

3D Copy&Paste: A Photogrammetry solution for 3D scanning of salient objects

Jan Jiménez Serra, B.Eng.

A Dissertation

Presented to the University of Dublin, Trinity College
in partial fulfilment of the requirements for the degree of

**Master of Science in Computer Science (Augmented and
Virtual Reality)**

Supervisor: Gerard Lacey

August 2021

Declaration

I, the undersigned, declare that this work has not previously been submitted as an exercise for a degree at this, or any other University, and that unless otherwise stated, is my own work.

Jan Jiménez Serra

August 31, 2021

Permission to Lend and/or Copy

I, the undersigned, agree that Trinity College Library may lend or copy this thesis upon request.

Jan Jiménez Serra

August 31, 2021

3D Copy&Paste: A Photogrammetry solution for 3D scanning of salient objects

Jan Jiménez Serra, Master of Science in Computer Science
University of Dublin, Trinity College, 2021

Supervisor: Gerard Lacey

Current 3D scanning solutions for single objects with hand-held devices require a high level of technical knowledge and expensive external devices. An alternative solution is to generate 3D models from images taken on a phone. This is known as Photogrammetry, which is a tedious process often done manually. It also incorporates unwanted background data to the 3D reconstruction of the model.

This dissertation presents an end-to-end 3D scanning solution which automates Photogrammetry with a mobile device. The solution comprises three interconnected modules. First, a mobile application, called "3D Copy&Paste", is used to obtain images from the camera sensor of the phone. A server then automatically removes the background from the input images using deep-learning based salient object detection, the resulting images are then used to generate a 3D object using Photogrammetry. Finally, an add-on for a desktop 3D software is able to import the generated model seamlessly.

3D models of single objects without background were generated at least 30% faster with the tested Photogrammetry pipeline. Also, with an average of 96% scale accuracy, this solution provides faster and lower cost 3D scanning in mobile devices with similar quality to the State of the Art. The code of the implementation can be found online at: <https://github.com/Janjs/3DCopy-Paste>.

Acknowledgments

My sincere thank you to my supervisor, Gerard Lacey, for the valuable guidance, assistance, and providing the hardware that I needed. I would also like to thank my family for their support.

JAN JIMÉNEZ SERRA

*University of Dublin, Trinity College
August 2021*

Contents

Abstract	iii
Acknowledgments	iv
Chapter 1 Introduction	1
1.1 Motivation	1
1.2 Salient object detection for single object 3D reconstruction	2
1.3 Structure of the dissertation	2
Chapter 2 State of the Art	4
2.1 3D Scanning technologies	4
2.1.1 Contact	5
2.1.2 Non-contact	5
2.2 Mobile Applications	6
2.3 3D Modeling	7
2.4 Lidar	7
2.5 Photogrammetry	8
2.6 Salient Object Detection	11
2.6.1 Image segmentation	12
2.6.2 U ² net	13
Chapter 3 Design	16
3.1 Alternative Approaches	17
Chapter 4 Implementation	19
4.1 Photogrammetry server	19
4.1.1 Salient object detection	21
4.1.2 Background removal	21
4.1.3 Photogrammetry solution	22
4.2 API	22

4.3	Mobile application	24
4.4	3D software Add-on	26
Chapter 5 Evaluation		28
5.1	Experiments	28
5.2	Results	29
5.2.1	Lidar vs photogrammetry	30
5.2.2	Background removal	31
Chapter 6 Conclusions & Future Work		34
6.1	Future Work	34
Bibliography		36

List of Figures

2.1	Example of passive non-contact scanning.	5
2.2	Using Qlone 3D scanning mobile app.	7
2.3	LiDAR sensor vs face recognition sensor on new Apple devices.	8
2.4	Photogrammetry example.	9
2.5	Representation of how SfM works.	10
2.6	Meshroom node-based system.	10
2.7	Difference between types of segmentation problems.	12
2.8	Illustration of U ² net architecture.	14
2.9	Results of U ² net.	15
3.1	Flow of information	16
4.1	Technical Structure	20
4.2	Feature matching to extract centroid	23
4.3	Flow iOS app	24
4.4	App Gallery View	25
5.1	Compared 3D model with and without its background removed	30
5.2	Compared 3D model from our solution against a professionally done 3D reconstruction.	31
5.3	Compared 3D model generated with LiDAR vs Photogrammetry.	32
5.4	Limitations of SOD in combination of Photogrammetry.	33

Chapter 1

Introduction

A salient object is the most noticeable or important object in a scene. 3D scanning is the process of generating a 3D representation of the real-world as a mesh of vertices. Often, after 3D scanning of salient objects, there is a manual post-processing step of removing unwanted background data. This requires technical knowledge of 3D modeling tools.

3D scanning also has a monetary barrier, where expensive hardware and software is required.

There are two main objectives presented in this dissertation that aim to solve these issues. First, to create an automated solution that generates 3D scans of single salient objects. Second, to make the solution work with widely available hardware. That is why a mobile application that improves the state-of-the-art 3D scanning apps is presented. The app, called "3D Copy&Paste", is an iOS app capable of generating 3D models with the help of a server. It also allows that 3D model to be exported to Blender [1], a 3D modeling software, to further automate the 3D scanning process outside of the 3D reconstruction itself.

A proof-of-concept project was made for the Masters Augmented Reality module (CS7GV4) using a LiDAR sensor from a 2020 iPad Pro. This project build off of that, but with a different approach, mainly because that LiDAR sensor is not powerful enough, nor is widely available on mobile devices.

Instead, the 3D scan is generated using Photogrammetry, a popular algorithm used for 3D reconstruction from a group of overlapping images of an object in different angles.

1.1 Motivation

From the dissertation proposal, the idea of a 3D copy and paste application came from "AR Cut&Paste" [2], a prototype from 2020 by Cyril Diagne that allows cutting elements

from your surroundings and pasting them in an image editing software. AR Cut&Paste was later developed into a product, ClipDrop [3].

The beginning goal of the project was to interpolate that in three dimensions, so instead of a image editing software, the idea was to “copy” a real world object into a 3D model and “paste” it into a 3D modeling software. Therefore creating an illusion that the object is being copied and pasted in the real world.

Another motivation was using the LiDAR sensors on new Apple devices to generate those 3D models, but it became obvious quickly that it was not going to work for any small or medium sized objects.

1.2 Salient object detection for single object 3D reconstruction

RGB Salient object detection is a task based on a visual attention mechanism, in which algorithms aim to explore objects or regions more attentive than the surrounding areas on the scene or RGB images.

As mentioned, when doing Photogrammetry the whole scene is taken into account which will result in unwanted background data on the output 3D object. An innovative solution that automatically removes it is presented. It does background removal in 3D objects by doing SOD on the input images used later on in Photogrammetry. This dissertation evaluates if this is a plausible solution.

1.3 Structure of the dissertation

The dissertation is structured the following way.

Chapter 1 Introduction: This chapter serves as introduction to the problems in the field of 3D reconstruction in mobile devices that the dissertation has the objective to solve.

Chapter 2 State of the Art: Looks at the current research in the field of 3D scanning. It starts by laying out the field of 3D scanning and its methods. LiDAR and Photogrammetry, the two methods tried, are described in detail. It dives deep into the pros and cons of both technologies, as well as a study of the best solution available for the use cases of dissertation. It also explains the State of the Art of Salient Object Detection. This chapter also looks at current 3D scanning applications for mobile phones. It critiques current issues with them, and lays out possible solutions that are then implemented in the following chapters.

Chapter 3 Design: This chapter explains the methodology used behind the dissertation. Comprised of three different parts. The design follows a client-server architecture, combined with an plugin for a 3D modeling software. The chapter also shows how the data flows between all modules. Different approaches were tried throughout the development of the dissertation which are also explained in this chapter.

Chapter 4 Implementation: It does an in-depth explanation of how all parts of the application work. It starts with the main research and innovative part of the dissertation, the Photogrammetry server. Inside this section, it is explained how background removal is preformed by utilizing U²net. Then, it explains how the images are used to perform Photogrammetry, and most importantly, how Photogrammetry can perform its function without background data on its input images.

Chapter 5 Evaluation: The goals explained in the first chapter are evaluated in this one. It describes how the experiments were chosen in relation to the goals they tried to accomplish. It also explains what the experiments where not trying to accomplish, such as evaluating 3D reconstruction methods in general, as these have already been extensively researched. Instead, experiments were focused on evaluating performance of background removed images as an input to Photogrammetry. The second section describes the results obtained from the experiments, which are evaluated and critiqued. Performance of the implementation is explained, as well as the limitations that come with this solution.

Chapter 6 Conclusion and Future Work: It summarises the main contributions of this dissertation. It also explains future use cases of the this solution and proposes ways of improving the methodology. Other Salient Object Detection methods could be used in the pipeline, those methods are mentioned, future work would include studying them and their performance. Other improvements could be made in the mobile application which are also explained. Finally, it also describes a way to make this dissertation into a product.

Chapter 2

State of the Art

In this chapter, the State of the Art on 3D scanning technologies is described. Specifically, the technologies used for this project.

There is a focus on Photogrammetry works and how the theory of adding background removal fits into it.

2.1 3D Scanning technologies

3D scanning is the process of reconstruction of a real-world object or environment to collect data on its shape and its appearance if wanted, that data is then used to construct digital 3D models. 3D scanning is done through a 3D scanner.

The purpose of a 3D scanner is usually to create a 3D model, which consists either of a point cloud of geometric samples on the surface of the subjects or a polygonal mesh representation of a shape, known as polygonal 3D models. These points or models are then used to recreate the shape of the subject.

Data collected from 3D scanning is used extensively by the entertainment industry in films and video games, which sparked the field of virtual reality, augmented reality and mixed reality.

3D scanning has been made with a variety of different technologies. Most of those technologies come with limitations, mainly about the kind of objects that can be digitalised, as well as the devices needed to do so.

These technologies work with sensors, including optical, acoustic, laser scanning, radar, thermal, and seismic. One way of classifying those technologies is contact and non-contact [4].

2.1.1 Contact

Contact 3D scanning is also known as digitizing, it probes the subject through physical touch while the object is firmly held in place. A CMM (coordinate measuring machine) is an example of that. Although it is used mostly in manufacturing and can be very precise, it does not have a mass consumer use as it has its disadvantages. Those include that it might modify or damage the subject, and physically moving the arm that the probe is mounted on can be very slow compared to an optical system for example.

2.1.2 Non-contact

Non-contact scanning can be further divided into active and passive.

Active scanning is performed by emitting some kind of light or radiation and detecting its reflection or radiation passing through the subject, which can be an object or environment. For example, light, ultrasound or x-ray.

Examples of passive non-contact scanning are hand-held laser scanners. This method consists of a triangulation mechanism to create a 3D image. A laser is projected onto the subject from a hand-held device and a sensor measures the distance to the surface. In order for this to work, the position of the scanner must be determined so that the data collected can be used in relation to an internal coordinate system. Using reference features on the surface being scanned or using an external tracking method such as integrated cameras, we can determine the position of the scanner.

Active non-contact scanning relies on semi-automatic generation of 3D reconstruction from sensor data. One of the most popular methods is through stereo 2D images, this process is also known as Photogrammetry.



Figure 2.1: Example of passive non-contact scanning. Source: Sense 2 by 3D systems

As seen in Figure 2.1, external devices are often needed. That limits massively the user base that is going to do any sort of 3D scanning. One of the main perks of Augmented Reality is its portability, and while 3D content is become widely available, the creation of that content is left for a few with the monetary or technical ability to do it.

That is the reason Photogrammetry in a mobile device was chosen as a cheaper alternative. But as will be explained in the next section, there are still a few issues with it.

2.2 Mobile Applications

The field of 3D scanning in mobile applications is small, within a search on the Apple App Store of "3D scan", outside the top 10 results, the search shows apps that do other functionalities. When looking at the most popular applications, a main observation can be made.

There is a stark contrast between promotional images and the results you can get from using the app. The marketing strategy some use is having a free version for newer apple devices, where it just uses the LiDAR sensor. If you want any accuracy on the 3D reconstruction, you have to pay and wait a tiresome amount of time for a 3D model to be generated. An example of this is "3D Scanner App" [5], the most popular app on the Apple App store. Behind the surface, the 3D model is also generated using Photogrammetry. But whereas current mobile apps do that process manually, the solution developed in this dissertation is completely automated end-to-end so the amount of time the user would have to wait for a 3D model to be generated is way less than current solutions.

As later found in the implementation, LiDAR on new apple devices such as the 2020 iPad Pro and iPhone 12 Pro is not accurate enough for any small to medium devices. The reason being that the intention behind adding LiDAR in Apple devices is to create more accurate AR experiences and better image quality when taking pictures, not to 3D scan.

There is some exceptions in the State of the Art though. For example Qlone [6] is an interesting application that generates high quality 3D objects.

The solution they use is similar to the one used in this dissertation. In the fact that they also do background removal before photogrammetry. But as seen in Figure 2.2, the drawback is that it requires a special mat so the size of object you can scan may be limited by the paper size of your printer. The solution presented removes this issue.

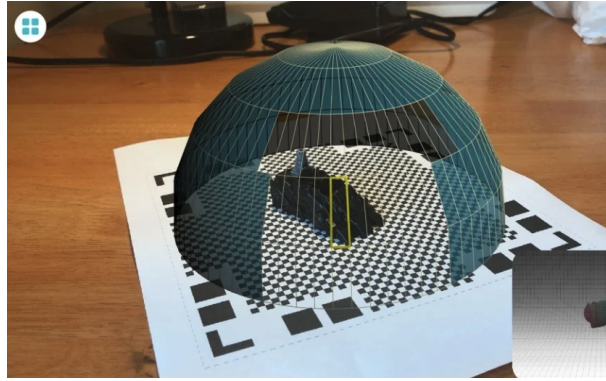


Figure 2.2: Using Qlone 3D scanning mobile app.

2.3 3D Modeling

3D modeling describes software tools for producing and modifying 3D representations of the world. 3D modeling artists use these software tools to manipulate vertices in a virtual space that form the 3D model mesh [7].

These 3D objects can be generated automatically or created manually by deforming the mesh, or otherwise manipulating vertices.

3D modeling is often used with combination of 3D scanners in post-processing. It is most likely that the objects generated from mobile 3D scanning are then sent to a 3D modeling software for a clean up process or to prepare it for the later use in the many applications of 3D content. This transition process is often tedious and done manually.

That is the reason why a 3D modeling software is used to showcase the capabilities of the "paste" functionality of this project. By automating sending and retrieving the 3D object, the time that this process would take is reduced significantly.

The 3D modeling software of choice is Blender, one of the most popular open source tools there is currently.

2.4 Lidar

LiDAR stands for Light Detection and Ranging or Laser Imaging Detection and Ranging, it can also be seen as LIDAR or Lidar. LiDAR is a sensor that measures distance by projecting the light of a pulsating laser into an object or surface and measuring the time it takes for the light to come back. LiDAR technology is used in geology, seismology, self-driving vehicles, atmospheric physics and more.

The concept of LiDAR come from RADAR, where instead of using radio waves, LiDAR uses light as a measuring system. LiDAR was originally a combination of light and RADAR, which is why the capitalization is done this way.

One of the main applications of LiDAR is 3D reconstruction. Apple recently added this technology into its flagship devices for both tablets and phones. Therefore, LiDAR was first thought as a possible implementation for 3D Copy&Paste.

Once tried out in the first version of the solution, it became obvious quickly that it was not going to be powerful enough for any medium to small devices. That is why Photogrammetry was used in the final solution.

For Apple, the reasoning behind adding LiDAR was mainly to improve their camera and adding support to new Augmented Reality (AR) capabilities. On their press release [8], Apple focused on supporting pro photo and video apps.

Although they also mention its depth-sensing capabilities, it was mostly centered around their measuring app and adding more accuracy on AR solutions, such as moving 3D objects behind real-world objects.



Figure 2.3: Obtained point cloud of front-facing face detection sensor on an iPad Pro 2020 on the left vs back-facing LiDAR sensor on the same device.

Consequently, Apple does not offer an easy way to obtain a 3D mesh from the LiDAR sensor. As shown in the implementation, it is possible to obtain it via a not straightforward way. But the data it obtains, as shown in Figure 2.3, is not nearly enough for 3D scanning. That study and the picture comparison was done by iFixIt [9].

2.5 Photogrammetry

Photogrammetry is the science of making measurements from photographs, so if photography is the projection of a 3D scene onto a 2D plane, the goal of Photogrammetry is to reverse that process. It does so by chaining computer vision-based pipelines, the main one being “Structure-from-motion” (SfM). SfM has the objective to understand the geometric relationship between the observations provided by the input images, most im-

portantly, the features that can be extracted from those. Figure 2.4 shows an example of how pictures can be used to make a 3D object.

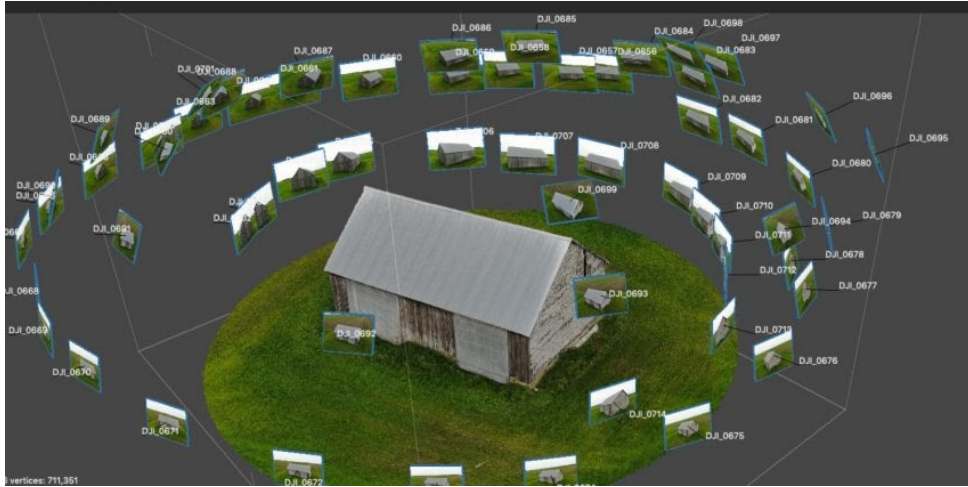


Figure 2.4: Example of predicted picture positioning over the 3D model generated by Photogrammetry.

Whereas other types of 3D scanning require expensive hardware, Photogrammetry can take input from a mobile device camera to generate the 3D model. A mix of Photogrammetry and other less-common sensors from phone devices is what is usually used as a solution in mobile applications.

There are a handful of state-of-the-art Photogrammetry solutions. Such as Mesh-Lab [10], 3DF Zephyr [11], Autodesk ReCap [12], or the recent Apple’s own Object Capture [13]. The software used in this implementation is Meshroom [14] [15].

One of the main reasons why Meshroom was chosen is the fact that it is a proven solution that produces high quality 3D models from phone cameras of different calibre. Whereas other solutions are expensive and usually require a per-month subscription, Meshroom is completely free. Most importantly, Meshroom is an open source software, meaning all the process is explained and can be trusted, while also providing any customization needed for this implementation.

Meshroom’s Photogrammetry process chains two computer vision-based pipelines: “Structure-from-Motion” [16] and “Multi View Stereo” [17].

The first step is feature extraction, which has the objective of extracting distinctive groups of pixels invariant to changing camera viewpoints.

The feature detection algorithm is SIFT [18][19].

The next step is matching images that are looking at the same area of the image. It is done through simplifying the image into a compact image descriptor, used to compute the distance between images. Image descriptors are done through a vocabulary tree approach

[20].

After that, the features extracted are matched between the image pairs found by the last step.

Enough data is extracted for structure for motion. SfM computes the fundamental matrix between two image pairs and considers that the first one is the origin of the coordinate system. Now that we know the pose of the two first cameras, we can triangulate the corresponding 2D features into 3D points [21]. This process is known as Bundle Adjustment: on a pair of images, the program searches for common points and, determining their offset or error from image to image, calculates how far the image is taken and what rotation the camera made. A schematic representation of this process can be seen at Figure 2.5.

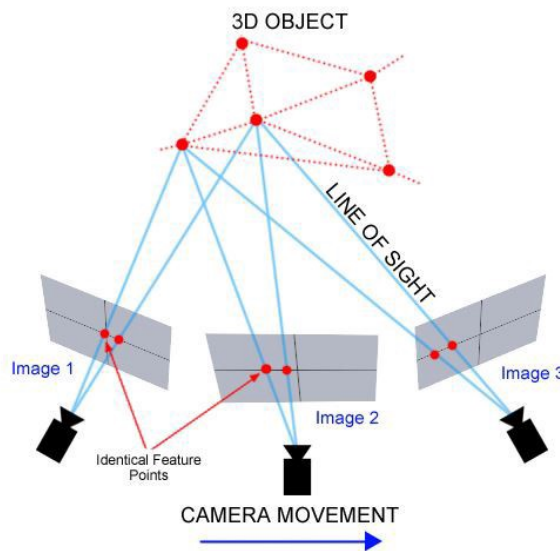


Figure 2.5: Representation of how SfM finds 3D points from 3D images.
Source: <https://thehaskinssociety.wildapricot.org/photogrammetry>

Then, Meshroom generates a depth map estimation by Semi-Global Matching [22][23].

Finally, meshing is done to create a dense geometric surface representation of the scene, done through 3D Delaunay tetrahedralization [24] and a Graph Cut Max-Flow [25]. If wanted, Meshroom provides texturing by generating automatic UV maps [26].

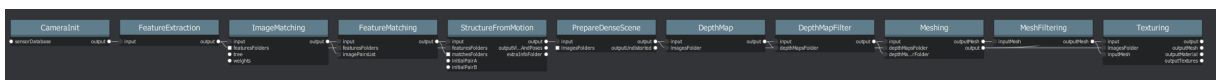


Figure 2.6: Screenshot showing the node-based system of Meshroom.

Meshroom works with a node or block based development system. As seen in Figure 2.6, the node-based system serves as a pipeline. Behind every node, there are the main algorithms explained before, as well as other several additional auxiliary algorithm.

Although Meshroom is the best open-source based Photogrammetry solution, it has its issues. Compared to other professional solutions, their solution can have a lower performance. It also tends to take more time than other Photogrammetry solutions, that is due to the fairly old algorithms that have not been improved upon or those improvements have not been yet implemented in Meshroom. That is another reason why automation is one of the main goals in order to reduce the total time.

Meshroom being open source means their resources are limited. At the time that this dissertation is written, it only has 36 contributors. Consequently, it will have trouble being capable of handling different kinds of textures. For example, textures of road tiles, parquet or carpets are known to be superimposed on themselves. It also has trouble with glossy or reflective surfaces.

Some of those problems are mitigated by the fact that only single objects are fed into Meshroom. By removing most of the background, Meshroom has much less data to handle. Therefore, the speed of generating 3D models of single objects can be improved up to 50%.

2.6 Salient Object Detection

Salient object detection (SOD) is a speciality of object segmentation. Object segmentation or image segmentation describes categorizing each pixel of an image to a particular object or class. Humans are very good at distinguishing different parts of a scene and drawing boundaries. But for humans, that process is very time consuming, that is why historically this has been one of the major challenges that computer vision tries to solve.

There is a difference between image segmentation and object detection. Object detection will build a bounding box to each class in an image, the task here is just detecting the different objects within a scene. Instead, image segmentation creates masks for each of the objects. Although SOD has object detection in its name, salient describes the most important object or part of a scene, that means single image segmentation can be meant as Salient Object Detection with posterior segmentation, which is the task in hand for this dissertation.

Image segmentation can be further divided into semantic segmentation and instance segmentation. Semantic segmentation assigns a label to every pixel in the image. It treats multiple objects of the same class a single entity. That is in contrast of instance segmentation, which treats multiple objects of the same class a individual instances.

A visualization of this different types of identification in an image can be seen in Figure 2.7. This image is extracted from the COCO dataset [27], a historical dataset to perform image segmentation.

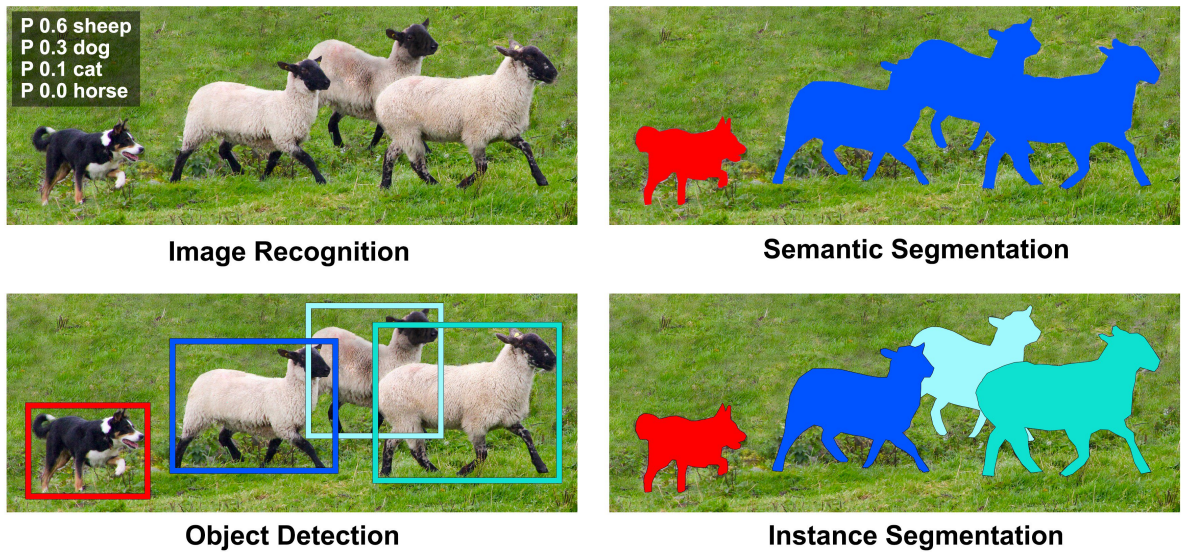


Figure 2.7: A sample annotated image from the COCO dataset, illustrating the difference between image recognition, semantic segmentation, object detection, and instance segmentation.

2.6.1 Image segmentation

Historically, many computer vision approaches based on algorithms were tried with not a lot of success. That is before the rise of Machine Learning and Neural Networks specifically, which revolutionized the field of computer vision, obtaining performance similar or in cases better than humans, at a fraction of the time.

Most of the state-of-the-art Neural Networks dedicated to this task are capped by how good the datasets are. COCO and other datasets, such as UCI image segmentation datasets, PASCAL VOC 2012, CityScapes dataset and KITTI semantic segmentation dataset, allowed the models to perform exponentially better.

Deep learning ability to find correlation in massive datasets has been proven to be the best current method of providing a solution to the most difficult tasks within image segmentation, such as being able to predict an object just by a small part of it. That ability is crucial as it allows a starting point for other, more general Artificial Intelligence problems such as self-driving cars.

One of the most important papers showcasing a state-of-the-art model is "Fully Convolutional Networks for Semantic Segmentation" [28], which started off the use of Convolutional Neural Networks for this task. That improved over classic Neural Networks like AlexNet, the VGG net, and GoogLeNet. Another ground-breaking research paper is "DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs" [29] which added more layers in the CNN model, also

known as Deep Convolutional Neural Networks.

This leads to "U-Net: Convolutional Networks for Biomedical Image Segmentation" [30] by , which breaks the path of deeper Neural Network in favour of a more focus on using data augmentation to more efficiently use the available dataset and its annotated samples. It does so by contracting path to capture context and a symmetric expanding path that enables precise localization. In 2015, the network outperformed the prior best method on the ISBI challenge [31] for segmentation of neuronal structures.

Even though the network was used in a specific use case, in the biometric field, that same concept of an end-to-end trained neural network with simple and fast architecture was what sparked U²net.

2.6.2 U²net

SOD has not had the same attention as image segmentation as a focus in research, as its use cases can be very different than general image segmentation. Although the field has made excellent advances, they are mostly focused on a more green-screen type use cases. Other than that, previous work focuses more on region accuracy, leaving boundary quality as an after-thought, as in use cases where that is needed, a green-screen is used.

Even though U²net takes its name from U-net, it is not from the same authors or even related to the future work of it. It just takes the same concepts when making a more general object detection architecture. U²net is based off of BASNet [32], a previous work done in 2019 from the same authors that showed state-of-the-art results for SOD.

BASNet proposes boundary-aware SOD using a predict-refine architecture. The architecture is composed of a densely supervised Encoder-Decoder network and a residual refinement module, which are respectively in charge of saliency prediction and saliency map refinement.

This focus on boundaries when segmenting objects was the main reason why it was chosen.

Future work of BASNet produced U²net in 2021. U²net used the same concept as U-net for a more general SOD implementation.

The architecture, which can be seen at Figure 2.8, stacks U-structure for Salient Object Detection. Although the concept is not new, before U²net, these methods stack U-Net-like structures sequentially to build cascaded models and can be summarized as "U \times n-Net", where n is the number of repeated U-Net modules. The issue is that the computation and memory costs get magnified by n . U²net proposes stacking U-structure for salient object detection. The exponential notation refers to nested U-structure rather than cascaded stacking. Theoretically, the exponent n can be set as an arbitrary positive

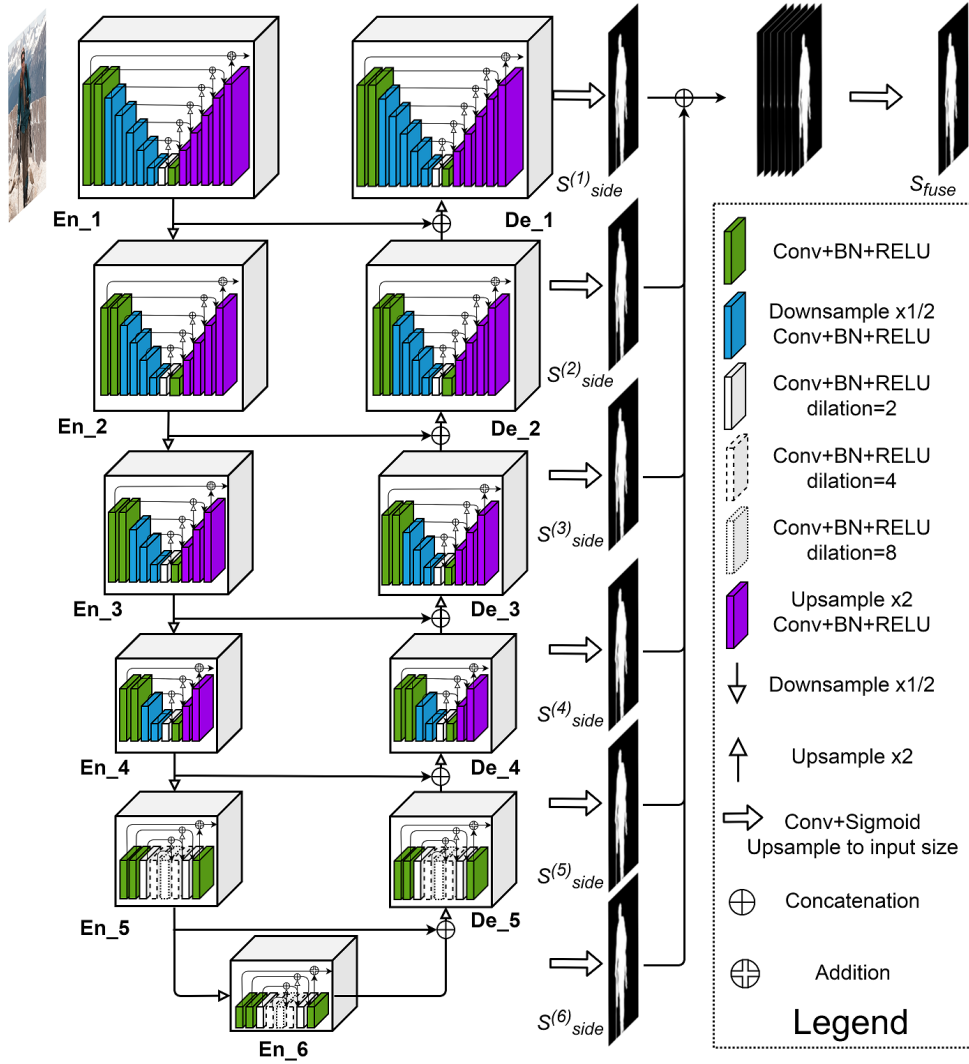


Figure 2.8: Illustration of U^2 net architecture.

integer to achieve single-level or multi-level nested U-structure. But architectures with too many nested levels will be too complicated to be implemented and employed in real applications.

This architecture enables training a deep network from scratch. And provides the results seen in Figure 2.9.

On the paper there are over 15 qualitative measurements comparing it to other state-of-the-art SOD solutions, performing around the top in all of them. More examples of the results that U^2 net provides can be seen in the evaluation section.

The pros of using U^2 net are mainly the boundary accuracy provided by the work laid out with BASNet, and its speed provided by the simplicity of a U-net type architecture. That low latency works best with this dissertation as Photogrammetry already takes a lot of time, and background removal can be done fast to save time using this model.

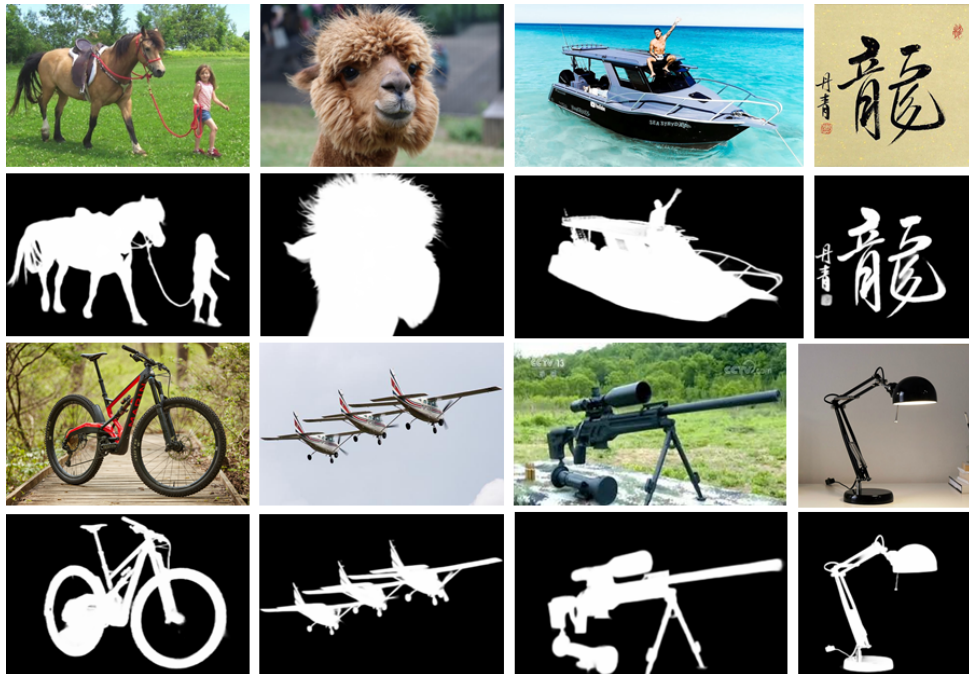


Figure 2.9: Results of U²net.

Chapter 3

Design

The design of the solution comprises three different parts. First, the iOS mobile application as the user interface, which also has the task of obtaining image data from the camera sensor of the device. If available on the system, it also obtains LiDAR sensor information. Then, a server runs a single object Photogrammetry solution to generate the 3D model. Finally, a Blender add on to import the 3D model, showcasing the automated "paste" process.

Figure 3.1 explains how the information flows through the three modules.

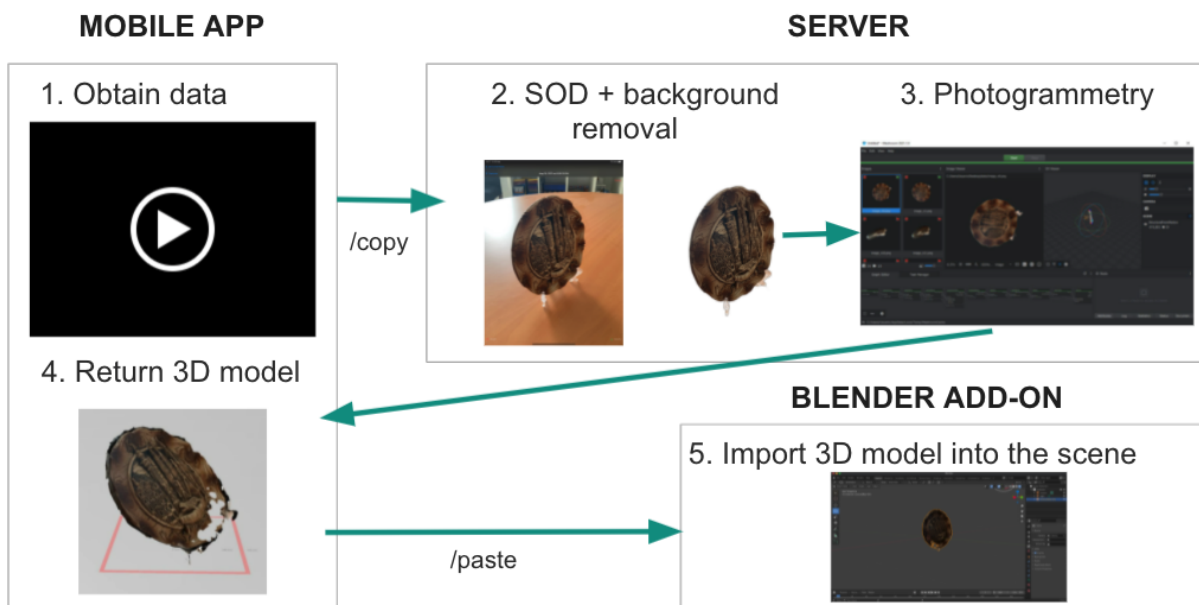


Figure 3.1: Flow of information throughout the 3D Copy&Paste process.

The first step is to obtain data from the phone device. For that, the user is asked to take pictures around the object. Those images are then stored locally in the device.

Then, the mobile app calls the `copy` endpoint on the server and sends the obtained

images. After that, the server first does salient object detection, and then background removal from the generated mask. After the images with the background removed are generated, those are then fed to Meshroom.

Once Meshroom has generated the 3D model, it is then sent back to the mobile phone on the same request.

Finally, once the user wants to send the generated 3D model to Blender, the user clicks the paste button and that object is automatically imported into Blender.

At the moment, the server runs in the same computer device as Blender. Therefore, the Blender add-on only needs to import the object at the location where the server has stored the 3D model. In future work, the server will be moved to a cloud solution, and the Blender add-on would fetch the 3D model from there.

It is important to know that this process could be done manually, but it would take hours, depending on how fast you are. The solution presented is completely automated end-to-end by using HTTP and Python scripts.

3.1 Alternative Approaches

Automated end-to-end 3D reconstruction is not a task that has been publicly taken before. That is due to the fact that 3D reconstruction requires many different moving parts that have to be connected together.

Consequently, the design chosen was not based in a single previous work. Instead, the implementation is a mixture of ways to connect different software. For example, although some Blender add-on are able to import a 3D model from a source, that source is not the end of a 3D scanning software.

Many distinct approaches could be taken to tackle this or any of the other tasks. For this one in concrete, the used method is a fairly simple way of obtaining the 3D model by monitoring a folder for it. An alternative approach, explained in the future work, could have been made with Websockets.

This is just an example of the many choices taken during this dissertation.

The first proof-of-concept used an Apple device for its new LiDAR scanner. As Augmented Reality becomes a more popular topic, other depth sensors have been natively added to mobile devices. For example, Samsung recently announced a sensor to compete with Apple, instead of LiDAR, that sensor is a Time of Flight (ToF) sensor on the Galaxy S22 known as ISOCELL Vizion 33D [33]. Although Photogrammetry has its advantages, in the future an on-device sensor will probably be powerful enough to handle real-time 3D scanning of smaller objects. When that time comes, it can be easily implemented to the 3DCopy&Paste pipeline.

The LiDAR sensor on Apple devices was also the reason iOS was chosen as the platform of the mobile application. That being said, the Photogrammetry solution could be used in any other platform.

With no prior experience with iOS development, building the iOS application was challenging. Apple developer documentation is also not very developer friendly. Two different User Interface frameworks are used, UIKit for the LiDAR implementation, and SwiftUI for Photogrammetry. That reason frameworks were switched mid-implementation was that SwiftUI provides much faster development, which was needed for the complexity of the Photogrammetry image capturing steps. It is also better incorporated with the programming language Swift.

A client-server architecture was chosen to communicate between the modules. Although other options are available and improvements can be made, client-server architecture provides a robust and proven solution. Python was chosen as the language for the server, that is because of its ease of use and wide range of libraries that provide all the necessary functionalities with a fraction of written lines of code compared to other languages. For the API, Flask was chosen against other web development Python frameworks like Django for its simplicity and customization, it is way quicker to build a REST API with Flask as no boilerplate code is needed.

The design of the code can be improved in many ways. During the development, the design was constantly changed to fit better implementations. It is a safe to say that more modifications on the design will be made in future work, mainly to improve speed. That being said, the design meets the end-to-end automation goals and is already a significant usability and speed improvement over current mobile 3D scanning solutions.

Chapter 4

Implementation

Last chapter laid out the parts and overall functionality of the project. In this chapter, every module is explained in detail. From the technologies, languages, frameworks and libraries used, to integrating third party software.

Automation was one of the main principles in mind when making the implementation. Because of the need to improve the time it takes to 3D scan from a mobile device and also require no technical knowledge from the user side.

Another principle was applying state-of-the-art solutions for Augmented Reality, Salient Object Detection, and Photogrammetry.

The technical architecture seen at Figure 3.1 was made with these principles in mind.

The code implementation folders are the same three modules, "server", "app", and "scripts", which is what is used for the Blender add-on. There is also a "scans" folder, which is the path that the Blender add-on looks at when importing the model. The server is further divided into the LiDAR and the Photogrammetry implementation.

4.1 Photogrammetry server

The Photogrammetry server implements the research side of the dissertation. One of the main goals of the dissertation was to make 3D models of just single objects. Often, when Photogrammetry is used to generate 3D reconstructions, the whole scene is taken into account. Therefore, the output has unwanted data from the background that has to be removed manually. This requires technical knowledge of 3D modeling software, which does not fit the other goal to make the solution usable regardless of previous technical knowledge.

Therefore, the background needs to be removed from the final 3D object automatically.

There are two different ways to tackle this. First, removing the background after

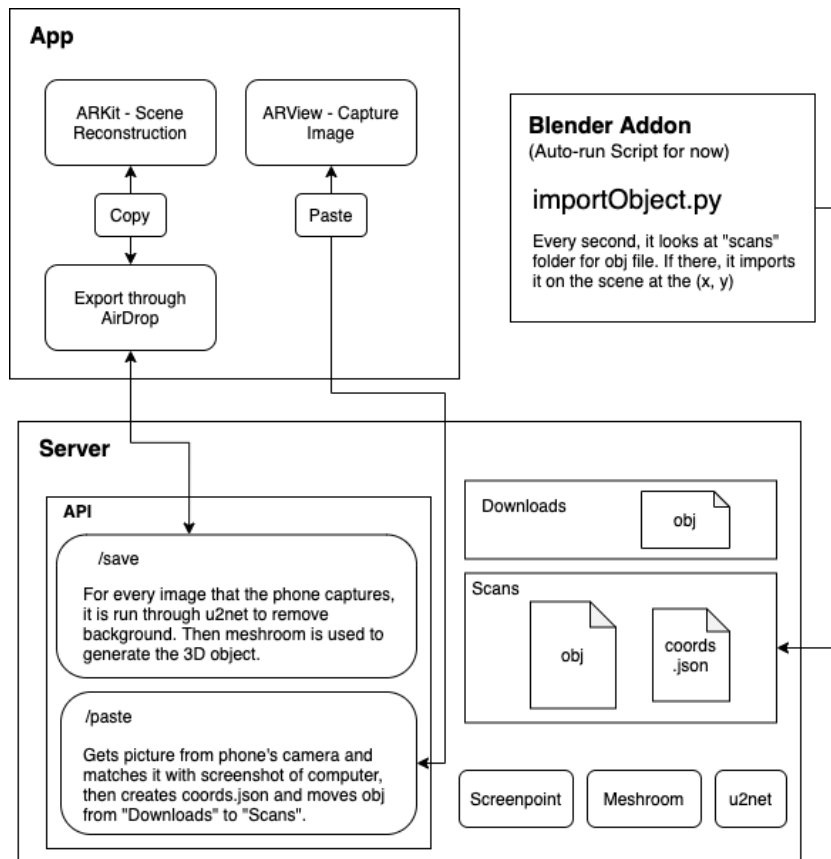


Figure 4.1: Technical structure of the project.

the 3D object is done. Although there have been solutions lately showcasing progress in 3D object segmentation [34][35], this solution was discarded as it requires very high computational power and it is not as developed compared to the next solution. Current solutions also mostly focused on segmentation needed for self-driving technology.

The other solution, and the one that is implemented, is doing background removal before 3D reconstruction with Photogrammetry on the input images. The field of object segmentation in 2D image is massive in both computer vision and machine learning [36]. The solution used for salient object detection is U²Net [37], which is seen in detail in the State of the Art section. Once the background and foreground is detected, it generates a mask that can then be superimposed with the original image to have its background removed.

The whole process is done on a server written in Python and using Flask to generate the API to communicate with the phone app.

4.1.1 Salient object detection

SOD Deep Learning solutions rely on an image input and a mask output with the foreground and background clearly defined, where the foreground is the salient object.

U²Net allows training from scratch and achieves comparable or better performance than those based on existing pre-trained backbones. This works best for this project as it only adds around 10% to the total time of the solution as seen in the results section.

The training of the model is done through DUTS-TR dataset [38] and tested on six public salient object detection datasets.

The server leverages off the Python and PyTorch implementation of the neural network that the authors of U²Net published [39].

U²Net Python implementation is used as a Python library, SOD masking is a simple function of that library which can be used very easily.

4.1.2 Background removal

Once a mask dividing foreground and background is generated by U²Net, also known as Binary Matte, simple masking is done to leave just the object from the input images. This is done through Python's Pillow library and the composite function [40] it provides to conduct masking. That is seen in the following Python code, where this process is done for each image and saved into a particular location:

```
1 from PIL import Image
2 import u2net
3 import numpy as np
4
5 for image in images:
6     filename = image.filename
7     image_data = image.read()
8
9     # Convert string data to PIL Image
10    img = Image.open(io.BytesIO(image_data))
11
12    # Ensure image size is under 1024
13    if img.size[0] > 1024 or img.size[1] > 1024:
14        img.thumbnail((1024, 1024))
15
16    # Process Image
17    mask = u2net.run(np.array(img)).convert("L")
18
19    # masking
20    empty = Image.new("RGBA", (img.size), 0)
```

```
21     img_bg_removed = naive_cutout(img, mask)
22     img_bg_removed.save()
```

Listing 4.1: Background removal code.

As will be seen in the evaluation section, the main issues are on the edge area. It is to be expected that the inaccuracy will show up in that area.

In order to obtain more accurate 3D objects, a bit of margin is added in the foreground matte.

4.1.3 Photogrammetry solution

Once all the images have had their background removed, they are fed to Meshroom. It is crucial that background removed images works well with Meshroom's Photogrammetry solution. The important step of the pipeline for this use case is the first step, which is feature extraction. As mentioned in the State of the Art chapter, the algorithm which Meshroom is based on for this part is SIFT, which means scale-invariant feature transform.

SIFT is an algorithm to extract features with similar characteristics in images, invariant to rotation, translation, and scale. SIFT was first developed by David Lowe in 2004, and one of the main challenges of that paper was to extract features of objects independent to the background. As SOD takes care of removing the background, this challenge is extracted from Photogrammetry.

Although this solution adds other challenges, input images of single objects without background works with this implementation of Photogrammetry.

On another note, in order to automate the process, Meshroom allows for command line based commands, so a bash script was made, which takes the images from a folder in the server and outputs it in the folder that Blender is looking at.

4.2 API

There are two HTTP endpoints that the server provides as its API.

On one side, the *copy* endpoint provides a POST method that allows for a group of images to be send. Those images are then used within the server to go through the 3D reconstruction pipeline, which includes background removal and 3D model generation with Photogrammetry. The HTTP function returns the generated 3D object as an OBJ file, with the material MTL file and texture file as a UV MAP.

In future work, instead of a normal HTTP function, HTTP long polling could be added so the user on the mobile application will not have to wait for the model to be generated and can meanwhile use the application.

The *paste* endpoint, which is also a POST method, has two functionalities. First, it obtains an image that the mobile application captures of what the camera is looking at, at the point when the paste button in the UI of the mobile application is pressed. Second, the code takes a screenshot of the computer screen. This two images are then compared with the same feature matching algorithm (SIFT) as Meshroom uses.

The reason this is done is to know where the user is looking at when "pasting" the 3D model to the computer by obtaining the centroid of two images 4.2. This concept is the same one that ClipDrop uses.

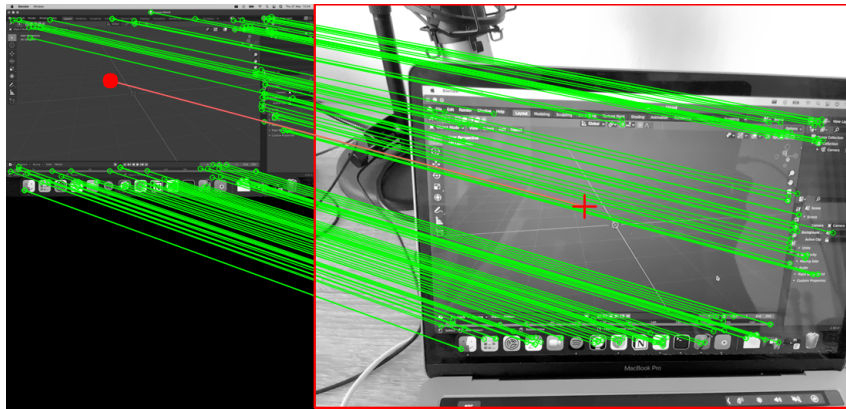


Figure 4.2: Feature matching to extract centroid of a screenshot of the computer and what the phone camera is looking at.

This is done with Python's library Screenpoint [41].

Lastly, the *paste* endpoint saves the coordinates extracted from feature matching as a JSON file, and moves them and the 3D model to a location that the Blender add-on is permanently looking at.

The API documentation can be seen at the following table 4.1.

URL	Method	Function	Parameters	Returns
/copy	POST	Remove background from images and make a 3D model make a 3D model out of them with Meshroom.	Array of images	3D object (OBJ, MTL)
/paste	POST	Obtain centroid coordinates and move 3D object in location that Blender knows	Image of the Camera view	Accepted or denied

Table 4.1: API documentation.

4.3 Mobile application

The iOS mobile application is written in Swift. The first proof of concept of the solution was made just with LiDAR sensor information. Whereas the final implementation used Photogrammetry.

Because of that, the landing page shows the option for both solutions as seen at Figure 4.3. The LiDAR option was built with UIKit and uses ARKit to extract LiDAR’s information into an OBJ file. It also uses ARKit’s sceneReconstruction [42] to show in real-time the information that LiDAR sees.

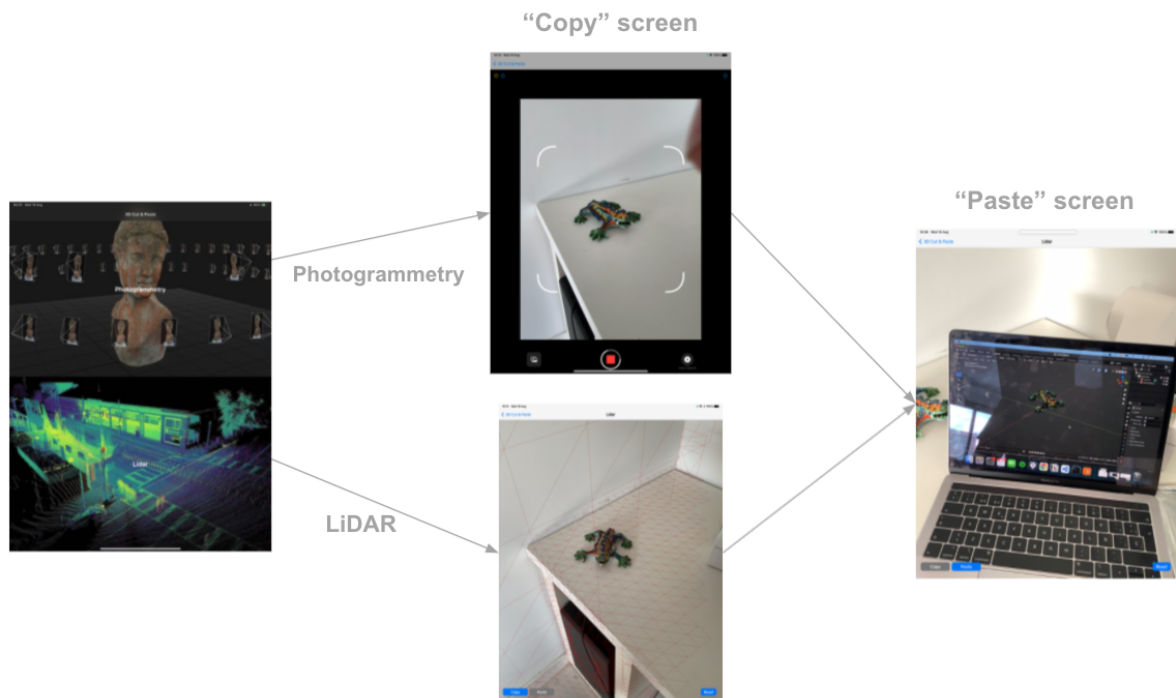


Figure 4.3: Flow of the main screens of the iOS application.

Getting the scene reconstruction into an OBJ file is not straightforward. Firstly, all *ARMeshAnchor* are fetched, which are anchors for physical objects that ARKit detects and recreates using a polygonal mesh. Then, each *ARMeshAnchor* is converted into a *MDLMesh* by converting each vertex of the geometry from the local space of their *ARMeshAnchor* to the world space. Each *MDLMesh* is added into a *MDLAsset*, which is the iOS container for 3D objects. Finally, the *MDLAsset* can be easily exported as an OBJ file using *UIActivityViewController*. The code for that can be seen at the *saveButtonPressed* function in *ViewController*. This method is similar to “iPadLiDARScanExport” by Stefan Pfeifer [43].

The photogrammetry option shows a complex embedded SwiftUI app. The bare-bones of it are similar to Apple’s “Taking Pictures for 3D Object Capture” app [44].

It allows for the user to take pictures of different angles of an object, autocapture them, which automatically shutters the capture button every second, and save those images into a local folder, which can later be browsed by date of the images abstraction.

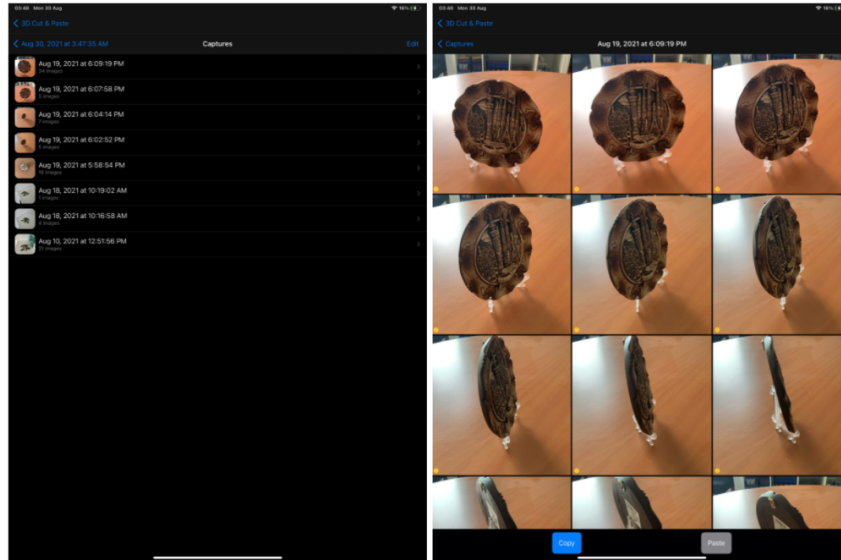


Figure 4.4: List of captures grouped by date on the left, gallery view of the images extracted in each group on the right.

The app has a table view of all the images extracted on different dates, therefore the images could be used anytime the user wanted to generate a newer 3D object, just by pressing the "copy" button. That table view can be viewed on the left side of Figure 4.4, on the right side, the images in a gallery view are displayed to the user.

The UI for both options also has two buttons, the copy and the paste button. Both call their respective API endpoints that were explained on the last section, with the data that the server requires.

The API requests are done with Swift library Alamofire. All the images are appended into a multipart formdata. The code for that is the following, where *CaptureFolderState* has a list of *Capture* objects, a data object that contains one image and its associated metadata:

```

1 func generate3DModel(captureFolderState: CaptureFolderState) {
2     let imageToUpload = captureFolderState.captures
3     let url = "\(urlServer)/copy"
4
5     AF.upload(multipartFormData: { multipartFormData in
6         let count = imageToUpload.count
7
8         for i in 0..

```

```

10         let image = try ImageLoader.loadImageSynchronously(url:
imageToUpload[i].imageUrl)
11             multipartFormData.append(image.pngData()!, withName: "
data[\(i)]", fileName: "image_n\(i)", mimeType: "image/png")
12         } catch {
13             print(error.localizedDescription)
14         }
15     }
16     print(multipartFormData)
17 }, to: url)
18 .responseData { response in
19     print(response)
20 }
21 }

```

Listing 4.2: API copy request from iOS app.

4.4 3D software Add-on

The blender add-on is a way to demonstrate as a proof-of-concept that you can "paste" the 3d model into basically anything that allows 3d content. Blender allows for python scripting that can be run automatically once you open a Blender project.

The Python script simply looks for a specific folder for the 3D model and the coordinates it has to import it in.

For now it looks at the folder every half a second in a subprocess of the computer so that it does not affect the speed of Blender. That can be seen in the following code. A further improvement would be creating a websocket so that the 3D model can be pushed once it is generated.

```

1 import bpy
2 import os.path
3
4 file_path_obj = './scans/texturedMesh.obj'
5 file_path_coords = './scans/coordinates.json'
6
7 pasted = False
8 def checkPaste():
9     global file_path
10    global pasted
11    if os.path.isfile(file_path_obj) and not(pasted):
12        imported_object = bpy.ops.import_scene.obj(filepath=
file_path_obj)

```

```
13     obj_object = bpy.context.selected_objects[0]
14     print('Imported name: ', obj_object.name)
```

Listing 4.3: API copy request from iOS app.

The object is imported at the 2D coordinates found by Screenpoint. Obviously Blender works in a 3D space. Therefore, the third axis missing means the pasting might look skewed. The solution was to programmatically show the z axis view, therefore moving the screen into 2D when "pasting".

Chapter 5

Evaluation

Evaluation was focused on determining the quality and usability of the solution. Mainly measuring the quality of 3D models using Photogrammetry from images with their background removed.

Experiments were also conducted in different scenarios, such as object sizes and different types of backgrounds.

User experience was also measured with the goal of measuring speed and usability compared to current 3D scanning solutions.

There was no evaluation of pros and cons of using LiDAR as a technology for 3D scanning, as that has been extensively researched already [45]. Instead, the experiments were conducted to find out if current on-device LiDAR sensors were accurate enough for 3D scanning.

The experiments were conducted in a iPad Pro 2020 for the mobile app, and in a Windows environment with CUDA-enabled GPU for the server.

5.1 Experiments

The most important evaluation was to compare 3D models with the background removal against a 3D model with the original images.

Aside from the apparent visual difference between the models, more objective experiments were conducted like comparing their scales.

As Photogrammetry by itself is not scale accurate. Ratio of width and height was used instead. In order to remove human error from the measuring, images were taken of 10 real world objects and their ratio measured through software.

Three of the results of this experiment can be seen at Figure 5.1.

Accuracy describes how close by absolute value the ratio of the 3D model is against


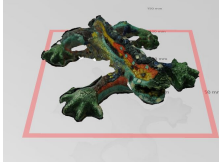




Object	W/H original	W/H 3D	Accuracy	3D model
	14.8 / 15.5	14.5 / 15.7	96%	
	14.2 / 14.3	12.3 / 12.3	99%	
	23.3 / 19.7	15.3 / 13.1	92%	

Table 5.1: Ratio of width divided by height compared between objects vs their respective generated 3D models

the original object. They show an average accuracy of 96%. Although the accuracy of that was high, the small sample size has to be taken into account, as well as the fact that object with similar width and height.

Further experiments comparing 3D models could be made, with tools such as Autodesk ReCap Photo [12].

5.2 Results

The results are promising, where the tested output 3D models do not have much noisy background data compared to how they would look with the original images.

Photogrammetry generates background data even in monotone backgrounds, as seen in Figure 5.1. Which is easily removed by this solution.

But this solution also has its limitations. Those are basically the same as the limitations with salient object detection in general. There might be a decrease in quality if the background removal is not done properly. Usually the problem lies in the edge area, were more background data is removed that needed, leading to incomplete 3D models.

In Figure 5.2 compares a professionally made 3D object, in this case an egyptian statue of the British Museum [46], against the output 3D model of our solution. Although the quality is significantly reduced, it still does an excellent job of extracting just the object



Figure 5.1: Compared 3D model with and without its background removed

data without the need of human work which might have taken hours. Instead, within 20 minutes, a 3D model is generated out 127 images without any technological knowledge needed. Moreover, our 3D model is 9MB compared to 50MB of the 3D model made by the British Museum.

In general, the solution struggles when not enough or good data is fed, which is a known issue of Photogrammetry.

The results of the width and height ratio experiments can be boiled down to differences between the edges of the generated 3D object. Other than that, this solution provides an accurate representation scale wise of the real-world object.

For the user experience, it is a huge improvement compared to current cumbersome solutions. Current solutions for 3D scanning from RGB images taken from a mobile phone take hours. In the tested environment, the solution presented takes five minutes to generate a 3D model from thirty images.

On average, the background removal process takes 0.5 seconds per image, while Photogrammetry takes around 25 seconds per image. Photogrammetry actually takes way shorter time, atleast 30% and up to 50% on the tested environment, due to the fact that it does not have to run all its algorithms on the full image, just on the object itself.

If LiDAR is used, data is taken in almost real-time. For the "paste" process, it takes less than a second.

5.2.1 Lidar vs photogrammetry

As shown in Figure 5.3, the mesh generated from the iPad Pro 2020 LiDAR sensor is not accurate enough even for large objects.

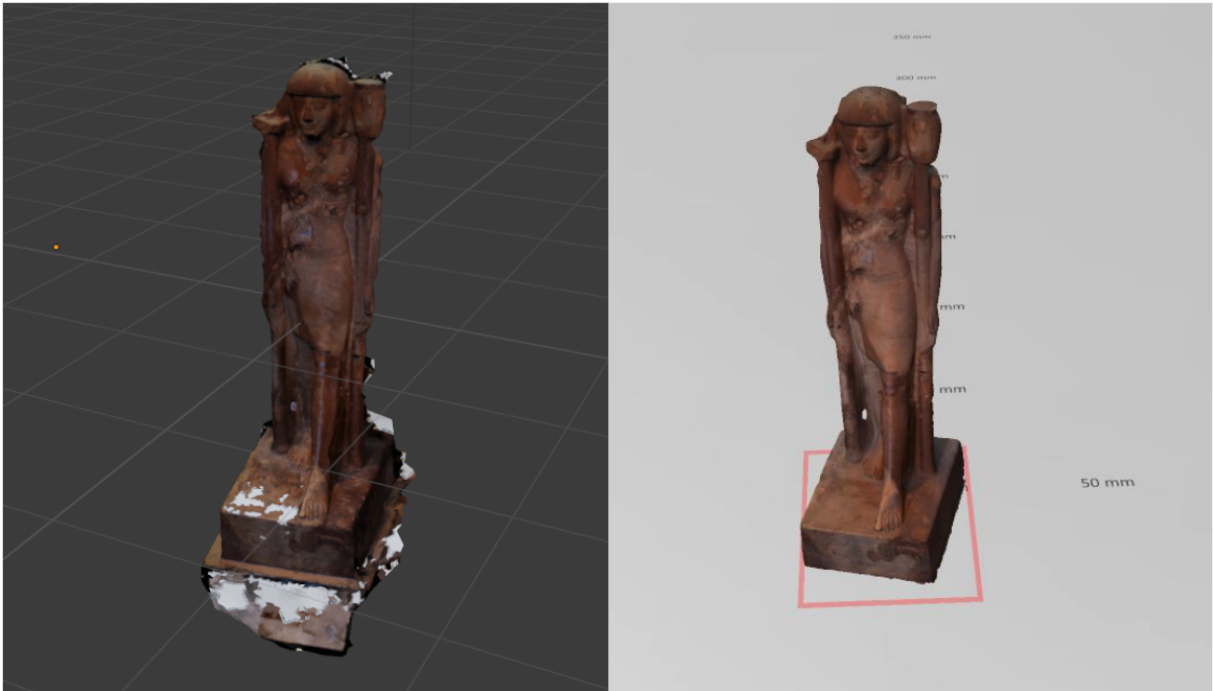


Figure 5.2: 3D model from our solution (left) compared to professionally done (right).

Although LiDAR is widely used for 3D scanning, in mobile devices it still needs higher quality sensors to be useful. Meanwhile, the presented solution generates quality 3D models from any modern phone camera.

A future improvement would be obtaining depth data from LiDAR and combining it with the RGB images, creating RGBD images which work better in Meshroom. Currently, Apple does not provide a way of obtaining that information easily for iPad devices [47].

5.2.2 Background removal

Background removal relies on the generated mask by U^2 net. U^2 net results are impressive, working in a wide range of settings and objects. As mentioned, some margin was added to each mask to improve the edge area issue.

As SOD is done in each image separately without context of the others, getting a model in all the angles can be tricky. For example, as seen in Figure 5.4, some foreground data might not be obtained in all images, therefore it can lead to incomplete 3D models. A further improvement could be made to add more context between images so that the same object is extracted in all the input data.

This is a perfect example showcasing the advantages of this solution, mainly removing background data, against the disadvantages, less accuracy compared with the 3D object

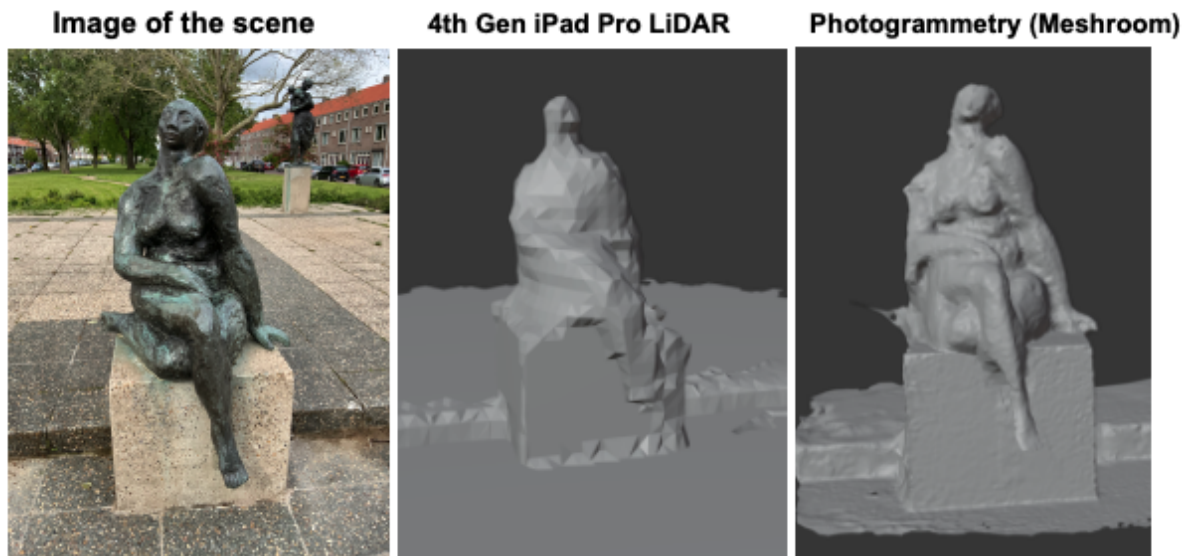


Figure 5.3: Compared 3D model generated with a mobile device LiDAR sensor vs RGB images with Photogrammetry.

generated from the same dataset without the background removed. So even though the produced 3D object is of slightly lower quality, the fact that there is no need for any major post-processing is a big advantage for non-technical users. The example dataset is 245 images showing a flower pot [48], both 3D models are created with Meshroom.

It is important to take enough pictures, the recommended amount is at least 20, and center the salient object in a similar position when taking pictures. Further improvement could be done in re-centering the images.

Also, for this solution, a better SOD algorithm could have been RGB-D Salient Object Detection via 3D Convolutional Neural Networks [49], which does salient object detection in a neural network where depth data is taken into account.

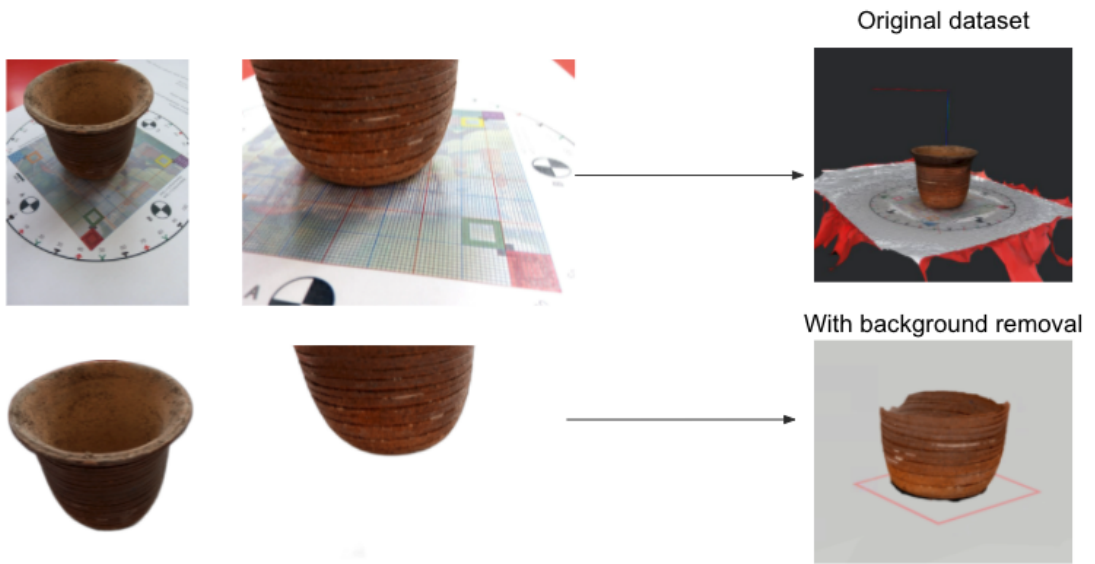


Figure 5.4: Limitations of SOD in combination of Photogrammetry for 3D scanning.

Chapter 6

Conclusions & Future Work

A 3D scanning solution of salient objects for mobile devices was successfully made with a unique pasting feature that sends the scan to a 3D modeling software in short time. The solution is automated end-to-end in order to improve current cumbersome user experience when 3D scanning through a mobile phone application.

There is enough proof to suggest that adding background removal to the Photogrammetry pipeline for salient object 3D reconstruction can be done.

That being said, in Photogrammetry the quality of the generated model will depend on the quality of the images, obviously a cellphone camera is not as good as a professional camera for now. Therefore generated 3D models from phone camera images might not be ready for a professional setting where a perfect replica of the real world is wanted.

That being said, this application would be perfect for hobbyist or augmented reality programmers that do not necessarily have time or money to make professional 3D models. As it removes a level of abstraction by automating processes in the 3D reconstruction pipeline, which is a huge improvement with current cumbersome methods of generating 3D models from a mobile device.

6.1 Future Work

The main improvements have to be done in the user interface, mainly on making sure the user understands each part of the process.

The unique pasting system could also be incorporated into any other software that uses or shows 3D content.

More experiments are required in order to obtain the best possible background removal for this solution. For example, 2D images could be overlaid with the 3D object and remove the background that are on the background part of the 2D mask.

In order to build a product out of this dissertation there is one main feature left to do. The server could be moved into a cloud environment. As a result, the user would not have to run the server locally. With just the phone app and the Blender add-on, the whole application would be usable by anyone. The Blender add-on would have to be modified accordingly so that the 3D object is fetched from the server instead of imported locally.

Also, the HTTP protocol currently does not support OBJ files or any other kind of 3D object type, as seen in the list of MIME types [50]. Future work would include supporting the standardization of 3D content types in the modern web.

Bibliography

- [1] Blender (2.29). (1994). free and open-source 3d computer graphics software toolset. blender foundation. <https://www.blender.org/>.
- [2] Cyril Diagne. Ar cutpaste, <https://github.com/cyrildiagne/ar-cutpaste>, 2020.
- [3] Clipdrop by init ml, <https://clipdrop.co/>, 2020.
- [4] Brian Curless. From range scans to 3d models. *ACM SIGGRAPH Computer Graphics*, 33 (4):38–41, 2000.
- [5] 3d scanner app by laan labs, <https://apps.apple.com/es/app/3d-scanner-app/id1419913995>, 2020.
- [6] Qlone, the all in one tool for 3d scanning, <https://www.qlone.pro/>, 2017.
- [7] Josh Petty. What is 3d modeling what’s it used for?, concept art empire, <https://conceptartempire.com/what-is-3d-modeling/>, 01 2019.
- [8] Apple. Apple unveils new iPad Pro with breakthrough LiDAR Scanner and brings trackpad support to iPadOS, 08 2021.
- [9] Kevin Purdy. How LiDAR Works, and Why It’s in the iPhone 12 Pro, 08 2021.
- [10] Meshlab, <https://www.meshlab.net/>, 2005.
- [11] 3Dflow SRL. 3df zephyr - photogrammetry software - 3d models from photos, <https://www.3dflow.net/3df-zephyr-photogrammetry-software/>, 05 2021.
- [12] Autodesk recap, <https://www.autodesk.es/products/recap/overview>, 04 2021.
- [13] Object capture - augmented reality, <https://developer.apple.com/augmented-reality/object-capture/>, 2021.
- [14] Pierre Moulon, Pascal Monasse, and Renaud Marlet. Adaptive structure from motion with a contrario model estimation. In *Proceedings of the Asian Computer Vision Conference (ACCV 2012)*, pages 257–270. Springer Berlin Heidelberg, 2012.

- [15] Michal Jancosek and Tomas Pajdla. Multi-view reconstruction preserving weakly-supported surfaces. In *CVPR 2011*. IEEE, jun 2011.
- [16] Johannes L Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4104–4113, 2016.
- [17] Michael Goesele, Brian Curless, and Steven M Seitz. Multi-view stereo revisited. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 2402–2409. IEEE, 2006.
- [18] G Lowe. Sift-the scale invariant feature transform. *Int. J*, 2(91-110):2, 2004.
- [19] Ives Rey Otero and Mauricio Delbracio. Anatomy of the SIFT Method. *Image Processing On Line*, 4:370–396, 2014. <https://doi.org/10.5201/ipol.2014.82>.
- [20] David Nister and Henrik Stewenius. Scalable recognition with a vocabulary tree. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 2161–2168. Ieee, 2006.
- [21] Jiaxiang Wu Hainan Cui Hanqing Lu Jian Cheng, Cong Leng. Fast and accurate image matching with cascade hashing for 3d reconstruction. *CVPR*, 2014.
- [22] Heiko Hirschmuller. Accurate and efficient stereo processing by semi-global matching and mutual information. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 807–814. IEEE, 2005.
- [23] Heiko Hirschmuller. Stereo processing by semiglobal matching and mutual information. *IEEE Transactions on pattern analysis and machine intelligence*, 30(2):328–341, 2007.
- [24] Pavel Maur. Delaunay triangulation in 3d. *Technical Report, Departmen. of Computer Science and Engineering*, 2002.
- [25] Yuri Boykov and Vladimir Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE transactions on pattern analysis and machine intelligence*, 26(9):1124–1137, 2004.
- [26] Adam Baumberg. Blending images for texturing 3d models. In *Bmvc*, volume 3, page 5. Citeseer, 2002.

- [27] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014.
- [28] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. *CoRR*, abs/1411.4038, 2014.
- [29] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *CoRR*, abs/1606.00915, 2016.
- [30] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597, 2015.
- [31] About the 2D EM segmentation challenge — ISBI Challenge: Segmentation of neuronal structures in EM stacks, 2012.
- [32] Xuebin Qin, Zichen Zhang, Chenyang Huang, Chao Gao, Masood Dehghan, and Martin Jagersand. Basnet: Boundary-aware salient object detection. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.
- [33] Isocell vizion 33d — tof sensor — samsung semiconductor, <https://www.samsung.com/semiconductor/minisite/isocell/vizion-sensor/isocell-vizion-33d/>, 2020.
- [34] Xinge Zhu, Hui Zhou, Tai Wang, Fangzhou Hong, Yuexin Ma, Wei Li, Hongsheng Li, and Dahua Lin. Cylindrical and asymmetrical 3d convolution networks for lidar segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9939–9948, 2021.
- [35] Ran Cheng, Ryan Razani, Ehsan Taghavi, Enxu Li, and Bingbing Liu. 2-s3net: Attentive feature fusion with adaptive feature selection for sparse semantic segmentation network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12547–12556, 2021.
- [36] Robert M Haralick and Linda G Shapiro. Image segmentation techniques. *Computer vision, graphics, and image processing*, 29(1):100–132, 1985.
- [37] Xuebin Qin, Zichen Zhang, Chenyang Huang, Masood Dehghan, Osmar R Zaiane, and Martin Jagersand. U2-net: Going deeper with nested u-structure for salient object detection. *Pattern Recognition*, 106:107404, 2020.

- [38] Lijun Wang, Huchuan Lu, Yifan Wang, Mengyang Feng, Dong Wang, Baocai Yin, and Xiang Ruan. Learning to detect salient objects with image-level supervision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 136–145, 2017.
- [39] Xuebin Qin. GitHub - xuebin Qin/U-2-Net: The code for our newly accepted paper in Pattern Recognition 2020: "U² - Net : Going Deeper with Nested U - Structure for Salient Object Detection.", 2020.
- [40] Python pillow, <https://python-pillow.org/>, 1995.
- [41] Cyrildiane Diagne. GitHub - cyrildiane/screenpoint: Project an image centroid to another image using OpenCV, 2020.
- [42] scenereconstruction - apple developer documentation, <https://developer.apple.com/documentation/arkit/arworldtrackingconfiguration/3521376-scenereconstruction>, 2020.
- [43] Zeitraumdev Pfeifer. GitHub - zeitraumdev/iPadLIDARScanExport: Export an OBJ file of ARKit 3.5 iPad Pro LIDAR scans, 2021.
- [44] Taking pictures for 3d object capture - apple developer documentation, 2020.
- [45] Maximilian Vogt, Adrian Rips, and Claus Emmelmann. Comparison of ipad pro's lidar and truedepth capabilities with an industrial 3d scanning solution. *Technologies*, 9(2):25, 2021.
- [46] BritishMuseumDH Pett. GitHub - BritishMuseumDH/egyptianStanding: Model files, 2016.
- [47] Depth for Object Capture on iPad Pro 2020, 06 2021.
- [48] Natowi Natowi. GitHub - natowi/dataset_lowerpot : *Dataset for photogrammetry*, 2018.
- [49] Qian Chen, Ze Liu, Yi Zhang, Keren Fu, Qijun Zhao, and Hongwei Du. Rgb-d salient object detection via 3d convolutional neural networks. *arXiv preprint arXiv:2101.10241*, 2021.
- [50] Common MIME types - HTTP — MDN, 08 2021.