Realistic Visualization of Accessories within Interactive Simulation Systems for Garment Prototyping

Vom Fachbereich Informatik
der Technischen Universität Darmstadt
genehmigte

DISSERTATION

zur Erlangung des akademischen Grades eines
Doktor-Ingenieurs (Dr.-Ing.)
von

Dipl.-Inform. Martin Knuth

geboren in Hanau, Hessen

Referenten der Arbeit:	Prof. Dr. techn. Dieter W. Fellner Technische Universität Darmstadt
	Prof. Dr. rer. nat. Jan Bender RWTH Aachen
Tag der Einreichung:	19.01.2017
Tag der mündlichen Prüfung:	03.03.2017

Darmstädter Dissertation, 2017
D 17

Danksagung

Wie bei vielen Arbeiten dieser Art, war die Entstehung meiner Dissertation ein langer und manchmal mühevoller Prozess. Ich kann mir nur schwer vorstellen, dass meine Arbeit ohne die Motivation von und die vielen Diskussionen mit Freunden und Kollegen in ihrer jetzigen Form vorliegen würde oder überhaupt zu einem erfolgreichen Abschluss gebracht worden wäre.

Für die Möglichkeit, im Forschungsbereich der Graphischen Datenverarbeitung arbeiten zu dürfen, die Betreuung meiner Arbeit und die Übernahme des Referats danke ich Herrn Professor Dieter Fellner. Besonders die Diskussionen über den Kern meiner Arbeit halfen mir die einzelnen Teilmengen zu bündeln und mich so auf die wesentlichen Teile zu konzentrieren. Ebenso danke ich Herrn Professor Jan Bender für die Übernahme des Koreferats und die gemeinsamen inspirierenden und motivierenden Diskussionen im Vorfeld, die zu einer Vielzahl von Publikationen und schließlich zu dieser Arbeit führten.

Ein spezieller Dank für die gute Zusammenarbeit und die unermüdlichen, vielen Anregungen gehen an den Leiter der Abteilung Informationsvisualisierung und Visual Analytics – Herrn Professor Jörn Kohlhammer und seinen Stellvertreter Thorsten May - ohne den forschungsfreundlichen Grundtenor in der Abteilung wäre die Arbeit schlicht nicht machbar gewesen.

In diesem Zusammenhang möchte ich auch Herrn Professor Arjan Kuijper danken für seine Unterstützung bei den diversen Publikationsprojekten, welche zum Großteil die Kernforschungen dieser Arbeit repräsentieren. Ich danke Herrn Professor Michael Gösele, dass er - zusammen mit Jan Bender und Arjan Kuijper - geholfen hat, meine Kernpublikation zu veröffentlichen. Es war für mich sehr lehrreich und hat ganz wesentlich die Art und Weise beeinflusst, wie ich wissenschaftliche Ergebnisse für eine Publikation verpacke.

Mein Dank geht auch an die vielen ehemaligen und gegenwärtigen Kollegen aus dem Institut, deren Anregungen und Motivation mich in meiner Arbeit weiterbrachten. Ich danke Arnulph Fuhrmann, der mich überhaupt erst in dieses Thema hineinbrachte, Clemens Groß, Thorsten May, Jörg Sahm, Ingo Soetebier, Sascha Schneider, Marcus Hoffmann, Tatiana von Landesberger, Marco Hutter, Tobias Ruppert, Andreas Bannach, Tobias Franke, Daniel Weber, Thomas Gerlinger, Christian Stab, Kawa Nazemi, Martin Steiger, Sebastian Maier, James Twellmeyer, Pedro Santos, Martin Ritz und Constanze Fuhrmann.

Gabi Knöß, Patricia Häg und Eva Kühnel danke ich dafür, dass sie mir nach Kräften von der organisatorischen Seite her den Rücken freigehalten haben.

Auch meinen Master- und Bachelorstudenten, meinen Praktikanten und meinen wissenschaftlichen Hilfskräften möchte ich für die gute Zusammenarbeit danken. Insbesondere danke ich meinen Diplom- und Masterstudenten Jens Tinz, Christian Altenhofen, Fabian Bauer und Nikolas Schmitt für ihre sehr gute Arbeit. Ein ganz besonderer Dank geht an meine Eltern und meine Schwestern, die mich aushalten mussten, wenn es stressig wurde und die mir durch ihr beharrliches Nachfragen, wie es denn mit der Dissertation vorangeht, wahrscheinlich mehr geholfen haben, als mir bewusst ist. Ihnen und euch allen noch einmal ein herzliches Dankeschön!

Martin Knuth, Darmstadt, August 2017

Abstract

In virtual garment prototyping, designers create a garment design by using Computer Aided Design (CAD). In difference to traditional CAD the word "aided" in this case refers to the computer replicating real world behavior of garments. This allows the designer to interact naturally with his design. The designer has a wide range of expressions within his work. This is done by defining details on a garment which are not limited to the type of cloth used. The way how cloth patterns are sewn together and the style and usage of details of the cloth's surface, like appliqués, have a strong impact on the visual appearance of a garment to a large degree. Therefore, virtual and real garments usually have a lot of such surface details.

Interactive virtual garment prototyping itself is an interdisciplinary field. Several problems have to be solved to create an efficiently usable real-time virtual prototyping system for garment manufacturers. Such a system can be roughly separated into three sub-components. The first component deals with acquisition of material and other data needed to let a simulation mimic plausible real world behavior of the garment. The second component is the garment simulation process itself. Finally, the third component is centered on the visualization of the simulation results. Therefore, the overall process spans several scientific areas which have to take into account the needs of each other in order to get an overall interactive system. In my work I especially target the third section, which deals with the visualization.

On the scientific side, the developments in the last years have shown great improvements on both speed and reliability of simulation and rendering approaches suitable for the virtual prototyping of garments. However, with the currently existing approaches there are still many problems to be solved, especially if interactive simulation and visualization need to work together and many object and surface details come into play. This is the case when using a virtual prototyping in a productive environment.

The currently available approaches try to handle most of the surface details as part of the simulation. This generates a lot of data early in the pipeline which needs to be transferred and processed, requiring a lot of processing time and easily stalls the pipeline defined by the simulation and visualization system. Additionally, real world garment examples are already complicated in their cloth arrangement alone. This requires additional computational power. Therefore, the interactive garment simulation tends to lose its capability to allow interactive handling of the garment.

In my work I present a solution, which solves this problem by moving the handling of design details from the simulation stage entirely to a completely GPU based rendering stage.

This way, the behavior of the garment and its visual appearance are separated. Therefore, the simulation part can fully concentrate on simulating the fabric behavior, while the visualization handles the placing of surface details lighting, materials and self-shadowing.

Thus, a much higher degree of surface complexity can be achieved within an interactive virtual prototyping system as can be done with the current existing approaches.

Zusammenfassung

Beim Virtuellen Prototyping (VP) von Bekleidung erstellt ein Designer ein Kleidungsstück mittels Computer Aided Design (CAD). Im Unterschied zum herkömmlichen CAD steht das Wort "Aided" in diesem Fall für ein Replizieren des physischen Verhaltens des Kleidungsstücks. Hierdurch ist es dem Designer möglich ähnlich, wie mit einem physischen Prototyp zu interagieren. Im Kleidungsdesign hat der Designer eine breite Palette von Ausdrucksformen für sein Werk, welche nicht auf die Form oder Auswahl von Stoffen beschränkt ist. Dies sind z.B.: Die Art und Weise, wie Stoffmuster gefügt, Verzierungen angebracht und Oberflächenobjekte platziert werden. Diese Applikationen haben einen starken Einfluss auf das Aussehen eines Kleidungsstücks. Reale Kleidungsstücke weisen in der Regel viele dieser Merkmale auf. Es ist daher für den Designer wichtig, dass diese auch am virtuellen Prototypen vorhanden sind.

Virtuelles Prototyping von Bekleidung selbst umfasst ein interdisziplinäres Gebiet. Mehrere Probleme müssen gelöst werden, um ein effizientes nutzbares Virtual Prototyping-System für Bekleidungshersteller zu schaffen. Ein solches System kann in drei Abschnitte unterteilt werden: Der Erste befasst sich mit der Übernahme des Materials und anderer Parameter, die für die Simulation eines plausiblen, realistischen Verhaltens notwendig sind. Der Zweite ist auf das Simulationsverfahren selbst zentriert. Der dritte und letzte Abschnitt befasst sich mit der Darstellung der Simulationsergebnisse. Um effizient zu sein, müssen die hierbei überspannten wissenschaftlichen Bereiche auf die Bedürfnisse der jeweils anderen Techniken Rücksicht nehmen, um ein interaktives Gesamtsystem zu erhalten. Meine Arbeit ist hier speziell in der dritten Komponente verortet, mit dem Ziel der Simulationskomponente Arbeit abzunehmen.

In wissenschaftlicher Hinsicht wurden in den letzen Jahren große Verbesserungen sowohl bei Geschwindigkeit und Zuverlässigkeit der Simulation als auch bei den Rendering-Ansätzen erzielt, die für das Virtuelle Prototyping von Kleidungsstücken geeignet sind. Bei den derzeit existierenden Ansätzen bestehen jedoch noch viele Probleme, die gelöst werden müssen. Dies ist besonders der Fall, wenn interaktive Simulation und interaktive Visualisierung zusammenarbeiten müssen und eine hohe Komplexität durch Objektdetails gegeben ist. Dies ist oft der Fall im produktiv eingesetzten VP-System.

Aktuelle Ansätze versuchen Oberflächenobjekte als Teil der physikalischen Stoffsimulation zu behandeln. Während dies theoretisch direktes Feedback innerhalb der Simulation ermöglicht, wird andererseits eine mitunter sehr große Datenmenge früh in die Pipeline eingespeist. Diese Datenmenge muss übertragen und verarbeitet werden. Dies kostet viel Verarbeitungszeit und kann leicht dazu führen, dass das System aus dem interaktiven Rahmen fällt. Zusätzlich sind reale Bekleidungen bereits so komplex, dass allein schon die

Simulation der Stoffe den Simulator auslastet. In der vorliegenden Arbeit stelle ich eine Lösung vor, die dieses Problem entschärft, indem sie Oberflächendetails aus der Simulationsphase herauslöst, diese unabhängig in einem komplett auf der GPU laufenden Verfahren verarbeitet und während des Rendering Prozesses hinzufügt.

Mein Ansatz erlaubt eine getrennte Verarbeitung des physikalischen Verhaltens des Kleidungsstücks (Schnitt) und des Oberflächenstylings. Nun kann sich der Simulationsprozess voll auf die Simulation des Stoffverhaltens konzentrieren, während die Visualisierung sich um Oberflächendetails, Beleuchtung, Materialien und Selbstabschattung kümmert. Des Weiteren stehen dem Visualiserungsprozess hierdurch semantische Informationen zur Verfügung, die eine Reduktion der Komplexität des Schattierungsproblems erlauben. Auf diese Weise kann ein viel höherer Grad an Oberflächenkomplexität in einem interaktiven virtuellen Prototypen-System erreicht werden, als es mit den gegenwärtigen bestehenden Ansätzen durchgeführt werden kann.

Contents

1 Introduction

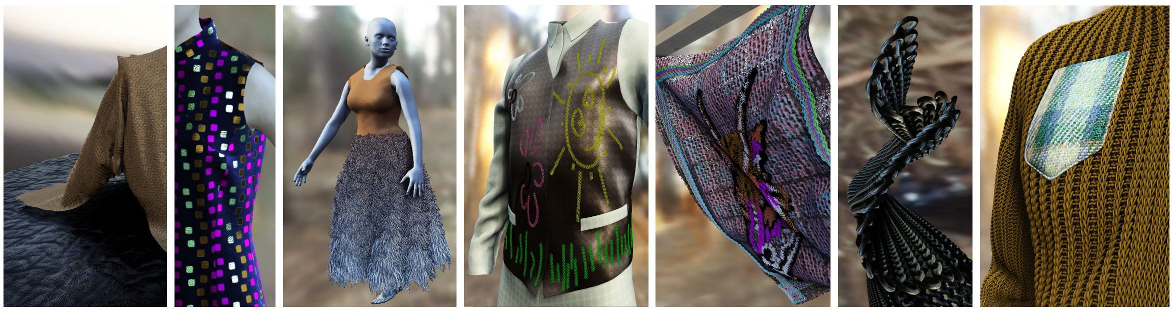


Figure 1.1: In this work I present and combine a real-time rendering approach with handling of surface objects especially found in garment design. The rendering itself is capable of handling various materials under natural lighting conditions taking into account directional occlusion (leftmost image). The combined approach is capable of handling different collections of surface objects including the complete replacement of the original garment's surface (rightmost image). My approach allows to vary and change all model related parameters in real-time, which makes it usable for interactive modeling in conjunction with garment simulation in garment design applications.

1.1 Motivation

My work is motivated by the recent developments of virtual prototyping (VP) within the garment industry in the last years. In the garment industry time to market for issuing new fashion collections is one of the key aspects in order to stay in business. A garment company has to face a fast changing market with many competitors. Therefore, the pressure to keep both costs and time low is strong. For classical mass production, this can be done by scaling up the production. However, markets like the European one, are more and more driven by made to measure and user customization. In this case the whole process needs more flexibility, a typical issue proposed in other industrial markets in Europe (industry 4.0).

A way for the garment industry is to implement VP throughout the development processes. The classical process consists of a design flow which incorporates the construction of several physical prototypes, (see Figure 1.2). This includes tailoring, which is up to now

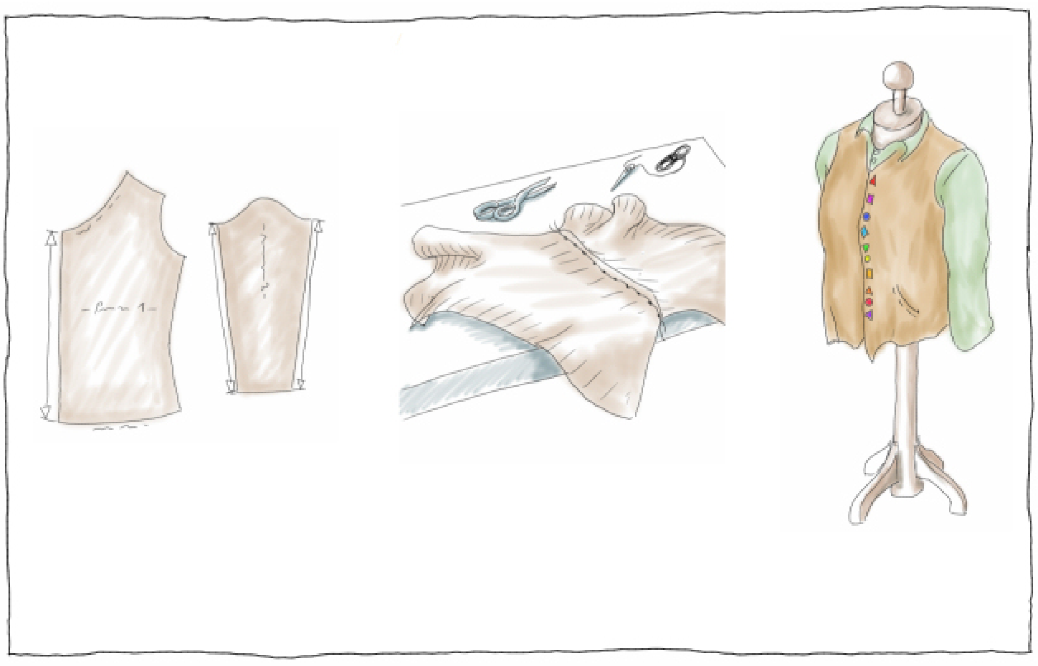


Figure 1.2: Process outline of the core fashion development process. From left to right: Starting from an idea conception, 2D patterns are generated. These are cut from cloth and sewn together to build a first garment prototype. This prototype is then used in a try-on session in order to evaluate fit, quality and the appearance of the new design. This process is repeated several times until a satisfying product is created. The prototype production is a mixed reality process. Pattern design is done on a computer, while sewing and try-on is done in reality. Therefore, data has to be transmitted from virtual to real world and back again.

a by hand process which cannot be automated. Thus, mixing virtual and real production steps stall the prototype production pipeline by requiring interfacing between reality and the virtual world, since this prototype production is run in a loop. Especially the cutting and sewing necessary to get from the virtual world to a physical prototype result in a mayor bottleneck, which costs time and resources.

Here, virtual prototyping can significantly reduce the necessary number of physical samples. However, its diffusion and efficient use in the industry is still low. This is due to the complexity and low fidelity the currently available tools have. In my work I present the results of scientific research within several projects with the garment industry. The main goal of these projects where to increase the speed of the simulated virtual prototypes and to reduce the complexity of the technology and its use.

The focus of my work is one of the core components of this virtual process - the interactive and still realistic visualization of the prototype.

1.2 Garment Development Process

Garment design is different to classical 3D modeling processes. In a classical modeling process a designer aims for full control over his model. Which is modified until it "looks good".

In garment design the main difference to the classic modeling process is given by its necessity to be produced from planar sheets of cloth.

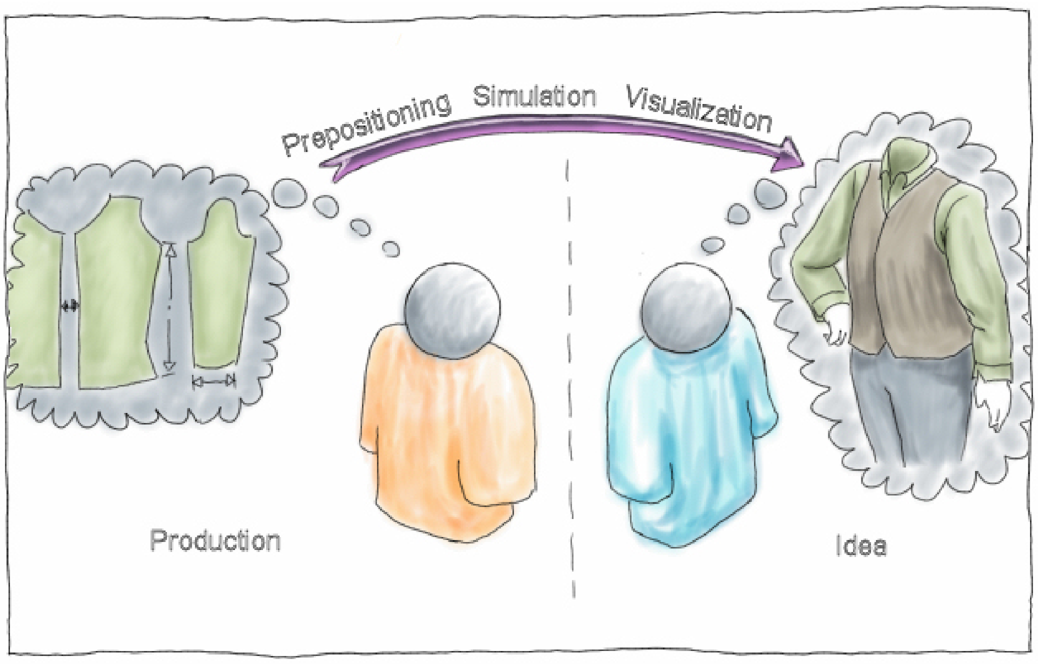


Figure 1.3: Iterations in the garment design process: The designer (right) discusses the vision of a design with the pattern maker (left), who transfers this vision into 2D sewing patterns, which can be used for production. In the classical case this feedback is done by creating a physical prototype which is then used within a try-on session. In order to give the designer an immediate 3D feedback in form of a virtual garment, several steps need to be performed. The patterns need to be prepositioned, simulated and visualized, as depicted by the center arrow.

In garment design two stakeholders are involved. The first is the garment designer, which is basically the artist. The second is the pattern maker, who works as an engineer. While the role of the artist is to generate new ideas, the role of the engineer is to transform the ideas into a producible format. The three core steps in Figure 1.2 show this design process. These two actors work as a control loop, where the pattern maker controls and changes the parameters and the designer judges whether the result matches the expectations (see Figure 1.3).

This is in big contrast to classical modeling, as implemented in the film or game industry. Garment design is an engineering process which has to follow the physical rules the planar pieces of cloth demand. If the process is not regulated along these rules, the garment cannot be manufactured. Therefore, all operations performed have to be related to this 2D world of planar cloth, represented by the the 2D sewing patterns. This property of the process forbids the use of classical pure 3D modeling processes creativity tools that Maya, 3D Max or ZBrush provide, except for design studies, which are not used for production.

Looking at the interaction between the pattern maker and the garment designer gives an impression of what a VP system for clothes needs to provide. In the classical approach the pattern maker generates sewing patterns from the sketches the designer created. These are then cut and sewn into a prototype as feedback for the designer. The designer discusses changes which are then performed by the pattern maker again. This decision process is run several times until a prototype is ready for the second part of the technical development (see Figure 1.3).

The main benefit of introducing virtual prototyping of garments into this process is to reduce the time and amount of necessary physical prototypes. The goal is to virtually resemble all steps within the loop. Results from the European project Future Fashion Design [wFD15] have shown the usefulness of the process when applied to most of the steps in the loop. Real world results from this project have shown that at least one prototype is needed for the final evaluation and the sales man. Therefore, the number of physical prototypes and the number of time intensive steps can be significantly reduced even in a real production scenario.

In order to be beneficial, such a VP system needs to reflect the two physical prototyping parts (the sewing and the try-on) in a virtual manner. If this is done in a sufficient way, feedback discussion and first changes can be directly done on the computer, skipping several physical prototypes.

Aside from a sewing pattern CAD system, the core of such a VP system for garments consists of three components:

1. A pre-positioning step, which places the sewing patterns around a 3D figurine.
2. A real-time garment simulation system mimicking the mechanical behavior of cloth.
3. A tightly coupled real-time visualization system mimicking the optical behavior of the cloth.

The last two integrate into an application, which allows the user to interact with the garment's cloth, while allowing to judge the visual appearance under natural lighting conditions with all details of the garment available. Changes to the garment model are still done in 2D by the pattern maker, guaranteeing the producibility of the garment. The details of creating such a system are presented by Fuhrmann in [Fuh06] An introduction of the complexity of the decisions behind garment modeling and and how VP systems assist these processes in general is given by Fan et al. in [JFH04].

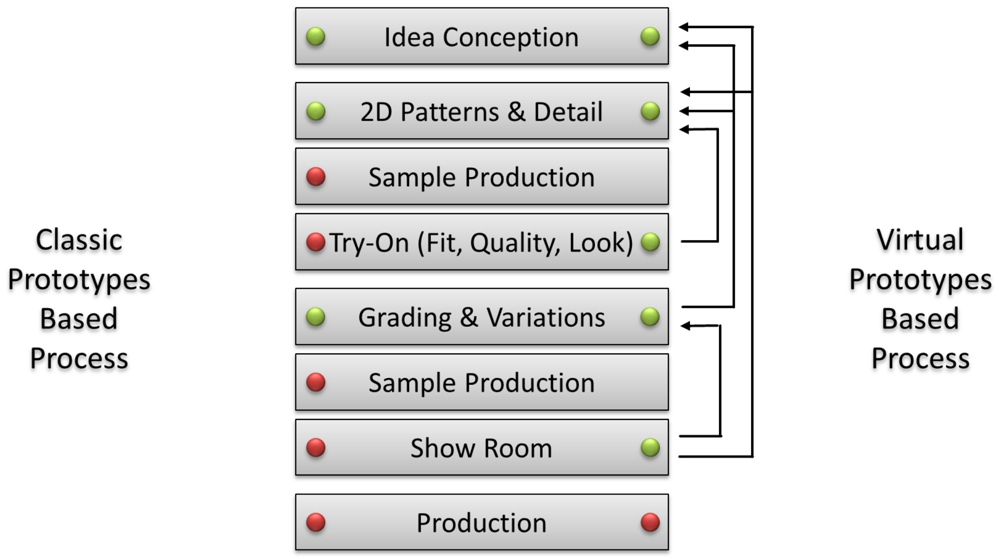


Figure 1.4: Overview of the complete garment design process including parts, which can benefit from virtual prototyping. On the left side the physical based process is described. On the right side an ideal virtual process is shown. Green dots depict steps performed on the computer, red dots physical processes. Arrows represent feedback between the single stages of the development

1.3 Virtual Prototyping of Garments Development Process

The challenge for building a VP system for garments is given by the requirement to provide high interactivity and fidelity of the representation required for the try-on process. Especially in the early design and technical development phase everything can be subjected to changes. Decorations on the garment are changed dynamically, cloth types are replaced, the garment is draped with needles, sewing patterns are changed. For a VP system this demands a high reactivity in order to change both the sewing patterns and the surface decorations. Always a prompt, plausible visual feedback needs to be provided, suitable for realistically judging the visual appearance of the new design. The VP system needs to be able to handle changes in the patterns, surface decorations and optical materials (change of the textile) in real time. This is necessary in order to cope with the dynamic changes the designers and the pattern makers perform on the garment in this conception phase.

In order to reduce costs and production time further, it is a natural extension to create complete collections with the help of virtual prototyping. Luckily, virtual prototyping is not limited to the core development. Figure 1.4 gives an overview of a classic garment design process and shows parts, which can benefit from VP. In case the core prototype production results in a virtual prototype with a sufficient optical and detailing quality, its use can be extended to the sales-man samples and the show room. In both cases it can be applied to show different configurations or options of the garments without the necessity to have the appropriate sample at hand.

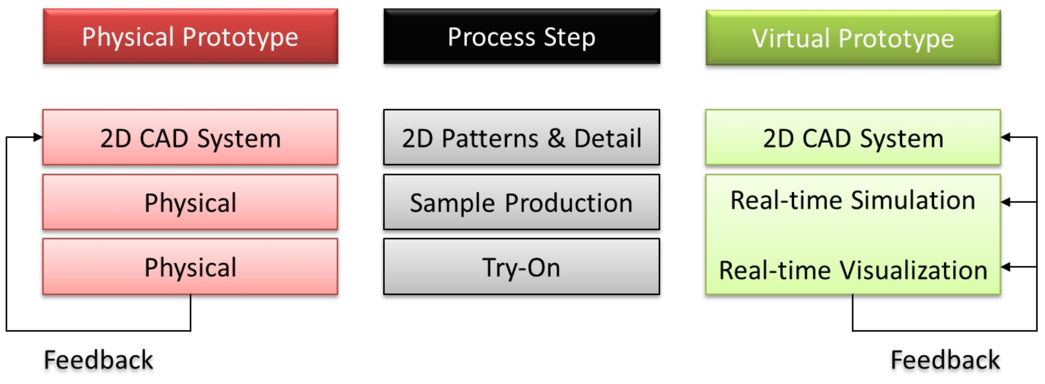


Figure 1.5: Overview of how physical steps of the core development are translated into virtual steps in VP utilizing a system providing real-time interaction with the virtual garments.

A virtual prototyping system for garments requires several components in order to function. An overview of the single steps is given in Figure 1.5. Here, both the classic physical prototyping and the new virtual prototyping are shown in comparison. Since VP has to replace physical prototypes, one of its most important features is its capability to provide real-time feedback to user interaction. Now, the core conditions for the virtual prototyping pipeline can be described the following way:

1. A sewing component has to reflect the interconnection between the single 2D sewing patterns in the virtual world.
2. A prepositioning component must arrange the single sewing patterns around a 3D avatar which is later used for the try-on. It provides the initial state for the physical simulation.
3. A real-time physical simulation of the garment behavior needs to be in charge for all cloth behavior you will see in the real world. It controls the draping behavior, folds, etc. This simulation needs to be real-time capable in order to allow life interaction with the virtual garment. These interactions can be pulling or fixing of parts of the cloth, attaching of several cloth parts on each other or even more complex operations.
4. A geometry post-processing stage details the look of the garment. It adds cloth thickness, appliqués and additional surface details to the garment.
5. A real-time visualization of the simulated model needs to be fast and realistic enough in order to allow interaction on the ever changing geometry and plausible lighting behavior for the evaluation of the look.

My special interest and the results presented in my work focus on the forth and the final stage, which were developed in the course of several research projects.

During these projects with the garment industry several requirements could be identified.

- Data handling: Construction data needs to stay in 2D. It is possible to mold parts of a textile, but the mayor form of a garment is defined by its 2D sewing patterns. Therefore, staying in 2D is mandatory in order to ensure producibility. Due to the same reason the VP system needs to be able to react to both changes in 2D (Sewing pattern CAD) and on the 3D (Virtual prototype) side.
- Interactivity: Within a try-on session the designers work directly with the garment. The same is required for the virtual prototype, demanding a high interactivity of the entire VP system.
- Designers work a lot with surface details like appliqués, borders and seams. Aside from the sewing pattern itself these three are the main degrees of freedom in the design phase. Design decisions are based on these details and the look of the garment.

To address these points successfully the prototyping system needs to be interactive in conjunction with a suitable real-time simulation and visualization. The physical simulation needs to be fast and deliver a sufficient quality in order to be interactive and judge the form of a garment realistically. The system needs to be able to process a high amount of surface details for editing and visualization. The visualization component needs to deliver a plausible view of the model. This includes optical material effect, surface objects and plausible self-shadowing, which is very important to judge the form of an object.

These requirements were too complex to be solved directly and stated several scientific questions, which have to be solved first in order to address the requirements.

Simulation Speed

Research projects with the garment industry have highlighted the need for an increase of speed within the simulation system. With the current technology, a speedup of the simulation system is directly used up by the use of simulation meshes with finer resolution, in order to create a more complex and realistic behavior of the textiles. This rises the question of how much resolution is possible and how much is required?

The following example could offer a response: Consider a simple shirt with roughly 1.5 square meters of cloth, which has a bending radius of 0.5 mm. Interpreting the surface as a signal makes it possible to utilize the Shannon Nyquist Sampling Theorem to estimate the minimal resolution of the simulation grid. A radius of 0.5 millimeters results in a wavelength of 2 mm. According to Shannon-Nyquist we need at least the double frequency for sampling, which means 1mm of resolution. Since we have a 2D grid and diagonal elements, the mesh resolution needs to be $\frac{1}{\sqrt{2}}$ mm of resolution at least. For 1.5 m^2 this would result in roughly 8 millions of triangles for the simulation.

It is possible to build a simplified simulation system which will simulate this in real-time. However, it will target only the simplest parts of garment simulation with a single piece of garment leaving out all kinds of intercollisions [SKBK13].

1.4 Realistic Visualization of Accessories within Interactive Simulation Systems for Garment Prototyping

Up to now I have outlined the environment in which my research is embedded. I will now focus on the part of the process, which can be improved using my approach.

In order to reduce the required computational effort of the simulation I propose an approach which allows to move the computation of surface details from the simulation component into the visualization component of the virtual garment modeling process. This originates in the observation that garments contain a lot of surface details, which are not directly related the physical simulation. To mimic these in the virtual world, the necessary computations consist of decoration and modification of the simulated cloth and have two goals. First, the simulation mesh is modified for rendering in order to create cloth with a thickness and a visible border (hem) (see [Fuh06]) Even thickness is considered, the physical simulation mesh itself is just a flat, thin mesh, which should not be used for rendering. Second, the surface of a garment typically contains a lot of appliqués, such as seams, stickers or buttons. My underlying approach is to offload the calculations of these details from the simulation process into the rendering stage. At the same time we get an organizational split of object types. Now there are objects, which need to be processed on pattern CAD level and new objects, which can be modified on the fly in a classic modeling level. Additionally, there are accessory objects which are only related to the figurine wearing the virtual cloth, such as gloves, shoes and hats, which need to be visualized.

Appliqués are objects which have in common that they are related to the cloth surface. However, they differ in the kind of behavior they show when the underlying cloth is deformed. For example, buttons are attached to a single point on the surface. They will move and tilt according to the surface, but are rigid in their structure. Seams follow the curvature on the garments surface. If the garment is deformed they will follow the deformation. Stickers behave similarly, but in two dimensions. Another kind of appliqués are accessories such as shoes, glasses, gloves etc. Even not related to the cloth surface, they still need to be handled. Their relation to the cloth surface can be complex: different parts of the appliqué can behave differently. Figure 1.6 shows an overview of different appliqué types. Appliqués can be classified as follows:

- (Cuff) buttons, decorations. They are attached to a single point.
- Seams, hem and border. They follow curves on the cloth surface. However, hem and border are part of the fabric geometry and need to be processed differently.
- Stickers, cloth structure, dress handkerchief. These objects behave in relation to an area of the fabric. Stickers and the dress handkerchief are located on top of the cloth surface, while changing the cloth structure itself, requires a different approach.
- Tie, bow tie, belt. These object mix several attachment types. Therefore, they need to be split according to them.
- Headgear, glasses, shoes, gloves. These objects are accessories, which are not related to a fabric surface, but to the avatar, wearing them.

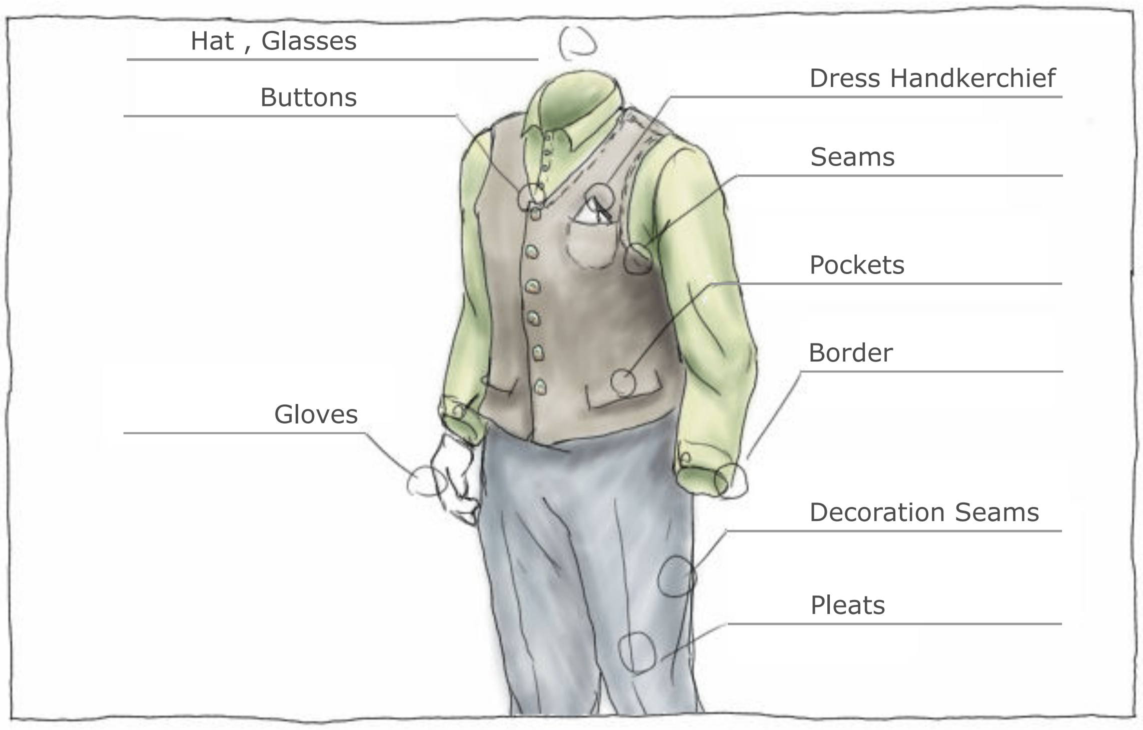


Figure 1.6: Types of common appliqué found on garments.

In order to be able to handle all of these cases, a configurable attachment method is needed.

Scientific Questions to be Solved

In order to process finishing operations apart from the physical simulation two main scientific questions need to be solved simultaneously. These can be formulated as follows:

- How can surface details be processed outside the simulation process?

- How can realistic real-time visualization of complex deforming objects be done?

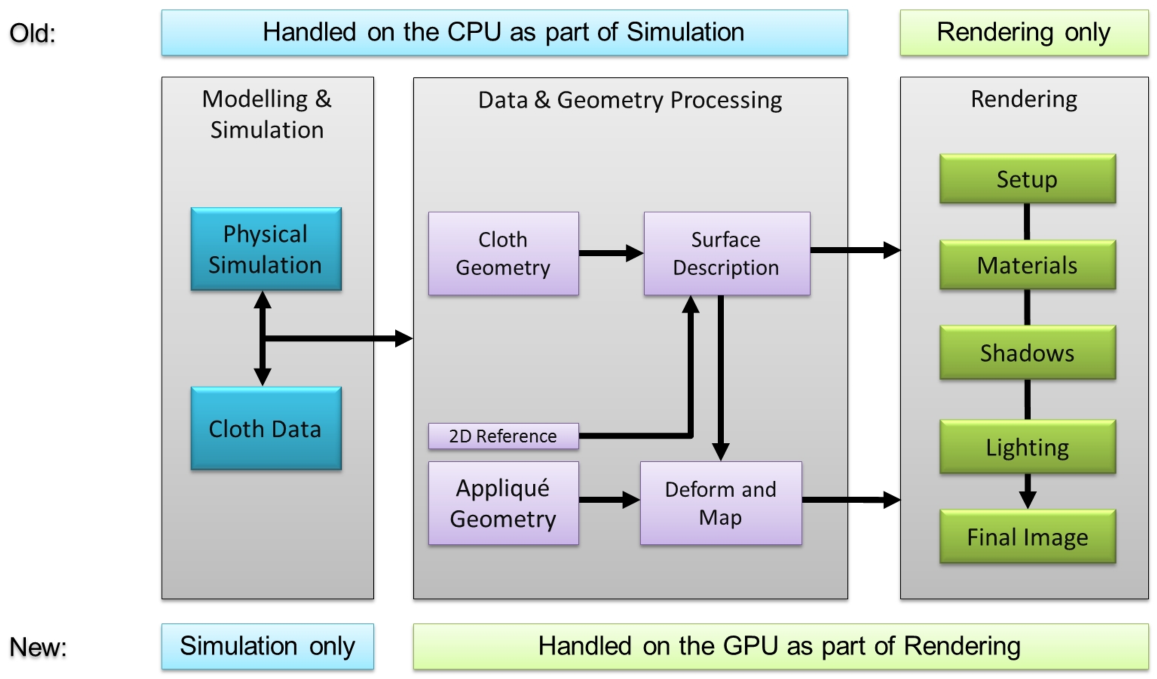


Figure 1.7: Difference between the architecture of a classical interactive garment simulation with visualization to my proposed approach using a visualization subsystem optimized for this kind of application.

1.5 Approach

In my conception I target both questions at the same time.

The methods presented within my concept (Chapter 3) ensure to perform appliqué processing as a geometry manipulation process inside the rendering stage. The key idea is to take advantage in having available both the coarse and the detailed garment model within the visualization part.

The architecture uses a two-step approach:

- First, my geometry processing stage targets the problem of having to process the appliqués outside the simulation with a method, that allows in a flexible way to attach all appliqués on the cloth surface directly by using the 2D CAD data from the sewing pattern space. This way I can handle a wide variety of attachment types in conjunction with real-time deformed cloth surfaces.
- Second, the processed geometry is visualized in real-time to the physical simulation results. As stated above, the overall visualization needs to be fast and plausible in order to not break the overall interactivity of the system. The scene needs to be rendered, incorporating effects like material, self-shadowing and occlusion, which are important for the viewer to get the right optical impression of the garment.

Within classical interactive garment simulation, the geometry processing is performed in the CPU side as part of preparing the visualization data for the GPU based rendering. This preparation generates a lot of detail geometry, which needs to be transferred and processed on a per frame basis by the GPU. Within my approach it is now possible to move this complete preparation step to the GPU side (see Figure 1.7). Now, only the current simulation result and the control data for the details need to be transferred per frame. This is an advantage, since the GPU is better suited for processing larger amounts of data due to its higher memory throughput, releasing the CPU from this task. This way the goal to increase the optical quality and fidelity of the garment for rendering without increasing the load for the simulation system can be achieved.

1.6 Outline of the Thesis

In the following chapters I present my approach sketched in Figure 1.7. In Chapter 2 I give an overview about work related approaches and their specific requirements.

In Chapter 3 I present a conception for detailing and visualization of the virtual prototype, attached directly after cloth simulation. My approach combines techniques in a way which contradicts the weaknesses of the single techniques by spanning both geometry processing and rendering.

In Chapter 4 I present the implementation of the approach. I first present an outline of the architectures and decisions I used for implementation. This is followed by going into detail regarding the specific steps of the single techniques.

Chapter 5 shows results achievable with the techniques and their implementation presented in the previous chapters. The results are centered on the two main aspects of my work, which are geometry processing and visualization in the context of virtual prototyping of garments. For geometry processing I differentiate between objects not related to a garments surface (macro objects) and objects attached to the cloth's surface (appliqués). This is accompanied by a tessellation scheme to improve the surface sampling for low resolution surface objects. On the rendering side, I show results of the two self-occlusion techniques which I use to handle self-occlusion separated into low- (material and geometry-based occlusion) and high-frequency (screen-space based occlusion) occlusion.

In Chapter 6 I conclude my work with an overview of ongoing research and an outlook into future work.

The content of the following chapters is based on my work in several publications. My idea for a hierarchical, cuboid-based deformation technique was presented in [KKK10a]. It is described as method to handle objects placed on figurines. Further, a surface-based deformation technique called deferred warping is presented in [KBGK15]. In this work, this technique is highlighted as cornerstone for handling all kinds of garment surface related objects like sequins and seams solely on the GPU. My idea for fast and flexible GPU-based tessellation was presented in [KKK10b]. Here I use this technique for increasing the mesh resolution of objects placed on garment surfaces.

My vision for a fast and accurate rasterization technique capable of handling self-shadowing of the garments was first presented in [KF05]. A major update of this technique was presented in [KAKB14] and is mainly used for handling the self-shadowing of the scenes utilizing various surface materials. In this work this technique is combined with the hybrid self occlusion approach, outlined in [KK09] in order to be able to efficiently handle fine shadow details of high polygon surface objects.

2 Related Work

My overall approach is split into the conceptions for the geometry processing on one hand and for the visualization part on the other hand. While the goal of the geometry processing part is to assemble a final detailed 3D model, the purpose of the visualization is to apply optical material properties and self-occlusion under natural lighting conditions. This split is present in all chapters.

In this section, for the geometry processing part, my focal point will be set on techniques, which allow processing of geometry objects in relation to surfaces and surface refinement. These techniques are then compared in relation to the originally stated problem of attaching appliqués. For the visualization part I will first focus on rasterization methods for image based lighting in conjunction with directional low frequency occlusion. For high frequency occlusions I present work related to image based approaches, which is good at handling occlusions of fine detail geometry. This is followed by an analysis of how these techniques can be used to solve the problem of real time rendering for highly detailed garments in conjunction with deformation and modification.

2.1 Geometry Processing

2.1.1 Deformation and Transformation in Relation to a Scene

For handling objects within a scene a wide variety of techniques exists. The simplest way of dealing with objects of a 3D scene is to place them object by object into a scene. A more advanced way is to make use of a hierarchy. Handling objects in a hierarchy allows to define relations of objects to each other. This process is exploited by scene graphs, which allow manipulation and modeling of objects within a graph structure. Here, usually linear transformations are used in order to handle positioning and manipulation of 3D objects. However, there are more complex transformations, which allow bending and deformation of 3D objects in a nonlinear fashion. Since both are independent from each other scene graph techniques and deformation techniques can be even combined in order to provide graphs of hierarchical deformations as shown in this section.

Scene Graphs

Graph structures deal efficiently with hierarchical relations of objects within scenes. These so called scene graphs are widely used within graphic applications. Several systems and application programming interfaces (APIs) like X3D, Open Inventor [WHR97], OpenGL Performer [RH94], Java3D [SRD98], OpenSG [RVB02], Open Scene Graph or the NVIDIA NVSG provide scene graph based scene management functionality. Being powerful toolkits, scene management and the rendering subsystem are often mixed and difficult to exchange. To circumvent this, the authors of [RGSS09] present a scene graph system which is especially designed to fit to different rendering methods. This is done by allowing the rendering process using a deferred mode - starting the rendering process after scene traversal.

Transformation and Deformation

In garment design it is interesting to attach accessories like shoes or glasses to the avatar wearing a garment. Since avatars exist in different body sizes and forms these accessories have to adapt to the dimensions of the avatar they are attached on. This is not only a simple change of size which can be expressed with linear transformations. What is needed are deformations of these accessories within the scene graph's hierarchy. Candidates for replacing the linear transformation system of a scene graph are presented by Gomes et al. in [GCDV98]. In this book a survey over different transformation techniques is given with focus on warping and morphing techniques. Several of the presented 2D techniques can be easily extended to 3D. An overview over existing deformation and animation techniques is given by Chen et al. in [CCI*05].

Physical simulation often uses deformation techniques to apply the simulation result to a target mesh which cannot be directly manipulated by the simulation. Examples for this approach are presented by Nealen et al. in [NMK*05].

An early deformation technique with focus on proper handling of the surface normal can be found in [Bar84]. Here, Barr presents a group of deformation methods, which additionally allow the computations of proper deformations of the normal. The nesting of several subsequent deformations is presented by Raviv and Elber in [RE99] with focus on free-form sculpting and modeling. Free Form Deformations, presented by Sederberg and Parry in [SP86] allow an intuitive way to manipulate objects with deformation using a control grid. Both methods use local and global deformations to create level of detail mechanisms for modifying an object.

Another application for deformation is the handling and manipulation of models with a high polygon count. A handle based approach for manipulation is presented by Robert et al. [SSP07]. In [EP09] Eigensatz and Pauly present a deformation method based on the manipulation of parts of the surface's properties.

Cage based deformation methods typically surround the object by some kind of control structure which has a lower resolution then the target mesh, simplifying the control process. In these kinds of applications the smooth interpolation between the single control elements is crucial and a variety of methods exist which ensure a certain degree of smoothness. In [LS08] Langer and Seidel propose a deformation method which extends the concept of barycentric coordinates in order to achieve smooth transitions between the deformation elements. In [BPWG07] Botsch et al. present a method which is based on elastic coupling of cells to achieve a smooth transition between user constraints. In order to generate skin deformation on articulated 3D characters, user specified chunks are deformed by using a finite element method to create realistic looking deformations of the articulated mesh as it is presented by Gou and Wong in [GW05].

2.1.2 Surface Related Deformations

Up to now I have presented techniques which deal with direct transformation or deformation of an object in order to handle or manipulate the object. These techniques have in common that they require external information on how and where they should transform or deform a given object. With the number of objects increasing a additional control information needs to be processed and generated. This additional information can slow down the overall process.

A way to avoid this is to attach the deformation target to the surface of another object. This way the surface of the second object acts as the deformation controller, leaving only the required information of where to place it.

A wide range of approaches exists to attach and visualize details to a smooth surface.

Decals

Decal rendering is a mapping method in which the deformation object is a texture patch and the deformation is the local projection of the texture onto a geometry. Mapping requires texture coordinates. However, coordinate computation on a general 3D object can be difficult. A solution for this problem is presented by Schmidt et al. [SGW06] who compute the needed coordinates on the fly using exponential maps. More recent work in this area focuses on the integration of geometry as decal into an existing surface and uses complex methods for fitting the geometry decal onto the surface as it is presented by Schneider et al. in [SGW09].

Decals alone can be seen as stickers which are glued to a place on the surface. Therefore it is useful to be able to calculate surface coordinates which are independent of the surface parametrization. This is beneficial, especially when a decal has to span several surfaces with sudden changes in the surface parametrization. A different approach is used by mapping techniques. They exploit directly the parametrization of the surface in order to texture a complete surface, for example with a repeating pattern. The goal is to have the mapped object (for example classical 2D texture mapping) aligned to the parametrization. In order to map structures beyond 2D textures several techniques exist.

Height-Field Based Approaches

A method for height-field based mapping is presented by Oliveira et al. [OBM00] called relief mapping. They map a texture extended with an orthogonal displacement per texel onto a polygon to render surface details. However, the surface itself stays flat and it is difficult to use the approach in conjunction with current graphics hardware. The approach requires efficient random memory writes, which is known not to be very efficient on current stream-processor based GPU architectures.

Instead, to achieve the same effect on GPUs ray casting steps are typically used for this kind of problem. It requires the use of binary search to find the nearest intersection with the viewing ray [POC05, Tat05]. It is fast and gives a plausible impression of the depth of the structured surface. However, special care has to be taken on borders of a surface and discontinuities. In a later work Policarpo and Oliveira [PO06] extended relief mapping to handle non-height field detail geometry using multiple height field layers. This is faster and more memory friendly as using true volumetric textures as presented by Neyret [Ney98]. However, the use of volumetric textures instead of layers allows the use of more complex geometry with a lot of undercuts. This is a problem similar to depth peeling based approaches as presented by Liu et al. in [LHLW09], which are layer based approaches for rasterization of transparent objects. Common in these techniques is the requirement to convert detail geometry, intended to be glued to a surface, into maps which can be sampled more easily.

Geometry Based

For directly using geometry as a texture other approaches are needed. Wang et al. [WTL*04] present a real-time approach based on five-dimensional Generalized Displacement Maps (GDM). These maps are used to speed up a ray marching process which creates the image of the detail geometry. To perform ray marching, the surface target is resembled by prisms, which are used to map world coordinate rays into surface local space rays. Porumbescu et al. [PBFJ05] use a very similar approach. Instead of using a 5D-map, the detail geometry is traced directly. To achieve this they warp geometric detail inside a tetrahedral cage layer attached to the target surface. Jeschke et al. [JMW07] improve on this by ensuring a consistent warping between neighboring cages. Brodersen et al. [BMPB08] later extended this for implicit representations, allowing advanced attachment modes such as summation and subtraction. This can be used to create the smooth transition between target surface and detail geometry. They also propose a near real-time variant where the detail geometry is represented explicitly. A common property of these techniques is that they are based on direct volume rendering or require ray casting based visualization methods.

There exist techniques which avoid this problem by directly mapping geometry to a surface called Deformation Displacement Mapping. Schein et al. [SKE05] present a real-time implementation of Deformation Displacement Mapping which was developed by Elber et al. [Elb02]. In order to map geometry onto a target surface they compute a position and normal texture from the surface. This step is performed as an offline process while the deformation mapping itself is performed on the GPU.

Aside from detailing surfaces directly garment modeling has another kind of important surface detail. There are techniques especially for garment modeling, which aim to increasing the detail of a simulated cloth surface by adding wrinkles. A GPU-based approach for creating wrinkles on textile materials using deformation was proposed by Loviscach in

[Lov06] for high speed processing of the deformations. A completely different use case is presented by Popa et al. in [PZB*09]. The authors use deformations as a tool to model wrinkles of garments, which have been captured from video frames resulting in a highly detailed 3D capture result. Both methods target the problem of the incapability of low resolution cloth simulations to create fine wrinkles within the garment and are very specialized for this task.

2.1.3 Comparison of Deformation Techniques

In order to discuss each related techniques' capabilities to handle appliqués, I will now describe the intended appliqués and how they are related to the surface of a fabric or garment pattern.

I have categorized appliqués according to the type of attachment on the 2D cloth surface. Roughly they can be split into several attachment types:

- Appliqués, which are attached to one point on the surface. This is often used for hard objects like buttons, but also for many decoration objects.
- Appliqués, which are placed alongside a curve on the surface. This is for example the case for seams, zippers, etc.
- Things like stickers are sewn or glued flat to a cloth surface and resemble an area-based attachment type.
- Objects which mix different attachment types (Ties, belts, etc).

Aside from appliqués there exist objects, which are worn by a person, but are not elements of a cloth's surface. These are accessories like headgears, glasses, shoes or gloves.

In order to judge the technique's capabilities in handling a certain appliqué I have grouped them in Table 2.1.

Along the columns, the table shows techniques arranged by their rough behavior of the approach they use. On rows the appliqué types are listed, grouped by their attachment style and the information how far a technique supports the process.

Looking at the existing methods mentioned in Table 2.1 with the intention to let them handle appliqués one can observe the effects described below:

Texture Mapping Based Methods

Decal mapping is basically normal texture mapping without repeat and special texture coordinates [SGW06, SGW09]. Since fabrics have always a good U/V parameterization the same effect can be generated by performing linear transformations on the texture coordinates. The U/V space of a fabric is usually not orthogonal after draping is performed. This is a difference to the computed coordinates in recent decal mapping techniques. In order to use plastic structures in conjunction with texturing, height fields need to be sampled [POC05, Tat05]. This leads to problems on creasing observation angles. Here, sampling

Appliqué	Type	Decal Mapping	Relief Mapping	Displacement Mapping	Scene Graph	Nested Deformation	Control-based Deformation	Coarse grid/ fine Geometry	My Method
		(Texture Based)		(Geometry Based)			(Free From / Grid Based)		
		[SGW09] [SGW06] [PCCS11]	[PO06] [POC05] [Tat05] [OBM00] [Ney98]	[BMPB08] [JMW07] [PBFJ05] [SKE05] [WTL*04] [Elb02]	[RGSS09] [RVB02] [SRD98] [WHR97] [RH94]	[KKK10a] [RE99] [SP86]	[EP09] [LLCO08] [LS08] [SSP07] [NMK*05]	[YKJM12] [MC10] [TWL07] [BPWG07] [Lov06] [GW05]	
Seam	C	No	No	No	No	Low res.	No	Low res.	Yes
Cloth structure	C	No	Yes	Yes	No	Low res.	No	Low res.	Yes
Hem & Border	C,A	No	Area	Area	No	Low res.	No	Low res.	Yes
Tie	P,C	No	No	No	No	Yes	Partly	Yes	Yes
Bow tie	P,C	No	No	No	Partly	Yes	Yes	Yes	Yes
Belt	P,C	No	No	No	No	Yes	Yes	Low res.	Yes
Cuff buttons	P	No	No	No	Yes	Yes	Yes	Yes	Yes
Decorations	P,C	No	No	No	Yes	Yes	Yes	Yes	Yes
Stickers	A	Yes	Partly	Partly	No	Low res.	No	Low res.	Yes
Headgear & Glasses	P	No	No	No	Yes	Yes	Yes	Yes	Yes
Gloves & Shoes	P,A	No	No	No	No	Yes	Partly	Low res.	Yes
Dress handkerchief	A	No	No	No	No	Yes	Yes	Low res.	Yes

Table 2.1: Comparison of the capabilities of existing techniques in relation to appliqués found in the area of garment design. The type column depicts the attachment type necessary for representing the appliqué. P = Point based, C = Curvature based, A = Area based. The existing techniques allow handling subsets. However, they are specialized to a certain attachment type. This specialization is not present in my proposed method, since it can handle different types of attachment.

rays travel within the height field without hitting the field's surface or even leave the field again, if curved. In conjunction with decal mapping special care has to be taken in order to handle artifacts introduced by the sudden start of the decal.

For the use in conjunction with fabrics, mapping can be used as long as it is possible to compute a coordinate set for the surface it is applied to. This is applicable for area (standard texture mapping) and point based (decals) attachments. However, it is difficult to compute texture coordinate sets for curves. Points of the surface can be defined in several ways at least when the curve overlaps itself. This makes it difficult to use texture-mapping based techniques for representing appliqués which have a curve-attachment type. Additionally, geometry applied to the surface has to be converted into a set of height fields in order to be usable in conjunction with mapping-based methods. Complex geometry with fine details require a high number of height fields with a high resolution in order to capture all details. This increases the computation time and memory consumption drastically. The process is an imaging process - the surface details are only visible in the rendering process and do not generate new geometry.

Additional methods exist which compute their own U/V parametrization. This is necessary, when a given U/V parametrization is not sufficient or nonexistent. An example for this kind of techniques are exponential maps [SGW06]. Automatic creation of a 2D parametrization for a given 3D object can be tricky when models get complex. A way to do it with the help of the user is presented by Pietroni et al. in [PCCS11]. However, in the scope of garment design the algorithms used should either automatically generate coordinates or, in an optimal case, work with the coordinates provided from the sewing patterns.

Geometry Mapping Based Methods

Another approach is to directly use the detail geometry within the mapping process. This way sampling or layering problems as they exist in a height-field based approach are prevented and the detail geometry can be used in full detail. However, the needed computational effort is high, since these techniques rely on tracing or pre-computation.

Examples for direct rendering of displacement maps are proposed by Wang et al. [WTL*04]. Porumbescu et al. [PBFJ05] and Jeschke et al. [JMW07] use a cage for the deformation based on the surface grid. The use of implicit surface definitions is possible, too. This way intersection operations with the surface are possible as it is presented by Brodersen et al. [BMPB08]. The techniques are a mix between displacement mapping and grid based deformation.

The common property of these approaches is in the intention to be used for texturing of a surface with geometry. For this fast texture-like usage of surface details Koniaris et al. present a survey and comparison of existing techniques in [KCYM14].

The analog to decals for detail geometry can be seen within scene graphs (as mentioned earlier [WHR97, RH94, SRD98, RVB02, RGSS09]). The hierarchical arranging of objects can be used to place objects on an other object's surface. However, this is intended for static use or animation of the linear transformations within the scene graph object hierarchy. There is no direct relation to a certain surface point. Therefore, in order to use this technique for surface objects attached to animated surfaces an update process is needed. The required computational effort for the transformations is very low. However this does not include the updating process necessary when geometry is deformed. This can be a bottleneck if large quantities of objects attached to a surface need to be handled.

Free Form and Grid Based Deformation

The above mentioned techniques allow handling area and point based attachments. In order to freely place detail geometry onto a surface on a curve type attachment more control is required. Deformation techniques for handling highly detailed models can be used to solve this problem [EP09, SSP07, NMK*05, YKJM12, MC10, TWL07, BPWG07, Lov06, GW05]. Basically, they can be used to perform all types of mappings. However, they suffer the same limitation as it is present for the scene graph. Since a control structure is used, this control structure needs to be updated every time the geometry they are attached to is deformed.

I note that the prism based approach introduced in shell maps and GDM could be used as a type of cage based deformation system for the detail geometry similar to the ones described in the previous section. Such an approach could even be very fast, but require a very smooth interpolation function. These interpolation functions are typically considered the holy grail of cage based deformation. Examples are high order barycentric coordinates [LS08] or Green coordinates [LLCO08] to create smooth transitions. However, all of these algorithms are much more complex and would reduce the performance by a large degree compared to prisms with hard boundaries.

2.1.4 Surface Processing

Aside from surface details which can be described by additional detail geometry, the fabric surface itself can undergo modification. This can be the case for example when fabrics are sewn tightly together. In this case dents will appear near the seam on the surface, since fabrics are compressible to a certain degree.

In order to show these details the surface of the garment needs to be processed. Height-field based surface geometry displacement techniques are well suited to add these kinds of details to a surface. However, since this involves sampling at the vertices of the surface geometry, a refinement of the surface geometry is required to have enough sampling points.

Here, I focus especially on local refinement techniques. Local refinement is well suited for shader-based approaches since it only operates with data provided by a single patch of the control mesh.

A refinement process can be roughly divided into three stages. First, a patch setup is performed. The patches describe the surface curvature. They are controlled by the input mesh, which is to be refined. Second, a tessellation logic is required, which divides the basic input mesh into a finer representation, creating interpolation coordinates. The third stage uses these coordinates in conjunction with the patch data to compute the positions of the new vertices.

Today, aside from software refinement and compute shaders, there are two ways to perform the logic for the refinement in GPU Hardware.

The first way is to use hardware tessellation. In the rendering API's of DirectX 11 and OpenGL 4.0 there are special shaders which directly resemble these stages. Both API implement a similar mechanism. A description of this process inside OpenGL can be found in the specification of OpenGl 4.0 [www10]. The new stages allow a direct hardware tessellation. It resides between the vertex processing unit and the triangle rasterizer. To allow a high flexibility, the stages need a new primitive type. This type is patch based and allows a definable number of vertex indices per patch. The tessellation logic itself is a configurable fixed function stage.

The second alternative approach is to use the GPU's geometry shader stage that allows us to create a flexible, specialized tessellation engine, which can be adapted directly to an existing rendering system with only few modifications. This shader type is more common, since it was introduced already in the previous shader model [Bly06] (see Figure(2.1)). It was improved with the update from DX10 class hardware to DX11 class hardware. In addition to speed improvements the output buffer size was increased from 512 output elements to 32k output elements. This alone allows 64 times more complex geometry. Hardware

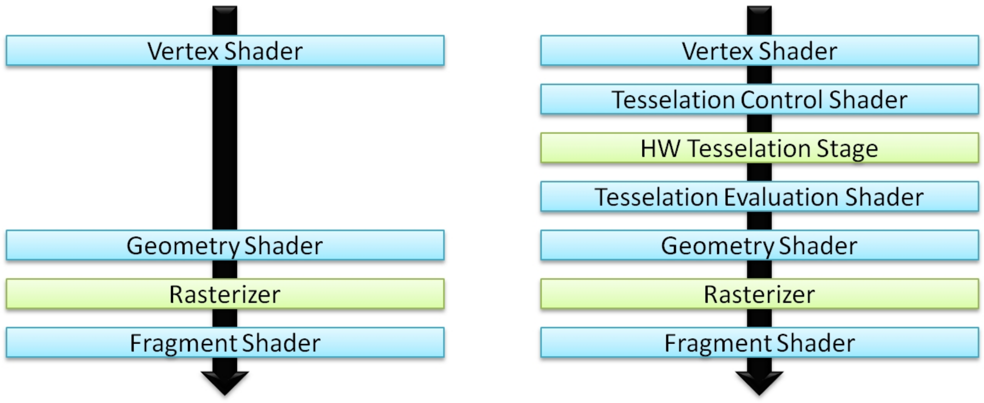


Figure 2.1: DX10/OpenGL3.2 (left) versus DX11/OpenGL4.0 (right): With DX11 3 new shading stages are available. Tessellation can now be achieved within the three new stages or by the geometry shader. Green blocks mark fixed function stages, while blue blocks are freely programmable.

tessellation is not necessarily superior to a geometry shader based approach (see Concheiro et al. [RMM10]). A geometry-shader based tessellation has a higher flexibility than the hardware-based method, since the generated topology can be chosen freely.

The new stages now allow to handle surface refinement on a GPU render pipeline level. In [LSNC09] Loop et al. describe a technique for approximating subdivision surfaces with Gregory patches. Their technique allows the approximation of Catmull-Clark subdivision surfaces directly on the GPU. Additionally they give an overview of the hardware tessellation unit in DX11 class graphics hardware.

Subdivision surfaces themselves create very smooth surfaces for a given mesh. However, they require special mesh structures and index sets in order to be quickly evaluated, ideally using recursion. Therefore, in order to do this on the GPU patch based approximations are used, since these can be evaluated without recursion.

Aside from the smoothing process itself, in order to integrate surface details my main goal lies in the increment of the resolution. Since the fabric model usually already has a smooth appearance due to low pass filtering (damping) within the physical simulation process of the fabric, the smoothness of the surface comes in second place. For this reason I especially focused on local triangle refinement techniques.

Triangular Patches

There exist several approaches regarding smooth surface patches usable for local triangle refinement techniques. The patches represent the curvature described by the control or input mesh sent to the refinement process. Unfortunately, position and normal are the only curvature data source available for patch setup in my situation.

A fast technique allowing a patch setup under these conditions is Phong tessellation [BA08]. It is a technique which uses a triangular patch of second order in conjunction with phong-shading to create a smooth surface. To work properly it needs a sufficient tessellated base mesh, since it cannot handle infliction points sufficiently.

Without inflexion limitation, triangular patches of cubic degree can be used to create smooth surfaces. A first approach for using the triangle mesh data directly as control mesh is called curved pn triangles and is presented in [VPBM01]. To provide a control mechanism for creases by additional vertex data, Boubekeur et al. [BRS05] extend this technique. The methods rely on cubic triangular Bézier patches. Since the center control point of the Bézier patch is shared by all 3 edges of the patch, geometric continuity of degree one (G1) can only be reached at the vertices of the patch. A technique promising a nice G1 continuity among patch borders is presented in [FMHF08]. It combines several triangular patches of cubic degree to grant continuity on the edges and blends them by a rational function to build the final surface. Unfortunately, this inspiring technique needs data from neighbor triangles for parameter computations. There is a representation for such adjacency information available for the GPU's geometry shader, but it replaces the given index set of the base geometry.

Tessellation and Adaptive Refinement

Early GPU based refinement approaches were limited to the vertex and fragment shader. The GPUs used did not contain stages for direct mesh manipulation. Thus, it was necessary to output the new geometry data in the form of embedded textures, to be able to process it. A theoretical framework for GPU based refinement is presented by Shiue et al. in [SGP03]. A list of approaches utilizing the GPU as a general purpose processing unit for refinement can be found in the work of Boubekeur and Schlick [BS05].

A first approach on tessellation inside consumer graphics hardware was introduced in 2001 by ATI inc. The implemented technique internally uses curved pn triangles [VPBM01]. The achieved refinement has a uniform topology and allowed a linear increment of the number of edge vertices. A uniform refinement is fast to compute. On the other hand it does not allow the change of the refinement level within the same geometry without topological inconsistencies.

Nowadays, with the existence of geometry and topological shader stages, mechanisms exist which simplify the process chain for adaptive mesh refinement on the GPU. The data necessary to represent smooth transition topologies between the different refinement levels for adaptive triangle refinement can be pre-computed. These topologies can be stored inside a lookup table sent to the GPU as geometry data. The Adaptive Refinement Kernel (ARK) technique presented by Boubekeur and Schlick in [BS08] uses this technique. Another similar method using adaptive topological patches called Dynamic Mesh Refinement is presented by Lorenz and Döller in [LD08]. The patterns describing the necessary topology have to fit all possible combinations of refinement levels. Thus the number of topological patches can be very high. This problem is reduced in the work presented by Lenz et al. in [LCNV09]. Here, a permutation technique is used to reduce the overall number of patterns necessary by permutation of the barycentric coordinates used for describing positions in the patch surface.

These methods rely on large topology catalogs in order to deal with the different combinations of refinement levels on the patch edges. The topological patches collected in these catalogs allow a high flexibility for choosing refinement patterns. Additionally, they can be computed in a pre-processing step. As a drawback the patterns use a lot of memory space and have to be addressed. Thus these algorithms usually perform several drawing passes until the final refined geometry is drawn.

An alternative to a catalog of topological patterns is presented in [DRS09]. The technique is a topological consistent extension to the work presented in [DRS08]. Their edge based refinement technique uses a simple mesh topology based on dyadic uniform triangle meshes. At first a uniform mesh is chosen representing the highest refinement level provided on the edges. On edges with lesser refinement level the vertices are snapped to the lower refinement position. This grants a watertight topologically consistent patch. As a drawback the snapping on the edge can create sudden resolution changes since only the vertices at the edge are moved.

GPU tessellation aims not only at creating smoother meshes. It can be used to reduce the necessary bandwidth of a 3D application. Especially for mobile devices this is an issue. In [CYKK09] Chung et al. address this problem and present a shader based solution for mobile phones. In addition they discuss the differences between the ARK technique and the HW tessellation system of DX11 in detail.

2.2 Rendering

After processing and assembly of the final geometry, the result needs to be visualized. The problem I target consists of three parts. First, a model based on a bidirectional reflectance distribution function (BRDF) is needed which is capable to handle various material effects. Second, natural illumination and directional occlusion of the scene are required. Third, in order to be used in combination with a real-time simulation system, everything related to a simulated geometry needs to be computed from scratch in each frame.

Especially, the computation of directional occlusion can be very time consuming if computed in high quality. To target this problem, I have researched hybrid occlusion techniques, which combine techniques specialized in low frequency and high frequency lighting.

Visualization, Lighting, and Occlusion

The technique I present belongs to the category of natural illumination techniques, where a scene is illuminated by a distant high dynamic range light source. A good overview of the different methods in this area is given by Ritschel et al. [RDGK12]. Their state-of-the-art report provides a good survey over current global illumination techniques. Using images of environments in order to illuminate objects has a long tradition in computer graphics.

Environment mapping allows storing directional lighting information in an efficient way. If shadows are not necessary, the scene can be directly illuminated by an environment map. For this task fast environment mapping methods (like [Gre86, RH01, HS99]) exist. The technique of environment mapping was originally introduced by Blinn and Newel [BN76]. Environment mapping techniques can be used to simulate a wide set of phenomena in real-time computer graphics. An overview of the most common techniques is given by Akenine-Möller et al. [AMHH08].

Environment maps can be used in conjunction with high dynamic range (HDR) images in order to capture natural illumination. Debevec [Deb98] presented an acquisition method for such images and how they can be used to render scenes under very realistic lighting conditions. This process is called image based lighting. The tricky part for image based lighting is the integration of self-shadowing of the scene. Contrary to Ambient Occlusion (AO) [SZ98] the environment is not a homogeneous light source but colored, which makes the incoming light dependent of its direction.

Since this technique does not support directional shadowing or occlusion (DO) directly, it can be necessary to compute a carefully chosen set of light sources to approximate the lighting effect of a light probe. For real-time applications this can be done by a set of lights generated by an importance sampling of the environment map. Examples for this are given in [HSK*05, Deb08, ADM*08]. To get an effective set, several methods exist, which perform an analysis of the distribution of light in the light probe with metrics and generate directional lights according to it. This results in a better distribution of lights, compared to a uniform distribution [ARBJ03, KK03, ODJ04, SSSK04]. In order to raster scenes with a set of light sources generated by importance sampling the use of real-time shadowing techniques is required. A survey of real time-shadowing methods is given by Hasenfratz et al. in [HLHS03]. Some of the algorithms presented work with small area light sources, but can be computationally costly if these lights become large. Fast shadow calculations for simple point or directional lights within dynamic scenes can be done inside modern GPUs. Mainly two different approaches are used: Shadow volumes presented by Crow in [Cro77] and shadow maps, presented by Williams in [Wil78]. Shadow maps are fast to compute and need less fill rate than shadow volumes. On the other hand, care has to be taken of sampling artifacts. Approaches for minimizing shadow buffer artifacts and softening of the borders of shadows can be found in [RSC87, DL06].

Shadow based techniques can be extended to even perform fast global illumination computations as it was presented by Keller in [Kel97]. He uses single shadow casting lights to illuminate the scene in an implicit way to speed up the process. In generalization these methods are called many light rendering approaches (see Ritschel et al. [RDGK12]).

However, all current approaches in this area do not perform well with highly glossy surfaces [HSK*05, RDGK12]. The reason for this is that the all-frequency information of the environment map is substituted with a compressed one. Using highly glossy surfaces will reveal the compressed information, which is given by single light sources. Annen et al. [ADM*08] reduce this problem by using area light sources. However, these light sources still represent one constant value over an area, not revealing the full environment map information in case of highly glossy surfaces.

The alternative is given by methods that can be applied to handle directional lighting information directly. The represented occlusion or lighting data is then used to process the environment map. These methods need to store directional data in a way which allows fast folding computations between the environment map and the occlusion map. A method often found in this area is the one called Spherical Harmonics, which covers a wide area of visualization problems related to storing information into a direction-related way, see for example [SKS02, NRH03, SHHS03]. The type of data used often requires pre-computation. This is problematic when using animated geometry. This problem is addressed by Kautz et al. in [KLA04]. By using model hierarchies they accelerated the calculations to interactive frame rates. Their approach is limited to small model sizes and low frequency shadows. In addition model hierarchies are costly to compute. A method with similar results is presented by Sattler et al. in [SSZK04]. It evaluates the visibility of a given set of directional light sources per vertex, taking into account the color of the environment map, by sampling the map in an uniform way. Visibility calculation is done inside the GPU, and is then read back by the CPU for occlusion computation.

A look on the ray tracing side, shows that ray tracers are well suited to compute occlusions [WPS*03, WBS03, PBMH02]. Nowadays, real-time frame rates can be achieved by using a PC-cluster, the GPU or dedicated ray tracing hardware, which is still in development. Software solutions require the power of an efficient cluster system and at last GPU ray tracing.

By utilizing a real time ray tracer global illumination can be simulated. In [CHL04] a direct approach towards global illumination on the GPU is presented, which works for small scenes. However, it is still required to perform folding computations between the environment map and the occlusion.

Recently, screen space methods have been developed which address ambient and directional occlusion computations. Screen-Space Ambient Occlusion (SSAO) covers methods and algorithms to solve the occlusion integral [SZ98] in screen-space using texture sampling. Working in screen space promises to be independent of the scene complexity. Screen space ambient occlusion was first discussed by Mittring in [Mit07] and used in the video game Crysis. The basic idea of the approach is to compute AO from the depth information contained inside the rendered frame. To achieve this, the screen content is treated as a height-field for which self-occlusion is approximated. Since its introduction in 2007, research in this field has led to numerous extensions and contributions in order to improve the quality, performance and usability. To improve the stability of the process Shanmugam and Arikan [SA07] extend the technique to incorporate distant occluders not visible on the screen. Horizon-based Ambient Occlusion by Bavoil et al. [BSD08] improves the shadow quality by ray marching in image-space. To cover larger sampling kernels while still maintaining high performance, Huang et al. [HL10] presented a multiresolution algorithm (MRSSAO) at the expense of texture memory. An extension towards directional occlusion (SSDO / MRSSDO) is presented by Ritschel et al. [RGS09]. Such techniques mainly rely on depth information

Requirement	Environment Mapping [Deb98] [BN76]	Filtered Env. Mapping [KM00] [KVHS00] [HS99]	Many Light Occlusion [DR08] [Deb08] [HSK*05]	Path Tracing [JC07] [WPS*03] [PBMH02]	Screen-space Occlusion [BKBK13] [KRES11] [RGS09] [Mit07]	Fast Shadow-based [DGR*09] [ADM*08]	My Approach
Dynamic Geometry	Yes	Yes	No	No	Yes	Yes	Yes
Image Based Lighting	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Scene Level Occlusion	No	No	Yes	Yes	No	Yes	Yes
Detail Level Occlusion	No	No	Partly	Yes	Yes	Partly	Yes
RT@High Polygon Count	Yes	Yes	No	No	Yes	No	Yes
Single Lights	No	No	Yes	Yes	No	Yes	Yes
Material Approximation	Partly	Partly	Partly	Yes	Partly	Partly	Yes
Arbitrary Ref. Model	No	Partly	No	Yes	No	Partly	Yes

Table 2.2: Capabilities of various techniques for the use of visualization for virtual prototyping of garments.

of the current view for their computations. This is problematic, since cavities are ignored, which are hidden in the current view. Thus, the whole scene does not contribute to the occlusion and the deviations from a ground truth differ more and more for the low frequency shadowing effects. However, SSAO techniques use the screen depth information to approximate the darkening. This does not directly cover cavities hidden by the current view.

In [VPG13] Vardis et al. present a technique, which additionally uses the depth information present inside the shadow buffers of the present lights of a scene. This way they get additional views of the scene which allows them to fuse multiple SSAO computations into a single image.

Another approach for taking into account the whole scene is presented by Thiedemann et al. [THGM11]. They use a voxel representation of the scene in order to perform fast global illumination computations. A drawback is the high amount of memory necessary to store the voxel representation and the additional time needed for geometry voxelization. Another mechanism for fast ambient occlusion computations is presented by McGuire in [McG10]. Instead of voxelization, polygonal volumes are built based on the scene geometry. This is done inside a geometry shader. These volumes represent the influence of the occlusion. As a geometrical approach, the technique is cost effective, but requires processing of the scene geometry.

2.2.1 Comparison

I will now describe the differences of the single image based lighting methods and their capability to handle the requirements for virtual prototyping of garments. Table 2.2 lists requirements, existing real-time rendering techniques and which parts they can cover.

(Filtered) Environment Mapping

Classical environment mapping is one of the first implementations of image based lighting [BN76]. It is used to handle reflecting objects by providing a direct look up for surface reflection vectors. In the basic approach the scene will not be seen in the reflection, since the map only contains the environment.

In order to extend the range of effects usable with environment mapping filtering can be applied [HS99]. Filtered environment mapping is an extension, which allows to incorporate BRDF data in the look up by integrating the reflection properties into the environment map, creating a new environment map [KVHS00, KM00]. Instead of using the reflection ray, sampling rays need to match the BRDF's function. An example for this behavior is irradiance mapping, which is generated by integrating diffuse reflection into the environment map. The sampling rays in this case are given by the surface normal [AMHH08]. However, this direct method is limited to anisotropic BRDFs only. Since the BRDF is integrated into the environment map it will always represent the appearance of the surface for the given environment and BRDF. In order to represent different BRDFs, additional maps need to be stored, one per BRDF.

Methods exclusively based on environment-mapping are very fast at rendering time, since it is a simple texture lookup. They can be used in conjunction with several surface reflection phenomena by using pre-filtering. In conjunction with High Dynamic Range (HDR) images environment maps can be used as a light source for natural lighting environments as presented by Debevec in [Deb98]. Since the support for material properties relies on pre-filtering, it is difficult to include self-shadowing of the scene. A solution to this is given by using ambient occlusion for the self-shadowing, ignoring the color of the environment in the shadowing process.

Better results can be achieved by using directional occlusion, which uses the information of the environment map in the shadow computations, too.

Many Light Occlusion

Directional occlusion can be implemented by several ways. [DR08] [Deb08] [HSK*05] They all consist in using a fast way to compute convolutions between the environment map and the directional occlusion.

For static scenes this can be done by using the frequency domain, like it is done when using spherical harmonics, for example [KLA04]. Techniques exist, allowing for faster computation of the parameters. However, they are limited to low frequency computations with small parameter sets.

Another, more direct way is to merge suitable neighboring areas into one entity and use this in conjunction with classical lighting and shadow computations. This is the same mechanism as it is used in many light approaches used for solving global illumination. The difference is in the light distribution. So called many light approaches use point light sources, which are distributed inside the scene in order to simulate the light transfer between single regions. For directional occlusion these lights are moved outside to the infinity shell [DR08,Deb08,HSK*05]. The light sources need to be distributed over the environment map. Aside from a equidistant mapping several importance sampling methods exist. They allow creating higher light source densities at important points on the environment map.

For diffuse applications high rendering speeds can be achieved. Light source based approaches do not perform well with highly glossy surfaces. These tend to reveal the structure of the single lights. This can be reduced by using area light sources instead, which are more complex to compute and difficult to handle in shadow computations.

Path Tracing

Ray-tracing based approaches easily generate high quality images using directional occlusion and environment mapping [WPS*03]. With path or multi-sampling techniques complex surface material can be evaluated [JC07]. The computation effort to generate images is very high, since the convolution is basically solved by sampling the map. This is still problematic when using fast or real time ray tracing approaches [PBMH02].

Screen Space Based Occlusion

Screen space occlusion techniques manage to decouple scene complexity and occlusion computation which is a very interesting feature. The techniques are very fast and independent of the scene's complexity [Mit07]. Screen space is limited to the scene visible within the view. In order to reduce artifacts additional information can be used. However, computational effort is increased. The screen space techniques require area sampling. Sampling is always costly. Multi-resolution approaches exist to speed this process up [HL10]. However, multi resolution requires multiple passes increasing the computational effort. Screen space occlusion is fastest, when only a small area around a feature causing occlusion is observed. Therefore, the technique is well suited to be used to shadow high frequency details within the scene. Additionally, with larger filter kernels the overall restrictions of the technique get more and more visible.

The inverse to screen space techniques are geometry only based techniques. Direct geometry based techniques generate a lot of geometry for creating the AO effect. Additionally the method requires a high fill-rate for large filter kernels, which makes it less suited for high detail geometry.

An interesting mix are methods which perform AO computations based on a voxelspace from the scene [BKBK13]. This can be completely performed on current GPUs. However the quality of the result is dependent of the voxelspace's resolution. The technique is best for being used for low frequency AO in conjunction with low detail geometry.

Fast Shadow Based

The techniques presented by Annen et al. [ADM*08] and Dong et al. [DGR*09] rely on the capability of modern GPUs to be able to compute a lot of shadows in real time. Therefore it is sufficient to reduce the amount of light sources to a necessary minimum and then do brute force shadow computations in real time. Aside from tricky shadow computations, which need to be done for area light-sources, one major drawback of this technique is its limitation to mostly diffuse surfaces. This is due to the low number of light sources, which would be revealed quickly when dealing with glossy materials.

2.3 Summary

In this chapter I have presented related work regarding geometry processing and visualization. This reflects my overall approach's logical split in the conceptions for the geometry processing on one hand and for the visualization part for the other. While the goal of the geometry processing is to assemble a final detailed 3D model, the purpose of the visualization is to apply optical material properties and self-occlusion under natural lighting conditions.

For the geometry processing, my focal point was set to techniques which allow processing of geometry objects in relation to surfaces and their their surface refinement. These techniques where compared in relation to the originally stated problem of attaching appliqués presented in Chapter 1. For the visualization part I first focused on rasterization methods for image based lighting in conjunction with directional low frequency occlusion. For high frequency occlusions I presented work related to image based occlusion, which is good at handling occlusions of fine detail geometry. This was followed by an analysis on how these techniques can be used to solve the problem of real time rendering for highly detailed garments in conjunction with deformation and modification.

However, in difference to my approach, all these techniques are not directly intended to be used in conjunction with each other. The results are differences in data structures used and the overall approaches (for example geometry generation versus direct tracing of the surface).

On the geometry side, I searched for an algorithm which works similar to the one presented by Schein et al. [SKE05], by directly generating geometry based on a target surface. However, my approach is not limited to simple mapping of a mesostructure to u/v coordinates, but in a more controllable way to handle all kinds of attachment types as described in Chapter 1.

On the visualization side, self-occlusion is mandatory when dealing with garments. However, geometry-based self occlusion using shadow calculations gets computational expensive when dealing with a lot of geometry.

How I deal with the above mentioned limitations and the overall concept and details of the single parts of my solution are described in detail in the next chapter.

3 Concept

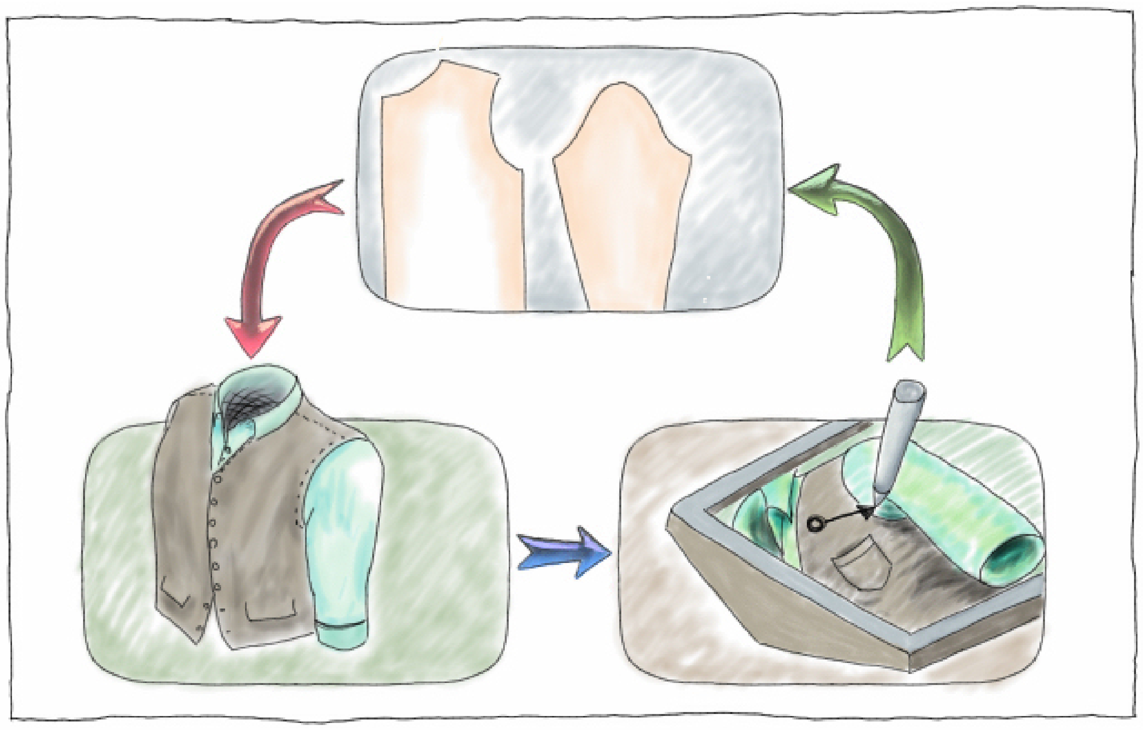


Figure 3.1: The kind of operation a designer performs can be depicted as the well known model-view-control pattern. The Model (top) is in this case the 2D sewing patterns. The View (left) is the 3D Presentation of the virtual garment and the Control (right) the interaction and operations the designer performs.

When using virtual prototyping of garments a designer needs to work and interact with the garment in real time. The type of interaction is a classical model view control pattern (see Figure 3.1). In order to guarantee producibility of the final product, all operations are related to the flat 2D sewing patterns. From these patterns the virtual garment is generated/updated and visualized. The designer defines operations on the garment, which are translated back into 2D pattern space of the sewing pattern information.

The main problem of the existing techniques in this area is that they can address only single facets of the problems arising with virtual prototyping of garments. Additionally, the way how parts are solved cannot be combined to a full system without massive drawbacks. Building the concept for the whole process was challenging. On one hand a high amount of

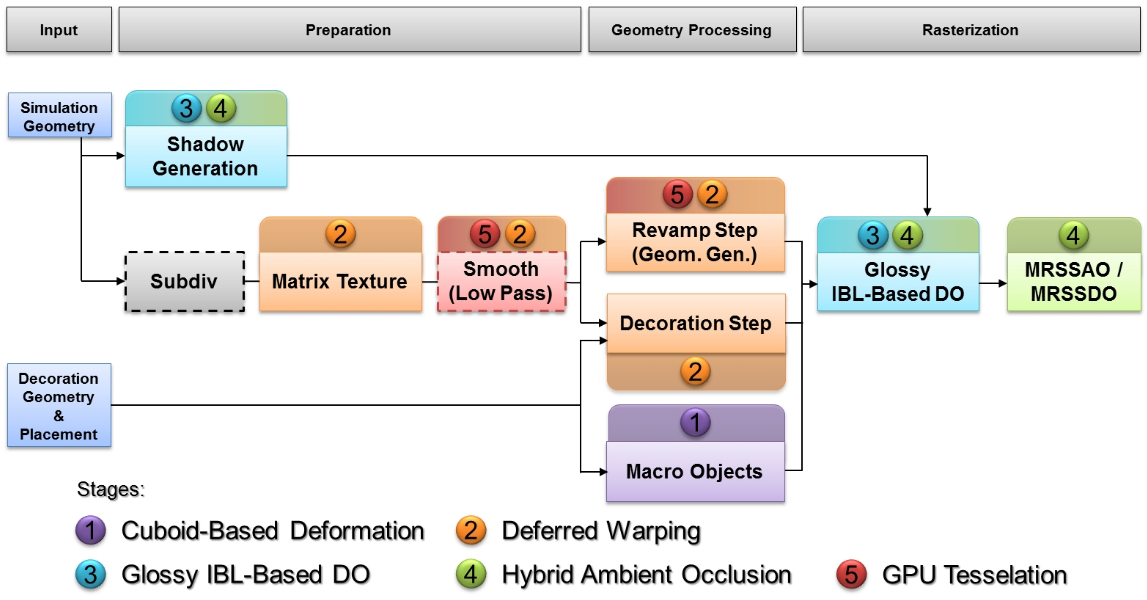


Figure 3.2: Overview of single steps within my proposed approach for a realistic representation of appliqués within real-time virtual prototyping systems for the garment industry. The colored dots depict the single techniques I use and where they are located inside the process. The acronyms used in the two rightmost blocks stand for image based lighting (IBL), directional occlusion (DO), multi resolution screen space ambient occlusion (MRSSAO) and its directional occlusion counterpart (MRSSDO).

detail is required to satisfy the needs of designers. On the other hand techniques for example for creating plausible self occlusion and lighting effects are needed. The latter contradict the use of highly detailed geometry, since a lot of geometry slows down the process of shadow computations or occlusion, if the shadow computation is geometry based.

I present a conception for detailing and visualization of the virtual prototype, located directly after simulating the cloth. My approach combines techniques in a way which contradicts the weaknesses of the single techniques. It can be roughly split into four sections - Input, Preparation, Geometry processing and Rasterization. An overview of the of the single parts of my concept is shown in Figure (3.2).

Input

The input for the process is given by the current geometry frame from the simulation process and information regarding the decoration's geometry and placing.

Preparation

During preparation step computations are performed which prepare later processes like shadow map computations and the matrix texture used for deferred warping.

Geometry Processing

For Geometry processing I propose a split approach:
The first part allows to place and customize appliqués related to the used avatar wearing the virtual garment. The placements can be performed hierarchically. The outcome is similar to a scene graph but additionally allowing complex deformations of the objects placed.

The second part places surface appliqués directly using the information provided in 2D sewing pattern space. It allows to freely choose the type of attachment to the surface. In relation to the existing approaches I can state three main differences:

First, the presented methods with the exception of [SKE05] use a representation that requires ray-tracing techniques to render the detail geometry or approximated data. In contrast, my method (orange stages depicting the use of deferred warping in Figure 3.2) as well as the one of Schein et al. [SKE05] simply render the geometry in the same way the target geometry is rendered using arbitrary rendering systems including rasterization. Both methods can also post-process the attached geometry within the rendering pipeline as needed. I show an example with further deformation later on in Section 3.1.2

Second, in contrast to [SKE05] and all other methods my approach defines a complete orthogonal reference frame at every vertex position. Having a full reference frame has the advantage to allow support for advanced attachment types like point and curve attachments. Note that these types do not sheer the detail geometry even in the case of a bad U/V parametrization. In previous works the attached geometry is always mapped directly to the target surface and will undergo bending when the surface is deformed.

Third, I need no offline pre-processing step since my matrix textures are computed on the fly. This way detail geometries can be attached to animated or manipulated surfaces. Moreover, the contained matrices can be modified in this process which provides more flexibility to the user of the approach.

Rasterization

Having both control over the detailing and the visualization process allows me to feed different levels of detail at different sections inside the visualization pipeline (see Figure 3.2).

Level of detail is a common method to reduce rendering complexity often found in computer games to reduce the complexity of objects in the distance. I combine this with the idea of imperfect shadow maps [RGK*08], which uses the observation that the accuracy of a single shadow is quite unimportant when used within a many lights approach.

My concept of a visualization system for virtual garment prototyping capable of handling surface details combines both approaches. I use the observation that the surface details added by the appliqués are tiny and do not contribute much to the overall low frequency self-shadowing of the scene. Therefore, the full detail is not needed for geometry based shadow computations and thus, the stage "Glossy IBL-Based DO" in Figure 3.2 can work with low detail geometry for the global shadows. The fine details are later on added within a screen-space stage called "MRSSAO/MRSSDO" in Figure 3.2. It is optimized to create the high frequency occlusion caused by the detail geometry using screen-space based methods.

For generation of the low frequency directional occlusion I use a many light approach with light sources generated from an HDR environment map. Typically, these approaches suffer from accuracy when highly glossy surfaces are used. However, my approach circumvents this problem by using the original environment map in the illumination computation. This way the original all-frequency information of the light probe can be used. To support the usage of shadowing techniques, e.g. [DL06, AMB*07], I substitute the environment map by a set of light sources. Annen et al. [ADM*08] show that a shadowing technique requires soft shadowing and a fitting penumbra computation in order to obtain a smooth transition between the shadows of the single light sources. Various soft shadowing techniques for real-time rendering exist, which have all their advantages and disadvantages. A survey of suitable methods is given in [KS14].

Approach Details

I will now present the details of the components of my concept. Again, there is the split into the two major function blocks:

First, I will present a real-time geometry processing stage (methods one, two and five in Figure 3.2), in which geometry from the simulation process is modified. This stage handles all objects which are related to the surface of the cloth like appliqués, seams or structures of the fabric itself. Additionally, it includes the projection of appliqués onto the garment's surface based on their location on the 2D cloth surface. This stage is described in section 3.1.1, section 3.1.2 and section 3.1.3.

Second, the real-time visualization stage combining image based lighting, directional occlusion and a material system using phong lobes. It is the key to give designers live feedback of what they are doing on the model. Again the split into a base model and the appliqués allow a distinction between coarse and detailing geometry. For the coarse geometry lighting I present a self shadowing and material system method in section 3.2.1 (method three in Figure 3.2). The detailing appliqués are lit the same way, however detailing shadow is then taken into account using a hybrid approach presented in section 3.2.2 (method four in Figure 3.2).

3.1 Geometry Processing

3.1.1 Macro Objects - Cuboid-Based Deformation

My concept starts with a description of the scene management I have developed in order to deal with the garment scene to be rendered. I use a cuboid based scene graph I have presented in [KKK10a]. The cuboids replace the linear transformation of an ordinary scene graph and additionally allow local low frequency deformation of the objects in the scene. This way I handle macro objects like shoes, glasses or hats in order to make them fit a given figurine.

The scene graph systems presented in section 2.1.1 do not take deformation into account as a low level transformation process, the aforementioned publications dealing with deformation instead focus mainly on animation and modeling aspects. They leave out the possibility of handling hierarchical deformations in conjunction with object placement in 3D scenes for real time applications. Even though the idea of using a hierarchy of deformations is not new [SP86] [RE99] the requirements posed by the replacement of a real time scene graph's transformation system are high. However, it is important to be able to compose non-linear transformations over the scene graph hierarchy within a real time system. It adds the functionality to not only group objects, but align these groups along curvatures, thus increasing the scene graph's functionality to some degree. In this section I show my approach to embed deformation into the scene graph of a real time rendering system.

For using cuboids as a scene graph for transformation, the linear transformation system is replaced by trilinear transformations.

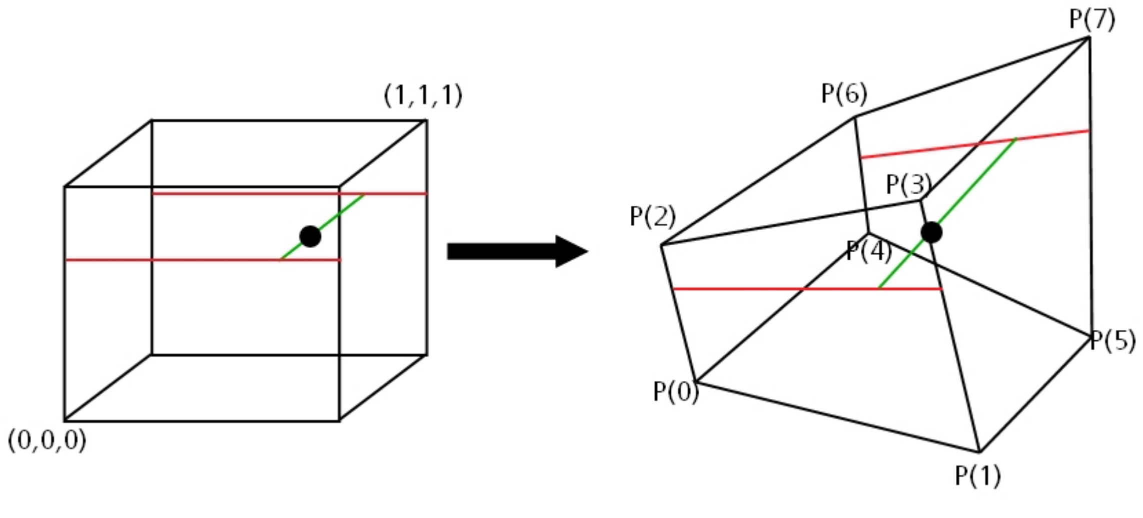


Figure 3.3: Trilinear transformation: a point within a unit cube is transformed by using its three coordinates as coefficients for a trilinear interpolation between the eight corners of the cuboid.

Trilinear Transformations

As described in [GCDV98] trilinear transformation is an extended version of bilinear transformation. It is a function, mapping points of $\mathbb{R}^3$ to $\mathbb{R}^3$ defined by 8 points forming a cuboid, see Figure 3.3.

In contrast to linear transformations each corner point of the cuboid defines its own coordinate system. It is created by the three adjacent corner points. A hierarchy of these transformations describes a hierarchy of local subspaces within a world space. This is completely different to a linear system, which would describe a hierarchy of local coordinate systems within a world coordinate system.

Vertex Transformations

In general there are two methods to perform the trilinear transformation of a point in space. The first one uses the coordinates of the cuboid directly to transform vertices, the second one uses a polynomial representation allowing fast transformation of many vertices. Additionally, the necessary coefficients allow for a simple test to detect whether the transformation represents a parallelepiped or a real cuboid.

The first method uses the corner points $\vec{p}_0..\vec{p}_7$ of the cuboid (see Figure 3.3) and the transformation T, defined as

$$T(x,y,z) = \begin{pmatrix} \vec{p}_0 \\ \vec{p}_1 \\ \vec{p}_2 \\ \vec{p}_3 \\ \vec{p}_4 \\ \vec{p}_5 \\ \vec{p}_6 \\ \vec{p}_7 \end{pmatrix}^T \cdot \begin{pmatrix} (1-x)(1-y)(1-z) \\ x(1-y)(1-z) \\ (1-x)y(1-z) \\ xy(1-z) \\ (1-x)(1-y)z \\ x(1-y)z \\ (1-x)yz \\ xyz \end{pmatrix}. \tag{3.1}$$

Here x,y and z are the coordinates of a point to be transformed. The coordinates have the range [0..1]. This equation can be used to directly transform a point. A nice feature of this approach is the direct usage of the image (cuboid) of the transformation.

However, with multiplication and sorting by x,y,z I get the second method which is a transformation process using a *polynomial representation*, where a coefficient 8×3 matrix

$\mathcal{C}$ is created from the corner points of the cuboid:

$$\mathcal{C} = \begin{pmatrix} (\vec{p}_7 - \vec{p}_6 - \vec{p}_5 + \vec{p}_4 - \vec{p}_3 + \vec{p}_2 + \vec{p}_1 - \vec{p}_0)^T \\ (\vec{p}_6 - \vec{p}_4 - \vec{p}_2 + \vec{p}_0)^T \\ (\vec{p}_5 - \vec{p}_4 - \vec{p}_1 + \vec{p}_0)^T \\ (\vec{p}_3 - \vec{p}_2 - \vec{p}_1 + \vec{p}_0)^T \\ (\vec{p}_4 - \vec{p}_0)^T \\ (\vec{p}_2 - \vec{p}_0)^T \\ (\vec{p}_1 - \vec{p}_0)^T \\ (\vec{p}_0)^T \end{pmatrix}^T . \tag{3.2}$$

Additionally, a parameter vector $\vec{v}$ is built from the coordinates of the point in space which I want to transform:

$$\vec{v} = (xyz, yz, xz, xy, z, y, x, 1) . \tag{3.3}$$

The transformation is now performed by multiplying the vector $\vec{v}$ with the matrix $\mathcal{C}$

$$\vec{v'} = \mathcal{C} \cdot \vec{v}. \tag{3.4}$$

Since matrix $\mathcal{C}$ is valid for all points to be transformed, the only computations to be done are the construction of $\vec{v}$ and its multiplication with $\mathcal{C}$. I describe linear transformations as a special case of the trilinear transformation by keeping the coordinate systems constant over the volume defined by the cuboid. This is the case if the cuboid represents a parallelepiped. This allows a detection of linear cases after propagation and composition, see Figure 3.4.

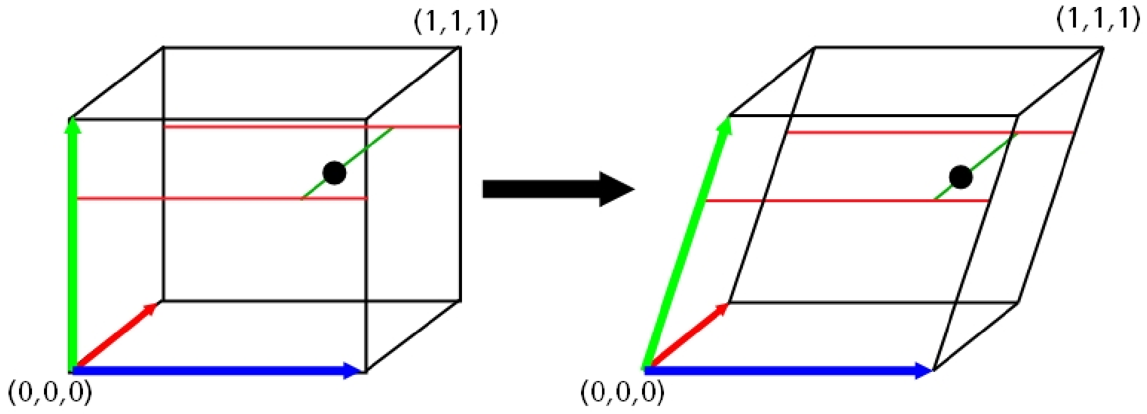


Figure 3.4: Warping of a unit cube using linear transformation results in a parallelepiped. This is a special case of a cuboid.

Taking a closer look at the 8×3 matrix $\mathcal{C}$ and the generation of coefficients two things become clear. The right half of $\mathcal{C}$ describes a linear 4x3 matrix. It contains a coordinate system plus an offset. The remaining columns of $\mathcal{C}$ can be interpreted as four vectors describing the difference between the parallelepiped defined by the right half 4x3 matrix and the intended cuboid. If the transformation describes a linear transformation this left half of $\mathcal{C}$ contains only zeros. This knowledge allows one to test whether a trilinear transformation is truly a deformation or just a linear transformation. This is a nice feature, since this test can be performed after composing the overall transformation for a geometry directly before rendering. This test allows one to limit the higher computational effort necessary for deformation to the objects really requiring deformation without the need of an extra protocol.

Both methods describe a warping of the whole space. This allows handling any geometric structure which can be represented in that space.

Approximating Arbitrary Deformations and Composition

Until now I discussed simple trilinear transformations using only a single cuboid. I will call this type simple transformation. In order to cope with real deformations I need a more flexible tool. Fortunately, trilinear transformations have a local character and can be attached side by side to form control grids. These grid structures can be used to approximate complex arbitrary deformations, like it is proposed by [RSSSG01]. Additional refinements of the grid are presented to allow local details of the deformation. Since my aim is to directly use 3D textures on the GPU to store these grids, I decided to use simple uniform grids. Additionally, this way trilinear interpolation of the texture stage can be used to perform the necessary interpolation within a single cuboid of the grid.

Since I intend to combine composite transformations with simple transformations, I have to take care of the following sampling issue: transforming the cuboid of a transformation is a sampling process that will only approximate the original transformation. If this is another simple trilinear transformation, this is no problem. A problem arises when an assembled transformation is sampled by one with lower frequency. This will lead to under-sampling. I prevent this problem by propagating the transformation with the highest resolution.

3.1.2 Decoration - Deferred Warping

Transforming objects to a surface of the garment is a special case. Cuboid based deformation could be used in order to place objects on the surface and let them follow the curvature. However, cuboids suffer from the problem of the necessity to control the structure which causes the deformation. With a surface which is deformed in real-time, updating these controls can be problematic when a lot of objects are attached to a surface. Instead I present a technique which not only has an implicit automatic update, but also can work directly with object position specifications within the production data of the planar sewing pattern. I call this technique deferred warping since it is a two step method:

First, the transformation field of a deformed surface is extracted on the fly. Second, this field is used to transform detail geometry on the surface, which I also call attaching geometry to a surface. Note that I use the terms *source object* for the geometry I want to attach to the deformed surface and *target object* for the externally animated or deformed geometry. I assume that a suitable parameterization of the target object exists, since this is the case when used with garments. In different applications such a parameterization can be generated using well-known techniques (see, e.g., Akenine-Möller et al. [AMHH08]). Parameterization and parameterization quality is discussed in more detail in Section 4.1.2.

In the following I first give an overview of deferred warping. Then I show how the transformation field of a deformed surface is determined. Finally, I demonstrate how different attachment types are realized by simply defining two mapping functions and introducing the most important types (see Figure 3.5). These attachment types can be used to realize a large variety of effects (see Figure 5.4). Deferred warping is, however, not limited to these types. Other definitions are also possible, e.g., to support the animation of the source geometry on the target surface.

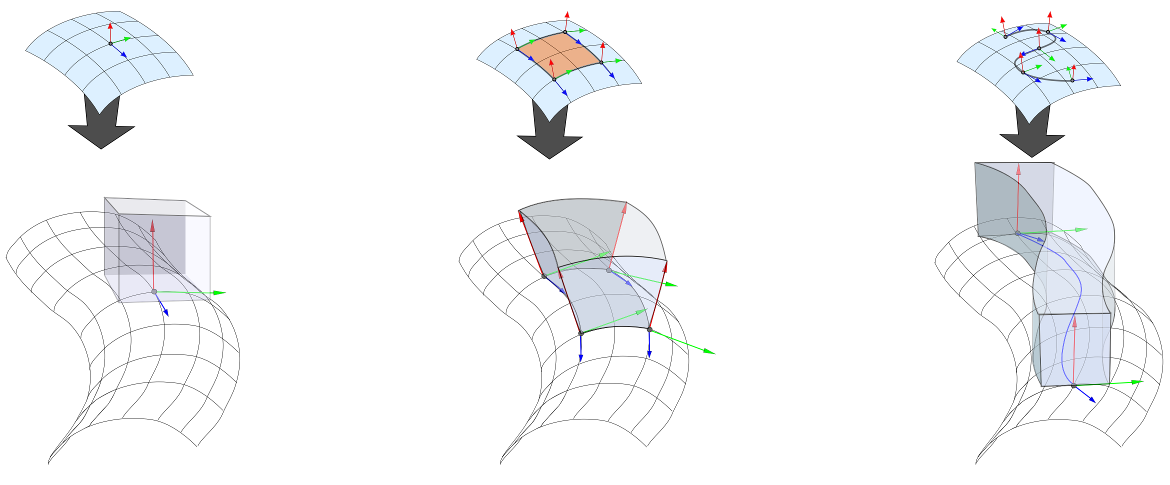

(a) (b) (c)

Figure 3.5: Possible attachments of a 3D geometry to a 2D surface. (a) Point attachment: The geometry is directly transformed by the matrix retrieved from the transformation field for the given position, placed on the surface, and oriented according to the principal axes of the subspace. (b) Area attachment: Texture coordinates are determined by projecting the source geometry on the *xz*-plane. The source geometry follows the curvature of the target surface. The *y*-coordinate of a source point defines the distance to the target surface in normal direction. (c) Curve attachment: A user-defined parametric curve function defines the texture coordinates for the matrix look-up. One component of a source point is used as parameter for the curve while the other components determine the final position in the plane which is orthogonal to the curve.

Overview

In the first step of my method the transformation field of a deformed surface is determined, which is defined by a 3×4 linear transformation matrix for each vertex. This field defines a look-up table from texture space into tangent space. With the transformation field known I can now define a *matrix texture* as the function $\mathbf{f}_{\text{mat. tex.}} : [0,1]^2 \mapsto \mathbb{R}^{3\times 4}$ which maps the surface parameter coordinates (u,v) to the corresponding 3×4 matrix (see Section 3.1.2). Hence, this matrix texture yields a local coordinate system on the deformed surface of the target object at any surface position (u,v).

In the second step the matrix texture is used to realize different attachment types. To obtain the corresponding local coordinate system for a point of the detail geometry, I need a mapping function $\mathbf{f}_{\text{uv}} : \mathbb{R}^3 \mapsto \mathbb{R}^3$ which yields a valid texture coordinate $(u,v,1)$ for each detail point. This function can be defined in different ways to realize different attachment types. Furthermore, I introduce the matrix $\mathbf{M}_{\text{tex}} \in \mathbb{R}^{2\times 3}$ to allow for a linear transformation in texture space. This gives the user the possibility to change the position, rotation, and scale of an attachment on the target surface easily and dynamically. Finally, the local coordinate system of a detail point $\mathbf{p}$ on the deformed surface is determined by:

$$\mathbf{M}_{\text{mat. tex.}}(\mathbf{p}) = \mathbf{f}_{\text{mat. tex.}}\left(\mathbf{M}_{\text{tex}} \cdot \mathbf{f}_{\text{uv}}(\mathbf{p})\right). \tag{3.5}$$

The mapping function $\mathbf{f}_{\text{uv}}$ only depends on $\mathbf{p}$. However, deferred warping supports the use of arbitrary functions as long as they provide valid texture coordinates. For example, an animated attachment can be realized by using a mapping which depends on a time parameter.

The matrix $\mathbf{M}_{\text{mat. tex.}}$ can be used to transform detail geometry on the target surface. But instead of applying this transformation directly, I introduce another transformation matrix $\mathbf{M}_{\text{tang}} \in \mathbb{R}^{4\times 4}$ to give the user more flexibility. This matrix provides the possibility to transform the detail geometry before attaching it to the deformed surface, e.g. to scale the geometry in the direction of the surface normal. The final position of an attached detail point $\mathbf{p}$ is determined by:

$$\mathbf{p}_{\text{world}} = \mathbf{M}_{\text{mat. tex.}}(\mathbf{p}) \cdot \mathbf{M}_{\text{tang}} \cdot \mathbf{f}_{\text{map}}(\mathbf{p}), \tag{3.6}$$

where $\mathbf{f}_{\text{map}} : \mathbb{R}^3 \mapsto \mathbb{R}^4$ is another mapping function which is required to realize the different attachment types in Sections 3.1.2-3.1.2.

In summary I require $\mathbf{f}_{\text{mat. tex.}}$, the user-defined functions $\mathbf{f}_{\text{uv}}$ and $\mathbf{f}_{\text{map}}$ as well as optionally the transformation matrices $\mathbf{M}_{\text{tex}}$ and $\mathbf{M}_{\text{tang}}$ to attach detail geometry. In the following subsections I will introduce the definitions of these functions.

The Transformation Field of a Surface

A surface in 3D space can be parameterized using two variables u,v, i.e., $(x,y,z) = \mathbf{f}(u,v)$. The tangent space plane can be computed from this surface parameterization for each (u,v) pair using partial derivatives in u and v as $\mathbf{t}_u = \frac{\partial \mathbf{f}}{\partial u}$ and $\mathbf{t}_v = \frac{\partial \mathbf{f}}{\partial v}$. These vectors define the directions of the principal axes of u and v in 3D space. Given the tangent vectors I can compute the normal of the surface at position (u,v) as $\mathbf{n} = \mathbf{t}_u \times \mathbf{t}_v$ if the surface is not degenerated and if the parameterization on the surface is not pathological (i.e., $\mathbf{t}_u$ and $\mathbf{t}_v$ are not parallel). $\mathbf{n}$, $\mathbf{t}_u$ and $\mathbf{t}_v$ are then linearly independent and form a subspace in $\mathbb{R}^3$ which is aligned with the surface's tangent space. The tangent vectors and the normal are orthogonalized after their computation to obtain an orthogonal coordinate system. This is required to prevent a distortion of the attached geometry.

My key idea is to extend the determined subspace with the position (x,y,z) on the surface to form a 3×4 linear transformation matrix. The matrix texture function which maps the surface parameter coordinates (u,v) to the corresponding transformation matrix is then defined by:

$$\mathbf{f}_{\text{mat. tex.}}(u,v) = \begin{pmatrix} t_{u,x} & n_x & t_{v,x} & x \\ t_{u,y} & n_y & t_{v,y} & y \\ t_{u,z} & n_z & t_{v,z} & z \end{pmatrix}. \tag{3.7}$$

Point Attachments

To attach a detail geometry to a point on a target surface (see Figure 3.5(a)), I first must define its target position (u_g, v_g) in texture space. Then the corresponding transformation matrix is determined by Equation (3.7). By applying this transformation the source geometry is transformed onto the surface and oriented according to the principal axes of the subspace.

Since the goal position (u_g, v_g) of the detail geometry on the surface is predefined, the first mapping function is defined by

$$\mathbf{f}_{\text{uv}}(\mathbf{p}) = (u_g, v_g, 1)^T. \tag{3.8}$$

The detail geometry must be defined in the coordinate system of the tangent space. Since this geometry should only be attached at a single point, the second mapping function is defined by

$$\mathbf{f}_{\text{map}}(\mathbf{p}) = \begin{pmatrix} p_x & p_y & p_z & 1 \end{pmatrix}^T. \tag{3.9}$$

The fur in the armadillo example of Figure 5.10 was fixed to the bunny using points attachments. The more complex area and curve attachments are described in the following subsections.

Area Attachments

In order to introduce an area attachment (see Figure 3.5(b)), I first need to define a mapping function $\mathbf{f}_{\text{uv}} : \mathbb{R}^3 \mapsto \mathbb{R}^3$ which yields valid texture coordinates $(u, v, 1)$ for each point of the detail geometry. Since the detail geometry is given in the coordinate system of the tangent space, the x, z coordinates correspond to the two tangent directions. Therefore, I define the mapping function by a projection onto the xz-plane:

$$\mathbf{f}_{\text{uv}}(\mathbf{p}) = (p_x, p_z, 1)^T. \tag{3.10}$$

If the resulting coordinates is not in the range of [0..1], I scale the whole geometry to that domain to get valid texture coordinates.

Since I determine a tangent space matrix for each point, the position on the target surface is well-defined. The distance of a point to the surface is defined by its y coordinate. Therefore, I get the final position for a point of the detail geometry by moving its corresponding surface position in normal direction by p_y. Mapping the point $\mathbf{p}$ the function

$$\mathbf{f}_{\text{map}}(\mathbf{p}) = \begin{pmatrix} 0 & p_y & 0 & 1 \end{pmatrix}^T \tag{3.11}$$

yields the desired result. Recall that the first and the third column of $\mathbf{M}_{\text{mat. tex.}}$ correspond to the two tangent vectors, while the second and the fourth column correspond to the normal vector and the surface position (see Equation (3.7)).

Curve Attachments

The curve attachment is the most complex type. The source geometry must be attached to a curve on the deformed target surface (see Figure 3.5(c)). To define this curve the user has to provide a parametric curve function $\mathbf{f}_{\text{curve}} : \mathbb{R}^2 \mapsto [0,1]^2$. This function delivers two-dimensional coordinates which can be directly used to define the mapping function for the texture coordinates in Equation (3.5):

$$\mathbf{f}_{\text{uv}}(\mathbf{p}) = (\mathbf{f}^x_{\text{curve}}(p_z), \mathbf{f}^y_{\text{curve}}(p_z), 1)^T . \tag{3.12}$$

I used the z coordinate of the source point as parameter for the curve function. Hence, the curve is oriented along the vector t_v which is no limitation. For my examples (see Figure 5.4) I defined the curve function $\mathbf{f}_{\text{curve}}$ using a cubic Bézier spline which is transformed to the desired position in texture space by $\mathbf{M}_{\text{tex}}$.

To define the offsets of a point $\mathbf{p}$ in the directions $\mathbf{t}_u$ and $\mathbf{n}$ I use the following mapping function in Equation (3.6):

$$\mathbf{f}_{\text{map}}(\mathbf{p}) = \begin{pmatrix} p_x & p_y & 0 & 1 \end{pmatrix}^T . \tag{3.13}$$

Since I used the z component of the source point as parameter for the curve the corresponding value of $\mathbf{f}_{\text{map}}(\mathbf{p})$ is set to 0.

The mapping function of Equation 3.13 attaches a detail geometry to a curve on the target surface but does not align the geometry to the curve. If the source geometry is also to be aligned to the curve I apply a different mapping function:

$$\mathbf{f}'_{\text{map}}(\mathbf{p}) = \begin{pmatrix} n^x_{\text{curve}} \cdot p_x & p_y & n^y_{\text{curve}} \cdot p_x & 1 \end{pmatrix}^T , \tag{3.14}$$

where $\mathbf{n}_{\text{curve}} \in \mathbb{R}^2$ is the normal vector of the curve at p_z. This function determines the position of the point $\mathbf{p}$ in the plane which is orthogonal to the curve, where p_y is the height over the surface and p_x is the offset perpendicular to the curve.

Vertex Post Processing

An important advantage of deferred warping is its seamless integration into the vertex transformation process. This provides the possibility to apply pre- and post-processing steps to the vertex data and yields a large flexibility. As an example, I demonstrate this flexibility in my fur rendering approach which uses a post-processing step in order to emulate gravitational forces acting on the hairs (see Figure 5.10). The hair geometry is transformed on the surface of the bunny by using point attachments of small fur patches which are randomly placed and rotated.

Without a post-processing step the single hairs would all stand perfectly orthogonal to the surface which looks unrealistic. To obtain more realistic results I manipulate the vertex positions of a hair which I get from Equation (3.6). First, the y coordinate of each point is reduced depending on its squared distance to the surface:

$$\mathbf{p}'_{\text{world}} := \mathbf{p}_{\text{world}} - \begin{pmatrix} 0 & c \cdot p_y^2 & 0 \end{pmatrix}^T, \tag{3.15}$$

where c is a user-defined value which scales the magnitude of the effect. After this step all single hairs hang down but their lengths vary widely. Inspired by [MKC12], I mitigate this problem by adapting the position $\mathbf{p}'_{\text{world}}$ so that it has the same distance to the corresponding surface point $\mathbf{p}_{\text{surf.}}$ as the original point $\mathbf{p}_{\text{world}}$:

$$\mathbf{p}''_{\text{world}} = \mathbf{p}_{\text{surf.}} + |\mathbf{p}_{\text{world}} - \mathbf{p}_{\text{surf.}}| \frac{\mathbf{p}'_{\text{world}} - \mathbf{p}_{\text{surf.}}}{|\mathbf{p}'_{\text{world}} - \mathbf{p}_{\text{surf.}}|}. \tag{3.16}$$

In this way a simple gravitational fur bending effect was introduced into the transformation. The result can be seen in Figure 5.10.

3.1.3 Smoothing - Tessellation

I have presented a method for placing single objects inside the scene, taking into account different sizes and forms. Additionally, it is now possible to place objects on a garment's surface using curves specified on the planar production sewing patterns. However, since I'm sampling a surface I need to make sure to have enough sampling points to follow the curvature. Therefore it can be necessary to increase the mesh resolution of the objects I want to attach to the surface. A simplified example is given in Figure 3.6.

The hybrid GPU-based tessellation technique I developed and presented in [KKK10b] is ideal for this task. Aside from increasing the number of triangles guided by tessellation levels it allows to additionally modify the triangle density.

Semi uniform adaptive triangle tessellation techniques (see Chapter 2) are adaptable for a geometry shader implementation, too. But the limitation to uniform triangles as basic refinement strategy leads to unnecessary high polygon counts when dealing with thin triangles (see Figure 3.7). Additionally, the snapping algorithm can lead to sudden resolution changes

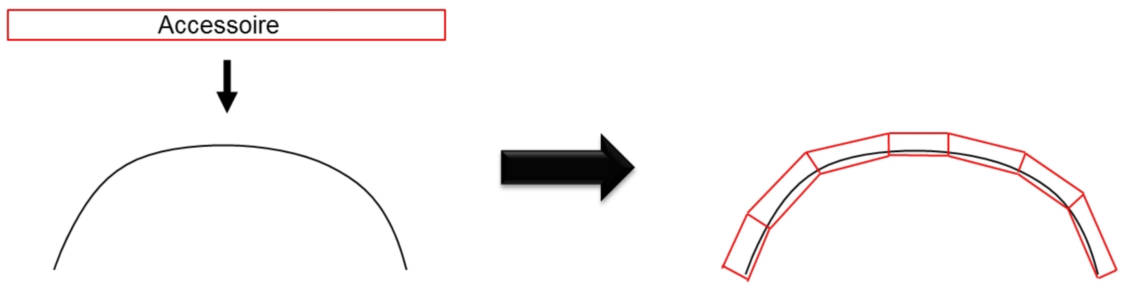


Figure 3.6: Attaching an accessory geometry to a surface can require tessellation of the object in order to make it follow the curvature of the surface in a better way.

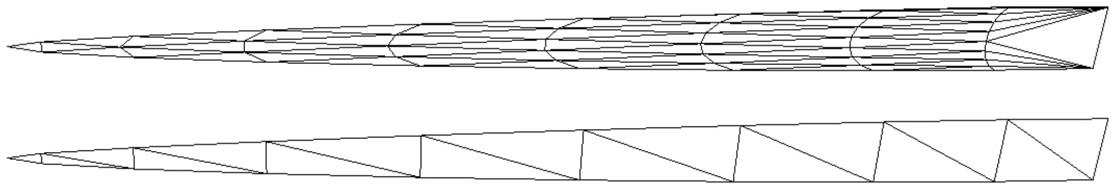


Figure 3.7: Comparison between semi-uniform triangle tessellation and my approach: Tessellation of a thin tetrahedron. Semi uniform triangle tessellation uses the highest edge refinement level for its internal structure, resulting in a high triangle count. My scheme uses triangle based tessellation only on the tip and fills the remainder with a quad patch, which can be tessellated with different resolutions for the x and y axis. This results in a much lower polygon count in this example and triangles that are more adapted to the resolution on the edges.

on the patch border. My presented refinement algorithm uses a quad- and a triangle refinement to circumvent these two problems. The triangle refinement uses a uniform mesh while the quadrangular mesh is based on a semi adaptive approach.

In this section I present my novel hybrid adaptive refinement method I use inside the geometry processing stage of the GPU. In the GPU shader model of DX10 a new shader stage was introduced [Bly06]. The stage is located between vertex processing and triangle rasterization. These geometry shaders take mesh primitives as input and create new primitives from them. The shader has limited resources, especially regarding the number of new vertices created from an input primitive. In order to minimize the number of vertices in my generated geometry I use triangle strips. Another limiting factor is the restricted accessibility of lookup data from the shader side. A catalogue of topological patches is not suitable under this restriction. Thus, I generate my topology on the fly. Since I combine different refinement strategies on parts of the patch surface I have to represent relative positions on the patch. I use barycentric coordinates as suggested in [LCNV09]. This allows an easy interoperability of the two different refinement strategies. Additionally, it allows the modification of the triangle density on the patch as a post process after refinement and prior to the surface evaluation of the patch. This warping process is described later in this section.

Refinement Stage

To perform the refinement of a triangle patch several steps are performed. The patch is rotated, split into a triangular and rectangular part, which then are tessellated using two specialized mesh generators.

Rotation

I first rotate the triangle patch (TP) to have the edge with the lowest refinement level on its base. Thus, I always have the same starting condition. This process was inspired by the work of Dyken [DRS09] which uses a permutation process to reduce the number of possible refinement states. After rotation, the quad patch is placed in the lower part, while the triangle part is located in the top part of the TP.

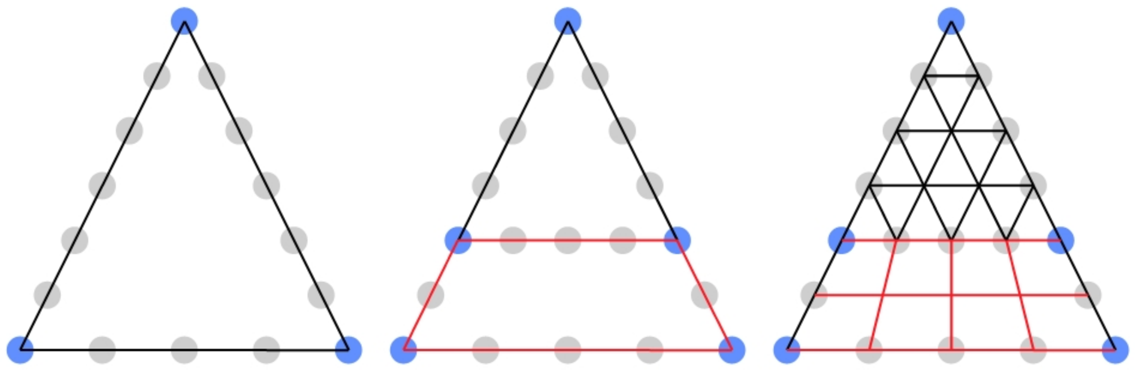


Figure 3.8: Overview of my refinement process: Given a triangular patch and three refinement levels on its edges (left), I perform a split between the two edges with the highest refinement level (middle). The remaining triangle is filled with a uniform triangle mesh with a resolution equal to the lowest resolution of the patch. The gap is filled by a trapezoidal quad matching the surrounding edge resolutions (right) of the remaining area.

Split

After rotation, I perform a split of the TP into a triangular part and a trapezoidal one (see Figure 3.8). The triangular part is processed by a triangle mesh generator. The quad part is processed by a rectangular mesh generator. The split of the TP is performed on the two edges with the highest refinement level. This allows the reduction of the maximum refinement level used in the two refinement generators.

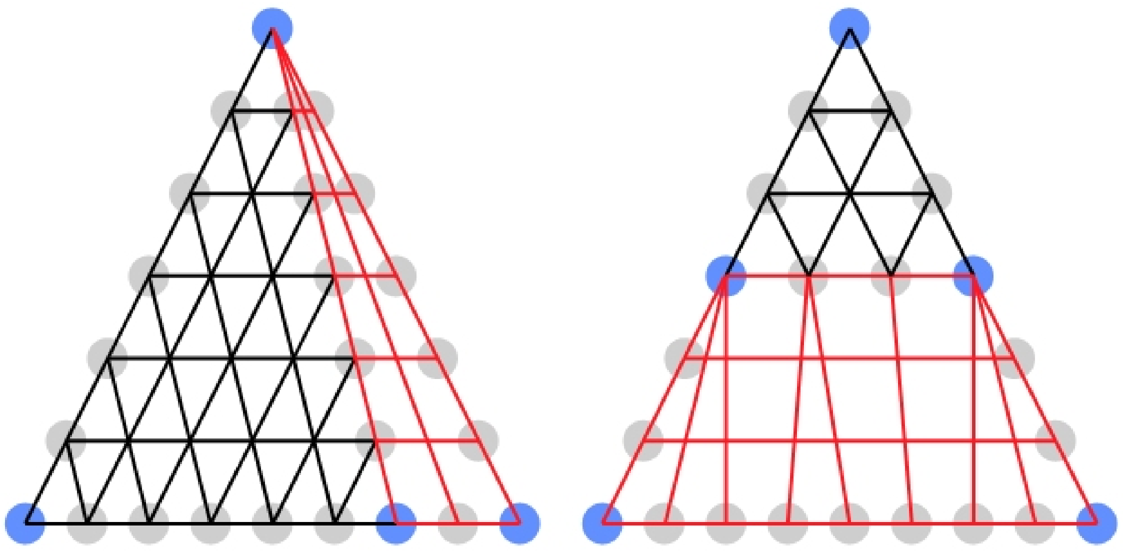


Figure 3.9: Special case: Two equal low resolution edges and one high resolution edge can lead to awkward topologies. In this case I rotate the quad patch and place it on the high resolution edge (left). This allows the adaptation of the high resolution edge to the other low resolution edges (right).

I start by defining the refinement level for the triangle mesh generator. My quad mesh generator is only adaptive for the left and the right side. Thus the resolution of the TP's base is not modified by the quad patch and defines the triangles resolution. The triangle mesh itself always has a uniform topology. By the initial rotation the left and right edge of the TP have a resolution higher than or equal to the TP base edge. Thus, the remaining difference on the two edges has to be chosen as target resolution for the left and the right side of the quad mesh generator.

A problem arises in the case of two equally low resolution edges and one high resolution edge. In this case large fan-like structures are generated. I circumvent this case by rotating the quad part by 90 degrees. This is possible, since both sides of the quad have the same resolution. Now the resolution of the refinement side can be lowered toward the top triangle (see Figure 3.9)

Mesh Generators

My approach uses two different types of mesh generators. The first one creates a uniform triangle mesh for a given refinement level. The second generates a rectangular tessellation which is semi uniform and uses a snapping mechanism inspired by the work presented by Dyken [DRS09]. Top and bottom of the rectangle share the same refinement level. In contrast the left and the right side allow different refinement levels (see Figure 3.10). The vertical resolution represents the maximum of both refinement levels. To create a watertight mesh a snapping process adapts the high resolution mesh to the lower refinement level. In contrast to [DRS09], I perform the vertex snapping only for the vertical axis of generated triangle strips from the quad refinement patch. Performing adaptation only on one axis simplifies the mesh generator. It allows the performance of snap operations in conjunction with linear interpolation for the vertical axis of the generator. The snapping process itself can be moved outside the inner loop of the tessellation procedure this way.

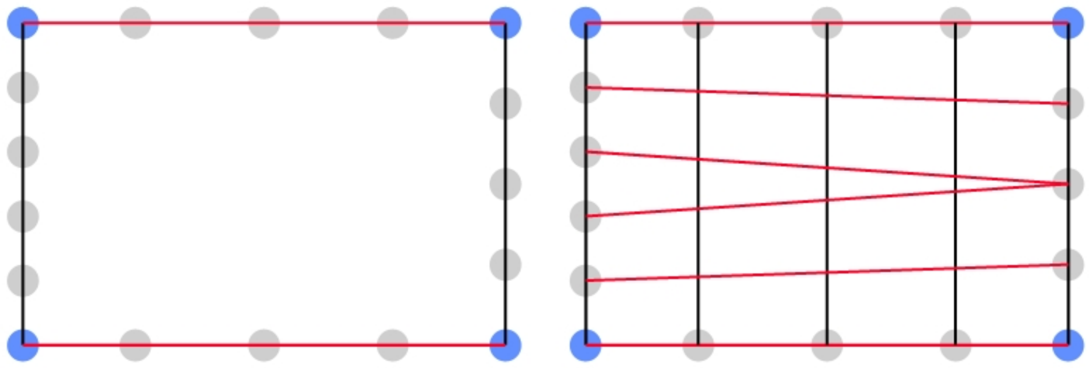


Figure 3.10: Overview of my adaptive quad generation scheme: In order to match the different resolutions for a given patch, I create strips from left to right. Utilising a snapping algorithm, the middle strip's right side is pinched to a single vertex in order to match the provided resolution for the right edge. By using linear interpolation for coordinate generation the change of resolution is distributed over the quad.

Barycentric Warping - Improving the View Dependency

The method as described so far already allows an adaptive tessellation of the input triangle patch. However, the method can only create a uniform distribution of vertices along an edge. My edge based level of detail (LOD) is derived from a vertex-based LOD scheme. Thus, it is obvious to use the vertex-based data to derive a density distribution for the generated triangles.

To provide a vertex-based density control mechanism I perform a warping on the barycentric coordinates generated by the refinement stage. The warping itself is performed by a quadratic Bézier triangle patch. The triangular patch is a quadratic function:

$$f : \mathbb{R}^3 \mapsto \mathbb{R}^3, u+v+w \leq 1, u,v,w \geq 0 \tag{3.17}$$

$$f(u,v,w) = \sum_{i+j+k<2} C_{ijk} \cdot u^i \cdot v^j \cdot w^k \tag{3.18}$$

$$f(u,v,w) = C_{200}u^2 + C_{020}v^2 + C_{002}w^2 \tag{3.19}$$

$$+C_{110}uv + C_{011}vw + C_{101}uw \tag{3.20}$$

It provides 6 control points. Three control points resemble the corners of a triangle (C_{200};C_{020};C_{002}). The remaining three represent points on the triangle edges (C_{110};C_{011};C_{101}).

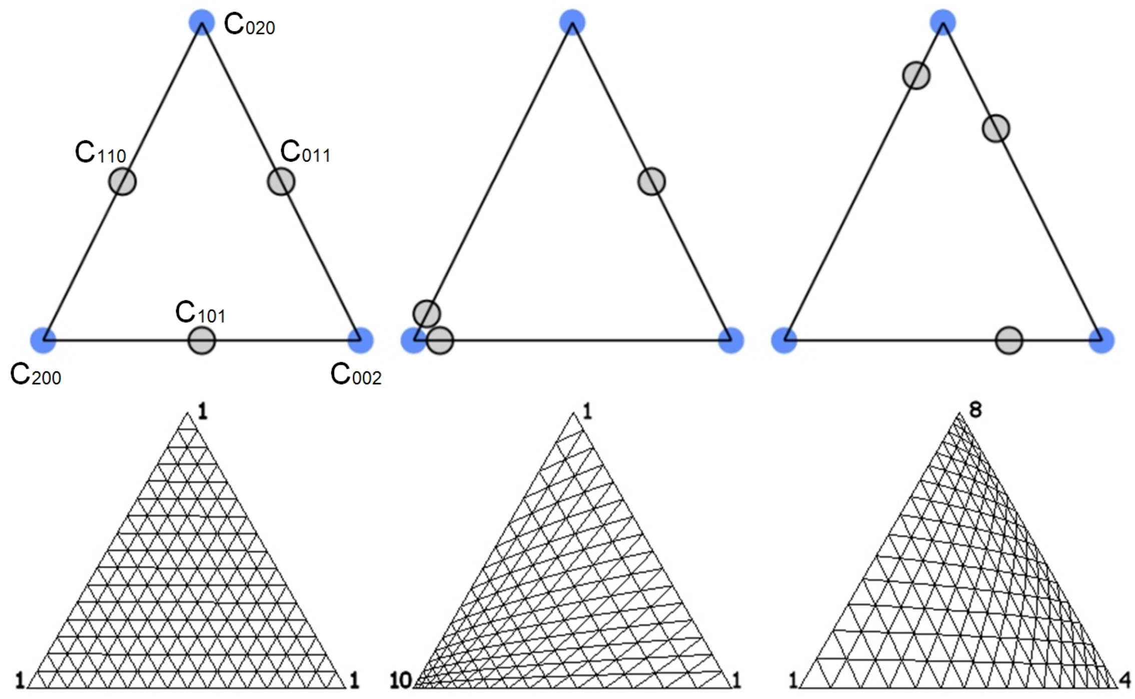


Figure 3.11: Coordinate warping of barycentric coordinates generated from the refinement: Using a triangular Bézier patch of degree two allows us to change the density of the generated structures on the patch. I compute a level of detail (LOD) on each vertex. The proportion of the two LOD levels of an edge is used to place the corresponding edge control vertex of the patch. Equal levels result in the control point to be placed on the centre of the edge. The mesh stays in its original state (left). Increasing the level of a vertex results in moving the control points and the resulting density toward this vertex (middle and right).

The corner control positions are initialized with their corresponding barycentric positions.

$$\vec{C_{200}} = (1,0,0) \tag{3.21}$$
$$\vec{C_{020}} = (0,1,0) \tag{3.22}$$
$$\vec{C_{002}} = (0,0,1) \tag{3.23}$$

The edge control points represent the relation of the vertex refinement level projected onto their common edge. Additionally, I have a set of level of detail values for the three corner points (L_{100};L_{010};L_{001}). I compute the 6 necessary weighting values (W_{210};W_{120};W_{021};W_{012};W_{201};W_{102}) as follows:

$$W_{210} = L_{100}/(L_{100} + L_{010}) \tag{3.24}$$
$$W_{120} = L_{010}/(L_{100} + L_{010}) \tag{3.25}$$
$$W_{021} = L_{010}/(L_{010} + L_{001}) \tag{3.26}$$
$$W_{012} = L_{001}/(L_{010} + L_{001}) \tag{3.27}$$
$$W_{201} = L_{100}/(L_{001} + L_{100}) \tag{3.28}$$
$$W_{102} = L_{001}/(L_{001} + L_{100}) \tag{3.29}$$

To compute an edge control point I linearly blend between their endpoints with the weights:

$$\vec{C_{110}} = \vec{C_{200}} \cdot W_{210} + \vec{C_{020}} \cdot W_{120} \tag{3.30}$$
$$\vec{C_{011}} = \vec{C_{020}} \cdot W_{021} + \vec{C_{002}} \cdot W_{012} \tag{3.31}$$
$$\vec{C_{101}} = \vec{C_{200}} \cdot W_{201} + \vec{C_{002}} \cdot W_{102} \tag{3.32}$$

This way the control points stay on the edge, always guaranteeing a straight triangular outline (see Figure 3.11). The result of this process is a mesh with a varying distribution of triangles (see Figure 5.13). In contrast to [LCNV09] I not only use barycentric coordinates to represent coordinates on the patch, but I manipulate them prior to their usage to get a better distribution of the generated triangles.

3.2 Rendering

Having processed all geometry it is now time to render the scene. For this purpose I have developed a hybrid occlusion process based on rasterization. It allows real-time rendering of the various materials of the garment with plausible self occlusion and natural lighting even when a high amount of details is present in the scene. I present this approach in two sections.

First, I explain how I evaluate the scene's directional occlusion (DO) gained from high dynamic range environment lights taking surface material properties into account. This part is based on my ideas presented in [KAKB14]. Second, I explain how I split occlusion computations in order to achieve both fast global low frequency occlusions and fast high frequency occlusions of the details using a hybrid occlusion approach. My ideas for this part where presented in [KK09].

Described first in the following is the global low frequency part.

3.2.1 Image Based Lighting, Material and Geometry Based Occlusion

My main goal for the low frequency lighting part is to achieve natural lighting for real-time rendering with directional occlusion, not only for diffuse and low gloss surfaces but also for

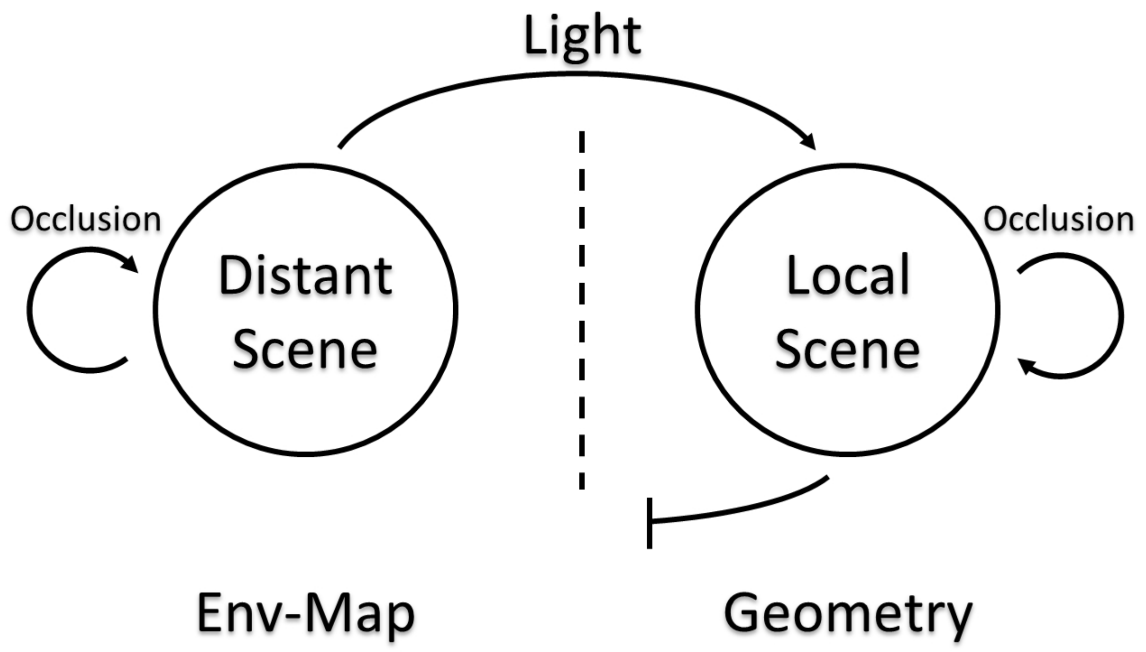


Figure 3.12: The lighting model I use in my approach. Since the whole scene is small the setup is only divided in distant scene and local scene. The distant scene is represented by an environment map. The local scene is the geometry, which has to be illuminated. The distant scene illuminates the local scene (top arrow), however, the lighting of the local scene has no impact on the distant scene (bottom). This model is a simplification of Debevec's model presented in [Deb98] for rendering with natural light.

high glossiness up to mirror-like surfaces. The basic lighting situation is depiced in Figure 3.12.

I identified two main problems of existing methods that I must address at the same time to reach this goal. First, the necessity to use all information of the environment map for the lighting process. Existing methods substitute the environment map with a set of light sources which is problematic when dealing with high gloss surfaces and limits the algorithms to diffuse and low gloss surfaces [RDGK12]. In contrast to previous methods I use the full image information of the environment map in the lighting process and compute only shadows with a set of virtual light sources. This allows the handling of high gloss surfaces without problems. Second, a reflection model is required which supports a smooth transition between diffuse and mirror-like glossiness. To meet this requirement I propose a reflection model based on filtering of the environment maps.

The basic outline of my new algorithm is:

1. Subdivide the original environment map into discrete regions and create a placeholder light source for each region. In this way I will do the occlusion computations for the self-occlusion of the scene.

2. Filter the generated discrete regions with my proposed method to get a lookup for the reflection model. I call the resulting filtered maps partial environment maps (PEM). This filtering allows the use of the PEMs as a direct lookup table for lighting, already including the glossiness of the materials. The novelty of this approach is that the lighting information is not only stored for a direction but also for a given glossiness value.
3. Use shadow mapping to get the visibility information for every light source, providing information about the scene's self-occlusion.
4. Determine the color for each pixel by combining the visibility information of the shadow maps with the color information of the (filtered) environment maps. This concludes the algorithm by combining occlusion values with their corresponding light information. Since this light information already contains the glossiness term of the material system, the result is a value containing material properties, occlusion and the lighting information of the whole environment map.

Environment Map Subdivision

In order to compute shadows with image based lighting, the environment map is converted into a discrete set of light sources, where each light represents a certain area of the map. Similar to the works of Havran et al. [HSK*05] and Annen at al. [ADM*08] I perform an importance sampling over the environment map to place the light sources depending on the brightness distribution of the environment map. The resulting partitioning should fulfill the following requirements for a good approximation:

1. Each point in the image space of the environment map belongs to exactly one partition:

$$\bigcup_{i=1}^{N} p_i = P, \qquad \bigcap_{i=1}^{N} p_i = \emptyset, \tag{3.33}$$

   where p_i represents the i-th partition of the original environment map P and N is the total number of partitions.
2. The intensity of the light source l_i is equal to the intensity of its partition p_i:

$$I(l_i) = I(p_i) = \int_{p_i} I(\mathbf{x}) d\mathbf{x}, \tag{3.34}$$

   where $I(\mathbf{x})$ represents the intensity of a point $\mathbf{x} \in p_i$.
3. Each partition is equally important for the final lighting.

Requirements 2 and 3 control two alternative ways to perform the environment map subdivision: If the resulting light sources are modeled individually to match their corresponding area with respect to shape, intensity and color (req. 2), requirement 3 can be neglected because the importance of each partition is weighted correctly by the parameters of the light source. On the other hand, if all light sources are modeled identically, it is mandatory for them to have equal influence on the overall lighting (req. 3). In this case, requirement 2 is automatically fulfilled up to a global scaling factor.

In order to work directly with a set of PEMs as a lookup table for the illumination for a given glossiness value each PEM must embed the reflection model. The lighting of the scene is the sum of the single light sources. This is also the case for summing up the single regions of the environment map. My choice to represent the material system as a set of energy-conserving filter functions exploits this similarity: It is equivalent whether I filter the sum of the single regions or whether I sum up the filtered regions. The latter allows the use of the single PEMs directly for lighting purposes including the reflection model: Instead of using the light sources directly to perform the lighting computations, I use the lights only for shadow computation while using their corresponding PEMs for the lighting itself. Therefore, requirement 2 can be neglected in my case and I only need a subdivision algorithm that satisfies points 1 and 3.

Convolution Based Material Model

My motivation to use a simple, cosine based reflection model is based on the results of Lafortune et al. [LFTG97] and Filip et al. [FH04] showing that these functions can be used to approximate complex BRDFs and Bidirectional Texturing Functions (BTF). Thus, my BRDF model is a combined approximation of diffuse, Phong and mirror reflection BRDF models [AMHH08], which are based on cosine functions and their exponents. In a second step I approximate this reflection model with a convolution based approach, which allows embedding the model directly into the PEMs. This is important since the convolution between the original environment map and the reflection model would take too much time if computed on the fly while rendering.

Cosine Based Reflection Model

In the following I introduce my tunable reflection model. There already exist two approaches, which correspond to the two extrema for my method. First, classical environment mapping corresponds to the mirror case. Second, irradiance mapping corresponds to the perfect diffuse case. My idea is to find a controllable function which defines a good transition between both extrema going from diffuse over shiny to a perfect mirror. A suitable model is the cosine based Phong reflection model. The color for a perfect reflection $\mathbf{c}_{reflect}$ and a diffuse reflection $\mathbf{c}_{irr}$ can be obtained by the functions $f_{env} : \mathbb{R}^3 \mapsto \mathbb{R}^3$ and $f_{irr} : \mathbb{R}^3 \mapsto \mathbb{R}^3$:

$$\mathbf{c}_{reflect} = f_{env}(\mathbf{r}), \qquad \mathbf{c}_{irr} = f_{irr}(\mathbf{n}), \tag{3.35}$$

where $\mathbf{r}$ is the surface reflection vector and $\mathbf{n}$ is the surface normal. The irradiance f_{irr} of a surface point is given by convolution of all incoming light with the Lambert reflection function for the current surface normal $\mathbf{n}$:

$$f_{irr}(\mathbf{n}) = \int_{\Omega^+} f_{env}(\omega) \cdot (\mathbf{n} \cdot \omega) d\omega. \tag{3.36}$$

Here, Ω^+ is the positive hemisphere for the direction **n** and ω describes all vectors belonging to this hemisphere.

Phong based specular reflection is evaluated a bit differently. The specular intensity I_{spec} is computed by evaluating the dot product between the light vector **l** and the surface reflection vector **r**. The result is potentiated by the parameter s_{Phong} in order to create a sharp small highlight:

$$I_{spec} = (\mathbf{r} \cdot \mathbf{l})^{s_{Phong}}. \tag{3.37}$$

If this parameter is set to very high values, the reflection model will start to approximate environment mapping. This is due to the fact that a high value of s_{Phong} will filter all values with $\mathbf{r} \neq \mathbf{l}$ taking only lights which lie in the direction of the reflection vector.

The ground truth for perfect diffuse surfaces is the Lambertian reflection used e.g. in classical Gouraud shading:

$$I_{Lambert} = (\mathbf{n} \cdot \mathbf{l}). \tag{3.38}$$

In contrast to the Phong model, the Lambert reflection model builds the dot product between the surface normal **n** and the light vector **l**. Both functions become the same when $s_{Phong} = 1$ and $\mathbf{n} = \mathbf{r}$. This is the case when looking perpendicularly onto a surface. However, having a Phong model with a low value for s_{Phong} does not make sense since the Phong model is only intended to be used with reasonably high values. To combine both models for the case $\mathbf{n} \neq \mathbf{r}$ I introduced a new glossiness value $g \in [0, 1]$ which controls both a new reflection value $\mathbf{r}'$ and the value of s_{Phong}:

$$\mathbf{r}' = \mathbf{n} \cdot (1 - g) + \mathbf{r} \cdot g, \qquad s_{Phong} = \frac{1}{1 - g}. \tag{3.39}$$

This function is a linear blend between the normal and the reflection vector dependent of the chosen glossiness. This is motivated by the observation that I need a transition between diffuse and full reflection. Note that also other interpolation techniques can be used. However, I did not notice any artifacts using linear blending. The combined reflection model can now be written as:

$$I_{combined} = (\mathbf{r}' \cdot \mathbf{v})^{s_{Phong}}. \tag{3.40}$$

However, having a combined reflection model does not yet solve the problem of the necessity of computing an integral over all incoming lights.

Embedding of the Reflection Model

My goal is to get rid of the integral over all incoming lights to increase the performance. This is achieved by embedding my proposed reflection model into the partial environment maps. The motivation is that a lookup table can be generated which allows to retrieve the lighting information for a given glossiness value using a texture lookup.

The application of the Phong model to a light probe results in a behavior similar to a variable low pass filter. Comparing this to the way environment maps are stored I get a very similar structure. An environment map is described by a function $f : \mathbb{R}^3 \mapsto \mathbb{R}^3$ providing a color $\mathbf{c}$ for a given direction $\mathbf{v}$:

$$\mathbf{c} = f(\mathbf{v}). \tag{3.41}$$

Let f_{light} be such a function for the light probe which represents the amount of incoming light for a given direction $\mathbf{v}$. I now use environment maps to store the result of the spherical convolution of the light probe f_{light} with my reflection model for a given glossiness value g and the direction $\mathbf{r}'$:

$$f_{embedded}(\mathbf{r}', g) = \int_{\Omega^+} f_{light}(\omega) \cdot (\mathbf{r}' \cdot \omega)^{s_{Phong}} d\omega. \tag{3.42}$$

This leads to a set of functions which allow to directly look up the result of the convolution of f_{light} for a given glossiness g. These functions allow the embedding of my reflection model into the environment map. The function $f_{embedded}$ has the form $f : \mathbb{R}^4 \mapsto \mathbb{R}^3$. Hence, mapping it to environment maps requires one environment map per value of g.

Approximation

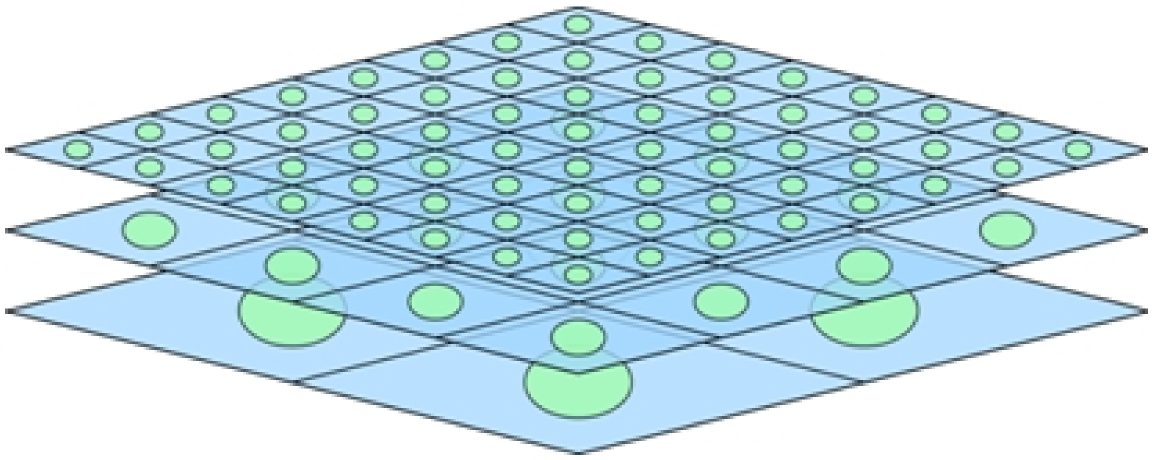


Figure 3.13: Using texture mipmapping for the low pass filtering of the original lightmap: Depending on the glossiness value I store light probe / Phog lobe convolutions in the corresponding mipmap level. For the depicted three level texture the results for reflection would be stored in the top level, glossy in the middle and diffuse in the bottom level.

Using the function set defined by Equation (3.42) directly as a rendering approach is unpractical. First, the set is too large to be stored on the GPU for rendering. Second, the pre-computation of the single maps will take a long time even on current machines since the convolution has to be performed always on the full resolution of the environment map f_{light}. Instead I make use of mipmapping which is available for most environment mapping

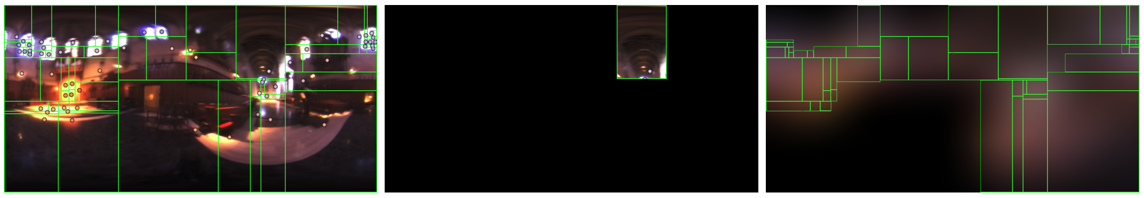


Figure 3.14: Environment map partinioning and PEM generation. Left: the result of the median cut algorithm [Deb08]. Middle: one single PEM extracted from the original texture, Right: the filtered sum of multiple PEMs. The images are adapted from [Deb08]. Instead of using the generated light sources directly, I apply the effect of the whole associated area onto the scene, combined with the light source's shadow.

techniques. The mipmap levels of an environment map are already low pass filtered versions of the original (see Figure 3.13). Thus, the stack of mipmap environment maps already behaves similar to my set of functions if g is chosen carefully (see dotted violet line in Figure 5.20). In this case a given g corresponds to a given mipmap level, leading to a function $f : \mathbb{R}^4 \mapsto \mathbb{R}^3$ which allows the lookup of direction and glossiness. Furthermore, I added a convolution based 5×5 filter for each mipmap generation stage. In this way the approximation can be improved significantly and I get a simple to use and very fast lookup method for the lighting.

Occlusion Computations

The introduced mipmap based approximation allows me to use the generated environment maps as a representation of real area light sources of arbitrary form and coloring while the shadows are computed based on simple directional lights representing these areas. Since the PEMs already contain the convolution of various filter kernel sizes the material lookup can be done with just a single lookup in the corresponding PEM (see Figure 3.14).

To compute the visibility for each PEM I use an approach based on shadow mapping. Since every PEM can be interpreted as an area light source, a soft-shadowing algorithm for area light sources is needed. Annen et al. [ADM*08] introduce a suitable method with high quality shadows for differently large light sources.

Final Lighting Computation

For each pixel I compute the sample vector for texture lookup just as for traditional environment mapping. I can use the same sample vector for each PEM. Then I compute the final color $\mathbf{c}$ of a pixel as a weighted sum over the corresponding texels of each PEM:

$$\mathbf{c} = \sum_{i=1}^{N} VIS_i(x,y) \cdot PEM_i(x,y), \tag{3.43}$$

where N is the number of PEMs and $PEM_i(x,y)$ denotes the color of pixel (x,y) stored in the i-th PEM. The weights are given by the visibility factor VIS_i which is computed using soft shadow mapping.

3.2.2 Screen Space Techniques - Hybrid Occlusion

The problem of the this approach is its geometry dependency of the shadowing computations. This limits the approach to low resolution geometry far from the amount of detail needed for virtual prototyping. For this reason, I apply the process only to the raw surface geometry, leaving out fine surface details while generating global shadows. In order to apply self occlusion of the fine details I use a second method using local occlusions (see Figure 3.15), which I will present now.

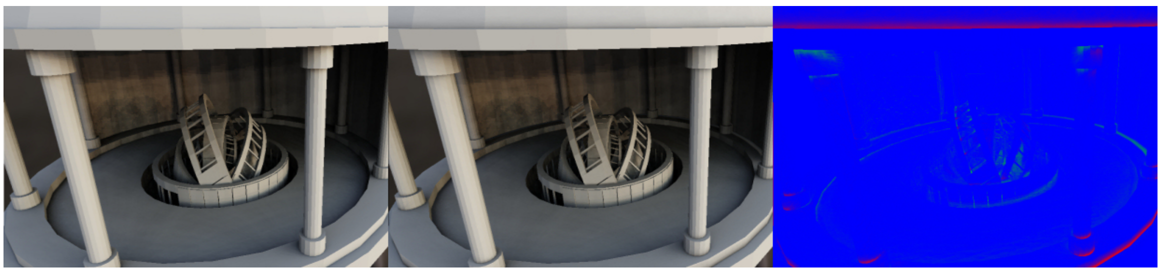


Figure 3.15: Difference between the occlusion of single light sources and an ideal hybrid directional/ambient occlusion combination. Left: shadowing of the single light sources. Middle: result of an ideal hybrid rendering. Right: difference image. Shadows of the single light sources are computed on a coarse level to reduce computational costs. Details are processed by a local AO technique.

My key idea is to separate the complex self shadowing problem into two parts (see Figure 3.16). The first step is the global shadow calculation on a level coarse enough to allow interactive frame-rates by using the approach mentioned in the previous section. While I propose to use a shadow map-based approximation for the directional occlusion, my hybrid approach is not restricted to this specific class of algorithms.

The benefit of this approach is mainly given by the characteristics of the local part. In the local part occluders are only tested near the current object of interest. This reveals high frequency details only and has the low computational costs of a local process. The main strategy is the reduction of overall computational costs by reducing the costs in the global part by far more then the costs introduced by the local part.

The Global Part

My intention within the global part is to calculate an approximation of the directional occlusion effects of the entire scene. The process starts by converting the light probe into a set of directional light sources, for which shadow maps are computed to approximate the

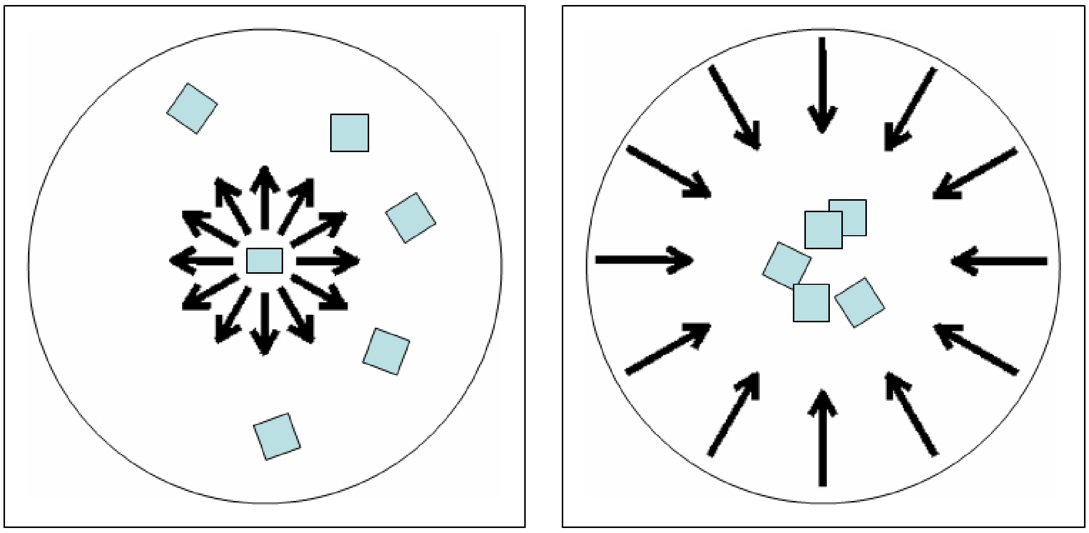


Figure 3.16: Differences between local (left) and global (right) part. In the local part an object tries to figure out its occlusion. In the global part light from outside the scene is occluded by the scene.

directional occlusion (see Figure 3.17). This is the process described in the previous chapter. The conversion process has to be done once since the light probe has a static property. However, the shadowing process itself is dynamic and can be performed completely inside the GPU. Additionally, I intend to overcome the appearance of single shadow borders by adding noise to the sample position of the shadow map. Unfortunately, this increases the amount of light bleeding since the higher threshold has to be chosen in order to avoid shadowing the occluder accidentally. This is due to virtual increasing of the area of a texel in the shadow map, which introduces the same artifacts as decreasing the shadow map resolution. The noising allows the reduction of the shadow map resolution, reducing the fill rate used to generate the shadow maps, thus saving computational time. Unfortunately, the whole shadowing process will tend towards low frequency shadows, since the high frequency parts are blurred away as well. These two issues, light bleeding and missing details are then addressed in the local part.

The Local Part

The idea of the local part is to add missing details to the scene shadowing, and to correct artifacts introduced by the global parts approximation. In this part I want to retrieve missing high frequency shadows and additionally I reduce the light bleeding artifacts. The basic approach is to apply an algorithm to the output image of the global part, which darkens concave areas of the scene with a local operation. Since this wanted property is similar to ambient occlusion it fulfills both purposes.

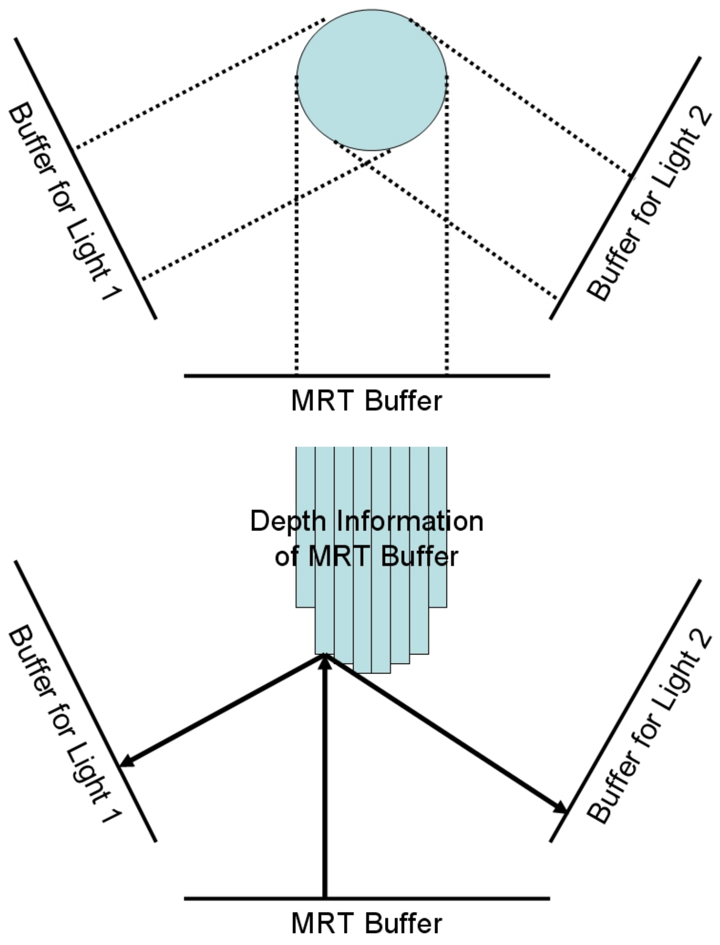


Figure 3.17: Principle functionality of my global part workflow. Upper image: I first render the whole scene into combined scene/shadow buffers (MRT buffer). Lower image: in the lighting stage only the data inside the buffers is needed to perform the shadow computations. The world position is calculated from depth and camera information. This position is then transformed into shadow map world space for the lookup.

Combination

In a final stage I have to combine the results of the two used methods. To simplify this I constructed the whole process around a deferred shading rendering system. The local part outputs an attenuation value with which the result from the global pass can be modulated. Modulation is a sufficient operation in this case, since directional information provided by the global part cannot be applied to the local part algorithm. So both outputs are combined in a final stage and not on a per light source level. Otherwise the output factors of both algorithms would have to be merged regarding their direction. By using an ambient occlusion technique in the local part I can avoid this problem.

3.3 Summary

In this section I have presented my conception of a visualization system allowing a garment designer to get real time feedback of what they do in conjunction with real time simulation as part of a virtual prototyping system. In order to guarantee producibility of the final product, all operations of the proposed method can be related to the flat 2D sewing patterns. My conception for the detailing and visualization of the virtual prototype is located directly after simulating the cloth. This means it can be run in sync with a real-time cloth simulation system. My approach combines techniques in a way which contradicts the weaknesses of the single techniques in order to visualize a high amount of surface details at the same time as plausible self occlusion. I achieve this by moving the geometry processing step of the surface detailing process from the simulation to the visualization component. This allows me to differentiate between basic geometry and surface details which are treated differently when computing self occlusion. Another beneficial effect of this change is the availability of the construction data of surface details within the grasp of the GPU as it is described in the next chapter.

4 Implementation

In this chapter I will present the implementation of my approach. I will first outline the architectures and decisions I used for implementation. Then I will go into details regarding the specific steps of the single techniques.

On the architectural side I used two distinct software frameworks to test my techniques. The first one is used for testing and research of the proposed algorithms which I called "Werkstatt". To tap into the full process containing garment simulation and design I had the opportunity to work with the program "Vidya" of the company Assyst [ws16]. It is a complete virtual prototyping system to assist the communication between pattern maker and designer in order to show a fresh garment design virtually without the need of a physical prototype.

Architecture of the Prototype Research Framework "Werkstatt"

For research on rendering algorithms I have build a framework based on the idea of visual programming. Its goal is the representation of a modular, node based representation of a complete rendering system. The set of nodes represents a domain specific language by means of large function blocks. An overview of the system architecture is shown in Figure 4.1. The system is controlled by the rendering loop and uses worker threads for managing resources and communication. Resources are managed using unique identifiers and lazy resource management. This allows the system to specify operations independently from the availability of the data used. Prior to rendering, the graph is evaluated in order to assemble all information needed, similar as it is done in normal scene graph APIs. However, aside from pure scene graph data I include additional contexts aside from scenes. This ranges from generative commands of the geometry placed in the scene (for example physical simulation of a mesh) up to the definition of the rendering process itself near the root node of the graph. Graphs with different meaning can be built. One example is the ability to specify new nodes using a built-in library code generator (see Figure 4.2), another one is used for specifying scenes and renderers (see Figure 4.3).

An example usage for the domain of user interfaces is the ability to build control menus by defining a menu description graph. These menus can be applied to expose selected controls to the 3D preview window. This reduces control complexity when working with new approaches.

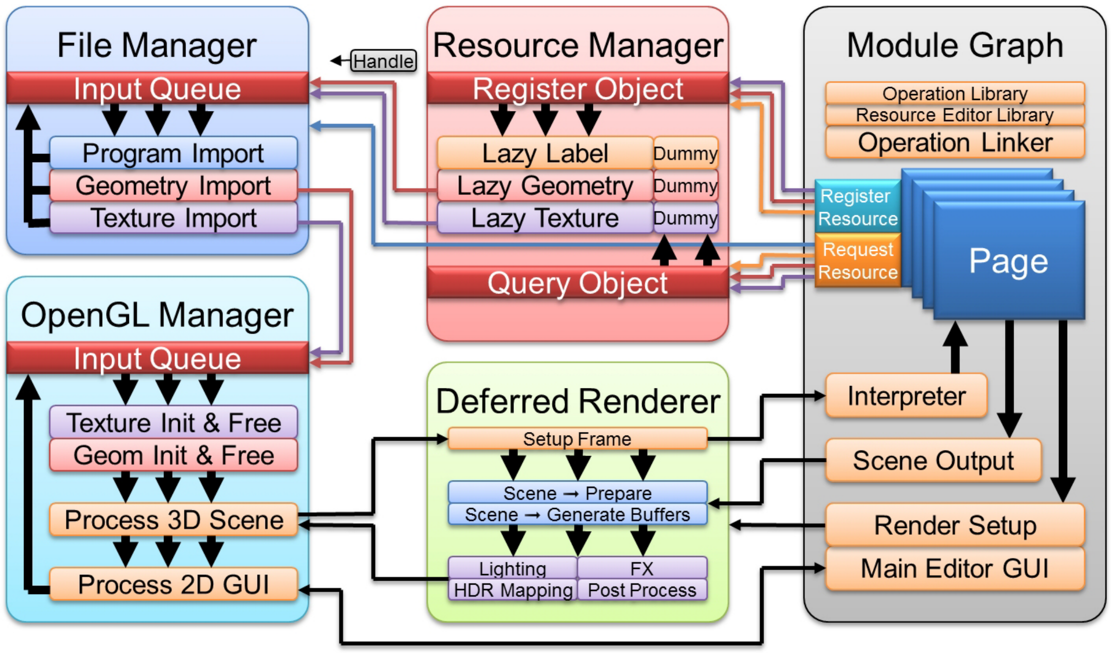


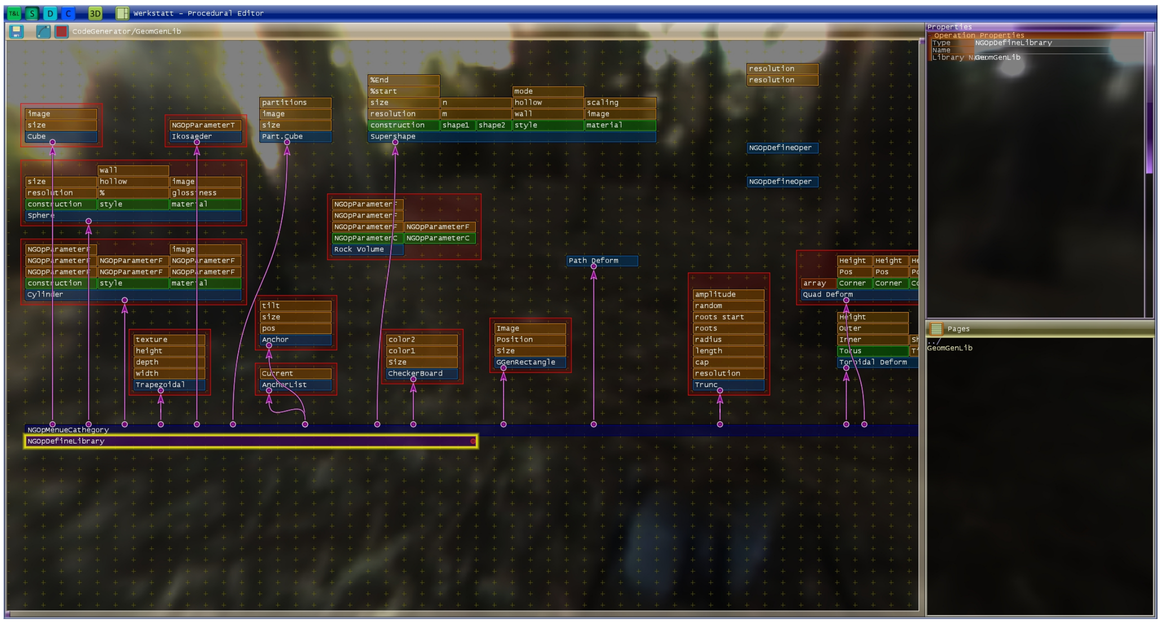
Figure 4.1: Architecture overview of my research and demonstration framework "Werkstatt". The system is based on the idea of graphical programming and the module graph. The core functionality is therefore centered on resource management (top and right side), as well as the handling and updating of the module graphs (left side). Everything is synchronized by the main rendering loop (bottom) which has trigger points inside the module graph.

Data Generation and Real-Live Tests

Input data for testing the algorithms were generated using:

- A simple built in simulation for quick testing and debugging
- Assyst Vidya for testing with real live garment designs
- Autodesk Maya for testing complex simulated dynamic scenes

In addition to development and testing in my own framework the material system, parts of the hybrid ambient occlusion technique and the tesselation method were integrated into Assyst Vidya for testing in conjunction with real live garment production. Analogously to the garment design process described in Section 1.2 the virtual prototyping system is divided into several sections. The prototype development starts within a 2D CAD system which allows to design the sewing patterns. These patterns are extended by sewing information before sending to the simulation program, which gives the cloth sheets a physical behavior. The results of the simulation are then visualized with the possibility of manipulating single parts of the garments by draping them virtually. These functional parts are directly repre-


Werkstatt - Procedural Editor
CodeGenerator/GeomGenLib
image size Cube
NGOpParameterT Ikosaeder
partitions image size Part.Cube
%End %start size resolution construction n m shape1 shape2 mode hollow wall style scaling image material Supershape
resolution resolution
NGOpDefineOper
NGOpDefineOper
wall size hollow image resolution % glossiness construction style material Sphere
NGOpParameterF NGOpParameterF NGOpParameterF NGOpParameterF NGOpParameterC NGOpParameterC Rock Volume
Path Deform
NGOpParameterF image NGOpParameterF NGOpParameterF NGOpParameterF NGOpParameterF NGOpParameterF NGOpParameterF construction style material Cylinder
tilt size pos Anchor
Current AnchorList
texture height depth width Trapezoidal
color2 color1 Size CheckerBoard
Image Position Size GGenRectangle
amplitude random roots start roots radius length cap resolution Trunc
Height Height Pos Pos array Corner Corner Quad Deform
Height Outer Inner Torus Toroidal Deform
NGOpMenueCathegory
NGOpDefineLibrary
Properties
Operation Properties
Pages
../
GeomGenLib

Type NGOpDefineLibrary
Name
Library Name GeomGenLib

Figure 4.2: Werkstatt in action: The screenshot shows a code generation graph which is used for defining the content of new nodes. As output the generator creates code for a c++ library containing function stubs and glue logic necessary to make the new functionality available in the system. The graph is traversed from bottom to top.

sented by the software architecture of the virtual prototyping system. An overview is given in Figure 4.4

I will now describe specifics regarding the implementation within my own framework of the single sections of my technique proposed in this work.

4.1 Geometry Processing

4.1.1 Macro Objects - Cuboid-Based Deformation

In order to process macro objects the input objects need to fit into a unit cube since it is a local transformation. Therefore, I additionally use bounding boxes to map these parts of the scene into unit cubes. These additional bounding boxes are defined for each non-linear transformation. All content of a bounding box is then warped into a cuboid. So in addition to the transformation hierarchy of the scene graph there is a bounding box hierarchy. This way the images of the child transformations are contained inside the parent's bounding box as needed in order to stay where interpolation of the cuboids is possible.

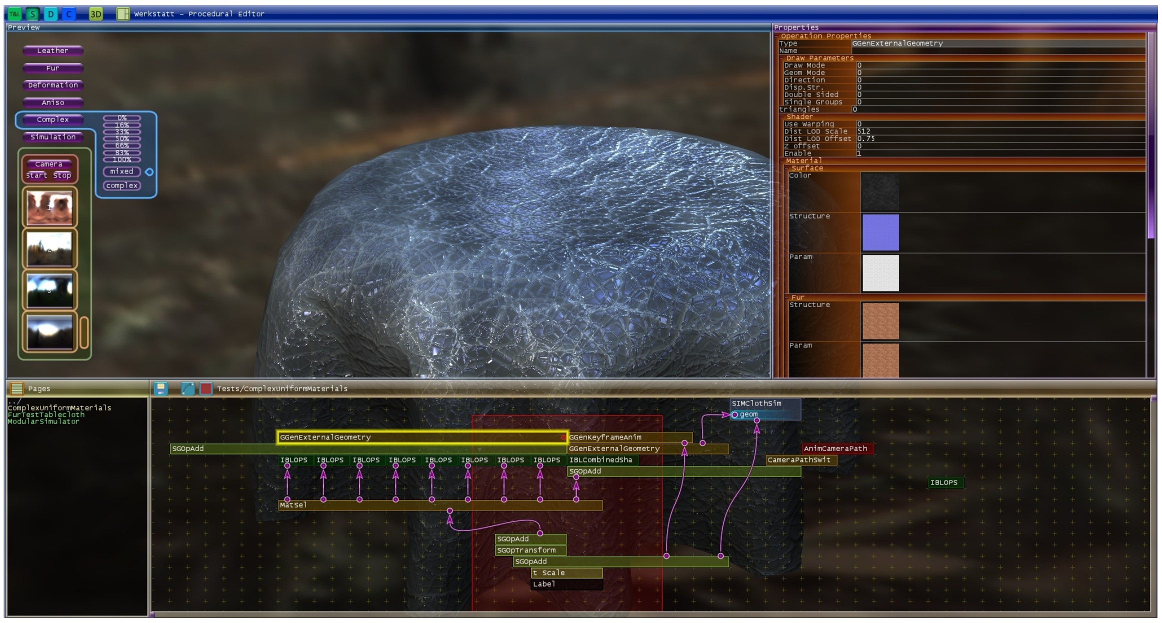


Figure 4.3: Werkstatt in action: The right side shows the normal state used for defining scenes and renderer.

The Transformation Process

In order to create propagation and composition I use one transformation ($\mathcal{G}$) to warp the other ($\mathcal{L}$). In principle the composition ($\mathcal{C}$) of two trilinear transformations $\mathcal{C} = \mathcal{G} \cdot \mathcal{L}$ is computed by transforming the positions of $\mathcal{L}$'s cuboid grid by the transformation $\mathcal{G}$.

A problem arises from this composition. My goal is to compose two trilinear transformations into a new one. However, composing a child node's transformation with the parent node's transformation does not result mathematically in a new trilinear transformation. This happens due to the ability of the trilinear transformation to be able to map lines to curves, which is always the case, if the image cuboid does not resemble a parallelepiped. To circumvent this problem I use the child's transformation cuboid to approximate the deformation represented by the parent's cuboid. Aside from introducing an approximation error this allows for getting one final composed transformation at a geometry leaf which is trilinear, and has to be applied to the geometry in the vertex processing stage.

Looking at the transformation hierarchy inside a scene graph I now perform the propagation and composition process the following way: I propagate the transformation from the world space towards the local space of the geometry. At each step I have to transform a more local transformation ($\mathcal{L}$) with the composition of the more global transformations ($\mathcal{G}$). The new composed transformation ($\mathcal{C}$) is now given by transforming the grid data of the local transformation $\mathcal{L}$ by the old composed transformation $\mathcal{G}$. As a result the new composition transformation is a transformed copy of $\mathcal{L}$.

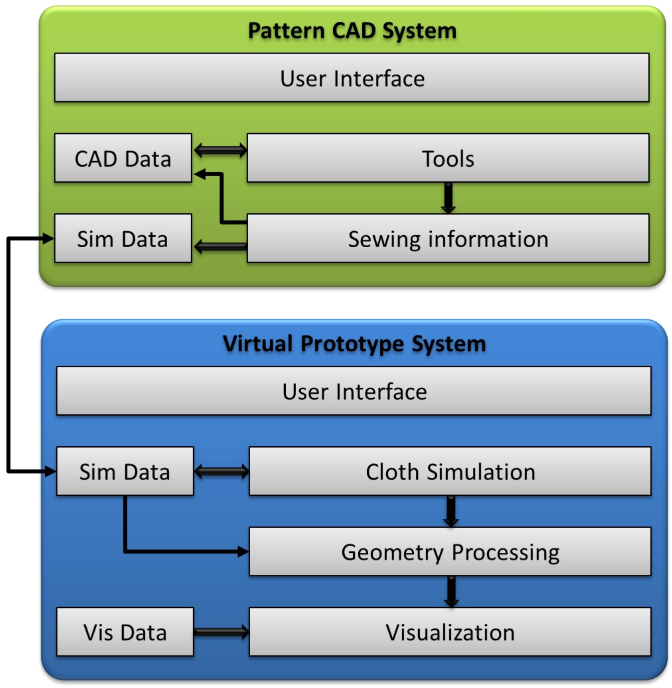


Figure 4.4: Coarse architecture overview of the prototyping tool "Vidya" closely coupled with the sewing pattern CAD software. Both softwares exchange data on a per-pattern base in order to allow fine granular design changes.

As described in Section 3.1.1 this composition of transformations resembles a sampling process. To prevent loss of information I have to choose a sampling grid for $\mathcal{L}$ with cells equal or finer in structure than the grid performing the transformation. Otherwise I have to handle this problem by increasing the resolution of $\mathcal{L}$'s grid, e.g. by resampling. I chose the grid size in the following way: Since I use a bounding box hierachy, $\mathcal{L}$ and $\mathcal{G}$ having the same resolution will automatically result in a proper sampling of $\mathcal{G}$ since $\mathcal{L}$ is smaller in size. Otherwise I check which transformation uses the highest resolution and use its grid resolution for computing $\mathcal{C}$. On the GPU I use interpolated 3D textures to represent the transformation grids. Since a 3D texture already has a domain defined on a unit cube, it is only necessary to perform the mapping from the bounding box to a unit cube in advance. Positions are represented by float values instead of colors in the texture.

I represent linear transformations only by using transformations with a grid size of one (one cuboid). As an optimization this allows us to check whether the transformation represents a parallelepiped. If the final transformation contains a larger grid I always consider it to handle a non-linear transformation.

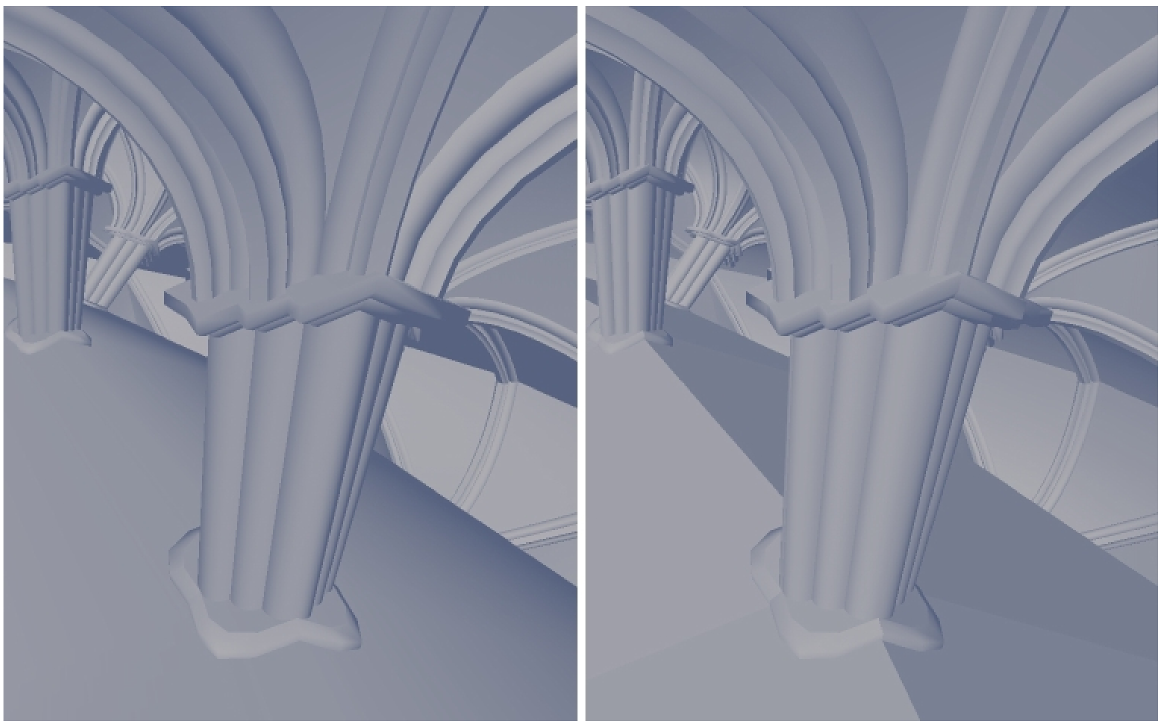


Figure 4.5: For rendering the lighting of the scene it is necessary to handle the deformation of the geometry's surface normals. I use two methods. Method 1 (left) simply transforms the normal with the given transformation. The resulting normal is not necessarily orthogonal to the surface, creating a smooth normal transition at the cuboid's boundary. Method 2 (right) shows the results transforming the surface's tangent space. Since the normal is now orthogonal to the surface, visible seams at the cuboid's border are noticeable.

Normal Transformations

Since groups of trilinear transformations are of C_0 continuity at their borders, special care has to be taken when computing the normals for lighting the contained objects [RSSSG01]. There are several methods for computing the normals used for shading surfaces (see [AMHH08]). For my implementation I have chosen two computation methods. Using the first method, I transform the tangent space representation of the normal. This results in normals orthogonal to the deformed surface. With the second method, I transform the normals directly. This results in a smooth lighting transition between cuboids. In both methods, transformations have to be performed with respect to the position of the vertex the normal or tangent space belongs to. This has to be done due to the position dependency of the coordinate system in a non parallelepiped cuboid. Both methods are valid and can be used with this approach (see Figure 4.5).

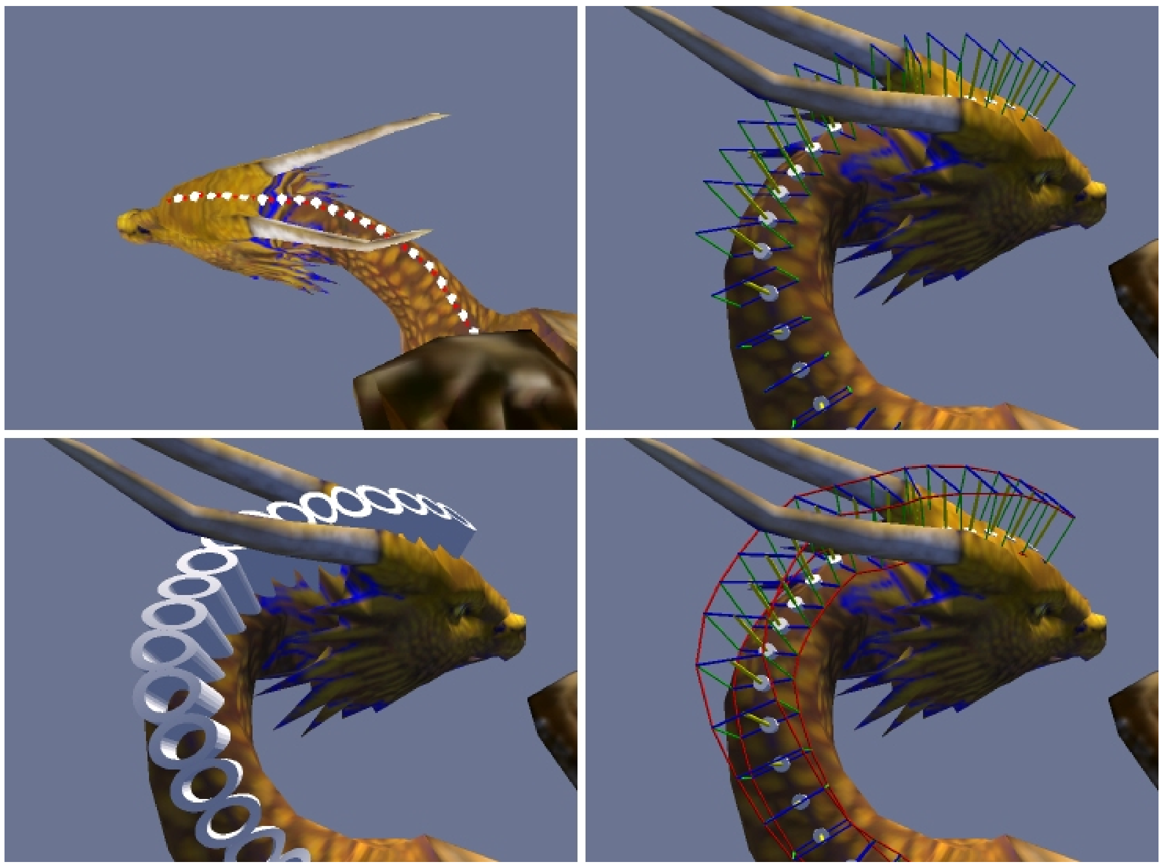


Figure 4.6: Automated handling of attached geometry in my method: At first I place a set of anchor points on the figurine (top left). Barycentric coordinates allow updating these points if the figurine is deformed or animated. These points represent local coordinate system (top right). The coordinate systems are used to create a deformation grid on the fly (bottom right). The control grid enables the scene graph to deform the attached geometry (bottom left).

Application Example

As many other deformation techniques, using my cuboid base deformation requires the presence of additional control structures. These structures need to be up to date when the surface changes, see Figure 4.6. In my case, this structure is generated by a list of anchors placed on the surface of the target's geometry, a transformation controlled by the anchors, and the geometry object which has to be attached. An anchor is defined by two points, which are registered and mapped onto a triangle of the basic mesh using barycentric coordinates. The first point resembles the anchor position. The second point is used to specify the orientation of the tangent vector. From these two points and the triangle's surface normal a coordinate system is computed. The transformation has to compute a trilinear transformation for this list of anchors, depending on which behavior the attachment is to have. An update of the anchors local coordinates system is performed by barycentric interpolation of the points of the triangle. This allows the computation of anchor position and alignment by only using the information of the geometry, see Figure 4.6. This concept is independent of the animation concept which is used on the basic mesh the anchor is applied to. The coordinate system defined by an anchor resides within the local space of the geometry. Thus the transformations dealing with the attachments have to work within the same space. In my example I use arrays of anchors to produce 1D or 2D deformation grids used by the cuboids. This method allows me to attach objects to the figurine wearing cloth since it is independent of the surface parametrization.

For the parametrized surfaces of the garments I use the deferred warping method. It is well suited in this situation since it allows direct specifying of locations of objects within the surface's parameter space which implicitly defines the control structure.

4.1.2 Decoration - Deferred Warping

The main difference between appliqués attached on a figurine and appliqués attached to cloth is the granularity. The number of details placed on a cloth surface can exceed by several magnitudes the number of appliqués attached to a figurine. Examples for this are sequins placed on cloth. In this case the algorithm handling and updating locations of the single sequins in real time has to handle several thousands of objects. Updating control structures as it is necessary for cage based deformation techniques and transferring the updated data onto the GPU is simply not practical due to the sheer mass of information. Deferred warping as described in the previous chapter here shows the advantage of performing all operations on the GPU solely based on the placement information of the single appliqué which only changes when edited. Therefore, this technique is my chosen one for this implementation case. In order to take advantage by using deferred warping I first had to consider some preconditions.

Texture Parametrization

The target surface I want to combine with deferred warping needs to fulfill some conditions. Since the surface's state is rendered into a texture, the surface needs a U/V parametrization without self intersections. Every point on the surface must be mapped to a unique location within the texture. However, areas of the texture may be left blank. The use of a texture atlas can be combined with my algorithm. Basically, my techniques face the same problems every mapping technique is affected by when using texture coordinates.

Luckily these conditions are perfectly met when dealing with geometry generated for garment design. Here everything has its origin within the flat 2D world of the sewing patterns. Additionally, for manufacturing the sewing patterns may not overlap and have to be packed tight. This is done in order to maximize the output from a single panel of cloth.

In order to maximize the space used on the texture I calculate a bounding box around the texture coordinates and fit the matrix texture inside it. A texture atlas was used for models consisting of several groups in order to keep the uniqueness of the mapping between surface and texture.

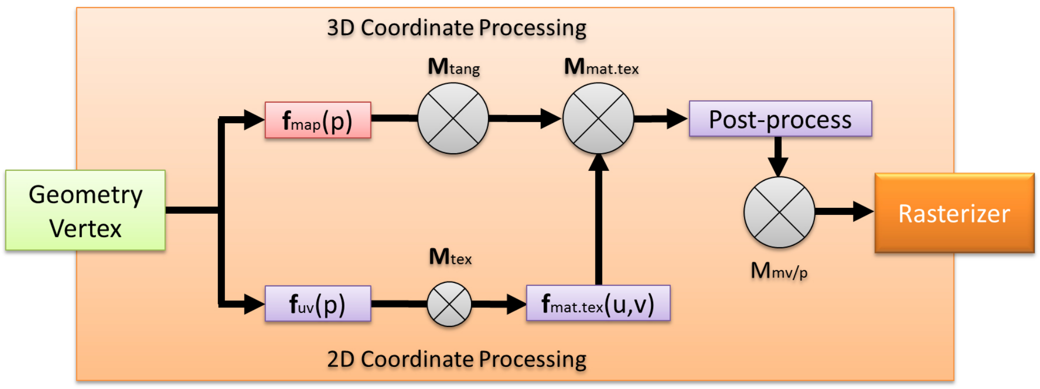


Figure 4.7: Vertex transformation block diagram. The upper pipeline processes 3D coordinates from tangent space to world space. The lower pipeline processes 2D coordinates defining how the object is placed in texture space on the surface and therefore determines $\mathbf{M}_{\text{mat. tex.}}$. The matrices $\mathbf{M}_{\text{tex}}$ and $\mathbf{M}_{\text{tang}}$ allow the independent positioning inside tangent and texture space. After an optional post-process of the transformed vertex it is passed to the standard model view transformation.

The minimal required resolution of the matrix texture can be estimated using the sampling theorem. According to the theorem, I need at last the double resolution of the size of the smallest feature I want to sample. In my case the smallest feature would be the shortest triangle edge of the target surface mesh in texture space. Within my examples the matrix texture is assembled by four RGB textures using floating-point components with a resolution of 512x512 pixels.

Using Deferred Warping

Deferred warping is performed in a two-pass rendering process at run time (see Algorithm 4.1.2.1). In the first rendering pass the matrix texture is updated. For animated target geometry I update the texture per frame to keep track of the animation. For the generation of this texture I need to determine the transformation (see Equation 3.7) of each vertex of the surface. This transformation consists of the vertex position as well as the normal and tangent vectors of the surface at the vertex position. Which is already known. The required vectors can be easily obtained since state-of-the-art rendering engines often support dot product bump mapping (see [AMHH08] for an overview). For this kind of bump mapping, tangent space data is needed and typically stored per vertex in conjunction with the surface normal. The shader transforms these vectors into world space. Additionally, tangent and binormal are orthogonalized before a matrix is rendered into the texture.

The matrix texture is then used in the second rendering pass in conjunction with the attachment mechanisms introduced in Section 3.1.2.

Each attachment type can be interpreted as an additional transformation step inside a vertex transformation stage of a renderer which is located between the local object transformation and the view and projection transformation. The transformation of the normal and tangent space of the source geometry is performed analogously to the vertex transformation. The three attachment types were implemented as vertex shader programs. Since only few values are required to control the mapping functions of the different attachments, I recommend using bulk processing methods like instancing on the GPU to maximize the geometry throughput. For this purpose I submit an array of parameters to the shader which holds placement data for up to 256 geometry instances. Thus, larger blocks of object instantiations need to be broken up into multiples of 256, giving a good speedup and still allowing arbitrary numbers of attached objects. A schematic view of the complete vertex transformation stage of my implementation is shown in Figure 4.7.

Algorithm 4.1.2.1 Deferred Warping

```
// Pass1:
for all texels $i$ do
    $\mathbf{M}$ = getLocalMatrix($i$)
    toWorldSpace($\mathbf{M}$)
    orthogonalizeTangentspace($\mathbf{M}$)
    writeTexel($\mathbf{M}$,$i$)
end for

// Pass2:
for all instances $i$ do
    for all vertices $v$ do
        $\mathbf{f}_{\text{uv}}$ = getFuvFor($i$)
        $\mathbf{M}_{\text{tex}}$ = getMtexFor($i$)
        $\mathbf{f}_{\text{map}}$ = getFmapFor($i$)
        $\mathbf{M}_{\text{tang}}$ = getMtangFor($i$)
        $\mathbf{p}_{in}$ = vertex($v$).pos
        $\mathbf{M}_{\text{mat. tex.}} = \mathbf{f}_{\text{mat. tex.}}(\mathbf{M}_{\text{tex}} \cdot \mathbf{f}_{\text{uv}}(\mathbf{p}_{in}))$
        $\mathbf{p}_{\text{world}} = \mathbf{M}_{\text{mat. tex.}} \cdot \mathbf{M}_{\text{tang}} \cdot \mathbf{f}_{\text{map}}(\mathbf{p}_{in})$
        postprocess($\mathbf{p}_{\text{world}}$)
    end for
end for
```

Outliers and Seams

In my approach a problem arises when detail geometry tries to sample the matrix texture at positions not used by the target surface. My solution to this problem is to initialize the matrix texture with a zero matrix. For a single vertex the algorithm can now detect whether it is outside by checking the length of the orientation vectors of the matrix. If the length deviates from one it is most likely for the vertex to have left the surface area. In my implementation I set a flag in this case within the vertex shader. The fragment shader can then discard all fragments which have something in common with this vertex. This way, I can effectively detect and discard outliers and their illegal states without much performance loss.

Aside from outliers another problem I faced were areas, where different cloth patterns meet. Such seams represent a discontinuity of the u/v space in both location and orientation. A curve or area appliqué crossing a seam will suddenly have a different u/v space on the adjacent cloth pattern. I deal with this by transforming the points near the seam by both u/v spaces and blending between them. This is detected by the same method as I use to detect outliers. The interpolation between texels of the map allows me to infer which u/v space a point belongs to. I use this information substantially to perform a linear interpolation between the spaces of the patterns.

4.1.3 Smoothing - Tessellation

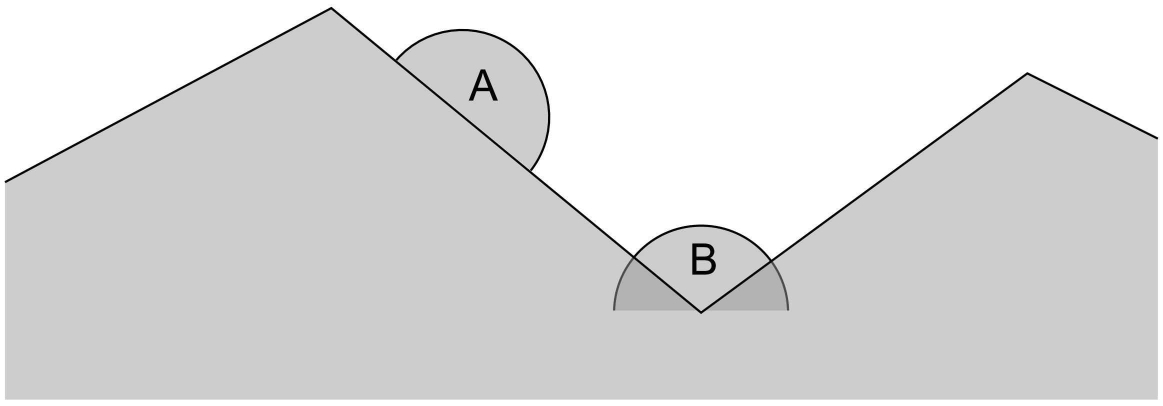


Figure 4.8: Mixing low resolution meshes and high resolution meshes for SSAO computations emphasizes edges on the low resolution mesh. Since the occlusion of fine details is evaluated the hemisphere used for searching is smaller than the faces of the coarse geometry. Therefore hemispheres placed on the center of the face (A) are less occluded than hemispheres on edges. This leads to a strong accentuation of the edges of the low resolution mesh which is not intended. Smoothing the depth values used for SSAO prevents the algorithm from pronouncing the edges of the single faces.

In my implementation I use tessellation to solve two different problems. First, by applying to appliqués, which do not have enough vertices to be directly placed on a curved surface to prevent gaps caused by undersampling (see Figure 3.6). Second, by using it to smooth depth values of the cloth surface when applying SSAO to the final rendering image. SSAO has the ability to shadow fine details and edges. This does not stop for edges of adjacent triangles. Since the cloth surface is smooth but tessellated this will create a lot of fine structures where they should not be (see Figure 4.8). Smoothing the depth values for the cloth parts prior to applying SSAO removes these issues.

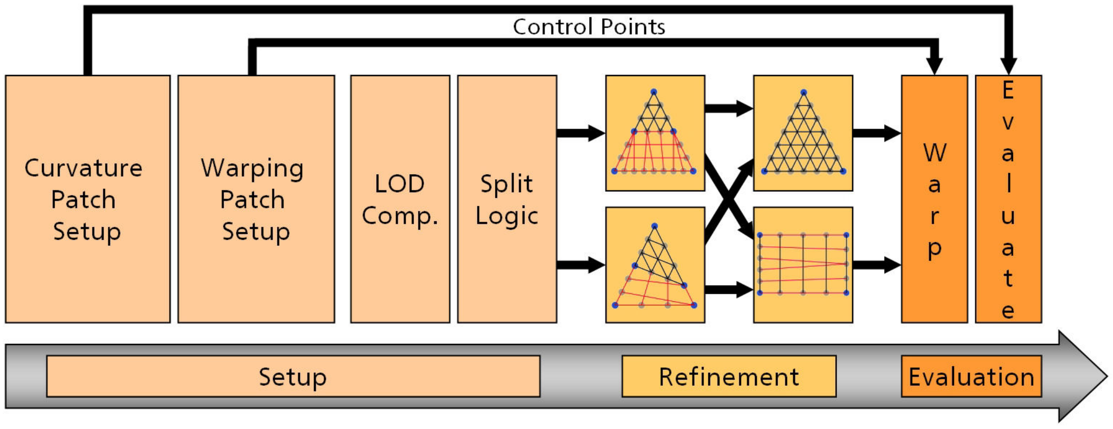


Figure 4.9: Overview of my algorithm implemented as a geometry shader: I start with the control coefficient computation for surface and warping patch. This is followed by view dependent refinement computations. Depending on the refinement data, a split is chosen dividing the patch into a quad and a triangular area. After the refinement the generated coordinates are warped and then evaluated.

I implemented my two tessellation approaches utilizing the OpenGL Shading Language (GLSL) (see [www10]) with the geometry shader extension within an OpenGL based rendering system. The process inside the shader is roughly divided in 3 stages (see Figure 4.9). In the setup phase I prepare data, which is consistent for the whole triangle patch. For surface smoothing I use the pn triangle patch approach presented by Vlachos in [VPBM01]. Additionally, a quadratic Bézier patch is set up to perform the coordinate warping post process. Initial splitting into the quad and the triangle refinement pattern is performed as well. The refinement stage hosts the two refinement processes and their setup. Finally, the quadratic patch is evaluated to compute the final coordinates used in the surface evaluation by the cubic pn triangle Bézier patch.

View Dependency

My refinement method presented is independent of the LOD computation mechanism used. However, for refinement testing purposes I had to choose one. I decided to combine a simple distance-based refinement mechanism and a silhouette refinement mechanism.

Distance based refinement aims to define different detail levels according to the distance of the object to the observer. My computation is roughly inspired by the technique presented in Sander et al. [SM05]. The observer is a point in 3D space which describes the camera position $O \in R^3$. The distance to a mesh vertex V is now simply the length of the vector from point O to point V. In order to get the current LOD for a distance I additionally define a minimum and a maximum distance, which define where to use the minimum and the maximum LOD. The distance is then scaled and normalized to represent a refinement level of one at the max distance and the maximal refinement level at the min distance. Additionally, a logarithmic falloff between both is chosen to take into account the effects of perspective projection. On the other hand, the main intention behind silhouette refinement is to increase the detail on the surface's creasing angle. I use a simple approach which is roughly based on the technique described by Hoppe in [Hop97]. The factor is computed per vertex by the dot product of the normalized distance vector from O to V and the surface normal at the position of V.

Smoothing Cloth Surfaces for SSAO

The tessellation method I use in conjunction with SSAO is different from the one used for tessellation of geometry. For a smooth appearance I require a pixel-wise smoothness. I achieve this by computing a per triangle patch setup as it is done for the geometry. However, the real tessellation step for the geometry is skipped. Instead, I forward the patch setup information directly to the fragment shader. This allows me to evaluate the patch within the fragment shader on a per-pixel level. This leads to smooth depth values for the SSAO computation reducing the self shadowing of concave edges of adjacent triangles.

4.2 Rendering

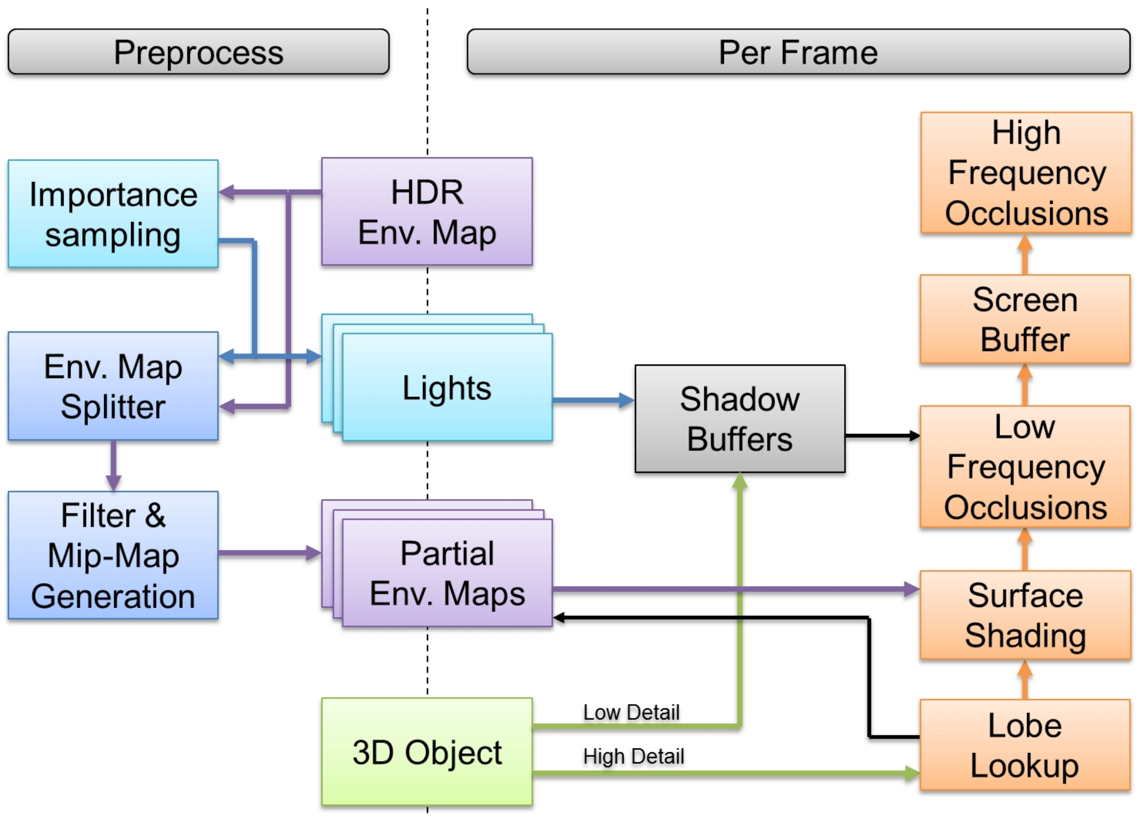


Figure 4.10: Architecture overview of my hybrid rendering process. Low frequency self-shadowing of the scene is computed based on directional occlusion of the low detail geometry. High frequency self-shadowing of the scene is performed by an image space based technique.

My goal of the rendering process is to achieve a high reactivity and interaction with the scene and at the same time to deliver a plausible material and self shadowing behavior with a high detail level. Both is needed by the designer who requires to edit the object in the scene and to judge what they get as feedback. To achieve this, I used a two level approach which handles coarse geometry data in another way than the fine geometry data. The coarse geometry is given by the current frame of the scene from the garment simulation system. The fine geometry is the additional data added within my geometry processes as appliqués onto the figurine and garment. On the implementation side this goes hand in hand as outlined in Figure 4.10. An overview of the implementation of the coarse geometry occlusion computation is given in subsection 4.2.1. This is followed by the implementation details of the hybrid occlusion which combines this first step together with SSAO.

4.2.1 Image Based Lighting, Material and Geometry Based Occlusion

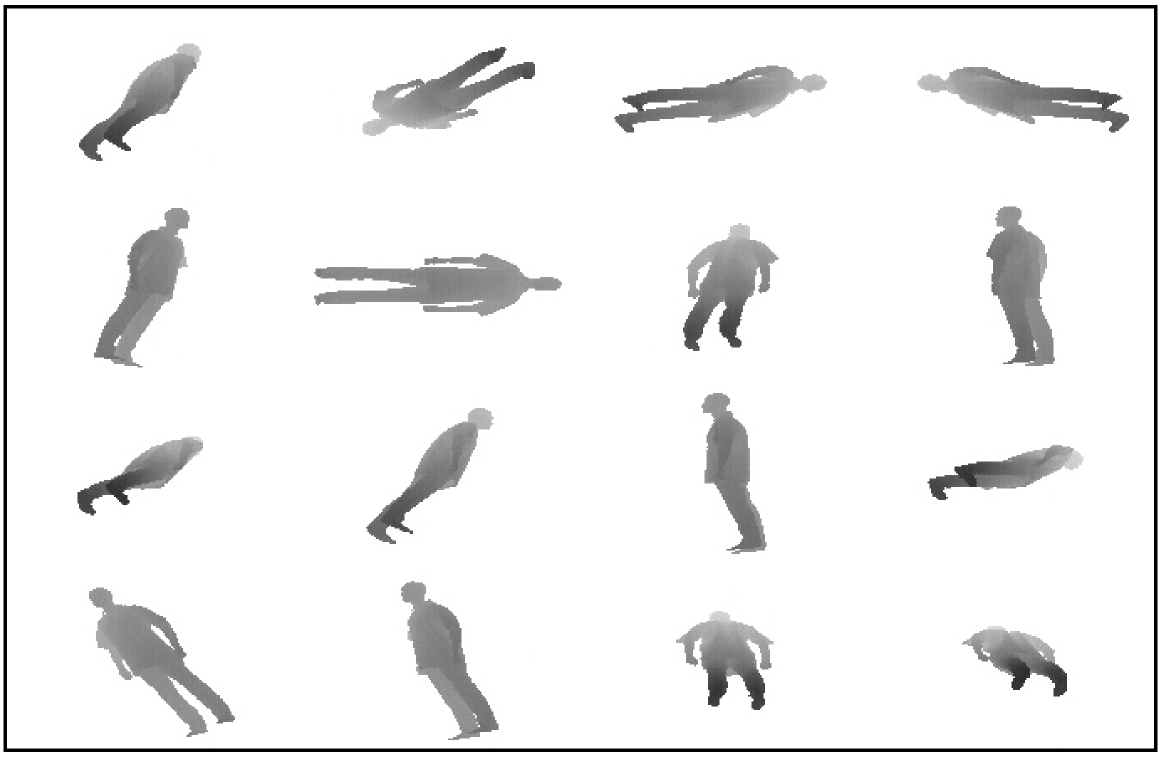


Figure 4.11: Bulk calculations of shadow maps for the rendering of coarse geometry occlusions.

The self shadowing of the coarse geometry is generated by computing all shadow buffers at once. This is done by using the geometry shader as a multiplier for the geometry's faces. Within the shader a face is generated for each shadow to be cast. Additionally, it is transformed to a location on a texture atlas containing all shadow buffers of the single light sources. The result of this process can be seen in Figure 4.11. The details of this approach are described in my work of [KF05].

For illumination I used cube maps as a model for environment maps. They are directly supported by the GPU and allow fast and direct access to their mipmap levels within shaders. To get the light source positions for the shadow calculations I sample the initial environment map using Debevec's median cut algorithm [Deb08]. It is a fast subdivision algorithm which fulfills the requirement that all resulting regions have to be equally important for the global lighting. Bright areas in the map yield multiple small regions whereas dark areas yield only few large regions. This process gives me the set of light sources and the set of regions required for my algorithm described in 3.2.1.

As a filter for the convolution in the mipmap generation process of the PEMs I tested several functions. I investigated filters based on the Gaussian distribution, the cosine function, several exponentiations of the cosine function and the box filter function mapped to the 5×5

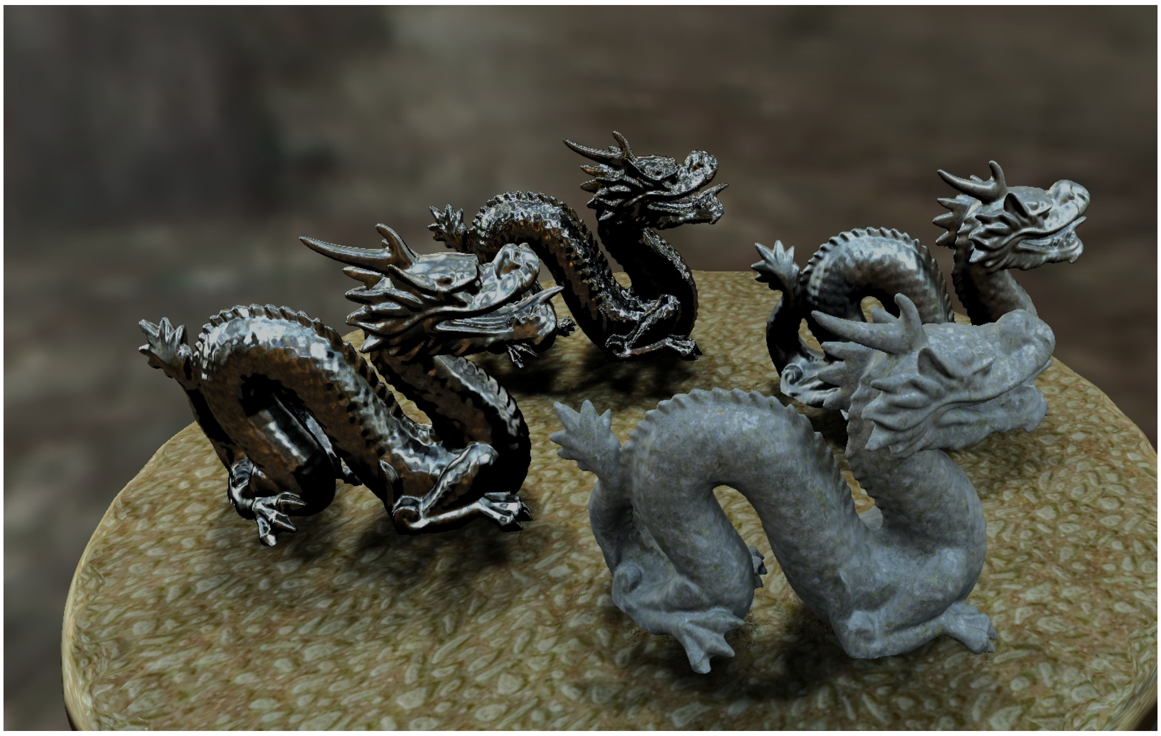


Figure 4.12: Applying different filtering levels to the environment map creates more or less glossy surfaces from mirror-like (back) to completely diffuse (front).

filter size. Results of the filtering effects are shown in Figure 4.12. Filtering can improve the approximation significantly. Especially a filter based on the square root of the cosine function shows a nice approximation behavior which is discussed in Section 6.2.1.

The resulting PEMs are stored as individual textures in GPU memory. For an easy and fast implementation I used cube maps. For each PEM I create a plain copy of the original environment map and just blacken all texels that do not belong to the current PEM (see Figure 3.14) before I start filtering. This technique increases the memory consumption but allows some major simplifications while lighting the scene.

I use soft shadowing based on directional light sources to handle occlusions. As a soft shadowing technique I decided to use Variance Shadow Maps (VSMs) [DL06, AMHH08] since they are simple and fast. The shadows are softened by applying a Gaussian filter over the entire depth texture. The size of the filter kernel determines how "soft" the shadows will be. The filter kernel size depends on the area of the corresponding PEM to simulate an area light source of that size.

4.2.2 Screen Space Techniques - Hybrid Occlusion

As described previously, I use an environment map as a large spherical area light source surrounding the scene. This is a light probe as described by Devebec in [Deb98]. From this light probe a light setup is generated and used for self occlusion computation purposes.

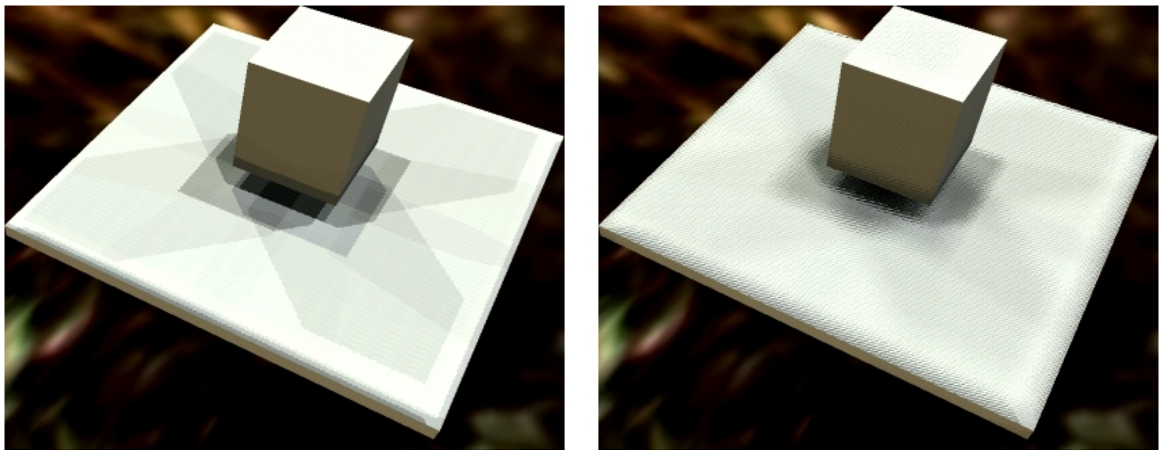


Figure 4.13: To smooth the transition between single shadows I introduce noise to the current sample position of the shadow map. Left image: shadow map sampling using only the map position, right image: shadow map sampling with noise, resulting in a smoother appearance of the shadows but in this case with a seemingly rougher appearance of the surface.

These light sources are used in conjunction with shadow buffers to create the self shadowing effect. The number of lights generated and the resolution of the shadow buffers are two parameters I can control in order to achieve a tradeoff between speed and accuracy. Knowing that I use a second algorithm to deal with high frequency self occlusions, I chose relatively low resolutions for the shadow maps. This was done with the intention to create soft, low frequency shadow data.

To enhance the image quality of the borders between shadowed and non shadowed areas, I introduce noise to the shadow map sample position, as seen in figure 4.13. The noise data is stored in a small tiled texture, which is repeated over the screen. This method introduces only low extra cost in difference to other smoothing or soft shadow methods. Additionally the shadow maps are applied in a screen space based method, which makes this part of the algorithm independent of the scene complexity. The cost of the shadowing process only depends on the number of shadows applied.

Global Part

To compute the coarse shadows generated by the shadow maps I apply them to all objects of the scene. These are the ones representing the coarse geometry and the ones representing the appliqués. Since the shadow buffer content is only generated by the coarse geometry, appliqués are not included within the overall self occlusion of the scene. This is done by a local occlusion part which is SSAO-based.

Local Part

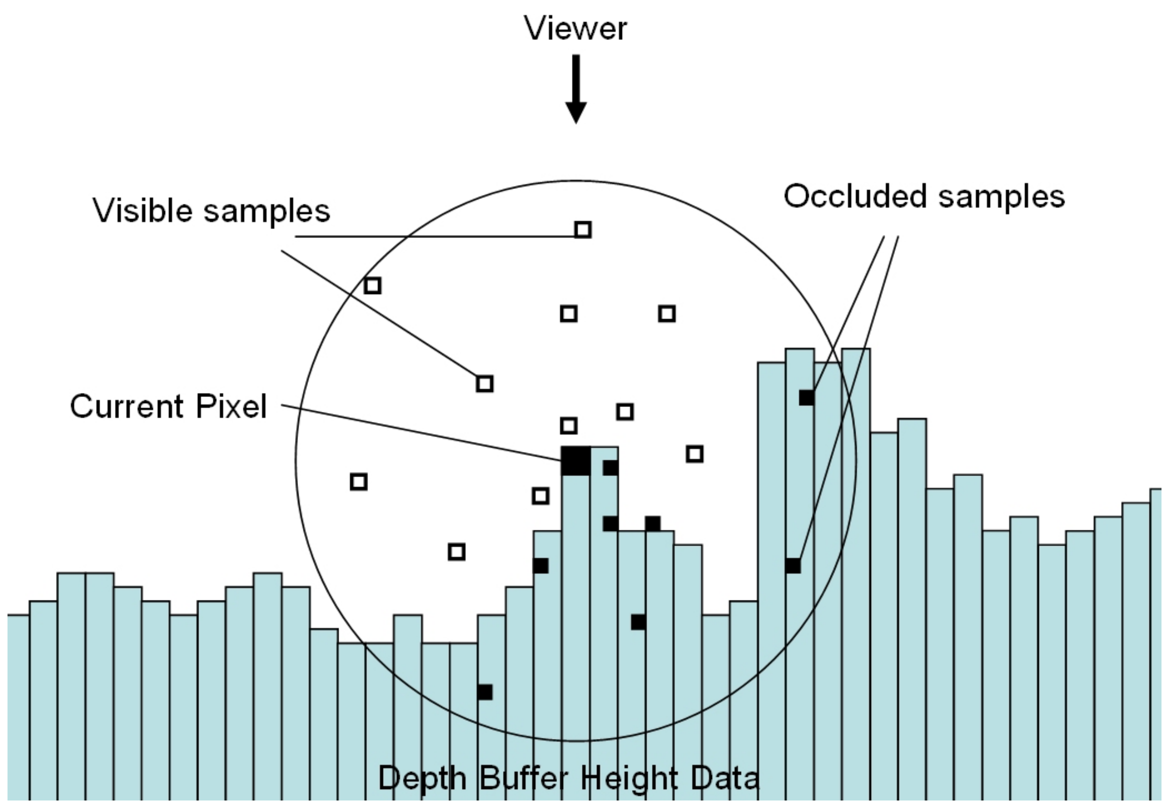


Figure 4.14: Self occlusion of the depth image, showing one line interpreted as height field: I test points inside a spherical region. The test results are related to the current sample point in the centre to get an approximation of its occlusion by its surroundings.

For the local self shadowing effects I have chosen a screen spaced ambient occlusion technique similar to the one presented in [SA07] (see Figure 4.14). It only processes data of the depth component of a multiple render target (MRT) buffer. In difference to the SSAO technique presented in [SA07] I used some simplifications. My version samples the occlusion of the spherical area around the currently analyzed pixel. I do not take into account the current normal. The distance attenuation of the importance of an occluder is automatically given by using a denser sampling towards the center of the sphere. The amount of occlusion is simply approximated by averaging the test results of the sampled positions. Due to this sampling method, objects occluding less than the half of the sphere would appear brighter than usual.

Since this is the case for all convex objects, I clamp the attenuation factor generated by the algorithm to the range between [0..1].

As it can be seen in Figure 4.10 this part of the rendering system was implemented via a deferred shading approach. This allows a modular design and reduces the number of rendering passes for algorithms working on the current frame of the scene. I start with a standard scene traversal and rendering of the scene. The result of the traversal is a light list which is processed later in the pipeline. The scene is rendered to a multiple render target buffer which holds the necessary information for the later deferred shading processes. In my case this buffer contains the color component, the normal component and the depth component of the scene. I use the camera parameters to reconstruct the world coordinates of pixel data from the depth buffer information. The global pass is performed with the light list, an associated shadow map list, the color and normal information from the MRT buffer. The local pass consists of the SSAO algorithm which works on the depth component of the MRT buffer. To reduce the number of texture reads inside the algorithm I make use of dithering. After the SSAO stage an adaptive filtering stage is applied to get a smooth appearance of the local pass output image. For the final combination the image from the lighting stage is attuned to the image from the local pass.

4.3 Summary

In this chapter I have presented the details of the implementations of the methods presented in Chapter 3. I outlined the architecture and decisions I used to implement my approach followed by details regarding the specific steps of the single techniques. For development of my algorithms I used my own framework. This framework used for testing and the research of the proposed algorithms is called "werkstatt". Additionally, in order to tap into the full process containing garment simulation and design I had the opportunity to work with the program "Vidya" of the company Assyst. It is a complete virtual prototyping system to assist the communication between pattern maker and designer in order to show a fresh garment design virtually without the need of a physical prototype. Here, in addition to development and testing in my own framework, the material system, parts of the hybrid ambient occlusion technique and tesselation method were integrated into Assyst Vidya for testing in conjunction with real live garment production. Aside from a simplified simulation system, test data for the use as input for testing the algorithms was generated using Assyst Vidya and Autodesk Maya. This way a wide variety of complex input data could be generated. Results based on these implementations and generated data are shown in the next chapter.

5 Results

This chapter shows results achievable with the techniques and their implementation I have presented in the previous chapters. The results are centered on the two main aspects of my work, which are geometry processing and visualization in the context of virtual prototyping of garments. For geometry processing I differentiate between objects not related to a garment's surface (macro objects) and objects living on the cloth's surface (appliqués). This is accompanied by a tessellation scheme to improve the surface sampling for low resolution surface objects. On the rendering side, I show results of the two self occlusion techniques which I use to handle self occlusion separated into low- (material and geometry-based occlusion) and high-frequency (screen-space based occlusion) occlusion.

5.1 Geometry Processing

The results of geometry processing are organized along the domains of macro objects, appliqués and tessellation. Macro objects are a field of application for which I demonstrate the results of my method to handle accessory items of the avatar. The results of my method for details of the cloth surface take effect in the area of handling appliquès. In the tessellation part I present results related to surface refinement of objects used for surface detailing and the effects of smoothing the depth values for SSAO.

5.1.1 Macro Objects - Cuboid-Based Deformation

I have tested the hierarchical deformation system with several scenes (see Figures 5.1, 5.2, 5.3). I focused on two aspects in my implementation. Besides measuring the speed difference between trilinear transformations and linear transformations I had to differentiate between the GPU and the CPU part. In order to measure the vertex throughput in the GPU I used a high polygon model as deformation target. The high polygon count was created by attaching several highly tessellated spheres to an animated object. The chain of spheres is deformed according to the animation of the base mesh. For the CPU side I created a helix using a large number of small objects to measure the composition throughput. In both tests I compared linear vs. trilinear throughput in frames per second.

The tests were performed on a system containing a GForce 8800 and an Intel Core 2 Quad 6600. The tables show a comparison between the use of trilinear transformation and the linear transformed scene in frames per second. Additionally, the number of objects and triangles are exposed. The software was implemented in OpenGL. All geometry was

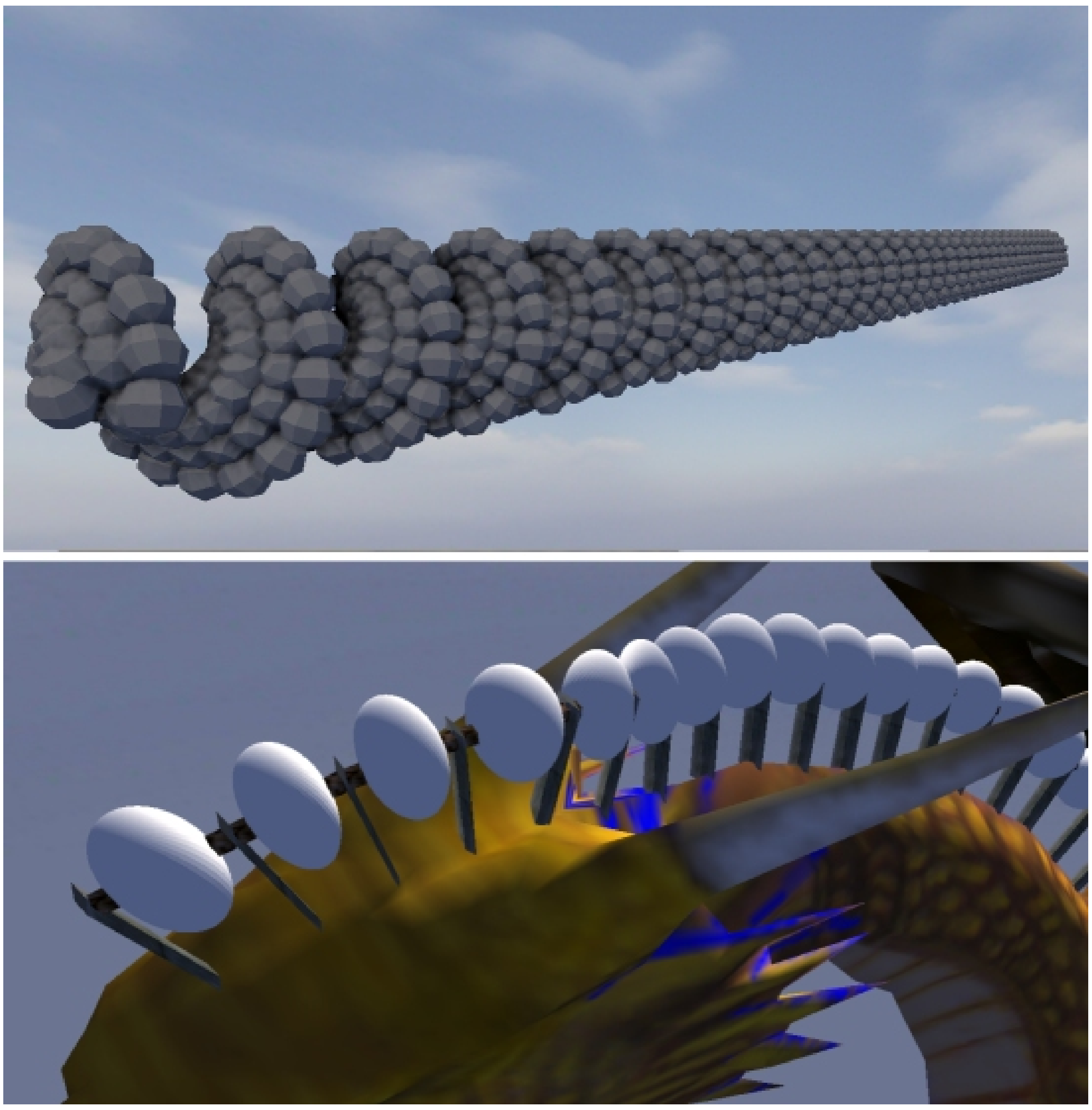


Figure 5.1: The scenes I used for experimentation: For the CPU tests, I created a helix consisting of many small low resolution spheres to create a high amount of composition. For testing the vertex throughput of the GPU I attached some high resolution spheres to an animated model. These spheres are deformed with respect to the animation, using my method.

stored in Vertex Buffer Objects (VBO) on the GPU side. The composition of the trilinear transformations within the scene graph hierarchy is performed on the CPU. Deformation of the geometry leaf nodes is done per GPU vertex shader. The trilinear transformation part of the used shader is roughly two and a half times more complex compared to the four scalar products required by a linear transformation.

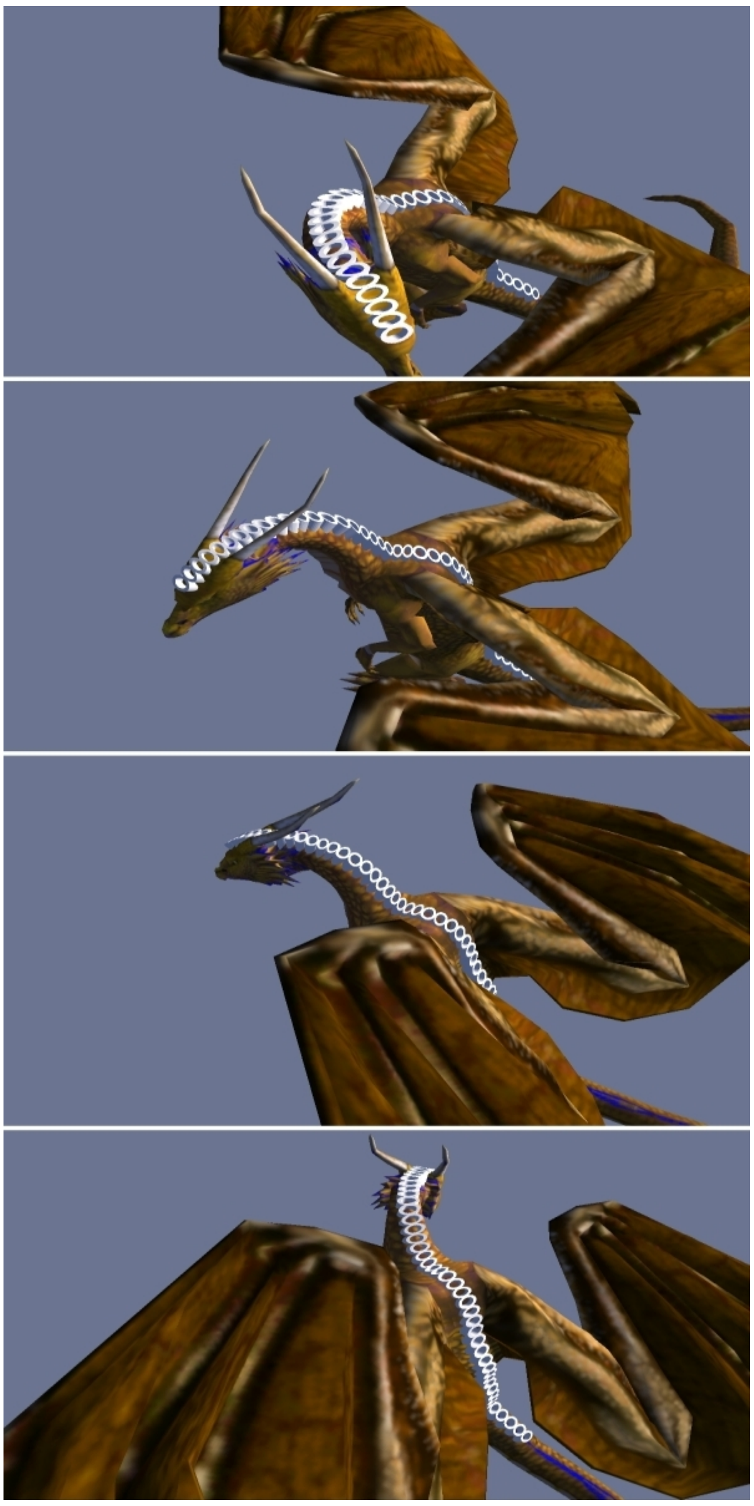


Figure 5.2: Results: 4 Frames from the vertex-animated figurine with a geometric attachment on its back, demonstrating the deformation of the attached geometry. The whole process is directly managed by the scene graph.

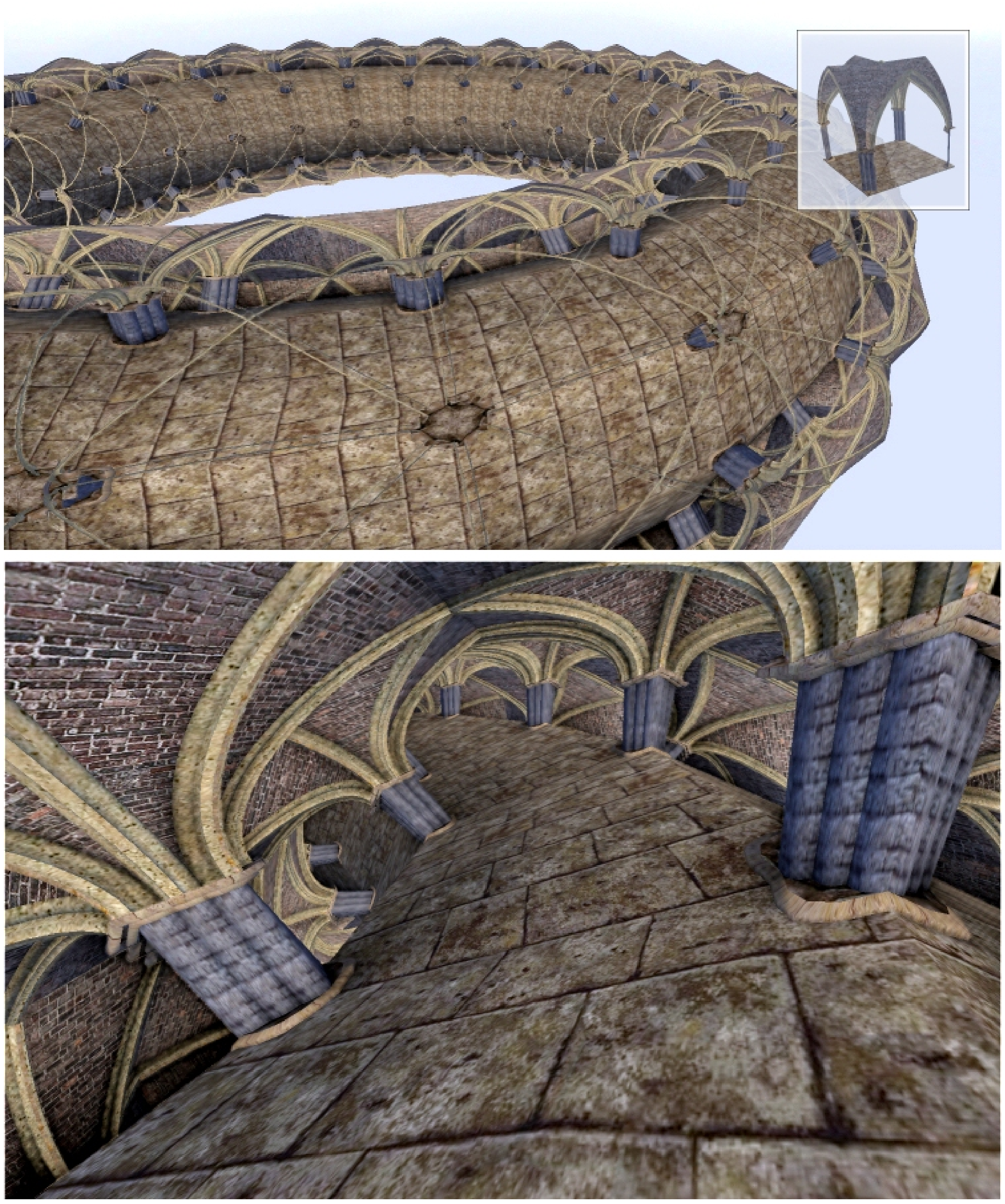


Figure 5.3: A geometric tile (top right) of a hall mapped on a torus (top). View of the 3D model (bottom). The embedded transformation system allows direct deformation of the geometric tiles to achieve a seamless (C_0) joining at their borders.

An experiment was performed with an object composed of 1.5 million triangles to compare the GPU side of the transformation stage. Even though the vertex shader for the trilinear

(FPS/T) transformation is more complex than the linear (FPS/L) one, I achieve more then 75 percent of the frame rate required by the linear transformation vertex shader.

scene	objects	Tri.	FPS/L	FPS/T	perf.
spheres	185	1.5M	19.5	14.9	76%
helix	4617	164k	16.5	13.3	80%
helix	2566	656k	10.8	8.4	77%

The composition test (CPU) had different results. Using a large number of low poly (around 100 faces) models resulted in a performance of 80 percent (second row). Increasing the polygon rate by decreasing the amount of objects (third row) shifts the performance to 77 percent (third row). As expected, this behavior is a result of moving slowly the amount of operations from composition (CPU) towards transformation (GPU).

The results show the tendency of the algorithm to start being a bottleneck when applying a lot of objects on separate locations since the CPU is needed to generate the initial control grid. This is exactly the problem when dealing with appliqués attached onto cloth surfaces. On the other hand it allows to freely choose arbitrary deformations as long as the number of objects placed this way in the scene is low. Therefore, this method is well suited for attaching macro objects onto an avatar, but not the best choice for attaching appliqués onto a cloth's surface. This is the domain of deferred warping, which has a much better performance in this area, as it can be seen in the following section.

5.1.2 Decoration - Deferred Warping

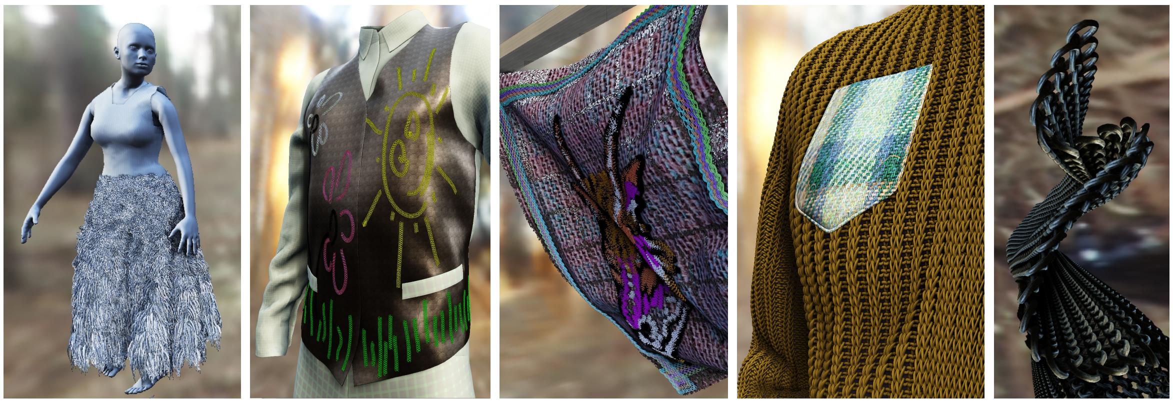


Figure 5.4: Deferred warping allows to attach arbitrary detail geometry to an animated or manipulated surface in real-time which is required in interactive garment simulations. Details can be attached at single points (a, c), along a curve (b, c) or on an area (d, e). In the latter case, the original geometry was even completely replaced with geometry representing the fabric structure.

In difference, deferred warping offers much better scaling with increasing amounts of attachments and their complexity.

Figure 5.4 shows results for point (a, c), curve (b, c) and area attachments (d, e) as well as combinations of them (c).

I tested the performance of my approach on an Intel Xenon X6550 (2,67 GHz) system with a NVIDIA GeForce GTX 580. For the test I attached a button geometry containing 928 triangles up to 65000 times to a target surface resulting in up to 60.1 MTriangles. This test was performed with a point, an area and a curve attachment shader. The results are shown in Figure 5.5. The complexities of all three shader programs are almost equal. The curve shader has to evaluate cubic Bézier splines and is therefore a bit slower. I executed the area shader using software instancing. The same batch size was generated in this case by aggregating the single tile into one geometry. This shows that hardware instancing yields a significant performance gain. Please note: for this test no culling or other acceleration techniques were used. The test demonstrates how much performance can be expected using a naive implementation of the algorithm.

In difference to the cuboid-based transformation, deferred warping does not require much help from the CPU at run-time, which allows to deal with large numbers of attachments and geometric complexity. While cuboids are the more general approach the limitation of deferred warping onto parametrized surfaces allows a complete GPU-based implementation, removing the bottleneck between CPU and GPU. In order to maintain flexibility of the way

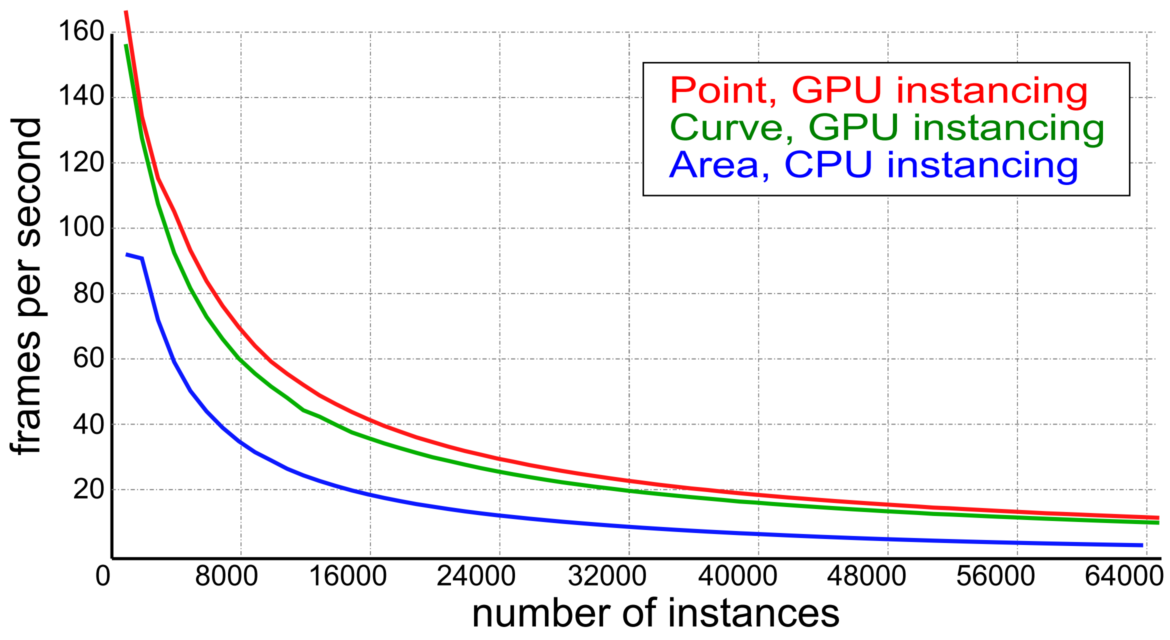


Figure 5.5: Performance of deferred warping using an NVIDIA GeForce GTX 580 for massive instancing of a 928 triangles model in conjunction with my three attachment types. At the maximum, the pipeline handled 60.1 MTriangles. 65 measurements were taken per curve. CPU instancing was performed by assembling 256 tiles to one large geometry block prior to rendering. GPU instancing was realized via the corresponding OpenGL extension.

objects are attached three typical attachment types were identified and implemented (see Figure 5.6).

The area attachment type allows for replacing of the complete surface. This can be used to replace a cloth model which consists of a simple triangle mesh by a complex knitwear model (see Figure 5.7) at interactive frame rates. The curve attachment shader can use detail geometry which is repeated along the z axis. The geometry is repeated n times via instancing and is mapped to the length of the curve (see Figure 5.8). The curves I use in the examples are cubic Bézier splines, read from scalable vector graphics (SVG) files. If I implement $\mathbf{f}_{\mathrm{uv}}$ as a time-dependent mapping function to determine the texture coordinates, I can even animate the source geometry on a waving flag (see Figure 5.9). By using a post-processing step deformation of the attached geometry is possible (see Figure 5.10).

All renderings for tests were performed using the proposed rendering approach. The SSAO component for handling the local self shadowing of the high frequency details is based on the work proposed by Hoang et al. [HL10]. The cloth simulations shown use the finite element method (FEM) of Etzmuss et al. [EKS03]. The adaptive cloth simulation in Figure 5.8 was computed by using the method presented by Bender and Deul in [BD13]. The suit and the knitted shirt are based on the Assyst Vidya VP system. The deformations

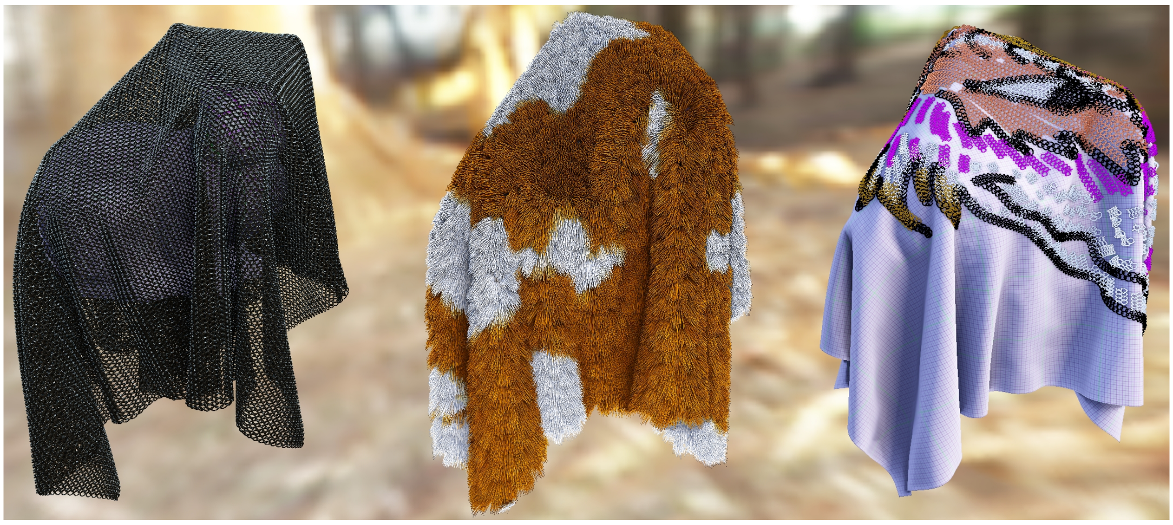


Figure 5.6: Examples of geometry processing for the different attachment modes. Left: Area. In this example the whole surface was replaced by a detail geometry, covering the original surface. Middle: Point. The fur like appearance is created by attaching 65000 strands at random positions onto the cloth surface. Right: Curve. A loop like geometry was placed alongside 2D spline curves which were warped onto the cloth surface.

of the armadillo seen in the video (and in Figure 5.10) was simulated using shape matching [MHTG05].

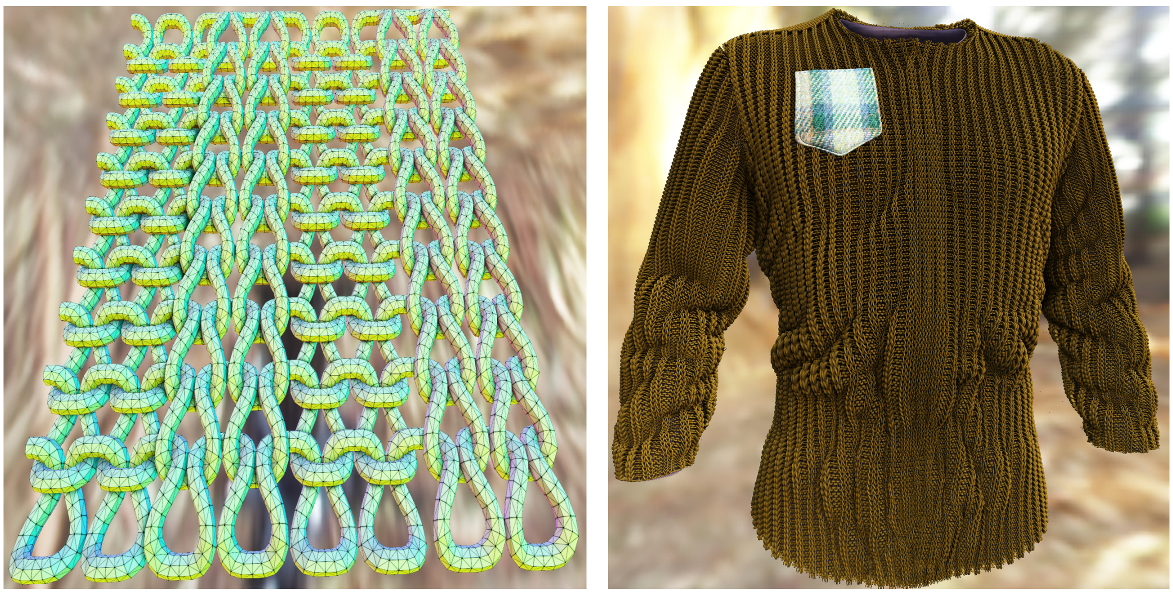


Figure 5.7: Knitwear model (right) created by replacing the original cloth mesh by the knitwear pattern (left) using area attachment.

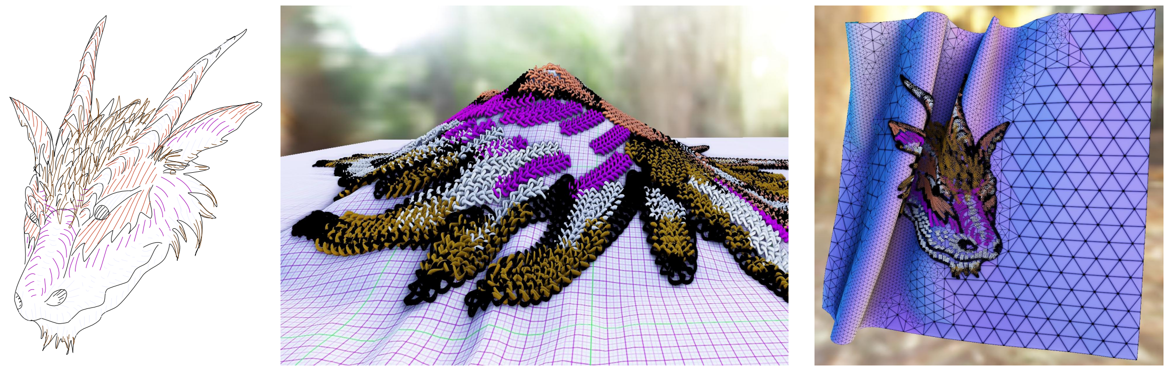


Figure 5.8: My implementation handles line drawings (left), which are specified as vector images. Each curve is evaluated in the vertex shader to attach a source geometry repeatedly on the surface (middle). Deferred warping is independent of the mesh topology of the target surface and can therefore be directly used in adaptive cloth simulations (right).

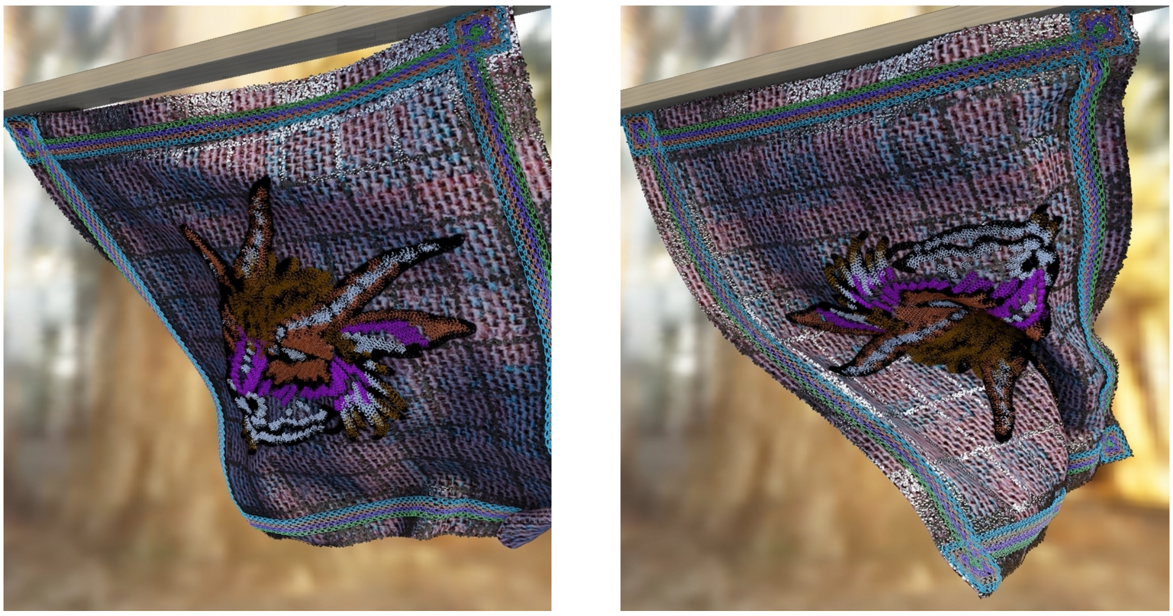


Figure 5.9: Waving flag with a rotating logo as a demonstration of a time-dependent mapping function.

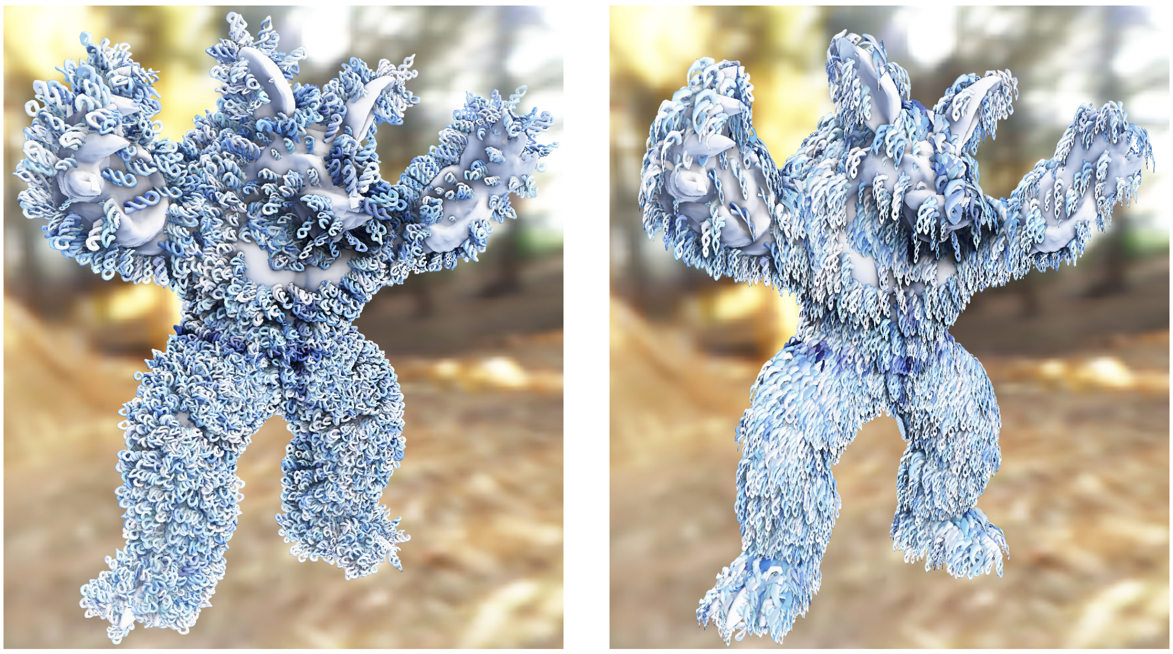


Figure 5.10: Armadillo model with point attachment based fur. The detail geometry was attached without (left) and with my fur post-processing step (right).

5.1.3 Smoothing - Tessellation

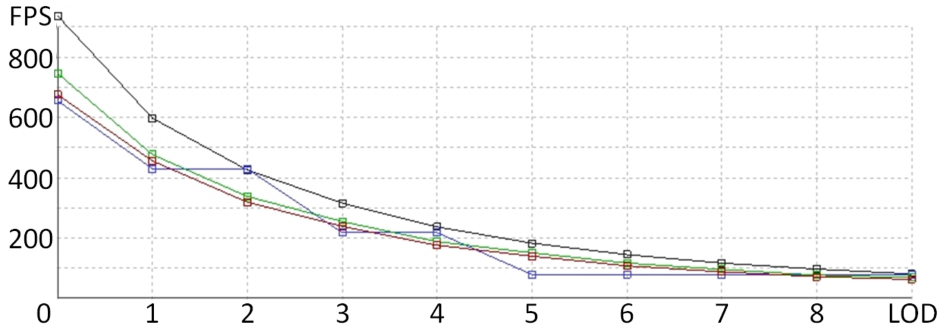


Figure 5.11: Comparison: The black line represents a simple uniform refinement scheme like it is used by the pn triangle scheme. The blue line represents a dyadic adaptive semiuniform tessellation scheme. The green line represents my method without warping applied. The red line shows the same experiment with warping enabled. Y axis: frames per second. X axis: refinement level. Since the dyadic scheme cannot represent all refinement levels it's curve has a stair-like structure.

I tested my concept on two different hardware architectures, both capable of performing geometry operations. An NVIDIA GTX 280 was chosen as a representative of the shader model 4 league. Additionally, an AMD Radeon 5770 was used as a representative of the shader model 5 league. Besides differences in computational power, the second GPU has a larger geometry shader output buffer for generated geometry data. This allows us to achieve higher tessellation levels in hardware.

I compared my solution against an implementation of dyadic semiuniform adaptive tessellation [DRS09] and pn triangles [VPBM01]. All tests were performed with geometry shader based implementations to create comparable results.

I have tested and compared different refinement strategies and LOD computation methods. Additionally, I performed a worst-case scenario test for my scheme (see Figure 5.11). The test was performed on the NVIDIA system. In addition, I have compared the performance of the ATI and the NVIDIA-based systems (see Figure 5.12). As a reference I used a simple uniform refinement strategy (black line) which is, in practice, a geometry-shader-based implementation of the pn triangle technique. For my method I tested two candidates. In number one (green line) the coordinate warping was disabled. In number two (red line) it was enabled. Additionally, both create the maximum triangles for a given refinement level using the longest path in the shader. This way a worst case scenario was created. The best case for my method is a uniform grid. Thus, in practice, performance ranges between the values of the red (green) and the black line. In my tests warping adds a loss of approximately nine percent of frames per second to the overall process. As seen in Figure 5.13, 5.14, 5.15 and 5.16 my proposed scheme grants a much smoother grid density control. Especially in the tablecloth and torus example the sagging of the tablecloth in its center is clearly noticeable.

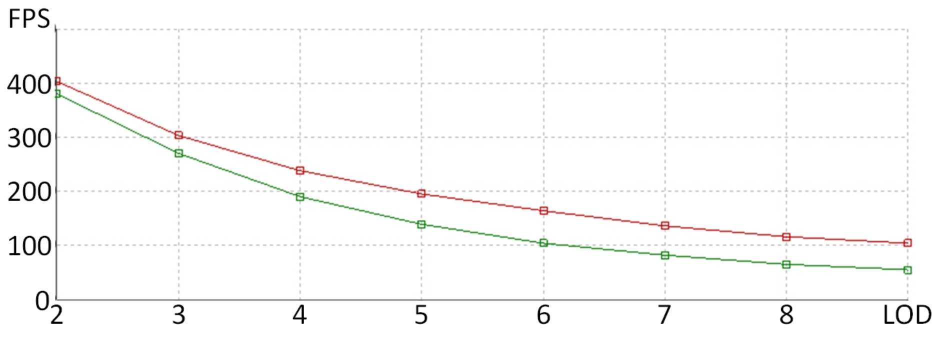


Figure 5.12: Comparison of a DX11 class ATI 5770 (red line) to a DX10 class NVIDIA GTX 250 (green line). The ATI performs much better on the geometry shader tessellation even though it is the less powerful GPU in other disciplines. A geometry shader program performs much better on DX11 class hardware in comparison to DX10 class hardware. Thus, both GPUs perform similarly at low tessellation levels. At higher tessellation levels they diverge strongly.

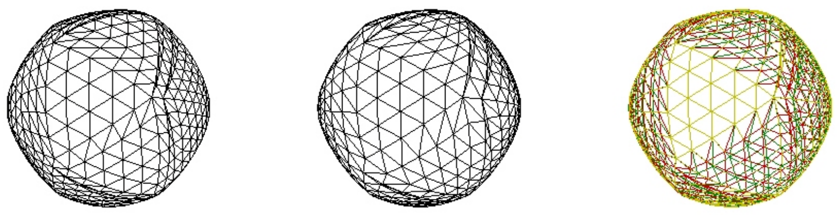


Figure 5.13: Coordinate warping used on an icosahedron: The low polygon count of the initial mesh and its high curvature introduces high changes in the LOD level for the vertices. In the left image the refinement without warping is visualized. The middle image shows refinement in conjunction with warping. The right image visualizes the difference of both images. The result is a smoother transition of the structure size on patch borders.

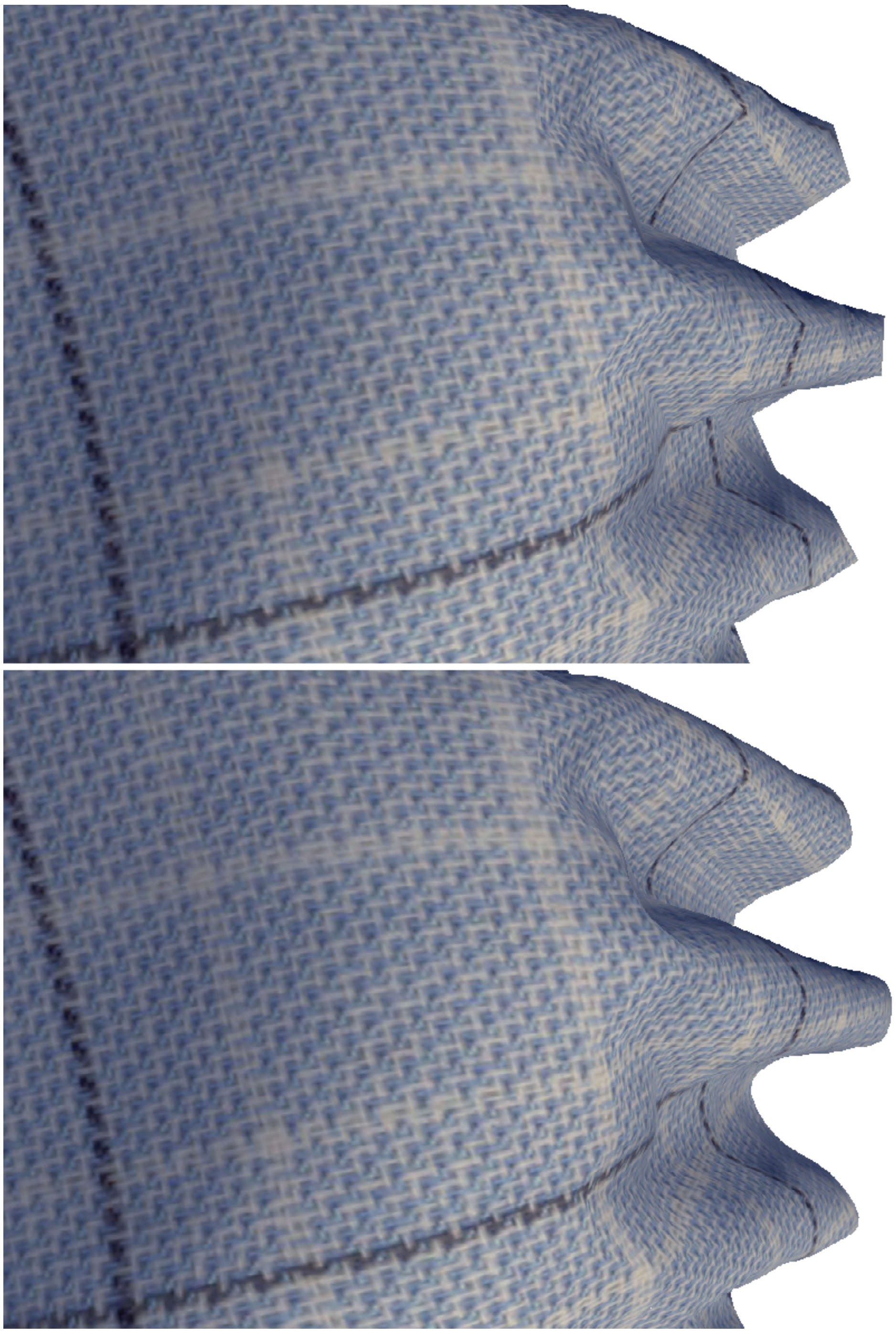


Figure 5.14: Overhead view rendering of a tablecloth draped over a torus: The upper image shows the direct output of the cloth simulator. Since I want to perform real-time simulation the resolution of the simulation mesh is quite low. Using my tessellation method inside the rendering pipeline smoothes the appearance of the folds of the tablecloth noticeably (lower image).

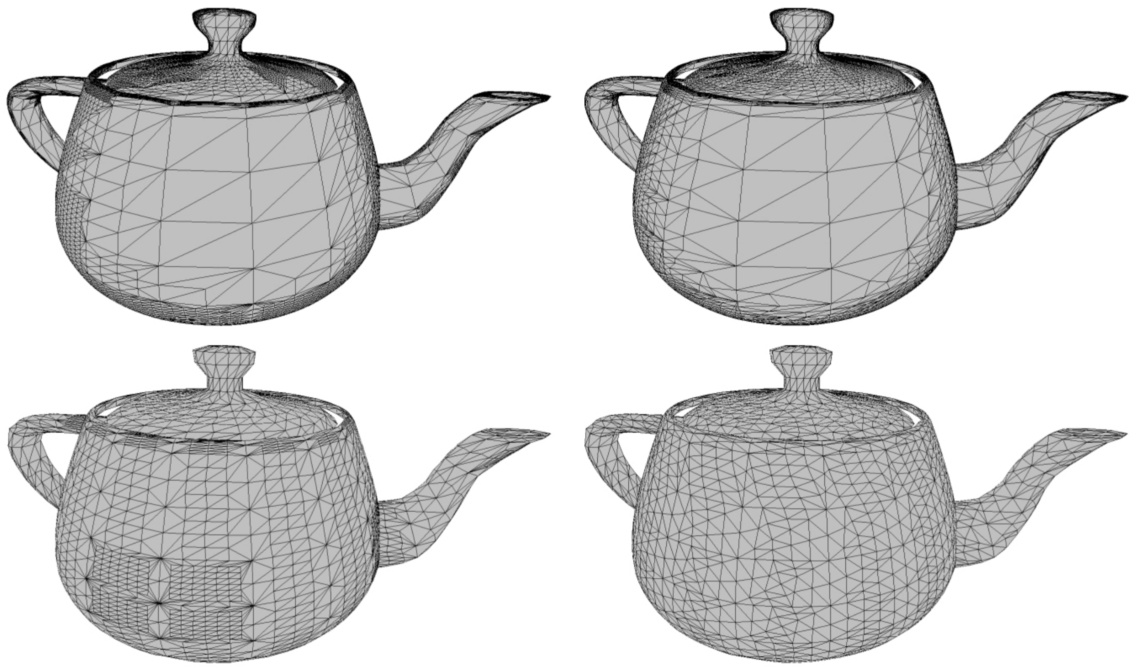


Figure 5.15: Teapot example: the left side was created using dyadic adaptive semi uniform tessellation, the right side using our algorithm. Two different metrics for LOD computations were used: In the upper row I used a distance- and angle-based scheme. In the lower row the LOD is chosen to reach a maximal edge length. In the 3D teapot model the triangles with largest edge lengths can be found on the side surfaces. Thus, a maximum resolution is reached here which is slightly over the trigger level of the next dyadic refinement level. This results in the sudden increase of resolution visible in the bottom left image.

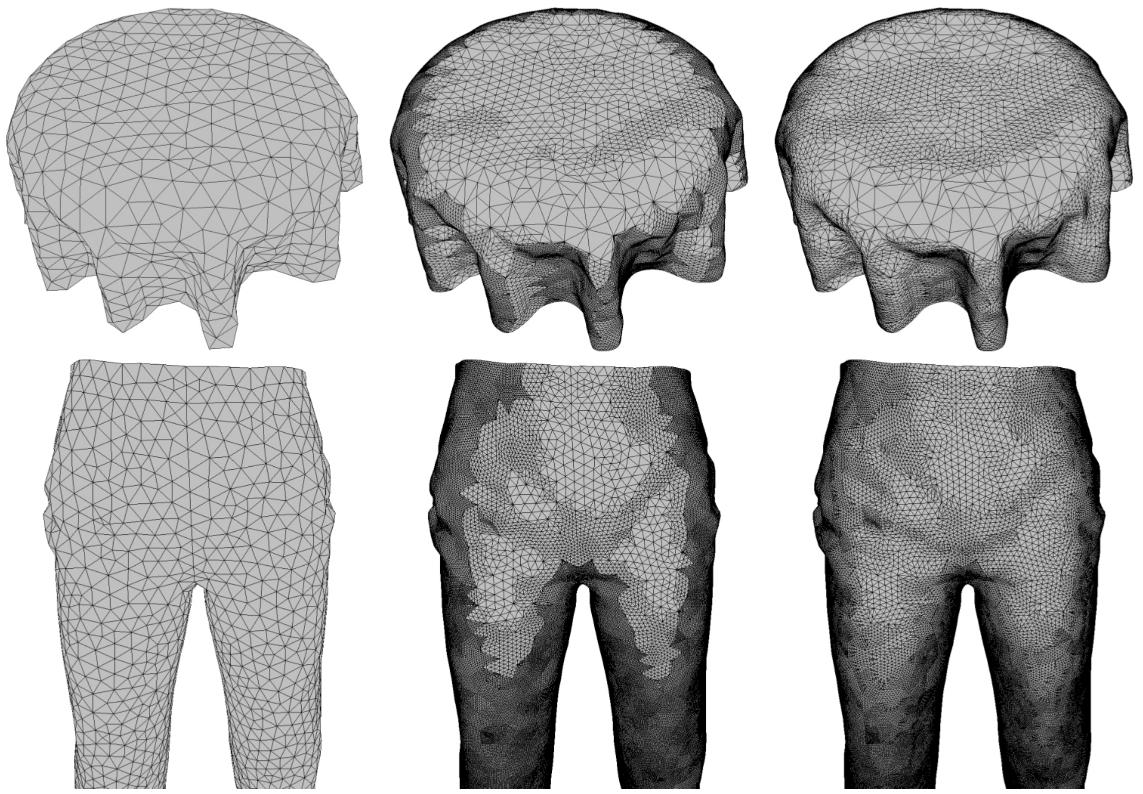


Figure 5.16: Cloth example: Tablecloth on a torus (top row) and a pair of trousers (bottom row). The left side shows the original mesh. In the centre column I applied dyadic adaptive semiuniform tessellation. On the right side my algorithm was used. Using a non-dyadic scheme a finer granularity of possible refinement steps is applied. In conjunction with the warping scheme proposed a much smoother appearance of the refined mesh is created.

Smoothing Cloth Surfaces for SSAO

The algorithm for smoothing depth data for SSAO operations on the garment surface performs well with nearly no impact on the frame rate of the overall rendering. The effect returns the original smooth appearance to the surface again, as it can be seen in the normal field generated per fragment by SSAO (see Figure 5.17).

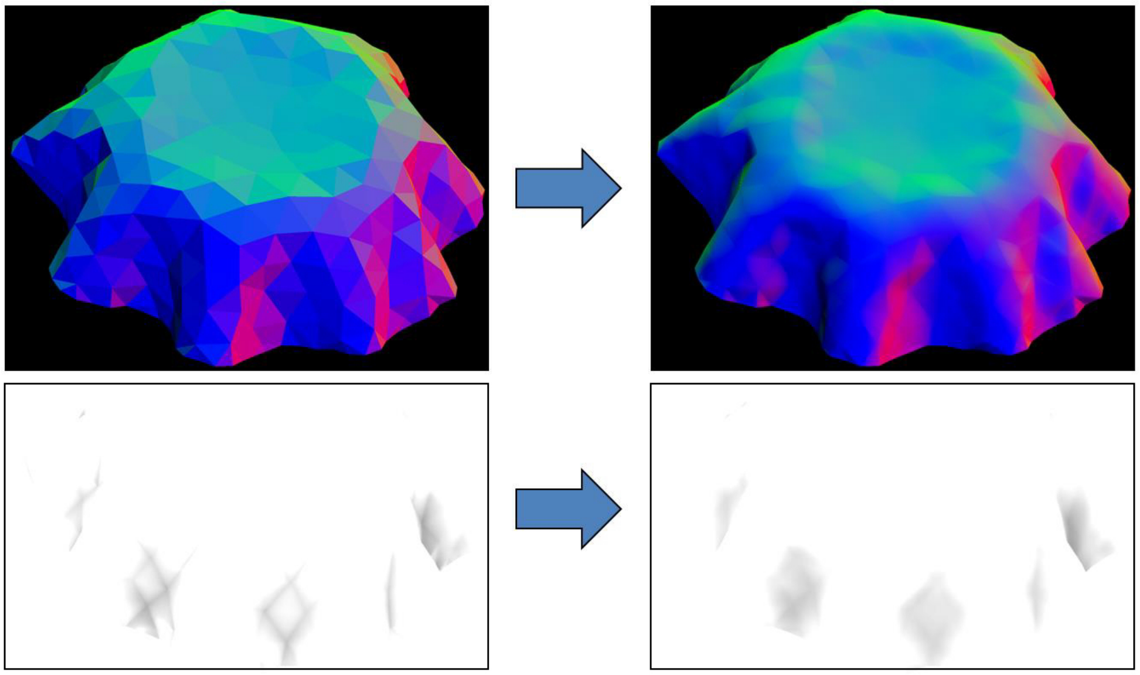


Figure 5.17: SSAO of a tablecloth without (left) and with use of smoothed depth values (right) utilizing per fragment patch evaluation. The upper row depicts the normals of the surface generated per fragment by SSAO based on the image depth values. The lower row shows the influence on the SSAO darkening only.

5.2 Rendering

The results for rendering are organized along the techniques of material and geometry based occlusion and screen-space based occlusion. Material and geometry based occlusion targets the results of rendering low frequency details like the cloth surface. Screen-space based occlusion on the contrary is intended for high frequency details and here I present results for combining both low frequency and high frequency parts. Results of the process can be seen in Figure 5.18.

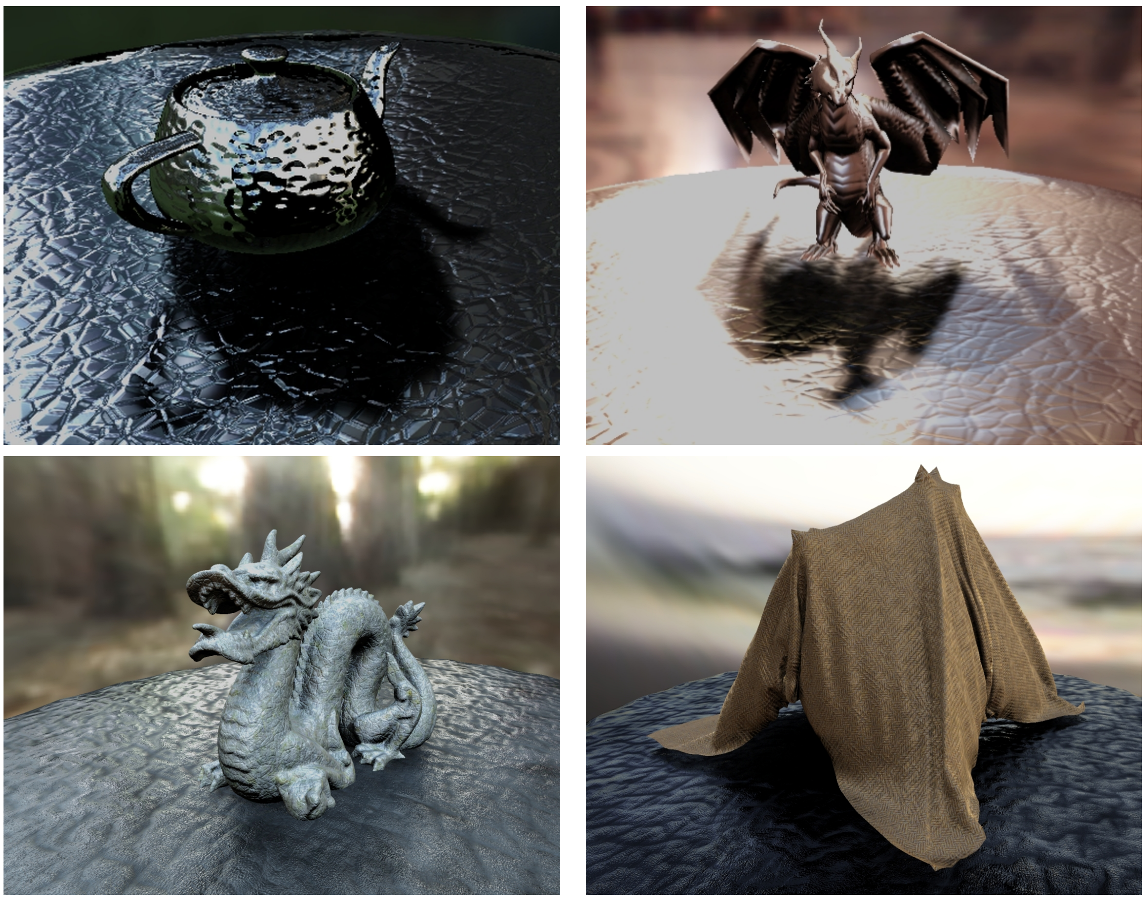


Figure 5.18: Several scenes rendered using this approach.

5.2.1 Image Based Lighting, Material and Geometry Based Occlusion

Filter Quality Embedding the whole material system into the mipmaps of the environment maps was not used before in conjunction with natural illumination. Therefore, I first want to investigate the quality and performance of my approximation by filter functions. To compare my embedded reflection model (see Equation (3.42)) with its approximation differential images can be used. However, in my case it is possible to compare the quality of the basic

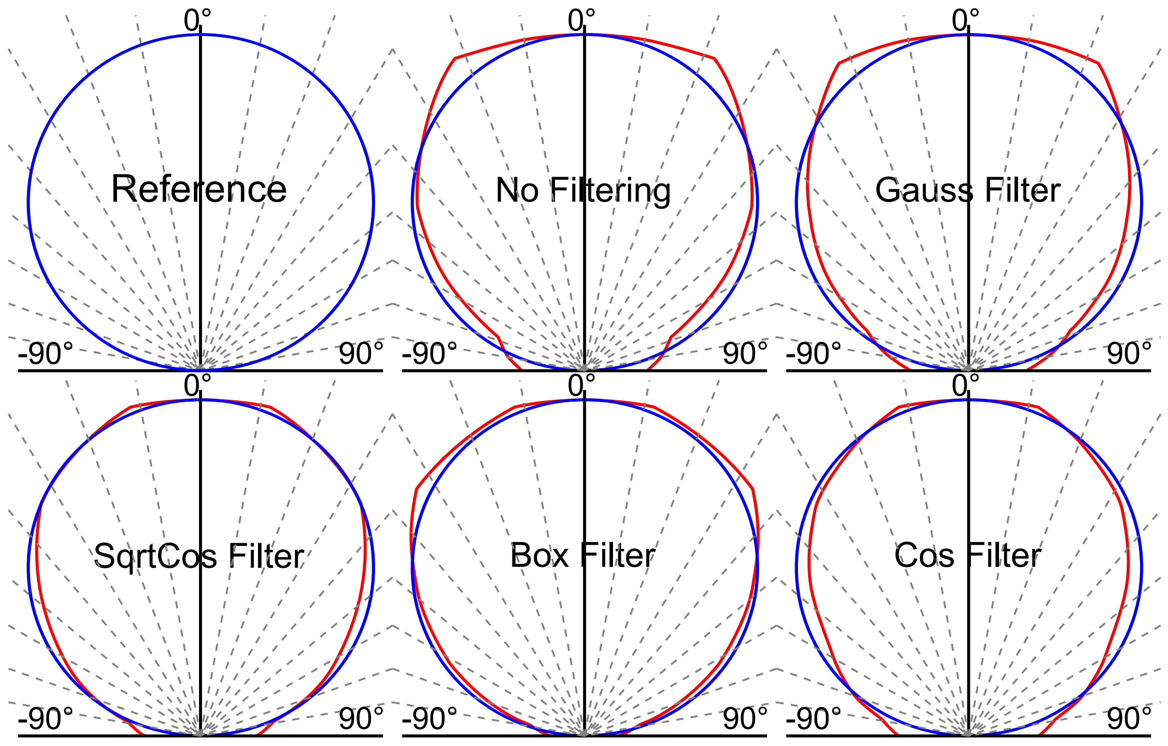


Figure 5.19: Cross sections of 3D radial plots of the effects of the filter functions used to approximate lighting lobes. The blue lobe corresponds to a given diffuse reflection while the red lobes are the computed approximations. To increase the quality I additionally apply different 5×5 low pass filters (gaussian, cosine, box and square root cosine filter).

functions generated by both methods directly.To perform this comparison I have fixed the direction vector. For the approximation this means having a black environment map with one white pixel which is then filtered. To compare the generated lobes I fit their volumes by finding the corresponding mipmap level for a given glossiness. The remaining differences in the volumes are then used to compute the relative error. Note that this approach can also be used to calibrate the approximation since the lobe volume is characteristic for a given glossiness. I tested five different filter functions. For $s_{Phong} = 1$ the cross sections of the radial plots of the function compared to the mipmap based approximation are shown in Figure 5.19. In this figure an approximation of $s_{Phong} = 1$ (blue curve) is performed. The red curves represent the approximation of the different filter functions. Especially for the case $s_{Phong} = 1$ I aimed to get a good approximation since it corresponds to pure diffuse reflection. In my test the box cosine filter performed best with an error of roughly 4%.

In the next step I investigated the deviations introduced by the linear interpolation between the mipmap levels. These deviations are shown in Figure 5.20. The choice of the filter function gradually changes the impact on the deviations. As it can be seen, no filtering (dotted violet) and the box filter (red) both introduce high spikes of relative error for s_{Phong}

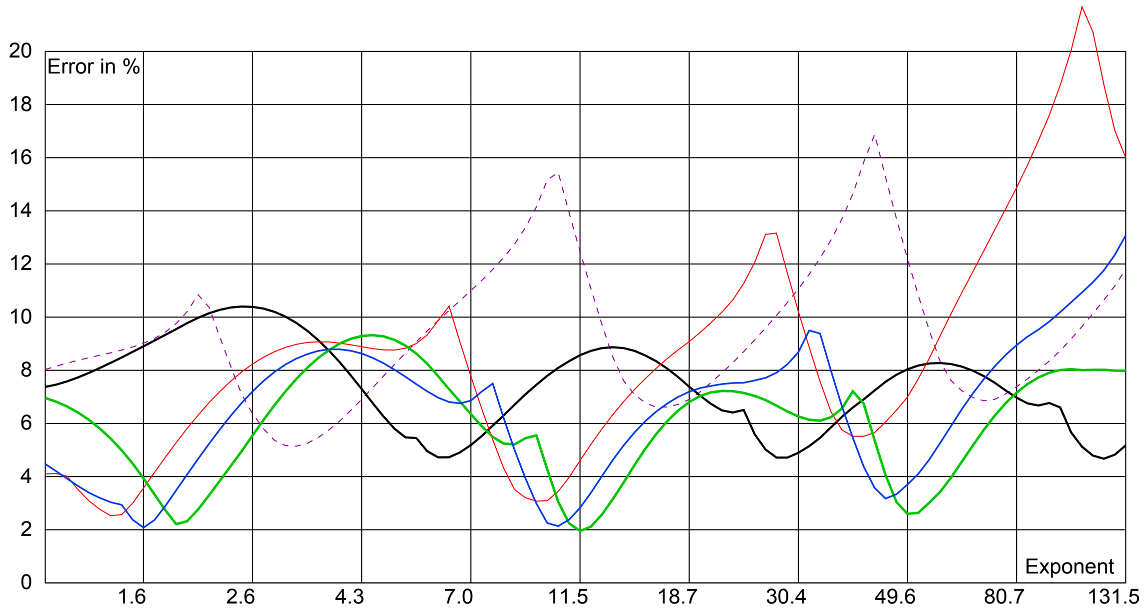


Figure 5.20: Results for different filters: no filtering (dotted purple line), gaussian filter (black), box filter (red), cosine filter (green) and the square root cosine filter (blue). The relative error is computed as the difference between the target lobe and the approximated lobe when both have the same volume. The error is given in percent of the target lobe volume.

increasing. The gaussian filter (black) is stable but has a high overall error rate. Using the cosine filter (green) creates results which are always below 10% of the error but have strong deviations for the important diffuse reflection case. The blue curve depicts the behavior for the square root cosine filter. This filter shows promising results but it tends to introduce high deviations when the exponent takes on high values which are the case for representing mirroring reflection. In practice the deviation for very high exponents is not that critical since it is near the transition to the reflection case. Note that the resolution of the environment map is the limiting factor of my approximation. For high exponents of the cosine function the resolution limit is reached which leads to stronger deviations. This can be seen on the right side of Figure 5.20. Due to these results I have chosen to use the square root cosine filter. It performs nearly as good as the box filter for the diffuse case but has a much better behavior when increasing s_{Phong}. A comparison to often used Ambient Occlusion techniques demonstrates that considering the reflectivity for the occlusion computation is clearly necessary in case of high glossiness (see Figure 5.21).

Performance

All tests were computed on a desktop PC with an AMD Phenom II X4 955 CPU at 3.2 GHz, 4 GB RAM and an AMD Radeon HD 6850 GPU at a clock rate of 775 MHz with 1 GB

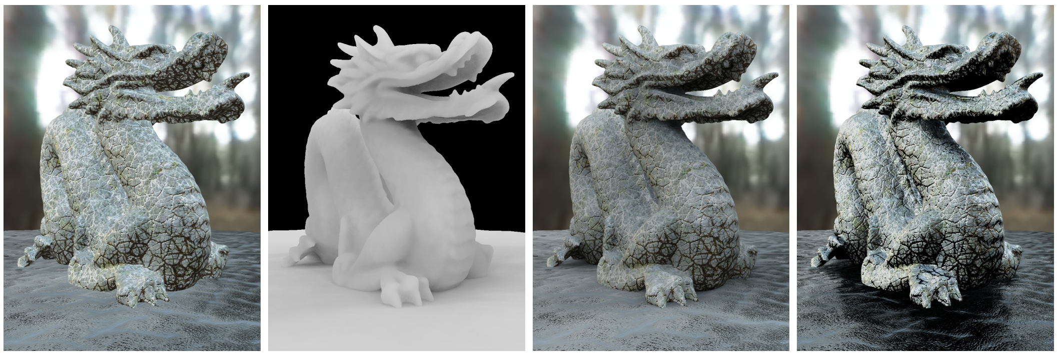


Figure 5.21: Comparison between my method and ambient occlusion (AO). From left to right: No occlusion, ambient occlusion shadow only, with color, my method. My approach considers the glossiness of the surface in the occlusion computation which leads to a completely different visual appearance of the object.

# lights	PEMs		VSMs				
	128^2	256^2	64^2	128^2	256^2	512^2	1024^2
1	1.5	6	0.015625	0.0625	0.25	1	4
12	18	72	0.1875	0.75	3	12	48
32	48	192	0.5	2	8	32	128
64	96	384	1	4	16	64	256
128	192	768	2	8	32	128	512
256	384	1536	4	16	64	256	1024

Table 5.1: Memory footprint (in MB). The PEMs use 16Bit floats with four color channels (*GL_RGBA*16*F*) while the VSMs have only two channels (*GL_RG*16*F*).

memory. Due to the high number of environment maps multi-pass rendering was used applying 12 PEMs and their corresponding shadow maps per pass.

The main performance factor is the number of subdivisions used (see Figure 5.22). Each region has its own PEM and a shadow map which have to be stored on the GPU. An overview of the necessary passes and the memory consumption is shown in Table 5.1

The results of rendering scenes while using the box filter are shown in Figure 5.18 and within the context of garment simulation in Figure 5.23.

As an alternative to the usage of cube maps I have considered parabolic mapping. The parabolic maps can be scaled to fit the region used by the PEM which reduces the amount of memory required significantly. This can be easily explained: Assuming that all six sides of a cube map have a resolution of 512×512 pixels and consist of 64 regions and that the corresponding 64 environment maps are stored with mipmaps and floating point color

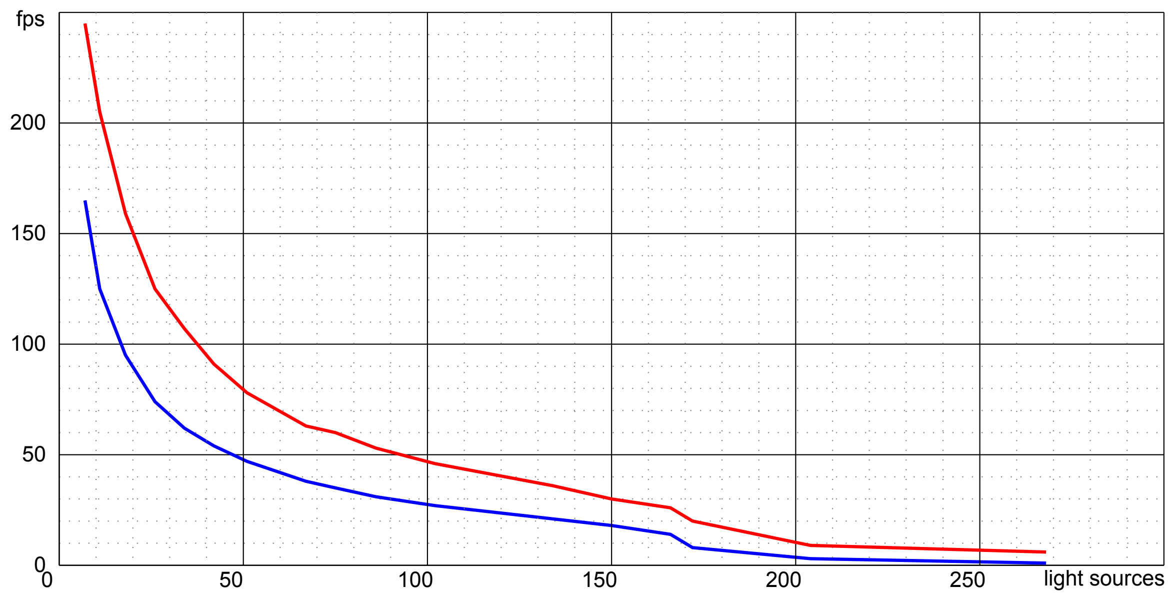


Figure 5.22: Frames per second for the teapot (red) and dragon (blue) scene with increasing number of light sources. Notice the drop for both scenes at about 200 light sources when total memory consumption becomes too large to completely fit into the GPU memory space.

values, consequently 1.5 GB of memory are roughly required. Since the parabolic version only needs to cover the parts containing information, I end up using a resolution of roughly 256×256 per parabolic map. Using 64 regions consumes about 70 MB of memory. However, using parabolic maps is much more complex than using cube maps. Due to their different sizes they behave differently in the approximation. Thus, for each map different values have to be applied in order to find the corresponding mipmap level for a given g. Therefore, the case $s_{Phong} = 1$ is not a constant mipmap level anymore which complicates the lookup. Additionally, the mipmaps correspond to different areas of the original environment map. This complicates the lookup even further and is the main reason why I decided to use a cube map-based environment mapping instead.

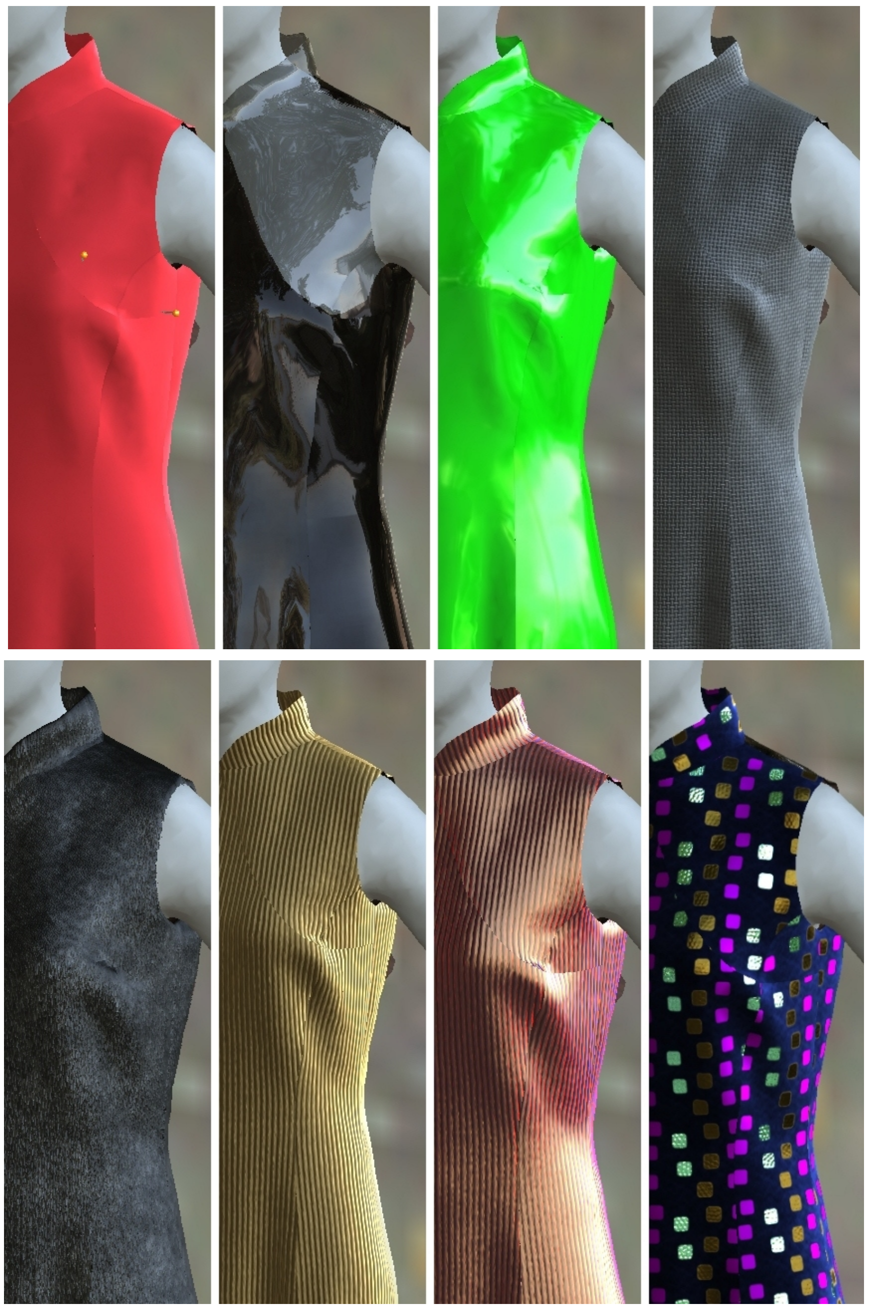


Figure 5.23: Several materials applied to a garment model using the lighting model presented in this work. The images show its use in conjunction with other methods for modeling materials within real-time graphics: Parameter maps allow a control over the material properties within a single material while bump-maps give the material a macroscopic structure.

5.2.2 Screen Space Techniques - Hybrid Occlusion

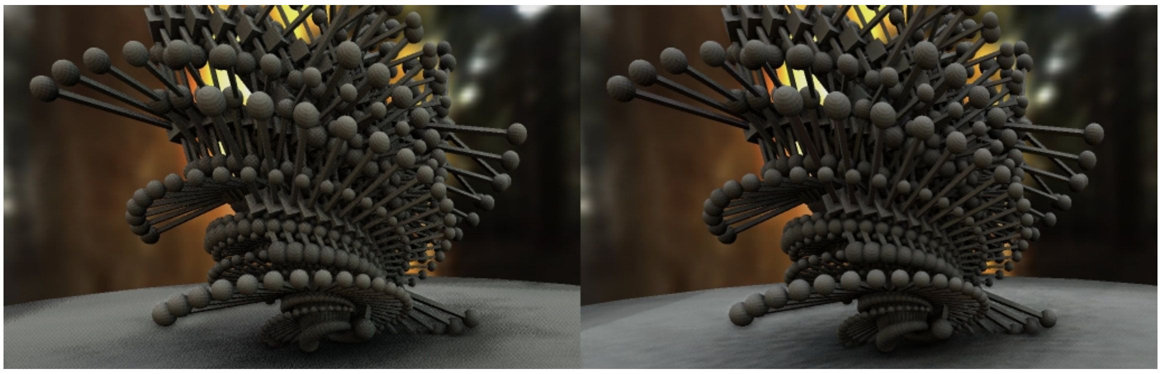


Figure 5.24: Comparison between different amounts of light sources in the global pass. Left image: 128 light sources at 15 fps. Right image: 512 light sources at 6 fps.

The presented hybrid technique for calculating the self-shadowing of a fully dynamic scene achieves interactive to real-time frame rates. A result for different numbers of light sources can be seen in Figure 5.24. With this approach I split the process into a local and a global independent method. This way I was able to combine the advantages of both. The methods can be chosen independently and are not limited to the ones I used. For the implementation of the two parts I presented a combination of a shadow buffer-based self-shadowing method and a screen space ambient occlusion algorithm. I have shown the capability of the local part to reduce light bleeding effects introduced by the method used in the global part. Additionally it adds shadowing details which are not covered by the global part. Both parts are evaluated on the GPU using a deferred shading approach which is decoupled from the scene complexity. In my implementation I reach around 6 FPS when using a light probe approximation of 512 light sources and around 15 FPS when using an approximation of 128 lights. The results show how the local and the global part improve independently from each other. Figure 5.25 shows how the local characteristics improve the missing high frequency detail of the coarse global shadowing pass. Additionally, light bleeding caused by insufficient shadow buffer depth resolution is less noticeable since it is masked out by the local process (see Figure 5.26). The combination allows to treat local and global shadowing in separate ways (see Figure 5.27), which allows to control the balance between speed and quality of the global part independently of the detail shadows (see Figure 5.24).

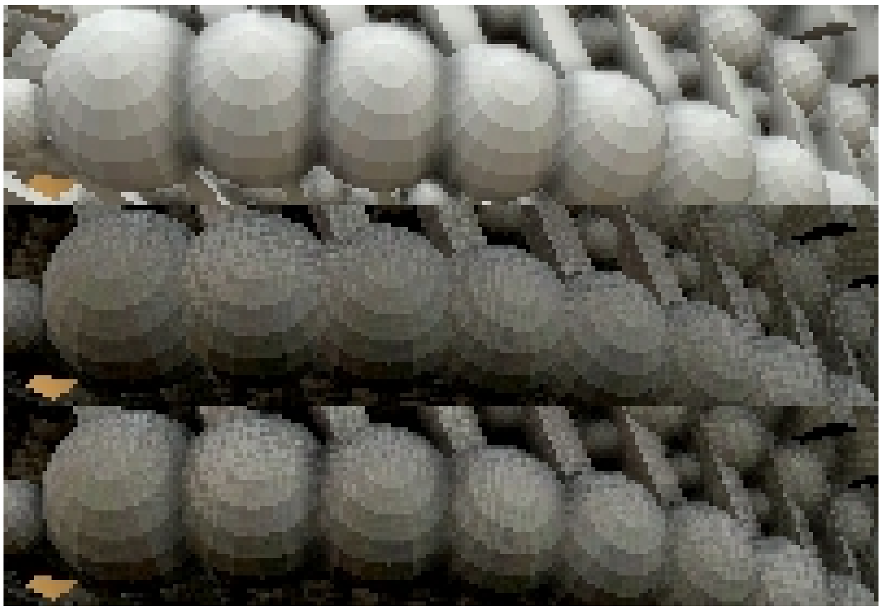


Figure 5.25: The effects of the details introduced by the local part. Top: local output. Middle: global output. Bottom: the final combination. Please note the difference of the shadows at the spheres between the middle image and the bottom image.

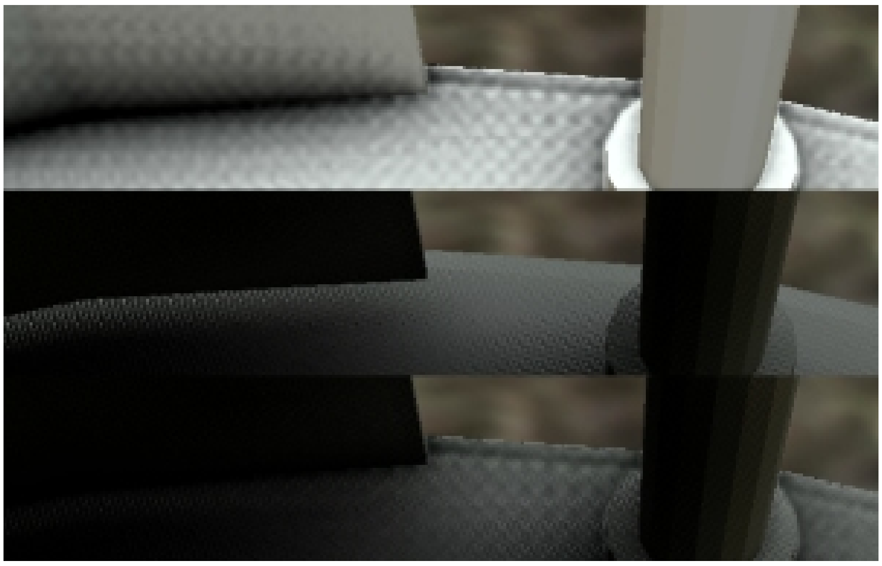


Figure 5.26: Light bleeding reduction introduced by the local part. Top: local output. Middle: global part output. Bottom: the final combination. The pattern-like artifacts visible in this images are introduced by the sampling method. Both shadow and ambience data is sampled by a dithering pattern which is visible within the magnified image region.

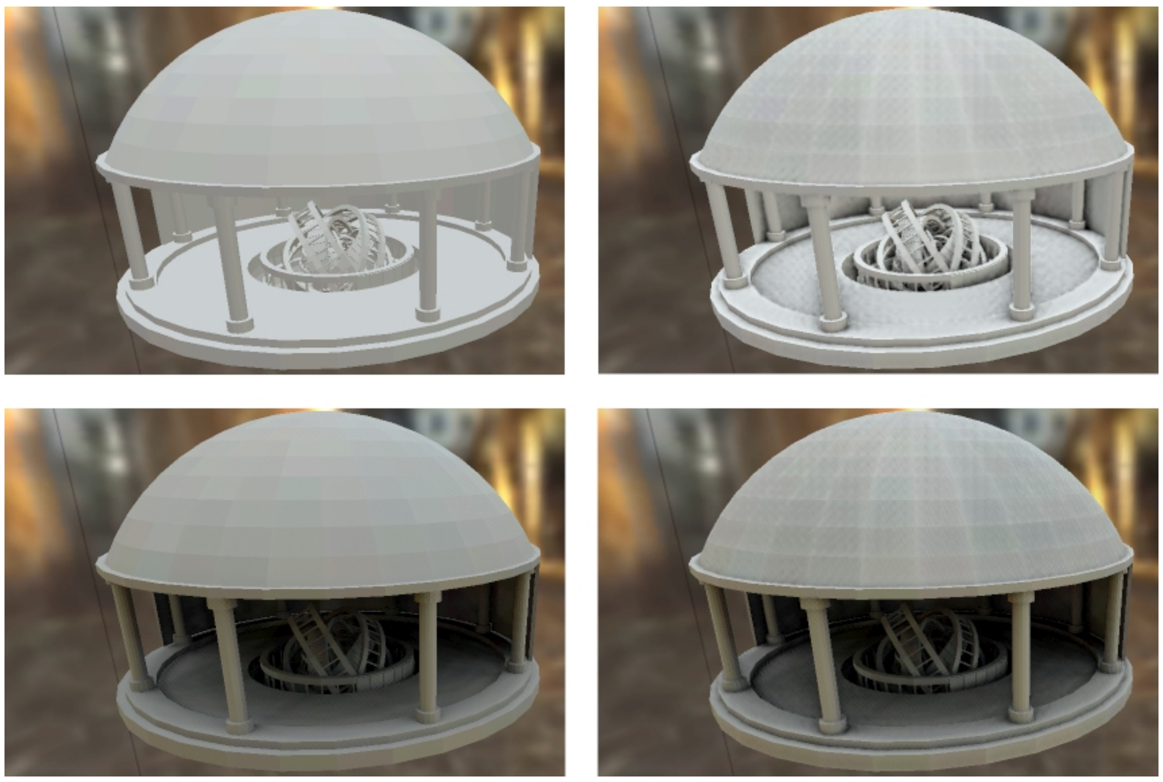


Figure 5.27: Comparison between the different effects of the hybrid ambient occlusion. Upper left: no self shadowing. Upper right: local part. Lower left: global part. Lower right: combination of local and global part. The light probe was sampled using 512 light sources. An average of 6 fps was achieved.

5.3 Summary

I have shown results achievable with the techniques and their implementation presented in the previous chapters. The results were focussed on the two main aspects of my work which are geometry processing and visualization in the context of virtual prototyping of garments. For geometry processing I differentiate between objects not related to a garments surface (macro objects) and objects living on the cloth surface (appliqués). This is accompanied by a tessellation scheme to improve the surface sampling for low resolution surface objects. On the rendering side I have shown results of the two self occlusion techniques which I use to handle self occlusion separated into low- (material and geometry-based occlusion) and high-frequency (screen-space based occlusion) occlusion. All techniques presented are capable of achieving real-time performance on current hardware and can be used in combination. This makes them well suited for use in the context of virtual prototype design. In the next chapter I will discuss the results and give an outlook over future development work.

6 Discussion and Future Work

In this work I have provided a method for interactive visualization of virtual garment prototypes with high surface detail count.

Nowadays, real time visualization of semi dynamic scenes like the ones found in computer games give a photo-realistic impression. However, the same is not the case for fully dynamic scenes like the ones found when interactively modifying or modeling objects under physical constraints in conjunction with high resolution details. For this scenario I have presented an approach, which allows to create a solution for the specific case of virtual garment design. It allows giving visual feedback for the designers interaction even when a lot of detail objects are attached to a garment's surface.

In this chapter I will discuss the single methods of my work. The discussion section is split into geometry processing and rendering to reflect and follow the single stages of my approach.

6.1 Geometry Processing

The discussion of the results for geometry processing is organized along the issues *macro objects*, *appliqués* and *tessellation*. The aspect *macro objects* is being discussed regarding its use for handling accessory items of the avatar. In the *appliqués* section I discuss the utility of the deferred warping process in relation to the use for the garments surface details. In the *tessellation* part I discuss the results of my GPU based surface tessellation approach, which is used for increasing the resolution of low resolution detail objects.

6.1.1 Macro Objects - Cuboid-Based Deformation

I have presented an approach to embed deformations in a scene graph system by replacing the linear transformation system by a more generic one. This extends the usage of transformation nodes to warping. This has been archived by combining the idea of trilinear transformation with the hierarchical organization structure of a scene graph. I support arbitrary deformations by using an approximation scheme. As a large benefit I perform all geometric transformations on the GPU within constant time. The composition of the scene graph's transformations is still performed inside the CPU. Theoretically, this could lead to a performance bottleneck, but only if a huge multitude of complex transformations have to be composed and send to the GPU. However, I did not observe this within any of my experiments. As an application I presented the attachment of arbitrary geometry to the surface of other deformable or dynamic geometry. For handling surface normals I have presented two methods. The first one performs a direct deformation of the normals. The second method guarantees orthogonality to the deformed surface by using the tangent space for normal representation.

According to [GCDV98] the inverse of a trilinear transformation can be computed analogously to the bilinear transformation's inverse. However the computation is complex, since there can be ambiguous areas for both, which make it difficult for the use with intersection tests. I did not cover this issue, since I do not need the inversion for my rasterization-based rendering concept. Within other rendering approaches the inverse is often a critical feature, for example for finding ray / objects intersections. Additionally it allows mapping one deformation onto another. Trough an inversion, the integration of structure based animation concepts (like skeleton based character animation) could be mapped into the scene graph structure. The non trivial inversion problem and C_0 continuity are current drawbacks of the used trilinear deformation. So a more complex, but invertible and more continuous transformation scheme could be of higher benefit in these cases.

6.1.2 Decoration - Deferred Warping

The cuboid's potential bottleneck of transferring thousands of transformations to the GPU and the continuity issues are problematic. Therefore, I introduced deferred warping, a novel approach for real-time deformation of 3D objects attached to an animated or manipulated surface. The method consists of two steps, creating the matrix texture which contains the transformations and applying those to the detail geometry. One of its key advantages compared to previous work is the fact that detailed geometry is directly transformed, allowing additional manipulation in a post-processing step as well as efficient rendering using a standard rasterization-based pipeline. I demonstrated an implementation in the vertex shading stage of modern GPUs. Another key advantage is the flexibility to use different types of attachment which are typically not supported by existing techniques.

Deferred warping requires an U/V parametrization to work. Within garment design a good parametrization is always present. Alternatively an arbitrary parametrization can be used, however inconsistencies within the parametrization will reflect onto the placing and deformation of the detail object (see Figure 6.1).

My approach opens several directions for future work. First, I combine deferred warping with procedurally generated surface geometry. Current techniques such as Li et al. [LBZ*11] or Yuksel et al. [YKJM12] generate geometry directly related to surfaces, but are not fast enough for interactive use. Applying deferred warping after the generation process could be a way to integrate procedurally generated surface geometry in real-time animations. Another topic I want to address in future is the missing collision feedback when using deferred warping in a simulation framework. Using the attached geometry directly for collision detection is too expensive for interactive simulations. To solve this problem one could introduce a collision geometry which approximates the final surface and transform this geometry simply by deferred warping. Finally, I believe that my technique could greatly increase the visual

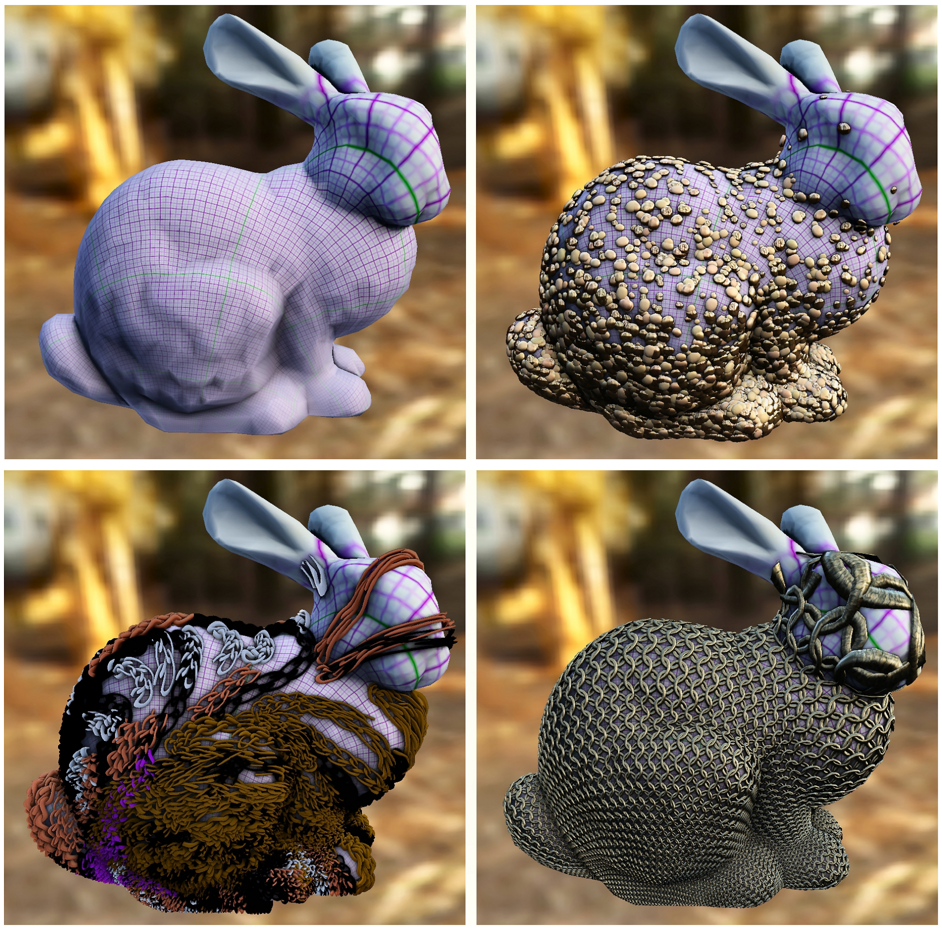


Figure 6.1: The presented warping process can be used on arbitrary U/V parametrization. However, it has to be taken care of distortions within the parametrization in order to get a consistent mapping.

complexity in various application domains including garments visualized in virtual try-on scenarios and computer games.

The reason for this is the geometry-based character of deferred warping, which acts as geometry transformation. It can be easily applied to other rendering systems aside from real-time rasterization.

In the scenario of my work this method enables the user to add fine details to an evolving surface. Therefore, it allows to discern between low detail and high detail geometry. This is directly used within the rendering stage by applying different self shadowing techniques for the detail types.

6.1.3 Smoothing - Tessellation

In order to deal with low resolution detail geometry, I have presented a GPU-based approach for adaptive mesh refinement. My approach makes use of the GPU's geometry shader to perform the necessary geometry processing in real time within an existing rendering pipeline. In contrast to existing work, I use a quad and a triangle-shaped refinement strategy to combine on the fly topology computation with LOD transition inside the patch. It creates reasonable tessellations on thin triangles by utilizing the quad mesh independency of vertical and horizontal resolution. The non-dyadic tessellation scheme grants a finer control of the generated triangles as opposed to a dyadic scheme. This finer resolution step size results in additional smaller transitions of level changes on patch borders. To grant an additional smoothing of this border transition I have introduced a warping mechanism which manipulates the generated barycentric coordinates of the patch. This allows control over the density of generated triangles inside the patch. The method presented is intended to be used for silhouette refinement of a garment's geometry. Especially the creasing angle of folds and curvatures reveal rough triangular structures, which have to be smoothed.

The geometry-shader-based approach has the advantage of allowing full control over the complete refinement process. The size of generated refinement meshes is limited this way, which is not a problematic aspect in my use case. The garment mesh already has a good resolution and my goal is to smooth the simulator's output mesh. GPU based hardware tessellation, on the other hand, allows a smooth transition of refinement levels. However, its tessellation method seems to introduce two new vertices per refinement step and has one resolution for the complete inner area. My method has a finer granularity in this case, but is limited to fixed LOD transitions. Future work can address this issue. Currently my approach uses a simple split algorithm which could be extended to feature recursive splits. Allowing a much finer control over the generated mesh structure, it would be a nice tradeoff between accuracy and speed. Replacing the uniform triangle mesh generator in my approach with a semi-uniform one would increase the number of possible splits of the split logic. This can result in better refinement meshes and is a topic of future research.

6.2 Rendering

The discussion of the results for rendering is organized along the issues *material and geometry based occlusion* and *screen-space based occlusion*. Material and geometry based occlusion discusses the utility of the results for rendering low frequency details like the cloth surface, including self shadowing and materials. Within the screen-space based occlusion section I discuss the combination of low and high frequency self shadowing as well as the impact of the tessellation based depth value smoothing for the SSAO process.

6.2.1 Image Based Lighting, Material and Geometry Based Occlusion

In order to incorporate a plausible material, lighting and self shadowing method I introduced a novel natural illumination method which allows the real-time rendering of highly glossy surfaces with self-shadowing. Since the evaluation of my tunable reflection model is too slow for real-time rendering, I presented a fast approximation method. This approximation uses information stored in the environment map to compute the illumination under changing glossiness values.

In contrast to previous works, my natural illumination approach for real-time rendering supports high gloss to mirror-reflective surfaces. Furthermore, my approximation scheme for the material system allows a fast computation of the necessary environment maps. Using my filtering technique speeds up the process to a point where video input can be used as a source for the environment map.

Aside from being fast, easy to implement and creating high quality results, my method has also some limitations. Embedding the material system into PEMs allows a fast evaluation but requires a lot of memory capacity since maps for several glossiness values have to be stored. Here the use of texture arrays instead of mipmapping in combination with a parabolic mapping could be investigated to optimize the memory consumption. The use of a linear interpolation to get the lookup values for the PEM yields plausible results, but it is only a first step. Here the goal is to use BRDF approximations instead of the current linear interpolations. This should give much more accurate results and allows the use of measured BRDFs and BTFs.

My proposed method of embedding the material system inside the PEMs results in the same run-time characteristics as standard image based lighting using diffuse materials. The big difference lies within the capability to represent the full range between diffuse and reflective materials.

An application area for my technique is e.g. virtual prototyping of garments. If fabrics are woven or knitted using different yarns with varying reflectance properties, this will result in surfaces with a complex pattern of changing reflectance factors. With my novel technique, it is for the first time possible to handle these changes in one reflection model. Moreover, several extensions to my method are possible. One of the most interesting extensions would be to use the method to approximate measured BRDFs and BTFs. Basically my method allows a fast evaluation of cosine based radiation lobes on a given environment map. Thus, it can be used to speed up the process of evaluating the single cosine terms of an approximation of measured functions. Here, one of the interesting points of my method is, that it is not limited to cosine based lobes.

6.2.2 Screen Space Techniques - Hybrid Occlusion

In order to deal with high frequency details separately from the low frequency ones, I have presented an hybrid self occlusion approach. A benefit of the presented implementation is the possibility to use standard lighting models inside the global part or the presented one. A major problem of combining a directional occlusion and an ambient occlusion technique is over occlusion. Since both methods do not communicate with each other, light can be occluded several times. A future local approach should take into account the directional character of the incoming light. It would allow combining the directional occlusion information of both parts correctly. Additionally the lighting information of the light probe would also be taken into account in the local part. Another topic is the use of global illumination techniques inside the global and local part. The presented technique should be flexible enough to allow such modifications. However, an interesting aspect will be the combination of the local and the global part. With increasing complexity of the used algorithms it will be more and more necessary to exchange information between local and global part to get a seamless merge of the local and the global domain.

To reduce edge accentuation from SSAO methods on garment surfaces, I have presented a smoothing method based on the setup information of pn triangle tessellation. It allows to compute smooth depth values for the SSAO stage. Prior to that it was not possible to use SSAO together with garment simulation since it strongly accents edges. This is a wanted feature for details, but not for a smooth surface.

6.3 Next Steps

Aside from the completed scientific work presented here, there are open questions regarding how these methods are to be used inside and in conjunction with a virtual prototyping system.

In order to have a high utility to the garment designer, such a system needs to have similarity between textile materials visualized in the interactive virtual prototyping system and the later offline rendering processes used for product visualization. At this point Bidirectional Texturing Functions (BTF) [SSK03] and simplified models come into play. In order to reach a higher interplay with these measured materials my current work is focused on extending the presented illumination system towards BTF support directly using the methods presented in section 3.2.1.

Another problem for real-time rendering in this domain is the support of the special properties of cloth with transparencies. Transparency itself is difficult to handle alone and a lot of techniques exist in order to create transparency effects. Additional phenomena like the partly shadowing of the single transparent layers are usually ignored. Research was performed in order to bring together transparency and self-shadowing. The results where presented in [BKBK13]. While the presented method combines order independent transparency and ambient occlusion techniques, it is still open how to combine this with the hybrid occlusion technique presented in this work.

The last technical issue is the current combination of a CPU based simulation with a GPU based visualization system. While my presented approach decreases the bottleneck problem between CPU and GPU by adding the details on the GPU side, the simulation on the CPU side can be a bottleneck. Since cloth simulation consists basically of a large number of elements which communicate with each other, it is to be expected to get a mayor speedup when performing the cloth simulation on the GPU side, too. This is analyzed in [SKBK13], based on the idea to limit the simulation model mainly to the communication problem itself. The results show promising speedups due to a novel hybrid multi-grid simulation approach with low resolution CPU computations and high resolution GPU simulation. The two computation units target different complexities, which adds to the performance of the approach. Again this still needs to be combined with my work presented in this document.

6.4 Other Applications Within Garment Design

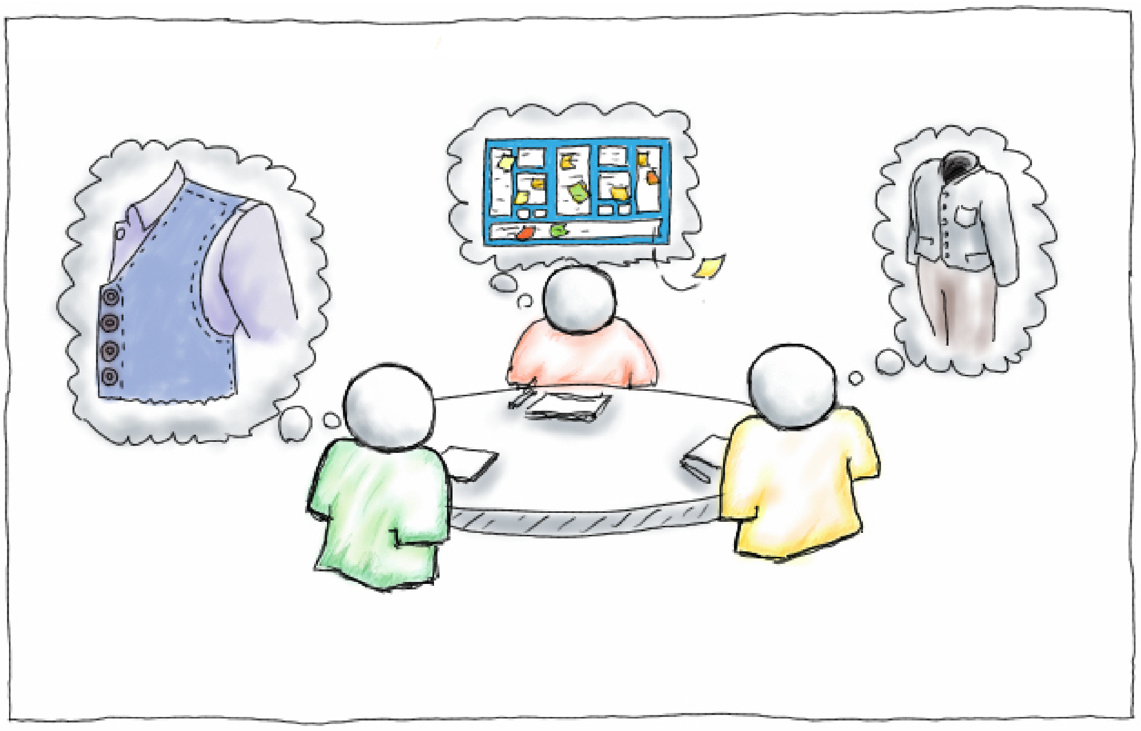


Figure 6.2: The designers can work as a team, observed by a supervisor, who guides the creativity towards specified directions, for example certain product lines of a company.

The methods presented in this work are not limited to be used together with garment simulation. They can be used stand-alone as shown in the following application example.

What needs to be done in order to let a designer (artist) do art, without endangering the producibility of the product as mentioned in the introduction? Or to be described in a more technical manner: how can I transform the things the designer does into parameter structures necessary for production and give him a direct visual feedback of what he does?

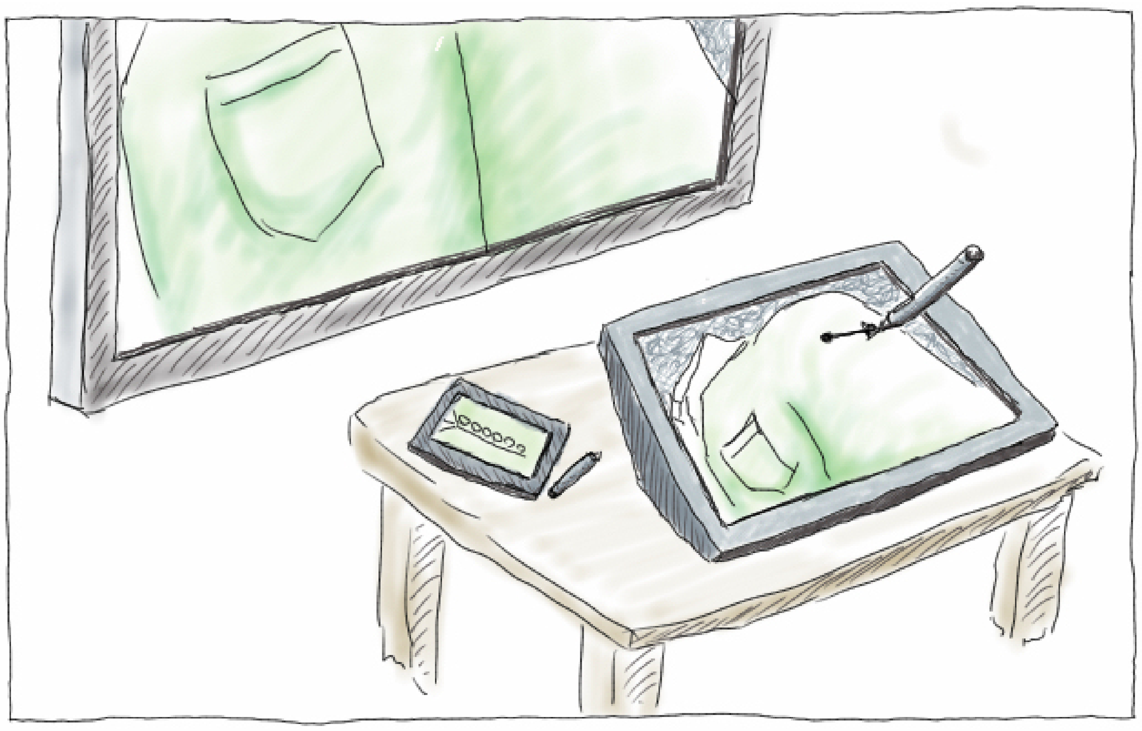


Figure 6.3: The kind of operation a designer performs can be depicted as a classical model-view-control pattern. The model is in this case the arrangement of appliqués on a 2D sewing patterns. The view is the 3D presentation of the appliqués on the virtual garment and the control is the interaction and operations the designer performs, changing the initial arrangements, closing the loop.

One method to ensure this is to let all data he generates stay in the 2D world of the sewing patterns. For appliqués it is now possible to build a design tool, which can be used to support design teams in their initial creativity process. This process can be performed by small design teams optionally supported by a supervisor, which guides the design processes into certain directions (see Figure 6.2). Assuming new designs are often influenced on existing designs, it is possible to modify appliqués on the existing virtual prototype's surface using digitizers on tablet computers.

Here, my presented techniques can be used to ensure the generated data to stay in the 2D world of the sewing pattern: When using a digitizer on a virtual prototype on a tablet computer it is trivial to get the corresponding 2D coordinates within sewing pattern space, since the sewing pattern space is basically a linear transformed version of the texture space on a pattern in 3D. Therefore, appliqués can be directly specified in the 2D pattern space using a digitizer. For visualization feedback my presented techniques can now directly re-project the appliqués into the cloth surface of the virtual prototype. This closes the feedback loop necessary to allow interactive editing (see Figure 6.3). Since no simulation is involved, current tablet computers are powerful enough to allow this kind of operation in real time.

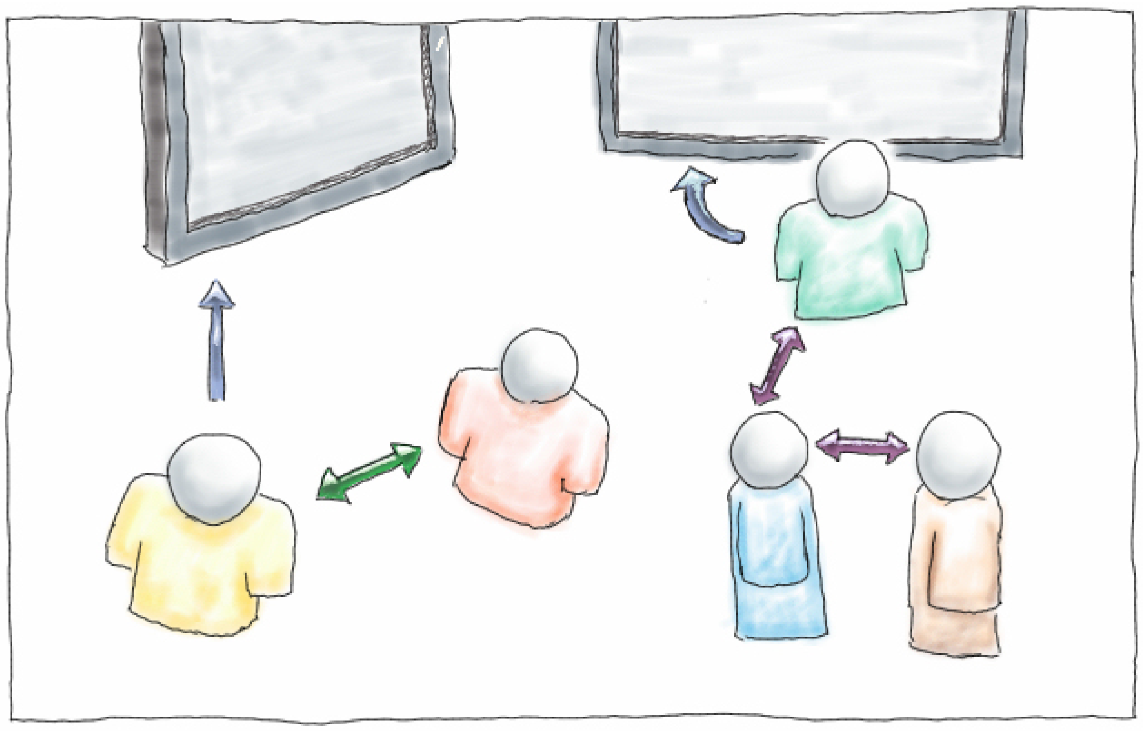


Figure 6.4: In a creative team anyone can communicate with everyone. This needs to be reflected within the creativity tool process.

The data footprint of the necessary operation is small. It mainly consists of information regarding what appliqué has to be placed where in a 2D world. Since tablet computers usually work in a network environment and the generated data has a small footprint it is possible to quickly exchange designs between single members of the design team or present them to the whole team (see Figure 6.4).

This would create a versatile tool for the creative design process. However, its details and its real usefulness is another topic and open for future research.

In the overall summary, my presented method can handle changes of the scene within geometry level, photo-realistic rendering and surface detailing in a 2D CAD friendly way at the same time. However, parts of this work can be used as stand-alone, too, for example to provide stand-alone editing capabilities.

A Publications and Talks

The thesis is partially based on the following publications and talks:

A.1 Publications

1. Arnulph Fuhrmann, Clemens Groß, Martin Knuth and Jörn Kohlhammer
   Virtual Prototyping of Garments.
   ProSTEP iViP Science Days 2005, September 2005.
2. Martin Knuth and Arnulph Fuhrmann
   Self-Shadowing of dynamic scenes with environment maps using the GPU.
   WSCG FULL papers proceedings, pages 31–38, February 2005.
3. Martin Knuth and Jörn Kohlhammer
   A hybrid ambient occlusion technique for dynamic scenes.
   WSCG 2009. Communication Papers Proceedings, pages 1–8, February 2009.
4. Martin Knuth and Jörn Kohlhammer
   Embedding Hierarchical Deformation within a Real-time Scene Graph.
   VISIGRAPP 2010, International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications. Proceedings, pages 246–253, May 2010.
5. Martin Knuth and Jörn Kohlhammer
   A geometry-shader-based adaptive mesh refinement scheme using semiuniform quad/triangle patches and warping.
   VRIPHYS 2010, 7th Workshop on Virtual Reality Interaction and Physical Simulation, pages 21–29, November 2010.
6. Fabian Bauer, Martin Knuth, Jan Bender and Arjan Kuijper
   Screen-space ambient occlusion using A-buffer techniques.
   13th International Conference on Computer-Aided Design and Computer Graphics, pages 140–147, November 2013.
7. Nikolas Schmitt, Martin Knuth ,Jan Bender and Arjan Kuijper
   Multilevel cloth simulation using GPU surface sampling.
   VRIPHYS 2013, 10th Workshop on Virtual Reality Interaction and Physical Simulation, pages 1–10, November 2013.

8. Martin Knuth, Christian Altenhofen, Jan Bender and Arjan Kuijper
   Efficient Self-Shadowing Using Image-Based Lighting on Glossy Surfaces.
   VMV 2014, pages 159–166, October 2014.
9. Martin Knuth, Jan Bender, Michael Goesele and Arjan Kuijper
   Deferred Warping.
   IEEE Computer Graphics and Applications, March 2016.

B Supervising Activities

The following list summarizes the student bachelor, diploma and master theses supervised by the author. The results of these works were partially used as an input into the thesis.

B.1 Diploma and Master Thesis

1. Michael Benz, Cluster based Rendering with Distributed Frame Buffers, TU-Darmstadt, FG Graphisch interaktive Systeme (GRIS), FB Informatik, 2005
2. Carsten Schoger, Globale Beleuchtung interaktiver Szenen, TU-Darmstadt, FG Graphisch interaktive Systeme (GRIS), FB Informatik, 2007
3. Jens Tinz, GPU-basierte Bekleidungssimulation, TU-Darmstadt, FG Graphisch interaktive Systeme (GRIS), FB Informatik, 2007
4. Christian Altenhofen, TU-Darmstadt, FG Graphisch interaktive Systeme (GRIS), FB Informatik, 2010
5. Fabian Bauer, Realistic Realtime Rendeirng of Garment with Transparency and Ambient Occlusion, TU-Darmstadt, FG Graphisch interaktive Systeme (GRIS), FB Informatik, 2013
6. Nikolas Schmitt, Multilevel cloth simulation through GPU surface sampling, TU-Darmstadt, FG Graphisch interaktive Systeme (GRIS), FB Informatik, 2013

B.2 Bachelor Thesis

1. Benedikt Führer, Approximation von Shadow-Maps mit Hilfe von Parallax-Mapping TU-Darmstadt, FG Graphisch interaktive Systeme (GRIS), FB Informatik, 2010
2. Julian Puhl, Materialsysteme für das realistische Echtzeit-Rendering von Szenen in Anwesenheit von Flüssigkeitssimulationen und Image-Based Lighting TU-Darmstadt, FG Graphisch interaktive Systeme (GRIS), FB Informatik, 2014

C Curriculum Vitae

Personal Data

Name	Martin Knuth
Birth date & place	14.03.1975, Hanau
Nationality	Germany

Education

2004	Graduation as Dipl. -Inform. in computer science with minor physics at Technische Universität Darmstadt, Germany

Work Experience

2005 – 2015 Researcher at Fraunhofer Institute for Computer Graphics, IVA group, Darmstadt, Germany, Focus: Virtual prototyping of garments.

Here, I did research in the context of several research projects for the garment industry. The ones with the most impact to this work are listed in the following:

- Virtual Try-On was a project funded by the German ministry of education and science (Bundesministerium fuer Bildung und Forschung - BMBF). This project which made the simulation technology available to allow garment simulation with interaction.
- In the project "SiMaKon" (Simulation von Maßkonfektion) one work strand was to extract appliqués from the simulation pipeline and shift them into the rendering process.
- In "ViVoTex" (Entwurf und Realisierung von Verfahren zur virtuellen Produktentwicklung von volumetrischen Objekten umschlossen von textilem Material) I developed new methods for

visualizing complex surface materials and their illumination for the use in real-time garment visualization. The project funded by the German ministry of economy and technology (Bundesministerium fuer Wirtschaft und Technologie - BMWi).

- During "FFD" (Future Fashion Design) I developed improvements to the appliqué extraction and combination of lighting and shadows together with complex materials.

Bibliography

[ADM*08] ANNEN T., DONG Z., MERTENS T., BEKAERT P., SEIDEL H.-P., KAUTZ J.: Real-time, all-frequency shadows in dynamic scenes. *ACM Trans. Graph. 27*, 3 (2008). 24, 26, 29, 34, 51, 55

[AMB*07] ANNEN T., MERTENS T., BEKAERT P., SEIDEL H.-P., KAUTZ J.: Convolution shadow maps. In *Proc. EGSR* (2007), Eurographics Association, pp. 51–60. 34

[AMHH08] AKENINE-MÖLLER T., HAINES E., HOFFMAN N.: *Real-Time Rendering 3rd Edition*. A. K. Peters, Ltd., Natick, MA, USA, 2008. 24, 27, 39, 52, 66, 69, 76

[ARBJ03] AGARWAL S., RAMAMOORTHI R., BELONGIE S., JENSEN H. W.: Structured importance sampling of environment maps. *ACM Trans. Graph. 22*, 3 (2003), 605–612. 24

[BA08] BOUBEKEUR T., ALEXA M.: Phong tessellation. In *SIGGRAPH Asia '08: ACM SIGGRAPH Asia 2008 papers* (2008), pp. 1–5. 22

[Bar84] BARR A. H.: Global and local deformations of solid primitives. *SIGGRAPH Comput. Graph. 18*, 3 (1984), 21–30. 14

[BD13] BENDER J., DEUL C.: Adaptive cloth simulation using corotational finite elements. *Computers & Graphics 37*, 7 (2013), 820 – 829. 87

[BKBK13] BAUER F., KNUTH M., BENDER J., KUIJPER A.: Screen-space ambient occlusion using A-buffer techniques. In *2013 International Conference on Computer-Aided Design and Computer Graphics (CAD/Graphics)* (2013), IEEE, pp. 140–147. 26, 28, 112

[Bly06] BLYTHE D.: The direct3d 10 system. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Papers* (New York, NY, USA, 2006), ACM, pp. 724–734. 20, 44

[BMPB08] BRODERSEN A., MUSETH K., PORUMBESCU S., BUDGE B.: Geometric texturing using level sets. *IEEE TVCG 14*, 2 (2008), 277–288. 16, 18, 19

[BN76] BLINN J. F., NEWELL M. E.: Texture and reflection in computer generated images. *Commun. ACM 19*, 10 (1976). 24, 26, 27

[BPWG07] BOTSCH M., PAULY M., WICKE M., GROSS M.: Adaptive space deformations based on rigid cells. *Computer Graphics Forum (Proc. EUROGRAPHICS) 26*, 3 (2007), 339–347. 14, 18, 19

[BRS05] BOUBEKEUR T., REUTER P., SCHLICK C.: Scalar tagged pn triangles. In *EUROGRAPHICS 2005 (Short Papers)* (2005), Eurographics. 22

[BS05] BOUBEKEUR T., SCHLICK C.: Generic mesh refinement on gpu. In *HWWS '05: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware* (2005), ACM, pp. 99–104. 22

[BS08] BOUBEKEUR T., SCHLICK C.: A flexible kernel for adaptive mesh refinement on gpu. *Computer Graphics Forum 27*, 1 (2008), 102–114. 22

[BSD08] BAVOIL L., SAINZ M., DIMITROV R.: Image-space horizon-based ambient occlusion. In *ACM SIGGRAPH talks* (2008), ACM, p. 22. 25

[CCI*05] CHEN M., CORREA C., ISLAM S., JONES M. W., Y. SHEN P., SILVER D., WALTON S. J., WILLIS P. J.: Deforming and animating discretely sampled object representations. *Eurographics State of the Art Reports* (2005). 14

[CHL04] COOMBE G., HARRIS M. J., LASTRA A.: Radiosity on graphics hardware. In *GI '04: Proceedings of the 2004 conference on Graphics interface* (2004), Canadian Human-Computer Communications Society, pp. 161–168. 25

[Cro77] CROW F.: Shadow algorithms for computer graphics. *j-COMPGRAPHICS 11*, 2 (July 1977), 242–248. 24

[CYKK09] CHUNG K., YU C.-H., KIM D., KIM L.-S.: Technical section: Shader-based tessellation to save memory bandwidth in a mobile multimedia processor. *Comput. Graph. 33*, 5 (2009), 625–637. 23

[Deb98] DEBEVEC P.: Rendering synthetic objects into real scenes: bridging traditional and image-based graphics with global illumination and high dynamic range photography. In *Proc. SIGGRAPH* (1998), pp. 189–198. 24, 26, 27, 50, 77

[Deb08] DEBEVEC P.: A median cut algorithm for light probe sampling. In *ACM SIGGRAPH Classes* (2008), ACM. 24, 26, 27, 28, 55, 75

[DGR*09] DONG Z., GROSCH T., RITSCHEL T., KAUTZ J., SEIDEL H.-P.: Real-time indirect illumination with clustered visibility. In *VMV* (2009), pp. 187–196. 26, 29

[DL06] DONNELLY W., LAURITZEN A.: Variance shadow maps. In *I3D '06: Proceedings of the 2006 symposium on Interactive 3D graphics and games* (New York, NY, USA, 2006), ACM, pp. 161–165. 24, 34, 76

[DR08] DECORO C., RUSINKIEWICZ S.: Subtractive shadows: A flexible framework for shadow level of detail. *J. Graphics Tools 13*, 1 (2008), 45–56. 26, 27, 28

[DRS08] DYKEN C., REIMERS M., SELAND J.: Real-time gpu silhouette refinement using adaptively blended bézier patches. *Comput. Graph. Forum 27*, 1 (2008), 1–12. 23

[DRS09] DYKEN C., REINERS M., SELAND J.: Semi-uniform adaptive patch tessellation. *Computer Graphics Forum 28(8)*, 8 (2009), 2255–2263. 23, 45, 46, 47, 91

[EKS03] ETZMUSS O., KECKEISEN M., STRASSER W.: A fast finite element solution for cloth modelling. In *Proc. Pacific Graphics* (oct. 2003), pp. 244 – 251. 87

[Elb02] ELBER G.: Geometric deformation-displacement maps. In *Proceedings of the 10th Pacific Conference on Computer Graphics and Applications* (Washington, DC, USA, 2002), PG '02, IEEE Computer Society, pp. 156–. 16, 18

[EP09] EIGENSATZ M., PAULY M.: Positional, metric, and curvature control for constraint-based surface deformation. *Computer Graphics Forum (Proc. EUROGRAPHICS) 28(2)*, 2 (2009), 551–558. 14, 18, 19

[FH04] FILIP J., HAINDL M.: Non-linear reflectance model for bidirectional texture function synthesis. In *Proc. Pattern Recognition (ICPR)* (2004), vol. 1, pp. 80–83 Vol.1. 52

[FMHF08] FÜNFZIG C., MÜLLER K., HANSFORD D., FARIN G.: Png1 triangles for tangent plane continuous surfaces on the gpu. In *GI '08: Proceedings of graphics interface 2008* (2008), pp. 219–226. 22

[Fuh06] FUHRMANN A.: Interaktive animation textiler materialien. In *Dissertation: TU Darmstadt, Fachbereich Informatik* (2006). 4, 8

[GCDV98] GOMES J., COSTA B., DARSA L., VELHO L.: *Warping and morphing of graphical objects*. Morgan Kaufman Publishers, San Francisco, CA, 1998. 14, 36, 108

[Gre86] GREENE N.: Environment mapping and other applications of world projections. *IEEE Comput. Graph. Appl. 6*, 11 (1986), 21–29. 24

[GW05] GUO Z., WONG K. C.: Skinning with deformable chunks. *Computer Graphics Forum (Proc. EUROGRAPHICS) 24*, 3 (Sept. 2005), 373–382. 15, 18, 19

[HL10] HOANG T.-D., LOW K.-L.: Multi-resolution screen-space ambient occlusion. In *Proc. VRST* (2010), pp. 101–102. 25, 28, 87

[HLHS03] HASENFRATZ J.-M., LAPIERRE M., HOLZSCHUCH N., SILLION F.: A survey of real-time soft shadows algorithms. In *Eurographics* (2003), Eurographics, Eurographics. State-of-the-Art Report. 24

[Hop97] HOPPE H.: View-dependent refinement of progressive meshes. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1997), ACM Press/Addison-Wesley Publishing Co., pp. 189–198. 73

[HS99] HEIDRICH W., SEIDEL H.-P.: Realistic, hardware-accelerated shading and lighting. In *Proc. SIGGRAPH* (1999), pp. 171–178. 24, 26, 27

[HSK*05] HAVRAN V., SMYK M., KRAWCZYK G., MYSZKOWSKI K., SEIDEL H.-P.: Importance sampling for video environment maps. In *ACM SIGGRAPH Sketches* (2005), ACM. 24, 26, 27, 28, 51

[JC07] JENSEN H. W., CHRISTENSEN P.: High quality rendering using ray tracing and photon mapping. In *ACM SIGGRAPH 2007 Courses* (New York, NY, USA, 2007), SIGGRAPH '07, ACM. 26, 28

[JFH04] J. FAN W. Y., HUNTER L.: *Clothing appearance and fit: Science and technology*. CRC Press, 2004. 4

[JMW07] JESCHKE S., MANTLER S., WIMMER M.: Interactive smooth and curved shell mapping. In *Proc. EGSR* (2007), pp. 351–360. 16, 18, 19

[KAKB14] KNUTH M., ALTENHOFEN C., KUIJPER A., BENDER J.: Efficient self-shadowing using image-based lighting on glossy surfaces. In *VMV 2014, Vision, Modeling, and Visualization* (2014), pp. 159–166. 12, 49

[KBGK15] KNUTH M., BENDER J., GOESELE M., KUIJPER A.: Deferred warping. In *To be published* (2015). 11

[KCYM14] KONIARIS C., COSKER D., YANG X., MITCHELL K.: Survey of texture mapping techniques for representing and rendering volumetric mesostructure. *Journal of Computer Graphics Techniques* (2014). 19

[Kel97] KELLER A.: Instant radiosity. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques* (1997), ACM Press/Addison-Wesley Publishing Co., pp. 49–56. 24

[KF05] KNUTH M., FUHRMANN A.: Self-shadowing of dynamic scenes with environment maps using the gpu. In *WSCG'2005 Full Papers Conference Proceedings* (2005), pp. 31–37. 12, 75

[KK03] KOLLIG T., KELLER A.: Efficient illumination by high dynamic range images. In *EGRW '03: Proceedings of the 14th Eurographics workshop on Rendering* (2003), Eurographics Association, pp. 45–50. 24

[KK09] KNUTH M., KOHLHAMMER J.: A hybrid ambient occlusion technique for dynamic scenes. In *WSCG'2009 Communication Papers Proceedings* (2009), pp. 1–8. 12, 49

[KKK10a] KNUTH M., KOHLHAMMER J., KUIJPER A.: Embedding hierachical deformation within a realtime scene graph. In *VISIGRAPP 2010, Proceedings* (2010), pp. 246–253. 11, 18, 35

[KKK10b] KNUTH M., KOHLHAMMER J., KUIJPER A.: A geometry-shader-based adaptive mesh refinement scheme using semiuniform quad/triangle patches and warping. In *VRIPHYS 2010, 7th Workshop on Virtual Reality Interaction and Physical Simulation* (2010), pp. 21–29. 11, 43

[KLA04] KAUTZ J., LEHTINEN J., AILA T.: Hemispherical rasterization for self-shadowing of dynamic objects. In *Proceedings Eurographics Symposium on Rendering 2004* (2004). 25, 27

[KM00] KAUTZ J., MCCOOL M. D.: Approximation of glossy reflection with pre-filtered environment maps. In *Graphics Interface* (2000), pp. 119–126. 26, 27

[KRES11] KLEHM O., RITSCHEL T., EISEMANN E., SEIDEL H.-P.: Bent normals and cones in screen-space. In *VMV* (2011), Eurographics Association, pp. 177–182. 26

[KS14] KOLIVAND H., SUNAR M.: Anti-aliasing in image based shadow generation techniques: a comprehensive survey. *Multimedia Tools and Applications*

(2014), 1–27. 34

[KVHS00] KAUTZ J., VÁZQUEZ P.-P., HEIDRICH W., SEIDEL H.-P.: Unified approach to prefiltered environment maps. In *Proc. Eurographics Workshop on Rendering Techniques* (2000), Springer-Verlag, pp. 185–196. 26, 27

[LBZ*11] LI Y., BAO F., ZHANG E., KOBAYASHI Y., WONKA P.: Geometry synthesis on surfaces using field-guided shape grammars. *IEEE TVCG 17*, 2 (2011), 231–243. 108

[LCNV09] LENZ R., CAVALCANTE-NETO J. B., VIDAL C. A.: Optimized pattern-based adaptive mesh refinement using gpu. In *SIBGRAPI '09: Proceedings of the 2009 XXII Brazilian Symposium on Computer Graphics and Image Processing* (2009), IEEE Computer Society, pp. 88–95. 22, 44, 49

[LD08] LORENZ H., DÖLLER J.: Dynamic mesh refinement on gpu using geometry shaders. In *WSCG'2008 Full Papers Conference Proceedings* (2008), pp. 97–104. 22

[LFTG97] LAFORTUNE E. P. F., FOO S.-C., TORRANCE K. E., GREENBERG D. P.: Non-linear approximation of reflectance functions. In *Proc. SIGGRAPH* (1997), ACM, pp. 117–126. 52

[LHLW09] LIU F., HUANG M.-C., LIU X.-H., WU E.-H.: Efficient depth peeling via bucket sort. In *Proceedings of the Conference on High Performance Graphics 2009* (2009), pp. 51–57. 16

[LLCO08] LIPMAN Y., LEVIN D., COHEN-OR D.: Green coordinates. *ACM Trans. Graph. 27*, 3 (2008), 78:1–78:10. 18, 20

[Lov06] LOVISCACH J.: Wrinkling coarse meshes on the gpu. *Computer Graphics Forum (Proc. EUROGRAPHICS) 25*, 3 (2006), 467–476. 17, 18, 19

[LS08] LANGER T., SEIDEL H.-P.: Higher order barycentric coordinates. *Computer Graphics Forum (Proc. EUROGRAPHICS) 27(2)*, 2 (2008), 459–466. 14, 18, 20

[LSNC09] LOOP C., SCHAEFER S., NI T., CASTA NO I.: Approximating subdivision surfaces with gregory patches for hardware tessellation. In *SIGGRAPH Asia '09: ACM SIGGRAPH Asia 2009 papers* (New York, NY, USA, 2009), ACM, pp. 1–9. 21

[MC10] MÜLLER M., CHENTANEZ N.: Wrinkle meshes. In *Proc. SCA* (2010), pp. 85–92. 18, 19

[McG10] MCGUIRE M.: Ambient occlusion volumes. In *Proc. High Performance Graphics* (2010), Eurographics Association, pp. 47–56. 26

[MHTG05] MÜLLER M., HEIDELBERGER B., TESCHNER M., GROSS M.: Meshless deformations based on shape matching. *ACM Trans. Graph. 24*, 3 (2005), 471–478. 88

[Mit07] MITTRING M.: Finding next gen: Cryengine 2. In *ACM SIGGRAPH courses* (2007), ACM, pp. 97–121. 25, 26, 28

[MKC12] MÜLLER M., KIM T.-Y., CHENTANEZ N.: Fast Simulation of Inextensible Hair and Fur. In *Proc. VRIPHYS* (2012), pp. 39–44. 43

[Ney98] NEYRET F.: Modeling, animating, and rendering complex scenes using volumetric textures. *IEEE Transactions on Visualization and Computer Graphics 4*, 1 (Jan. 1998), 55–70. 16, 18

[NMK*05] NEALEN A., MUELLER M., KEISER R., BOXERMAN E., CARLSON M.: Physically based deformable models in computer graphics. *Eurographics State of the Art Reports* (2005). 14, 18, 19

[NRH03] NG R., RAMAMOORTHI R., HANRAHAN P.: All-frequency shadows using non-linear wavelet lighting approximation. In *ACM SIGGRAPH 2003 Papers* (2003), pp. 376–381. 25

[OBM00] OLIVEIRA M. M., BISHOP G., MCALLISTER D.: Relief texture mapping. In *Proc. SIGGRAPH* (2000), pp. 359–368. 15, 18

[ODJ04] OSTROMOUKHOV V., DONOHUE C., JODOIN P.-M.: Fast hierarchical importance sampling with blue noise properties. *ACM Transactions on Graphics 23*, 3 (2004), 488–495. Proc. SIGGRAPH 2004. 24

[PBFJ05] PORUMBESCU S. D., BUDGE B., FENG L., JOY K. I.: Shell maps. In *Proc. SIGGRAPH* (2005), pp. 626–633. 16, 18, 19

[PBMH02] PURCELL T. J., BUCK I., MARK W. R., HANRAHAN P.: Ray tracing on programmable graphics hardware. *ACM Trans. Graph. 21*, 3 (2002), 703–712. 25, 26, 28

[PCCS11] PIETRONI N., CORSINI M., CIGNONI P., SCOPIGNO R.: An interactive local flattening operator to support digital investigations on artwork surfaces. *IEEE Transaction on Visualization and Computer Graphics (Proceedings of Visualization 2011) 17*, 12 (December 2011), 1989–1996. 18, 19

[PO06] POLICARPO F., OLIVEIRA M. M.: Relief mapping of non-height-field surface details. In *Proc. I3D* (2006), pp. 55–62. 16, 18

[POC05] POLICARPO F., OLIVEIRA M. M., COMBA J. A. L. D.: Real-time relief mapping on arbitrary polygonal surfaces. In *Proc. I3D* (2005), pp. 155–162. 16, 17, 18

[PZB*09] POPA T., ZHOU Q., BRADLEY D., KRAEVOY V., FU H., SHEFFER A., HEIDRICH W.: Wrinkling captured garments using space-time data-driven deformation. *Computer Graphics Forum (Proc. EUROGRAPHICS) 28(2)*, 2 (2009), 427–435. 17

[RDGK12] RITSCHEL T., DACHSBACHER C., GROSCH T., KAUTZ J.: The state of the art in interactive global illumination. *Comput. Graph. Forum 31*, 1 (Feb. 2012), 160–188. 23, 24, 50

[RE99] RAVIV A., ELBER G.: Three dimensional freeform sculpting via zero sets of scalar trivariate functions. In *SMA '99: Proceedings of the fifth ACM symposium on Solid modeling and applications* (New York, NY, USA, 1999), ACM, pp. 246–257. 14, 18, 35

[RGK*08] RITSCHEL T., GROSCH T., KIM M. H., SEIDEL H.-P., DACHSBACHER C., KAUTZ J.: Imperfect shadow maps for efficient computation of indirect illumination. In *ACM SIGGRAPH Asia 2008 Papers* (2008), pp. 129:1–129:8. 33

[RGS09] RITSCHEL T., GROSCH T., SEIDEL H.-P.: Approximating Dynamic Global Illumination in Screen Space. In *Proc. Interactive 3D Graphics and Games* (2009). 25, 26

[RGSS09] RUBINSTEIN D., GEORGIEV I., SCHUG B., SLUSALLEK P.: Rtsg: Ray tracing for x3d via a flexible rendering framework. In *Proc. of the 14th International Conference on 3D Web Technology 2009* (2009), ACM, New York, NY, USA. 14, 18, 19

[RH94] ROHLF J., HELMAN J.: Iris performer: a high performance multiprocessing toolkit for real-time 3d graphics. In *SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1994), ACM, pp. 381–394. 14, 18, 19

[RH01] RAMAMOORTHI R., HANRAHAN P.: An efficient representation for irradiance environment maps. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (2001), ACM Press, pp. 497–500. 24

[RMM10] R.CONCHEIRO, M.AMOR, M.BÓO.: Synthesis of bezier surfaces on the gpu. In *Proceedings of the International Conference on Computer Graphics Theory and Applications (GRAPP 2010)* (2010). 21

[RSC87] REEVES W. T., SALESIN D. H., COOK R. L.: Rendering antialiased shadows with depth maps. *SIGGRAPH Comput. Graph. 21*, 4 (1987), 283–291. 24

[RSSSG01] REZK-SALAMA C., SCHEUERING M., SOZA G., GREINER G.: Fast volumetric deformation on general purpose hardware. In *HWWS '01: Proc. of the ACM SIGGRAPH/EUROGRAPHICS workshop on Graphics hardware* (2001), ACM, pp. 17–24. 38, 66

[RVB02] REINERS D., VOSS G., BEHR J.: Opensg: Basic concepts. In *In 1. OpenSG Symposium OpenSG* (2002). 14, 18, 19

[SA07] SHANMUGAM P., ARIKAN O.: Hardware accelerated ambient occlusion techniques on GPUs. In *Proc. Interactive 3D graphics and games* (2007), ACM, pp. 73–80. 25, 78

[SGP03] SHIUE L.-J., GOEL V., PETERS J.: Mesh mutation in programmable graphics hardware. In *HWWS '03: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware* (2003), pp. 15–24. 22

[SGW06] SCHMIDT R., GRIMM C., WYVILL B.: Interactive decal compositing with discrete exponential maps. *ACM Trans. Graph. 25*, 3 (July 2006), 605–613. 15, 17, 18

[SGW09] SCHNEIDER J., GEORGII J., WESTERMANN R.: Interactive geometry decals. In *Proceedings of Vision, Modeling, and Visualization 2008* (2009). 15, 17, 18

[SHHS03] SLOAN P.-P., HALL J., HART J., SNYDER J.: Clustered principal components for precomputed radiance transfer. In *ACM SIGGRAPH 2003 Papers* (2003), pp. 382–391. 25

[SKBK13] SCHMITT N., KNUTH M., BENDER J., KUIJPER A.: Multilevel cloth simulation using gpu surface sampling. In *VRIPHYS 2013, 10th Workshop in Virtual Reality Interactions and Physical Simulations* (2013), pp. 1–10. 7, 113

[SKE05] SCHEIN S., KARPEN E., ELBER G.: Real-time geometric deformation displacement maps using programmable hardware. *The Visual Computer 21*, 8-10 (2005), 791–800. 16, 18, 29, 33

[SKS02] SLOAN P.-P., KAUTZ J., SNYDER J.: Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. In *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques* (2002), pp. 527–536. 25

[SM05] SANDER P. V., MITCHELL J. L.: Progressive buffers: view-dependent geometry and texture lod rendering. In *SGP '05: Proceedings of the third Eurographics symposium on Geometry processing* (Aire-la-Ville, Switzerland, Switzerland, 2005), Eurographics Association, p. 129. 72

[SP86] SEDERBERG T. W., PARRY S. R.: Free-form deformation of solid geometric models. *SIGGRAPH Comput. Graph. 20*, 4 (1986), 151–160. 14, 18, 35

[SRD98] SOWIZRAL H., RUSHFORTH K., DEERING M.: *The Java 3D API Specification*. Addison-Wesley, 1998. 14, 18, 19

[SSK03] SATTLER M., SARLETTE R., KLEIN R.: Efficient and realistic visualization of cloth. In *Proceedings of the 14th Eurographics Workshop on Rendering* (2003), pp. 167–177. 112

[SSP07] SUMNER R. W., SCHMID J., PAULY M.: Embedded deformation for shape manipulation. *ACM Trans. Graph. 26*, 3 (2007), 80. 14, 18, 19

[SSSK04] SZECSI L., SBERT M., SZIRMAY-KALOS L.: Combined correlated and importance sampling in direct light source computation and environment mapping. *Computer Graphics Forum (Eurographics 04) 23*, 3 (2004). 24

[SSZK04] SATTLER M., SARLETTE R., ZACHMANN G., KLEIN R.: Hardware-accelerated ambient occlusion computation. In *Vision, Modeling, and Visualization 2004* (November 2004), Girod B., Magnor M., Seidel H.-P., (Eds.), Akademische Verlagsgesellschaft Aka GmbH, Berlin, pp. 331–338. 25

[SZ98] S. ZHUKOV A. IONES G.: An ambient light illumination model. In *Proc. Rendering Techniques* (1998). 24, 25

[Tat05] TATARCHUK N.: Practical dynamic parallax occlusion mapping. In *Proc. SIGGRAPH Sketches* (2005). 16, 17, 18

[THGM11] THIEDEMANN S., HENRICH N., GROSCH T., MÜLLER S.: Voxel-based global illumination. In *Proc. Interactive 3D Graphics and Games* (2011), ACM, pp. 103–110. 26

[TWL07] TOLEDO R. D., WANG B., LEVY B.: Geometry textures. In *Proc. SIBGRAPI* (2007), pp. 79–86. 18, 19

[VPBM01] VLACHOS A., PETERS J., BOYD C., MITCHELL J. L.: Curved pn triangles. In *I3D '01: Proceedings of the 2001 symposium on Interactive 3D graphics* (2001), pp. 159–166. 22, 72, 91

[VPG13] VARDIS K., PAPAIOANNOU G., GAITATZES A.: Multi-view ambient occlusion with importance sampling. In *Proc. Interactive 3D Graphics and Games* (2013), ACM, pp. 111–118. 26

[WBS03] WALD I., BENTHIN C., SLUSALLEK P.: Interactive global illumination in complex and highly occluded environments. In *EGRW '03: Proceedings of the 14th Eurographics workshop on Rendering* (2003), Eurographics Association, pp. 74–81. 25

[wFD15] WWW.FUTURE FASHION-DESIGN.EU: Future fashion design real-time, accurate fabric to garment virtual prototyping in collaborative environments, On 07.07.2015. 4

[WHR97] WANG D., HERMAN I., REYNOLDS G. J.: The open inventor toolkit and the premo standard, 1997. 14, 18, 19

[Wil78] WILLIAMS L.: Casting curved shadows on curved surfaces. In *SIGGRAPH '78: Proceedings of the 5th annual conference on Computer graphics and interactive techniques* (1978), ACM Press, pp. 270–274. 24

[WPS*03] WALD I., PURCELL T. J., SCHMITTLER J., BENTHIN C., SLUSALLEK P.: Realtime ray tracing and its use for interactive global illumination. In *Eurographics State of the Art Reports* (2003). 25, 26, 28

[ws16] WWW.HUMAN SOLUTIONS.COM: Assyst gmbh, part of the human solutions group, On 10.09.2016. 61

[WTL*04] WANG X., TONG X., LIN S., HU S., GUO B., SHUM H.-Y.: Generalized displacement maps. In *Proceedings of the Fifteenth Eurographics conference on Rendering Techniques* (Aire-la-Ville, Switzerland, Switzerland, 2004), EGSR'04, Eurographics Association, pp. 227–233. 16, 18, 19

[www10] WWW.OPENGL.ORG: Opengl 4.0 specification, On 08.04.2010. 20, 72

[YKJM12] YUKSEL C., KALDOR J. M., JAMES D. L., MARSCHNER S.: Stitch meshes for modeling knitted clothing with yarn-level detail. *ACM Trans. Graph. 31*, 4 (2012), 37:1–37:12. 18, 19, 108