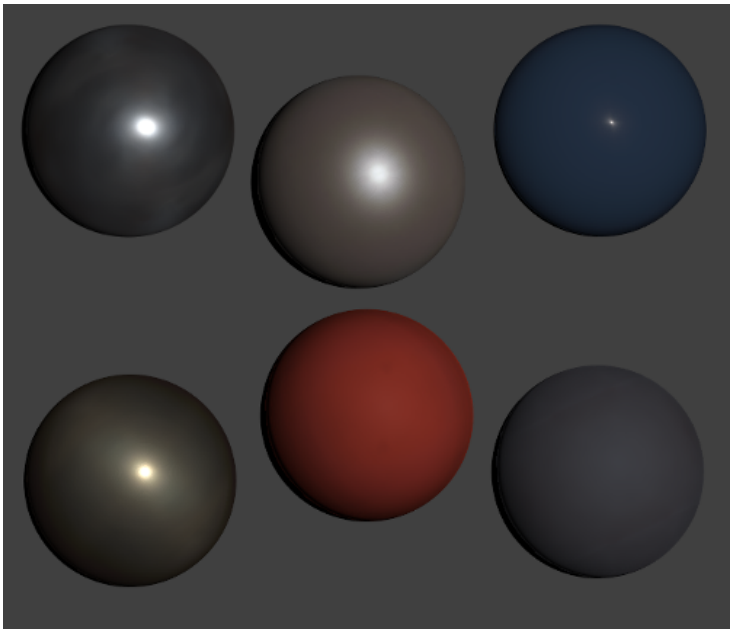
Exploration de l'espace des BRDFs pour la modélisation de matériaux

---

Méthodes et Algorithmes

.



Adrien Grosjean
Ludovic Burg
Karim Salama
Adèle Saint-Denis

Superviseur : Mathias Paulin

UNIVERSITÉ TOULOUSE III PAUL SABATIER
Université de Toulouse

18 novembre 2018

Introduction

Dans le cadre du Master 2, nous avons choisi de réaliser notre chef d'oeuvre sur le sujet basé sur le papier *A Versatile Parameterization for Measured Material Manifolds* de [Soler et al., 2018]. Celui-ci propose une solution au problème suivant :

Lors du processus de design de modèles et de scènes 3D, les graphistes doivent être capable de choisir le matériau qui correspond à leur attentes. Pour ce faire ils ont besoin d'outils leur permettant de visualiser et de modifier ces matériaux. Le logiciel *BRDF Explorer* de Disney répond à ces besoins en proposant une visualisation d'une centaine de matériaux dont les données proviennent de la base MERL.

Le problème réside dans le fait que ces représentations manquent d'ergonomie : d'une part la navigation dans la liste des 100 matériaux est pénible, et d'autre part les paramètres fournis pour les modifier sont insuffisants. L'idée du papier cité plus haut pour résoudre ces deux problèmes est de créer un espace de dimension q inférieur permettant d'explorer intuitivement l'espace des matériaux. Le choix de q dépend de la qualité souhaitée et de la manière dont nous souhaitons naviguer dans l'espace. Un espace 2D à l'avantage de pouvoir être représenté sous la forme d'une carte et d'être simple à visualiser.

Notre projet se déroulera donc en deux temps : Dans un premier temps, nous implémenterons l'algorithme permettant de fournir une paramétrisation acceptable de l'espace des données mesurées par le MERL vers un espace à q dimension : une variété topologique à q dimensions de l'espace des BRDFs.

Ensuite nous choisirons q égal à deux, générerons le modèle de l'espace correspondant et serons en mesure de créer une image pour fournir une représentation de cet espace 2D. Ce faisant, nous obtiendrons une carte que les graphistes pourrons explorer. Notre modèle lui, permettra de reconstruire des données semblables à celles du MERL, c'est à dire produire un nouveau matériau qui n'existait pas dans la base à partir d'un point de la carte, le tout implémenté dans *BRDF Explorer*.

En mettant des points de référence sur la carte (les points de la variété topologique correspondant aux données d'origine), le graphiste pourra alors aisément choisir un matériau entre l'or et l'aluminium par exemple.

CHAPITRE I

Présentation de l'existant

I.1 La base de données MERL

MERL, pour *Mitsubishi Electric Research Laboratories* [Matusik et al., 2003], est une base de données contenant les mesures de 100 matériaux analysés en laboratoire. Ces données proviennent d'un échantillonnage précis de la fonction de la réflectance des matériaux : la BRDF. Pour chacun des matériaux, elle fournit 1,458,000 couples de vecteurs (incident et réfléchi) sur 3 canaux de couleurs (rouge, vert et bleu). La dimension de cet espace selon ces données est donc d'à peu près 4 millions, d'où la nécessité de faire une réduction de dimension pour pouvoir l'explorer.

Le site du laboratoire fourni un code pour charger les données des matériaux dont la forme en sortie est la suivante :

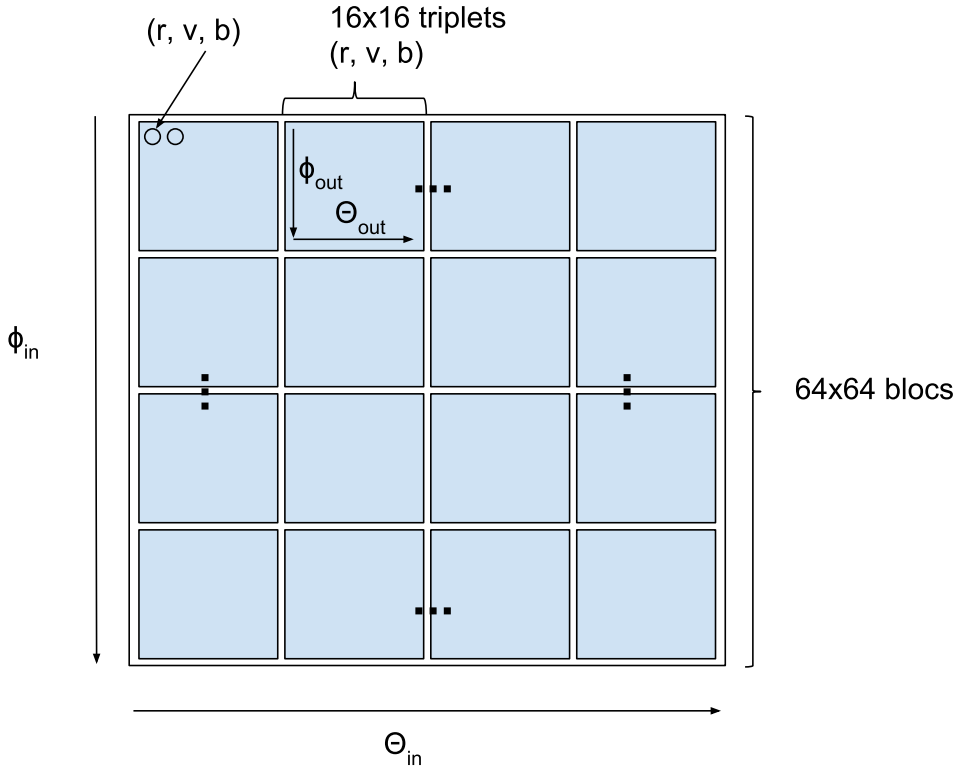


Figure I.1 – Organisation des données récupérées depuis la base MERL

I.2 BRDF Explorer

BRDF explorer est le logiciel développé par Disney qui permet de visualiser et de modifier des matériaux. Cependant, ces modifications ne sont limitées qu'à certains paramètres du matériau, comme sa rugosité.

C'est sur ce logiciel qui utilise la librairie graphique Qt que nous allons implémenter la solution de l'exploration de l'espace des BRDFs sur une variété topologique 2D. Avec cette représentation, le graphiste sera alors capable de choisir facilement un matériau à partir de points de références. Par exemple un matériau interpolé entre l'or et l'aluminium.

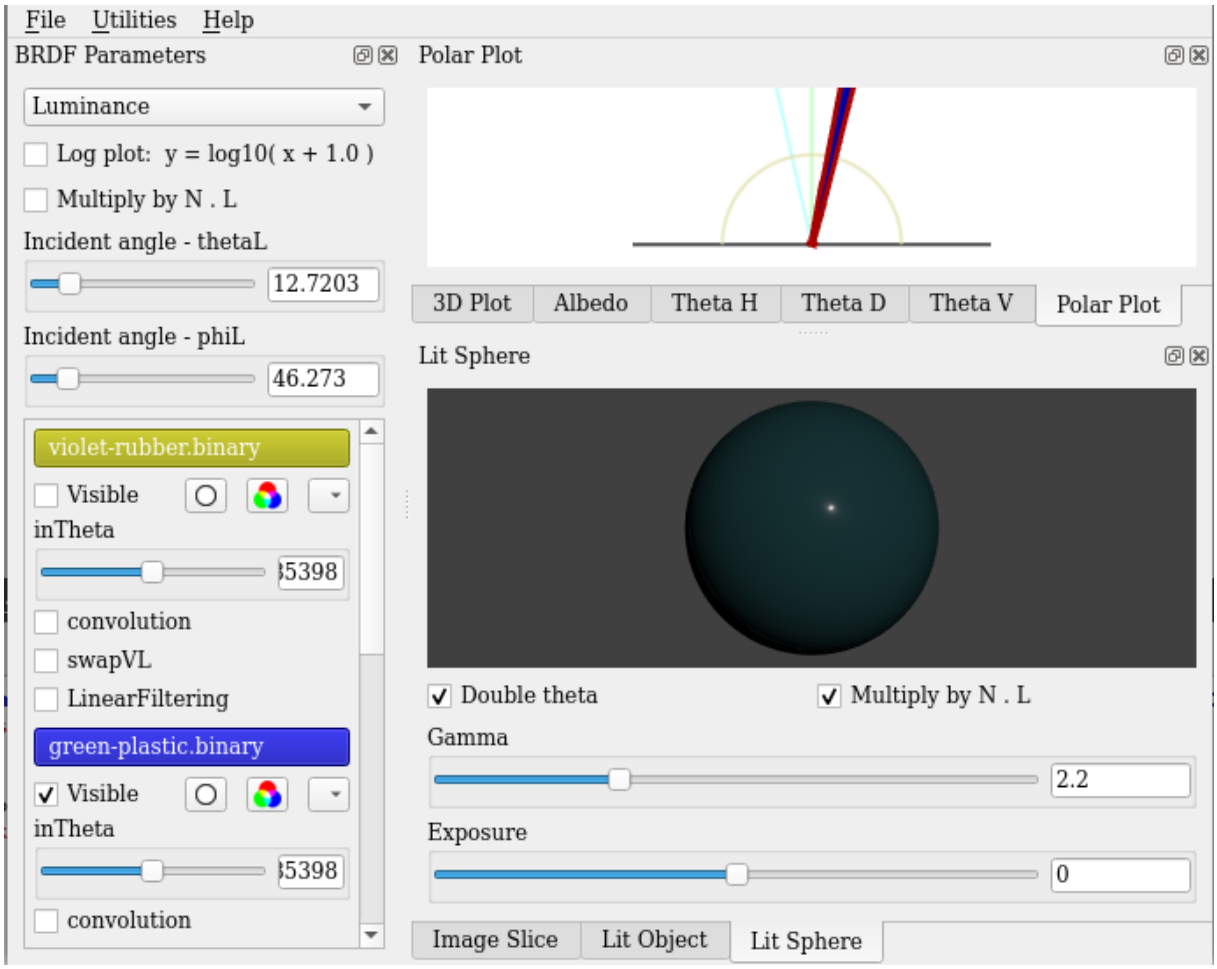


Figure I.2 – Interface de *BRDF Explorer*

CHAPITRE II

Objectif

II.1 Première Partie : Paramétrisation de l'espace des BRDFs

Nous allons implémenter la méthode décrite dans le papier de [Soler et al., 2018].

Une BRDF est considérée comme étant un point dans un espace à N dimensions, avec N ici environ égal à 4 millions. Nous voudrions définir une paramétrisation de cet espace vers un espace de dimension inférieure : c'est ce que nous appellerons **l'espace latent**. Dans ce projet, nous produirons une variété topologique de la même dimension que **l'espace latent** grâce à un apprentissage par processus gaussien (*Gaussian Process Latent Variable Model*). Une variété topologique d'une certaine dimension est une surface paramétrable en cette dimension. Chaque point dans l'espace originel correspond à un point dans l'espace latent, sur la variété topologique. Ces points de correspondance seront nommés **les variables latentes**.

Concrètement, la méthode consiste à trouver le maximum de la fonction de vraisemblance qui se trouve dans le papier. La qualité de paramétrisation produite découle directement de la formule utilisée pour la vraisemblance. Elle induit des contraintes pour permettre d'obtenir de bonnes reconstructions. Par exemple il est préférable d'écarter le plus possible **les variables latentes** dans l'espace latent afin de minimiser les imprécisions dues aux instabilités numériques sur calcul flottant, et de permettre une reconstruction cohérente. La recherche de ce maximum sera faite à l'aide d'un algorithme d'optimisation décrit plus bas.

Nous utiliserons C++ comme language de programmation. Eigen et OpenMp serviront respectivement au calcul matriciel et à la parallélisation. Une fois **l'espace latent** crée, nous exporterons le modèle mathématique nécessaire pour la reconstruction.

Test de l'erreur sur la reconstruction Afin de mesurer la qualité de construction de la variété topologique, nous calculerons l'erreur commise lors de la reconstruction des matériaux à partir des **variables latentes**, selon différentes dimensions de l'espace latent.

II.2 Deuxième Partie : Intégration de la carte des matériaux dans *BRDF Explorer*

La deuxième partie du projet consiste à générer l'image de la représentation de **l'espace latent** et à en faire un nouvel élément d'interface dans *BRDF Explorer*.

Nous produirons grâce à notre code une paramétrisation 2D dont le modèle nous permettra de calculer une représentation image sous forme de carte. Sur cette carte, chaque pixel représentera donc un matériau. La couleur de ces pixels sera donnée par l'albédo du materiau obtenu par reconstruction : celui-ci donne une première idée de l'aspect du matériau car il représente sa couleur moyenne. Cette image servira ensuite de carte pour la sélection des matériaux dans *BRDF Explorer*. Nous rajouterons donc à *BRDF Explorer* la possibilité de choisir une BRDF à partir de la carte des matériaux.

À cette étape nous aurons produit 4 éléments de ressource :

les variable latentes
la matrice K^{-1} permettant de passer de l'espace latent au primal
la matrice Z des données
l'image de la carte des matériaux

Nous serons donc en mesure de créer un élément d'interface : l'utilisateur choisira un point de la carte et validera la coordonnée en appuyant sur un bouton ce qui déclenchera l'appel de la fonction de reconstruction de BRDF. Le callback sera un *slot* dans la classe *ParameterWindow* (conformément à l'organisation déjà en place dans le code) et ouvrira la BRDF une fois reconstruite de la même manière que si elle avait été choisie dans la liste d'items.

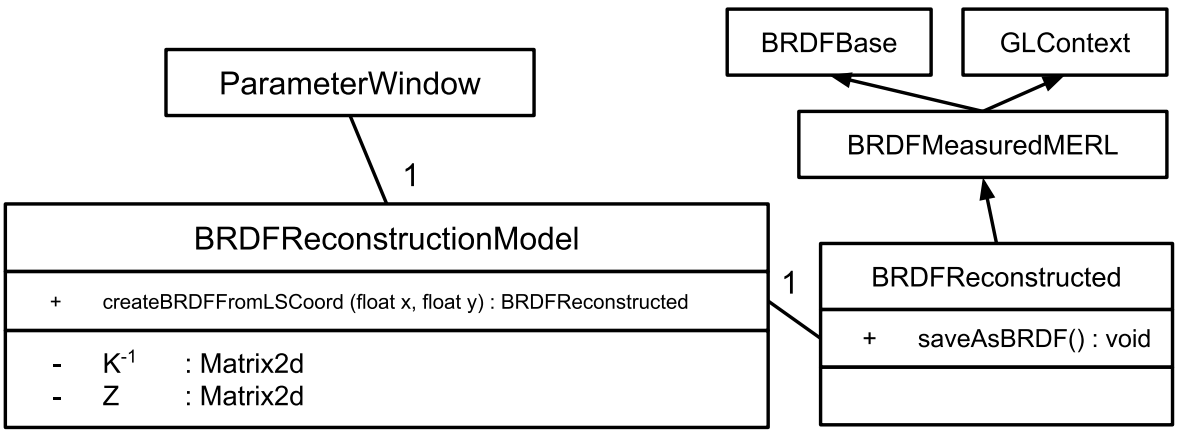


Figure II.1 – Diagramme UML des modules à intégrer dans BRDF explorer

Pour reconstruire une BRDF, nous avons besoin d'une nouvelle classe qui hérite de BRDFMeasuredMERL. Cette classe représentera notre BRDF reconstruite par la classe BRDFReconstuctionModel. De plus on prévoit une fonction dans la classe BRDFReconstructed servant à enregistrer la BRDF reconstruite dans le même format que celles du MERL.

II.2.1 Chargement de Z dans *BRDF Explorer*

Dans notre cas, la matrice Z aura une taille de 7Go. Cela est dû au grand nombre de dimensions des BRDFs.

Nous reconstruirons donc Z depuis *BRDF Explorer*. Il faudra faire attention à charger les BRDFs dans le même ordre dont on a stocké les variables latentes. Pour cela nous utiliserons la bibliothèque standard *FileSystem* de *c++17* qui permet de charger toutes les BRDFs en une fois.

Le chargement de Z dans la mémoire vive prendra par conséquent beaucoup de place. Dans le but de pouvoir s'abstraire de l'emplacement mémoire de Z (dans la mémoire vive ou sur disque avec un fichier temporaire) nous utiliserons la bibliothèque stxxl qui gérera automatiquement cette problématique. Il en découle que nous utiliserons la classe *Map* de *Eigen* afin de manipuler Z écrite dans un type de *stxxl* depuis un type de *Eigen*.

CHAPITRE III

Méthode

III.1 L'algorithme de Hooke et Jeeves

L'apprentissage de la variété topologique 2D par processus gaussien nécessite de trouver le maximum de la fonction de vraisemblance du papier III.1.
La recherche de ce maximum sera effectuée à l'aide de la méthode de [Hooke and Jeeves, 1961] dont la solution représentera **les variables latentes**.
Cet algorithme ne garantit pas l'obtention du maximum global mais est dans la pratique bien adapté à la fonction de vraisemblance en question et produit donc une *bonne* solution.

$$L = 0.5 \times d \times \log|K| + 0.5 \times trace(K^{-1}ZZ^t)$$

Figure III.1 – L est l'opposée du logarithme de la fonction de vraisemblance : c'est la fonction de coût à minimiser. Sa minimisation revient à maximiser la vraisemblance.

Pour la suite de l'explication, nous définissons ces variables :

- **Z :** la matrice où sont stockés les matériaux. Chaque ligne contient les valeurs d'une BRDF.
- **X :** le vecteur contenant les variables latentes. Chaque variable latente, composée de deux coefficients, est mise en colonne et concaténée dans X.
- **K :** la matrice de covariance des variables latentes. Chacun de ses coefficients donne la covariance entre deux variables latentes. La covariance est calculée avec la fonction de l'équation III.2.
- **N :** le nombre de matériaux. Ici N est égal à 100.
- **d :** le nombre de dimensions des BRDFs. Ici d est environ égal à quatre millions.
- **q :** le nombre de dimensions de l'espace latent. Nous prendrons plus tard q égal à 2 pour notre implémentation dans BDRF Explorer.

Initialisation Afin que le processus gaussien converge vers la moyenne des BRDFs, nous commençons par centrer Z sur zéro. Pour cela, nous soustrayions à chaque coefficient de Z sa moyenne.

$$c(x, x') = \mu \times \delta(x - x') + \exp\left(-||x - x'||^2/(2 \times l^2)\right)$$

Figure III.2 – c est la fonction de covariance entre les variables latentes x et x'. μ et l sont des hyperparamètres.
Dans notre implémentation, μ sera par défaut à 10^{-4} et l à 1.
δ est la fonction de Dirac.
$||x - x'||$ est la norme L2 entre x et x'.

Nous initialisons X en effectuant une analyse en composantes principales de Z afin de se rapprocher au mieux du X optimal. Pour cela, nous utiliserons la méthode de [Matusik, 2003].

Algorithme 1 : Analyse en composantes principales (ACP)

Données : Z : une matrice N × d
Résultat : X : un vecteur $(q \times N) \times 1$
début
  $B = Z^T \times Z$;
  Λ = matrice de zéros avec les q plus grandes valeurs propres de B au début de sa diagonale, dans l'ordre décroissant;
  V = matrice des vecteurs propres en colonnes;
  $X = \Lambda^{\frac{1}{2}} \times V^T$;
  X = vectorisation(X);
fin

Ensuite, nous effectuerons des modifications sur X afin de trouver un vecteur de déplacement de X qui améliore la solution, c'est à dire qui fait diminuer la fonction de coût III.1. Nous appellerons ce processus celui des **déplacements d'exploration**.

Un **déplacement d'exploration** consiste à modifier un coefficient de X, et de regarder si cette modification se traduit par une diminution de la fonction de coût.

Soit le pas étant une constante.
Pour tout les coefficients de X :
On incrémente le coefficient avec la valeur du pas et on évalue la fonction de coût.
Si celle-ci a diminuée, on retiens l'incrément dans le vecteur de déplacement de X.
Sinon on teste de la même manière le déplacement dans l'autre sens en décrémentant cette fois-ci le coefficient avec la valeur du pas :
Si la fonction de coût a diminué, on retiens le décrément dans le vecteur de déplacement de X.
Si ce n'est toujours pas le cas, on prend 0 pour le déplacement selon cette coordonnée.

À l'issue de ce processus nous obtenons un vecteur de déplacement de X qui fait diminuer la fonction de coût.

Algorithme 2 : Déplacements d'exploration

Données : X : un vecteur $(q \times N) \times 1$, L : un scalaire
Résultat : X' : un vecteur $(q \times N) \times 1$, L' : un scalaire
début
  pour *chaque coefficient* x_i *de* X **faire**
    $x_i \longleftarrow x_i + pas$;
    $L' \longleftarrow fonctionDeCout(X)$;
    si $L' < L$ **alors**
      $L \longleftarrow L'$;
    sinon
      $x_i \longleftarrow x_i - 2 \times pas$;
      $L' \longleftarrow fonctionDeCout(X)$;
      si $L' < L$ **alors**
        $L \longleftarrow L'$;
      sinon
        $x_i \longleftarrow x_i + pas$;
      fin
    fin
  fin
fin

Nous optimisons ensuite de manière itérative afin de réduire l'erreur entre l'espace des matériaux et son espace latent.

À chaque itération, nous appliquons sur X la modification apprise avec les déplacements d'exploration. Cette modification s'appellera **le déplacement appris**.

Le déplacement appris consiste à appliquer sur X le vecteur de déplacement calculé précédemment.

Algorithme 3 : Déplacement appris

Données : X et XPred : deux vecteurs $(q \times N) \times 1$
Résultat : X' : un vecteur $(q \times N) \times 1$, L' : un scalaire
début
  $X' \longleftarrow X + (X - XPred)$;
  $L' \longleftarrow fonctionDeCout(X')$;
fin

Nous pourrions ensuite effectuer des **déplacements d'exploration** afin de raffiner la direction de translation de X pour l'itération suivante. Cependant, cette action prend un temps non négligeable. Surtout, elle n'améliore pas de manière significative le X final. Donc nous n'effectuerons pas cette action (la fonction L est coûteuse à calculer).

Si après avoir effectué un **déplacement appris**, X n'est pas amélioré, nous diminuons l'amplitude des modifications à effectuer sur X (le pas). Ensuite il faut refaire des **déplacements d'exploration** sur X pour redéfinir le vecteur de déplacement.

Nous arrêtons l'optimisation lorsque le pas est inférieur à un seuil.

Algorithme 4 : Optimisation avec Hooke et Jeeves

Données : Z : une matrice N $\times$ d, pasMin : nombre réel
Résultat : X : un vecteur $(q \times N) \times 1$
début
  $pas \longleftarrow constante$;
  $reducteurPas \longleftarrow constante$; `/* doit appartenir à ]0, 1[ */`;
  $centrerSurZero(Z)$;
  $X \longleftarrow ACP(Z)$;
  $L \longleftarrow fonctionDeCout(X)$;
  Faire
    $X', L' \longleftarrow deplacementExploration(X, L)$;
    tant que $L' < L$ **faire**
      $L \longleftarrow L'$;
      $XPred \longleftarrow X$;
      $X \longleftarrow X'$;
      $X', L' \longleftarrow deplacementAppris(X, XPred)$;
    fin
    $pas \longleftarrow reducteurPas \times pas$;
  Tant que $pas \geq pasMin$;
fin

Pour obtenir une convergence rapide nous utiliserons la même méthode que dans l'article de [Soler et al., 2018] : *constante = 0.5*.

Il faudra alors normaliser le résultat de l'ACP et ensuite optimiser l'espace latent dans l'hypercube centré de côté 2 : chaque coordonnée doit être dans l'intervalle [-1, 1]. Cela rajoute des tests à faire lors des déplacements de X dans le processus d'optimisation (nous rejetterons les coordonnées en dehors de cet hypercube).

La normalisation en elle même ne change pas la solution, elle permet simplement d'avoir des valeurs de pas plus explicites.

III.2 Optimisations de l'algorithme

L'algorithme doit être optimisé au maximum car celui-ci peut prendre beaucoup de temps à s'exécuter.

Comme nous manipulons un grand nombre de données, il faudra faire attention à la précision numérique afin de générer le moins d'erreurs possible.

Lors d'un déplacement d'exploration, nous modifions une seule variable latente. Cette modification a pour seul conséquence la modification d'une seule ligne et d'une seule colonne de la matrice de covariance K. Or, à chaque déplacement d'exploration et à chaque déplacement appris, pour calculer la fonction de coût III.1, il nous faut re-calculer l'inverse et le déterminant de K. Ces deux opérations peuvent être optimisées en temps de calcul grâce à la formule de Sherman-Morisson [Press et al., 2007] et du lemme du déterminant de la matrice [wikipédia, 2018] respectivement.

La modification d'une ligne de K peut être représentée avec l'équation III.3 tandis que la modification d'une de ses colonnes peut s'écrire avec l'équation III.4.

$$\begin{aligned} K + u \times v^T &= K + \begin{pmatrix} 0 & \cdots & 0 & 1 & 0 & \cdots & 0 \end{pmatrix}^T \times (-k_i + k'_i)^T \\ &= K + \begin{pmatrix} 0 & \vdots & 0 & \vdots & 0 \\ 0 & \vdots & \vdots & \vdots & 0 \\ -k_{i1} + k'_{i1} & \vdots & -k_{ij} + k'_{ij} & \vdots & -k_{iN} + k'_{iN} \\ 0 & \vdots & \vdots & \vdots & 0 \\ 0 & \vdots & 0 & \vdots & 0 \end{pmatrix} \end{aligned}$$

Figure III.3 – Représentation de la modification d'une ligne de K. u est un vecteur unitaire avec son i-ième coefficient à un. k_i et k'_i sont respectivement l'ancienne et la nouvelle ligne de K.

$$\begin{aligned} K + u \times v^T &= K + (-k_j + k'_j) \times \begin{pmatrix} 0 & \cdots & 0 & 1 & 0 & \cdots & 0 \end{pmatrix} \\ &= K + \begin{pmatrix} 0 & \cdots & -k_{1j} + k'_{1j} & \cdots & 0 \\ 0 & \vdots & \vdots & \vdots & 0 \\ 0 & \vdots & -k_{ij} + k'_{ij} & \vdots & 0 \\ 0 & \vdots & \vdots & \vdots & 0 \\ 0 & \vdots & -k_{Nj} + k'_{Nj} & \vdots & 0 \end{pmatrix} \end{aligned}$$

Figure III.4 – Représentation de la modification d'une colonne de K. v est un vecteur unitaire avec son j-ième coefficient à un. k_j et k'_j sont respectivement l'ancienne et la nouvelle colonne de K

III.2.1 Le calcul de l'inverse de K à chaque modification d'une variable latente

Nous appliquons une première fois la formule de Sherman-Morisson (III.5) pour calculer K avec une de ses lignes modifiée. Dans ce cas, u et v sont définis dans l'équation III.3. De la même manière, nous appliquons une deuxième fois la formule III.5 pour calculer l'inverse de K après avoir changé une ses colonnes. Cette fois-ci, u et v sont définis dans l'équation III.4.

$$(K + u \times v^T)^{-1} = K^{-1} - \frac{K^{-1} \times u \times v^T \times K^{-1}}{1 + v^T \times K^{-1} \times u}$$

Figure III.5 – Formule de Sherman-Morisson

III.2.2 Le calcul du déterminant de K à chaque modification d'une variable latente

Nous appliquons une première fois le lemme du déterminant de la matrice (III.6) pour calculer K avec une de ses lignes modifiée. Dans ce cas, u et v sont définis dans l'équation III.3. De la même manière, nous appliquons une deuxième fois la formule III.6 pour calculer le déterminant de K après avoir changé une ses colonnes. Cette fois-ci, u et v sont définis dans l'équation III.4.

$$det(K + u \times v^T) = (1 + v^T \times K^{-1} \times u) \times det(K)$$

Figure III.6 – Lemme du déterminant de la matrice

Grâce à ces optimisations, le processus d'optimisation devrait prendre un peu moins de 2 minutes.

III.3 Génération de la carte des matériaux

Nous avons à cette étape une matrice X de taille q $\times$ N contenant les coordonnées des N **variables latentes**.

Pour calculer la carte, nous commençons par choisir une résolution en pixel de la représentation de **l'espace latent** $n \times m$ *(i.e. 1024×1024)*.

Extraction d'une nouvelle BRDF Pour chaque pixel, afin de construire une nouvelle BRDF à partir d'un point en q dimensions (q = 2 ici) de l'espace latent, nous utiliserons l'équation III.7.

$$BRDF = k \times K^{-1} \times Z + \mu$$

Figure III.7 – $BRDF$ est représentée en un vecteur ligne.
k est aussi un vecteur ligne tel que $k = [c(x_0, x), c(x_1, x), ..., c(x_{N-1}, x)]$ avec x un point de l'espace latent et x_i les variables latentes.
La fonction de covariance c est définie dans l'équation III.2.
$k \times K^{-1}$ donne un vecteur de poids pour chaque BRDF contenu dans Z. Par construction, la somme de ses poids est égale à un.
μ est un scalaire. Il s'agit de la moyenne de Z.

Ensuite, nous intégrons la BRDF sur l'hémisphère de la surface du matériau pour obtenir l'albédo :

$$albedo = \int_{w_i} \int_{w_o} (BRDF(w_i, w_o)(w_o \cdot n)\mathrm{d}w_o)(w_i \cdot n)\mathrm{d}w_i$$

Figure III.8 – w_i et w_o sont respectivement une direction d'incidence et une direction de réflectance. $n = (0, 1, 0)$

Il faudra réorganiser les données des BRDFs pour que les accès en mémoire du spectre de réflectance soient contigus (et donc efficace) : à w_i fixe, il faut avoir le spectre de réflectance du matériau à w_o variable aligné en mémoire. Il nous faudra donc coder une fonction de réorganisation des données à partir de la forme présentée au début de ce document vers cette organisation là.

Afin de fournir une information sur la courbure de la variété topologique, nous ajouterons des courbes de niveaux sur la carte des matériaux. Ces courbes de niveaux tracent les zones où la valeur de la luminance des pixels est constante.

En ce qui concerne la navigation sur cette carte, nous permettrons à l'utilisateur de se déplacer grâce à la souris et à zoomer sur la carte.

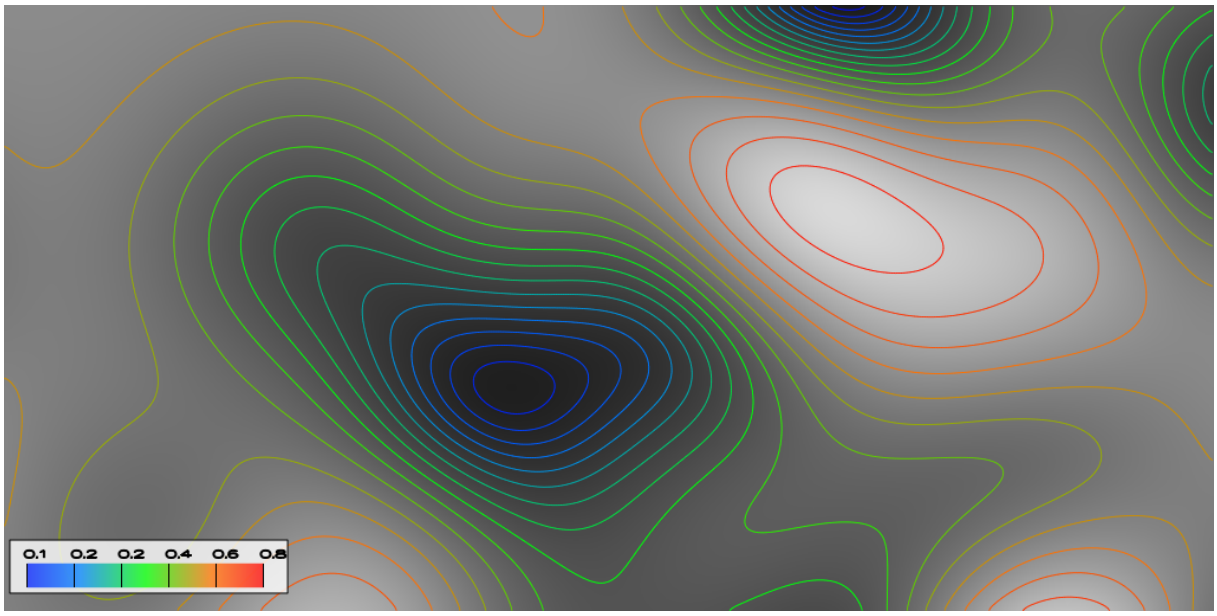


Figure III.9 – BRDFs interpolées sur espace latent

Pour aller plus loin

Lors de notre travail de compréhension des méthodes et algorithmes que nécessite la réalisation de notre sujet, nous avons dégagé des pistes d'explorations potentiellement intéressantes ne faisant pas partie des exigences initiales. Parmis celles-ci :

- Implémenter le calcul de la carte d'albédo en utilisant un processeur graphique *(GPU)* grâce à la technologie *CUDA*. Cette méthode consiste à définir un *kernel* qui décrit le calcul intensif en question, allouer la mémoire nécessaire pour exécuter le calcul et finalement, passer les données aux processeur graphique.

- Utiliser une intégration par méthode de Monte Carlo pour calculer l'intégrale de l'albédo lors du calcul de la carte et mesurer la qualité de la construction en fonction du temps de calcul pour déterminer s'il est possible de calculer la carte en temps réel avec une qualité *raisonnable* (cette notion étant à définir).

Conclusion

Pour conclure, ce chef d'oeuvre est composé de plusieurs parties principales :

Dans un premier temps nous implémenterons une méthode de paramétrisation de l'espace des données vers un espace latent de dimension inférieure. Cette paramétrisation optimisera les coordonnées des données d'origine dans l'espace latent afin de maximiser la qualité de reconstruction.

Par la suite nous produirons une telle paramétrisation en choisissant un espace latent à 2 dimensions, à partir de quoi nous générerons une image pour visualiser cet espace dans laquelle chaque pixel portera la couleur de l'albedo du matériau reconstruit à partir de ses coordonnées latentes.

Finalement nous implémenterons une interface graphique permettant de manipuler cette image sous la forme d'une carte de BRDFs dans le logiciel de Disney *BRDF Explorer* et visualiser les BRDFs reconstruites.

La première partie sera entièrement implémentée dans un logiciel dédié aux tâches décrites plus tôt. Cette application fournira en sortie, les données plongées dans l'espace latent (variables latentes), la matrice de covariance permettant de passer de l'espace latent à l'espace originel et la carte des matériaux dans le but de les utiliser dans le logiciel *BRDF Explorer*.

Bibliographie

Robert Hooke and T. A. Jeeves. " direct search" solution of numerical and statistical problems. *J. ACM*, 8(2) :212–229, April 1961. ISSN 0004-5411. doi : 10.1145/321062.321069. URL http://doi.acm.org/10.1145/321062.321069.

Wojciech Matusik. *A data-driven reflectance model*. PhD thesis, Massachusetts Institute of Technology, 2003.

Wojciech Matusik, Hanspeter Pfister, Matt Brand, and Leonard McMillan. A data-driven reflectance model. *ACM Transactions on Graphics*, 22(3) :759–769, July 2003.

William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes 3rd Edition : The Art of Scientific Computing*. Cambridge University Press, New York, NY, USA, 3 edition, 2007. ISBN 0521880688, 9780521880688.

Cyril Soler, Kartic Subr, and Derek Nowrouzezahrai. A Versatile Parameterization for Measured Material Manifolds. *Computer Graphics Forum*, 37(2) :1–10, April 2018. URL https://hal.inria.fr/hal-01702722.

wikipédia. Matrix determinant lemma, 2018. URL https://en.wikipedia.org/wiki/Matrix_determinant_lemma.