Optimal Transport for Interactive Interpolation Problems in Computer Graphics

Dissertation

zur Erlangung des Doktorgrades (Dr. rer. nat.)
der
Mathematisch-Naturwissenschaftlichen Fakultät
der
Rheinischen Friedrich-Wilhelms-Universität Bonn

vorgelegt von

Dipl.-Inform. Tim Golla
aus
Dachau

Bonn, Juli 2021

Angefertigt mit Genehmigung der Mathematisch-Naturwissenschaftlichen
Fakultät der Rheinischen Friedrich-Wilhelms-Universität Bonn

1. Gutachter:	Prof. Dr. Reinhard Klein
2. Gutachter:	Prof. Dr. Thomas Schultz
Tag der Promotion:	14.1.2022
Erscheinungsjahr:	2022

Acknowledgements

I would like to thank Prof. Dr. Reinhard Klein for supervising this thesis and always having new ideas. I would like to thank my family, my friends and my colleagues for supporting me.

Abstract

Modern computer graphics require realistic surface materials and geometries for generating convincing imagery. Both types of data are inherently complex for most applications. To obtain them, measurement of real-world samples is a widespread method. A frequently required editing operation in computer graphics is the creation of intermediate states of measured assets. In the case of complex assets such as highly-detailed surface materials or complicated geometries this as a difficult task.

For the purpose of interpolating this type of data, the theory of optimal transport is applied in this thesis. In general, this theory deals with the optimal allocation of resources. Mathematically speaking, it aims to warp one probability measure onto another at minimum cost. The common theme and overarching question of this thesis is how how measured input data can be processed and represented in such a way that an optimal transport problem can be set up and how its solution can provide a meaningful result. This allows achieving results of a visual quality that was impossible to obtain before.

Overall the proposed approach consists of the following key steps. The first step is to represent the input data as probability measures. Then, a cost measure that suits the needs of the problem at hand has to be chosen. The next step is to set up the optimal transport problem in a computationally manageable way and to solve it. The final step comprises interpreting the problem's solution in a way that is useful for our purpose, for instance, to produce meaningful, visually appealing results.

The proposed approach is applied here in two areas of computer graphics for which satisfactory solutions have not yet been found. This thesis first focuses on representing the sparkling effect of metallic car paints as measured bidirectional texture functions (BTFs). It explains how to represent these in a novel, memory-efficient statistical way that is also suitable for real-time rendering. Having obtained this statistical representation, it is described how to set up and solve an optimal transport problem between two such representations. The interpolation of the metallic paint BTFs is then performed using the solution of the optimal transport problem in real-time, allowing for an interactive application.

Following this, the temporal upsampling of time series of 3D point clouds obtained from 3D scans of plants is addressed. The first step of the developed solution is to represent the point clouds as hierarchies of point cloud clusters that correspond to the natural segments of the plant. Then, a matching of the hierarchically organized clusters using the optimal transport-based Wasserstein distance is to be found. A solution to the optimal transport problem between all matched pairs of clusters is computed. The information gained through the optimal transport solution is then used to generate an arbitrary number of intermediate states between two consecutive scans of a time series, which allows us to generate a temporal upsampling of the whole sequence, resulting in smooth animations for temporally sparse 3D scans.

In summary, the methods proposed in this thesis solve the problem of generating interpolations of measured data that were previously hard or impossible to generate. The methods provide realistic interpolations, using optimal transport theory as a basis, underlining its usefulness in computer graphics applications. The findings can be used for artistic applications in the field of rendering as well as for scientific applications in the field of plant growth visualization.

Zusammenfassung

Moderne Computergrafik erfordert realistische Oberflächenmaterialien und -geometrien, um überzeugende Bilder zu erzeugen. Beide Arten von Daten sind für die meisten Anwendungen von Natur aus komplex. Um sie zu erhalten, ist die Messung von realen Proben eine sehr weit verbreitete Methode. Eine häufig erforderliche Bearbeitungsoperation in der Computergrafik ist die Erstellung von Zwischenzuständen gemessener Daten. Bei komplexen Daten wie hochdetaillierten Oberflächenmaterialien oder komplizierten Geometrien ist dies eine schwierige Aufgabe.

Zum Zwecke der Interpolation dieser Art von Daten wird in dieser Arbeit die Theorie des optimalen Transports angewendet. Im Allgemeinen befasst sich diese Theorie mit der optimalen Allokation von Ressourcen. Mathematisch gesehen zielt sie darauf ab, ein Wahrscheinlichkeitsmaß zu minimalen Kosten auf ein anderes abzubilden. Das grundlegende Thema und die übergreifende Frage dieser Arbeit ist, wie gemessene Eingabedaten so aufbereitet und dargestellt werden können, dass ein optimales Transportproblem aufgestellt werden kann und wie seine Lösung ein nutzbares Ergebnis liefert. Auf diese Weise können Ergebnisse von einer visuellen Qualität erzielt werden, die vorher unmöglich zu erreichen war.

Insgesamt besteht der vorgeschlagene Ansatz aus den folgenden Schlüsselschritten. Der erste Schritt besteht darin, die Eingabedaten als Wahrscheinlichkeitsmaße darzustellen. Dann muss ein Kostenmaß gewählt werden, das den Erfordernissen des vorliegenden Problems entspricht. Der nächste Schritt besteht darin, das optimale Transportproblem in einer rechnerisch

handhabbaren Weise aufzustellen und zu lösen. Der letzte Schritt umfasst die Interpretation der Lösung des Problems in einer Weise, die für unsere Zwecke nützlich ist, zum Beispiel um aussagekräftige, visuell ansprechende Ergebnisse zu erzielen.

Der vorgeschlagene Ansatz wird hier in zwei Bereichen der Computergrafik angewandt, für die es bisher keine befriedigenden Lösungen gibt. Diese Arbeit konzentriert sich zunächst auf die Darstellung des Glitzereffekts von Metallic-Autolacken als gemessene bidirektionale Texturfunktionen (BTFs). Es wird erklärt, wie diese auf eine neuartige, speichereffiziente statistische Weise dargestellt werden können, die auch für Echtzeit-Rendering geeignet ist.

Nachdem diese statistische Darstellung vorliegt, wird beschrieben, wie ein optimales Transportproblem zwischen zwei solchen Darstellungen aufgestellt und gelöst werden kann. Die Interpolation der BTFs für Metallic-Lacke wird dann unter Verwendung der Lösung des optimalen Transportproblems in Echtzeit durchgeführt, wodurch eine interaktive Anwendung ermöglicht wird.

Im Anschluss daran wird das zeitliche Upsampling von Zeitreihen von 3D-Punktwolken, die durch 3D-Scans von Pflanzen behandelt. Der erste Schritt der hier entwickelten Lösung besteht darin, die Punktwolken als Hierarchien von Punktwolken-Clustern darzustellen, die den natürlichen Segmenten der Pflanze entsprechen. Danach wird ein Matching der hierarchisch organisierten Cluster unter Verwendung der Wasserstein-Distanz gefunden. Es wird eine Lösung für das optimale Transportproblem zwischen allen Paaren von Clustern berechnet. Die durch die Lösung des Transportproblems gewonnene Information wird dann dazu verwendet, eine beliebige Anzahl von Zwischenzuständen zwischen zwei aufeinanderfolgenden Scans einer Zeitreihe zu erzeugen, was uns erlaubt, ein zeitliches Upsampling der gesamten Sequenz zu erzeugen, wodurch sich flüssige Animationen für zeitlich gering aufgelöste Reihen von 3D-Scans ergeben.

Zusammenfassend lässt sich sagen, dass die in dieser Arbeit vorgestellten Methoden das Problem der Erzeugung von Interpolationen von Messdaten lösen, welche vorher nur schwer oder gar nicht zu erzeugen waren. Die

Methoden liefern realistische Interpolationen auf der Grundlage der Theorie des optimalen Transports, was deren Nützlichkeit für Computergrafikanwendungen unterstreicht. Die Ergebnisse können sowohl für künstlerische Anwendungen im Bereich des Renderings als auch für wissenschaftliche Anwendungen im Bereich der Visualisierung von Pflanzenwachstum verwendet werden.

Contents

1 Introduction

1.1 Motivation and Contributions

In modern computer graphics, realistic surface materials and geometries are required for generating convincing imagery. Both of these types of data are, however, complex by nature for most applications. For obtaining these data, there are two approaches: (1) manually creating them or (2) measuring real-world samples. The manual creation has the advantage of providing exactly what the artist had in mind and it allows for great flexibility in the modification of the assets due to the nature of their generation with e.g. analytical parameters. Creating assets with realistic detail, however, is a resource-intensive task, demanding expert knowledge and experience. Measuring real-world samples, on the other hand, requires – thanks to the availability of commercial scanning devices – less expertise and can reproduce very fine detail. It is the equivalent of comparing painting to taking a photograph with a modern camera. By measuring real-world samples using 3D scanners, even biological processes such as the growth of plants can be captured with a realistic detail that is hard to achieve with manual or computational methods. Despite these advantages, measured data provides less flexibility in terms of customization. Data modifications are more difficult. For instance, modifications of unstructured point clouds resulting from a 3D scan are much harder than modifications of a 3D mesh generated with 3D software. Measured data also represents only snapshots of reality. Only what is physically available at the current point in time can be captured. When e.g. scanning a plant, the data represent its current state of growth. When scanning a metallic car paint, only the existing sample can be used. It will however frequently happen that the existing real-world input data is not perfectly adequate for the application at hand. As an example, the designer might want to change the appearance of the paint of a car after the acquisition process has ended already. A real-world sample of what he has in mind might not be available. Thus, it becomes necessary to modify the data after the acquisition process.

A frequently-used operation in many computer graphics applications is the *interpolation* between discrete states of input data, i.e. generating smooth transitions between them. This operation is also intuitive for the user. Interpolation in general has been used in computer graphics in numerous applications, from fundamental techniques like spline interpolation to more complex methods like view interpolation [CW93], motion figure interpolation [WH97] and mesh interpolation [ACOL00]. One of the first interpolation applications more noticeable to the wider audience was the movie special effects technique of *morphing*, thanks to its ability to generate previously impossible imagery. In early motion pictures, this effect was achieved by blending one image over the other – a simple interpolation technique. A similar effect also exists in physical form: The tabula scalata, a technique known since the end of the 16th century. Here, two images are painted over a corrugated surface. Depending on the view direction, either the one or the other is visible. This effect is still used today for example in children's toys. The first digital morphing effects were used in the 1980s, e.g. in the movies "Flight of the Navigator" (1986) and "The Golden Child" (1986). These were already a big step forward from the previous blending technique. In the 1990s, a more realistic computer-aided technique became popular, where the user would have to manually select corresponding points in two images. Then, e.g. by using the algorithm of Beier and Neely [BN92], it is possible to warp one image's shape and interpolate its colors towards that of a second image. Well-known examples showcasing such more advanced morphing techniques are the Michael Jackson music video "Black or White" (1991) and the movies "Star Trek VI: The Undiscovered Country" (1991) and "Terminator 2: Judgment Day" (1991).

In recent years, the theoretical framework of optimal transport has gained popularity in the field of computer graphics, e.g. as shown in [SDGP+15].

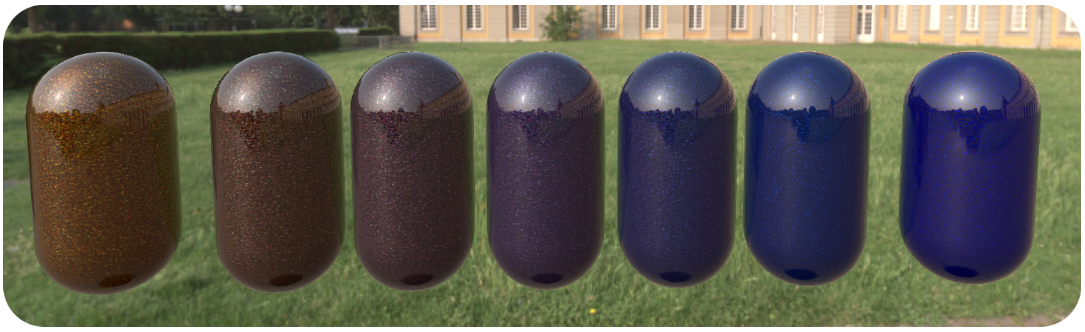


Figure 1.1: *Rendering and interpolation of metallic paints. On the left and on the right are two original metallic effect car paints. Note that the material on the left exhibits an appearance with a strong sparkling effect, while the blue material on the right only exhibits a weak sparkling. In between are interpolated paints, generated with the method described in this thesis.*

This thesis investigates the use of optimal transport with the aim of generating interpolated states of complex measured data. The focus is on cases for which the generation of interpolated states is particularly difficult. Intuitively, optimal transport theory deals with matching supply from given sources (like factories) to given targets (e.g. warehouses). For each pair of source and target, a transport cost, usually the distance, has to be defined. The optimal transport problem is to find the allocation of goods from sources to targets at minimal cost. In this thesis, possibilities to prepare measured data used for computer graphics in a way such that the interpolation of two datasets can be posed as an optimal transport problem are explored. The solution to this problem is used for creating intermediate steps and animations in a way that makes sense visually for a human observer.

This thesis first focuses on metallic car paints. These paints consist of several layers and are visually complex. They exhibit a sparkling effect which is represented here in a data-driven way as measured bidirectional texture functions (BTFs). These BTFs are memory-intense. A novel, memory-effcient representation for the metallic paint BTFs, which is also suitable for real-time rendering, is introduced in this thesis. The key to this is to find clusters of uniform distribution in color space. The distributions, as well as their boundaries, are stored in textures on the GPU. A specialized shader is able to render the metallic effect from this representation. Having obtained this statistical representation, it is described how to set up and solve an optimal transport problem between two such BTFs. The interpolation of the metallic paint BTFs is then performed using the solution of the optimal transport problem by transporting the color clusters accordingly through color space. The interpolation is performed in real-time, allowing for an interactive application. A rendering of two measured metallic car paints and interpolations between them is shown in Fig. 1.1.

Following this, the temporal upsampling of time series of 3D point clouds obtained by 3D scans of plants by interpolation is addressed. The key is to represent the point clouds as hierarchies of point cloud clusters that correspond to the natural segments of the plant. It is described how to find a matching of the hierarchically organized clusters using the optimal transport-based Wasserstein distance. For each pair of clusters, an optimal transport problem is set up. Due to the size of the point clouds, setting up the optimal transport problem in the standard way, using a full distance matrix is often impossible. It is described how to set up the problem in a sparse way, regularize it and then solve it efficiently. The information gained through the optimal transport solution allows to identify associations between the points of two subsequent scans of a time series and subsequentially to generate an arbitrary amount of intermediate states between them in real-time. This allows for generating a temporal upsampling of the whole sequence, yielding smooth animations for temporally sparse 3D scans. An exemplary upsampling of a komatsuna plant scan is shown in Fig. 1.2. An upsampling of a synthetically generated scan of a tree is shown in Fig. 1.3.

In both areas, the methods described deliver high quality interpolation results at comparatively low computational demands.

1.2 Contributions

In summary, the contributions of this thesis are:

- A novel, statistical representation for the data-driven BTF used for metallic car paint, which is

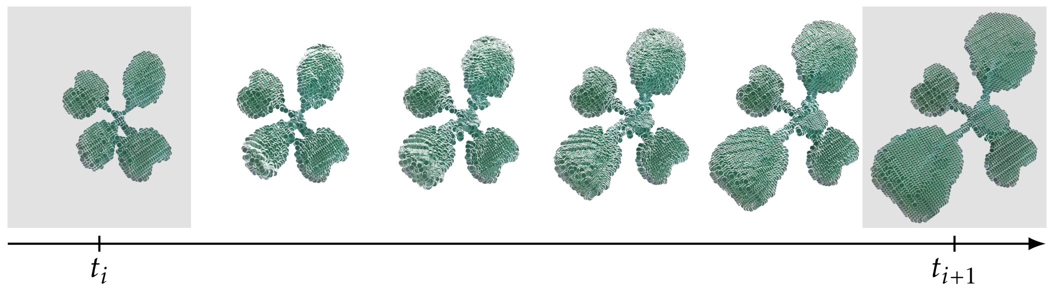


Figure 1.2: *Temporal upsampling of scans of a komatsuna plant. The original point clouds for the time steps t_i and t_{i+1} are highlighted with a gray background. The in-between states have been computed with the method described in this thesis.*

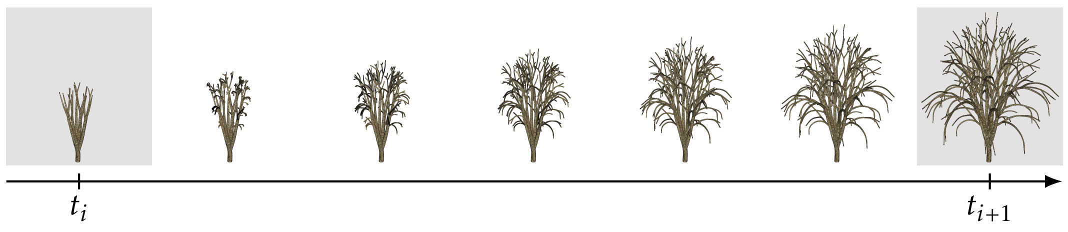


Figure 1.3: *Temporal upsampling of two synthetically generated tree point clouds obtained with the method described in this thesis. The original point clouds for the time steps t_i and t_{i+1} are highlighted with a gray background. The in-between states have been computed with the upsampling method described in this thesis.*

    - memory-efficient.
    - fast to compute.
    - suitable for decompression and real-time rendering on the GPU.
    - allows for the generation of arbitrarily-sized BTFs, despite the limited size of the original input data.
- A novel method for interpolating metallic car paints using the previously introduced representation and the theory of optimal transport, which
    - exhibits a high visual quality.
    - is suitable for real-time applications.
    - allows to separately interpolate the different visual aspects of the metallic paints, that is the larger-scale reflective properties, including the basic color hue, the local color hue and the sparkling intensity of the metallic paints.
    - allows to generate interpolations of three or more metallic car paints.
- A novel method for temporal upsampling of sequences of 3D scans of plants, based on optimal transport theory, which
    - doesn't require exact point correspondences, shape priors or huge databases.
    - identifies and handles topological changes like growing or decaying leaves and generates corresponding virtual segments and thus interpolations.
    - includes a heuristic segmentation method suitable for thin structures such as occurring in plants like maize or trees.
    - yields smooth temporal upsamplings, allowing for realistic growth animations with moderate computational effort.

1.3 Mathematical Foundations of Optimal Transport

In the following, we outline the mathematical foundations of optimal transport, which facilitates a better understanding of the rest of this thesis. Optimal

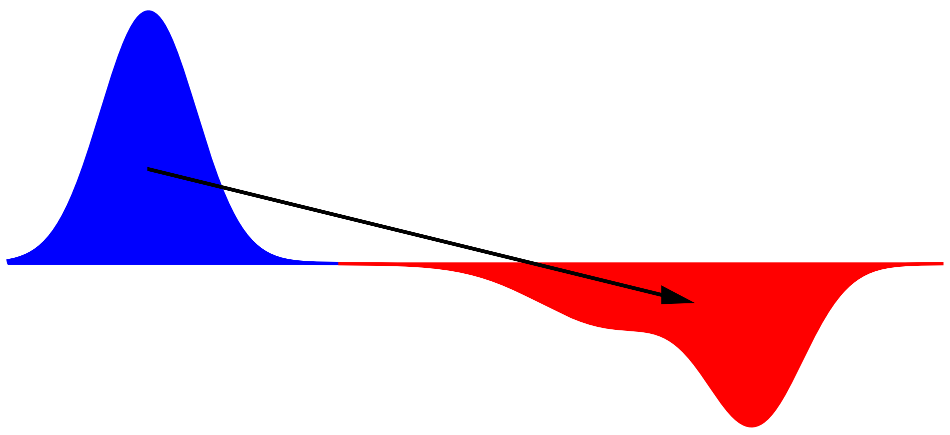


Figure 1.4: *Illustration of the intuition behind the Monge Problem. The goal is to transport the pile of sand on the left (blue) into the hole on the right (red) at minimum cost.*

Transport is a *linear optimization* or *linear programming* problem. The French mathematician Gaspard Monge (1746–1818) was the first to formalize the problem of optimal transport [Mon81]. He described the problem as follows: An existing pile of sand has to be moved to a target hole that has a specific shape. The total effort spent on this task has to be minimized. This is also called *Monge problem* and is illustrated in Fig. 1.4. Interpreted in a mathematical way, the problem can be stated as follows: With two given probability distributions we want to find the optimal way of morphing or transporting them onto each other.

Mathematically speaking, let $\mathcal{M}$ be a Riemannian manifold, μ_0 (the "pile of sand") and μ_1 (the "hole") two probability measures on $\mathcal{M}$ and c a distance measure on $\mathcal{M}$. Our task is to find a map τ that minimizes:

$$W(\mu_0, \mu_1) = \int_{\mathcal{M}} c(x, \tau(x)) d\mu_0(x) \tag{1.1}$$

with

$$\mu_0(\tau^{-1}(\Omega)) = \mu_1(\Omega) \forall \Omega \in \mathcal{M}. \tag{1.2}$$

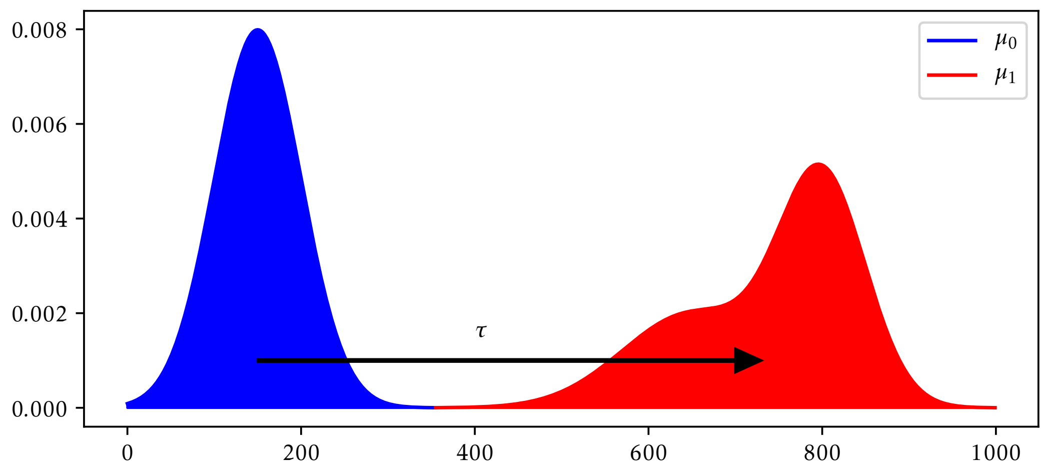


Figure 1.5: *Illustration of the basic optimal transport problem. Our goal is to find the map τ that transports the probability measure μ_0 to μ_1 at minimal cost.*

μ_1 is sometimes also called the pushforward of μ_0 by τ, denoted as $\mu_1 = \tau \# \mu_0$. $W(\mu_0, \mu_1)$ defines the total transport cost of μ_0 to μ_1. In the case $c(x, \tau(x)) = d(x, \tau(x))^p, p > 1$ and d is the geodesic distance on $\mathcal{M}$, $W(\mu_0, \mu_1)^{1/p}$ is also called the pth *Wasserstein distance*. An illustration of the problem is given in Fig. 1.5.

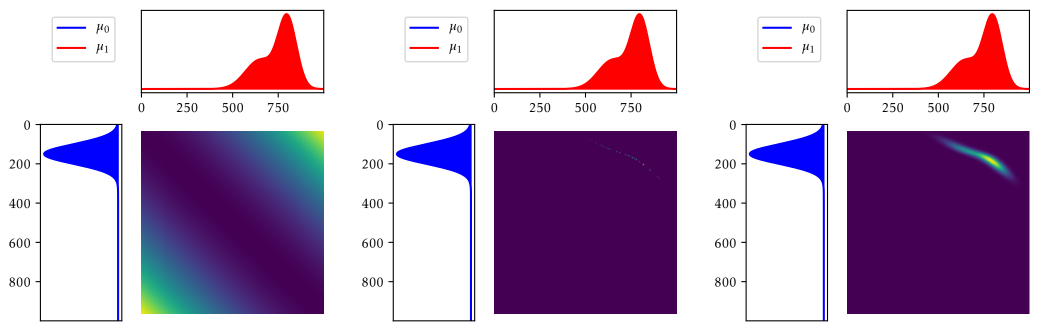
In the 1930s and 40s, Leonid Vitaliyevich Kantorovich (1912-1986) was working in the field of linear optimization [Kan39]. The problem he was trying to solve was how to move soldiers to destinations in the most efficient way. In 1942 [Kan42], he described the following problem, also called *Monge-Kantorovich problem*, and a solution to it: Find a probability measure π that minimizes:

$$W(\mu_0, \mu_1) = \inf_{\pi \in \Pi(\mu_0, \mu_1)} \int_{\mathcal{M}} \int_{\mathcal{M}} c(x, y) d\pi(x, y), \tag{1.3}$$

with the constraints

$$\pi(U \times \mathcal{M}) = \mu_0(U) \forall U \subseteq \mathcal{M}, \tag{1.4}$$

$$\pi(\mathcal{M} \times V) = \mu_1(V) \forall V \subseteq \mathcal{M}, \tag{1.5}$$



(a) *Distances* (b) *Exact Solution* (c) *Approximate Solution*

Figure 1.6: *Illustration of the setup of a 1D optimal transport problem. The probability measure μ_0 (blue) is to be transported to the probability measure μ_1 (red). The measures and the distances between two positions is illustrated in Fig. 1.6a. Brighter colors mean larger values. An exact solution for the probability measure π in the optimal transport problem is visualized in Fig. 1.6b. Note that the transport is so selective that it is barely visible in the picture. An approximate solution to the optimal transport problem is illustrated in Fig. 1.6c. This solution is slightly "blurred", which also makes it easier to see in the visualization.*

where $\Pi(\mu_0, \mu_1)$ is the set of probability measures fulfilling these constraints, i.e.

$$\begin{aligned}\Pi(\mu_0,\mu_1) = \{\pi \in Pr(\mathcal{M}\times\mathcal{M})\,|\pi(U\times\mathcal{M}) = \mu_0(U)\forall U \subseteq \mathcal{M},\\ \pi(\mathcal{M}\times V) = \mu_1(V)\forall V \subseteq \mathcal{M}\},\end{aligned} \tag{1.6}$$

with $Pr(\mathcal{M}\times\mathcal{M})$ being the set of probability measures on $\mathcal{M}\times\mathcal{M}$. An illustration of the problem setup and solutions is given in Fig. 1.6. Analogous to before, we can define the pth Wasserstein distance as:

$$W_p(\mu_0,\mu_1) := \left(\inf_{\pi\in\Pi(\mu_0,\mu_1)} \int_{\mathcal{M}}\int_{\mathcal{M}} d(x,y)^p d\pi(x,y)\right)^{1/p}, \tag{1.7}$$

with $d(\cdot,\cdot)$ being the geodesic distance on $\mathcal{M}$.

Around the same time as Kantorovich, Tjalling Charles Koopmans came to similar conclusions. In 1975, they were awarded the Nobel Prize in economics.

The Monge-Kantorovich problem can be discretized as follows. The probability measures μ_0 and μ_1 are discretized as histograms $\bar{\mu}_0 = (h_{0a})_a \in [0,1]^n, a \in \{1,\dots,n\}$ with n bins resp. $\bar{\mu}_1 = (h_{1b})_b \in [0,1]^m, b \in \{1,\dots,m\}$ with m bins. A discretized version of Eq. (1.3) can then be written as: Find a matrix $\boldsymbol{T} = (T_{ab})_{ab} \in \mathbb{R}^{n\times m}$ such that

$$W(\bar{\mu}_0,\bar{\mu}_1) = \min_{\boldsymbol{T}} \sum_{a=1}^{n}\sum_{b=1}^{m} T_{ab}\cdot C_{ab}, \quad C_{ab} = c(x_a, y_b) \tag{1.8}$$

$$\text{s.t. } T_{ab} \geq 0 \qquad \forall a \in \{1,\dots,n\}, \forall b \in \{1,\dots,m\} \tag{1.9}$$

$$\sum_{a=1}^{n} T_{ab} = h_{1b} \qquad \forall b \in \{1,\dots,m\} \tag{1.10}$$

$$\sum_{b=1}^{m} T_{ab} = h_{0a} \qquad \forall a \in \{1,\dots,n\} \tag{1.11}$$

This problem formulation is a linear program with *mn* variables, *mn* inequality constraints and $m+n$ equality constraints. An illustration of the problem setup is given in Fig. 1.7 and Fig. 1.8.

Towards the end of the 1990s, optimal transport became popular in the field of computer vision, most notably under the name *Earth Mover's Distance*. Very influential work regarding this was published by Rubner and his collaborators [RTG98, RTG00, RT01]. Also known as *Wasserstein Distance*, the Earth Mover's Distance describes the cost of the optimal transport and can e.g. be used as a measure for image similarity.

As stated before, the discretized optimal transport problem is a linear program with *nm* unknowns, thus it can be solved with a standard approach like Dantzig's Simplex algorithm. For large problems however, it turned out [Cut13] that it is advantageous to solve a modified problem: The regularized

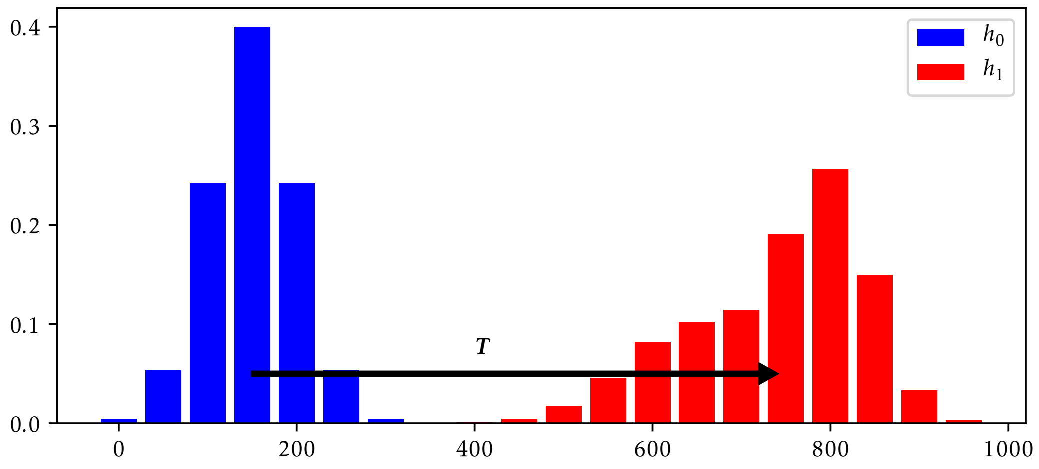
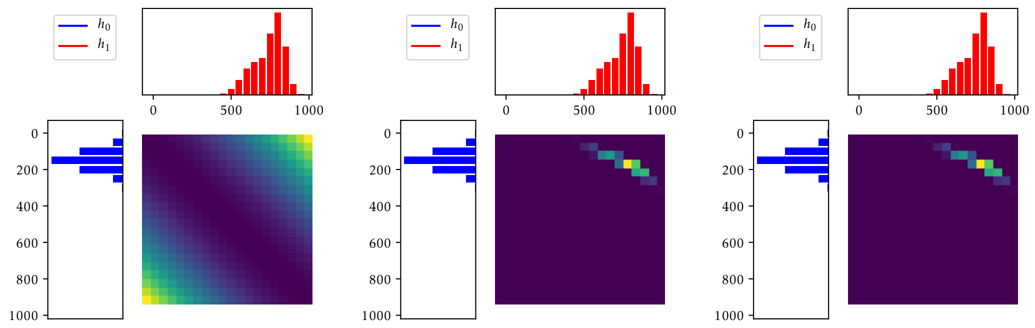


Figure 1.7: *Illustration of the discretized basic optimal transport problem. Our goal is to find the map $\boldsymbol{T}$ that transports the histogram $\boldsymbol{h_0}$ to $\boldsymbol{h_1}$ at minimal cost.*



(a) *Distances* (b) *Exact Solution* (c) *Approximate Solution*

Figure 1.8: *Illustration of the setup of a discrete 1D optimal transport problem. The histogram $\boldsymbol{h_0}$ (blue) is to be transported to the histogram $\boldsymbol{h_1}$ (red). The histograms and the distances between two positions is illustrated in Fig. 1.8a. Brighter colors mean higher cost. An exact solution for the transport matrix $\boldsymbol{T}$ in the optimal transport problem is visualized in Fig. 1.8b. Brighter colors mean larger values. An approximate solution to the optimal transport problem is illustrated in Fig. 1.8c. Note that in this case, the difference between the exact and approximate solution is very small.*

optimal transport problem. In the regularized transport problem, the objective function in Eq. (1.8) is modified to

$$\min_{\boldsymbol{T}} \sum_{a=1}^{n} \sum_{b=1}^{m} T_{ab} \cdot C_{ab} - \gamma H(\boldsymbol{T}), \tag{1.12}$$

where $H(\boldsymbol{T})$ is the entropy of the matrix $\boldsymbol{T}$, defined as

$$H(\boldsymbol{T}) = -\sum_{a=1}^{n} \sum_{b=1}^{m} T_{ab} \log T_{ab}. \tag{1.13}$$

Inserting Eq. (1.13) into Eq. (1.12) yields

$$\min_{\boldsymbol{T}} \sum_{a=1}^{n} \sum_{b=1}^{m} T_{ab} \cdot C_{ab} + \gamma \sum_{a=1}^{n} \sum_{b=1}^{m} T_{ab} \log T_{ab}. \tag{1.14}$$

For solving this problem, a more efficient algorithm exists: The Sinkhorn algorithm. It can be shown [Cut13] that minimizing Eq. (1.14) leads to the following condition for the elements of $\boldsymbol{T}$:

$$T_{ab} = u_a K_{ab} v_b \tag{1.15}$$

where $K_{ab} = \exp(-C_{ab}/\gamma)$ are the entries of a matrix $\boldsymbol{K}$ and $\boldsymbol{u} = (u_a)_a, \boldsymbol{v} = (v_b)_b$ are unknown vectors to be determined. Note that with the vectors $\boldsymbol{u}$ and $\boldsymbol{v}$ we only have a linear number of unknowns, as opposed to the quadratic number in the matrix $\boldsymbol{T}$. If we insert Eq. (1.15) into the problem constraints Eq. (1.10) and Eq. (1.11), we obtain the following relationships:

$$\sum_{a=1}^{n} T_{ab} = h_{1b} \Rightarrow v_b \sum_{a=1}^{n} K_{ab} u_a = h_{1b} \Rightarrow v_b = \frac{h_{1b}}{\sum_{a=1}^{n} K_{ab} u_a} \tag{1.16}$$

$$\sum_{b=1}^{m} T_{ab} = h_{0a} \Rightarrow u_a \sum_{b=1}^{m} K_{ab} v_b = h_{0a} \Rightarrow u_a = \frac{h_{0a}}{\sum_{b=1}^{m} K_{ab} v_b} \tag{1.17}$$

Using these relations iteratively as an algorithm for finding the solution was originally proposed by Sinkhorn [Sin64] in a different context and introduced to optimal transport by Cuturi [Cut13]. Initialize the vectors $\boldsymbol{u} = (u_a)_a = \mathbf{1}, \boldsymbol{v} = (v_b)_b = \mathbf{1} \forall a \in A, b \in B$. The Sinkhorn algorithm then finds an approximate solution by iterating the following steps:

1. $\boldsymbol{u} \leftarrow \boldsymbol{h_0} \oslash \boldsymbol{K}\boldsymbol{u}$
2. $\boldsymbol{v} \leftarrow \boldsymbol{h_1} \oslash \boldsymbol{K}\boldsymbol{v}$,

where $\oslash$ denotes an element-wise division. The elements of the approximate solution matrix $\boldsymbol{T}$ are then $T_{ab} := u_a K_{ab} v_b$. The algorithm usually only takes few iterations to converge.

The questions that arise for most practitioners that want to use optimal transport in order to solve their problems are:

- How to state the problem at hand in a way such that it can be transformed into a standard optimal transport problem. How to represent the data at hand? When these questions are answered, an existing algorithm can be used for solving the problem.
- How to interpret and use the result?

While for some problems, such as the problem of finding the optimal logistics from a number of sources (factories) to a number of targets (warehouses), setting up and interpreting the problem are relatively obvious. For many other applications, this is not clear.

This thesis gives answers to these questions in two fields, where these have not been tackled yet: The rendering of data-based metallic car paint representations (BTFs) and in the field of the temporal upsampling of 3D scans of plants. It is shown how to transform the measured input data into forms that are applicable as an input to an optimal transport problem and how to interpret the results for interpolation in real-time applications.

1.4 Optimal Transport In Computer Graphics

In this section, an overview of the use of optimal transport in the field of computer graphics is given. Well-known are the works of Rubner et al. [RTG98, RTG00], who used optimal transport in the form of the Earth Mover's

Distance for measuring the similarity of images. They computed the transportation cost between color distributions and texture feature distributions for measuring the distances between images. Pele and Werman presented a fast algorithm for thresholded Earth Mover's Distances [PW09] that they used for image retrieval. Solomon et al. [SRGB14] described how to compute the Earth Mover's Distance on discrete surfaces. Mérigot presented a multiscale approach to optimal transport with an application in image interpolation [Mér11].

A widely-used application for optimal transport in computer graphics is color transfer. The following papers are all in this field: Pitié et al. used optimal transport with regularization while maintaining the gradient of the original picture for color transfer [PKD07]. Rabin et al. [RP11] use approximate Wasserstein constraints on color statistics and geometric regularization for color transfer in images. Ferradans et al. used regularized discrete optimal transport within a unified convex variational framework [FPPA14]. Rabin et al. [RFP14] used relaxed optimal transport with a regularization that respects the spatial distribution of colors. Frigo et al. [FSDH14] separately matched illuminant and optimal transported dominant colors, which was regularized by thin plate splines. Chizat et al. extended optimal transport based on entropy regularization to unbalanced problems [CPSV16].

De Goes et al. [DGCSAD11] described a method for 2D shape reconstruction and simplification from defect and noisy point clouds. In the following year, De Goes et al. [DGBOD12] described an algorithm to generate blue noise with the help of optimal transport. Digne et al. [DCSA$^+$14] used optimal transport for feature-preserving surface reconstruction from point clouds and their simplification. Merigot et al. [MMT18] described an algorithm for optimal transport between a point cloud and a simplex soup. Lavenant et al. [LCCS18] dealt with optimal transport on discrete surfaces. Qin et al. [QHL$^+$19] used optimal transport for topology-aware skeleton extraction from point clouds.

The most closely related work in the context of this thesis deals with interpolation in computer graphics. These papers are described in this paragraph. Rabin et al. [RPDB11] used optimal transport for interpolating textures. They used a sliced approximation over 1D distributions to replace the original

Wasserstein metric. This allows them to employ a fast stochastic gradient descent algorithm. This yields a new notion of the barycenter of probabilities, which they use for 2D texture interpolation. Bonneel et al. [BVDPPH11] used optimal transport for bidirectional reflectance distribution function (BRDF) interpolation. They represent distributions or functions as sums of radial basis functions and solve a transport problem between these representations. Solomon et al. [SDGP$^+$15] introduced convolutional Wasserstein distances which exhibited faster convergence in their applications and were applied for interpolating BRDFs and volumetric representations. They interpolated 3D data represented as density functions in a fixed 3D voxel grid and BRDFs.

The works in this thesis are – to the best of the author's knowledge – the first ones to use optimal transport to solve interpolation of measured BTFs and point clouds.

1.5 Publications

This cumulative thesis comprises the following publications:

1. Tim Golla and Reinhard Klein. An Efficient Statistical Data Representation for Real-Time Rendering of Metallic Effect Car Paints. In *Virtual Reality and Augmented Reality: 14th EuroVR International Conference, EuroVR 2017*, pages 51–68. Springer, Cham, 2017
2. Tim Golla and Reinhard Klein. Interactive Interpolation of Metallic Effect Car Paints. In *Vision, Modeling & Visualization, VMV 2018*, EG VMV '18, page 11–20. The Eurographics Association, 2018
3. Tim Golla, Tom Kneiphof, Heiner Kuhlmann, Michael Weinmann, and Reinhard Klein. Temporal Upsampling of Point Cloud Sequences by Optimal Transport for Plant Growth Visualization. *Computer Graphics Forum*, 39(6):167–179, September 2020

The following papers were also published during the PhD period and are in the context of this thesis, but were not included in it in order to put the focus on interpolation of measured data using optimal transport:

1. Tom Kneiphof, Tim Golla, and Reinhard Klein. Real-time Image-based Lighting of Microfacet BRDFs with Varying Iridescence. *Computer Graphics Forum*, 38(4), July 2019
2. Tom Kneiphof, Tim Golla, Michael Weinmann, and Reinhard Klein. A Method for Fitting Measured Car Paints to a Game Engine's Rendering Model. In *Workshop on Material Appearance Modeling*, pages 27–31. The Eurographics Association, 2018
3. Tim Golla and Reinhard Klein. Real-time Point Cloud Compression. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 5087–5092. IEEE, 2015
4. Tim Golla, Christopher Schwartz, and Reinhard Klein. Towards Efficient Online Compression of Incrementally Acquired Point Clouds. In *Vision, Modeling & Visualization*. The Eurographics Association, 2014

1.6 Outline

The rest of this cumulative thesis is structured as follows: Chapter 2 introduces a memory-efficient statistical data representation for the memory-intensive bidirectional texture function (BTF) required for the metallic effect in the data-driven metallic car paint representation, suitable for real-time rendering. Chapter 3 describes how to use this representation as input for an optimal transport problem. Its solution is used for creating interpolations of two or more car paints suitable for interactive applications. Chapter 4 investigates how 3D point clouds of plants can be represented in a hierarchy of segments in order to facilitate the definition of an optimal transport problem. The solution of this problem can be used to generate an arbitrary number of intermediate states of real-world data, i.e. to create a temporal upsampling. The thesis closes with a summary, a conclusion and an outlook in Chapter 5, followed by the bibliography, a list of figures and a list of tables.

2 An Efficient Statistical Data Representation for Real-time Rendering of Metallic Effect Car Paints

Abstract Realistic virtual reality applications require highly-detailed geometry as well es convincing surface representations. In many applications, especially in the automotive industry, the realistic rendering of metallic effect paints is necessary. Due to their complex appearance, this is a demanding problem. Previous methods either use a computationally heavy-weight and often hand-tuned simulation approach or a data-driven approach. The former are thus not well-suited for real-time applications. The latter have the advantage of lower computational complexity and virtually no manual hand-tuning, but the disadvantage of requiring large amounts of the graphics card's memory, making them problematic for larger scenes with numerous materials as required in VR applications. In this paper, we describe an efficient representation for metallic car paints, based on computing the statistical properties of measured real-world samples. Our approach is suited for real-time rendering, poses only moderate requirements on the computing power, uses a low amount of memory and displays high-quality results, as shown in our evaluation section. As an additional advantage, our representation allows the generation of BTFs of arbitrary resolution.

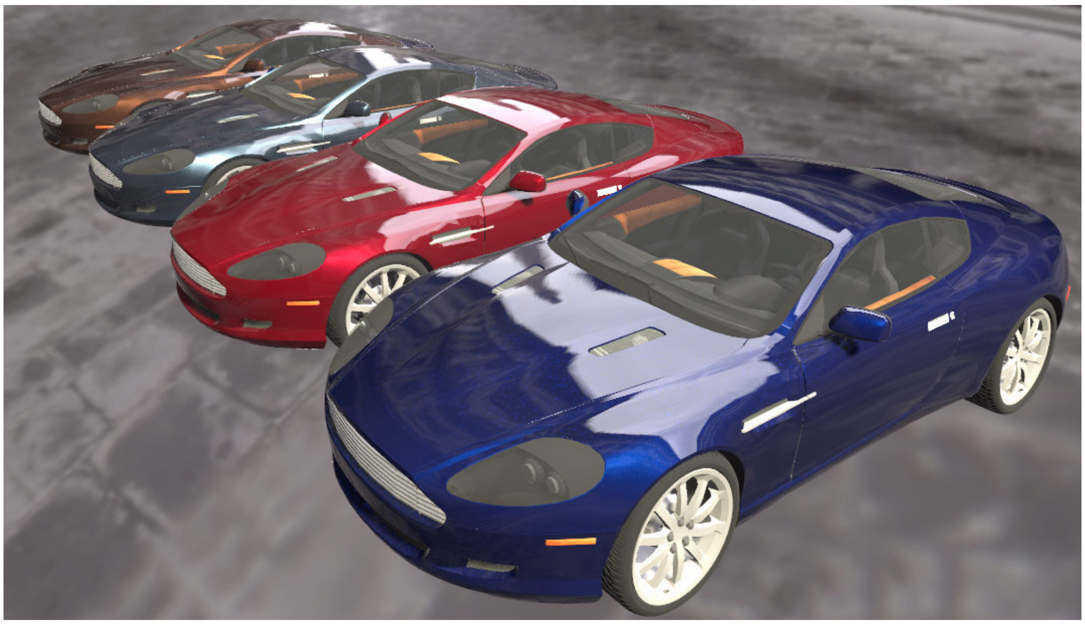


Figure 2.1: *Scene displaying various metallic paints on cars in a virtual showroom, as may be employed to show color choices to a customer. Thanks to our efficient statistical representation, various different digitally acquired real-world metallic paints can be rendered in real-time on commodity hardware.*

This chapter corresponds to the publication Tim Golla and Reinhard Klein. An Efficient Statistical Data Representation for Real-Time Rendering of Metallic Effect Car Paints. In *Virtual Reality and Augmented Reality: 14th EuroVR International Conference, EuroVR 2017*, pages 51–68. Springer, Cham, 2017 [GK17].

2.1 Introduction

Virtual reality applications are gaining popularity in the automotive industry, where applications range from virtual showrooms for customers to tools for designers. Metallic effect paints are among the most popular finishes for cars and thus must be included in these applications. The metallic paints consist of multiple layers and exhibit – among other optical effects – a metallic-looking sparkling effect. This is achieved by including additional effect pigments to the base paint, which are usually small aluminium flakes. A variation are pearlescent paints, that change color depending on the view and light angles, an effect also caused by flakes. Recreating realistic-looking car paints in computer graphics is a demanding problem. The sparkling effect of metallic paints is particularly difficult to recreate.

The rendering equation [Kaj86] is the theoretical basis for most rendering algorithms and can be written in simplified form as:

$$L_\mathrm{o}(\mathbf{x},\mathbf{o}) = L_e(\mathbf{x},\mathbf{o}) + \int_{\Omega_i^+} f_r(\mathbf{x},\mathbf{i},\mathbf{o}) L_\mathrm{i}(\mathbf{x},\mathbf{i})(\mathbf{i}\cdot\mathbf{n}) d\mathbf{i}, \tag{2.1}$$

where $\mathbf{x}$ is the position on the surface, $\mathbf{i}$ the incoming, $\mathbf{o}$ the outgoing light direction, $\mathbf{n}$ the surface normal, L_o the outgoing radiance, L_i the incoming radiance, L_e the emitted radiance, Ω_i^+ the hemisphere over $\mathbf{x}$ and f_r a reflectance function. The recreation of the metallic car paint effect can be described as defining a suitable reflectance function $f_r(\mathbf{x},\mathbf{i},\mathbf{o})$. For homogenous materials, the dependency on the spatial variable $\mathbf{x}$ can be omitted. Especially for the latter case, many analytical models exist [GGH+17, GGG+16]. When one wants to take non-local effects into account, one has to extend this equation to:

$$L_\mathrm{o}(\mathbf{x},\mathbf{o}) = L_e(\mathbf{x},\mathbf{o}) + \int_A \int_{\Omega_i^+(\mathbf{x}_i)} S(\mathbf{x}_i,\mathbf{i},\mathbf{x},\mathbf{o}) L_\mathrm{i}(\mathbf{x}_i,\mathbf{i})(\mathbf{i}\cdot\mathbf{n}) d\mathbf{i} d\mathbf{x}_i, \tag{2.2}$$

where A is the object's 2D surface and S is a scattering function. With this definition, light incident at $\mathbf{x}_i$ and scattered to $\mathbf{x}$ is taken into account.

Another approach is *image-based rendering*. Here, we compute *light fields* $L_{\cdot,V}$ parameterized on a volume's surface V. We are interested in the outgoing light field $L_{o,V}$, which can be computed by the equation

$$L_{o,V}(\mathbf{x},\mathbf{o}) = \int_V \int_{\Omega_i^+(\mathbf{x}_i)} R_V(\mathbf{x}_i,\mathbf{i},\mathbf{x},\mathbf{o}) L_{i,V}(\mathbf{x}_i,\mathbf{i}) d\mathbf{i} d\mathbf{x}_i, \tag{2.3}$$

where R_V is the reflectance field, which was introduced by Debevec et al. [DHT+00]. The equation is called *image-based relighting equation*. When assuming a directional incident light field $L_{i,V}^d$, which does not depend on the surface position $\mathbf{x}_i$, i.e. the light sources are in an infinite distance resulting in only parallel light rays, also called far field assumption, one can reduce the 8-dimensional reflectance field R_V to a 6-dimensional reflectance field B_V, which is called Bidirectional Texture Function (BTF) [Dis98]. The outgoing light field can then be computed with the following equation:

$$L_{o,V}(\mathbf{x},\mathbf{o}) = \int_{\Omega_i^+} B_V(\mathbf{i},\mathbf{x},\mathbf{o}) L_{i,V}^d(\mathbf{i}) d\mathbf{i} \tag{2.4}$$

Depending on the choice of the approach used for rendering metallic car paints, either model-based, specifying a BRDF or scattering function, or image-based by capturing the reflectance function, the corresponding equation is discretized and the final image can be computed. In any case, for a high-quality virtual reality application the approach of choice should fulfill the following requirements:

- High-quality look in order to convince and impress customers and be useful for designers.
- Relative ease of acquisition or generation to be usable for non-expert users and save time for experts.
- Evaluable in real-time, using as few computational power as possible to enable high frame rates.
- Low memory footprint in order to support the rendering of many different paints simultaneously and allow the use of more complex car and scene geometry.

The latter two points are even more important on mobile devices. Simple model-based approaches are unable to recreate the complex effects of car paints. Nevertheless, achieving high-quality results with model-based approaches is possible, as e.g. shown by Yan et al. [YHJ+14, YHMR16] and Jakob et al. [JHY+14], but these are computationally demanding and thus not suited for real-time applications as required for virtual reality. Many of these approaches furthermore require manual setting of numerous parameters, making them difficult to handle for non-experts.

Alternatively, image-based approaches, where a real-world sample of the material to be represented is digitized by taking images under varying illumination, deliver high-quality results, but inherently require large amounts of memory, which again hinders the usage of these representations in a VR scenario. To overcome this problem, Rump et al. [RMS+08] introduced a hybrid representation and decomposed the full BTF of a car paint into a BRDF and an easier to compress remaining BTF containing effects caused by the metallic flakes, which we will call in the following *flake BTF*. Although this representation yields high-quality renderings and is relatively straightforward to generate, it still requires large amounts of memory. In a VR application, this permits only the simultaneous use of a small number of different car paints. Based on this approach, X-Rite developed the car paint model used in the AxF file format [ML15], which is used by the TAC7 device and has already been included in several commercially available software packages like Autodesk VRED, Nvidia Iray or X-Rite Pantora. This representation fulfills all requirements except for its memory footprint. We thus use it as a basis for our new representation for metallic paints.

The flake BTF requires by far the largest amount of memory of the model. While in the original representation by Rump et al., as well as the AxF format, the flake BTF is represented as a 4D matrix, we compute a statistical representation of this data, suited for real-time reconstruction on the graphics card. Our representation requires much less memory and only moderate computing power. As an additional advantage, our representation allows the generation of BTFs of arbitrary resolution, independent of the resolution of the originally acquired data.

The rest of this paper is structured as follows: In Section 2.2, we describe the related work. In Section 2.3, we explain the original representation used in the AxF car paint model. In Section 2.4, we describe our model for the metallic flakes, our statistics generation algorithm and our real-time reconstruction algorithm. In Section 2.5, we provide results and an evaluation. In Section 2.6, we describe limitations of the approach, followed by our conclusion in Section 2.7.

2.2 Related Work

We only focus on related work specific to the rendering of metallic car paints. For a more general overview, we refer the reader to the literature like the SIGGRAPH 2017 course *Material Capture and Representation with Applications in VR* by Guarnera et al. [GGH+17], the state of the art report by Guarnera et al. [GGG+16] and the textbook *Digital modeling of material appearance* by Dorsey et al [DRS10].

The first works on measurement and rendering of car paints were done by Takagi et al. in 1990 [TTOO90]. Another team lead by Takagi published more findings 15 years later [TWB05]. Dumont-Bècle et al. [DBFK+01] presented a multi-texture approach. They didn't show result images, which makes it hard to compare their results to others. Kitaguchi [Kit08] provided a detailed introduction into the physics of metallic paints. Ershov et al. [EKK99] presented a physically-based, analytical model, which they improved further in their subsequent publications [EKM01, EĎKM04]. They achieved good-looking results, which however require a large amount of parameters, which are difficult to set by hand. Ďurikovič and Martens [ĎM03] modeled the flake sparkling by explicitly modeling their geometry. This however makes the approach not well-suited for real-time applications. Ngan et al. [NDM05] showed that the Cook-Torrance model [CT82] is well-suited for car paints. Günther et al. [GCG+05] described the complete process from measuring to real-time rendering of car paints. They fit analytical models to their measurements. For the metallic flakes, they draw on ideas from Ershov et al. [EKK99] and Ďurikovič and Martens [ĎM03] and procedurally generate a normal map which represents the flakes. Rump et al. [RMS+08] introduced the combined model for metallic and pearlescent paints, on which the AxF car paint model [ML15] and hence our model is based. It is similar to the model by Günther et al. [GCG+05], with the main difference of using a measured BTF for the metallic flakes instead of the procedural approach. They experimented with PCA-based compression for the BTF part, but achieved only a moderate compression ratio of 1:4 on their data. Later, they proposed a flake BTF compression algorithm based on selecting representative image patches [RSK09].

The patch computation is quite involved. They report compression rates of 1:18 to 1:46 on their data, depending on the dataset. The AxF format builds on this compressed representation. Later, Ďurikovič and Mihálik described a similar approach [ĎM13]. They used an 8 Bit texture to generate the sparkle effect. The shown results are good, but the BTF approach seems to deliver a higher quality.

Kurt et al. [KSKK10] suggested a novel BRDF model that surpasses the Cook-Torrance model for car paint in their evaluation. Another high-quality BRDF model for glossy surfaces was presented by Löw et al. [LKYU12]. Being homogeneous BRDFs, their models however cannot account for the spatially varying metallic flakes.

Yan et al. [YHJ+14, YHMR16] and Jakob et al. [JHY+14] presented high-quality simulation models for rendering of glints and metal surfaces. Their approaches are computationally intensive and thus not ideal for real-time applications. Related is also the publication by Raymond et al. [RGB16], who render scratched metal surfaces, but no metallic paints. Atanasov and Koylazov [AK16] presented an approach specialized on metallic flakes rendering, which is however not real-time capable.

2.3 Original AxF Car Paint Model

Our representation for metallic paints is based on the car paint model used in the AxF file format [ML15], which has already been included in several commercially available software packages like Autodesk VRED, Nvidia Iray or X-Rite Pantora. The format is largely based on the work of Rump et al. [RMS+08] and consists of a combination of multiple models:

- A measured clear cloat layer, that changes incoming and outgoing directions $\mathbf{i}, \mathbf{o}$ to $\bar{\mathbf{i}}, \bar{\mathbf{o}}$, depending on the thickness and refraction index of this layer.
- A Lambertian BRDF $\frac{a}{\pi}$

- A multi-lobe Cook-Torrance BRDF [CT82] for the brightness, where the k-th lobe is defined as:

$$f^{CT}_{s_k,\alpha_k,F_{0,k}}\left(\bar{\mathbf{i}},\bar{\mathbf{o}}\right) = \frac{s_k}{\pi}\frac{D_{\alpha_k}(\bar{\mathbf{h}})F_{F_{0,k}}\left(\bar{\mathbf{h}},\bar{\mathbf{o}}\right)G\left(\bar{\mathbf{i}},\bar{\mathbf{o}}\right)}{\bar{\mathbf{i}}_z\,\bar{\mathbf{o}}_z}, \tag{2.5}$$

where $\bar{\mathbf{h}}$ is the half vector, s_k is the specular coefficient,

$$D_{\alpha_k}(\bar{\mathbf{h}}) = \frac{1}{\alpha_k^2\bar{\mathbf{h}}_z^4}e^{\frac{\bar{\mathbf{h}}_z^2-1}{\bar{\mathbf{h}}_z^2\alpha_k^2}} \tag{2.6}$$

is the microfacet distribution,

$$F_{F_{0,k}}(\bar{\mathbf{h}},\bar{\mathbf{o}}) = F_{0,k} + (1-F_{0,k})(1-\bar{\mathbf{h}}\cdot\bar{\mathbf{o}})^5, \tag{2.7}$$

where

$$F_{0,k} = \left(\frac{n_{1,k}-n_{2,k}}{n_{1,k}+n_{2,k}}\right)^2 \tag{2.8}$$

is Schlick's approximation [Sch94] of the Fresnel term, $n_{1,k}, n_{2,k}$ are the refractive indices and

$$G(\bar{\mathbf{i}},\bar{\mathbf{o}}) = \min\left(1, \frac{2\bar{\mathbf{h}}_z\bar{\mathbf{o}}_z}{\bar{\mathbf{h}}\cdot\bar{\mathbf{o}}}, \frac{2\bar{\mathbf{h}}_z\bar{\mathbf{i}}_z}{\bar{\mathbf{h}}\cdot\bar{\mathbf{o}}}\right) \tag{2.9}$$

is the geometry term, where

$$\mathbf{y}_z = \mathbf{y}\cdot\mathbf{n}, \mathbf{y}\in\{\bar{\mathbf{h}},\bar{\mathbf{i}},\bar{\mathbf{o}}\} \tag{2.10}$$

denotes the dot product of $\mathbf{y}$ with the surface normal.
- A 2D color table $\chi(\theta_{\bar{\mathbf{h}}},\theta_{\bar{\mathbf{i}}})$, modulating the brightness in order to take view-dependent color shifts into account. It is parametrized by the angles $\theta_{\bar{\mathbf{h}}}$ and $\theta_{\bar{\mathbf{i}}}$, where $\theta_{\bar{\mathbf{h}}} = \arccos(\bar{\mathbf{h}}_z)$ is the angle between half vector and normal and $\theta_{\bar{\mathbf{i}}} = \arccos(\bar{\mathbf{h}}\cdot\bar{\mathbf{i}})$ is the angle between half vector and incoming direction.
- A BTF taking the effects caused by the metallic flakes into account. It is parameterized by $\theta_{\bar{\mathbf{h}}}$, $\theta_{\bar{\mathbf{i}}}$ and $\mathbf{x}\in\mathbb{R}^2$, the position on the surface, i.e. it is

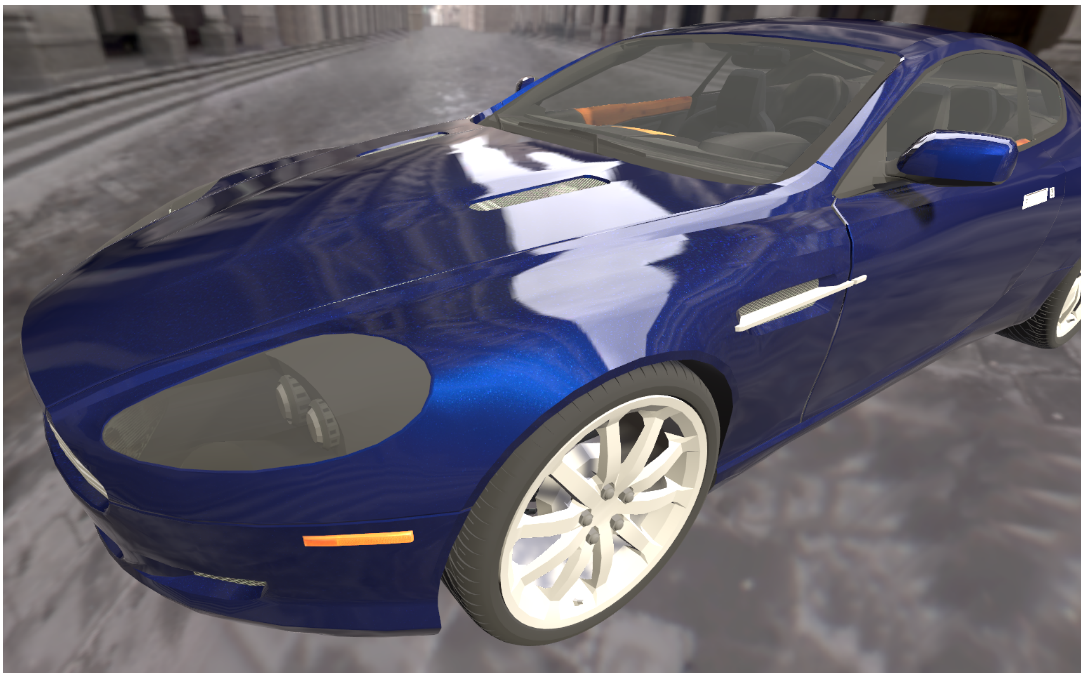


Figure 2.2: *Real-time rendering of a car with our metallic car paint representation.*

a 4D table, denoted as $\Xi(\mathbf{x}, \theta_{\bar{\mathbf{h}}}, \theta_{\bar{\mathbf{i}}})$. According to Rump et al. [RMS+08], the angular lifetime of a metallic flake is around 6-7 degrees, which is why an angular sampling of 24-30 samples along each direction was chosen. Each combination of $\theta_{\bar{\mathbf{h}}}$ and $\theta_{\bar{\mathbf{i}}}$ results in a 2D texture. A typical AxF file contains 68 of these. In the following we call this function *flake BTF*.

The complete model is then:

$$f(\mathbf{x}, \bar{\mathbf{i}}, \bar{\mathbf{o}}) = \chi(\theta_{\bar{\mathbf{h}}}, \theta_{\bar{\mathbf{i}}}) \left(\frac{a}{\pi} + \sum_{k=1}^{K} f^{CT}_{s_k, \alpha_k, F_{0,k}} \left(\bar{\mathbf{i}}, \bar{\mathbf{o}} \right) \right) + \Xi(\mathbf{x}, \theta_{\bar{\mathbf{h}}}, \theta_{\bar{\mathbf{i}}}) \tag{2.11}$$

Choosing three lobes, i.e. $K = 3$, was shown to deliver good results [GCG+05, ML15].

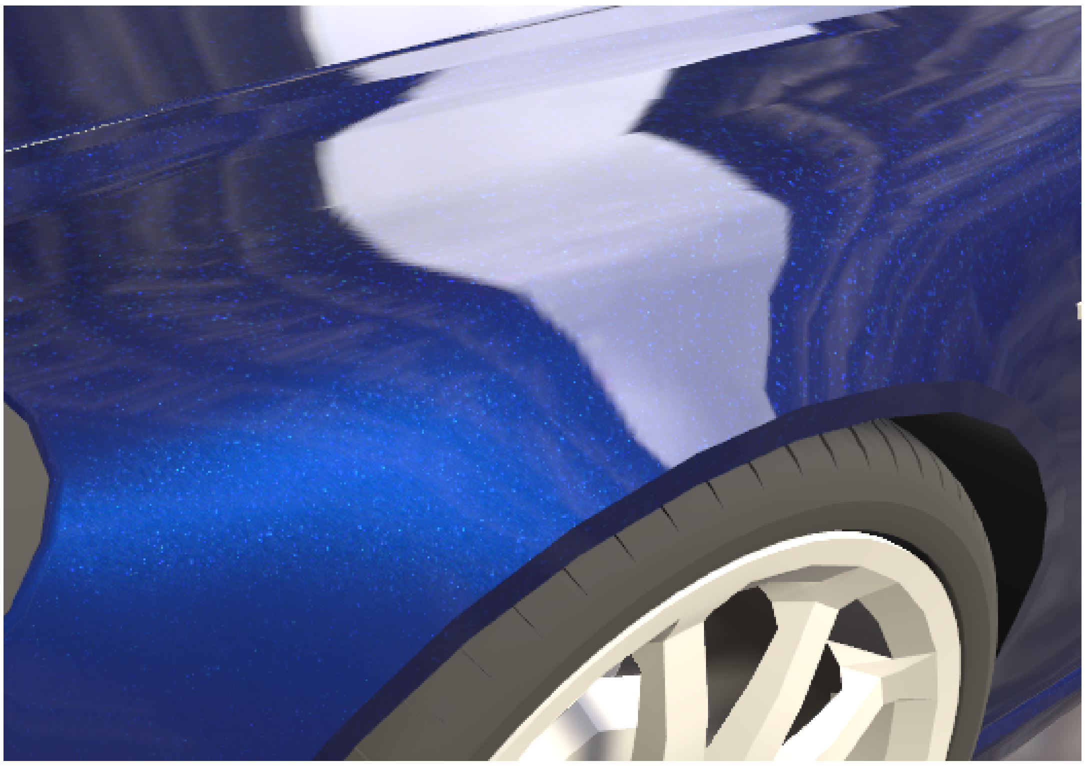


Figure 2.3: *Closeup of the car shown in Figure 2.2*

2.4 Our Representation and its Generation

The AxF car paint model delivers high-quality results, but uses a great amount of the graphics card's memory for each material. While most parts of the model are described by only a small number of parameters, the flake BTF $\Xi(\mathbf{x}, \theta_{\bar{\mathbf{h}}}, \theta_{\bar{\mathbf{i}}})$ consists of a big number of – possibly large – textures, and thus is by far the most memory-consuming part of the model. As pointed out by Rump et al., traditional high-ratio image compression methods like JPEG are not suited for the BTF, as they tend to smear out or remove the very small flakes and also may introduce artifacts and noise. We therefore use the BRDF and color table parts of the original model, but replace the memory-intensive BTF with our statistical model which is computed on the measured data. Our representation is based on the following observations:

- The absolute position of the BTF's pixels representing e.g. the flakes is irrelevant for a metallic paint's unique look.
- The flake BTF colors and their distributions are very important for a paint's unique look.
- These two properties vary strongly with the light and viewpoint orientation.
- The positions of the flakes follow a uniform random distribution on the surface.
- Due to their extremely small size, each metallic flake usually only covers one pixel in an image generate by current acquisition devices.
- The positions of the color pixels follow a uniform random distribution on the surface.

These observations allow us to reduce the flake BTF to a statistical representation, from which a similar-looking BTF can be reconstructed in real-time on the graphics card.

2.4.1 Generation of the Statistical Representation

We start with the car paint representation used in the AxF format. In an offline step, running on the CPU, we convert the flake BTF representation $\Xi(\mathbf{x}, \theta_{\bar{\mathbf{h}}}, \theta_{\bar{\mathbf{i}}})$ to our statistical representation.

The AxF car paint BTF representation discretizes the view- and lighting hemisphere by the angles $\theta_{\bar{\mathbf{h}}}$ and $\theta_{\bar{\mathbf{i}}}$. Usually 68 angle combinations are used, in combination with interpolation to generate a high-quality look. This number was chosen so that one combination roughly encompasses the angular lifetime of one flake [RSK09]. Each of the following steps is performed independently for each of the angular combinations.

Color Octree Generation

For each combination of $\theta_{\bar{\mathbf{h}}}$ and $\theta_{\bar{\mathbf{i}}}$ chosen in the original model, i.e. one flake texture, we compute statistical information on the color distribution. For this, we employ an octree, somewhat similar to octree color quantization [GP88, BX06]. We first compute a sparse color octree of a predefined depth d for all pixels in one BTF texture. d should be chosen as large as possible in order to achieve maximum precision. Note however, that the number of leaf nodes can be up to 8^d. All pixel color values are stored in the leaf nodes. 8^d should not exceed the number of pixels, since then no gain in precision is possible anymore, as each leaf will then contain a single color value. One leaf may represent multiple pixels, since they can have the same color.

Now the pruning process starts. We iteratively remove the octree leaves that are most balanced concerning the number of color values per volume in color space. This means that sibling octree nodes are only combined if they contain a similar number of colors. The idea behind this is that we want to generate a predefined maximum number c_{max} of color clusters – defined by the contents of the final leaf nodes – whose contents are as uniformly distributed as possible. The larger c_{max}, the more accurate the reconstruction will be. However, memory usage rises linearly with it. Note that the resulting number of leaf nodes/clusters l may be less or equal c_{max}. The algorithm is given in pseudo-code in Algorithm 1.

Statistics Computation

After generating the tree, we sort the color clusters defined by the leaf nodes in an arbitrary order. For each cluster, we count the number of colors in it and compute the bounding box in color space. From the numbers of colors for all

Algorithm 1 Density-based Octree Pruning

```
function getAverageChildDensityDisparity(Node $n$)
    $d \leftarrow [\infty, \dots, \infty] \in \mathbb{R}^8$
    for $i \in [0, \dots 7]$ do
        if $i$th child node is a leaf then
            $v \leftarrow$ volume of bounding box of colors in child node $i$
            $c \leftarrow$ number of color values in the leaf node
            if $c > 0$ then
                $d[i] \leftarrow \frac{c}{v}$
            else
                $d[i] \leftarrow 0$
            end if
        end if
    end for
    $\bar{d} \leftarrow \frac{1}{8} \sum d[i]$
    $\forall i : d[i] \leftarrow d[i] - \bar{d}$
    return $\frac{1}{8} \sum d[i]$                ▷ returns $\infty$ if not all child nodes are leaves
end function

function testAndAppend(n,C,D)
    $d \leftarrow$ getAverageChildDensityDisparity($n$)
    if $d \neq \infty$ then
        append $n$ to $C$
        append $d$ to $D$
    end if
end function

function Density-based Octree Pruning(Octree $O$, max nr of leaves $c_{\max}$ )
    Initial Candidate Nodes $C_{init} \leftarrow$ all inner nodes
    $C \leftarrow []$
    $D \leftarrow []$
    for all $n \in C_{init}$ do
        testAndAppend($n, C, D$)
    end for
    while number of leaves $> c_{max}$ do
        $i \leftarrow \arg\min D[i]$
        for all child node $n$ of $C[i]$ do
            Collect color data from $n$ and attach to $C[i]$
            Remove $n$ from $O$
        end for
        $p \leftarrow$ parent($n$)
        testAndAppend($p, C, D$)
        Remove $D[i]$ from $D$
        Remove $C[i]$ from $C$        ▷ $C[i]$ became a leaf and is no longer a candidate
    end while
end function
```

clusters, we can compute their discrete probability distribution. From this, we compute the cumulative distribution function.

Mipmapping

We compute the mipmaps separately for each angle combination on the original data. We then generate the octrees and statistics separately for each mipmap level.

Texture Generation

As we want to reconstruct the flake BTF in the GPU fragment shader, we store all required information in textures. We have a cumulative distribution function per angle combination and per mipmap level. Each function can be represented as one row in a single color floating point texture. We first store all functions of mipmap level 0, then all functions of level 1 and so on in a single texture. See Figure 2.4 for an example. Note that this texture has a fixed width of c_{max}, the previously specified maximum number of leaf nodes. However, as the color octrees might have a smaller number of leaf nodes l, the discrete cumulative distribution functions will also have only l sample points. We assign a probability of 0 to the remaining $c_{max} - l$ points, i.e. they have a cumulative value identical to their predecessor, usually 1.

Similarly, we store two opposing leaf nodes' bounding box corner points in a separate three color texture of width $2c_{max}$. Again, as there may be only l leaf nodes, we fill the remaining $2(c_{max} - l)$ pixels with black or any other arbitrary color. These "fill" colors are never used since their probability is 0. See Figure 2.5 for an example.

In addition, for the reconstruction on the graphics card, we require pseudo-random numbers that remain fixed on the surface in order to guarantee a constant look from frame to frame. If we would generate new random number for each frame or when changing the viewpoint, new colors would be generated every time, manifesting in flicker. For this, we generate a floating point, single color texture with pseudo-random pixel values between 0 and 1. See Figure 2.6 for an example. In order to obtain a BTF of the same resolution as in the original AxF representation, for simplicity we choose the random

Figure 2.4: *Example cumulative distribution function texture for one mipmap level. One row represents one discrete cumulative distribution function. Black means zero probability, the brightest values mean probability 1.*

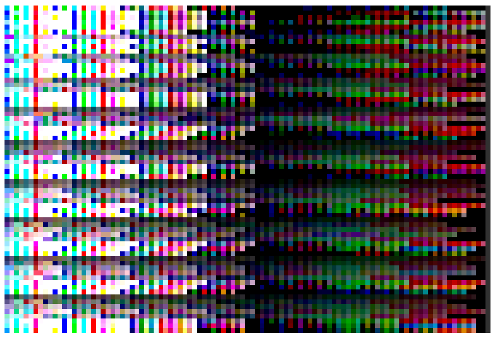


Figure 2.5: *Example color bounding box corners for one mipmap level. One row represents the color bounding boxes belonging to one discrete cumulative distribution function. Even pixels are the colors of "lower left" box corners, the pixels on the right to each the respective opposite "upper right" corner color.*

texture's resolution identical to the AxF BTF's resolution. In our case, this was 480×480. Note that with the help of coordinate transformations, one could choose different resolutions to obtain similarly-looking results. As an additional advantage, our representation also allows the generation of BTFs of arbitrary resolution by varying the random texture's resolution or coordinate transformation. The wrapping mode for the random texture should be set to *repeat* – see Section 2.4.2.

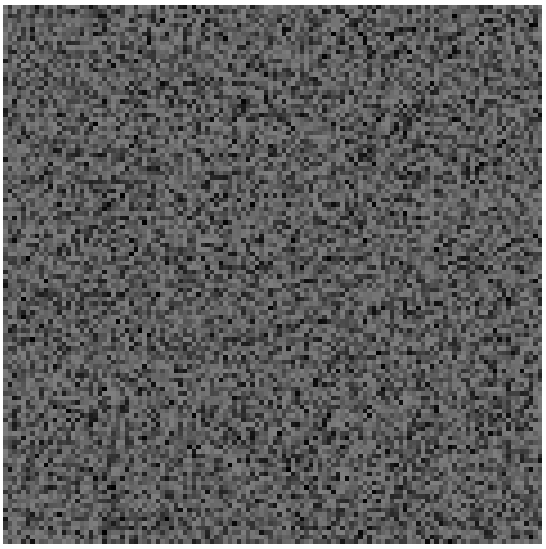


Figure 2.6: *Part of an example pseudo-random value texture. It is used for BTF reconstruction.*

2.4.2 Real-time BTF Reconstruction and Rendering

In the fragment shader, we first compute the angle combination $\theta_{\bar{\mathbf{h}}}$ and $\theta_{\bar{\mathbf{i}}}$ and its discretization step, as in the AxF specification. Additionally, we compute the mipmap level. According to these values, we can compute the correct row in our cumulative distribution function and bounding box textures. According to the standard *uv* values selected by the default pipeline, a pseudo-random value is read from the pseudo-random texture.

In a loop, we walk along the columns of the cumulative distribution function texture, till we reach a value larger than the random value previously picked. This gives us the column number i. Note that while this approach requires a loop of size c_{max} (usually small) and as many texture lookups, in comparison to using the inverse cumulative distribution function, which would only require one texture lookup, we prefer it, as it has a *natural* discretization, whereas the inverse function would have to be newly discretized, possibly leading to sampling artifacts and in this case also the requirement of more sampling points, i.e. a larger texture, in order to reduce the artifacts.

i specifies the column numbers in the color bounding box texture we have to use: $2i$ and $2i+1$. The row is the same as in the cumulative distribution function texture. Accessing these returns the color bounding box corners. We now require three additional pseudo-random numbers in order to generate a color value within the bounding box – one for each color channel. While in a naive implementation, one could use the previously generated random number, this would couple the generated color to the (arbitrary) position of the color cluster, resulting in a tendency to generate colors nearer to the "lower left" bounding box corner, i.e. darker colors, for color cluster with a low index and brighter colors for clusters with a high index. While one could generate additional pseudo-random textures in order to generate these pseudo-random values, we chose a more memory-efficient approach: We specify arbitrary, fixed offsets $(0,0) < o_1, o_2, o_3 < (1,1)$ and read the pseudo-random values $(u,v) + o_j, j \in \{1,2,3\}$ from our original pseudo-random texture. Note that these texture coordinates may be larger than $(1,1)$, which is why the texture mode has to be set to *repeat*. With these three random values, we can determine a color in the color bounding box, which is our final flake color. In order to provide smooth results when slowly changing the observed angle combination, i.e. changing the camera's or light's orientation, we perform a bilinear interpolation between the colors resulting from neighboring angle discretization steps.

2.5 Evaluation

2.5.1 Visual Comparison

See Figure 2.7 for a visual comparison between the AxF representation and our statistical representation of a light gray metallic paint. Figure 2.8 contains a comparison between the data-driven AxF rendering and a rendering with our statistical representation of a gray-blue metallic paint. Figure 2.9 shows extreme close-up views of the AxF and our statistical representation of the light gray paint. Note that on the pixel level, one can clearly see differences between the two representations. Under normal viewing conditions, both representations convey very similar impressions. Figures 2.10 and 2.11 contain further examples. Figure 2.12 contains a particularly interesting example: A pearlescent or flip-flop effect paint, that changes its color from green to blue.

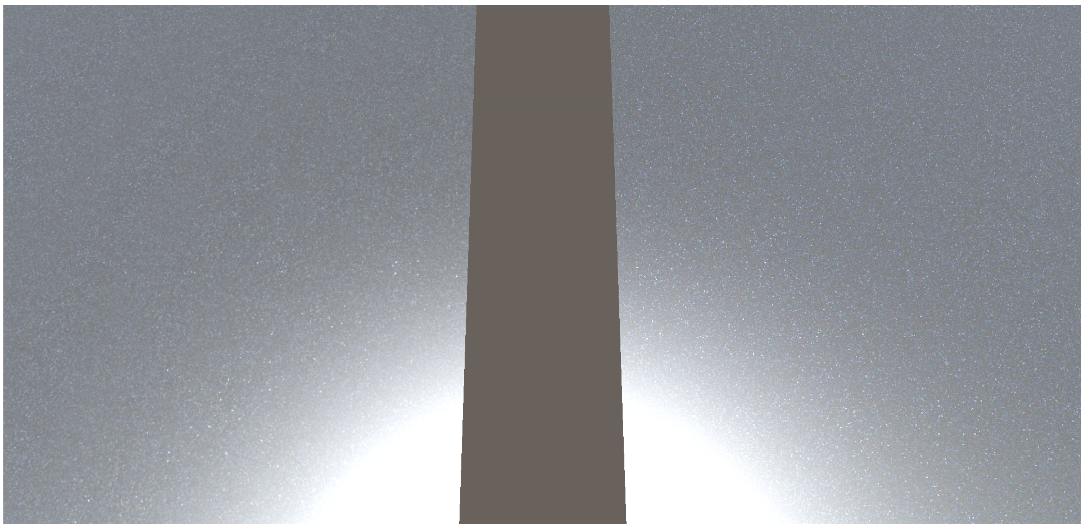


Figure 2.7: *A light gray metallic paint. Left: Rendering of the original AxF car paint representation. Right: Rendering of our statistical representation, which uses only a fraction of the AxF's memory requirement.*

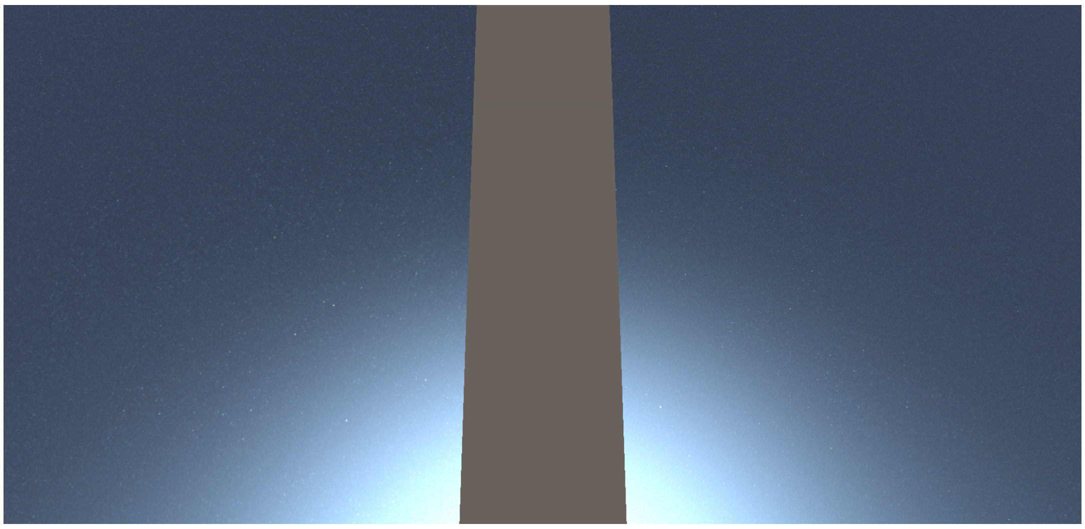


Figure 2.8: *A gray-blue metallic paint. Left: Rendering of the original AxF car paint representation. Right: Rendering of our statistical representation, which uses only a fraction of the AxF's memory requirement.*

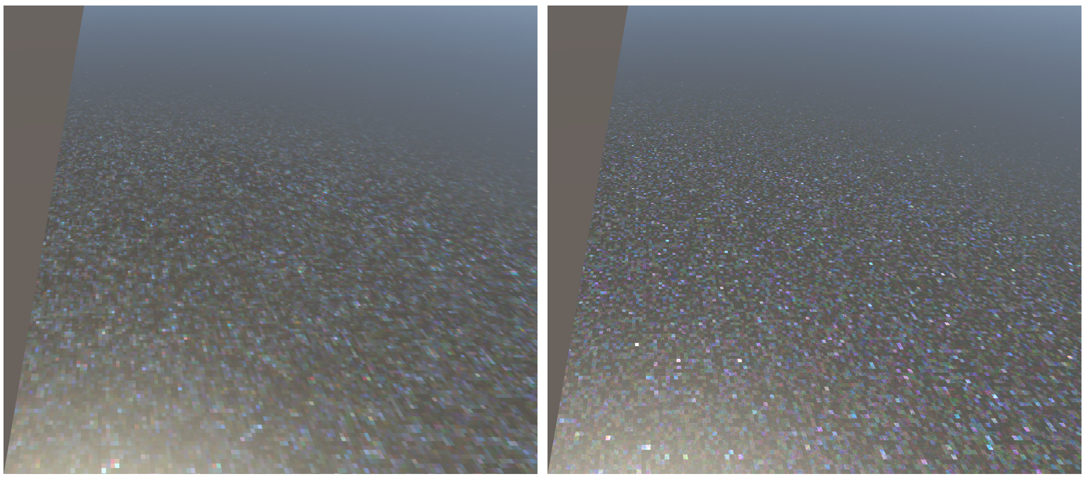


Figure 2.9: *Extreme close-up view of the light-gray paint. Left: AxF representation. Right: Our statistical representation. Note that on the pixel level, one can clearly see differences between the two representations.*

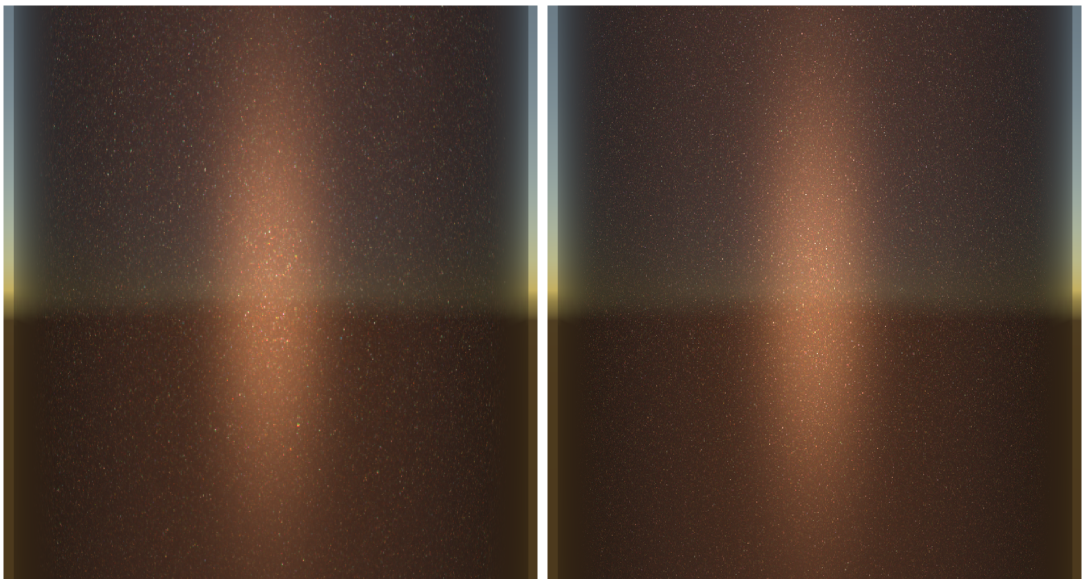


Figure 2.10: *A brown metallic paint on a cylinder. Left: AxF representation. Right: Our statistical representation.*

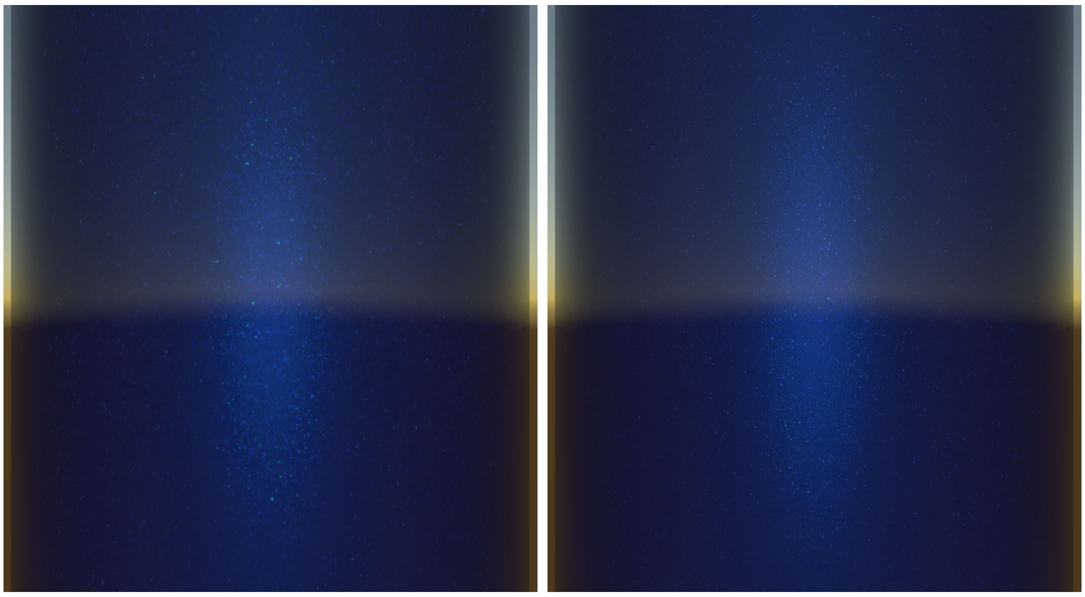


Figure 2.11: *An intense blue metallic paint on a cylinder. Note that in this case, most flakes are blue, too. Left: AxF representation. Right: Our statistical representation.*

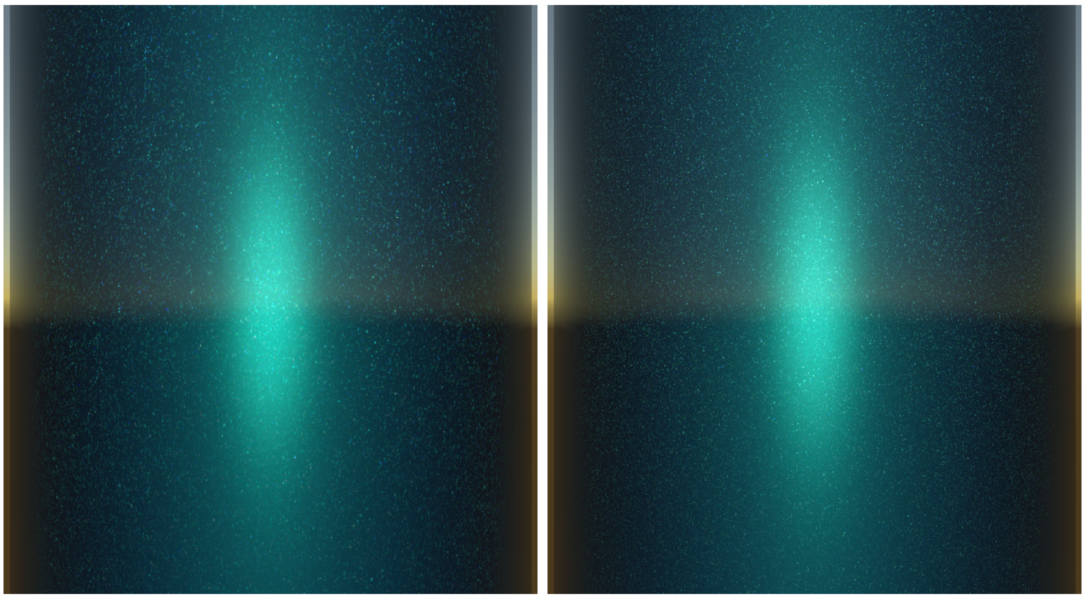


Figure 2.12: *A pearlescent effect paint on a cylinder. It changes its color from green to blue, depending on the light-view combination. Left: AxF representation. Right: Our statistical representation.*

Figure 2.2 shows the blue metallic paint on a car model, Figure 2.3 shows a closeup.

2.5.2 Memory Usage and Rendering Times

In order to measure the memory usage and computation times, we use a simple test scene, consisting only of a quad, one directional light, one camera and and a simple environment map. We used a first-generation Nvidia GeForce Titan for evaluation. When no material is applied to the quad, the graphics memory requirement of the scene is 19.2 MB. Average rendering time was 2.0 ms.

The following numbers are valid for all car paints we tested, as their AxF representation as well as their statistical representation is identical concerning memory and computation time usage. We set $c_{max} = 50$. Generation of the representation took 53 s in our unoptimized Python implementation, including file IO on a standard hard drive.

When applying the original AxF car paint material, the memory usage went up to 202.8 MB. Deducing from this, the memory requirement of the material is 183.6 MB. Average rendering time was 4.6 ms.

Applying the material in our statistical representation resulted in a total graphics memory usage of 23.8 MB. Deducing from this, the memory usage of the material alone is 4.6 MB. This means that our representation uses only 2.5 % of the AxF car paint's required amount of memory, respectively a compression ratio of about 1:40 in comparison to the AxF format. Average rendering time was 5.8 ms.

Assuming 1 GB (1024 MB) of graphics memory available for materials, the original AxF car paint representation would allow up to 5 different metallic paints per scene. Our representation would allow up to 222 different paints.

2.6 Limitations

By design it is impossible to reconstruct a BTF from our representation that is identical to the original representation on the pixel level. As argued above, this is irrelevant for the metallic paints. The described approach is only valid as long as the assumption holds, that one metallic flake only occupies one pixel in the acquired data. This was the case for all samples we studied.

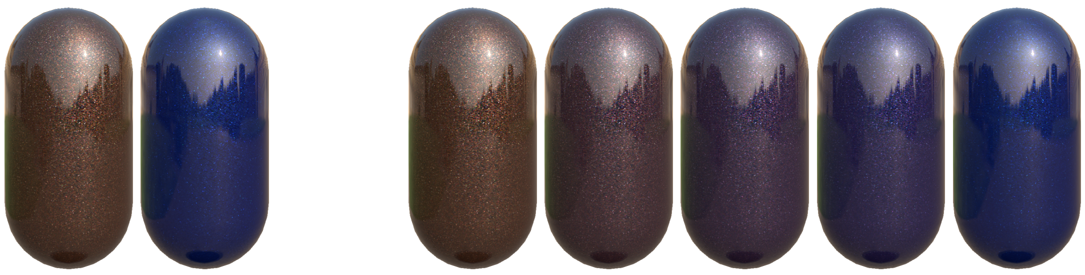
2.7 Conclusion

We presented an efficient statistical data representation for metallic effect car paints, based on measured real-world samples, allowing for real-time high-quality renderings in VR applications. We described our representation generation algorithm and our real-time reconstruction shader in detail. Despite the limitations described in the previous section, our approach proved very useful in practice. Our representation shares the high-quality look and relative ease of use of previous data-driven approaches. In addition, it allows to generate car paint materials of arbitrary size and resolution, independent of the acquisition resolution. While only slightly increasing computation times, it greatly reduces the amount of graphics memory required. This allows for using a much higher number of different metallic paints in virtual reality environments than previously possible with data-driven representations, as required by applications in the automotive industry.

Acknowledgments

We would like to thank Volkswagen and X-Rite for providing measurements of the metallic paints. Uffizi Gallery Light Probe Image ©1999 Paul Debevec, http://www.debevec.org/Probes/

3 Interactive Interpolation of Metallic Effect Car Paints



(**a**) *Measured metallic car paints*

(**b**) *Measured and interpolated metallic car paint materials*

Figure 3.1: *(a) Two measured metallic car paints. (b) Measured car paints and car paints generated with our approach: Interpolation of the brown paint towards the blue paint in 25% steps. All parameters are interpolated, including the metallic sparkling effect caused by metallic flakes.*

Abstract Metallic car paints are visually complex materials that, among others effects, exhibit a view-dependent metallic sparkling, which is particularly difficult to recreate in computer graphics. While capturing real-world metallic paints is possible with specialized devices, creating these materials computationally poses a difficult problem. We present a method that allows for interactive interpolation between measured metallic automotive paints, which can be used to generate new realistic-looking metallic paint materials. By clustering the color information present in the measured bidirectional

texture function (BTF) responsible for the metallic sparkling effect, we set up an optimal transport problem between metallic paints' appearances. The design of the problem facilitates efficiently finding a solution, based on which we generate a representation that allows for real-time generation of interpolated realistic materials. Interpolation happens smoothly, no flickering or other visual artifacts can be observed. The developed approach also enables to separately interpolate the larger-scale reflective properties, including the basic color hue, the local color hue, and the sparkling intensity of the metallic paint. Our method can be used intuitively in order to generate automotive paints with a novel appearance and explore the space of possible metallic paints spanned by given real-world measurements. The resulting materials are also well suited for real-time rendering in standard engines.

This chapter corresponds to the publication Tim Golla and Reinhard Klein. Interactive Interpolation of Metallic Effect Car Paints. In *Vision, Modeling & Visualization, VMV 2018*, EG VMV '18, page 11–20. The Eurographics Association, 2018 [GK18].

3.1 Introduction

Metallic paints are complex materials that are widely used in the car industry. They exhibit a sparkling effect that is caused by small effect particles – usually aluminium flakes – which is difficult to render in a convincing way. Metallic automotive paints convey a premium look and thus are the most popular kind of paints used for new cars. Realistic rendering of metallic paints in real-time is of great importance for the car industry. Typical applications are in the field of marketing, where real-time applications for customers are becoming more and more common, for example on the car manufacturer's websites as well as in the design area. Further fields are the game industry and even the movie industry where real-time rendering is used for quick prototypical viewing of virtual scenes.

For designers not only realistic rendering but also the ability to generate new variants of the paints is very important. This imposes tremendous technical challenges. Finding the desired paint should be intuitive and efficient, which requires interactive editing of the appearance, since otherwise the exploration of the space of possible paints is very difficult.

Current models for car paints are either fully analytical, partially, or completely data-driven. Analytical models provide great freedom with the ability to generate more or less any kind of paint. They however have drawbacks in terms of usability: It is often tedious for non-expert users to generate new automotive paints' appearances because of the large number of parameters which are not intuitively usable. Testing parameters to realize the diversity of appearances is usually time-consuming. Furthermore, the high quality models are computationally heavy and thus not suited for real-time applications. Fully or partially data-driven models are easier to use, because real-world materials can be measured. They are often also computationally simpler than analytical models, making them better suited for real-time applications. It is however much harder to generate new materials from given measured real-world samples and specialized synthesis methods have to be used.

In this paper, we present a novel method for interactively synthesizing new metallic paints by inter- and extrapolating measured materials. For this purpose we build on a recently developed real-time capable car paint model, which among other parameters exploits the statistics of the measured materials. We formulate the correspondence computation between two of these statistical representations as an optimal transport problem, which can be solved efficiently. Based on the resulting correspondence map, a fast interactive inter- and extrapolation of measured paints is realized. The key contributions of our work can be summarized as follows:

- We present a simple and fast method for solving the interpolation problem for measured car paints. In comparison to the related works of Bonneel et al.[BVDPPH11] and Solomon et al.[SDGP+15], our method is easier to implement and handles a measured bidirectional texture function (BTF) representation.
- We show how to separately interpolate the larger-scale reflective properties, including the basic color hue, the local color hue and the sparkling

intensity of the metallic paints' appearances in order to allow greater artistic freedom.

- We describe a representation suited for real-time editing and rendering of the metallic paints.

3.2 Related Work

We focus on the related work most closely related to this paper. For a more general overview of material acquisition and rendering, we refer the reader to the literature like the SIGGRAPH 2017 course *Material Capture and Representation with Applications in VR* by Guarnera et al. [GGH+17], the state of the art report by Guarnera et al. [GGG+16] and the textbook *Digital modeling of material appearance* by Dorsey et al [DRS10].

3.2.1 Metallic Car Paint Rendering

This subsection largely follows the overview given by Golla and Klein [GK17]. In 1990, first research on measurement and rendering of car paints were done by Takagi et al. [TTOO90]. Takagi continued to work in this field and published further research later [TWB05]. Further pioneers in this area were Dumont-Bècle et al. [DBFK+01], who presented a multi-texture approach. Ershov et al. [EKK99] presented a good-looking physically-based, analytical model. They further improved this in their later publications [EKM01, EĎKM04]. A disadvantage of their method is the large amount of parameters, which are difficult to select for non-expert users. A detailed introduction to the physics behind the appearance of metallic paints was given by Kitaguchi [Kit08].

In the approach of Ďurikovič and Martens [ĎM03] the geometry of metallic flakes is modeled explicitly in order to simulate their sparkling. This however leads to a huge number of polygons, which is not suitable for many applications, especially in the real-time context. Ngan et al. [NDM05] were able to show that the Cook-Torrance model [CT82] can represent the large-scale shininess of car paints well. A complete process from measuring to real-time rendering of car paints was presented by Günther et al. [GCG+05]. They rely on fitting analytical models to their measurements. To reproduce the sparkling, they used ideas from Ershov et al. [EKK99] and Ďurikovič and Martens [ĎM03] and procedurally generate a normal map which represents the flakes.

A combined model for metallic and pearlescent paints was presented by Rump et al. [RMS+08]. Later the AxF car paint model [ML15] was developed based on their research. They use a measured BTF for the metallic flakes, in contrast to the otherwise similar model by Günther et al. [GCG+05], where a procedural technique was used. Rump et al. also experimented with PCA-based compression for the BTF part. They achieved a moderate compression ratio of 1:4 on their data. In their follow-up paper [RSK09], they described a compression algorithm based on selecting representative image patches for the flake BTF. Depending on the dataset, they report compression rates of 1:18 to 1:46 on their data, The AxF format builds on this compressed representation. Golla and Klein [GK17] based their model on the AxF car paint model, but replaced the image-based BTF by a statistical representation based on the measurements. This way, they are able to achieve very good compression ratios.

Another approach was presented by Ďurikovič and Mihálik [ĎM13], who used an 8 bit texture to generate the sparkle effect. While yielding good results, the BTF approach seems to deliver a higher quality. A new BRDF model surpassing the Cook-Torrance model for car paints was introduced by Kurt et al. [KSKK10]. Another novel BRDF model for glossy surfaces was suggested by Löw et al. [LKYU12]. However, their models cannot account for the spatially varying metallic flakes.

Recently, high-quality simulation models for rendering of glints and metal surfaces were presented by Yan et al. [YHJ+14, YHMR16] and Jakob et al. [JHY+14]. These approaches deliver good results, but are computationally intensive and thus not ideal for real-time applications. Related is also the publication by Raymond et al. [RGB16], who render scratched metal surfaces, but no metallic paints. An approach for rendering metallic flakes was presented by Atanasov and Koylazov [AK16]. It is not real-time capable, however.

3.2.2 BTF Synthesis and Interpolation

The following publications all present methods for the synthesis of BTFs that look similar to a given – usually small – sample, but are parameterized on different – usually larger – surfaces. Tong et al. [TZL+02] described a method for BTF synthesis using so-called textons. They generate BTFs that look similar to samples from a database, but can be parameterized on arbitrary surfaces. Kawasaki et al. [KSOF05] presented a patch-based method for the same application. Meseth et al. [MMK03] presented a method for real-time rendering of BTFs, which relies on synthesis of similar-looking BTFs from small samples. Zhou et al. [ZDW+05] provided a method that synthesizes arbitrarily sized BTFs using a graph cut based algorithm. In a successive step, they add additional detail – imperfections, like scratches – to the BTF and generate seamless transitions. Haindl and Filip [HF04] described a probabilistic approach for BTF synthesis, which also provides good compression. Later, the same authors [HF07] presented a different method for strong compression and synthesis of BTFs, that look similar to the original data. Haindl et al. [HHCD05] presented a patch-based method for synthesizing BTFs. Liu et al. [LHZ+04] approximate a BTF sample by 4D point appearance functions. These, combined with 2D geometry maps can then be used for synthesis and real-time rendering. A good overview of BTF acquisition, synthesis and rendering was given by Müller et al [MMS+05].

Müller et al. [MSK07] presented a procedural method for editing BTFs. Their approach seems to be suited best for materials that exhibit a relatively coarse geometry like leather or corduroy. Kautz et al. [KBD07] presented an interactive approach for editing BTFs, which is well suited for materials like cloth, but not for the highly specular automotive paints. Ruiters et al. [RSK13] described a method for interpolating BTFs of different materials. Although their results look good, they report runtimes in the range of hours and require some manual markups. The major problem of these approaches is the correct interpolation of textures, which is hard to solve.

3.2.3 Optimal Transport in Computer Graphics

Optimal Transport problems and algorithms solving them have widespread use in many fields. We will only focus on applications in computer graphics. One of the best-known applications of optimal transport is the Wasserstein metric, also known as Earth Mover's Distance. Some of the best-known research in this context was published by Rubner et al. [RTG98, RTG00, RT01], who showed its usefulness for measuring the similarity of images. The earth mover's distance is frequently used in the computer vision area, as shown by Levina and Bickel [LB01], Ren et al. [RYZ11], Ling and Okada [LO07], Graumann and Darrell[GD04] and Pele and Werman [PW09].

Another widespread use of optimal transport, which is also closely related to our method, is color transfer for 2D images. Examples for this are the publications by Pitié et al. [PKD07], Rabin and Peyré [RP11], Ferradans et al. [FPPA14], Rabin et al. [RFP14], Frigo et al. [FSDH14] and Chizat et al. [CPSV16]. Rabin et al. [RPDB11] showed an application of optimal transport for 2D texture interpolation. Most closely related to our work are the publications by Bonneel et al.[BVDPPH11] and Solomon et al.[SDGP+15], who both employed optimal transport for BRDF interpolation. Bonneel et al.[BVDPPH11] described a method of mass transportation for BRDF interpolation. Solomon et al.[SDGP+15] employed convolutional Wasserstein distances for faster convergence and showed that their framework can also be used for BRDF interpolation.

3.3 The Statistical Car Paint Model

Our representation is based on the statistical car paint model presented by Golla and Klein [GK17], which itself is based on the car paint model defined in the AxF file format [ML15], which we will explain first.

3.3.1 The Basic Car Paint Model

The basic car paint model we use, which is also used in the AxF format [ML15], consists of a combination of multiple models. The model for a specific paint can be obtained by measurements, where the analytical parts are fitted to the measurements. The model consists of these components:

- A clear cloat layer, that changes incoming and outgoing directions $\mathbf{i}, \mathbf{o}$ to $\bar{\mathbf{i}}, \bar{\mathbf{o}}$, depending on the thickness and refractive index of this layer.
- A Lambertian BRDF $\frac{a}{\pi}$
- A multi-lobe Cook-Torrance BRDF [CT82] for the brightness, where the k-th lobe is defined as:

$$f^{CT}_{s_k,\alpha_k,F_{0,k}}\left(\bar{\mathbf{i}},\bar{\mathbf{o}}\right) = \frac{s_k}{\pi}\frac{D_{\alpha_k}(\bar{\mathbf{h}})F_{F_{0,k}}\left(\bar{\mathbf{h}},\bar{\mathbf{o}}\right)G\left(\bar{\mathbf{i}},\bar{\mathbf{o}}\right)}{\bar{\mathbf{i}}_z\,\bar{\mathbf{o}}_z}, \tag{3.1}$$

where $\bar{\mathbf{h}}$ is the half vector, s_k is the specular coefficient,

$$D_{\alpha_k}(\bar{\mathbf{h}}) = \frac{1}{\alpha_k^2\bar{\mathbf{h}}_z^4}e^{\frac{\bar{\mathbf{h}}_z^2-1}{\bar{\mathbf{h}}_z^2\alpha_k^2}} \tag{3.2}$$

is the microfacet distribution,

$$F_{F_{0,k}}(\bar{\mathbf{h}},\bar{\mathbf{o}}) = F_{0,k} + (1-F_{0,k})(1-\bar{\mathbf{h}}\cdot\bar{\mathbf{o}})^5, \tag{3.3}$$

where

$$F_{0,k} = \left(\frac{n_{1,k}-n_{2,k}}{n_{1,k}+n_{2,k}}\right)^2 \tag{3.4}$$

is Schlick's approximation [Sch94] of the Fresnel term, $n_{1,k}, n_{2,k}$ are the refractive indices and

$$G(\bar{\mathbf{i}},\bar{\mathbf{o}}) = \min\left(1, \frac{2\bar{\mathbf{h}}_z\bar{\mathbf{o}}_z}{\bar{\mathbf{h}}\cdot\bar{\mathbf{o}}}, \frac{2\bar{\mathbf{h}}_z\bar{\mathbf{i}}_z}{\bar{\mathbf{h}}\cdot\bar{\mathbf{o}}}\right) \tag{3.5}$$

is the geometry term, where

$$\mathbf{y}_z = \mathbf{y}\cdot\mathbf{n}, \mathbf{y}\in\{\bar{\mathbf{h}},\bar{\mathbf{i}},\bar{\mathbf{o}}\} \tag{3.6}$$

denotes the dot product of $\mathbf{y}$ with the surface normal.

- A 2D color table $\chi(\theta_{\bar{\mathbf{h}}}, \theta_{\bar{\mathbf{i}}})$, that is used to represent large-scale color shifts, observed in pearlescent paints. It is parametrized by the angles $\theta_{\bar{\mathbf{h}}}$ and $\theta_{\bar{\mathbf{i}}}$, where $\theta_{\bar{\mathbf{h}}} = \arccos(\bar{\mathbf{h}}_z)$ is the angle between half vector and normal and $\theta_{\bar{\mathbf{i}}} = \arccos(\bar{\mathbf{h}} \cdot \bar{\mathbf{i}})$ is the angle between half vector and incoming direction.
- A BTF representing the sparkling effects caused by the metallic flakes. It is parameterized by $\theta_{\bar{\mathbf{h}}}$, $\theta_{\bar{\mathbf{i}}}$ and $\mathbf{x} \in \mathbb{R}^2$, the position on the surface, i.e. it is a 4D table, denoted as $\Xi(\mathbf{x}, \theta_{\bar{\mathbf{h}}}, \theta_{\bar{\mathbf{i}}})$. An angular sampling of 24-30 samples along each direction is usually chosen. This is according to research by Rump et al. [RMS+08], who observed that the angular lifetime of a metallic flake is around 6-7 degrees. Each combination of $\theta_{\bar{\mathbf{h}}}$ and $\theta_{\bar{\mathbf{i}}}$ results in a 2D texture. For clarity, we will call this function *flake BTF* in the following.

The complete model is:

$$f(\mathbf{x}, \bar{\mathbf{i}}, \bar{\mathbf{o}}) = \chi(\theta_{\bar{\mathbf{h}}}, \theta_{\bar{\mathbf{i}}}) \left(\frac{a}{\pi} + \sum_{k=1}^{K} f^{CT}_{s_k, \alpha_k, F_{0,k}} \left(\bar{\mathbf{i}}, \bar{\mathbf{o}} \right) \right) + \Xi(\mathbf{x}, \theta_{\bar{\mathbf{h}}}, \theta_{\bar{\mathbf{i}}}) \tag{3.7}$$

According to the literature [ML15, GCG+05], good results can be achieved with three lobes.

3.3.2 The Statistical Model

Golla and Klein [GK17] replaced the memory-intensive flake BTF with a statistical representation for compression purposes. We will explain their flake model in the following. They keep the basic discrete parametrization of the flake BTF by angles $\theta_{\bar{\mathbf{h}}}$ and $\theta_{\bar{\mathbf{i}}}$. Each discretized angle combination yields a 2D texture, which we will call *flake slice*. Their approach is based on representing each slice by a number of cuboid-shaped *color clusters*. Within each cluster, they assume a uniform distribution of colors. Each of these clusters has a probability p_i, where $\sum p_i = 1$. In order to compute these color clusters, they employ an algorithm based on reducing an octree in color space to a pre-defined number of leave nodes n. The latter is the only parameter in their method, and thus in ours. They suggest using $n = 50$, which we adopt.

In order to render metallic car paints in real-time from this representation, they provide a GPU-friendly representation. Each flake slice is represented by one row of two textures. They represent the color clusters by storing two opposing corners of the cluster cuboid of each cluster in a common texture. For representing the probabilities p_i, they compute the discrete cumulative distribution function F and store its kth value in the kth pixel of a single channel texture row, i.e. the texture row's kth pixel value equals $\sum_{i=1}^{k} p_i$. They do this to obtain the best possible compression ratio. This however forces their algorithm to read the complete texture row for each reconstruction. Since our focus is on speed, we slightly modify their approach by storing F^{-1}. This forces us to use a different discretization. Instead of the original 50 discretization steps for F, we use 500 steps for F^{-1}. The overall texture size remains comparatively small, such that this does not pose an issue memory-wise. For real-time reconstruction, random numbers are required. In order to ensure frame-to-frame coherence they generate a texture of pseudo-random numbers.

Real-time reconstruction is then performed on the GPU in the fragment/pixel shader. First, the discrete angle combination $\theta_{\bar{\mathbf{h}}}$ and $\theta_{\bar{\mathbf{i}}}$ is computed for the current pixel to be shaded. The random value texture is read for the current u, v position, which yields a random value. Different from Golla and Klein's approach, the random value is identical to the u coordinate in the inverse CDF texture. The inverse CDF texture value is the index of the color cluster to be used. We read the appropriate cluster cuboid corner values from the respective texture. Like Golla and Klein, we then read additional random values from the random value texture by using fixed offsets. Using these values, we generate a color value in the color cluster.

3.4 Interpolation Method

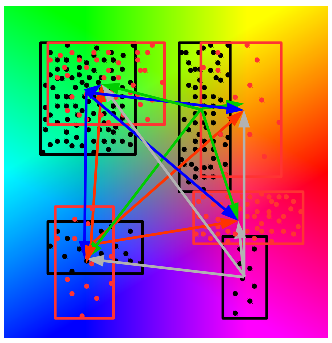
For synthesis, we wish to compute inter- and extrapolations of measured metallic paints. For simplicity, we first consider two metallic paints, where we regard one paint as the source paint and the other as target paint. In order to be able to interpolate the flake BTF responsible for the metallic sparkling effect, we preprocess it, which is explained in the following. The other components of the model require no preprocessing.

3.4.1 Preprocessing for the Interpolation of the Flake BTFs

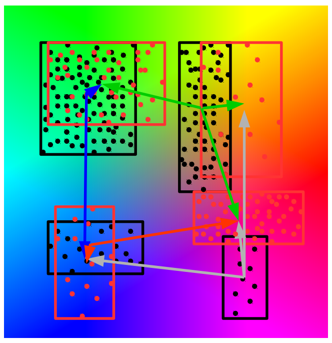
First, the statistical model according to Golla and Klein [GK17] is generated for both metallic flake BTFs. Based on these representations, we set up an optimal transport problem. This idea is related to the Wasserstein metric[Was69, Dob70], also known as Earth Mover's Distance[RTG00, RT01]. Note that all computations are done in the Lab color space, where distances and interpolations behave in a perceptually plausible way. The interpolation has to be performed for all flake BTF texture slices, i.e. separate textures of the two metallic paint BTFs. We assume an identical angle discretization for both BTFs. The interpolation is then performed on each matching pair of BTF texture slices of the two paints. In the following we will consider one flake BTF slice pair of a source paint and a target paint. The process is repeated analogously for each slice pair.

For each BTF slice, we compute the centers of the color cluster cuboids. We assign each cluster's probability to the respective center. The problem is now to transport the probabilities of the source BTF slice to those of the respective target BTF slice. The transportation cost is given by the Euclidean distance of the centers in the Lab color space. It can be visualized as a bipartite graph where each center of the source BTF slice is connected to each center of the target BTF slice – see Figure 3.2. The edge cost is the Euclidean distance. This can be formulated as a linear programming problem. The number of variables is equal to the number of edges, which is $n_s \cdot n_t$, where n_s, n_t is the number of color clusters of the source, respectively target BTF slice. Note that n_s does not have to be equal to n_t. Let $p_i, i \in \{1, \dots, n_s\}$ be the probabilities of the source BTF slice color clusters, $q_j, j \in \{1, \dots, n_t\}$ the probabilities of the target BTF slice color clusters. Let $\|\cdot\|$ be the Euclidean norm and $c_i, d_j, i \in \{1, \dots, n_s\}, j \in \{1, \dots, n_t\}$ be the source, respectively target, BTF slice color clusters' centers in the Lab color space. Let

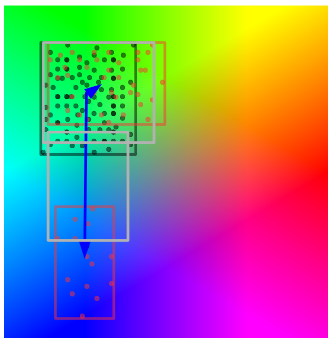
$$w_{ij} = \|c_i - d_j\|, i \in \{1, \dots, n_s\}, j \in \{1, \dots, n_t\} \tag{3.8}$$



(a) *Schematic of the setup of the transportation transport problem. Black: Color clusters of the source car paint. Red: Color clusters of the target paint. Each cluster of the source material is connected to each cluster of the target material. The distances of the cluster centers in color space are the transportations costs. Not shown: Each cluster has a probability relative to the number of color data points it encompasses.*



(b) *A solution to the optimal transport problem. The arrows indicate that some of the probability of the cluster is transported to the respective target cluster.*



(c) *For each transport path used, a copy of the source color cluster's bounding box is generated – shown in gray. During real-time interpolation, the boxes are transported and transformed through the color space, until the respective target cluster's bounding boxes positions and shapes are reached at 100% interpolation. A 50% interpolation for only the top left cluster is shown.*

Figure 3.2: *Our interpolation method for metallic paints is based on solving an optimal transport problem. After clustering the colors present in the car paint BTF, we set up an optimal transport problem and use its solution for interpolating between the paints. Schematics of the involved steps are given in the subfigures.*

be the Euclidean distances of the color clusters' centers.

The optimization problem's objective function, which is to be minimized, is:

$$f(\boldsymbol{x}) = \sum_{i=1}^{n_s} \sum_{j=1}^{n_t} w_{ij} x_{ij}, \tag{3.9}$$

where $\boldsymbol{x} = (x_{ij}) \in \mathbb{R}^{n_s \cdot n_t}$ under the constraints:

$$x_{ij} >= 0 \; \forall i \in \{1, \ldots, n_s\} \; \forall j \in \{1, \ldots, n_t\} \tag{3.10}$$

$$\sum_{i=1}^{n_s} x_{ij} = q_j \; \forall j \in \{1, \ldots, n_t\} \tag{3.11}$$

$$\sum_{j=1}^{n_t} x_{ij} = p_i \; \forall i \in \{1, \ldots, n_s\} \tag{3.12}$$

The constraints also imply that $x_{ij} \in [0, 1]$ and

$$\sum_{i=1}^{n_s} \sum_{j=1}^{n_t} x_{ij} = 1. \tag{3.13}$$

These properties implicate that the x_{ij} can be used as a probability measure for some countable collection $\{E_{ij}\}$. The solution can be computed by one of the existing solvers for linear programming problems – we used the Cbc solver from the COIN-OR project[LH03].

Using this information, we generate a new representation which allows interpolation between the two original BTF slices. We will now explain how to generate the collection $\{E_{ij}\}$. Let $C_i, i \in \{1, \ldots, n_s\}$, be the source and $D_j, j \in \{1, \ldots, n_t\}$, the target BTF slice color clusters. For each $x_{ij} > 0$, we create copies of the source BTF slice's ith color cluster, as well as of the target BTF slice's jth color cluster, i.e. $E_{ij} = (C_i, D_j)$. The tuple (x_{ij}, E_{ij}) then represents an *interpolatable color cluster* of our new *interpolatable* material. In theory, we could also generate the tuples (x_{ij}, E_{ij}) for $x_{ij} = 0$, but they would never be used for rendering and thus be useless. The number n_u of non-zero x_{ij} is at most $n_s + n_t - 1$[Flo53, BVDPPH11].

3.4.2 Real-Time Interpolation

The coefficients of the analytical parts of the original paints can be inter- and extrapolated in a straightforward way. For the inter- and extrapolation of the diffuse color lookup table, we convert its entries to the Lab color space and interpolate linearly in this space.

To generate a *realization* of the interpolatable material of two metallic paints, we choose interpolation values $\alpha, \beta \in [0,1], \alpha + \beta = 1$. Besides the interpolation of the analytical parameters, we have to compute the flake BTF for this realization. Let us consider one interpolatable color cluster $(x_{ij}, E_{ij}) = (x_{ij}, (C_i, D_j))$. The probability x_{ij} was already computed appropriately in the preprocessing step to be valid for all realizations. We only need to compute an interpolated color cluster. For this, we consider two opposing corners g_i and g'_i of the cuboid representing C_i and the matching opposing corners h_j and h'_j of D_j. Without loss of generality, g_i and h_j are chosen with the smallest possible coordinates and g'_i and h'_j with the largest possible coordinates. The respective corners l_{ij}, l'_{ij} of the realization of the interpolatable color cluster are then computed as

$$l_{ij} = \alpha g_i + \beta h_j \tag{3.14}$$

and

$$l'_{ij} = \alpha g'_i + \beta h'_j. \tag{3.15}$$

This can be computed in real-time, on the CPU or the GPU, e.g. in a compute shader. In both cases, only the texture representing the cluster corners has to be updated, while the other properties of the flake BTF remain constant. The analytical parameters and the color lookup table have to be updated as well, which also require only a small amount of memory. Thus the necessary bandwidth is relatively low. Computing on the GPU is faster and has only the disadvantage of using slightly more memory, because all parameters have to be kept in the video RAM.

By dropping the requirement $\alpha, \beta \in [0,1]$, extrapolations can be generated. Dropping the requirement $\alpha + \beta = 1$ is also possible, however the results are less intuitive.

3.4.3 Interpolation of Multiple Materials

The approach can be generalized to an arbitrary number of metallic paints. While it would be possible to set up a transportation problem for many paints, this is not recommended, because the number of variables in the linear programming problem strongly increases. Let n_s, n_t, n_o be the number of color clusters of matching flake BTF slices of the three metallic paints and $F_l, l \in \{1 \dots n_o\}$ the color clusters of the third paint's BTF slice. The number of variables in the objective function would be $n_s \cdot n_t \cdot n_o$. We therefore choose a different approach: We iteratively solve transportation problems. We start with two original materials and generate their resulting interpolatable material. From this material's realization with $\alpha = \beta = 0.5$, we compute the interpolatable material with a new paint. Let us denote the first interpolatable material's probabilities $x_{ij} \neq 0$ by $x_k, k \in \{1, \dots, n_u\}$ and $k \leftrightarrow (i,j)$ for the respective i, j. As mentioned, in practice $n_u << n_s \cdot n_t$. Let $r_l, l \in \{1 \dots n_o\}$ be the probabilities of the third paint. The second optimization problem can then be set up analogously to the first with variables $y_{kl}, k \in \{1, \dots, n_u\}, l \in \{1 \dots n_o\}$. The new interpolatable material will then have color clusters that are represented as tuples $(y_{kl}, (C_i, D_j, F_l)), k \leftrightarrow (i,j)$. A realization will be computed as a linear combination of the three clusters and the analytical parameters with interpolation parameters $\alpha, \beta, \gamma \in [0,1], \alpha + \beta + \gamma = 1$. Again, extrapolations are possible. This process can be repeated with every additional paint.

Note that the number of color clusters and thus the memory usage typically increases with each additional paint added to the material. For practical applications however, we consider three to five paints being the maximum number of paints being intuitively usable. Since the statistical model and thus the interpolatable version are very memory efficient, this does not pose a problem in practice.

While solving the transportation problem is computationally somewhat intensive, it has to be done only once for each additional new paint and thus is considered a preprocessing step. In our implementation, the whole process of loading the original paints and computing the interpolatable material typically took around 50 seconds. The actual inter- and extrapolation, i.e. material synthesis can be done in real-time.

3.4.4 Separate Interpolation of the Flake Intensity and the Color

By introducing smaller modifications to the method, we can create further useful applications. The first is to only interpolate the lightness color channel of the flake BTF, but leaving the hue channels as they are and also leaving all other parameters at those of the source paint. This makes the flakes sparkling intensity look more like that of the target paint, while keeping the color hue of the source paint. See Figure 3.3 for an example. The "dual" operation is also possible: One can keep the flake BTF's lightness information while interpolating its hue channels and the other paint parameters. This results in paints with a new color impression while keeping the flake sparkling intensity. See Figure 3.4 for an example. Note that some results can be achieved with both techniques. Another interesting experiment is to remove the color, i.e. a and b color channel information from the flake BTF. The result looks slightly less convincing as a metallic paint, because it has a very smooth look. See Figure 3.5. This also shows that the color (Lab ab) information in the flake BTF contributes to the overall look.

3.5 Results

All if our experiments were performed on a standard PC with an Intel Core i7-4930K CPU, 64 GB RAM and an Nvidia Geforce Titan first generation graphics card. Including reading and writing to disk, the computation of an interpolatable material from two paints took 50 seconds in our unoptimized Python implementation. Interpolation of three paints took 103 seconds.

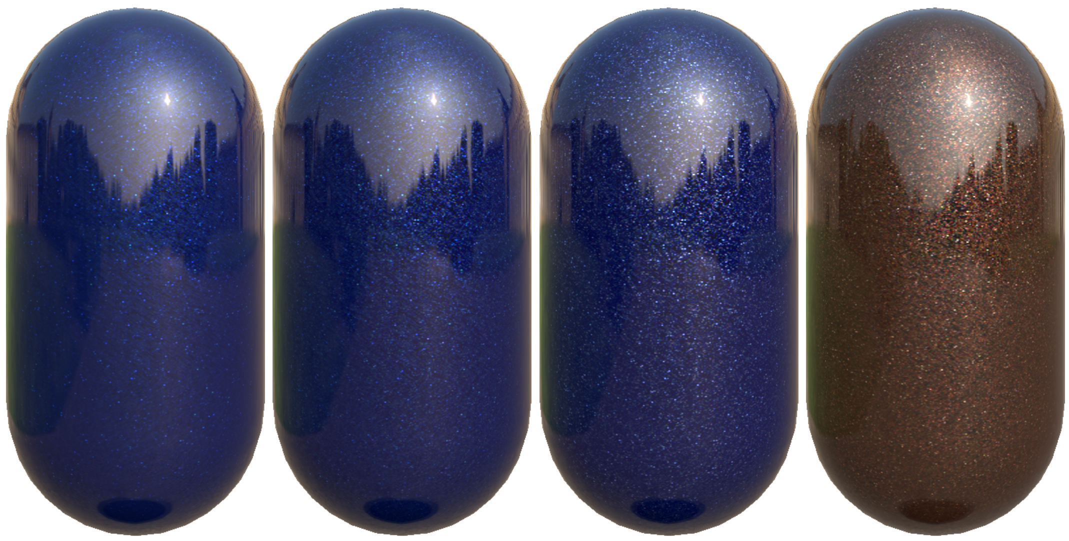


Figure 3.3: *Manipulating the metallic sparkling intensity by interpolation of only the flake BTF's lightness. From left to right: Original blue paint, blue paint with flake BTF lightness interpolated half-way between blue and brown paint, blue paint with flake lightness set to match the brown paint's flake BTF lightness, original brown paint. The basic color impression of the interpolated material remains blue, while the metallic sparkling intensity increases, matching that of the brown paint.*

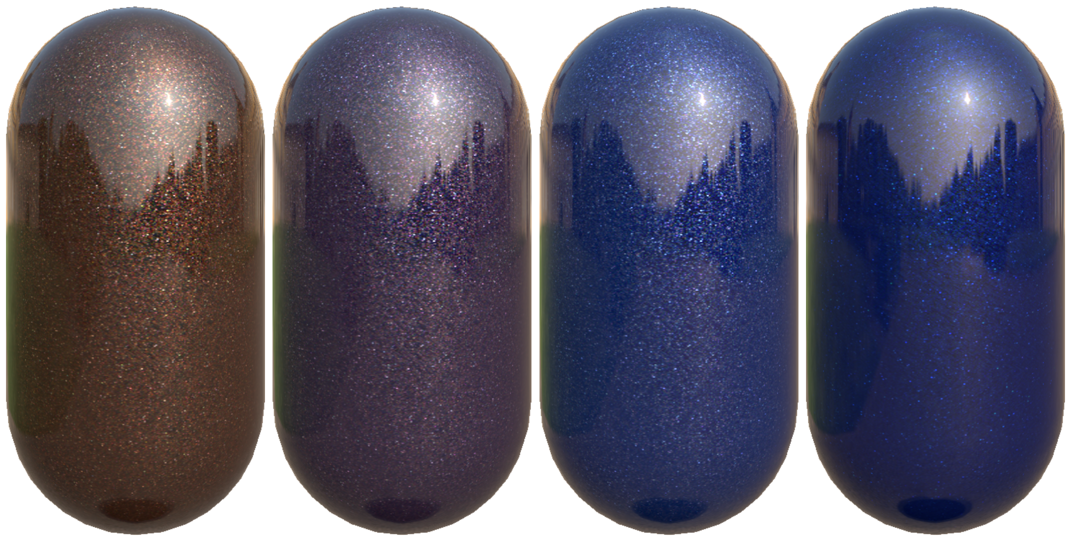


Figure 3.4: *Partial Interpolation. Left to right: Original brown paint, brown paint with all parameters except the flake BTF's lightness interpolated half-way between the brown and the blue paint's, brown paint with all parameters except the flake BTF's lightness interpolated to the blue paint's, original blue paint*

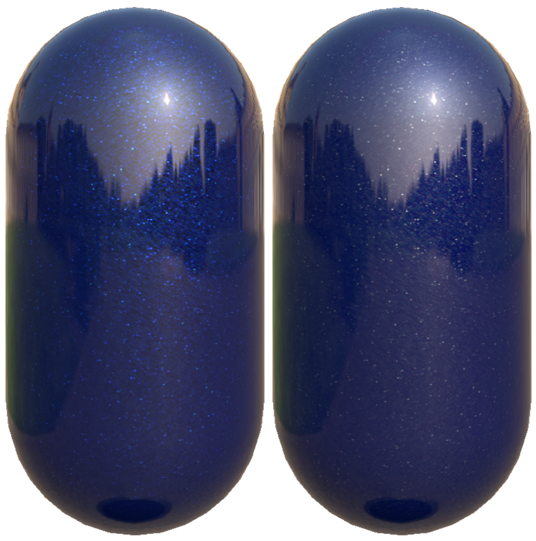


Figure 3.5: *Left: Original blue car paint material. Right: Blue paint with the flake BTF's color hue information set to white. This yields a blue material with white sparkles. However, it looks less convincing as a metallic paint, because it looks too smooth.*

Figure 3.6 shows several different interpolated materials. In each row, the paints in the left and right column are the original measured paints, paints in between are interpolated between them. The green-blue paint in the bottom picture is a flip-flop paint, which is also correctly interpolated. The flip-flop effect gets weaker the more one goes towards the gray paint.

We devised an interactive demo with three capsule models with original measured materials (blue, brown and gray), one capsule with a material that is interpolatable between the other three and a car with the same interpolatable material. The user can move the capsule with the interpolatable material and this material will be set according to the distance of this capsule to the other three capsules. That is, if the capsule with the interpolatable material is e.g. moved close to the blue capsule, it will be mostly blue. This demo is shown in Figure 3.7. This demo was running with 105 frames per second, when the materials remained constant. It dropped to 28 frames per second when the user started moving the capsules, i.e. the materials had to be interpolated. We are thus able to maintain real-time speeds even on slightly outdated hardware. The interpolation was performed on the CPU. We expect the frame rate to be

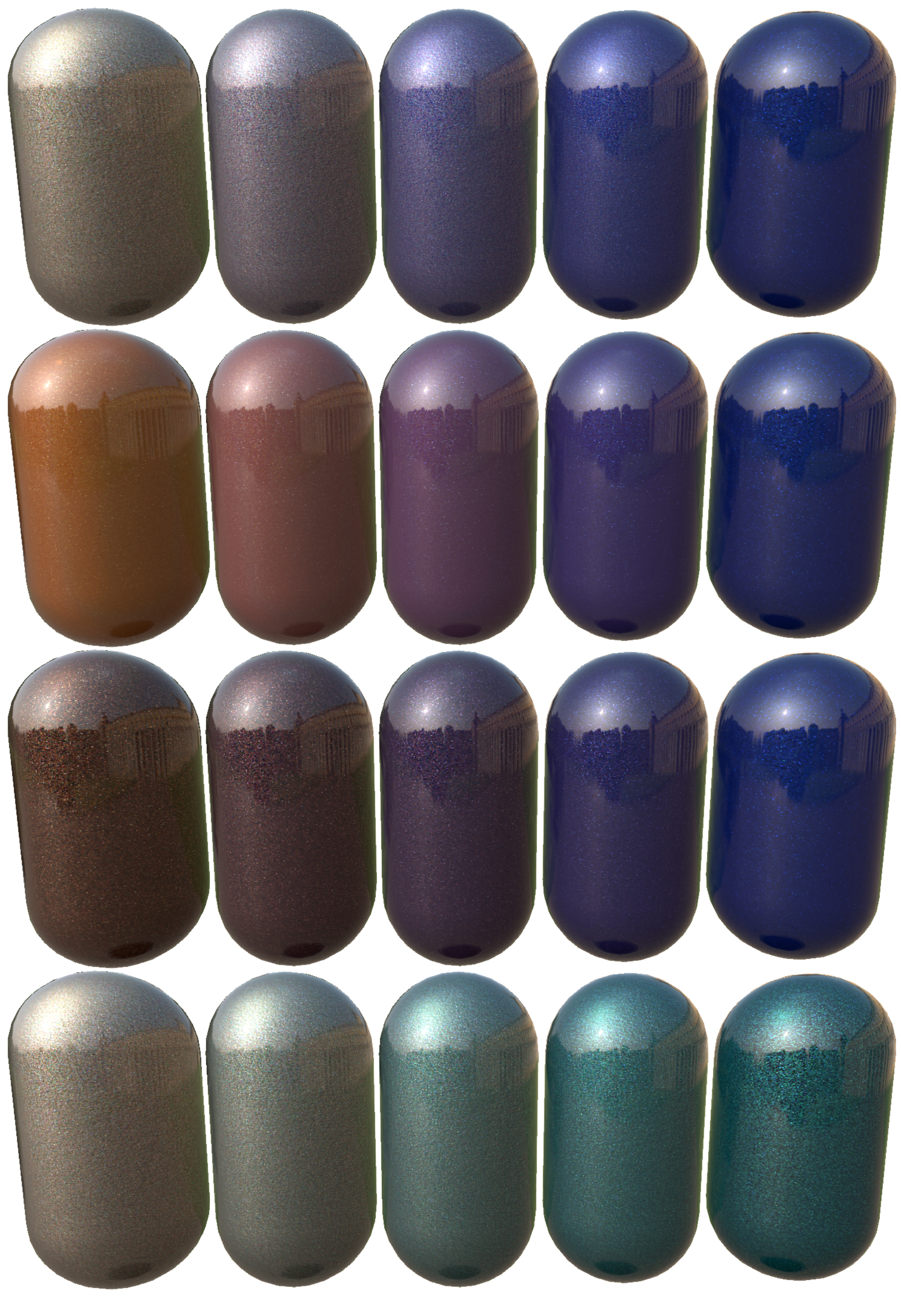


Figure 3.6: *Several interpolations: In each row, the capsules in the left and right column are have been assigned the original measured paints, capsules have been assigned materials interpolated between the respective measured materials. The green-blue paint in the bottom picture is a flip-flop paint, which is also correctly interpolated. The flip-flop effect gets weaker the more one goes towards the gray paint.*

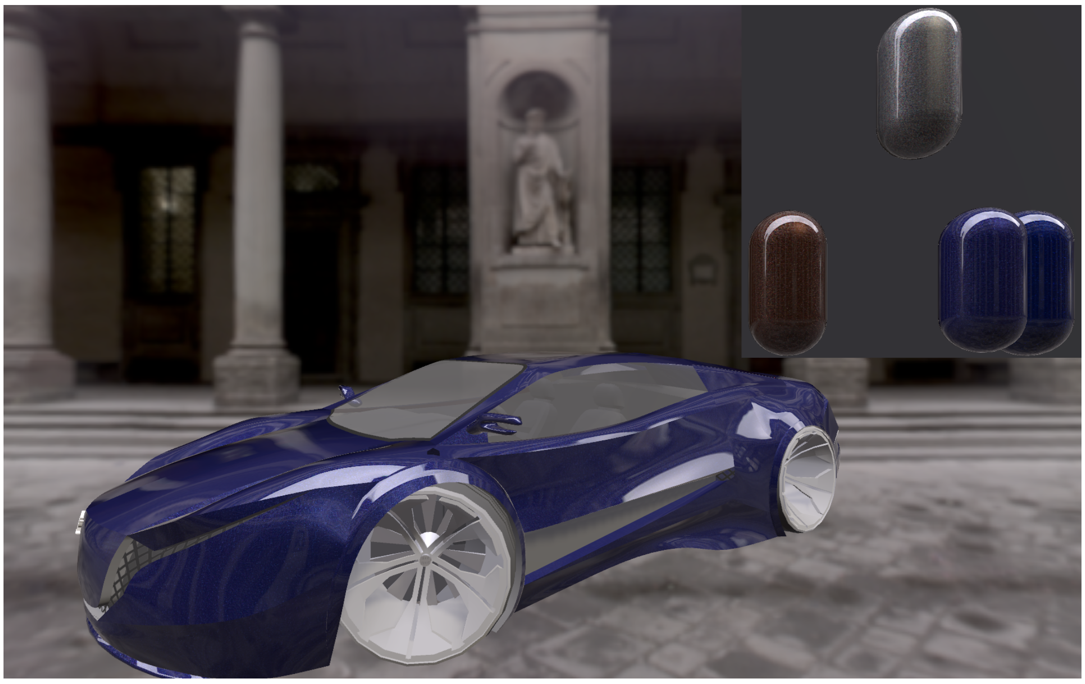
even higher when the interpolation is performed on the GPU. The total VRAM requirement of this demo was 131 MB.

Figure 3.3 shows an example where only the flake BTF lightness is interpolated, but the paint's basic color remains that of the source paint. On the left, the original blue metallic paint is shown, which has flakes of only a low sparkling intensity. On the right, the original brown paint is shown, which has more intensely sparkling flakes. The second capsule from the left consists of a 50 % interpolation of the BTF lightness of the two original paints. Visually its sparkling intensity is between the original paints. The material of the third capsule from the left has the blue base color and also the bluish tint of the flakes, but the sparkling intensity of the brown paint. It looks like a version of the blue paint with more and brighter flakes.

Figure 3.4 shows the orthogonal operation: Starting with the brown paint on the right, we interpolate its base color and the flake BTF hue, but keep the flakes BTF lightness. The material of the second capsule from the right is a 50 % interpolation. It has a violet color, which is between blue and brown in the Lab color space. Its sparkling intensity remains that of the brown paint. The second paint from the left is the brown paint, with its hue fully interpolated to blue. Note that the result looks very similar to the third paint from the left in Figure 3.3, where the orthogonal operation, starting from the blue paint was performed. The leftmost paint is again the original blue metallic paint.

When only small manipulations are desired, it is also possible to just manipulate one paint. Figure 3.5 shows the blue paint with the flake representation's color hue set to white. This yields a blue material with white sparkles. However, it looks less convincing as a metallic paint, because it looks too smooth.

Figure 3.8 shows materials generated by extrapolation. The measured brown and blue metallic paints are shown in Figure 3.8b. The result from extrapolating in 25% steps are shown in Figures 3.8a and 3.8c. The results are yellow-brown (3.8a) and intensely blue materials (3.8c).



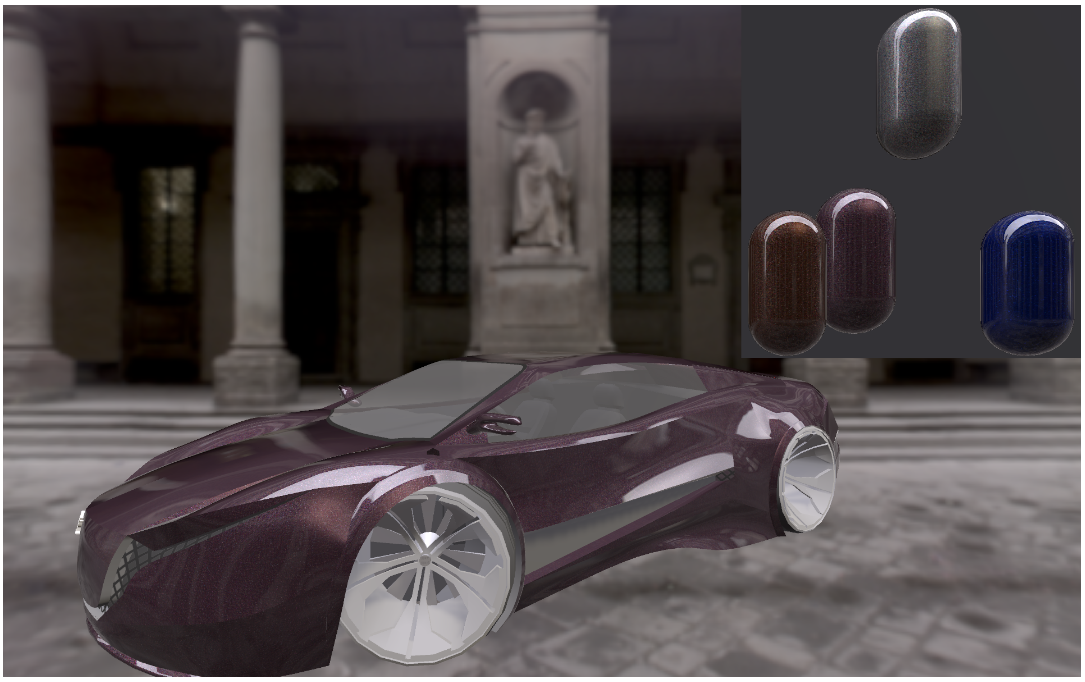
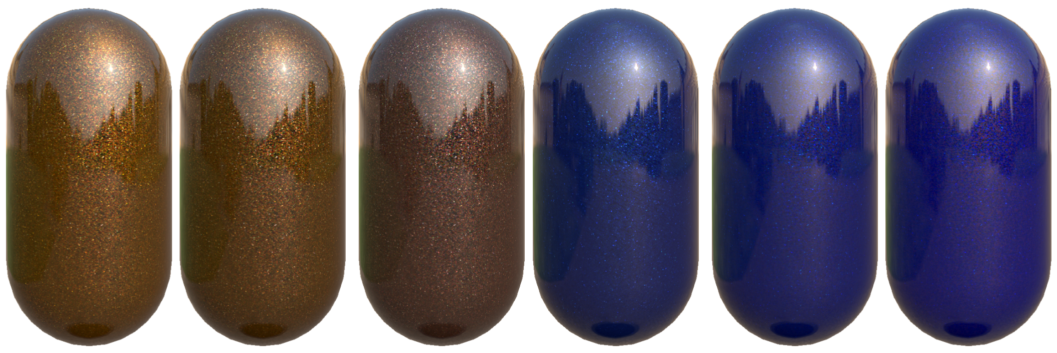


Figure 3.7: *Example Application: The capsules can be moved. The central capsule's material is interpolated from the other three capsules' materials, depending on the distance to them. The car has the same interpolated material. This way the user can generate a desired material in real-time. See also the accompanying video.*



(a) *Extrapolated* (b) *Measured* (c) *Extrapolated*

Figure 3.8: *Our approach can also be used to generate extrapolated versions of the metallic paints. The two paints displayed in the center (b) are the original paints. To the left (a) and right (b) are extrapolations in 25% steps. The respective interpolations are shown in Figure 3.1b.*

3.6 Conclusion

We presented a method that allows to generate realistic complex metallic car paint materials in real-time, based on interpolating captured real-world car paints. Our method requires no manual parameter tuning, only a short preprocessing step and is intuitive to use. We showed how to separately interpolate the larger-scale reflective properties, including the basic color hue, the local color hue, and the sparkling intensity of the metallic paints' appearances, allowing for even greater artistic freedom. Our method is simple to implement and efficient in terms of memory and computation requirements and thus only has moderate hardware requirements. Using our approach facilitates intuitively exploring the space of possible metallic paints spanned by given real-world measurements. We believe that our method will be useful for designers in the game, car and movie industry.

Acknowledgments

We would like to thank Volkswagen and X-Rite for providing measurements of the metallic paints. Uffizi Gallery Light Probe Image ©1999 Paul Debevec, http://www.debevec.org/Probes/

4 Temporal Upsampling of Point Cloud Sequences by Optimal Transport for Plant Growth Visualization

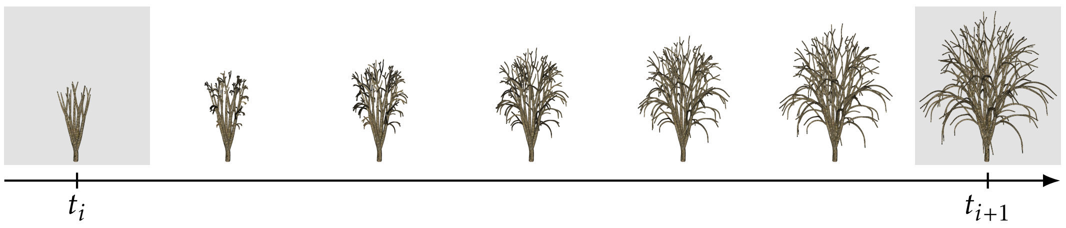


Figure 4.1: *Temporal upsampling of two synthetically generated tree point clouds obtained with our method. The original point clouds for the time steps t_i and t_{i+1} are highlighted with a gray background and the in-between states have been computed based on our method.*

Abstract Plant growth visualization from a series of 3D scanner measurements is a challenging task. Time intervals between successive measurements are typically too large to allow a smooth animation of the growth process. Therefore, obtaining a smooth animation of the plant growth process requires a temporal upsampling of the point cloud sequence in order to obtain approximations of the intermediate states between successive measurements. Additionally, there are suddenly arising structural changes due to the occur-

rence of new plant parts such as new branches or leaves. We present a novel method that addresses these challenges via semantic segmentation and the generation of a segment hierarchy per scan, the matching of the hierarchical representations of successive scans and the segment-wise computation of optimal transport. The transport problems' solutions yield the information required for a realistic temporal upsampling, which is generated in real-time. Thereby, our method does not require shape templates, good correspondences or huge databases of examples. Newly grown and decayed parts of the plant are detected as unmatched segments and are handled by identifying corresponding bifurcation points and introducing virtual segments in the previous, respectively successive time step. Our method allows the generation of realistic upsampled growth animations with moderate computational effort.

This chapter corresponds to the publication Tim Golla, Tom Kneiphof, Heiner Kuhlmann, Michael Weinmann, and Reinhard Klein. Temporal Upsampling of Point Cloud Sequences by Optimal Transport for Plant Growth Visualization. *Computer Graphics Forum*, 39(6):167–179, September 2020 [GKK+20].

4.1 Introduction

Modeling, visualizing and analyzing plant growth is of great relevance in various domains including biology, agriculture and computer graphics with applications in movie and game productions. Therefore, the coupling of respective plant models to measurements of growing plants allows a better understanding regarding the underlying growth processes for individual plant species. While a conventional video recording has shown great potential for plant monitoring, its limitations regarding the flexibility of inspecting plants from arbitrary views and performing 3D measurements can only be overcome by directly considering a complete 3D scan of the plant of interest. However, capturing the plant growth with all its details relies on an (ideally) temporally continuous scanning of the plant state which is hindered by the fact that capturing the plant state at a certain time step usually involves a time-consuming manual acquisition. This induces larger time intervals between successive measurements. As a plant often has grown significantly between successive scans, a smooth animation of the growth process from coarsely taken discrete measurements is not easily possible. Additionally, suddenly arising structural changes induced by bifurcation or the evolution of leaves have to be taken into account when aiming for a smooth completion of the intermediate states between successive measurements. Such topological changes of the underlying ground truth model and the fact that the measurement process inherently does not guarantee perfectly corresponding points or point numbers for successive point cloud scans cannot be handled by trajectory-based deformation and morphing approaches. These require exact point correspondences that can only be reliably obtained for small deformations and motions and, hence, are by design not suitable for analyzing growth processes with changing topology [XSWL15]. Further limitations of previous work on deformations based on exact point correspondences regarding their applicability on growth processes include the assumption of piecewise rigid motion as used for object tracking [BIZ18], the requirement of a 3D object template that is deformed and fitted to the point clouds in adjacent time steps using respective priors (without enforcing temporal coherence) [ZFG+17], the involvement of a visual hull prior that biases the optimization in the context of mesh-based approaches [LLV+12], or the need for large databases required by learning-based methods [WLX+18] that are hard to acquire due to the time-consuming nature of the scanning process and the growth of the plants themselves.

In this paper, we address the aforementioned limitations by a novel method that allows temporal upsampling of coarsely acquired point cloud sequences of growth processes including topological changes without requiring exact point correspondences, shape priors or huge databases. For this purpose, our approach involves an initial point cloud segmentation and the generation of a hierarchy on the segments for each of the individual scans, the matching of the segment hierarchies of successive scans, and the final computation of a segment-wise regularized optimal transport (OT) to morph the point sets associated with matched segments. Newly grown and decayed parts of the plant, that are detected as unmatched segments, are handled by identifying corresponding bifurcation points and introducing virtual segments in the previous, resp. successive time step. Furthermore, we present a heuristic segmentation approach that is particularly suitable for thin structures as occurring for maize plants or tree structures. By using a memory-efficient formulation of the optimal transport problem, our approach is also capable of handling large point clouds. As demonstrated in the evaluation, our approach yields smooth temporal upsamplings, allowing for realistic growth animations with moderate computational effort.

4.2 Related Work

Temporal upsampling for modeling and visualizing growth and deformation processes can be approached based on several strategies such as procedural modeling, template-based approaches, correspondence-based approaches and data-driven approaches. In the following, we will discuss respective developments in the context of plant growth simulation. In addition, we provide a review of applications of optimal transport between data distributions in the context of graphics applications as the use of optimal transport allows our method to overcome the need for perfect one-to-one correspondences or templates.

Procedural modeling for plant growth Modeling structures and phenomena from sets of rules has received a lot of attention in the contexts of facade modeling and plant modeling. However, the underlying rules are often not directly known and have to be derived, for example, using inverse procedural modeling. In the context of modeling plant growth, L-systems have been extended by growth rules such as relational growth grammars [KKBS05] or tree-species-specific growth rules [LKMH05], component-wise time functions of logistic growth [CAJBS05], guiding vectors to provide local control over branch orientation during tree growth [XM15], generic rules to provide sequences of symbols and branching [SBM+10], the localization and tracking of topological events like budding and bifurcation [LPY+12], growth models [SPK+14] or plant modules for capturing species-specific branching pattern characteristics that can be combined to model whole individual plants [MHS+19]. While individual instances can be derived via the use of meaningful parameters and rules, a smooth interpolation between individual objects or process instances is challenging as smooth deformations of plants cannot simply be achieved by interpolating within the parameter space. In addition, inverse procedural growth models may fail to capture the nature of input examples that do not capture all of the relevant growth phenomena and may be limited to certain species.

Template-based and mesh-based interpolation Early work on template-based growth analysis and visualization include the modeling of plant growth in terms of flexible spatio-temporal templates to represent non-rigid motions [LXZ+97]. Regarding mesh morphing, the morphing based on local transformations between source and target meshes such as ARAP [ACOL00] and affine transformations [SZGP05] is not capable of handling large deformations. The latter also applies to morphing based on intrinsic shape information such as differential coordinates [Ale03], linear rotation-invariant (LRI) coordinates [LSLCO05], Laplacian coordinates with invariance to rotation and isotropic scaling [SCOL+04], mean curvature flow Laplacian coordinates [HLW07], gradient fields [XZWB06] or the combination of near-rigid priors with a local gradient field interpolation [CL09]. Further work used spatio-temporal skeleton-based models [ZLLZ12]. Bansal et al. [BT18] introduced a mesh morphing and interpolation framework based on the Lie Bodies representation. For large deformations, the authors perform mesh segmentation and apply the Lie Bodies framework on the respective components. Furthermore, Chen et al. [CFB16] presented a method for the simultaneous morphing between two mesh sequences for blending motions and interpolating shapes. Zhao and Barbic [ZB13] segment plants from the geometric information to establish a structural hierarchy that is connected with a FEM simulation model to allow deformations. However, template-based approaches rely on the availability of 3D object templates that are deformed and fitted to the observations in adjacent time steps as well as adequate priors and larger deformations cannot be handled easily.

Correspondence-based interpolation Instead of exploring templates and shape priors given by the mesh representation, several approaches focused on unstructured representations such as point clouds, where the missing topological information complicates the morphing. This has been addressed based on improving the robustness of correspondences based on local mappings between the projections of the input point clouds onto a common parameter space which can be performed in a segment-wise manner for complex-shaped objects [XZPF04]. Alternatives include the guidance of the morphing by registration and warp functions [THCF06], exploiting differential geometry and continuum mechanics to get physically-meaningful transitions [BGQ05] or the guidance of the interpolation based on gradient fields [TZZ09]. Wang et al.[WZH12] presented a method for morphing point-sampled geometry based on parameterizing the point clouds onto spheres and then comparing features. Solenthaler et al. [SZP07] refine irregularly sampled, dynamic points in the context of particle-based fluid simulation while preserving edges and avoiding point collisions. Further work on temporal upsampling by interpolating between temporally consecutive point cloud measurements includes the use of 4D spatio-temporal models as proposed in the context of large dynamic urban scenes [JBB13]. Lie et al. [LLV+12] incorporate the visual hull as a shape topology prior and surface bending energy for shape completion. Subsequently, inter-frame correspondences are used to obtain temporally coherent dynamics. The approach can handle larger deformations and certain topological changes, however, the topology of the output is tied to the possibly incorrect topology of the visual hull. Li et al. [LFM+13] exploit a forward-backward analysis for spatial and temporal analysis of 4D point clouds for tracking plant parts over time. The co-segmentation framework proposed by Yuan et al. [YLX+16] decomposes articulated point cloud sequences into near-rigid moving parts. Segments derived using trajectory analysis are progressively propagated to neighboring frames and merged through all frames using space-time segment grouping. This results in robustness regarding noise, occlusions as well as variations of pose and view. Similarly, Bertholet et al. [BIZ18] address motion segmentation from RGB-D videos by representing motion as a sequence of rigid transformations through all input frames in an energy optimization framework. Vlachos et al. [VLSM18] focus on the improvement of animation reconstruction based on rank minimization while imposing spatial coherence between successive frame clusters. Furthermore, Xu et al. [XSWL15] approach trajectory-based shape interpolation by solving Poisson equations defined on a domain mesh. Interpolated shapes are reconstructed from interpolated gradient fields that exploit both point coordinates and surface orientations instead of directly using point coordinates. The technique shows stable area and volume changes and overcomes the shrinkage problem of linear shape interpolation. Zheng et al. [ZFG+17] used a template-based dynamic tracking algorithm for the temporal reconstruction of geometry and deformation in point cloud sequences, tailored to the reconstruction of flower petals. The use of specific priors to assist the tracking allows handling occlusions and collision-based interactions at the cost of requiring a suitable template. However, trajectory-based deformation/-morphing approaches require exact point correspondences that can usually only be reliably obtained for small deformations/motions and, hence, are by design not suitable for analyzing growth processes with changing topology due to newly added branches or leaves [XSWL15]. Furthermore, the points in adjacent point clouds may not exactly represent corresponding surface parts as the point distribution is determined by the scanning process. Our approach overcomes these limitations by considering the point distributions instead of the point cloud itself. As a result, we can circumvent the establishment of exact point correspondences that are hard to establish for growing objects with possibly changing topology.

Data-driven and learning-based interpolation Data-driven approaches exploit the availability of large collections of related exemplars to describe the characteristic deviations between objects and, hence, to guide the interpolation between object states, which allows handling large deformations. This has been used in the context of mesh interpolation based on patch-based linear rotation invariant (LRI) coordinates [BVGP09], using an iterative Gauss-Newton method for shape interpolation [FB11], solving a shortest path problem in the local linear subspaces derived from a given model database [GLHH13], using low-dimensional shape spaces that exploit the specific structure of the interpolation problem [VTSSH15] and using rotation/translation-invariant representations that define plausible deformations in a global continuous space [GCLX17]. Furthermore, the approach by Wang et al. [WLX+18] explores a shape space technique that relies on the correspondence computation between trees of different structure and using geodesics to obtain the continuous blending between trees. However, such collections of examples that represent the complete range of possible variations are often not available due to the impractically time-consuming capture processes.

Optimal Transport in Graphics and Vision Optimal transport has been used successfully in various computer graphics and computer vision applications. Early applications include its use for the measurement of the similarity of images by defining the earth mover's distance between color distributions and between texture feature distributions as used by Rubner et al. [RTG98, RTG00]. Furthermore, optimal transport has been successfully used for color transfer in 2D images based on the regularization with the preservation of the gradient of the original picture [PKD07], an approximate variational formulation using an approximate Wasserstein constraint on color statistics and a generic geometric-based regularization [RP11], regularized discrete optimal transport within a unified convex variational framework [FPPA14], relaxed optimal transport using a regularization that considers the spatial distribution of colors [RFP14], by using separate illuminant matching and optimal transfer of dominant colors regularized by thin plate splines [FSDH14] and extending standard entropy regularization based optimal transport to unbalanced problems [CPSV16].

In addition, optimal transport has been applied for interpolation. Respective examples include the interpolation between textures [RPDB11], reflectance models such as bidirectional reflectance distribution functions (BRDFs) [BVDPPH11, SDGP+15] or bidirectional texture functions (BTFs) [GK18] based on statistical representations [GK17] as well as geometric models in terms of volumetric representations [SDGP+15]. Digne et al. [DCSA+14] used optimal transport for surface reconstruction and simplification from point clouds and Merigot et al. [MMT18] investigated optimal transport between a simplex group and a point cloud.

Plant Segmentation The numerous approaches for plant segmentation include the combination of color information and 3D models [ADT11, MBW+18], adapted surface feature-based techniques [PDMK13], model-based approaches based on cylinder representations of stems and separate segmentation tailored to leaves [GDHB17], skeleton-based stem and leaf point recognition approaches [WWX+19], facet region growing approaches after an initial oversegmentation [LCT+18] as well as automated, data-driven approaches for plant structure segmentation based on geometric features and Random Forests classifiers (e.g. [DNC+18]) or geometric features and clustering [WPKM15].

4.3 Temporal Upsampling of Point Cloud Sequences Using Optimal Transport for Plant Growth Visualization

Given a time series of point cloud scans obtained by a 3D scanner, our goal is to produce a temporal upsampling of the plant growth that happened between the individual measurements that results in a realistic, smooth plant growth visualization. As illustrated in Fig. 4.2, our method consists of the following steps: Initially, the scans are coarsely aligned to each other (Section 4.3.1). While it is possible to set up and solve a single optimal transport problem for the whole dataset, the result is not satisfactory, as artifacts like unnatural-looking breaking up of plant parts occur (see Fig. 4.3). In order to alleviate this, we perform the following steps: First, a hierarchical segmentation is derived for each scan individually (Section 4.3.2). Afterward, the corresponding segments of successive scans are matched (Section 4.3.3), which is facilitated by the initial coarse alignment. Finally, for each pair of corresponding segments, a regularized optimal transport problem is set up (Section 4.3.4). Its solution delivers the information required for a temporal upsampling, which is obtained by interpolation.

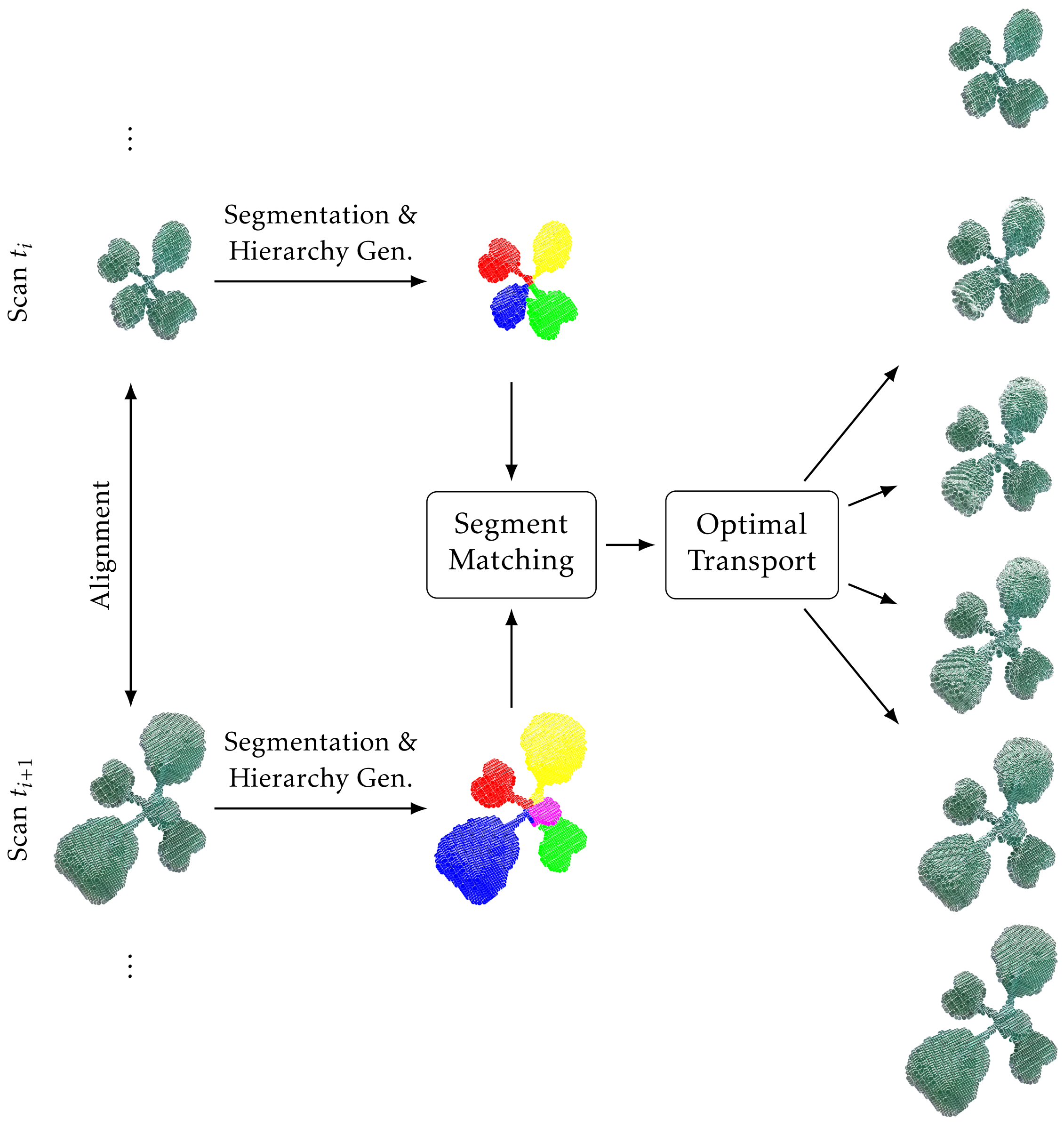


Figure 4.2: *Overview of our method: First, a hierarchical segmentation is generated for each scan, which is followed by a matching of the segments of point clouds obtained for successive measurements at the time steps t_i and t_{i+1} and the computation of segment-wise optimal transport solutions used for temporal upsampling.*

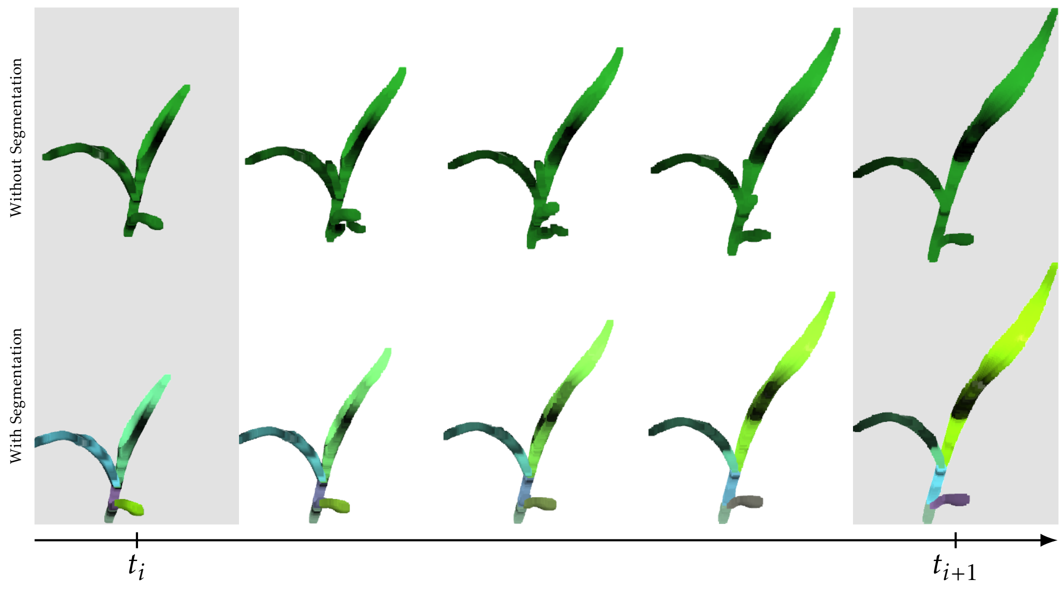


Figure 4.3: *Comparison between animations generated from the same data without segmentation (top row) and with segmentation (bottom row). Images with a gray background correspond to the input scans. Images with a white background represent upsampled data. In the case without segmentation (top row), the plant breaks up in an unrealistic way. Using segmentation (depicted in the bottom row), this problem is solved, because transport is restricted to take place only within a segment.*

4.3.1 Coarse Alignment

Continuously monitoring plant growth with a fixed arrangement of scanners is complicated due to self-occlusions induced by growing plant parts and results in incomplete 3D scans. As a consequence, detailed plant scans require a manual scanning process, ideally with a movable scanner platform that can be guided by a user to obtain complete scans. As this may require the relocation of plants to a certain scanner platform, successive scans have to be aligned in a common coordinate system in order to register the individual plant segments and to allow the consideration of changes induced by the growth process. For this purpose, we align each scan in the time series coarsely via the iterative closest point algorithm [BM92] to its predecessor. We assume that scans in a time series have a consistent scaling.

4.3.2 Hierarchical Segmentation

If used without any further constraints, the later described optimal transport solution will lead to an unrealistic breaking up of the dataset in the upsampling as shown in Fig. 4.3. To avoid this problem, we segment the respective point cloud P_i of each time step t_i into segments $s \in S_{P_i}$, onto which we restrict the optimal transport problem setup. These segments are not necessarily matched. Additionally, new segments might appear due to growth or might disappear due to cropping. In order to be able to better match the segments and handle growth and decay events, we introduce a hierarchy H_{P_i} on the segments S_{P_i} of each point cloud P_i.

Creating a Hierarchy on Existing Segmentations

In principle, different segmentation techniques may be used to derive the segmentation S_{P_i}. However, they should be suitable for the expected plant structures. For a given segmentation of a plant into its semantic entities like stem, branches, leaves or blossom petals, we first identify the root segment of the hierarchy H_{P_i} as the segment containing the point with the lowest vertical coordinate. Alternatively, the respective segment may also be directly chosen by the user.

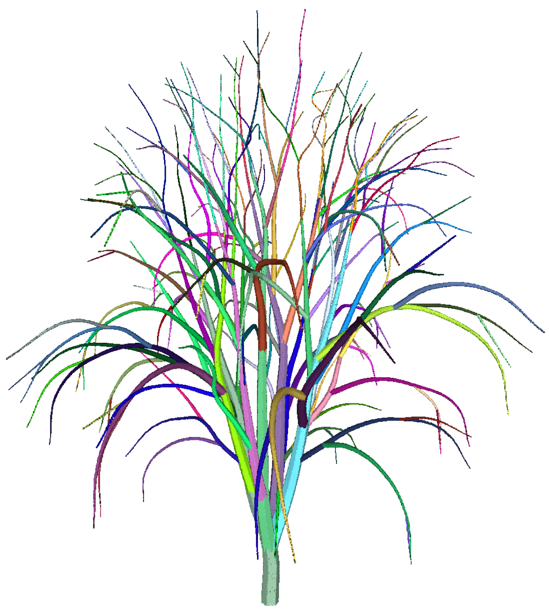
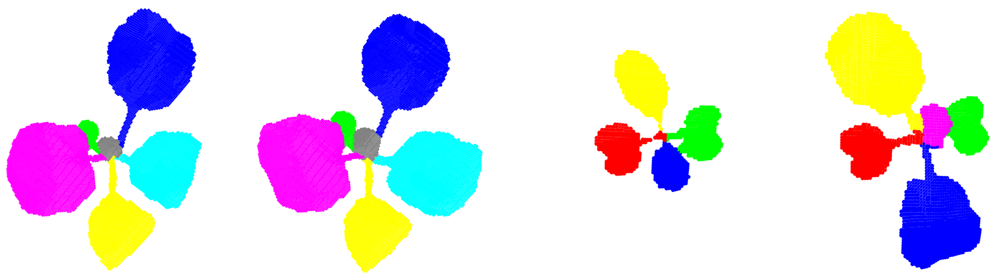


Figure 4.4: *Exemplary segment hierarchy obtained from an initially provided plant segmentation: The artificially generated tree has a natural segmentation. We impose a hierarchy by heuristically or manually selecting a root (the tree trunk) and then performing successive nearest neighbor searches in order to generate a hierarchy on the segments.*

Then, successive nearest neighbor searches are performed to populate the hierarchy H_{P_i}. All segments containing points that are closer than a user-specified threshold to points of the currently considered segment are added as children of this segment in the hierarchy. This is iterated until all segments have been processed. An exemplary decomposition into a segment hierarchy derived for a synthetic tree model is illustrated in Fig. 4.4. Here, the ground truth segmentation of the tree into the stem and individual branches was directly obtained from the synthesis and we chose the trunk as the root segment, which is also the segment with the lowest vertical coordinate. In the case of a given segmentation with no meaningful root segments, such as in the Komatsuna dataset [USM+17], we introduce a virtual, empty root segment and add the other segments as direct successors in the hierarchy H_{P_i}. An example is shown in Fig. 4.5.



(a) *Small Growth* (b) *Significant Growth & New Part*

Figure 4.5: *Labeled point cloud scans from the Komatsuna dataset [USM+17]. (a) shows two successive point cloud scans which are consistently labeled and only a small amount of growth happened. In this case, no further matching of the segmentations is necessary. (b) shows two point clouds where there has been significant growth in the meantime. While consistent labels are assigned to the individual parts, for the newly grown part a small new virtual segment is added to the first point cloud and matched to the unmatched segment of the second point cloud.*

Simultaneous Segmentation and Hierarchy Generation for Thin Structures

For plants with thin structures, we can derive the segment hierarchy H_{P_i} together with the corresponding segmentation S_{P_i} without the requirement of an initial segmentation of a plant into its parts.

First, we build a k-nearest neighbor graph on the point cloud, where k is a user-defined parameter. We found $k = 15$ to deliver good results in our experiments. On this graph, we compute a minimum spanning tree (MST). We assume the point with the smallest value along the vertical axis to be the root point of the MST, thus inducing a direction on the graph. When operating on single plants, this is a reasonable assumption as this point corresponds to the location where the stem exits the ground.

For each node v in the MST, we compute the longest possible path length m_v to a leaf node in the MST subtree rooted at v. Starting with the root, we traverse the MST. If we encounter a node v in the tree where the number of successors $u \in \text{succ}(v)$ with longest path length $m_u > c$ is more than one, for a user-defined threshold c, we found a bifurcation. All of v's successor nodes u with

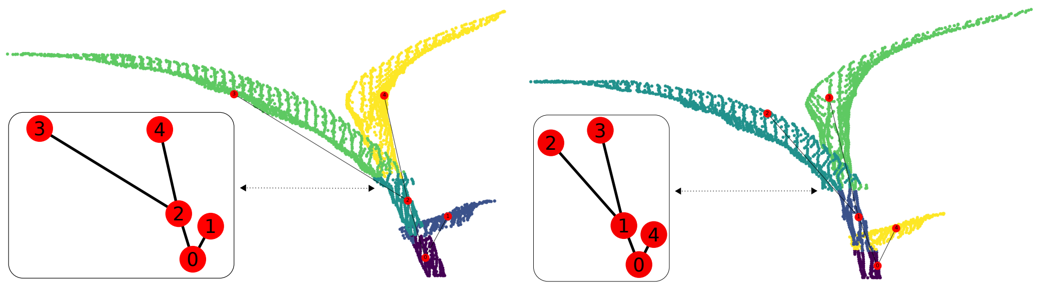


Figure 4.6: *Segmented point clouds of two successive scans of a maize plant. The hierarchy graphs are plotted on top of the plants and shown additionally in the boxes. Although the numberings of the nodes, indicated in terms of the segment colors, differ, our method allows a correct matching of corresponding segments.*

$m_u > c$ are then considered to be roots of trees corresponding to new biological branches. The edges from v to these successors are inserted into a set E of edges to be removed from the MST, which also define the connectivity between created segments. After visiting all nodes, the edges $e \in E$ are removed from the MST and the connected components of the resulting graph, respectively the corresponding 3D points, define the segmentation S_{P_i} on the point cloud P_i. At the same time, a separate graph representing the hierarchy H_{P_i} of the segments in S_{P_i} is created. The root segment is chosen such that it contains the MST root node. Two segments $s_1, s_2 \in S_{P_i}$ are connected in H_{P_i} if an edge $(u,v) \in E$ exists with $u \in s_1$ and $v \in s_2$. An example of the resulting segmentation and the corresponding hierarchy graph is depicted in Fig. 4.6.

4.3.3 Segment Matching

Given segmentations of two point clouds P and Q for successive time steps, we match the segments of P and Q using the hierarchies H_P and H_Q built in the previous step. We differentiate the following cases:

1. The labels are consistent and the number of segments is identical. This is usually the case with manually segmented data, when no new segments have grown. An example is shown in Fig. 4.5(a).
2. The labels are consistent and the number of segments varies. This is usually the case with manually segmented data, when new structures represented by new segments have grown or parts have decayed. An

example is shown in Fig. 4.5(b). For growth scenarios, small new virtual segments are added to the first point cloud P and matched to the otherwise unmatched segments of the second point cloud Q. These virtual segments are used to generate a growth animation. For decay scenarios, we add a point set that represents the decayed leaf. As a heuristic, we generate this point set by moving the segment's points towards the smallest vertical coordinate, which is perceived as the floor. Alternatively, this segment may be simply faded out. We emphasize that the respective choice depends on the preferences of the user.

3. The labels are inconsistent. This can occur if the labeling has been generated automatically with no additional information. In this case, we perform a matching algorithm, which is described in Section 4.3.3. Growth or decay may also occur and is handled analogously to the scenario with a consistent labeling.

Segment Matching with Inconsistent Labeling

If different IDs are assigned to corresponding segments s_P, s_Q in consecutive scans, we use the Wasserstein distance $d_w(s_P, s_Q)$ (see Eq. (4.1)) as a similarity measure between the two segments s_P, s_Q, that is, the cost of warping the point distributions onto each other. Initially, only the root segments of the hierarchy are considered to be matched. We then alternate between merging and matching segments in the hierarchy, which is explained in detail in the following paragraphs.

Merging For each newly created match of the segments $(s_P \in H_P, s_Q \in H_Q)$, we try to merge the segment in the second point cloud s_Q with one of its children $c_Q \in \mathrm{succ}(s_Q)$, as long as this reduces the Wasserstein distance $d_w(s_P, s_Q)$ between the associated point clouds. The rationale behind this step is to adapt to different segment hierarchies induced by growth or decay processes. For example, a new biological branch might have appeared in the later states of the plant growth, which leads to the splitting of the segment of the previous scan at the location where the new part starts growing. This process is repeated until no further improvement is possible.

Matching When no further merging steps are possible, we create a matching of the remaining child segments. Since the scans are segmented individually, the association of corresponding parts between the segmentations has still to be computed in order to be able to set up an optimal transport problem between segment pairs. An example of the segmentations and the associated graphs of two consecutive scans is shown in Fig. 4.6.

In order to resolve this ambiguity, we perform the following matching step: We compute the pairwise Wasserstein distances $d_w(c_P, c_Q)$ between the point clouds associated with the child segments $c_P \in \text{succ}(s_P)$ and $c_Q \in \text{succ}(s_Q)$. The pairwise costs are stored in a cost matrix. The optimal association of the child segments is computed with optimal transport.

Segment Growth and Cropping/Decay A plant's growth may yield new parts like new branches or leaves. Analogously, there may also be a loss of parts such as the falling of leaves or the breaking of branches. This is reflected in the segment hierarchy. If the current target segment s_Q has more children than s_P, a new part has grown. If s_Q has fewer segments, plant parts have been cropped or decayed and fallen off the plant. In order to handle these cases, we create virtual matching segments. The child segment with the largest Wasserstein distance to any child segment of the other point cloud is considered unmatched and a virtual segment is generated in the other scan.

For growth scenarios, the point cloud associated with the virtual segment is generated by searching l points in the point cloud of s_P, that are closest to the points of the unmatched segment of point cloud Q. l has to be chosen by the user, depending on the density of the point cloud. In our experiments, we found values between 10 and 100 to deliver good results. These points are duplicated, labeled as a new segment and added to the segment hierarchy. The newly generated virtual segment is matched with the previously unmatched segment. This procedure is repeated for all unmatched segments.

For cropping or decaying scenarios, our goal is to generate an animation of the cropped part falling to the ground. Note that other solutions like the fading out of the respective points or their immediate removal are also viable and can be generated analogously. We duplicate the points of the cropped segment, but set their vertical axis coordinate to the point cloud's minimum. These points are added as a new segment to Q, which is then matched to the previously unmatched segment of P.

4.3.4 Optimal Transport Plan for Two Point Clouds

So far, we have outlined how point clouds are segmented and how the segments are associated between different time steps. Based on this information, we perform temporal upsampling by warping respectively matching the point distributions of previously matched segments. Let P and Q be their respective point clouds, which we want to match onto each other using optimal transport. Solving the optimal transport problem for given point clouds P and Q involves the computation of the *Wasserstein distance* $d_w(P,Q)$, which is essentially the cost of the optimal transport solution, that is, the cost of warping one distribution onto the other, and the temporal upsampling is performed by using the solution for the interpolation of the point clouds.

In the (discretized) optimal transport problem, each point $p_a \in P$, $1 \leq a \leq |P|$ and $q_b \in Q$, $1 \leq b \leq |Q|$ is assigned a mass m_a and m'_b, respectively, such that $\sum_{1 \leq a \leq |P|} m_a = 1 = \sum_{1 \leq b \leq |Q|} m'_b$. The goal is to find an optimal transport plan, which transports all mass from P to Q, minimizing the cost, which is given by a cost matrix $\boldsymbol{C} \in \mathbb{R}^{|P| \times |Q|}$, with entries C_{ab} defining the cost of transporting a piece of mass from p_a to q_b, where one common choice is given by the euclidean distance. The optimal solution is a transport matrix $\boldsymbol{T} \in \mathbb{R}^{|P| \times |Q|}$ with entries T_{ab} denoting the amount of mass transported from $p_a \in P$ to $q_b \in Q$, chosen such that the total transport cost is minimized. The optimal transport problem can be written as a linear program:

$$\begin{aligned} d_{w_2}^2(P,Q) = \min_{\boldsymbol{T}} & \sum_{1 \leq a \leq |P|} \sum_{1 \leq b \leq |Q|} T_{ab} \cdot C_{ab} \\ \text{s.t.} & \sum_{1 \leq a \leq |P|} m_a \cdot T_{ab} = m'_b \qquad \forall 1 \leq b \leq |Q| \wedge b \in \mathbb{N} \\ & \sum_{1 \leq b \leq |Q|} m'_b \cdot T_{ab} = m_a \qquad \forall 1 \leq a \leq |P| \wedge a \in \mathbb{N}, \end{aligned} \tag{4.1}$$

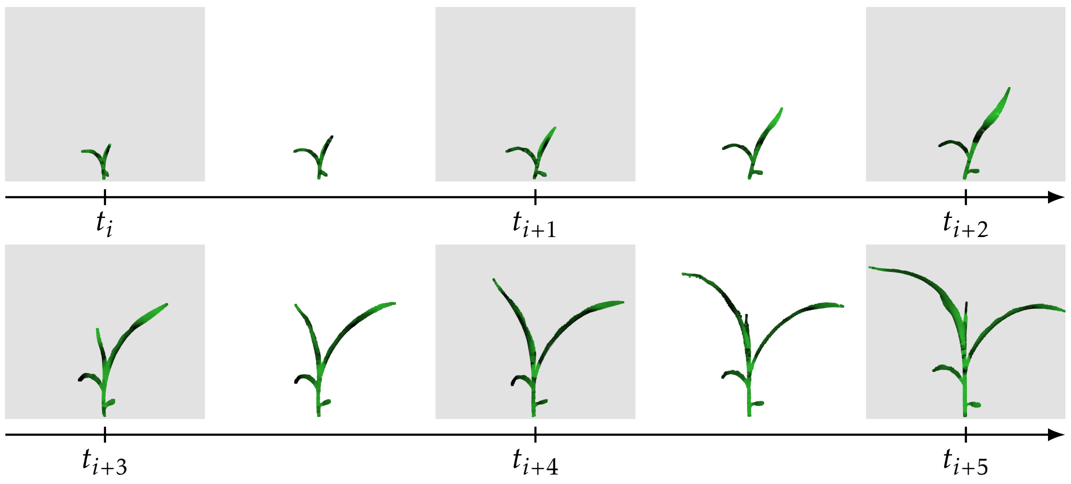


Figure 4.7: *Several stages of a growing maize plant, visualized with our method. Real measurements are highlighted with a gray background, while temporally upsampled data generated with our method is shown with a white background.*

which minimizes the (discretized) squared 2-Wasserstein distance $d_w(P, Q)$ between the point clouds P and Q. For small point clouds, this problem can be solved by standard algorithms.

Transport Problem Setup In order to compute the optimal transport from point cloud P to Q, we have to define the point masses m_a and m'_b, and the entries C_{ab} of the cost matrix $\boldsymbol{C}$. Since we do not have any particular preference for certain points, we define their masses uniformly as $m_a = 1/|P|$ and $m'_b = 1/|Q|$ for each point $p_a \in P$ and $q_b \in Q$, respectively.

For each point $p_a \in P$, we collect the k nearest neighbors $q_b \in Q$. This graph is represented by a sparse matrix $\boldsymbol{C}_P$, where entry $C_{P,ab} = d(p_a, q_b)$ is the distance between p_a and q_b if q_b is one of the nearest neighbors. If q_b is not among the k nearest neighbors of p_a, we define $C_{P,ab} = \infty$. The matrix $\boldsymbol{C}_P$ is the cost matrix of transport from the source cloud's points to the target cloud's points. This is a reasonable approximation to the true dense cost matrix, since points in Q that are far away from a point p_a are unlikely to be used as targets for transporting mass from p_a in the solution of the full optimal transport problem. The cost for transporting from p_a to non-neighbor points is thus infinite.

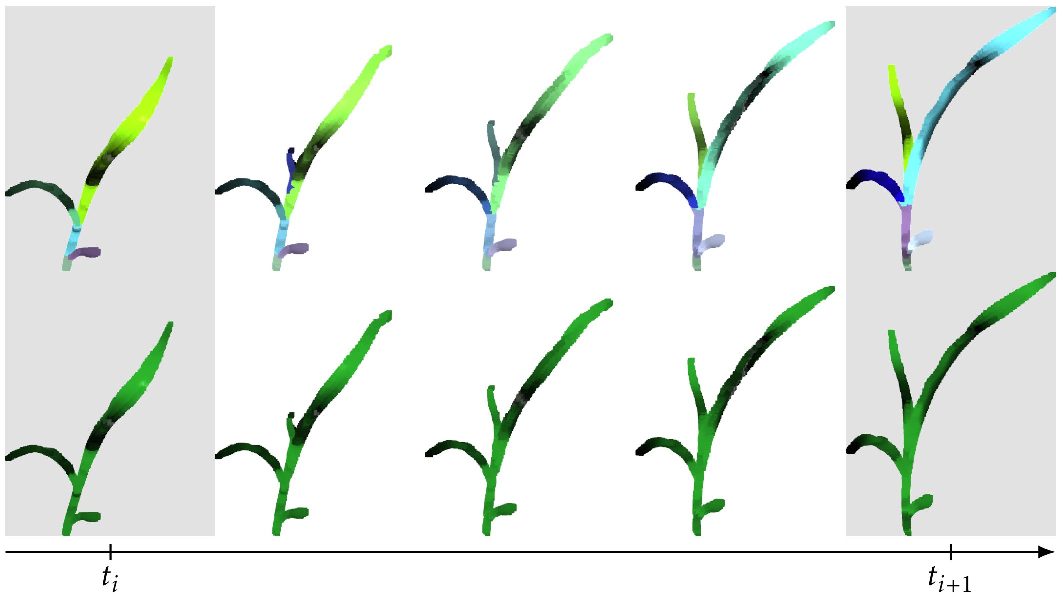


Figure 4.8: *Several stages of a growing maize plant. Images with a gray background are input data. Images with a white background were generated with our method. Between the two successive scans, the plant has grown regarding its size as well as a newly-evolved leaf. The budding process was generated automatically. In the upper row, the points were colorized according to their label. Note that the labels and their respective colors in the input data (gray background) do not match, because the input scans are segmented individually. Our matching method automatically produces correct assignments. The label color of matched segments is interpolated over time (white background) in order to visualize this.*

It is possible that some points of the target point cloud are not are not among the nearest neighbors. Since we want to compute a solution to the optimal transport problem that transports from each point and to each point, this is not desirable. In order to alleviate this, we also compute the k nearest neighbors $p_a \in P$ for points $q_b \in Q$. This is also represented by a sparse matrix $\boldsymbol{C}_Q$, where entry $C_{Q,ab} = d(p_a, q_b)$ represents the distance between p_a and q_b if p_a is among the nearest neighbors of q_b. The resulting cost matrix $\boldsymbol{C}$ is then defined as the coefficient-wise minimum of $\boldsymbol{C}_P$ and $\boldsymbol{C}_Q$. k determines the sparsity of the matrix and is user-defined. We found $k = 15$ to deliver good results in our experiments. Thus, in the resulting graph, there are at least k connections from each point in the source cloud to points in the target cloud and also at least k connections from each point in the target cloud to points in the source cloud.

The Sinkhorn Algorithm for Regularized Optimal Transport The optimal transport problem is approximated by the regularized optimal transport problem, which we solve using the Sinkhorn algorithm [Cut13]. In the regularized transport problem, the objective function in Eq. (4.1) is modified to

$$\min_{\boldsymbol{T}} \sum_{1 \le a \le |P|} \sum_{1 \le b \le |Q|} T_{ab} \cdot C_{ab} - \gamma H(\boldsymbol{T}), \tag{4.2}$$

where $H(\boldsymbol{T})$ is the entropy of the matrix $\boldsymbol{T}$, defined as

$$H(\boldsymbol{T}) = - \sum_{1 \le a \le |P|} \sum_{1 \le b \le |Q|} T_{ab} \log T_{ab}. \tag{4.3}$$

Inserting Eq. (4.3) into Eq. (4.2) yields

$$\min_{\boldsymbol{T}} \sum_{1 \le a \le |P|} \sum_{1 \le b \le |Q|} T_{ab} \cdot C_{ab} + \gamma \sum_{1 \le a \le |P|} \sum_{1 \le b \le |Q|} T_{ab} \log T_{ab}. \tag{4.4}$$

It can be shown [Cut13] that minimizing Eq. (4.4) leads to the following condition for the elements of $\boldsymbol{T}$:

$$T_{ab} = u_a K_{ab} v_b \tag{4.5}$$

where $K_{ab} = \exp(-C_{ab}/\gamma)$ are the entries of a matrix $\boldsymbol{K}$ and $\boldsymbol{u} = (u_a)_a, \boldsymbol{v} = (v_b)_b$ are unknown vectors. $\boldsymbol{K}$ is also a sparse matrix and the infinite distances in $\boldsymbol{C}$ become 0 in $\boldsymbol{K}$. This property is important in order to be able to handle large point clouds. A full cost matrix for large point clouds would not fit into the computer's memory. We initialize the vectors $\boldsymbol{u} = (u_a)_a = \mathbf{1}, \boldsymbol{v} = (v_b)_b = \mathbf{1} \forall a \in A, b \in B$. The Sinkhorn algorithm then finds an approximate solution by iterating the following steps: $\boldsymbol{m} = (m_a)_a, \boldsymbol{m'} = (m'_b)_b$

1. $\boldsymbol{u} \leftarrow \boldsymbol{m} \oslash \boldsymbol{K}\boldsymbol{u}$
2. $\boldsymbol{v} \leftarrow \boldsymbol{m'} \oslash \boldsymbol{K}\boldsymbol{v}$,

where $\oslash$ denotes an element-wise division. The elements of the approximate solution matrix $\boldsymbol{T}$ are then $T_{ab} := u_a K_{ab} v_b$.

Using the Solution of the Transport Problem for Interpolation The solution of the transport problem is a transport matrix $\boldsymbol{T}$ of the same size as $\boldsymbol{C}$, where each entry T_{ab} denotes the amount of mass that has to be transported from p_a to q_b. Since we are interested in an interpolation animation that is visually appealing, we set up the following requirements:

- No point can disappear, that is, each point of P has to move to at least one point of Q.
- No point can appear out of nowhere, that is, for each point of Q there must be at least one point of P moving to it.
- For efficiency, as few as possible point pairs should be used for the transport.

A solution to the first two requirements is to use each entry $T_{ab} > 0$ of the transport matrix and interpolate p_a to q_b. If there are several $T_{ab} > 0$ for fixed a, we create copies of p_a, which follow different animation paths to

Dataset	Segmentation	Matching + OT
Maize Plant	12.1 s	50 s
Cropped Maize	3 s	6 s
Synthetic Tree 1	4.2 s	42 s
Synthetic Tree 2	-	88 s
Komatsuna Plant 0	-	105 s
Komatsuna Plant 0/10	-	8 s

Table 4.1: *Overview of computation times for the segmentation step and the combined matching and optimal transport step for different datasets. For datasets that are already segmented, we do not report timings for segmentation.*

their respective q_b. Since the cost matrix $\boldsymbol{C}$ allows up to k targets for each p_a, the number of transported points is at most $k \cdot (|P| + |Q|)$ instead of $|P| \cdot |Q|$. Yet, this may be an unnecessarily huge amount of transported points, which is inefficient for real-time animations as the number of point copies may still become very big. In order to alleviate this, we perform the following optimization: We create a matrix $\boldsymbol{T}'$, which contains a 1 only at the maximum entry of each row of $\boldsymbol{T}$ at their respective position. All other entries are set to 0. We create another matrix $\boldsymbol{T}''$, which is 1 only at the column maxima of $\boldsymbol{T}$ and 0 otherwise. We then generate a matrix $\hat{\boldsymbol{T}} = \frac{1}{2}(\boldsymbol{T}' + \boldsymbol{T}'')$ which is used for the transport instead of $\boldsymbol{T}$.

Having this transport plan, we can visualize the result as an animation, where each point p_i and its copies are transported linearly to the q_j's positions in real-time.

4.4 Results

In this section, we show results obtained with our method. All experiments were performed on a PC with an Intel Core i7-8700K CPU and 64 GB RAM. We implemented our approach in Python. An overview of the computation times, separated into segmentation (where applicable) and segment matching and solving of the optimal transport problem is given in Table 4.1. The upsampling using the optimal transport solution and the visualization run in real-time, thus no computation times are given for this step.

Maize Plant Time Series We generated a smooth animation from eight scans of a maize plant. These were acquired with a hand-held laser scanner, in a fashion comparable to the one described by Paulus et al. [PSKL14]. Exemplary input scans and upsampled data are shown in Fig. 4.7 and a full animation is shown in the accompanying video. Fig. 4.8 shows two input scans from this dataset and their upsampling in more detail. The segmentation of these scans is shown by the coloring of the plant. The labels and their respective colors do not match in the input scans, because the point clouds were segmented individually. Our matching method automatically assigns the correct segments to each other. This is visualized by interpolating the segment color in the upsampled data. The scans in this dataset have between 1700 and 5000 points, which took 12.1 s. The point cloud matching and computation of optimal transport took 50.0 s. The total computation time was 62.2 s. The visualization of the interpolation runs in real-time.

We also computed an animation without computing a segmentation, using the described interpolation method on the whole point clouds, as shown in Fig. 4.3. Here, parts of one leaf get transported to a different leaf, because it exhibits a stronger growth than before. Employing the segmentation prohibits this artifact because the transport is restricted to take place only between matched segments.

Cropped Maize In order to test the handling of cropped plant segments, we manually removed a leaf of the maize plant. As a first test, we used the cropped plant as the source and the complete plant as the target. The result is an animation of a growing leaf, shown in Fig. 4.9. For the second test, we used the full plant as the source and the cropped plant as the target. Our method is able to generate an animation of the cropped leaf falling to the ground, shown in Fig. 4.10 and the accompanying video. The full plant had 5080 points, the cropped one 4100. The total time required for segmentation was 6 s. The time for computing the upsampling was 3 s in both cases.

Synthetic Tree 1 Time Series We generated a dataset of four virtual scans of a simulated growing tree with 778 to 2253 points. The segmentation took

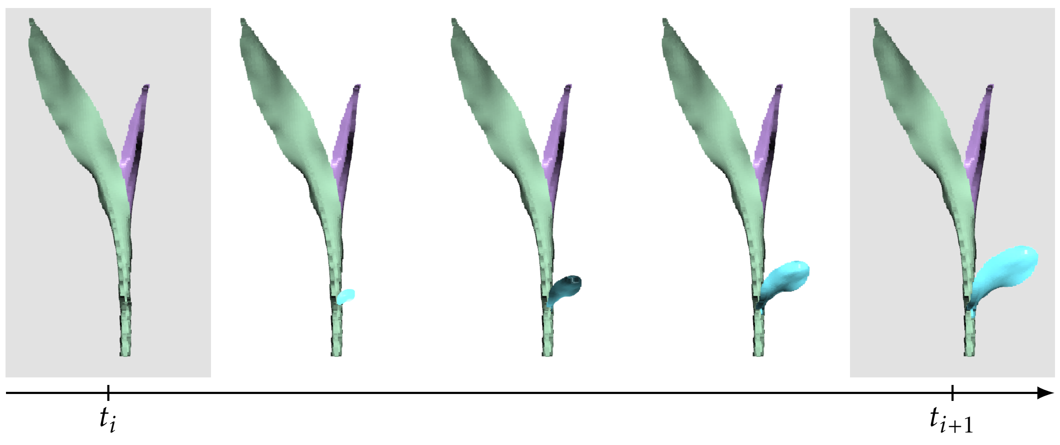


Figure 4.9: *Temporal upsampling generated from the two input point clouds highlighted with a gray background. The inputs are identical, except that the light blue leaf was removed in t_i, simulating the growth of a new plant part. The new leaf grows smoothly.*

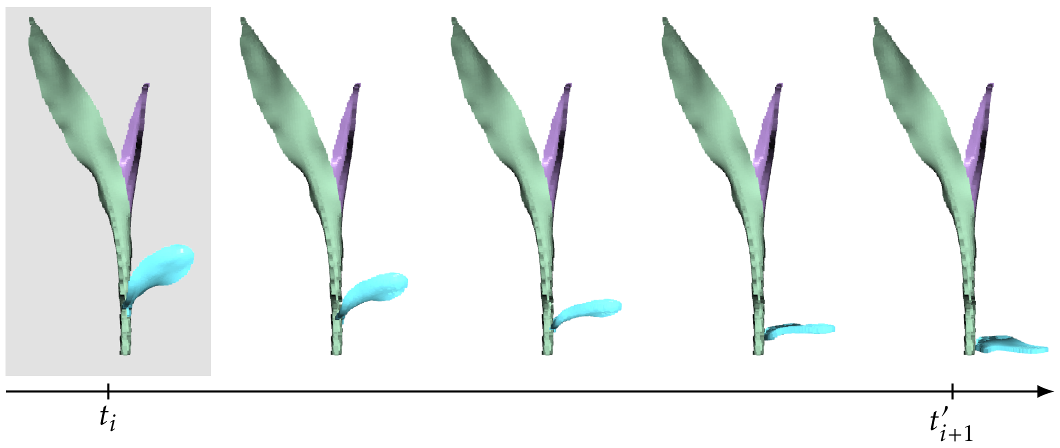


Figure 4.10: *Animation generated from the two input point clouds from Fig. 4.9 in reverse order. The inputs are highlighted with a gray background, depicting the full plant as the source and the cropped plant as the target (automatically modified version shown here). The cropped leaf falls slowly to the ground.*

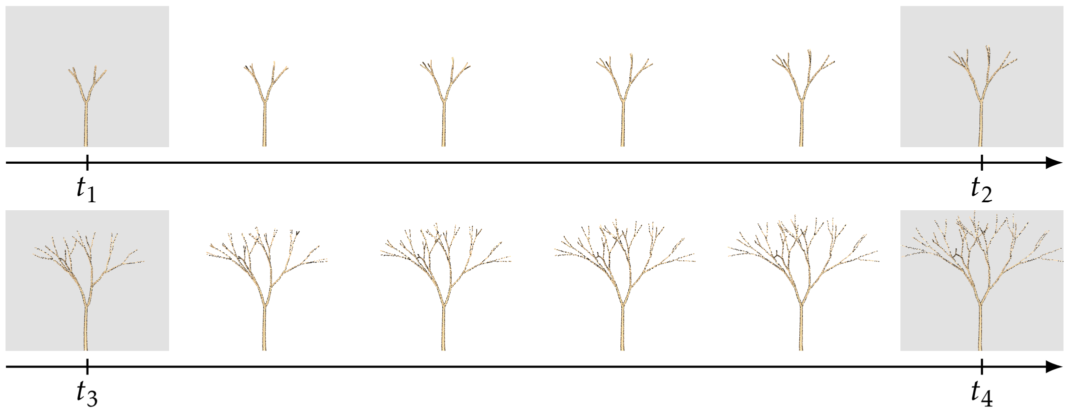


Figure 4.11: *The synthetic tree 1 dataset consists of four time steps t_1 to t_4. Input scans are highlighted with a gray background. The dataset was temporally upsampled with our method in the intervals $[t_1, t_2]$ and $[t_3, t_4]$, respectively, shown with a white background.*

4.2 s. Matching and optimal transport took 42.0 s. Our visualization runs in real-time. The dataset and its upsampling are shown in Fig. 4.11 and the accompanying video.

Synthetic Tree 2 Dataset We generated a synthetic tree dataset consisting of two point clouds, representing tree states for different times with a considerable growth in-between including the growth of many new branches, where the first has 77448 points and the second one 326032 points. A huge amount of growth with many new branches occurred. The segmentation was given due to the synthesis, as shown in Fig. 4.4. We created a hierarchy on the segments as described in Section 4.3.2. The computation of the matching and optimal transport plan took 88 s. Our algorithm was able to generate a smooth and realistic animation, which is depicted in Fig. 4.1 and shown in the accompanying video.

Komatsuna dataset for instance segmentation, tracking and reconstruction The Komatsuna RGB-D dataset described by Uchiyama et al. [USM+17] consists of a time series of RGB and depth images of five plants acquired with an RGB-D camera, as well as a manually created labeling of the plant parts.

All plants were recorded six times a day over a period of ten days. For our experiments, we used plant 0 and constructed colored 3D point clouds from the RGB-D images for each plant part. Each label in the dataset defines a segment in our context and is interpolated separately. In order to obtain a hierarchical representation, we create an empty virtual root node and added child nodes representing the plant's leaves.

Due to noise in the original data, the depth-discontinuities at leaf boundaries are not precisely aligned with the RGB images and label data, and the depth of thin branches is not well-captured. Therefore, we apply a semi-automatic preprocessing step before creating the 3D point clouds: First, we remove all depth values which we consider to be incorrect in a manual process and mark them as "unknown", which usually occurs at the boundaries of leaves. In a second step, we fill in the "unknown" depth values based on the known valid depth values, similar to the approach of Desbrun et al. [DMSB99]. This is carried out for each label of each plant separately.

This dataset shows partial occlusions and in some cases slightly misplaced manual labels. In Fig. 4.12 and in the accompanying video, it can be seen, that, while the general animation is smooth, these two aspects lead to some visual artifacts. The temporal occlusion leads to leaves seemingly shrinking away from the center of the plant and growing back to the center shortly later. The misplaced labeling leads to some points moving between leaves. Both are limitations of our method and may be addressed either in preprocessing or in future work. The computation time was 105 s.

As an additional experiment, we used only each 10th scan of the Komatsuna plant 0 dataset. The resulting upsampling looks convincing and shows fewer visual artifacts since by skipping input data, we essentially removed high-frequency temporal noise as also demonstrated in the accompanying video. The computation time was 10 s.

Verification: Reconstruction of the Komatsuna dataset by skipping original data To analyze the quality of the upsampling, we omit every second frame of the original dataset and compare it with a reconstruction generated with our method. As an example, we show scan 3 of day 9 of plant 0 of the

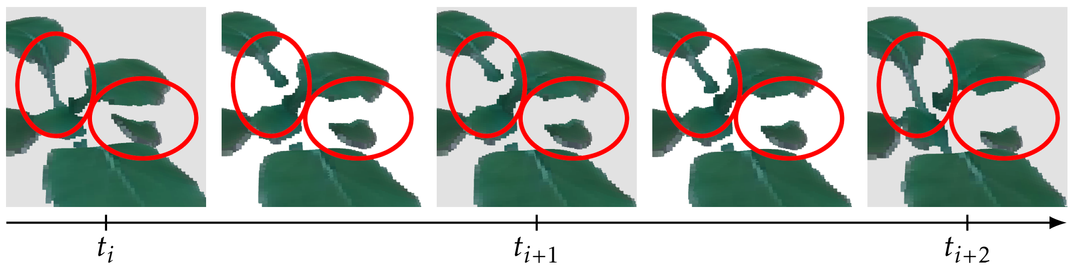
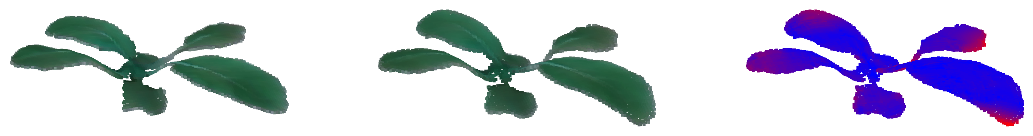


Figure 4.12: *Artifacts in the input data, like the temporarily severed stalk in the back and the partially occluded leaf on the right in this example are temporally upsampled as shrinking and growing animations. Input scans are highlighted with a gray background. We suggest a pre- or post-processing in order to mitigate these artifacts for future work.*

dataset. The whole dataset was rescaled to fit into the $[0, 1]^3$ unit volume. Comparing the reconstruction to the original data resulted in a mean distance of 0.001728 unit lengths, a standard deviation of 0.001130 unit lengths and a maximum distance of 0.00787517 unit lengths. A visual comparison is shown in Fig. 4.13.

4.5 Limitations and Future Work

The segmentation algorithm described in Section 4.3.2 is designed for thin structures as occurring in the plant data considered, whereas the matching and optimal transport components of our method do not have such limitations. In the future, we intend to investigate the combination of our method with other segmentation techniques, such as PointNet [QSMG17]. Furthermore, our method is tailored to data that can be represented as a hierarchy of segments, and the results of our method depend on the quality of the input data. If the input data seemingly moves back and forward, as in the Komatsuna dataset, this is also mirrored in our upsampling. The plant movement can be due to the natural movement of the plant or due to scanner noise. Varying sampling in the input data can also lead to counter-intuitive movement of the individual points in the upsampled dataset in some cases. This can either be resolved by resampling the data in a preprocessing step or by introducing additional constraints to the optimal transport problem, which would be an interesting



(**a**) *Original data from time step t_i*

(**b**) *Reconstruction from neighboring time steps t_{i-1} and t_{i+1}*

(**c**) *Error*

Figure 4.13: *Reconstruction of a point cloud of the Komatsuna dataset. Left: Original data of scan 3 of day 9 of plant 0. Center: Reconstruction of this data, using scans 2 and 4 of day 9 of plant 0. Right: Color-coded reconstruction error. Lower values are colored in blue, higher values in red. The maximum error is* 0.00787517 *unit lengths, with input data scaled to the unit volume.*

direction for future work. Noisy points are not smoothed out by the method itself. In this case, we rely on preprocessing. Furthermore, our method does not fill in temporarily occluded regions of the point cloud data. This can be resolved by preprocessing the data with existing methods like 3D shape inpainting.

Our method matches the segments traversing the hierarchy in a greedy fashion. While this strongly reduces computational complexity, it can also lead to errors that could be prevented by comparing all possible successor combinations. Segments that were pruned completely can be recognized and handled in a user-defined fashion. Partially pruned segments are currently not recognized and thus result in a shrinking animation. In the future, we would like to explore possibilities for handling these cases, e.g. by comparing the volume and only allowing volume increases for the animation. Volume decreases could be either faded out or filled.

4.6 Conclusion

We presented a novel, efficient approach for the temporal upsampling of point cloud sequences that, by design, circumvents limitations of previous approaches that rely on exact point correspondences, shape priors or the availability of huge datasets and allows handling topological changes to be expected for several growth processes. For this purpose, our approach involves an initial automatic alignment of successive scans, the generation of a segment hierarchy for each individual scan, the matching of the segment hierarchies obtained for successive scans and the final computation of segment-wise regularized optimal transport in a memory-efficient way that allows morphing the distributions of the point sets associated with matched segments onto each other. As demonstrated by our evaluation, our approach allows the generation of a smooth temporal upsampling of the input point cloud sequence and, hence, a realistic growth animation. We believe that our method will be useful for biologists and researchers in the field of agriculture, as well as in the entertainment industry.

Acknowledgments We would like to thank the reviewers and the editor for providing valuable feedback and suggestions for improvements. We would like to thank Alexander Schier for providing the synthetic tree 1 dataset and Lasse Klingbeil for his helpful comments.

5 Summary, Conclusion and Future Work

5.1 Summary and Conclusion

This thesis presented methods for transforming measured real-world input data into forms that can be interpreted as probability measures. These representations are suitable for setting up optimal transport problems. Here, it was demonstrated how to solve these problems efficiently and use the solutions for creating realistic interpolations in real-time.

The first type of measured data investigated were metallic effect car paints. These paints exhibit a complex appearance, which makes using them in real-time rendering applications a demanding task. Previous methods for representing them can be separated into two general fields: simulation-based and data-driven approaches. While both are able to achieve good results, simulation-based approaches are computationally complex and thus not well suited for most real-time applications. Data-driven approaches are less computationally complex but require larger amounts of memory. Rump et al. [RMS+08] introduced a mixed analytical and data-driven model. The analytical component is fitted to the measured raw data. The metallic sparkling effect is represented as a bidirectional texture function (BTF). This BTF is still memory-consuming to store. In Chapter 2, this problem was solved by introducing a new compact statistical representation for the BTF. This representation is generated by finding clusters of nearly uniform distribution in color space and computing their probability and their boundaries. This information is stored in textures. Using a specialized shader, it is possible to reconstruct a BTF representing the sparkling effect from this representation in real-time. This representation uses only 2.5 % of the memory that the original one used, while the visual impression remains very similar to the original representation and the computational cost is only slightly higher.

Chapter 3 showed that the statistical data representation is also useful for solving another problem: Interpolating the complex metallic effect car paints. While the analytical components of the model used are straightforward to interpolate, it is difficult to interpolate the high-frequency data-driven BTF component. Using the previously developed statistical BTF representation, it was shown how to set up an optimal transport problem between the BTFs of two metallic car paints. No manual parameter tuning is necessary for this process. The solution to this problem then serves as input for an interactive application that allows the user to interpolate metallic paints in real-time in an intuitive way. By adding the possibility to separately interpolate the individual aspects of the reflectance model, including the basic color hue, the local color hue, and the sparkling intensity of the BTFs, the user is given even more options for altering the appearance of metallic paints. The results are convincing while the computational cost is low.

In Chapter 4 the interpolation of a different type of measured data is investigated: Sequences of 3D scans of plants. The goal was to create a temporal upsampling of these sequences. As before, it was shown that by generating a suitable data representation and applying the theory of optimal transport appropriately, we can achieve convincing results. Each timestep of the data is represented as a hierarchy of segments on the point clouds is created. E.g. in the case of a tree, the point cloud representing its stem is the root of the hierarchy, followed by the branches. For subsequent point clouds, these hierarchies are traversed and the associated segments are matched by computing their pairwise Wasserstein distances – an optimal transport-based distance measure. After computing the matching by using the combination with the least total distance, an optimal transport problem per pair of matched segments is set up. The solution to this problem provides the associations between the points of the subsequent point clouds. These are used for interpolating the point clouds in real-time. Applying this method to a sequence of scans allows to generate a realistic-looking temporal upsampling of arbitrary precision.

In summary, this thesis introduced methods for generating interpolations of measured input data in areas where this was previously difficult to achieve. The methods work by first transforming the input data to a suitable representation and then making use of optimal transport theory. The final results are achieved by using the solutions to optimal transport problems in an appropriate way for interactive interpolation. The algorithms presented are comparatively easy to implement while also being efficient in terms of memory and computational complexity. We believe that the methods presented in this thesis will find widespread use and are looking forward to future applications and extensions.

5.2 Future Work

In this section, possible future directions for research, based on the results of this thesis are given. A limitation of the metallic car paint representation developed in Chapter 2 is that it can only reproduce metallic paint sparkles that have a size of one pixel. This limitation facilitated a perfectly parallel extraction in the graphics card's pixel shader. For the given data, the sparkles created by the metallic flakes were indeed so small that they only occupied one pixel at the camera resolution. When data of higher resolution become available, a modification will be required. Such an approach will have to take neighboring pixels into consideration when the BTF is regenerated. On current graphics card models, such a modification might lead to a negative impact on performance. Furthermore, research might be performed in representing other materials that exhibit stochastic patterns in a way similar to the described one, such that the ideas presented in this thesis can be transferred to new material types.

The BTF interpolation method based on optimal transport described in Chapter 3 may also be used for other materials if they can be represented as probability measures similar to the metallic paint BTFs. It may even be possible to interpolate materials of different natures.

The method for interpolation and temporal upsampling of point clouds described in Chapter 4 enables numerous applications. It can also be interpreted as generating non-injective, yet surjective, i.e. one-to-many, many-to-one and many-to-many correspondences between point clouds. One possibility for future applications is the combination with other segment matching methods and point path generation methods. A candidate would be the work of Wang et al.[WLX$^+$18], which describes morphing of 3d meshes of trees. In combination with the method presented in this thesis, it would allow for the morphing of scans of very different tree species.

In general, this thesis showed that in combination with a suitable data transformation and result interpretation, the theory of optimal transport allows for the generation of realistic intermediate states of measured input data. We are looking forward to future interesting applications based on our findings.

6 Bibliography

[ACOL00] Marc Alexa, Daniel Cohen-Or, and David Levin. As-rigid-as-possible shape interpolation. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '00, pages 157–164, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[ADT11] G. Alenya, B. Dellen, and C. Torras. 3d modelling of leaves from color and ToF data for robotized plant measuring. In *2011 IEEE International Conference on Robotics and Automation*, pages 3408–3414, Shanghai, China, May 2011. IEEE.

[AK16] Asen Atanasov and Vladimir Koylazov. A practical stochastic algorithm for rendering mirror-like flakes. In *ACM SIGGRAPH 2016 Talks*, page 67. ACM, 2016.

[Ale03] Marc Alexa. Differential coordinates for local mesh morphing and deformation. *The Visual Computer*, 19(2):105–114, May 2003.

[BGQ05] Yunfan Bao, Xiaohu Guo, and Hong Qin. Physically based morphing of point-sampled surfaces. *Computer Animation and Virtual Worlds*, 16(3-4):509–518, July 2005.

[BIZ18] P. Bertholet, A.E. Ichim, and M. Zwicker. Temporally consistent motion segmentation from RGB-D video. *Computer Graphics Forum*, 37(6):118–134, 2018.

[BM92] Paul J. Besl and Neil D. McKay. Method for registration of 3-D shapes. In *Sensor Fusion IV: Control Paradigms and Data Structures*, volume 1611, pages 586–606. International Society for Optics and Photonics, April 1992.

[BN92] Thaddeus Beier and Shawn Neely. Feature-based image metamorphosis. *ACM SIGGRAPH computer graphics*, 26(2):35–42, 1992.

[BT18] Sumukh Bansal and Aditya Tatu. Lie bodies based 3d shape morphing and interpolation. In *Proceedings of the 15th ACM SIGGRAPH European Conference on Visual Media Production*, CVMP '18, pages 5:1–5:10, New York, NY, USA, 2018. ACM.

[BVDPPH11] Nicolas Bonneel, Michiel Van De Panne, Sylvain Paris, and Wolfgang Heidrich. Displacement interpolation using lagrangian mass transport. *ACM Transactions on Graphics (TOG)*, 30(6):1–12, 2011.

[BVGP09] Ilya Baran, Daniel Vlasic, Eitan Grinspun, and Jovan Popović. Semantic deformation transfer. *ACM Transactions on Graphics (TOG)*, 28(3):36:1–36:6, July 2009.

[BX06] Guo Baolong and Fu Xiang. A modified octree color quantization algorithm. In *Communications and Networking in China, 2006. ChinaCom'06. First International Conference on*, pages 1–3. IEEE, 2006.

[CAJBS05] Somporn Chuai-Aree, Willi Jäger, Hans Georg Bock, and Suchada" Siripant. Simulation and Visualization of Plant Growth Using Lindenmayer Systems. In *Modeling, Simulation and Optimization of Complex Processes*, pages 115–126. Springer Berlin Heidelberg, 2005.

[CFB16] Xue Chen, Jieqing Feng, and Dominique Bechmann. Mesh sequence morphing. *Computer Graphics Forum*, 35(1):179–190, February 2016.

[CL09] Hung-Kuo Chu and Tong-Yee Lee. Multiresolution mean shift clustering algorithm for shape interpolation. *IEEE Transactions on Visualization and Computer Graphics*, 15(5):853–866, September 2009.

[CPSV16] Lenaic Chizat, Gabriel Peyré, Bernhard Schmitzer, and François-Xavier Vialard. Scaling algorithms for unbalanced transport problems. *arXiv preprint arXiv:1607.05816*, 2016.

[CT82] Robert L Cook and Kenneth E. Torrance. A reflectance model for computer graphics. *ACM Transactions on Graphics (TOG)*, 1(1):7–24, 1982.

[Cut13] Marco Cuturi. Sinkhorn Distances: Lightspeed Computation of Optimal Transport. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 2292–2300. Curran Associates, Inc., 2013.

[CW93] Shenchang Eric Chen and Lance Williams. View interpolation for image synthesis. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 279–288, 1993.

[DBFK⁺01] Patricia Dumont-Bècle, Eric Ferley, Andras Kemeny, Sylvain Michelin, and Didier Arquès. Multi-texturing approach for paint appearance simulation on virtual vehicles. In *Proceedings of the driving simulation conference*, pages 123–133, 2001.

[DCSA⁺14] Julie Digne, David Cohen-Steiner, Pierre Alliez, Fernando de Goes, and Mathieu Desbrun. Feature-Preserving Surface Reconstruction and Simplification from Defect-Laden Point

Sets. *Journal of Mathematical Imaging and Vision*, 48(2):369–382, February 2014.

[DGBOD12] Fernando De Goes, Katherine Breeden, Victor Ostromoukhov, and Mathieu Desbrun. Blue noise through optimal transport. *ACM Transactions on Graphics (TOG)*, 31(6):1–11, 2012.

[DGCSAD11] Fernando De Goes, David Cohen-Steiner, Pierre Alliez, and Mathieu Desbrun. An optimal transport approach to robust reconstruction and simplification of 2d shapes. In *Computer Graphics Forum*, volume 30, pages 1593–1602. Wiley Online Library, 2011.

[DHT+00] Paul Debevec, Tim Hawkins, Chris Tchou, Haarm-Pieter Duiker, Westley Sarokin, and Mark Sagar. Acquiring the reflectance field of a human face. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 145–156. ACM Press/Addison-Wesley Publishing Co., 2000.

[Dis98] Jean-Michel Dischler. Efficiently rendering macro geometric surface structures with bi-directional texture functions. *Rendering Techniques '98*, 98:169–180, 1998.

[ĎM03] Roman Ďurikovič and William L Martens. Simulation of sparkling and depth effect in paints. In *Proceedings of the 19th spring conference on Computer graphics*, pages 193–198. ACM, 2003.

[ĎM13] Roman Ďurikovič and Andrej Mihálik. Metallic paint appearance measurement and rendering. *Journal of Applied Mathematics, Statistics and Informatics*, 9(2):25–39, 2013.

[DMSB99] Mathieu Desbrun, Mark Meyer, Peter Schröder, and Alan H Barr. Implicit fairing of irregular meshes using diffusion and curvature flow. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, SIGGRAPH

'99, pages 317–324, USA, 1999. ACM Press/Addison-Wesley Publishing Co.

[DNC+18] Sundara Tejaswi Digumarti, Juan Nieto, Cesar Cadena, Roland Siegwart, and Paul Beardsley. Automatic Segmentation of Tree Structure From Point Cloud Data. *IEEE Robotics and Automation Letters*, 3(4):3043–3050, October 2018.

[Dob70] R. L. Dobrushin. Prescribing a System of Random Variables by Conditional Distributions. *Theory of Probability & Its Applications*, 15(3):458–486, January 1970.

[DRS10] Julie Dorsey, Holly Rushmeier, and François Sillion. *Digital modeling of material appearance*. Morgan Kaufmann, 2010.

[EĎKM04] Sergey Ershov, Roman Ďurikovič, Konstantin Kolchin, and Karol Myszkowski. Reverse engineering approach to appearance-based design of metallic and pearlescent paints. *The Visual Computer*, 20(8-9):586–600, 2004.

[EKK99] Sergey Ershov, Andrei Khodulev, and Konstantin Kolchin. Simulation of sparkles in metallic paints. In *Proceeding of Graphicon*, pages 121–128, 1999.

[EKM01] Sergey Ershov, Konstantin Kolchin, and Karol Myszkowski. Rendering pearlescent appearance based on paint-composition modelling. In *Computer Graphics Forum*, volume 20, pages 227–238. Wiley Online Library, 2001.

[FB11] Stefan Fröhlich and Mario Botsch. Example-driven deformations based on discrete shells. *Computer Graphics Forum*, 30(8):2246–2257, 2011.

[Flo53] Merrill M. Flood. On the Hitchcock distribution problem. *Pacific Journal of Mathematics*, 3(2):369–386, 1953.

[FPPA14] Sira Ferradans, Nicolas Papadakis, Gabriel Peyré, and Jean-François Aujol. Regularized discrete optimal transport. *SIAM Journal on Imaging Sciences*, 7(3):1853–1882, 2014.

[FSDH14] Oriel Frigo, Neus Sabater, Vincent Demoulin, and Pierre Hellier. Optimal transportation for example-guided color transfer. In *Computer Vision – ACCV 2014*, pages 655–670, Cham, Germany, 2014. Springer International Publishing.

[GCG+05] Johannes Günther, Tongbo Chen, Michael Goesele, Ingo Wald, and Hans-Peter Seidel. Efficient acquisition and realistic rendering of car paint. In *Vision, Modeling, and Visualization*, volume 5, pages 487–494, 2005.

[GCLX17] Lin Gao, Shu-Yu Chen, Yu-Kun Lai, and Shihong Xia. Data-driven shape interpolation and morphing editing. *Computer Graphics Forum*, 36(8):19–31, 2017.

[GD04] Kristen Grauman and Trevor Darrell. Fast contour matching using approximate earth mover's distance. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 1, pages I–I. IEEE, 2004.

[GDHB17] William Gelard, Michel Devy, Ariane Herbulot, and Philippe Burger. Model-based Segmentation of 3d Point Clouds for Phenotyping Sunflower Plants:. In *Proceedings of the 12th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*, pages 459–467. SCITEPRESS - Science and Technology Publications, 2017.

[GGG+16] Dar'ya Guarnera, Giuseppe Claudio Guarnera, Abhijeet Ghosh, Cornelia Denk, and Mashhuda Glencross. BRDF representation and acquisition. In *Computer Graphics Forum*, volume 35, pages 625–650. Wiley Online Library, 2016.

[GGH+17] Giuseppe Claudio Guarnera, Abhijeet Ghosh, Ian Hall, Mashhuda Glencross, and Dar'ya Guarnera. Material capture and representation with applications in virtual reality. In *ACM SIGGRAPH 2017 Courses*, page 6. ACM, 2017.

[GK15] Tim Golla and Reinhard Klein. Real-time Point Cloud Compression. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 5087–5092. IEEE, 2015.

[GK17] Tim Golla and Reinhard Klein. An Efficient Statistical Data Representation for Real-Time Rendering of Metallic Effect Car Paints. In *Virtual Reality and Augmented Reality: 14th EuroVR International Conference, EuroVR 2017*, pages 51–68. Springer, Cham, 2017.

[GK18] Tim Golla and Reinhard Klein. Interactive Interpolation of Metallic Effect Car Paints. In *Vision, Modeling & Visualization, VMV 2018*, EG VMV '18, page 11–20. The Eurographics Association, 2018.

[GKK+20] Tim Golla, Tom Kneiphof, Heiner Kuhlmann, Michael Weinmann, and Reinhard Klein. Temporal Upsampling of Point Cloud Sequences by Optimal Transport for Plant Growth Visualization. *Computer Graphics Forum*, 39(6):167–179, September 2020.

[GLHH13] Lin Gao, Yu-Kun Lai, Qi-Xing Huang, and Shi-Min Hu. A data-driven approach to realistic shape morphing. *Computer Graphics Forum*, 32(2pt4):449–457, 2013.

[GP88] Michael Gervautz and Werner Purgathofer. A simple method for color quantization: Octree quantization. In *New Trends in Computer Graphics*, pages 219–231. Springer Berlin Heidelberg, 1988.

[GSK14] Tim Golla, Christopher Schwartz, and Reinhard Klein. Towards Efficient Online Compression of Incrementally Acquired Point Clouds. In *Vision, Modeling & Visualization*. The Eurographics Association, 2014.

[HF04] Michal Haindl and Jiří Filip. A fast probabilistic bidirectional texture function model. In *Image Analysis and Recognition*, pages 298–305. Springer Berlin Heidelberg, 2004.

[HF07] Michal Haindl and Jiri Filip. Extreme compression and modeling of bidirectional texture function. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(10), 2007.

[HHCD05] Michal Haindl, Martin Hatka, M Chantler, and O Drbohlav. BTF roller. In *Proceedings of the 4th International Workshop on Texture Analysis and Synthesis*, pages 89–94, 2005.

[HLW07] Jianwei Hu, Ligang Liu, and Guozhao Wang. Dual laplacian morphing for triangular meshes. *Computer Animation and Virtual Worlds*, 18(4-5):271–277, September 2007.

[JBB13] O. Józsa, A. Börcs, and C. Benedek. Towards 4D virtual city reconstruction from lidar point cloud sequences. In *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, volume II-3/W1, pages 15–20, 2013.

[JHY+14] Wenzel Jakob, Miloš Hašan, Ling-Qi Yan, Jason Lawrence, Ravi Ramamoorthi, and Steve Marschner. Discrete stochastic microfacet models. *ACM Transactions on Graphics (TOG)*, 33(4):115, 2014.

[Kaj86] James T Kajiya. The Rendering Equation. In *ACM Siggraph Computer Graphics*, volume 20, pages 143–150. ACM, 1986.

[Kan39] Leonid V Kantorovich. The mathematical method of production planning and organization. *Management Science*, 6(4):363–422, 1939.

[Kan42] Leonid V Kantorovich. On the translocation of masses. In *Dokl. Akad. Nauk. USSR (NS)*, volume 37, pages 199–201, 1942.

[KBD07] Jan Kautz, Solomon Boulos, and Frédo Durand. Interactive editing and modeling of bidirectional texture functions. In *ACM Transactions on Graphics (TOG)*, volume 26, page 53. ACM, 2007.

[KGK19] Tom Kneiphof, Tim Golla, and Reinhard Klein. Real-time Image-based Lighting of Microfacet BRDFs with Varying Iridescence. *Computer Graphics Forum*, 38(4), July 2019.

[KGWK18] Tom Kneiphof, Tim Golla, Michael Weinmann, and Reinhard Klein. A Method for Fitting Measured Car Paints to a Game Engine's Rendering Model. In *Workshop on Material Appearance Modeling*, pages 27–31. The Eurographics Association, 2018.

[Kit08] Saori Kitaguchi. *Modelling texture appearance of gonioapparent objects*. PhD thesis, University of Leeds, 2008.

[KKBS05] Winfried Kurth, Ole Kniemeyer, and Gerhard Buck-Sorlin. Relational growth grammars – a graph rewriting approach to dynamical systems with a dynamical structure. In *Unconventional Programming Paradigms*, pages 56–72. Springer Berlin Heidelberg, 2005.

[KSKK10] Murat Kurt, László Szirmay-Kalos, and Jaroslav Křivánek. An anisotropic BRDF model for fitting and monte carlo rendering. *ACM SIGGRAPH Computer Graphics*, 44(1):3, 2010.

[KSOF05] Hiroshi Kawasaki, Kyoung-Dae Seo, Yutaka Ohsawa, and Ryo Furukawa. Patch-based BTF synthesis for real-time rendering. In *Image Processing, 2005. ICIP 2005. IEEE International Conference on*, volume 1, pages I–393. IEEE, 2005.

[LB01] Elizaveta Levina and Peter Bickel. The earth mover's distance is the mallows distance: Some insights from statistics. In

Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on, volume 2, pages 251–256. IEEE, 2001.

[LCCS18] Hugo Lavenant, Sebastian Claici, Edward Chien, and Justin Solomon. Dynamical optimal transport on discrete surfaces. *ACM Transactions on Graphics (TOG)*, 37(6):1–16, 2018.

[LCT+18] Dawei Li, Yan Cao, Xue-song Tang, Siyuan Yan, and Xin Cai. Leaf Segmentation on Dense Plant Point Clouds with Facet Region Growing. *Sensors*, 18(11):3625, November 2018.

[LFM+13] Yangyan Li, Xiaochen Fan, Niloy J. Mitra, Daniel Chamovitz, Daniel Cohen-Or, and Baoquan Chen. Analyzing growing plants from 4d point cloud data. *ACM Transactions on Graphics (TOG)*, 32(6):1–10, November 2013.

[LH03] R. Lougee-Heimer. The Common Optimization INterface for Operations Research: Promoting open-source software in the operations research community. *IBM Journal of Research and Development*, 47(1):57–66, January 2003.

[LHZ+04] Xinguo Liu, Yaohua Hu, Jingdan Zhang, Xin Tong, Baining Guo, and Heung-Yeung Shum. Synthesis and rendering of bidirectional texture functions on arbitrary surfaces. *IEEE transactions on visualization and computer graphics*, 10(3):278–289, 2004.

[LKMH05] Lars Linsen, Brian J Karis, E Gregory McPherson, and Bernd Hamann. Tree Growth Visualization. *Journal of WSCG*, page 8, 2005.

[LKYU12] Joakim Löw, Joel Kronander, Anders Ynnerman, and Jonas Unger. BRDF models for accurate and efficient rendering of glossy surfaces. *ACM Transactions on Graphics (TOG)*, 31(1):9, 2012.

[LLV+12] Hao Li, Linjie Luo, Daniel Vlasic, Pieter Peers, Jovan Popović, Mark Pauly, and Szymon Rusinkiewicz. Temporally coherent completion of dynamic shapes. *ACM Transactions on Graphics (TOG)*, 31(1):2:1–2:11, February 2012.

[LO07] Haibin Ling and Kazunori Okada. An efficient earth mover's distance algorithm for robust histogram comparison. *IEEE transactions on pattern analysis and machine intelligence*, 29(5):840–853, 2007.

[LPY+12] Yingying Liu, Juan Pan, Li Yang, Xiaodong Zhu, and Na Zhang. Visualization of Virtual Plants Growth Based on Open L-System. In Daoliang Li and Yingyi Chen, editors, *Computer and Computing Technologies in Agriculture V*, volume 368, pages 90–96. Springer Berlin Heidelberg, 2012.

[LSLCO05] Yaron Lipman, Olga Sorkine, David Levin, and Daniel Cohen-Or. Linear rotation-invariant coordinates for meshes. *ACM Transactions on Graphics (TOG)*, pages 479–487, 2005.

[LXZ+97] J. J. Loomis, Xiuwen Liu, Zhaohua Ding, K. Fujimura, M. L. Evans, and H. Ishikawa. Visualization of plant growth. In *Proceedings. Visualization '97 (Cat. No. 97CB36155)*, pages 475–478. IEEE, October 1997.

[MBW+18] Anders Krogh Mortensen, Asher Bender, Brett Whelan, Margaret M. Barbour, Salah Sukkarieh, Henrik Karstoft, and René Gislum. Segmentation of lettuce in coloured 3d point clouds for fresh weight estimation. *Computers and Electronics in Agriculture*, 154:373–381, November 2018.

[Mér11] Quentin Mérigot. A multiscale approach to optimal transport. In *Computer Graphics Forum*, volume 30, pages 1583–1592. Wiley Online Library, 2011.

[MHS$^+$19] Miłosz Makowski, Torsten Hädrich, Jan Scheffczyk, Dominik L Michels, Sören Pirk, and Wojtek Pałubicki. Synthetic Silviculture: Multi-scale Modeling of Plant Ecosystems. *ACM Transactions on Graphics (TOG)*, 38(4):14, July 2019.

[ML15] Gero Müller and Francis Lamy. Axf - appearance exchange format. Technical report, X-Rite, Inc., 4300 44th St. SE, Grand Rapids, MI 49505, 2015. Version 1.0.

[MMK03] Jan Meseth, Gero Müller, and Reinhard Klein. Preserving realism in real-time rendering of bidirectional texture functions. In *OpenSG Symposium*, pages 89–96. The Eurographics Association, 2003.

[MMS$^+$05] Gero Müller, Jan Meseth, Mirko Sattler, Ralf Sarlette, and Reinhard Klein. Acquisition, synthesis, and rendering of bidirectional texture functions. In *Computer Graphics Forum*, volume 24, pages 83–109. Wiley Online Library, 2005.

[MMT18] Quentin Mérigot, Jocelyn Meyron, and Boris Thibert. An Algorithm for Optimal Transport between a Simplex Soup and a Point Cloud. *SIAM Journal on Imaging Sciences*, 11(2):1363–1389, January 2018.

[Mon81] Gaspard Monge. Mémoire sur la théorie des déblais et des remblais. *Histoire de l'Académie Royale des Sciences de Paris*, 1781.

[MSK07] Gero Müller, Ralf Sarlette, and Reinhard Klein. Procedural editing of bidirectional texture functions. In *Proceedings of the 18th Eurographics conference on Rendering Techniques*, pages 219–230. Eurographics Association, 2007.

[NDM05] Addy Ngan, Frédo Durand, and Wojciech Matusik. Experimental analysis of BRDF models. *Rendering Techniques*, 2005(16th):2, 2005.

[PDMK13] Stefan Paulus, Jan Dupuis, Anne-Katrin Mahlein, and Heiner Kuhlmann. Surface feature based classification of plant organs from 3d laserscanned point clouds for plant phenotyping. *BMC Bioinformatics*, 14(1):238, July 2013.

[PKD07] François Pitié, Anil C Kokaram, and Rozenn Dahyot. Automated colour grading using colour distribution transfer. *Computer Vision and Image Understanding*, 107(1):123–137, 2007.

[PSKL14] Stefan Paulus, Henrik Schumann, Heiner Kuhlmann, and Jens Léon. High-precision laser scanning system for capturing 3d plant architecture and analysing growth of cereal plants. *Biosystems Engineering*, 121:1–11, May 2014.

[PW09] Ofir Pele and Michael Werman. Fast and robust earth mover's distances. In *Computer vision, 2009 IEEE 12th international conference on*, pages 460–467. IEEE, 2009.

[QHL$^+$19] Hongxing Qin, Jia Han, Ning Li, Hui Huang, and Baoquan Chen. Mass-driven topology-aware curve skeleton extraction from incomplete point clouds. *IEEE Transactions on Visualization and Computer Graphics*, 2019.

[QSMG17] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 652–660. IEEE, 2017.

[RFP14] Julien Rabin, Sira Ferradans, and Nicolas Papadakis. Adaptive color transfer with relaxed optimal transport. In *2014 IEEE International Conference on Image Processing (ICIP)*, pages 4852–4856. IEEE, 2014.

[RGB16] Boris Raymond, Gael Guennebaud, and Pascal Barla. Multiscale rendering of scratched materials using a structured SV-

BRDF model. *ACM Transactions on Graphics (TOG)*, 35(4):57, 2016.

[RMS+08] Martin Rump, Gero Müller, Ralf Sarlette, Dirk Koch, and Reinhard Klein. Photo-realistic rendering of metallic car paint from image-based measurements. In *Computer Graphics Forum*, volume 27, pages 527–536. Wiley Online Library, 2008.

[RP11] Julien Rabin and Gabriel Peyré. Wasserstein regularization of imaging problem. In *2011 18th IEEE International Conference on Image Processing*, pages 1541–1544. IEEE, 2011.

[RPDB11] Julien Rabin, Gabriel Peyré, Julie Delon, and Marc Bernot. Wasserstein barycenter and its application to texture mixing. In *International Conference on Scale Space and Variational Methods in Computer Vision*, pages 435–446. Springer Berlin Heidelberg, 2011.

[RSK09] Martin Rump, Ralf Sarlette, and Reinhard Klein. Efficient resampling, compression and rendering of metallic and pearlescent paint. In *Vision, Modeling, and Visualization*, pages 11–18, 2009.

[RSK13] Roland Ruiters, Christopher Schwartz, and Reinhard Klein. Example-based interpolation and synthesis of bidirectional texture functions. In *Computer Graphics Forum*, volume 32, pages 361–370. Wiley Online Library, 2013.

[RT01] Yossi Rubner and Carlo Tomasi. The earth mover's distance. In *Perceptual Metrics for Image Database Navigation*, pages 13–28. Springer, 2001.

[RTG98] Yossi Rubner, Carlo Tomasi, and Leonidas J Guibas. A metric for distributions with applications to image databases. In *Sixth International Conference on Computer Vision*, pages 59–66. IEEE, 1998.

[RTG00] Yossi Rubner, Carlo Tomasi, and Leonidas J Guibas. The earth mover's distance as a metric for image retrieval. *International Journal of Computer Vision*, 40(2):99–121, 2000.

[RYZ11] Zhou Ren, Junsong Yuan, and Zhengyou Zhang. Robust hand gesture recognition based on finger-earth mover's distance with a commodity depth camera. In *Proceedings of the 19th ACM international conference on Multimedia*, pages 1093–1096. ACM, 2011.

[SBM$^+$10] O. Stava, B. Beneš, R. Měch, D. G. Aliaga, and P. Krištof. Inverse Procedural Modeling by Automatic Generation of L-systems. *Computer Graphics Forum*, 29(2):665–674, May 2010.

[Sch94] Christophe Schlick. An inexpensive BRDF model for physically-based rendering. In *Computer graphics forum*, volume 13, pages 233–246. Wiley Online Library, 1994.

[SCOL$^+$04] O. Sorkine, D. Cohen-Or, Y. Lipman, M. Alexa, C. Rössl, and H.-P. Seidel. Laplacian surface editing. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, SGP '04, pages 175–184, New York, NY, USA, 2004. ACM.

[SDGP$^+$15] Justin Solomon, Fernando De Goes, Gabriel Peyré, Marco Cuturi, Adrian Butscher, Andy Nguyen, Tao Du, and Leonidas Guibas. Convolutional wasserstein distances: Efficient optimal transportation on geometric domains. *ACM Transactions on Graphics (TOG)*, 34(4):66, 2015.

[Sin64] Richard Sinkhorn. A relationship between arbitrary positive matrices and doubly stochastic matrices. *The annals of mathematical statistics*, 35(2):876–879, 1964.

[SPK$^+$14] O. Stava, S. Pirk, J. Kratt, B. Chen, R. Měch, O. Deussen, and B. Benes. Inverse Procedural Modelling of Trees: Inverse Procedural Modeling of Trees. *Computer Graphics Forum*, 33(6):118–131, September 2014.

[SRGB14] Justin Solomon, Raif Rustamov, Leonidas Guibas, and Adrian Butscher. Earth Mover's Distances on Discrete Surfaces. *ACM Trans. Graph.*, 33(4):67:1–67:12, July 2014.

[SZGP05] Robert W. Sumner, Matthias Zwicker, Craig Gotsman, and Jovan Popović. Mesh-based inverse kinematics. *ACM Transactions on Graphics (TOG)*, 24(3):488–495, 2005.

[SZP07] Barbara Solenthaler, Yanci Zhang, and Renato Pajarola. Efficient Refinement of Dynamic Point Data. In M. Botsch, R. Pajarola, B. Chen, and M. Zwicker, editors, *Eurographics Symposium on Point-Based Graphics*. The Eurographics Association, 2007.

[THCF06] Haishan Tian, Yuanjun He, Hongming Cai, and Lirong Feng. Efficient Metamorphosis of Point-Sampled Geometry. In *16th International Conference on Artificial Reality and Telexistence–Workshops (ICAT'06)*, pages 260–263, Hangzhou, Zhejiang, China, 2006. IEEE.

[TTOO90] Atsushi Takagi, Hitoshi Takaoka, Tetsuya Oshima, and Yoshinori Ogata. Accurate rendering technique based on colorimetric conception. In *ACM SIGGRAPH Computer Graphics*, volume 24, pages 263–272. ACM, 1990.

[TWB05] Atsushi Takagi, Akihiro Watanabe, and Gorow Baba. Prediction of spectral reflectance factor distribution of automotive paint finishes. *Color Research & Application*, 30(4):275–282, 2005.

[TZL$^+$02] Xin Tong, Jingdan Zhang, Ligang Liu, Xi Wang, Baining Guo, and Heung-Yeung Shum. Synthesis of bidirectional texture functions on arbitrary surfaces. In *ACM Transactions on Graphics (ToG)*, volume 21, pages 665–672. ACM, 2002.

[TZZ09] G. Tan, S. Zhang, and Y. Zhang. Shape Morphing for Point Set Surface Based on Vertex Deformation Gradient. In *2009 WRI World Congress on Software Engineering*, volume 2, pages 466–471. IEEE, May 2009.

[USM+17] Hideaki Uchiyama, Shunsuke Sakurai, Masashi Mishima, Daisaku Arita, Takashi Okayasu, Atsushi Shimada, and Rin-ichiro Taniguchi. An Easy-to-Setup 3d Phenotyping Platform for KOMATSUNA Dataset. In *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*, pages 2038–2045, Venice, Italy, October 2017. IEEE.

[VLSM18] E. Vlachos, A. S. Lalos, A. Spathis-Papadiotis, and K. Moustakas. Distributed consolidation of highly incomplete dynamic point clouds based on rank minimization. *IEEE Transactions on Multimedia*, 20(12):3276–3288, Dec 2018.

[VTSSH15] Christoph Von-Tycowicz, Christian Schulz, Hans-Peter Seidel, and Klaus Hildebrandt. Real-time nonlinear shape interpolation. *ACM Transactions on Graphics (TOG)*, 34(3):34:1–34:10, May 2015.

[Was69] Leonid N Wasserstein. Markov processes over denumerable products of spaces describing large systems of automata. *Problems of Information Transmission*, 5(3):47–52, 1969.

[WH97] Douglas J Wiley and James K Hahn. Interpolation synthesis of articulated figure motion. *IEEE Computer Graphics and Applications*, 17(6):39–45, 1997.

[WLX+18] Guan Wang, Hamid Laga, Ning Xie, Jinyuan Jia, and Hedi Tabia. The Shape Space of 3d Botanical Tree Models. *ACM Transactions on Graphics (TOG)*, 37(1):1–18, January 2018.

[WPKM15] Mirwaes Wahabzada, Stefan Paulus, Kristian Kersting, and Anne-Katrin Mahlein. Automated interpretation of 3d laserscanned point clouds for plant organ segmentation. *BMC Bioinformatics*, 16(1), December 2015.

[WWX+19] Sheng Wu, Weiliang Wen, Boxiang Xiao, Xinyu Guo, Jianjun Du, Chuanyu Wang, and Yongjian Wang. An Accurate Skeleton Extraction Approach From 3d Point Clouds of Maize Plants. *Frontiers in Plant Science*, 10, 2019.

[WZH12] Renfang Wang, Changwei Zhang, and Jie Hu. Smooth Morphing of Point-Sampled Geometry. In Tai-hoon Kim, Hyun-seob Cho, Osvaldo Gervasi, and Stephen S. Yau, editors, *Computer Applications for Graphics, Grid Computing, and Industrial Environment*, Communications in Computer and Information Science, pages 16–23. Springer Berlin Heidelberg, 2012.

[XM15] Ling Xu and David Mould. Procedural Tree Modeling with Guiding Vectors. *Computer Graphics Forum*, 34(7):47–56, October 2015.

[XSWL15] W. Xu, M. Salzmann, Y. Wang, and Y. Liu. Deformable 3D fusion: From partial dynamic 3D observations to complete 4D models. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 2183–2191. IEEE, Dec 2015.

[XZPF04] Chunxia Xiao, Wenting Zheng, Qunsheng Peng, and A. R. Forrest. Robust morphing of point-sampled geometry. *Computer Animation and Virtual Worlds*, 15(3-4):201–210, July 2004.

[XZWB06] Dong Xu, Hongxin Zhang, Qing Wang, and Hujun Bao. Poisson shape interpolation. *Graphical Models*, 68(3):268 – 281, 2006.

[YHJ+14] Ling-Qi Yan, Miloš Hašan, Wenzel Jakob, Jason Lawrence, Steve Marschner, and Ravi Ramamoorthi. Rendering glints on high-resolution normal-mapped specular surfaces. *ACM Transactions on Graphics (TOG)*, 33(4):116, 2014.

[YHMR16] Ling-Qi Yan, Miloš Hašan, Steve Marschner, and Ravi Ramamoorthi. Position-normal distributions for efficient rendering of specular microstructure. *ACM Transactions on Graphics (TOG)*, 35(4):56, 2016.

[YLX⁺16] Qing Yuan, Guiqing Li, Kai Xu, Xudong Chen, and Hui Huang. Space-time co-segmentation of articulated point cloud sequences. *Computer Graphics Forum*, 35(2):419–429, 2016.

[ZB13] Yili Zhao and Jernej Barbič. Interactive authoring of simulation-ready plants. *ACM Transactions on Graphics (TOG)*, 32(4):84:1–84:12, July 2013.

[ZDW⁺05] Kun Zhou, Peng Du, Lifeng Wang, Yasuyuki Matsushita, Jiaoying Shi, Baining Guo, and Heung-Yeung Shum. Decorating surfaces with bidirectional texture functions. *IEEE Transactions on Visualization and Computer Graphics*, 11(5):519–528, 2005.

[ZFG⁺17] Qian Zheng, Xiaochen Fan, Minglun Gong, Andrei Sharf, Oliver Deussen, and Hui Huang. 4D reconstruction of blooming flowers. *Computer Graphics Forum*, 36(6):405–417, 2017.

[ZLLZ12] Q Zeng, X Lin, H Liu, and T Zhu. Plant's 3d modeling and growth simulation system based on skeleton extraction. *Journal of Information & Computational Science*, 9:1357–1363, May 2012.

List of Figures

List of Tables