MINISTÈRE
DE L'ÉDUCATION
NATIONALE,
DE LA JEUNESSE
ET DES SPORTS
Liberté
Égalité
Fraternité

EAE SIN 2

SESSION 2021

AGREGATION
CONCOURS EXTERNE

Section : SCIENCES INDUSTRIELLES DE L'INGÉNIEUR

Option : SCIENCES INDUSTRIELLES DE L'INGÉNIEUR ET INGÉNIERIE INFORMATIQUE

MODÉLISATION D'UN SYSTÈME, D'UN PROCÉDÉ OU D'UNE ORGANISATION

Durée : 6 heures

Calculatrice électronique de poche - y compris calculatrice programmable, alphanumérique ou à écran graphique – à fonctionnement autonome, non imprimante, autorisée conformément à la circulaire n° 99-186 du 16 novembre 1999.

L'usage de tout ouvrage de référence, de tout dictionnaire et de tout autre matériel électronique est rigoureusement interdit.

Si vous repérez ce qui vous semble être une erreur d'énoncé, vous devez le signaler très lisiblement sur votre copie, en proposer la correction et poursuivre l'épreuve en conséquence. De même, si cela vous conduit à formuler une ou plusieurs hypothèses, vous devez la (ou les) mentionner explicitement.

NB : Conformément au principe d'anonymat, votre copie ne doit comporter aucun signe distinctif, tel que nom, signature, origine, etc. Si le travail qui vous est demandé consiste notamment en la rédaction d'un projet ou d'une note, vous devrez impérativement vous abstenir de la signer ou de l'identifier.

Tournez la page S.V.P.

INFORMATION AUX CANDIDATS

Vous trouverez ci-après les codes nécessaires vous permettant de compléter les rubriques figurant en en-tête de votre copie.

Ces codes doivent être reportés sur chacune des copies que vous remettrez.

Concours	Section/option	Epreuve	Matière
EAE	1417A	102	2680

Sommaire

Ce document se décompose en trois parties :

- Le sujet (pages 1 à 24)
- Les documents techniques DT1 à DT4 (pages 25 à 32)
- Les documents réponses DR1 à DR6 (pages 33 à 43)

IMPRIMERIE NATIONALE – 21 0096 – D'après documents fournis

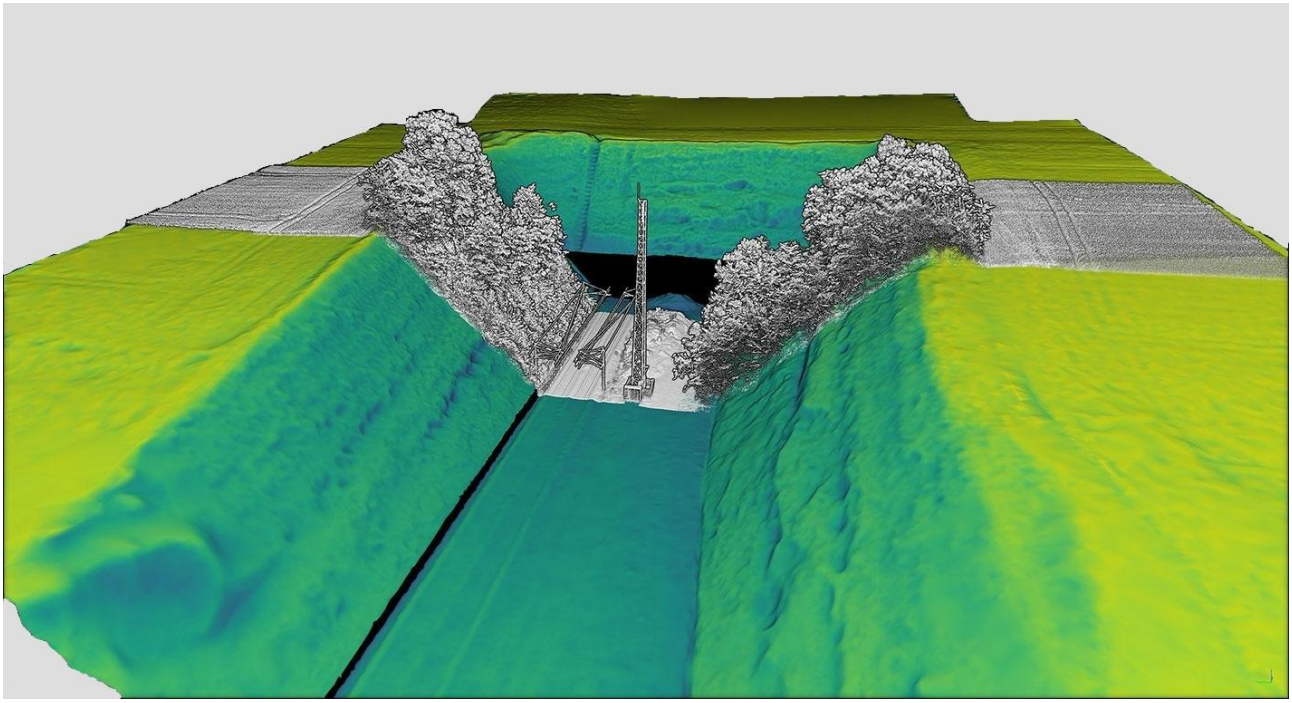
Inspection de parois rocheuses pour la sûreté du réseau ferroviaire

Présentation

Avec plus de 50 000 km de voies, 1 600 tunnels, 32 305 ponts-rails, 11 700 ponts-routes, et 1 350 passerelles pour piétons, le réseau ferré français pose un défi permanent pour les équipes chargées de leur maintenance.

Altametris est une filiale de SNCF Réseau spécialisée dans les services d'aide à la gestion des actifs industriels en réalisant des collectes de données 3D, des traitements et des analyses de ces dernières.

Ces traitements aboutissent à une représentation 3D relativement fidèle de l'environnement étudié. Les experts ont ainsi accès à des informations précises et facilement exploitables informatiquement.



La collecte de données 3D peut être réalisée suivant 3 techniques :

- la lasergrammétrie terrestre ;
- la lasergrammétrie aéroportée ;
- la photogrammétrie.

Lasergrammétrie terrestre :

Afin de modéliser numériquement les parois rocheuses des voies ferrées, Altametris utilise un scanner-Lidar (figure 1) fixe dont l'orientation et la position dans l'espace sont connues avec précision (centrale inertielle et GPS). Le balayage de l'espace est réalisé suivant deux axes de rotation.

Tournez la page S.V.P.

Le modèle 3D texturé (figure 1) d'une paroi rocheuse illustre le résultat du traitement numérique qui associe les données collectées par le scanner-Lidar et les données d'une image issue d'une caméra numérique intégrée.

Figure n°1 : Scanner-Lidar 2 axes de rotation et le modèle 3D d'une paroi (1 770 102 points 3D, résolution en profondeur 2 cm)

Lasergrammétrie aéroportée:

Cette technique remplace la lasergrammétrie terrestre pour augmenter la vitesse de numérisation et pour numériser des lieux difficiles d'accès. Dans ce contexte, Altametris utilise un drone octocoptère de 23 kg (RiCOPTER) transportant un scanner-Lidar dont le balayage est opéré suivant un seul axe de rotation. La direction du vol décrit le deuxième axe de balayage.

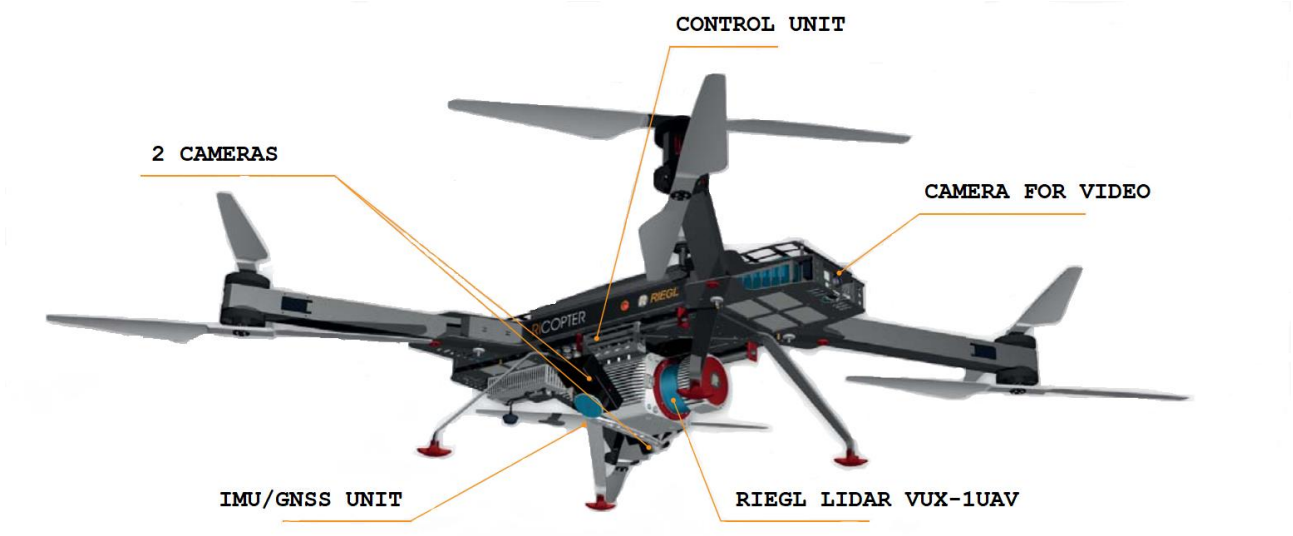


Figure n°2 : Drone RiCOPTER et son scanner-Lidar

Photogrammétrie :

Cette solution n'utilise pas de scanner-Lidar mais un appareil photo numérique dont la position et l'orientation sont connues également.
Dans la deuxième partie de l'image ci-après (figure 3), sont représentés, le modèle 3D et les différentes positions/orientations de l'appareil photo. La reconstruction 3D utilise des techniques de corrélation d'images et de triangulation.

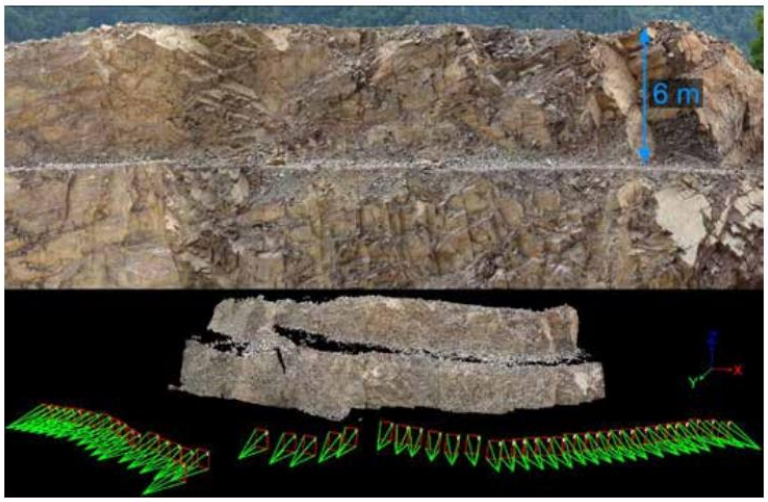


Figure n°3 : Modèle 3D d'une paroi rocheuse par photogrammétrie (30 paires d'images)

Logiciel d'inspection de parois rocheuses :

C'est un logiciel qui assure le traitement des nuages de points 3D issus soit de la lasergrammétrie, soit de la photogrammétrie. Il permet à un utilisateur « expert » de déterminer les discontinuités d'une paroi rocheuse dans un objectif de maintenance préventive permettant ainsi d'identifier les zones à risque et d'anticiper les chutes de rochers.

Depuis l'interface du logiciel, l'expert a accès aux fonctionnalités suivantes :

- mesure des distances par déplacement de la souris ;
- visualisation de la carte des profondeurs en 2D et 3D ;
- extraction des familles de discontinuités ;
- classification des plans de discontinuités.

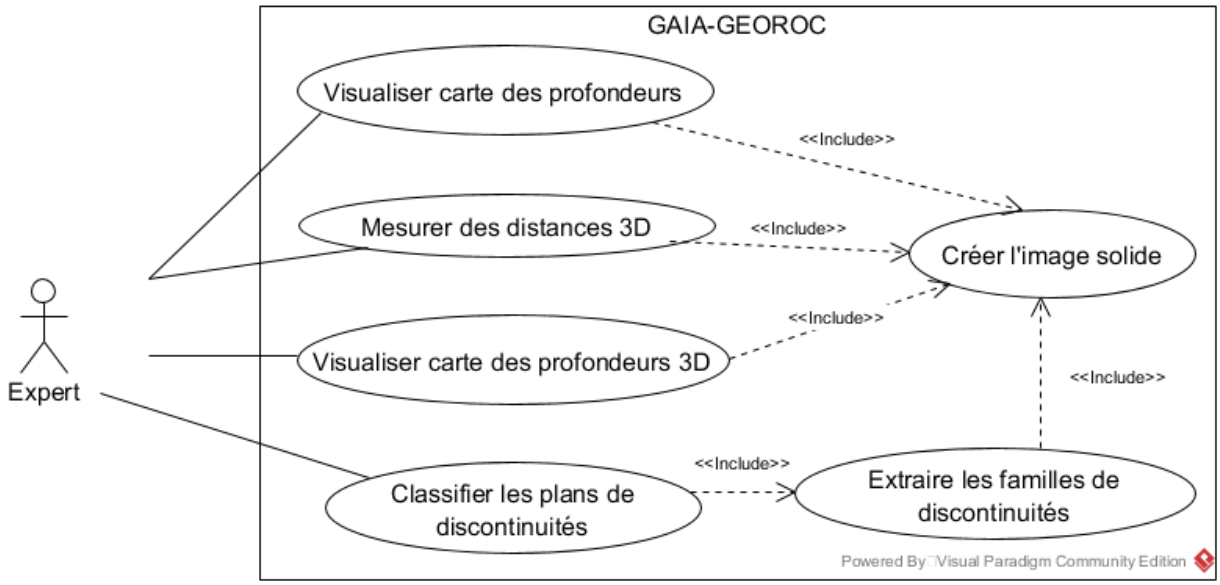


Figure n°4 : Cas d'utilisation partiel du logiciel d'inspection de parois rocheuses

Ces fonctionnalités reposent sur la construction préalable de l'image solide (figure 5). C'est une structure de données qui est fabriquée à partir d'un nuage de points 3D et d'une image numérique.

Tournez la page S.V.P.

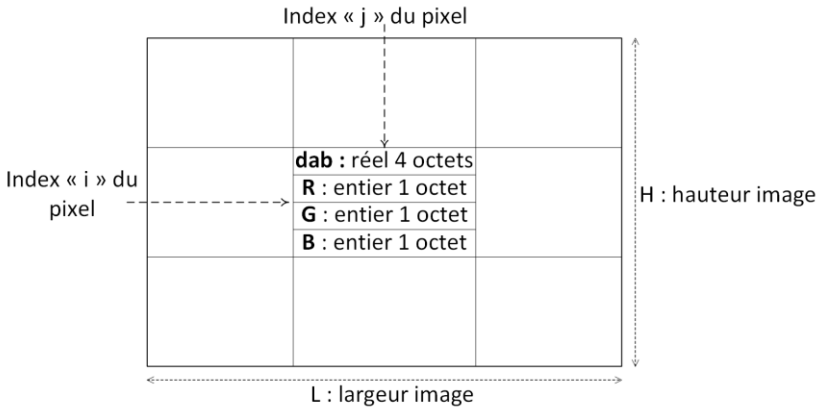
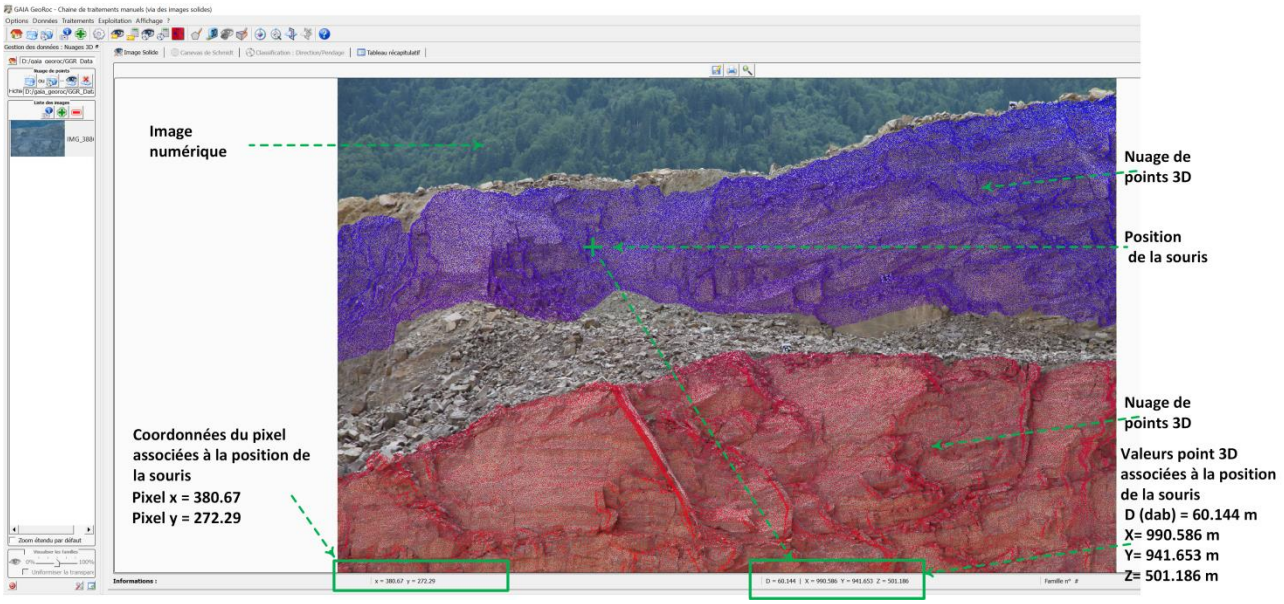


Figure n°5 : Principe de l'image solide (avec dab, distance du point 3D)

Ce logiciel est composé de deux couches :

- une couche C++ associée à l'interface homme-machine et aux entrées/sorties du type fichier texte et image ;
- une couche de scripts réalisés dans le langage « R » orienté manipulation de données pour le traitement du nuage de points et des images.



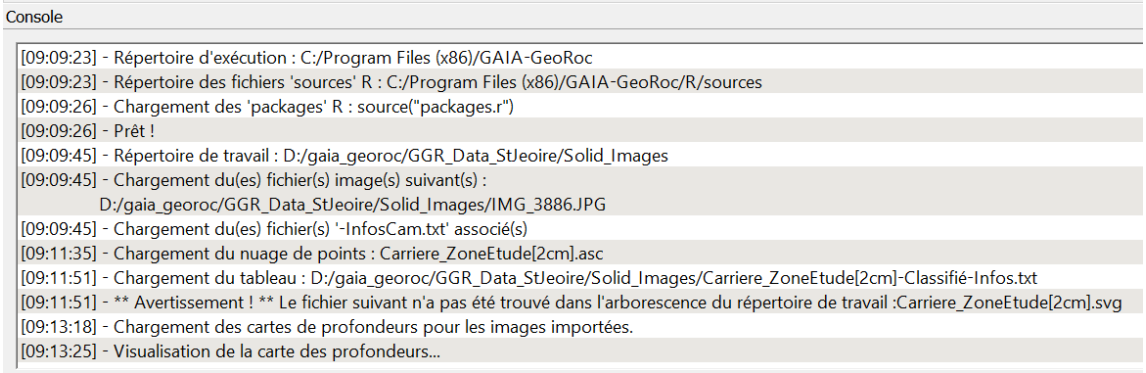


Figure n°6 : Nuage de points 3D projetés dans l'image « IMG_3886.jpg » et console d'exécution des tâches logicielles

L'interaction entre les différentes couches est représentée par le diagramme d'activité (figure 7) dans lequel les actions utilisateurs (boutons) sont définies dans le couloir de gauche. Les données manipulées (fichiers de mesures, images) apparaissent comme des flots de contrôle et des flots d'objets.

Les fichiers suivants utilisés dans ce diagramme sont décrits dans le DT2 :

- « img_3886_infosCam.txt » : informations caméra ;
- « Carriere_ZoneEtude[2cm].asc » : nuage de points 3D.

Ce diagramme décrit les cas d'utilisation « Créer l'image solide », « Visualiser carte des profondeurs 2D » et « Visualiser carte des profondeurs 3D ».

Après importation d'une image de la paroi, des données du nuage de points 3D et des données de la caméra, l'utilisateur déclenche le calcul de l'image solide décrit dans le couloir « Script R ».
Les fichiers « R » sont associés aux différents nœuds et leur exécution permet de construire l'image solide. Les programmes « R » ne seront pas étudiés dans ce sujet.

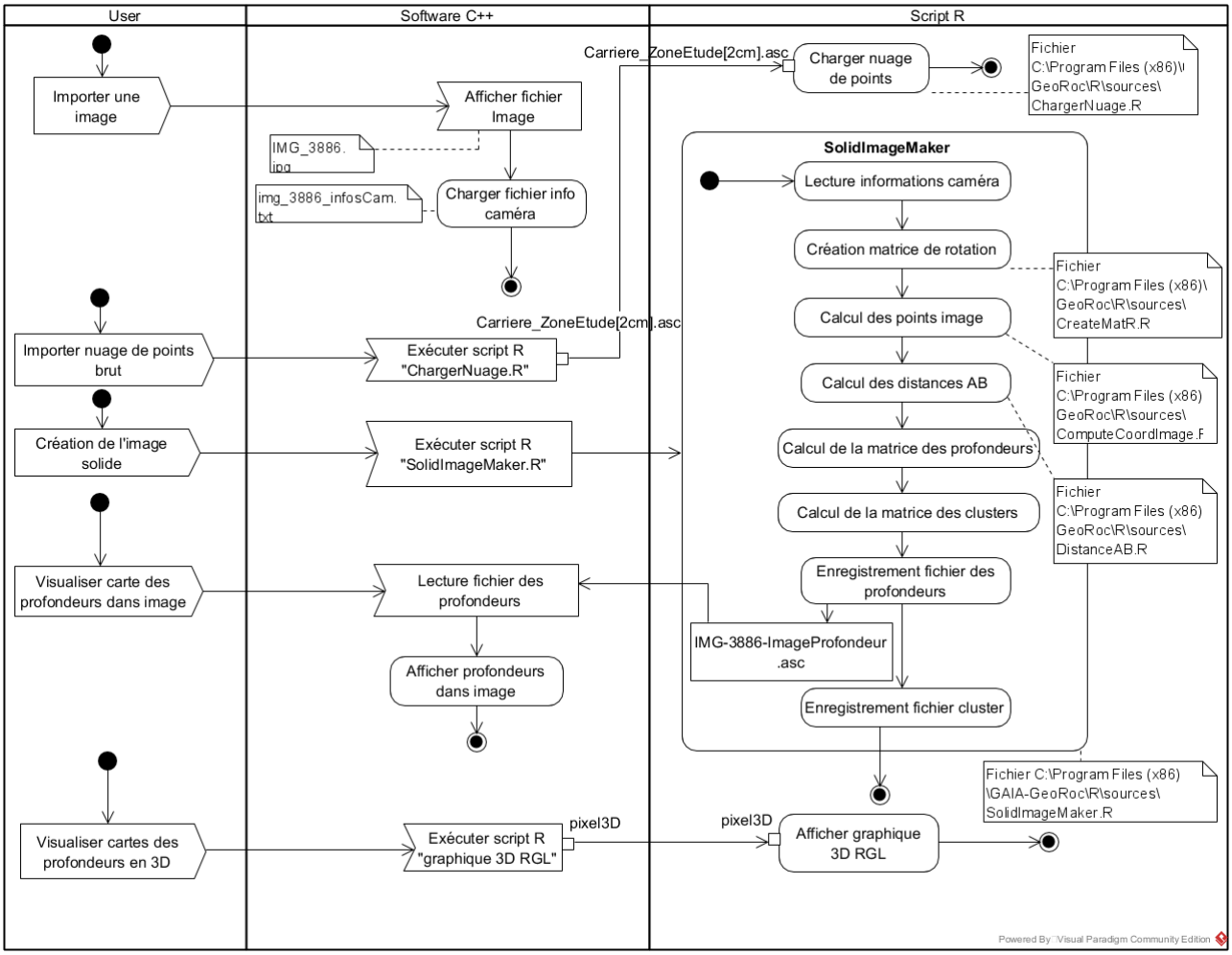


Figure n°7 : Inspection de parois rocheuses : interaction entre couches logicielles et utilisateur

Les réflexions suscitées par les questions de ce sujet doivent permettre de caractériser les données manipulées et les traitements numériques mis en œuvre dans les techniques de lasergrammétrie et de photogrammétrie afin d'évaluer ces dernières dans le contexte de surveillance des parois rocheuses bordant le réseau ferroviaire.

Tournez la page S.V.P.

Ce sujet est constitué de 4 parties indépendantes :

- partie 1 : acquisition 3D par scanner-Lidar ;
- partie 2 : calcul de l'image solide par lasergrammétrie ;
- partie 3 : solution de photogrammétrie ;
- partie 4 : analyse des images par segmentation.

Partie 1. Acquisition 3D par scanner-Lidar

Objectifs :
Caractériser le nuage de points mesuré (résolution spatiale, densité, quantité d'informations) par un scanner-Lidar embarqué sur un drone. Disposer d'un modèle de programmation orienté objet du scanner-Lidar et de l'acquisition aéroportée.

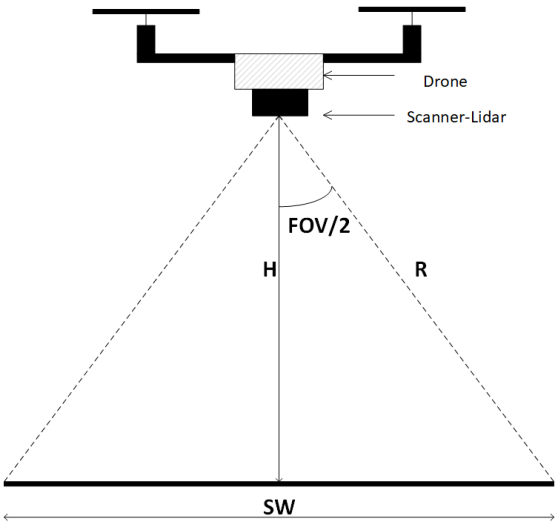
L'acquisition du nuage de points par un scanner-Lidar terrestre nécessite de déplacer ce dernier sur des positions référencées permettant ainsi de numériser la surface totale de la paroi rocheuse. Afin de supprimer le chevauchement de mesures, des cibles identifient différentes zones de la paroi puis un algorithme de consolidation est utilisé.

L'utilisation d'un scanner-Lidar aéroporté (DT1) pourrait simplifier l'acquisition du nuage de points. En effet le parcours d'un drone embarquant un scanner-Lidar parallèlement à la paroi peut produire dans le cas idéal un nuage de points uniforme semblable à une matrice de points.

Le drone RiCOPTER muni d'un scanner-Lidar assure l'acquisition de nuages de points 3D aériens à très haute densité. Le DT1 présente le principe de fonctionnement ainsi que les caractéristiques principales d'un scanner-Lidar à un seul axe de balayage.

Q 1. En considérant la réponse à une impulsion Laser émise par le Lidar, déterminer les distances qui séparent le drone de la caténaire, de la végétation et du sol.

Le déplacement horizontal du drone réalise un balayage perpendiculaire au balayage suivant la direction portée par le segment SW (Swath Width : largeur de bande), décrivant ainsi une matrice d'empreintes (« footprints »). Par la suite, l'hypothèse d'une répartition uniforme des empreintes est considérée.



Q 2. Déterminer avec les éléments du croquis ci-avant, l'expression de la largeur de bande SW en fonction de la hauteur H et de l'angle de champ de vision FOV. Déterminer la surface balayée A à vitesse v et à temps de vol ΔT donnés.

Q 3. Exprimer le nombre de lignes $\mathrm{N_l}$ scannées en fonction du temps de vol ΔT et de la fréquence de balayage SR. Exprimer la vitesse angulaire Ω en fonction de la fréquence de balayage SR et de l'angle de champ de vision FOV. Donner l'expression du pas angulaire $\Delta\theta$ (angle décrit par le faisceau durant la période d'un pulse).

Q 4. Exprimer le nombre d'empreintes $\mathrm{N_f}$ lors d'un unique balayage. En déduire la densité de points $\mathrm{d_p}$ (nombre d'empreintes par $\mathrm{m^2}$) et donner les expressions de l'espace inter-empreintes a et de l'espace interligne b dont la répartition est considérée uniforme.

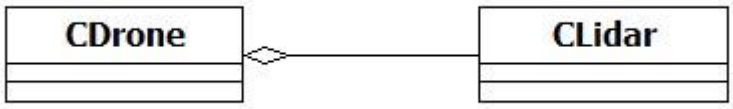
La paroi rocheuse à numériser est considérée comme verticale et plate. Ces dimensions sont de 80 m de longueur et de 16 m de hauteur.

Q 5. En considérant un balayage à fréquence donnée ($SR = 200\ Hz$) suivant des lignes parallèles à la hauteur de la paroi, un déplacement horizontal du drone et une distance $D = 30\ m$ du drone par rapport à la paroi, déterminer l'angle de champ de vision, la fréquence PRR et la vitesse du drone afin d'obtenir une résolution spatiale de 2 cm.

Le DT1 présente également le format des fichiers stockés dans un scanner-Lidar.

Q 6. Calculer le volume de données en octets pour la campagne de mesures décrite à la question précédente pour un temps de vol $\Delta T = 20\ s$.

Les calculs précédents ont été implémentés dans le langage Python selon la structure logicielle ci-dessous dans laquelle deux classes collaborent.



Q 7. Compléter le corps des méthodes de la classe CLidar (DR1) décrites ci-dessous :

Méthode	Variable associée
`getOmega`	Vitesse angulaire Ω
`getDeltaTheta`	Pas angulaire $\Delta\theta$
`getNf`	Nombre d'empreintes $\mathrm{N_f}$

Q 8. Compléter le corps des méthodes de la classe CDrone (DR1) décrites ci-dessous :

Méthode	Variable associée
`getSW`	Largeur de bande SW
`getA`	Surface balayée A
`getNl`	Nombre de lignes $\mathrm{N_l}$
`getDensity`	Densité de points $\mathrm{d_p}$
`getWidthSpace`	Espace inter-empreintes a
`getLengthSpace`	Espace inter-empreintes b

Tournez la page S.V.P.

Q 9. Écrire le corps de la méthode **`listeFootPrints`** de la classe CDrone permettant de remplir la liste **`self.__lPoints`** de l'ensemble des points associés à une campagne de mesures.

Le nuage de points ainsi acquis est appelé « nuage de points dense ».

Q 10. Justifier ce terme et conclure sur les traitements post-acquisition à réaliser dans le cas d'une répartition non uniforme des empreintes.

Partie 2. Calcul de l'image solide par lasergrammétrie

Objectifs :
Modéliser le lien entre les coordonnées des points 3D et les pixels de l'image numérique d'une paroi rocheuse.
Réaliser et caractériser le programme Python de calcul de l'image solide à partir du nuage de points 3D et des pixels de l'image.

La mise en relation géométrique des points image et des points 3D est illustrée par la figure ci-dessous qui s'appuie sur le modèle « sténopé » ou « pinhole camera ». C'est un modèle linéaire qui considère les hypothèses suivantes :

- tous les rayons lumineux passent uniquement par le centre optique (O) sans distorsion ;
- le plan de l'image non inversée est localisé à la distance focale f par rapport au centre optique représenté par le centre du repère caméra (O).

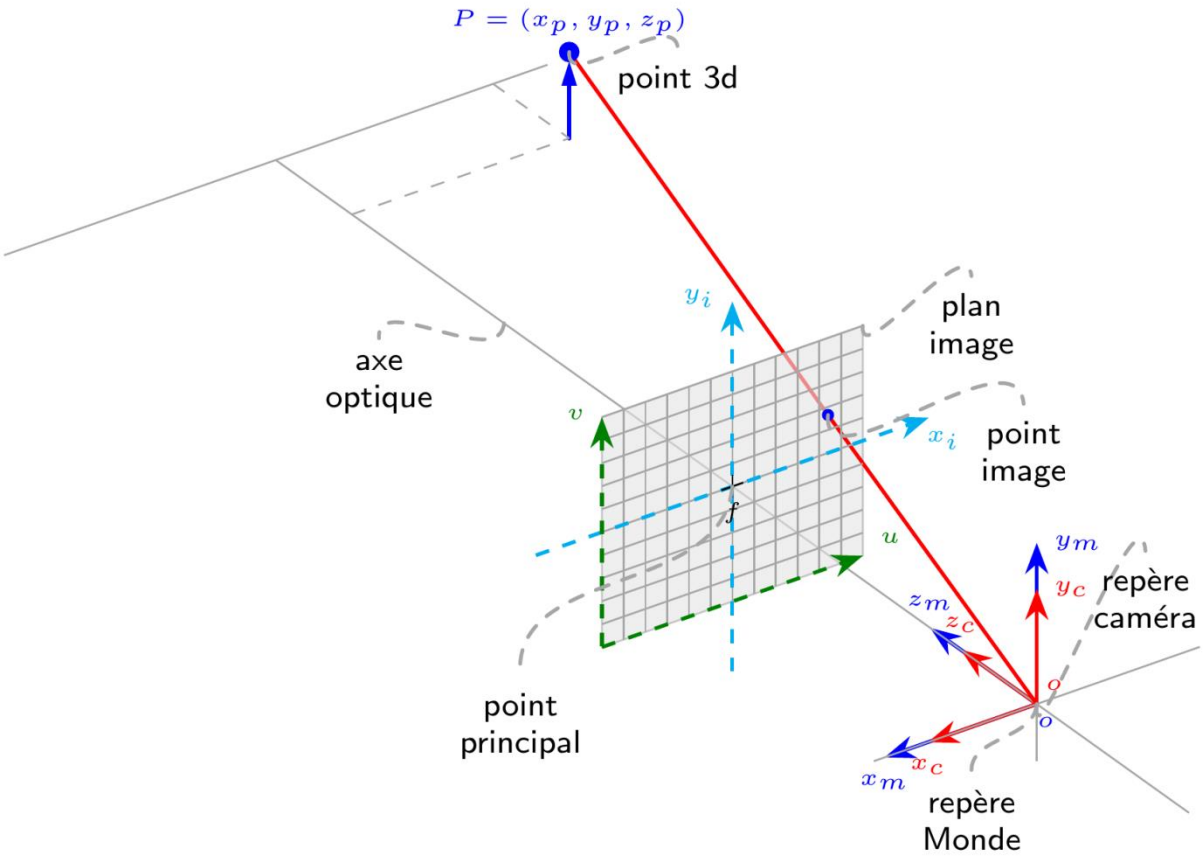


Figure n°8 : Construction de l'image d'une caméra

Pour les questions suivantes, le repère « caméra » (C) et le repère « Monde » (M) sont confondus.

Q 11. Réaliser les croquis illustrant la projection de la figure ci-dessus respectivement dans le plan horizontal (O, x_c, z_c) et dans le plan vertical (O, y_c, z_c). Donner les expressions des coordonnées (x_i, y_i) du point image en fonction de x_p, y_p, z_p et f.

Afin d'effectuer les transformations géométriques nécessaires, les coordonnées homogènes sont utilisées.
Un point 2D (x, y) est alors représenté par un vecteur de 3 composantes où les deux premières sont définies à un facteur près égal à la troisième composante :

$\vec{v} = (x, y, 1)^T \cong (zx, zy, z)^T = (x', y', z)^T$, avec $z \neq 0$

De la même manière, un point 3D (x, y, z) devient un vecteur à 4 composantes :

$\vec{v} = (x, y, z, 1)^T \cong (wx, wy, wz, w)^T = (x', y', z', w)^T$, avec $w \neq 0$

Par conséquent, la projection du point (P) de la figure précédente dans le plan image est définie par la relation suivante :

$$\begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} \cong \begin{bmatrix} x_i' \\ y_i' \\ z_i \end{bmatrix} = [F] \begin{bmatrix} x_p \\ y_p \\ z_p \\ 1 \end{bmatrix} = \begin{bmatrix} -f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_p \\ y_p \\ z_p \\ 1 \end{bmatrix}$$

Le vecteur $(x_p, y_p, z_p)^T$ en coordonnées homogènes donne $(x_p, y_p, z_p, 1)^T$ dont les valeurs des composantes restent inchangées.
En revanche, les vecteurs $(x_i, y_i, 1)^T$ et $(x_i', y_i', z_i)^T$ sont égaux à un facteur près.

Q 12. Donner les expressions de x_i', y_i' et de z_i et vérifier les résultats de la question précédente.

Afin de prendre en compte les déplacements de la caméra, il est nécessaire d'exprimer les nouvelles coordonnées du point 3D lorsque la caméra effectue des déplacements relatifs au repère « Monde ».

Dans ce cas, le repère « caméra » et le repère « Monde » ne sont plus confondus.

La translation de la caméra par rapport au repère « Monde » est obtenue par translation du centre optique dans le repère « Monde » par un vecteur $\vec{d} = (d_x, d_y, d_z)^T$. Ainsi, du point de vue du repère « caméra », les coordonnées du point 3D représentées par le vecteur $(x_p, y_p, z_p, 1)^T$ subissent la transformation géométrique suivante :

$$\begin{bmatrix} x_p' \\ y_p' \\ z_p' \\ 1 \end{bmatrix}_C = T \begin{bmatrix} x_p \\ y_p \\ z_p \\ 1 \end{bmatrix}_M = \begin{bmatrix} I_3 & d \\ 0_3^T & 1 \end{bmatrix} \begin{bmatrix} x_p \\ y_p \\ z_p \\ 1 \end{bmatrix}_M = \begin{bmatrix} 1 & 0 & 0 & -d_x \\ 0 & 1 & 0 & -d_y \\ 0 & 0 & 1 & -d_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_p \\ y_p \\ z_p \\ 1 \end{bmatrix}_M$$

Tournez la page S.V.P.

Les rotations de la caméra sont obtenues par des rotations de l'axe optique ayant pour origine le centre optique. Par exemple, une matrice de rotation associée à un axe de la caméra est représentée ci-dessous :

$$\begin{bmatrix} x'_p \\ y'_p \\ z'_p \end{bmatrix}_{\mathrm{C}} = R_\beta \begin{bmatrix} x_p \\ y_p \\ z_p \end{bmatrix}_{\mathrm{M}} = \begin{bmatrix} \cos\beta & 0 & -\sin\beta \\ 0 & 1 & 0 \\ \sin\beta & 0 & \cos\beta \end{bmatrix} \begin{bmatrix} x_p \\ y_p \\ z_p \end{bmatrix}_{\mathrm{M}}$$

Q 13. Justifier par calcul l'expression de la matrice T et l'expression de la matrice R_β.

Les matrices de rotation R_α et R_γ, respectivement associées aux rotations de la caméra autour des axes x_c et z_c sont données ci-après :

$$R_\alpha = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & \sin\alpha \\ 0 & -\sin\alpha & \cos\alpha \end{bmatrix} \qquad R_\gamma = \begin{bmatrix} \cos\gamma & \sin\gamma & 0 \\ -\sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

La combinaison de l'ensemble des rotations permet de définir une matrice de rotation globale :

$$R = R_\gamma R_\beta R_\alpha = \begin{bmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{bmatrix}$$

Ce qui permet d'exprimer les coordonnées homogènes du point image de la manière suivante :

$$\begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} \cong FP \begin{bmatrix} x_p \\ y_p \\ z_p \\ 1 \end{bmatrix} = F \begin{bmatrix} R & -Rd \\ 0_3^T & 1 \end{bmatrix} \begin{bmatrix} x_p \\ y_p \\ z_p \\ 1 \end{bmatrix}$$

Q 14. Justifier par calcul l'expression précédente en explicitant la matrice P nommée matrice de projection en fonction des éléments de la matrice R et du vecteur $\overrightarrow{d}$.

Les dimensions du plan image sont associées à la largeur L_{cmos} et à la hauteur H_{cmos} du capteur permettant l'acquisition des composantes (R, G, B) de chaque point 3D.
Le repère (u, v), dont l'origine est située en bas à gauche du plan image, constitue la référence de l'image. Les unités de u et de v sont en pixels.
Par ailleurs, les résolutions en pixels/m sont définies en fonction de la largeur L et de la hauteur H en pixels de l'image :

$$s_x = \frac{L}{L_{cmos}} \qquad s_y = \frac{H}{H_{cmos}}$$

Q 15. Exprimer le vecteur $(u, v, 1)^T$ comme le produit de deux matrices S, D à déterminer et le vecteur $(x_i, y_i, 1)^T$. Exprimer le vecteur $(u, v, 1)^T$ en fonction du vecteur $(x_p, y_p, z_p, 1)^T$.

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = SD \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$

Q 16. Justifier les expressions exactes des coordonnées du point image données ci-dessous.

$$u = s_x \left[\frac{L_{cmos}}{2} - f \frac{\left(R_{11}(x_p - d_x) + R_{12}(y_p - d_y) + R_{13}(z_p - d_z)\right)}{\left(R_{31}(x_p - d_x) + R_{32}(y_p - d_y) + R_{33}(z_p - d_z)\right)} \right]$$

$$v = s_y \left[\frac{H_{cmos}}{2} + f \frac{\left(R_{21}(x_p - d_x) + R_{22}(y_p - d_y) + R_{23}(z_p - d_z)\right)}{\left(R_{31}(x_p - d_x) + R_{32}(y_p - d_y) + R_{33}(z_p - d_z)\right)} \right]$$

L'image solide (figure 5) permet de composer, dans une unique structure de données, les informations colorimétriques de chaque pixel d'une image (R, G, B) avec la distance dab entre un point 3D et le centre optique de la caméra.

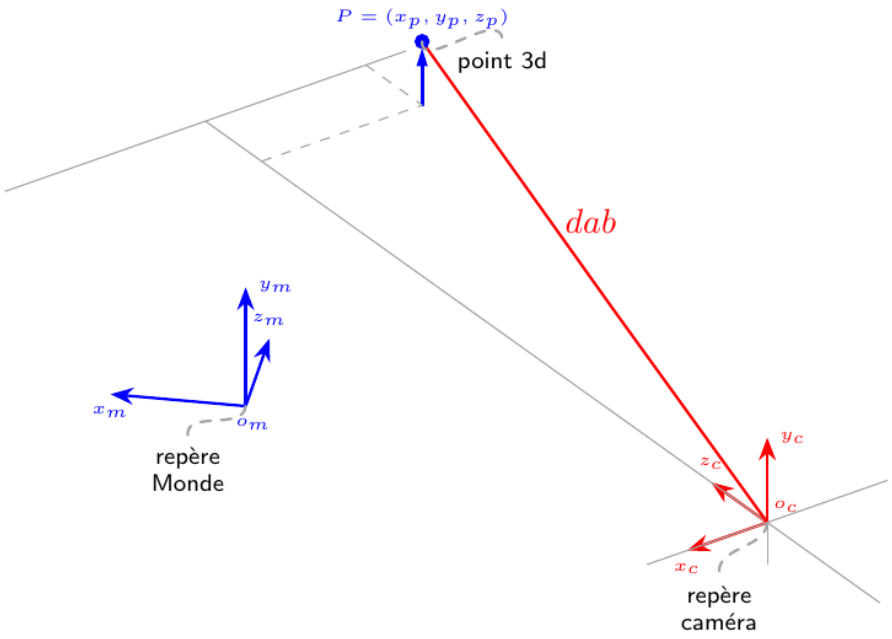


Figure n°9 : Distance dab

Les coordonnées d'un point 3D sont issues de mesure du type scanner-Lidar.

Le DT3 présente un extrait des fonctions de la bibliothèque Numpy et des fonctions de traitement des fichiers en Python.

Le DT2 présente le fichier des informations d'un nuage de points ainsi que le fichier des paramètres de la caméra utilisée pour la capture de l'image associée.

Q 17. Sans utiliser la bibliothèque « csv », compléter la fonction **`read3dPointTextFile`** du programme Python (DR2) permettant de stocker les coordonnées homogènes des points 3d dans la variable **`tabNuage`** et la direction du plan dans la variable **`tabVecNorm`**.

Q 18. Compléter les fonctions **`matRotation`** et **`matProjection`** associées respectivement à la matrice de rotation globale et à la matrice de projection.

Q 19. Déduire du programme principal du DR2, la structure de données permettant de stocker le vecteur $(u', v', w)^T$ de coordonnées homogènes $(u, v, 1)^T$ et justifier son type et ses dimensions.

Tournez la page S.V.P.

Q 20. À partir du programme principal, dresser un tableau permettant de mettre en relation les instructions Python et les traitements mathématiques permettant de calculer le vecteur $(u', v', w)^T$ puis le vecteur $(u, v, 1)^T$.

Instruction Python	Traitement réalisé
…	…

Q 21. Déduire du programme principal, l'expression de la distance dab entre un point image et un point 3d (Figure 9) et expliquer le rôle de la structure conditionnelle imbriquée dans la structure itérative.

En raison de la méthode d'acquisition des points 3D, la liste **`l_pixels_dab`** du programme principal (DR2) n'est pas triée et contient des doublons de pixel qui ne sont pas nécessairement consécutifs. Afin de réduire l'espace mémoire occupé inutilement, il est nécessaire de supprimer ces derniers.

```
In [2]: lpixels_dab[0:10]
Out[2]:
[[1190, 694, 50.86, 0],
 [1191, 695, 50.858, 1],
 [1190, 696, 50.841, 2],
 [1191, 696, 50.822, 3],
 [1191, 693, 50.887, 4],
 [1192, 693, 50.869, 5],
 [1193, 695, 50.852, 6],
 [1193, 695, 50.833, 7],
 [1193, 696, 50.814, 8],
 [1196, 696, 50.813, 9]]
```

Figure n°10 : Liste non triée selon dab, constituée de $[u \quad v \quad dab \quad index]$, présence de doublons de pixel $[u \quad v]$

Pour un pixel apparaissant plusieurs fois dans la structure **`l_pixels_dab`**, seul celui correspondant à la distance dab la plus petite est à garder.

Par conséquent, le premier traitement consiste à trier les données dans l'ordre décroissant de la distance dab tout en préservant la correspondance avec les coordonnées des pixels.

```
In [4]: lpixels_dab[0:10]
Out[4]:
[[1130, 49, 63.477, 299733],
 [1128, 49, 63.474, 299722],
 [1132, 49, 63.467, 325187],
 [1133, 49, 63.466, 299801],
 [1131, 48, 63.465, 325143],
 [1127, 48, 63.459, 325133],
 [1143, 47, 63.458, 324902],
 [1130, 49, 63.457, 299730],
 [1127, 50, 63.456, 299714],
 [1129, 50, 63.455, 299721]]
```

Figure n°11 : Tri de la liste en fonction de la distance dab

En raison de sa complexité en temps et en espace mémoire, le tri rapide (« quicksort »), dont le principe est présenté dans le DT4, est choisi pour réaliser le premier traitement.

Q 22. Justifier ce choix et compléter dans le DR3 le programme du tri rapide (dans sa version récursive) dans le cas d'une liste simple à trier dans l'ordre décroissant.

Q 23. Compléter dans le DR3, la fonction de partition **`partition_ag`** dans le cas d'une liste composée des index de pixels, de la distance dab et de l'index de la liste **`l_pixels_dab`** selon les valeurs décroissantes de la distance dab.

Le dernier traitement consiste à supprimer les pixels dupliqués. Le principe de ce traitement repose sur l'unicité de l'indice k d'un pixel calculé en fonction des indices de ligne et de colonne comme illustré ci-dessous pour une image de hauteur $H = 4$ et de largeur $L = 4$:

	$u = 0$	$u = 1$	$u = 2$	$u = 3$
$v = 0$	$k = 0$	1	2	3
$v = 1$	4	5	6	7
$v = 2$	…	…	…	…
$v = 3$	…	…	…	15

Q 24. Donner l'expression de k en fonction de u , v et L et écrire la fonction **`suppression_doublons(l_pixels_dab,H,L)`** permettant de retourner une nouvelle liste composée des éléments de **`l_pixels_dab`** sans doublon. La complexité en temps devra être au plus en $O(n)$ et la taille de la nouvelle liste devra correspondre au nombre de points 3D. Au préalable du programme Python, un énoncé du principe de l'algorithme devra être rédigé.

Q 25. Compléter le programme principal avec l'appel des fonctions de tri et de suppression des doublons.

La fin du traitement consiste à créer et à remplir une structure de données qui stocke les canaux R, G, B de l'image et la distance dab pour un pixel donné.

Ainsi à la fin du programme principal, deux variables ont été créées :

- **`imSrc`** : variable permettant de stocker l'image RGB ;
- **`img`** : variable permettant de stocker l'image solide (figure 5).

Q 26. Préciser la taille et le type des données associées à ces variables en considérant les composantes de couleur de l'image source définies comme des réels simple précision. Ajouter les instructions permettant de remplir la variable **`img`**.

Q 27. Conclure sur les performances en déterminant les complexités en temps et en mémoire associées à la création de l'image solide dans le meilleur des cas.

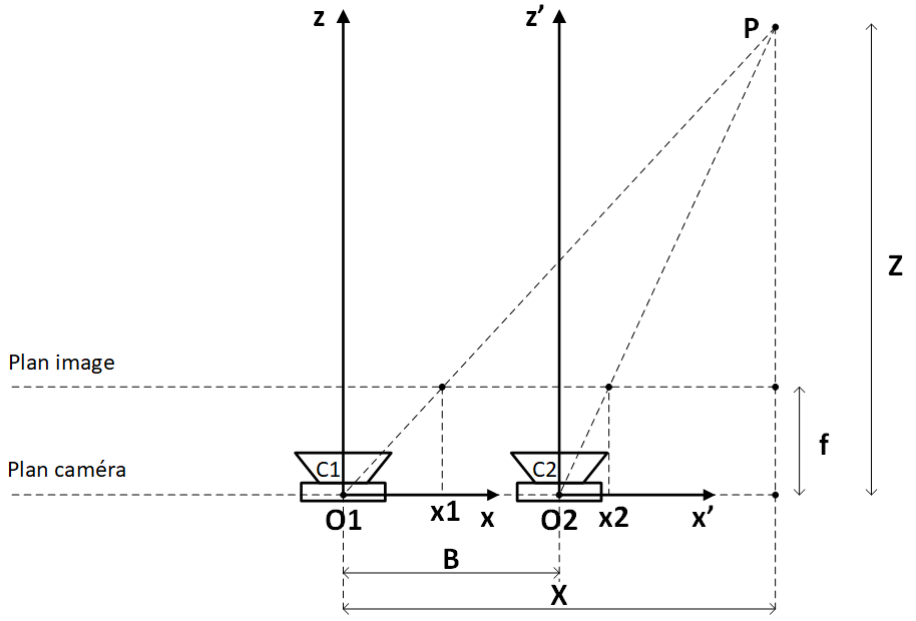
Tournez la page S.V.P.

Partie 3. Solution de photogrammétrie

Objectif : Comparer la solution d'acquisition des points 3D par lasergrammétrie avec une solution utilisant les principes de photogrammétrie.

Afin de diminuer le coût matériel, un système drone-caméra pourrait se substituer au système drone-Lidar-caméra en s'appuyant sur les principes de photogrammétrie.

La figure ci-dessous décrit un système stéréoscopique, équivalent à deux prises de vue consécutives depuis un drone. Ce système est composé de deux caméras calibrées (C1 et C2) fixées sur le même support permettant l'acquisition d'une paire d'images stéréo rectifiées (coplanaires).



Grandeur	Définition
B	Distance entre les deux caméras
f	Distance focale
x_1, y_1	Coordonnées d'un point dans l'image de la caméra C1
x_2, y_2	Coordonnées d'un point dans l'image de la caméra C2
$d = x_1 - x_2$	Disparité : distance entre deux pixels d'une même ligne
X, Y, Z	Coordonnées du point P dans le repère de la caméra C1

Q 28. D'après la figure précédente, démontrer l'expression matricielle ci-dessous décrivant les coordonnées du point P.

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \frac{B}{d} \begin{bmatrix} x_1 \\ y_1 \\ f \end{bmatrix}$$

Q 29. Déterminer la résolution spatiale ΔX et ΔY conformément aux contraintes matérielles décrites dans le tableau ci-après.

B	10 m
Z	30 m
Résolution d'une caméra	7360 x 4912 pixels
f	35 mm
Taille capteur caméra	35.9 mm x 24 mm

Q 30. Déduire de la relation de la question Q.28, la résolution ΔZ en fonction de $\Delta d, B, f, Z$. Faire l'application numérique pour une variation de disparité minimale de 1 pixel. Expliquer comment améliorer la résolution ΔZ à caméra numérique donnée et distance de prise de vue donnée.

La carte des profondeurs est déterminée par un algorithme de mise en correspondance des pixels de l'image de la caméra C1 et des pixels de l'image de la caméra C2 situés sur la même ligne (contrainte épipolaire).

En définissant la ligne de pixels k pour chacune des images :

- $L(k,j)$ où j est associé à l'index de colonne d'un pixel de C1 ;
- $R(k,i)$ où i est associé à l'index de colonne d'un pixel de C2.

$L(k,j)$	1	1	4	5	6

$R(k,i)$	4	5	6	8	8

Q 31. Définir l'équation qui lie un pixel de la ligne k de la caméra de C1 au pixel homologue dans la caméra C2 pour un même point P en faisant apparaître la disparité d.

La programmation dynamique est un algorithme très utilisé dans diverses applications comme la reconnaissance de formes, les fonctions de correction d'erreurs et la recherche d'homologues.
Il peut être vu comme une mesure du coût minimal pour transformer une chaîne d'éléments, définie comme la chaîne d'entrée vers une seconde définie comme la chaîne de référence.

En voici le principe algorithmique pour une ligne k :

- soient $L(j)$ et $R(i)$, respectivement les chaînes d'entrée et de référence, de taille N et indexées de 0 à $N-1$;

- soit $D(i,j) = |L(j) - R(i)|$, la différence entre deux pixels mis en correspondance ;

- le coût minimum $g(i,j)$ pour mettre en correspondance les chaînes $L(j)$ et $R(i)$, est défini comme la somme de $D(i,j)$ avec le minimum des coûts suivants :

    - $g(i,j-1)$, occlusion de pixel dans l'image de droite, équivalent à un déplacement horizontal (numéroté 1 dans la figure 12 ci-après) vers la cellule $g(i,j)$;

    - $g(i-1,j-1)$, correspondance entre pixels équivalent à un déplacement diagonal (numéroté 2) vers la cellule $g(i,j)$;

    - $g(i-1,j)$, occlusion de pixel dans l'image de gauche équivalent à un déplacement vertical (numéroté 3) vers la cellule $g(i,j)$;

Tournez la page S.V.P.

- pour le calcul de la première ligne de $g(0,)$, et de la première colonne $g(,0)$, les valeurs $g(-1,)$, et $g(,-1)$, utilisées lors de l'opération « min » possèdent un coût maximum (v_{max}=50) pour interdire ces possibilités ;

- la valeur d'initialisation de $g(0,0)$ est égale à $|D(0,0)|$.

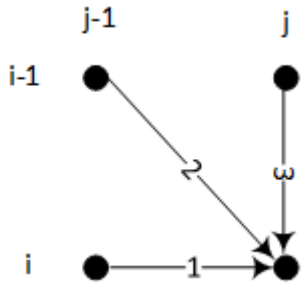


Figure n°12 : Mouvements possibles vers la cellule $g(i,j)$

L'argument choisi par l'opérateur « min » est alors mémorisé et permet, à la fin de l'exécution de l'algorithme, de reconstituer le chemin optimal, à savoir la liste des appariements choisis en remontant depuis le point de programmation dynamique ($N-1$, $N-1$) jusqu'à l'origine (0,0).

Une version en pseudo-code du principe précédent est proposée ci-dessous :

```
Algorithme d'appariement de points homologues

L, chaîne d'entrée,
R, chaîne de référence,
g(-1,.) et g(.,-1) possèdent un coût maximal
initialisation de g(0,0)
Pour j allant de 0 à N-1 faire
 Pour i allant de 0 à N-1 faire
  calcule D(i,j)
  evalue g(i,j)= D(i,j) + min(g(i,j-1), g(i-1,j-1), g(i-1,j))
  mémorise l'argument choisi argmin(i,j)
 Fin pour
Fin pour
```

Q 32. Compléter la table des coûts $g(i,j)$ (DR4) associée aux chaînes L et R et tracer en couleur le chemin qui minimise la fonction de coût. Préciser la résolution minimale de correspondance.

Q 33. Compléter le programme Python implémentant l'algorithme ci-dessus (DR4). Le tableau P permettra de stocker le type de déplacements (1, 2 ou 3) vers la cellule $g(i,j)$. Les fonctions **`amin`** et **`argmin`** de la bibliothèque Numpy (DT3) pourront être utilisées.

À la fin de l'algorithme et conformément aux valeurs des chaînes L et R définies ci-dessus, le tableau P est rempli comme illustré à la figure suivante.

```
In [3]: P
Out[3]:
array([[0., 0., 0., 0., 0., 0.],
       [0., 2., 1., 1., 1., 1.],
       [0., 3., 2., 2., 2., 1.],
       [0., 3., 2., 2., 3., 2.],
       [0., 3., 2., 3., 3., 3.],
       [0., 3., 2., 3., 3., 3.]])
```

Le traitement final permet d'obtenir une structure de données d contenant les index des pixels de L retrouvés dans R associés à leur disparité respective.

```
In [2]: d
Out[2]: [(2, 4), (2, 3), (2, 2)]
```

Q 34. Compléter le programme Python (DR4) permettant, à partir du tableau P, de remplir la structure de données d présentée ci-dessus.

Q 35. Déterminer la complexité en temps et en espace mémoire de l'algorithme d'appariement pour une paire d'images de dimensions égales.

La figure 6 présente une solution logicielle de post-traitement exécutable sur des machines du type ordinateur de bureau. Les systèmes embarqués actuels présentent des capacités de calcul intéressantes et dans un objectif d'amélioration de la réactivité du système de surveillance, certains traitements pourraient être embarqués.

Q 36. Comparer les performances de la technique de stéréoscopie avec la technique par lasergrammétrie en termes de plage de mesure, de résolution, de coût des traitements numériques et d'intégration dans une cible embarquée. Dresser un tableau synthétique comme ci-dessous en ajoutant les critères importants.

Critère	Solution de lasergrammétrie	Solution de photogrammétrie
Plage d'utilisation		
Résolution spatiale		
…		
…		
Intégration embarquée des traitements numériques		

Tournez la page S.V.P.

Partie 4. Analyse des images par segmentation

Objectif :
Analyser les images solides des parois rocheuses pour la détermination des signatures caractéristiques des fragilités.

Les traitements numériques sont basés sur l'exploitation des données, images 2D, points 3D et images solides acquises par un drone d'inspection. Ils doivent permettre la classification des discontinuités présentes dans les images acquises (figure 5, figure 18).

Ces discontinuités apparaissent comme des éléments filaires. Ce linéament est une donnée importante des parois rocheuses, car il constitue une trace visible en surface d'un accident profond qui peut être objectivé à travers plusieurs paramètres ; l'orientation des *linéaments*, l'orientation des plans de discontinuité, les espacements entre les *linéaments* et la persistance (surface des discontinuités).

Le premier traitement concerne la détection des contours.

La relation ci-dessous décrit la fonction de convolution d'une image en niveaux de gris $I(x,y)$ de taille $(H) \times (L)$ avec le noyau $h(x,y)$ de taille $(2p+1) \times (2p+1)$.

$$y(x,y) = I(x,y) * h(x,y) = \sum_{i=-p}^{p} \sum_{j=-p}^{p} I(x-i, y-j) \cdot h(i,j)$$

Q 37. Compléter le code Python du DR5 (bibliothèque Numpy DT3 imposée) permettant de déterminer l'image filtrée par le noyau h, puis compléter le DR5 avec les valeurs manquantes. La gestion des bords repose sur la technique de l'enroulage.

Le premier extracteur utilise un filtre multirésolution de type DoG – Difference of Gaussian. L'analyse multiéchelle permet de mettre en évidence différentes tailles de structure dans une image. Le facteur d'échelle représente une troisième composante σ associée à chaque pixel (x,y). La fonction $L(x,y,\sigma)$ est obtenue par le produit de convolution entre l'image $I(x,y)$ et une fonction gaussienne $G(x,y,\sigma)$ définie ci-dessous :

$$G(x,y,\sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2}$$

Q 38. Donner l'expression de la fonction $DoG = L(x,y,k\sigma) - L(x,y,\sigma)$, en fonction de $G(x,y,k\sigma)$, $G(x,y,\sigma)$ et de $I(x,y)$.

En utilisant la propriété suivante du produit de convolution :

$$\Delta L(x,y) = \frac{\partial^2 L}{\partial x^2} + \frac{\partial^2 L}{\partial y^2} = I(x,y) * \Delta G(x,y)$$

Q 39. Donner l'expression du laplacien de $L(x,y)$ et argumenter concernant l'appellation filtre LoG – Laplacian of Gaussian.

Q 40. Déduire par calcul que la différence de gausienne – DoG – représente une bonne approximation d'un Laplacien d'une Gaussienne – LoG, à un facteur près à déterminer.

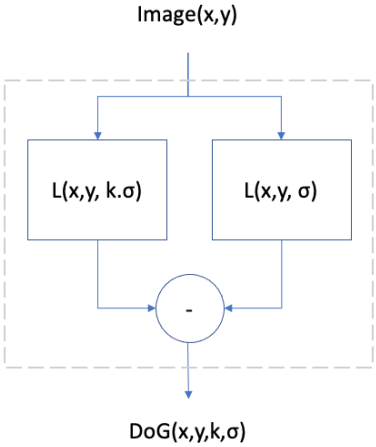
Q 41. Représenter l'allure des réponses impulsionnelles d'un DoG et du LoG, pour $\sigma = 1$ et $k = 2$. Évaluer l'influence de k.

Q 42. Définir une fonction **`masque2D(sigma,p)`** en Python qui calcule le masque normalisé associé à la fontion G de taille $(2p+1) \times (2p+1)$.

Q 43. À partir de la décomposition du filtre Gaussien en deux filtres 1D, montrer que le filtre Gaussien est un filtre séparable au sens de la convolution.

Q 44. Estimer le nombre d'opérations N_C pour un pixel pour une convolution 2D classique. Estimer le nombre d'opérations N_S en exploitant la séparabilité du filtre. En déduire le gain (rapport N_C/N_S).

La fonction DoG est illustrée suivant le schéma ci-dessous :



Q 45. Écrire la fonction DoG en Python et les fonctions dont elle dépend (DR6) qui renvoie l'image de la différence (résultat à la figure 13). La séparabilité du filtre sera utilisée.

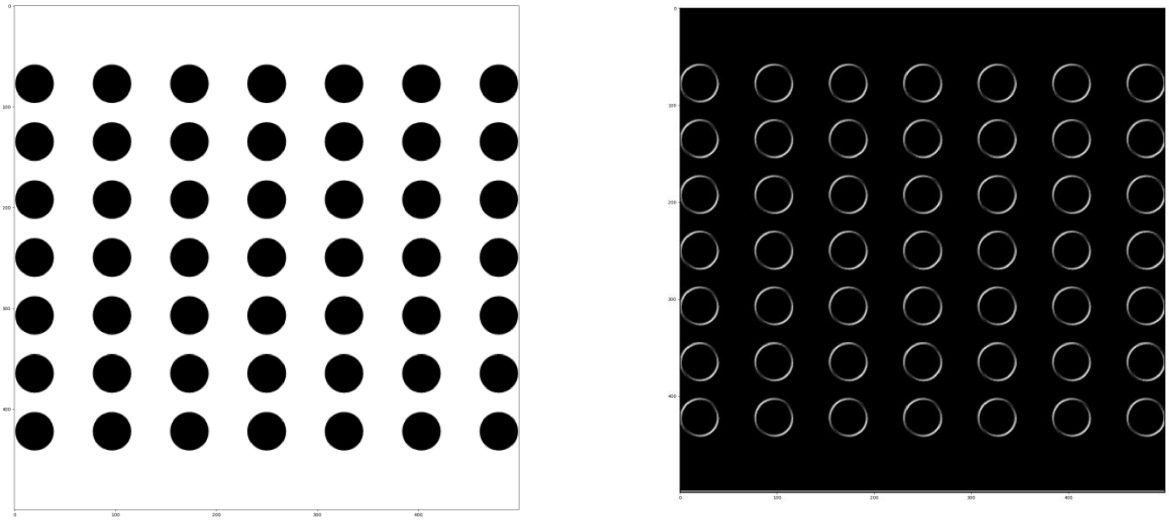


Figure n°13 : À gauche, image d'origine, à droite image du DoG $(k = \sqrt{2}, \sigma = 1)$

Q 46. En utilisant la fonction précédente $L(x, y, \sigma)$ écrite en Python, compléter la fonction Python $DoGN(Image, N, k, \sigma, p)$ du DR6 qui prend en entrée l'image à traiter, le nombre de gaussienne, la valeur de k, la valeur de σ et qui renvoie l'image filtrée constituée des pixels de valeur maximale (points d'intérêt) parmi les N DoG de l'octave 1.

Tournez la page S.V.P.

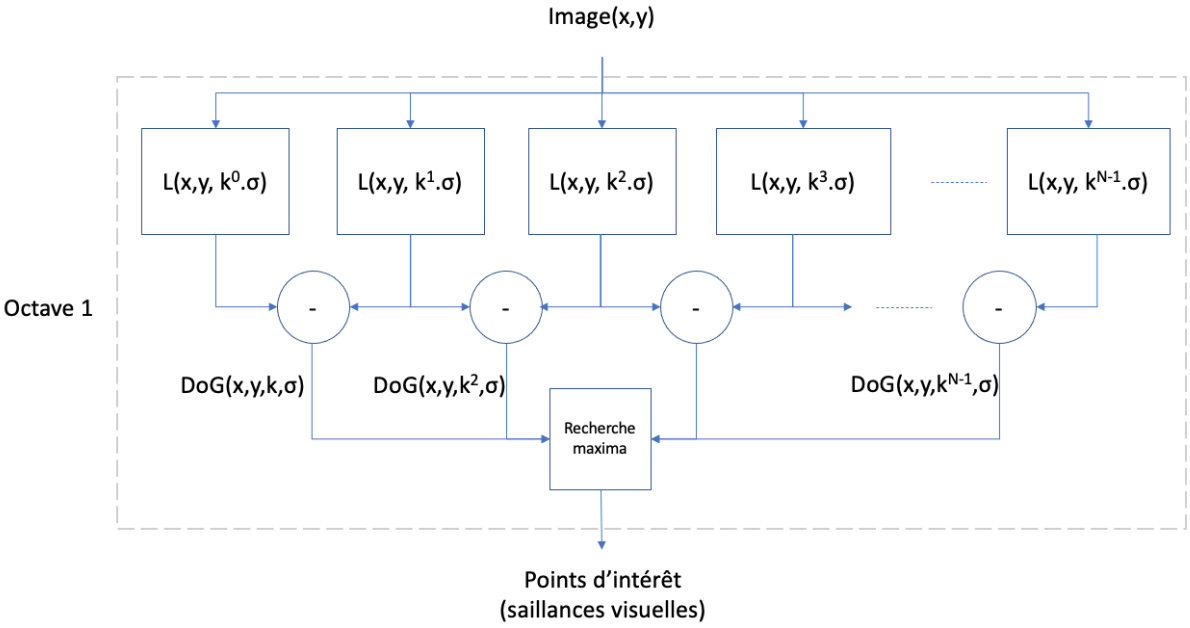


Figure n°14 : Fonction $DoGN$ sur N étages

Les octaves supérieures sont obtenues en sous-échantillonnant l'image d'entrée par 2^{n-1} (n représente le rang de l'octave) et en appliquant les filtres DoG (figure 15). Les points d'intérêt (ou saillance) sont obtenus en réalisant pour chaque pixel, la recherche de la valeur maximale entre les différentes octaves. Le sous-échantillonnage de l'image orginale est réalisé à base de slicing en Python.

Q 47. En utilisant la fonction de la question précédente, compléter la fonction de filtrage multirésolution, DoG_MR du DR6, qui reprend la structure du filtre de la figure 15 ci-après. Cette fonction permettra de réaliser un filtrage multirésolution sur l'image d'entrée, en vue d'extraire les points d'intérêts.

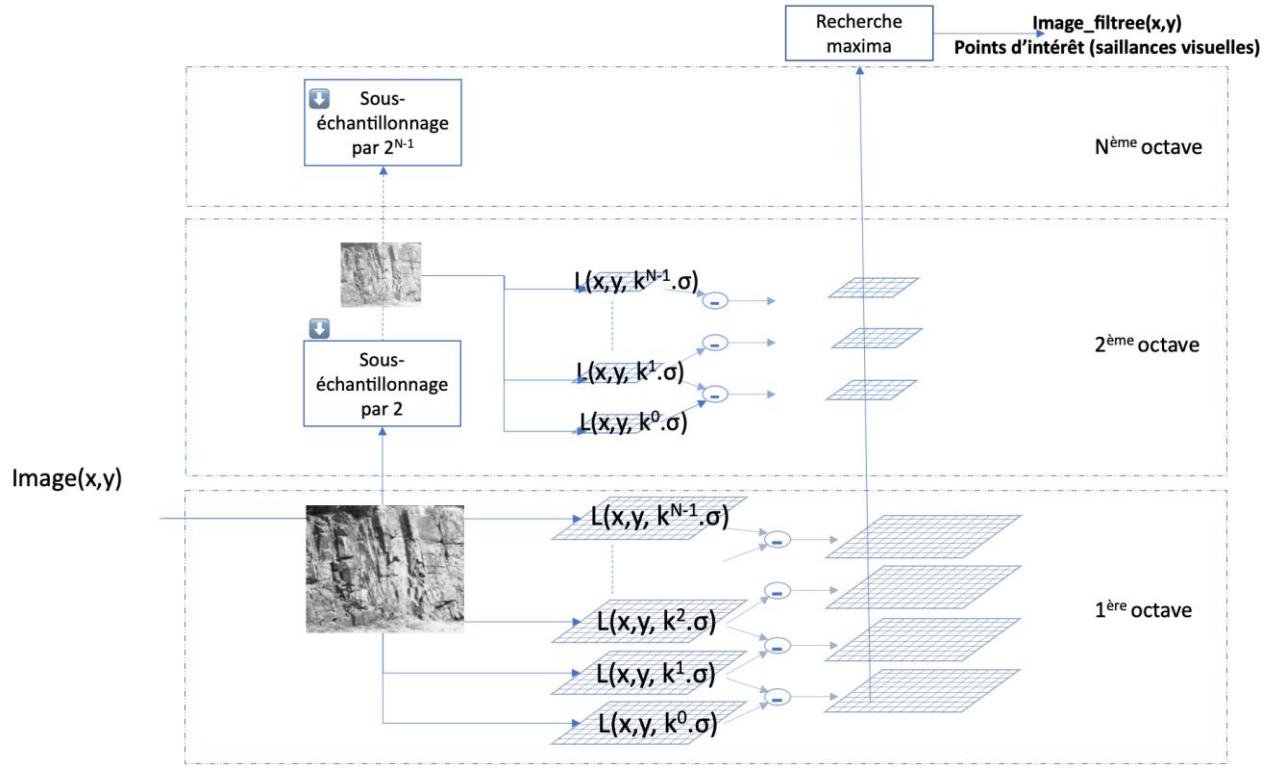


Figure n°15 : Filtre pyramide DoG_MR multirésolution

La sortie du filtre DoG_MR est seuillée avec un niveau fixe afin de produire une image de contours.

L'estimation des paramètres géométriques de chaque contour linéaire nécessite une étape préliminaire de segmentation qui permet de séparer chaque contour linéaire et de les labelliser. L'algorithme RANSAC – Random Sample Consensus – est utilisé pour effectuer ce traitement. Il procède de manière itérative sur un tirage aléatoire d'un jeu d'échantillons de taille suffisante afin d'estimer les paramètres du modèle mathématique recherché.

Pour chaque modèle estimé, l'algorithme ne retient que les points « valables » (*inliers*), c'est-à-dire situés dans un intervalle de tolérance fixé, et rejette les données aberrantes (*outliers*). L'estimation du modèle est alors réajustée sur les *inliers* tout en quantifiant leur niveau de correspondance avec le modèle estimé. La figure 17 présente le résultat de cet algorithme appliqué sur un contour linéaire en présence de bruit (*outliers*).

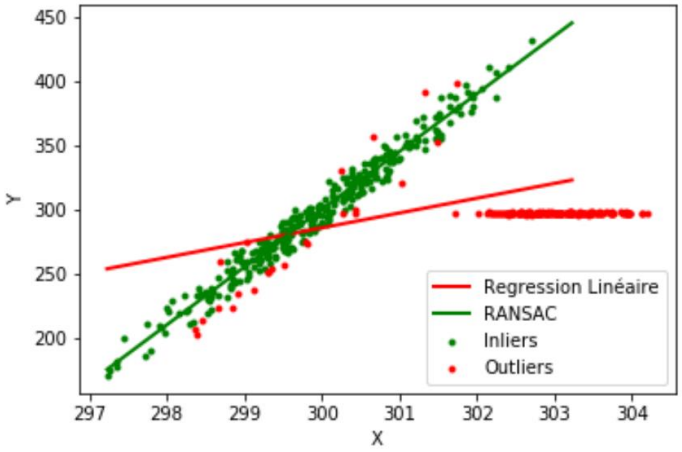


Figure n°17 : RANSAC appliqué sur un contour linéaire. Comparaison à une régression linéaire

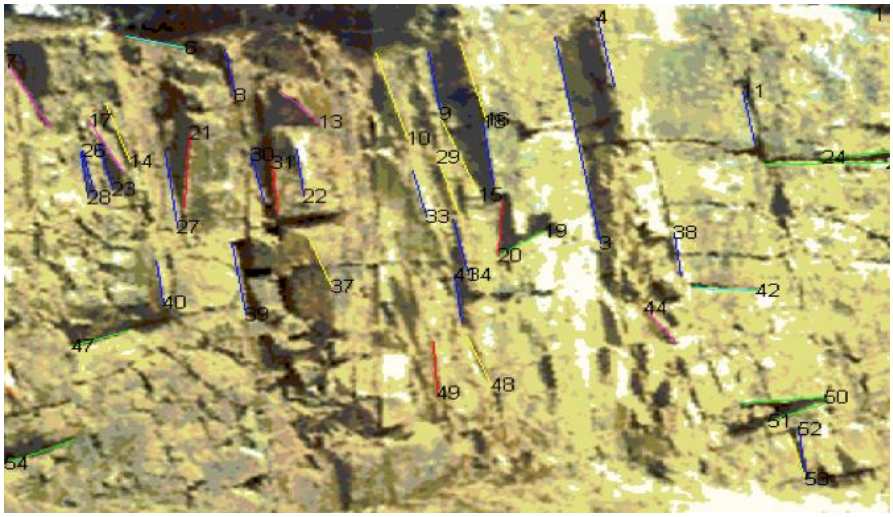


Figure n°18 : Exemple de contours segmentés par RANSAC

L'algorithme RANSAC étant itératif, le nombre d'itérations maximum peut être prédéfini avec une probabilité P donnée. Si w est la probabilité de tirer à chaque itération un point valable, et si les n points de l'échantillon sont sélectionnés de façon indépendante, alors la probabilité que ces n points soient valables est donnée par w^n.

Tournez la page S.V.P.

Q 48. En considérant, k : le nombre d'itérations (ou tirages), P : la probabilité que l'algorithme sélectionne n points valables sur k itérations, w : la probabilité de tirer un point valable à chaque itération, déterminer la relation lien k à P, w et n.

Q 49. En utilisant la fonction `linear_model.RANSACRegressor` (bibliothèque Scikit-learn DT3), écrire une fonction en Python qui estime les paramètres d'un segment de contour passé en argument. Les paramètres renvoyés seront les coefficients de la droite ainsi que son orientation 2D. Les paramètres estimés seront stockés dans un fichier « angle.csv ».

Q 50. Rappeler brièvement le principe du *clustering*. La classification des orientations des segments utilise l'algorithme K-means (bibliothèque Scikit-learn DT3). La labellisation est réalisée avec un a priori de 10 degrés en termes de résolution. Ce *clustering* sera appliqué sur le fichier « angle.csv » et renverra la classe du linéament. Écrire le programme Python associé à ce traitement en utilisant les bibliothèques ci-dessous.

```
from sklearn import linear_model,

datasets from sklearn.cluster import KMeans
```

Un ensemble de $\boldsymbol{n}$ éléments linéaires segmentés en fonction de leur orientation est obtenu suite à ces traitements. Le résultat de la classification obtenu sur la figure 18 avec un K-means de 6, est donné ci-dessous :

Linéaments	Angle (degrés)
Rouge	56.97
Vert	165.66
Bleu	80.56
Jaune	10.26
Cyan	72.27
Magenta	89

La méthode est maintenant étendue au formalisme 3D à travers le traitement des images solides. Il s'agit ici de classifier les familles de plans de discontinuités 3D en fonction de leur orientation (vecteur normal au plan de discontinuité). La fonction de Hough est utilisée pour segmenter les plans de discontinuité d'une image solide `img` passée en argument (figure 5).

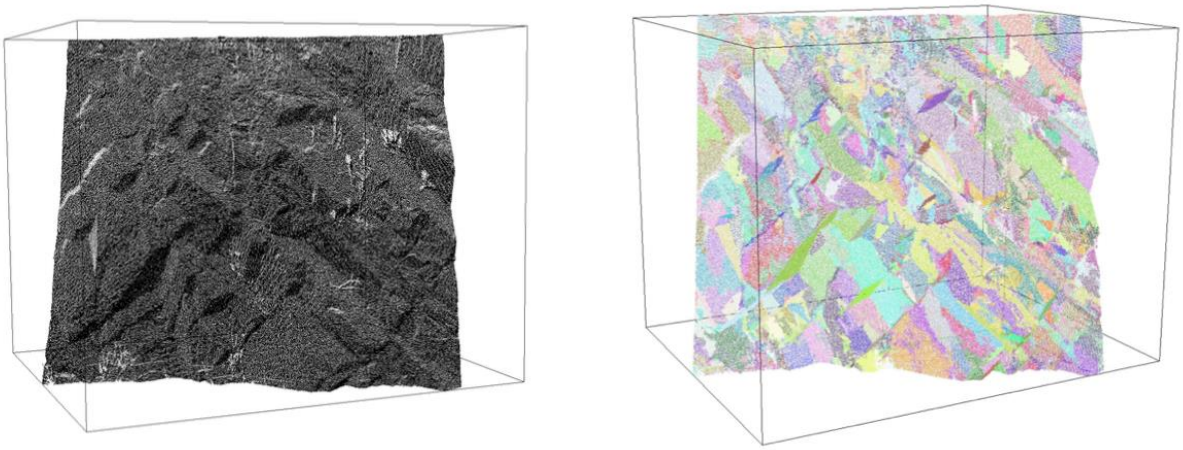


Figure n°19 : À gauche : nuage de points 3D brut acquis par le Lidar. À droite : nuage de points segmenté par application de la transformée de Hough, une couleur par plan détecté – extraction de 407 plans [Thèse S.Slob 2010].

L'équation d'un point $P(X_j, Y_j, Z_j)$ appartenant à un plan P_i de vecteur normal (a_i, b_i, c_i) et passant par le point $O(x_0, y_0, z_0)$ satisfait l'équation suivante :

$$a_i.(X_j - x_o) + b_i.(Y_j - y_o) + c_i.(Z_j - z_o) = O$$

Cette équation se réécrit :

$$Z_j = A_i.X_j + B_i.Y_j + C_i$$

Avec :

$$A_i = -\frac{a_i}{c_i} \qquad B_i = -\frac{b_i}{c_i} \qquad C_i = \frac{a_i.x_0 + b_i.y_0 + c_i.z_0}{c_i}$$

Les paramètres (A_i, B_i, C_i), notés E, sont estimés au sens des moindres carrés en minimisant la fonction de coût $M(A_i, B_i, C_i)$.

$$A_i.X_j + B_i.Y_j + C_i = P(X_j, Y_j) \qquad e_j = |P(X_j, Y_j) \quad -Z_j| \qquad M(A_i, B_i, C_i) = \sum_{j=1}^{n} e_j^2$$

Q 51. Démontrer l'équation ci-dessous.

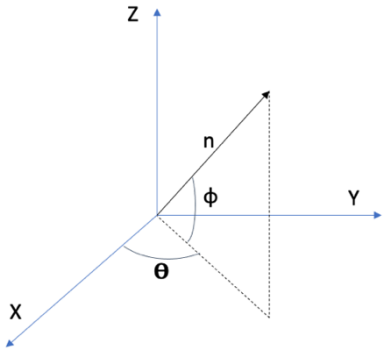
$$\begin{bmatrix} \sum_{j=1}^{n} x_j^2 & \sum_{j=1}^{n} x_j.y_j & \sum_{j=1}^{n} x_j \\ \sum_{j=1}^{n} x_j.y_j & \sum_{j=1}^{n} y_j^2 & \sum_{j=1}^{n} y_j \\ \sum_{j=1}^{n} x_j & \sum_{j=1}^{n} y_j & n \end{bmatrix} . \begin{bmatrix} A_i \\ B_i \\ C_i \end{bmatrix} = \begin{bmatrix} \sum_{j=1}^{n} x_j.z_j \\ \sum_{j=1}^{n} y_j.z_j \\ \sum_{j=1}^{n} z_j \end{bmatrix}$$

$$E = (F^T.F)^{-1}.F^T.D$$

Q 52. Identifier dans l'équation ci-dessus les paramètres E, F et D.

Q 53. À partir de l'équation ci-dessus, écrire une fonction en Python permettant d'estimer ces trois paramètres à partir d'un nuage de points 3D (X_i, Y_i, Z_i) segmentés issus de la fonction de Hough et passés en argument.

Q 54. Déterminer les paramètres d'orientation en azimut (θ) et élévation (ϕ) (voir figure ci-après) du plan P_i.



Tournez la page S.V.P.

Q 55. La classification des orientations des plans de discontinuités est basée sur l'utilisation d'un k-means à partir des valeurs extraites précédemment. Les plans sont contenus dans un hémisphère avec une résolution angulaire de 10°. Estimer la valeur de K. En utilisant la fonction **`kMeans,`** écrire le programme qui catégorise les plans de discontinuité à partir des paramètres d'orientation (θ_i, ϕ_i) de chaque plan P_i. Les valeurs (θ_i, ϕ_i) de chaque plan P_i sont stockées dans un fichier « angle3D.csv ». Les valeurs renvoyées par le programme sont les numéros de groupe pour chaque vecteur.

Afin de mesurer l'apport du traitement des images solides, il est possible de comparer les résultats obtenus avec des valeurs déterminées de manière classique ; mesures manuelles collectées in situ et mesures issues du traitement photogrammétrique. Le tableau ci-dessous résume les résultats obtenus pour 5 types de plan extraits (figure 20).

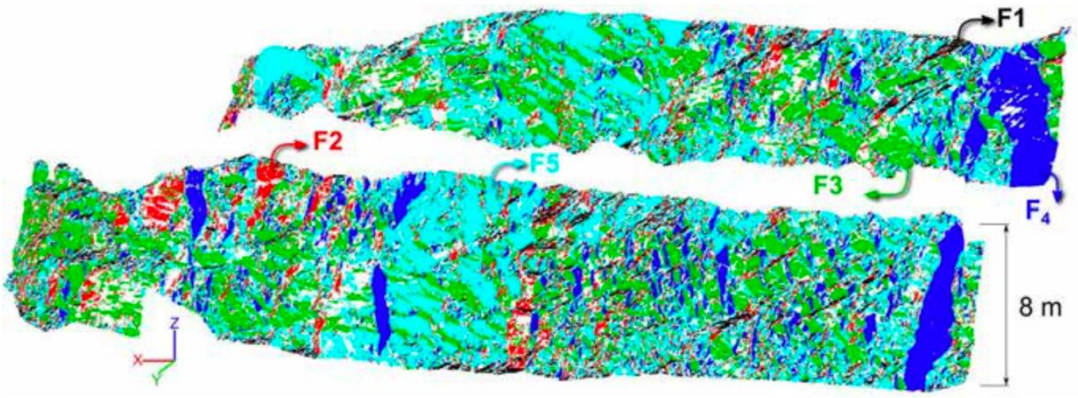


Figure n°20 : Exemple d'extraction des plans de discontinuité par le traitement des orientations à partir de l'image solide. Les F_i correspondent aux familles de plan de discontinuité, classées selon l'orientation du vecteur normal au plan

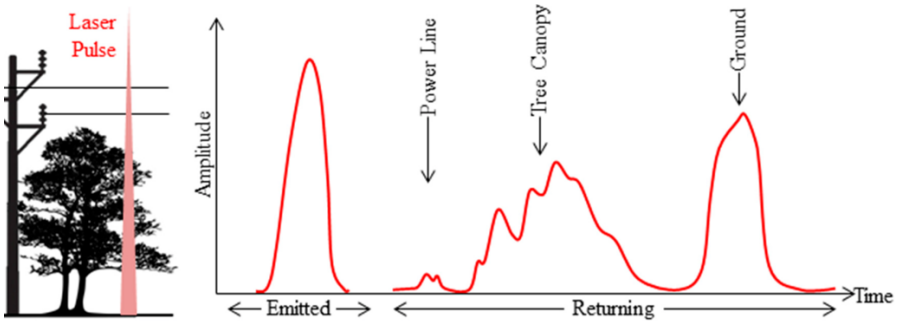
	Plan F1		***Plan F2***		***Plan F3***		***Plan F4***		***Plan F5***	
Orientations (en degrés)	θ	Φ	θ	Φ	θ	Φ	θ	Φ	θ	Φ
Relevés expérimentaux – vérité terrain	#	#	162.6	30.3	130.6	59.9	72.5	69.1	81.2	5.2
Estimation par traitement lasergrammétrique	339.1	10.2	156.7	28.8	127.5	57.8	73.6	70.2	89.2	4.8
Estimation par traitement photogrammétrique	343.7	15.5	140.3	24.2	121.8	52.5	62.7	65.9	86.0	2.5

Q 56. Pour les résultats consignés dans la table ci-dessus, estimé les erreurs moyennes des deux approches. Conclure sur l'approche du traitement des images solides.

Document technique DT1 : scanner-Lidar

Principe :
Le principe du Lidar (Light Detection And Ranging) repose sur deux opérations qui ont lieu consécutivement :

- la première assure l'émission d'une onde électromagnétique ;
- la seconde assure la mesure de la durée de propagation (aller-retour) des échos multiples ainsi que la mesure de leur intensité respective.



Extrait Kashani 2015

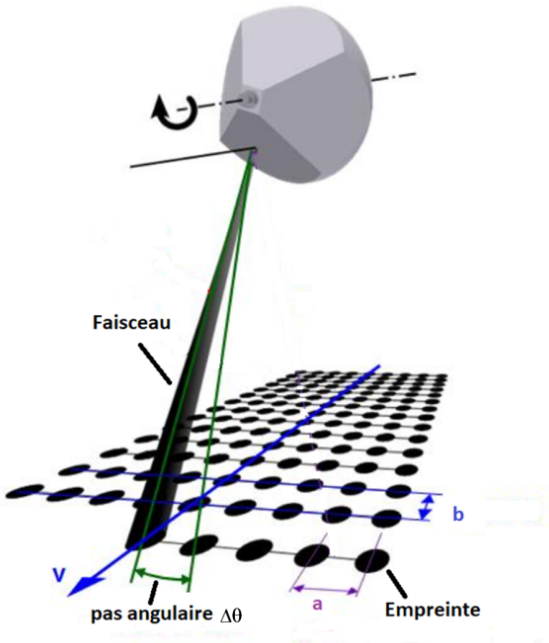
Durée en ns	
Emission – Caténaire	10
Emission – Arbre	12
Emission – Sol	20

La distance est calculée en fonction de la vitesse de la lumière c et la durée de propagation d'un écho.

$$d = \frac{c \times T}{2}$$

Échos multiples :
En fonctionnement, le Lidar génère périodiquement des pulses à la fréquence PRR (Pulse Repetition Rate). À chaque pulse peut correspondre 0 (obstacle à une distance supérieure à R définie dans le tableau ci-après), 1 et jusqu'à 6 échos.

Balayage :
Un dispositif de balayage du faisceau (miroir polygonal tournant) selon un seul axe de rotation permet d'acquérir plusieurs points (empreintes) situés sur une même ligne.

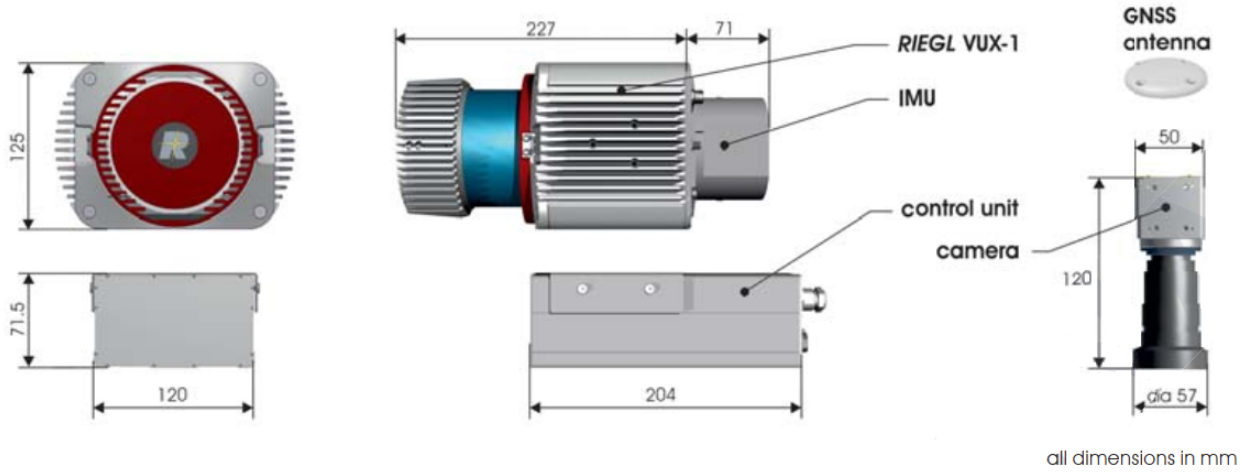


Tournez la page S.V.P.

C

La fréquence de scan (Scanning Rate, SR) définit le nombre de balayages du faisceau par seconde.
L'angle d'ouverture (Field Of View, FOV) définit la plage angulaire de balayage.
Les espaces entre les lignes et les empreintes sont définis respectivement par b et a.

Caractéristiques du scanner-Lidar Riegl vux-uav :



Caractéristique	Symbole	Valeurs	Unités
Pulse Repetition Rate	PRR	50-550 (min-max)	kHz
Scanning Rate	SR	10-200 (min-max)	Hz
Field Of View	FOV	330 (max)	Degrés
Range	R	340 (PRR@550kHz)	m
Height	H	190 (max)	m
Laser beam divergence	γ	0.5	mrad

Fichier de nuage de points :

Lors d'une campagne de mesures, les données sont enregistrées dans un fichier binaire au format « .las » qui constitue le format standard d'échange de données en lasergrammétrie et photogrammétrie. Le tableau ci-dessous présente le format d'une acquisition d'un fichier de ce type.

Item	Définition	Format	Taille (octets)
X	Coordonnées du point 3D	Long	4
Y		Long	4
Z		Long	4
Intensity	Amplitude de l'écho	Unsigned short	2
Return number	Index de l'écho	Bit 0, 1, 2	1
Number of returns	Nombre d'échos	Bit 3, 4, 5	
Scan direction flag	Sens de rotation du scanner	Bit 6	
Edge of flight line	Fin d'une ligne de scan	Bit 7	
Classification	Index du type d'écho (sol, végétation, etc.)	Unsigned char	1

Document technique DT2 : Extrait des fichiers de données

Le tableau ci-dessous présente le contenu du fichier « Carriere_ZoneEtude[2cm].asc » du type « csv » concernant le nuage de points 3D. En raison des contraintes d'acquisition, ces points ne sont pas triés.

1	954.423096	974.645935	498.174988	0.130873	0.976516	-0.171142
2	954.431946	974.628601	498.125000	0.318486	0.908274	0.271303
3	954.421997	974.638428	498.149994	0.217846	0.975570	-0.028415
…	…	…	…	…	…	…
…	…	…	…	…	…	…
1770100	991.772095	942.765198	499.184998	0.105114	0.548929	0.829233
1770101	991.757507	942.781006	499.183990	0.232139	0.525368	0.818596
1770102	991.741211	942.795288	499.183990	0.441274	0.365627	0.819509

Avec :

Colonne n°1	x_p	Coordonnées du point 3D dans le repère « Monde » (en m)
Colonne n°2	y_p	
Colonne n°3	z_p	
Colonne n°4	n_x	Direction du plan (normale à la surface construite à partir des points)
Colonne n°5	n_y	
Colonne n°6	n_z	

Le tableau ci-dessous présente le contenu du fichier « img_3886_test-InfosCam.txt » du type « csv » concernant les paramètres de la caméra.

1026.769437
989.403669
495.906265
-93.933868
41.039028
-175.523472
0.0992083
0.0359958
0.024
0.0181339
0.012035

Avec :

Ligne n°1	d_x	Coordonnées centre optique dans le repère « Monde » (en m)
Ligne n°2	d_y	
Ligne n°3	d_z	
Ligne n°4	α	Orientation de la caméra (en degrés)
Ligne n°5	β	
Ligne n°6	γ	
Ligne n°7	f	Focale (en m)
Ligne n°8	L_{cmos}	Largeur capteur (en m)
Ligne n°9	H_{cmos}	Hauteur capteur (en m)
Ligne n°10	ppx	Coordonnées du point principal ou point milieu du capteur de la caméra (en m)
Ligne n°11	ppy	

Tournez la page S.V.P.

Document technique DT3 : Extrait de données et de fonctions Python

Bibliothèque Numpy

An ndarray is a (usually fixed-size) multidimensional container of items of the same type and size. The number of dimensions and items in an array is defined by its shape, which is a tuple of N non-negative integers that specify the sizes of each dimension. The type of items in the array is specified by a separate data-type object (dtype), one of which is associated with each ndarray.

Indexing

ndarrays can be indexed using the standard Python x[obj] syntax, where x is the array and obj the selection. There are three kinds of indexing available: field access, basic slicing ([start : end : step]), advanced indexing. Which one occurs depends on obj.

Example

```
>>> a=np.eye(5,5)
>>> print a
[[ 1.  0.  0.  0.  0.]
 [ 0.  1.  0.  0.  0.]
 [ 0.  0.  1.  0.  0.]
 [ 0.  0.  0.  1.  0.]
 [ 0.  0.  0.  0.  1.]]
```

```
>>> a[:3,:4]=4
>>> print a
array([[4., 4., 4., 4., 0.],
       [4., 4., 4., 4., 0.],
       [4., 4., 4., 4., 0.],
       [0., 0., 0., 1., 0.],
       [0., 0., 0., 0., 1.]])
>>> a[::2,::2]
>>> print a
array([[4., 4., 0.],
       [4., 4., 0.],
       [0., 0., 1.]])
```

numpy.array(object)
Create an array.

Parameters

Object [array_like] : an array, any object exposing the array interface, an object whose __array__ method returns an array, or any (nested) sequence.

Returns : an array object satisfying the specified requirements.

Example

```
>>> numpy.array([[1, 2], [3, 4]])
array([[1, 2],
       [3, 4]])
```

numpy.zeros(shape, dtype=float)
Return a new array of given shape and type, filled with zeros.

Parameters

shape [int or tuple of ints] : shape of the new array, e.g., (2, 3) or 2.

dtype [data-type, optional] : the desired data-type for the array, e.g., numpy.int8. Default is numpy.float64.

Returns : array of zeros with the given shape and dtype.

Examples

```
>>> numpy.zeros((5,), dtype=int)
array([0, 0, 0, 0, 0])
>>> np.zeros((2, 1))
array([[ 0.],
       [ 0.]])
```

ndarray.transpose( )
Returns a view of the array with axes transposed. For a 2-D array, this is a standard matrix transpose.

Returns : View of a, with axes suitably permuted.

Example

```
>>> a = numpy.array([[1, 2], [3, 4]])
>>> a.transpose()
array([[1, 3],
       [2, 4]])
```

numpy.amax(a, axis)
Returns the maximum of an array or maximum along an axis.
Parameters
a [array_like] : input data
axis [none or int or tuple of ints optional]. If this is a tuple of ints, the maximum is selected over multiple axes, instead of a single axis or all the axes as before.
Return amax : ndarray of scalar. Maximum of a.
Example

```
>>> a = np.arange(4).reshape((2,2))
>>> a
array([[0, 1],
       [2, 3]])
>>> np.amax(a)
3
```

numpy.amin(a, axis=None)
Return the minimum of an array or minimum along an axis.
Parameters
a [array_like] : input data.
axis [None or int or tuple of ints, optional] : Axis or axes along which to operate. By default, flattened input is used
Returns : Minimum of a. If axis is None, the result is a scalar value. If axis is given, the result is an array.
Example

```
>>> a
array([[0, 1],
       [2, 3]])
>>> np.amin(a)#Minimum of the flattened array
0
np.amin(a, axis=0)#Minima along the first axis
array([0, 1])
```

numpy.argmin(a, axis=None)
Returns the indices of the minimum values along an axis.
Parameters
a [array_like] : input array.
axis [int, optional] : By default, the index is into the flattened array, otherwise along the specified axis.
Returns : index_array, ndarray of ints, array of indices into the array. It has the same shape as a.

numpy.matmul(x1, x2)
Matrix product of two arrays.
Parameters
x1, x2 : array_like, input arrays, scalars not allowed.
Returns: y ndarray, the matrix product of the inputs. This is a scalar only when both x1, x2 are 1-d vectors.
Example

```
>>> a = numpy.array([[1, 0],
...                  [0, 1]])
>>> b = numpy.array([[4, 1],
...                  [2, 2]])
>>> numpy.matmul(a, b)
array([[4, 1],
       [2, 2]])
```

Traitement de fichiers texte et chaînes de caractères

open(file, mode='r')
Open file and return a corresponding file object.
Parameters
file : a path-like object giving the pathname of the file to be opened.
mode: an optional string that specifies the mode in which the file is opened. It defaults to 'r' which means open for reading in text mode.

readlines(hint=-1)
Read and return a list of lines from the stream.

Tournez la page S.V.P.

Parameters
hint can be specified to control the number of lines read: no more lines will be read if the total size (in bytes/characters) of all lines so far exceeds hint.

str.split(sep=None, maxsplit=-1)
Return a list of the words in the string, using sep as the delimiter string.

Parameters
If maxsplit is given, at most maxsplit splits are done (thus, the list will have at most maxsplit+1 elements). If maxsplit is not specified or -1, then there is no limit on the number of splits (all possible splits are made).

If sep is given, consecutive delimiters are not grouped together and are deemed to delimit empty strings (for example, '1,,2'.split(',') returns ['1', '', '2']). The sep argument may consist of multiple characters (for example, '1<>2<>3'.split('<>') returns ['1', '2', '3']). Splitting an empty string with a specified separator returns [''].

Example
```
>>> '1,2,3'.split(',')
['1', '2', '3']
```

Read csv file
```
Import csv
with open('data.csv', newline='') as csvfile:
    reader = csv.reader(csvfile)
    for row in reader:
        print(row)
```
Write csv file
```
with open('write.csv', 'w', newline='') as csvfile:
    writer = csv.writer(csvfile)
    writer.writerow(('123', '456', '789'))
```

Listes

The most versatile compound data types are *list*, which can be written as a list of comma-separated values (items) between square brackets. Lists might contain items of different types, but usually the items all have the same type.

```
list1 = ['physics', 'chemistry', 1997, 2000];
list2 = [1, 2, 3, 4, 5];
list3 = ["a", "b", "c", "d"];
```

Create an empty list
```
List4=[ ]
```

list.append(x)
Add an item to the end of the list.

Indexing and slicing
```
>>> list3[3]
'd'
>>> list3[1:3]
['b', 'c']
```

Bibliothèque Scikit-learn

Scikit-image is a collection of algorithms for image processing.

KMeans(*n_clusters=8*, *n_init=10*, *max_iter=300*)

Parameters
n_clusters [int, default=8]: The number of clusters to form as well as the number of centroids to generate.
n_init [int, default=10]: Number of time the k-means algorithm will be run with different centroid seeds. The final results will be the best output of n_init consecutive runs in terms of inertia.
max_iter [int, default=300]: Maximum number of iterations of the k-means algorithm for a single run.

Methods

fit(X[, y, sample_weight])	Compute k-means clustering.
fit_predict(X[, y, sample_weight])	Compute cluster centers and predict cluster index for each sample.
fit_transform(X[, y, sample_weight])	Compute clustering and transform X to cluster-distance space.

get_params([deep])	Get parameters for this estimator.
predict(X[, sample_weight])	Predict the closest cluster each sample in X belongs to.
score(X[, y, sample_weight])	Opposite of the value of X on the K-means objective.
set_params(**params)	Set the parameters of this estimator.
transform(X)	Transform X to a cluster-distance space.

Example

```
>>>from pandas import DataFrame
>>>from sklearn.cluster import KMeans

>>>Data                                    =                                    {'x':
[25,34,22,27,33,33,31,22,35,34,67,54,57,43,50,57,59,52,65,47,49,48,35,33,44,45,3
8,43,51,46],
        'y':
[79,51,53,78,59,74,73,57,69,75,51,32,40,47,53,36,35,58,59,50,25,20,14,12,20,5,29
,27,8,7]
       }
>>>df = DataFrame(Data,columns=['x','y'])
>>>kmeans = KMeans(n_clusters=3.fit(df)
>>>centroids = kmeans.cluster_centers_
```

linear_model.RANSACRegressor(base_estimator=None, *, min_samples=None, residual_threshold=None, is_data_valid=None, is_model_valid=None, max_trials=100, max_skips=inf, stop_n_inliers=inf, stop_score=inf, stop_probability=0.99, loss='absolute_loss', random_state=None)

Parameters

base_estimator [object**,** optional]

Base estimator object which implements the following methods:

- fit(X, y): Fit model to given training data and target values.
- score(X, y): Returns the mean accuracy on the given test data, which is used for the stop criterion defined by stop_score. Additionally, the score is used to decide which of two equally large consensus sets is chosen as the better one.
- predict(X): Returns predicted values using the linear model, which is used to compute residual error using loss function.

If base_estimator is None, then base_estimator=sklearn.linear_model.LinearRegression() is used for target values of dtype float.

Note that the current implementation only supports regression estimators.

random_state [int] ***:*** RandomState instance, default=None

The generator used to initialize the centers. Pass an int for reproducible output across multiple function calls

Methods

fit(X, y[, sample_weight])	Fit estimator using RANSAC algorithm.
get_params([deep])	Get parameters for this estimator.
predict(X)	Predict using the estimated model.
score(X, y)	Returns the score of the prediction.
set_params(**params)	Set the parameters of this estimator.

Example

```
>>>import numpy as np
>>>from sklearn import linear_model, datasets
>>>n_samples = 1000
>>>X, y, coef = datasets.make_regression(n_samples=n_samples, n_features=1,
                                        n_informative=1, noise=10,
                                        coef=True, random_state=0)
>>>ransac = linear_model.RANSACRegressor()
>>>ransac.fit(X, y)
```


Tournez la page S.V.P.

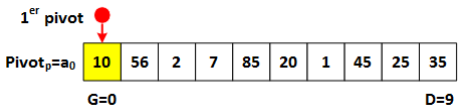
Document technique DT4 : Principe du tri rapide

Ce tri repose sur le principe du « Diviser pour régner ».

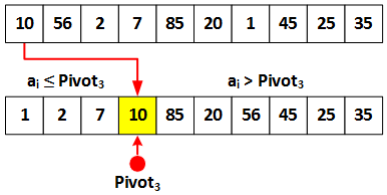
a) Choix du pivot

La première étape consiste à choisir à partir du tableau $T = [a_0, a_1, \dots, a_i, \dots, a_{N-1}]$ de N éléments, délimité par les index $G = 0$ et $D = N - 1$, un élément a_i appelé « pivot » et identifié par $Pivot_p$ avec $p \in [0, N-1]$.
Une solution arbitraire consiste à choisir le premier pivot tel que $Pivot_p = a_0$.

Choix du pivot



Partitionnement



b) Partitionnement

Le partitionnement consiste à déplacer les éléments de part et d'autre du pivot $Pivot_3 = 10$ placé à l'emplacement $i = 3$, tel que pour un tri croissant :

- les éléments sont placés à gauche du pivot si $a_i \leq Pivot_3$;
- les éléments sont placés à droite du pivot si $a_i > Pivot_3$.

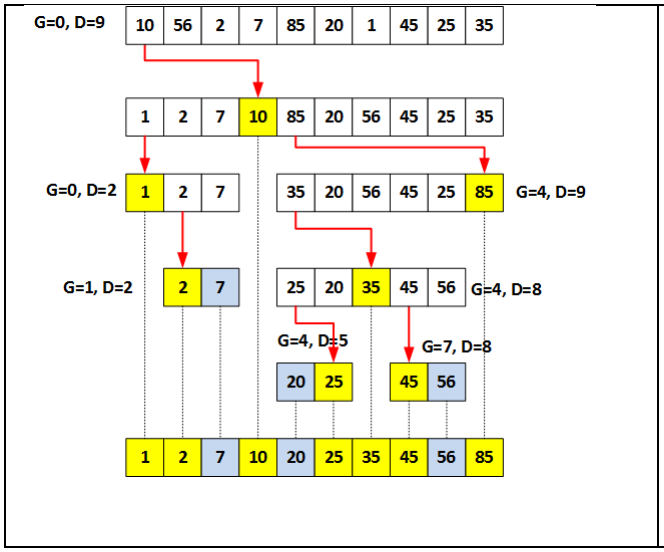
Après déplacement, les pivots occupent leur place définitive.

c) Récursivité

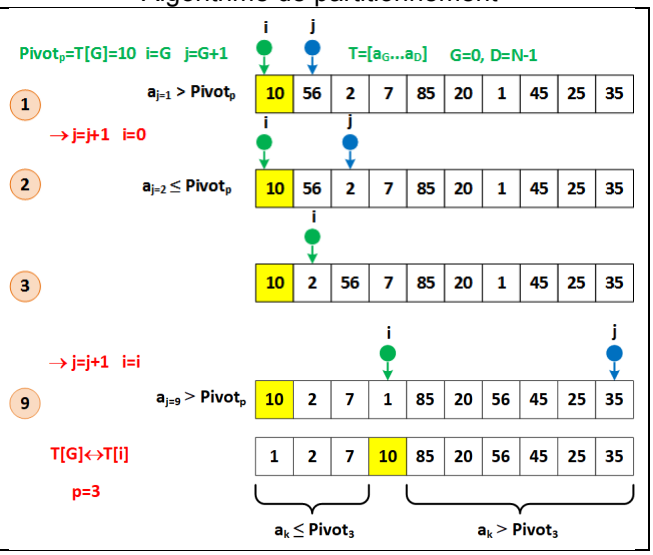
L'algorithme continue sur ce principe en considérant tout d'abord le sous-tableau situé à gauche du tableau dont le pivot $Pivot_0 = 1$ permet à nouveau de pratiquer le partitionnement des a_i avec $i \in [1,2]$ de part et d'autre de ce dernier. Ainsi :

- le premier sous-tableau de gauche est délimité par les bornes $G = 0,\ D = p - 1$ si $D > G$;
- le premier sous-tableau de droite est délimité par les bornes $G = p + 1,\ D = N - 1$ si $D > G$.

Récursivité



Algorithme de partitionnement



d) Algorithme de partitionnement

- un premier index initialisé $i = G$ permet de mémoriser l'emplacement du dernier $a_i \leq Pivot_p = 10$;
- un deuxième index initialisé $j = G + 1$ permet de parcourir tous les éléments du tableau afin de les comparer avec le pivot ;
- a_0 est choisi comme premier pivot, la valeur de l'index p n'est pas encore connue ;
- si $T[j] > Pivot_p$ alors $j = j + 1$;
- si $T[j] \leq Pivot_p$ alors permutation des éléments $T[j]\ et\ T[i+1]$ et $j = j + 1, i = i + 1$;
- à la fin des différentes comparaisons, le pivot a_0 est permuté avec le dernier élément déplacé puis l'index du pivot est retourné.

Modèle CMEN-DOC v2 ©NEOPTEC

Nom de famille :
(Suivi, s'il y a lieu, du nom d'usage)

Prénom(s) :

Numéro Inscription :

(Le numéro est celui qui figure sur la convocation ou la feuille d'émargement)

Né(e) le : ☐☐ / ☐☐ / ☐☐☐☐

(Remplir cette partie à l'aide de la notice)

Concours / Examen : …….. **Section/Spécialité/Série :** ……

Epreuve : …….. **Matière :** …….. **Session :** ……..

CONSIGNES

- *Remplir soigneusement, sur CHAQUE feuille officielle, la zone d'identification en MAJUSCULES.*
- *Ne pas signer la composition et ne pas y apporter de signe distinctif pouvant indiquer sa provenance.*
- *Numéroter chaque PAGE (cadre en bas à droite de la page) et placer les feuilles dans le bon sens et dans l'ordre.*
- *Rédiger avec un stylo à encre foncée (bleue ou noire) et ne pas utiliser de stylo plume à encre claire.*
- *N'effectuer aucun collage ou découpage de sujets ou de feuille officielle. Ne joindre aucun brouillon.*

DR1 - DR2 (1/2)

Tous les documents réponses sont à rendre, même non complétés.

NE RIEN ECRIRE DANS CE CADRE

Document réponse DR1 : Simulation scanner-Lidar

Question Q7

```
import math

class CLidar:
    def __init__(self,p_PRR,p_SR,p_FOV,p_gamma):
        self.__PRR=p_PRR
        self.__SR=p_SR
        self.__FOV=p_FOV*math.pi/180.0
        self.__gamma=p_gamma
        self.__deltaTheta=0.
        self.__omega=0.0
        self.__Nf=0.
    def getPRR(self):
        return(self.__PRR)
    def getSR(self):
        return(self.__SR)
    def getFOV(self):
        return(self.__FOV)
    def getGamma(self):
        return(self.__gamma)
    def getOmega(self):




        return(self.__omega)
    def getDeltaTheta(self):




        return(self.__deltaTheta)
    def getNf(self):




        return(self.__Nf)
```

```
class CDrone:
    #Paramètres:p_h:hauteur,p_v:vitesse,p_deltaT:durée,p_lidar:objet CLidar
    def __init__(self,p_h,p_v,p_deltaT,p_lidar):
        self.__h=p_h
        self.__v=p_v
        self.__deltaT=p_deltaT
        self.__Lidar=p_lidar
        self.__SW=0.
        self.__A=0.
        self.__Nl=0
        self.__Nf=0
        self.__density=0
        self.__widthSpace=0.
        self.__lengthSpace=0.
        self.__pointX=0
        self.__pointY=0
        self.__lPoints=[]
```

Question Q8

```
    def getSW(self):


        return(self.__SW)
    def getA(self):


        return(self.__A)
    def getNl(self):


        return(self.__Nl)
    def getDensity(self):


        return(self.__density)
    def getWidthSpace(self):


        return(self.__widthSpace)
    def getLengthSpace(self):


        return(self.__lengthSpace)

#Programme principal
lidar=CLidar(550e3,200,90,0.5e-3)
drone=CDrone(100,4,10,lidar)
```


Tournez la page S.V.P.

Document réponse DR2 (1/2) : Construction de l'image solide

Le programme Python ci-après correspond au diagramme d'activités permettant de construire l'image solide.
Les bibliothèques incluses ici sont considérées incluses pour tout le DR2.
Le programme principal appelant les différentes fonctions est localisé deux pages suivantes à partir de celle-ci (encadré « programme principal »).

Question Q17

```
import matplotlib.pyplot as plt
import numpy as np
import math
from time import process_time

def readCamTextFile(camFileName):
    fileCam=open(camFileName,'r')
    cam=fileCam.readlines()
    fileCam.close()
    camPos=np.array([[float(cam[0]),float(cam[1]),float(cam[2])]])
    camPos=np.transpose(camPos)
    camOrien=np.array([[float(cam[3]),float(cam[4]),float(cam[5])]])
    camOrien=(np.pi/180)*np.transpose(camOrien)
    f=float(cam[6])
    L_cmos=float(cam[7]); H_cmos=float(cam[8])
    ppx=float(cam[9]); ppy=float(cam[10])
    return(camPos,camOrien,f,L_cmos,H_cmos,ppx,ppy)

def read3dPointTextFile(pointFileName):
    fileNuagePoints=open(pointFileName,'r')
    linesNuage=fileNuagePoints.readlines()
    fileNuagePoints.close()
    N_Nuage=len(linesNuage)
    tabNuage=np.zeros((N_Nuage,4));tabVecNorm=np.zeros((N_Nuage,3))
    for i in range(len(linesNuage)):
        oneLine=linesNuage[i]
        tabNuage[i,0]=oneLine.split(' ')[0]









    return(tabNuage,tabVecNorm,N_Nuage)
```

Modèle CMEN-DOC v2 ©NEOPTEC

Nom de famille :
(Suivi, s'il y a lieu, du nom d'usage)

Prénom(s) :

Numéro Inscription :

Né(e) le : / /

(Le numéro est celui qui figure sur la convocation ou la feuille d'émargement)

(Remplir cette partie à l'aide de la notice)

Concours / Examen : …….. **Section/Spécialité/Série :** ……

Epreuve : …….. **Matière :** …….. **Session :** ……..

CONSIGNES

- *Remplir soigneusement, sur CHAQUE feuille officielle, la zone d'identification en MAJUSCULES.*
- *Ne pas signer la composition et ne pas y apporter de signe distinctif pouvant indiquer sa provenance.*
- *Numéroter chaque PAGE (cadre en bas à droite de la page) et placer les feuilles dans le bon sens et dans l'ordre.*
- *Rédiger avec un stylo à encre foncée (bleue ou noire) et ne pas utiliser de stylo plume à encre claire.*
- *N'effectuer aucun collage ou découpage de sujets ou de feuille officielle. Ne joindre aucun brouillon.*

DR2 (2/2) - DR3

Tous les documents réponses sont à rendre, même non complétés.

NE RIEN ECRIRE DANS CE CADRE

Document réponse DR2 (2/2) : construction de l'image solide

Question Q18

```
def matRotation(camOrien):
    matA=np.zeros((3,3))
    matA[0,]=np.array([1,0,0])
    matA[1,]=np.array([0,math.cos(camOrien[0,0]),math.sin(camOrien[0,0])])
    matA[2,]=np.array([0,-math.sin(camOrien[0,0]),math.cos(camOrien[0,0])])
```

```
    return(matR)
```

```
def matProjection(matR,camPos):
    matRt=np.zeros((3,1))
    matRt=-np.matmul(matR,camPos)
    matP=np.zeros((4,4))
```

```
    return(matP)
```

```
def matParamImage(ppx,ppy,sx,sy):
    matF=np.array([[-f,0,0,0],[0,f,0,0],[0,0,1,0]])
    matD=np.array([[1,0,ppx],[0,-1,ppy],[0,0,1]])
    matS=np.array([[sx,0,0],[0,sy,0],[0,0,1]])
    return(matF,matD,matS)
```

Programme principal

```
camPos,camOrien,f,L_cmos,H_cmos,ppx,ppy=readCamTextFile('img_4000_InfosCam.txt')
tabNuage,tabVec,N_Nuage=read3dPointTextFile('Carriere_ZoneEtude[2cm].asc')
#Taille de l'image Largeur, hauteur
L=1200;H=800
sx=L/L_cmos
sy=H/H_cmos
matR=matRotation(camOrien)
matF,matD,matS=matParamImage(ppx,ppy,sx,sy)
matP=matProjection(matR,camPos)

tabImage=np.zeros((N_Nuage,3))
u=np.zeros((N_Nuage,1))
v=np.zeros((N_Nuage,1))
dab=np.zeros((N_Nuage,1))
lpixels_dab=[]
index=0

for i in range(N_Nuage):
   tabImage[i,]=np.matmul(matF,np.matmul(matP,tabNuage[i,]))
   tabImage[i,]=np.matmul(matS,np.matmul(matD,tabImage[i,]))
   u[i,0]=round(tabImage[i,0]/tabImage[i,2])
   v[i,0]=round(tabImage[i,1]/tabImage[i,2])

   dab[i,0]=round(math.sqrt((camPos[0,0]-tabNuage[i,0])**2+(camPos[1,0]
tabNuage[i,1])**2+(camPos[2,0]-tabNuage[i,2])**2),3)

   if(u[i,0]<L and u[i,0]>=0 and v[i,0]<H and v[i,0]>=0):
      lpixels_dab.append([int(u[i,0]),int(v[i,0]),dab[i,0],index])
      index=index+1
```

Question Q25

Question Q26

```
#Lecture de l'image
imSrc=cv2.imread("IMG_4000.jpg")
#Stockage de l'image solide dans un tableau numpy à 4 éléments
dt=np.dtype((np.float32,(4)))
img=np.zeros((H,L),dtype=dt)
```


Tournez la page S.V.P.

Document réponse DR3 : Tri rapide et suppression des doublons

Question Q22

```
def partition_ag_simple(ldata,G,D):
    pivot=ldata[G];i=G









    return(i)

def quickSort_ag(ldata,G,D):
    if(G>=D):




    else:




ldab=[51.512,51.167,51.186,51.178,51.166,51.173,51.169,51.157,51.16,51.157]
quickSort_ag_simple(ldab,0,9)
```

Question Q23

```
def partition_ag(ldata,G,D):









    return(i)

lpixels_dab=[[1190,694,50.86,0],[1191,695,50.858,1],[1190,696,50.841,2],
             [1191,696,50.822,3],[1191,693,50.887,4],[1192,693,50.869,5],
             [1193,695,50.852,6],[1193,695,50.833,7],[1193,696,50.814,8],
             [1196,696,50.813,9]]
quickSort_ag(lpixels_dab,0,9)
```

Modèle CMEN-DOC v2 ©NEOPTEC

Nom de famille :
(Suivi, s'il y a lieu, du nom d'usage)

Prénom(s) :

Numéro Inscription :

(Le numéro est celui qui figure sur la convocation ou la feuille d'émargement)

Né(e) le : ☐☐ / ☐☐ / ☐☐☐☐

(Remplir cette partie à l'aide de la notice)

Concours / Examen : …….. **Section/Spécialité/Série :** ……

Epreuve : …….. **Matière :** …….. **Session :** ……..

CONSIGNES

- *Remplir soigneusement, sur CHAQUE feuille officielle, la zone d'identification en MAJUSCULES.*
- *Ne pas signer la composition et ne pas y apporter de signe distinctif pouvant indiquer sa provenance.*
- *Numéroter chaque PAGE (cadre en bas à droite de la page) et placer les feuilles dans le bon sens et dans l'ordre.*
- *Rédiger avec un stylo à encre foncée (bleue ou noire) et ne pas utiliser de stylo plume à encre claire.*
- *N'effectuer aucun collage ou découpage de sujets ou de feuille officielle. Ne joindre aucun brouillon.*

DR4 - DR5

Tous les documents réponses sont à rendre, même non complétés.

NE RIEN ECRIRE DANS CE CADRE

Document réponse DR4 : Programmation dynamique

Question Q32

	$j=-1$	$j=0$	$j=1$	$j=2$	$j=3$	$j=4$
$i=-1$						
$i=0$						
$i=1$						
$i=2$						
$i=3$						
$i=4$						

Question Q33

```
import numpy as np

L=[1,1,4,5,6]
R=[4,5,6,8,8]
N=len(L)
#Matrice des coûts
g=np.zeros((N+1,N+1))
#Stockage du choix du min
P=np.zeros((N+1,N+1))
#Coût maximal
lambda=50.
#Initialisation g(-1,) et g(,-1)
for i in range(N+1):




#Initialisation g(0,0)



#Calcul de la table des coûts
for j in range(1,N+1):
    for i in range(1,N+1):
```

Question Q34

```
#Parcours du chemin optimal
i=N
j=N
#Pixels communs et profondeur associée
d=[]
while(
```


Tournez la page S.V.P.

Document réponse DR5 : Fonction de convolution

Question Q37

```
def bords_enroulage(image):
    H,L=np.shape(image);imgR=np.zeros((H+2,L+2))

    for j in range(0,L):
        imgR[0,j+1]=image[H-1,j]
        imgR[H+2-1,j+1]=image[0,j]
    for i in range(0,H):
        imgR[i+1,0]=image[i,L-1]
        imgR[i+1,L+2-1]=image[i,0]

    imgR[0,0]=image[H-1,L-1];imgR[H+2-1,L+2-1]=image[0,0]
    imgR[0,L+2-1]=image[H-1,0];imgR[H+2-1,0]=image[0,L-1]

    for i in range(1,H+2-1):
        for j in range(1,L+2-1):
            imgR[i,j]=image[i-1,j-1]
    return(imgR)

def conv_y(image,h):
    H,L=np.shape(image)
    imgF=np.zeros((H-2,L-2))




    return(imgF)
```

L'image I et le noyau de convolution h sont donnés ci-dessous.

$I =$

13	92	64	10	28	55	96	97	16	98
96	49	81	15	43	92	80	96	66	4
85	94	68	76	75	40	66	18	71	4
28	5	10	83	70	32	96	4	44	39
77	80	19	49	45	65	71	76	28	68
66	17	12	50	96	35	59	23	76	26
51	70	90	96	55	14	15	26	85	26

$h =$

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

Suite question Q37 (compléter le tableau y)

$y =$

			53.55	45.33	53.11	63.44	64.11	57.11	50.55
			51.11	48.22	63.88	71.11	67.33	52.22	50.33
			57.88	58.44	66	58.22	60.11	38.44	48.55
53.33	51.77	53.77	55	59.44	62.22	52	52.66	39.11	49.33
45.11	34.88	36.11	48.22	58.33	63.22	51.22	53	42.66	50.22
53.44	53.55	53.66	56.88	56.11	50.55	42.66	51	48.22	55.88
51	52.77	55.66	55.66	48.77	50.33	46.66	54.77	52.55	50.77

Modèle CMEN-DOC v2 ©NEOPTEC

Nom de famille :
(Suivi, s'il y a lieu, du nom d'usage)

Prénom(s) :

Numéro Inscription :

(Le numéro est celui qui figure sur la convocation ou la feuille d'émargement)

Né(e) le : ☐☐/☐☐/☐☐☐☐

(Remplir cette partie à l'aide de la notice)

Concours / Examen : …….. **Section/Spécialité/Série :** ……

Epreuve : …….. **Matière :** …….. **Session :** ……..

CONSIGNES

- *Remplir soigneusement, sur CHAQUE feuille officielle, la zone d'identification en MAJUSCULES.*
- *Ne pas signer la composition et ne pas y apporter de signe distinctif pouvant indiquer sa provenance.*
- *Numéroter chaque PAGE (cadre en bas à droite de la page) et placer les feuilles dans le bon sens et dans l'ordre.*
- *Rédiger avec un stylo à encre foncée (bleue ou noire) et ne pas utiliser de stylo plume à encre claire.*
- *N'effectuer aucun collage ou découpage de sujets ou de feuille officielle. Ne joindre aucun brouillon.*

DR6

Tous les documents réponses sont à rendre, même non complétés.

NE RIEN ECRIRE DANS CE CADRE

Document réponse DR6 : Fonction DoG

Question Q46

```
#fonction qui calcule le masque gaussien 1D en fonction de x et sigma
def g1D(x,sigma):


  return(


#fonction qui calcule le masque gaussien de taille p, fonction de sigma, en 1D
def masque1D(sigma,p):









  return(

#fonction qui calcule L(x,y,sigma)
def conv_L(image,sigma,p):









  return(
```

Suite question Q46

```
#fonction qui calcule DoG(x,y,sigma)
def DoG(image,k,sigma,p):









  return(img_dog)

def DoGN(image,N,k,sigma,p):
    H,L=np.shape(image)
    #Stocke les valeurs maxi des pixels de la même octave
    maxi=np.zeros((H,L))
    #Stocke les N dog
    dog_liste=[]









    return(


def DoG_MR(image,N,k,sigma,p,nO):
    H,L=np.shape(image)
    dogmr_liste=[]









    return(
```