Gaussian Material Synthesis

KÁROLY ZSOLNAI-FEHÉR, TU Wien

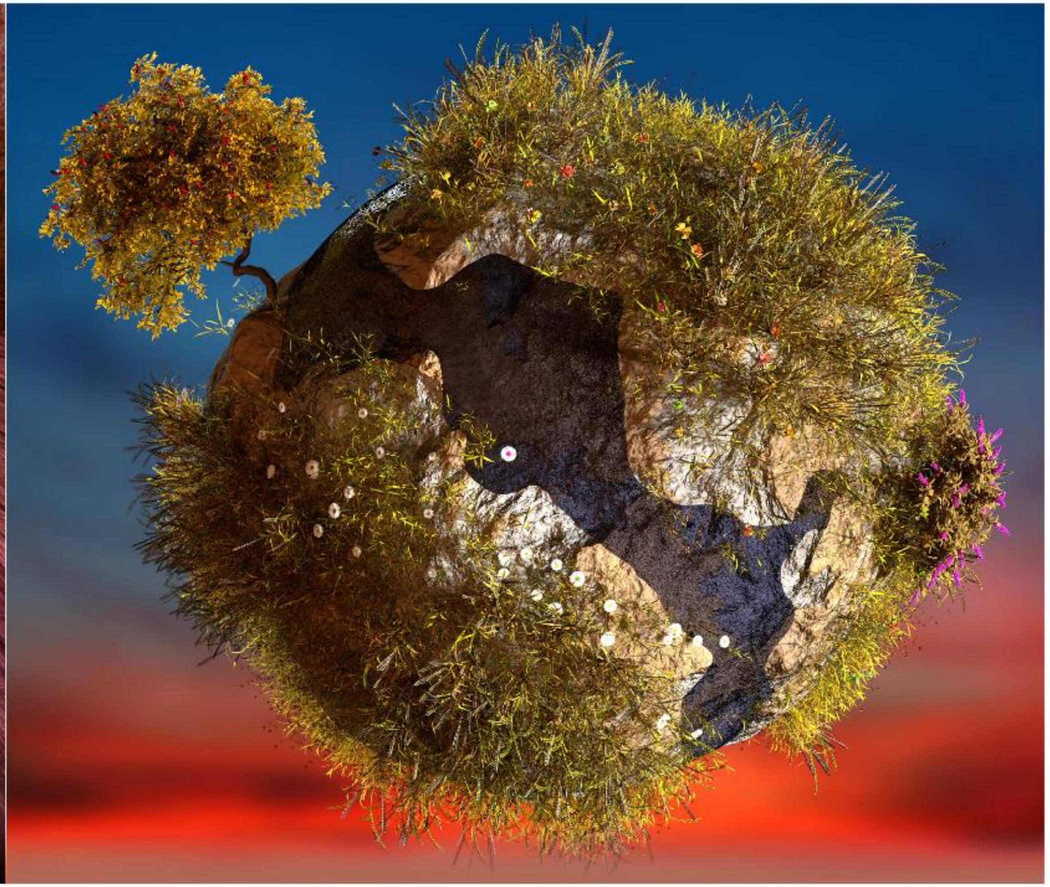
PETER WONKA, KAUST

MICHAEL WIMMER, TU Wien

ACM Transactions on Graphics, Publication date: August 2018

Presenter: MinKu Kang

Material Synthesis

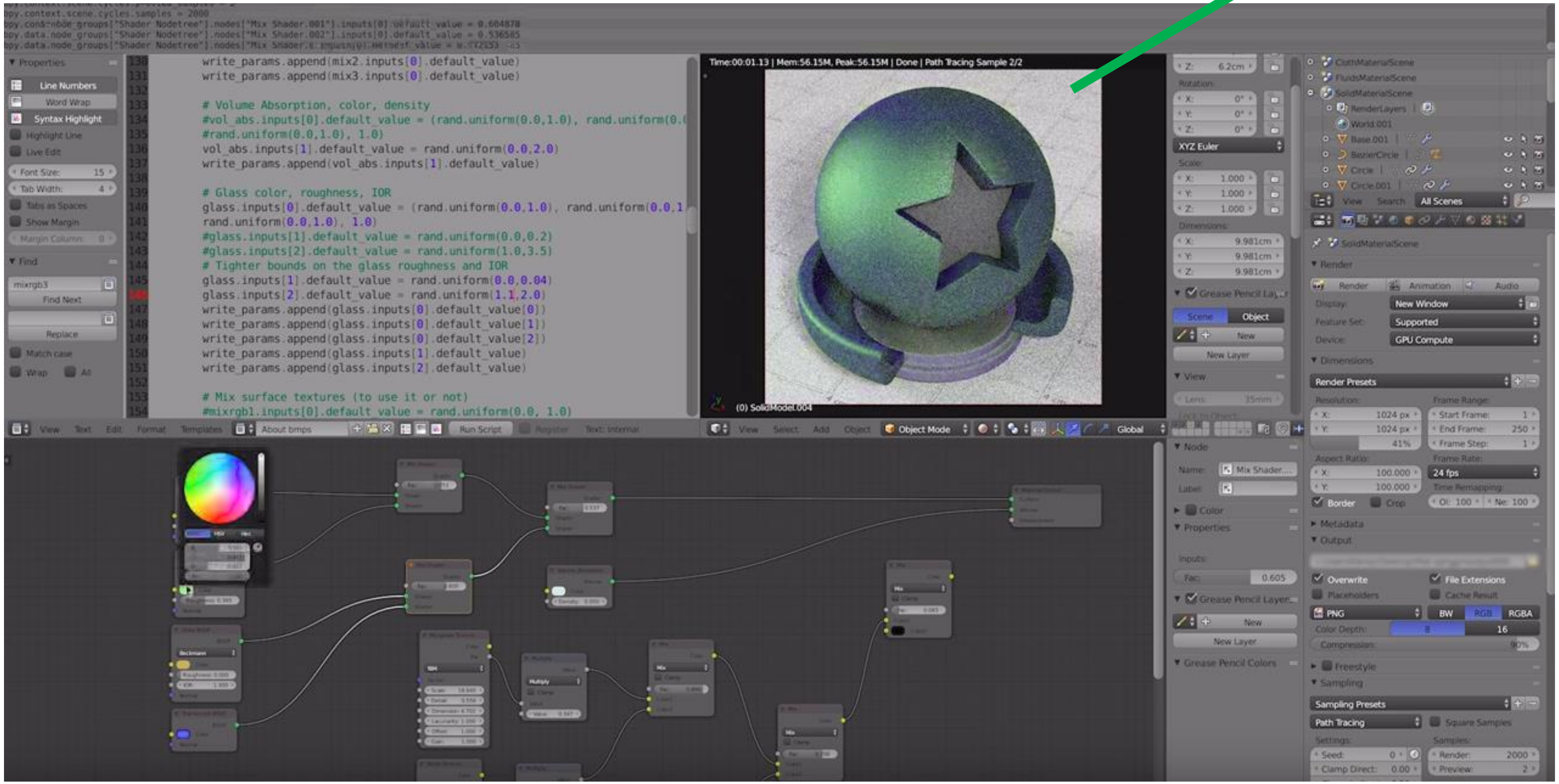


a scene with metals and minerals, translucent, glittery and glassy materials

more than a hundred synthesized materials and objects for the vegetation of the planet

Manual Material Synthesis is **Labor-intensive**

It takes time to render.

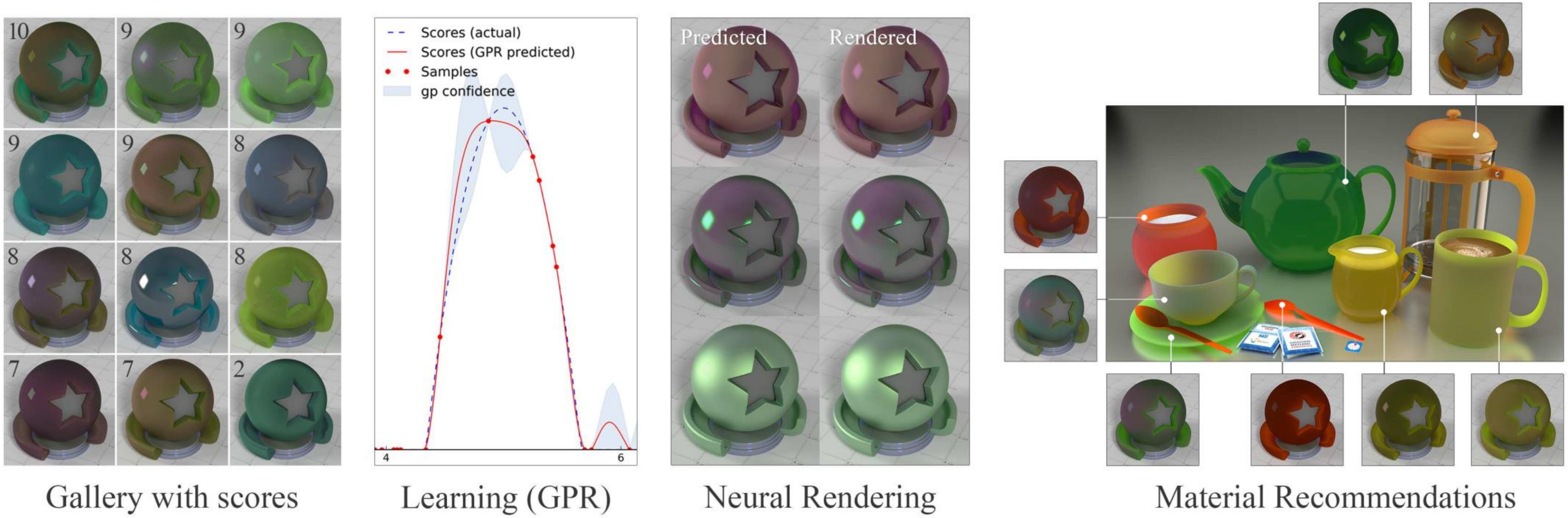


It might be a well fit for an expert, but ...

$\mathbf{x}_i \in \mathbb{R}^m$ Many parameters to tune.

From Authors Video: https://www.youtube.com/watch?v=6FzVhIV_t3s

Overview



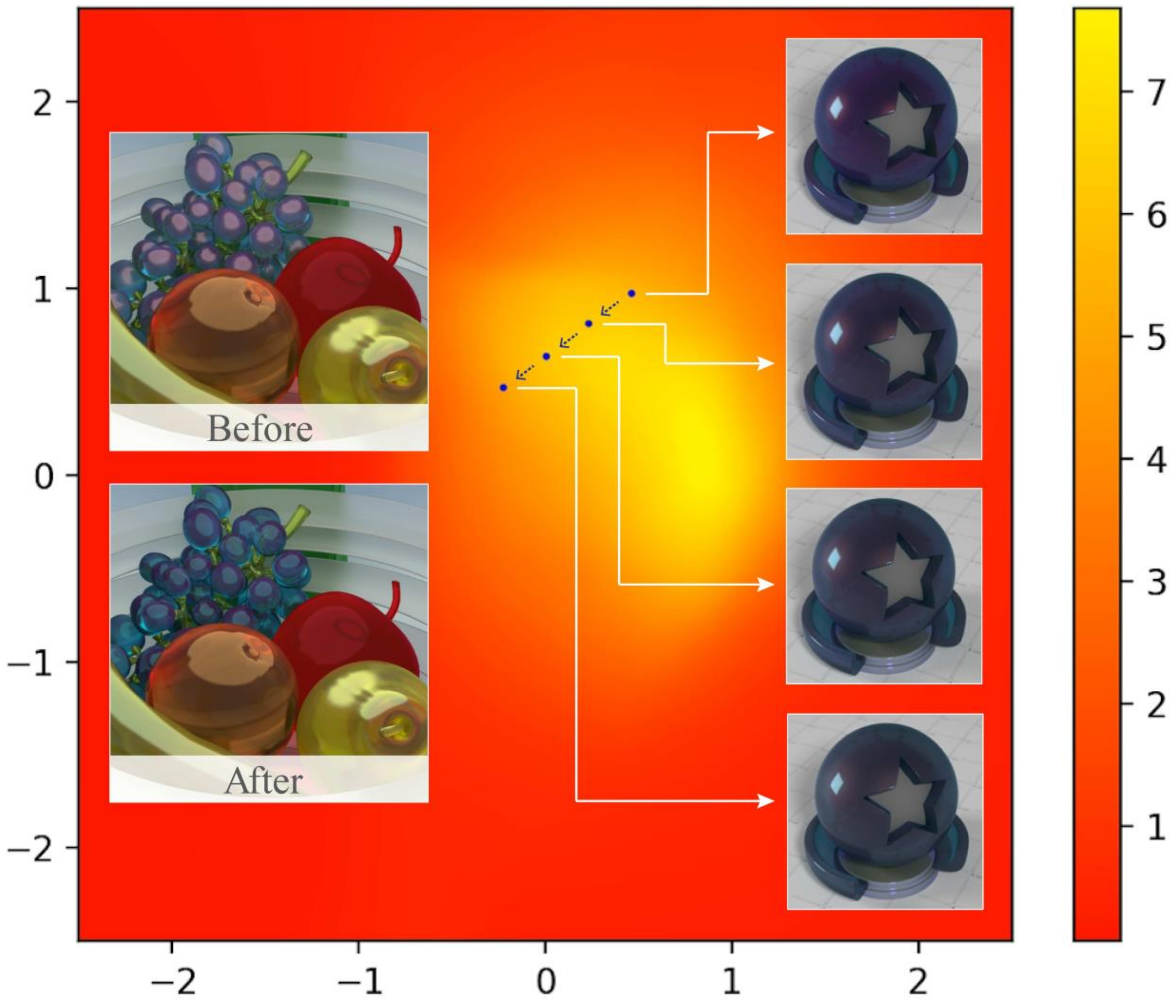
- Rapid mass-scale material synthesis for **novice** and expert users.
- This method takes a set of user **preferences** as an input.
- It **recommends** relevant new materials from the learned distributions.

Stage 1: A User Scores a Gallery

From Authors Video: https://www.youtube.com/watch?v=6FzVhIV_t3s

Stage 2: Recommendations are Generated

(Optional) Stage 3: Fine-searching in Latent Space



Stage 4: Applying materials to a Scene

Notations

Symbol	Description	Type
$\mathbf{x}$	BSDF description	Vector
$u^*(\mathbf{x})$	Preference function (Ground truth)	Scalar
$u(\mathbf{x})$	Preference function (GPR prediction)	Scalar
n	Number of GPR samples	Scalar
$\mathbf{x}^*$	Unknown BSDF test input	Vector

$$\mathbf{x}_i \in \mathbb{R}^m$$

A parameter space similar to Disney's **principled shader** that comes in two versions:

- **m = 19** variant spans the most commonly used materials,
i.e., a combination of diffuse, specular, glossy, transparent and translucent materials

- **m = 38** version additionally supports procedurally textured albedos and displacements

Gaussian Process Regression (GPR)

$$k(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \exp\left[-\frac{(\mathbf{x}-\mathbf{x}')^2}{2l^2} \right] + \beta^{-1}\delta_{xx'}$$

$$\mathbf{K} = \begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & k(\mathbf{x}_1, \mathbf{x}_2) & \dots & k(\mathbf{x}_1, \mathbf{x}_n) \\ k(\mathbf{x}_2, \mathbf{x}_1) & k(\mathbf{x}_2, \mathbf{x}_2) & \dots & k(\mathbf{x}_2, \mathbf{x}_n) \\ \vdots & \vdots & \ddots & \vdots \\ k(\mathbf{x}_n, \mathbf{x}_1) & k(\mathbf{x}_n, \mathbf{x}_2) & \dots & k(\mathbf{x}_n, \mathbf{x}_n) \end{bmatrix}$$

$$\mathbf{k}_* = \left[k(\mathbf{x}^*, x_1), k(\mathbf{x}^*, x_2), \dots, k(\mathbf{x}^*, x_n) \right]^T$$

Kernel function,
Kernel matrix

Joint distribution over scores

$$\begin{bmatrix} \mathbf{U} \\ u(\mathbf{x}^*) \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} \mathbf{K} & \mathbf{k}_*^T \\ \mathbf{k}_* & k_{**} \end{bmatrix} \right)$$

Conditional distribution over scores given **observations**

$$P(u(\mathbf{x}^*) \mid \mathbf{U})$$

test input

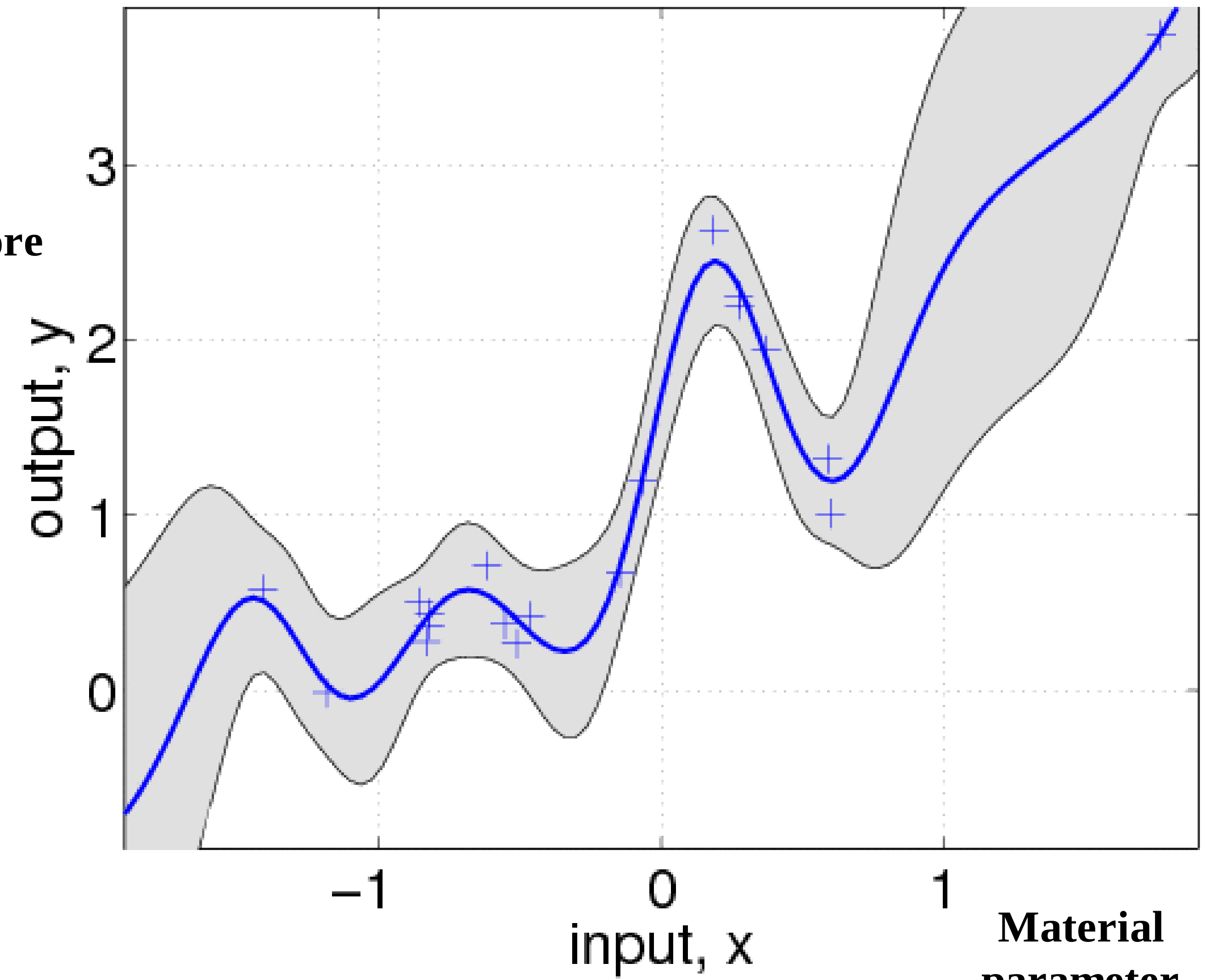
observations

$$u(\mathbf{x}^*) = \mathbf{k}_*^T \mathbf{K}^{-1} \mathbf{U},$$

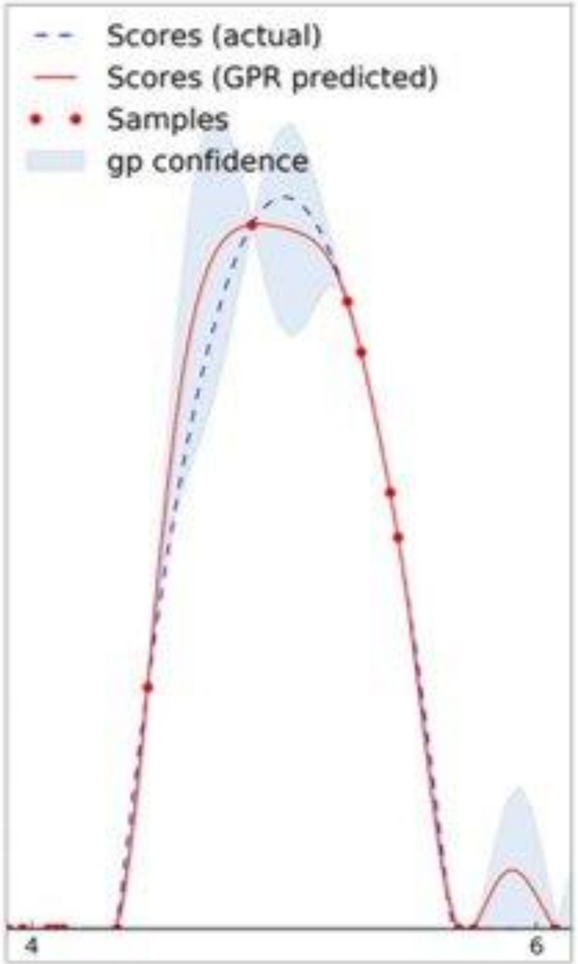
$$\sigma(u(\mathbf{x}^*)) = k_{**} - \mathbf{k}_* \mathbf{K}^{-1} \mathbf{k}_*^T$$

Gaussian Process Regression (GPR)

Score



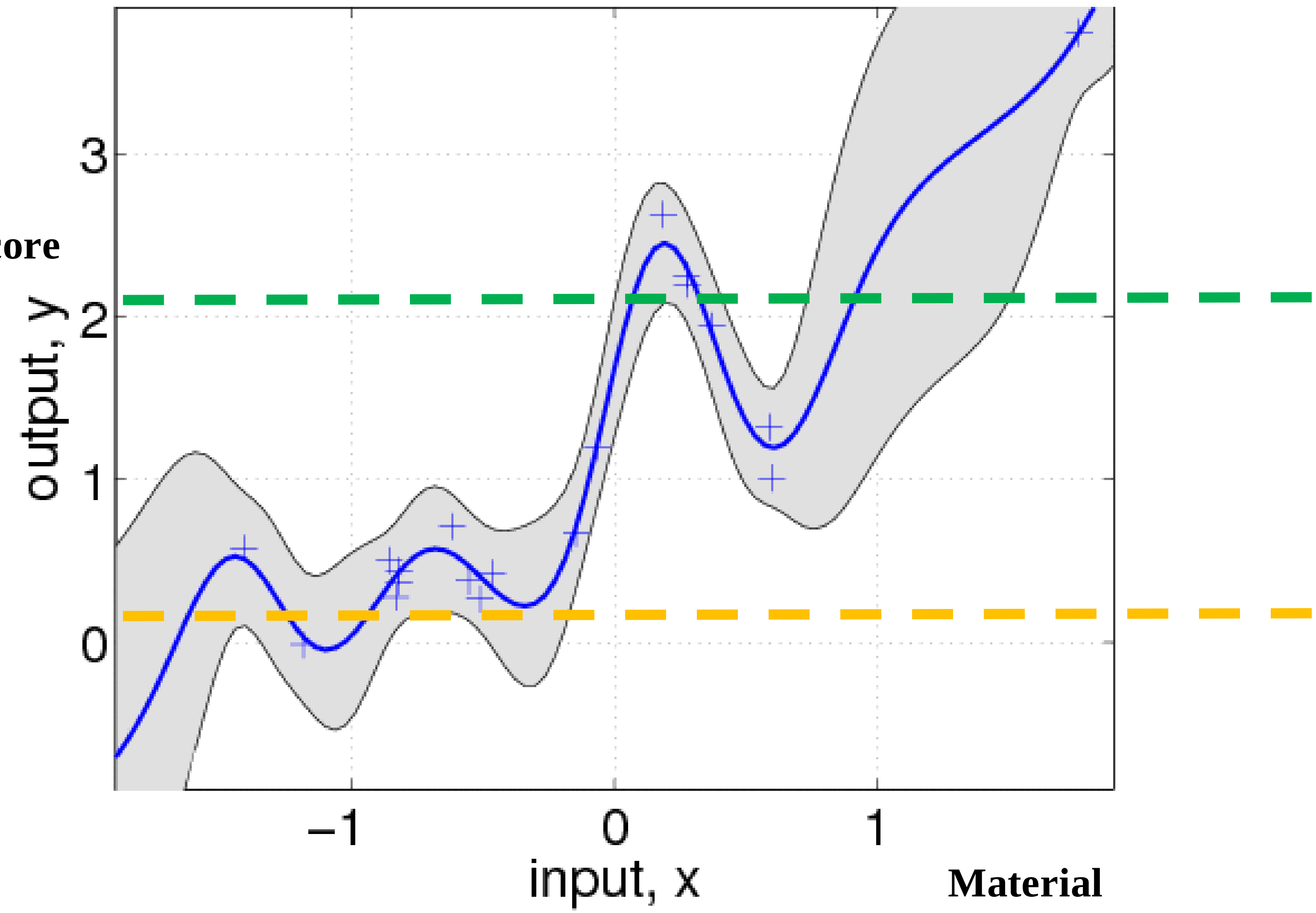
Material parameter



Learning (GPR)

Generating Recommendations

Score

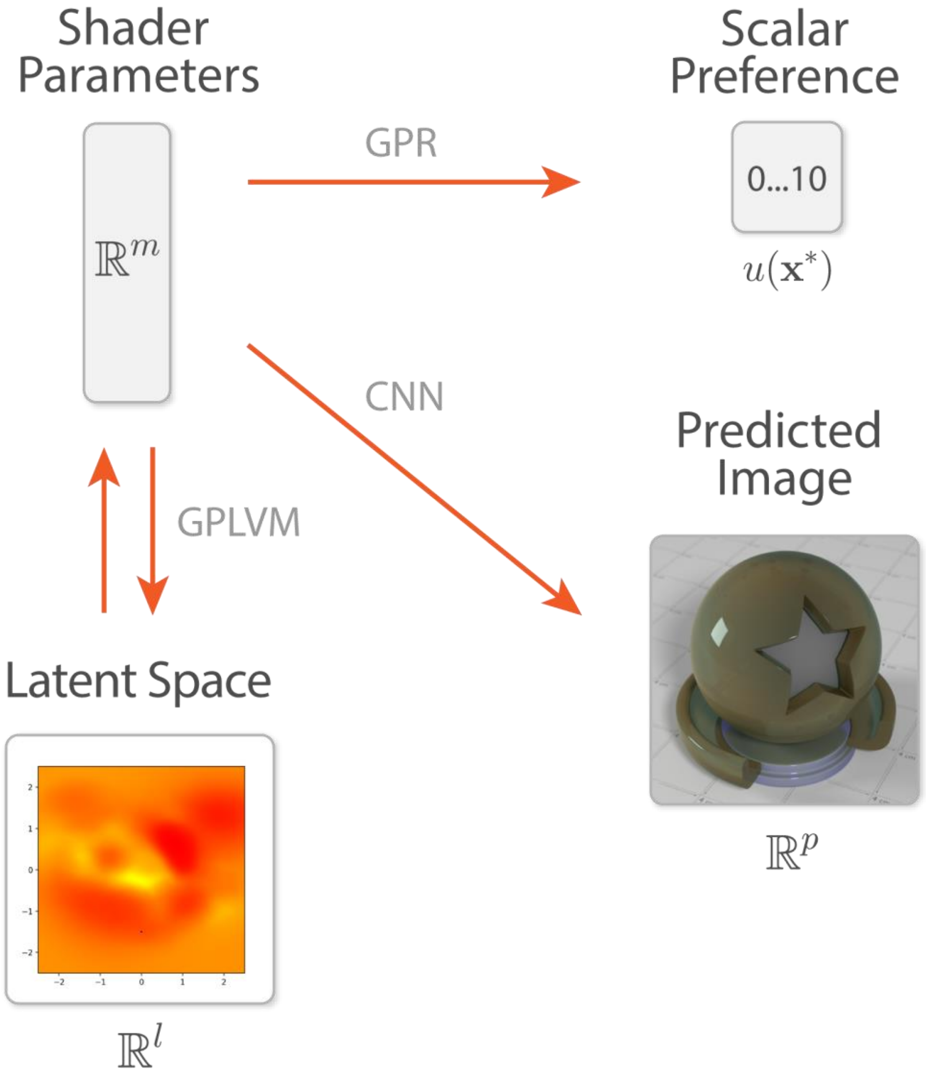


Material parameter

A high **threshold** will result in high-quality recommendations at the cost of decreased variety.

A larger variety of recommendations can be enforced by lowering the target **threshold**.

System Overview



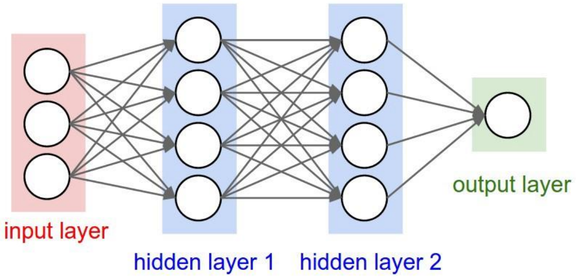
1. **GPR** is used to learn the user-specified material preferences
2. The system **recommends** new **materials** with **visualization**

3. Optionally, GPLVM can be used to provide an intuitive 2D space for variant generation.

Why GPR over other methods ?

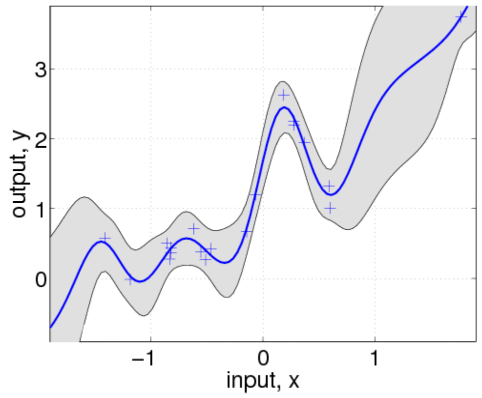
observations is quite small (= # material-score pairs labeled by the user), which is in the order of a few of tens.

Parametric Model



Data is absobed into the weights: new prediction is affected by the estimated parameter. It requires **many samples for an accurate parameter estimation**

Non-parametric Model



'**Let the data speaks**'
: new prediction is highly affected by the past observations

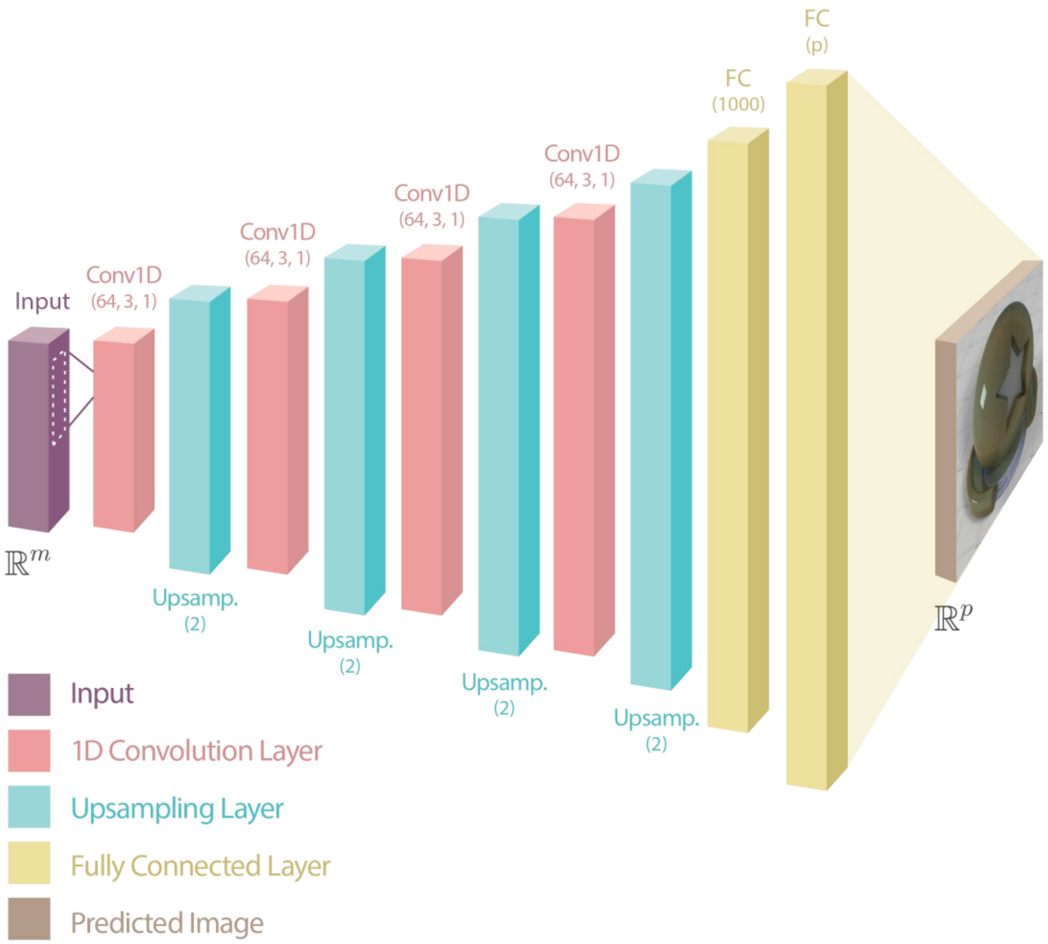
$$\mathbf{K} = \begin{bmatrix} k(\mathbf{x}_1,\mathbf{x}_1) & k(\mathbf{x}_1,\mathbf{x}_2) & \dots & k(\mathbf{x}_1,\mathbf{x}_n) \\ k(\mathbf{x}_2,\mathbf{x}_1) & k(\mathbf{x}_2,\mathbf{x}_2) & \dots & k(\mathbf{x}_2,\mathbf{x}_n) \\ \vdots & \vdots & \ddots & \vdots \\ k(\mathbf{x}_n,\mathbf{x}_1) & k(\mathbf{x}_n,\mathbf{x}_2) & \dots & k(\mathbf{x}_n,\mathbf{x}_n) \end{bmatrix}$$

$$\mathbf{k}_* = \left[k(\mathbf{x}^*,x_1), k(\mathbf{x}^*,x_2), \dots, k(\mathbf{x}^*,x_n)\right]^T$$

$$u(\mathbf{x}^*) = \mathbf{k}_*^T \mathbf{K}^{-1} \mathrm{U},$$

$$\sigma(u(\mathbf{x}^*)) = k_{**} - \mathbf{k}_* \mathbf{K}^{-1} \mathbf{k}_*^T$$

Neural Networks and Rendering



$$\phi: \mathbb{R}^m \rightarrow \mathbb{R}^p$$

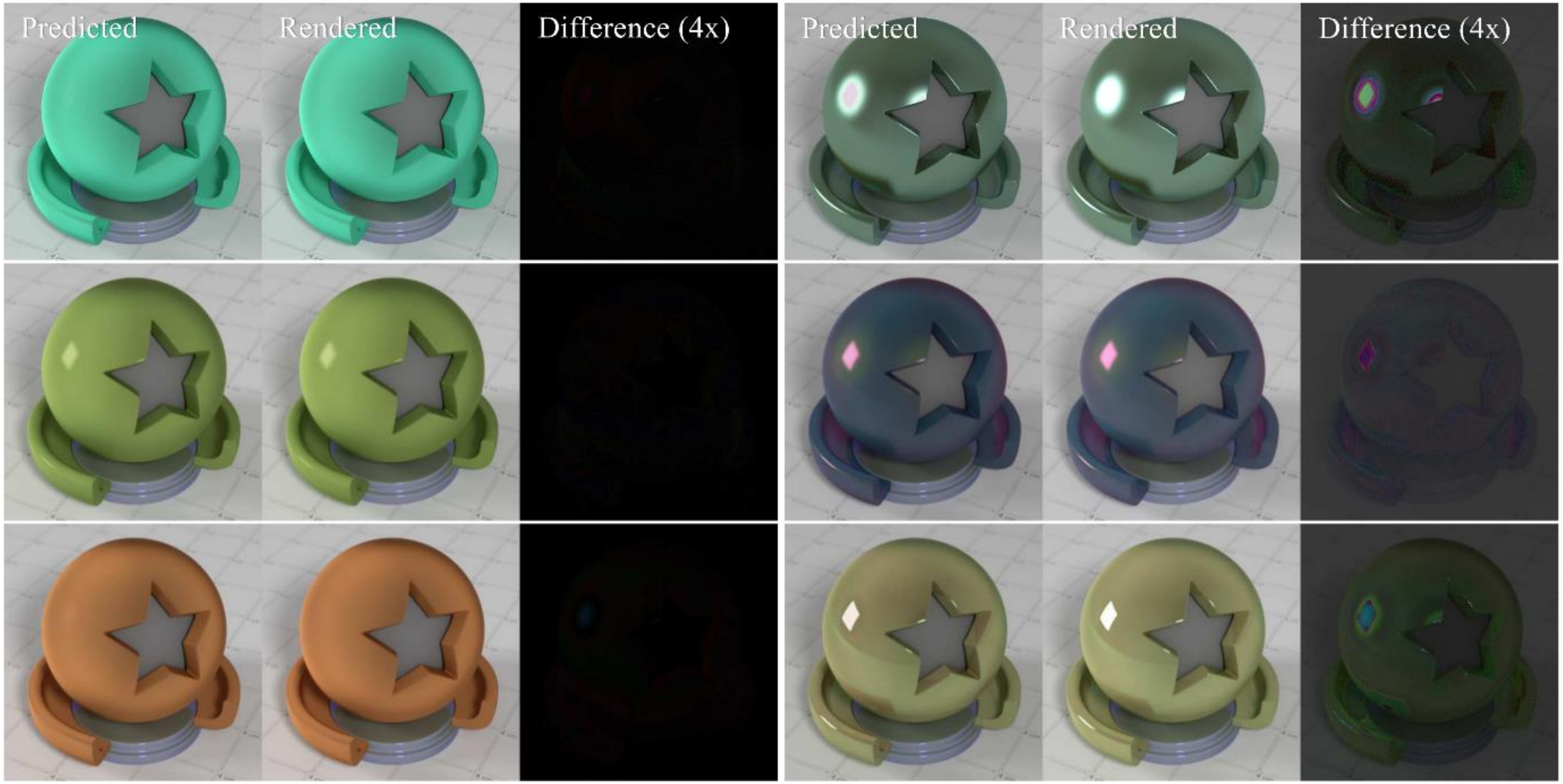
An atypical setting where the input shader dimensionality is orders of magnitude smaller than the output

m: order of tens
p: 410 by 410

Training set: 45000 shader-image pairs (250 spp)
Training time: over **4 weeks** on a consumer system with a NVIDIA GeForce GTX TITAN X GPU

This architecture is similar to the decoder part of Convolutional Autoencoders

Neural Networks and Rendering

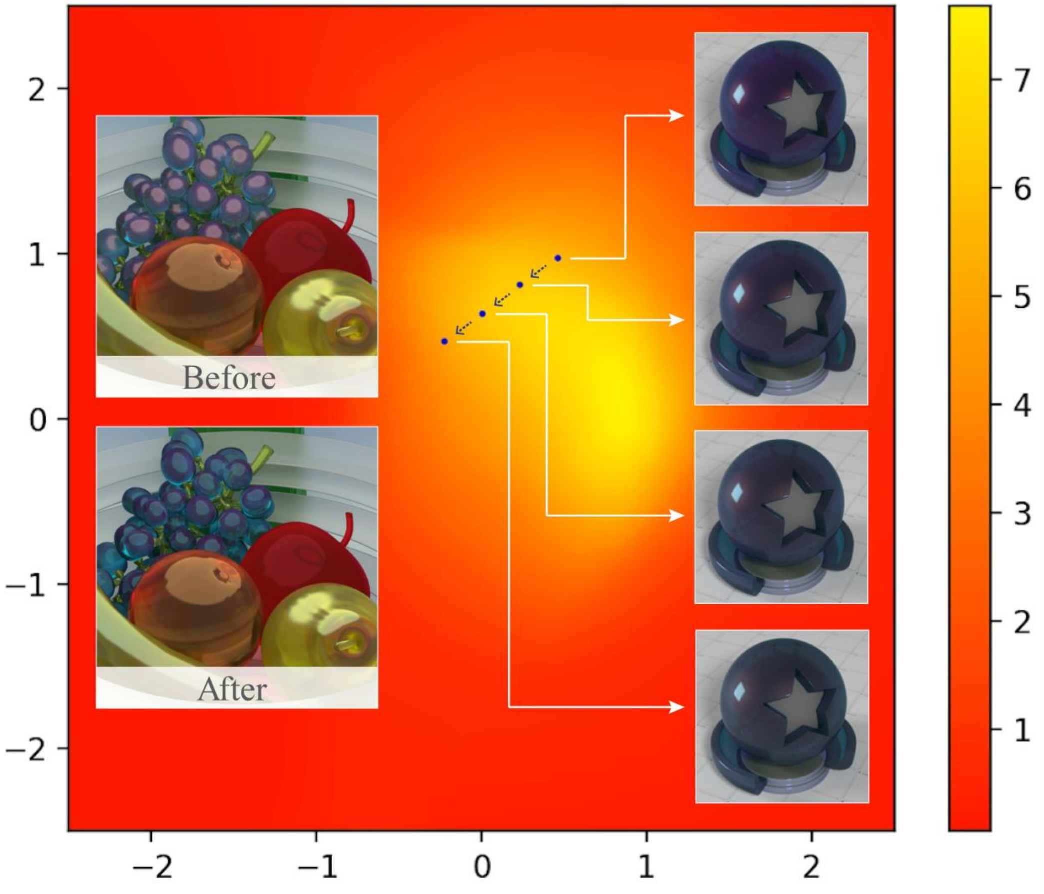


Best-case prediction

Worst-case prediction

~ 3 ms ~ 60 sec. (GI)

Latent Space Exploration



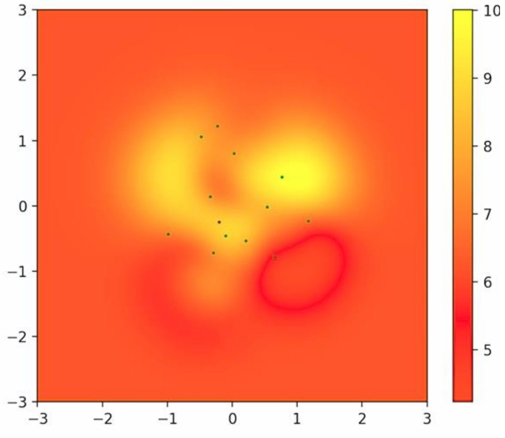
Low-dimensional latent space (m=2), with color representing score. GPLVM (Gaussian Process Latent Variable Model)

A few tens of high-scoring materials from the gallery are embedded into the low-dimensional latent space.

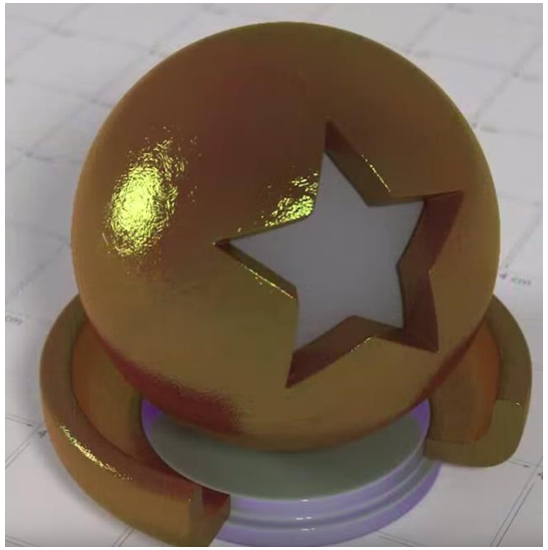
$$\mathbf{X} = \left[\cdots \mathbf{x}_i \cdots\right]^T \text{ with } \mathbf{x}_i \in \mathbb{R}^m$$

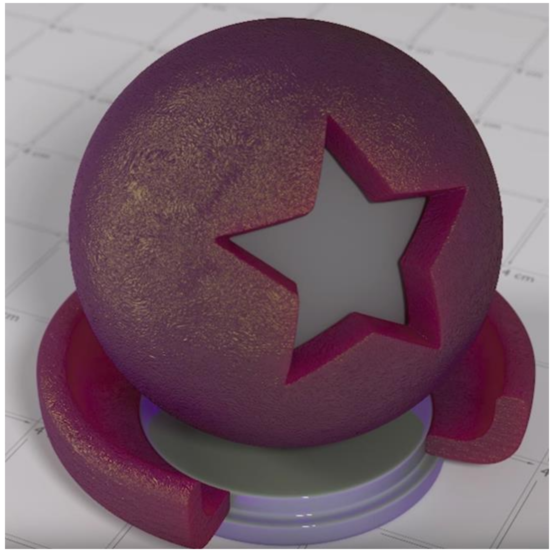
$$\mathbf{L} = \left[\cdots \mathbf{l}_i \cdots\right]^T \text{ with } \mathbf{l}_i \in \mathbb{R}^l$$

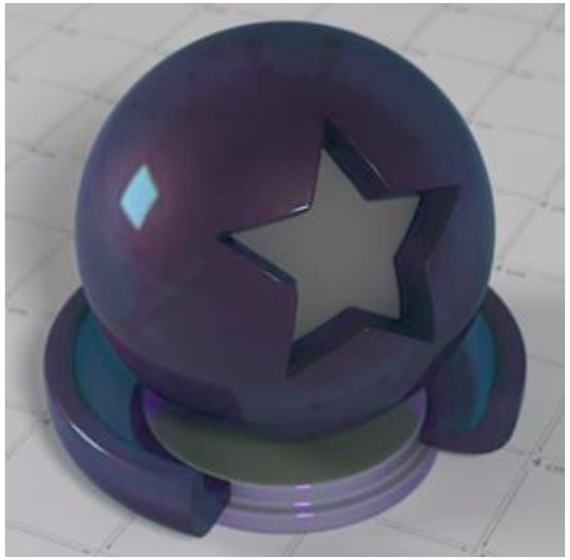
$$m \gg l$$



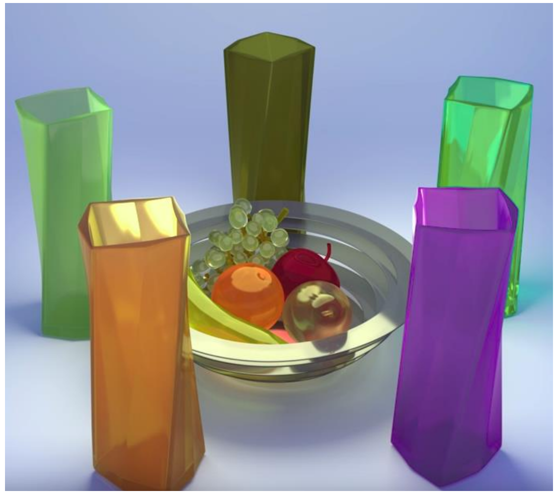
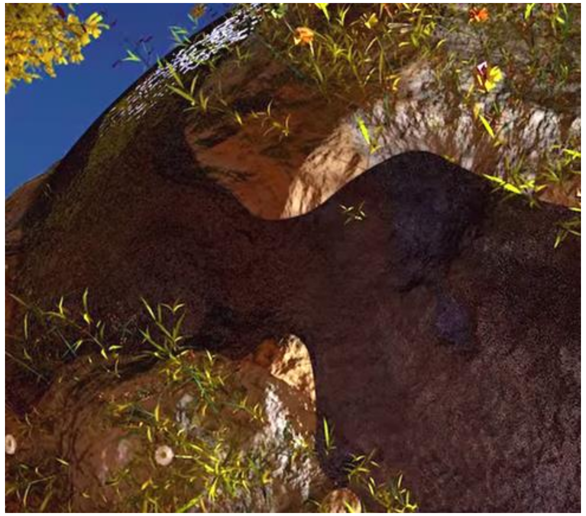
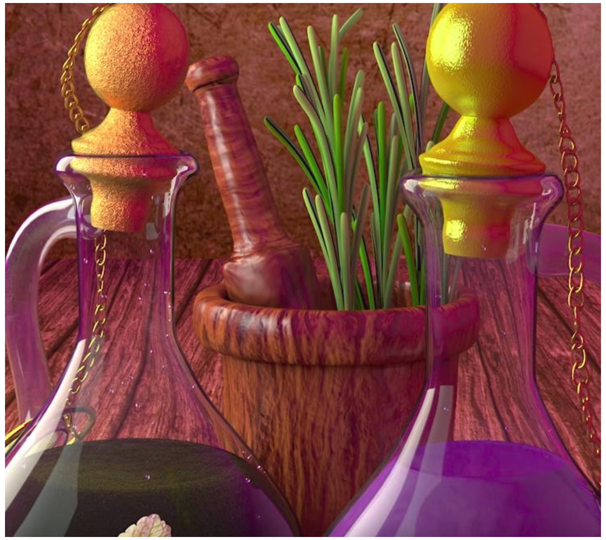
Resultant Materials (fine-tuned)



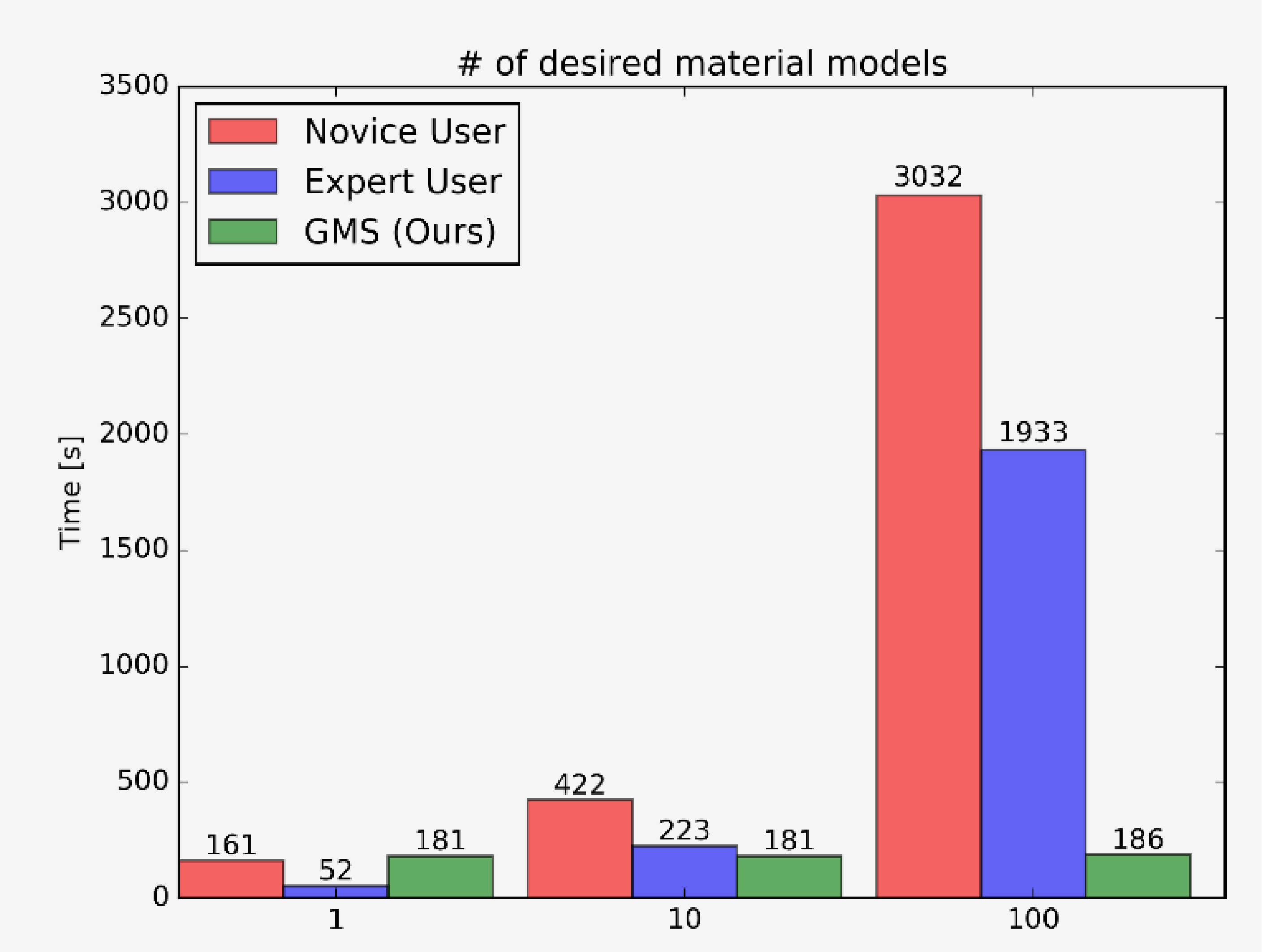




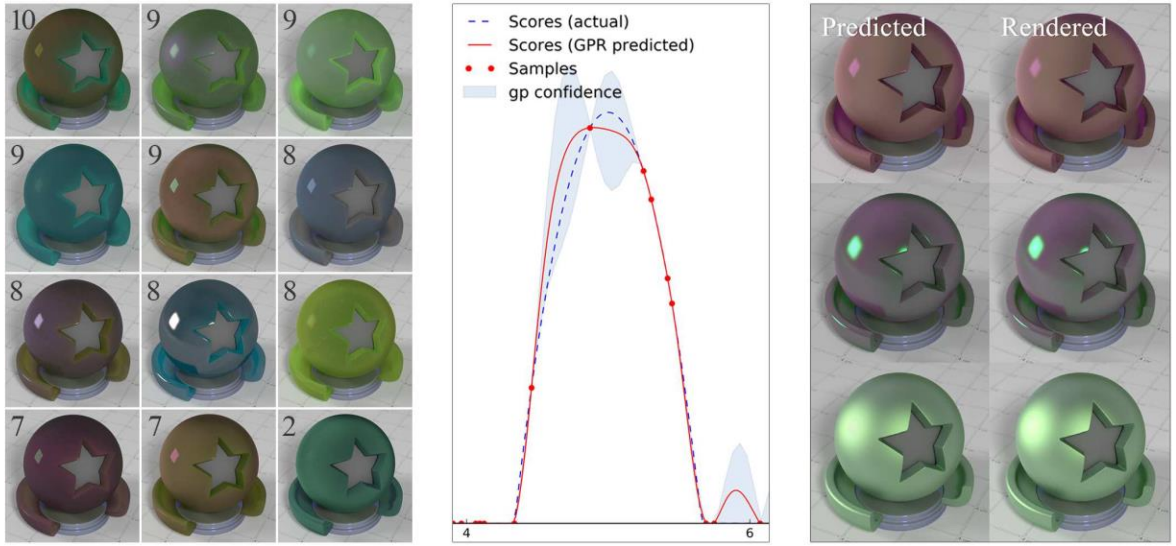
Resultant Materials (Application to Scenes)

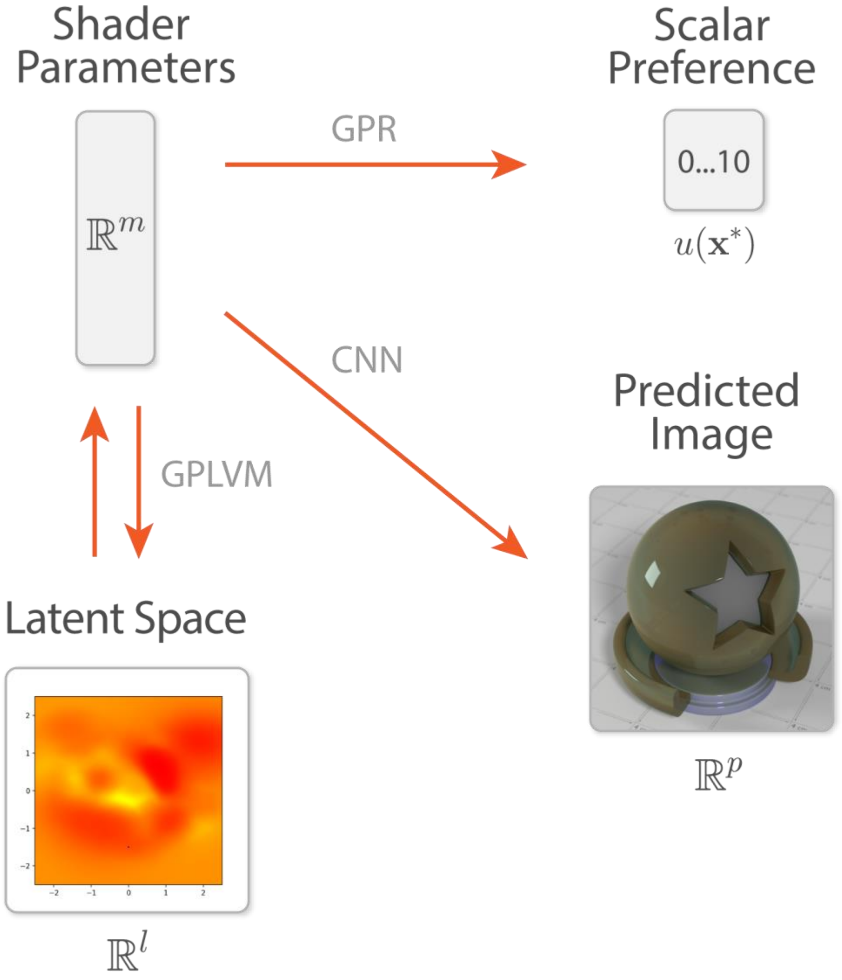
Time taken to generate similar materials



Summary



Gallery with scores　Learning (GPR)　Neural Rendering



Module 1: GPR for recommendations

Module 2: Inflated CNN for Neural Rendering

Module 3: Latent Space for Fine-Tuning