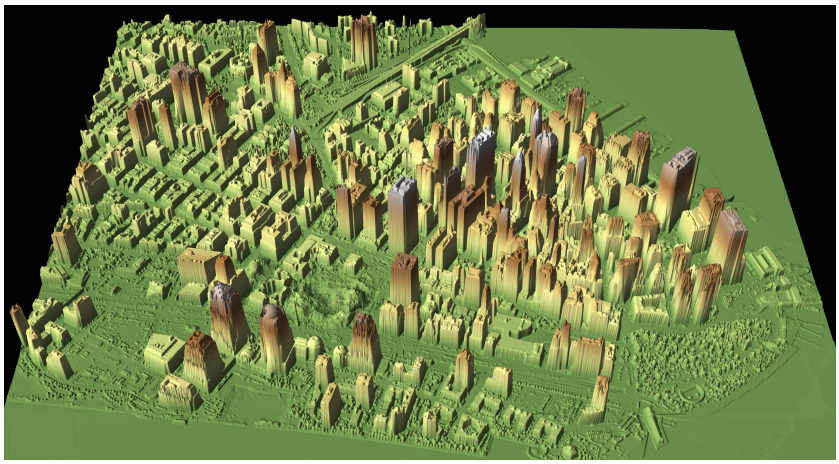
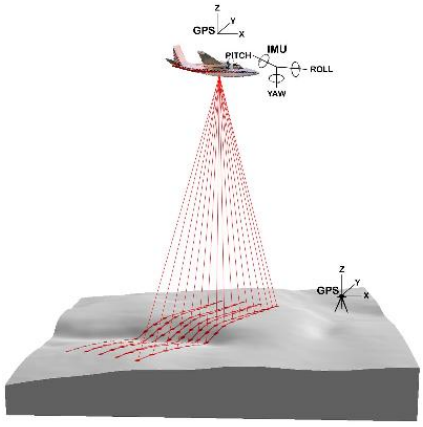
INSA Toulouse
Département Génie Mathématique et Modélisation
4ème année Méthodes et Modèles Numériques
Année 2010/2011



Modélisation, simulation et restitution d'images 3D d'objets : radar laser imageur

Marie Bonnasse-Gahot
Rémi Chauvin



Date de soutenance :
Mercredi 8 juin 2011

Jacques Isbert
Pierre Weiss

Table des matières

Introduction

Les capacités de l'oeil humain étant limitées, en particulier dans de mauvaises conditions climatiques ou la nuit, des technologies ont été élaborées pour combler ces lacunes. On peut ainsi citer l'imagerie laser, ou *ladar*, permettant d'obtenir un nuage de points 3D, en effectuant un balayage de la scène à représenter et en mesurant la distance entre un rayon laser et la première surface opaque trouvée. La qualité et la densité des informations obtenues (formation de nuages de points), constituent un avantage considérable et témoignent de leur succès. On retrouve l'utilisation de la télémétrie laser et de la triangulation dans des environnements très variés. Par exemple, elles ont été utilisées lors de l'accident de Tchernobyl pour modéliser des interventions d'urgence. Elles sont également utilisées dans le domaine militaire : certains drônes sont équipés de ladars permettant de détecter d'éventuels obstacles, des cibles ennemies ou pour cartographier précisemment des terrains. Les environnements non géométriques peuvent également demander leur utilisation comme par exemple dans le domaine de la carrosserie automobile ou pour représenter des durites de connexion sous le capot d'une voiture ou sous un train. Le principe de la télémétrie repose sur la mesure du temps de parcours de la lumière jusqu'à la surface à représenter.

Dans notre étude, nous travaillerons avec un ladar utilisant la méthode de mesure du temps de vol. Celle-ci repose sur l'évaluation du temps entre l'émission d'une impulsion lumineuse laser et le retour, l'observation du point lumineux. C'est le temps mis par la lumière pour effectuer un aller-retour. C'est une méthode directe demandant un matériel très perfectionné et sensible à un grand nombre de paramètres. La mesure du temps de vol est donc une technique qui peut être relativement imprécise.

Dans un premier temps, nous étudierons les performances du ladar en fonction de ses caractéristiques. Nous présenterons ensuite un simulateur modélisant une photo 3D prise par l'appareil. Enfin, nous travaillerons sur l'exploitation de cette photo afin de reconstituer l'image 3D.

Partie 1

Bilan radiométrique

Le bilan radiométrique nous permet de mesurer l'énergie transportée par les rayonnements du lidar. C'est grâce à lui que nous allons pouvoir définir les performances spatiales du capteur c'est-à-dire que nous allons connaître la distance maximale où le laser peut être situé pour avoir une image correcte.
A partir des trois paramètres suivants :

- la densité de puissance reçue par l'objet
- la puissance reçue à l'entrée de l'optique de réception
- la puissance reçue par le photorécepteur

nous pouvons établir la fonction de transfert radiométrique de l'appareil.

1.1 Fonction de transfert (ou bilan) radiométrique

La fonction de transfert radiométrique permet de calculer le nombre de photons qui revient lorsqu'on éclaire une cible. Nous pouvons donc l'assimiler à la puissance totale reçue par le récepteur.
La forme générale de cette fonction est :

$$P_{reue} = P_{laser}\rho T_A^2 T_R T_T \frac{4A_r A_{cible} cos(\theta_S)}{8\pi R^2 (\theta_T R + d_{ta})^2} \tag{1.1}$$

avec :

- P_{laser} est la puissance du laser.
- T_A, transmission atmosphérique
  $T_A = \exp^{-\gamma R}$ où γ est le coefficient d'affaiblissement atmosphérique et dépend de la visibilité (exprimée en km).
- ρ est la réflectance de la cible c'est-à-dire le rapport de l'énergie réfléchie sur l'énergie incidente ; elle dépend principalement de la nature du matériau et de la surface de la cible.
- T_R correspond aux pertes optiques lors de la réception.
- T_T , pertes optiques lors de l'émission.
- A_r est la surface de réception, exprimée en m^2.
- θ_S est l'angle entre la ligne de visée de l'instrument et la perpendiculaire à la surface de la cible, exprimé en radians.
- R est la distance entre la cible et le laser, exprimée en m.
- A_{cible} correspond à la surface de la cible.
- θ_T correspond à la divergence du faisceau laser.
- d_{ta} diamètre du faisceau gaussien à l'entrée de l'optique de collimation

P_{reue} est exprimée en Watt.
Cependant, suivant les dimensions de l'objet éclairé, nous pouvons simplifier cette expression.

1.1.1 Puissance reçue pour une cible résolue

On parle de cible résolue lorsque la cible a des dimensions supérieures aux dimensions du spot laser c'est-à-dire pour $A_{cible} \geq (\theta_T R + d_{ta})$.

$$P_{reue} = P_{laser}\rho T_A^2 T_R T_T \frac{4A_r cos(\theta_S)}{\pi R^2} \tag{1.2}$$

1.1.2 Puissance reçue pour une cible non résolue

Une cible non résolue est une cible dont les dimensions sont inférieures aux dimensions du spot laser c'est-à-dire pour $A_{cible} < (\theta_T R + d_{ta})$.

$$P_{reue} = P_{laser}\rho T_A^2 T_R T_T \frac{4A_r A_{cible} cos(\theta_S)}{\pi^2 \theta_T^2 R^4} \tag{1.3}$$

1.2 Rapport signal à bruit : SNR

Le rapport signal à bruit, SNR, *signal to noise ratio*, permet de mesurer la qualité de la transmission d'une information par rapport aux bruits et de déterminer la précision de la mesure de distance.
Il est défini comme étant le rapport de la puissance reçue par le capteur sur le bruit.

$$SNR = \frac{P_{reue}}{NEP} \tag{1.4}$$

où NEP, *noise equivalent power*, est la puissance (en Watt) reçue par le détecteur correspondant au bruit.

1.3 Comparaison de la distance maximale entre le laser et l'objet en fonction de la visibilité pour trois lasers

A l'aide d'un algorithme de descente de Newton, nous cherchons jusqu'à quelle distance maximale de la cible le laser peut émettre, en fonction de ses caractéristiques,fixées, et de la visibilité.
Pour cela, nous avons chercher la distance maximale R qui minimise la différence entre un SNR fixé et le rapport entre la puissance reçue fonction de la visibilité et de cette distance et le NEP de l'appareil.
Pour une cible résolue, on cherche :

$$\min_{R\in\mathbb{R}} SNR - \left(P_{laser}\rho T_R T_T \frac{4A_r cos(\theta_S)}{\pi}\right) \frac{\exp^{-2\gamma(V)R}}{R^2} \tag{1.5}$$

Pour une cible non résolue :

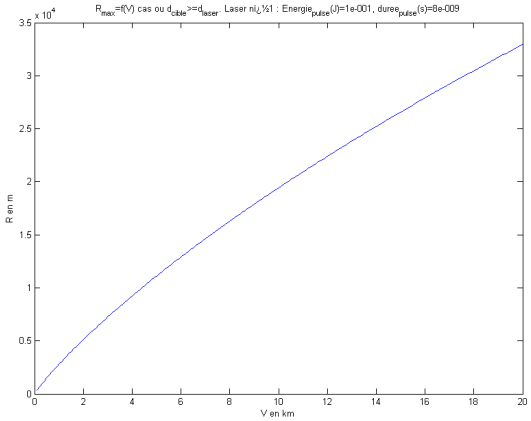
$$\min_{R\in\mathbb{R}} SNR - \left(P_{laser}\rho T_R T_T \frac{4A_r A_{cible} cos(\theta_S)}{\pi^2 \theta_T^2}\right) \frac{\exp^{-2\gamma(V)R}}{R^4} \tag{1.6}$$

Dans cette partie, nous fixons le SNR égal à 7 et nous considérons que la puissance émise est uniforme (et non gaussienne).

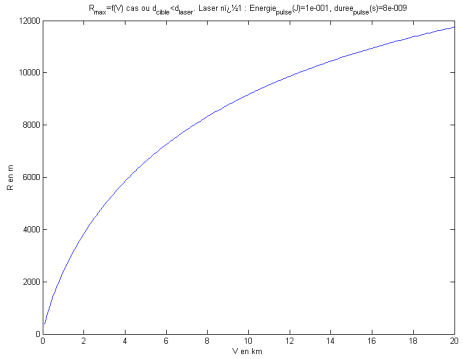
1.3.1 Test numérique pour le premier laser

Caractéristiques du laser 1 :
- Énergie du pulse : E=100 mJ et durée du pulse : 8 ns d'où une puissance P_{laser} = 12,5 MW
- Divergence du faisceau $\theta_S = 0.3 \times 10^-3$ rad
- $T_T = T_R = 0.1$
- Surface de réception 0.005 m^2
- Surface de la cible, pour une cible non résolue 0.01 m^2
- Surface de la cible, pour une cible résolue 1 m^2
- NEP = 3 nW
- $\rho = 0.2$
- $\gamma = 0.503 \times (\frac{3}{V})^{1.08}$

Cible résolue



Cible non résolue

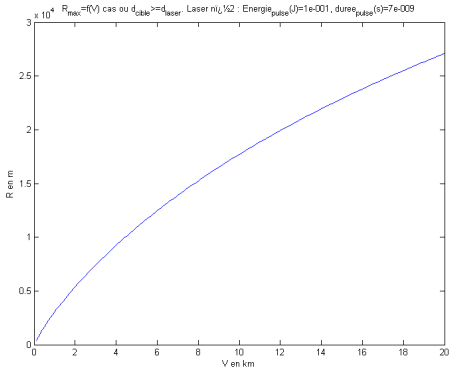


Pour une visibilité de 20 km, la distance maximale à partir de laquelle on peut émettre le faisceau laser 1 en ayant un SNR au moins égal à 7 est de 32,93 km pour une cible résolue et de 11,74 km pour une cible non résolue.

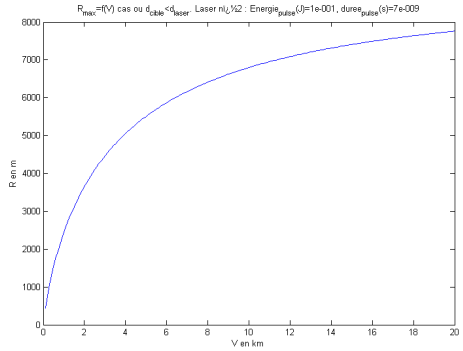
1.3.2 Test numérique pour le deuxième laser

Caractéristiques du laser 2 :
- Énergie du pulse : E=100 mJ et durée du pulse : 6,5 ns d'où une puissance P_{laser} = 15,4 MW
- Divergence du faisceau $\theta_S = 0.3 \times 10^-3$ rad
- $T_T = T_R = 0.1$
- Surface de réception 0.005 m^2
- Surface de la cible, pour une cible non résolue 0.01 m^2
- Surface de la cible, pour une cible résolue 1 m^2
- NEP = 6 nW
- $\rho = 0.25$
- $\gamma = 0.314 \times (\frac{3}{V})^{1.11}$

Cible résolue



Cible non résolue

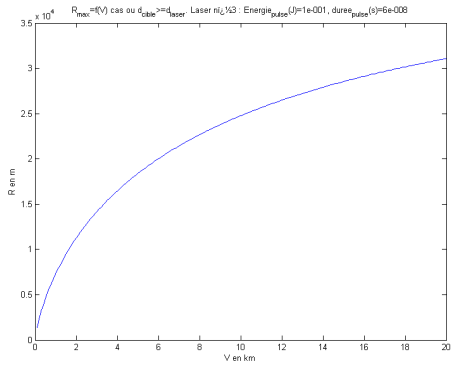


Pour une visibilité de 20 km, la distance maximale à partir de laquelle on peut émettre le faisceau laser 2 en ayant un SNR au moins égal à 7 est de 27,09 km pour une cible résolue et de 7,76 km pour une cible non résolue.

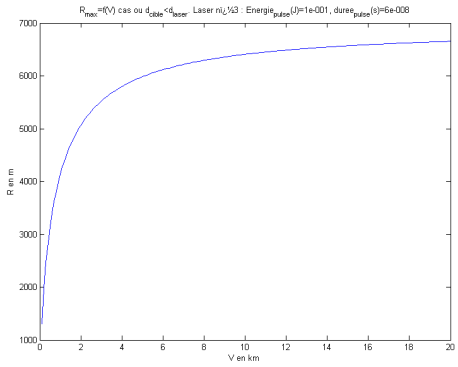
1.3.3 Test numérique pour le troisième laser

Caractéristiques du laser 3 :
- Énergie du pulse : E=100 mJ et durée du pulse : 60 ns d'où une puissance P_{laser} = 1,7 MW
- Divergence du faisceau $\theta_S = 0.3 \times 10^-3$ rad
- $T_T = T_R = 0.1$
- Surface de réception 0.005 m^2
- Surface de la cible, pour une cible non résolue 0.01 m^2
- Surface de la cible, pour une cible résolue 1 m^2
- NEP = 2 nW
- $\rho = 0.2$
- $\gamma = 0.082 \times (\frac{3}{V})^{1.01}$

Cible résolue



Cible non résolue



Pour une visibilité de 20 km, la distance maximale à partir de laquelle on peut émettre le faisceau laser 3 en ayant un SNR au moins égal à 7 est de 31,08 km pour une cible résolue et de 6,66 km pour une cible non résolue.

Partie 2

Analyse temporelle

2.1 Principes généraux et objectif de l'analyse temporelle

Dans cette partie, nous faisons une analyse temporelle du ladar. Cela nous permettra de déterminer la résolution de l'appareil en z, c'est-à-dire la résolution en profondeur. Cette étape ne peut se faire qu'après l'étape précédente de radiométrie. Le principe de l'étude temporelle est la suivante

- On se place à un $R < R_{max}$, en respectant le critère radiométrique.
- On simule l'envoi d'une impulsion gaussienne sur une zone contenant les plans d'équation $z = R$ et $z = R + \Delta R$. Cela va induire deux retours de signal, donc deux gaussiennes, qui vont se superposer.

L'objectif sera de déterminer le ΔR minimum pour lequel on peut distinguer les deux gaussiennes, c'est-à-dire pour lequel le *critère de Rayleigh* est respecté. Pour toute l'étude, nous travaillerons avec les paramètres suivants

Grandeur	Variable	Valeur
Longueur d'onde du laser	λ	$1.06\ \mu m$
Énergie pulse	E	$100\ mJ$
Écart type de pulse	σ_w	$5\ ns$
Puissance pulse	P	$12.5\ MW$
Surface de réception	S	$0.005\ m^2$
Réflectance	ρ	0.2
Visibilité	V	$23\ km$
Coefficient d'atténuation	γ	$0.503\left(\frac{3}{V}\right)^{1.08}$

Dans toute suite, c désigne la célérité de la lumière à savoir $3 \times 10^8\ m.s^-1$

2.2 Description de l'analyse

Dans la partie précédente, l'étude radiométrique nous a permis entre autres choses de déterminer la puissance du signal reçu en fonction des grandeurs mises en jeu. En particulier, dans le cas où la dimension de la cible est supérieure à la surface éclairées (*i.e.* la cible n'est pas entièrement éclairée), on a

$$P_{recue}(P) = P\rho T_T T_R \times \frac{d_r^2}{8R^2} cos(\theta_S) e^{-2\gamma R}$$

L'impulsion que nous envoyons est supposée gaussienne et a pour équation

$$P(t) = \frac{E}{\sigma_w \sqrt{2\pi}} e^{\frac{-t^2}{2\sigma_w^2}}\ ,$$

de sorte que $\int_{-\infty}^{+\infty} P(t)\,dt = E$

Cette impulsion émise par le laser sera réfléchie par une cible à une distance comprise entre une distance minimale R_{min} et maximale R_{max} qui sera notre fenêtre d'analyse de tous les évènements possibles et la largeur de la porte dans la fenêtre distance sera$R_{gate} = (T_{max} - T_{min})\frac{c}{2}$(en mètres) , avec $T_{min} = \frac{2R_{min}}{c}$ qui correspond au temps que met le rayon entre le moment où il est émis et le moment où il est détecté par la capteur, après avoir été réfléchi par le plan $z = R$. Dans cette formule, le facteur 2 est dû au fait que le rayon fasse un aller-retour.

Á une impulsion émise, une ou plusieurs impulsions reviennent vers le récepteur (plusieurs cibles éclairées et à distances égales ou différentes et vues par le récepteur.) D'autre part pour reconstituer correctement une impulsion il faut l'échantillonner et en général on prend 10 échantillons de chaque impulsion. Donc le pas théorique de l'échantillon est : $\Delta t = \dfrac{\sigma_w}{10}$

Dans notre simulation, nous prendrons $R_{min} = 100m$ et $R_{max} = 1km$.

L'échantillonnage du signal de retour sera donc constitué d'éléments rectangulaires de largeur Δt , d'indice temporel k et donc une boucle en indice pour incrémenter les échantillons. Ainsi, si t_k désigne le temps du $k^{\text{ième}}$ échantillon (*i.e.* la k^{ime} itération consiste à intégrer le signal entre $t_k - \frac{\Delta t}{2}$ et $t_k + \frac{\Delta t}{2}$) et en appelant $E(k)$ l'énergie reçue pour cet échantillon, on a

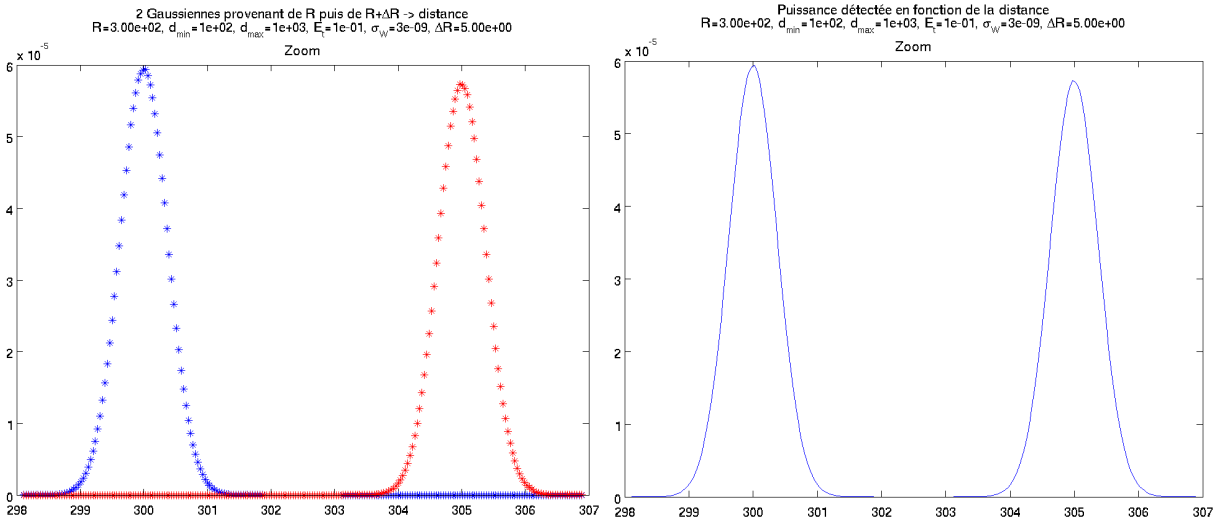
$$E(k) = \int_{t_k - \frac{\Delta t}{2}}^{t_k + \frac{\Delta t}{2}} P_{recue}\big(P(t - 2\frac{R}{c})\big)\, dt$$

et si $P(k)$ désigne la puissance de cet échantillon, on a donc $P(k) = \dfrac{E(k)}{\Delta t}$

Tous ces éléments nous permettent de réaliser un programme de simulation `TestTemps.m` dont le code est présenté en annexe de ce document.

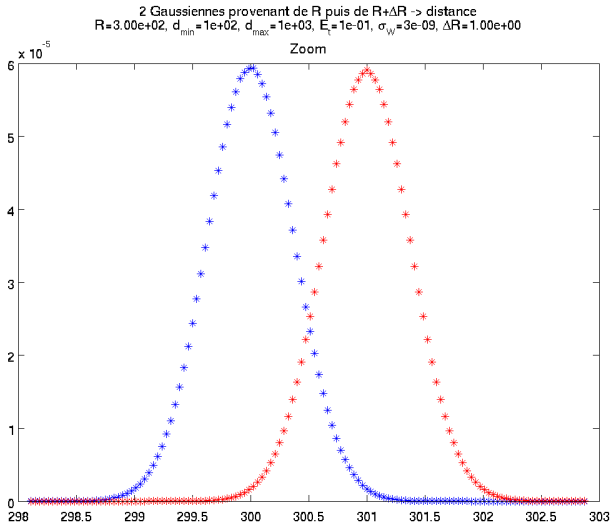
Nous présentons ici nos tests numériques. La première remarque à faire est que la résolution en espace ne dépend pas de R, donc prendrons donc un R constant égal à $300m$. Seul ΔR variera. Pour différentes valeurs de ΔR, nous donnerons 2 figures : la première affichera le retour des deux impulsions provenant des deux plans différents ($z = R$ et $z = R + \Delta R$) et la seconde la superposition des ces deux courbes, c'est-à-dire le signal "réellement" capté.

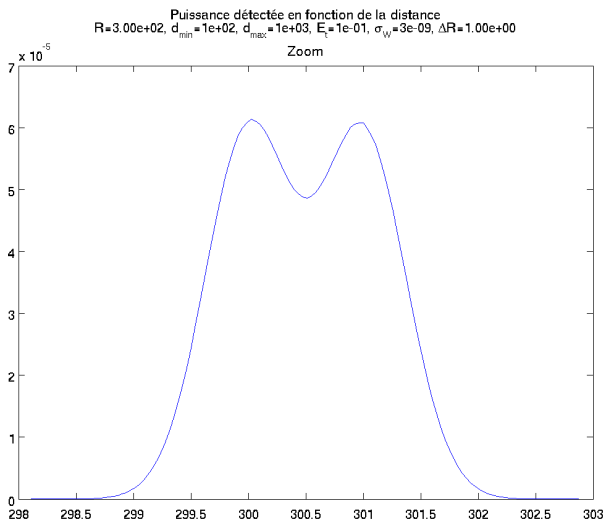
2.2.1 Simulation pour $\Delta R = 5m$



On voit que dans ce cas là, les impulsion sont totalement distinctes, la résolution est donc clairement inférieure

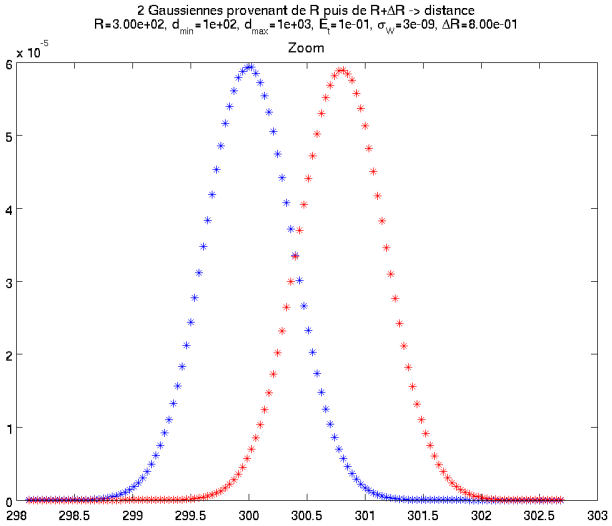
2.2.2 Simulation pour $\Delta R = 1m$

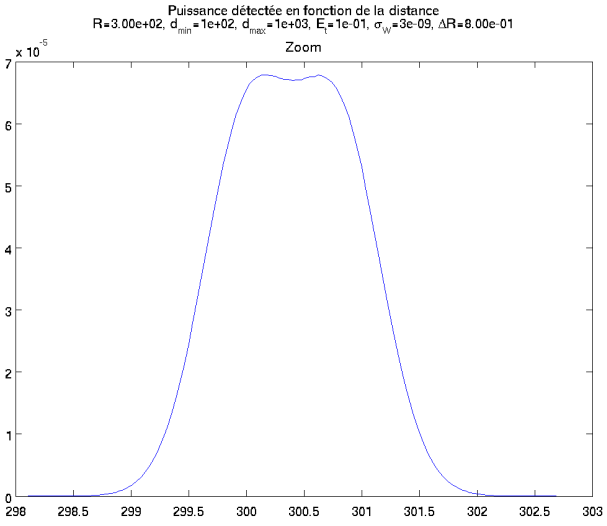




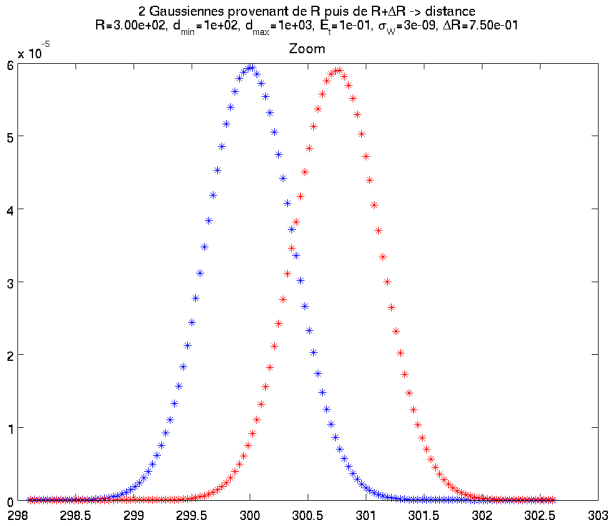
On voit maintenant que les impulsions se chevauchent mais on distingue cependant deux sommets bien identifiés. Le critère de Rayleigh est donc respecté.

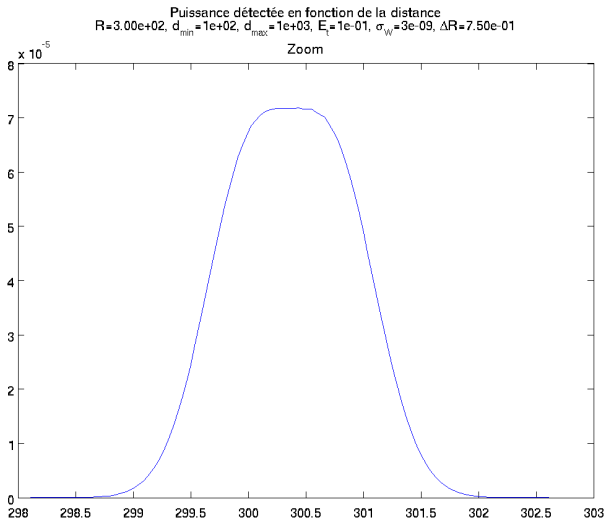
2.2.3 Simulation pour $\Delta R = 0.8m$





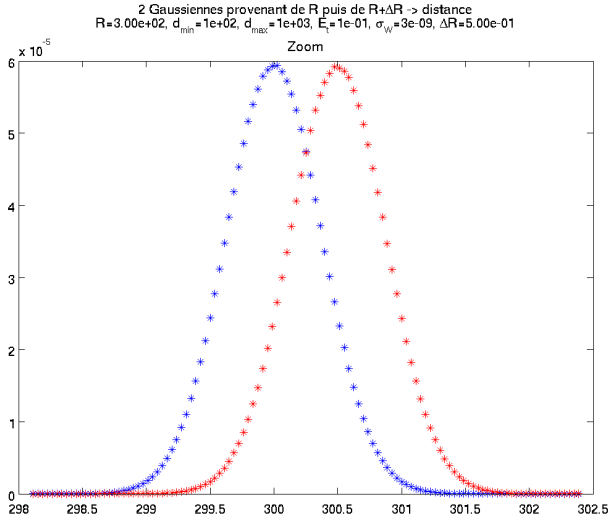
2.2.4 Simulation pour $\Delta R = 0.75m$

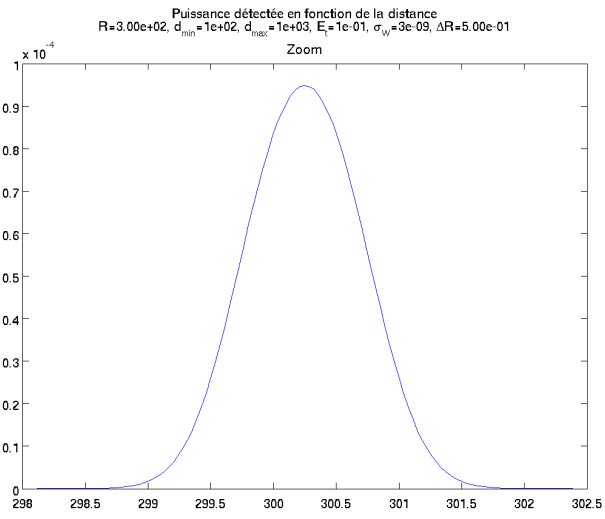




Les deux maxima sont maintenant très difficilement identifiables, on est donc à la limite du critère de Rayleigh.

2.2.5 Simulation pour $\Delta R = 0.5m$





On voit cette fois que le critère de Rayleigh n'est plus du tout respecté puisqu'il n'y a plus qu'un seul maximum identifiable.

La résolution de l'appareil est donc de $0.75m$. Cette résolution correspond à une différence d'altitude que l'on peut voir à l'œil nu. Cependant, on pourrait faire mieux en effectuant un traitement du signal. En effet, on voit sur que pour $\Delta R = 0.5$, les deux gaussiennes se superposent pour ne former qu'une courbe ressemblant à une autre gaussienne. En réalité, cette superposition n'est pas une gaussienne et un traitement du signal nous permettrait d'isoler les deux impulsions. La résolution pourrait donc être améliorée, et ce sans améliorer les caractéristiques de l'appareil.

Partie 3

Construction du simulateur

Dans cette partie, nous décrivons un procédé pour construire un simulateur modélisant une photo 3D du laser sur lequel nous travaillons. Le principe est le suivant : nous construisons un paysage avec un algorithme connu, *diamond-square*. Ensuite, nous simulons la photos en prenant en compte la plupart des bruits mis en jeu, ainsi que les différentes convolution. La partie suivante consistera à déconvoler et débruiter l'image.
Dans tout ce chapitre, $\mathbb{E}$ désigne l'espérance d'une variable aléatoire et $\mathbb{P}$ la probabilité d'un événement.

3.1 Les différents bruits

3.1.1 Le signal utile

Le détecteur CCD se trouvant dans l'appareil capte des photons qui ont une certaine énergie, dépendante de la fréquence du photon. Ces photons sont convertis en électrons avec un rendement η (1 photon donne en moyenne η électrons). Dans notre application, nous prendrons $\eta = 1$ et pourrons donc parler indifféremment de nombre d'électrons ou de photons
Dans un premier temps, on calcule la puissance détectée à l'aide des formules de radiométrie (cf. chapitre 1). L'énergie reçue pendant l'échantillonnage k s'écrit $E(k) = \int_{t_k-\frac{\Delta t}{2}}^{t_k+\frac{\Delta t}{2}} P_{det}(t)\,dt$, où P_{det} représente la puissance détectée. En la supposant constante sur l'intervalle Δt, on a alors $E(k) = P(t_k)\Delta t$. Or, l'énergie d'un photon est égale à $h\nu$ (avec ν fréquence du photon et h constante de Planck). En appelant K le nombre de photons détecté au $k^{\text{ième}}$ échantillonnage, on a donc

$$\mathbb{E}[K] = \frac{P_{det}(k)\Delta t}{h\nu} = \frac{P_{det}(k)\lambda\Delta t}{hc}$$

3.1.2 Modélisation des bruits

Le nombre de photons détecté par le capteur CCD est en réalité une variable aléatoire de Poisson de moyenne $\mathbb{E}[K]$. En appelant $\mathbb{P}[n,k]$ la probabilité d'avoir n photons captés au $k^{\text{ième}}$ échantillonnage, on a ainsi

$$\mathbb{P}[n,k] = \frac{(\mathbb{E}[K])^n\, e^{-\mathbb{E}[K]}}{n!}$$

Ce modèle de variable de Poisson se déduit de la mécanique quantique, nous ne préciserons pas plus cet aspect par manque de connaissance de la discipline.

S'ajoute ensuite un bruit dû à l'éclairement du soleil. En effet, le capteur CCD va recevoir des photons provenant du soleil et qui ne sont donc pas désirés. En appelant N_{bs} le nombre de ces photons détectés, A_B la surface de la cible vue par le plan focal, S_{IB} l'éclairage de la cible par le soleil (en $W.m^{-2}.\mu m$, ce qui représente une irradiance par unité de longueur d'onde exprimé en μm), et $\Delta\lambda$ la largeur de la bande-passante du filtre au niveau des capteurs), on a

$$\mathbb{E}[N_{bs}] = \frac{S_{IB}\Delta\lambda.A_b.\rho T_R T_A d_r^2 \Delta t}{4R^2 h\nu}$$

La simulation de cette variable est la même que précédemment, il suffit de changer la moyenne.

Le troisième bruit présent est le bruit d'obscurité. Ce bruit est associé au nombre d'électrons collectés pendant la mesure. Il est spécifique au détecteur et ne dépend pas du flux (utile ou parasite) reçu. Il est mesuré dans l'obscurité. En posant N_{dark} le nombre de photons provenant de ce phénomène, on a

$$\mathbb{E}[N_{dark}] = \frac{I_{dark}\Delta t}{q_e}$$

où q_e désigne la charge de l'électron, et I_{dark}, le courant d'obscurité, ne dépend que de l'appareil.

On peut donc rajouter ces deux signaux qui représentent la valeur moyenne de la distribution de Poisson

$$N_b = N_{bs} + N_{dark}$$

Enfin, nous avons un bruit thermique qui dépend de la température T(Kelvin) du circuit et de sa capacité C (Farad). Ce bruit est une gaussienne centrée, de variance

$$Q_n^2 = k_B T C / q_e^2$$

où k_B désigne la constante de Boltzmann, égale à l'inverse du nombre d'Avogadro.

Algorithme du simulateur d'une image 3D

Fonction simulation1prise(R,dmin,dmax,cosVerticaleNormale) :
  K : **Nombre d'échantillons**
  E_{detect} : **Vecteur échantillon indicé de 1 à** K
  Pour k **de** 1 **à** K **faire**
    $K(i) \leftarrow \int_{t_k-\Delta t/2}^{t_k+\Delta t/2} P_{detect}(t)dt$
    [P_{detect} calculé en combinant formules de radiométrie et temporelles]
    [dmin et dmax permettent de définir la fenêtre d'analyse temporelle]
  Fin Pour
  Retourner E_{detect}
Fin
Fonction simulateur(X,Y,Z,nbPixels,sigmaXY) :
  [(X,Y,Z) Paysage cartésien "réel", sigmaXY écart type du noyau de convolution de la lentille]
  Se donner dmin, dmax, K
  Signal, u_0,Signal_capté : **matrices nbPixels** $\times K$
  Image_detectée : **vecteur de dimension nbPixels**
  Pour i **de** 1 **à** nbPixels **faire**
    Signal(i, :)$\leftarrow$ simulation1prise(Z(i),dmin,dmax,cosVerticaleNormale)
  Fin Pour
  Pour k **de** 1 **à** K **faire**
    $u_0(:,k) = h*\text{Signal}(:,k)$
    [h gaussienne de variance sigmaXY, $*$ définit le produit de convolution]
    Signal_capté$(:,k) = \mathscr{P}(\text{Signal}(:,k)) + \mathscr{P}(N_b) + \mathscr{N}(0, Q_n^2)$
  Fin Pour
  [$\mathscr{P}(\lambda)$ désigne une loi de Poisson de moyenne λ, $\mathscr{N}$ la loi normale]
  Pour i **de** 1 **à** nbPixels **faire**
    Image_detectée(i) = max (Signal_capté$(i,:)$)
  Fin Pour
  Retourner Image_detectée
Fin

Algorithme 1: Algorithme succint du simulateur

Partie 4

Reconstruction de l'image 3D

Nous allons nous intéresser à la reconstruction de l'image 3D.
Pour cela nous avons utiliser deux méthodes : l'une plutôt "grossière", l'autre plus précise.

Sans les deux cas, le principe est sensiblement le même. Il s'agit de trouver un moyen de déconvoler l'image et de la débruiter. Nous allons voir immédiatement que ces deux critères sont contradictoires et qu'il nous faudra donc faire un compromis.

4.1 Position très générale du problème

Soit u l'image réelle. On la modélise comme étant une fonction de $\mathbb{R}^2$, c'est-à-dire qu'à un point (x, y) de $\mathbb{R}^2$ on associe une altitude u. Dans une optique imagerie, et donc pixels, u est ramené à un vecteur de $\mathbb{R}^n$. Soit u_0 l'image captée par le laser. u_0 est l'image réelle convolé par la lentille de l'appareil, dont le noyau de convolution h est supposé connu et gaussien. S'ajoutent ensuite des bruits que nous avons décrit précédemment, qui eux ne sont pas convolés (les bruits interviennent au niveau des capteurs et du circuit, donc après la lentille). On peut modéliser le bruit par une fonction de $\mathbb{R}^n$ appelée b. Nous supposons que b est un vecteur gaussien dont toutes les composantes sont de moyennes nulles, de variances égales et indépendantes.

On peut alors écrire

$$u_0 = h * u + b$$

L'application $u \to h * u$ étant linéaire, on peut écrire $h * u = Hu$, où H est un opérateur linéaire de $\mathbb{R}^n$ dans $\mathbb{R}^n$. On peut donc le représenter comme une matrice $\mathbb{R}^n \times \mathbb{R}^n$, qui a la particularité d'être diagonalisable dans l'espace de Fourier. On a ainsi le modèle général

$$u_0 = Hu + b \tag{4.1}$$

On appelle u^* la solution de l'image déconvolée et débruitée.
On peut écrire $b = u_0 - Hu$. b étant de moyenne nulle, on doit avoir $Hu^* - u_0$ proche de 0, donc $\|Hu^* - u_0\|_2^2$ proche de 0. De plus, le bruit est un phénomène associé à des hautes fréquences, et a donc pour conséquence de faire exploser ∇u^*. Pour se rapprocher de la solution, il faut que ∇u^* ne soit pas trop grand.

Pour palier à ces deux phénomènes a priori contradictoires, on va donc résoudre le problème de minimisation suivant

$$u^* = \underset{u \in \mathbb{R}^n}{\arg\min} \|\nabla u\|_p^p + \lambda \|Hu - u_0\|_2^2 \tag{4.2}$$

où $\alpha \in \{1, 2\}$ et $\lambda > 0$ sera un paramètre à choisir "intelligemment". Pour l'instant, nous ne présumons rien sur la norme utilisée pour ∇u. Remarquons que plus λ sera grand moins le bruit sera filtré. Á l'inverse, le fait d'augmenter λ permet de mieux déconvoler l'image.

On peut ensuite raffiner le modèle, en favorisant des pixels par rapport à d'autres. Par exemple, il se peut que dans notre imageur, certains pixels ne reçoivent pas d'information suffisante pour être traitée. Ce que l'on fait dans ces cas là consiste à se donner une matrice D de taille $n \times n$, diagonale. On met le terme diagonal à 1 si on reçoit de l'information, à 0 sinon. Le fait de mettre un 0 revient à dire qu'on n'utilise pas le signal reçu par le pixel. On pourrait également prendre n'importe quel nombre entre 0 et 1 : plus le pixel reçoit un signal important et plus son coefficient est proche de 1, et inversement. Le problème le plus général s'écrit donc

$$u^* = \underset{u \in \mathbb{R}^n}{\arg\min} \|\nabla u\|_p^p + \frac{\lambda}{2} \|D(Hu - u_0)\|_2^2 \tag{4.3}$$

4.2 Première méthode : utilisation de la norme 2 pour le gradient

Dans un premier temps, nous allons tenter de résoudre :

$$\min_{x\in\mathbb{R}^{3n}} ||\nabla x||_2^2 + \frac{\lambda}{2}||Hx - x_0||_2^2$$

où H est un noyau de convolution et ∇x indique une discrétisation de l'opérateur gradient.
Nous posons $J(x) = ||\nabla x||_2^2 + \frac{\lambda}{2}||Hx - x0||_2^2$.
Les conditions d'optimalités du problème de minimisation sont : $\nabla J(x) = 0$.
En calculant ∇J nous trouvons :

$$\nabla J(x) = 2\nabla^T \nabla x + \lambda H^T(Hx - x_0)$$

Le système linéaire résultant à résoudre est donc :

$$2\nabla^T \nabla x + \lambda H^T(Hx - x_0) = 0$$

d'où

$$x = \left(2\nabla^T \nabla + \lambda H^T H\right)^{-1} \lambda H^T H x_0$$

Nous utilisons la transformée de Fourier pour le résoudre, plus rapide qu'une méthode itérative de type gradient conjugué, car les opérateurs $\nabla^T\nabla$ et $H^T H$ peuvent être diagonalisés par la transformée de Fourier et donc plus faciles à manipuler pour le calcul.

Les résultats obtenus sont acceptables pour une première approche , mais peuvent cependant être nettement améliorés !

4.3 Deuxième méthode : l'algorithme de Chambolle-Pock

La méthode précédente nous limitait car on utilisait la norme 2 pour le gradient de u. Pour comprendre le problème soulevé, nous raisonnons maintenant en discret. Chercher u en minimisant $\|\nabla u\|_2^2$ entraine nécessairement que ∇u est dans L^2, donc $u \in H^1$. u n'admet donc aucune discontinuité le long des courbes. Or, dans une image 3D, le fait par exemple qu'il puisse se trouver un objet au milieu d'une scène engendre des discontinuités. L'espace utilisé n'est donc pas adapté. Pour palier le problème, on utilise $\|\nabla u\|_1$. Cette norme, plus compliquée à manipuler, autorise les discontinuités. De plus, elle est parfaitement adaptée au modèle utilisé car elle mesure les variations de u. La minimiser revient donc à minimiser les variations de u, et donc le bruit.

Cette deuxième méthode consiste donc à minimiser :

$$\min \|\nabla x\|_1 + \frac{\lambda}{2}\|D(Hx - x_0)\|_2^2 \tag{4.4}$$

où

- D est une matrice diagonale, permettant de favoriser certains pixels par rapport à d'autres
- H est le noyau de convolution

en utilisant l'algorithme de Chambolle-Pock [2]
Nous aurions pu utiliser un algorithme de points intérieurs dont le taux de convergence aurait été meilleur. Mais ce type d'algorithme, nous utilisons une méthode de Newton : il faut donc calculer la hessienne du problème et l'inverser à chaque itération. Notre problème ayant de grandes dimensions, le coût des opérations est alors très grand. C'est pourquoi nous n'avons pas choisi un algorithme de points intérieurs pour résoudre notre problème.
Nous choisissons donc l'algorithme de Chambolle-Pock qui consiste à résoudre le problème dual d'un problème de minimisation :

$$\min_{x\in X} F(Kx) + G(x) \text{ (problème primal)} \tag{4.5}$$

$$\max_{y\in\mathbb{R}^{2n}} -(G^*(-K^*y) + F^*(y)) \text{ (problème dual)} \tag{4.6}$$

où

- $X et Y$ sont deux espaces vectoriels munis d'un produit scalaire.
- $K : X \to Y$ est un opérateur linéaire
- $G : X \to \mathbb{R}^+$ et $F : Y \to \mathbb{R}^+$ sont des fonctions convexes, semi-continues inférieurement (s.c.i.)

L'algorithme de Chambolle-Pock est le suivant : - Initialisation : on choisit τ, $\sigma > 0$, $\theta \in [0,1]$, $(x^0, y^0) \in X \times Y$ et on pose $x^0 = x^0$
-Pour $n \geq 0$, on effectue l'itération suivante :

$$\begin{cases} y^{n+1} = (I + \sigma \partial F^*)^{-1}(y^n + \sigma K x^n) \\ x^{n+1} = (I + \tau \partial G)^{-1}(x^n - \tau K^* y^{n+1}) \\ x^{n+1} = x^{n+1} + \theta(x^{n+1} - x^n) \end{cases}$$

Nous pouvons remarquer qu'à l'heure actuelle, l'algorithme de Chambolle-Pock est la méthode la plus optimale pour résoudre notre type de problème.
Pour mieux comprendre, et donc utiliser, cet algorithme nous avons besoin de quelques définitions.

4.3.1 Définitions

Définition 4.3.1 (Opérateur proximal). *Soit $f : \mathbb{R}^n \to \mathbb{R}$ une fonction convexe, sci.*
L'opérateur proximal de f, ou résolvante de f, est défini par :

$$prox_f(x) = (I + \partial f)^{-1}(x) = arg \min_{x' \in \mathbb{R}^n} f(x') + \frac{1}{2}||x - x'||_2^2$$

Pour que x^* soit solution, on doit avoir

$$\begin{aligned} \partial f(x^*) + x^* - x \ni 0 &\Leftrightarrow x \in \partial f(x^*) + I(x^*) \\ &\Leftrightarrow x^* \in (I + \partial f)^{-1}(x) \end{aligned}$$

Nous pouvons remarquer que l'opérateur proximal est unique car il fait intervenir $||.||_2$.

Définition 4.3.2 (Fonction "duale"). *Soit $f : \mathbb{R}^n \to \mathbb{R}$ une fonction convexe, sci.*
La fonction "duale" (ou polaire ou conjugué de Fenchel) est définie par :

$$f^*(x) = \sup_{x' \in \mathbb{R}^n} < x, x' > - f(x')$$

Propriétés 1.

- *f^* est convexe sci*
- *$f^{**} = f$ (si f est convexe sci)*

Si f n'est pas convexe, f^{**} est tout de même convexe. Elle est appelée relaxation convexe de f.

4.3.2 Première position du problème

Pour notre problème nous posons dans un premier temps :

$$F(x) = ||x||_1 \tag{4.7}$$

$$G(x) = \frac{\lambda}{2}||D(Hx - x_0)||_2^2 \tag{4.8}$$

$$K = \nabla \tag{4.9}$$

$$\tag{4.10}$$

Nous avons donc besoin de calculer les opérateurs proximaux de G et F^* pour notre algorithme.

Opérateur proximal de F^*

$$F^*(y) = \sup_{y' \in \mathbb{R}^{2n}} < y', y > -||y'||_1$$

Nous pouvons décomposer cette équation en n problèmes indépendants.
Nous obtenons alors la formulation suivante : $\forall i \in [1 : 2n]$

$$\begin{aligned}(F^*(y))_i &= \sup_{y'_i \in \mathbb{R}^2} < y'_i, y_i > -||y'_i||_1 \\ &\leq ||y'_i||_2 ||y_i||_2 - ||y'_i||_1\end{aligned}$$

Pour maximiser le produit scalaire, les vecteurs y'_i et y_i doivent être colinéaires c'est-à-dire $||y'_i||_2 = \lambda ||y_i||_2, \lambda > 0$.
On a alors : $< y'_i, y_i >= ||y'_i||_2 ||y_i||_2$ et donc

$$\sup_{y_i \in \mathbb{R}^2} < y'_i, y_i > -||y'_i||_1 = \begin{cases} 0 \text{ si } ||y_i||_2 \leq 1 \\ +\infty \text{ sinon} \end{cases}$$

Opérateur proximal de G

$$(I + \tau \partial G)^{-1}(x) = arg \min x' \in \mathbb{R}^n \frac{\lambda}{2} ||D(Hx' - x_0)||_2 + \frac{1}{2} ||x' - x_0||_2$$

Les conditions d'optimalités du problème nous donnent :

$$\lambda H^T D^T (D(Hx' - x_0)) + x' - x_0 = 0$$

soit

$$(\lambda H^T D^T DH + I)x' = H^T D^T x_0 + x_0$$

Le problème que nous rencontrons ici est que la matrice H est beaucoup trop grande pour être stocker. Nous utilisons alors l'astuce de passer par sa transformée de Fourier :

$$\lambda H^T D^T DH + I = \lambda \mathcal{F}^{-1} D_H^* \mathcal{F} D^T D \mathcal{F}^{-1} D_H \mathcal{F} + I$$

Comme la matrice obtenue est définie positive et symétrique, nous pouvons alors utiliser une méthode de gradient conjugué linéaire pour résoudre le système.
Toutefois, cette méthode nous paraît un peu compliquée et nous préférons repositionner notre problème afin d'avoir une méthode de résolution plus simple.

4.3.3 Deuxième position du problème

Nous rappelons notre problème de départ :

$$\min ||\nabla x||_1 + \frac{\lambda}{2} ||D(Hx - x_0)|| \tag{4.11}$$

que nous résolvons à l'aide de l'algorithme de Chambolle-Pock dont le problème général est, rappelons-le :

$$\min_{x \in \mathbb{R}^n} F(Kx) + G(x) \text{ (problème primal)} \Leftrightarrow \max_{y \in \mathbb{R}^{2n}} -(G^*(-K^* y) + F^*(y)) \text{ (problème dual)} \tag{4.12}$$

Nous reformulons alors notre problème en posant :

$$\begin{aligned}&- K : \mathbb{R}^n \to \mathbb{R}^{3n} \\ &x \to \begin{pmatrix} \nabla x \\ DHx \end{pmatrix} \\ &- F : \mathbb{R}^{3n} \to \mathbb{R} \\ &\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \to || \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} ||_1 + \frac{\lambda}{2} ||x_3 - x_0||_2^2 \\ &- G(x) = 0\end{aligned}$$

Ainsi, même si nous avons besoin de calculer les opérateurs proximaux de G et F^* pour notre algorithme, l'opérateur proximal de G se calcule facilement :

$$(I + \tau\partial G)^{-1}(y) = y$$

De même que K^*, $K^* = K^T$ car nous sommes dans $\mathbb{R}$.
Seuls nous restent à trouver la fonction duale de F, F^*, et son opérateur proximal.

Fonction duale de F

$$\begin{aligned} F^*(y) &= \sup_{y' \in \mathbb{R}^{3n}} \langle y', y\rangle - F(y') \\ &= \sup_{y' \in \mathbb{R}^{3n}} \left\langle \begin{pmatrix} y'^1_1 \\ y'^1_2 \end{pmatrix}, \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} \right\rangle - \left\| \begin{pmatrix} y'^1_1 \\ y'^1_2 \end{pmatrix} \right\|_2^2 + \langle y'_3, y_3\rangle - \frac{\lambda}{2} \|y'_3 - x_0\|_2 \end{aligned}$$

Ce problème se décompose en n problèmes indépendants

$$F^*(y) = \sum_{i=1}^{n} \sup_{(y')^i \in \mathbb{R}^3} \left\langle \begin{pmatrix} (y')^i_1 \\ (y')^i_2 \end{pmatrix}, \begin{pmatrix} y^i_1 \\ y^i_2 \end{pmatrix} \right\rangle - \left\| \begin{pmatrix} (y')^i_1 \\ (y')^i_2 \end{pmatrix} \right\|_2^2 + \langle (y')^i_3, y^i_3\rangle - \frac{\lambda}{2} \left\|(y')^i_3 - x^i_0\right\|_2$$

Résolvons donc ces problèmes en 3 dimensions :

$$\begin{aligned} &\sup_{y' \in \mathbb{R}^3} \left\langle \begin{pmatrix} y'_1 \\ y'_2 \end{pmatrix}, \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} \right\rangle - \left\| \begin{pmatrix} y'_1 \\ y'_2 \end{pmatrix} \right\|_2^2 + \langle y'_3, y_3\rangle - \frac{\lambda}{2} \|y'_3 - x_0\|_2 \\ = &\sup_{y' \in \mathbb{R}^2} \left\langle \begin{pmatrix} y'_1 \\ y'_2 \end{pmatrix}, \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} \right\rangle - \left\| \begin{pmatrix} y'_1 \\ y'_2 \end{pmatrix} \right\|_2^2 + \sup_{y' \in \mathbb{R}} \langle y'_3, y_3\rangle - \frac{\lambda}{2} \|y'_3 - x_0\|_2 \end{aligned}$$

Après développement des calculs, on trouve

$$F^*(y) = \frac{1}{2\lambda} \|y_3\|_2^2 + \langle y_3, x_0\rangle + \begin{cases} 0 & \text{si } \forall i = 1, ..., n \quad \left\| \begin{pmatrix} (y')^i_1 \\ (y')^i_2 \end{pmatrix} \right\|_2 \leq 1 \\ +\infty & \text{sinon} \end{cases}$$

Opérateur proximal de F^*

Après calcul, on trouve

$$\begin{aligned} (I + \sigma\partial F^*)^{-1}(y) &= \arg\min_{y' \in \mathbb{R}^3} \sigma F^*(y') + \frac{1}{2} \|y' - y\|_2^2 \\ &= \begin{pmatrix} \Pi_\kappa \begin{pmatrix} y'_1 \\ y'_2 \end{pmatrix} \\ \dfrac{y_3 - \sigma x_0}{1 + \frac{\sigma}{\lambda}} \end{pmatrix} \end{aligned}$$

Où Π_κ représente la projection de $\begin{pmatrix} y_1 \\ y_2 \end{pmatrix}$ sur l'espace $\kappa = \left\{ \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} \in \mathbb{R}^2, \ \left\| \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} \right\|_2 \leq 1 \right\}$

Partie 5

Test numériques

Dans ce chapitre, nous exposons les tests numériques de la deuxième méthode exposée.

5.1 Cas images 2D

Dans un premier temps, nous testons l'algorithme sur des simples images 2D. Nous utilisons pour cela une image très classique en test de traitement d'image, *Le Cameraman*
Nous prenons cette image convolée avec un noyau gaussien puis bruitée par une gaussienne. Nous présentons les résultats obtenus. On appelle σ l'écart type du noyau de convolution, σ_b celui du bruit.

- $\sigma = 0.02$, $\sigma_b = 10^{-3}$

FIGURE 5.1 – Image réelle

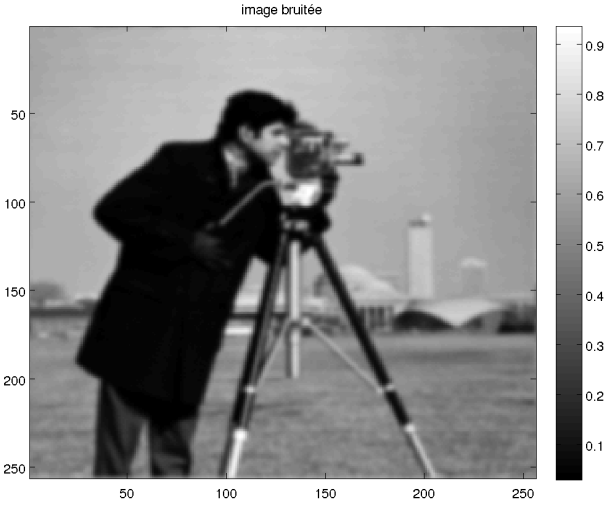


FIGURE 5.2 – Image convolée et bruitée

FIGURE 5.3 – Image traitée

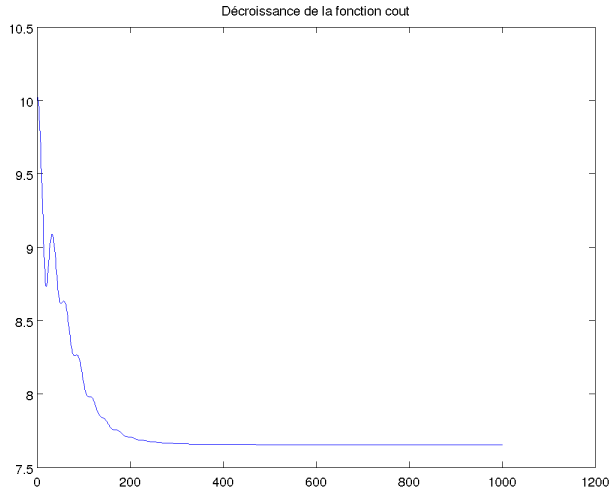


FIGURE 5.4 – Variation de la fonction coût

On observe que sur un cas "simple" (noyau de convolution et bruit de variances faibles), la reconstitution de l'image est très bonne.

- $\sigma = 0.02$, $\sigma_b = 0.05$
  On observe maintenant comment l'algorithme réagit avec plus de bruit. Nous n'avons pas réussi à trouver des paramètres donnant un résultat vraiment satisfaisant. Voici la meilleure image obtenue.

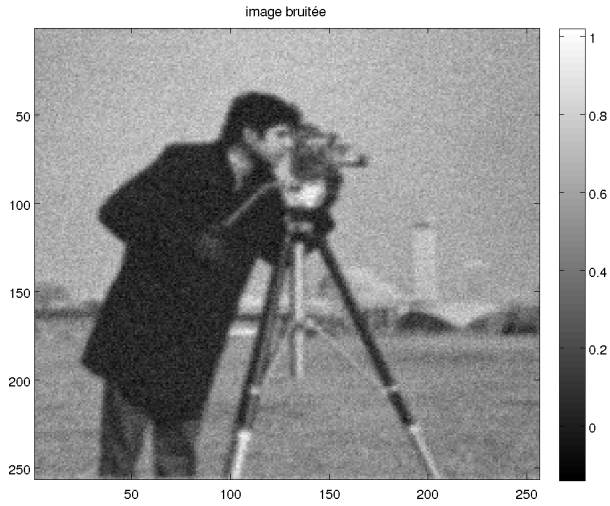


FIGURE 5.5 – Image convolée et bruitée

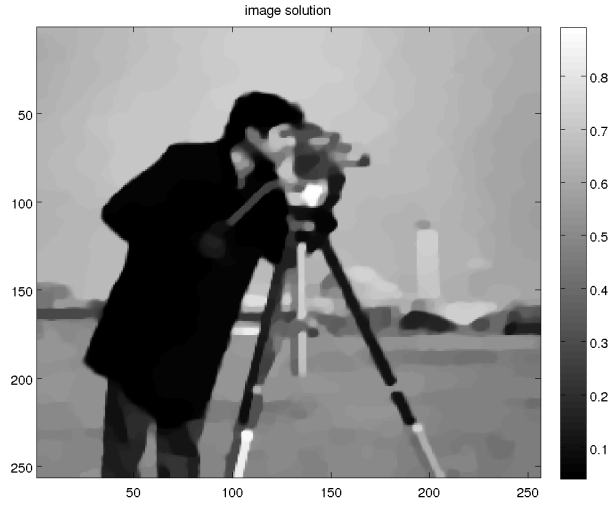


FIGURE 5.6 – Image traitée

- $\sigma = 0.02$, $\sigma_b = 10^{-3}$, 10% de pixels
  Nous reprenons maintenant les paramètres initiaux mais en perdant 90% des pixels. Cette fois, les résultats sont impressionnant car avec seulement 10% de l'information, on reconstitue l'image originelle de façon tout-à-fait acceptable ! Rappelons que les applications pour lesquelles nous travaillons sont purement militaires, et dans le cas présent, bien que la photos obtenue ne soit pas très "jolie", tous les objets sont identifiables.

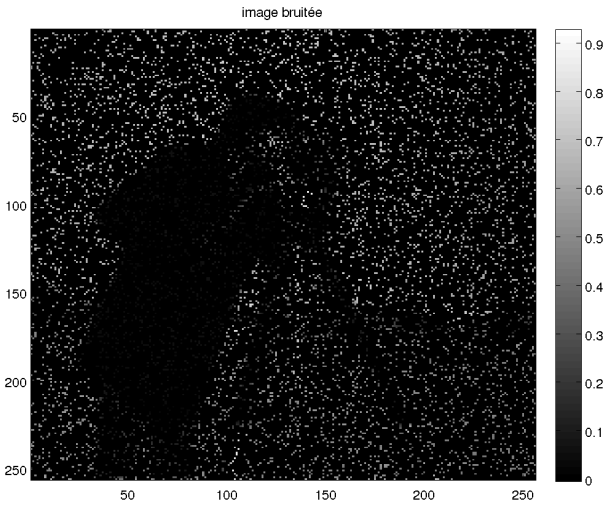


FIGURE 5.7 – Image convolée et bruitée

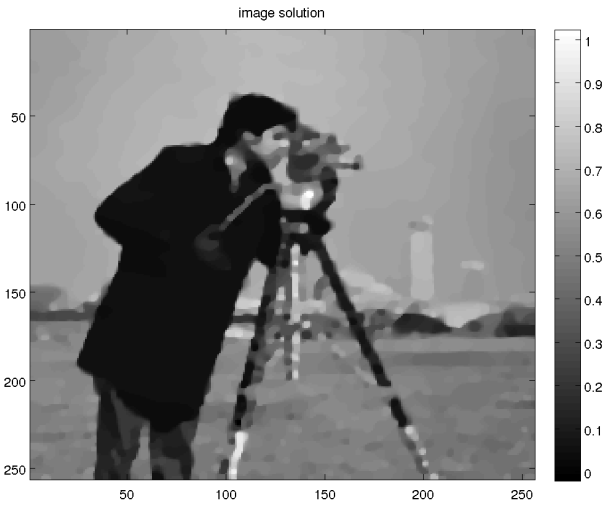


FIGURE 5.8 – Image traitée

5.2 Images 3D

5.2.1 Première approche via un objet simple

Nous avons réalisé une simulation sur un objet sensé représenter un cochon. Nous avons trouvé sur une base de donnée de l'INRIA [3] un cochon maillé en éléments finis triangulaire. Cette modélisation étant trop complexe dans le cas que nous traitons, nous avons transformé cette image en un objet sur une grille cartésienne : en tout point de $\mathbb{R}^2$, on associe une altitude. L'objet que nous traitons est alors le cochon vu de dessus. L'image obtenue n'étant pas très fidèle, nous prions le lecteur de faire preuve d'imagination !

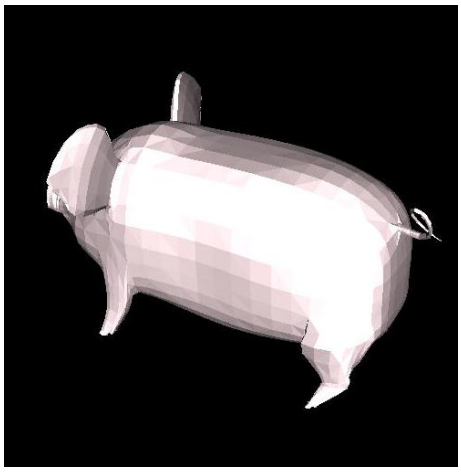


FIGURE 5.9 – Image maillée

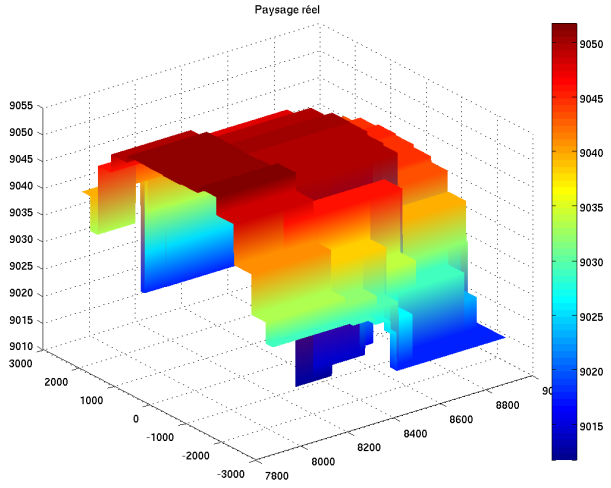


FIGURE 5.10 – Image cartésienne

Nous présentons maintenant l'image bruitée par le simulateur puis trois reconstitutions pour différents λ. Plus on augmente λ, moins on a tendance à pénaliser le gradient. La conséquence est la suivante : en augmentant λ, on augmente la netteté des contours mais on filtre moins de bruit.

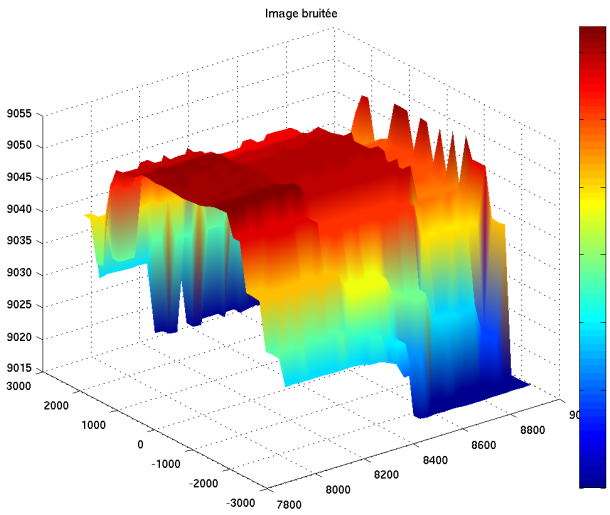


FIGURE 5.11 – Image convolée et bruitée

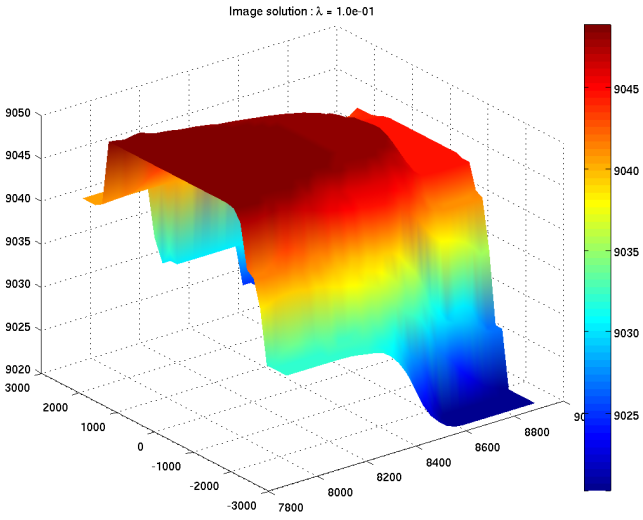


FIGURE 5.12 – Image traitée, $\lambda = 0.1$

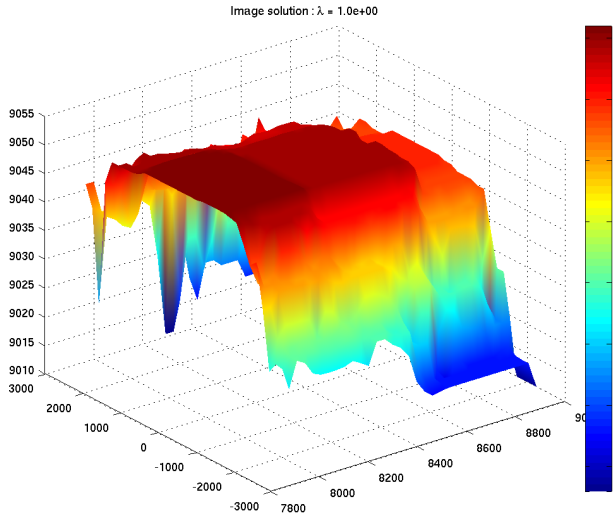


FIGURE 5.13 – Image traitée, $\lambda = 1$

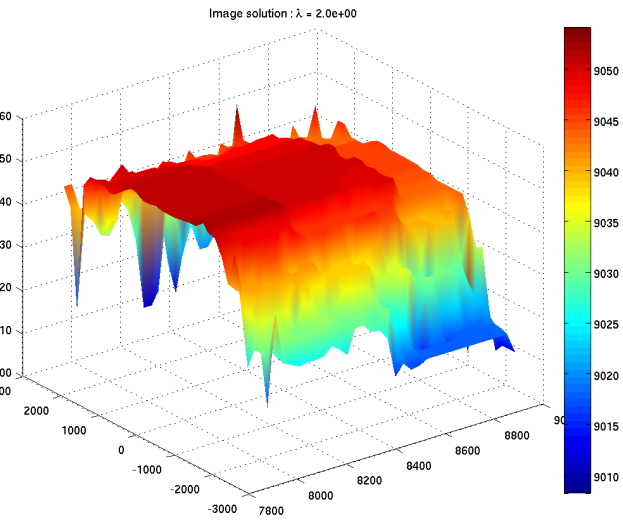


FIGURE 5.14 – Image traitée, $\lambda = 2$

La remarque faite au-dessus est confirmée par les tests numériques. Avec un petit λ pénalisant fortement le gradient, on observe une image lisse, dépourvue de bruit mais dont les contours sont très mal définis. En augmentant λ, les contours sont de plus en plus nets mais l'image est plus bruitée.

Bilan

Cette petite expérience nous permet de valider notre code pour des images 3D. La restitution d'image n'est cependant pas parfaite, mais il faut sans doute, comme nous l'expliquerons dans la suite de cette exposé, plus incriminer le modèle de la reconstitution que l'algorithme.

5.2.2 Reconstitution d'un paysage

Maintenant, nous travaillons avec un paysage provenant de la même banque de données. Dans la simulation précédente, nous avions placé l'objet à environ 9000 m du laser. Cette fois-ci, afin de diminuer le bruit, nous plaçons l'image à 5000 m. Dans cette simulation, nous verrons également comment réagit le modèle sur une image avec des pixels manquants.
Bien que la reconstitution ne soit toujours pas des plus fidèles, il est assez simple d'imaginer que la figure cartésienne est un paysage !

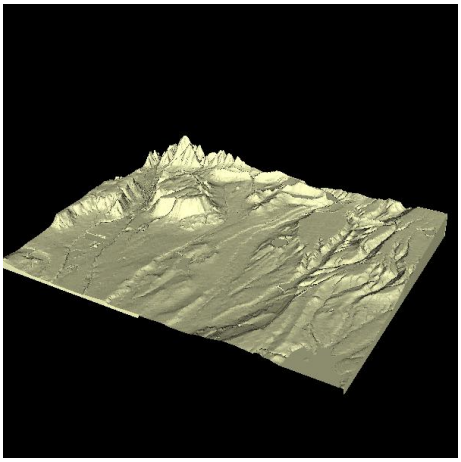


FIGURE 5.15 – Image maillée

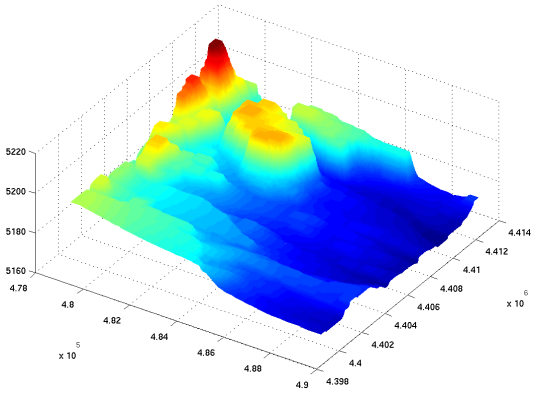


FIGURE 5.16 – Image cartésienne

Voici le paysage bruité en convolé que l'on obtient avec le simulateur.

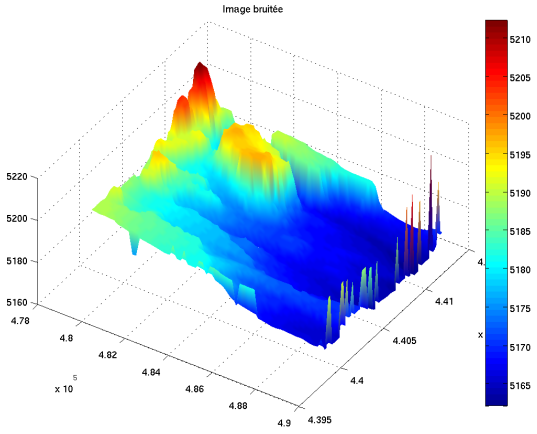


FIGURE 5.17 – Paysage bruité et convolé

Voici le résultat du traitement de l'image pour différents λ.

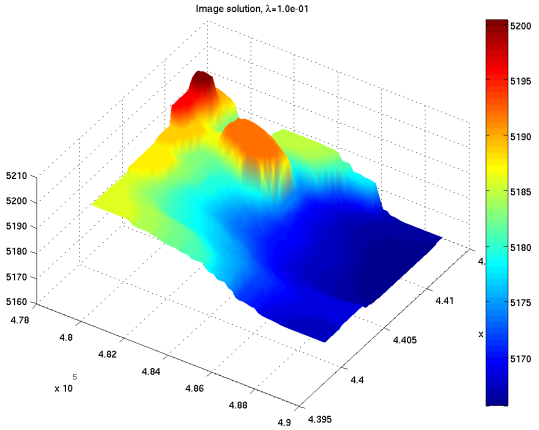


FIGURE 5.18 – $\lambda = 0.1$

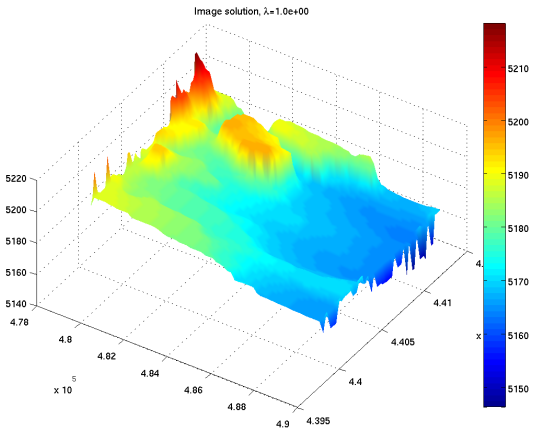


FIGURE 5.19 – $\lambda = 1$

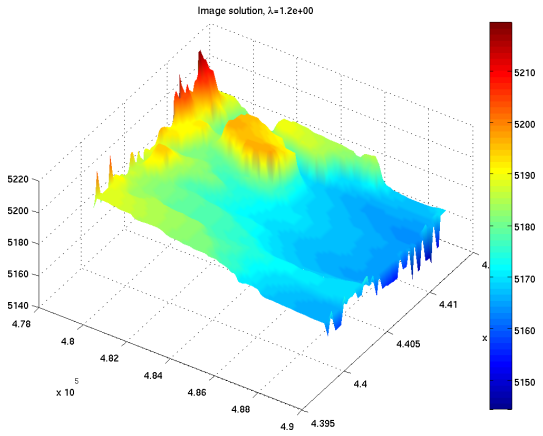


FIGURE 5.20 – $\lambda = 1.2$

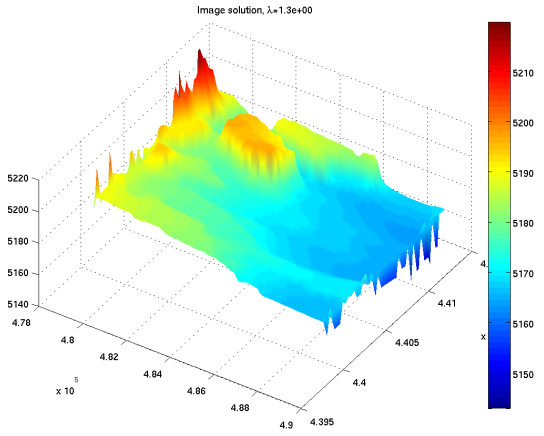


FIGURE 5.21 – $\lambda = 1.33$

Traitement d'une image avec perte de pixels

Nous allons maintenant considérer dans notre modèle qu'un pixel possède une probabilité $\frac{1}{2}$ de contenir de l'information. Nous perdons donc environ la moitié de l'information. Voici l'image que nous obtenons et l'image reconstituée, avec $\lambda = 1.2$.
Remarque : l'image avec perte affichée n'est pas vraiment représentative de la quantité d'information que l'on a, Matlab ne parvient pas à afficher correctement les images avec pixels manquant. En particulier pour 90% de perte, l'image est totalement vide !

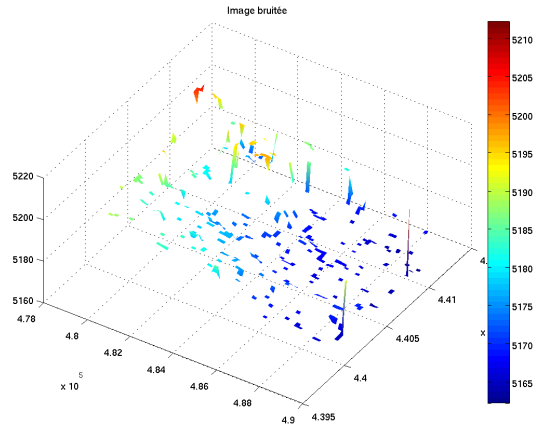


FIGURE 5.22 – Image avec 50% de perte

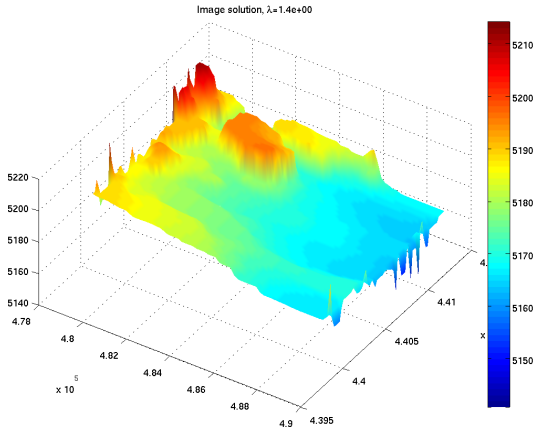


FIGURE 5.23 – Reconstitution de l'image

Les résultats sont assez surprenant, car nous parvenons à reconstituer l'image quasiment aussi bien que quand nous n'avons pas de perte dans les données. On peut considérer que la perte d'information, qui s'élève pourtant à 50%, est presque négligeable devant la convolution et le bruit. Réitérons le test avec cette fois 70% de perte. Expérimentalement, on constate qu'augmenter légèrement λ améliore la solution, on prend donc $\lambda = 1.4$.

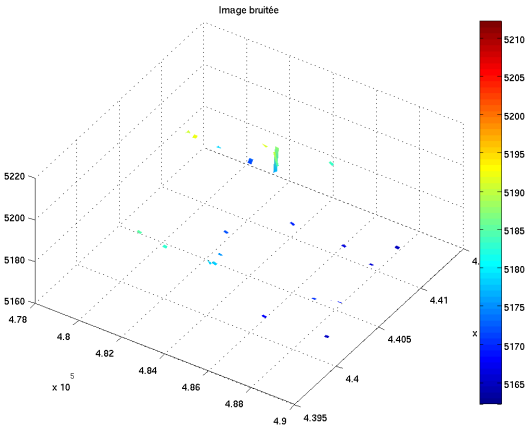


FIGURE 5.24 – Image avec 70% de perte

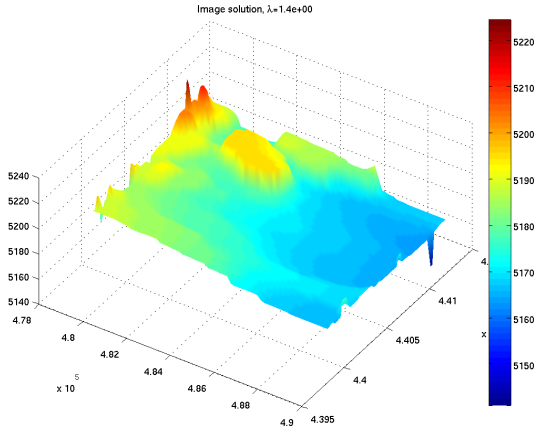


FIGURE 5.25 – Reconstitution de l'image

Le résultat reste tout-à-fait acceptable au vu de la quantité d'information. Même pour 90% de perte, le résultat est encore très surprenant :

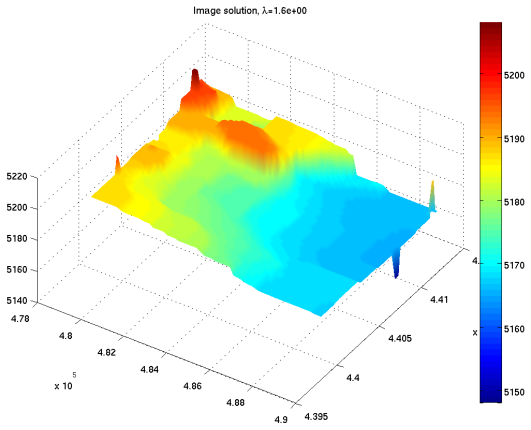


FIGURE 5.26 – Reconstitution de l'image avec 90% de perte

Conclusion

Nous avons donc essayé, à partir d'une image dégradée par un simulateur censé approcher les phénomènes physiques mis en jeux dans l'imagerie 3D, d'obtenir une image la plus nette et la plus proche de la réalité possible. Pour cela, il nous fallait à la fois débruiter l'image, ce qui avait tendance à la lisser, et à la fois la déconvoler, ce qui avait plutôt tendance à améliorer la netteté des contours. Nous avions donc deux critères contradictoires, que nous avons sommés et minimisés. L'algorithme de minimisation utilisé est en la matière le meilleur connu à ce jour pour le problème considéré.

Nous avons constaté que l'algorithme de Chambolle-Pock fonctionne très bien sur le problème d'imagerie 2D, et nous l'avons donc transposé à notre étude 3D. Malheureusement, les résultats s'en sont trouvés moins concluant. Nous voyons plusieurs raisons à ce problème : premièrement, le bruit considéré dans la simulation était un bruit purement gaussien pour lequel le modèle est tout-à-fait adapté. Dans notre application, nous avons considéré des bruits de Poisson couplés à des bruits gaussiens, mais avons traité ces bruits comme des bruits purement gaussiens. Bien que le théorème de la limite centrale nous indique qu'une loi Poisson tend en loi vers une gaussienne quand sa variance tend vers l'infini, et que nous travaillons avec un très grand nombre de photons, l'erreur commise dans cette approximation n'est peut-être pas négligeable. Par ailleurs, le modèle de la minimisation de gradient pour diminuer le bruit n'est pas toujours très pertinent, car il pénalise des droites allant fortement vers la profondeur de l'image, ce qui n'est pas forcément contraire à une image réelle. Une idée d'amélioration serait peut-être de rajouter dans la fonction coût la norme de la hessienne, ce qui pénaliserait les oscillations mais pas les droites partant dans la profondeur de l'image. Nous n'avons pas eu le temps de tester cette méthode, mais il serait intéressant de voir si elle permet d'améliorer les résultats...

Apports personnels du projet

Nous avons trouvé ce projet très intéressant car il est à l'interface de plusieurs disciplines, à savoir l'optique et le traitement d'image via l'optimisation. Cela nous a permis d'implémenter des algorithmes très performants et récents d'optimisation convexe. En outre, c'était la première fois que nous travaillions sur une publication scientifique. Par ailleurs, grâce à ce projet, nous avons appris à communiquer avec un chercheur venant d'une discipline différente de la notre, ce qui n'a pas toujours été évident.

Nous remercions sincèrement Jacques Isbert et Pierre Weiss de nous avoir encadrés sur ce projet.

Bibliographie

[1] Jacques Isbert *: La Télémétrie Laser et l'Imagerie Laser 3D*, Cours ISAE.

[2] Antonin Chambolle, Thomas Pock *: A First-Order Primal-Dual Algorithm for Convex Problems with Applications to Imaging* , Publication 2010

[3] Base de données d'images 3D de l'INRIA : `http://www-roc.inria.fr/gamma/download/Bib`

[4] Zoï Vahlas, Benoit Paupy, Nathalie Victorin, Louis Lapointe *: L'imagerie Active Laser 3D*,

Annexe A

Fontions radiométrie

```
function R=Rmax(V,P_laser,rho,T_R,T_T,dr,SNR,thetaS,NEP,x,z)
gamma=x.*((3/V).^z);
ratio=(P_laser.*(rho./pi).*T_R.*T_T.*(4.*dr).*cos(thetaS))/NEP;
epsilon=10^-10;
R=200;
erreur=1000;
Rprec=0;
while (erreur> epsilon)
  if Rprec==0
     Rprec=R;
  else
     Rprec=[Rprec;R];
  end
  f=SNR-ratio.*exp(-2.*gamma.*R.*10^-3)/(R^2);
  fprim=2.*ratio.*exp(-2.*gamma.*R.*10^-3).*(gamma.*R.*10^-3+1)/(R^3);
  R=R-(f/fprim);
  erreur=abs(R-Rprec)
end
```

Ce code calcule la distance maximal pour une cible résolue. POur une cible non résolue, le code est sensiblement le même, seules changent la fonction f et sa dérivée

Annexe B

Fontions analyse temporelle + simulateur

```
 function Pdetect= simulation1prise (R,dmin,dmax,cosVerticaleNormale)
global SigmaW Et Visibilite rho T_R T_T surface_reception thetaS gamma ...
    dr  dt c
Tmin=2*dmin/c;
Tmax=2*dmax/c;
k=[0:floor((Tmax-Tmin)/dt)];
[t]=gaussienne(Et,SigmaW);

% Puissance détectée à chaque prise de vue dans un cas "idéal",
% c'est-à-dire si l'on ne considère pas les pertes d'énergie
t1=t+2*R/c;
Pk1= Pt(k,t1,dmin,Et,SigmaW,R);
% Puissance detectée réelle (en tenant compte des atténuations)
Pdetect=Pdetectee(Pk1,rho,T_R,T_T,dr,R,gamma,thetaS)*cosVerticaleNormale;
```

```
#include <math.h>
#include "mex.h"
void mexFunction( int nlhs, mxArray *plhs[],
  int nrhs, const mxArray*prhs[] )
{
    int n,i;
    double Tmin,dmin,Et,SigmaW,R;
    double *tk,*k,*t;
    double *P2;

    /* Check for proper number of arguments */
    if (nrhs != 6) {
mexErrMsgTxt("Six input arguments required.");
    } else if (nlhs > 1) {
mexErrMsgTxt("Too many output arguments.");
    }

    /*Assigns variables*/
    k=prhs[0];
    t=prhs[1];
    dmin=(double*)prhs[2];
    Et=(double*)prhs[3];
    SigmaW=(double*)prhs[4];
    R=(double*)prhs[5];

    /*alloc space*/
    n = mxGetDimensions(prhs[0]);
```

```
    tk=mxMalloc(n*sizeof(double));
    plhs[0] = mxCreateDoubleMatrix(k, 1, mxREAL);
    P2=mxGetPr(plhs[0]);

    /*code*/
    Tmin=2*(*prhs[2])/(3*1e8);
    for (i=0;i<n;++i) tk[i]=Tmin+k[i]*SigmaW/10.0;
    for (i=0;i<n;++i) {
        P2[i]=0;
    }
    mxFree(tk);
    return;
}
```

```
function [XI,YI,ImageCaptee,hf]=  simulateur (X,Y,Z,nbPixels,sigmaXY)
IntervalleMesure=100;
global SigmaW Et Visibilite rho T_R T_T surface_reception thetaS gamma ...
dr  dt c
SigmaW=5;
Et=0.1;
Visibilite=23;   %en km
rho=0.2;
T_R=0.1;
T_T=0.1;
surface_reception=0.005;  % 0.3<R<0.9
thetaS=0;
gamma=0.503*(3/Visibilite)^1.08;
dr= sqrt(surface_reception /pi );
c=3*10^8 ; %célérité de la lumière
SigmaW=SigmaW*10^-9;
dt=SigmaW/10;
L=0;
[XI , YI ]=meshgrid(linspace(X(1,1)+L/5,X(end,end)-L/5,nbPixels),...
    linspace(Y(1,1)+L/5,Y(end,end)-L/5,nbPixels));
ZI=interp2(X,Y,Z,XI,YI);
costheta=angle(XI,YI,ZI);
Distance=(max(Z(:))+min(Z(:)))/2;
nbPrisesParPixel=length(simulation1prise(Distance,Distance-...
    IntervalleMesure/2,Distance+IntervalleMesure/2,1));
Signal=sparse(size(ZI,1)*nbPrisesParPixel,size(ZI,2));
nbLignes=size(ZI,1);
for i=1:size(ZI,1)
    for j=1:size(ZI,2)
        kk=i:nbLignes:nbLignes*nbPrisesParPixel;
        Signal(kk,j)= (simulation1prise(ZI(i,j),Distance-...
            IntervalleMesure/2,Distance+IntervalleMesure/2,costheta(i,j)));
    end
end
[X3d,Y3d]=meshgrid( XI(1,:) , YI(:,1));
%%%% Convolution de l'image
[XXX , YYY ] = meshgrid(linspace(-2,2,size(X3d,1)),...
    linspace(-2,2,size(Y3d,1)));
h=exp(-(XXX.^2+YYY.^2)/(2*sigmaXY^2)) ;
h=h/sum(h(:));
h=fftshift(h);
hf=fft2(h);
u0=sparse(size(Signal,1),size(Signal,2));
disp('début convolution gros con');
for kk=1:nbPrisesParPixel
```

```
    u0(((kk-1)*(nbLignes)+1):((kk)*(nbLignes)),1:nbLignes)=...
        ifft2(fft2(full(Signal(((kk-1)*(nbLignes)+1):...
        ((kk)*(nbLignes)),1:nbLignes))).*hf);
end
u0(u0<0)=0;
disp('fin convolution');
%%%  AJOUT DU BRUIT   %%%%%
A_B=0.01;
eta=1;
lambda=1.06;
Ek=sparse(size(Signal,1),size(Signal,2));
lambda = lambda*10^-6;
dlambda=5;        %largeur de la cible (nm)
Sib=500 ;         %Irradiance par unité de longueur(W/m^2.µm)
TempCircuit=300;  %(en Kelvin)
Capacite = 4 ;    %en pF
Idark=4 ;         %en nA
h_planck=6.626068 * 10^-34 ;%constante de planck
dlambda=dlambda*10^-9;
Capacite=Capacite*10^-12;
Idark=Idark*10^-9;
k_Boltzmann = 1.381*10^-23 ;
qe=1.60217646 * 10^-19 ;
nu=c/lambda;
Ek=u0*dt*lambda/(h_planck*c); %Conversion puissance->nb de photons
ENsignal=Ek*eta;  %Conversion nb de photons -> nb d'électrons
ENdark=Idark*dt/(qe);
ENb=(Sib*dlambda*A_B*rho*T_R*T_T*dr^2*dt*eta)/(4*Distance^2*h_planck*nu)...
    +ENdark;
Qn2=k_Boltzmann*TempCircuit*Capacite/qe^2;
[II1,II2]=find( ENsignal>0) ;
nbElectronsAttendus=sparse(size(Signal,1),size(Signal,2));
for ii= 1 :length(II1)
nbElectronsAttendus(II1(ii),II2(ii))=...
    poissrnd(Ek(II1(ii),II2(ii)));
end
disp('fin bruit Poisson');
nbElectronsEclairementObscurite=poissrnd(ENb);
nbElectrons=sparse(size(Signal,1),size(Signal,2));
nbElectronsTot=sparse(size(Signal,1),size(Signal,2));
nbElectrons(nbElectronsAttendus>0)=nbElectronsAttendus(...
    nbElectronsAttendus>0)  +nbElectronsEclairementObscurite;
nbElectronsTot(nbElectrons>0)=nbElectrons(nbElectrons>0)+...
    floor(randn(size(nbElectrons((nbElectronsAttendus>0))*sqrt(Qn2))));
ImageCaptee =zeros(nbPixels,nbPixels);
for i=1:nbPixels
    for j=1:nbPixels
        tmp=nbElectronsTot( i:nbPixels:nbPixels*nbPrisesParPixel , j);
        [tp2 , ImageCaptee(i,j)]=max(tmp);
    end
disp(i)    ;
end
ImageCaptee= ImageCaptee*dt*c/2+Distance-IntervalleMesure/2;
clear SigmaW Et Visibilite rho T_R T_T surface_reception thetaS gamma ...
dr  dt c
```

Annexe C

Traitement d'image

```
%function  xx = pock (x0,lambda,D,H)
%This function computes min_x ||nabla x||- (lamdba/2)||D(Hx-x0)||
function  [xx CF]= pock (x0,lambda,D,Hf,tau,sigma,theta,nitMax)
xn=x0;
yn=0;
xBarre=xn;
nit=0;
CF=zeros(1,nitMax);
while ( nit<=nitMax )
    nit=nit+1;
    ynp1=proxFetoile (sigma,lambda,yn+sigma*A(xBarre,Hf,D),x0);
    xnp1=xn-tau*Aetoile(ynp1,Hf,D);
    xBarre=xnp1+theta*(xnp1-xn);
    dx =drondx(xnp1);
    dy =drondy(xnp1);
    Hx= ifft2( Hf.*fft2(xnp1));
    valF=sum(sum((sqrt(dx.^2+dy.^2))));
    valH= sum(sum(D.*(Hx-x0).^2));
    %% cost function
    CF(nit)=valF+lambda/2*valH;
    xn=xnp1;
    yn=ynp1;
end
xx=xn;

function d=drondy(im)
d=zeros(size(im));
d(1:end-1,:)=im(2:end,:)-im(1:end-1,:);
function d=drondyT(im)
d=zeros(size(im));
d(2:end-1,:)=im(1:end-2,:)-im(2:end-1,:);
d(1,:)=-im(1,:);
d(end,:)=im(end-1,:);
function d=drondx(im)
d=zeros(size(im));
d(:,1:end-1)=im(:,2:end)-im(:,1:end-1);
function d=drondxT(im)
d=zeros(size(im));
d(:,2:end-1)=im(:,1:end-2)-im(:,2:end-1);
d(:,1)=-im(:,1);
d(:,end)=im(:,end-1);
function AA= A(x,Hf,D)
DHx= D.*ifft2( Hf.*fft2(x));
gradx= [ drondx(x);drondy(x) ];
AA=[gradx; DHx];
function AA= Aetoile(x,Hf,D)
```

```
n=size(x,1)/3;
m=size(x,2);
ii2=1:m;
ii1=1:n;
dx=x(ii1,ii2);
dy=x(n+ii1,ii2);
Hx=x(2*n+ii1,ii2);
AA=drondxT(dx)+drondyT(dy)+ ifft2(conj(Hf).*fft2(D.*Hx));
function yy = proxFetoile (sigma,lambda,y,x0)
yy=y;
ii1=1:(size(y,1)/3);
ii2=1:size(y,2);
norme=sqrt(y(ii1,ii2).^2 +  y(size(y,1)/3+ii1,ii2).^2);
norme(norme==0)=0.5;
yy(ii1,ii2) = (norme<1).*y(ii1,ii2) +(norme>=1).*y(ii1,ii2)./norme;
yy(size(y,1)/3+ii1,ii2) = (norme<1).*y(size(y,1)/3+ii1,ii2) +(norme>=1).*...
    y(size(y,1)/3+ii1,ii2)./norme;
yy(2*size(y,1)/3+ii1,ii2)=(y(2*size(y,1)/3+ii1,ii2)-sigma*x0)/...
    (1+sigma/lambda);
```