2019

[PHOTOGRAMMETRY AUTOMATION TOOLKIT]

Graduation Report

Martin Niehoff, 21.01.2019

For the Creative Media and Game Technologies at the Saxion University of Applied Science

Photogrammetry Automation Toolkit

Graduation Report

Written by

Martin Niehoff

info@martin-niehoff.de

As graduation topic in the Creative Media and Game Technologies Course at the Saxion University of Applied Science

Supervising Teachers

Patrick Huitema

21.01.2019

Summary

Photogrammetry is a technique that allows it to scan in real world objects for the use in a virtual environment, by transferring two-dimensional information from an image sequence into a set of three-dimensional data, which can be used in games or other kinds of digital media.

Although it offers great possibilities, its usage is not always a viable option for projects, even if it's a technical & artistic viable option. As it adds additional complexity to existing pipelines and requires some additional hard & software. Which makes it interesting to see,

How to increase the viability of Photogrammetry for smaller objects in a production scenario?

Therefore, this paper is investigating an existing photogrammetry workflow within the graduation company to find a method for increasing its viability for future projects. For this purpose, we are taking a closer look at the different production steps within the scanning process, to examine their influences on the overall production speed & quality. Afterwards we investigate how these influences could be adjusted to positively influence the overall workflow. Furthermore, we are using the gathered data to build our own automation toolkit to help us removing the bottlenecks, which we found throughout our previous research.

Finally, we are comparing the workflow of our automation toolkit with the previous manual workflow to draw a conclusion about its effectiveness and show an example project from the company.

This paper is no meant as step-by-step guide trough an ideal capturing process, but rather as overview about the remaining possibilities within existing workflows, regarding their optimization & automation to increase their viability. It lists the different steps of my research and explains the reasoning behind their implementation within the automation toolkit.

Our research showed that the current process of photo scanning objects via photogrammetry leaves a lot of manual and repetitive labor for an artist within the workflow. This starts with the photographing of the object and runs through the entire process up to the texture generation. Although the computer does all the complicated computations for the user, the workflow requires him to do all the simple boilerplate work to pipe data through different applications. Though this are small and simple tasks, their overall influence in a larger production can quickly add up to a big cost factor.

This is where the usage of an automation toolkit comes in handy, as it takes over many of these tasks from an artist, so that he/she must spend less time on the creation per asset, which makes this technique therefore viable for more projects.

Figure 1 *Final Scanner with its mascot*

Preface

First off, I want to thank C4Real for allowing me to my graduation on a topic, which I'm personally interested in. Of course, not to forget, my colleagues who gave me throughput the creation of this report a lot of useful input. The freedom I had throughout the project in combination with the possibilities of the studio did not only allow me to develop a useful tool, but also to push my boundaries & grow on a professional level. I was able to deepen my knowledge in a broad area of expertise, while having fun doing so.

Furthermore, I want to thank my teachers who guided me throughout the project and allowed me to present its outcome to a broader audience at the Universities Accreditation.

Last but not least, thanks to all the great people from the open source community, who release some of their work free of charge, as it wouldn't be possible to create such tools like mine without their work.

Table of Content

Reason for the assignment

Introduction

Photogrammetry isn't a new technique, but it got used more frequently over the last few years. One reason for this might be the greater availability of VRAM, which has limited the usage of bigger textures in the past. Certainly, also the trend of Virtual Reality with its demand for immersive environments.

I choose this topic for my research, because I found it always delightful to be able to scan in real world object for the use in a virtual environment. My interest in the development of games started by the creation of custom levels for various games and there's always a demand for high quality assets, which allow a more detailed representation of the real world.

Photogrammetry is something that I wanted to try out on my own, since I knew about it. So, I used my second specialization to gain some experiences about its possibilities & weaknesses. This research gave me some good insides into the workflow behind the technique, but it also showed me what issues it had.

Therefore, I came up with the idea of further automating things to make the entire scanning process a more pleasant experience for me and others. After some of my teachers told me that this would be a good graduation topic, I started to look for a company, which would allow me to do so. With C4Real, the company where I previously also did my internship, I also found such a company. Who saw the possibility of photogrammetry and the benefit of a toolkit to support their internal workflow.

Figure 2 *Wireframe of a Scanned Bug, refer to the usage chapter for further details.*

Target Audience

This report is primary intended for people for are familiar with photogrammetry, its terms & requirements. Its focus lies on the explanation of the research, that I did regarding the workflow optimization for photogrammetry & not the technique itself. Certain technical terms or explanations may be incomprehensible for people who are unfamiliar with the underlaying technique. If this is the case, I would recommend having a look at my previous research project about photogrammetry upfront, as this covers the basics terms used on the following pages. You can find it under **Appendix A: *"Beneficial use-cases for photogrammetry"*** in the appendices to this report.

Graduation Assignment

The development of a "toolkit" for an optimized photogrammetry workflow of smaller assets for the use in games & interactive media applications. The toolkit should reduce the amount of manual labor, which is required during the process of the creation of photo scanned assets for (real) time applications and renders.

Company Outline

C4Real is a small visual 3D Design company located in the center of Enschede with a passion to develop virtual solutions for transferring knowledge into visual & interactive solutions.

They help their national and international customers to transfer complex information about the products they produce into visual content, that can be easily understood and absorbed by their target audience. This includes 3D animations and films, as well as interactive 3D, or virtual reality applications. The combination of these different visual tools allows C4real to present the products of their customers in a way that fits best to their target audience.

C4REAL
VIRTUAL EXPERIENCE

C4reals clients are ambitious companies seeking to communicate more successful technical products in a B2B market. Their clients reach from the automotive to the energy or aerospace sector and many more. In the past C4real has been working in cooperation with big partners like TenCate, AkzoNobel, Brink or Thales.

For these clients they formerly mainly focused on creating 3D animations and product films. Besides that, they have a young but quickly growing interactive department as they are planning to allow their clients to get all their visual solutions in one package.

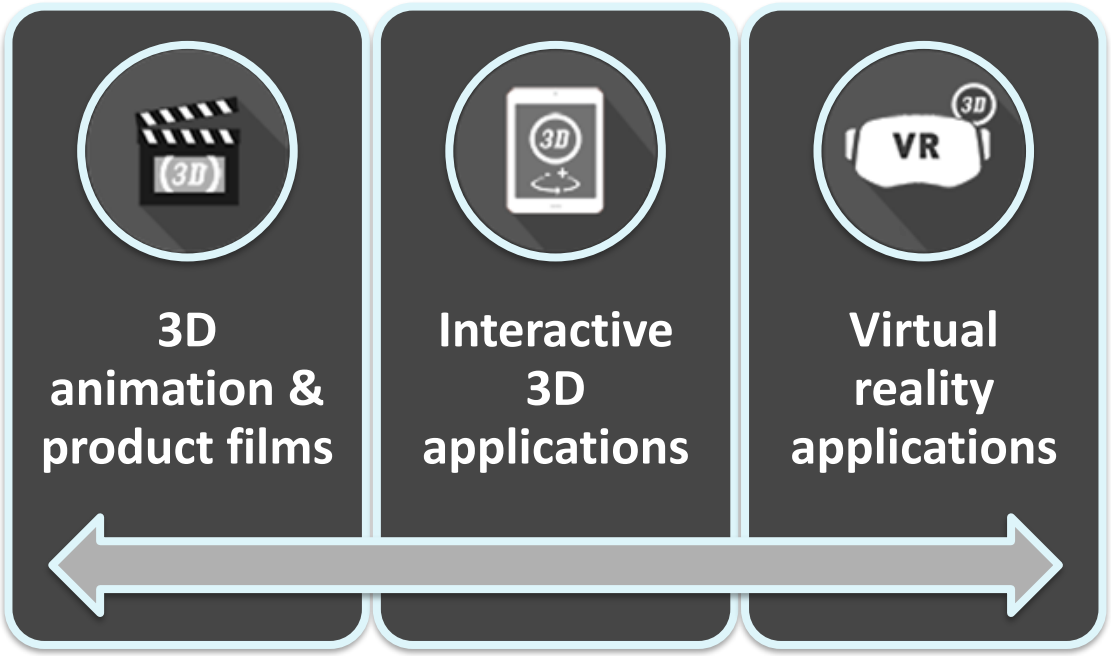


Figure 3 *Production "Pillars" of C4Real*

Objectives of the Client

C4Real is looking for a photogrammetry automation toolkit for their internal production pipeline. As the overall graphics quality in the industry is ever evolving the client is looking for a more efficient way to handle the creation of photo scanned assets, to increase its viability in productions. As photo scanned assets can help to accurately visualize real world objects, or to increase the visual fidelity & quality of the assets used in a production.

But as not every object can be scanned equally well and the creation of an asset involves multiple production stages, where different areas of expertise are required. Its usage isn't viable for many projects due to time & budget constraints.

Therefore, C4Real is looking for an automation toolkit, that helps to remove as many manual and repetitive tasks from the production of a photo scanned asset, as viable in the given time. It should help to decrease the production costs for this specific production technique with the goal to create assets for the usage in real time applications, for standalone & web deployment, as well as render productions.

It's not the goal of the project to develop a new way of scanning assets, or to reinvent existing production workflows, but to combine & optimize existing hardware & software for an overall increase in productivity. The focus lies on objects of small size, which are painless to process with currently available photogrammetry suites. The exact size of the objects is determined by the feasibility of on implementation within the timeframe of the graduation.

Preliminary problem definition

Photogrammetry is another workflow for artists to master and adds additional complexity to the existing workflow. Therefore, the client is looking for a solution to implement this technique into his existing production workflow, as he thinks, that his productions could benefit from its usage.

As the driving factor for the creation of an asset, is rather the required manpower than the used computation time, it's a viable option to automate as many parts of the asset generation as possible.

It's cheaper to buy another workstation, than hiring another skilled 3D artist.

Based on my previous experience with photogrammetry my initial guess was that most time within the scanning process could be saved by the initial generation of an asset.

The manual capturing process, image optimization, mesh and texture generation. As all these steps are equal to each asset and even include a lot of waiting time for files to be processed. Therefore, artists must wait for the computation output to continue their work, while in the worst case the computation is also slowing down their own workstation, hence negatively influencing other tasks, which they could do in the meantime.

So that the problem of the client is not necessary the complexity of the workflow itself, but rather the missing automation of its manual & repetitive parts.

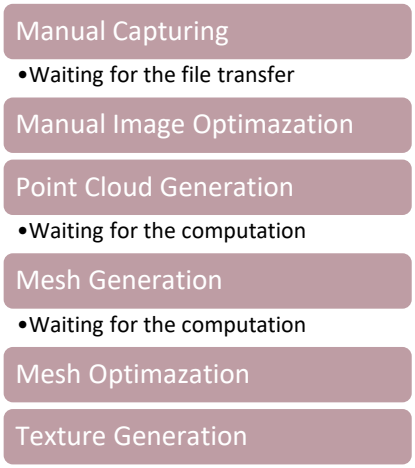
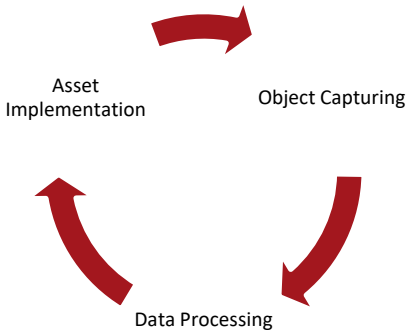


Figure 4 *Manual Workflow*

The creation of photo scanned assets, for the use in digital productions involves a lot of repetitive & time-consuming tasks, which drive the costs of its creation.

Therefore, it's to research, if the client can reduce the amount of manual labor in his photogrammetry workflow, to decrease the production costs and risks of photogrammetry in his current workflow. As this would make its usage more viable in a lot of production scenarios, what on the other hand helps to further improve the production quality.



Most projects requiring more than one asset, therefore its generation is getting a repetitive task.

Theory

I gained already some experience about the usage of photogrammetry during my specialization, which helped me to identify the bottlenecks of the current asset generation pipeline within the company. As the problem of the company is quite specific towards its own workflow it's hard to refer to any books, or publications about the problem itself. But many examples about similar problems and some solutions from companies who phased similar problems can be found on the internet.

Many companies worked on the creation of an asset pipeline for the creation of human characters with photogrammetry, like EA Sports for FIFA (DIMENSIONAL IMAGING LTD, n.d.), which involves the usage of many cameras, or Quixel (Quixel, 2014) & Ready at Dawn (Ready at Dawn, n.d.) who build custom scanners for the creation of PBR materials. Although our goal is neither the creation of characters, or materials, they faced the same problem.

Figure 5 *Material Scanner from (Ready at Dawn, 2018)*

The conversion from a real-world object into a digital asset involved a lot of steps & data processing, which must be repeated for each object. Although they all faced different problems, they all solved them by automating some of these tasks with the help of some additional hardware and/or software.

Tough, our requirements are still a bit different. We don't need to scan in moving objects, like humans, or gain the surface structure of flat objects. Our goal is to scan in small objects as 3D props.

Therefore, we don't require dozens of cameras to capture the entire object in a single moment, which drives the costs of such a setup. Or recreate the structure of a surface from a specific angle. Instead we need to capture and process and object from as many sides as feasible with as less human interaction as possible. This makes the usage of a turntable as already shown briefly during my specialization ideal (Niehoff, 2016), but in a polished and further optimized manner.

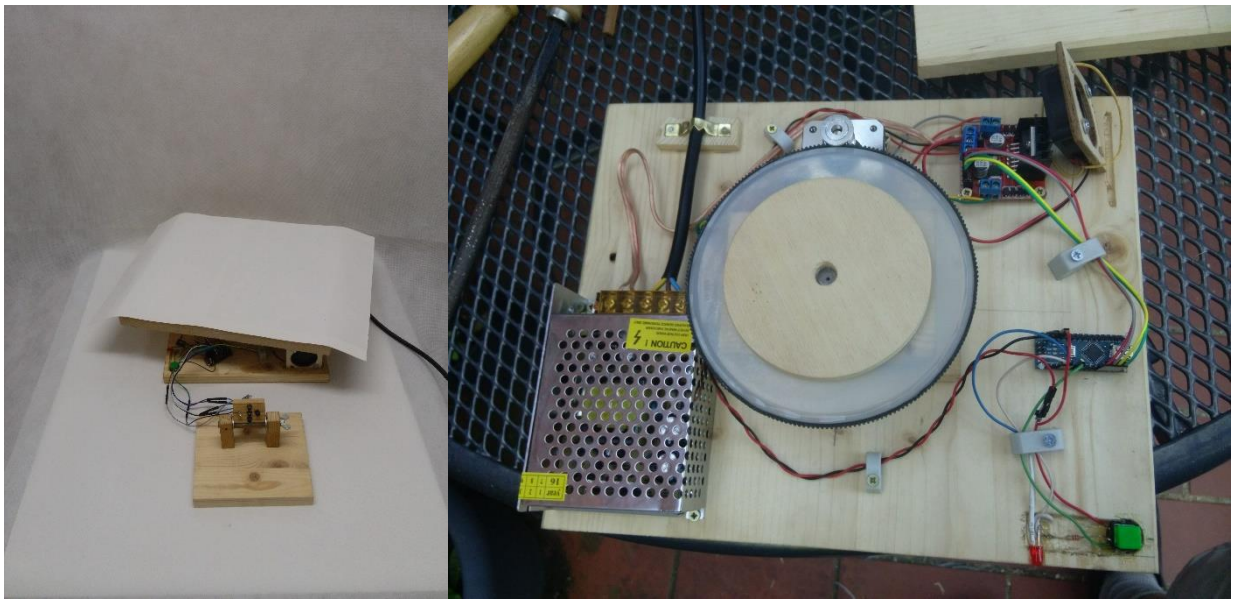


Figure 6 *First prototypes of an automated turntable during my Specialization*

The creation of an asset with the current workflow can be divided into the following steps.

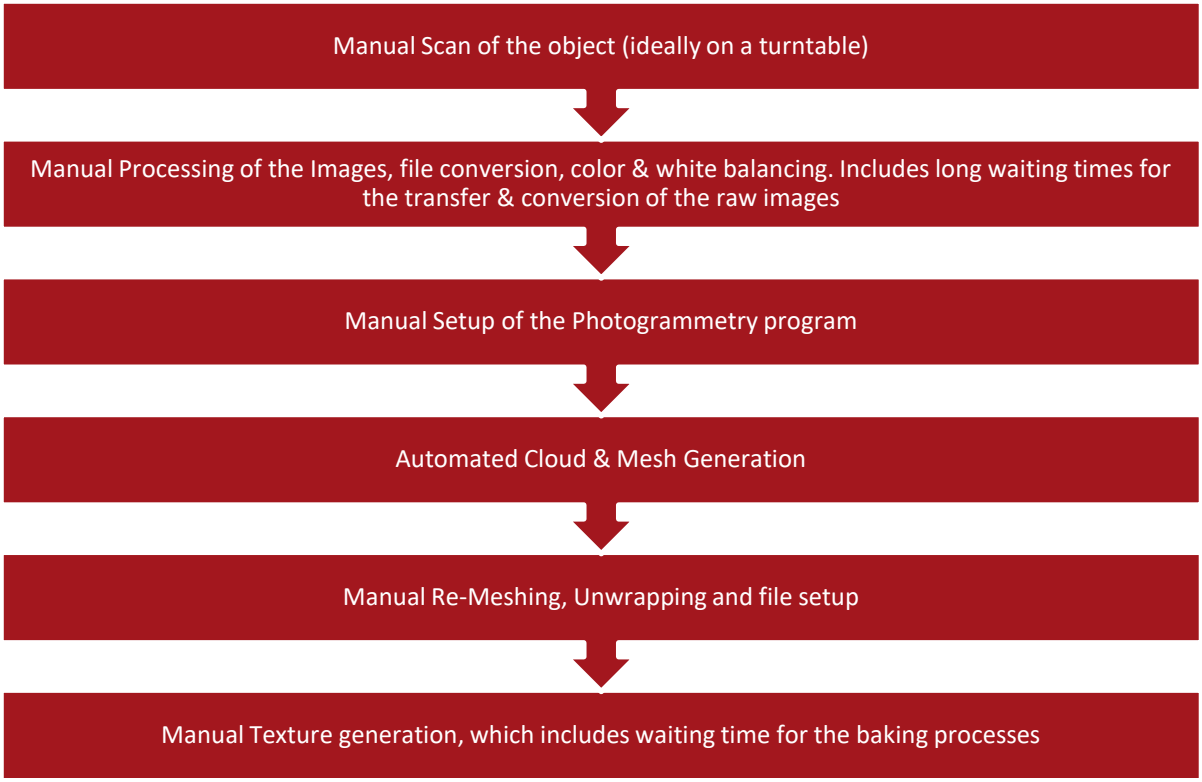


Figure 7 *Detailed overview of the manual workflow*

Many of these tasks are similar for each scanned object and involve quite some idling time for an artist, as he/she needs to wait for files being processed. Although some assets will (always) [until new sorts of hard & software are out] require manual work by an artist, like further remeshing or manual placement of UV-seams. But many of the above listed tasks can easily be automated, so the artist can work on other things while his asset is being processed.

Handling the automation of these tasks with our toolkit would help our client to decrease the production costs & risks of photogrammetry in his current workflow and therefore make it more useable for him.

Problem Definition

Based on the information I gained throughout my research it shows, that the client is looking for solution to minimize the workload of his artists throughout the process of photo scanning assets for a digital production usage, without losing the freedom of artistic choices, or quick iterations.

As the client is looking for a way to reduce the required effort for the usage of photogrammetry within its existing workflow. It's important to reduce the amount of manual labor that an artist must spend per asset during its production, to make it viable for more projects.

Main Question

Therefore, the following main question can be formed:

How to increase the viability of Photogrammetry for smaller objects in a production scenario?

Sub Questions

Now after our main research question is formulated, we must find out, what is needed to answer it, while staying in the scope of this report. To do so, we must clarify a few things up front:

- What are the time consuming tasks within the current workflow?
- How much manual labor involves each of these steps?
- Which of these parts could be further automated?

With answers to these sub questions it's possible to define the requirements for our desired automation toolkit. Conclude which parts are wise to automate and which parts should remain at the hands of the artist to optimize our existing workflow, based on our given timeframe.

Accordingly, we can define some guidelines for the ideal usage of the automation toolkit, which brings up additional questions. Like the ideal background & lighting setup, as well as an overview about the recommended camera usage.

Furthermore, there are also some technical challenges, which needs to be solved. What's the best approach to build the turntable? DSLR based, with a single, or with multiple raspberry pi cameras? How many mini computers or cameras do we need for this?

This brings up the question how to connect all the capture and render devices in a reliable manner. Another thing, which needs to be researched is the scalability of the used setup. What happens, if it required to capture bigger objects in the future?

The answer to most of these questions depends on their setup costs in comparison to the given results.

Research Approach

As the viability of a specific workflow within a production can be influenced by many different things, I started to work on this assignment by first defining its theoretical scope further. I looked for answers to the workflow related sub questions listed within the problem definition of this report. This helped me to value the importance of specific parts within the current workflow and to define more precise requirements for our automation toolkit.

With these requirements in mind I started to work on solving the technical challenges within our project scope. From the most challenging one towards the smaller chunks, until I had enough information to work on the implementation of the automation toolkit and was confident enough that the project was manageable in the given timeframe. This was mostly done by desk research per topic with some follow up experiments and the development of small independent prototypes to validate, if my desired solutions would work as expected.

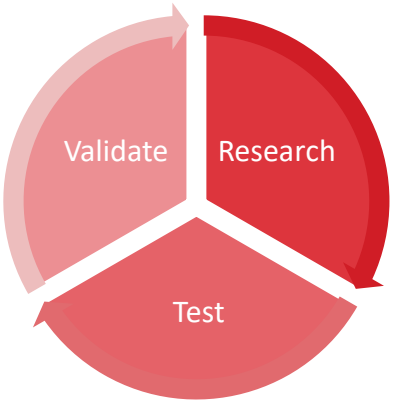


Figure 8 *Research Steps*

Initial Research – Defining the theoretical Scope

I started my research by talking with my co-workers about their existing approach to scan in objects via photogrammetry and compared that with my own experiences. This helped me to get a felling for the complexity of the current workflow within the company and how time consuming and complicated each step in the process is.

The current workflow can be broken down into the following main parts:

- Preparation
- Image Generation & Processing
- Data Generation
- Asset Generation

*For more details have a look at the **Appendix B: "Photogrammetry Asset Pipeline"** in the Appendices.*

I processed a couple of objects this was and kept track of the time which I required for the different steps in the process. Afterwards I tried to compare these values with experiences from my co-workers to validate my findings. I used this data as baseline to determine the complexity of each step in the current workflow and valued its importance for an automation based on it.

The preparation time may vary depending on the used objects and many tasks like the preparation of the equipment do not have to be repeated on every object, but only at the change of the location.

Although most of the overall time is spend on the Data Generation, which includes all the computation intensive tasks like the Point Cloud Generation, the Image generation and its processing requires most of the artists time on many objects. This shifts towards the asset generation, if an object requires a lot of manual cleanup or polishing.

Figure 9 *Influence of Asset-Complexity*

As the development of more time efficient tools for the data generation would blow the scope of this project and the time required for the cleanup of an asset depends on the object itself, rather than the used workflow. I tried to focus on the parts within the workflow, which are similar on all objects and involve a lot of manual and repetitive work, which the artist needs to repeat on each object in a similar manner. This involves the image generation, its processing and piping through to the tools in the data and asset generation.

It makes most sense to focus on these parts of the process, as their manual execution by an artist won't bring any benefits to the product, but rather block other parts within the pipeline. Furthermore, these are the simplest things within the current workflow to be automated, without over engineering certain parts, which aren't worth the investment, as they might become outdated, or to complex too quickly.

There is no benefit, if the artist walks with a camera around the object, or manually rotates the turntable instead of letting a motor doing it. Furthermore, the artist needs to wait multiple times for data to be processed within the current workflow just to pipe data afterwards into the next program. Either the artist waits for a task to finish, which reduces his/her productivity, or he works on other things and therefore might response with a delay on the finished task. Although a tool could to that for him immediately after the previous task is finished. If it is filled up with the proper information by the artist at the beginning of the process. This allows the artist to entirely focus on other things until he/she is required to perform some follow up tasks, which cannot be further optimized too easily.

Digging Deeper – Validating the Initial Research

After I determined the parts of the workflow, which I would like to automate I tried to determine, how much time I would be able to gain by this. To see if it would be worth the effort of automating it.

There are dozens of different hardware & software combinations, which can be used for the generation of photo scanned assets. Therefore, I started my research with a small series of "benchmarks" to compare the influence of different hard- & software configurations on the asset generation. This allowed me furthermore to gain some additional experience with the used photogrammetry suites to decide, which would be most suitable for our use case. You can find the Benchmark results under **Appendix C:** "**Photogrammetry Benchmarks**" in the Appendices.

But as the goal of this project is it to improve the current workflow within the company, I manually captured and processed the same object multiple times with the current workflow and tracked how long each step would take me on average. For that I took a small model of a bird, as its surface is well suited for photogrammetry & hence lowering the influence of capturing issues disturbing the results.

Figure 10 *"Analog" Turntable*

I captured a sequence of 74 images with the help of a tripod & turntable. Copied the files from the SD Card of my camera to the pc, where I converted the raw files before processing them with VisualSFM again.

Once I had this data, I checked how much of time I spend on the tasks, which I planned on optimizing and how often I was waiting for tasks to finish or block other tasks.

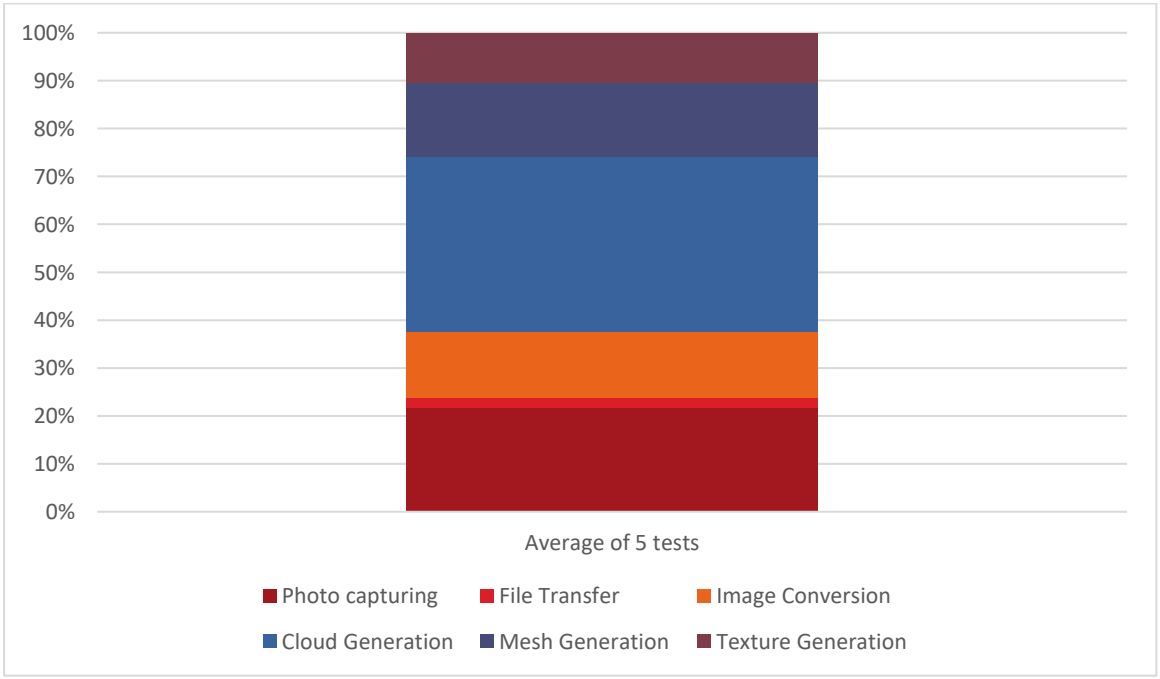


Figure 11 *Sample Times of manual processing steps & their impact on the overall time*

This simple proof of concept showed, that there is potential for the automation of the capturing process, as well as the following data generation. But it also made me aware of the fact, that it's not necessary enough to automatically pipe the data of one task into the following application, to avoid idling time for an artist. As some of the tools, which are used within the current workflow are so computation heavy that an artist cannot work on his machine anyway, while its processing.

Refining the Scope - The Requirements of the Toolkit

The validation of our initial research showed that there is enough optimization potential within the current workflow, but the question how to achieve this automation is still open.

Therefore, I continued by setting up a plan of approach to research the requirements of our toolkit and how to solve these requirements in the most suitable way. While looking at the complete picture of the process I thought that there are basically 3 different sub parts involved.

Figure 12 *Involved "actors" within our process.*

Our goal is it to reduce the work that our user/artist must carry out. As a lot of this involves the interaction between him/her and the two other sub parts, it makes sense to tackle their intersections with our research. So, I divided the project into 3 subparts, each with their own requirements and duties, based on their part within the current process, to make the entire research project more manageable.

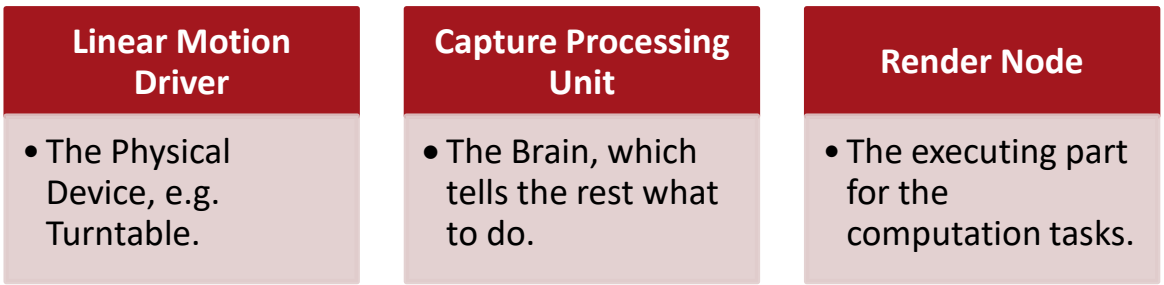


Figure 13 *Research Subparts*

I then went ahead and defined the requirements & duties for each of these parts as baseline for my further research.

Linear Motion Driver

The linear motion driver is the executing part of our automation toolkit, which handles the movement of the scannable object, which is required during the capturing process. Its whole purpose is to move the objects, which are placed on top of it with as little user interaction as possible. Therefore, it must be able of moving objects:

- 360 degree around one axis.
- precise enough to handle complex object types.
- with various shapes & forms.
- with different weights and a center of gravity.

This brought up the following questions:

- What type of motor is most suitable?
- How precise does it need to be?
- What power does it have?
- How to connect/interact it with the rest of our tools?

I had already some experience for such a setup from previous projects, so that I knew what to look for. Therefore, I thought the most suitable approach to answer these questions would be a small desk research of the available components, which are suitable for our use case. Followed up by a small prototype, which verifies if it fulfills our requirements & matches our expectations.

Capture Processing Unit

The capture processing Unit is the brain of the automation toolkit, it handles many of the tasks, which an artist must carry out manually otherwise. Or in other words, this is the part of the automation toolkit, which is the key behind our research. It carries out the capturing process with the turntable and the communication with the render nodes for the user. Therefore, it needs to:

- offer a (network) connection, for the communication with the render nodes
- be capable of handling multiple cameras
- have enough power to handle all the upcoming data

Tough it relies on the other components for its functioning I wasn't able to set any fixed scope for its requirements at this stage of my research. Therefore, I planned in some extra buffer time to redefine its scope and evaluate its requirements, once I have more detailed requirements of the other components.

Render Node

The render node is the actual worker of our automation toolkit, which does all the heavy lifting throughout the process. It listens for commands from the capture processing unit and carries them out on a powerful machine in the network. This includes all the file processing and data generation, which an artist must carry out manually otherwise.

So the main purpose of the render node is to:

- receive file from the capture processing unit
- deliver the output to the user

Therefore, it needs to be able to:

- connect to our capture processing unit & gather data from it
- control our computation applications
- transfer its output data

Furthermore, I considered a multi-platform implementation of the required software to increase its compatibility with existing soft & hardware solutions within the existing workflow/studio.

As the tasks of our render node are relatively simple, I didn't expect too much meaningful research for this part of the project. Never less I was aware that a lot of its tasks are something I personally didn't implement yet and therefore require me personally to study various application documentation and implementation concepts. That's why I expected this to be the part of the project, where I personally would learn most new stuff, that could help me in future projects as well.

Scope

As this project will be my graduation project, I must fit everything into the given timeframe for my graduation. Therefore, I have 18 weeks of time to develop a usable prototype and prepare the school related documents like this report. Hence, I had to cut some corners to keep this project within a manageable timeframe.

The project is focused on the combination and integration of existing tools and workflows into an already existing pipeline, to extend its feasibility in productive use cases.

It's not within the scope of this project to write new algorithms or tools to handle specific steps within the given workflow. But rather to look for existing resources that can be used in a more effective manner, to increase the productivity in the given workflow. This includes hard & software used within the pipeline.

Although the goal of the project is to have a production ready toolkit, it's not meant to be completely polished or perfect, but sufficient enough to support, or replace the existing workflow and to be used as a foundation for future development.

Requirements

The turntable is used to automate the scanning process of smaller objects. Admittedly you may ask now, what means automate, or what is referred to as small object?

What means automate the scanning process?

The process of capturing an object via photogrammetry involves a lot of repetitive tasks, as you basically repeat the same action over and over, with just a slight adaption. Usually you pick an object, which you want to scan and take a picture of it. But wait you don't take one image, you take a dozen of them, only from different angles, right?

That's a thing that we can easily automate, instead of manually pressing the capture button of your camera hundreds of times, we could just tell your camera to take a picture once it's in position. Furthermore, the moving of the camera is also just a linear path, which can be setup quite easily for certain objects, which leads us to the next question.

What means smaller objects and why did I choose them?

What I refer to as smaller objects, are things that you can easily pickup by hand and carry around if you like. The reason why I went for this type of objects, it's easier to gather a lot of them, which is useful for setting up a robust pipeline. Furthermore, it made me a bit more weather independent, as I had the space to setup a small indoor studio for storing & capturing these objects.

Of course, I could also build a bigger turntable, but to be able to setup a reliable pipeline in the given timeframe (and budget) I had to setup some boundaries for this project. I ended up with this rough baseline, to keep the hardware costs for this project in a more student affordable amount. This is also the reason why I went for a turntable as "driver" of the linear camera motion. It's easier to move smaller object around themselves (if you can lift them on the turntable), than to build a camera track around a big object.

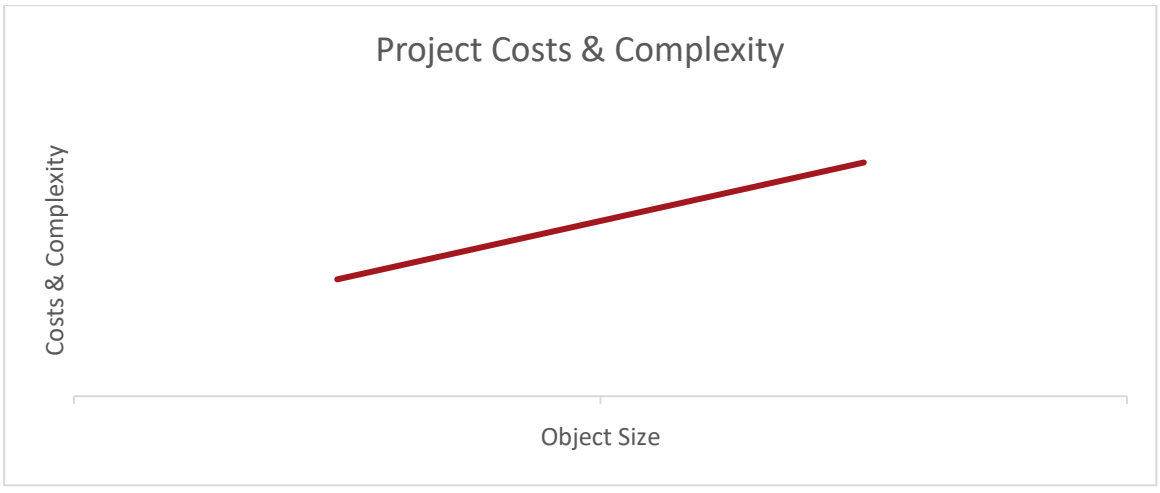


Figure 14 *Project Cost & Complexity*

In theory the same principles could be used to develop hardware for different scales and use-cases, but with increasing scale & weight of the objects, the costs of the required hardware for the turntable, studio space etc. will also increase. However, some objects, might be too big, or heavy, to be scanned with a turntable in this case you could replace the "linear motion driver" with some camera tracks or drone.

Outcome

In the end I want to have a production ready automation toolkit for the creation of photo scanned assets, for the use in in various digital productions, like games, interactive media and renders.

This toolkit should help the client to scan in various small objects up to ~50cm³ in a standardized manner, with an easy to understand & reliable workflow for production usage.

The automation toolkit should be:

Easy to Setup	Automated Capturing	Quicker Processing	Reduce Work
• The entire ecosystem should be easy to understand & install.	• It should automate the capturing process.	• Allow quicker image processing and less idling time.	• Reduce the manual & repetitive workload of the artists, who are using it.

Figure 15 *Planned Project Outcome*

Development Schedule

With the outlined requirements from my previous research I went ahead and defined a development plan for a first prototype.

Pre-Development

As I am planning this research project already for quite a while, I could do some preparation upfront. I did most of the initial research already before the start of my actual graduation period. Furthermore, I researched different components & software solution, which I thought I would require throughout this project. This helped me to avoid any tricky blockers, or expensive in-between solutions, for missing parts during the tight development phase.

Prototype Testing - 1 Week

To avoid any nasty surprises, test all required functions on the compute board before their detailed implementation. Are all required applications running? Is the performance sufficient?

Case Development - 2 Weeks

Develop a (if the time allows it) modular case for our turntable, with some simple scanning hats.

Proof of concept turntable – 2 Weeks

Setup a rough proof of concept of our planned pipeline (simple, without any customization) and test the prototype device early with it, to see if we missed any important usability aspects on the hardware.

Proof of concept, network rendering – 2 Weeks

Develop a proof of concept for our network-based preview calculations & asset generation, to avoid any nasty surprises in the later development.

At this stage, we can reshuffle our planning, if any discoveries require it

---

Develop a proper Web Interface – 3 Weeks

Polish our proof of concept interface. Add a proper design to the panel & include customization options for the scanner & scanning process. If that's done, we should do a couple of test scans, to test its usability & functionality.

Develop a stable network rendering – 4 Weeks

After we polished our turntable, we can continue with tweaking the asset generation over the network in the backend. Add customization options to the used hardware & extent their functionality.

Polishing – 2 Weeks

If both parts of our pipeline are working, properly we should stress test them a bit and maybe get some third options about the scanning process to further tweak the workflow.

Development of additional functions & devices – 4 Weeks?

If at this stage still some time remains, we could implement some additional software sided features from our wish list, or develop additional devices, like our camera node, or stand.

However, as I probably also need some time at the end to write my report & prepare my presentation, I left the development of the additional devices as functions as extra milestone, that I can use as buffer, if certain things take longer, or I need additional time for school related things.

Project Results

The project ended up being a quite complex and encapsulated product, which makes use of a variety of different technologies. As it would completely blow the scope of this report to handle every little detail of the toolkit, I will go only briefly through the in-depth research & implementation of each of the mentioned subparts. For further details please refer to the mentioned appendices.

Research – Solving the underlaying issues

Linear Motion Driver

It was very important to me, that the motor would give me a precise control over the rotation of the object, so I'm able to capture parts of objects, which aren't visible from every angle and therefore might won't be scanned correctly otherwise.

Therefore, I focused my research for the linear motion driver on the usage of a stepper motor as main component. I had already some experience with the development of smaller motorized devices driven by an Arduino and raspberry pi and therefore thought that a stepper driver would offer me the best mix between precise control & torque. As it's hard to achieve exact motions with dc motors alone and most servo drivers don't offer enough torque to drive heavier objects. Stepper motors allow precise control, offer more torque and are widely available at an affordable price range.

The wide availability of the hardware, its price range and my previous experience with it where also part of my decision to go for solution with a stepper motor.

Nether less it took a bit of extra research to figure out, which exact model would be most suitable for our use case and how to calculate its torque. Analogous, how much weight our scanner would be able to move without jamming.

Once I had a working motor, I build a simple prototype with an Odroid and a driver board, that I had left from another project. Though the initial prototyping went smooth I had to replace the driver board at a later stage, due to combability issues with my used processing board. Which resulted in some extra work on the already designed case.

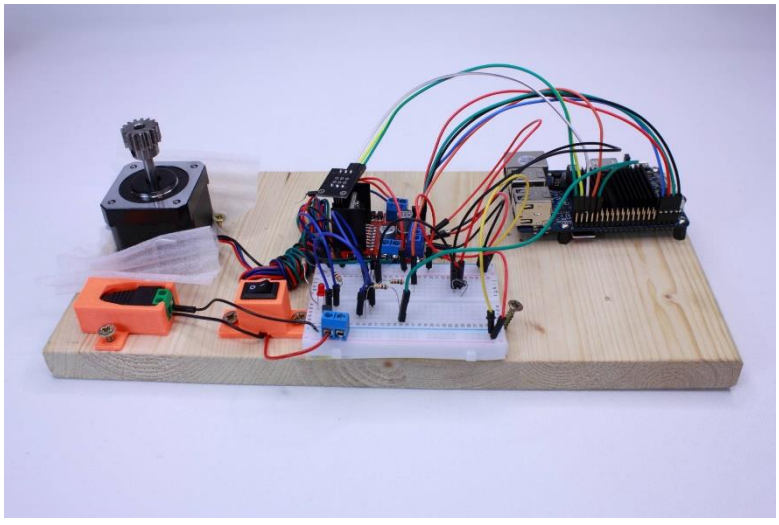


Figure 16 *Initial Prototype for an advanced turntable*

Central Processing Unit

I had some basic requirements for the functionality of the central processing unit in mind, when I started my research. It should drive our linear motion driver and be the bridge between the artist and the worker nodes. Due to the complexity of this, it was clear from the beginning, that we need a bit beefier minicomputer to handle all the incoming data, without slowing down the scanning process. So, I divided my basic requirements for the hardware into smaller chunks and started with a theoretical research where possible to have a foundation for my practical tests.

A good example about this is the required network functionality. I was aware that this will be one of the key points for the speed and therefore the usability of the toolkit, as Photogrammetry is a quite data intensive process, especially when dealing with raw footage. Not all mini computers offer the same connectivity. Some offer Wi-Fi, some ethernet, some none and all that with all kinds of variances in speed. That's why I wanted to ensure that I buy the most suitable device for our use case, instead of going ahead with the most obvious choice, like the raspberry pi, which isn't ideal in the end.

I calculated how much data our setup would produce in a worst-case scenario, many high-res cameras with raw footage. This removed the raspberry pi already from our list, as its single (internal) usb port and slow ethernet port would bottleneck our file transfer, even below the worst case.

This simple theoretical research allowed me to reduce the selection of boards but it also leaded me to the question, how to store all the upcoming data on the device. That's how one question leaded to another extending my list of requirements gradually. Admittedly this research was less time consuming as the usage of the wrong board would have been.

Figure 17 *Collection of tested boards*

As the research and development of the central processing unit ended up being the most time-consuming part of my project, at least in relation to the overall time frame from the start to the first fully working prototype. This was mostly caused by delivery delays of the ordered hardware parts and the in-compatibility between some parts. Therefore, I had to replace some components a few times throughout the process, although I did an in-depth research upfront to prevent this.

As it's the central point of our hardware, which connects everything together, there are a lot of variables, which come together. I had to test component by component separately for its proper function with the central processing unit to ensure, that I can combine everything to a fully functional prototype in the end. And naturally its always the last thing, that you test, which doesn't work as expected. These issues where mainly caused by the problem, that the hardware and software quality of the mini computers in the size of a raspberry pi vary a lot. While some have a good software supports, they lack some hardware features for our project. Others have in theory the perfect hardware for our needs, but missing software/driver support makes them unusable either. Many issues can be avoided by a look at the spec sheets of a board and its vendor forum, but some also require hands on tests.

Based on my research about a few currently available micro controller boards on the market, I concluded, that the Banana Pi M2 Ultra, or the Odroid C2 would offer the best starting point for my turntables control node. I started my prototyping phase with the Banana Pi but had to switch towards the Odroid at a later stage, due to some limitations. Unfortunately, the C2 wasn't anywhere on stock at that time, so that I had to go towards the next suitable choice, its predecessor the C1+.

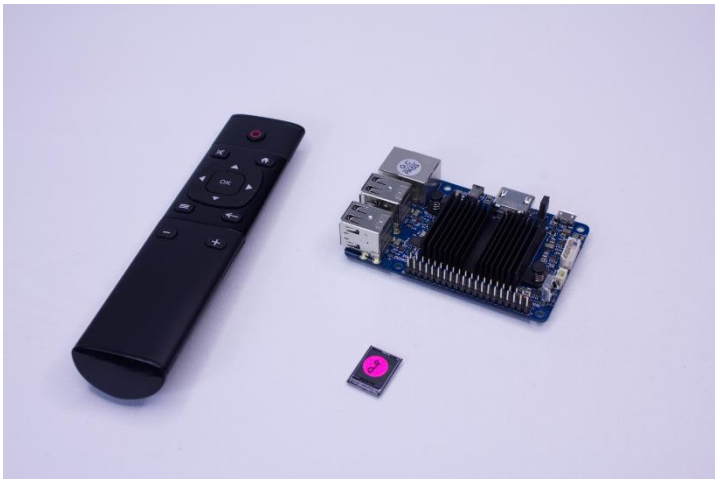


Figure 18 *Odroid C1+ with Remote & eMMC*

Nether less my research showed a couple of tricky aspects on my "ideal" setup for which I would like to develop my turntable, especially the required high transfer rates. Although there are a couple of alternatives, to bypass these issues, like the usage of higher tier arm or x86 board boards (I only looked at the bottom/mid end of the available arm boards) it should be possible to develop a device that matches almost all my requirements for a perfectly usage turntable, even with a relatively payable device as base node.

Render Node

The render node is in simple words just carrying out the work of the artist. This includes all the file processing and data generation, which an artist must carry out manually otherwise. Therefore, I had to research tools, which could carry out his work in a non-interactive way through a command line interface, without loosing too much control over its output. I had the list of tools, which have been used in the existing manual workflow, but not all of them where suited for further automation. As most of them offered command line options only in additional packages, or not at all. Furthermore, the workflow used a variety of tools, which had to manually filled up with data by a user for each scanned object.

Therefore, I started my research for the render node, by looking up existing tools on the market, which are suitable for the usage in an automation toolkit as mine. The main requirement was that they offered a simple to use command line interface and can produce consistent output data, while not requiring too much user interaction/corrections on the way.

Furthermore, I investigated the platform combability and pricing/licensing of each package to draw my conclusion. I wanted to keep the entire project as open source & platform independent as possible, as its main purpose is it to simplify tasks for the user. But many external dependencies or expensive node locked applications within the pipeline don't help to achieve that. As the render node is the actual worker of our automation toolkit, which does all the heavy lifting throughout the process, it can be counterproductive for the workflow of an artist, if it computes all the data on his workstation.

Therefore, I looked for multiplatform applications, ideally without to restrictive licensing options to make it easier to use the automation toolkit on multiple machines. To lower the burden of a multi machine setup, where a different machine in the network processes all the files, while the artist can continue to work on his workstation.

Though I didn't manage to find suitable open source alternatives for all commercial applications within the pipeline, I managed to remain relatively platform in depended, as all required software packages run on both Windows & Linux. As the worker nodes code base is also written on a multi-platform basis its easy to setup a Linux based render node for the processing, while the artist continues to work on his Window machine.

As there are many different tools for each of the involved steps within the pipeline this got a bit longer part of my research than initially expected. Therefore, I recommend you have a look at the **Appendix D:** "**Software Requirements**", if you would like to have any further details.

The ideal setup for me turned out to use:

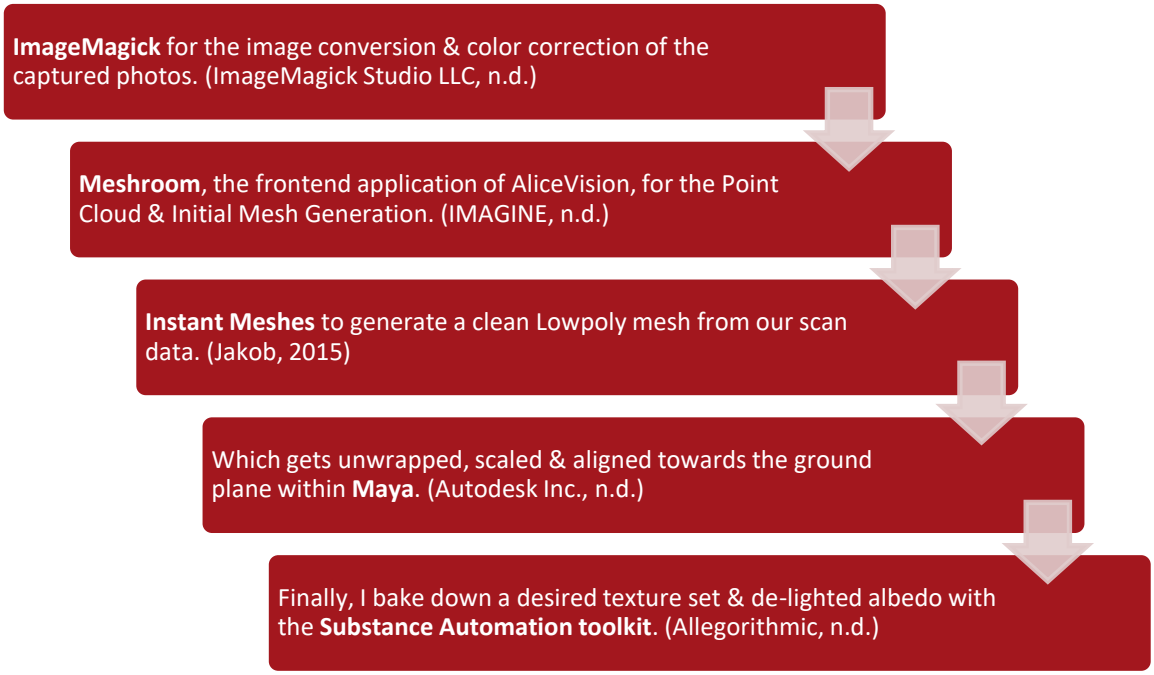


Figure 19 *Used Processing Packages*

The render node itself, with all the network logic is written with node-webkit (Community, n.d.) to keep everything platform in depended.

The exact output of the render node is defined by the quality & project settings the user defined at the beginning of the scan through the web interface. Tough, it's easy to bake down additional textures or iterate over the model afterwards due to the quick access of all production files, which is provided by the tray app. The default settings are aimed at providing the necessary input files for most of our production scenarios within C4Real, this includes a textured low poly mesh plus ambient occlusion and world space normal together with the original highpoly mesh used for the bakes.

User interface

The user interface of our automation toolkit consists out of two separate parts. The main part is the web interface, which is used to control the scanner & start up the scanning process. Furthermore, our render node others a small ui, which is used for its initial configuration and easy access of its output data.

I started the work on the user interface by defining a style & branding guide, as I wanted to give the project a more professional look, but also to gain some personal experience in these fields. In this phase, the name „Rapitry“ was born. Which is a simple combination of the word rapid, for the increase productivity with the automation toolkit and photogrammetry.

Also, the name fits to a rabbit as logo, which personally reminds me always of the so often used Stanford bunny in the 3d world.

Afterwards the design of the interface was a quite straight forward process as my previous research already produced a feature list that I could work off, piece by piece in an iterative process. Tough I made some smaller layout adjustments throughout the process, based on some user feedback I got from my coworkers, who tested the application once its basic functionality was implemented.

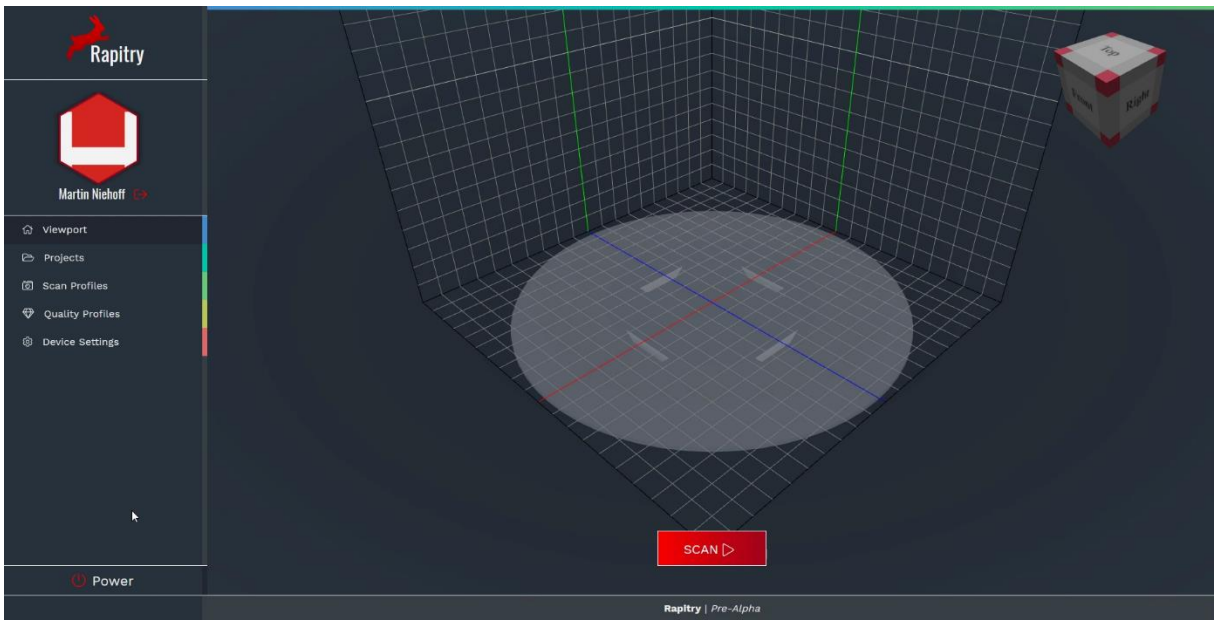


Figure 20 *Webinterface Home Page with 3D Viewer*

The web interface of the scanner is built with Vue.js (You, n.d.), Laravel (Otwell, n.d.) & PlayCanvas (PlayCanvas Ltd., n.d.).

I choose these frameworks as my previous experience with them allowed me to better estimate how much time the implementation of certain features would require me, while also allowing me to implement more features in the given time of my graduation.

Playcanvas is used for the WebGL based 3D viewer on the home screen, while Vue.js components are used on all pages of the frontend to speedup the development process & increase the usability of the application with reactive ui parts.

The php backend handles all the communication with the user through the frontend, a persistent data storage through a MySQL database, as well as the scan logic itself. Smaller chunks on the hardware level, like the direct stepper motor control are independent python or bash scripts, which are called through command line commands from Laravel.

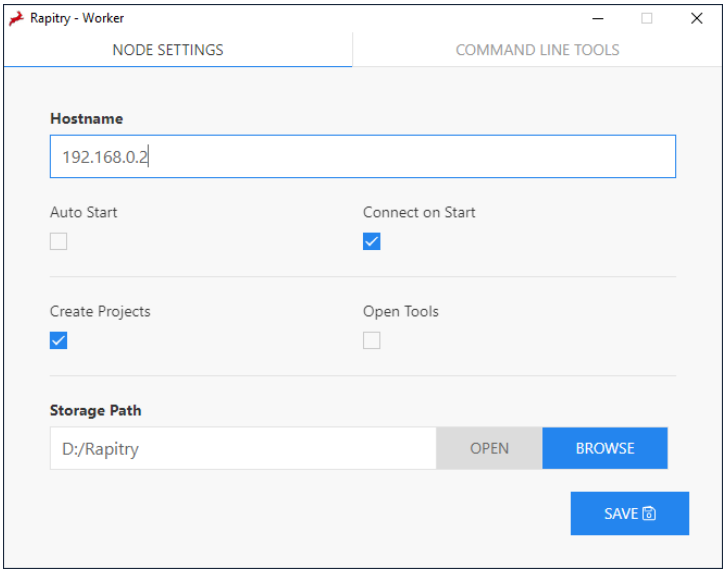


Figure 21 *Rapitry Tray Tool - Node Settings*

The render node on the other hand is built with node-webkit. The UI is based on the same Vue.js components as the web interface of the scanner, just with a lighter theme as alternative styling. The backend logic of the worker node is written in separated node.js modules, which allowed me to quickly test the implementation of different processing tools within the pipeline, without the need to rewrite too much code within the main app.

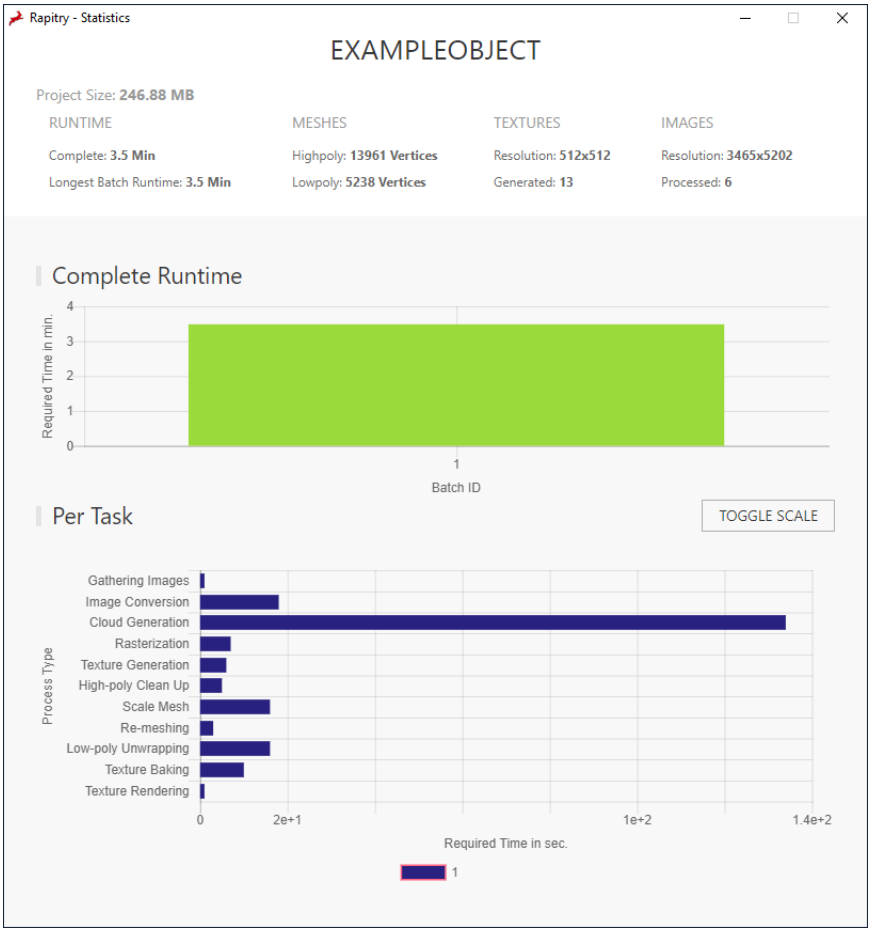


Figure 22 *Rapitry Tray Tool - Process Statistics*

Scanner Case

Although the housing of a device seems to be relatively irrelevant for the functionality of a tool in the first place. It's still an important part of it once you think closer. It does not only protect the sensitive parts in its interior, it also influences how handy the tool is to use. Furthermore, its appearance influences how we perceive its functionality.

I planned in a dedicated timeframe for the development of the case for my scanner to ensure the scanner does not only work from a technical perspective but is also nice to use. As its not helpful to have a scanner which is so unhandy to setup, that you avoid using it.

I started with a small list of things, which I would like to incorporate within the design of the case. Some artistical, like matching colors between the case as physical part of the project and the digital parts like our user interface. Some technical like ventilation holes to prevent our hardware from overheating. Nether less the process of the case development consisted essentially of dozens of iterations from various case parts, as you can imagine from the image on the side.

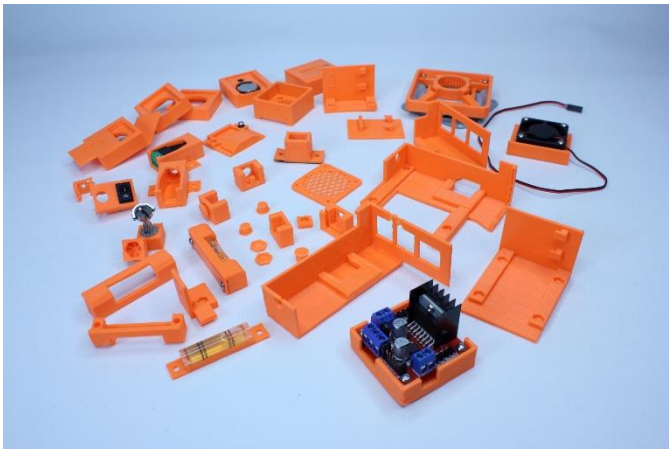


Figure 23 *Component Test Prints*

Tough I bought a cheap 3d printer a while upfront my graduation, it was my first bigger self-designed object that I tried to 3d print. Therefore, not everything that I modeled digitally did fit properly the first time I tried to print it. Some prints, where too small, some too large, some simply were not user-friendly to assembly, or use.

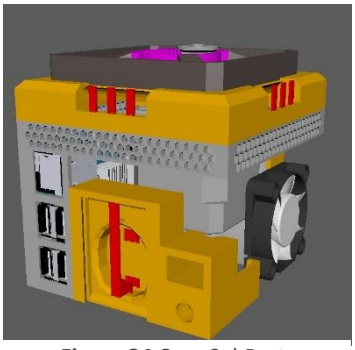


Figure 24 *Case SubParts*

Although 3D printing is known to allowing quick prototyping & rapid development, the word quick is rather relative from my experience, as printing a single piece often requires a couple of hours to print and increasing size & quality of the print quickly increase the print time. Therefore, I tried to slice my case into as many smaller test sections right from the start, so I could iterate mount by mount to ensure all my physical parts fit. Before combining everything into a bigger piece to simplify the assembly of the case. The final case consisted out of 3 bigger parts, which printed over 10 hours each and multiple smaller parts like the lids, so that the final print of the case with its assembly took me around a week.

Finally, I ended up spending even more time on the design & production of the case, than I initially planned. As some delayed shipping prevented me working on the hardware in the first place and some later hardware changes required further adjustments throughout the project. Tough this extra time payed off, as it allowed me to accommodate all the required hardware into a relatively small case, which still offers enough cooling for the hardware in its inside. Furthermore, the case allows easy access to all required hardware parts and is portable enough for mobile usage. Two levels on the top of the case and a tripod mount on the bottom encourage its transportability even further. Tough the most important part of the design is the detachable top, which enables artists to use a top, which fits best to the target objects, which they like to scan.

Outcome – Workflow of the Automation Toolkit

The goal of this project was it to develop an automation toolkit, which could increase the viability of Photogrammetry for smaller objects in a production scenario. My research showed me that I needed to build two things to achieve this, a physical helper, our scanner, which works in combination with a custom software package to decrease the workload of our artists in the given production scenario.

These two helpers allowed us to greatly simplify the capturing process of smaller objects within the company. The scanner is quick & easy to setup and lightweight enough to transport, due to its small form factor. While the additional documentation makes it easier to explain this workflow towards newer team members, compared to our old workflow.

We went from our old manual workflow:

Figure 25 *Legacy Workflow Steps*

Towards a leaner process which leaves most work to our toolkit instead of our artists.

Figure 26 *Rapitry Workflow Steps*

Red Arrows Visualize Work carried out by an artist, while green arrows are carried out by the PC.

The biggest time safer here is that our artists have less interaction points with different software packages. Furthermore, the scanner allows them to scan multiple objects in a batch, which increases their efficiency even further. The render node will than process one object after another. Tough this is only advisable, for scanning in similar objects, or objects you know to scan well.

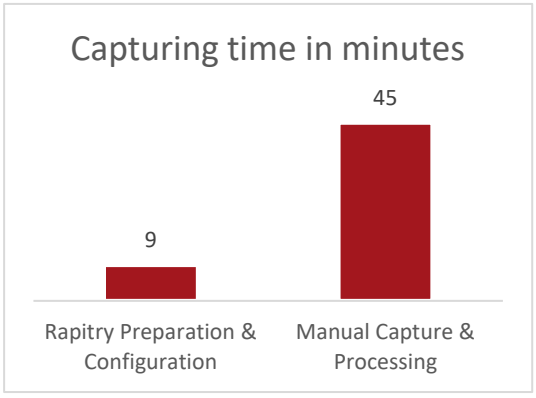


Figure 27 *Workflow Time Comparison*

Figure 27 shows a comparison of the time required for capturing and processing of an image sequence.

For this we repeated our test process described by Figure 11 with our automation toolkit. Although the processing times of both workflows remain the same, the reduction of the workload for our artists is significant.

The hardware costs of the current iteration of the scanner are only around 150€, ignoring the initial research costs. As the scanner can be connected to regular DSLR cameras, which we have plenty at the studio, no further hardware is required on top of our old manual workflow.

Furthermore, C4Real didn't need to obtain any additional software licenses, as all third-party software packages, which are used in the current version of the automation toolkit, are either free to use, or are already in use at the company.

Conclusion & Discussion

The viability of Photogrammetry for smaller objects in a similar production scenario can be greatly increased by an optimization of the underlaying workflow. As the process of the asset generation involves many repetitive tasks with time consuming computation times, its beneficial to automate their execution. As this greatly decreases the time an artist must spend per asset. Which in turn increases its viability from a business standpoint.

Though the viability of such an implementation depends on many factors, like the complexity and required flexibility of the underlaying workflow. Furthermore, in the case of photogrammetry, on the amount and type of assets which needs to be processed. As not every object is suited equally well for photogrammetry and too much manual clean-up per asset might eat up the benefit of the automation again.

In addition to that, the limitations of the turntable must be considered within the requirements of the project. Though the size of the turntable doesn't matter for the automation toolkit, it does for the evaluation of its viability. If your common workflow involves the scan of objects form various sizes you will end up with a giant tedious to use piece of hardware, or one that's too small for many of the objects you like to scan.

So that the question how to increase the viability of Photogrammetry for smaller objects in a production scenario, really depends on the given production scenario.

Recommendations

The development of an automation toolkit like the one descripted throughout this report is a valid option for workflows, which offers a similar level of repetition. Tough this is only valid, if the workflow is also used frequently enough to justify the investment of development time for its creation.

Furthermore, the complexity and adjustability of the workflow should be considered beforehand. As well as the target audience of the tool. The development of a cli tool for a static workflow is a way less time-consuming task, than the automation of a pipeline, which requires an easy to use interface with the possibility to adjust the underlaying workflow.

These are also the things, which remain on the todo list for this project at the end of my graduation. Although the current iteration of our automation toolkit offers a well-polished user interface and robust asset generation pipeline, there are many more things to be improved in future versions. Like the integration of a user interface for the selection of different processing tool, as not every artist or studio prefers the same tools as we do.

In addition to that, the hardware could be extended for the usage of remote camera nodes, or automated camera stands and additional capture hats.

Usage – How it's being used

Lastly, I would also like to give a glimpse of a smaller project, that was already done with my automation toolkit at C4Real, before the hand in of this report.

Figure 28 *Asset Example Usecase*

C4Real did a small test project to scan in bugs via photogrammetry for the Museums factory in Hengelo, as case study for digital archiving & conversation of their bug collection.

Figure 29 *First Production Test Object*

Figure 30 *First Production Scan*

There is more for the museum to scan and the plan is to use further use the tool within the studio in the future, if a project could benefit from the usage of photo scanned assets.

References

Agisoft LLC. (n.d.). *Agisoft.* Retrieved from Agisoft Metashape: https://www.agisoft.com/

Allegorithmic. (n.d.). *Automation Toolkit Home - Substance Automation ToolKit.* Retrieved from Automation Toolkit Home: https://support.allegorithmic.com/documentation/sat

Autodesk Inc. (n.d.). *Maya | Computer Animation & Modeling Software | Autodesk.* Retrieved from Autodesk | 3D Design, Engineering & Entertainment Software: https://www.autodesk.eu/products/maya/overview

CapturingReality.com. (n.d.). Retrieved from CapturingReality.com: https://www.capturingreality.com

Community. (n.d.). *NW.js.* Retrieved from NW.js: https://nwjs.io/

DIMENSIONAL IMAGING LTD. (n.d.). *FIFA.* Retrieved from di4d: http://www.di4d.com/work/entertainment/fifa/

ImageMagick Studio LLC. (n.d.). *Convert, Edit, or Compose Bitmap Images @ ImageMagick.* Retrieved from ImageMagick Home: https://imagemagick.org

IMAGINE. (n.d.). *AliceVision | Photogrammetric Computer Vision Framework.* Retrieved from AliceVision: https://alicevision.github.io/

Jakob, W. (2015, 10 20). *Interactive field-aligned mesh generator* . Retrieved from Instant Meshes: https://github.com/wjakob/instant-meshes

Niehoff, M. (2016, 06 22). *What are beneficial use-cases for photogrammetry in a 3D game?* Retrieved from Martin-Niehoff: http://photogrammetry.martin-niehoff.de

Otwell, T. (n.d.). *Laravel - The PHP Framework For Web Artisans.* Retrieved from Laravel: https://laravel.com/

PlayCanvas Ltd. (n.d.). *PlayCanvas WebGL Game-Engine.* Retrieved from Playcanvas: https://playcanvas.com/

Quixel. (2014, 03 18). *megascans.se*. Retrieved from Quixel Megascans: https://www.youtube.com/watch?v=8alYZgkwClM

Ready at Dawn. (n.d.). *Crafting a Next-Gen Material Pipeline for The Order: 1886*. Retrieved from GDC Vault: https://www.gdcvault.com/play/1020162/Crafting-a-Next-Gen-Material

Wu, C. (n.d.). *VisualSFM : A Visual Structure from Motion System.* Retrieved from Changchang Wu: ccwu.m

You, E. (n.d.). *Vue.js - The Progressive.* Retrieved from Vue.js: https://vuejs.org/

Appendices

Appendix A: *"Beneficial use-cases for photogrammetry"*

2016

[PHOTOGRAMMETRY]

What are beneficial use-cases for photogrammetry in a 3D game?

Martin Niehoff, 22.08.2016
For the Game Creation and Producing at the Saxion University of Applied Science

Photogrammetry

What are beneficial use-cases for photogrammetry in a 3D game?

Written by

Martin Niehoff

info@martin-niehoff.de

As specialization topic in the Game Creation and Producing Course at the Saxion University of Applied Science

Supervising Teachers

Bram den Hond & Taco van Loon

22.08.2016

Summary

Photogrammetry is a technique, which is already around for quite some time, but it's also some kind of trend that gets used more and more in games recently. One reason for this might be the greater availability of VRAM for bigger textures, but certainly also the trend of creation immersive environments for Virtual Reality.

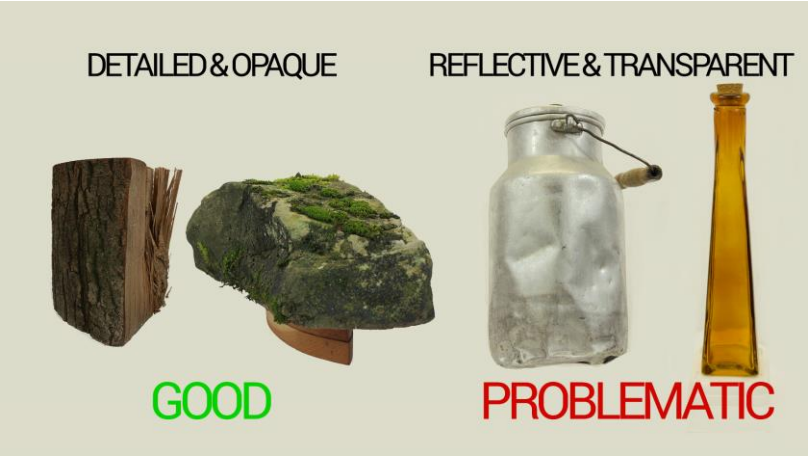
The term Photogrammetry describes a technique to reconstruct three dimensional scenes, or objects, from a series of overlapping photographs. As the accurate recreation of real world objects for games is a quiet time consuming process with a regular modeling & texturing workflow, it's interesting to see,

What are beneficial use-cases for photogrammetry in a 3D game?

Therefore, this paper is examining the differences between a regular modeling & texturing workflow and photogrammetry as alternative, or addition. To examine its influences on the production speed & quality. Furthermore, this paper shows the challenges, which arise while applying photogrammetry, as well as its advantages. It's not meant as an exhaustive, step-by-step guide to photogrammetric scanning, but rather an introduction and collected set of hints and tips.

The gathered data shows that photogrammetry can offer a delightful addition to a regular asset pipeline, if the project requires a certain type of assets, which is ideal for photogrammetry. It offers a valid alternative in situations, where objects with a lot of surface details, like wear & tear, or weathering effects are needed. These type of objects are ideal, as the captured photographs allow an exact recreation of these little influences in a very detailed and realistic manner.

However, when objects are not ideal for capturing, the entire capturing and processing gets way more time intensive for an artist. As its time consuming to compensate these problematic material types. This makes photogrammetry less usable for projects, which require a lot of objects with reflective, or transparent parts, due to the increased effort that the creation of certain object types requires.



Therefore, photogrammetry offers a nice addition to the regular modeling workflow, to increase the quality of certain assets. Tough, it's not advisable to fully replace the manual asset creation, due to the increased effort that the creation of certain object types requires. It's ideal use-case are projects, which require old objects with a lot of wear & tear and history on their surfaces. These additional surface details are not only looking most impressive with photogrammetry, also they simplify the scanning process.

Preface

First off I want to thank my teacher who convinced me to actually write this report about photogrammetry itself, as this allowed me to focus a bit more in detail on this awesome technology.

Furthermore, I want to thank the great team behind the Playcanvas engine, as they are building a great tool, without that I probably couldn't make all the interactive examples for this paper in the extend it is now. Of course not to forget, my colleagues who gave me throughput the creation of this report a lot of useful input.

Table of Content

Introduction

The gaming industry is quite rapidly evolving industry, that is always pushing for new technologies. One thing that is already talked about for years, is the so called “Photo realistic” game, or environment. Although we are closer to it than ever before, with all the stunning cinematic game trailers nowadays and the recent trend in the industry to switch to a “Physically Based Rendering” (Russell, 2016) method, which caused a big step forward in realistic real-time rendering, we are still quite a bit away from a fully photo realistic game.

Nevertheless, there is already a technique, that produces some “real” photo realism in games, admittedly in a different meaning, the so called “Photogrammetry”. Photogrammetry is already around for quite some time, but it’s some kind of trend that gets used more and more in games recently. Whether it’s to capture accurate facial likeness of athletes in FIFA (DIMENSIONAL IMAGING LTD, 2016), or to visualize real world locations as accurate, as possible, like in the VR applications “Realities” (realities.io inc, 2016). Photogrammetry is a technique, that allows to convert a bunch of photos of an object, into a textured 3D model. This model includes all the details of the real world object, like wear and tear and all the influences the world had over time on the object.

This is precisely, that makes photogrammetry so interesting and useful. It’s not only making a virtual copy of an object in form of a mesh, but furthermore it also captures the history of the object, in form of scratches, dirt and all that stuff, which texture artists need a lot of time and experience for, to accurately recreate it.

For this reason, this report is examining what exactly photogrammetry is, which boundaries and benefits it has and in which use-cases it offers a good alternative to the standard way of manually modeling and texturing objects from scratch.

> *What are beneficial use-cases for photogrammetry in a 3D game?*

As usual, there is no perfect, or only one way to look at a technique like this, as its usability always depends on the given use case and sometimes, personal preference. But this report is trying to give you a detailed inside into Photogrammetry, to allow you, to build your own opinion about it.

Reason

Photogrammetry is something I am interested in already for quite a while, as I found it always delightful to be able to put real world objects into a virtual environment.

Since I knew a bit more about the technique and the workflow behind it, it's something, that I wanted to try out a bit more in depth on my own, but never had the time for.

Therefore, this "free" time of my study to write this report offers the perfect opportunity for me to look a bit more in detail at Photogrammetry. As Photogrammetry seems to be used more and more in the industry recently and the company, where I was doing my internship is also planning to look into photogrammetry, the topic is more relevant than ever.

The basic requirements to start with photogrammetry are quite simple, you only need a camera and a software to convert your captured images into a 3d model and its texture. Tough there are always a couple more things to be aware of to get ideal results.

Therefore, I would like to have a look at the general boundaries and benefits of photogrammetry and look at its usability in a few different use cases to be able to draw a conclusion, for which type of project its suitable. So that I am able to recommend at the end of my research, how and when photogrammetry can be beneficial to be used in a project.

Preliminary Problem statement

Therefore the following preliminary problem statement was built:

> *What are efficient use-cases for photogrammetry?*

Theoretical framework

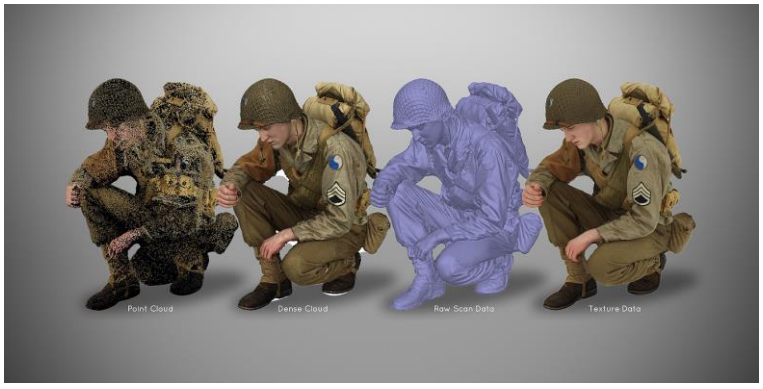
The theoretical framework is used to build a foundation for the further research done throughout this report. Furthermore, it defines the used terminologies and technologies for the reader to make sure they can follow the further explanations throughout this report.

Definitions

This report is going to look a bit more into detail at Photogrammetry, to conclude for which kind of projects, or use cases, it would be a beneficial alternative to the regular modeling & texturing workflow. This assumes of course, what both terms are actually standing for in the context of this report. Therefore, before getting started, lets clarify these definitions, to avoid misinterpretation.

What does the term Photogrammetry mean?

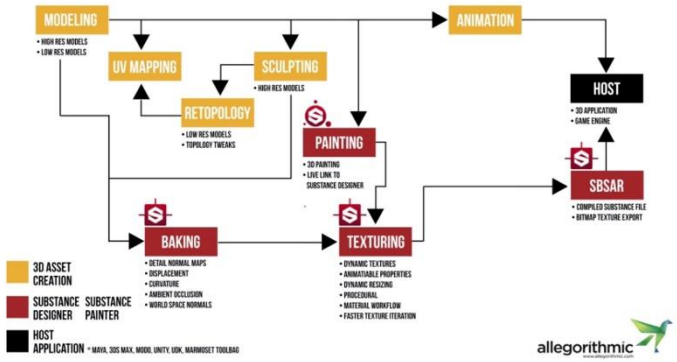
The term *"Photogrammetry"* stands for the technology to reconstruct three dimensional scenes or objects, from a series of overlapping photographs. It's used throughout this report as reference to the technique of generating 3D models & textures for games from a series of photographs. This includes the process of taking the images of the object, as well as its processing in a photogrammetry software and its optimization for the use in a game engine.



(FBFX Digital - What we do, 2016)

What means a regular modeling & texturing workflow?

The term "*regular modeling & texturing workflow*" used in this report stands for the workflow of creating three dimensional models in a 3D modeling application like 3Ds Max, or Maya (Polycount, PropsModeling - Polycount, 2016) and to texture them afterwards in Photoshop, or Substance (Polycount, PropsTexturing - Polycount, 2015) based on some reference images, or concept art.



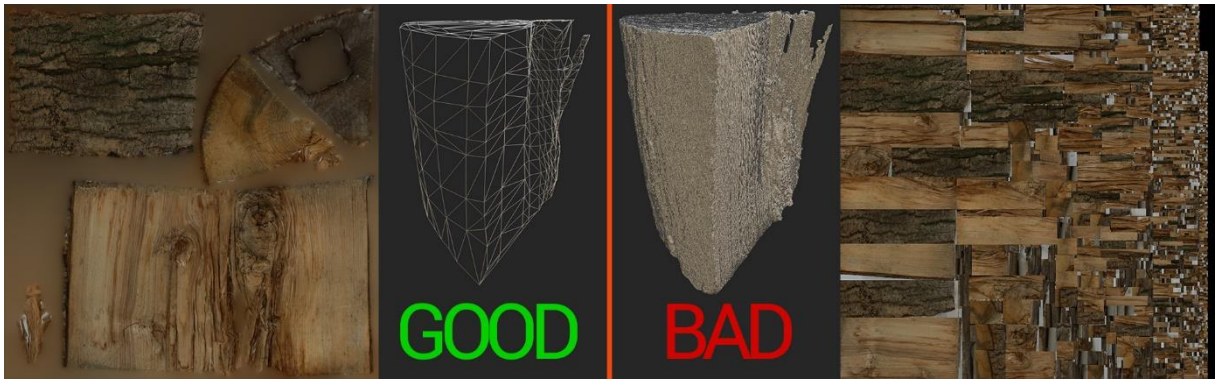
(Allegorithmic, 2016)

General Model Requirements

The number of different model & objects types is probably as high as the number of projects you can think of. It makes of course a difference if you are going to model a small object like a bucket, or try to capture an entire scene like a cathedral. Sometimes your objects will require a proper texture unwrap, but sometimes tile-able textures or shaders will be more beneficial for you.

To handle all this different use cases would of course blow up the scope of this report, therefore we are only going to compare a single type of objects to give a general overview about the boundaries and benefits of photogrammetry. Tough I will try to explain the different aspects of photogrammetry, as clear as possible, so it should be transferable to different use-cases and projects as well.

Therefore, this report is mainly focusing on the creation of smaller assets, to dress up a scene for a game. The kind of assets, that 3D artists have to create in masses to detail their game environments. These assets should have a game ready topology and a proper unwrap with PBR ready textures.



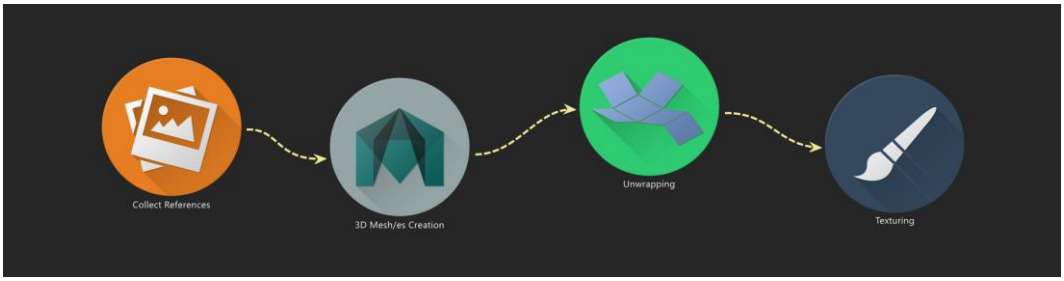
Game-ready means, that their polygon count stays is a reasonable and shape supporting extent, to be still render-able as part as complex environments in a real-time application, like a game. Furthermore, the UV-layout should be clean enough to avoid additional draw-calls.

Technique Comparison

Before we are able to compare both techniques, lets first have look at some example workflows for both techniques and what general advantages & disadvantages they have.

Regular Modeling & Texturing

Most 3D models for games are created manually (or with the help of some scripts & tools) in a specialized 3D modeling software like 3Ds Max, or Maya. Afterwards they get textured again by hand in a texturing suite like Substance, or Photoshop. This requires of course a lot of manual work for an artist and also a lot of experience, if the objects should have some realistic aging and wear & tear.



Example workflow:

An example workflow for this technique would be that;

1. Somebody has to create, or search reference photos of the object, which should be created.
2. An artist builds the 3D model in his modeling application of choice, based on the reference images.
    a. Depending on the type of model and quality requirements, a high- and a low-poly model has to be created, to bake down additional details onto the game-ready low-poly model.
3. The model has to be unwrapped by the artist.
4. Afterwards the model has to be textured in a separate application again, based on the provided reference images.

That's a rather simplified overview about the different creation steps without consideration of additional requirements as collision models etc. just to give an overview about the steps, which are interesting for our comparison.

Advantages & Disadvantages

Pros:

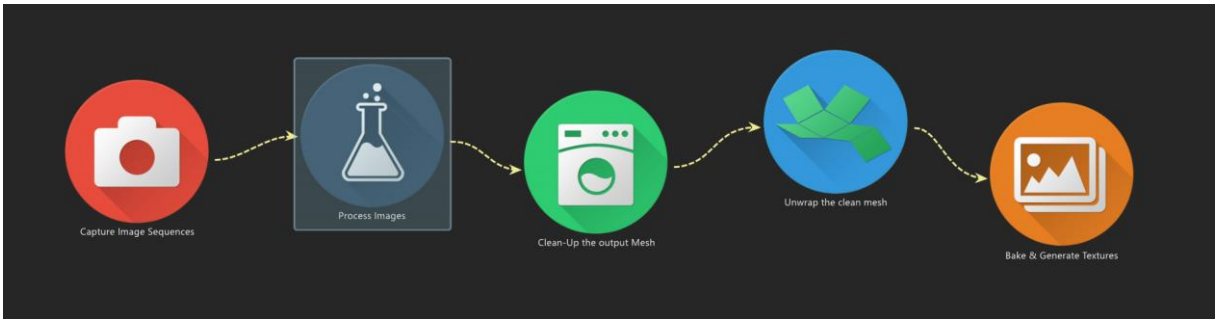
- Most 3D Artists know this technique.
- It does not necessary require additional hardware, or space to capture images of the objects, as enough images of most objects can be found on the internet as well.
- It works with all kind of objects.
- Fictive objects can be created based on concept art.

Cons:

- Requires a lot of manual work.
- Requires a lot of skill for the texture creation to not look too artificial.

Photogrammetry

For this technique a series of photos will be taken from different angles of a real world object. These photos will then be transformed into a point cloud by a special software, to be able to generate 3D models and textures from the images.



Example workflow:

An example workflow for this technique would be that;

1. An artist captures a series of photographs of the object.
    a. If necessary, the images will be cleaned up.
2. The photographs are taken into a photogrammetry software.
3. The output model of the Photogrammetry software gets re-topologized by an 3D Artist.
4. The re-topologized model has to be unwrapped by the artist.
5. Afterwards texture maps from the Photogrammetry Software can be baked down onto the re-topologized model.
6. Additional texture maps, like roughness and metalness have to be generated by a texture artist, based on the diffuse texture, with a software like Bitmap2Material.

On the first look, photogrammetry requires more steps in the creation process of a game ready model. However, some of these steps are pretty much automated, so that artists don't have to do too much manual work in the end. *(For more details see Research and Design)*

Advantages & Disadvantages

Pros:

- Can be automated to a certain extend.
- Generates accurate and realistic looking models with real world aging and wear & tear.

Cons:

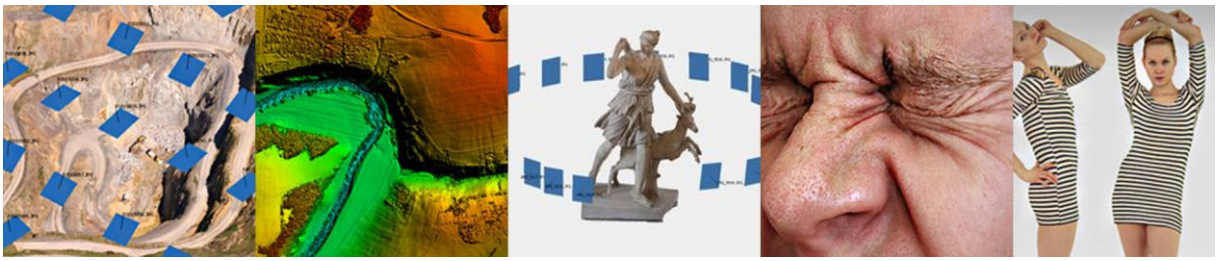
- Not every 3D artist has experience with this technique.
- It requires additional equipment like Cameras & sufficient Capture space.
- Not every object is scan able, like transparent or strong reflecting objects.
- You need the real world object at hand.

Background Information

Overview of Software

There are a couple of applications out there (wikipedia, 2016) to convert photographs into 3D Models, as usual, each with its own advantages and disadvantages.

These applications are reaching from specialized software to convert images from drones (UAVs) into meshes of entire landscapes, to software that aims at capturing facial expressions of real world characters as detailed as possible.



Composition of images from: (Agisoft LLC, 2016)

As this report is only explaining photogrammetry in general and not its use in a specific software I won't talk in detail about all the different software solutions, which are out there, but I would like to give you some insights from my research, so you can decide what's worth looking into for you.

The probably most used Photogrammetry Software out there for Game Development is AgiSoft Photoscan, which was used in games like "The Vanishing of Ethan Carter" (Poznanski, 2014) who did a phenomenal job. Photoscan offers an all in one solution to convert your images into 3D Meshes and textures in one go (except the re-topologizing ofc.), its relatively easy to use and offers an affordable small version, with enough features to start your photogrammetry project.

Another commonly used software is 123Catch from Autodesk, which performs all the heavy calculation in the cloud. Its available in a free version as app for your phone and pc, therefore it's nice to gain some first insights into photogrammetry (Autodesk, 2016). Grab your Mobile phone and an object and fiddle a bit around with it. Although it's nice to play around with the principle of photogrammetry it's not really useable for a game production, due to the limitations Autodesk gives you on the amount and resolution of used photographs. So if this report raised your interest in photogrammetry, give it a try, but don't expect perfect results.

A bit more complicated but more adaptable and free software is VisualSFM. With it alone you won't come far, as its just handling one part of the pipeline (Wu, 2016). In combination to this you will need a dense reconstruction tool like PMVS/CMVS and a software like "MeshLab" to process their output mesh in the end (you still need to re-topologize and bake this output to get a proper model). A good tutorial for using this bundle can be found here.

Although this setup might sound way more complex, than the use of AgiSoft (in fact it is), it also offers some very nice advantages. Its relatively modular, as things like the dense reconstruction tools it uses, can be simply swapped out and in addition to that it also offers a command line option as well as versions for Windows, Linux and Mac OSX.

That was actually the main reason why I looked more in depth into this software “bundle”. I like to automate as much repetitive work as possible, with closed source software that can be tricky sometimes. Furthermore, photogrammetry is quite computation extensive, therefore my work machine is unusable for a few minutes every time I start to calculate some meshes.

The command line options and multi-OS support of VisualSFM allows to build a custom render farm, by simply splitting parts of the process across different machines in the office.

Hardware Requirements

Photogrammetry has some requirements on the used hardware, although they are greatly influences by the quality level you want to achieve. In theory you can capture almost everything as good with your mobile phones camera as with an expensive DSLR Camera, but obviously the results will differ in the end.

Most of the previously mentioned Photogrammetry Software’s have own guides mentioning the required hardware to run their software, as well as the required capturing equipment. So that’s stuff you should check out, before you start shooting.

A rough general guideline is, that you should use a camera with at least 5 megapixels, to get useable images, while around 12 are ideal. It is reasonable to remember that the resolution of the input images influences the quality of the processing results, but also the calculation time. Therefore, don’t go for a much higher resolution, if you don’t miss information in your photographs, or you calculate for days on every single object. More important than the pure resolution, is the size of the cameras sensor and the option to lock its ISO, since photogrammetry works on identifying small details across photos, you want shots to be as sharp and as free from sensor noise as possible.

In addition to the camera, it’s also worth investing in a proper tripod or single pod, as blurred images are useless. A cheap alternative for a professional single stick is to simply mount a camera holder onto a telescopic bar from some gardening equipment.

Limitations of Object/Material Types

Due to the fact how Photogrammetry works, identifying small details across multiple photos, there are some limitations regarding the objects which are capture-able. While things like cliffs, or stone walls capture perfectly well, due to their natural roughness, it's hard, or impossible to capture shiny objects like cars, or transparent objects like glasses. This is caused by the fact, that the algorithm calculating the three dimensional models in the software is trying to match the pixels between the different images which are provided. If it finds enough matching pixels between the images, it will be able to generate a point cloud from them, but specular highlights and reflections will horribly confuse it.

Therefore, try to capture matte or opaque objects only, but if you really need to capture shiny or transparent objects, you can try to cover all specular highlights and reflections. Some ways to do this is painting the object, or to sprinkle calk dust over it, but keep in mind that you won't get any texture information from your objects this way.

Influence of Lights & Shadows

Another important aspect of photogrammetry to keep in mind, is the way you light your object while capturing it.

For objects, like props, that you want to place in different environments in your game, it's imperative to capture them without any direct light pointing at them. As this light would be baked into the texture in the end, which results in odd lighting, if not all objects in the scene are captured with the same light influence.

On the other hand, if you capture an entire scene, like a small room, this direct light influence might be exactly what you want. As this baked light will look very realistic (because it is real ;D) and does not require any additional light setup in your game, if you don't need any dynamic light sources.

Another thing to keep in mind, that applies to both mentioned use-cases, is to avoid strong shadows, or highlights caused by a wrong light setup. As each photograph needs comparable pixels with its neighbors in the image sequence, its self-speaking, that strong shadows, or highlights would confuse the comparing algorithm. The same part of the object would have different color values, or totally white or black pixels, that end up in artefacts, or skipped points in the point cloud.

Therefore, it's important to capture objects in as diffuse light as possible, to avoid any unwanted shadows, or strong highlights in the image sequence. If you want to capture smaller objects and have some equipment like a light diffuser to hand, it's advisable to light your object equally from all sides with them, to get rid of the shadows. If you don't have access to this hardware just try to avoid any direct light on your object, or capture you're your scene on a cloudy day, if the sun isn't shining too strong.

Definition of the problem

The previous research shows some limitations of photogrammetry regarding the type of objects and materials, as well as the surrounding in form of light and reachability by the photographer. Furthermore, its visible that with photogrammetry created 3D models can offer beautiful natural and irregular textures, which are hard to create for artists. In addition, the creation of certain meshes can be quicker with photogrammetry, compared to the regular modelling & texturing workflow depending on their characteristics and the available resources.

Main Question

Therefore, the following main question can be formed:

What are beneficial use-cases for photogrammetry in a 3D game?

Sub Questions

Now after our main research question is formulated, we have to find out, what is needed to answer it, while staying in the scope of this report. Therefore, we have to clarify a few things up front:

- *How useful is photogrammetry for scanning objects that are used in a 3D game?*
- *What kind of objects are suitable for photogrammetry?*
- *What challenges are there when applying photogrammetry?*

With answers to these sub questions it's possible to conclude which technique can be preferred on a given use case. Although as the previous research showed, that both techniques have their own limitations and ideal use cases, this paper is not going to give a fixed rule, but rather a recommendation for certain use cases, in which one technique could be preferred over the over.

Scope

The scope of this report is bound to the research, that is required to recommend one of the listed asset creation techniques, for certain use cases, or project requirements.

Factors which are influencing this recommendation are the technical limitations and advantages of both techniques, as well as their usability for the artist.

Points that are out of the scope of this report, but could have influenced the recommendation of this report are, although briefly mentioned, partly automated asset pipelines with self-written scripts, or additional software packages and additional hardware like automated turntables.

Research and design

The main research method, which is used to gather the data required for the recommendation of this paper is to validate the gathered background data with some praxis oriented test cases.

These test cases are built to answer the list of sub questions, which raised while collecting the background information needed for this research. While the collection of these background information helped to figure out the more and less important aspects of the compared asset creation workflows, they give no clear answer to the main question. Therefore, the test cases are used to verify the collected theoretical data and also to give a more in depth answer to the given main question. In addition, they offer some interesting content to visualize the more theoretical parts of the investigated techniques.

To be able to find out the pros and cons of photogrammetry I selected some best and worst case scenarios based on my previous research and looked for suitable objects to represent each of these scenarios. These scans should help to verify the previous made research, or to find additional point of interest.

I ended up with the following objects for the different scan scenarios:

1. **Wooden log** – Represents an ideal use case for photogrammetry due to the detailed organic shape. In addition to that its easy moveable, thus simplifying the scanning process.
2. **Owl Decoration** – Represents a more problematic, but still doable use case for photogrammetry due to the transparent parts of the object.
3. **Milk Can** – Represents a bad, but still doable use case for photogrammetry due to the strong reflection and smaller parts of the surface.
4. **Mossy Stone** – A comparison of the effect, that the used camera resolution has on the output model and its texture.
5. **Woodwaste** – Represents additional challenges, which arise if scanning a not moveable objects outside.

Note: Additional use cases that I haven't covered here, but I recommend people to look into, if they are interested in photogrammetry. The scan of fully transparent objects like glasses, or the scan of room with interiors, or building facades.

A positive side effect of building the test projects is that their creation allows a basic comparison of the usability of both workflows in a production environment. As its generally not feasible to use a technique, even if it offers slightly better results, if it's too hard to use, or too time consuming. Therefore, this is another important aspect, that should be taken into account for a recommendation.

Research results

1. Wooden log

A wooden log is a relatively easy object to scan, so it is a quite good starting point to get used to the process of photogrammetry, while learning the dos and don'ts in an easy way.

A split block of wood as the one shown on the side offers quite a few interesting things for photogrammetry.

- It is relatively small and lightweight, what makes it easier to find a good place to scan it. (You don't need a big room, or think about the weather conditions)
- It has an interesting organic surface with a lot of details, which would require quite some time to sculpt by hand. In addition, this is beneficial for the photogrammetry software, as it's easy to find matching points.
- The splintered parts on the side show nicely some problematic shapes for photogrammetry, as they can produce some floating parts in the geometry, if they get too small.

Note: I made two scans from of the log, one upside down to be able to fully capture all sides of the log. Their output was merged afterwards to have a fully captured model from all angles. Both scans container 3 image rows, top, mid and bottom of 24 images per row.

3D-Viewer:

https://playcanv.as/p/FLpNI0EL/?object=wooden-log

Conclusion

A Split piece of wood offers a perfect use case for photogrammetry, because the scan process is a quick and easy task, due to the handiness of the object. Furthermore, the object offers an interesting and detailed surface, that would require a lot of time to manually create it. The only thing to watch out for are small splintered parts, as they can result in floating points and scan errors.

2. Owl Decoration

A small decoration object like the solar owl lamp on the side shares the handiness and surface detail (the feathers add some nice surface detail) of the above shown wooden log. Additionally, it has a particularity, that makes it a bit harder to proper capture it.

- The eyes and the solar panel on the object are transparent/semi-transparent, which produces artifacts and noise areas on the output mesh.
- A simple solution for object like this is to cover the transparent/semi-transparent parts of the object with something opaque, that you can easily remove later on. This way it's possible to capture at least the surface of these objects as well.
- Planar transparent parts like the ones on this owl, can be easily textured manually afterwards by the use of an additional photographed texture, which can be projected onto the mesh again in an application like Substance Painter.

Note: This mesh is made with a single scan of 3 images rows (top, mid, bottom), with the transparent parts covered by some duct tape. These parts are afterwards manually textured with the help of additional images of the affected parts.

3D-Viewer:

https://playcanv.as/p/FLpNI0EL/?object=owl-decoration

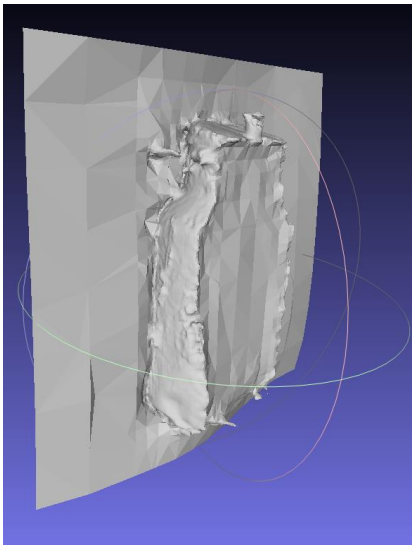
Conclusion

The transparency of objects is a bit problematic for photogrammetry, as it causes artifacts and unclear surface definitions, but with a few tricks it's also possible to capture these types of objects. Although if it's worth the effort, depends on the complexity of the object. This owl allowed a relatively quick and easy manual texturing of the transparent objects, because all transparent parts are planar. For more complex shapes this manual texturing might not be as easy anymore and therefore might reduce the advantage of scanning the object.

3. Milk Can

An old dented milk can like the one on the side might looks like a nice object for photogrammetry, as its quite handy and shows some nice wear and tear, which shows some nice history. Although the shiny material makes it a bit harder to capture, than it might look like.

- Due to the reflectivity of the surface the light in the capture area has to be reduced, to avoid strong highlights and reflections.
- The reduced light intensity requires adjustment of the camera to still get bright enough images.
- Small parts like the wire of the handle, offer similar problems as the splintered parts of the log shown above.
- As the surface of the can is beside the dents quite detail less, it's not offering a lot of matching points, which the photogrammetry software could work with. Therefore, it's hard for the software to recreate the proper surface into a mesh, this makes the post processing more complex, as the point cloud tends to produce output meshes like the one on the right.
- If it's the goal to only capture the surface, without its texture detail, the capture process of this type of objects can be greatly simplified with covering up the surface, like shown on the owl above.



Note: This mesh is made with a single scan of 3 images rows (top, mid, bottom), but it required a few test runs to find the right mix of light intensity and camera settings. Furthermore, the image post processing required a lot more time than on the other models shown above.

3D-Viewer:

https://playcanv.as/p/FLpNI0EL/?object=milk-can

Conclusion

To capture reflective or shiny objects like this milk can is still possible with photogrammetry, as long as they offer enough surface detail and don't reflect too much of their environment. So that the software can find enough matching points, to proper reconstruct its surface. Although the capturing process of these objects tend to be way more complex, than for less reflective objects. It requires way more thinking about the proper light setup and image correction, to take good images for the use in a photogrammetry software. This makes the overall process more complex and less useful for simpler objects compared to a manual modeling and texturing workflow.

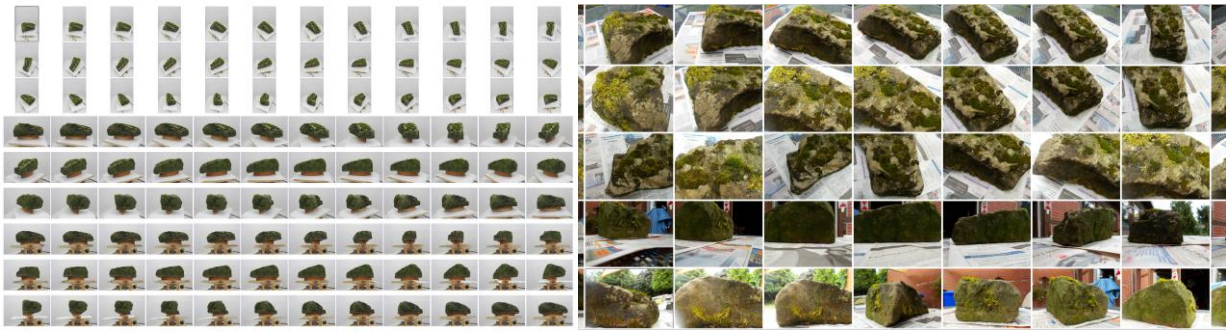
4. Mossy Stone

A mossy stone like the one shown on the side offers a quite ideal use case for photogrammetry, as the moss and weather influences offer cool and unique surface details with a lot of match-able points.

That's in fact one of the reason, why it was one of the first objects, which I tried to scan with an old compact camera I still had lying around. After a while I switched to a newer DSLR camera and asked myself, what influence the camera actually has on the output quality, so I recaptured the stone with the DSLR and made a comparison of both output meshes.

- The overall shape is perfectly matching with both cameras, though the higher resolution of the DSLR adds a bit more micro definition to the mesh.
- The higher resolution of the DSLR allows the creation of a texture with a slightly higher end resolution, which is beneficial for close up, or hero models, but it also increases the computation time.

Note: The two meshes were not captured with a fully identical setup, as I captured both scans with different cameras and some temporal distance between the shots. In addition, the capturing setup differs a bit, since I made the first scan on our garden table, while the second one was made in a lid studio. If somebody is interested in actually comparing the influences of different aspects of used cameras, like their resolution, more in depth, I suggest to do some more comparable scans with identical environment setups.



3D-Viewer:

https://playcanv.as/p/FLpNI0EL/?object=mossy-stone

Conclusion

Although the two scans of the above shown stone are not fully identical, due to different camera models and environment conditions, they show some advantages of using a bit more advanced camera for photogrammetry. But they certainly also show that it is not necessary required, to use an expansive DSLR camera, if you don't plan to go too close up to the scanned meshes. Tough a DSLR certainly offers more options to the user while shooting, which might be helpful while capturing larger amounts of images.

5. Woodwaste

After I captured quite a few smaller objects, which I could easily move around and light up correctly, I was a bit more curious about the additional challenges which arise, while capturing bigger objects outside. So I went outside and started to look for something interesting to capture. I found an old pile of rotten wood, which was ideal, as I its surrounding was relatively open, so I could easily move around it.

A few things I have noticed, which should be considered while capturing objects outside:

- Watch the weather forecast, as its really annoying to do the same capture twice, just because it starts to rain 5minutes before you got all your images. The sun can get really annoying as well, if it appears every second image behind a cloud.
- A single pod to mount your camera makes capturing way easier and quicker.
- For outside shooting it's even more important to make a few more images, than you think you need, as it's hard to get equal lighting conditions again. So it can happen, that you have to reshoot everything, even if you only need one additional image.

3D-Viewer:

https://playcanv.as/p/FLpNI0EL/?object=woodwaste

Conclusion

Capturing objects outside can make quite some fun and offers a unique opportunity to add objects with a lot of weathering effects to your virtual environments, but it can also be a bit tricky and frustrating. A lot more factors have to be thought of, which you cannot necessary control, like the weather at the capture scene.

Conclusion and discussion

The use of photogrammetry offers a valid alternative for the regular modeling workflow in situations, where objects with a lot of surface details and/or wear & tear, or weathering effects with a realistic appearance are needed. These types of objects are the strength of photogrammetry, as it shows all these little influences in a very detailed and realistic manner. In addition, the capturing process of these object types is relatively quick and easy.

However, when objects don't have a lot of surface definition, or problematic material types like reflective, or transparent parts, the capturing and processing gets way more time intensive. Furthermore, the results of these objects tend to be less impressive, as stuff like glass has to be faked with shaders. Moreover, actual surface detail can be lost, because the photogrammetry software cannot find enough matching points on these material types, to reconstruct its surface details.

While this shows that the use of photogrammetry offers a nice addition to the regular modeling workflow, to increase the quality of certain assets. It also shows that it cannot fully replace the manual creation of assets from scratch, due to the increased effort that the creation of certain object types requires. So that the usability of photogrammetry remains a question of the project requirements, rather than limitations of hard- or software, as the starting barrier with the variety of payed and free software with little hardware requirements is quite low.

Therefore, beneficial use cases for photogrammetry are projects which include objects with mostly opaque and rough surfaces. The more surface details an object has, the easier the scanning gets. The strength of photogrammetry is the scan of old and worn down objects with a lot of wear & tear and history on their surface. As the virtual recreation of these influences is the easier and more impressive usage of photogrammetry.

> As mentioned earlier, this article is not meant as an exhaustive, step-by-step guide to photogrammetric scanning, its rather an introduction and collected set of hints and tips. Therefore, the use of additional equipment or software could influence the quality of certain scans.

Recommendations

The use of Photogrammetry offers a valid addition to a regular modeling workflow for projects, which:

- Require objects with a lot of surface details and/or wear & tear, or weathering effects with a realistic appearance.
- Don't contain a lot of objects with reflective, or transparent parts.

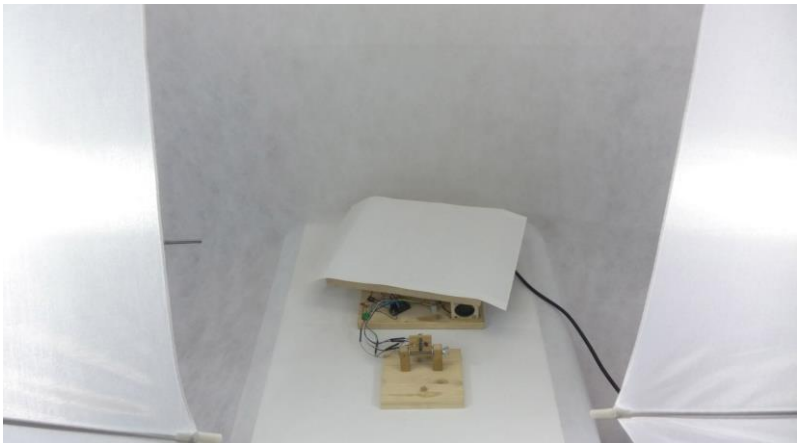
Furthermore, it's advisable to start with easier objects like the shown wooden log and some simple available equipment and then to gradually increase the difficulty of the objects to get a feel for the things, which might work, or don't work. Before spending a lot of money on additional equipment.

For rough scans of normal props, a simple camera with 5-9 megapixels should be okay for an up to date game, but if close-up, or hero models should be scanned in its useful to have a few megapixels more to get some additional surface details into the scans. Moreover, the more control you have over your camera, the better. As the ability to manual focus the target object and the ability to adjust the ISO and other options makes the capturing process way easier.

For scanning smaller moveable objects, a turntable is highly recommended, as it will greatly increase your capture speed. Furthermore, some studio lights will allow you to lit up your objects more evenly, so you can capture them regardless of the current lighting. In addition, a green screen will make your life a bit easier, as it allows you to cut off the background, which makes the post processing a lot easier.

What is important, however, while using a turntable is that you don't capture a lot of the static background. As the background would produce more matching point than your object on the turntable, which will result in broken scan. A common trick is to put a piece of newspaper under the object, so that bigger parts of your image are moving with the object. A slightly reflecting piece of paper works even better, as most photogrammetry applications cannot reproduce it, so that you don't have to cut it away afterwards. Another good trick is to lift your object a bit with some kind of pillar, to capture the detail on the bottom of the object. Furthermore, it helps to avoid ambient occlusion on the lower part of the object.

To get as much information into your scans as possible its recommended to do three rows of image sequences of your object. Once from a high angle, once directly from the front and once from a lower angle. Each row should contain a 360-degree sequence with images taken every 5 – 15 degrees depending on the complexity and desired amount of surface details.



For capturing outdoor scenes, or objects its quite important to plan in the weather and sun conditions on the planned scene, to avoid unnecessary breaks, or additional shooting days. Ideal is an overcast, or cloudy day with defuse light. Furthermore, it's advisable to shot always a couple more images, than you actually need, to have some angles in reserve for the case you miss some parts in the reconstruction days after the actual shooting. In addition, a single stick with an adjustable height is a great helper for capturing bigger objects from different heights, it also allows you to quicker capture sharp images, than with a tripod.

Graduation Products

The prototype created for this paper can be found under the following link:

https://playcanv.as/p/FLpNI0EL/

References

Agisoft LLC. (2016, 08 15). *Agisoft Photoscan.* Retrieved from Agisoft: http://www.agisoft.com/

Allegorithmic. (2016, 06 16). Retrieved from https://www.allegorithmic.com/: https://www.allegorithmic.com/

Autodesk. (2016, 06 30). *123 Catch How To.* Retrieved from 123dapp.com: http://www.123dapp.com/howto/catch

DIMENSIONAL IMAGING LTD. (2016, 06 14). *FIFA.* Retrieved from di4d: http://www.di4d.com/work/entertainment/fifa/

FBFX Digital - What we do. (2016, 08 16). Retrieved from FBFX Digital: http://www.fbfxdigital.co.uk/#!what-we-do/c71j

Polycount. (2015, 01 22). *PropsTexturing - Polycount.* Retrieved from Polycount: http://wiki.polycount.com/wiki/Category:PropsTexturing

Polycount. (2016, 05 27). *PropsModeling - Polycount.* Retrieved from Polycount: http://wiki.polycount.com/wiki/Prop_Modeling

Poznanski, n. (2014, 03 25). *VISUAL REVOLUTION OF THE VANISHING OF ETHAN CARTER.* Retrieved from theastronauts.com: http://www.theastronauts.com/2014/03/visual-revolution-vanishing-ethan-carter/

realities.io inc. (2016, 06 14). *realities.* Retrieved from realities: http://realities.io/

Russell, J. (2016, 06 14). *PBR Theory | Marmoset.* Retrieved from Marmoset: https://www.marmoset.co/toolbag/learn/pbr-theory

wikipedia. (2016, 05 26). *Comparison of photogrammetry software | wikipedia.* Retrieved from wikipedia: https://en.wikipedia.org/wiki/Comparison_of_photogrammetry_software

Wu, C. (2016, 06 30). *VisualSFM : A Visual Structure from Motion System.* Retrieved from ccwu.me: http://ccwu.me/vsfm/doc.html#usage

Additional Resources:

http://steamcommunity.com/games/250820/announcements/detail/117448248511524033

http://steamcommunity.com/games/250820/announcements/detail/117448248511523471

http://steamcommunity.com/games/250820/announcements/detail/117448248511522945

http://www.photogrammetry.com/

http://www.geodetic.com/v-stars/what-is-photogrammetry.aspx

http://www.zarria.net/nrmphoto/nrmphoto.html

http://www.gdcvault.com/play/1020162/Crafting-a-Next-Gen-Material

Appendix B: *"Photogrammetry Asset Pipeline"*

Photogrammetry Asset Pipeline

Overview about the different steps in my photogrammetry pipeline.

Preparation

- Prepare the equipment, like a turntable, lights, diffuser
- Calibrate the Camera, take a reference image with a color checker
- Prepare the object
    - Cover it with matte powder, clean it, or make it dirty/wet to match the desired style

Image Generation & Processing

- Create a LUT from the reference image
- Convert all raw images into calibrated jpegs
    - Apply LUT
    - Remove Vignette, keep lens distortion
    - Apply Naming Convention
- Generate Point Cloud
    - Manual/Automatic, Mask setup if needed
    - Export, Sparse & dense cloud
    - Generate Highpoly Mesh with textures

Data Generation

- Generate a lowpoly mesh via one of the following methods:
    - Decimation
    - Wrapping
    - Retopology
- Generate an optimized UV Map
- Bake required texture maps with
    - Substance, or
    - Knald/xNormal for better/quicker results
- De-lightify the albedo map of the highpoly mesh with Substance

Asset Generation

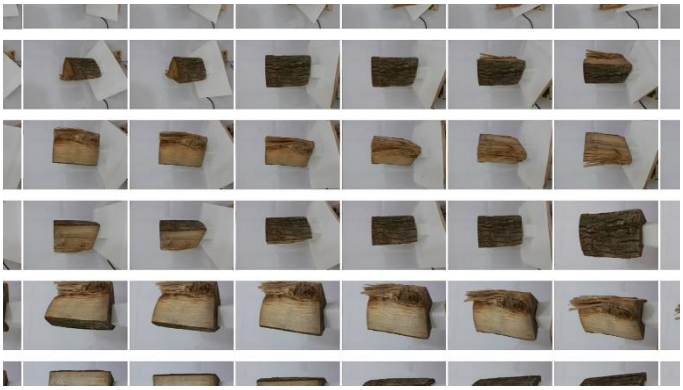
- Generate a game ready texture pack based on the previously baked textures with substance
- Render a nice preview image
    - Optional but nice for an asset library/browser

Appendix C: *"Photogrammetry Benchmarks"*

Photogrammetry Benchmarks

Following is an overview about a couple of benchmarks I did to investigate the influence of the used hard- & software on the production time of a photo scanned asset for my graduation research.

This isn't meant as a comparison between the used applications, nor a fully accurate comparison of their performance, as this would require the usage of more workstations and additional tests to gain a solid dataset. Sadly, this wasn't possible within the scope of my research due to hardware & time constraints.



All tests have been made with the same image sequence, a simple turntable capture of a wooden log. Furthermore, each process was repeated 3 times per machine, the shown data below is an average of those data sets. All applications used their default presets.

You can find it as attachment to this report, if you are interested in a comparison with your own setup.

I used 2 different machines one desktop pc and one laptop, for a comparison between "VisualSFM" (Wu, n.d.) in Version 0.5.26, "Agisoft Photoscan" (Agisoft LLC, n.d.) as Demo Version 1.3.2 and "RealityCapture" (CapturingReality.com, n.d.) as Demo Version 1.0.2.2810. I used the same ssd, with a fresh windows 10 installation on each of the test to avoid other software distorting the tests.

The exact specs of the used computers are listed below.

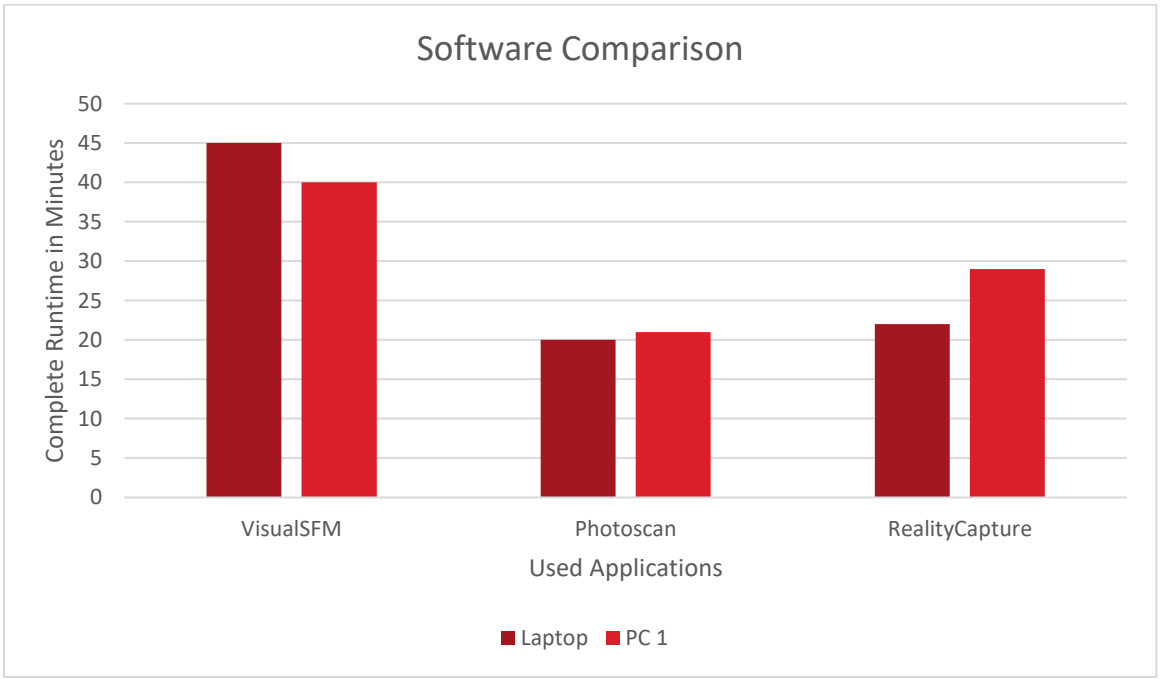
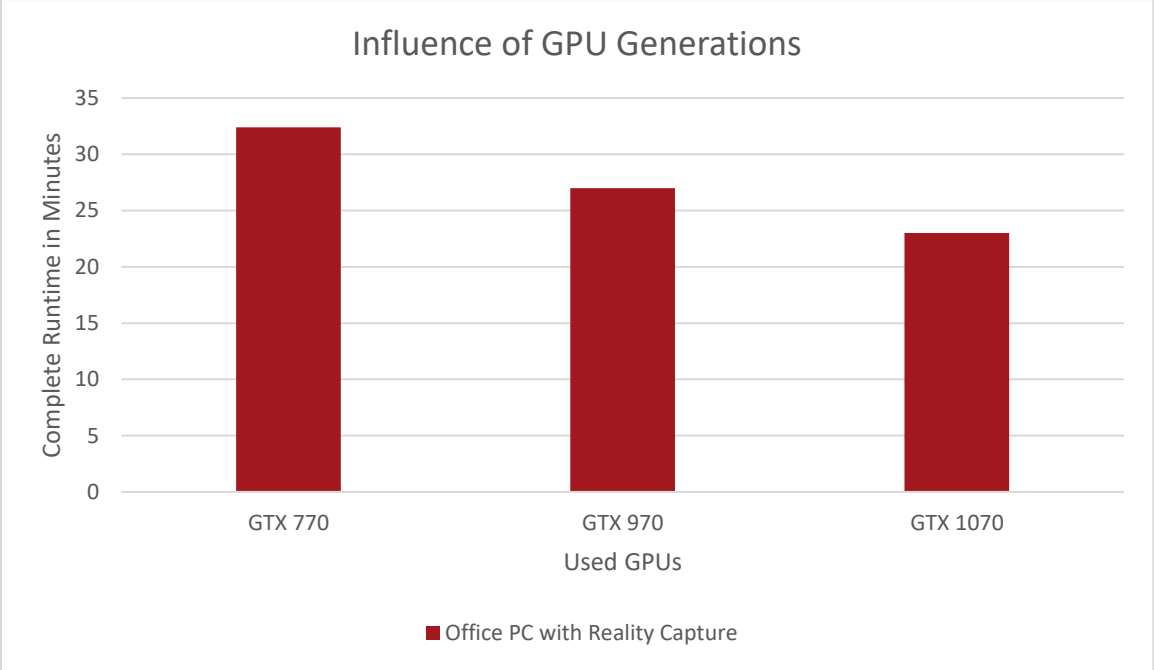
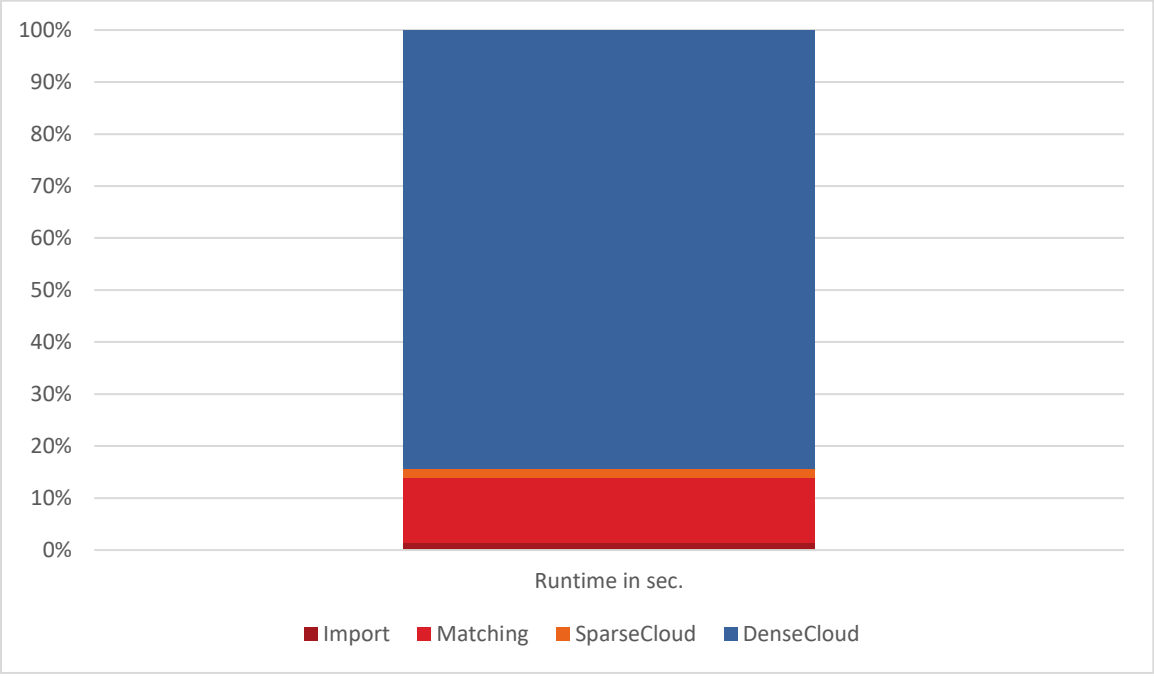


***Figure 1** Software Comparison Benchmarks*

Afterwards, I compared what influence the usage of different GPU generations has on a cuda enabled software like reality capture. I though this would be another interesting test to do, as I had access to 3 generations of Nvidia GPUs of the same price segment in the office.



However, more important than the overall computation time of an asset, are the required times per subtask for my research. Therefore, I looked at their individual runtimes with VisualSFM, as its with its open source nature seemed to be the most suitable application for my purpose.



It shows clearly that in the case of VisualSFM most of the generation time is spend during the creation of the dense cloud, which matches the results of the other applications altough their runtimes differ.

Device Specs

Laptop:

i7 4600MQ 8GB Ram & Gforce765m

PC 1

i5 3570k 16GB Ram & GTX 770

Office PC

i7 920 @2.67GHZ & 24GB RAM

Appendix D: *"Software Requirements"*

Software Requirements

Now as we know, what functions we would like our turntable to have & what hardware we require for it, we can define, which applications we need for this, and what requirements they have. Additionally, we can plan what custom applications we need to develop on our own.

Functionality

The requirements for our base functionality are quite short, we only drive our stepper for the turntable & need to trigger our connected cameras, therefore we need:

- Stepper controller software, gPio library
- Camera controller library

Usability

The list of requirements for our usability is already a bit longer.

- Web server, to allow access to an easy user interface
- Storage server, how do we storage & transfer all the data?
- Slave control, how do we manage all our render nodes?
- Update functionality, how can be update the used applications?

This list excludes basic requirements, like the drivers or system components, as these should come with the used board.

Custom Developments

To streamline the workflow with our device, we will certainly need to develop a couple of own applications to glue everything together, these include:

- Web Panel, to manage the entire device
  - Scan Control
  - Scan Preview
  - Scanner Settings
  - Asset Browser
  - Pipeline settings (Used software & naming conventions)
- Slave control, to manage our render & processing nodes
- Slave client, computes the from our control node given tasks on the host device

Software Functions

Overview about the planned features, their requirements & estimated implementation times, priorities as MoSCoW list. [M, S, C, W].

Our pipeline will be primarily based on five custom developed applications, which link everything together, that is required for a smooth workflow.

Web Interface

The turntable will be fully controllable over a custom front end. This application combines the other parts of our software stack and allow an easy interaction for the end user, during scans.

- 3D Window - M
  - Indicate current scanning step on a 3d turntable - M
  - Render Preview Mesh - S
- Scan Control Panel - M
  - Start Scan - M
  - Pause Scan - M
  - Abort Scan – M
  - Options Menu – M
    - Presets Selection/Save - S
    - Number of images per row - S
    - Number of rows - S
    - Manual/Multi Camera or stand selection – S/ C
    - Preview Toggle - C
      - Slave selection - C
      - Quality Option - C
    - Asset Generation Toggle - C
      - Select Generation Preset - C
- Scan Statistics - S
  - Display Selected Settings - S
  - Show estimated scan time & current time - M
  - Display scan meta, like required storage size - C
- Scanner Controls - M
  - Shutdown - M
  - Reboot - M
  - Network Settings - M
- Pipeline Settings - S
  - Allow Presets - S
  - Define Used Software Suites for - C
    - Point Cloud Generation - C
    - Map Baker - C
    - Texture Generation - C
    - Low-poly Generation/decimation/remeshing - C
    - Preview/Thumbnail Rendering - C
  - Customize naming conventions - S

- Storage Settings - S
  - Internal storage - S
  - NAS connection - S
  - Online provider integration? - C

❖ Slave Control - S
- Create/Edit/Remove Nodes - S
- Assignee Node tasks - S
- Give nodes priorities (which node should be used first) - C
- Give Nodes, render weight (to balance stronger/weaker nodes) - C
- AWS integration? - C

Turntable Driver

The turntable driver is a small python based custom application, which is used to control the stepper motor, which is connected to our driver unit. The reason, why I separated this logic into an independent application, is that it's easier for me to develop & maintain hardware related application in python, than with php, which is used for the front-end.

❖ Setup the gPio pins - M
❖ Receives its configurations via command line properties, to make its front-end connection easier - M
❖ Calculates the degrees per step, based on the desired amount of images - M
❖ Auto Home option? – C

Slave Client

Our slave server listens for commands from the compute node & executes their commands on its host device. This includes primarily command line launches of computation applications, with the from our turntable provided files.

❖ Listen for commands over the network - M
❖ Receive/ gather required files over the network - M
❖ Launch computation applications - M
❖ Send computation results over the network to the target storage - M

For cross platform combability it could be a good idea to develop this application with nodejs and wrap it into a custom application via nwjs.

Turntable Manager

The turntable manager handles the interaction between the user interactions/configurations from the web interface and the executing applications, like the turntable driver and slave client on the backend. For simplicity, this application is written in php, as this was a more comfortable choice for me, to implement the desired features in the given timeframe.

- Store Turntable & Pipeline Settings - M
- Generate Statistics/Meta Data - C
- Manage Scan tasks - M
- Manage Slaves - S
- Manage Slave Tasks - S