A Versatile Parameterization for Measured Material Manifolds

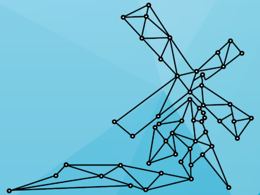
Cyril Soler[1], Kartic Subr[2], Derek Nowrouzezahrai[3]

[1] INRIA, Grenoble University, France

[2] University of Edinburgh, Scotland

[3] McGill University, Canada

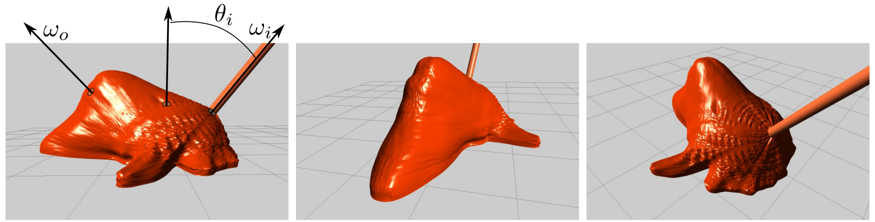
EUROGRAPHICS 2018



- motivation
- gaussian process regression
- evaluation
- applications: leveraging linearity

Problem statement

- Measured/Tabulated BRDFs are non trivial functions...



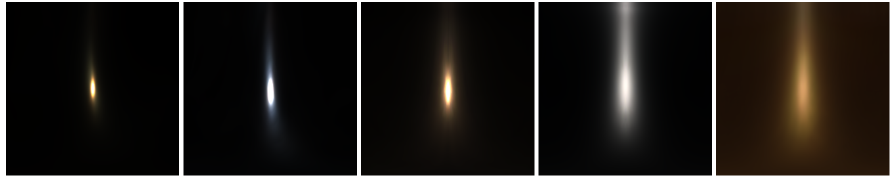
MERL "`blue rubber`" [Matusik'2003]

Dimension: 4.10^6 data points

1. **Tabulated BRDFs is** what you get from measurements with a gonioreflectometer.
2. **They come as a vector of values** corresponding to how much of the incident light in a direction ω_i is refltected into a direction ω_o
3. **These functions are generally non trivial**, as can be seen here on the blue rubber measurement from the MERL database
4. **However, when aligning multiple measurements together**, it's quite visible that different materials share lots of similarities. As such they probably belo to a space of much lower dimension than the size of the tabulated data
5. **This is our motivation for this work:** find a low dimensional parameterization of measured materials. On this slide we found a 2D map where each poir corresponds to a full tabulated BRDF
6. **We would like to use this:** for instance to browse materials, during a design process. So we need to be able to look into any point in the low dimensiona space and see what BRDF lies there

Problem statement

- Measured/Tabulated BRDFs are non trivial functions...
- ...but share similarities



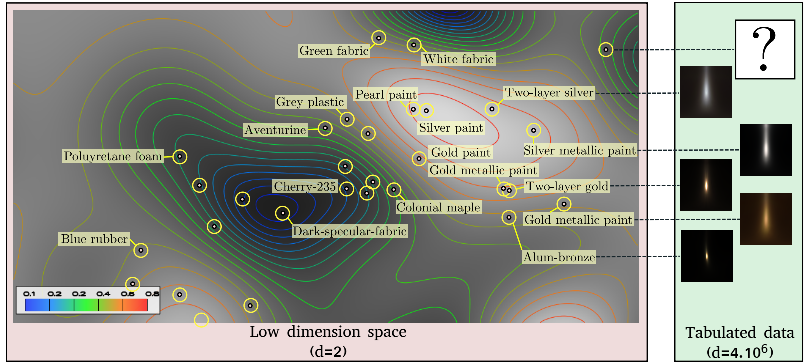
$\theta_i = \frac{\pi}{4}$

1. **Tabulated BRDFs is** what you get from measurements with a gonioreflectometer.
2. **They come as a vector of values** corresponding to how much of the incident light in a direction ω_i is refltected into a direction ω_o
3. **These functions are generally non trivial**, as can be seen here on the blue rubber measurement from the MERL database
4. **However, when aligning multiple measurements together**, it's quite visible that different materials share lots of similarities. As such they probably belo to a space of much lower dimension than the size of the tabulated data
5. **This is our motivation for this work:** find a low dimensional parameterization of measured materials. On this slide we found a 2D map where each poir corresponds to a full tabulated BRDF
6. **We would like to use this:** for instance to browse materials, during a design process. So we need to be able to look into any point in the low dimensiona space and see what BRDF lies there

Problem statement

- Measured/Tabulated BRDFs are non trivial functions...
- ...but share similarities

  $\rightarrow$ belong to a low dimension manifold

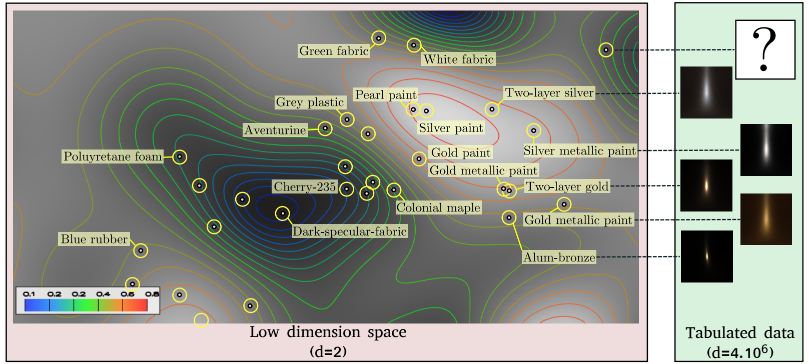


1. **Tabulated BRDFs is** what you get from measurements with a gonioreflectometer.
2. **They come as a vector of values** corresponding to how much of the incident light in a direction ω_i is refltected into a direction ω_o
3. **These functions are generally non trivial**, as can be seen here on the blue rubber measurement from the MERL database
4. **However, when aligning multiple measurements together**, it's quite visible that different materials share lots of similarities. As such they probably belo to a space of much lower dimension than the size of the tabulated data
5. **This is our motivation for this work:** find a low dimensional parameterization of measured materials. On this slide we found a 2D map where each poir corresponds to a full tabulated BRDF
6. **We would like to use this:** for instance to browse materials, during a design process. So we need to be able to look into any point in the low dimensiona space and see what BRDF lies there

Problem statement

We need a (drastic) dimensionality reduction technique that...

- interpolates the BRDF database
- lets us choose the manifold dimension
- maps low to high dimensions
- allows fast rendering



1. **Therefore we need** a pretty drastic dimensionality reduction technique, with some additional properties:
2. **It needs to reproduce the input data**, rather than approximate it
3. **because we want to use that for material design** it should be compatible with very low dimensions. But we may also want to use maps of higher dimensions to increase the possibility to more cleverly pack the materials.
4. **because we want to compute the BRDFs** at other points, we need the dimensionality reduction technique to map the low dimension space into the high dimension space
5. **and last but not least,** it would be good to be able to efficiently use BRDFs at these new points
6. **So if we review the classical dimensionality reduction techniques:**, namely PCA, kernel PCA, Dictionnary models which provide some sort of convex combination of the input data; Multidimensional scaling that provides coordinates in a euclidian space that match a pairwise distance matrix between the input data; Isomap that is a combination of MDS with shortest path calculation in a neighborhood graph; And finally, because we're dealing with BRDFs, analytical models can be a way to parameterize BRDFs too.
7. **Saddly, only these three** actually interpolate the input data, among which only MDS and isomap allow us to choose the dimension of the parameter spac and neither of them actually allow us to map the low dimension space to the tabulated space
8. **So we're not happy!**
9. **This is without considering** Gaussian Process Regression, which actually exactly fits our needs very nicely. And our contribution was precisely to adap this technique to BRDF data interpolation

Problem statement

We need a (drastic) dimensionality reduction technique that...

- interpolates the BRDF database
- lets us choose the manifold dimension
- maps low to high dimensions
- allows fast rendering

Traditional techniques:

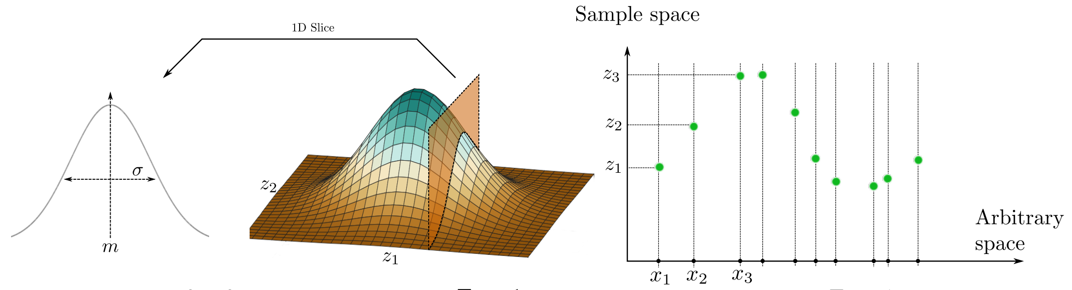
- PCA (Principal Component Analysis)
- Kernel PCA
- Dictionnary models
- MDS (Multidimensional Scaling)
- IsoMap
- Analytical models (GGX, Beckman, Lafortune, SGD, etc)

1. **Therefore we need** a pretty drastic dimensionality reduction technique, with some additional properties:
2. **It needs to reproduce the input data**, rather than approximate it
3. **because we want to use that for material design** it should be compatible with very low dimensions. But we may also want to use maps of higher dimensions to increase the possibility to more cleverly pack the materials.
4. **because we want to compute the BRDFs** at other points, we need the dimensionality reduction technique to map the low dimension space into the high dimension space
5. **and last but not least,** it would be good to be able to efficiently use BRDFs at these new points
6. **So if we review the classical dimensionality reduction techniques:**, namely PCA, kernel PCA, Dictionnary models which provide some sort of convex combination of the input data; Multidimensional scaling that provides coordinates in a euclidian space that match a pairwise distance matrix between the input data; Isomap that is a combination of MDS with shortest path calculation in a neighborhood graph; And finally, because we're dealing with BRDFs, analytical models can be a way to parameterize BRDFs too.
7. **Saddly, only these three** actually interpolate the input data, among which only MDS and isomap allow us to choose the dimension of the parameter spac and neither of them actually allow us to map the low dimension space to the tabulated space
8. **So we're not happy!**
9. **This is without considering** Gaussian Process Regression, which actually exactly fits our needs very nicely. And our contribution was precisely to adap this technique to BRDF data interpolation

- motivation
- gaussian process regression
- evaluation
- applications: leveraging linearity

Gaussian process regression

and we will apply this in parallel to all pairs of directions.



$$f(z) = e^{-(z-m)^2/\sigma^2} \qquad f(\mathbf{z}) = e^{-(\mathbf{z}-\mathbf{m})^{\mathbf{T}}\mathbf{K}^{-1}(\mathbf{z}-\mathbf{m})} \qquad f(\mathbf{z}) = e^{-(\mathbf{z}-\mathbf{m})^{\mathbf{T}}\mathbf{K}^{-1}(\mathbf{z}-\mathbf{m})}$$

$$\underbrace{\mathbf{K}}_{\text{Covariance matrix}}$$

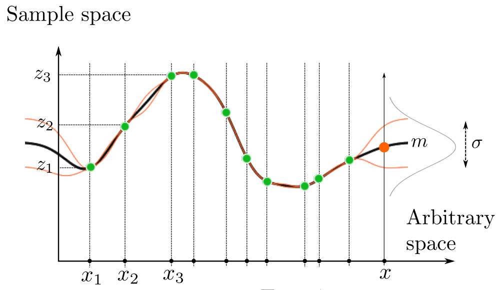
$$\mathbf{K}_{ij} = \underbrace{c}_{\text{Covariance function}}(x_i, x_j)$$

1. **You all know what a Gaussian distribution is**: it's a 1-dimensional probability distribution defined by its mean and variance, from which you can draw samples.
2. **A Gaussian distribution can be generalized** to a multi-dimensional vector space, which allows to draw multi-dimensional samples. What's interesting is that when I slice this distribution (in any direction), the posterior distribution is also a Gaussian distribution
3. **A Gaussian process is a generalization of this to arbitrary dimensions**: a vector of any dimension can be sampled from it, using a Gaussian distribution which covariance matrix is build using a symmetric function over an arbitrary space. Each entry in the covariance matrix is the value of this function for pairs of points in this space
4. **What's interesting here is that given some existing samples**, the posterior distribution for the value at a new point x is also a Gaussian distribution. S the mean of the posterior distribution at a varying point x can be considered to be a function defined over this arbitrary space, that implicitely interpolates the existing points.
5. **The value of that function is given by the following formula**, that involves the covariance between x and all other points in the parameter space, and t existing values at the known points.
6. **So let me rephrase this:** with this technique, we choose an arbitrary space, and some positions x_i in this space to which we assign the z_i values. This formula will turn this space into a parameterisation of the manifold that implicitly passes through our data points.
7. **In order to use this, we will consider BRDFs values** at a specific pair of directions to be the samples to interpolate.
8. **Therefore we pack all BRDFs** from the input database into a long matrix, where each line is a material and each column is independently interpolated b the Gaussian Process regression
9. **Doing so, we have constructed** an implicit manifold that interpolates BRDFs, and is parameterized in an arbitrary space.

Gaussian process regression

and we will apply this in parallel to all pairs of directions.

$$m = \mathbf{k}^{\mathbf{T}}\mathbf{K}^{-1}\mathbf{z}$$
$$\sigma^2 = c(x,x) - \mathbf{k}^{\mathbf{T}}\mathbf{K}^{-1}\mathbf{k}$$
$$\mathbf{k} = [c(x,x_0), ..., c(x,x_n)]$$



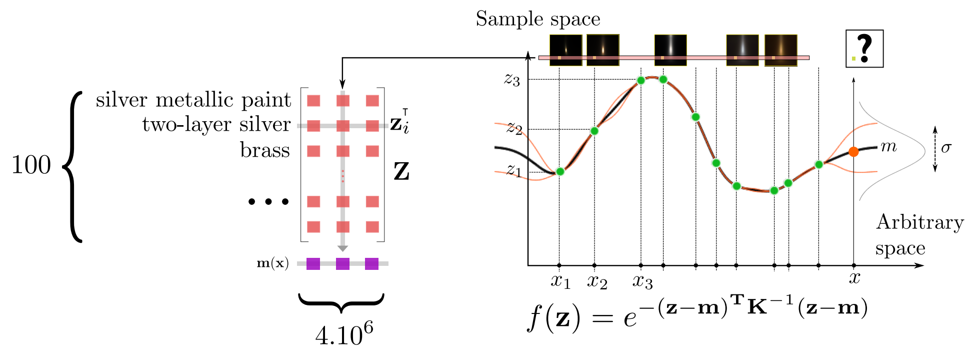
$$f(\mathbf{z}) = e^{-(\mathbf{z}-\mathbf{m})^{\mathbf{T}}\mathbf{K}^{-1}(\mathbf{z}-\mathbf{m})}$$

$$\mathbf{K}_{ij} = \underbrace{c}_{}(x_i, x_j)$$
↳ Covariance function

1. **You all know what a Gaussian distribution is**: it's a 1-dimensional probability distribution defined by its mean and variance, from which you can draw samples.
2. **A Gaussian distribution can be generalized** to a multi-dimensional vector space, which allows to draw multi-dimensional samples. What's interesting is that when I slice this distribution (in any direction), the posterior distribution is also a Gaussian distribution
3. **A Gaussian process is a generalization of this to arbitrary dimensions**: a vector of any dimension can be sampled from it, using a Gaussian distribution which covariance matrix is build using a symmetric function over an arbitrary space. Each entry in the covariance matrix is the value of this function for pairs of points in this space
4. **What's interesting here is that given some existing samples**, the posterior distribution for the value at a new point x is also a Gaussian distribution. S the mean of the posterior distribution at a varying point x can be considered to be a function defined over this arbitrary space, that implicitely interpolates the existing points.
5. **The value of that function is given by the following formula**, that involves the covariance between x and all other points in the parameter space, and t existing values at the known points.
6. **So let me rephrase this:** with this technique, we choose an arbitrary space, and some positions x_i in this space to which we assign the z_i values. This formula will turn this space into a parameterisation of the manifold that implicitly passes through our data points.
7. **In order to use this, we will consider BRDFs values** at a specific pair of directions to be the samples to interpolate.
8. **Therefore we pack all BRDFs** from the input database into a long matrix, where each line is a material and each column is independently interpolated b the Gaussian Process regression
9. **Doing so, we have constructed** an implicit manifold that interpolates BRDFs, and is parameterized in an arbitrary space.

Gaussian process regression

and we will apply this in parallel to all pairs of directions.



$$f(\mathbf{z}) = e^{-(\mathbf{z}-\mathbf{m})^{\mathbf{T}}\mathbf{K}^{-1}(\mathbf{z}-\mathbf{m})}$$

$$\mathbf{K}_{ij} = c(x_i, x_j)$$

↳ Covariance function

1. **You all know what a Gaussian distribution is**: it's a 1-dimensional probability distribution defined by its mean and variance, from which you can draw samples.
2. **A Gaussian distribution can be generalized** to a multi-dimensional vector space, which allows to draw multi-dimensional samples. What's interesting is that when I slice this distribution (in any direction), the posterior distribution is also a Gaussian distribution
3. **A Gaussian process is a generalization of this to arbitrary dimensions**: a vector of any dimension can be sampled from it, using a Gaussian distribution which covariance matrix is build using a symmetric function over an arbitrary space. Each entry in the covariance matrix is the value of this function for pairs of points in this space
4. **What's interesting here is that given some existing samples**, the posterior distribution for the value at a new point x is also a Gaussian distribution. S the mean of the posterior distribution at a varying point x can be considered to be a function defined over this arbitrary space, that implicitely interpolates the existing points.
5. **The value of that function is given by the following formula**, that involves the covariance between x and all other points in the parameter space, and t existing values at the known points.
6. **So let me rephrase this:** with this technique, we choose an arbitrary space, and some positions x_i in this space to which we assign the z_i values. This formula will turn this space into a parameterisation of the manifold that implicitly passes through our data points.
7. **In order to use this, we will consider BRDFs values** at a specific pair of directions to be the samples to interpolate.
8. **Therefore we pack all BRDFs** from the input database into a long matrix, where each line is a material and each column is independently interpolated b the Gaussian Process regression
9. **Doing so, we have constructed** an implicit manifold that interpolates BRDFs, and is parameterized in an arbitrary space.

The good properties of Gaussian processes

- exact interpolation is guarantied
- non linear w.r.t. reduced space, but **linear** w.r.t. the **training data**!
- extrapolation is very fast
- ...independently of $\{\mathbf{x}_i\}$, dimensionality, and $c(,)$

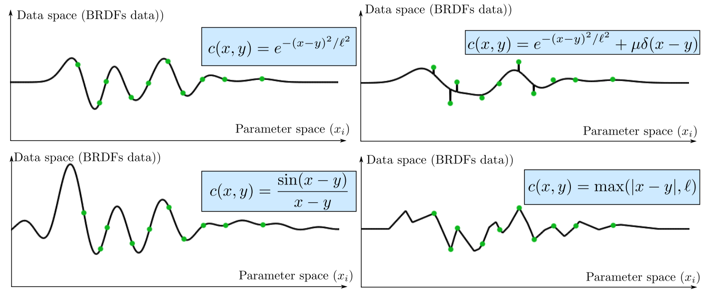
$$m(\mathbf{x}) = \quad \mathbf{k}(\mathbf{x})^{\mathbf{T}} \quad \underbrace{\mathbf{K}^{-1} \quad \mathbf{Z}}_{\text{Precomputed}}$$

1. **So let's look** at the properties we get from this. The equation to compute a new BRDF at point **x** is the following...
2. **What happens if x is one of the x**$_i$; well the covariance vector $\mathbf{k}^*$ becomes equal to one line of the covariance matrix **k**, so their product is a canonical vector, which when multiplied by the matrix of BRDFs returns exactly one of them. That means interpolation is guarrantied.
3. **The computed value does not linearly depend** on **x** because elements in vector **k** are computed using the covariance that is not linear. But $m(\mathbf{x})$ is line in Z, which means that whatever calculation we do that is a linear operation on the BRDFs can be interpolated using the same process.
4. **Finally, computation cost is low:** if we precompute the product between $\mathbf{K}^{-1}$ and **Z**, generating an interpolated point just needs to compute the covariance vector and multiply it with the precomputed matrix.
5. **And most of all:** all these properties are totally independent on the choice of the covariance function, the dimension of the parameter space and the positions of the $\mathbf{x}_i$ in that space

Latent variables and covariance function

- covariance mainly encodes the shape of the functions

$$c(\mathbf{x}_i, \mathbf{x}_j) = e^{-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{\ell^2}}$$



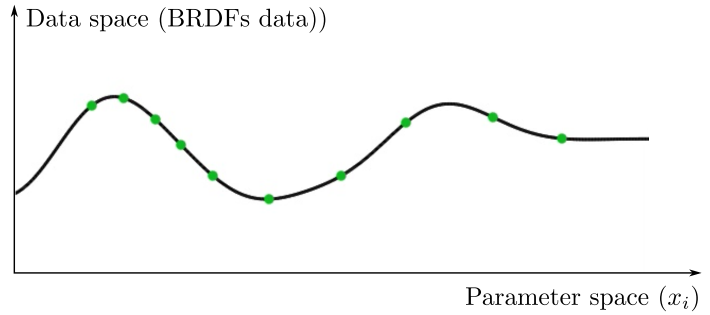
1. **The covariance function mainly encodes** the nature of the interpolant. The images below show the various shapes obtained from the same points usin different covariance functions. As you can see the nature of the covariance function can be guessed from the interpolant. Using a sync for instance will create an interpolant that keeps oscillating away from the samples.
2. **In our case, we want the interpolant to be as smooth as possible** in between samples, so we use a Gaussian. Adding a small delta to the covariance makes the interpolant non continuous, but increases the numerical stability.
3. **The "consistency" of the interpolant is very sensitive** to the placement of latent variables. We need therefore to optimize for the best choice of latent variables
4. **Fortunately the log-likelyhood** of the Gaussian process gives exactly the measure of this consistency. It's visible on the bottom left of the video.
5. **consequently, we optimise** the position of the $\mathbf{x}_i$ in order to maximize the log likelyhood, which gives us the smoothest possible manifold.
6. **Finally, we still ahve a scale parameter** in the covariance function. Because changing this scale is equivalent to scalign all the $\mathbf{x}_i$, we set it to an arbitrai value of 1

Latent variables and covariance function

- covariance mainly encodes the shape of the functions

$$c(\mathbf{x}_i, \mathbf{x}_j) = e^{-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{\ell^2}}$$

- points needs to be spaced according to covariance
  $\rightarrow$ optimize for point placement (Hooke-Jeeves)



1. **The covariance function mainly encodes** the nature of the interpolant. The images below show the various shapes obtained from the same points usin different covariance functions. As you can see the nature of the covariance function can be guessed from the interpolant. Using a sync for instance will create an interpolant that keeps oscillating away from the samples.
2. **In our case, we want the interpolant to be as smooth as possible** in between samples, so we use a Gaussian. Adding a small delta to the covariance makes the interpolant non continuous, but increases the numerical stability.
3. **The "consistency" of the interpolant is very sensitive** to the placement of latent variables. We need therefore to optimize for the best choice of latent variables
4. **Fortunately the log-likelyhood** of the Gaussian process gives exactly the measure of this consistency. It's visible on the bottom left of the video.
5. **consequently, we optimise** the position of the $\mathbf{x}_i$ in order to maximize the log likelyhood, which gives us the smoothest possible manifold.
6. **Finally, we still ahve a scale parameter** in the covariance function. Because changing this scale is equivalent to scalign all the $\mathbf{x}_i$, we set it to an arbitra value of 1

Latent variables and covariance function

- covariance mainly encodes the shape of the functions

$$c(\mathbf{x}_i, \mathbf{x}_j) = e^{-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{\ell^2}}$$

- points needs to be spaced according to covariance
  $\rightarrow$ optimize for point placement (Hooke-Jeeves)

$$L = -d \log|\mathbf{K}| - \mathrm{tr}(\mathbf{K}^{-1}\mathbf{Z}\mathbf{Z}^{\mathbf{T}})$$

1. **The covariance function mainly encodes** the nature of the interpolant. The images below show the various shapes obtained from the same points usin different covariance functions. As you can see the nature of the covariance function can be guessed from the interpolant. Using a sync for instance will create an interpolant that keeps oscillating away from the samples.
2. **In our case, we want the interpolant to be as smooth as possible** in between samples, so we use a Gaussian. Adding a small delta to the covariance makes the interpolant non continuous, but increases the numerical stability.
3. **The "consistency" of the interpolant is very sensitive** to the placement of latent variables. We need therefore to optimize for the best choice of latent variables
4. **Fortunately the log-likelyhood** of the Gaussian process gives exactly the measure of this consistency. It's visible on the bottom left of the video.
5. **consequently, we optimise** the position of the $\mathbf{x}_i$ in order to maximize the log likelyhood, which gives us the smoothest possible manifold.
6. **Finally, we still ahve a scale parameter** in the covariance function. Because changing this scale is equivalent to scalign all the $\mathbf{x}_i$, we set it to an arbitrai value of 1

Latent variables and covariance function

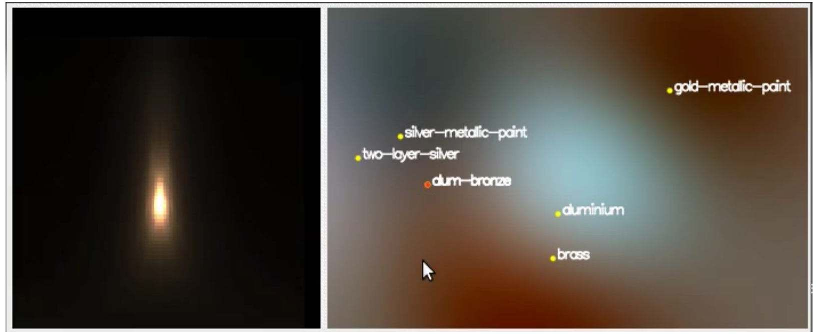
- covariance mainly encodes the shape of the functions

$$c(\mathbf{x}_i, \mathbf{x}_j) = e^{-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{\ell^2}}$$

- points needs to be spaced according to covariance
  $\rightarrow$ optimize for point placement (Hooke-Jeeves)
- optimal ℓ and x_i linked with a scale $\Rightarrow$ fix $\ell = 1$

1. **The covariance function mainly encodes** the nature of the interpolant. The images below show the various shapes obtained from the same points usin different covariance functions. As you can see the nature of the covariance function can be guessed from the interpolant. Using a sync for instance will create an interpolant that keeps oscillating away from the samples.
2. **In our case, we want the interpolant to be as smooth as possible** in between samples, so we use a Gaussian. Adding a small delta to the covariance makes the interpolant non continuous, but increases the numerical stability.
3. **The "consistency" of the interpolant is very sensitive** to the placement of latent variables. We need therefore to optimize for the best choice of latent variables
4. **Fortunately the log-likelyhood** of the Gaussian process gives exactly the measure of this consistency. It's visible on the bottom left of the video.
5. **consequently, we optimise** the position of the $\mathbf{x}_i$ in order to maximize the log likelyhood, which gives us the smoothest possible manifold.
6. **Finally, we still ahve a scale parameter** in the covariance function. Because changing this scale is equivalent to scalign all the $\mathbf{x}_i$, we set it to an arbitrai value of 1

Does it work??



Live capture

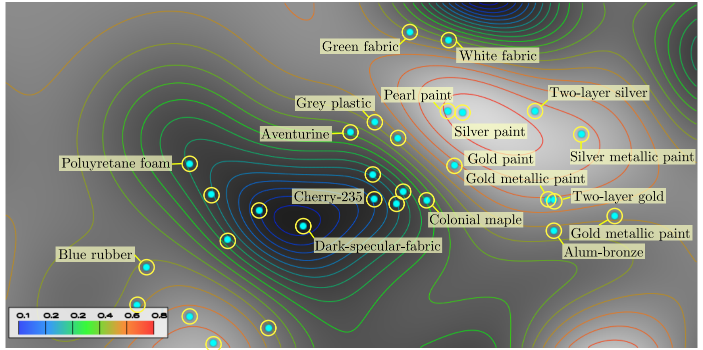
1. **So, does it work??** Yes it does. You can see the parameter space here at right and the interpolated BRDF at left. At you can see it varies smoothly in between samples and does not produce anything fancy far from the samples either.

- motivation
- gaussian process regression
- evaluation
- applications: leveraging linearity

Physical correctness / stability

Interpolation reproduces the input data exactly. But what happens in between the $\mathbf{x}_i$?

- preserves reciprocity
- interpolates albedo
- maximized log-likelyhood ensures stability (See paper)

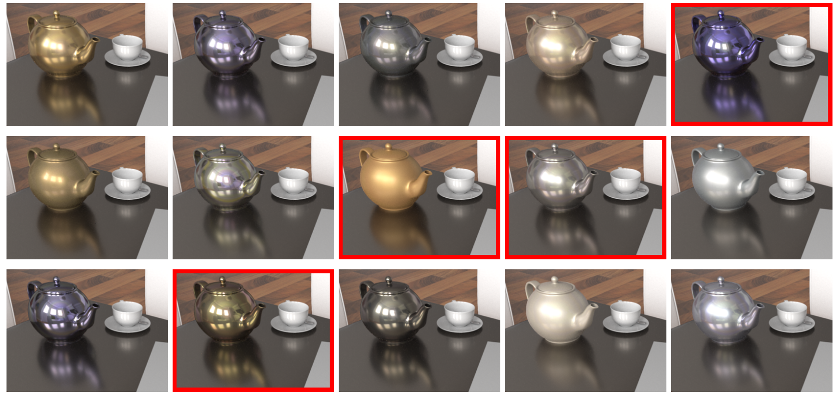


Albedo map

1. **So, it looks ok, but do** the interpolated BRDFs behave like real BRDFs?
2. **First, our model by construction preserves** reciprocity and interpolates the albedo
3. **We also show in the paper** and supplemental that the effect of maximizing the log likelihood is precisely to keep the interpolant from doing fancy things between the input samples

Plausibility

- plausibility experiment

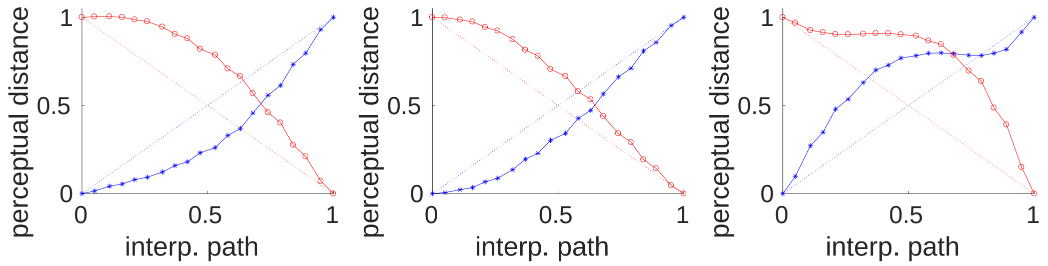


11 points from our manifold + 4 actual MERL BRDFs, rendered with Mitsuba

1. **Next, we wonder about the plausibility** of the interpolant: does it look like a real BRDF?
2. **here we rendered 15 images** with Mitsuba, 11 of which use materials picked in our manifold, and the remaining 4 actually correspond to real MERL data Can you guess which?
3. **The answer is** that all these are not real materials, yet most of them look very realistic
4. **We also performed a more serious** experiment in which we compare the perceptual distance developped by Pereira and Ruzinkiewicz when interpolati two BRDFs in the manifold. We do this by linearly interpolating the coordinates in the low dimension space and we compare the perceptual distance to th euclidian distance in the parameter space
5. **It works reasonnably well,** provided by the parameter space is of dimension large enough to allow the optimizer to cleverly cluster the various materials according to both color, shininess and material type

Plausibility

- plausibility experiment
- preservation of perceptual distance
  - ...when interpolating in parameter space
  - works best if space dimension is >= 5



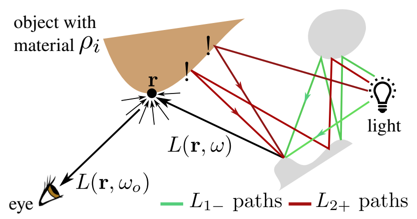
(using Pereira & Rusinkiewicz'2012 similarity metric)

1. **Next, we wonder about the plausibility** of the interpolant: does it look like a real BRDF?
2. **here we rendered 15 images** with Mitsuba, 11 of which use materials picked in our manifold, and the remaining 4 actually correspond to real MERL data Can you guess which?
3. **The answer is** that all these are not real materials, yet most of them look very realistic
4. **We also performed a more serious** experiment in which we compare the perceptual distance developped by Pereira and Ruzinkiewicz when interpolatir two BRDFs in the manifold. We do this by linearly interpolating the coordinates in the low dimension space and we compare the perceptual distance to th euclidian distance in the parameter space
5. **It works reasonnably well,** provided by the parameter space is of dimension large enough to allow the optimizer to cleverly cluster the various materials according to both color, shininess and material type

- motivation
- gaussian process regression
- evaluation
- applications: leveraging linearity

Application: interpolating materials in GI images

- use the near-linearity between pixels and materials
- approximation error is hardly noticable

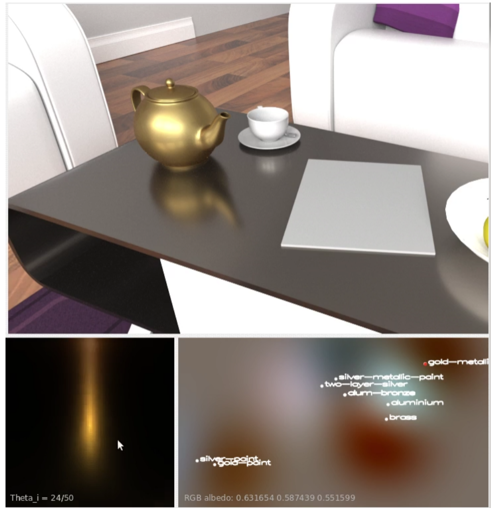


$$I = \underbrace{I_1}_{affine} + I_2$$

$$I_1 = \mathbf{N}\mathbf{z} + P$$

1. **Now I'm switching to applications** where we use this nice linearity property of Gaussian Processes.
2. **For instance, when doing Monte-Carlo global illumination**, you may consider that the image is made of two components: the one for which light paths that touch a given object at most once, and the one where light paths that touch this object more than once.
3. **The first part which we call** I_1 here is afine w.r.t. the material on the object, and it also represents in most situations the main part of the energy in the image.
4. **that means we can interpolate** global illumination images with a Gaussian Process when changing one of the materials in the scene with that same Gaussian Process
5. **Here is what is looks like** in a live captured session. You can see that when I'm browsing the manifold, the reflection of the teapot updates as expected
6. **This is only for static images**. Can we do that on dynamic geometry and lighting? The answer is yes

Application: interpolating materials in GI images



1. **Now I'm switching to applications** where we use this nice linearity property of Gaussian Processes.
2. **For instance, when doing Monte-Carlo global illumination**, you may consider that the image is made of two components: the one for which light paths that touch a given object at most once, and the one where light paths that touch this object more than once.
3. **The first part which we call** I_1 here is afine w.r.t. the material on the object, and it also represents in most situations the main part of the energy in the image.
4. **that means we can interpolate** global illumination images with a Gaussian Process when changing one of the materials in the scene with that same Gaussian Process
5. **Here is what is looks like** in a live captured session. You can see that when I'm browsing the manifold, the reflection of the teapot updates as expected
6. **This is only for static images**. Can we do that on dynamic geometry and lighting? The answer is yes

Application to real time shading (1/2)

- compute shading of tabulated BRDFs in real time
  ... using a basis of rotated zonal harmonics [Soler'2015]

$$I(p,\omega_o) = \int_S \rho(\mathbf{R}_n\omega_o, \mathbf{R}_n\omega)E(\omega)\cos\theta d\omega$$

using

$$\rho(\omega_o,\omega) = \sum_{l,m} \lambda_l^m(\omega_o)\mathcal{Y}_l^0(\mathbf{R}_m^{-1}\omega)$$

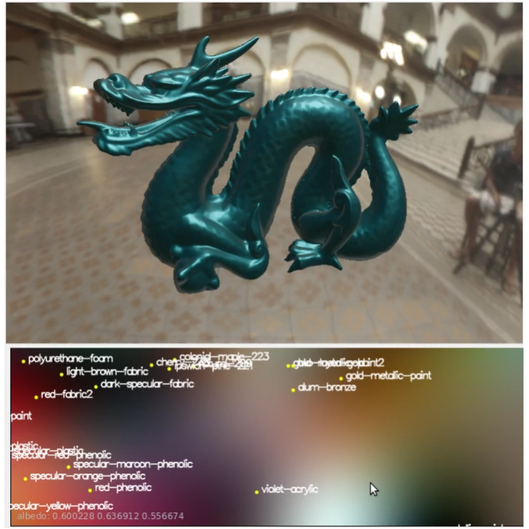
$$\lambda = \mathbf{M}\rho$$

- directly apply Gaussian Process Regression to the λ_l^m
- send to the shader

$\Rightarrow$ RT rendering of interpolated BRDFs, with free view+geometry+lighting!

1. **We recently produced** a technique that allows to render measured materials in real time
2. **in this technique we basically compute** the shading equation on the GPU, after projecting the BRDF onto a basis of rotated zonal harmonics
3. **but because the coefficients of a material** in this basis depend linearly on the material data, we can compute these coefficients instantly for any point i the manifold by simply interpolating them with the gaussian process
4. **Doing so, we render the interpolated material** exactly by supplying the interpolated coefficients to the GPU shader that does the rendering
5. **this is what it looks like** on the full MERL database. This kind of application allows to browse BRDFs and experiment the look of the BRDF which changing the geometry and lighting in real time.
6. **the interpolation here is done on the CPU** and the renderng is done on the GPU

Application to real time shading (2/2)



Future work

What we haven't tried yet:

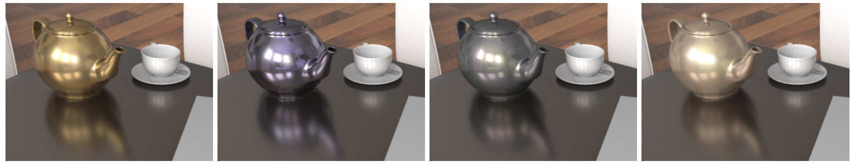
- interpolation between isotropic and anisotropic BRDFs
  $\longrightarrow$ should work out of the box

Ongoing work:

- single shot BRDF capture
- material replacement in images (requires a consistent illumination/normals pair)
  - handling of visibility (totally possible)
  - using consistency constraints in optimization

Questions

Thank you for your attention!



Funding:

Project “CaLiTrOp” ANR-16-CE33-0026

Royal Society’s University Research Fellowship