
Projet

POLYTECHNIQUE
MONTRÉAL

LE GÉNIE
EN PREMIÈRE CLASSE



Construction de panorama par recalage
Points remarquables, descripteurs et homographie

INF8725 - Traitement de signaux et d'images

Automne 2017
Département de génie informatique
École Polytechnique de Montréal

Dernière mise à jour: 7 décembre 2017

Baptiste Levasseur	1856305
Theo Rubenach	1800314

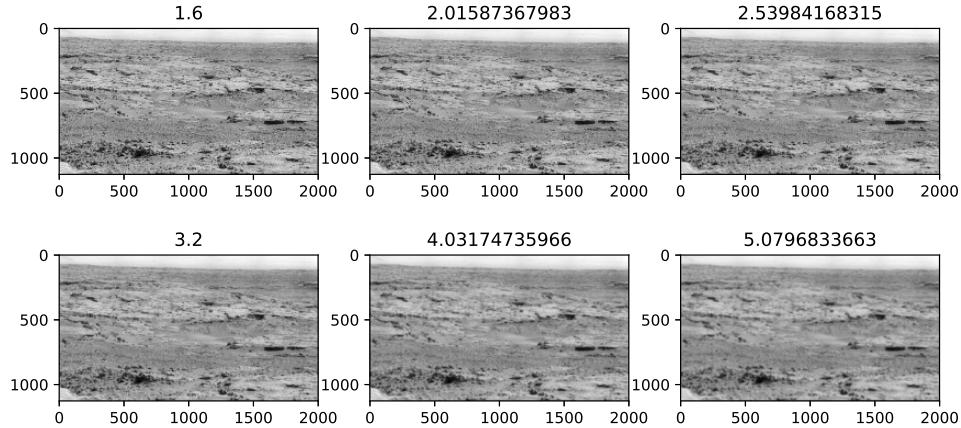


FIGURE 1 – Exemple de pyramide de gaussienne obtenue pour $s = 3$. On trace ici la première octave. on voit que l'image es de plus en plus flou, comme attendu.

1 Détection des extrema et descripteurs

1.1 Construction des différences de gaussiennes

1.1.1 Question 1

Dans une première partie, il est nécessaire de construire l'espace des échelles qui sera utilisé pour détecter les points clés de l'image. On procède donc dans un premier temps par la détermination de la pyramide de gaussienne pour une octave précise. On reproduira alors cette fonction pour les différentes octaves. Cette fonction prend donc en argument l'image, le nombre d'intervalles final et le numéro de l'octave sur lequel on travaille selon la signature suivante :

```
def pyramideDeGaussiennes(image,s,no_octave):
```

On obtient en sortie la liste de $s + 3$ fonctions convolées avec masque gaussien de taille $n = 21$ avec un σ croissant commençant à 1.6. ($\sigma_i = 1.6 * k^i$ avec $k = 2^{1/2}$ pour $i = 0..s + 3$).

En traçant alors le résultat pour l'octave numéro 1, on obtient alors les résultats suivants pour l'image `gauche.jpg`, on obtient alors les résultats suivants visibles sur la figure 1.

Dans la suite de l'article, on utilisera toujours l'image `gauche.jpg`

1.1.2 Question 2

Après avoir calculé la pyramide de $s + 3$ gaussiennes pour obtenir $s + 1$ intervalles finaux, il faut calculer la différence de gaussiennes. Contrairement à la pyramide de gaussiennes, celle-ci renvoie une liste de `numpyarrays` où l'array i correspond aux $s + 2$ différences de gaussiennes pour l'octave i .

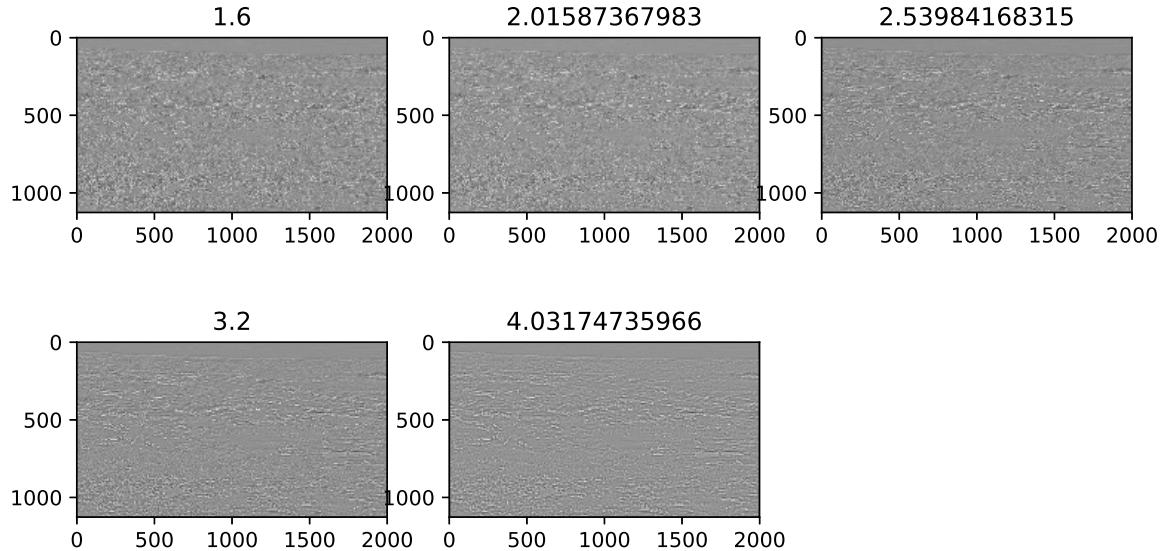


FIGURE 2 – Exemple de différence de gaussiennes sur la même octave que précédemment. Les premiers étages affichent davantage les petits détails, les étages supérieurs des tendances plus générales.

En traçant alors le résultat encore une fois pour la première octave, on obtient les résultats de la figure 2. Le résultat rappelle grandement le résultat d'un filtre passe-haut, puisque seules les arêtes (à peu près) sont visibles.

Au terme de cette fonction, on obtient donc les $s + 2$ différences de gaussiennes pour chacune des n octaves.

1.2 Détection des points clés

1.2.1 Question 1

Le calcul de la hessienne peut se faire simplement en codant préalablement la fonction `gradient(image)` : qui pour une image $n \times m$ renvoie deux images $n \times m$ en utilisant les différences finies.

La fonction `gradient(image)` : calcule pour tous les éléments intérieurs de l'image les différences finies d'ordre 2 :

$$\frac{\partial f}{\partial x} = \frac{f(x+1) - f(x-1)}{2}$$

Pour les limites à gauche de l'image, on utilise des différences finies d'ordre 1 :

$$\frac{\partial f}{\partial x} = f(x+1) - f(x)$$

On fait de même pour les éléments sur les limites à droite de l'image.

Finalement, le gradient fournit en sortie 2 matrices de taille $n \times m$ qui sont la dérivée en x ainsi que la dérivée en y

Pour le calcul de la hessienne, il suffit alors d'appliquer cette même fonction de gradient à chacune des composantes du gradient de l'image. On obtient alors finalement 4 matrices de taille $n \times m$ qui correspondent à la matrice hessienne :

$$H = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{yx} & D_{yy} \end{bmatrix}$$

Remarque : La fonction hessienne a été adaptée pour fonctionner avec des matrices de dimension 3 afin de pouvoir calculer le gradient et la hessienne de la différence de gaussienne (nécessaire pour le développement de Taylor)

1.2.2 Question 2

Il faut désormais à partir de la différence de gaussienne pour tous les octaves calculer les différents points clés pour chacune de ces octaves. Pour chaque octave, on va dans un premier temps chercher les extrema en parcourant tous les pixels de la matrice de différence de gaussienne pour l'octave en question de taille $n \times m \times s + 2$ et le comparer avec ses 26 voisins (dans les 3 dimensions). La fonction qui permet de faire cette opération est :

`def detectionExtrema(DoG):`

A la fin de cette étape, on obtient alors un grand nombre de points clés qu'il va falloir élaguer. Pour ce faire, on va faire passer ces points clés par plusieurs fonctions discriminantes :

Elimination des points à faible contraste : On vérifie que la valeur de la différence de gaussienne de l'extrema est supérieur à 0.03. Pour améliorer la stabilité, on peut utiliser l'expansion de Taylor à 3 dimensions :

$$D(\mathbf{x}) = D + \frac{\partial D}{\partial \mathbf{x}}^T \mathbf{x} + \frac{1}{2} \mathbf{x}^T \frac{\partial^2 D}{\partial \mathbf{x}^2} \mathbf{x}$$

où $\frac{\partial D}{\partial \mathbf{x}}$ est le gradient de la fonction 3D pris en \mathbf{x} et $\frac{\partial^2 D}{\partial \mathbf{x}^2}$ est la matrice Hessienne de la fonction 3D pris \mathbf{x} .

Comme évoqué précédemment, les fonctions gradient et hessienne à 3 dimensions ont déjà été implémentées. Une tentative de calcul de l'offset avec une correction des points clés et une évaluation de la différence de gaussienne interpolée a été réalisée conformément aux indications du code. Cependant, le calcul de l'offset $\hat{\mathbf{x}}$ renvoyait des valeurs trop grandes malgré une limitation du calcul pour des déterminants suffisamment grands (numpy.linalg.inv calculant l'inverse d'une matrice même si celle-ci n'est pas inversible). Nous avons donc décidé de laisser de coté cette partie même si la correction des offsets même supérieurs à 1 a été implémentée.

Elimination des arêtes : Pour l'élimination des arêtes, le calcul de la hessienne en 2D a été repris pour vérifier la condition suivante :

$$\frac{Tr(\mathbf{H})^2}{Det(\mathbf{H})} < \frac{(r+1)^2}{r}$$

Suppression des points clés près des bords : Pour pouvoir appliquer la transformation pour le calcul des descripteurs, il est nécessaire d'éliminer les pixels sur les bords. De plus, il faut prévoir la rotation de l'image lors du changement de référentiel de la zone du descripteur. On élague donc les bords avec un rayon $n_{zone} \times n_{pixel,zone} \times \sqrt{2}/2$

On obtient finalement l'évolution suivante du nombre de points clés :

Octave	Extrema Local	Faibles contrastess	Points d'arêtes	Bords	Nombre Total
1	21811	19775	573	336	1127
2	5052	4417	245	161	229
3	1071	864	84	76	47
4	331	185	80	60	6

1.2.3 Question 3 et 4

Après avoir obtenus les différents points clés, on va augmenter leur nombre en définissant une orientation pour chacun d'entre à partir de l'histogramme comme décrit dans l'énoncé :

```
def orientationPointCle(point_cle,L,sigma_list):
```

On obtient finalement l'évolution du nombre de points clés pour les différentes octaves :

Le fichier `PointsCles.npy` contient alors ces différents points clés.

1.2.4 Question 5

On trace désormais le résultat des points clés orientés pour l'image de gauche (visible sur la figure 3)

On observe alors que plus l'échelle est élevée (c'est à dire plus on en bas dans l'espace d'échelle), plus les points clés vont représenter des éléments grossiers (comme des cailloux) tandis que plus l'échelle est petite, plus les descripteurs seront fins.

1.3 Descripteur

Dans cette partie, il s'agit d'obtenir les descripteurs de l'image pour chacun des points clés obtenus. Pour ce faire, on utilise plusieurs sous fonctions. Dans un premier temps, pour rendre invariant le descripteurs aux changements de repères, on va appliquer une rotation de la zone de $n_{pixel,zone} \times n_{zone}$ par l'orientation du point clé. (`rotationGradient(point_cle,L,n_pixel)`) Il est à noter que la zone est divisée en un nombre pair de pixels. Nous avons donc décidé de placer le point de référence dans le coin inférieur droit de la zone supérieure gauche du point clé.

On applique enfin l'algorithme de l'énoncé pour le calcul des vecteurs des descripteurs. Cependant, le calcul de l'interpolation trilinéaire n'a pas été effectué.

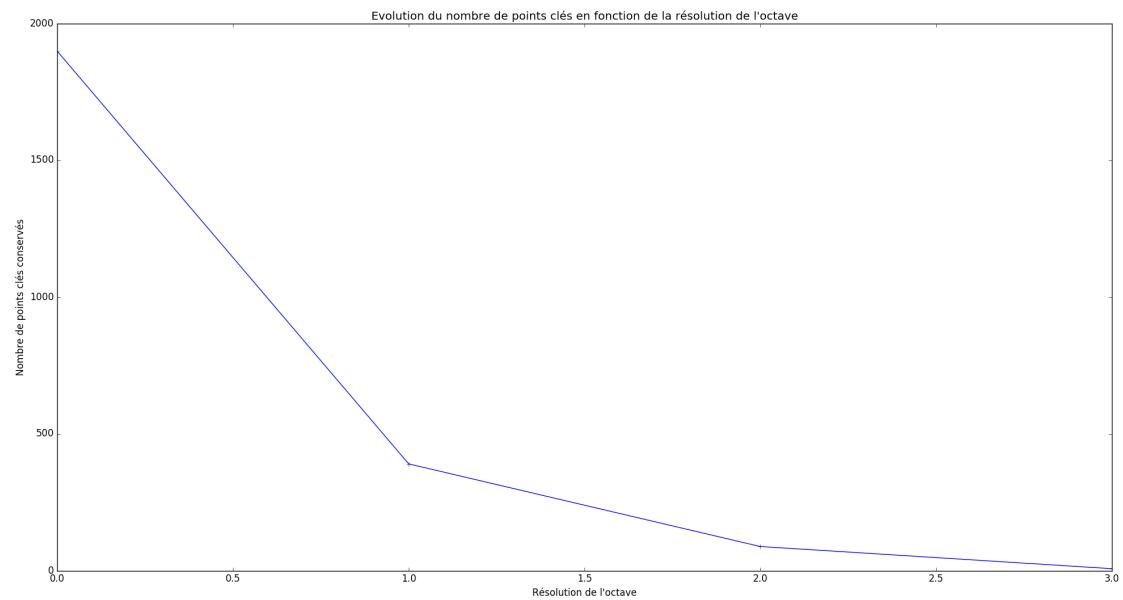


FIGURE 3 – Evolution du nombre de points clés

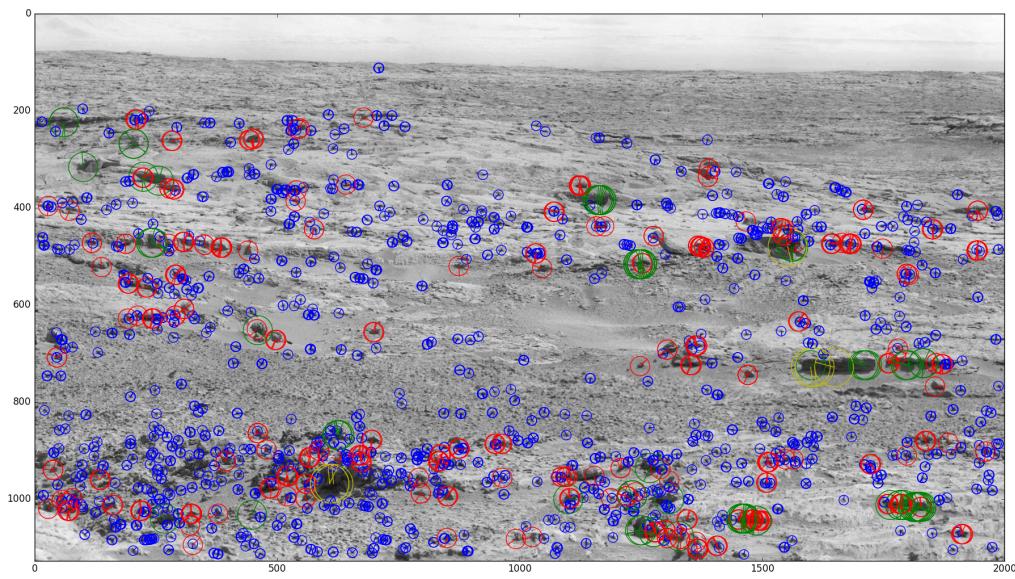


FIGURE 4 – Lieux des points clés pour l'image gauche.jpg. Les petits cercles bleus sont les points clés de la première octave, et plus l'octave du point clé augmente, plus le cercle est grand (la couleur change également).

On normalise enfin le vecteur obtenu pour le rendre insensible aux changements d'illuminations puis on élague les valeurs trop élevées.

La matrice $N_{img} \times 130$ des descripteurs est alors obtenue.

2 Matching et homographie

2.1 Recherche de couples amis

2.1.1 Question 1

Le calcul de la matrice des distances est assez classique et ne nécessite pas d'explications particulières.

2.1.2 Question 2

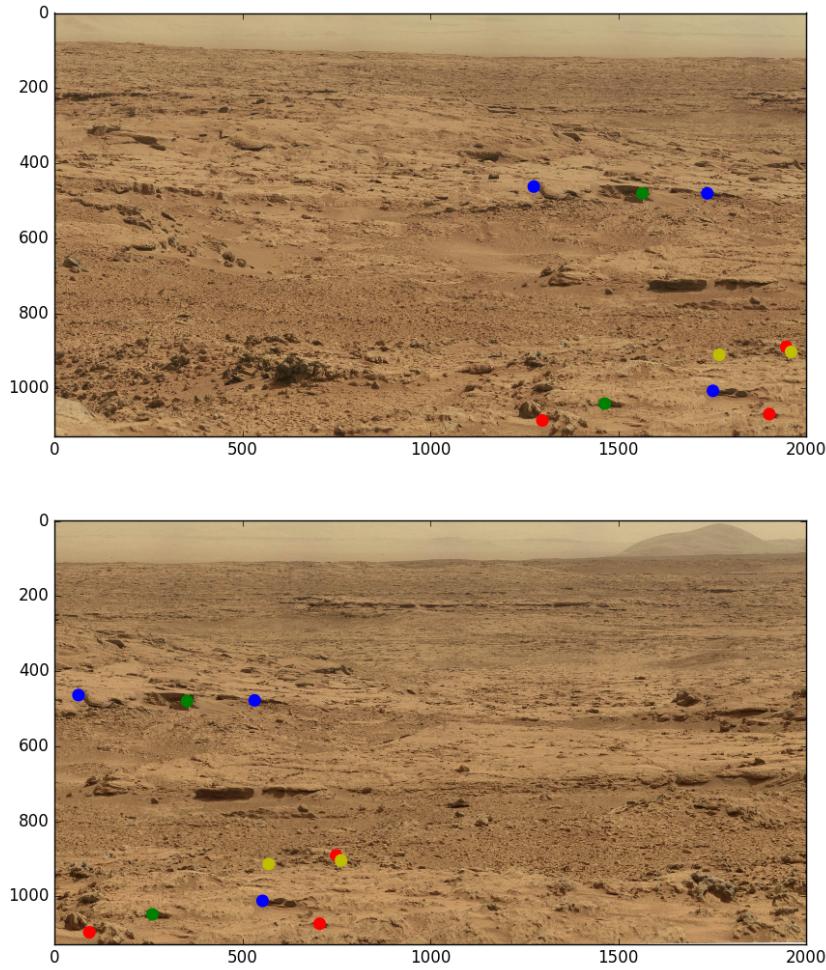


FIGURE 5 – Exemple de correspondance entre les $n = 10$ descripteurs les plus proches entre les deux images.

Dans la suite du code, on fixera le nombre de plus proches voisins à considérer pour l'homographie à 6. En effet, durant nos tests, il est arrivé plusieurs fois que trois descripteurs retenus soient quasiment parfaitement alignés, ce qui rendait le système $A \cdot H = 0$ non inversible, et il est arrivé une fois que 4 descripteurs soient alignés (sur une image de basses dimensions). Les solutions renvoyées par notre algorithme n'avaient alors aucun sens, comme on peut le voir sur la figure 6. En revanche, il ne faut tout de même pas trop augmenter le nombre de descripteurs retenus car on augmente alors la probabilité d'avoir un couple apparié par erreur, ce qui fausserait le calcul de la matrice d'homographie.

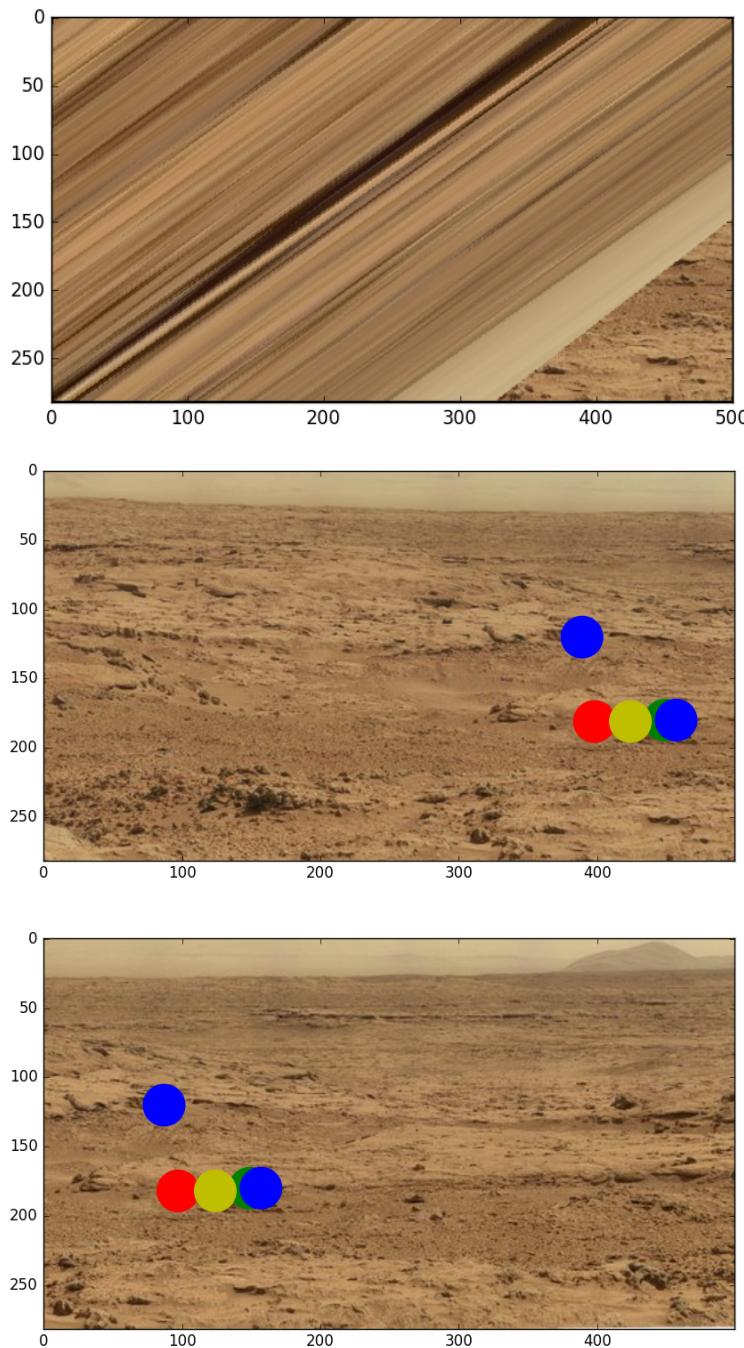


FIGURE 6 – Exemple de mauvais matching. On voit ici que la matrice d'homographie n'est pas la bonne. On remarque que les descripteurs retenus sont alignés, ce qui rend le système dégénéré. Le problème ne se pose pas quand les points ne sont pas alignés.

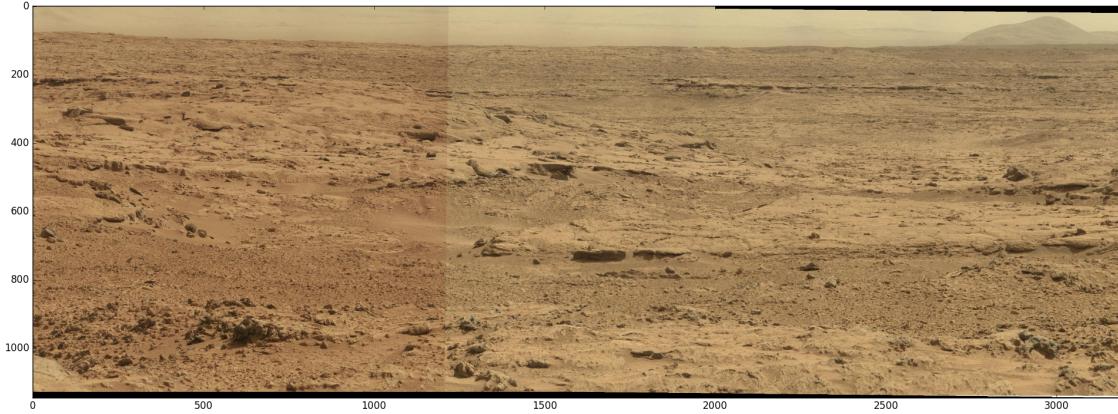


FIGURE 7 – Panorama reconstruit. On note la rupture d'exposition à la frontière des deux images. La continuité géométrique est cependant assurée.

2.2 Homographie

2.2.1 Question 1

Le code affiche systématiquement la norme euclidienne de la différence entre les deux matrices H (l'une obtenue par SVD, l'autre par les vecteurs propres de $A^T A$). La différence est toujours inférieure à 10^{-5} , ce qui est très acceptable en considérant qu'avec plus de 4 couples de points, il n'y a pas de solution exacte.

Le système $A \cdot H$ n'est pas toujours parfaitement résolu, il peut arriver que le résultat (affiché en console) ne soit pas négligeable (surtout quand on utilise davantage de descripteurs, ce qui est tout à fait logique, la solution exacte n'existant que pour 4 couples de points). Cependant cela n'a pas d'impact sur le résultat final.

2.2.2 Question 2

Même sur les photos de haute résolution (2000x1127) fournies, la méthode par SVD et la méthode par $A^T A$ prennent moins de 10ms. On a quand même souvent une meilleure performance pour la SVD. On remarque également que l'erreur résiduelle $\epsilon = A \cdot H$ est souvent moindre en utilisant la valeur de H donnée par la SVD. On prend donc la SVD pour la suite des opérations, puisqu'elle semble plus performante tant en termes de résultats qu'en termes de rapidité.

2.2.3 Question 3

Après reconstruction du panorama, on obtient l'image de la figure 7. Un examen plus approfondi nous révèle un léger décalage dans la zone de surimpression, visible sur la figure 8, mais ce décalage léger est tout à fait acceptable puisqu'invisible sur le rendu final.

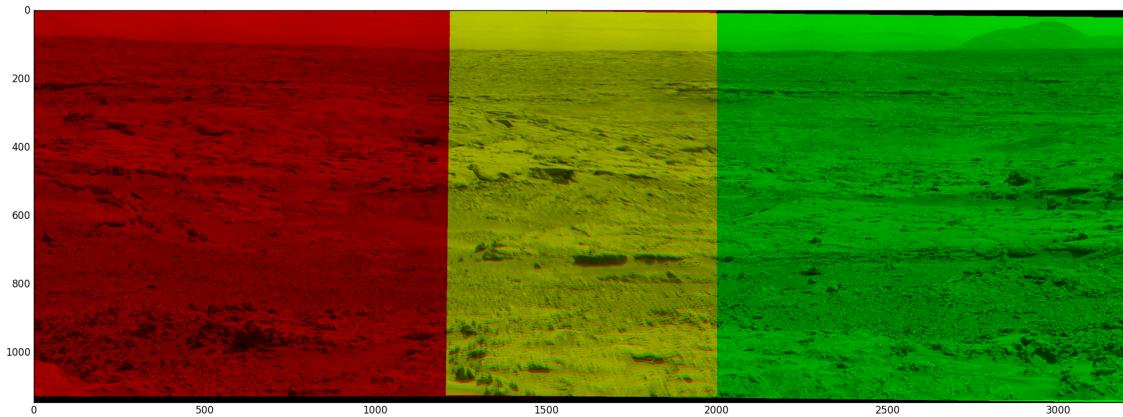


FIGURE 8 – Panorama reconstruit. L'image de gauche est sur le canal rouge, l'image de droite sur le canal vert ; la zone jaune est la zone de superposition. On note un léger dé-doublage vers la droite de la zone jaune : la transformation n'est pas parfaite.

2.2.4 Question 4

Dans le but d'améliorer le rendu visuel, nous avons essayé de corriger la différence d'exposition. Comme les deux photos sont malgré tout très semblables, nous avons privilégié une approche assez basique.

Nous avons calculé les valeurs moyennes des histogrammes des trois canaux (R,G,B) des deux images, puis nous avons ajouté la différence entre les deux moyennes à chaque pixel de la deuxième image (selon son canal). Ainsi les valeurs moyennes sur chaque canal sont les mêmes entre les deux images. Les valeurs négatives ou supérieures à 1 sont ramenées dans les bornes acceptées. Le rendu est plutôt bon, comme on peut le voir sur la figure 9. On notera cependant que cette technique est assez basique et ne marcherait certainement pas aussi bien sur des images plus contrastées (trop de pixels dépasseraient 1 ou seraient en dessous de 0), ou avec une plus grosse différence entre les deux images (même phénomène).

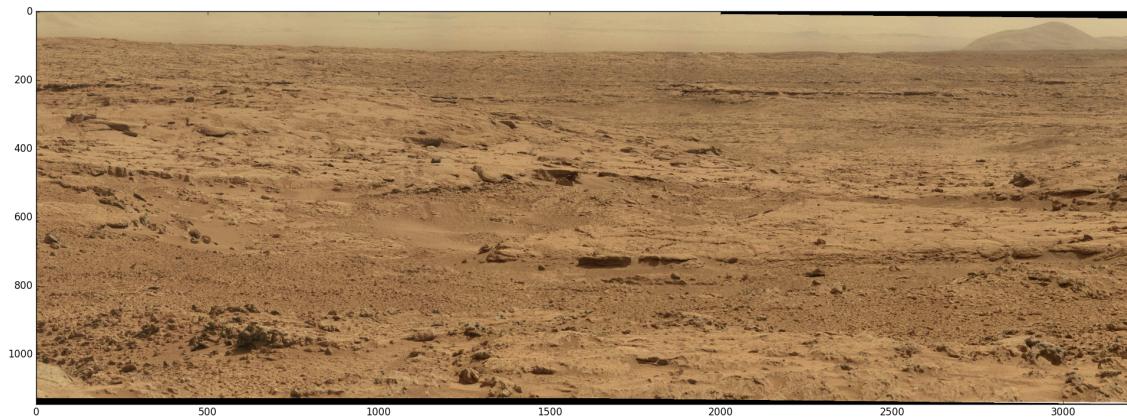


FIGURE 9 – Exemple de panorama où la différence d'exposition entre les deux images est corrigée.