DO 2D PARA 3D DESENHANDO

SERGIO NUNES DA SILVA JÚNIOR

DO 2D PARA 3D DESENHANDO

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação do Instituto de Ciências Exatas da Universidade Federal de Minas Gerais como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação.

Orientador: Erickson Rangel do Nascimento

Belo Horizonte

Fevereiro de 2017

SERGIO NUNES DA SILVA JÚNIOR

FROM 2D TO 3D BY DRAWING

Dissertation presented to the Graduate Program in Computer Science of the Universidade Federal de Minas Gerais - Departamento de Ciência da Computação in partial fulfillment of the requirements for the degree of Master in Computer Science.

Advisor: Erickson Rangel do Nascimento

Belo Horizonte

February 2017

Silva Júnior, Sergio Nunes da

S586f From 2D To 3D By Drawing / Sergio Nunes da Silva Júnior. — Belo Horizonte, 2017
xxviii, 68 f. : il. ; 29cm

Dissertação (mestrado) — Universidade Federal de Minas Gerais - Departamento de Ciência da Computação
Orientador: Erickson Rangel do Nascimento

1. Computação - Teses.. 2. Computação Gráfica. 3. Reconstrução 3D. 4. Voxel. 5. 3D Mesh. I. Título.

CDU 519.6*84(043)

UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

FOLHA DE APROVAÇÃO

From 2d To 3d by drawing

SERGIO NUNES DA SILVA JUNIOR

Dissertação defendida e aprovada pela banca examinadora constituída pelos Senhores:

PROF. ERICKSON RANGEL DO NASCIMENTO - Orientador
Departamento de Ciência da Computação - UFMG

PROF. LEANDRO AUGUSTO FRATA FERNANDES
Departamento de Ciência da Computação - UFF

PROF. LUIZ CHAIMOWICZ
Departamento de Ciência da Computação - UFMG

PROF. RENATO ANTÔNIO CELSO FERREIRA
Departamento de Ciência da Computação - UFMG

Belo Horizonte, 03 de fevereiro de 2017.

Acknowledgments

First of all, I would like to thank him, who has been the entire time with me, who has allowed me to do everything, who has always known all my doubts and has been with me in the happy and sad moments. So, thank you, myself.

Also, I would like to thank a couple of people, who were important in my journey through my Master's degree. Antonio C. Nazaré Junior, for putting me in contact to, in that time, my future advisor. My parents, for supporting me throughout my scholarship budget life. My girlfriend Amanda, for those lovely Sunday afternoons of double studying. My advisor professor Erickson R. Nascimento, for guiding me throughout this learning process and helping me to achieve this presented work. Bruno Miranda, Jordane Almeida, Paulo Ângelo, Tarcísio Rezende and Vinícius Bicalho, for helping me with their modeling perks in the experiments. All colleagues from VeRLab, for giving me a hand with Linux and the company to the *Bandejao*.

"Pardon me, I was absorbed in thought."
(Siegward of Catarina)

Resumo

Nesta dissertação é abordado o problema de criação de modelos 3D a partir de artes conceituais para aplicações 3D de tempo real, como por exemplo, Jogos Eletrônicos. É apresentado uma metodologia automática para construção de modelo 3D baseada no método Visual Hull. O resultado é uma malha 3D texturizada criada a partir de três desenhos bidimensionais. A criação de malhas 3D de objetos da vida real para aplicações em tempo real pode ser uma tarefa trabalhosa, que é comumente produzida manualmente, demandando recursos específicos e altamente especializados como artistas 2D e 3D. Uma técnica difundida na indústria de jogos eletrônicos é a criação iterativa a partir de um modelo grosseiro para um refinado. Nesse processo, um artista 3D começa a modelagem do zero criando uma representação grosseira do objeto final e vai refinando-o iterativamente. Embora, esse processo esteja consolidado na indústria e vem sendo usado assertivamente na prática para a criação de modelos de alta qualidade, ele é lento e custoso. A metodologia apresentada neste trabalho requer apenas três artes conceituais do objeto alvo em pontos de vista diferentes, e como resultado gera uma malha 3D e um mapa de textura. A proposta para essa malha é de ser utilizada como o modelo grosseiro no primeiro passo da modelagem iterativa. Além disso, é proposta uma nova abordagem para melhorar a qualidade dos métodos baseados em Visual Hull, os quais originalmente não conseguem modelar partes côncavas. A qualidade dos modelos 3D gerados é avaliada em experimentos entre artistas 3D, nos quais é medido o quanto a metodologia acelera a criação de modelos 3D. Além de comparações com metodologias similares e baseadas em foto-consistência, para medir o quão acurado é a malha gerada quando comparadas as outras metodologias. Nossos experimentos demostram também que nossa abordagem é capaz de criar modelos por volta de 6% mais acurados que processos similares e baseados em foto-consistência quando se é utilizado artes conceituais. Nossos resultados demonstram que a metodologia acelera na média em 33% a modelagem, e em 13% o processo de criação de conteúdo 3D em geral, quando se é provido o modelo grosseiro automaticamente.

Palavras-chave: Reconstrução de Modelos, Computação Gráfica, Voxel, Geração de Textura, Malhas 3D.

Abstract

In this thesis, we address the problem of creating 3D meshes from pieces of concept art for real-time 3D applications such as Video Games. We present a fully automatic 3D modeling methodology based on silhouette carving, creating textured 3D meshes from three 2D drawings. Creating 3D meshes from real-life objects for real-time applications can be a laborious task, usually produced manually demanding specific, highly specialized human resources such as 2D and 3D artists. A wide-spread modeling approach is iteratively performed in a coarse-to-refined basis. A 3D artist starts modeling from scratch a coarse representation of the given model illustrated by concept arts and then, refines it iteratively. Although this process is consolidated in the game industry and it has been used successfully in practice to create high-quality meshes, it is slow and expensive. Our method uses a set of three concept arts of the target object in different views as input. It generates a 3D mesh automatically and a diffuse color map for texturing that object. This mesh is intended to replace the coarse modeling step in the modeling approach. Furthermore, we propose a novel constraint to improve the silhouette carving methods, which were originally not capable of modeling concave parts. We evaluate the quality of our 3D meshes by assessing its speed up in an experiment with experienced 3D artists and comparing it with other techniques of similar approach and photo-consistency based methods. Our experiments show that our approach creates around 5% more accurate meshes compared to the similar approach and photo-consistency based methods when working with pieces of concept art. Also, we show that it speeds up in an average of 33% the modeling step, and 13% the whole 3D content production time by providing the coarse model automatically.

Palavras-chave: Model Reconstruction, Computer Graphic, Voxel, Texture Generation, 3D Mesh.

Resumo

Nesta dissertação é abordado o problema de criação de modelos 3D a partir de artes conceituais para aplicações 3D de tempo real, como por exemplo, Jogos Eletrônicos. É apresentado uma metodologia automática para construção de modelo 3D baseada no método Visual Hull. O resultado é uma malha 3D texturizada criada a partir de três desenhos bidimensionais. A criação de malhas 3D de objetos da vida real para aplicações em tempo real pode ser uma tarefa trabalhosa, que é comumente produzida manualmente, demandando recursos específicos e altamente especializados como artistas 2D e 3D. Uma técnica difundida na indústria de jogos eletrônicos é a criação iterativa a partir de um modelo grosseiro para um refinado. Nesse processo, um artista 3D começa a modelagem do zero criando uma representação grosseira do objeto final e vai refinando-o iterativamente. Embora, esse processo esteja consolidado na indústria e vem sendo usado assertivamente na prática para a criação de modelos de alta qualidade, ele é lento e custoso. A metodologia apresentada neste trabalho requer apenas três artes conceituais do objeto alvo em pontos de vista diferentes, e como resultado gera uma malha 3D e um mapa de textura. A proposta para essa malha é de ser utilizada como o modelo grosseiro no primeiro passo da modelagem iterativa. Além disso, é proposta uma nova abordagem para melhorar a qualidade dos métodos baseados em Visual Hull, os quais originalmente não conseguem modelar partes côncavas. A qualidade dos modelos 3D gerados é avaliada em experimentos entre artistas 3D, nos quais é medido o quanto a metodologia acelera a criação de modelos 3D. Além de comparações com metodologias similares e baseadas em foto-consistência, para medir o quão acurado é a malha gerada quando comparadas as outras metodologias. Nossos experimentos demostram também que nossa abordagem é capaz de criar modelos por volta de 6% mais acurados que processos similares e baseados em foto-consistência quando se é utilizado artes conceituais. Nossos resultados demonstram que a metodologia acelera na média em 33% a modelagem, e em 13% o processo de criação de conteúdo 3D em geral, quando se é provido o modelo grosseiro automaticamente.

Palavras-chave: Reconstrução de Modelos, Computação Gráfica, Voxel, Geração de Textura, Malhas 3D.

List of Figures

List of Tables

List of Algorithms

Contents

Chapter 1

Introduction

In this chapter, we introduce the context and motivation of this thesis by presenting an overview of how the entertainment industry deals with 3D content creation. Also, we state the problem tackled in this work and our main contributions.

1.1 3D Content Creation

Three-dimensional meshes are ubiquitous structures in a myriad of computer graphics applications such as the generation of synthetic images, virtual scenarios and characters for games and movies, architectural modeling and scientific visualization to name a few. High-quality 3D meshes play a central role in achieving good results in both rendering and animating algorithms as they are the kernel of 3D graphical applications and they interact directly with an important sense: our sight.

In general, the production of any 3D model asset starts with creating pieces of concept art. Concept art is an 2D illustration to convey information of a visual structure such as characters, buildings, and scenario props. Typically, a 2D artist tries different designs by illustrating the object in order to finally deliver a set of concept art as blueprint to a 3D artist. Blueprint is a technical illustration that depicts a target object in canonical viewpoints (front, side, and top) in an orthogonal perspective. Figure 1.1 illustrates each step of the process of concept art creation. In the very first step, a rough drawing known as sketch is produced for idea validation.

The task of three-dimensional modeling is a hard and painstaking process that demands considerable effort even from a skilled 3D artist. A widespread modeling approach within the game industry is accomplished by performing this task iteratively, known as Box Modeling. In Box Modeling, 3D artists start the modeling process with a triangular 3D primitive mesh, for example, a cube or sphere. Next, they deform

(a) Sketch (b) Conception (c) Blueprint

Figure 1.1. Illustrations depicting a 2D artist workflow of concept art creation as blueprint. (a) A rough drawing known as Sketch is produced for ideia validation, (b) designs from different viewpoints are produced for conception. (c) A final set of concept art as Blueprint is handed over to a 3D artist. (Reproduced from *http://hoatheartist.com/wp-content/uploads/2011/09/Lion-O_Concept.jpg*).

and extend the primitive towards to the target object by using operations such as extruding, slicing, translating, rotating and scaling. They begin modeling a coarse representation of the target object, and then, turn it into a refined one. Although this technique allows to produce high quality 3D models, it holds a time overhead by the generation of useless midway models. Figure 1.2 illustrates a Box Modeling progression of a human head as example.

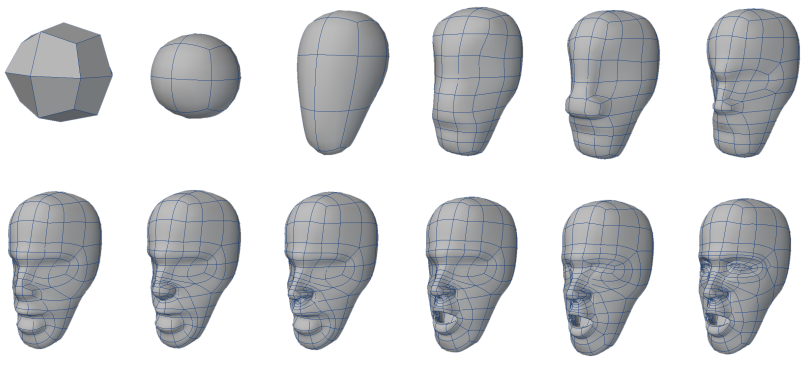


Figure 1.2. An example of a progression of a 3D human head modeled iteratively with the Box Modeling technique. Starting from a cube-shaped primitive, it is deformed and extended until it becomes a refined 3D human head. (Reproduced from *http://forums.newtek.com/showthread.php?131319-Another-Box-Model-Head-Steps-Thread*).

1.2 Motivation and Problem Definition

In the game industry, most specifically AAA productions (blockbuster games with budget of millions of dollars), the artistic process is one of the most time consuming and expensive in the entire production pipeline, as we can check in [Ahearn, 2001], a web article where the author advises how to budgeting and scheduling your game. The 3D content creation appears in the early stages of the game production, being demanded in prototypes and ideas validation of graphics and gameplay for example. It is not rare to see game development teams be composed of a majority of artists. In addition, virtually all tools for 3D modeling have a steep learning curve, which reflects the cost of its utilization. Methodologies that could facilitate the task of three-dimensional modeling by automating and speeding it up, which would consequently decrease its process cost, are a valuable tool that would have a strong appeal in this industry.

Over the last few decades, several systems and methodologies have been developed for automating parts of the creation of detailed 3D models. Multi-view reconstruction and Sketch-based modeling are the most representative approaches of this automation effort. The former approach has shown impressive results by using digital image of scenes and objects; however, it has to tackle with the known issues related to the digital image processes such as noise and camera parameters estimation. The latter has gained the attention of the industry, since these techniques may perfectly fit in the artist's workflow. However, it demands changes in the production pipeline as a new tool is added to it, which may not be desirable.

Motivated by the game industry high demand of art content and the lack of smart and automatic solutions for 3D mesh and texture generation that do not change the artistic pipeline, we intend to reduce the cost of the 3D content creation for entertainment purposes by increasing its production speed. We propose a methodology capable of generating a 3D mesh to be used as a head start for 3D artists on the early stage of the modeling process. We place our work as a gathering of the best of multi-view reconstruction and sketch-based modeling approaches. Multi-view reconstruction is used as we perform a surface estimation using three 2D images, and Sketch-based modeling is used as we use concept art rather than digital image. Our approach is an automatic 3D modeling method that does not require any user interaction throughout the process. In order to be akin to the artistic creation pipeline, our methodology works only with three pieces of concept art from well-defined viewpoints: front, side and top (canonical viewpoints used in the modeling process) as input. Our solution for converting 2D to 3D is based on the Visual Hull [Laurentini, 1994] methodology. Although one can argue

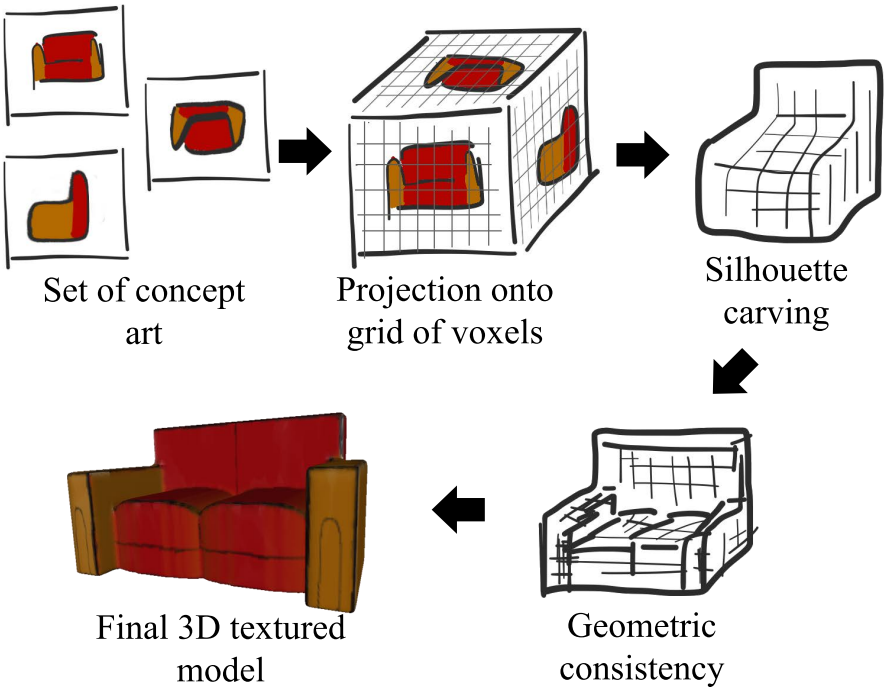


Figure 1.3. The overview of our method. By using as input three pieces of concept art of different viewpoints of an object: front, side and top view, it starts with a grid of voxels projecting the drawings onto the grid, performing a silhouette carving. Then, it applies the geometric consistency to check any concave part. At last, it generates a texture color map based on the concept art.

that adding more views could aid to solve any ambiguity issue this method holds, one of our main motivations is not to increase the already heavy content creation workflow. We accomplish that by mimicking the artistic creation pipeline. The concept art must be drawn in an orthogonal view, just as artists usually do. Furthermore, we propose an approach for modeling self-occluded parts that are not tackled on the Visual Hull methodology. We address it as geometric consistency. In addition, our methodology creates a texture map that is applied onto the final mesh, which gives its surface color. We estimate this texture map based on the color information within the input pieces of concept art. Figure 1.3 presents an overview of our methodology.

1.3 Contributions

A summary of the main contributions of this thesis to the 3D content creation for entertainment purposes is as follows:

- a fully automatic method for 3D mesh creation from three pieces of concept art

based on silhouette carving methodology;

- a new approach to overcome the ambiguity problem presented in the regular silhouette carving not based on texture information;
- an approach to enhance spherical shapes for the regular silhouette carving;
- a technique for extraction of a color texture map based on colors within the concept art;

1.4 Organization

We have organized this thesis in five chapters and two appendices. Regardless this introduction, the remaining of this document is presented as follows:

Chapter 2 - Related Work: discusses relevant work in the area of 3D modeling and texture generation.

Chapter 3 - Methodology: presents our proposed solution of 3D modeling and texture estimation from concept art. Furthermore, we depict a user interface of our technique.

Chapter 4 - Experiments: presents the quantitative and qualitative experiments performed to validate our methodology and shows several results.

Chapter 5 - Conclusions: closes this thesis with our conclusions, work limitations and directions for future work.

Appendix A - User Interface user case: shows a step-by-step how to use our user interface.

Appendix B - 3D Artists Experiment Time Discrimination: divulges the times spent individually by artists 3D in our experiment.

Chapter 2

Related Work

In this chapter, we present previous approaches of 3D modeling, automatic and interactive, most related to our method. We depict techniques similar to ours, automatic 3D modeling from concept art. Also, we present techniques concerning automatic texture generation.

2.1 3D Reconstruction

Reconstructing the geometry and texture of objects of the world remains one of the key challenges in computer vision and computer graphics. In the computer vision field, we witnessed a large body of proposed approaches based on digital images for creating 3D models [Seitz and Dyer, 1999], [Kutulakos and Seitz, 2000], [Furukawa and Ponce, 2010], [Oswald, 2015] (Figure 2.1) while in the computer graphics field, a common procedure consists of using interactive tools such as Teddy [Igarashi et al., 1999] (Figure 2.2), 3Sweep [Chen et al., 2013], SecondSkin [De Paoli and Singh, 2015], Andre and Saito system [Andre and Saito, 2011], Cohen et al. interface [Cohen et al., 1999], EverybodyLovesSketch [Bae et al., 2009] or synthetic images such as two-dimensional drawings of a simplified model to guide the modeling [Rivers et al., 2010]. More recently, Buchanan et al. [Buchanan et al., 2013] and Levi et al. [Levi and Gotsman, 2013] proposed automatic approaches for 3D modeling from concept art.

For the sake of methodology similarity, we group those 3D modeling methods in three classes based on their strongest feature: Digital Image-Based Methods, Interactive Techniques, and Automatic Methods.

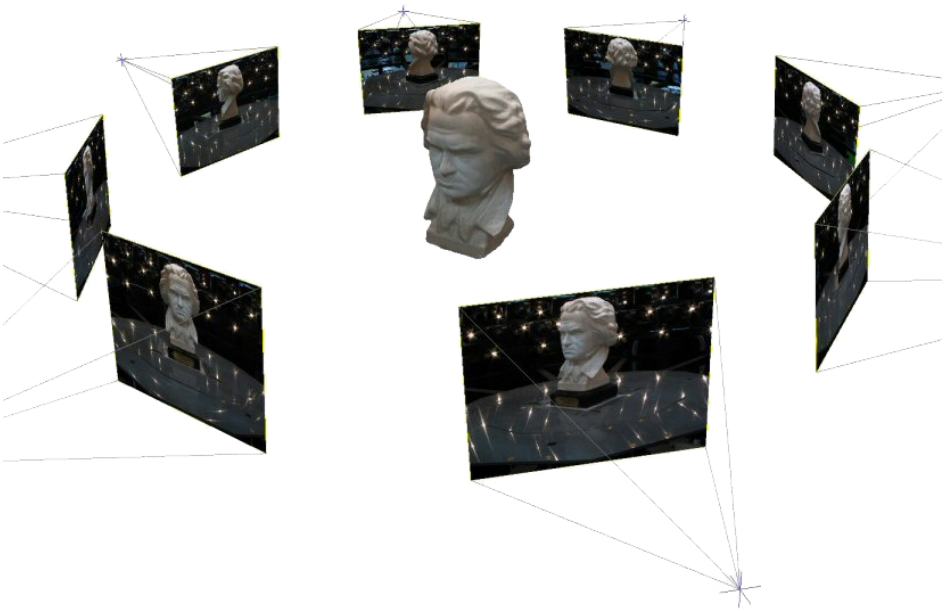


Figure 2.1. An example of multi-view reconstruction. (reproduced from [Oswald, 2015])

2.1.1 Digital Image-Based Methods

Techniques such as Voxel Carving [Seitz and Dyer, 1999], Space Carving [Kutulakos and Seitz, 2000], and Multi-View Reconstruction [Furukawa and Ponce, 2010] and [Oswald, 2015] use digital images captured by real cameras as input data. In general, these methods demand extra work, due to the need of knowledge of the intrinsic and extrinsic camera parameters necessary to the process. They also have to tackle noise problems caused by the image acquisition.

Voxel Carving generates the model by carving the object silhouette onto a grid of voxels. Since it takes into account only the object silhouette, it is not capable of modeling concave parts within the silhouette. Even though Space Carving also creates a based-voxel model, it overcomes the Voxel Carving drawback by applying a photo-consistency check. Photo-consistency enhances the modeling procedure by looking for pairs of points between the images as well as the silhouette carving. The points pairing in Photo-consistency is performed by keypoints matching, a usual approach in Computer Vision for image recognition and object detection. Although Photo-consistency improves reconstruction by imposing matching constraint, it highly relies on texture of the objects' surfaces to perform properly its keypoint matching.

Multi-View Reconstruction performs surface reconstruction and photo-consistency in a more sophisticated way by inferring the camera's rigid transformation amongst the images viewpoint. Even though, it is capable of reconstructing complex

Figure 2.2. An example of Teddy's user interface and some results achieved by the tool. (Reproduced from [Igarashi et al., 1999])

surfaces, the camera transformation requirement lightly prohibits Multi-View Reconstruction working with drawings. It must have a physically plausible rigid transformation between viewpoints and that is not guaranteed for drawings. Also, it deeply depends on texture of the object as well.

In general, concept art lacks of texture information as artists represent only the borders of the surface. This fact severely complicates the utilization of these methods when modeling from drawings. For an extensive comparison of Multi-view reconstruction approaches, the reader is referred to the work of Seitz et al. [2006].

2.1.2 Interactive Techniques

Other approaches, such as Teddy [Igarashi et al., 1999] (Figure 2.2), propose the use of interactive tools. This method is based on inflated geometrical surfaces to create simple toy models. It can generate interesting results for organic modeling such as stuffed animals and character models. 3Sweep [Chen et al., 2013], for its turn, is based exclusively on one digital image for the 3D reconstruction. First, the user is required to draw a profile curve of a generalized cylinder. Next, by dragging the user's mouse, the algorithm makes a generalized cylinder along the swept path snapping it into the outline of the 2D shape onto the image. Nonetheless, the process offers a limited range of primitives that can be used to generate the surface and cannot manage occlusion.

Andre and Saito [2011] proposed a single view approach based on the sweeping algorithm (generalized cylinders). It requires the user to draw two outlines (cross section of the shape and an ellipse perpendicular to the cross section) to delimit shapes on the sketch. It is able to generate complex models from a single viewpoint.

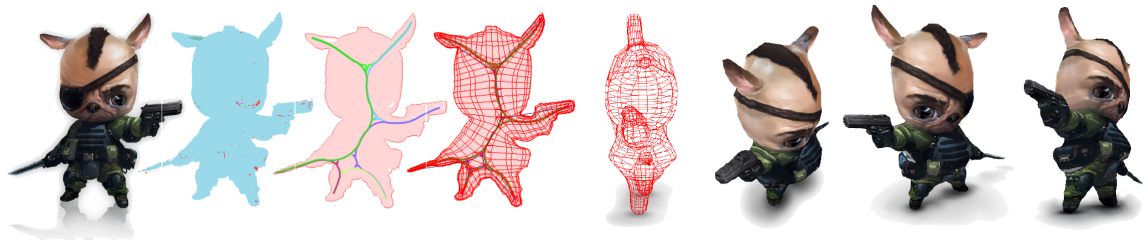


Figure 2.3. An example of the automatic approach proposed by Buchanan. They reconstruct a rigged 3D character from a single view concept art. (Reproduced from [Buchanan et al., 2013])

Rivers et al. [2010] proposed a simple algorithm towards computing the silhouette cylinders and applying Constructive Solid Geometry (CSG) operators to build the 3D model. They assume that silhouettes of the sub-parts of a 2D drawing from two different views (front and side) can provide enough information to create a 3D model. Their approach shows acceptable results in generating models for man-made objects.

Brazil et al. [2015] proposed a framework for sketch-based modeling using concepts of adaptive meshes. They proposed the system called *Detail Aware Sketch-Based Surface Modeling* (DASS), which allows the user to create meshes from a sketch and refine it by controlling local modifications while maintaining the surface outside of the area of interest unchangeable. It presents a fast-paced solution to generate coarse sketch-based models.

We point out that even though all these tools are capable of generating models from sketches, they still require extensive user interaction. Overloading the already heavy artist workflow, which is clearly an undesired effect. Nonetheless, they usually do not deliver a model ready for use as the regular manual 3D modeling pipeline does.

2.1.3 Automatic Methods

In regard to automatic modeling methods from concept art, Buchanan et al. [2013] (Figure 2.3) proposed an approach to reconstruct character model from a single view image. Their methodology produces a 3D mesh based on a single view input, creates a skeleton that rigs the mesh, which permits the mesh to be deformable for animation. In their method, they start creating a skeleton on the input artwork concerning the artwork outline, then build a rigged triangulated mesh on the skeleton so that the surface matches the artwork outline.

Entem et al. [2015] proposed an approach to generate a 3D model of animals from a single view sketch from a sagittal plane. In their methodology, the drawing is con-

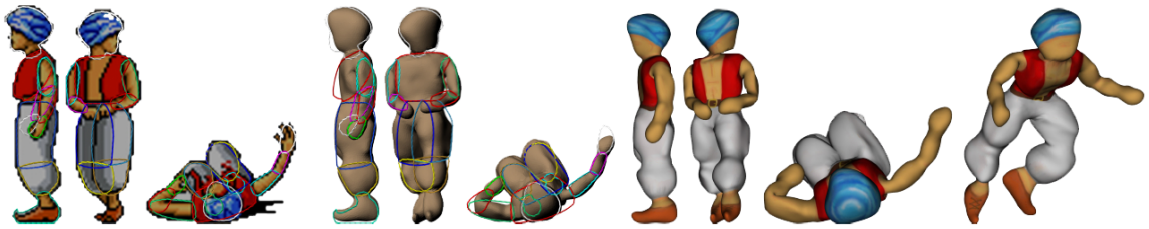


Figure 2.4. An example of the automatic multi-view approach proposed by Levi. They reconstruct a rigged 3D character from a set of frames from a spritesheet. (Reproduced from [Levi and Gotsman, 2013])

verted to a set of parametric curves, assembled in cycles and then classified under one of four categories that they propose. They process each cycle part by part, representing them as a kind of implicit surface, scale invariant integral surfaces. Even though their approach proved to run rapidly, it demonstrates to be on early development stage having several limitations such as every animal must hold such as nonself-overlapping limbs, and the user always must draw both contours for a pair of symmetric limbs. Levi and Gotsman [2013] (Figure 2.4) proposed a system that generates a rigged 3D model from a spritesheet (a file that gathers a group of images of the same object in different poses. It is mostly used for 2D animation purposes). It is capable of reconstructing surfaces from views that are not in canonical views (front, top and side view) and does not remain in the same pose among them. They address each view as a frame. For each frame, the user has to mark the rigid parts, producing joints, on the frames and then matching them between frames. Since it does not work on canonical views, they estimate the extrinsic camera parameters based on those rigid transformation parts. Next, they reconstruct the surface based on Level Set Method (LSM) upon a grid of voxels representation. It is able to work with images that are represented in the perspective view; however, it presents better results when working with the orthographic view.

Our methodology is in part an extension and enhancement of the work proposed by Viana et al. [2014]. They tackled the problem of 3D modeling creation from concept art. Their work is built upon a silhouette carving approach using three points of view. However, they do not address the self-occlusion problem present in silhouette carving based methods, which is one of the limitations of their work.

Despite those methods being automatic, those that work with only one image are extremely limited to the kind of models they are able to generate. Entem et al. [2015] can only produce animal models and Buchanan et al. [2013] can only generate

character models with very limited shape complexity. On the other hand, our method shows broader capacity of the kind of models it can create by working with three drawings similarly to a 3D artist. Levi and Gotsman [2013] tackle the 3D modeling problem using multi images, albeit they do not address the concavity problem present by working with silhouettes, and it could not be considered completely automatic as the user must mark the rigid parts. Our method addresses this problem and proposes a new approach to overcome it, regardless of being completely automatic.

2.2 Automatic Texture Generation

Texture Generation plays an important role in 3D content creation as well. Textures are images applied onto the 3D model's mesh enhancing the realistic or artistic appeal of the 3D model. A texture map is a 2D image that gives information about the model surface such as color and light effects, e.g., Figure 2.5. The process of automatic texture map generation consists of techniques and algorithms for assessing and estimating of texture maps automatically.

3Sweep extracts color information of the mesh surface from the input digital image. Space Carving method also extracts the surface color from the input images. It projects back a ray from the surface to the images, similarly photo-consistency methods. Maillot et al. [1993] provided an interactive approach for texture mapping using an energy minimization function, where it allows handling non-continuous texture maps needed for complex models. In their process, they offer tools for guiding the map generation and an algorithm that automatically generates an atlas for any type of object.

Sander et al. [2001] proposed an automatic texture map approach for progressive

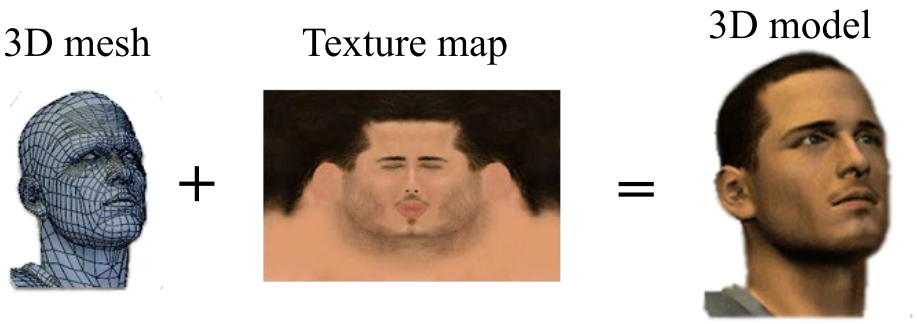


Figure 2.5. An illustration of how a texture map can enhance a 3D mesh by giving diffuse color and properties of how light interacts with the mesh surface. (Reproduced from *https://nickdesaulniers.github.io/RawWebGL/texture.gif*)

meshes problem, where it provides a global texture map that holds valid among levels of the model refinement. Matsuyama et al. [2004] provided a texture map for 3D models reconstructed from videos. They reconstruct the texture from the frames of a video. To define the surface color, it uses information from the video's frames and the surface normal.

Aware of how valuable is a texture map for the sake of 3D model quality, we propose an automatic method for color extraction to be used as texture map for the model. Similarly to Space Carving approach, our approach is based on projecting back the model's surface to the input pieces of concept art.

To summarize, our methodology is strongly connected to the third aforementioned 3D modeling group. However, differently from the relevant related work, we go further and address the concavity problem, which is a strong drawback of silhouette-based modeling methods, in the drawing context. As texture discrimination and physically plausible reference cannot be guaranteed in the drawing context, we propose a geometric consistency to tackle this problem.

Chapter 3

Methodology

In this chapter, we present our methodology to tackle the problem of 3D modeling and automatic texture generation. The output of our methodology can be used as a head start by an artist in 3D content creation for entertainment applications as well as entry geometry for digital sculpting tools. Additionally, we present a user interface that has implements the methodology. It has widgets for drawing and let the user see the resulting 3D model within the application in a fast-paced mode.

In general, a 3D artist creates the model using three images from different points of view of the object provided by a 2D artist as is shown in Figure 3.1. Those images in many cases are pieces of concept art from the front, side (right) and top view of the object considering an orthogonal projection, commonly addressed as blueprints. Concept

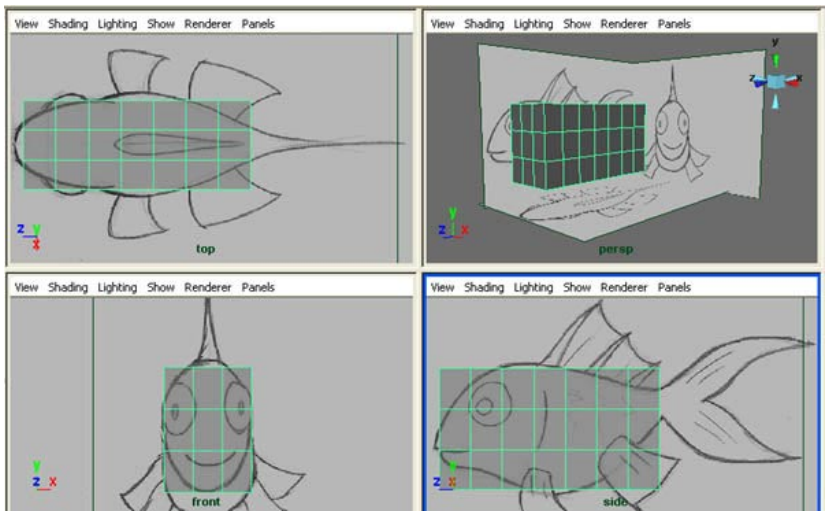


Figure 3.1. An example of how the interface of a 3D modeling software looks like when configured for modeling purpose. (Reproduced from *http://www.free3dtutorials.com/cartoon-fish-tutorial.php*)

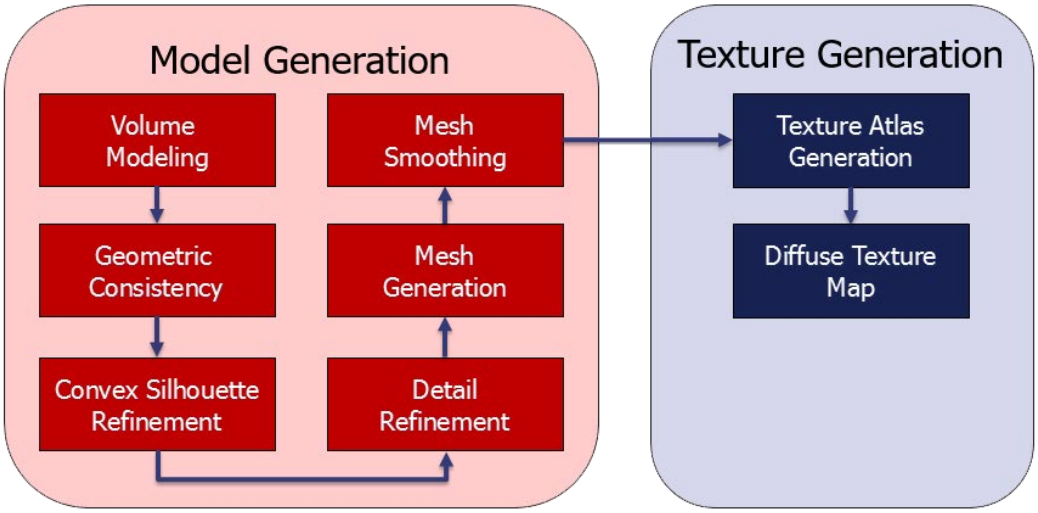


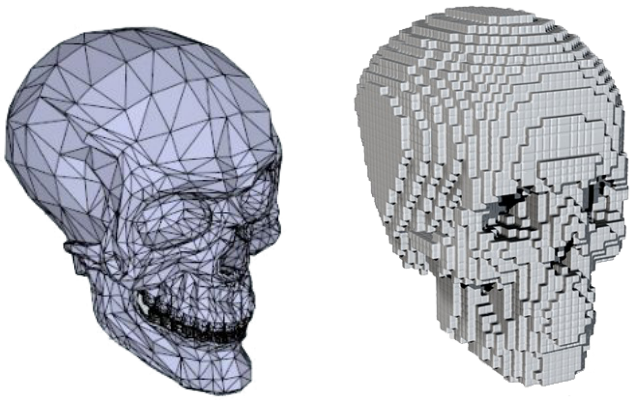
Figure 3.2. The overview of the main steps and sub-steps of our methodology.

art, differently from sketches, are carefully crafted drawing pieces. Our methodology receives these pieces of concept art as inputs and automatically delivers a coarse textured model for use as a head start in a refinement step or sculpting by 3D artists.

Our technique splits into two main steps, modeling and texturing. The first step is composed of six main routines. It starts with a volume modeling, creating a coarse mesh of the object by extracting the silhouettes from the concept art and back projecting them into a grid of voxels. Then, if any surface inconsistency occurs due to silhouette carving issues, we apply our geometric consistency procedure. In case of having any convex silhouette, we apply an approach that we named as convex silhouette refinement, this solution enhances rounded shapes. Next, we apply a detail refinement to strengthen details that are skipped by modeling in a grid of voxels. Finally, we perform a conversion from our grid of voxels to a triangular mesh representation and apply a mesh smoothing to lessen the voxels ridges. For the second step, we first generate a texture atlas that maps each triangle of the model to a position in a squared plane representation. Then, we create a texture map to be applied to the model surface. The color intensities are a linear combination of image patches from the input concept art. An overview that summarizes our methodology is present on Figure 3.2.

3.1 Model Generation

Three-dimensional models are mathematical representations that simulate the surface, as well as the interior in some cases, of the world's objects. They are widely used in the entertainment industry such as Digital Games and Animation Movies. However,



(a) Triangulated model (b) Voxel-based model

Figure 3.3. Illustration of the 3D representation's types. (a) A triangulated 3D Skull. We can see the edges and the triangles faces. (b) A voxel-based 3D Skull. We can check the discrete way the shape feels. (Reproduced from *www.3dxtras.com/content.php?page=3dmodels_detail&prodid=10305* and *www.123dapp.com/smb-123D_Design/Voxel-Skull/1557274*)

they are not restricted to these areas. We can witness them in medical use, e.g., computerized tomography scan visualization, in civil engineering, Computer-Aided Design software, to name a few.

There are several ways of representing a 3D model such as triangulation, implicit surfaces, surfels, and voxel-based volume. A triangulated 3D model is represented by vertices and edges (Figure 3.3(a)), which form the faces of polygons (most of the time, triangles). This kind of representation is the preferred one in real time applications such as Digital Games, as it is intuitively manipulated and computationally lighter than other representation methods. Nonetheless, it often represents only the model's surface. In a voxel-based volume representation (Figure 3.3(b)), the 3D world is discrete and a voxel is the smallest unit as just as a pixel is the smallest unit within a screen. The voxels compound the whole object, which makes it solid (having surface and interior).

In this thesis, we have used a representation of voxels in the initial modeling process, because it is intuitively easier to work with volume for silhouette carving. Then, after the rough surface is carved in voxels, we turn it into a triangular representation for texture map estimation and, mainly, because it is the format that 3D artists are familiar with.

3.1.1 Volume Modeling

Similar to Visual Hull algorithm [Laurentini, 1994], on the first step of our modeling technique, we extract the silhouettes of the object from the concept art and project

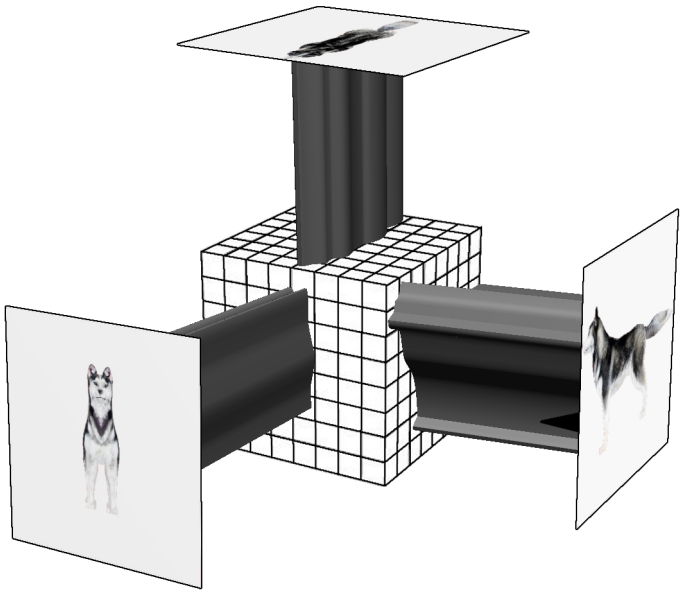


Figure 3.4. Modeling by silhouettes illustration. We have a whole voxel's cube where the concept art silhouettes are projected. The silhouettes intersection gives the first coarse model.

them onto a voxel-based cube-shaped volume as we can check on Figure 3.4. However, we go beyond and instead of only considering a pixel being either background or object such as Visual Hull, we consider that a pixel can represent either background, flat surface, or border (surface discontinuity) within the object. Thus, a voxel that is represented by a patch of pixels, can be labeled as containing any sort of the three states. Let $\mathcal{V}_x^s$ be the set of voxels labeled having flat surface pixels for the viewpoint x and $\mathcal{V}_x^b$ be the set of voxels labeled having border pixels for the viewpoint x. Our volume is given by

$$volume = \bigcap_{vp}^{viewpoints} (\mathcal{V}_{vp}^s \cup \mathcal{V}_{vp}^b), \tag{3.1}$$

using a grid of voxels.

Initially all voxels of the initial grid are set as visible. Each voxel is represented by one patch of pixels from each input piece of concept art. Patches of exclusively background regions have their voxels carved by setting them as invisible, as is presented in Algorithm 3.1.

The method *project*() in the image object checks all pixels within that patch giving that voxel a label as is shown in Algorithm 3.2. The label is an unsigned integer variable that holds three-valued flag, which can be BACKGROUND, SURFACE, and

Algorithm 3.1 Algorithm of silhouette carving.

```
1: inputs: the grid of voxels, the set of pieces of concept art
2: function silhouetteCarving(grid, frontImg, sideImg, topImg)
3:   for x = 0 to grid.getSize() do
4:     for y = 0 to grid.getSize() do
5:       for z = 0 to grid.getSize() do
6:         if  frontImg.project(x,y) == BACKGROUND
7:           or sideImg.project(z,y) == BACKGROUND
8:           or topImg.project(x,z) == BACKGROUND then
9:           grid.getVoxel(x,y,z).visible = false
10:        end if
11:      end for
12:    end for
13:  end for
14: end function
```

EDGE. *PatchSizeInPixels* is the ratio of image dimension in pixel by size of the grid of voxel.

As we intersect all of the silhouettes projection, it is crucial that all pieces of concept art match among themselves. We grant that performing an alignment by their bounding boxes[1] before the projection phase. The bounding box of each piece of concept art is given by the first non-background pixel in each direction, i.e., the

[1]A bounding box is a struct with two 2D coordinates that represent the minimum and maximum points of a box shape.

Algorithm 3.2 Algorithm of project method.

```
1: inputs: initial x pixel position pX, initial y pixel position pY
2: function project(pX, pY)
3:   var projectionLabel = 0
4:   for x = pX to pX + patchSizeInPixels do
5:     for y = pY to pY + patchSizeInPixels do
6:       if  pixelBuffer.At(x,y) == BACKGROUND then
7:         projectionLabel |= BACKGROUND
8:       else if  pixelBuffer.At(x,y) == EDGE then
9:         projectionLabel |= EDGE
10:      else if  pixelBuffer.At(x,y) == SURFACE then
11:        projectionLabel |= SURFACE
12:      end if
13:    end for
14:  end for
15:  return projectionLabel
16: end function
```

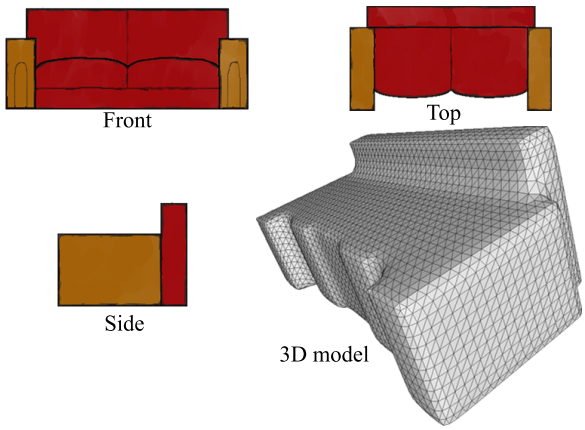


Figure 3.5. The figure presents an example of 3D modeling using a standard Visual Hull method. Since this standard method only employs the information from of the shape's silhouette to generate the 3D surface, they cannot handle occluded regions. Notice that the couch's seat is not correctly modeled due to the occlusion produced by the arm in front of the seat on the side view, even though there is information about the seat in the concept art of the front and top views.

most left non-background pixel regardless its Y coordinate is considered the $minx$ of the bounding box and so on. Thus, having the bounding box of each piece of concept art, we scale them forcing a matching amongst them. For example, the x extent of the front concept art must match the x extent of the top concept art, and y extend of the top concept art must match the x extent of the side concept art.

3.1.2 Geometric Consistency

The main drawback of a silhouette-based approach is its inability to model concave parts that are not seen by all views. Even a simple model could have a wrong estimation of its shape as it is illustrated in Figure 3.5. A standard Visual Hull algorithm failed to model the seat of the couch since the arm of the couch is occluding it on the side view. In this example, even if more viewpoints were added, it would not model the expected surface. We call attention to that as the addition of view is commonly proposed as a solution to this silhouette carving issue.

Image-based modeling methods such as Space Carving [Kutulakos and Seitz, 2000] handle concave, i.e., self-occlusions, regions by applying a photo-consistency test, where the color of each voxel should be similar in all cameras that can see it for being considered a valid point on the surface. Although these methods are capable of constructing 3D models for complex objects with concave regions, they are extremely dependent on color and grayscale variance. Therefore, they are doomed to fail when working with

concept art as they usually lack texture information.

Figure 3.6 illustrates the challenge of carving concave parts from silhouettes. The projection of the 3D point $\mathbf{P}$ to the $\mathbf{p_f}$ pixel coordinate on the front concept art is given by $\mathbf{p_f} = \Pi\mathbf{P}$, where Π is the orthographic projection matrix:

$$\Pi = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}. \tag{3.2}$$

The corresponding pixel $\mathbf{p_s}$ in the side concept art lies in the line s (red line in Figure 3.6), since any point in this line projects onto $\mathbf{p_s}$, we cannot estimate what point $\mathbf{P}$ projects to $\mathbf{p_s}$ and thus it is not possible to carve the right voxel.

To overcome this limitation, our method performs a geometric consistency by checking out the grid of voxels. The information of the extracted edges from the concept art can act as a constraint in the carving process given by

$$\forall \mathbf{v} \in \mathcal{V}, \quad is\_consistent(\mathbf{v}), \tag{3.3}$$

where $\mathcal{V}$ is the set of all voxels in the volume; and $is\_consistent(\mathbf{v})$ is defined as

$$\forall vp \in viewpoints, \quad \mathbf{v}_{vp}^{visible} + \mathbf{v}_{vp}^{border} = edge(\mathbf{v}, vp), \tag{3.4}$$

where $\mathbf{v}_{vp}^{visible}$ informs whether the given voxel $\mathbf{v}$ is visible by viewpoint vp; $\mathbf{v}_{vp}^{border}$ informs whether the given voxel $\mathbf{v}$ is surrounded by border voxels in viewpoint vp; and $edge()$ checks whether the given voxel $\mathbf{v}$ lies in an edge in the concept art patch from the viewpoint vp.

Whenever a voxel does not satisfy Equation 3.3 and

$$\sum_{vp}^{viewpoint} edge(\mathbf{v}, vp) \geq \varepsilon, \tag{3.5}$$

the geometric consistency is broken under the assumption that all edges represent surface discontinuity. In our current work, we use $\varepsilon = 2$. In particular, this is true for concept art, where strong strokes usually represent a discontinuity in the surface and define the shape of the model. We call this consistency check as geometric because it enforces the visibility of a voxel if this voxel projects to an edge in two different viewpoints at least. Currently in our method, the artist must highlight the discontinuity edges in the drawing simply stroking with a totally black brush.

Let $\hat{\mathcal{V}}$ be the set of all voxels that project onto an edge in at least ε views in all

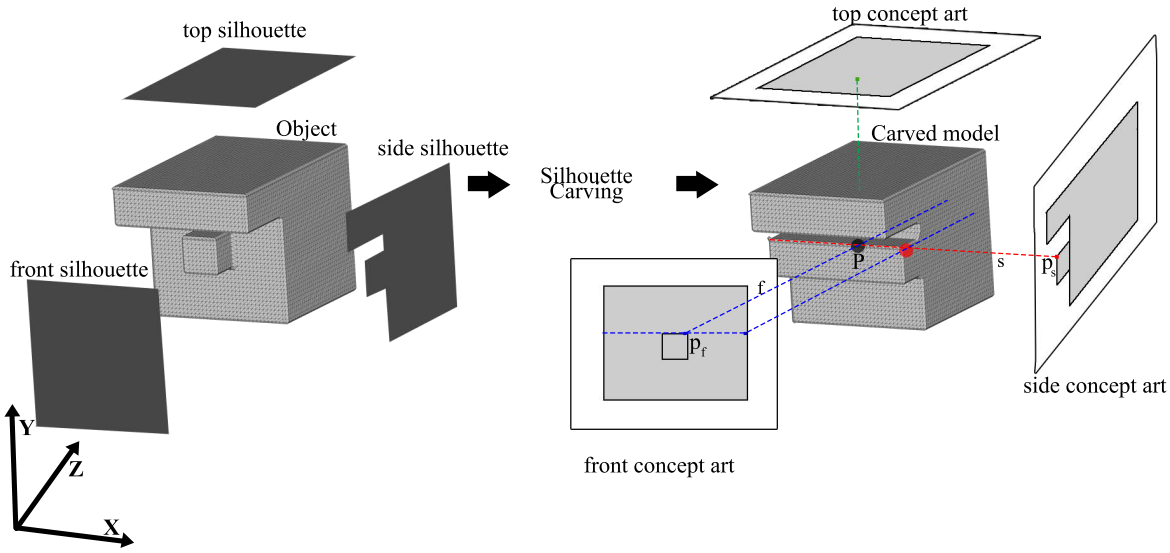


Figure 3.6. This diagram illustrates the inability of the Visual Hull method to model concave surfaces. Any point in a viewpoint projects onto a line when transformed to another viewpoint. The Visual Hull method uses the intersections to generate the surface. However, concerning only the silhouettes, the intersections may be misplaced (red point). It is clear that the right ray intersection occurs at the point **P** (black point).

viewpoints. For each voxel $\mathbf{v} \in \hat{\mathcal{V}}$, we test the geometrical consistency by projecting $\mathbf{v}$ onto an image plane (concept art) as follow:

$$\mathbf{p_k} = \Pi^k \mathbf{v}, \tag{3.6}$$

where Π^k is the projection matrix for the k-th view.

If $\mathbf{p_k}$ does satisfy Equation 3.5 but does not Equation 3.3, we can conclude that $\mathbf{v}$ is occluded from one of these views. Then, we evaluate each voxel along the six directions $\{X, -X, Y, -Y, Z, -Z\}$ for potential carving. In Figure 3.7 (in this example, the top view was hidden for the sake of visualization) we can see that the green voxel is not visible from side concept art since it is being occluded by the light red voxels. This breaks the geometric consistency on the green voxel as it lies on edge pixels in both images and forces our method to search along all of the six directions to make it visible in all pieces of concept art.

The next step is to find in which direction the voxel is being occluded and should be visible. There are three possible cases:

1. a direction in which there are no voxels to carve (directions $Y, -Y$ and $-Z$ in Figure 3.7);

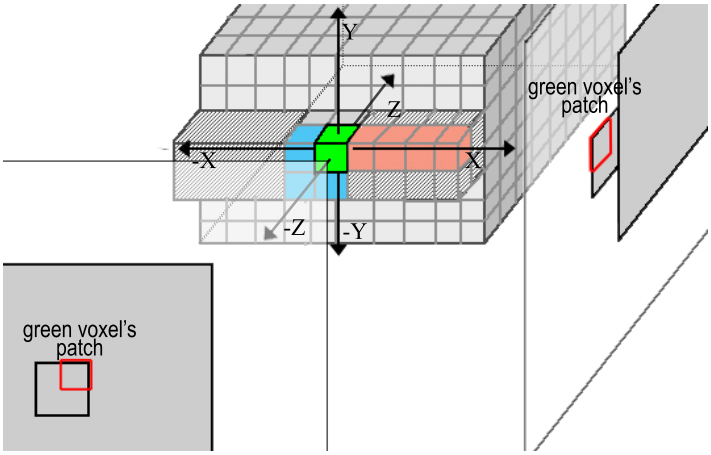


Figure 3.7. In order to check out in what direction a voxel **v** (green voxel) should be visible, our method looks for it in six directions $\{X, -X, Y, -Y, Z, -Z\}$. If the voxel **v** is occluded by voxels that project into a surface in any view (red voxels), we carve these voxels making **v** visible. Otherwise, if **v** is occluded by voxels that project into edge pixels (blue voxels), they remain opaque.

2. a direction where the voxel is occluded by another voxel that projects to an edge pixel, the light blue voxels in Figure 3.7;

3. a direction where the voxel is occluded by voxels that project on a flat region of the concept art.

Our method selects to carve all voxels, starting from the occluded voxel, along each direction that has a neighbor that does not project to an edge pixel (the light red voxels in Figure 3.7). The pseudo-code is shown in Algorithm 3.3. The method $satifiesGeometricConsistency()$ checks if the given voxels is labeled as edge for at least two view points, and in positive case, checks if it is seen or laying amidst a region of other edge voxels. The method $carveVoxelsFromTo$ is shown in Algorithm 3.4.

Figure 3.8 shows a result of the same couch generated by a standard Visual Hull algorithm and one amended by our geometric consistency. One can clearly see that our approach successfully models the couch seat. Although our approach showed effectiveness for single depth levels (when there are only two levels of depth: foreground and background), it may not work properly when there are several levels of depth that need solving.

3.1.3 Convex Silhouette Refinement

Forwardly, in the case of having any silhouette that is convex, we dynamically refine the grid of voxels by adjusting the convex silhouette in regard to the other two corre-

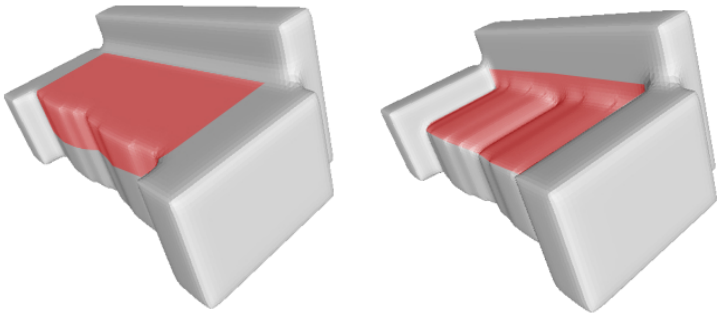
Algorithm 3.3 Algorithm of geometric consistency.

```
1: inputs: All Visible Voxels 𝒱
2: function applyGeometricConsistency(𝒱)
3:   for all v ∈ 𝒱 do
4:     if not satifiesGeometricConsistency(v) then
5:       for all d ∈ {X, −X, Y, −Y, Z, −Z} do
6:         for all v̂ ∈ 3D-neighborhood do
7:           p_d = projects(v̂, d)
8:           if p_d is on a flat region then
9:             carveVoxelsFromTo(v, d)
10:          end if
11:        end for
12:      end for
13:    end if
14:  end for
15: end function
```

spondent silhouettes. This approach enhances spherical shapes. Reconstruction of this kind of shape by a Visual Hull method usually presents a flat surface as it was chopped

Algorithm 3.4 Algorithm of CarveVoxelsFromTo method.

```
1: inputs: voxel v, direction d
2: function carveVoxelsFromTo(v, d)
3:   v = v.nextVoxelInDirection(d)
4:   while v.isValid() do
5:     v.visible = false
6:     v = v.nextVoxelInDirection(d)
7:   end while
8: end function
```



Voxel Carving Our approach

Figure 3.8. A comparison between two 3D models created by Voxel Carving, a standard Visual Hull based method, and ours. The highlighted area shows the couch's seats, which was ignored by the Voxel Carving algorithm.

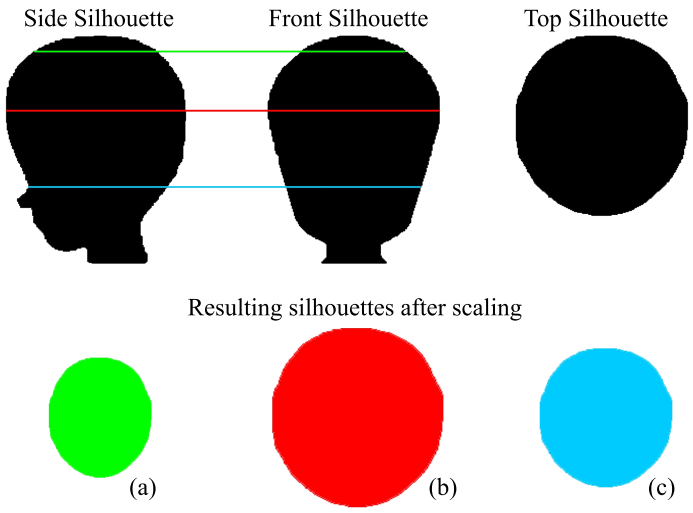


Figure 3.9. Illustration of as our scaling and scanning approach deals with the silhouettes. (a) is the resulting top silhouette when scaled at green row; (b) is the resulting top silhouette when scaled at red row; (c) is te resulting top silhouette when scaled at blue row. Those scaled silhouettes are applied to the model using an intersection operator.

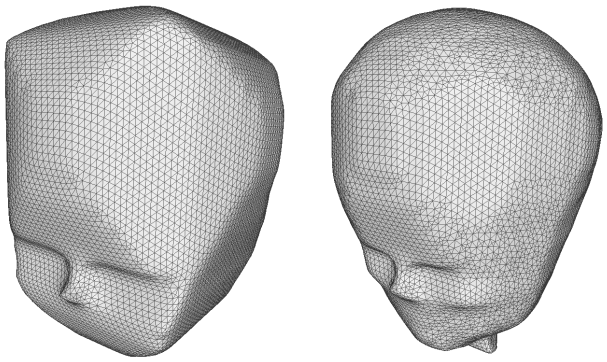
or sculpted. We perform this refinement similar to a scanning. For each slice of the grid of voxels, we resize the convex silhouette by the other two silhouettes boundaries at the row position. We traverse the grid treating it as being a set of layers. The resized silhouette is applied using an intersection operator onto the grid of voxels. We can verify how this approach enhances curved shapes in Figure 3.10. Algorithm 3.5 presents the pseudo-code and Figure 3.9 illustrates the scaling process. Notice that front and side view used in the Algorithm 3.5 as scaling view by considering the top view as convex. If the convex view was another rather than the top, those scaling views would be different. Also, a silhouette mask is an black and white image, where white is the object shape and black is the background.

3.1.4 Detail Refinement

Next, we improve the grid of voxels by smoothing the discretization implied by the voxel representation. The grid smoothness is accomplished by performing the Marching Intersects algorithm [Rocchini et al., 2001], which traverses the border on the concept art detecting intersections between vertices of a voxel. The Marching Intersects procedure detects the exact position of the edge intersection within a region of pixels and updates the vertex of that voxel to the intersection position. This approach generates

Algorithm 3.5 Algorithm for refinement by convex silhouettes.

```
1: inputs: the grid of voxels, the concept art silhouette masks
2: function convexRefinement(grid, front, side, top)
3:   for all viewpoint ∈ inputConceptArt do
4:     if viewpoint is convex then
5:       for row = 0 → grid.getSize() do
6:         mask = viewpoint
7:         xScale = front.widthAt(row) / mask.widthAt(row)
8:         yScale = side.widthAt(row) / mask.heightAt(row)
9:         mask = scale(mask, xScale, yScale)
10:        grid = intersectionOp(row, grid, mask)
11:      end for
12:    end if
13:  end for
14: end function
```



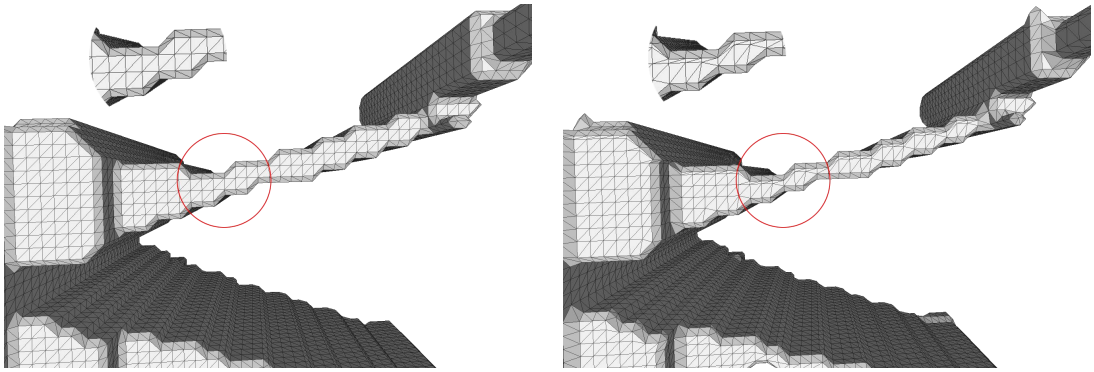
Before After

Figure 3.10. Comparison of the reconstructed surfaces without and with refinement by convex silhouettes.

the surface more accurate by taking into account the contours of the concept art and highlighting small details. As surface comparison is shown in Figure 3.11.

3.1.5 Mesh Generation

Since the texture coordinates are specified at each vertex of a given triangle, we first convert the representation of the grid of voxels to a triangular mesh by applying the Marching Cubes algorithm [Lorensen and Cline, 1987].



Without Marching Intersects With Marching Intersects

Figure 3.11. A surface comparison with before and after the Marching Intersects procedure.

3.1.6 Mesh Smoothing

In order to reduce the mesh jaggedness, we perform a vertex reposition based on Laplacian operator [Bardyn et al., 2010]

$$\triangle(v_i) = \frac{1}{|\mathcal{N}_1(v_i)|} \sum_{v_j \in \mathcal{N}_1(v_i)} (v_j - v_i),$$

where v_i is the vertex to be smoothed, $\mathcal{N}(v_i)$ is the local 1-ring neighborhood of a vertex v_i and vertices v_j are adjacent of v_i. It adjusts the vertices in relation to their neighbors by using a relaxation approach

$$v_i' = v_i + \triangle(v_i). \tag{3.7}$$

We take the average of the edges of neighbor vertices to obtain a smooth vertex position among their neighborhood.

3.2 Texture Generation

In the last step of an approach to completely generate a textured 3D model generation, we create a texture map based on the colors within the input pieces of the concept art and applies it to the surface of the object.

The texture is applied to the 3D model surface in two main steps: (1) creating a texture atlas, which binds a 3D surface to a 2D image and (2) generating a diffuse color map that stores the final color of each triangle of the mesh in an image. Figure 3.12 illustrates these two steps and the final textured 3D model.

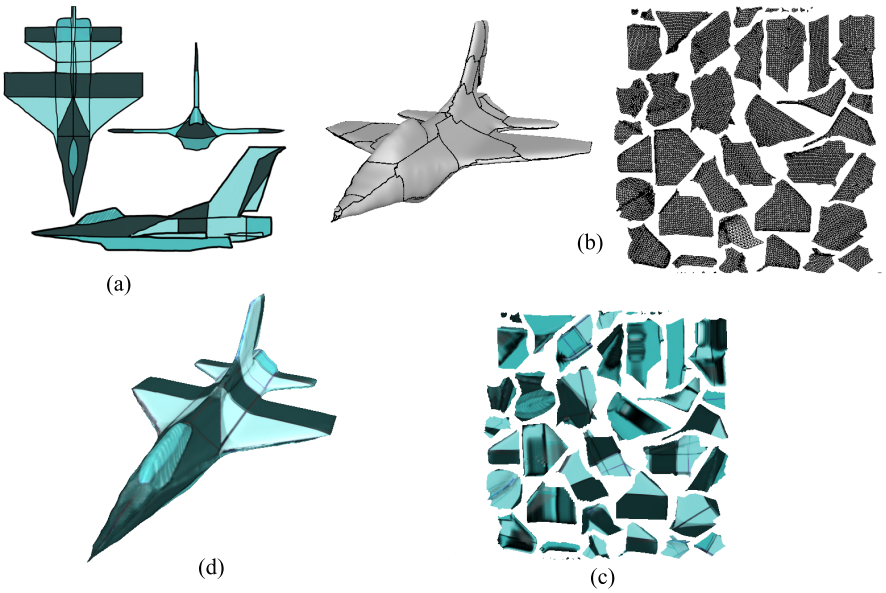


Figure 3.12. The procedure of texture generation starts mapping every triangle of the mesh onto a planar space, the texture atlas. By combining the image patches from concept art where the triangle projects, we compute the colors in the texture map. Image (a) shows an example of pieces of concept art of an airplane object used as input to create the 3D model and the texture atlas (b). The image used as the texture is the result of a diffuse color map generated upon the texture atlas concerning the input pieces of concept art (c). The final textured 3D model is presented in (d).

3.2.1 Texture Atlas

We iterate through all 3D model's triangles and rearrange them into clusters that project onto a plane. At the same time, we attempt to maintain as minimum distortion as possible. This task in a nutshell is a problem NP-Hard known as the bin packing problem. In the bin packing problem, objects of different volumes must be packed into a finite number of bins or containers each of volume V in a way that minimizes the number of bins used. Thus, the best exact solution is computationally exorbitantly expensive to computed. To determine these clusters, we use the work of Sander et al. [2001], which creates a texture atlas by using mesh progression. It starts with a base mesh and then creates a sequence of n refinements of the base mesh with operations called vertex splits. The result of the refinement is a group of similar meshes with different numbers of vertices. The final output is a global texture atlas, where the final mesh can be adjustable to a different level of details.

In the texture atlas creation, there are two entry parameters: the maximum permitted distortion and the maximum number of triangle clusters, each one affects

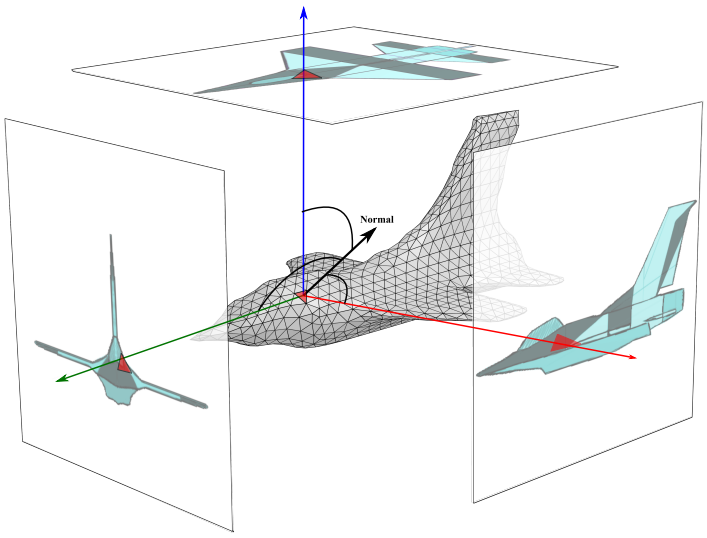


Figure 3.13. The diffuse map is created by iterating through all 3D model's triangles and warping the concept art patches to the triangle plane by using Homography. The final color is a linear combination of the concept art patches and the triangle normal.

the other. In the worst case, for example, $maxdistortion = 0$, every triangle could be placed on their own cluster. Figure 3.12 (b) shows an example of an airplane model showing its texture mapping seams. Each seam is a group of triangles. It is advisable to group triangles of similar material into the same seam.

3.2.2 Diffuse Map Creation

Once the texture atlas is built, our method assigns a texture coordinate (u, v) in the texture atlas for every vertex on the 3D mesh. It binds every vertex to a position within the texture image providing us the equality

$$texture_{x,y} = f(vertex_{u,v}), \tag{3.8}$$

where $texture_{x,y}$ are the pixel coordinates on a texture image, and $vertex_{u,v}$ are the vertex's texture coordinates. Thus, it is possible to calculate the color within a triangle by its three vertices. Similarly, we can project a vertex back to the concept art as well.

We define a patch as a group of pixels from a piece of concept art image. The patch P of a triangle T_i is the region within pixels of a piece of concept art in which T_i would lie when projected back. Notice that depending on the normal's triangle, the patches from the front, side and top concept art would have different areas as shown

Algorithm 3.6 Algorithm of diffuse texture map creation.

```
1: inputs: the mesh, and the set of concept art
2: function calcDiffuseMap(mesh, frontImg, sideImg, topImg)
3:    texture = new Texture()
4:    for all triangle ∈ mesh.triangles do
5:       patchWarped = new Patch[3]
6:       patchWarped[0] ← warp(frontImg, triangle.uv)
7:       patchWarped[1] ← warp(topImg, triangle.uv)
8:       patchWarped[2] ← warp(sideImg, triangle.uv)
9:       normal ← clamp(triangle.normal)
10:      texture ← linearComb(patchWarped, normal)
11:   end for
12:   return texture
13: end function
```

on Figure 3.13. In reason of that, we warp all patches to the triangle plane by using Homography, making the patches overlap themselves, as follows

$$\mathsf{P}_v^w = H(\mathsf{P}_v, \mathcal{V}), \tag{3.9}$$

where P_v is the original patch of a triangle T from the piece of concept art of some *view*, $\mathcal{V}$ are the vertices of the triangle T, and H is the Homography transformation from the concept art plane to the triangle's plane. Notice that, as a Homography system must have four points from each plane to be solved, a fourth coplanar vertex is virtually created to the triangle.

The final color of that triangle is defined as follows

$$color = \mathsf{P}_f^w \times cl(\overrightarrow{\mathbf{n}} \bullet \overrightarrow{\mathbf{z}}) + \mathsf{P}_s^w \times cl(\overrightarrow{\mathbf{n}} \bullet \overrightarrow{\mathbf{x}}) + \mathsf{P}_t^w \times cl(\overrightarrow{\mathbf{n}} \bullet \overrightarrow{\mathbf{y}}), \tag{3.10}$$

where $\mathsf{P}_f^w, \mathsf{P}_s^w, \mathsf{P}_t^w$ are the warped patches from front, side and top images, $\overrightarrow{\mathbf{n}}$ is the triangle's normal and $\overrightarrow{\mathbf{z}}, \overrightarrow{\mathbf{x}}, \overrightarrow{\mathbf{y}}$ are normalized vectors from z, x and y axes. The function cl is a clamp function to force the dot product to be in a range between 0 and 1 and it maintains the three dot products normalized. It prevents the final color of being under or super estimate. We perform this for all triangles adding them into the final texture as shown on Algorithm 3.6. Also, we provide the possibility of mirroring the right concept art when the triangle's normal points to the left direction. This showed to be a fine approach for symmetric objects.

In Figure 3.12, we can see the results from each step using three pieces of concept art of an airplane object (a). The texture atlas and its projection on the 3D model

(b). The diffuse color map calculated onto the texture atlas (c). The final textured 3D model (d).

3.3 User Interface Application

For the sake of demonstration, we developed a user interface application that makes use of our method. It allows the user to test interactively our methodology (Figure 3.14). The user interface application consists mainly of three 2D drawable widgets for drawing the front, side, and top concept art of the object, and a 3D scene widget in which the generated 3D model is presented. Furthermore, there is a configuration panel in the interface's right with tools and tweaks to manage both 2D and 3D widgets such as brush color, brush size, 3D scene background color and so on. More detail regarding the configuration panel and a user case are presented in Appendix A.

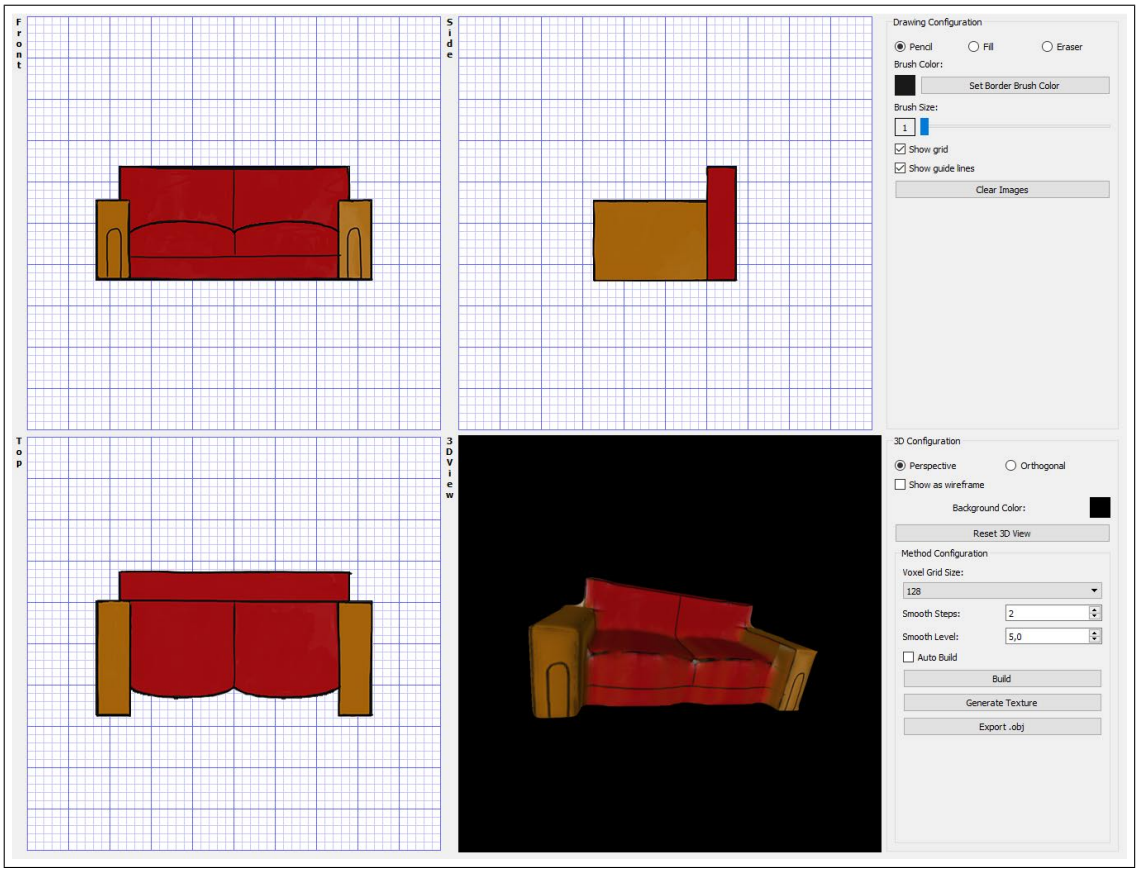


Figure 3.14. A user interface application that allows the user work interactively with our methodology. It is composed by four widgets, three are 2D drawable widgets from drawing the front, side, and top concept art as well as a 3D widget in which the generated 3D model is presented.

Chapter 4

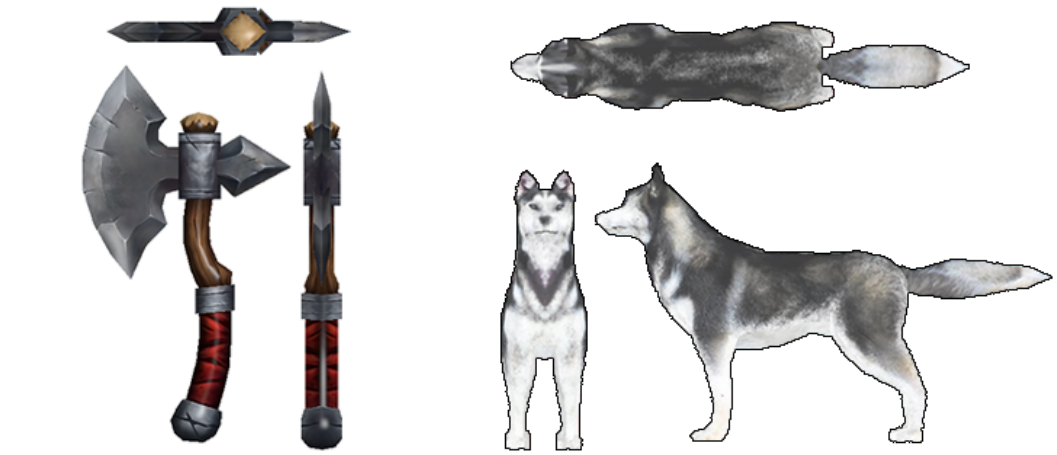
Experiments

In this chapter, we present our experiments to provide both quantitative and qualitative evaluation. We performed experiments using synthetic concept art for surface reconstruction accuracy assessment and compare our geometric consistency against a photo-consistency based method, both to assess the quality our 3D surface modeling method. Additionally, we assess the processing time and perform an analysis of the trade-off between the processing time and surface reconstruction accuracy by changing the resolution of the grid of voxel. Also, we performed an experiment with 3D artists to assess the applicability our method. As our goal is speed up the artistic pipeline by giving the coarse model at the beginning of the 3D content creation process. We close our experiments with a qualitative assessment using several pieces of concept art.

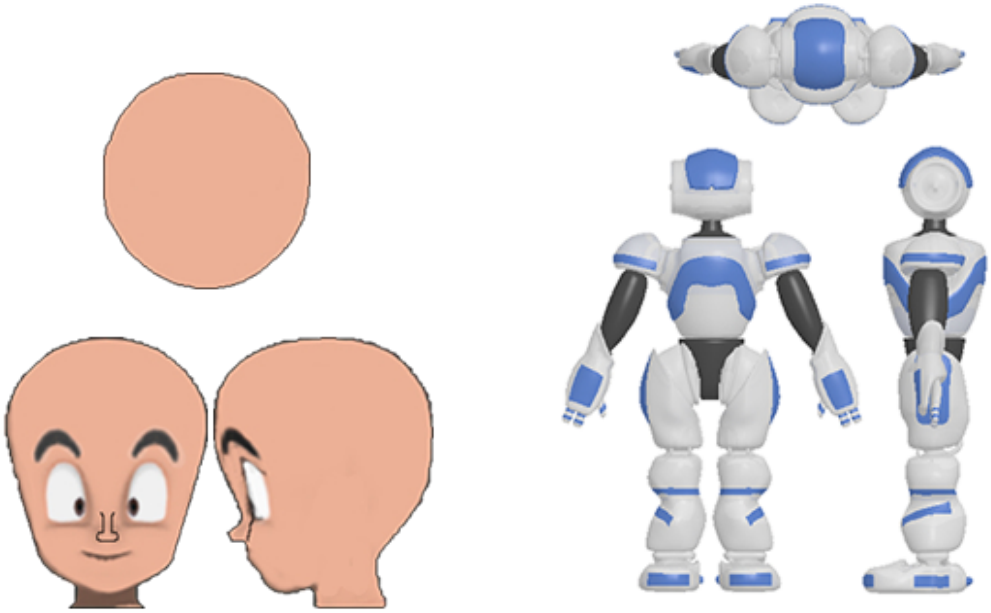
The synthetic pieces of concept art are images that were extracted from 3D models made by a 3D artist and rendered following the concept art pattern: three views captured by a camera of orthogonal projection. We reference them as base 3D models and the three views of each model used in our experiments are shown in Figure 4.1. The pieces of concept art were hand-drawings created on a graphical software by a 2D artist. All of our experiments were performed using images with a dimension of 512×512 and grid of voxels varying from 32 to 256 voxels of grid resolution.

4.1 Quantitative Analysis

In our quantitative analysis, we compare the accuracy of the generated surfaces by our methodology to another silhouette-based method. Thanks to the original mesh of the model, we could measure the surface distance error in generic units along a variation of the grid of voxels dimension. To verify that the geometric consistency



Axe model Dog model



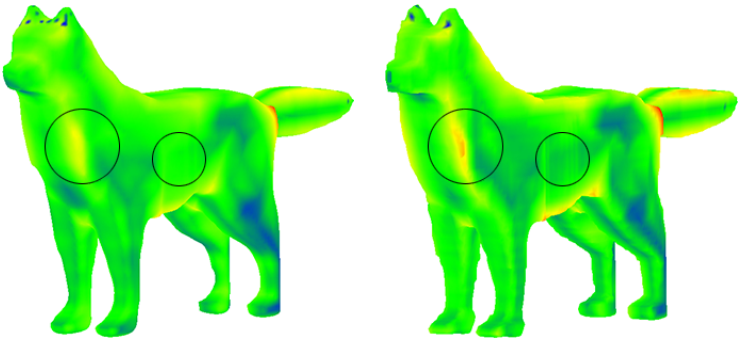
Head model Robot model

Figure 4.1. The images used in the quantitative experiments. We created three images from four 3D models using orthogonal projection simulating concept art drawings.

contributes to the success of our approach, we also compare to a methodology based on photo-consistency.

4.1.1 Surface Reconstruction Accuracy

In the surface reconstruction experiment, we evaluate the accuracy of the reconstruction provided by our method and the Voxel Carving algorithm, the Voxel Carving source code is the original one provided by the authors in their website. We used four 3D



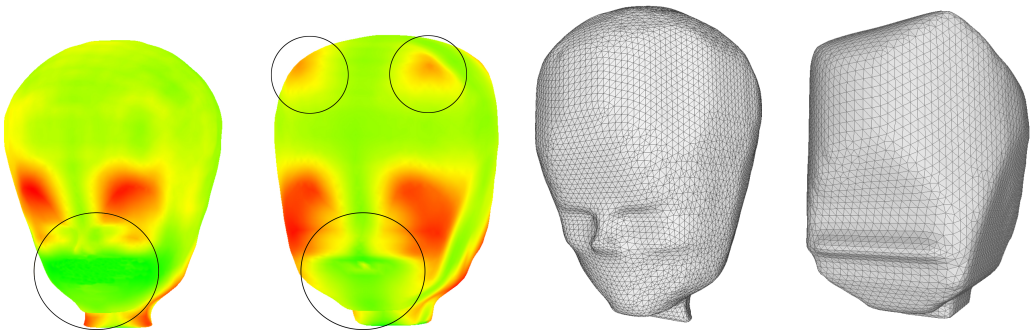
Ours Voxel Carving

Figure 4.2. Comparison of the reconstructed surfaces with the base 3D model. The heat maps show the surface distances. The color gamut varies from greener for closer distance to the red for farther one. The highlighted areas show that our method estimated better accurate surface compared to the Voxel Carving algorithm.

models of different objects: an axe, a dog model, a human head model and a robot model. Figure 4.1 shows the three viewpoints of each model used in the evaluation. We created three images from four 3D models using orthogonal projection simulating concept art drawings. The head model is one particularly challenging object to be modeled due to the occlusion of the nose by the forehead on the top view.

We used a software called CloudCompare to assess the surface distances. CloudCompare's default way to compute distances is using the nearest neighbor distance, for each vertex of the compared model, CloudCompare searches the nearest triangle in the reference model and computes their euclidean distance.

Figures 4.2 and 4.3 show the heat map of surface distances of the dog and head. The color gamut varies from green for close distance to yellow for farther distance and



Ours Voxel Carving Ours Voxel Carving

Figure 4.3. Surface estimation for the head model. As can be seen in the heat maps (left) and the meshes (right), our method was capable of modeling the nose and eyebrows regions, while Voxel Carving was not.

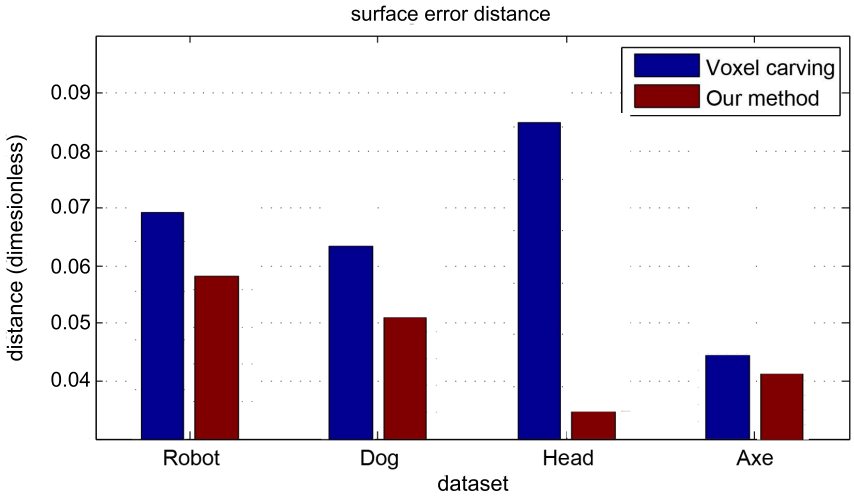


Figure 4.4. Surface reconstruction accuracy. The bars show the surface error distance of the models estimated by the methods compared to the 3D base model.

the red for the farthest one. Yellow means that the generated surface has inflated and the blue means that the surface has shrunk. We draw the following observations. First, our method outperforms the Voxel Carving even in its best case, where there is no concave part within the silhouette (dog model). Second, as expected, the geometric consistency detects and models concave parts as can be checked around the nose in the head model. It is worth noting that a photo-consistency based method would not able to model these regions due to the lack of texture on the input images.

It can be seen in Figure 4.4 that in all cases, our proposed method leads the performance. This means that the surface estimated by our method is the closest to the base 3D models. As far as the reconstruction accuracy is concerned, our method presents a significant improvement compared to Voxel Carving.

4.1.2 Geometric Consistency vs Photo-consistency

Techniques such as Space Carving [Kutulakos and Seitz, 2000] label the voxels as opaque if they pass the photo-consistency test. This test matches the pixel intensity space to decide whether it should continue carving. Different from geometric consistency, the photo-consistency test relies on the pixel appearance. Therefore, the model quality estimated by a method based on photo-consistency is a function of the image discriminant information of object shape. The texture is a crucial feature that must be present in the image. In general, concept art does not present rich intensity patterns and thereby does not have high discriminant texture information. However, by comparing our methodology with the Clustering Views for Multi-view Stereo (CMVS) [Fu-

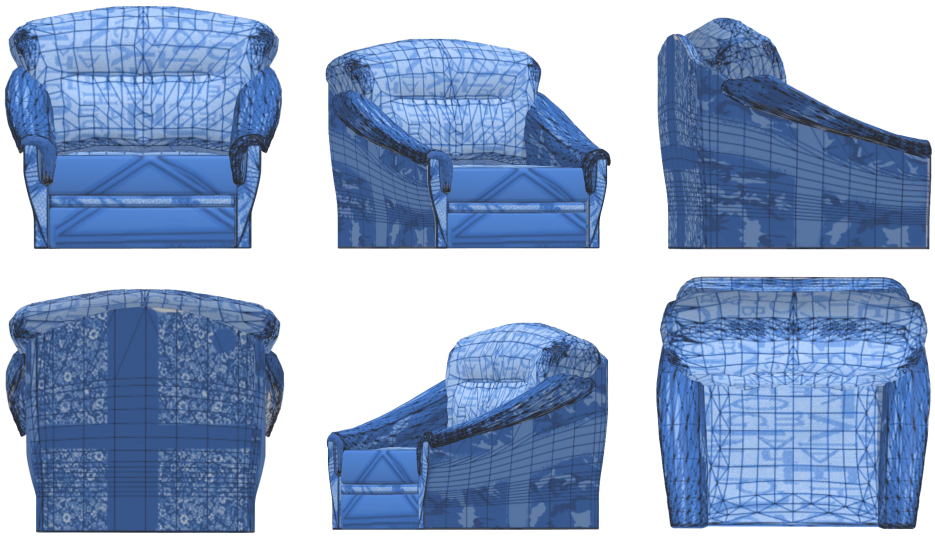


Figure 4.5. Six images from a textured 3D model generated to be used in the CMVS methodology. The whole set is composed of one image from a top view and the others 36 are from the ground level rotating the model 10 degrees.

rukawa and Ponce, 2010], a sophisticated methodology for 3D reconstruction that uses photo-consistency, we show the benefits of our methodology in a textureless image. We have used an executable provided by the authors in their website.

Although our methodology can work with only three images, this showed not to be possible for CMVS. To generate 3D models using CMVS, we used 37 images, where 36 images were rendered from the ground level view with an offset of 10 degrees among them and one from the top view. In addition, we also included a texture map to the 3D model with high level of discrimination aiding CMVS in performing the photo-consistency check. Figure 4.5 shows six views from the set of 37 images used in the experiment. We guarantee the best scenario for a photo-consistency approach by using these images.

To assess the reconstructed surfaces, we used the same heat map metric presented in Section 4.1.1. One can clearly see in Figure 4.6 that our model outperforms CMVS models for both scenarios, textureless, and texture-rich images. It presents a greener shading than the CMVS models. Even though it was added enough viewpoints to CMVS, the CMVS textureless erroneous result is totally expected as discriminant texture information is absent in the input images, which is the main feature that CMVS relies on. On the other hand, CMVS with rich texture information would have a fair result if it was compared to surfaces generated by Image-Based Reconstruction Methods. However, regarding the drawing context, which is our focus, our method delivers a better reconstruction quality. Quantitative values are presented on Table 4.1.

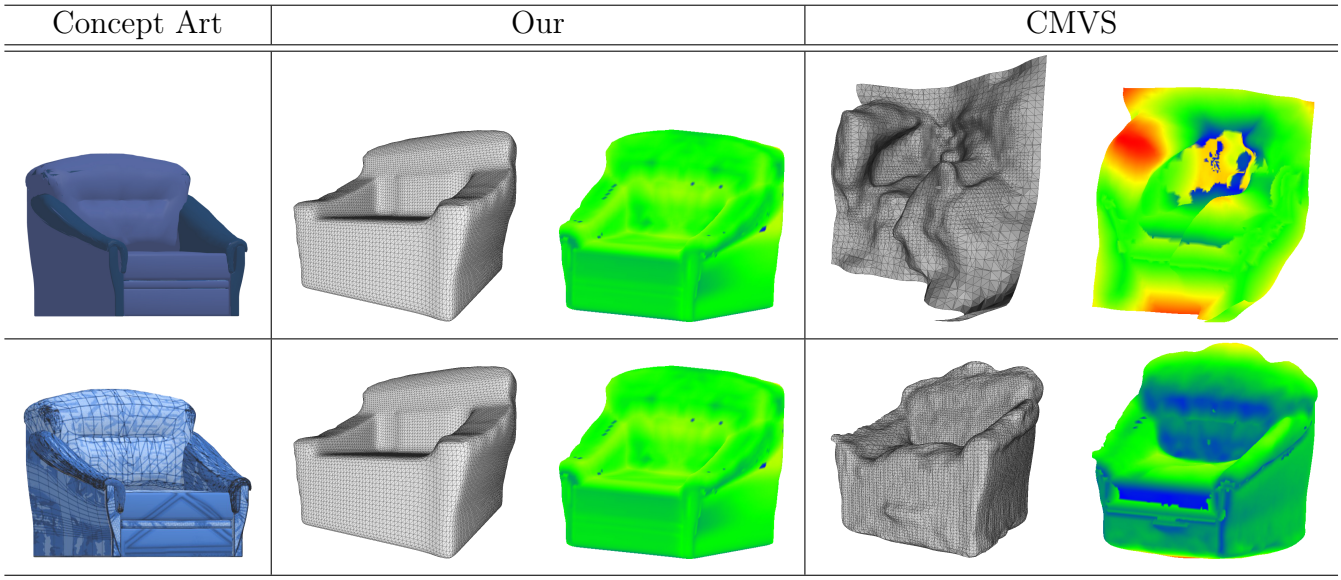


Figure 4.6. Heat map and mesh of the surfaces generated by our methodology and CMVS approach using textureless (first row) and texture images (second row). It assesses surface distance from a base model, where the color gamut varies from greener for closer distance to the red for farther one.

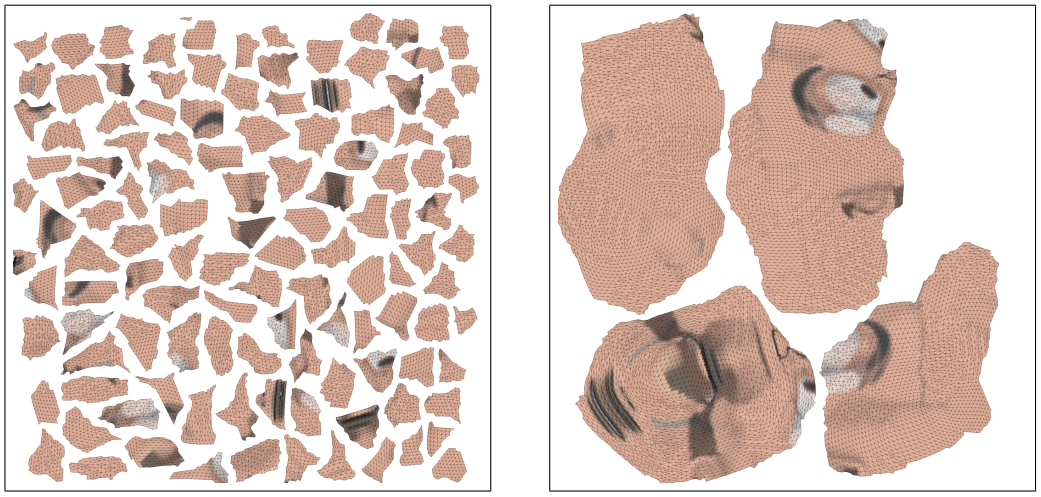
Method	Error distance (dimensionless)
Ours	**0.031**
CMVS	0.036

Table 4.1. The surface distance between the groundtruth mesh, our method (geometric consistency) and CMVS (photo-consistency). Small distances represent a more accurate model.

4.1.3 Speed Up Assessment

In this experiment, we performed an analysis to assess quantitatively how much our approach can speed up the 3D content creation. In order to evaluate that, we took four experienced 3D artists, their experience ranging from 5 to 9 years, and gave to them concept art of objects and their respective models generated by our methodology. We asked them to create two ready-to-use models of each object, one modeling from scratch and another starting from our coarse model and recording how long they took to create them. In addition, they took a survey where they expressed their thoughts about the methodology.

The table 4.2 presents the recorded times for our experiment. The times are the average of the four 3D artists time, Appendix B presents all times separately. Notice that our methodology expressively accelerated the modeling step, for example, the X-Wing modeling time was cut almost in a half. Also, The overall 3D artists' work



Max cluster: 120, max distortion: 0.02 Max cluster: 1, max distortion: 0.05

Figure 4.7. The difference between texture atlas after changing the entry parameters of texture atlas generation. The texture atlas generated by decreasing the max cluster parameter and increasing the max distortion one offered a more workable map to the artist work with.

was sped up in an average of 14%. Nonetheless, our methodology could not speed up the texture map step. The artists pointed out that the generated texture atlas were not wisely divided creating a great quantity of clusters and that has overloaded their work on the step of texture generation. Though, this issue was not enough to overload their whole process. Moreover, as we mentioned in section 3.2, max distortion and max number of clusters are entry parameters of the texture atlas generation. After the survey feedback, we changed the entry parameters giving more importance to the cluster number than distortion, Figure 4.7 shows the difference between the texture atlas. A comparison between a two models generated in the experiment, one modeled from scratch and another from our methodology, is shown in Figure 4.8

In the survey, they were asked to answer some qualitative questions about their feeling towards the methodology. One of the questions, the complete questionnaire with the average values follows below, was how much they would be open to adopting the methodology in their work pipeline, varying from 0 (no open at all) to 5 (totally open). The opening to our methodology had an average of 4.3, which can be interpreted as great inclination in adopting our methodology in their pipeline. Another question was, varying from 0 (no effort at all) to 5 (prohibitive effort), how much effort to start modeling from a coarse model was. It had an average of 2.3, which is lightly tended to lessen their effort. They stated that the main effort point was a topology issue, sometimes the mesh loops were not optimal to work with. Despite this issue, they claimed that beginning with a coarse model is faster than from scratch as we

Model	Dog		Head		X-Wing	
	from scratch	coarse given	from scratch	coarse given	from scratch	coarse given
Modeling Time (hh:mm)	2:55	**2:18**	3:15	**2:10**	2:45	**1:40**
Modeling Speed up	21%		33%		40%	
Texturing Time (hh:mm)	2:29	**2:13**	**2:15**	2:40	**1:45**	2:32
Texturing Speed up	11%		-18%		-44%	
Total Time (hh:mm)	5:24	**4:30**	5:30	**4:50**	4:30	**4:12**
Speed up	20%		13%		7%	

Table 4.2. Speed Up Assessment experiment assessed how much our technique sped up the 3D content creation work. The table presents the average times spent by the four 3D artists to create both models, coarse-given (our methodology) and from scratch.

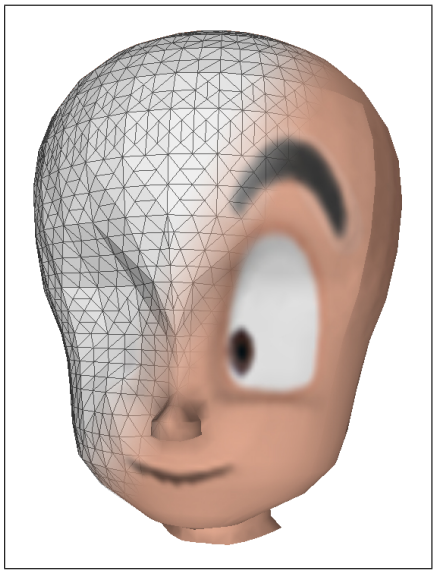
can clearly verify in the table 4.2 as well. In addition, one of the artists, who was an assiduous user of digital sculpting tool, added that the coarse model is useful as a base geometry for digital sculpting.

Qualitative questionnaire:

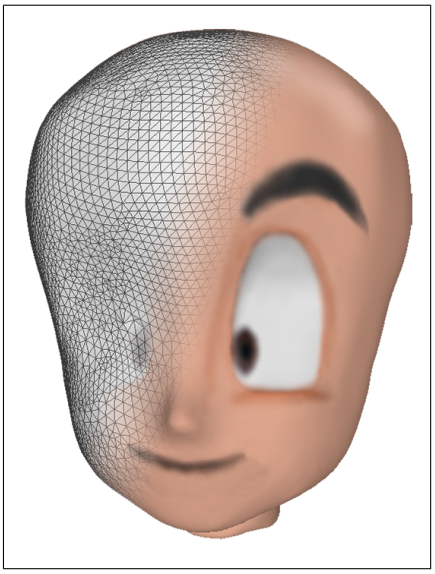
- From 0 (No open at all) to 5 (Totally open), how much would you be open to use our methodology in your workflow? *4.3*
- From 0 (No effort at all) to 5 (Prohibitive effort), how difficult was it to adequate our model to the final one? *2.3*
- From 0 (No effort at all) to 5 (Prohibitive effort), how difficult was it to adequate our texture to the final one? *2.75*
- From 0 (Nothing at all) to 5 (Really improved), how much has our method eased your work? *3.75*

4.1.4 Processing Time

On average, 3D artists spend several hours to create textured 3D models such as the models are shown in Figure 4.11, as we can see on Table 4.2. In Table 4.3, we present the processing time spent by our methodology in each step of the process for a grid



Head model from scratch (total time 5:30)



Head model coarse-given (total time 4:50)

Figure 4.8. The results of the head concept art after an artist has worked on it. From scratch model was completely modeled by an artist and was taken five and a half hours. Whereas coarse-given (our methodology), the artist modeled from our methodology output and spent only four hours and fifty minutes. It was a speed up of 13%.

	Volume	Mesh	Texture	Total
Wagon	1.90	1.21	973.66	976.76
Couch	1.86	0.80	631.36	634.01
Spaceship	1.33	0.98	1131.64	1133.95
Head	1.64	0.81	587.73	590.18
Robot	1.40	0.99	883.88	886.28
Axe	1.48	0.96	772.54	774.99
Dog	1.52	1.61	681.96	685.09
Average	1.59	1.05	808.96	811.61

Table 4.3. Processing time (in seconds) for each step using a grid of 128 voxels.

of dimensions 128. We used a machine with core i7-4510U CPU @ 2.00GHz processor and 8 GB of RAM for processing time computation.

Despite a long time on texture generation, the resulting time presents a significant speed up on 3D model generation. The modeling process only spends 2 seconds on average, which is quite fast, since a 3D artist can spend up to eight hours on similar models of those used in the experiments. The texture generation had such variation in time due to the process is dependent on the number of triangles.

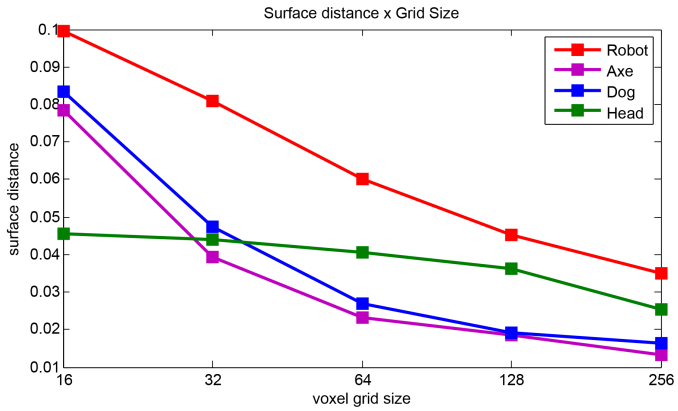


Figure 4.9. The influence of grid size in the surface modeling accuracy. A bigger grid provides a more precise surface.

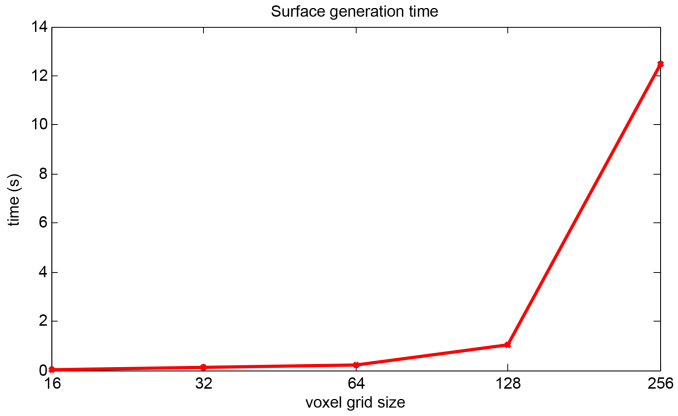


Figure 4.10. Processing time versus the grid size. Although the process time grows exponentially, a grid with 128 of size can provide high accuracy models in a small processing time.

4.1.5 Grid Size and Reconstruction Accuracy

For the four synthetic models, we tested the influence of grid size in the modeling accuracy. We can see in Figure 4.9 that model surface accuracy raises as the grid of voxels increases. It is a logical effect as the greater the voxel grid resolution, the smaller is the pixel patch that each voxel represent within the images, therefore the volume can hold more details. On the other hand, we see in Figure 4.10 that the processing time grows polynomially as the grid of voxels increases. It is due to $\Omega(n^3)$ (n is the grid size) for the voxel grid traveling as we must go through in all 3 axis (X, Y, Z). In the worst case, where geometric consistency is highly processed, it can hypothetically turn to $\Theta(n^4)$. We found that a grid of dimension 128, which is one-quarter of the concept art's resolution, is the best trade-off between accuracy and processing time.

Figure 4.11. The resulting 3D models generated by the synthetic concept art.

4.2 Qualitative Assessment

Our qualitative assessment was performed to evaluate the quality of 3D models created by using concept art. We chose several objects with different shapes from complex surfaces such as the concept art of a spaceship to others with concave parts such as the concept art of a couch. Figures from 4.12 to 4.20 present the set of concept art used as input and the resulting 3D model of our methodology. Figure 4.11 presents the results from the synthetic pieces of concept art from the quantitative experiment 4.1.

In Figure 4.12, we can see the result of our methodology for an airplane model and the input concept art. The airplane structure shows to be an ideal surface for being represented in those three viewpoints as its silhouette does not present any self-occlusion.

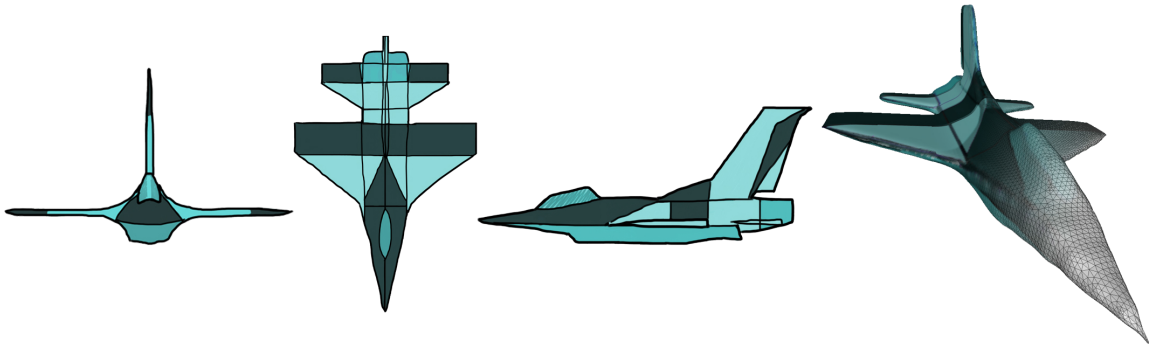


Figure 4.12. Airplane concept art and model.

In Figure 4.13 we can verify, as aforementioned in Chapter 3, how our geometric consistency performs an important role in the right surface estimation, even though the seats of the couch is self-occluded in the right viewpoint.

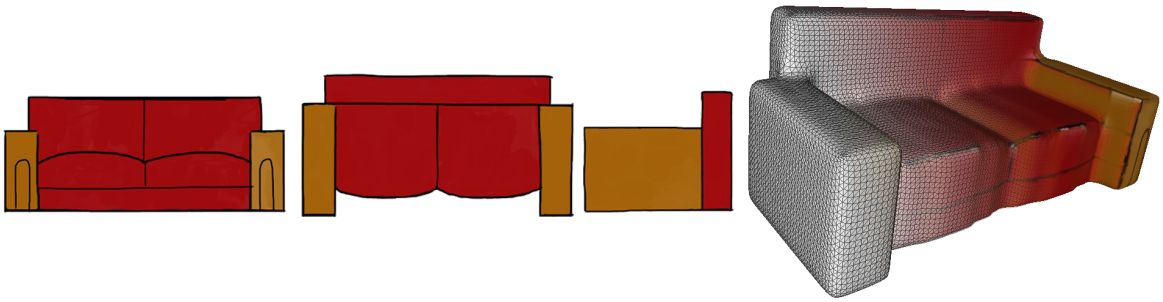


Figure 4.13. Couch concept art and model.

In Figure 4.14, the wagon concept art presents a self-occlusion problem similar to the couch model. However, our methodology was able to estimate the right surface successfully by relying in our geometric consistency.

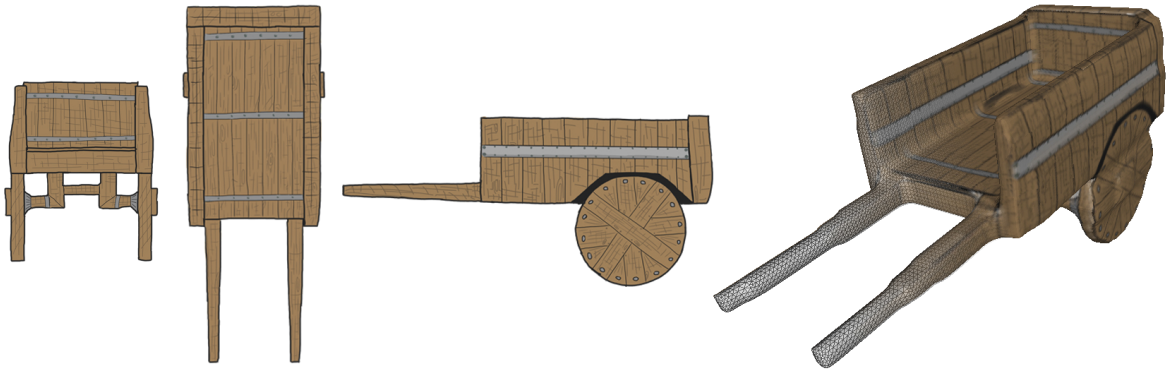


Figure 4.14. Wagon concept art and model.

In Figure 4.15, we can see pieces of concept art illustrating an X-Wing spaceship, which is a complex object with small details and not so obvious surface. We can readily verify that our methodology created a pretty fair 3D model representation with the iconic fuselage painting.

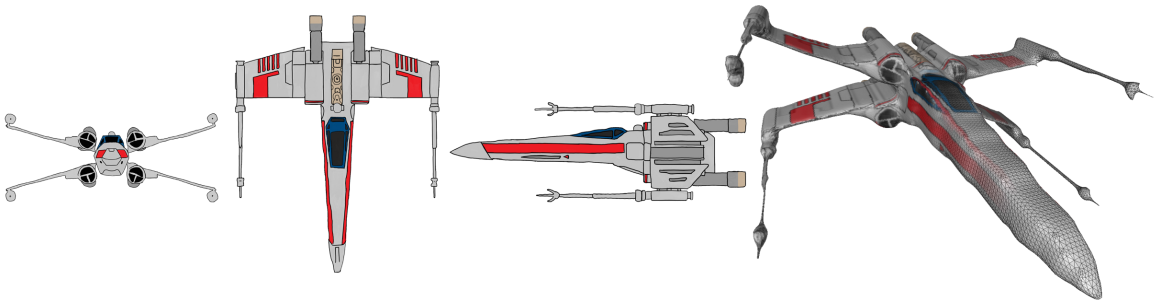


Figure 4.15. X-Wing concept art and model.

In Figure 4.16, we have a representation of a regular car, a simple surface that was successfully modeled by our methodology. Our methodology assertively created the model surface and its texture map.

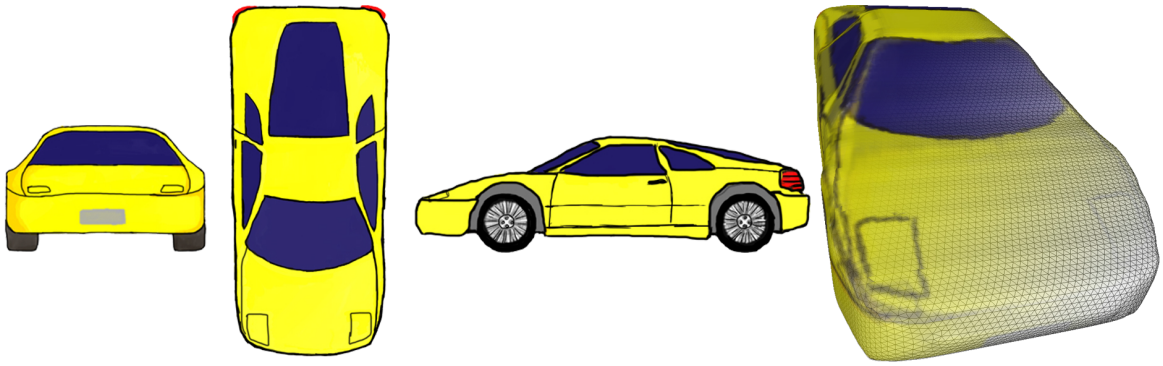


Figure 4.16. Car concept art and model.

In Figure 4.17, the pieces of concept art present a handgun, a very ordinary item in nowadays video games. The resulting model present quality to be an effortlessly created asset used for prototyping.

Figure 4.17. Handgun concept art and model.

In Figure 4.18, it is presented a dragon, another complex object with a challenging surface. The final 3D model expressively presents the surface and colors on the input images.

Figure 4.18. Dragon concept art and model.

In Figure 4.19, the jet object surface follows closely the airplane from Figure 4.12, which presents a favorable scenario for silhouette-based modeling approaches. Thus, as the airplane result, our methodology created the model in the concept art without any further problem.

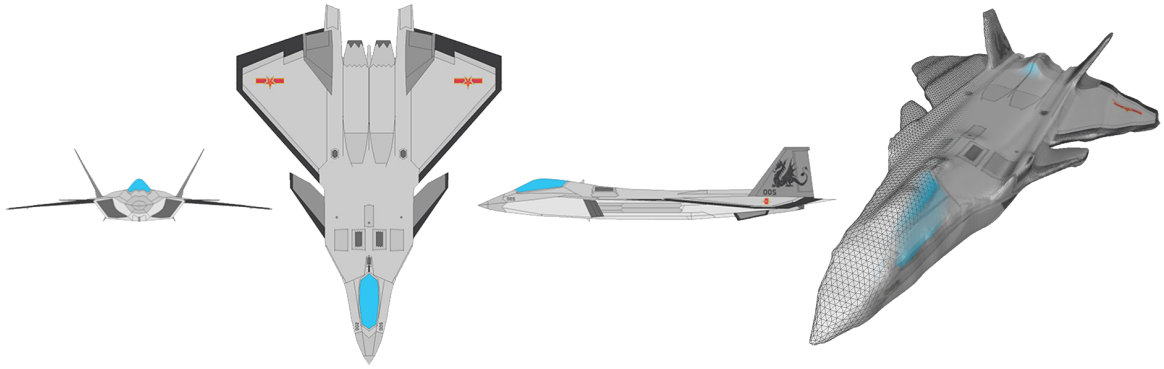


Figure 4.19. Jet concept art and model.

In Figure 4.20, the arm chair object presents the same problem of the couch from Figure 4.13. Nonetheless, the arm chair self-occlusion problem is successfully overcame by our geometric consistency as well.

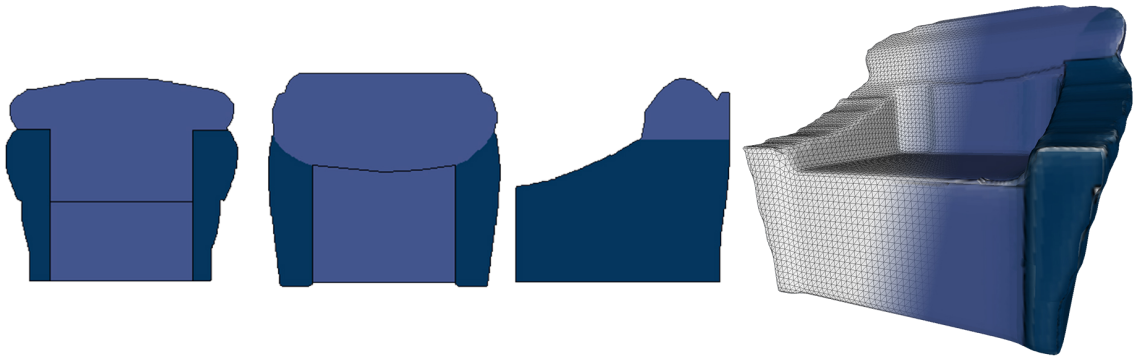


Figure 4.20. Armchair concept art and model.

Chapter 5

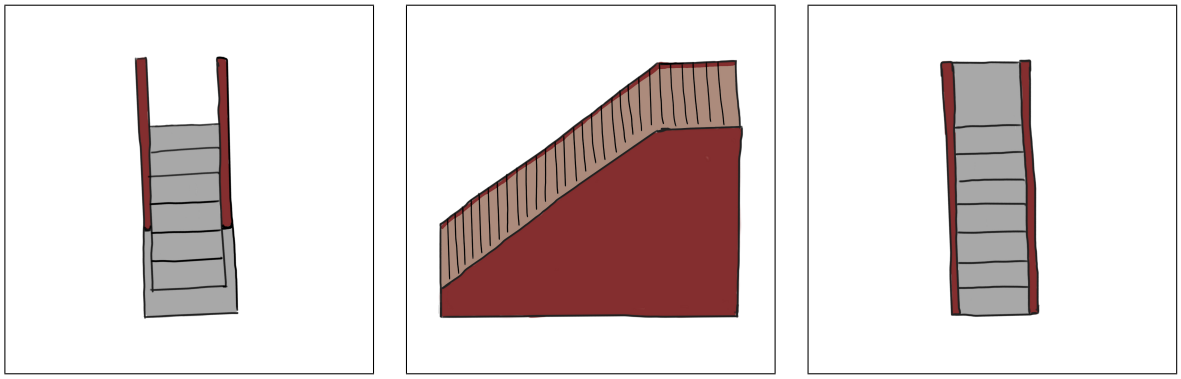
Conclusion

In this work, we proposed a fully automatic methodology to create textured 3D models from 2D drawings, which focuses on speeding up the early stages of 3D content creation. Our approach improves the shape estimation of volumes based on new constraints, which performs the consistency of the model's geometrical features by considering the concept art. In addition, we presented a method to estimate intensities to generate a diffuse texture map automatically based on the colors presented within concept art.

Our experiments confirmed our insight that granting a coarse model to the 3D artists increases productivity in developing 3D content for the entertainment industry. We were able to speed up the whole productivity in an average of 14%, while maintaining the artistic work pipeline. The modeling step had an impressive speed up of 31% itself. Thus, having room for improvement in the texture step, which at first were not so satisfying, we could increase the 14% speed up around three times if texture step followed the modeling improvement.

Our method presented a better performance in estimation and treatment of concave surfaces compared to methodologies based on photo-consistency check when working with pieces of concept art. It held a 14% better accuracy in the best scenario for photo-consistency methods, images rich in texture. Motivated by those results, our geometric consistency approach is worth to be extended for being able to check inconsistency on any viewpoint rather than only those canonical ones covered in this work. Hence, we could bring it to other areas such as 3D reconstruction by images in Computer Vision. Applying our geometric consistency in synergy with photo-consistency could improve the reconstruction quality.

Even though our generated texture was not the strongest point of our method, one can readily see that the diffuse color map is a reasonable starting point for a refinement performed by an artist.



Front concept art Side concept art Top concept art

Figure 5.1. An illustrated example of the several levels of depth limitation. The stair' steps are occluded on side concept art and our Geometric Consistency is not able to estimate all the concave parts by using only front and top concept art.

5.1 Limitations

Despite the proposed geometric consistency working properly for two level concave parts (it is well defined what is foreground and what is background), it cannot infer when there are several levels of concave parts present on, e.g. stair' steps occluded on side concept art, our geometric consistency would not be able to estimate all the concave parts by using only front and top concept, this example is shown Figure 5.1. Further, the matching of edges amongst the viewpoints is crucial to geometric consistency work properly. However, it is not as critical as pixel precision. Our algorithm is robust enough to treat some pixel misalignment provided that the edge pixels project in the same voxel. Also, some textures show a ghosting effect (Figure 5.2), which is caused by the blending process of the patches extracted from the concept art. Because of the normal vector of the triangle, the contribution may be wrong.

5.2 Future Work

As future work, we plan to improve the robustness of the Geometric Consistency. As pointed in section 5.1, it can only resolve for two depth levels with the current constraints. Probably, new constraints could be applied to enable more level solving. Modeling it as a constraint satisfaction problem (CSP) is an available option as well. CSP is approach of intense research in artificial intelligence, where the problem is defined as a set of objects with states that must satisfy a finite number of constraints or limitations.

Additionally, we plan to present an alternative method for calculating texture

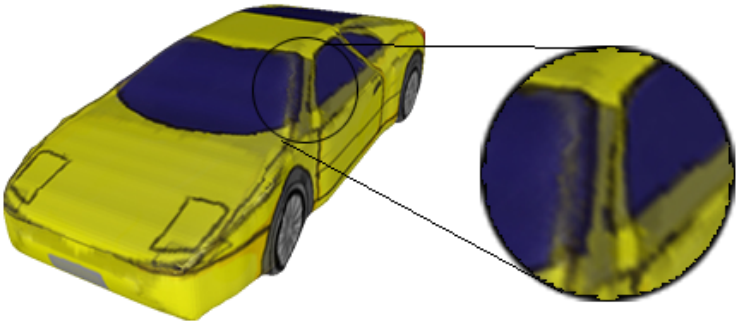


Figure 5.2. One drawback of our approach occurs in the texture generation. Due to the warping and blending that we perform to the input concept art for the diffuse color estimation, it presents a ghosting effect depending on the surface angles.

mapping in order to improve their execution time as Table 4.3 shows that this step is a bottleneck for the whole process. Currently, it is modeled as a bin packing, a np-hard problem. Some further investigation must be carried in good heuristics for the bin packing problem or modeling it differently.

Moreover, we intend to propose a better blending process considering other geometrical and image features for the texture generation process such as the triangle neighborhood. It could tackle with ghosting effect (Figure 5.2).

Finally, in regard to enhancement, there is room to propose a smarter approach for mesh loops generation, as it was one of the issues pointed by the artists in our survey. In the current work, we only prevent the mesh of having Ngons, forcing it to be completely composed by triangles. Ngons in 3D modeling is the effect of a mesh having a polygon with more than 4 vertices. Also, our convex silhouette refinement to enhance rounded shapes could be extended to use the surface shading to detected and smooth rounded shapes.

Bibliography

Ahearn, L. (2001). Budgeting and scheduling your game. *http://www.gamasutra.com/view/feature/3083/budgeting_and_scheduling_your_game.php*.

Andre, A. and Saito, S. (2011). Single-view Sketch Based Modeling. In *Proceedings of the Eighth Eurographics Symposium on Sketch-Based Interfaces and Modeling*, SBIM '11, pages 133--140, New York, NY, USA. ACM.

Bae, S.-H., Balakrishnan, R., and Singh, K. (2009). Everybodylovessketch: 3d sketching for a broader audience. In *Proceedings of the 22Nd Annual ACM Symposium on User Interface Software and Technology*, UIST '09, pages 59--68, New York, NY, USA. ACM.

Bardyn, T., Reyes, M., Larrea, X., and Büchler, P. (2010). Influence of smoothing on voxel-based mesh accuracy in micro-finite element. In *Computational biomechanics for medicine*, pages 85--93. Springer.

Brazil, E. V., Amorim, R., Sousa, M. C., Velho, L., and de Figueiredo, L. H. (2015). Sketch-based modeling and adaptive meshes. *Computer & Graphics*, 52(C):116--128. ISSN 0097-8493.

Buchanan, P., Mukundan, R., and Doggett, M. (2013). Automatic single-view character model reconstruction. In *SBIM*, SBIM '13, pages 5--14, New York, NY, USA. ACM.

Chen, T., Zhu, Z., Shamir, A., Hu, S.-M., and Cohen-Or, D. (2013). 3Sweepp: Extracting Editable Objects from a Single Photo. *ACM Transactions on Graphics*, 32(6):195:1--195:10. ISSN 0730-0301.

Cohen, J. M., Markosian, L., Zeleznik, R. C., Hughes, J. F., and Barzel, R. (1999). An interface for sketching 3d curves. In *Proceedings of the 1999 Symposium on Interactive 3D Graphics*, I3D '99, pages 17--21, New York, NY, USA. ACM.

De Paoli, C. and Singh, K. (2015). SecondSkin: Sketch-based Construction of Layered 3D Models. *ACM Trans. Graph.*, 34(4):126:1--126:10. ISSN 0730-0301.

Entem, E., Barthe, L., Cani, M.-P., Cordier, F., and van de Panne, M. (2015). Modeling 3d animals from a side-view sketch. *Computer & Graphics*, 46:221--230.

Furukawa, Y. and Ponce, J. (2010). Accurate, dense, and robust multiview stereopsis. volume 32, pages 1362--1376, Washington, DC, USA. IEEE Computer Society. ISSN 0162-8828.

Igarashi, T., Matsuoka, S., and Tanaka, H. (1999). Teddy: A sketching interface for 3d freeform design. In *International Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '99, pages 409--416, New York, NY, USA. ACM Press/Addison-Wesley Publishing Co.

Kutulakos, K. N. and Seitz, S. M. (2000). A Theory of Shape by Space Carving. *Internationl Journal of Computer Vision*, 38(3):199--218. ISSN 0920-5691.

Laurentini, A. (1994). The visual hull concept for silhouette-based image understanding. *PAMI*, 16(2):150--162. ISSN 0162-8828.

Levi, Z. and Gotsman, C. (2013). ArtiSketch: A System for Articulated Sketch Modeling. *Computer Graphics Forum*. ISSN 1467-8659.

Lorensen, W. E. and Cline, H. E. (1987). Marching cubes: A high resolution 3d surface construction algorithm. In *SIGGRAPH*, SIGGRAPH '87, pages 163--169, New York, NY, USA. ACM.

Maillot, J., Yahia, H., and Verroust, A. (1993). Interactive texture mapping. In *International Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '93, pages 27--34. ACM.

Matsuyama, T., Wu, X., Takai, T., and Nobuhara, S. (2004). Real-time 3d shape reconstruction, dynamic 3d mesh deformation, and high fidelity visualization for 3d video. *Computer Vision and Image Understanding*, 96(3):393--434. ISSN 1077-3142.

Oswald, M. R. (2015). *Convex Variational Methods for Single-View and Space-Time Multi-View Reconstruction*. PhD thesis, Technische Universität München, Germany.

Rivers, A., Durand, F., and Igarashi, T. (2010). 3D Modeling with Silhouettes. *ACM Transactions on Graphics*, 29(4):109--109. ISSN 0730-0301.

Rocchini, C., Cignoni, P., Ganovelli, F., Montani, C., Pingi, P., and Scopigno, R. (2001). Marching Intersections: an Efficient Resampling Algorithm for Surface Management. In *SMI*, pages 296--305.

Sander, P. V., Snyder, J., Gortler, S. J., and Hoppe, H. (2001). Texture mapping progressive meshes. In *International Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '01, pages 409--416, New York, NY, USA. ACM.

Seitz, S. M., Curless, B., Diebel, J., Scharstein, D., and Szeliski, R. (2006). A comparison and evaluation of multi-view stereo reconstruction algorithms. In *IEEE Conf. on Comp. Vision and Pattern Recog.*, volume 1, pages 519–528. ISSN 1063-6919.

Seitz, S. M. and Dyer, C. R. (1999). Photorealistic Scene Reconstruction by Voxel Coloring. volume 35, pages 151--173, Hingham, MA, USA. Kluwer Academic Publishers. ISSN 0920-5691.

Viana, R. D., Nascimento, E. R., and Ferreira, R. A. C. (2014). On the Development of a Fully Automatic Methodology to Create Smooth Mesh for 3D Models from Concept Arts. In *Proceedings of the XIII Brazilian Symposium on Games and Digital Entertainment - SBGames 2014*, Porto Alegre, Brazil.

Appendix A

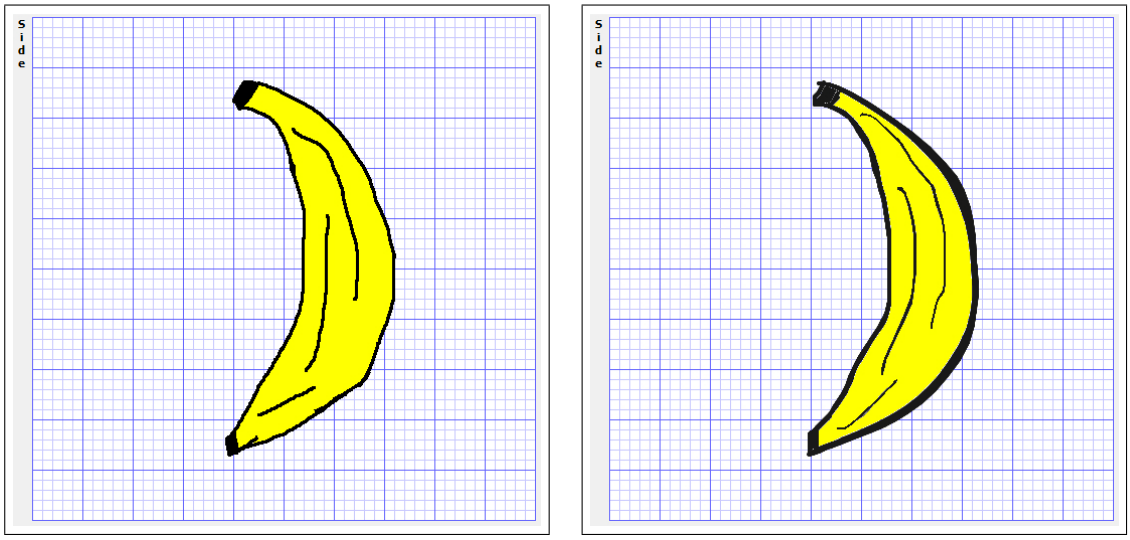
User Interface user case

In this appendix, we present a user case of our user interface and explain further the implementation of the 2D and 3D widgets, and parameters in the configuration panel.

A.1 User Interface explained

A.1.1 2D Drawable Widget

The 2D drawable widget is a Paintbrush-like component that let the user draw a bitmap image. It has a 512×512 pixels dimension as just as the same size our methodology works with. Differently from Paintbrush, our application implements a more stylized



Default Paintbrush brush Our stylized brush

Figure A.1. An example of the stylized brush. Notice the strokes present a more natural flow along their paths.

brush, which takes into account the mouse pointer speed and angle to stroke, providing it a more natural drawing feeling, as we can compare on Figure A.1, given by

$$\begin{aligned} From_t &= To_{t-1}, \\ \Delta_t &= From_t - MousePos_t, \\ To_t &= (\Delta_t \times PertStrength) \times Pertubation, \end{aligned} \tag{A.1}$$

where $MousePos_t$ is the current mouse position, $PertStrength$ indicates how strong is the flow and $Pertubation$ is a random offset given by $U([0,1]) * 0.1 + 0.5$. Then, we trace a line using $From$ and To as initial and end points. In order to achieve this effect, any stroke is actually a set of small strokes calculate by Equation A.1, which are strengthened as much as fast and curved is the stroke path.

Furthermore, it supports zoom and pan increasing the accuracy of concept art details. In addition, we provide a grid as background to assist in drawing as it is supposed that all the views match amongst them. Also, the drawable widgets show a guideline, which represents the brush projection (mouse's cursor) from the widget being drawn in that widget. It is easily achieved as the three drawable widgets are from well-defined viewpoints and have a one-to-one correspondence e.g. when the user traces a line in the front drawable widget, a guideline shows up on side and top drawable widgets.

A.1.2 3D Scene Widget

The 3D scene widget (Figure A.2) is an OpenGL-based 3D viewer component that presents the generated 3D model. It lets the user rotate, translate and scale the 3D model inside the 3D scene with the mouse. The resulting model will be shown in the 3D scene widget either whenever the user hit Build button or as soon as a modification is computed in any 2D drawable widget if Auto-build checkbox is ticked. Furthermore, the user can toggle between orthographic and perspective projection. Also, the object can be visualized as solid or wireframe (The triangles are hidden and is only shown the edges).

A.1.3 Configuration Panel

The configuration panel let the user configure several features that customizes the user experience within the application such as brush color and size, type of 3D projection in the 3D scene widget, and so on.

Figure A.2. The 3D View widget presents the generated 3D model and let the user manipulate the model.

A.1.3.1 Drawing Configuration Panel

Figure A.3 shows the interface of drawing configuration panel, and the meaning of the fields is as follows:

Pencil, Fill, Eraser: are the available brush types. Pencil is a brush that draws the path performed by the mouse pointer; Fill fills the area following the same pixel color in which it started similar to the bucket tool on Paintbrush; Eraser works similar to Pencil but it erases instead of painting.

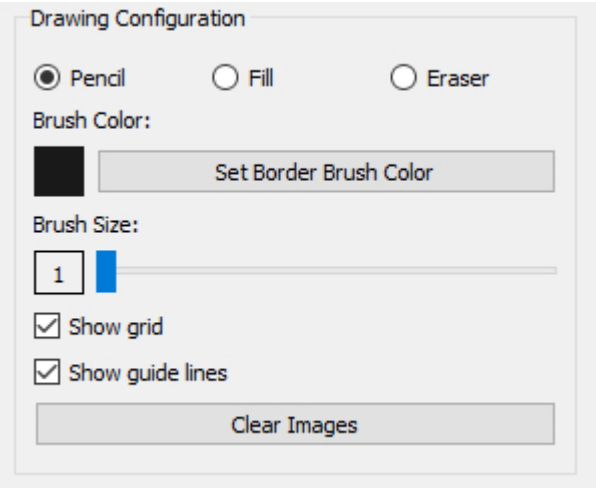


Figure A.3. The interface of drawing configuration panel.

Brush Color: is straight forward, it is the color of the brush for both Pencil and Fill tools. To change it just click on colored rectangle.

Set Border Brush Color: sets the brush color to the specific color used on Geometric Consistency, it informs on the concept art where the surface discontinuity is.

Brush Size: configures the size of the brush, which only affects Pencil and Eraser tool. The size is given in pixels, however, due to brush stylization; it could be thicker depending on the pencil motion.

Show grid: toggles between showing and hiding the grid background in the drawable widgets. Notice that this background is not taken as part of the concept art when processing our methodology.

Show guidelines: toggles between showing and hiding the guidelines presented when drawing in the drawable widgets.

Clear Images: vanishes any drawing performed in the drawable widgets.

A.1.3.2 3D Configuration Panel

Figure A.4 shows the 3D configuration panel interface and the meaning of the fields are as follows:

Perspective, Orthogonal: toggles the projection between perspective and orthogonal.

Show as wireframe: toggles the 3D object surface presentation between as solid and wireframe.

Background Color: defines the 3D scene background color. To change the color just click on colored rectangle.

Reset 3D View: removes any translation, scale, and rotation that has been made on the object.

Method Configuration regards to parameters used by our methodology, and it is as follows:

Voxel Grid Size: is the voxel grid resolution that the object will be built upon. As big is the grid resolution as more details the 3D object has.

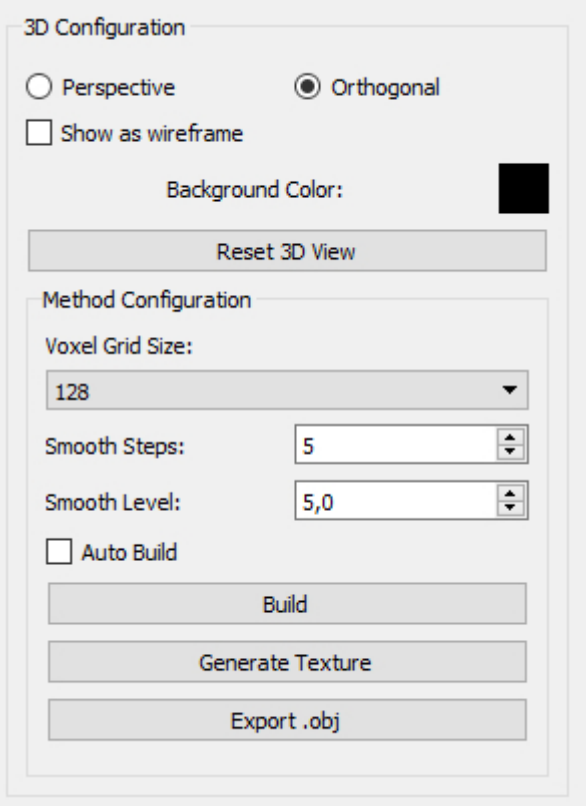


Figure A.4. The interface of 3D configuration panel.

Smooth Steps: defines how many times the smoothing will be applied to the model.

Smooth Level: defines the max angle between two normal that are eligible to be smoothed.

Auto Build: if checked, whenever a change is computed in any drawing widget, the build routine is called.

Build: calls our methodology to reconstruct the 3D model surface only. Due to the time spent to generate the texture, texture generation is separated in another field.

Generate Texture: calls our methodology to compute the texture. Since it takes a long time, this process is separated from the 3D model reconstruction step.

Export .obj: exports and save on the disk the generated 3D model as a .obj file. If generate texture was not clicked, the 3D model is exported without texture information.

A.2 User Case

The short user case below is a demonstration how straightforwardly and conveniently our methodology can be fitted to the artistic pipeline.

Firstly, the user selects the Pencil tool and the brush size:

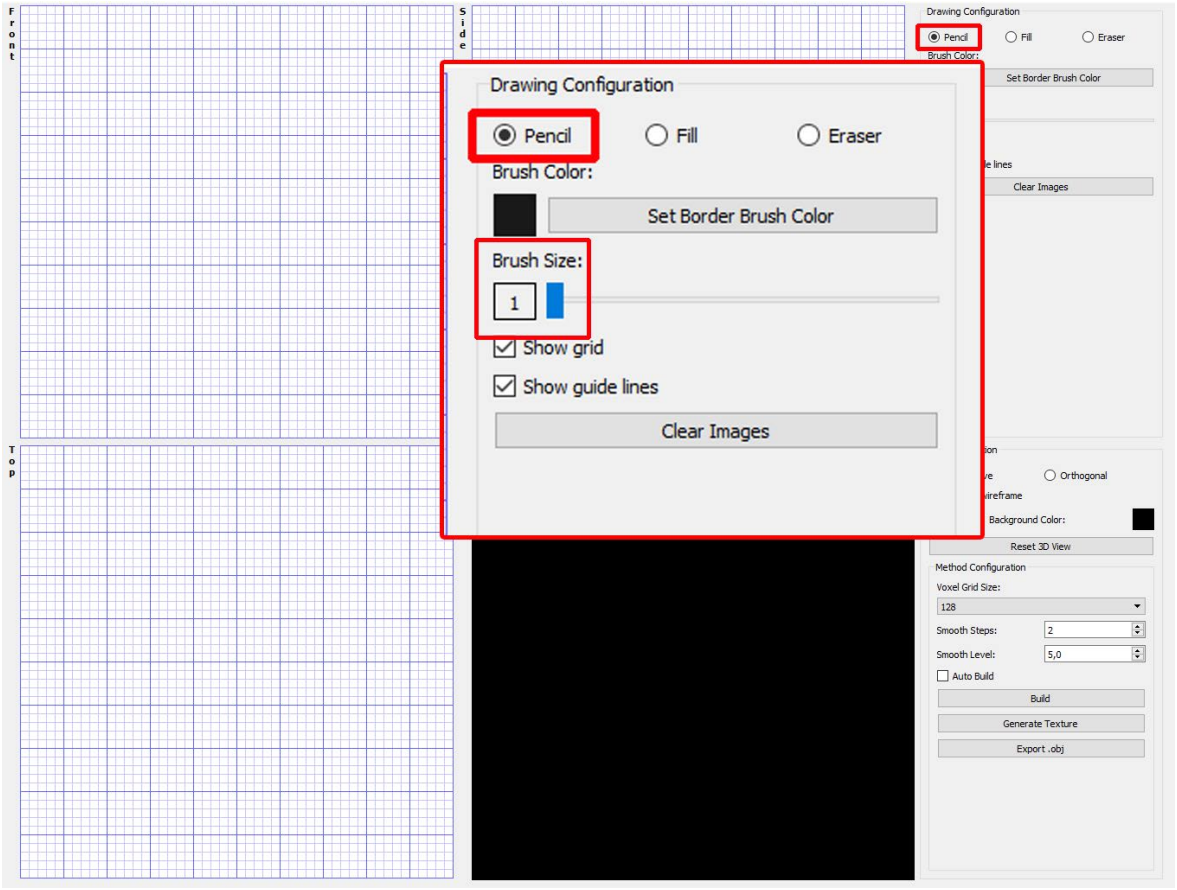


Figure A.5. Pencil tool and brush size selection.

Then, sketches the object shape onto 2D widgets:

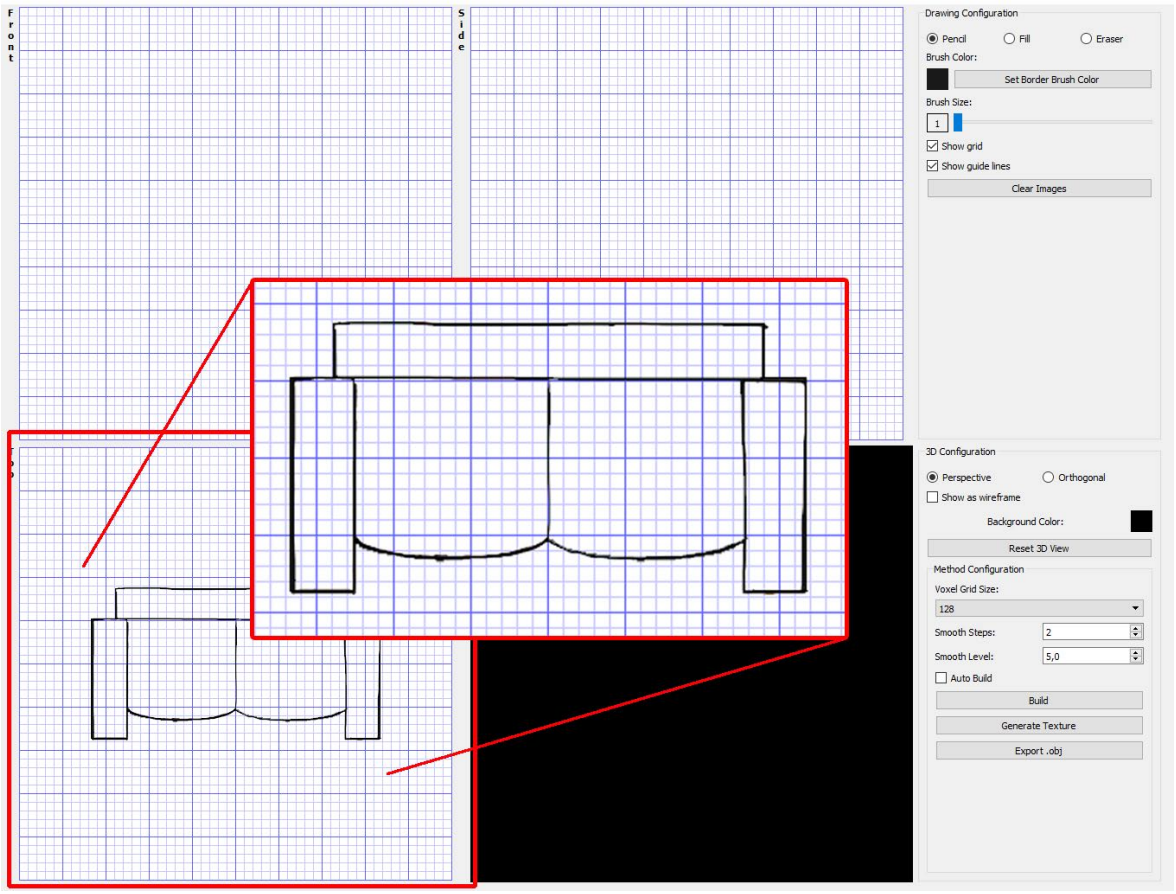


Figure A.6. Shape sketching.

Using the Fill tool and setting the brush color, the user defines the object's color surface:

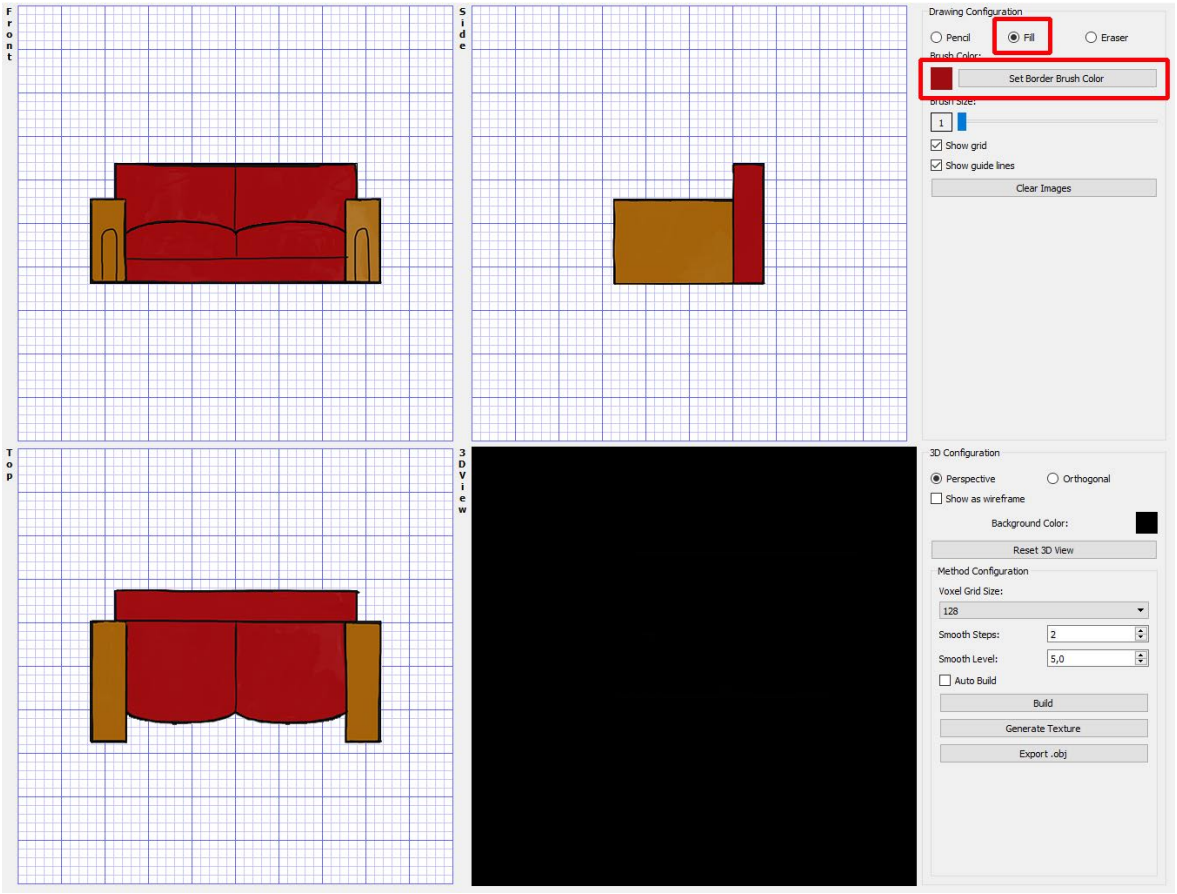


Figure A.7. Fill tool and brush color for filling the drawings.

After the pieces of concept art are finished, the user just needs to press the Build button, and our methodology shows the generated 3D model in the 3D widget almost instantly:

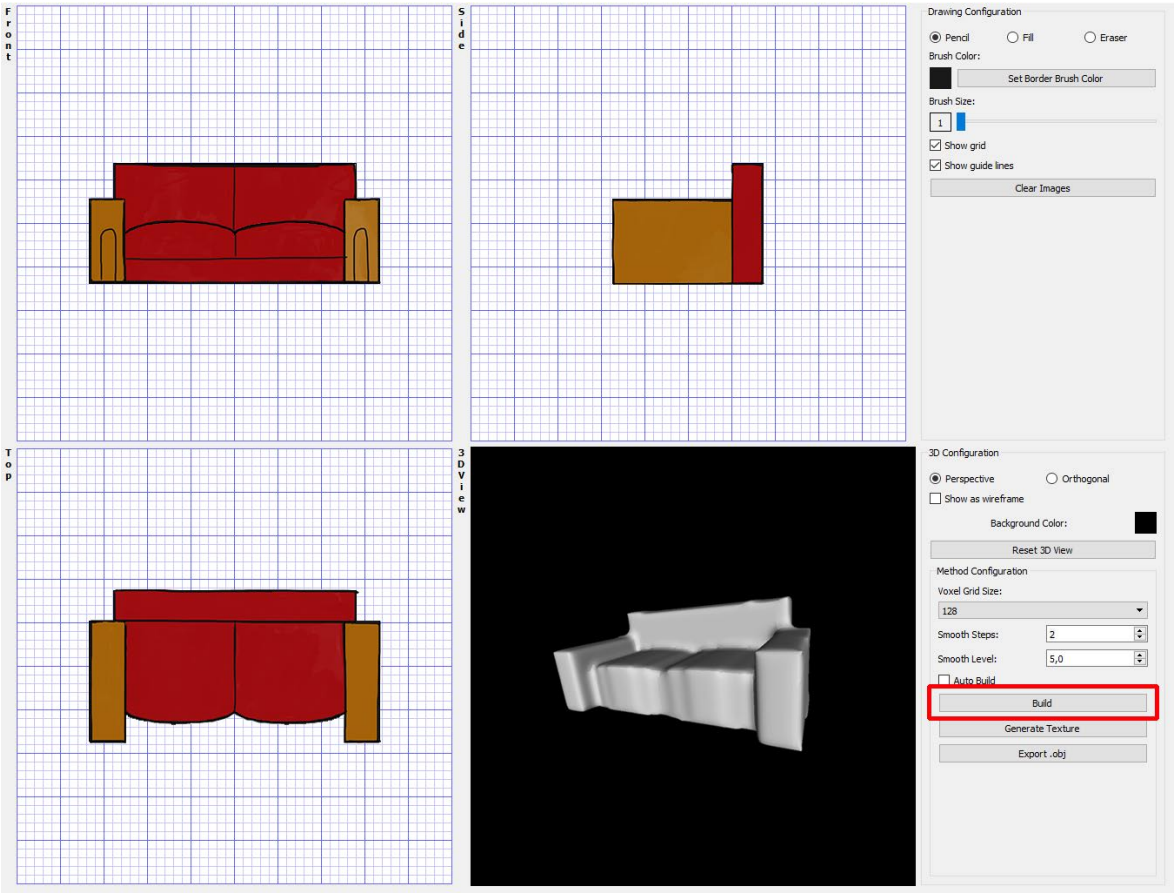


Figure A.8. The result 3D models is presented in the 3D widget.

By hitting the Generate Texture button, the texture map is calculated and the final result shows up in the 3D widget.

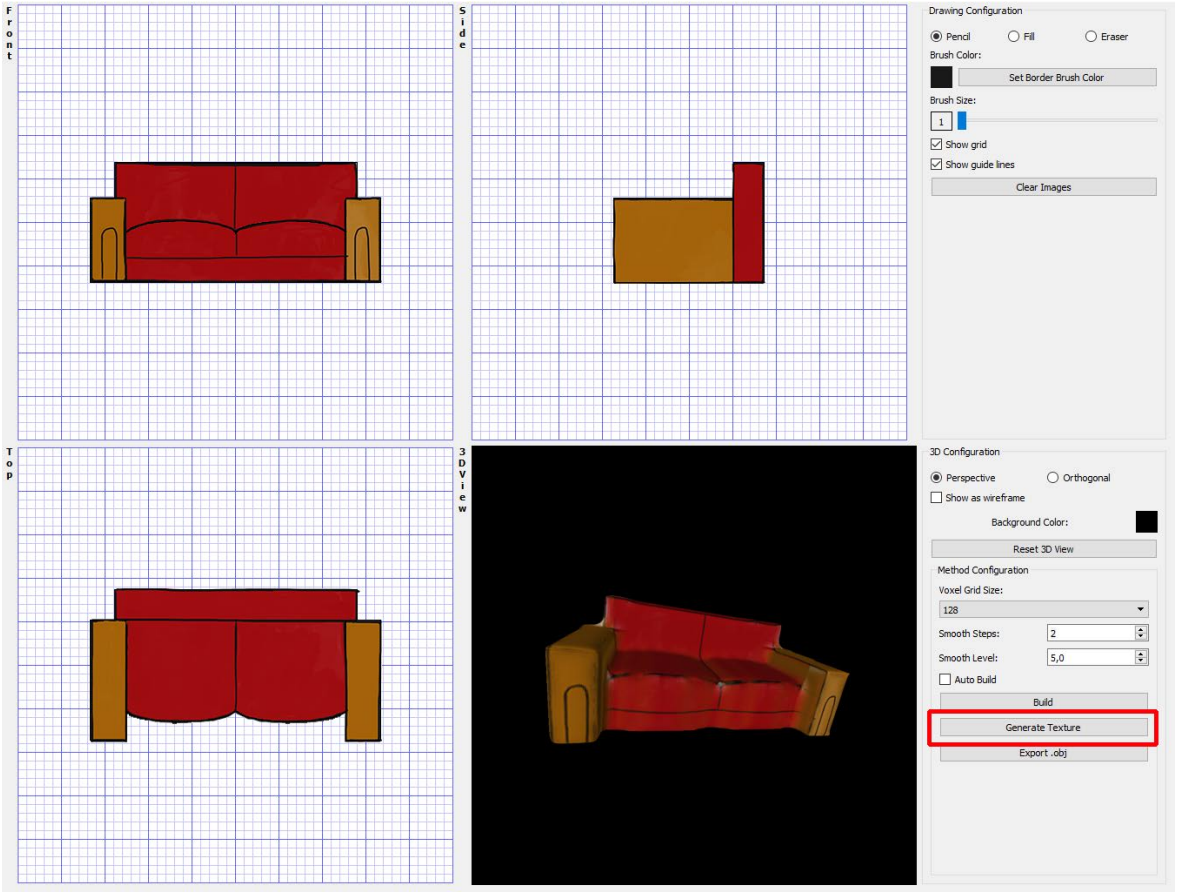


Figure A.9. Texture generation and final result.

Appendix B

3D Artists Experiment Time Discrimination

As in the Chapter 4 we presented the average of the 3D artists creation time for the sake of simplicity and conclusion, in this appendix, we divulge below the times assessed in the experiments individually.

In regard to the artists experience, both artist 1 and artist 2 claimed they had 5 years of 3D modeling experience for video games. Artist 3 has 3 years of experience in 3D modeling, however, he works with digital sculpting tools as well. Artist 4 was the most experienced amongst them having 9 years of 3D modeling experience.

B.1 Time Discrimination

Artist 1	Dog		Head		X-Wing	
	from scratch	coarse given	from scratch	coarse given	from scratch	coarse given
Modeling Time (hh:mm)	3:00	**2:06**	4:00	**2:30**	3:00	**1:30**
Texturing Time (hh:mm)	2:30	2:30	**3:30**	4:00	**1:30**	2:18
Total Time (hh:mm)	5:30	**4:36**	7:30	**6:30**	4:30	**3:48**

Table B.1. Artist 1 modeling and texturing time.

Artist 2	Dog		Head		X-Wing	
	from scratch	coarse given	from scratch	coarse given	from scratch	coarse given
Modeling Time (hh:mm)	3:30	**2:36**	3:45	**3:00**	3:15	**1:48**
Texturing Time (hh:mm)	3:00	**2:48**	**2:00**	2:12	**2:00**	3:30
Total Time (hh:mm)	6:30	**5:24**	5:45	**5:12**	**5:15**	5:18

Table B.2. Artist 2 modeling and texturing time.

Artist 3	Dog		Head		X-Wing	
	from scratch	coarse given	from scratch	coarse given	from scratch	coarse given
Modeling Time (hh:mm)	3:00	3:00	3:00	**2:06**	3:00	**1:48**
Texturing Time (hh:mm)	2:24	**1:30**	**2:24**	2:42	**1:30**	2:24
Total Time (hh:mm)	5:24	**4:30**	5:24	**4:48**	4:30	**4:12**

Table B.3. Artist 3 modeling and texturing time.

Artist 4	Dog		Head		X-Wing	
	from scratch	coarse given	from scratch	coarse given	from scratch	coarse given
Modeling Time (hh:mm)	3:00	**1:30**	2:15	**1:00**	1:45	**1:30**
Texturing Time (hh:mm)	2:00	2:00	**1:07**	2:00	**2:00**	2:00
Total Time (hh:mm)	5:00	**3:30**	3:22	**3:00**	3:45	**3:30**

Table B.4. Artist 4 modeling and texturing time.