

Langevin Monte-Carlo avec différence finie

par

Francis BEAUCHAMP

PROJET PRÉSENTÉ À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE  
COMME EXIGENCE PARTIELLE À L'OBTENTION DE LA MAÎTRISE  
AVEC PROJET  
M. Ing.

MONTRÉAL, LE 16 AVRIL 2025

ÉCOLE DE TECHNOLOGIE SUPÉRIEURE  
UNIVERSITÉ DU QUÉBEC



Francis Beauchamp, 2025



Cette licence Creative Commons signifie qu'il est permis de diffuser, d'imprimer ou de sauvegarder sur un autre support une partie ou la totalité de cette oeuvre à condition de mentionner l'auteur, que ces utilisations soient faites à des fins non commerciales et que le contenu de l'oeuvre n'ait pas été modifié.

**PRÉSENTATION DU JURY**

**CE PROJET A ÉTÉ ÉVALUÉ**

**PAR UN JURY COMPOSÉ DE:**

M. Adrien Gruson, directeur de projet  
Département de Génie logiciel et des TI, École de technologie supérieure

M. Sheldon Andrews, examinateur externe  
Département de Génie logiciel et des TI, École de technologie supérieure



## **REMERCIEMENTS**

Nous tenons particulièrement à remercier Adrien Gruson, enseignant au département des technologies de l'information à l'École de technologie supérieure, pour les nombreuses discussions utiles à ce travail qui ont eu lieu tout au long de sa rédaction. Nous remercions également Joey Litalien, pour ses contributions à l'outil de visualisation utilisé pour générer les courbes de convergence utilisées à des fins de comparaison entre les différents algorithmes présentés dans cet ouvrage.



# Langevin Monte-Carlo avec différence finie

Francis BEAUCHAMP

## RÉSUMÉ

La convergence des techniques d'intégration Monte-Carlo dépend de quelques facteurs ; la facilité avec laquelle l'intégrale de la contribution lumineuse peut être approximée par des échantillons tirés proportionnellement à une densité de probabilité, et à quel point la technique d'échantillonnage utilisée parvient à générer des échantillons distribués selon une fonction cible. Le premier de ces facteurs peut être mitigé par l'usage d'une technique Monte-Carlo basée sur les chaînes de Markov. Il s'agit d'un processus itératif qui fait subir séquentiellement des transformations aux échantillons avant de décider de les accepter ou de les rejeter. La mutation employée dans cette approche a une grande influence sur l'exploration de l'espace des états. Lorsque des interactions entre la lumière et des matériaux anisotropes surviennent, la qualité des explorations locales est affectée par la nature isotrope d'une perturbation uniforme. Des méthodes plus avancées comme « Metropolis Adjusted Langevin Algorithm » (MALA) permettent de régler ce problème en utilisant des systèmes d'autodifférenciations pour calculer le gradient de la fonction cible et proposer des mutations orientées vers les plus grosses variations. Ce faisant, le taux d'acceptation des mutations augmente et permet de mieux parcourir l'espace des états, ce qui diminue la corrélation entre les échantillons. Cet ouvrage étudie les problèmes liés aux interactions anisotropes et propose d'utiliser les différences finies pour évaluer le gradient. À l'inverse des autres approches, notre méthode permet de prendre en charge des mutations anisotropes sans avoir recours à un système d'autodifférentiation. La méthode est simple et s'intègre facilement à des moteurs de rendu photoréalistes.

**Mots-clés:** Markov-Chain Monte-Carlo, Metropolis Light Transport, Langevin, simulation du transport de la lumière



# Langevin Monte Carlo with finite difference

Francis BEAUCHAMP

## ABSTRACT

The convergence of Monte Carlo integration techniques depends on a few factors ; the ease with which the integral of the light contribution can be approximated by samples drawn in proportion to a probability density, and the extent to which the sampling technique used manages to generate samples distributed according to a target function. The first of these factors can be mitigated by the use of a Monte Carlo technique based on Markov chains. It is an iterative process that sequentially transforms samples before deciding whether to accept or reject them. The mutation employed in this approach has a great influence on the exploration of the state space. When interactions between light and anisotropic materials occur, the quality of local explorations is affected by the isotropic nature of the uniform perturbation. More advanced methods such as Metropolis Adjusted Langevin Algorithm (MALA) can solve this problem by using auto-differentiation systems to calculate the gradient of the target function and propose mutations oriented towards the largest variations. In doing so, the mutation acceptance rate increases and allows for better exploration of the state space, which decreases the correlation between samples. This work studies the problems related to anisotropic interactions and proposes to use finite differences to evaluate the gradient. Unlike other approaches, our method can manage anisotropic mutations without resorting to an auto-differentiation system. The method is simple and easily integrates with photorealistic rendering engines.

**Keywords:** Markov-Chain Monte Carlo, Metropolis Light Transport, Langevin, light transport simulation



## TABLE DES MATIÈRES

	Page
INTRODUCTION .....	1
CHAPITRE 1 PRÉLUDE .....	5
1.1    Simulation du transport de la lumière .....	5
1.1.1    Radiométrie .....	5
1.1.2    Réflectivité bidirectionnelle .....	7
1.1.3    Équation de rendu .....	9
1.2    Méthodes numériques d'intégration .....	9
1.2.1    Monte-Carlo .....	10
1.2.2    Échantillonnage par importance multiple .....	11
1.2.3    Monte-Carlo par chaînes de Markov .....	13
1.2.4    Metropolis-Hastings .....	14
1.3    Metropolis Light Transport .....	15
1.3.1    Espace primaire .....	16
1.3.2    Dynamique de Langevin .....	17
1.4    Rendu dans le domaine des gradients .....	18
1.4.1    Fonction de décalage .....	19
1.4.2    Reconstruction .....	21
CHAPITRE 2 TRAVAUX CONNEXES .....	23
2.1    MCMC avec mutation isotrope .....	23
2.1.1    PSSMLT .....	23
2.1.2    GDMLT .....	25
2.2    MCMC avec mutation adaptative .....	27
2.2.1    Stratification .....	27
2.2.2    MALA .....	28
CHAPITRE 3 DESCRIPTION DE LA MÉTHODE .....	31
3.1    Estimation des gradients .....	31
3.1.1    Fonction de décalage .....	32
3.1.2    Fonction cible et noyaux .....	33
3.2    Mutations .....	35
3.2.1    MALA .....	35
3.2.2    MALA avec adaptation .....	37
3.3    Reconstruction itérative .....	39
CHAPITRE 4 PRÉSENTATION DES RÉSULTATS .....	41
CONCLUSION ET RECOMMANDATIONS .....	55

ANNEXE I      RÉSULTATS ADDITIONNELS .....	59
BIBLIOGRAPHIE .....	61

## **LISTE DES TABLEAUX**

	Page
Tableau 4.1      Erreur quadratique moyenne des intégrateurs avec un budget en temps de rendu .....	43
Tableau 4.2      Erreur quadratique moyenne des intégrateurs avec un budget d'échantillons .....	43



## LISTE DES FIGURES

	Page	
Figure 1.1	Différents types d'illumination .....	5
Figure 1.2	Unités de mesure en radiométrie .....	7
Figure 1.3	Composantes d'une BRDF .....	8
Figure 1.4	Chemin de lumière .....	9
Figure 1.5	Convergence de Monte-Carlo .....	10
Figure 1.6	Échantillonnage par importance multiple .....	12
Figure 1.7	Mutations utilisées par MLT .....	16
Figure 1.8	Espace primaire .....	17
Figure 1.9	Calcul d'une image final avec les gradients .....	19
Figure 1.10	Fonctions de décalage .....	20
Figure 1.11	Reconstruction de poisson .....	22
Figure 2.1	Mutation de PSSMLT stratifiée .....	29
Figure 2.2	Mutation gaussienne pour MALA .....	30
Figure 3.1	États d'un rayon décalé .....	33
Figure 3.2	Fonction de décalage pour MALA et GDMALA .....	34
Figure 3.3	Reconstruction itérative .....	40
Figure 4.1	Cornelbox : comparaison à temps égal .....	46
Figure 4.2	Cornelbox : comparaison à nombre d'échantillons égal .....	47
Figure 4.3	Classroom : comparaison à temps égal .....	48
Figure 4.4	Classroom : comparaison à nombre d'échantillons égal .....	49
Figure 4.5	Door ajar : comparaison à temps égal .....	50
Figure 4.6	Door ajar : comparaison à nombre d'échantillons égal .....	51

Figure 4.7	Bidir : comparaison à temps égal .....	52
Figure 4.8	Bidir : comparaison à nombre d'échantillons égal .....	53
Figure 5.1	Augmentation du nombre de gradients .....	55
Figure 5.2	Différents types de noyau .....	56
Figure 5.3	Impact des variables de contrôle sur la reconstruction itérative .....	57

## **LISTE DES ABRÉVIATIONS, SIGLES ET ACRONYMES**

LMC	« Langevin Monte-Carlo »
MALA	« Metropolis Adjusted Langevin Algorithm »
MCMC	Monte-Carlo par Chaîne de Markov
MH	Metropolis-Hastings
MLT	« Metropolis Light Transport »
MIS	Échantillonnage par importance multiple
GDMLT	« Metropolis Light Transport » dans le domaine des gradients
PSSMLT	« Primary Sample Space Metropolis Light Transport »
NEE	« Next Event Estimation »
BRDF	« Bi-directional Reflectance Distribution Function »
GDMALA	« Metropolis Adjusted Langevin Algorithm » dans le domaine des gradients
RSR	« Random Sample Replay »
PDF	Fonction de densité
RMSE	Racine de l'erreur quadratique moyenne
MAPE	Pourcentage d'erreur absolue moyenne



## LISTE DES SYMBOLES ET UNITÉS DE MESURE

$\Phi$	Flux
$E$	Éclairement
$M$	Radiosité
$\omega$	Angle solide
$I$	Intensité
$L$	Luminance
$W$	Watts
$sr$	Steradian
$\Omega$	Espace des chemins
$\mathcal{P}$	Espace primaire



## INTRODUCTION

Le rendu photoréaliste est une branche de l'infographie qui s'intéresse plus particulièrement à la simulation du comportement de la lumière afin de générer des images en accord avec les lois de la physique. L'équation fondamentale de rendu permettant ce type de simulation a pour la première fois été introduite par Kajiya (1986), pour ensuite être mise en pratique par différentes techniques d'intégration Monte-Carlo. En accumulant plusieurs chemins de lumière générés aléatoirement, on obtient une approximation du comportement physique par lequel la lumière se propage au sein d'une scène. Cela permet entre autres de représenter correctement des effets de lumière complexes tels que des caustiques et des phénomènes de transluminescence. Lorsque le nombre de ces effets augmente dans une scène, les méthodes Monte-Carlo perdent vite en efficacité, étant donné la forte concentration de chemins de lumières. Des techniques plus avancées comme Monte-Carlo par chaîne de Markov (MCMC) (Metropolis, Rosenbluth, Rosenbluth, Teller & Teller, 1953) permettent de pallier ces problèmes. Au lieu de générer indépendamment les chemins, un algorithme MCMC utilise une chaîne de Markov pour piloter une marche aléatoire dans l'espace des états et produire des chemins de lumière statistiquement corrélés. Le but est de naviguer dans l'espace global et trouver des chemins dont la contribution lumineuse est plus importante afin d'en explorer l'espace local. Metropolis Light Transport (MLT) (Veach & Guibas, 1997) introduit une formulation de cet algorithme qui modifie directement la géométrie des vertex qui composent un chemin de lumière, ce qui est à la fois coûteux et complexe à mettre en place ; en fonction des interactions lumineuses avec les matériaux, plusieurs mutations différentes doivent être appliquées sur les vertex de l'espace des chemins  $\Omega$ . Des simplifications subséquentes ont été apportées à MLT avec l'espace primaire  $\mathcal{P}$  (Kelemen, Szirmay-Kalos, Antal & Csonka, 2002) qui manipule directement les nombres aléatoires des chemins via le mappage  $\mathcal{P} \rightarrow \Omega$ , facilitant l'implémentation de techniques d'intégration Monte-Carlo basées sur les chaînes de Markov. MLT dans le domaine des gradients (GDMLT) (Lehtinen *et al.*, 2013) ajoute à MLT une dimension additionnelle : en calculant les gradients de l'image, nous pouvons résoudre

un problème de minimisation qui donne un résultat avec une faible variance. Cependant, la méthode n'est pas parfaite ; des gradients bruités entraînent des artefacts et des discontinuités qui sont très facilement perceptibles. Finalement, « Metropolis-Adjusted Langevin Algorithm » (MALA) (Roberts & Tweedie, 1996) aborde le problème sous un angle différent. En simulant un système de particules dynamiques propulsées par un mouvement brownien, la marche aléatoire peut être dirigée dans des régions plus intéressantes de l'espace des états, ce qui permet de réduire la variance avec de plus grandes mutations qui conservent un taux d'acceptation élevé. Cet algorithme a la particularité de nécessiter le calcul de dérivées de premier ordre, ce qui est traditionnellement rendu possible grâce à un système d'autodifférentiation (Jakob, Speierer, Roussel & Vicini, 2022). Cela est problématique pour son implémentation dans les moteurs de rendu en industrie ; il devient nécessaire d'apporter des changements importants au code et d'implémenter plusieurs surcharges d'opérateurs. Incidemment, cela limite l'usage de cette technique au monde académique.

Nous proposons d'utiliser les différences finies obtenues par la fonction de décalage du domaine des gradients pour calculer une approximation des dérivées employée par MALA. Cela nous donne l'occasion d'expérimenter avec l'usage de cet algorithme pour le rendu photoréaliste sans avoir recours à un système d'autodifférentiation. Cette simplification de la méthode donne lieu à deux principales contributions :

- Une adaptation de MALA qui remplace le terme de dérivée analytique  $\nabla \log \pi(X(t))$  par des différences finies. L'estimation de ces dérivées retire la dépendance à un système d'autodifférentiation, facilitant grandement la mise en place de la technique.
- Une adaptation de GDMLT qui utilise la dynamique de Langevin pour diriger le processus de marche aléatoire. GDMALA calcule le gradient de la fonction cible avec des noyaux dont le centre est décalé dans les axes du gradient de l'image.

Plus particulièrement, notre approche utilise des noyaux dont le pixel central sert d'ancrage aux pixels voisins. Il suffit de combiner ces noyaux pour pouvoir en faire la différence finie et calculer la discrétisation de Langevin. Ce faisant, notre méthode ouvre la possibilité d'exploiter des techniques basées sur la dynamique de Langevin dans des applications qui n'ont pas accès à la différentiation automatique.



# CHAPITRE 1

## PRÉLUDE

### 1.1 Simulation du transport de la lumière

L'illumination globale, aussi appelée la simulation du transport de la lumière, tente à la fois de modéliser l'éclairage des sources visibles à partir d'une surface (lumière directe) et la lumière qui rebondit de surface en surface (lumière indirecte) (Fig. 1.1). L'avantage de ce type d'illumination est que cela permet d'obtenir des images beaucoup plus photoréalistes. Nous allons commencer par aborder dans cette section les différentes valeurs fondamentales en radiométrie. Nous verrons par la suite comment elles sont utilisées pour calculer la réflectivité bidirectionnelle. Finalement, l'équation de rendu sera introduite et servira de cadre pour tous les algorithmes présentés dans cet ouvrage.

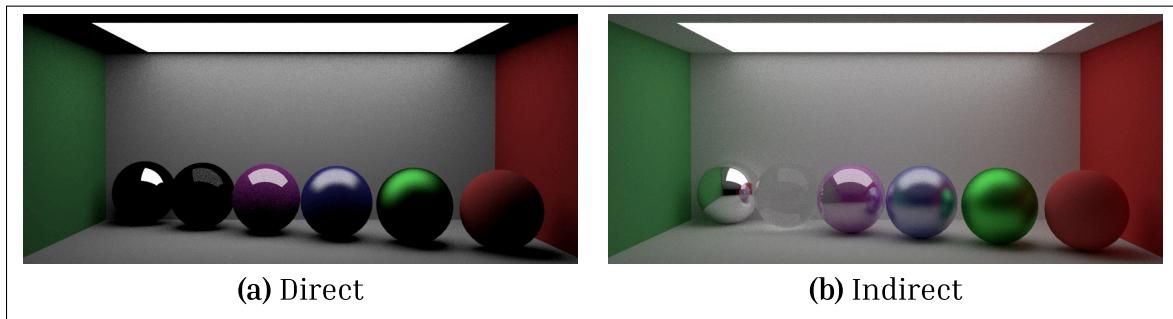


Figure 1.1 L'illumination directe **(a)** ne permet pas de représenter le comportement de la lumière de façon réaliste. Plus particulièrement, les surfaces métalliques et diélectriques en **(b)** nécessitent plusieurs rebonds de lumière pour être simulées correctement.

#### 1.1.1 Radiométrie

La radiométrie est la science qui dicte les comportements physiques de la lumière. Cela inclut entre autres la quantification de mesures comme l'intensité, l'éclairement, le flux, la luminance et la radiosité (Fig. 1.2). Les techniques abordées dans cet ouvrage vont y faire référence à mainte reprise ; il est donc pertinent d'en voir les formulations mathématiques.

Le **flux**  $\Phi$  représente la quantité totale d'énergie lumineuse qui passe par une surface quelconque et se calcule en watts ( $W$ ). C'est la mesure de l'énergie des photons qui entrent en contact avec des objets. Soit  $Q = \frac{hc}{\lambda}$  l'énergie d'un photon en joules ( $J$ ), où  $h$  est la constante de Plank,  $h \approx 6.626 \times 10^{-34} m^2 kg/s$ ,  $c$  la vitesse de la lumière,  $299472458 m/s$  et  $\lambda$  la longueur d'onde. On obtient le flux  $\Phi$  en mesurant l'énergie pour une période donnée avec l'équation différentielle :

$$\Phi = \frac{dQ}{dt}. \quad (1.1)$$

L'**éclairement**  $E$  est la mesure du flux lumineux qui arrive par unité de surface et se calcule en watts par mètre carré ( $W/m^2$ ). C'est en quelque sorte une façon de décrire la densité moyenne d'un flux lumineux. Soit  $A$ , l'aire d'une surface illuminée centrée en  $x$ . On obtient l'éclairement  $E$  en mesurant le flux par rapport à une surface avec l'équation différentielle :

$$E(x) = \frac{d\Phi(A)}{dA(x)}. \quad (1.2)$$

La **radiosité**  $M$  est en quelque sorte l'inverse de l'éclairement et se calcule sur une surface émettrice. Il s'agit de la mesure du flux lumineux sortant par unité de surface et se calcule en watts par mètre carré ( $W/m^2$ ). Soit  $A$ , une toute petite unité de surface lumineuse centrée en  $x$ . On obtient la radiosité  $M$  par l'équation différentielle :

$$M(x) = \frac{d\Phi(A)}{dA(x)}. \quad (1.3)$$

L'**angle solide**  $\omega$  est une unité utilisée par quelques mesures en radiométrie et se calcule en stéradians ( $sr$  ou  $\omega$ ). Pour bien le comprendre, il faut d'abord s'imaginer la région de la circonférence d'un cercle contenu dans un angle  $\theta$  en radians ( $rad$ ). L'angle solide inclut de façon analogue cette même région. On l'obtient en délimitant l'aire  $A$  de la base d'un cône orienté en coordonnées polaires par  $\theta$  et  $\phi$  et dont le sommet est au centre d'une sphère de rayon  $r$ .

$$\omega = \frac{A}{r^2}. \quad (1.4)$$

L'**intensité**  $I$  est la mesure du flux d'une surface émettrice par angle solide dans une direction donnée et se calcule en watts par stéradian ( $W/sr$ ).

$$I(\omega) = \frac{d\Phi(A)}{d\omega}. \quad (1.5)$$

La **luminance**  $L$  vient fermer la boucle en mesurant la densité du flux par unité de surface, par angle solide. Elle se calcule en watts par steradian par mètre carré ( $W \cdot sr \cdot m^2$ ). Soit  $I$  l'intensité,  $\theta$  l'angle entre la normale de la surface émettrice et la direction de l'angle solide. On utilise le cosinus pour tenir compte de la loi de Lambert qui stipule que la densité de lumière est réduite en fonction de l'orientation d'une surface. On obtient donc la luminance  $L$  par l'équation différentielle :

$$L(\omega) = \frac{dI(\omega)}{dA \cos \theta}. \quad (1.6)$$

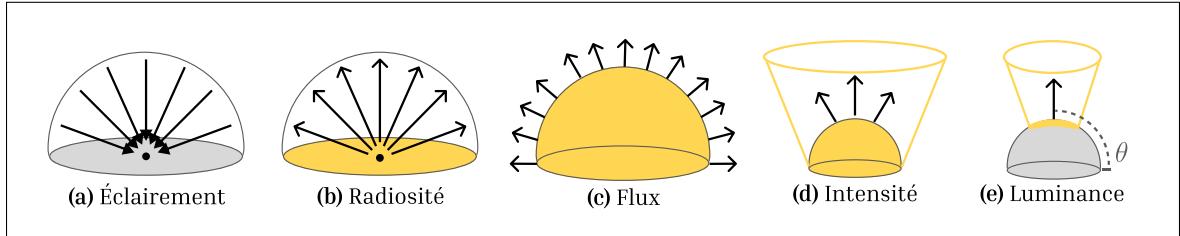


Figure 1.2 Les différentes mesures utilisées en radiométrie.

### 1.1.2 Réflectivité bidirectionnelle

La lumière qui parcourt une scène entre en contact avec des surfaces dont les propriétés dictent l'orientation et la couleur de sa réflexion. Pour parvenir à calculer une direction sortante, on utilise une « Bidirectional Reflectance Distribution Function » (BRDF) (Pharr, Jakob & Humphreys, 2023). Cette fonction prend en considération deux choses pour calculer la couleur et l'orientation de la lumière réfléchie :

1. La **direction incidente**  $\omega_i$  de la lumière caractérisée par les angles  $\theta_i$  et  $\phi_i$  en coordonnées sphériques.

2. La **fonction de distribution**  $f_v(x, \omega_i, \omega_o)$  qui dicte la probabilité qu'un rayon incident  $\omega_i$  soit réfléchi dans une direction sortante  $\omega_o$ .

Il y a quelques propriétés de  $f_v$  qui doivent être respectées si l'on veut s'assurer d'un résultat physiquement réaliste. Premièrement, puisqu'il s'agit d'une probabilité, le résultat ne peut jamais être négatif (c.-à-d.  $f_v \geq 0$ ). Ensuite, en accord avec le théorème de réciprocité de Lorentz, la réflexion doit être réciproque (c.-à-d.  $f_v(x, \omega_i, \omega_o) = f_v(x, \omega_o, \omega_i)$ ). Finalement, les réflexions de la lumière doivent être en accord avec la loi de conservation de l'énergie, ce qui revient à s'assurer que l'on n'intègre jamais des quantités lumineuses supérieures à 1 :  $\int_{H^2} f_v(x, \omega_i, \omega_o) \cos \theta d\omega_r \leq 1$ .

Il existe plusieurs modèles de BRDF comme l'approximation de Phong (1975), le modèle de Schlick (1994) ou encore le modèle Lambertien (Taillet, Villain & Febvre, 2023). Ces derniers vont généralement utiliser une combinaison de trois grandes catégories : diffus, brillant et spéculaire (Fig. 1.3). Les surfaces diffuses vont réfléchir la lumière identiquement dans toutes les directions. Les surfaces brillantes comme du métal ou du plastique vont principalement réfléchir la lumière dans la direction d'un lobe spéculaire. Les surfaces parfaitement spéculaires n'auront qu'une direction de réflexion possible. D'autres modélisations plus complexes permettent de traiter la transmission dans le verre avec des indices de réfraction, ou encore des réflexions avec plusieurs lobes.

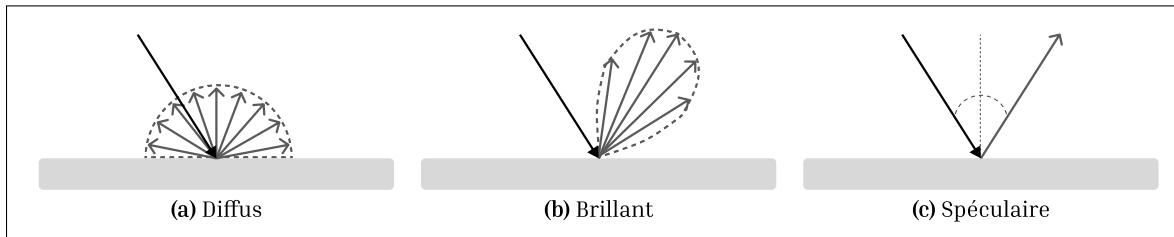


Figure 1.3 Les principales composantes d'une BRDF sont illustrées avec la lumière incidente en noir et la modélisation des réflexions possibles en gris.

### 1.1.3 Équation de rendu

Difficile d'aborder les problèmes en transport de la lumière sans prendre connaissance de l'équation de rendu fondamentale proposée par Kajiya (1986). L'équation 1.7 décrit la propagation de la luminance  $L$  par des rayons de lumière entrants et sortants  $\omega_i, \omega_o$  (Fig. 1.4) qui naviguent dans l'espace des chemins  $\Omega$  d'une scène vers une caméra située en  $x_0$

$$L_o(x, \omega_o) = L_e(x, \omega_o) + \int_{\Omega} f_v(x, \omega_i, \omega_o) L_i(x, \omega_i) (\omega_i \cdot n) d\omega_i, \quad (1.7)$$

où son application récursive de surface en surface donne  $F(\bar{x})$  la contribution lumineuse du chemin  $\bar{x}$ . Pour  $n$  interactions, on a donc une liste de  $n + 1$  vertex  $\bar{x} = \{x_0, \dots, x_n\}$ . Nous verrons que des méthodes numériques permettent d'avoir un résultat sans utiliser de formule d'intégration.

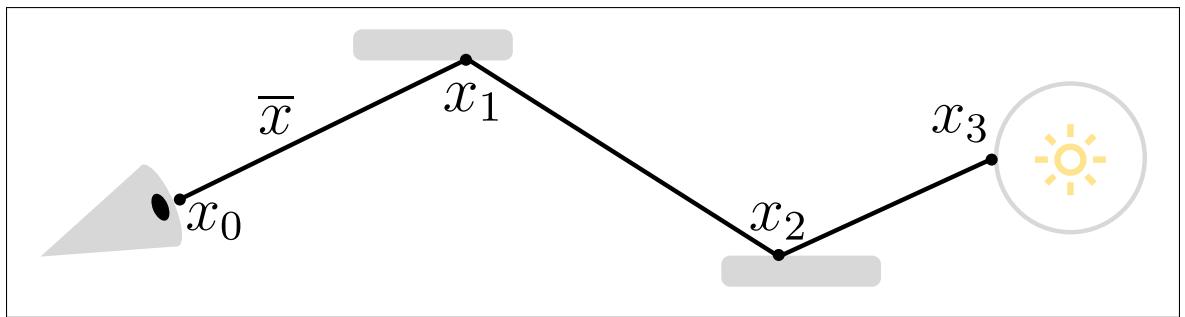


Figure 1.4 Exemple de chemin de lumière  $\bar{x}$  où la ligne continue des interactions débute par la source lumineuse  $x_3$  et se termine sur la caméra en  $x_0$ . Les vertex  $x_1$  et  $x_2$  désignent une interaction avec une surface.

### 1.2 Méthodes numériques d'intégration

Le concept d'infinité présent dans le comportement physique de la lumière (p.ex. lorsqu'elle réfléchit sur une surface diffuse toutes les directions) présente des difficultés du point de vue des calculs d'intégration. Pour faciliter l'implémentation d'algorithmes, il existe des méthodes numériques qui permettent d'approximer les intégrales avec suffisamment de précision. C'est généralement ce genre d'approche qui sera utilisée en rendu.

### 1.2.1 Monte-Carlo

Une technique commune pour intégrer l'équation de rendu consiste à générer plusieurs échantillons et d'en faire la somme pondérée avec un estimateur Monte-Carlo (Alg. 1). Nous obtenons donc  $N$  chemins  $\bar{x}_n$  pour chaque pixel  $ij$  proportionnellement à la fonction de densité (PDF)  $p(\bar{x}_n)$ , ce qui donne l'équation 1.8. Pour qu'un estimateur de cette forme soit non biaisé, il est important que la valeur de  $p$  soit positive lorsque la valeur de  $F$  l'est aussi. De façon générale, la variance dépend largement du fait que  $F$  soit proportionnel à  $p$ , mais cette relation est difficile à obtenir en pratique :

$$I_{ij} \approx \frac{1}{N} \sum_{n=1}^N \frac{F(\bar{x}_n)}{p(\bar{x}_n)}. \quad (1.8)$$

Cette estimation accumule une erreur tout au long du processus. Dans notre cas, elle se traduit en bruit dans l'image finale. Nous pouvons amoindrir les effets avec plus d'échantillons (Fig. 1.5); il reste que la convergence  $O(1/\sqrt{N})$  de l'estimateur Monte-Carlo (Hua *et al.*, 2019) est lente et présente des problèmes au niveau de la performance.

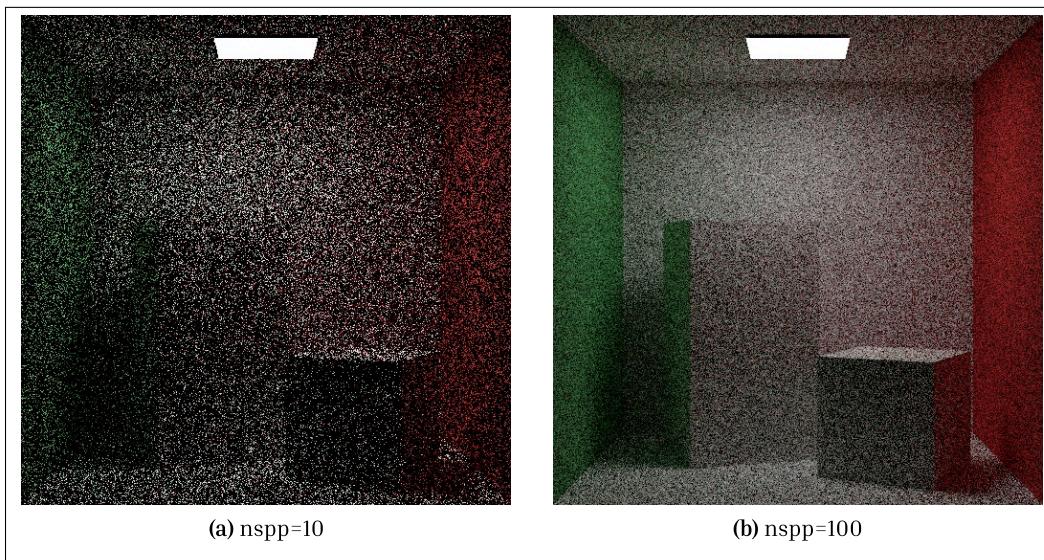


Figure 1.5 La convergence de Monte-Carlo est illustrée avec un rendu n'utilisant pas le « next event estimation » (NEE) des sources de lumière. Par exemple, avec 10 échantillons par pixel (a) et 100 échantillons par pixel (b).

### Algorithme 1.1 Intégration par Monte-Carlo

```

1 Algorithme : Monte-Carlo avec N échantillons par pixel
Input : Nombre d'échantillon par pixel  $N \in \mathbb{R}_{\geq 0}$ , taille de l'image  $\{X, Y\}$ 
Output : Image estimée par l'intégration  $I$ 

2 Initialisation de l'image  $\forall x, y : I[x, y] \leftarrow zero;$ 
3 for all pixels  $I_{ij} \in (i = 1, \dots, X), (j = 1, \dots, Y)$  do
4   for all samples  $\in (n = 1, \dots, N)$  do
5      $\bar{x}_n \leftarrow trace(I_{ij});$ 
6      $I_{ij} += F(\bar{x}_n)/p(\bar{x}_n);$ 
7   end for
8 end for
9  $I = I/N;$ 
10 output image  $I;$ 
```

#### 1.2.2 Échantillonnage par importance multiple

Monte-Carlo nous permet d'évaluer des intégrales de la forme  $\int F(\bar{x})d\bar{x}$  en tirant des échantillons proportionnellement à une densité  $p(\bar{x})$ . Cependant, il arrive que nous devions évaluer l'intégrale du produit de plusieurs fonctions  $\int f(\bar{x})g(\bar{x})d\bar{x}$ . Par exemple, le calcul de l'illumination directe prend la forme  $\int f_v(x, \omega_i, \omega_o)L_i(x, \omega_i)(\omega_i \cdot n)d\omega_i$ . L'échantillonnage proportionnellement à  $f_v$  ou  $L_i$  risque de donner de mauvais échantillons et engendrer une variance élevée. On peut s'imaginer que des directions choisies aléatoirement et dont l'angle est rasant par rapport à la surface touchée causent un cosinus proche de zéro. Les rayons subséquents n'auront aucune contribution pour l'échantillonnage des surfaces  $f_v$ . À l'inverse, l'échantillonnage de l'illumination directe  $L_i$  via les sources de lumière pourra nous donner des contributions, mais il existe tout de même des situations pour lesquelles cet échantillonnage n'est pas tout à fait approprié.

La combinaison de plusieurs techniques d'échantillonnage dont les PDF sont pondérés d'un poids (Fig. 1.6) permet d'avoir une meilleure approximation de la contribution lumineuse dans l'ensemble. La formulation de ce calcul par importance multiple (Veach & Guibas, 1995) ajoute

un facteur de pondération à l'estimateur Monte-Carlo

$$I_{ij} \approx \sum_{s \in S} \frac{1}{N_s} \sum_{n=1}^{N_s} w_s(\bar{x}_{s,n}) \frac{F(\bar{x}_{s,n})}{p_s(\bar{x}_{s,n})}, \quad (1.9)$$

où la valeur du pixel  $ij$  est approximée par un ensemble de techniques de rendu  $S$ , dont les contributions de  $N_s$  échantillons sont pondérées par un poids positif et spécifique à chaque technique  $w_s \in [0...1]$ . Ces poids doivent avoir leur somme égale à un (c.-à-d.  $\sum w_s = 1$ ) et avoir une valeur de 0 lorsque leur probabilité  $p_s$  est nulle.

Rappelons que l'objectif de cette pondération est d'obtenir une estimation dont la variance est minimale. La « balance heuristic » de Veach & Guibas (1995) permet justement de choisir le poids associé aux techniques de façon à minimiser la variance totale en calculant le ratio de la densité  $p_s$  d'une technique  $s$  par rapport à celles des autres  $k$  techniques utilisées :

$$w_s(\bar{x}) = \frac{N_s p_s(\bar{x})}{\sum_k N_k p_k(\bar{x})}. \quad (1.10)$$

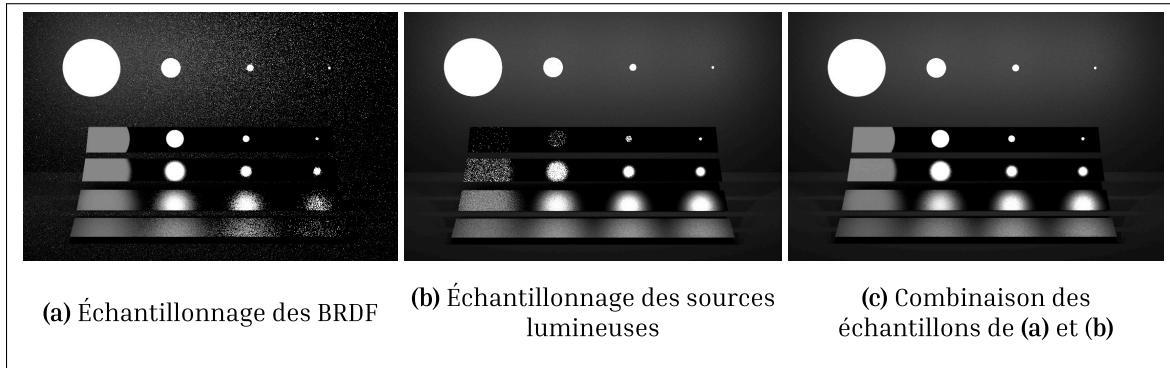


Figure 1.6 Différentes techniques d'échantillonnage sont utilisées avec un estimateur Monte-Carlo. La scène contient quatre sources lumineuses sphériques dont les rayons varient. Il y a quatre plaques métalliques avec des coefficients de rugosité allant de 0.004 à 0.14, disposées de façon à réfléchir la lumière vers la caméra. Les images sont générées avec 768x512 pixels et 100 échantillons par pixel. Les échantillons de (a) sont choisis aléatoirement et proportionnellement à la BRDF des surfaces spéculaires. Les échantillons de (b) sont choisis en échantillonnant d'abord uniformément une source de lumière et ensuite une direction  $w_i$  comprise dans le cône dirigé vers la source choisie. Les échantillons de (c) sont choisis en utilisant une combinaison pondérées des échantillons de (a) et (b).

### 1.2.3 Monte-Carlo par chaînes de Markov

Les chaînes de Markov sont un outil permettant de générer des séquences de nombres aléatoires corrélés. En les combinant à un estimateur Monte-Carlo, nous obtenons Monte-Carlo par chaînes de Markov (MCMC) (Metropolis *et al.*, 1953), un algorithme d'échantillonnage dont les échantillons suivent une distribution donnée. Pour que les nombres aléatoires générés soient cohérents, les chaînes de Markov qui sont utilisées doivent respecter un ensemble strict de conditions.

**Definition 1.1.** *Une chaîne de Markov est une suite de variables aléatoires  $X_n$  dont les valeurs sont tirées d'un ensemble fini  $E$ , où pour chaque valeurs de  $n \in \mathbb{N}$ , on a*

$$P(X_{n+1} = i_{n+1} \mid X_1 = i_1, \dots, X_n = i_n) = P(X_{n+1} = i_{n+1} \mid X_n = i_n). \quad (1.11)$$

*L'ensemble  $E$  d'où la suite  $X_n$  tire ses valeurs est l'espace des états et chaque élément de  $E$  décrit un état possible de la chaîne.*

Autrement dit, pour chaque état  $X_n$  subséquent, en commençant par  $X_0$ , la transition vers le prochain état  $X_{n+1}$  ne dépend que de l'état courant  $X_n$ .

**Definition 1.2.** *Une chaîne de Markov est dite **apériodique** si, quand nous commençons à l'état  $X_i$ , il est impossible de savoir exactement lorsque nous retournerons au même état  $X_i$ . Cet état pourrait aussi bien survenir après 1, 2, 3, 4, ... transitions.*

**Definition 1.3.** *Une chaîne de Markov est dite **invariante** ou **stationnaire** si, quand nous obtenons un échantillon  $X_i$  suivant la distribution  $\pi$ , une transition est assurée de nous retourner un échantillon  $X_{i+1}$  suivant lui aussi la distribution  $\pi$ .*

**Definition 1.4.** *Une chaîne de Markov est dite **ergodique** si, quand la chaîne a subit un nombre  $N$  de transitions suffisamment élevé et fini, nous aurons parcouru l'espace des états au complet.*

**Definition 1.5.** *Une chaîne de Markov est dite **irréductible** si, pour toutes les paires d'états  $i$  et  $j$ , la probabilité de passer de l'état  $i$  à  $j$  est strictement positive.*

**Definition 1.6.** Une chaîne de Markov est dite **réversible** quand sa distribution initiale  $\pi$ ,  $\forall i, j \in E$  satisfait la condition de **balance détaillée** par rapport à une densité de probabilité  $P(X | Y) = T(X | Y)\alpha(X | Y)$ , où  $T$  est la probabilité de transition et  $\alpha$  la probabilité d'accepter l'état :

$$\pi(X_i)P(X_j | X_i) = \pi(X_j)P(X_i | X_j). \quad (1.12)$$

#### 1.2.4 Metropolis-Hastings

Algorithme 1.2 Metropolis-Hastings

```

1 Algorithme : Metropolis-Hastings
  Input : Nombre d'échantillons  $N \in \mathbb{R}_{\geq 0}$ 

2 Choisir l'échantillon initial  $X_0$ ;
3 for all samples  $X_n \in (n = 1, \dots, N)$  do
4   Générer la proposition  $v \leftarrow \text{mutate}(X_n)$ ;
5   Calculer la probabilité d'acceptation  $\alpha(X_n \rightarrow v)$ ;
6   if  $\alpha > \text{rand}(0 \dots 1)$  then
7     Accepter la proposition  $X_{n+1} \leftarrow v$ ;
8   else
9     Rejeter la proposition  $X_{n+1} \leftarrow X_n$ ;
10  end if
11 end for
```

L'algorithme Metropolis-Hastings (MH) (Hastings, 1970) est un algorithme MCMC utilisé dans plusieurs domaines afin de générer des nombres aléatoires. Un pseudocode est présenté dans l'algorithme 1. Pour commencer, nous débutons avec l'état initial  $X_0$  (ligne 2), et à chaque itération  $n$ , appliquons une mutation sur l'état courant  $X_n$  pour obtenir l'état suivant  $v$  (ligne 4). Ces mutations permettent en pratique d'assurer l'ergodicité (Déf. 1.4) de la chaîne en naviguant dans le voisinage d'un état, communément décrit comme l'exploration locale. La probabilité d'accepter un nouvel état  $a(X_n \rightarrow v)$  sachant l'état courant est ensuite calculée (ligne 5). Sa

valeur est donnée par

$$a(X \rightarrow v) = \min \left( 1, \frac{\pi(v) \cdot \mathcal{T}(v \rightarrow X)}{\pi(X) \cdot \mathcal{T}(X \rightarrow v)} \right), \quad (1.13)$$

où  $\pi$  est la fonction cible pilotant le processus Markovien et  $\mathcal{T}(X \rightarrow v)$  la fonction de transition provisoire donnant la densité de probabilité de générer  $v$  lorsque l'état courant est  $X$  (Veach & Guibas, 1997). L'objectif de cette formule est de maximiser la vitesse de convergence de la chaîne vers l'état stationnaire (Déf. 1.3). Les échantillons générés par ce processus sont distribués proportionnellement à la fonction cible  $\pi$  tant que l'échantillon initial est lui aussi proportionnel à  $\pi$ . Une façon d'assurer cette condition est d'utiliser une « burning phase » dans laquelle on tire des échantillons jusqu'à l'obtention d'un premier échantillon proportionnel à  $\pi$ . Il est aussi commun d'éliminer le biais initial en calculant la constante de normalisation  $b$  (Éq. 1.15) avec un algorithme de tracé de rayon (p. ex. unidirectionnel). La constante est ensuite appliquée aux estimations Monte-Carlo. Avec l'une de ces techniques, nous assurons la condition d'invariance (Déf. 1.3) :

$$p(X) = \frac{\pi(X)}{b}, \quad (1.14)$$

$$b = \int \pi(X) d\Omega \approx \frac{1}{N} \sum_{n=1}^N \frac{\pi(X_0, n)}{p(X_0, n)}. \quad (1.15)$$

### 1.3 Metropolis Light Transport

La première utilisation d'un algorithme MCMC en infographie a été proposée par Veach & Guibas (1997) avec Metropolis Light Transport (MLT). MLT utilise Metropolis-Hastings et l'applique directement sur l'espace des chemins (Sik & Krivanek, 2020). La chaîne va tirer ses états de l'ensemble des chemins de lumière  $\Omega$ , et un état correspond à un chemin  $\bar{x}$ . Lorsque des états sont proposés, l'algorithme va perturber des chemins et assurer la condition d'ergodicité (Déf. 1.4) avec une mutation particulière. En fonction des surfaces rencontrées par le chemin, différentes mutations vont être plus efficaces. La figure 1.7 présente les différentes mutations de MLT ; par exemple, la mutation lentille (**a**) est particulièrement bien adaptée pour des chemins contenant des interactions spéculaires directement visibles par la caméra. À l'inverse, la perturbation

caustique (**b**) est mieux adaptée aux chemins où la lumière se concentre sur une caustique (généralement dû à une interaction avec une surface diélectrique). D'autres mutations ont été développées plus tard afin de couvrir des scénarios d'illumination spécifique. Entre autres, le « manifold exploration » de Jakob & Marschner (2012) permet de mieux gérer les caustiques engendrées par les interactions spéculaires complexes. Des méthodes plus avancées comme la perturbation « geometry aware » de Otsu, Hanika, Hachisuka & Dachsbacher (2018) limite la taille de la mutation de façon adaptative afin d'éviter les chemins dont la contribution est nulle. Ultimement, il n'y a pas de stratégie parfaite et les différentes interactions avec les matériaux d'une scène dicteront comment muter un chemin de façon appropriée.

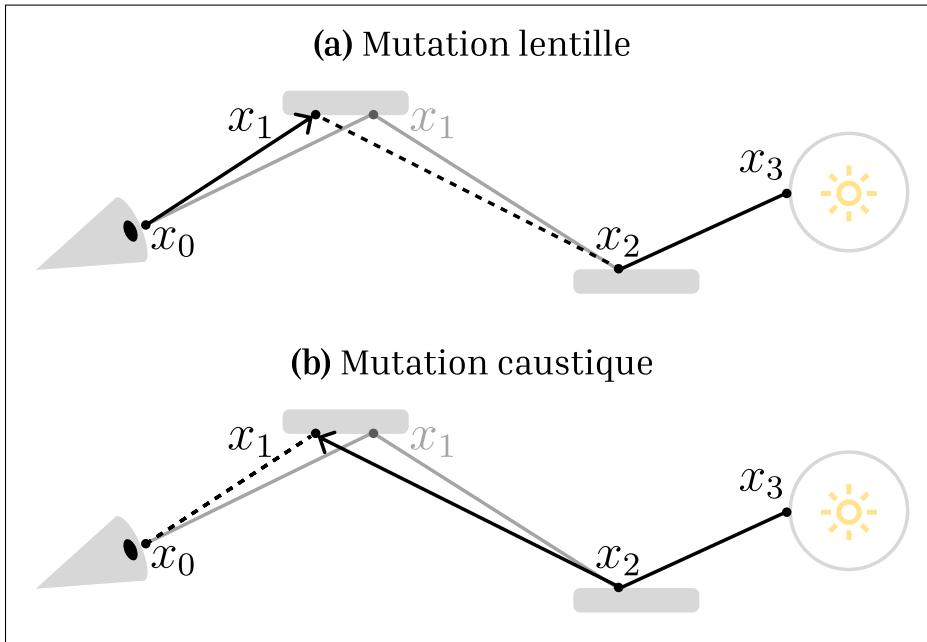


Figure 1.7 Les différentes mutations utilisées par MLT. La mutation lentille (**a**) change un chemin visible à partir de la caméra. La mutation caustique (**b**) fait l'inverse et change un chemin visible à partir d'une source de lumière.

### 1.3.1 Espace primaire

Si l'espace des chemins  $\Omega$  est l'ensemble explicite contenant la géométrie de tous les chemins de lumière possibles, l'espace primaire est obtenu en décrivant tout chemin  $\bar{x}$  par rapport à la

suite de nombres aléatoires  $u$  qui le génère. Cette idée est rendue possible grâce au mappage direct qui existe entre l'espace primaire et l'espace des chemins  $m : \mathcal{P} \rightarrow \Omega$  (Fig. 1.8).

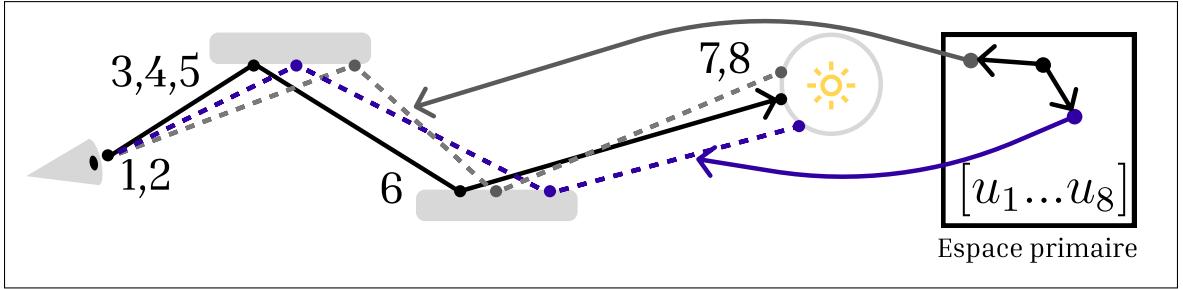


Figure 1.8 Lien entre l'espace primaire et les chemins qu'il génère. Ici, de petites perturbations sur les nombres aléatoires de l'espace primaire provoquent un changement dans les chemins qui touchent la source de lumière. Les nombres générés à chaque segment sont numérotés et correspondent aux valeurs de l'espace primaire  $[u_1 \dots u_8]$ .

Cette reformulation de l'espace donne lieu à une adaptation de l'équation 1.7

$$I(m(u)) = \int_{\mathcal{P}} \frac{F(m(u))}{p(m(u))} du, \quad (1.16)$$

où nous tenons compte du changement de densité de l'échantillonnage avec le jacobien  $1/p(u)$ , équivalent à l'inverse de la densité de probabilité d'échantillonner le chemin  $\bar{x}$  étant donné la fonction de mappage  $m$  (Sik & Krivanek, 2020). L'équation 1.16 peut être approximée en utilisant MCMC

$$I(m(u)) \approx \frac{b}{N} \sum_{i=1}^N \frac{F(m(u))}{p(m(u))F^*(m(u))}, \quad (1.17)$$

où  $b$  est la constante de normalisation estimée (Éq. 1.15),  $N$  le nombre d'échantillons et  $F^*(m(u))$ , la luminance issue de  $F(m(u))/p(m(u))$ .

### 1.3.2 Dynamique de Langevin

En utilisant la dynamique de Langevin pour modéliser l'état d'une chaîne de Markov comme une particule animée dans l'espace, on peut diriger les mutations de l'espace des états vers le gradient  $\nabla$  où la contribution sera plus élevée, résultant en un meilleur taux d'acceptation des

états proposés. À la base, un système dynamique de Langevin est un processus continu décrit par une équation différentielle stochastique qui modélise le mouvement aléatoire de plusieurs particules (Ermak & McCammon, 1978)

$$M\ddot{X} = -\nabla U(X) - \zeta \dot{X} + \sqrt{2\zeta K_B T} R(t), \quad (1.18)$$

où  $M$  est la masse de la particule,  $\zeta$  le coefficient de friction de la substance dans laquelle se retrouve la particule,  $\nabla$  est l'opérateur de gradient,  $U(X)$  est le potentiel d'interaction de la particule,  $\ddot{X}$  est l'accélération de la particule,  $\dot{X}$  est la vitesse de la particule,  $T$  est la température,  $K_B$  est la constante de Boltzmann et  $R(t)$  est un bruit aléatoire.

La dynamique brownienne (Ermak & McCammon, 1978) permet de grandement simplifier l'équation 1.18 et ne requiert pas de dérivée seconde. C'est ce type de dynamique qui est utilisée par des algorithmes tels que Langevin Monte-Carlo (LMC) (Roberts & Tweedie, 1996) ou Monte-Carlo Hamiltonien (Duane, Kennedy, Pendleton & Roweth, 1987) afin de générer les états dans la fonction de transition  $\mathcal{T}$ . On se retrouve donc avec une équation différentielle de premier ordre

$$\dot{X} = -\frac{D}{K_B T} \nabla U(X) + \sqrt{2D} R(t), \quad (1.19)$$

où  $D$  est un coefficient de diffusion. Les chemins de lumière générés par les algorithmes MCMC plus traditionnels utilisent une fonction de transition  $\mathcal{T}$  soumise à une fonction d'acceptation (Éq. 1.13) qui est principalement influencée par la contribution lumineuse. Les systèmes dynamiques vont de l'avant avec une fonction de transition  $\mathcal{T}$  qui génère des échantillons proportionnellement aux gradients décrits par un système physique. La difficulté de cette technique provient historiquement de la façon avec laquelle on parvient à calculer ces gradients.

## 1.4 Rendu dans le domaine des gradients

Le rendu dans le domaine des gradients permet de grandement réduire le bruit uniforme en calculant les différences d'intensité lumineuse entre pixels subséquents. Cette information est utilisée pour évaluer la fonction cible  $\pi$  en la pondérant par la valeur de ces gradients. L'idée

est de concentrer les échantillons dans des régions à haute variation d'intensité lumineuse pour mieux pouvoir en capturer la différence. Cela permet entre autres d'appliquer des algorithmes de reconstructions qui engendrent des intensités uniformes là où la variation du gradient est nulle. Il en résulte donc que le bruit perçu sur ces surfaces diminue grandement, surtout lorsque nous arrivons à calculer une bonne estimation des gradients. Ainsi, nous obtenons d'abord l'estimation des gradients horizontaux  $I_{dx}$  et verticaux  $I_{dy}$  de l'image (Fig. 1.9). Nous obtenons en même temps une version grossière de l'image  $I_G$  avec les intensités calculées. L'espace des chemins étendu  $\Omega'$  ajoute aux échantillons un décalage  $\delta_x, \delta_y$  vers un pixel du voisinage. Ce nouvel espace devient donc

$$\Omega' = \Omega \times \{(1, 0), (-1, 0), (0, 1), (0, -1)\}. \quad (1.20)$$

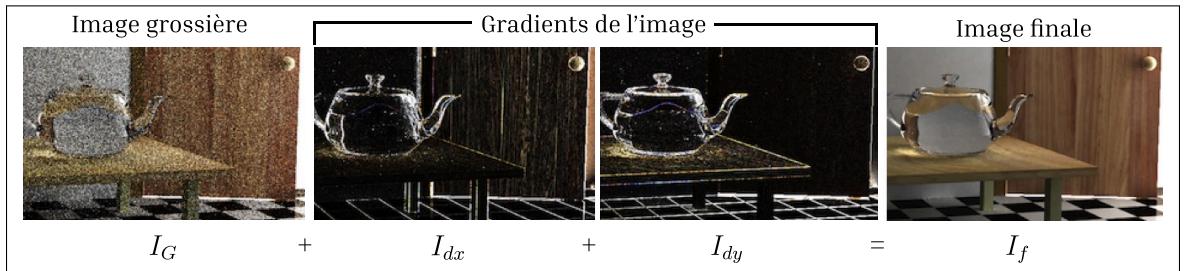


Figure 1.9 L'image grossière  $I_G$  est combinée aux gradients  $I_{dx}$  et  $I_{dy}$  pour résoudre un problème de minimisation donnant l'image finale  $I_f$ . Images tirées de Lehtinen *et al.* (2013)

#### 1.4.1 Fonction de décalage

Une fonction de décalage permet de muter un chemin  $\bar{x}$  afin que ses coordonnées dans le plan image se déplacent d'un pixel  $(\delta_x, \delta_y)$  dans l'une des quatre directions  $\in \{(1, 0), (0, 1), (-1, 0), (0, -1)\}$ , sans que son origine  $x_0$  change. Ce décalage  $T_{\delta_x, \delta_y}(\bar{x})$  est décomposé en plusieurs parties :

1. L'origine de la caméra  $x_0$ , identique pour  $\bar{x}$  et  $T_{\delta_x, \delta_y}(\bar{x})$ .
2. Les interactions spéculaires décalées  $S_d = \{x_1, \dots, x_b\}$  qui se terminent dès la première interaction diffuse  $x_b$  et dont le prochain sommet  $x_{b+1}$  du chemin de base est diffus. On

retrouve aussi les interactions spéculaires  $S_f = \{x_{b+1}, \dots, x_{c-1}\}$  suivant  $x_{b+1}$ . En fonction de la scène,  $S_d$  et  $S_f$  peuvent ne contenir aucune valeur.

3. Les sommets de base inchangés  $\{x_c, \dots, x_k\}$ , terminant sur une source de lumière.

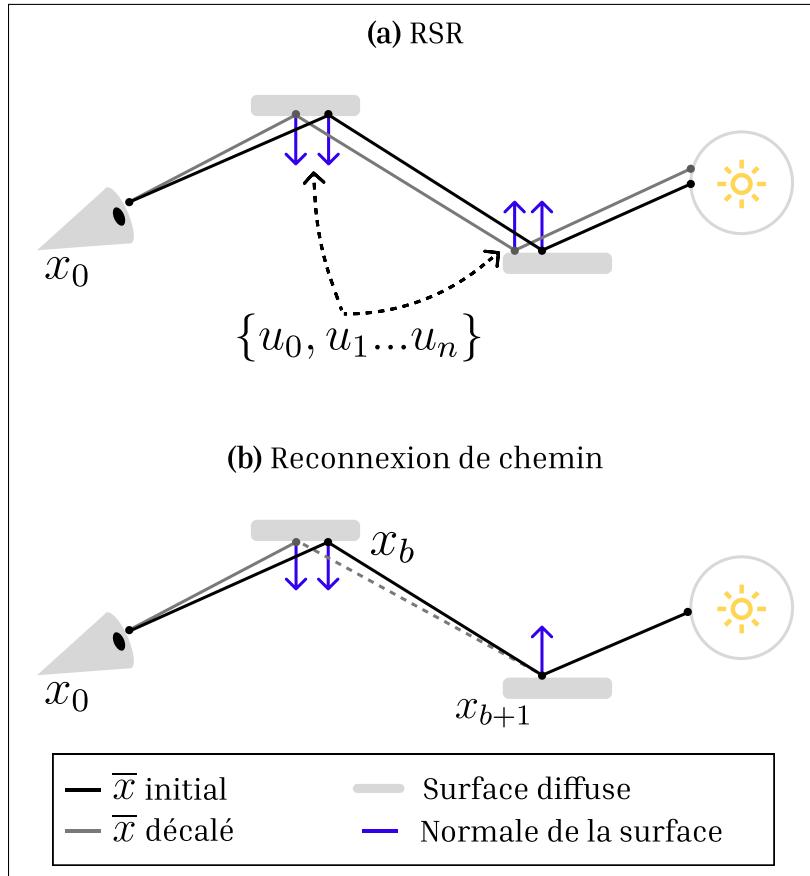


Figure 1.10 Différentes fonctions de décalage sont illustrées. La technique en **(a)** est plus simple d'implémentation mais comporte un surcoût computationnel important puisque chaque décalage doit tracer un nouveau rayon en entier. Avec la reconnexion de chemin **(b)**, ce coût est grandement amorti en réutilisant le chemin de base dès que possible.

Différentes techniques permettent de faire ce type de décalage avec des résultats variables. Par exemple, « random sample replay » (RSR) **(a)** (Fig. 1.10) exploite la réutilisation des nombres aléatoires  $u_0, u_1 \dots u_n$  générés par le chemin de base. L'hypothèse est qu'un chemin  $\bar{x}$  devrait changer très peu lorsqu'il est décalé d'un pixel. En pratique, ce n'est pas tout le temps le cas et peut causer une augmentation importante de la variance. Une meilleure approche **(b)** (Fig. 1.10) consiste à reconnecter les chemins décalés au chemin de base dès qu'une interaction valide  $x_b$

est rencontrée. Une fois reconnecté, le chemin décalé réutilise le chemin de base. Cela entraîne cependant un changement dans la densité d'échantillonnage de l'espace des états et doit être corrigé par le déterminant du jacobien  $d\{\mu \circ T_{1,0}\}/d\mu$ .

#### 1.4.2 Reconstruction

Les gradients  $I_{dx}$ ,  $I_{dy}$  et l'image grossière  $I_G$  sont utilisés à la toute fin pour reconstruire l'image finale  $I_f$ . En ayant une approximation de la variation des intensités dans les axes horizontaux et verticaux, il est possible de résoudre un problème de minimisation qui cherche à faire converger  $I_G$  vers  $I_f$  où ses gradients seront aussi proches que possible de  $I_{dx}$ ,  $I_{dy}$ . La reconstruction de Poisson est l'un des algorithmes qui permettent d'obtenir de bons résultats pour cette étape. L'idée est de minimiser la différence entre les gradients de l'image  $I_f$  et les gradients  $I_{dx}$ ,  $I_{dy}$  obtenus itérativement :

$$\underset{I_f}{argmin} \left( \left\| \begin{pmatrix} H_{dx} I_f \\ H_{dy} I_f \end{pmatrix} - \begin{pmatrix} I_{dx} \\ I_{dy} \end{pmatrix} \right\|_n^n + \|\alpha(I - I_G)\|_n^n \right), \quad (1.21)$$

où  $H_{dx}$ ,  $H_{dy}$  contiennent les différences finies de  $I_f$  pour calculer le gradient,  $\alpha$  un paramètre de pondération du poids de l'image grossière dans la reconstruction, et  $n$  la valeur qui choisi entre les normes L1 et L2. Cependant, l'utilisation de la norme L2 est souvent privilégiée, car elle permet de retirer le biais obtenu par l'usage de la norme L1 (Fig. 1.11).

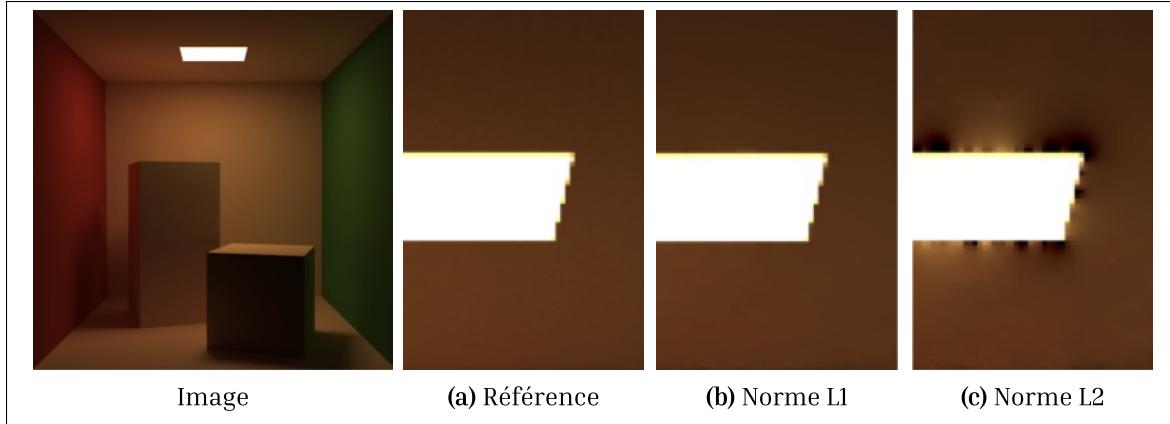


Figure 1.11 Bien qu'elle soit non biaisée, la reconstruction de poisson utilisant la norme L2 est particulièrement susceptible aux artefacts aux alentours des sources lumineuses (c).

En utilisant la norme L1 (b), ce problème est retiré aux dépens d'un biais, ce qui est généralement à éviter. Images tirées de Lehtinen *et al.* (2013)

## CHAPITRE 2

### TRAVAUX CONNEXES

Nos travaux s’inspirent de plusieurs autres ouvrages, plus particulièrement dans les domaines des chaînes de Markov et du rendu MCMC, du calcul des gradients et des dérivées analytiques de la contribution lumineuse. Les mutations dans l'espace primaire utilisées par « Primary Sample Space Metropolis Light transport » (PSSMLT) (Kelemen *et al.*, 2002) sont faciles à mettre en place et nous donnent un référentiel de base pour concevoir des mutations plus sophistiquées. Par exemple, en adaptant la perturbation tout au long de la chaîne (Szirmay-Kalos & Szécsi, 2017), on peut améliorer la convergence de nos algorithmes. Des mutations adaptatives basées sur la dynamique de Langevin comme MALA (Roberts & Tweedie, 1996) puisent dans cette idée afin de diriger les nombres de l'espace primaire vers le gradient de la fonction cible. Finalement, des mutations basées sur les gradients de l'image calculées dans GDMLT (Lehtinen *et al.*, 2013) permettent de résoudre le problème de convergence sous un autre angle tout en utilisant des dérivées comme pour MALA.

#### 2.1 MCMC avec mutation isotrope

Les travaux présentés dans cette section ont la particularité d'avoir des mutations isotropes. Elles sont relativement faciles à mettre en place, car elles ne changent pas au cours de l'exécution de la chaîne de Markov et ont généralement des fonctions de transition symétriques.

##### 2.1.1 PSSMLT

PSSMLT (Kelemen *et al.*, 2002) est un algorithme MCMC qui privilégie un espace des états simplifiés. Tandis que MLT change directement les chemins de lumière avec des mutations adaptées aux interactions avec les surfaces, PSSMLT propose d'appliquer une mutation uniforme sur le vecteur de nombres aléatoires  $u$  (Fig. 1.8).

Il y a deux types de perturbations utilisées pour diriger la marche aléatoire : des mutations locales et des mutations globales. Lors de la génération d'un nouvel échantillon  $u_{n+1}$ , la mutation

locale échantillonne uniformément une distribution exponentielle pour choisir un déplacement  $du$  comme dans la mutation lentille de MLT (Veach & Guibas, 1997)

$$du = r_2 \cdot \exp(-\log(r_2/r_1)U),$$

$$u_{n+1} = u_n \pm du,$$

où  $r_1$  et  $r_2$  sont deux valeurs laissées au choix de l'utilisateur tel que  $r_1 < r_2$  (dans notre cas  $r_1 = 1/1024$  et  $r_2 = 1/64$ ), et  $U$  est un nombre aléatoire généré uniformément entre 0 et 1.

Par exemple, on obtient un petit déplacement horizontal ou vertical sur le plan image pour les deux premiers nombres  $u_1$  et  $u_2$ . Cela nous permet d'explorer des régions où la contribution lumineuse est plus élevée en suivant la règle d'acceptation de MH (Éq. 1.13). La mutation globale vient aléatoirement choisir un nouvel échantillon qui n'est pas corrélé de façon à assurer la condition d'ergodicité de la chaîne (Déf. 1.4). Par exemple, cela donnerait un échantillon qui n'est pas nécessairement dans le voisinage de l'échantillon précédent.

### Algorithme 2.1 MLT dans l'espace primaire

#### 1 Algorithme : PSSMLT

**Input :** Nombre d'échantillons  $N \in \mathbb{R}_{\geq 0}$ , ratio de mutations larges  $p\_large$

```

2 Initialiser l'image  $I$  à zéro;
3 Choisir l'échantillon initial  $X_0$ ;
4 for all samples  $X_n \in (n = 1, \dots, N)$  do
5   Choisir le type de mutation  $large \leftarrow rand(0 \dots 1) < p\_large$ ;
6   Générer la proposition avec la mutation sélectionnée  $v \leftarrow mutate(X_n, large)$ ;
7   Calculer la probabilité d'acceptation  $\alpha \leftarrow min(1.0, F^*(X_n)/F^*(v))$  ;
8   if  $\alpha > rand(0 \dots 1)$  then
9     | Accepter la proposition  $X_{n+1} \leftarrow v$  ;
10    else
11      | Rejeter la proposition  $X_{n+1} \leftarrow X_n$  ;
12    end if
13    Accumuler la luminance de  $F^*(v)$  dans  $I$ 
14 end for
15 return  $I/N$ 
```

Un pseudocode est présenté dans l'algorithme 2.1. Comme dans MH, nous débutons avec un échantillon initial  $X_0$  (ligne 3). Le type de mutation est choisi en échantillonnant uniformément  $\mathcal{N}(0...1)$  (ligne 5) pour ensuite être appliqué sur le vecteur de nombres aléatoires courant (ligne 6). Cela nous donne un nouveau chemin de lumière dont les nombres aléatoires sont dans  $v$ . La luminance de l'état courant  $F^*(X_n)$  est ensuite comparée à celle du nouvel état  $F^*(v)$  pour calculer la probabilité d'acceptation  $\alpha$  (ligne 7). Cette formulation est possible grâce à la mutation symétrique. Nous échantillonnons uniformément  $\mathcal{N}(0, 1)$  et comparons la valeur obtenue à  $\alpha$  pour accepter ou rejeter le nouvel état (ligne 8) puis mettre à jour l'état courant (lignes 9 et 11). Finalement, nous accumulons l'état  $F^*(v)$  dans l'image (ligne 13) qui sera normalisée à la toute fin avant d'être retournée (ligne 15).

### 2.1.2 GDMLT

« Gradient Domain Metropolis Light Transport » (GDMLT) est un algorithme MCMC qui accumule les gradients de l'image tout au long d'une marche aléatoire pilotée par des chaînes de Markov. Le gradient localisé sur un pixel  $j$  se calcule par la différence avec l'intensité des pixels voisins ;  $I_j^{dx} = I_{j+1} - I_j$  pour l'axe horizontal et pareillement pour l'axe vertical. Cela nous donne la formulation pour l'intensité d'un pixel voisin et aussi pour la différence avec le pixel  $j$

$$I_{j+1} = \int_{\Omega} h_j(\bar{x}) F^*(T_{1,0}(\bar{x})) \frac{d\{\mu \circ T_{1,0}\}}{d\mu}(\bar{x}) d\mu(\bar{x}), \quad (2.1)$$

$$I_j^{dx} = I_{j+1} - I_j = \int_{\Omega} h_j(\bar{x}) \overbrace{\left[ F^*(T_{1,0}(\bar{x})) \frac{d\{\mu \circ T_{1,0}\}}{d\mu}(\bar{x}) - F^*(\bar{x}) \right]}^{F_{1,0}(\bar{x})} d\mu(\bar{x}). \quad (2.2)$$

Le déterminant du jacobien  $d\{\mu \circ T_{1,0}\}$  permet de tenir compte du fait que les chemins décalés vont engendrer un changement de la densité des chemins dans certaines régions de l'espace des chemins  $\Omega$ . Un chemin augmenté  $\bar{z} = \{\bar{x}, \delta_x, \delta_y\}$  est un chemin de lumière auquel on ajoute un déplacement horizontal et vertical. Le chemin issu du déplacement  $\delta_x, \delta_y$  se calcule par la fonction de décalage  $\bar{z} = T_{\delta_x, \delta_y}(\bar{x})$ . Cela mène à la formulation de la contribution lumineuse

d'un chemin augmenté

$$F^*(\bar{z}) = n(\bar{z}) \begin{cases} F_{1,0}(\bar{x}), & (\delta_x, \delta_y) = (-1, 0) \\ -F_{-1,0}(\bar{x}), & (\delta_x, \delta_y) = (1, 0) \\ F_{0,1}(\bar{x}), & (\delta_x, \delta_y) = (0, 1) \\ -F_{0,-1}(\bar{x}), & (\delta_x, \delta_y) = (0, -1) \end{cases}, \quad (2.3)$$

où  $n(\bar{z})$  prendra la valeur  $w_{ij}$  ou  $w_{ji}$  selon la fonction de décalage utilisée ; 0.5 dans le cas de « random sample replay » (RSR). Lorsque la contribution d'un chemin  $\bar{z}$  est accumulée dans nos images, le gradient doit être calculé en fonction des bons index  $\delta_x, \delta_y$ . Les filtres  $h_j^x(\bar{z})$  et  $h_j^y(\bar{z})$  décidant du pixel dans lequel accumuler les contributions tiennent compte du fait que les valeurs  $\delta_x = -1$  et  $\delta_y = -1$  estiment les différences finies pour  $I_j - I_{j-1}$  plutôt que  $I_{j+1} - I_j$  et doivent être accumulées dans le pixel  $j - 1$

$$h_j^x(\bar{z}) = \begin{cases} h_j(\bar{x}), & (\delta_x, \delta_y) = (-1, 0) \\ h_{j+1}(\bar{x}), & (\delta_x, \delta_y) = (1, 0) \\ 0, & \delta_x = 0 \end{cases}. \quad (2.4)$$

La fonction cible utilisée pour le calcul de l'acceptation d'un état (Éq. 1.13) doit être légèrement modifiée par rapport à MLT de façon à tenir compte de la valeur des gradients. Étant donné que l'étape de reconstruction est sensible aux gradients bruités, la nouvelle fonction mitige ce problème en tirant davantage d'échantillons dans les régions à gradients élevés. D'ainsi, nous avons

$$\pi(\bar{z}) = \sum \left( \text{abs}(F_{ij}^*(\bar{z})) \right) + \alpha \cdot 0.25 \cdot \text{abs}(F^*(\bar{x})), \quad (2.5)$$

où la contribution lumineuse du chemin de base est pondérée par un paramètre utilisateur  $\alpha \cdot 0.25$  pour tenir compte de son apparition dans les quatre décalages de  $\Omega'$ . Nous y ajoutons la valeur absolue de la contribution des gradients (pouvant être négatifs dans des régions de diminutions de l'intensité lumineuse) et cela nous donne la valeur finale pour un chemin  $\bar{x}$ .

## 2.2 MCMC avec mutation adaptive

Les travaux présentés dans cette section utilisent des mutations adaptatives dont le noyau de transition n'est pas symétrique. Cela permet entre autres de perturber les nombres aléatoires proportionnellement à des distributions plus complexes. Ce type de mutation permet ainsi d'explorer l'espace local de régions anisotropes.

### 2.2.1 Stratification

Les mutations proposées par Kelemen *et al.* (2002) ont le désavantage d'être isotropes. Qui plus est, la taille des perturbations ne change pas en fonction de la région explorée, menant à une marche aléatoire uniforme parfois trop lente dans des sections de l'image moins importantes (p.ex. où la luminance est faible). Inversement, des mutations trop grandes dans des régions importantes peuvent nous amener à explorer des régions de l'image qui sont moins intéressantes. Il en résulte que l'acceptation des mutations locales est rarement optimale. Afin de pallier ces problèmes, Szirmay-Kalos & Szécsi (2017) proposent d'utiliser une mutation asymétrique et adaptative utilisant le niveau d'acceptation courant et la fonction cible pour en changer la taille.

Une bonne stratification des échantillons, générés de façon à suivre la densité de probabilité de l'image, permet d'améliorer le processus d'acceptation de nos mutations. Une façon d'y parvenir est en jouant sur la taille de la mutation. Cependant, cela peut s'avérer désavantageux ; par exemple, lorsque les mutations sont trop petites, le biais initial est plus important. Inversement, lorsque les mutations sont trop grandes, l'acceptation d'un nouvel échantillon est moins probable et entraîne des duplicitas. Ainsi, la mutation stratifiée de Szirmay-Kalos & Szécsi (2017) utilise des bornes  $[s_{min}, s_{max}]$  entre lesquelles la valeur  $s$  d'une mutation  $X(t + 1) = X(t) + s$  sera comprise avec une densité de probabilité  $p(s)$  (Fig. 2.1). En augmentant la valeur d'un paramètre  $\alpha$ , il sera plus probable que la mutation soit proche de  $s_{min}$ , ce qui donne lieu à une dépendance linéaire entre ce paramètre et la fonction cible  $\pi$  normalisée par le facteur  $b$  :

$$\alpha = K(t) \frac{\pi}{b}. \quad (2.6)$$

Le paramètre  $K$  permet de contrôler l'adaptativité de la mutation afin d'obtenir une valeur  $s$  fortement asymétrique (en jaune dans la Fig. 2.1) ou une valeur  $s$  plus uniforme comme proposé par Kelemen *et al.* (2002)

$$K(t+1) = K(t) + \frac{\lambda}{t}(a_{cible} - a_t), \quad (2.7)$$

où l'adaptation est obtenue avec  $a_{cible}$  et  $a_t$ , les taux d'acceptation désiré et courant pour la mutation locale, et  $\lambda$  pour contrôler la rapidité de convergence vers  $a_{cible}$ . Une valeur de  $\lambda = 10$  est utilisée par Szirmay-Kalos & Szécsi (2017).

Finalement, nous pouvons obtenir la taille  $s$  de la mutation avec une fonction exponentielle, étant donné qu'elle répond à deux critères importants : pour une valeur de  $\pi$  fixe, la densité décroît avec  $s$  et a une intégrale inversible :

$$s = \frac{1}{\alpha} \log \left( \exp(-\alpha s_{min}) - \frac{\alpha r}{\beta} \right), \quad (2.8)$$

où  $r$  est un nombre aléatoire proportionnel à  $\mathcal{N}(0, 1)$ . Intuitivement, la mutation aura plus de chance d'être proche de  $s_{min}$  lorsque  $\pi$  est élevé, faisant en sorte que l'exploration locale est adaptée par rapport à la fonction cible.

### 2.2.2 MALA

« Metropolis-Adjusted Langevin Algorithm » (Roberts & Tweedie, 1996) simule la dynamique de Langevin (section 1.3.2) pour diriger une marche aléatoire dans la direction des gradients. Cela fait en sorte que l'accumulation des échantillons se concentre aux endroits qui présentent des variations d'intensité plus importantes et se traduit par des mutations adaptatives telles que décrites dans la section 2.2.1. Nous pouvons générer des chemins de lumière avec une légère reformulation de l'équation 1.19 afin qu'elle s'applique mieux à l'échantillonnage dans l'espace primaire

$$dX(t) = \frac{1}{2} \nabla \pi(X(t)) dt + dW(t), \quad (2.9)$$

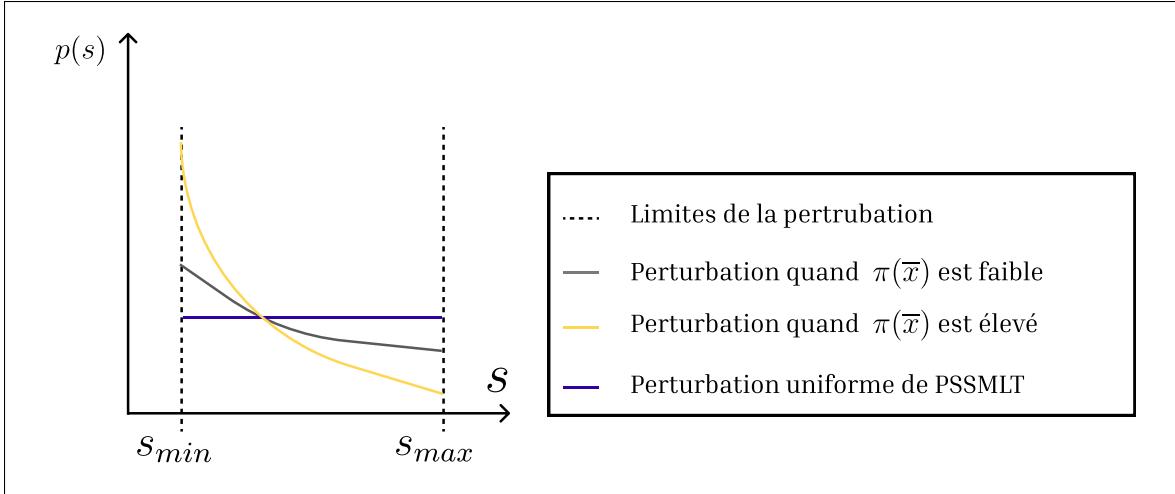


Figure 2.1 La taille de la mutation stratifiée est adaptative. En jaune, la taille a une plus grande probabilité d'être petite lorsque la fonction cible est élevée. En gris, la taille, lorsque la fonction cible est faible, se rapproche de la mutation PSSMLT uniforme en bleu.

où la suite de nombres aléatoires  $X(t)$  générée par la chaîne de Markov est proportionnelle au gradient  $\nabla \log \pi(X(t))$  et au terme de mouvement brownien  $W(t)$ .

Comme en simulation physique, l'équation est discrétisée de façon à pouvoir l'utiliser dans nos programmes. L'une des formulations répandues est la méthode d'Euler (Maruyama, 1955) qui donne lieu à une légère modification de la chaîne de Markov pour que la mutation tienne compte

$$X(t+1) = X(t) + \frac{1}{2}\epsilon \nabla \pi(X(t)) + \sqrt{\epsilon} W, \quad (2.10)$$

où  $\epsilon$  influence la taille du déplacement vers le gradient et  $W$  est un nombre aléatoire échantillonné proportionnellement à une distribution normale  $\mathcal{N}(0, 1)$ . L'un des problèmes avec cette formulation est que les erreurs de discrétisation vont s'accumuler et briser la condition de stationnarité de la chaîne (Déf. 1.3), puisqu'elle ne va pas converger exactement vers la distribution du processus continu (Luan, Zhao, Bala & Gkioulekas, 2020). Une façon de corriger cette erreur est en ayant recours à la règle d'acceptation de Metropolis-Hastings (Éq. 1.13) pour accepter ou rejeter les différents états proposés dans la chaîne. La fonction de transition provisoire  $\mathcal{T}_{mala}$  de

MALA utilise une densité de probabilité asymétrique :

$$\mathcal{T}_{mala}(x' | x) \propto \exp\left(-\frac{1}{4\epsilon}||x' - x - \epsilon \nabla \log \pi(x)||_2^2\right). \quad (2.11)$$

Finalement, les mutations de l'espace des états vont tenir compte à la fois du gradient et du terme de mouvement brownien. Par exemple, une mutation de l'espace primaire (Fig. 2.2) va prendre l'état courant, diriger le nouvel échantillon vers les gradients  $du_0$  et  $du_1$ , pour finalement échantillonner une fonction gaussienne de taille  $s$  donnant lieu à un mouvement brownien  $\bar{W}$  centré à cet endroit.

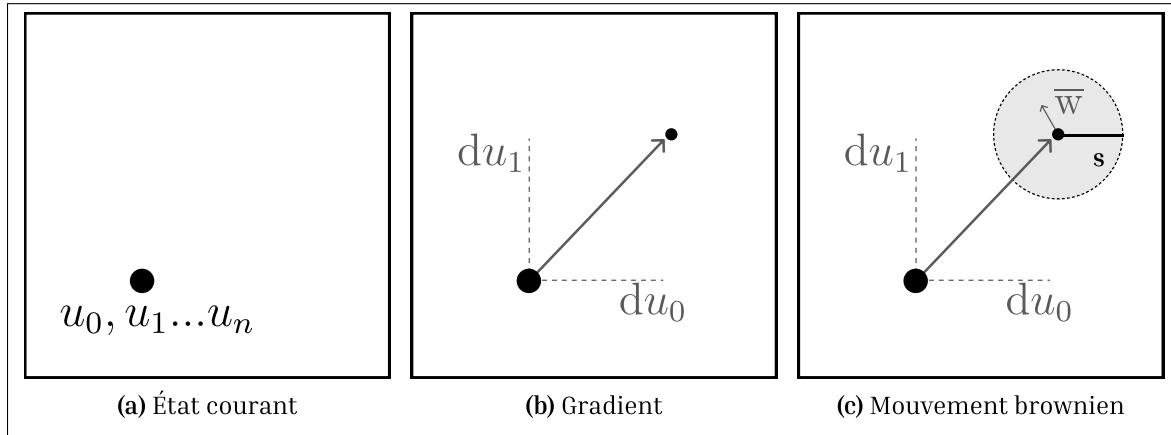


Figure 2.2 L'état courant donne le point de départ (a). Les nombres de l'espace primaire se déplacent dans la direction de leur gradient (b). Le terme de mouvement brownien (c) est échantillonné aléatoirement dans une fonction gaussienne de taille  $s$  centrée en  $[u_0, u_1] + 0.5\epsilon[du_0, du_1]$ . Cela nous permet d'obtenir la valeur de la mutation.

## CHAPITRE 3

### DESCRIPTION DE LA MÉTHODE

Notre approche consiste à utiliser l'espace primaire et le mécanisme de « lazy evaluation » décrit par Kelemen *et al.* (2002) pour tirer des échantillons. Cette approche garde en mémoire les nombres aléatoires afin de les modifier au besoin en fonction de la dimension du vecteur nécessaire pour générer un chemin de lumière. Nous définissons une interface `PrimarySpaceMutation` et utilisons l'injection de dépendance pour changer le comportement des perturbations :

```
interface PrimarySpaceMutation {  
    func mutate(u: [PrimarySample], i: Int) -> Float  
}
```

L'ajout de différentes mutations nous permet de tester librement plusieurs approches. Par exemple, la mutation MALA (section 2.2.2) avec et sans adaptation décrite par Luan *et al.* (2020) nous permettent de mieux explorer l'espace local des régions anisotropes grâce à la nature asymétrique de la fonction de transition  $\mathcal{T}$ . Enfin, nous nous servons de la théorie du domaine des gradients pour approximer le gradient de la fonction cible par rapport aux deux axes de l'image en ayant recours aux différences finies (p.ex.  $\nabla_{ij} = \frac{I(x)+I(x+1)}{2}$ ). Ce choix est motivé par la facilité avec laquelle nous pouvons réutiliser les variations entre pixels subséquents (deux premiers nombres aléatoires) obtenues avec la fonction de décalage. Se limiter aux deux premières dimensions implique également que nous n'avons pas à évaluer le gradient par rapport à la géométrie du chemin de lumière. Ainsi, nous utilisons les gradients sur le plan image dans les intégrateurs MALA et GDMALA afin de diriger les perturbations dans leurs sens.

### 3.1 Estimation des gradients

Le gradient de la fonction cible nous permet de simuler la dynamique de Langevin, avec laquelle il devient possible de proposer des mutations dirigées vers des régions plus intéressantes de l'image. Les mutations de MALA et l'adaptation de Luan *et al.* (2020) utilisent cette information pour générer des chemins de lumière qui ont plus de chance d'avoir une forte contribution

lumineuse. Pour le calcul des gradients, nous proposons d'utiliser une fonction de décalage de type reconnection de chemin, comme dans GDMLT. Lorsqu'une scène comporte plusieurs surfaces métalliques, il peut arriver que RSR donne des résultats similaires. En pratique, cela vient de la difficulté à reconnecter les chemins dans de telles scènes, bien que la reconnection de chemin soit plus efficace dans des circonstances idéales. Ultimement, nous utilisons la méthode qui fonctionne le mieux et qui permet d'avoir les meilleurs gradients en fonction de la scène. Nous indiquerons quelle fonction de décalage est employée sur chacune des scènes de test. Nous débutons avec une approche simple basée uniquement sur la luminance, identique à celle utilisée par PSSMLT (Kelemen *et al.*, 2002). Ensuite, nous étendons notre algorithme à la fonction cible de GDMLT qui est basée à la fois sur la luminance et sur le gradient des contributions lumineuses. Cela donne lieu à la création d'un noyau de décalage qui permet de concevoir un algorithme basé sur la dynamique de Langevin dans le domaine des gradients (GDMALA).

### 3.1.1 Fonction de décalage

Nous utilisons la reconnection de chemin (Fig. 1.10), ce qui peut présenter certains problèmes lors des interactions avec des surfaces parfaitement spéculaires ou diélectriques. Nous séparons aussi la luminance des sources de lumières directement visibles à partir de la caméra. La raison de cette séparation devient apparente lors de la reconstruction. La source de lumière risque d'entraîner un gradient très fort, ce qui est particulièrement susceptible de créer des artefacts avec l'algorithme de reconstruction itérative que nous utilisons. Aussi, la fonction cible utilisant la luminance vient nécessairement entraîner un nombre important d'échantillons sur les sources lumineuses, ce qui n'est pas avantageux dans notre cas.

La contribution d'un chemin décalé est calculée en deux temps : d'abord par le calcul du direct par « next event estimation » (NEE) et ensuite par l'évaluation de la BRDF. Dans les deux cas, le rayon de lumière peut être dans l'un des états  $\mathcal{R} \in \{\text{failed}, \text{connected}, \text{recent}, \text{fresh}\}$  (Fig. 3.1). L'hypothèse de cette approche est qu'un chemin de lumière décalé d'un seul pixel sera similaire, dans la majeure partie des cas, au chemin de base. L'objectif est de rencontrer

rapidement des interactions permettant la reconnexion et le passage à l'état *connected* afin d'augmenter la corrélation entre les chemins et réduire la variance des gradients.

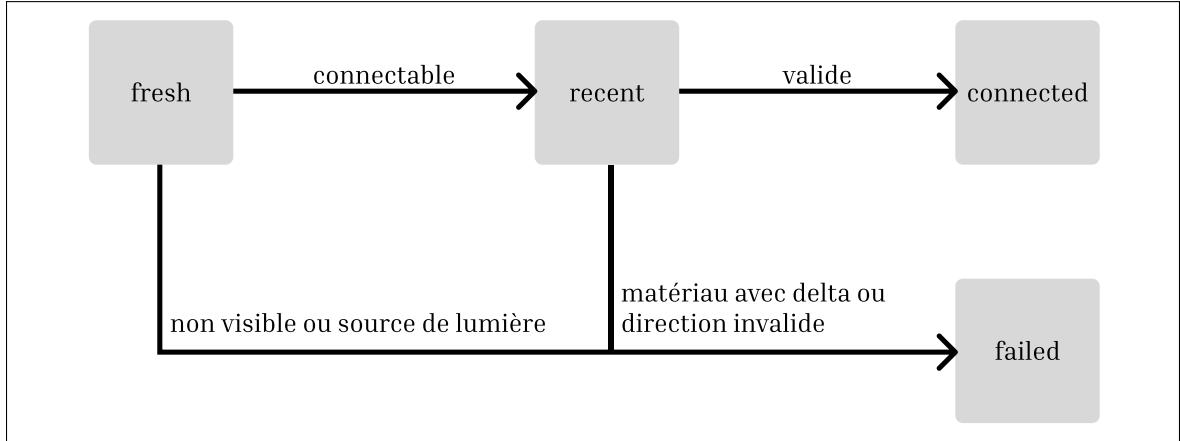


Figure 3.1 Les différents états  $\mathcal{R}$  d'un rayon décalé débutant à l'état *fresh*. En fonction des conditions de connectivité, le rayon passe soit à l'état *recent* ou *failed*. Une fois l'état *failed* atteint, le traçage du rayon prend fin immédiatement. Lorsqu'un rayon connecté récemment touche une surface valide et qu'une direction est générée au point de la connexion, le rayon décalé devient une copie conforme du rayon de base pour le reste du chemin.

Un rayon  $\bar{z}_{ij}$  débute dans l'état « *fresh* » à partir de la caméra, où  $i$  et  $j$  sont le décalage dans les axes horizontal et vertical. Lors de la première interaction permettant une reconnexion, il passe à l'état « *recent* » jusqu'au prochain rebond. À tout moment du tracé de rayon, un échec de reconnexion entraîne l'état « *failed* », qui se traduit par une contribution et un gradient avec une valeur nulle. Finalement, tous les rebonds subséquents seront dans l'état « *connected* » suite à une reconnexion. Dans cet état, la contribution de chacune des interactions du chemin de base  $\bar{x}$  est directement réutilisée dans  $\bar{z}_{ij}$ , ce qui mitige le surcoût associé aux décalages.

### 3.1.2 Fonction cible et noyaux

La fonction de transition provisoire  $\mathcal{T}$  est basée sur la fonction cible  $\pi$ . Dans le cas de MALA, nous utilisons la même formulation que PSSMLT qui se base sur la luminance. Il est également possible d'utiliser les pixels voisins découverts par la fonction de décalage afin d'obtenir une fonction cible version pondérée par la luminance du voisinage (Fig. 3.2).

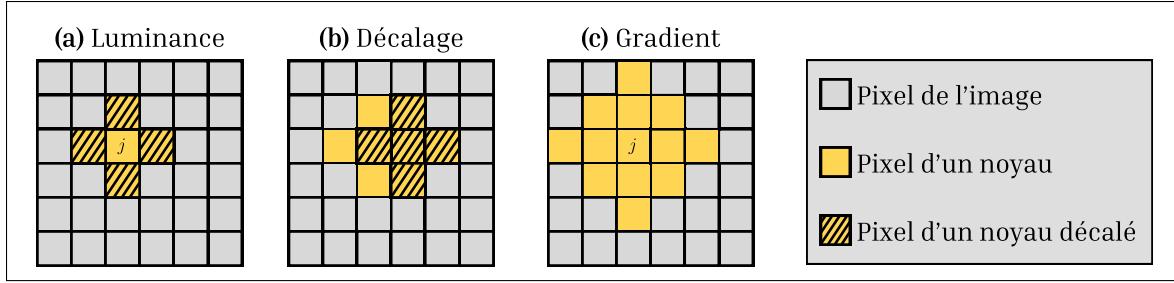


Figure 3.2 La fonction de décalage basée sur la luminance (a) calcule les différences finies entre la fonction cible du pixel  $j$  et de son voisinage. Ici, c'est la luminance des chemins qui est utilisée. Pour le domaine des gradients, la fonction cible ne prend pas seulement la luminance de  $j$ . Nous utilisons également la valeur des gradients de (a). Cela donne un noyau de base composé de 5 pixels (b) devant être décalé pour pouvoir en calculer la fonction cible. Ainsi, nous obtenons le noyau complet (c) lorsque nous décalons le noyau de base dans les 4 directions.

Lorsque nous passons vers le domaine des gradients, la fonction cible ne correspond plus uniquement à la luminance des chemins, mais à une adaptation du calcul de l'équation 2.5 pour chaque décalage. Ainsi, c'est un noyau de pixels qui est décalé (Fig. 3.2). À la différence de GDMLT, nous ne choisissons pas de direction aléatoire dans  $\Omega'$ . Nous utilisons plutôt l'ensemble des directions lors du calcul des gradients, ce qui équivaut à simplement avoir l'espace des états  $\Omega$ , et dans notre cas  $\mathcal{P} \rightarrow \Omega$  étant donné que nous utilisons des mutations sur l'espace primaire. La nouvelle fonction cible  $\pi^*$  d'un chemin de lumière est maintenant la somme des fonctions cibles des décalages  $\pi_{ij}(\bar{z})$  et de la fonction cible du chemin de base  $\pi(\bar{x})$  pondérée par un facteur  $\alpha$

$$\pi^*(\bar{x}) = \sum (\pi_{ij}(\bar{z})) + \alpha \cdot 0.25 \cdot \pi(\bar{x}). \quad (3.1)$$

Un pseudocode du calcul de la fonction cible basée sur les gradients est présenté dans l'algorithme 3.1. Pour commencer, nous traçons le chemin de base  $\bar{x}$  et les pixels voisins comme dans GDMLT (ligne 2). Après, pour chaque décalage, nous traçons le chemin central  $\bar{z}$  du noyau décalé (ligne 4). Les contributions  $\bar{z}^*$  des pixels voisins à  $\bar{z}$  sont évaluées (ligne 6) pour ensuite être utilisées dans le calcul de la fonction cible (ligne 8). Finalement, nous pouvons estimer le gradient de la fonction cible des noyaux en ayant recours aux différences finies (lignes 10 et 11).

### Algorithme 3.1 Gradient de la fonction cible avec GDMALA

```

1 Algorithme : Gradient de la fonction cible avec GDMALA
Input : Pixel  $j = \{x, y\}$ , Pondération du chemin de base  $\alpha$ 

2 Tracer le chemin de base  $\bar{x} \leftarrow trace(x, y)$ ;
3 for all shift  $\delta_x, \delta_y \in \{(1, 0), (-1, 0), (0, 1), (0, -1)\}$  do
4   Tracer le chemin central décalé  $\bar{z} \leftarrow trace(x + \delta_x, y + \delta_y)$ ;
5   for all offset  $i, j \in \{(1, 0), (-1, 0), (0, 1), (0, -1)\}$  do
6     Tracer le décalage dans le noyau  $\bar{z}^* \leftarrow trace(x + \delta_x + i, y + \delta_y + j)$ ;
7   end for
8   Calculer la fonction cible du noyau  $\pi(\bar{z})_{\delta_x, \delta_y} = \sum \left( F^*(\bar{z}_{ij}^*) \right) + \alpha \cdot 0.25 \cdot F^*(\bar{z})$ ;
9 end for
10 Calculer le gradient dans l'axe horizontal  $dx \leftarrow 0.5 \cdot (\pi(\bar{z})_{1,0} - \pi(\bar{z})_{-1,0})$ ;
11 Calculer le gradient dans l'axe vertical  $dy \leftarrow 0.5 \cdot (\pi(\bar{z})_{0,1} - \pi(\bar{z})_{0,-1})$ ;
12 return  $(dx, dy)$ 
```

## 3.2 Mutations

Les nombres aléatoires utilisés pour générer des chemins de lumière sont mutés dans un « sampler » avec une stratégie de mutation interchangeable. Nous appliquons cette opération simultanément sur les deux premiers nombres aléatoires qui génèrent une position  $x$  et  $y$  dans le plan image. Le reste des nombres aléatoires de  $X$  sont mis à jour par la mutation « fallback » de Kelemen *et al.* (2002) décrite dans la section 2.1.1 de façon indépendante. Puisque nous avons recours au mécanisme de « lazy evaluation », nous mettons à jour les nombres aléatoires un à la fois en utilisant un index de mutation  $i$ . Conséquemment, nos algorithmes dénotent l’usage d’une valeur dans le vecteur  $X$  pour le  $i$ ème état de la chaîne par  $X(t)[i]$ .

### 3.2.1 MALA

Ce mutateur propose de faire une perturbation uniforme, comme dans PSSMLT, mais avec la particularité que la fonction gaussienne soit centrée sur un décalage  $\lceil u_0$  et  $\lceil u_1$  dans la direction du gradient de la contribution lumineuse. La taille du décalage dans cette direction est pondérée par un paramètre  $\epsilon$ . Une plus petite valeur va diminuer le déplacement jusqu'à l'obtention d'une

mutation symétrique. Une plus grande valeur va diminuer le taux d'acceptation de la mutation, et donc la valeur optimale de ce paramètre dépend largement de la configuration de la scène. Finalement, un mouvement brownien est échantillonné à partir d'une distribution normale 2D de taille  $s \ N(0, s)$ (Fig. 2.2), de façon à représenter un bruit aléatoire.

Nous avons vu avec l'équation 2.11 la densité de la fonction de transition pour un état  $x$  vers l'état  $x'$ , correspondant au terme  $\mathcal{T}$  dans l'équation 1.13. L'une des particularités de la mutation symétrique de PSSMLT est que ce terme s'annule, donnant lieu à une simplification du calcul d'acceptation qui devient simplement le ratio des fonctions cibles  $\pi(x')/\pi(x)$ . Tout comme la mutation adaptative de Szirmay-Kalos & Szécsi (2017), la mutation de MALA n'est pas symétrique. Pour MALA, cela vient de son orientation dans le sens des gradients. Ainsi, la densité de probabilité de  $x$  vers  $x'$ , dénoté  $\mathcal{T}(x \rightarrow x')$ , n'est pas équivalente à la densité de probabilité de  $x'$  vers  $x$ , dénoté  $\mathcal{T}(x' \rightarrow x)$ . Il est donc nécessaire de calculer correctement le terme  $\mathcal{T}$  comme décrit dans les équations 1.13 et 2.11. Un pseudocode de la mutation est

### Algorithme 3.2 Mutation MALA

**1 Algorithme :** Mutation MALA

**Input :** Gradients  $g(t) = [dx, dy]$ , Step  $\epsilon$ , Mutation index  $i$ , Samples  $X(t)$

```

2 begin
3   if  $i >= \text{length}(g(t))$  then
4     return  $\text{fallback}(X(t)[i])$  /* mutation de Kelemen ou autre */
5   else
6     Calculer un mouvement brownien  $w \propto N(0, I)$ ;
7     return  $X(t)[i] + 0.5\epsilon \cdot g(t)[i] + \sqrt{\epsilon} \cdot w$ ;
8   end if
9 end
```

présenté dans l'algorithme 3.2. Le vecteur de nombres aléatoires  $X$  contient plus de valeurs que notre vecteur de gradients  $g(t)$ . Pour cette raison, nous utilisons une mutation « fallback » (ligne 4) lorsque nous appliquons une mutation sur un autre nombre aléatoire que ceux utilisés pour générer la position sur le plan image en  $x$  et  $y$ . Un mouvement brownien doit être échantillonné par rapport à une distribution normale multivariée de taille 2 (ligne 6). En pratique, il s'agit

de deux nombres aléatoires échantillonnés en même temps en utilisant la transformation de Box-Muller (Scott, 2011). Finalement, nous retournons une mise à jour de la valeur  $X(t)[i]$  via la dynamique de Langevin (ligne 7).

### 3.2.2 MALA avec adaptation

Ce mutateur propose d'améliorer la mutation isotrope de MALA en utilisant une descente de gradient basée sur l'algorithme Adam (Kingma & Ba, 2014). L'idée derrière Adam est de faire une optimisation stochastique de notre mutation en utilisant des gradients, et plus particulièrement dans notre cas, à partir de nos fonctions de décalages. Adam utilise normalement une matrice hessienne contenant les dérivées seconde pour faire l'optimisation. En pratique, nous prenons uniquement la diagonale de cette matrice, ce qui est substantiellement moins coûteux que de calculer le hessien (Luan *et al.*, 2020). Nous avons l'approximation du gradient de la fonction cible, par rapport aux deux premiers nombres aléatoires de  $X$  qui servent à générer une position  $(x, y)$  sur le plan image. Notre implémentation est composé de quatre termes importants :

1.  $G(t)$ , une matrice diagonale d'accumulation.
2.  $H(t)$ , une approximation de la matrice hessienne.
3.  $M(t)$ , une matrice diagonale de préconditionnement.
4.  $m(t)$ , un vecteur de momentum permettant d'accélérer la convergence.

Soit  $g(t)$ , le vecteur  $[dx, dy]$  de nos gradients pour les deux premiers nombres aléatoires,  $\beta$  et  $\alpha$  des poids qui décroient de façon exponentielle, et  $c_1, c_2$  les valeurs positives de l'adaptation diminuant au cours de la descente de gradient. Nous avons les formules pour le calcul de ces termes :

$$G(t) = \sum_{\tau=1}^t \frac{1}{\beta^{\tau-1}} (1 - \beta) g(\tau) \cdot g(\tau)^T, \quad (3.2)$$

$$H(t) = \delta I + \frac{1}{t^{c_1}} \cdot \sqrt{G(t)}, \quad (3.3)$$

$$M(t) = H(t)^{-1}, \quad (3.4)$$

$$m(t) = g(t) + \frac{1}{t^{c_2}} \sum_{\tau=1}^t \alpha^{\tau-1} (1-\alpha) g(\tau). \quad (3.5)$$

Un pseudocode de la mutation est présenté dans l'algorithme 3.3. Comme pour la mutation MALA, nous utilisons un « fallback » (ligne 8) lorsque nous appliquons une mutation sur un autre nombre aléatoire que ceux utilisés pour générer la position sur le plan image en  $x$  et  $y$ . Conséquemment, les formules pour les matrices d'accumulation, de préconditonnement et du moment sont légèrement adaptées par rapport à celles présentées par Luan *et al.* (2020) (ligne 10 à 13). L'index  $i$  dénote le gradient pour lequel ces valeurs sont mises à jour, où 0 correspond à  $dx$  et 1 correspond à  $dy$ . Un mouvement brownien doit être échantillonné comme dans la mutation MALA (ligne 14). Enfin, la valeur dans le vecteur  $X$  est mise à jour en appliquant la dynamique de Langevin (ligne 15).

### Algorithme 3.3 Mutation MALA avec adaptation

```

1 Algorithme : Mutation MALA avec adaptation
2   Input : Gradients  $g(t) = [dx, dy]$ , Decay  $\alpha, \beta$ , Diminishing adaptation  $c_1, c_2$ , Step  $\epsilon$ ,
      Constante  $\delta$ , Mutation index  $i$ , Samples  $X(t)$ 
3   begin
4     Initialiser les vecteurs  $G(0) \leftarrow [0, 0]$ ,  $m(0) \leftarrow [0, 0]$ ,  $d(0) \leftarrow [0, 0]$ ;
5     Initialiser un vecteur de diagonale identité  $I \leftarrow [1, 1]$ ;
6   end
7   begin
8     if  $i >= \text{length}(g(t))$  then
9       return fallback( $X(t)[i]$ ) /* mutation de Kelemen ou autre */
10    Calcule le vecteur d'acc.  $G(t)[i] \leftarrow \beta G(t-1)[i] + (1-\beta)g^2(t)[i]$ ;
11    Calcule le vecteur de pc.  $M(t)[i] \leftarrow I / ((\text{delta} * I[i]) + t^{-c_1} \cdot \sqrt{G(t)[i]})$ ;
12     $d(t)[i] \leftarrow \alpha d(t-1)[i] + (1-\alpha)g(t)[i]$ ;
13    Calcule le momentum  $m(t)[i] \leftarrow t^{-c_2}d(t)[i] + g(t)[i]$ ;
14    Calcule un mouvement brownien  $w \sim N(0, I)$ ;
15    return  $X(t)[i] + 0.5\epsilon M(t)[i] \cdot m(t)[i] + \sqrt{\epsilon} \cdot \sqrt{M(t)[i]} \cdot w$ ;
16   end if
17 end
```

### 3.3 Reconstruction itérative

Après le calcul des gradients et la mise en place d'une stratégie de mutation, il reste à reconstruire l'image par la résolution d'un problème de minimisation, tel que décrit dans la section 1.4.2. Hua *et al.* (2019) présente un algorithme de reconstruction utilisant un solveur itératif basé sur des variables de contrôle. L'idée est d'appliquer successivement une convolution sur chaque pixel de  $I_G$ , où leur valeur sera itérativement composée d'un mélange de leur contribution de base et des contributions de leurs pixels voisins. Ainsi, lors de chaque itération et pour chaque pixel  $I_{ij}$ , son voisinage  $\{I_g, I_d, I_h, I_b\}$  est utilisé pour calculer les gradients  $\{\Delta_g, \Delta_d, \Delta_h, \Delta_b\}$  dans leurs axes respectifs

$$\begin{aligned} \text{gauche} &\rightarrow (i-1, j) \xrightarrow{I_g} \Delta_g = (I_g - I_{ij}) \\ \text{droite} &\rightarrow (i+1, j) \xrightarrow{I_d} \Delta_d = (I_{ij} - I_d) \\ \text{haut} &\rightarrow (i, j-1) \xrightarrow{I_h} \Delta_h = (I_h - I_{ij}) \\ \text{bas} &\rightarrow (i, j+1) \xrightarrow{I_b} \Delta_b = (I_{ij} - I_b) \end{aligned} \quad (3.6)$$

En moyennant ces valeurs, on obtient la contribution lumineuse du pixel après l'itération  $n$ . Un facteur de pondération  $w$  est appliqué en fonction de la position du pixel  $I_{ij}$  dans l'image, où  $w$  est le nombre total de pixels pris en compte dans le calcul. Sur les bordures, cette valeur sera entre  $w = 3$  et  $w = 4$ , tandis que pour le reste de l'image, cette valeur sera de  $w = 5$ .

$$I_{ij}^{n+1} = \frac{1}{w} \cdot I_{ij}^n + (I_d^n - \Delta_d) + (I_g^n + \Delta_g) + (I_h^n + \Delta_h) + (I_b^n - \Delta_b). \quad (3.7)$$

Comme avec la reconstruction de poisson L2, cette technique présente des artefacts importants autour des sources de lumière, étant donné la forte variation du gradient et un suréchantillonnage de ces régions. Pour retirer ces artefacts, nous séparons les contributions lumineuses directes (Fig. 3.3), c'est-à-dire tous les rayons qui touchent directement une source de lumière à partir de la caméra. Ces contributions sont ajoutées une fois les itérations de reconstruction terminées. Pour nos tests, 50 itérations sont utilisées pour l'ensemble des scènes.

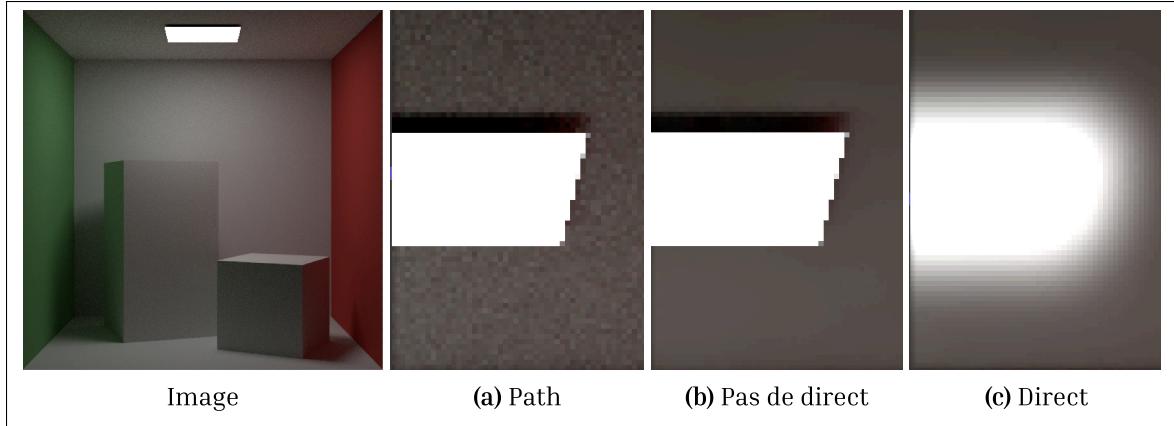


Figure 3.3 La séparation de l'éclairage direct (b) donne de meilleurs résultats aux alentours des sources lumineuses par rapport à la reconstruction les utilisant dans le calcul itératif (c).

## CHAPITRE 4

### PRÉSENTATION DES RÉSULTATS

Nos algorithmes sont implémentés dans notre propre moteur de rendu, développé en collaboration avec Adrien Gruson. Le code source en Swift de ces implémentations est disponible publiquement dans le moteur de rendu SwiftTracer. Nous comparons ces nouveaux intégrateurs, MALA et GDMALA, par rapport à d'autres méthodes non biaisées : tracé de rayon unidirectionnel (« path »), « primary sample space » MLT (PSSMLT) et « gradient domain » MLT (GDMLT). Tous les algorithmes ont été exécutés sur la même plateforme de test, un MacBook M4 Pro avec 48 GB de mémoire.

Nous effectuons nos tests sur un ensemble de quatre scènes, tirées de Bitterli (2024). **Door ajar** présente un scénario d'illumination complexe : une source de lumière à haute intensité entre dans la pièce par une porte entrouverte. **Bidir** est un exemple difficile pour le rendu unidirectionnel : deux sources de lumière et un oeuf en verre donnent lieu à des interactions lumineuses intéressantes. **Cornelbox** est une scène plus simple qui permet de vérifier le surcoût associé à une méthode de rendu complexe. Finalement, **classroom** contient des régions anisotropes (barreaux des chaises) qui permettent de valider l'utilité des mutations asymétriques.

Nous générerons la scène de référence (« ground truth ») avec un algorithme de tracé de rayon unidirectionnel sur le moteur de rendu Mitsuba pour door ajar et bidir. La référence des autres scènes a été générée en utilisant l'intégrateur path de notre moteur de rendu. Dans les deux cas, nous utilisons un nombre d'échantillons très élevé donnant une image dont le bruit est amoindri. Chaque algorithme comporte un certain nombre de paramètres. En outre, plusieurs mutations sont disponibles pour un même algorithme. La meilleure combinaison de paramètres-mutation a été choisie pour chacune des scènes. La taille maximale des chemins de lumière est de 16 pour toutes les scènes, à l'exception de **bidir** qui ne bénéficie pas particulièrement d'une telle longueur, étant donné la configuration particulière des chemins de lumière passant par l'oeuf en verre. La taille des chaînes  $N_{MC}$  a été choisie de façon optimale pour chacune des scènes, et le nombre d'échantillons pour l'intialisation de la constante de normalisation est de 100000.

$N_{MC}$  contrôle en même temps le nombre de chaînes total qui sera nécessaire. La fonction de décalage RSR a été utilisée sur les scènes classroom, bidir et door ajar. Notre implémentation de la reconnexion de chemin présente des problèmes importants sur les surfaces métalliques et mérite que le « manifold exploration » (Jakob & Marschner, 2012) ou la copie de « half vector » (Kettunen *et al.*, 2015) soit utilisé pour ce genre de scène. Nous appliquons une transformation sur la luminance finale de chaque pixel, de telle sorte que des intensités négatives sont remplacés par une intensité de zéro. Lorsque des valeurs trop fortes sont rencontrées (p.ex. l'importance d'un pixel  $I_{ij} > 100 \cdot \sum(I_{neighbours})$ ), la valeur est mise à l'échelle pour mitiger l'effet de « firefly ». Des méthodes plus complexes utilisant plusieurs tampons (Zirr, Hanika & Dachsbacher, 2018) amélioreraient la robustesse de ce filtrage.

La comparaison qualitative et quantitative est démontrée par le rendu final d'une scène avec un budget de temps fixe en minutes (Tab. 4.1) et avec un budget d'échantillons fixes par pixels (Tab. 4.2). Dans les deux cas, la racine de l'erreur quadratique moyenne (RMSE) et le pourcentage d'erreur absolue moyenne (MAPE) sont utilisés pour l'évaluation quantitative

$$RMSE = \sqrt{\frac{\sum_{i=0}^N (P_i - O_i)^2}{N}}, \quad (4.1)$$

$$MAPE = 100 \frac{1}{N} \sum_{i=0}^N \left| \frac{P_i - O_i}{P_i} \right|, \quad (4.2)$$

où N est le nombre total de pixels dans l'image,  $P_i$  est la valeur d'un pixel de notre image de référence, et  $O_i$  est la valeur observée d'un pixel dans l'image résultante.

Un algorithme non biaisé devrait éventuellement converger vers l'image de référence étant donné que le temps de rendu est suffisamment élevé. Par contre, les méthodes MCMC ont la particularité d'avoir des artefacts notables lorsque la chaîne reste prise dans un pixel ayant une contribution trop forte. Cela peut survenir si ce chemin à forte contribution est particulièrement difficile à trouver (p.ex. très faible probabilité dans l'espace des états). Ce phénomène se traduit par de fortes variations (« firefly »), malgré une convergence dans le reste de l'image, et est particulièrement problématique pour la génération de nos gradients.

Tableau 4.1 Erreur quadratique moyenne des intégrateurs avec un budget en temps de rendu

<b>Scène</b>	<b>Résolution</b>	<b>Budget</b> (min)	<b>Erreur quadratique moyenne (RMSE)</b>				
			<b>path</b>	<b>pssmlt</b>	<b>mala</b>	<b>gdmlt</b>	<b>gdmala</b>
Cornelbox	512x512	10	<b>4.00e-4</b>	1.58e-3	2.14e-3	<b>2.39e-4</b>	7.51e-4
Classroom	1280x720	10	<b>4.58e-3</b>	8.86e-3	3.56e-2	<b>4.61e-3</b>	4.65e-3
Bidir	1024x1024	20	3.01	3.43e-1	<b>2.28e-1</b>	<b>1.20e-1</b>	1.25e-1
Door ajar	1280x720	30	3.49e-1	<b>2.73e-2</b>	4.41e-1	<b>1.12e-1</b>	1.25e-1

L'utilisation d'un budget de temps de rendu permet d'obtenir des courbes de convergences (Annexe I). Cependant, il est parfois plus équitable de comparer les algorithmes avec un budget en nombre d'échantillons (Tab. 4.2). Cela permet de vérifier qu'une technique bien optimisée peut engendrer une réduction de la variance. Nous remarquons par exemple que GDMALA devient nettement plus intéressant avec ce type de comparaison. Nous en déduisons qu'en ayant une meilleure fonction de décalage dont le temps d'exécution serait optimal, les résultats à temps égal seraient grandement améliorés.

Tableau 4.2 Erreur quadratique moyenne des intégrateurs avec un budget d'échantillons

<b>Scène</b>	<b>Résolution</b>	<b>Budget</b> (nspp)	<b>Erreur quadratique moyenne (RMSE)</b>				
			<b>path</b>	<b>pssmlt</b>	<b>mala</b>	<b>gdmlt</b>	<b>gdmala</b>
Cornelbox	512x512	50	2.60e-3	1.62e-2	<b>3.45e-3</b>	<b>2.98e-4</b>	3.04e-4
Classroom	1280x720	50	<b>6.52e-3</b>	1.86e-2	8.43e-1	<b>1.92e-3</b>	1.94e-3
Bidir	1024x1024	30	1.28e+1	4.46e-1	<b>1.07e-1</b>	<b>7.99e-2</b>	8.51e-2
Door ajar	1280x720	50	9.07e-1	5.47e-1	<b>3.36e-1</b>	<b>9.44e-2</b>	9.84e-2

Les résultats à temps égal de la scène cornelbox (Fig. 4.1) démontrent la rapidité de convergence dans un scénario relativement facile pour tous les algorithmes : l'absence de surfaces métalliques et spéculaires rend les opérations de décalage relativement faciles. On constate que tous les algorithmes tendent à converger éventuellement (Fig. I-1). GDMLT a le mieux performé à échantillons égaux (Fig. 4.2), suivi de très près par GDMALA (facteurs respectifs de 8.7x et 8.55x). Les algorithmes dans le domaine des gradients comportent moins d'erreurs de façon générale, même si la variance est plus facilement perceptible dans les régions contenant des

arêtes franches entre les objets. Cela vient confirmer l'importance d'avoir une bonne fonction de décalage ; les discontinuités créent souvent des problèmes de visibilité avec le rayon de base et donnent lieu à des décalages dans l'état « failed ». Ce problème est visible dans la section au-dessus de la source de lumière. Dans le cas de MALA, cela entraîne une baisse de la qualité des mutations, puisque l'information du gradient n'est pas présente. Il faut aussi noter que la scène de référence a été générée avec l'algorithme path ; cela vient avantagez cette méthode sur les autres pour le budget à temps égal, puisque l'algorithme est certain de converger exactement à l'image de référence. Nous remarquons aussi que les algorithmes de PSSMLT et MALA ne viennent pas particulièrement aider dans le rendu d'une scène ayant un faible niveau de complexité sur les interactions lumineuse. Pour commencer à voir une amélioration dans les résultats par l'utilisation des gradients sur la mutation, il faut utiliser des scènes plus complexes comme bidir et doorajar (Fig. 4.7 et 4.5).

Les résultats à temps égal de la scène classroom (Fig. 4.3) sont très similaires à ceux de cornelbox, sauf que l'erreur globale est plus élevée pour l'ensemble des intégrateurs. GDMALA et GDMLT ont le mieux performé à échantillons égal (Fig. 4.4). Dans cette scène, la seule source de lumière est une carte d'environnement qui laisse entrer la lumière par des fenêtres de part et d'autre de la pièce. Les algorithmes tendent tous à converger, bien que MALA subit parfois des pics lorsque la chaîne de Markov reste prise dans un espace local restreint (Fig. I-2). Nous ne remarquons pas d'amélioration entre path et GDMALA à temps égal, ce qui laisse penser que la mutation isotrope combinée à la fonction de décalage RSR ne permet pas d'améliorer la convergence sur cette scène.

La figure 4.5 présente les résultats à temps égal de door ajar, une scène typiquement difficile pour le rendu étant donné que la source de lumière est cachée et que son intensité est très élevée. On remarque d'ailleurs que les techniques n'utilisant pas de phase de reconstruction tendent à avoir un bruit uniforme partout dans l'image, ce qui n'est pas surprenant puisque notre reconstruction itérative vient étendre ce bruit à peu près également dans le voisinage d'un pixel. Tous les algorithmes tendent à converger, bien que ceux dans le domaine des gradients y arrivent beaucoup plus vite (Fig. I-4). Les résultats à échantillons égaux de GDMLT et GDMALA ont

une RMSE particulièrement faible par rapport aux autres (Fig. 4.6), ce qui pourrait s'expliquer par la simplicité de la géométrie de la scène au niveau des murs. On constate un facteur de 2.79x entre la RMSE du domaine des gradients et de path. Nous remarquons de façon générale une réduction du bruit par les chaînes de Markov, particulièrement au niveau des interactions diffuses dans le voisinage des pixels de la porte et des théières. Cependant, on ne constate pas de différence notable entre GDMALA et GDMLT. L'usage de la mutation isotrope sur cette scène pourrait expliquer l'absence d'amélioration, car le nombre élevé d'interactions spéculaires et métalliques limite grandement l'utilité du déplacement de la mutation uniforme vers le gradient. Avec des stratégies de décalages mieux adaptées aux surfaces partiellement métalliques (plancher de doorajar), la variance dans le reste de l'image devrait être réduite.

Avec la scène bidir (Fig. 4.7), nous remarquons une grande réduction de la variance pour tous les algorithmes MCMC par rapport à path. Plus particulièrement, la convergence de path ne semble pas très prometteuse (Fig. I-3). Notamment, l'oeuf en verre se trouvant directement sur le chemin d'une source de lumière vient causer énormément de « firefly » et de bruit dans son voisinage sur cet intégrateur. Les erreurs utilisées sont la RMSE, étant donné que la valeur de la MAPE est très importante dans tous nos algorithmes et rend la comparaison qualitative difficile. Nous avons un facteur de 24x entre la RMSE de path et celle de GDMALA à temps égal. Cette scène est beaucoup mieux adaptée au rendu bidirectionnel, car l'échantillonnage des sources de lumière cause beaucoup de NEE sans visibilité directe. Le facteur de la RMSE entre GDMALA et PSSMLT à temps égal (2.74x) est un peu plus raisonnable et prouve l'utilité des chaînes de Markov sur ce genre de scène. On constate aussi des améliorations dans les résultats de GDMLT et GDMALA par rapport aux autres à échantillons égaux (Fig. 4.8).

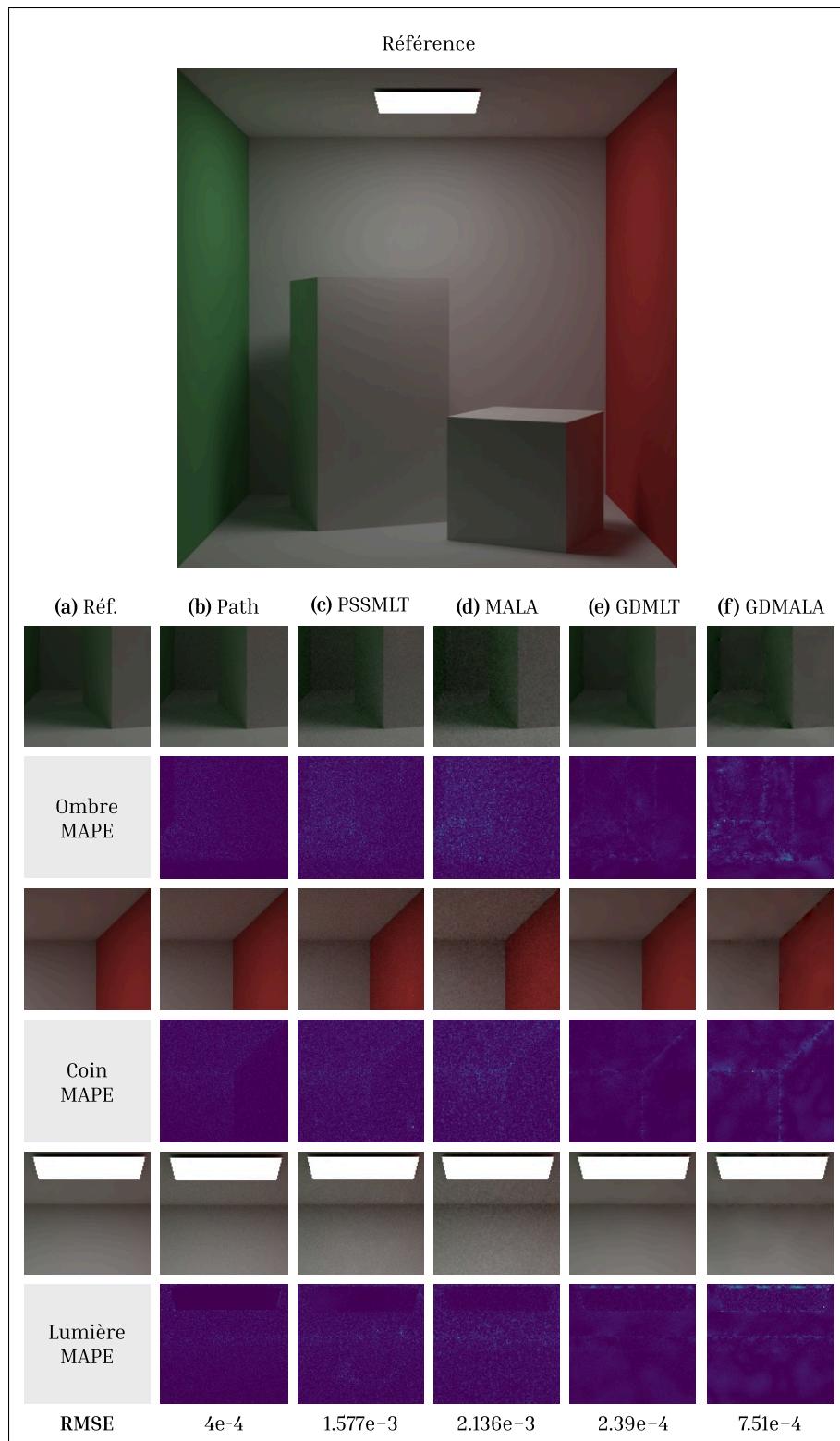


Figure 4.1 Comparaison à temps égal (10 mins.) du rendu de cornelbox avec nos algorithmes.

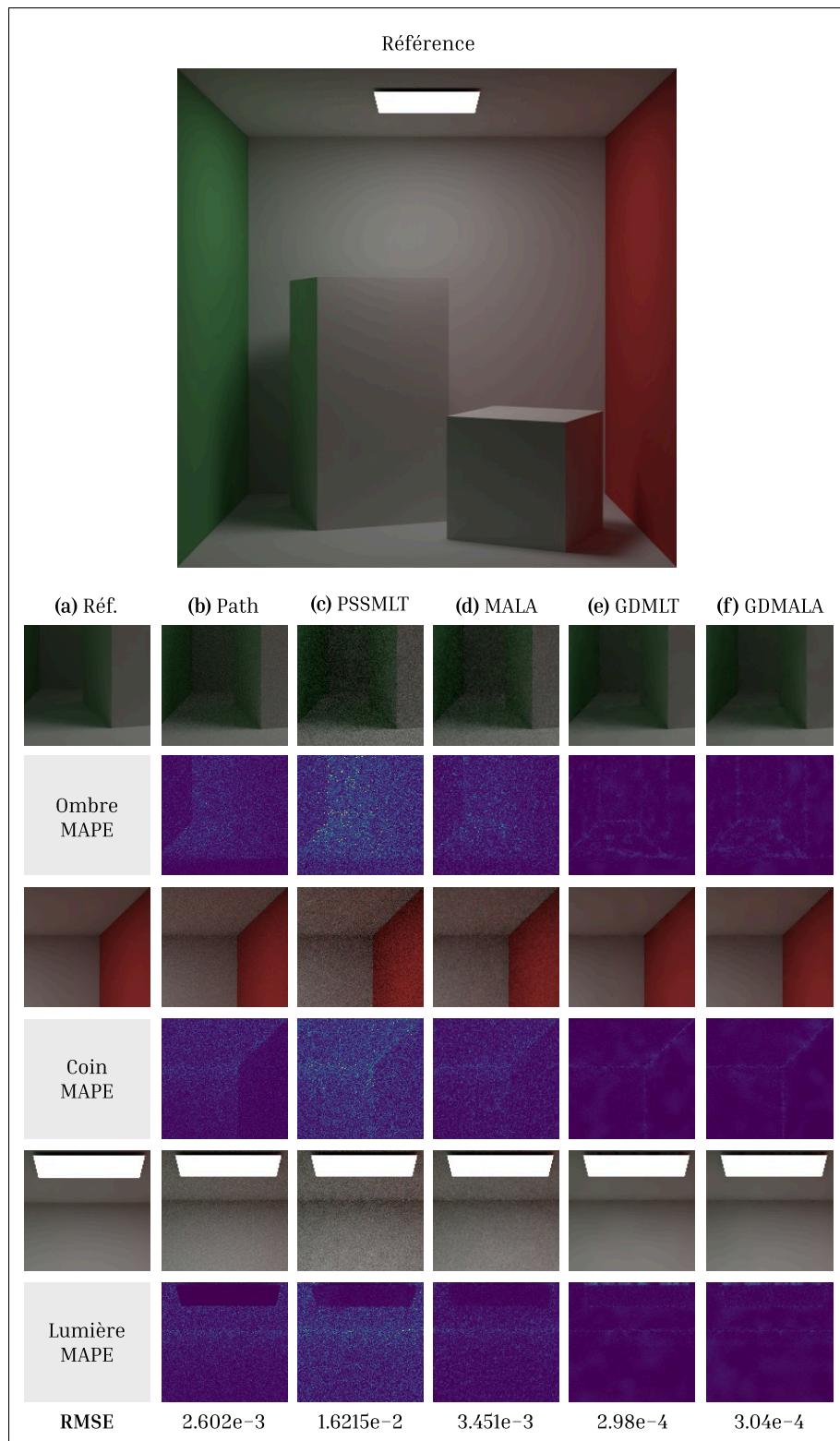


Figure 4.2 Comparaison à nombre d'échantillons égal (50) du rendu de cornelbox avec nos algorithmes.

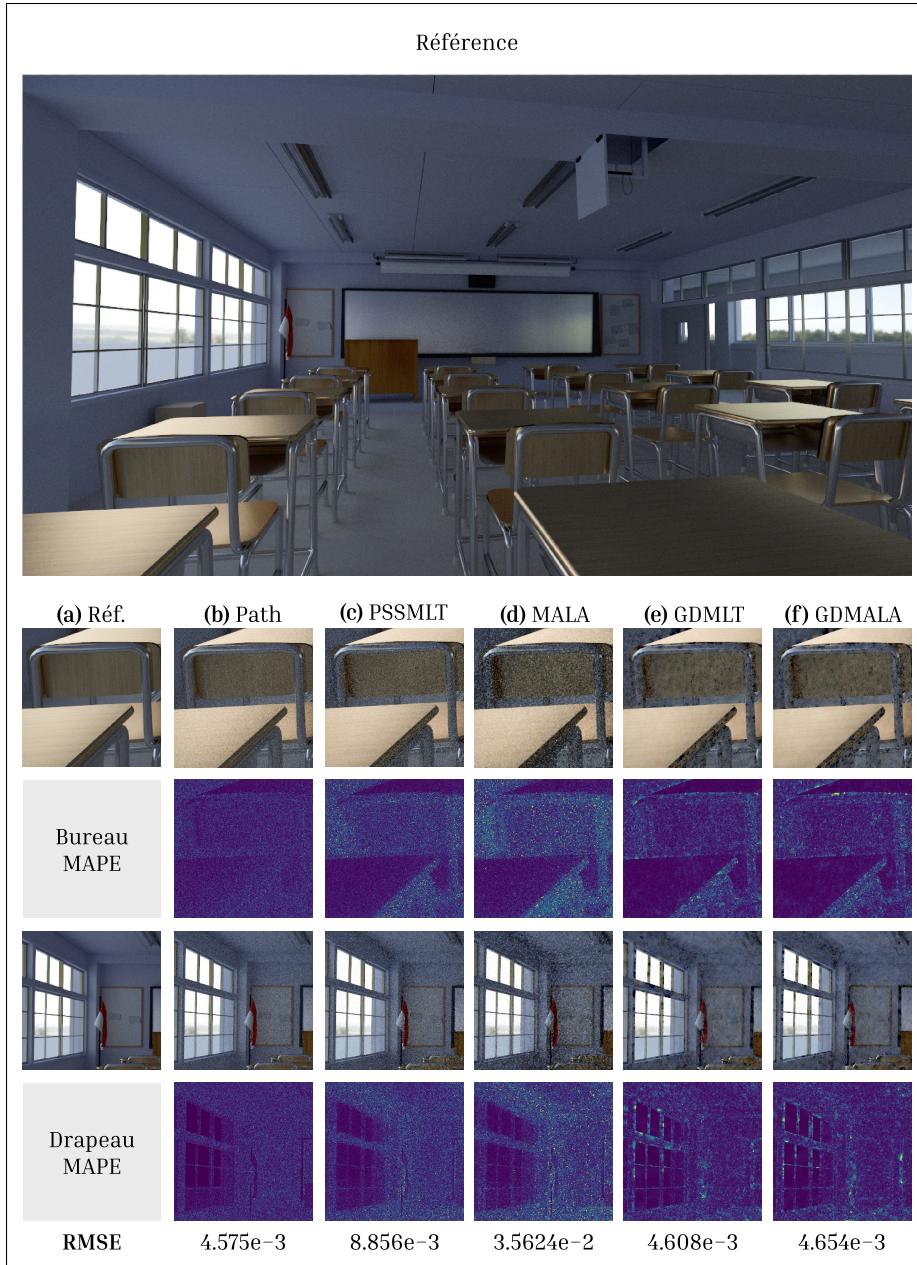


Figure 4.3 Comparaison à temps égal (10 mins.) du rendu de classroom avec nos algorithmes.

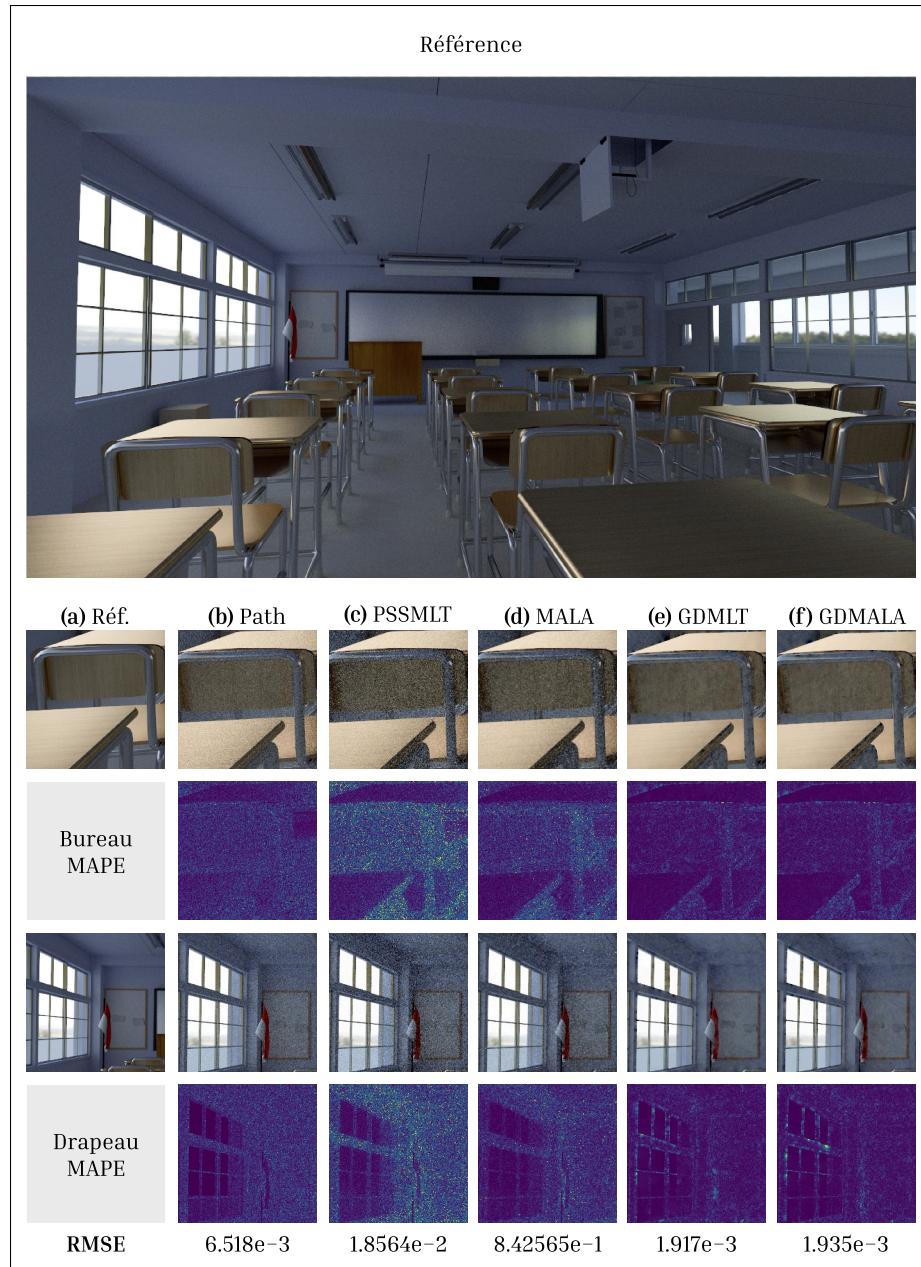


Figure 4.4 Comparaison à nombre d'échantillons égal (50) du rendu de classroom avec nos algorithmes.

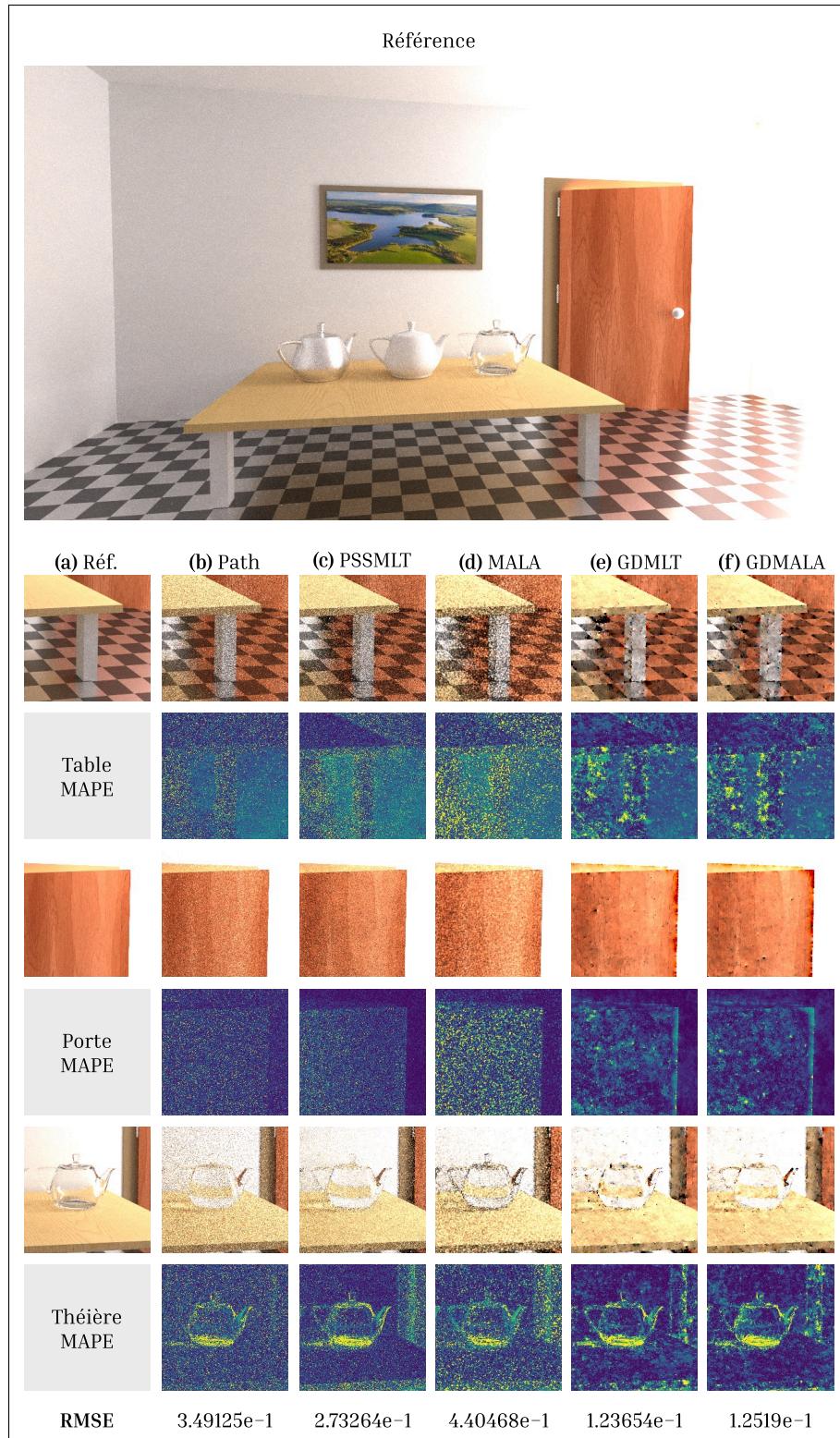


Figure 4.5 Comparaison à temps égal (30 mins.) du rendu de door ajar avec nos algorithmes.

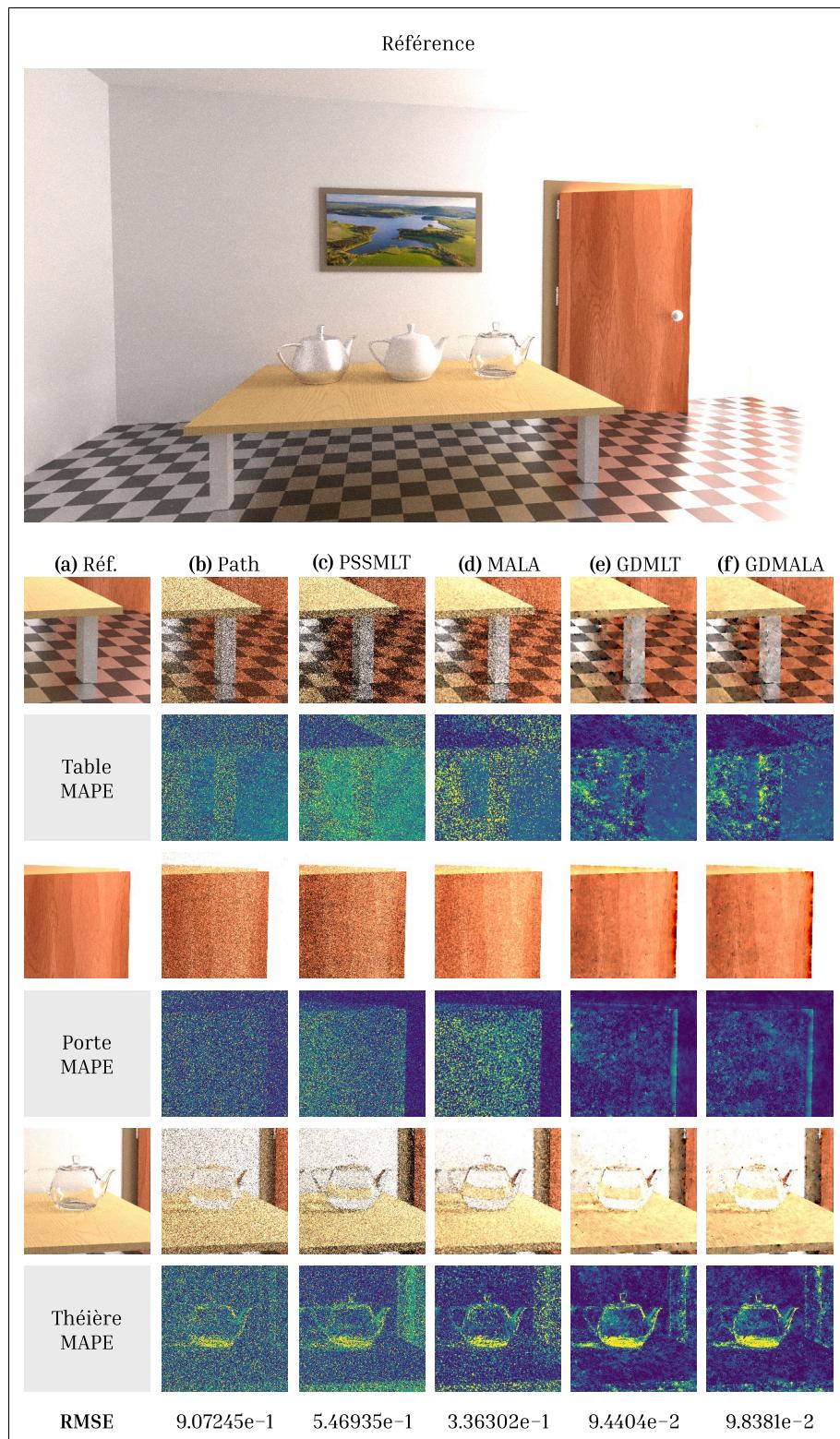


Figure 4.6 Comparaison à nombre d'échantillons égal (50) du rendu de door ajar avec nos algorithmes.

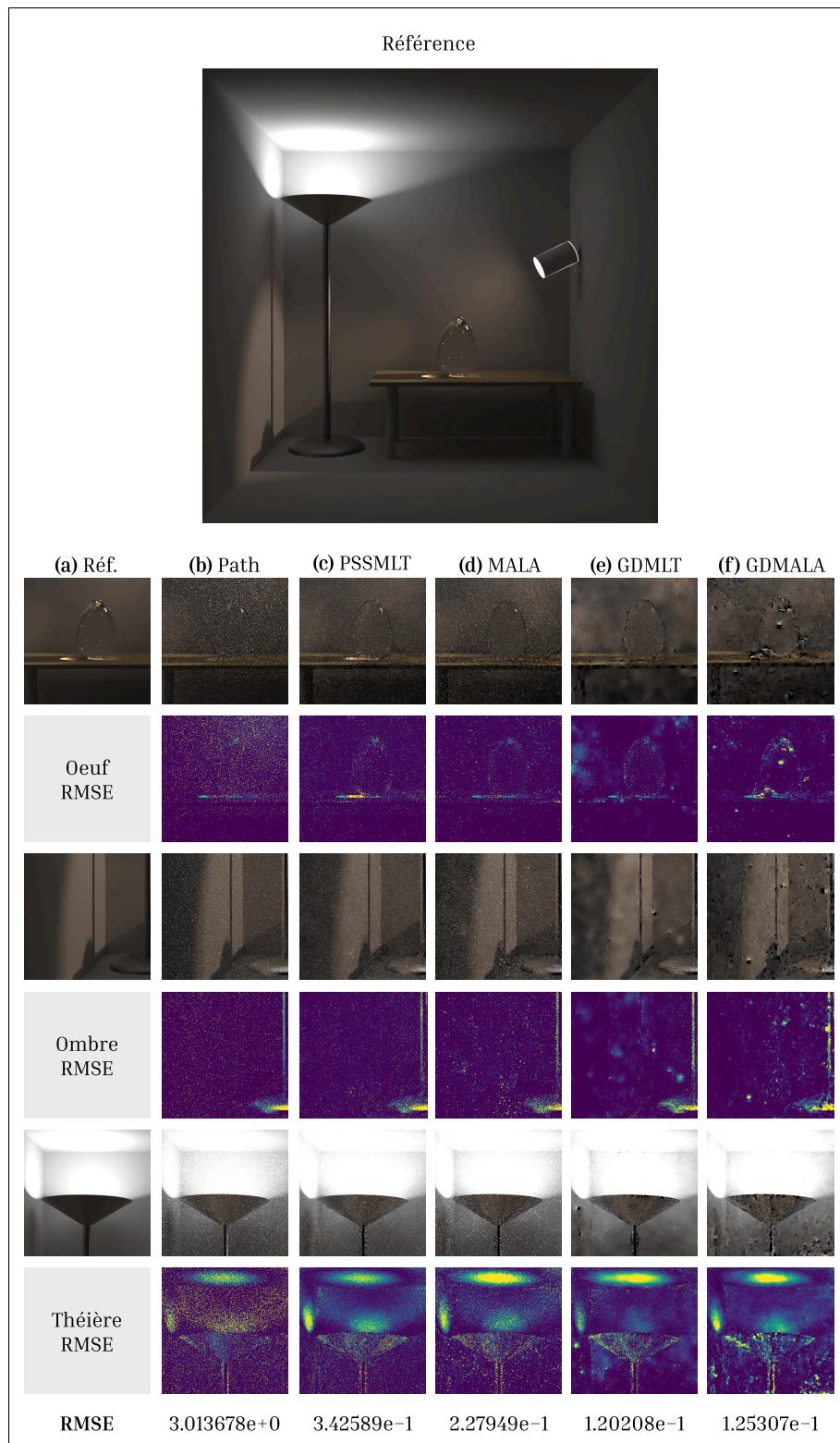


Figure 4.7 Comparaison à temps égal (20 mins.) du rendu de bidir avec nos algorithmes.

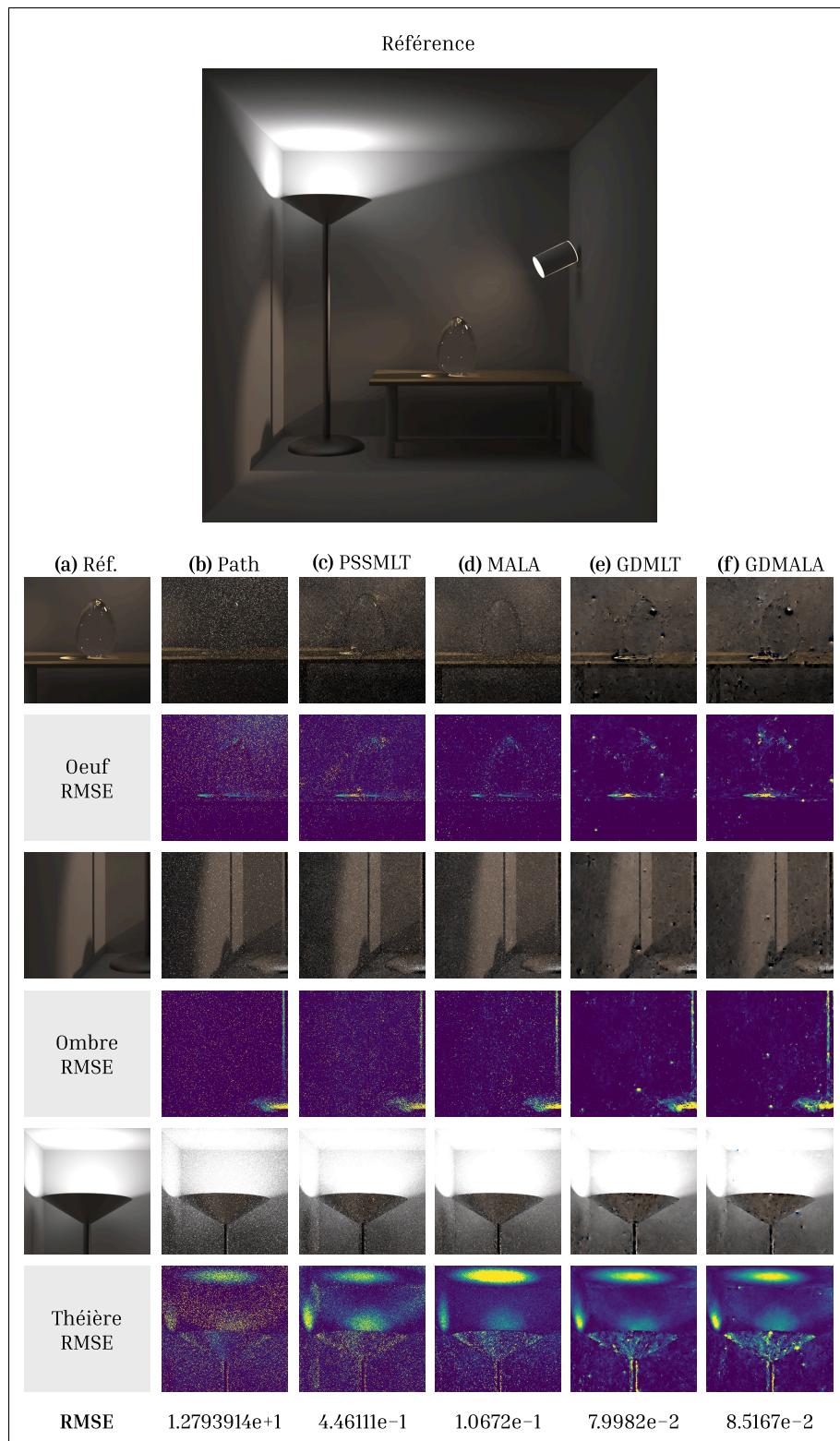


Figure 4.8 Comparaison à nombre d'échantillons égal (30) du rendu de bidir avec nos algorithmes.



## CONCLUSION ET RECOMMANDATIONS

Nous avons présenté GDMALA et MALA avec différence finie, deux algorithmes qui utilisent les différences finies de GDMLT pour diriger des mutations de type MALA sur l'espace primaire. Comme pour le domaine des gradients, nous avons une étape de reconstruction donnant lieu à la résolution d'un problème de minimisation. Nous avons fourni les résultats et les analyses de cette technique et les avons comparés à d'autres algorithmes tels que PSSMLT et GDMLT. Le principal avantage de notre approche est qu'elle permet de simuler la dynamique de Langevin sans avoir recours à un système d'autodifférentiation. Nous avons démontré que les différences finies sont un outil simple à mettre en place et offrent une approximation qui dépend largement sur la qualité de la fonction de décalage. En outre, les méthodes de RSR et reconnexions de chemin semblent insuffisantes lorsque le niveau de complexité d'une scène augmente. Des techniques plus complexes comme le « manifold exploration » (Jakob & Marschner, 2012) ou la copie du « half vector » (Kettunen *et al.*, 2015) pourraient améliorer nos résultats sur des scènes similaires à classroom, doorajar ou bidir.

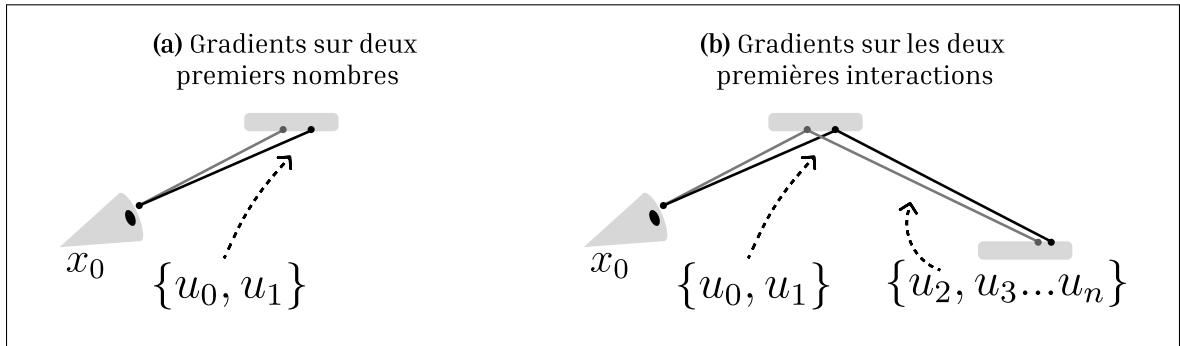


Figure 5.1 Notre technique utilise présentement le gradient des fonctions cibles à partir des décalages **(a)**. En allant d'un pas en avant, nous proposons d'évaluer l'augmentation de la précision des gradients en l'évaluant au point de la reconnexion.

Nos algorithmes comportent certaines limitations dans leur forme actuelle. Nous pensons donc que notre travail donne lieu à plusieurs pistes d'exploration, plus particulièrement sur l'estimation des gradients par le calcul des différences finies. Pour le moment, nous avons le gradient de  $\pi$  par

rapport aux deux premiers nombres aléatoires, lesquels sont utilisés pour générer une position  $x, y$  sur le plan image. En incluant dans ce calcul la dérivée par rapport aux prochains nombres aléatoires de  $u$  utilisés pour générer le premier rebond du chemin de lumière, nous pourrions potentiellement améliorer la précision du gradient (Fig. 5.1). Premièrement, cela serait facile pour la majeure partie d'une image composée principalement de surfaces diffuses ; un nombre élevé d'interactions avec des surfaces rencontrées lors du premier rebond satisferont la condition de connectivité décrite dans la section 1.4.1. Par ce fait, les gradients de la contribution lumineuse à ce rebond pourraient être calculés dans le voisinage du point de reconnexion. Nous pensons qu'avec cet ajout aux fonctions de décalage il serait possible, sans système d'autodifférentiation, d'obtenir une estimation numérique des gradients plus précise, ce qui se traduit typiquement par une meilleure phase de reconstruction et une réduction de la variance. Par contre, des matériaux plus complexes (p.ex. combinaison de composantes diffuses, spéculaires, diélectriques) ou avec des couches dont les propriétés varient peuvent rendre le travail d'une fonction de décalage difficile. En réalité, il n'existe pas de technique qui sera parfaite pour toutes les interactions et la décision de mettre fin à un décalage dépend largement d'un choix arbitraire basé sur des seuils de rugosité (Hua *et al.*, 2019; Lin *et al.*, 2022).

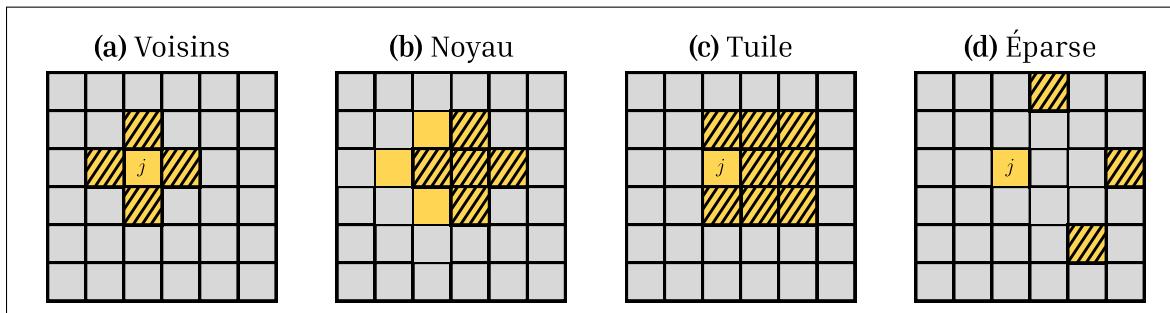


Figure 5.2 En utilisant des noyaux différents pour le décalage, nous pouvons approximer les gradients avec des coûts variables étant donné le nombre de rayons additionnel. La méthode en (a) est simple mais limité dans l'étendue de la valeur du gradient. Un noyau (b) permet d'approximer une fonction cible plus complexe dans le domaine des gradients. En décalant une tuile (c), on obtient une meilleure évaluation du voisinage dans son ensemble. Finalement, un décalage épars de pixels choisis en fonction de leur intensité (d) pourrait offrir des gains par rapport à une tuile.

Outre les gradients, différentes fonctions de décalage méritent d'être explorées (Fig. 5.2). Entre autres, en utilisant davantage de pixels (Tong & Hachisuka, 2024), il serait plus facile de capturer l'anisotropie des surfaces dans l'évaluation de la fonction cible. Cependant, l'usage de plusieurs pixels entraîne présentement une augmentation importante dans le nombre de rayons de lumière devant être tracés. Il devient donc apparent que l'optimisation de cette fonction, en relation avec le type d'interaction rencontrée, soit un point central dans la création d'algorithmes performants basés sur la dynamique de Langevin n'ayant pas recours à l'autodifférentiation.

Rousselle, Jarosz & Novák (2016) démontrent qu'il est possible d'améliorer les résultats de la phase de reconstruction des algorithmes dans le domaine des gradients avec une fonction itérative utilisant des variables de contrôles. L'idée est de calculer la variance de l'image grossière et des gradients et de prendre en compte cette valeur dans l'évaluation du poids  $w$  décrit par l'équation 3.7. Ces variables de contrôle viennent pondérer les gradients utilisés dans l'interpolation d'un pixel à chaque itération. Hua *et al.* (2019) notent une réduction importante du nombre d'artefacts dus à des gradients bruités (Fig. 5.3). Les travaux de Back, Hua, Hachisuka & Moon (2020) suggèrent que l'apprentissage machine pourrait également améliorer la reconstruction.

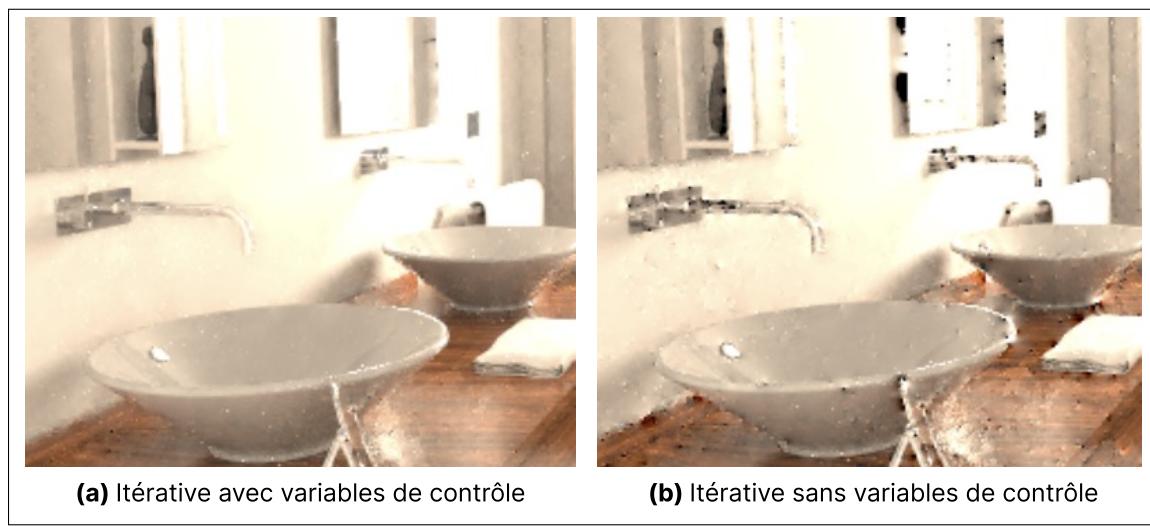


Figure 5.3 En utilisant des variables de contrôle (a), nous pouvons obtenir des gains substantiels au niveau de la qualité du résultat, qui contient une erreur quadratique moyenne plus faible qu'avec la méthode uniforme (a). Images tirées de Hua *et al.* (2019).



## ANNEXE I

### RÉSULTATS ADDITIONNELS

#### 1. Courbes de convergence

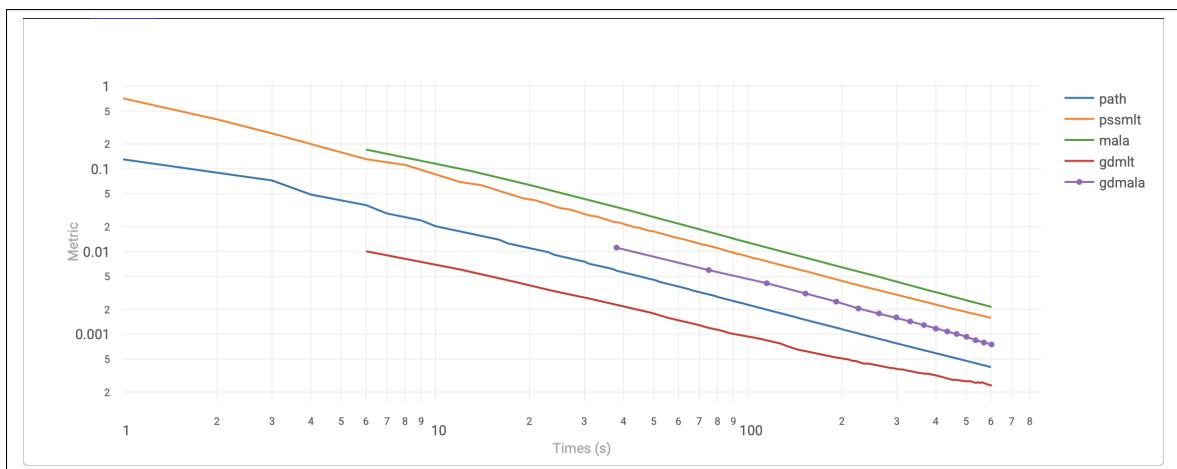


Figure-A I-1 Courbes de convergence pour la scène conrnelbox.

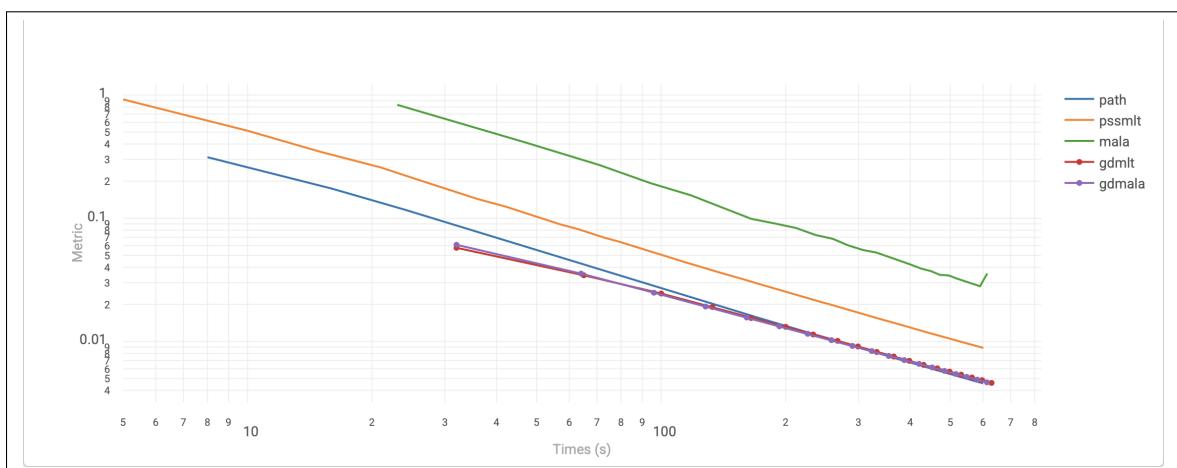


Figure-A I-2 Courbes de convergence pour la scène classroom.

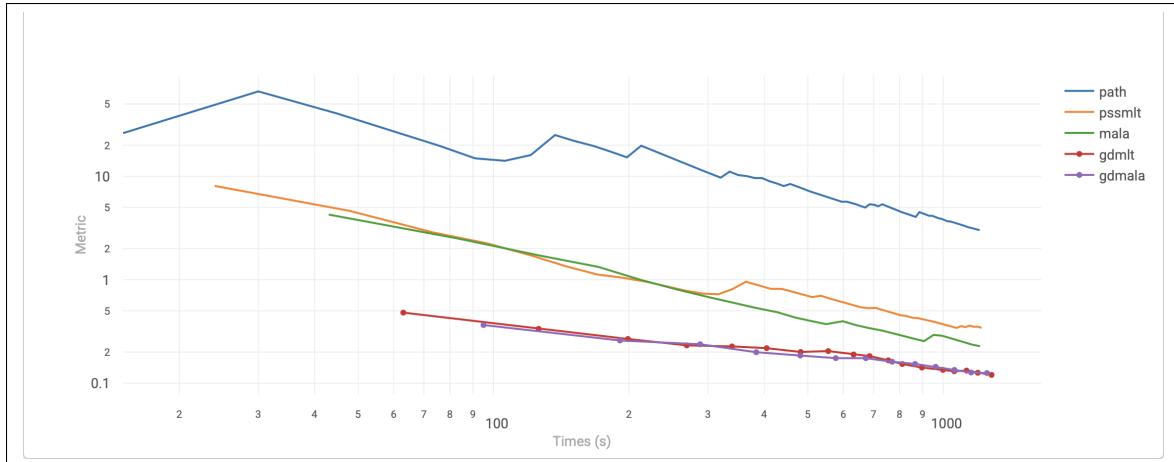


Figure-A I-3 Courbes de convergence pour la scène bidir.

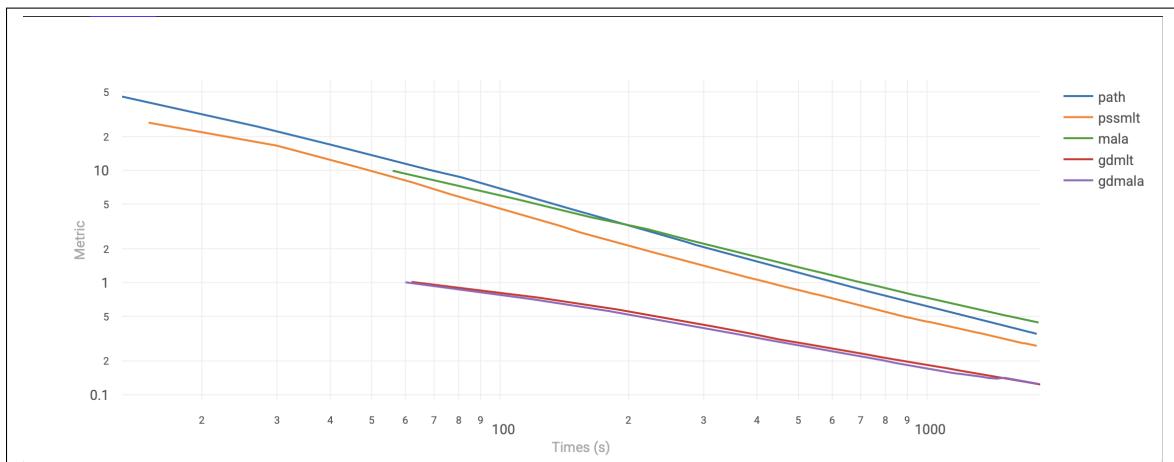


Figure-A I-4 Courbes de convergence pour la scène doorajar.

## BIBLIOGRAPHIE

- Back, J., Hua, B.-S., Hachisuka, T. & Moon, B. (2020). Deep combiner for independent and correlated pixel estimates. *ACM Transactions on Graphics*, 39(6), 1–12. doi : 10.1145/3414685.3417847.
- Bitterli. (2024). Rendering Resources | Benedikt Bitterli's Portfolio. Repéré le 2024-12-04 à <https://benedikt-bitterli.me/resources/>.
- Duane, S., Kennedy, A., Pendleton, B. J. & Roweth, D. (1987). Hybrid Monte Carlo. 195(2), 216–222. doi : 10.1016/0370-2693(87)91197-X.
- Ermak, D. L. & McCammon, J. A. (1978). Brownian dynamics with hydrodynamic interactions. 69(4), 1352–1360. doi : 10.1063/1.436761.
- Hastings, W. K. (1970). Monte Carlo sampling methods using Markov chains and their applications. 57(1), 97–109. doi : 10.1093/biomet/57.1.97.
- Hua, B., Gruson, A., Petitjean, V., Zwicker, M., Nowrouzezahrai, D., Eisemann, E. & Hachisuka, T. (2019). A Survey on Gradient-Domain Rendering. *Computer Graphics Forum*, 38(2), 455–472. doi : 10.1111/cgf.13652.
- Jakob, W. & Marschner, S. (2012). Manifold exploration : a Markov Chain Monte Carlo technique for rendering scenes with difficult specular transport. *ACM Transactions on Graphics*, 31(4), 1–13. doi : 10.1145/2185520.2185554.
- Jakob, W., Speierer, S., Roussel, N. & Vicini, D. (2022). DR.JIT : a just-in-time compiler for differentiable rendering. *ACM Transactions on Graphics*, 41(4), 1–19. doi : 10.1145/3528223.3530099.
- Kajiya, J. T. (1986). The rendering equation. *ACM SIGGRAPH Computer Graphics*, 20(4), 143–150. doi : 10.1145/15886.15902.
- Kelemen, C., Szirmay-Kalos, L., Antal, G. & Csonka, F. (2002). A Simple and Robust Mutation Strategy for the Metropolis Light Transport Algorithm. *Computer Graphics Forum*, 21(3), 531–540. doi : 10.1111/1467-8659.t01-1-00703.
- Kettunen, M., Manzi, M., Aittala, M., Lehtinen, J., Durand, F. & Zwicker, M. (2015). Gradient-domain path tracing. 34(4), 1–13. doi : 10.1145/2766997.
- Kingma, D. P. & Ba, J. [Version Number : 9]. (2014). Adam : A Method for Stochastic Optimization. arXiv. Repéré le 2025-04-12 à <https://arxiv.org/abs/1412.6980>.

- Lehtinen, J., Karras, T., Laine, S., Aittala, M., Durand, F. & Aila, T. (2013). Gradient-domain metropolis light transport. *ACM Transactions on Graphics*, 32(4), 1–12. doi : 10.1145/2461912.2461943.
- Lin, D., Kettunen, M., Bitterli, B., Pantaleoni, J., Yuksel, C. & Wyman, C. (2022). Generalized Resampled Importance Sampling : Foundations of ReSTIR. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2022)*, 41(4), 75 :1–75 :23. doi : 10.1145/3528223.3530158. (\*Joint First Authors).
- Luan, F., Zhao, S., Bala, K. & Gkioulekas, I. (2020). Langevin monte carlo rendering with gradient-based adaptation. 39(4), 140. doi : 10.1145/3386569.3392382.
- Maruyama, G. (1955). Continuous Markov processes and stochastic equations. 4(1), 48–90. doi : 10.1007/BF02846028.
- Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H. & Teller, E. (1953). Equation of State Calculations by Fast Computing Machines. 21(6), 1087–1092. doi : 10.1063/1.1699114.
- Otsu, H., Hanika, J., Hachisuka, T. & Dachsbaecher, C. (2018). Geometry-aware metropolis light transport. *ACM Transactions on Graphics*, 37(6), 1–11. doi : 10.1145/3272127.3275106.
- Pharr, M., Jakob, W. & Humphreys, G. (2023). *Physically based rendering : from theory to implementation* (éd. Fourth edition). The MIT Press.
- Phong, B. T. (1975). Illumination for computer generated pictures. 18(6), 311–317. doi : 10.1145/360825.360839.
- Roberts, G. O. & Tweedie, R. L. (1996). Exponential Convergence of Langevin Distributions and Their Discrete Approximations. 2(4), 341. doi : 10.2307/3318418.
- Rousselle, F., Jarosz, W. & Novák, J. (2016). Image-space Control Variates for Rendering. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)*, 35(6), 169 :1–169 :12. doi : 10/f9cphw.
- Schlick, C. (1994). An Inexpensive BRDF Model for Physically-based Rendering. 13(3), 233–246. doi : 10.1111/1467-8659.1330233.
- Scott, D. W. (2011). Box–Muller transformation. *WIREs Computational Statistics*, 3(2), 177–179. doi : 10.1002/wics.148.
- Sik, M. & Krivanek, J. (2020). Survey of Markov Chain Monte Carlo Methods in Light Transport Simulation. 26(4), 1821–1840. doi : 10.1109/TVCG.2018.2880455.

- Szirmay-Kalos, L. & Szécsi, L. (2017). Improved stratification for Metropolis light transport. 68, 11–20. doi : 10.1016/j.cag.2017.07.032.
- Taillet, R., Villain, L. & Febvre, P. (2023). Dictionnaire de physique : + de 7000 termes, nombreuses références historiques, 3700 références bibliographiques. Dans *Dictionnaire de physique : + de 7000 termes, nombreuses références historiques, 3700 références bibliographiques* (éd. 5e éd, pp. 531). De Boeck supérieur.
- Tong, X. & Hachisuka, T. (2024). Efficient Image-Space Shape Splatting for Monte Carlo Rendering. 43(6), 1–11. doi : 10.1145/3687943.
- Veach, E. & Guibas, L. J. (1995). Optimally combining sampling techniques for Monte Carlo rendering. *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, (SIGGRAPH '95), 419–428. doi : 10.1145/218380.218498.
- Veach, E. & Guibas, L. J. (1997). Metropolis light transport. *Proceedings of the 24th annual conference on Computer graphics and interactive techniques - SIGGRAPH '97*, pp. 65–76. doi : 10.1145/258734.258775.
- Zirr, T., Hanika, J. & Dachsbacher, C. (2018). Re-Weighting Firefly Samples for Improved Finite-Sample Monte Carlo Estimates. *Computer Graphics Forum*, 37(6), 410–421. doi : 10.1111/cgf.13335.