



Construction de panorama par recalage

Points remarquables, Descripteurs et Homographie

Implémentation Matlab ou Python

Projet de Cours INF8725



Table des matières

1	Introduction	3
1.1	Un peu de contexte	3
1.2	Objectifs	4
1.3	Rendus et matériel	4
1.3.1	Librairies autorisées	5
2	Détection des extrema et descripteurs	6
2.1	Construction des différences de Gaussiennes – <i>Barème</i>	0
2.2	Détection des points clés – <i>Barème</i> : 7	7
2.2.1	Descripteur – <i>Barème</i> : 5	8
3	Matching et homographie	10
3.0.1	Recherche de couples amis – <i>Barème</i> : 3	10
3.0.2	Homographie – <i>Barème</i> : 1	11



1. Introduction

Lors de votre travail préparatoire, vous vous êtes familiarisé avec la théorie de la détection de points caractéristiques SIFT au sein d'une image (Scale invariant feature transform). A chaque point caractéristique, un descripteur est associé, supposé capturer les caractéristiques discriminantes de ce point dans l'image.

Une des applications de ces descripteurs est la construction de panorama : l'ensemble des descripteurs d'une image sert à recaler cette image vers une image associée. Pour chaque descripteur de l'image de départ, on cherche celui qui lui est le plus proche dans l'image d'arrivée. Cela permet d'obtenir un ensemble de couple de points, faisant l'association entre les images. Les couples correspondent généralement aux même objets, présents dans les deux images mais acquis sous différentes conditions (position de prise de vue ou lumière différente, présence de niveaux de bruits, etc...).

A partir de l'analyse de ces couples de points, il est possible de construire une matrice de transformation, permettant d'assembler les deux images en une seule, en assurant une continuité naturelle à la jonction. Ce document va vous guider dans l'implémentation de ces différentes étapes.

1.1 Un peu de contexte

Le recalage d'image trouve des applications dans de très nombreux domaines, de l'imagerie médicale pour les images multi-modales à la création de mosaïques d'images (panoramas).

Nous vous proposons une application concrète de recalage, issue d'une problématique réelle : celle de la construction de panorama du paysage martien à partir d'images obtenus avec un champs de vision restreint.

Le 26 novembre 2011, le centre JPL, associé à l'agence spatiale américaine envoie une sonde spatiale vers Mars. A son bord se trouve un petit rover de près de 900kg : le désormais célèbre Curiosity. La descente et l'atterrissement (*et non "l'amarssissage"*) ont lieu le 6 août 2012. Depuis, le rover, bardé d'instruments scientifiques, a parcouru près de 14km sur le sol martien, avant d'achever sa mission le 24 juin 2014, au bout de 687 jours terrestres (669 jours martiens).

Fixé à un mât au dessus du rover, MASTCAM est un ensemble de deux caméras, chacune dotée

de focale fixe différente. L'une d'elle, MASTCAM 34 (focale de 34 mm), couvrant un champs de $15^\circ \times 15^\circ$, est capable de réaliser un panorama de 360° par l'assemblage de 150 photos, en environ 25min. Chaque image est acquise à la résolution 1200×1600 pixels, acquise au format RAW et compressée en format JPEG. L'image couleur (trois canaux, RGB) est obtenue en une seule acquisition. MASTCAM a été déployé au deuxième jour (martien) de la mission, le 8 août.

1.2 Objectifs

Ce projet vous invite à implémenter une méthode de reconstruction de panorama à partir de deux photographies prises par Curiosity. La méthode est imposée : vous devez utiliser celle décrite par Lowe dans la revue *International Journal of Computer Vision* (il s'agit de l'article que vous avez analysé en travail préparatoire). Les différentes étapes sont :

- L'implémentation de la détection d'extrema dans l'espace des échelles.
- La construction des descripteurs associés à chaque extrema.
- La recherche des points de correspondance entre les descripteurs des deux images que nous souhaitons fusionner.
- La construction d'une matrice d'homographie permettant de faire correspondre les deux images.

L'objectif final est de recomposer un panorama à partir des deux photographies suivantes :



(a) Première photographie de Curiosity



(b) Seconde photographie de Curiosity

FIGURE 1.1 – A l'origine, les deux photographies proviennent d'un panorama constitué par Curiosity. Il a été artificiellement redécoupé. Des opérations de transformations de couleurs, d'addition de bruit et une homographie ont été appliquées à l'image de droite. Cela va permettre d'éprouver la robustesse des descripteurs SIFT

1.3 Rendus et matériel

Dans le cadre du projet, il vous est imposé de rendre un rapport par binôme (maximum 15 pages, format *pdf* obligatoire), ainsi que l'intégralité du code. Le code doit être bien commenté, y compris s'il ne fonctionne pas. Veuillez remettre tous vos fichiers dans un seul fichier zip et nommez ce fichier selon vos matricules (Mat1_Mat2.zip).

A chaque étape, un certain nombre de figures vous seront demandées. Ce présent document vous indiquera les spécificités attendues, ainsi que des questions auxquelles vous devez répondre. Insérez les figures obtenues (légendées et nommées) et vos réponses dans votre rapport.

La notation prendra en compte l'effort de recherche, y compris si l'implémentation ne fonctionne pas. C'est pourquoi il vous est demandé de détailler vos essais, furent-ils non concluants.

Vous avez la liberté de choisir entre le langage Python ou MatLab.

1.3.1 Librairies autorisées

Aucune librairie/toolbox n'est imposée : vous êtes libre de choisir celles de votre choix pour un certain nombre d'opérations standards. Voici une liste **exhaustive** de ces opérations de traitement d'images dont l'implémentation n'est pas demandée et que vous pouvez utiliser librement :

- La lecture et l'écriture d'image.
- Les méthodes de dessins de formes géométriques usuelles (traits, flèches, cercle, etc...).
- La convolution 2D.
- La décomposition d'une matrice en valeurs singulières et la diagonalisation de matrice. Matlab et numpy proposent tous deux des algorithmes performants pour ces opérations.

Toutes autres opérations dont vous aurez besoin ne peuvent s'appuyer que sur votre propre code et l'utilisation judicieuse de ces précédentes méthodes.

En Python, toutes les matrices doivent pouvoir être interprétées par la librairie de référence **numpy**.

Références

 Pour plus d'information sur le Rover Curiosity, le site de la NASA regorge de ressources sur le sujet. Celles-ci sont synthétisées sur la page wikipedia du *Mars Science Laboratory*

2. Détection des extrema et descripteurs

2.1 Construction des différences de Gaussiennes – Barème : 4

La première étape consiste à construire la pyramide de gaussiennes.

La construction de la pyramide des DoG doit se faire au sein d'une fonction répondant à la nomenclature suivante :

```
fonction differenceDeGaussiennes(Mat image_initiale,  
                                Scal s,  
                                Scal nb_octave)  
  
...  
  
renvoie Liste des matrices DoG,  
        Vect des sigma correspondant
```

Question 1

Tracez la pyramide de gaussienne construite à l'octave de votre choix. Vous utiliserez au choix l'une des images fournies en ressources. (Voir 2.1)

Question 2

Tracez la différence de Gaussiennes correspondante. Quel type de filtre classique cela vous rappelle-t-il ? (Voir 2.2)



FIGURE 2.1 – Exemple de résultat attendu. La pyramide est obtenue pour $s = 3$. La légende indique le σ correspondant.

2.2 Détection des points clés – Barème : 7

La détection des points clés doit se faire au sein d'une fonction répondant à la nomenclature suivante. Cette fonction est appelée une fois par octave :

```
fonction detectionPointsCles(Mat DoG,
    Vect sigma, Scal seuil_contraste,
    Scal r_courb_principale,
    Scal resolution_octave)

    ...

renvoie Liste de points pour une octave.
```



Dans l'article de référence, les auteurs suggèrent de corriger la position d'un extremum trouvé en faisant un développement de Taylor à l'ordre 2 sur un espace tridimensionnelle (le point extremum correspondant à un lieu de dérivée nulle). Cette interpolation est une manière de corriger le biais de position de chaque extremum.

L'implémentation de cette interpolation n'est pas demandée, vous pouvez donc simplement sauter cette étape. En revanche, tout pseudo-code et/ou code proposant une implémentation de la méthode sera examiné et pourra donner lieu à un point bonus.

Le contraste sera donc évalué au point $D(x)$ et non $D(\hat{x})$

Question 1

Comment calculer de façon efficace (donc sans boucle for) la Hessienne pour tous les pixels de l'image ?

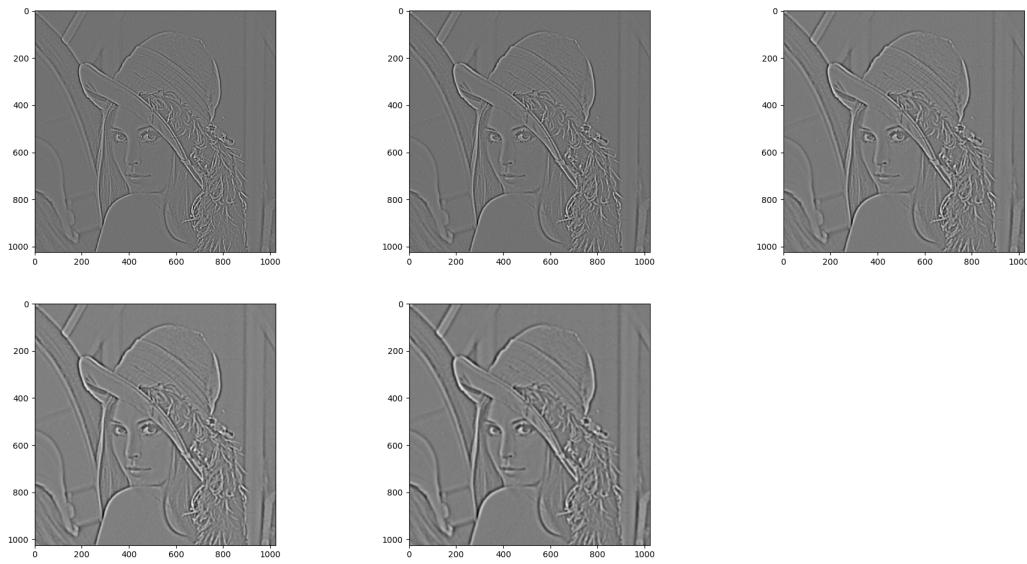


FIGURE 2.2 – Exemple de résultat attendu pour le tracé des différences de Gaussiennes

Question 2

Créez trois compteurs comptabilisant respectivement le nombre total d'extrema détectés, le nombre de point de faibles contrastes éliminés et le nombre de points d'arrêtes éliminés. Reportez les valeurs obtenues pour chaque octave dans un tableau.

Question 3

Tracez l'évolution du nombre de points conservés k_i (*keypoints*) dans l'octave i en fonction de la résolution de l'octave.

- R** Pour chaque point, vous devriez obtenir un vecteur de dimension 4 de la forme : **[#ligne, #colonne, échelle, angle]**. En posant $N_{img} = \sum_i k_i$, stockez vos points caractéristiques dans une matrice de taille $N_{img} \times 4$

Question 4

Pour chacune des images, joignez la matrice obtenue au .zip que vous rendrez. **Format : .mat (Matlab) ou .npy (Python)**

Question 5

Sur l'image de votre choix (parmi celles fournies), tracez les lieux de chaque point clé, tel que montré sur la figure 2.3.

2.2.1 Descripteur – *Barème : 5*

Avec les valeurs données dans l'article, à chaque point clé, vous devriez obtenir un descripteur de taille 128. Créez un vecteur de 130 éléments. Les deux premiers éléments seront réservés à la position du point clé **[#ligne, #colonne]**. Les 128 restants seront alloués au descripteur. Pour chaque point, vous obtenez ainsi un vecteur **[#ligne, #colonne, descripteur]**. Stockez chacun des vecteurs

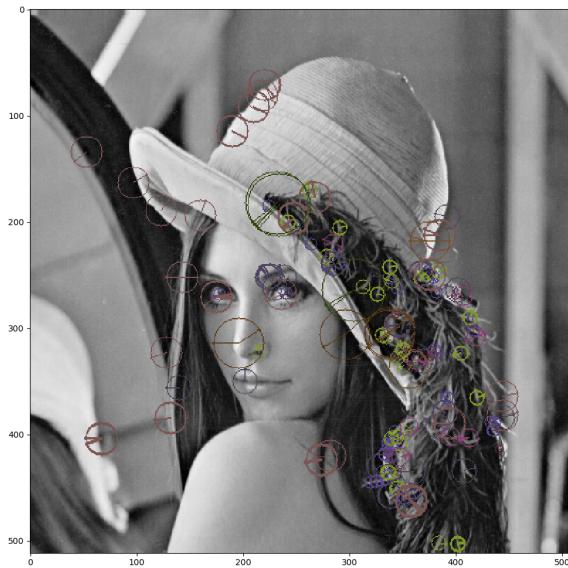


FIGURE 2.3 – Exemple de résultat attendu : le rayon du cercle correspond à l'échelle du point clé, la flèche marque la direction d'orientation principale du point.

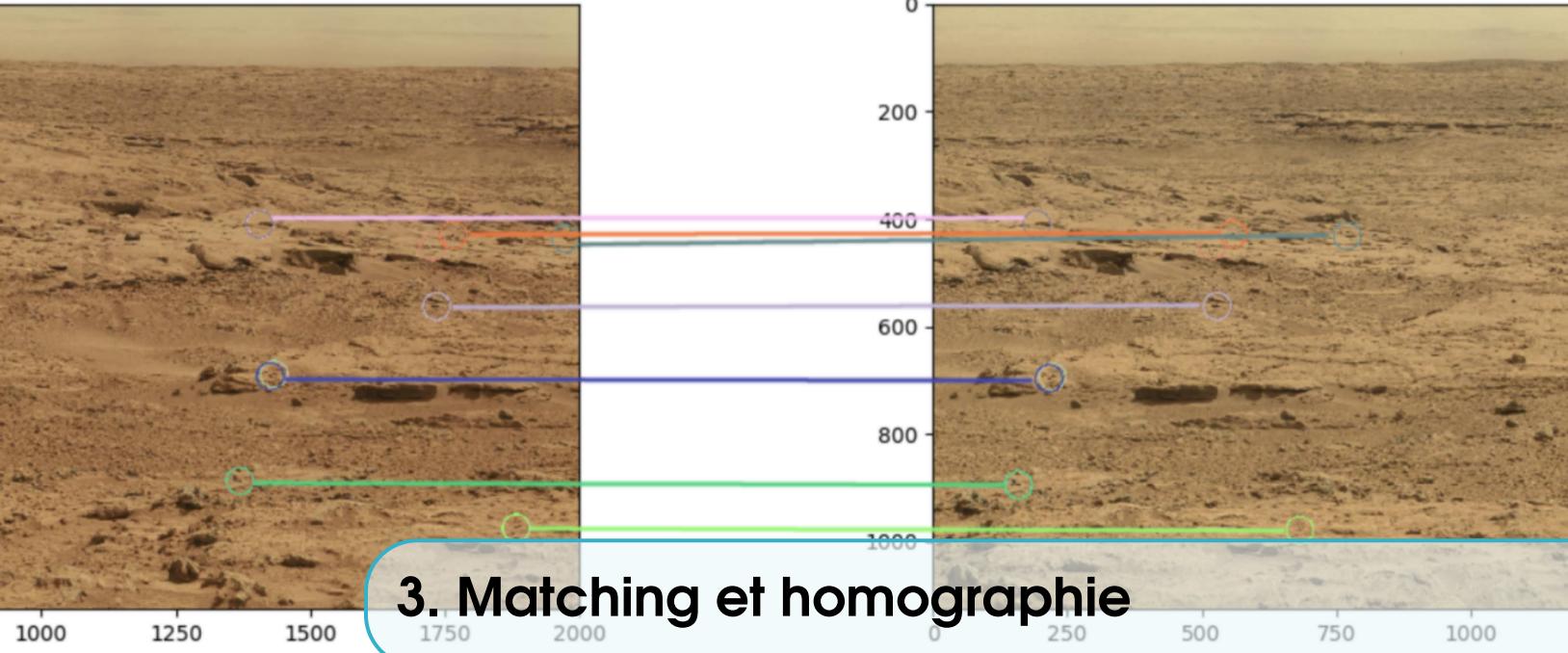
dans une matrice de taille $N_{img} \times 130$. Pour chacune des images, joignez la matrice obtenue au .zip que vous rendrez. **Format : .mat (Matlab) ou .npy (Python)**

La construction du(descripteur) doit se faire au sein d'une fonction répondant à la nomenclature suivante.

```
fonction descriptionPointsCles(Mat image_initiale,  
Liste points_cles)
```

```
...
```

```
renvoie Mat de descripteurs.
```



3. Matching et homographie

3.0.1 Recherche de couples amis – Barème : 3

Nous appelons "couple ami" tout couple de points formé par un point de chaque image dont les descripteurs sont proches. La notion de proximité se traduit généralement en terme de distance. Celles que nous vous proposons d'implémenter est la plus simple possible : il s'agit de la distance euclidienne, définie par :

$$d_{m,n} = \sqrt{\sum_i^{128} (descp_{img_1}^{(m)}(i) - descp_{img_2}^{(n)}(i))^2} \quad (3.1)$$

Question 1

Pour chaque descripteur de l'image 1, calculez sa distance euclidienne avec chacun des descripteurs de l'image 2. Tracez la matrice D.

La construction de la matrice de distance doit se faire au sein d'une fonction répondant à la nomenclature suivante.

```
fonction distanceInterPoints(Mat points_image1,
Mat points_image2)

    ...

renvoie Mat de distance
```

Question 2

Récupérez les n couples de points de distance minimales (c'est à dire les n plus petites distances de la matrice). Sur chaque image, tracez le lieu des points ainsi obtenus. Vérifiez visuellement que vous obtenez une correspondance entre les deux images.

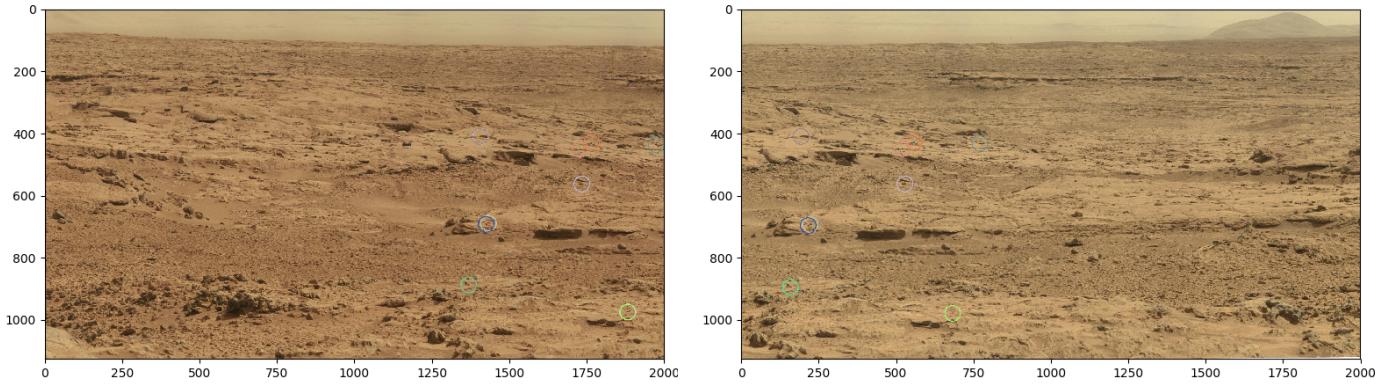


FIGURE 3.1 – Exemple de résultat attendu. Chaque couleur correspond à un couple ami.

3.0.2 Homographie – Barème : 1

Une homographie est une application projective bijective. Derrière ce formalisme mathématique, une homographie est une opération de transformation géométrique qui conserve la structure d'une image : une droite dans l'espace de départ restera une droite dans l'espace d'arrivée. De façon succincte, une homographie peut se résumer en une opération de transformation de coordonnées :

$$\omega \cdot \mathbf{v}_i = \begin{pmatrix} \omega x'_i \\ \omega y'_i \\ \omega \end{pmatrix} = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{pmatrix} \cdot \begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix} = \mathbf{H} \cdot \mathbf{u}_i \quad (3.2)$$

Pour un ensemble de n couples **uniques** $\{(\mathbf{u}_1, \mathbf{v}_1), \dots, (\mathbf{u}_i, \mathbf{v}_i), \dots, (\mathbf{u}_n, \mathbf{v}_n)\}$, on obtient les coefficients en résolvant :

$$\left| \begin{array}{ccccccccc} x_1 & y_1 & 1 & 0 & 0 & 0 & -x'_1 x_1 & -x'_1 y_1 & -x'_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -y'_1 x_1 & -y'_1 y_1 & -y'_1 \\ & & & \cdots & & & & & \\ & & & \cdots & & & & & \\ x_n & y_n & 1 & 0 & 0 & 0 & -x'_n x_n & -x'_n y_n & -x'_n \\ 0 & 0 & 0 & x_n & y_n & 1 & -y'_n x_n & -y'_n y_n & -y'_n \end{array} \right| \cdot \begin{pmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ h_{33} \end{pmatrix} = \mathbf{A} \cdot \mathbf{h} = \mathbf{0} \quad (3.3)$$

La notation $\mathbf{0}$ désigne le vecteur nul, et non un scalaire.

L'homographie étant défini à un facteur d'échelle près, il est possible (sans perte de généralité) de fixer un élément h_{ij} . En général, on fixe $h_{33} = 1$.

Ainsi défini, la détermination des 8 inconnues restantes ne nécessite a priori que quatre couples amis et une solution linéaire non-homogène peut être trouvée.

S'il y a plus de 4 couples amis, le problème se ramène à minimisation homogène de la quantité $\|\mathbf{A} \cdot \mathbf{h}\|^2$, avec la contrainte $|\mathbf{h}| = 1$ (pour éviter la solution triviale où tous les coefficients seraient nuls).

On peut montrer (voir remarque qui suit) que le vecteur \mathbf{h} satisfaisant cette condition est égale au vecteur propre de la matrice $\mathbf{A}^T \cdot \mathbf{A}$ associée à la plus petite valeur propre.

R Une façon intuitive de comprendre le lien entre vecteur propre de la matrice $\mathbf{A}^T \cdot \mathbf{A}$ est de voir le problème sous la forme d'une optimisation quadratique. On cherche \mathbf{h} tel que la distance f entre $\mathbf{A} \cdot \mathbf{h}$ et le vecteur nul soit minimale.

$$f(\mathbf{h}) = \frac{1}{2}(\mathbf{A} \cdot \mathbf{h} - \mathbf{0})^T \cdot (\mathbf{A} \cdot \mathbf{h} - \mathbf{0}) = \frac{1}{2}\mathbf{h}^T \mathbf{A}^T \mathbf{A} \mathbf{h} \quad (3.4)$$

Pour cela, on va annuler la dérivée de cette distance, prise par rapport à \mathbf{h}

$$\frac{df}{d\mathbf{h}} = 0 \quad (3.5)$$

$$\Leftrightarrow \frac{1}{2}(\mathbf{A}^T \mathbf{A} + (\mathbf{A}^T \mathbf{A})^T) \mathbf{h} = 0 \quad (3.6)$$

$$\Leftrightarrow \mathbf{A}^T \mathbf{A} \cdot \mathbf{h} = 0 = \lim_{\varepsilon \rightarrow 0} \varepsilon \cdot \mathbf{h} \quad (3.7)$$

En approximation du point d'annulation de la dérivée, on peut donc considérer \mathbf{h} comme le vecteur propre de $\mathbf{A}^T \mathbf{A}$ possédant la plus petite valeur propre.

L'explication donnée ici n'est qu'une description intuitive de la solution. Une démonstration rigoureuse se ramène en fait à un problème de minimisation utilisant des multiplicateurs de Lagrange

R On peut également obtenir ce vecteur propre en prenant le dernier vecteur singulier de la matrice d'analyse \mathbf{V} , obtenue par décomposition en valeurs singulières et définie par $\mathbf{A} = \mathbf{U} \mathbf{S} \mathbf{V}^T$, si on suppose que les valeurs singulières sont triées par ordre décroissant.

A nouveau, un vecteur propre étant définie à un facteur constant près (comme une homographie !), nous pouvons satisfaire la contrainte $h_{33} = 1$. Ceci est usuellement le cas pour une homographie. ($\hat{\mathbf{h}} = \mathbf{h}/h_{33}$).

En général, l'implémentation se fait selon le modèle :

Entrée : Matrice \mathbf{A} de taille $2n \times 9$

Sortie : Matrice \mathbf{H} de taille 3×3

```

 $A_{quadratique} \leftarrow \mathbf{A} \cdot \mathbf{T} \cdot \mathbf{A}$ 
 $valeursPropres \leftarrow \text{ValeursPropres}(A_{quadratique})$ 
 $vecteursPropres \leftarrow \text{VecteursPropres}(A_{quadratique})$ 
 $H_{flatten} \leftarrow \text{vecteursPropres}[:, \text{argmin}(valeursPropres)]$ 
 $H_{flatten\&norm} \leftarrow H_{flatten}/H_{flatten}[end]$ 
 $H_{norm} \leftarrow \text{Réordonner}(H_{flatten}, (3, 3))$ 

```

Renvoyer H_{norm}

Algorithm 1: Obtention de la matrice de transformation à partir des vecteurs propres. Celle-ci est normalisée de telle sorte que sa dernière valeur soit unitaire.

Question 1

Vérifiez numériquement que le dernier vecteur singulier de la décomposition SVD est bien égal au plus petit vecteur propre de la matrice carrée $\mathbf{A}^T \mathbf{A}$. La première étape consiste à transformer l'ensemble des couples amis sous la forme de la matrice \mathbf{A} . Pour n couples, vous devriez obtenir une matrice \mathbf{A} de taille $2n \times 9$.

Question 2

Comparez le temps d'exécution de la méthode de décomposition SVD et celle de décomposition en vecteurs propres. Qu'en concluez-vous ?

Question 3

Une fois la matrice H obtenue, la suite logique est de calculer la projection de l'image de droite sur l'image de gauche, ce qui nécessite une certaine réflexion sur le positionnement des coordonnées de l'image transformée.

La reconstruction du panorama doit respecter un certain nombre d'étapes. La première est d'inverser l'homographie calculée, car nous souhaitons l'image de droite sur celle de gauche. Pour reconstruire l'image complète, il est également nécessaire de calculer la dimension finale. Il suffit pour cela de calculer les coordonnées des points extrêmes de l'image de droite, après transformation. Si l'image de droite est de dimension $(x_{max_{droite}}, y_{max_{droite}})$, ces coordonnées s'obtiennent par :

$$(x_{max_{final}}, y_{max_{final}}) = H \cdot (x_{max}, y_{max}, 1)$$

L'image de gauche est mappée dans les intervalles $[0 : x_{max_{gauche}}, 0 : y_{max_{gauche}}]$ de l'image transformée.
Reconstruisez le panorama.

Question 4 Bonus

Si tout se passe bien, vous avez pu reconstruire le panorama suivant :

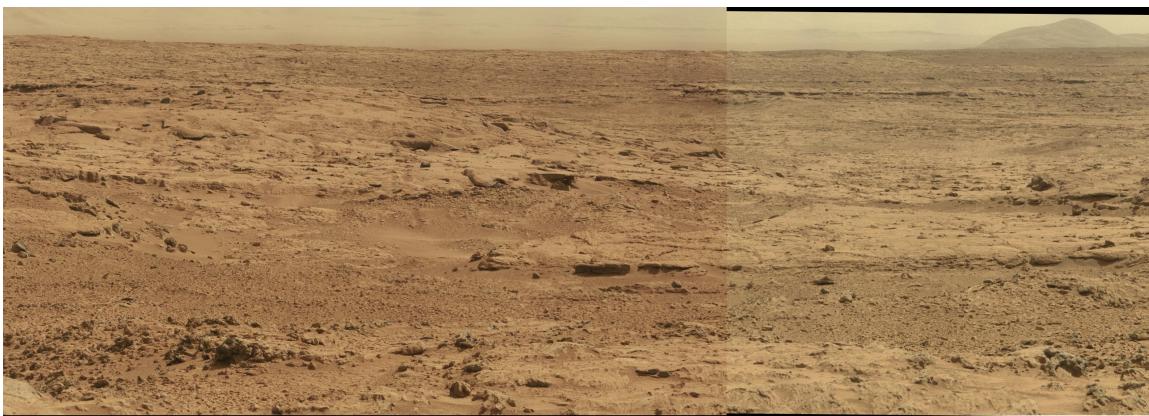


FIGURE 3.2 – Panorama martien reconstitué à partir de deux photographies du rover Curiosity. Remarquez que nous n'avons pas corrigé les différences d'exposition entre les photographies. En revanche, la continuité de la géométrie est préservée le long de la frontière.

Proposez une méthode pour corriger (automatiquement !) la différence d'exposition entre les deux images. *L'implémentation n'est pas demandée, vous avez donc le droit d'utiliser des fonctions pré-existantes si vous souhaitez vérifier votre hypothèse*



FIGURE 3.3 – Résultat final !