

Améliorer la localisation de points d'intérêt sur des images stéréographiques en milieu sous-marin.

Jules Royer *; Matthieu Colin †

April 3, 2024

Abstract

Localiser des paires de points d'intérêt sur des images stéréographiques avec une précision en dessous du pixel est une tâche difficile avec les méthodes actuelles. De plus, construire un modèle 3D d'un objet réel requiert de trouver un nombre conséquent de telles paires. La principale difficulté provient du manque de contraste des images sous-marines. Dans cet article, un nouvel algorithme de localisation de paires de points d'intérêt est présenté. Inspiré de SIFT, algorithme incontournable sur le sujet, il construit des histogrammes de gradients des voisins d'un point et les compare entre les images. Contrairement à SIFT, il ajoute l'information des courbures principales des images mais détecte uniquement des motifs de taille constante. Sur des images de synthèse, il détecte plus de paires que SIFT, au pixel près, mais ce n'est pas le cas sur des images réelles. Des pistes d'amélioration sont proposées.

Introduction

Contexte

Les algorithmes de *Computer Vision* pour la détection de points d'intérêt sont aujourd'hui extrêmement efficaces, en particulier depuis la publication de *SIFT* [1] en 2004, avec des applications dans de nombreux domaines : médecine, logistique et imagerie numérique pour ne citer qu'eux. Néanmoins, les algorithmes existants ont des résultats mitigés sur des images sous-marines. En effet, ils proposent de nombreuses paires présentant un écart de plus d'1 pixel, qui fausseraient une éventuelle reconstruction 3D. Une difficulté majeure du milieu sous-marin est la perte de contraste et le manque de luminosité aux profondeurs considérées.

Un domaine dans lequel cette lacune est particulièrement visible est la reconstruction de coraux par images sous-marines. Des images de l'organisme marin sont prises en stéréographie à l'aide d'un robot possédant deux caméras, et une reconstruction 3D est ensuite effectuée en comparant les images prises par les deux caméras pour déterminer la distance des points aux caméras. Cette détermination de la profondeur passe par un appariement entre les deux images : pour certains points notables dans l'image de gauche, appelés points d'intérêt, le point correspondant sur l'image de droite est recherché. Pour avoir une précision inférieure à 1mm, il est nécessaire d'avoir un grand nombre de paires : $N \approx 3000$ sur des images de résolution 3000×2000 dont la correspondance est exacte. Or parmi les paires renvoyées par *SIFT*, bon nombre ne sont pas

assez précises, dégradant significativement la qualité de la reconstruction 3D.

L'objectif de l'article est donc d'améliorer la détection et la caractérisation de points d'intérêt dans des images sous-marines pour pouvoir augmenter la précision des méthodes de reconstruction d'organismes sous-marins, qui sont amenées à se développer aujourd'hui pour faire face à la perte massive de récifs coraliens. L'entièreté du code informatique se trouve dans le dépôt *Github* suivant¹.

Objectif

Notre objectif est donc de dépasser le taux de paires de points localisées au pixel près sur les images stéréographiques par l'algorithme *SIFT*. Pour cela, nous avons cherché quel relief topologique autour d'un point maximisait la détection de celui-ci sur les deux images, et nous avons conçu des métriques pour quantifier la performance de notre algorithme et pour le comparer à *SIFT*.

Etat de l'art

Dans cette section, nous présentons les algorithmes de détection et caractérisation de points d'intérêt les plus utilisés à l'heure actuelle. Contrairement à du *pattern matching* fait à partir de corrélations par patchs (la corrélation entre le motif recherché et une fenêtre carrée glissante sur l'image est calculée, et le patch où elle est maximale est le pattern recherché), ces algorithmes ont été conçus pour résister à une variation d'échelle, une rotation des images est un changement de point de vue.

*Mél: jules.royer@etu.minesparis.psl.eu

†Mél: matthieu.colin@etu.minesparis.psl.eu

¹ Lien: https://github.com/SushiOfDestiny/Data_Sophia_Aquatic_SIFT

SIFT

SIFT (Scale Invariant Feature Transform) [1], algorithme fondateur dans le domaine, utilise les extrema de l'espace des échelles pour déterminer des points d'intérêt dans l'image. L'espace des échelles de l'image est constitué de différences de gaussiennes, i.e. des différences pixel par pixel de l'image convoluée par des filtres gaussiens d'écart-types différents. Un pixel à une échelle σ_i est alors considéré comme un point d'intérêt quand sa valeur est supérieure (resp. inférieure) aux 8 pixels voisins à l'échelle σ_i , aux 9 pixels voisins à l'échelle σ_{i+1} et aux 9 pixels voisins à l'échelle σ_{i-1} (c'est alors un extrémum de l'espace des échelles). Après un filtrage et une interpolation intermédiaires, les points d'intérêt restants sont alors caractérisés par un histogramme des gradients voisins. L'orientation et la magnitude du gradient d'un point est définie par les équations 2 et 1. Chaque point d'intérêt est ainsi représenté par un vecteur de dimension 128, contenant les informations d'un carré de 4×4 histogrammes autour du point, avec 8 directions enregistrées dans chaque cellule (cf Figure 4).

SURF

SURF [2] (*Speeded Up Robust Features*) est un algorithme de détection et de caractérisation de points d'intérêt, plus récent et plus rapide que *SIFT*. Pour caractériser un point d'intérêt, il utilise des images intégrales et des sommes de réponses à des ondelettes de Haar. *SURF* n'a pas été étudié ici car il est encore sous brevet, et demandait donc de refaire une implémentation *Python*.

DAISY

DAISY [3] a la même fonction que les deux algorithmes précédent. Il est développé pour être utilisé lorsque la distance entre les caméras stéréographiques (*baseline*) est grande. Pour effectuer des calculs rapides, il repose sur des convolutions. Chaque point d'intérêt est codé par un vecteur de dimension 200, à partir d'histogrammes de gradients calculés par convolution gaussienne avec un noyau symétrique circulairement. *DAISY* n'a pas été étudié non plus.

Données

Dans cette section, nous présentons les images sur lesquelles les performances des algorithmes ont été évaluées.

Images réelles

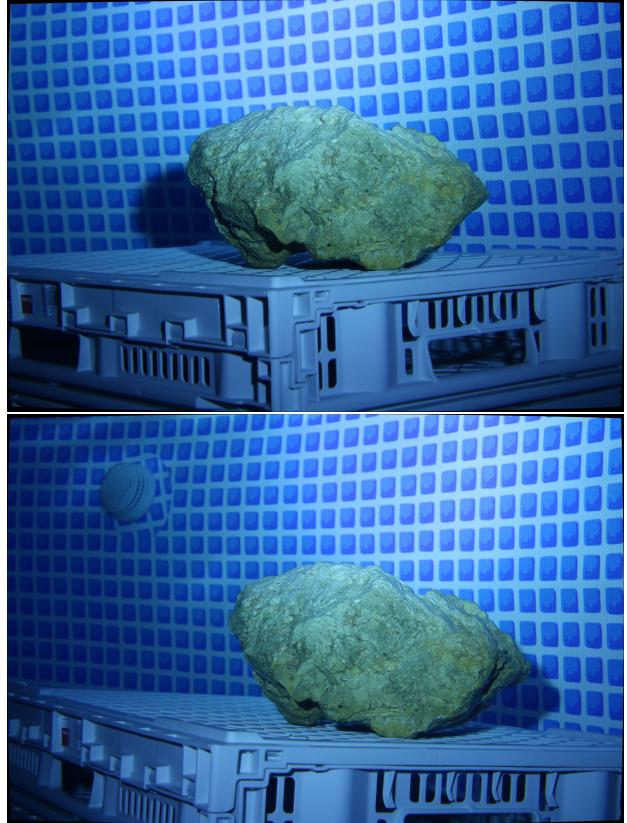


Figure 1: Exemple d'images réelles prises lors d'une expérience en bassin peu profond.

Les images réelles dont nous disposons proviennent d'une expérience en bassin peu profond, dans lequel un rocher d'environ $30\text{ cm} \times 20\text{ cm} \times 20\text{ cm}$ est photographié par un robot à deux caméras. Un exemple d'image est disponible en Figure 1. Les différences majeures entre des images en surface et des images sous-marines sont une perte de contraste significative et une modification des couleurs (visible à l'œil nu). Les images sont de résolution 3000×2000 , l'angle entre les caméras est estimé à 10° , les caméras sont situées à environ 1m du rocher, et ont une distance focale de 35mm.

Images de synthèse

La précision de paires de points d'intérêt sur des images réelles est souvent calculée par une méthode de filtrage avec la droite épipolaire. Cette méthode présente l'inconvénient d'être longue et coûteuse en calcul. De fait, une méthode alternative est développée pour pouvoir quantifier les performances entre l'algorithme présenté ici et *SIFT*.

Une scène 3D se voulant fidèle à la réalité et dont les paramètres sont parfaitement connus est construite avec le logiciel open-source *Blender*. Cette scène

comporte un rocher dont la texture est obtenue avec des déformations à bruit gaussien, et deux caméras simulant la stéréographie sous-marine. Les dimensions du rocher sont de $35.4 \text{ cm} \times 21.4 \text{ cm} \times 25.6 \text{ cm}$. Comme avec les images réelles, l'angle entre les caméras est de 10° , la distance focale des caméras est de 35 mm, la distance au rocher de 1 m. Les images produites sont de 1920×1080 pixels. Les scènes sont également disponibles sur le dépôt du projet.

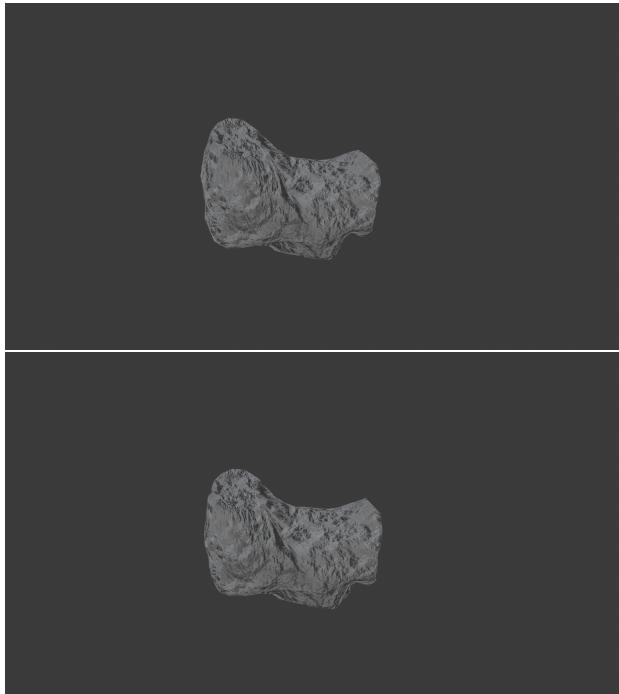


Figure 2: Exemple d'images de synthèse produites.

Sur ces images de synthèse, l'exactitude d'une paire de points est calculée automatiquement selon la procédure suivante. Pour chaque caméra, les rayons passant par le centre de la caméra et le pixel considéré situé sur l'image de la caméra sont tracés. Ce rayon intersecte le rocher 3D en un point de collision, qui est enregistré (cf Figure 3). La distance euclidienne entre les points de collision de chacune des caméras est utilisée pour évaluer la précision de la paire considérée. Si cette distance dépasse un certain seuil, alors la paire de pixels est rejetée.

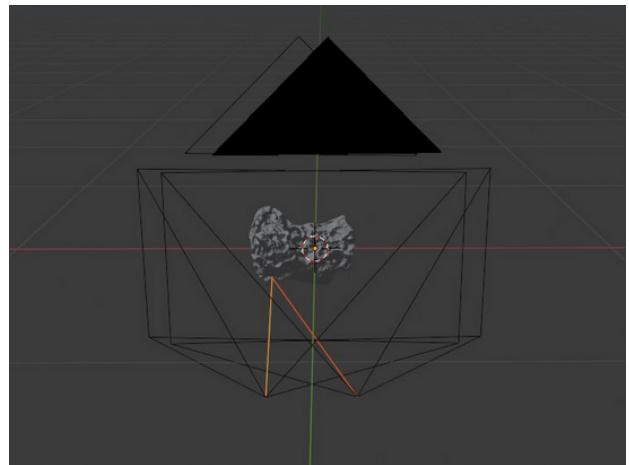


Figure 3: Tracé des rayons passant par les centres des caméras et un point d'intérêt interceptant l'objet simulé dans le logiciel Blender.

Méthodologie

Traitements de l'image

Les images sont transformées en niveau de gris par simple moyenne des 3 canaux de couleur RGB. Un recadrage manuel est effectué pour enlever au maximum les pixels de l'arrière-plan, car les points d'intérêts sont à chercher sur l'objet photographié. Ensuite un filtre gaussien est appliqué, d'écart-type 1 et uniquement sur les images de synthèse, car les images réelles ont peu de contraste. Pour pallier cela, des méthodes d'augmentation de contraste ont été testées, aucune amélioration notable n'a été constatée.

Notion de point d'intérêt

Il est certes possible de chercher les paires de points de façon brute entre l'ensemble des pixels de chaque image, mais c'est implique de longs temps de calcul, et n'est pas le plus efficace : beaucoup de points réels ne présentent que peu d'intérêt car leur voisinage présente peu de relief.

La question est alors de définir un bon point d'intérêt, i.e. un point dont le voisinage topologique varie de façon assez significative pour être reconnaissable sur les deux images. Un changement de point d'observation s'effectue pendant la prise de celles-ci, changeant alors le relief apparent et le contraste.

Pour la caractérisation des points, *SIFT* se concentre sur la répartition des gradients dans le voisinage. Il élimine en amont les points sur des arêtes, jugés trop sensibles au bruit, il calcule pour cela la trace et le déterminant de la hessienne, fonctions des courbures principales.

Les courbures principales sont les extrema des courbures en un point. Elles coïncident avec les valeurs propres de la hessienne de la surface en le point, et leurs directions respectives sont les vecteurs propres associés.

Ces courbures principales et leur directions mériteraient plus d'attention, et pourraient être considérées telles quelles. Dit grossièrement, elles contiennent une information à l'ordre 2 sur la surface et complètent l'information d'ordre 1 fournie par le gradient. Un bon point d'intérêt a selon nous dans son voisinage une variation marquée des courbures.

C'est pour cela qu'un préfiltrage grossier est effectué : sur chaque image, seul 30% des pixels de plus grande courbure principale moyenne (en norme) sont gardés.

Descripteur

Comme dans *SIFT*, un descripteur est construit, son but est d'encoder dans un vecteur des caractéristiques topologiques du voisinage du point, qu'on espère trouver sur les deux images le plus à l'identique possible.

Définissons d'abord des propriétés de robustesse que nous souhaitons pour notre descripteur:

1. Invariance par rotation: le robot sous-marin n'est pas parfaitement horizontal lorsqu'il photographie. Une même forme n'aura pas la même orientation sur les deux images. Nous souhaitons donc pouvoir identifier un point, même si son voisinage subit une rotation dans le plan de photographie.
2. Robustesse à la luminosité: la lumière sous-marine est naturellement inégale, que ce soit à cause de la lumière du soleil qui est réfractée dans l'eau et réfléchie sur les particules sous-marines, ou bien à cause des éclairages du robot sous-marin.

Il est important de souligner que l'invariance par échelle, propriété essentielle de *SIFT*, n'est pas implémentée. La raison est que cette propriété répond à un besoin très général d'identifier des points d'intérêt à différentes échelles. C'est très utile lorsque l'objet ciblé présente une structure et un relief très variés, par exemple une maison. Mais dans notre utilisation précise, nous voulons à terme photographier des récifs coraliens, et actuellement, en phase expérimentale, nous utilisons des rochers.

Le relief d'un rocher ne présente pas de motif notable, comme des arêtes marquées, mais tend au contraire à être uniforme et répétitif. Il semble raisonnable de tirer parti de la constance du dispositif expérimental, pour trouver assez de bons points

($N > 3000$) en comparant des voisinages d'une même taille fixe, déterminée à l'avance.

Le descripteur repose principalement sur le gradient et les courbures principales. Ces valeurs sont approximées par différences finies symétriques de l'image à l'ordre 1 et 2. Si l'on définit une image numérique I de dimension $(w, h) \in \mathbb{N}^2$, comme une fonction discrète: $I : \llbracket 0, w - 1 \rrbracket \times \llbracket 0, h - 1 \rrbracket \rightarrow [0, 1]$ d'abscisses x et d'ordonnées y , la norme du gradient et son orientation sont approximées par:

$$m(x, y) = \sqrt{(I(x+1, y) - I(x-1, y))^2 + (I(x, y+1) - I(x, y-1))^2} \quad (1)$$

$$\theta(x, y) = \tan^{-1} \left(\frac{I(x, y+1) - I(x, y-1)}{I(x+1, y) - I(x-1, y)} \right) \quad (2)$$

Pour calculer les courbures et directions principales, la hessienne H est d'abord calculée similairement, voir l'équation 4 en Annexes :

$$H = \begin{bmatrix} dxx & dxy \\ dxy & dyx \end{bmatrix} \quad (3)$$

Puis le module *Numpy* de *Python* est utilisé pour en obtenir les valeurs et vecteurs propres. Avant de construire l'histogramme, quelques choix techniques doivent être précisés. Dans tout l'algorithme, les angles sont en degrés et dans $[0, 360]$. Afin de n'avoir que des courbures positives, les courbures négatives sont multipliées par -1 et les vecteurs propres associés sont tournés de 180° .

Le voisinage autour d'un pixel est ensuite construit. En appelant θ l'orientation du gradient en le point, le voisinage en question est l'image par une rotation d'angle θ d'un carré de 15×15 pixels, centré en le pixel. Cela permet d'être invariant par rotation. Ce voisinage est ensuite découpé en 9 cellules de 25 pixels chacune. D'autres tailles de voisinage n'ont pas été testées en comparaison, à part un voisinage 5×5 qui a sans surprise donné de moins bons résultats.

Un histogramme des gradients des pixels est alors calculé en chacune des 9 cellules. Chaque gradient d'un voisin est une contribution à l'angle correspondant à l'écart entre l'orientation du voisin et celle du pixel central. La résolution angulaire de l'histogramme est 5° , donc celui-ci est un vecteur de taille $360/5 = 72$. La contribution d'un pixel est le rapport entre la norme de son gradient et celle du pixel central, seuillée à $\varepsilon = 10^{-6}$ afin d'éviter une division par 0. Cette contribution est pondérée par une fonction gaussienne de sa distance au centre, d'écart

type la moitié du voisinage, ici 7.5, et ceci afin de limiter l'influence des de la topologie aux bords du voisinage, voir la Figure 4. Les contributions sont ensuite transformées en pourcentage de la somme des contributions dans toutes les cellules, afin de fournir une invariance à la luminosité. Ainsi, le tableau des 9 histogrammes est de taille $9 \times 72 = 648$.

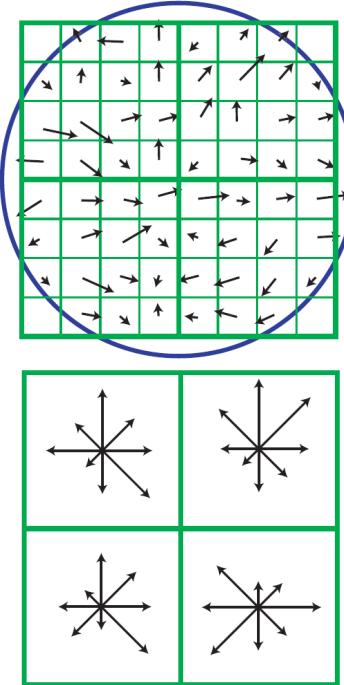


Figure 4: Schémas des gradients d'un voisinage 8×8 d'un pixel, la pondération gaussienne y apparaît en bleu, et des histogrammes des cellules. (Illustrations tirées de SIFT [1]).

Les deux histogrammes des courbures principales (ordonnées par valeur signée décroissante) sont calculés selon le même principe. Pour cela, l'orientation d'une courbure principale est définie comme l'angle fait entre sa direction et le vecteur horizontal $e_x = (x = 1, y = 0)$, et la contribution va, comme précédemment, à l'écart entre les orientations du voisin et du pixel central. Comme exemple, voir la Figure 7 en Annexes.

En concaténant les 3 tableaux d'histogrammes, on obtient le descripteur final du pixel, qui est un vecteur de taille $3 \times 648 = 1944$.

Appariement

Bien qu'il existe des méthodes analogues aux plus proches voisins adaptées à de la haute dimension, comme la *Best Bin First*, elles n'ont pas été utilisées car il aurait été nécessaire de les réimplémenter. En effet, celles des bibliothèques *Python* ne peuvent pas être précompilées en *C* par *Numba*, module utilisé pour avoir des temps de calcul "raisonnables".

L'appariement brut a donc été choisi: pour chaque pixel filtré de l'image de gauche, on l'associe à celui de l'image de droite ayant le descripteur le plus similaire, dans le sens où il minimise leur distance euclidienne. Ce choix de distance a encore été fait par défaut et aucune autre distance n'a été testée.

Il serait intéressant d'expérimenter avec une norme pondérée, accordant plus d'importance à l'information des gradients par rapport à celle des courbures.

Postfiltrage

Parmi les paires de points calculées, seul un nombre équivalent à 2% du nombre de pixels de l'image est gardé. Les paires gardées sont celles ayant les plus petites distances, i.e. avec le plus haut taux de confiance. Ce seuil est retenu puisqu'il correspond à un peu plus que l'objectif de 3000 bonnes paires sur les images réelles recadrées, de résolution 900×1400 . Le même seuil est utilisé sur les images de synthèses, même si la résolution est inférieure.

Par commodité, l'algorithme conçu dans l'article est appelé *HM*, pour *Home Made*.

Analyse du descripteur sur images de synthèse

Afin de savoir si le descripteur ainsi construit permet de distinguer des bons points d'intérêt des mauvais, pour chaque image, la moyenne des histogrammes spatiaux de la courbure principale 1 des bons points parmi les paires renvoyées par *HM* est calculée. Idem pour les mauvais points, cf Figure 8. Cependant, aucune différence notable apparaît entre le bon et mauvais descripteur, les différences visibles sont de l'ordre de 0.01%. Le même constat est fait pour les autres histogrammes spatiaux du descripteur. Ces observations ne permettent pas d'inférer un critère de sélection de bons points d'intérêt.

Résultats

Dans cette section, *HM* est comparé avec *SIFT*, paramétré par défaut, i.e. avec un seuil de contraste de 0.04, un seuil d'arête de 10, un écart type de gaussienne de 1.6, et un seuil de distance de 0.75. Il serait intéressant d'optimiser *SIFT* sur les images de synthèse pour avoir une meilleure comparaison.

Résultats sur image de synthèse

Pour des images de synthèse recadrées sur le rocher, de résolution 300×580 et d'angles entre caméras de 10° et 7° dégrés, les résultats sont dans le Tableau 1.

Angle entre caméras	10°		7°	
Algorithme (préfiltrage)	HM (60%)	SIFT	HM (70%)	SIFT
Nombre de bonnes paires parmi les 3480 de distance minimale (% de paires)	2579 (74.11%)	613 (17.61%)	2532 (72.76%)	604 (17.36%)
Statistiques sur les distances des bonnes (B) et mauvaises (M) paires	(B) (M)	(B) (M)	(B) (M)	(B) (M)
Min	10.98 14.11	27.48 33.18	11.97 12.35	15.56 38.74
Max	33.69 44.50	255.22 264.50	32.45 43.46	253.92 251.11
Moyenne	20.82 23.24	111.58 124.36	20.79 23.28	112.42 121.08
Ecart-type	1.99 2.72	49.09 52.81	2.03 2.75	50.70 48.92
Ecart entre les distances moyennes des bonnes et mauvaises paires divisé par l'écart-type des bonnes paires	1.22	0.26	1.23	0.17
Résultats avant postfiltrage (Sélection des 2%)				
Total de paires calculées	69600	931	52200	930
Total de bonnes paires renvoyées (% de paires)	17110 (24.58%)	613 (65.84%)	12647(24.23%)	604 (64.94%)
Temps de calcul (hh:mm:ss) :	00:15:39,		00:11:48,	
Image 1, Image 2, Appariement	00:15:27,	Total < 5s	00:11:37,	Total < 5s
	00:12:05		00:09:24	

Table 1: Comparaison des résultats sur deux paires d'images de synthèse de résolution 900×1400 de l'algorithme HM par rapport à SIFT réglé par défaut. Les bonnes paires ont été identifiée par l'algorithme Blender.

Surprenamment, HM a d'excellents résultats: la proportion de bonnes paires parmi celles gardées est haute par rapport à celle de SIFT: 74.11% contre 17.61% pour 10° .

Il est rassurant d'avoir une distance moyenne des bonnes paires en dessous de celle des mauvaises. De plus l'écart entre les deux moyennes est d'environ 1.22 fois l'écart-type des bonnes paires (pour 10°). Pour SIFT en comparaison, il est seulement de 0.27 fois. Sur ce test, le descripteur de HM est plus discriminant que SIFT car les distributions des bonnes et mauvaises paires sont plus espacées.

Quand on regarde les résultats pour 10° avant d'avoir fait le postfiltrage, donc avant de ne garder que l'équivalent des 2% des pixels de l'image, on voit que HM renvoie comme attendu un grand nombre de paires: en l'occurrence 30% ou 40% des pixels de l'image. La bonne nouvelle est que parmi elles se trouvent relativement beaucoup de bonnes paires: 24.58%. De l'autre côté, SIFT a la vertu d'être très parcimonieux et précis, renvoyant 931 paires dont 65.8% de bonnes. Cependant, le reproche qu'on peut faire à SIFT est de ne pas renvoyer assez de bonnes paires. On notera cependant que le pourcentage de bonnes paires renvoyées parmi le nombre de pixels de la sous-image est de 0.31% et donc supérieur à l'objectif de 0.24% correspondant à 3000 points dans une image 900×1400 (résolution d'une image réelle recadrée sur le rocher). Ainsi, SIFT valide dans ce cas les attendus, mais de peu, là où HM peut potentiellement détecter bien plus de bons points.

On peut voir sur la Figure 5 des paires proposées par HM. Un effet du préfiltrage par la courbure principale moyenne est que HM ne renvoie pas de paires dans les zones peu contrastées, là où SIFT en trouve plus, voir Figure 9 en Annexes.

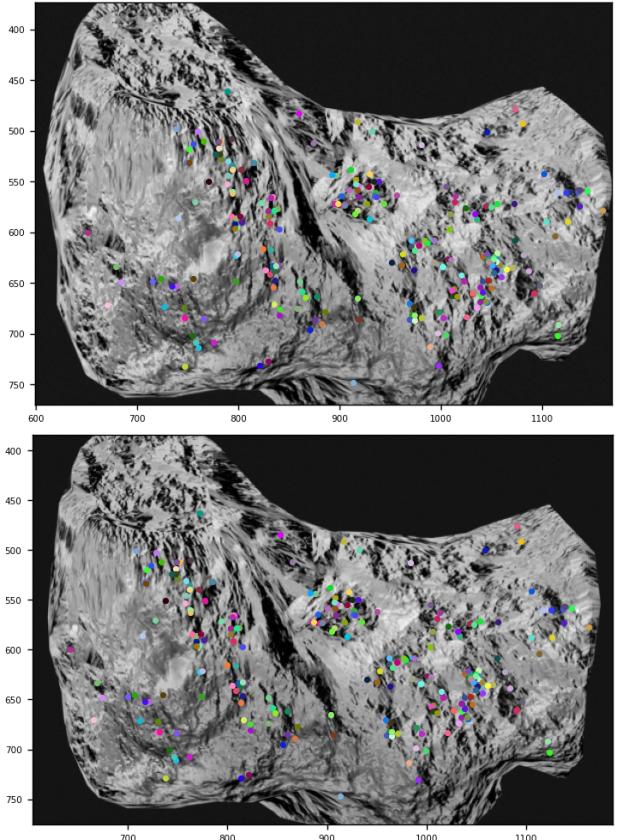


Figure 5: 250 paires tirées aléatoirement parmi celles proposées par HM.

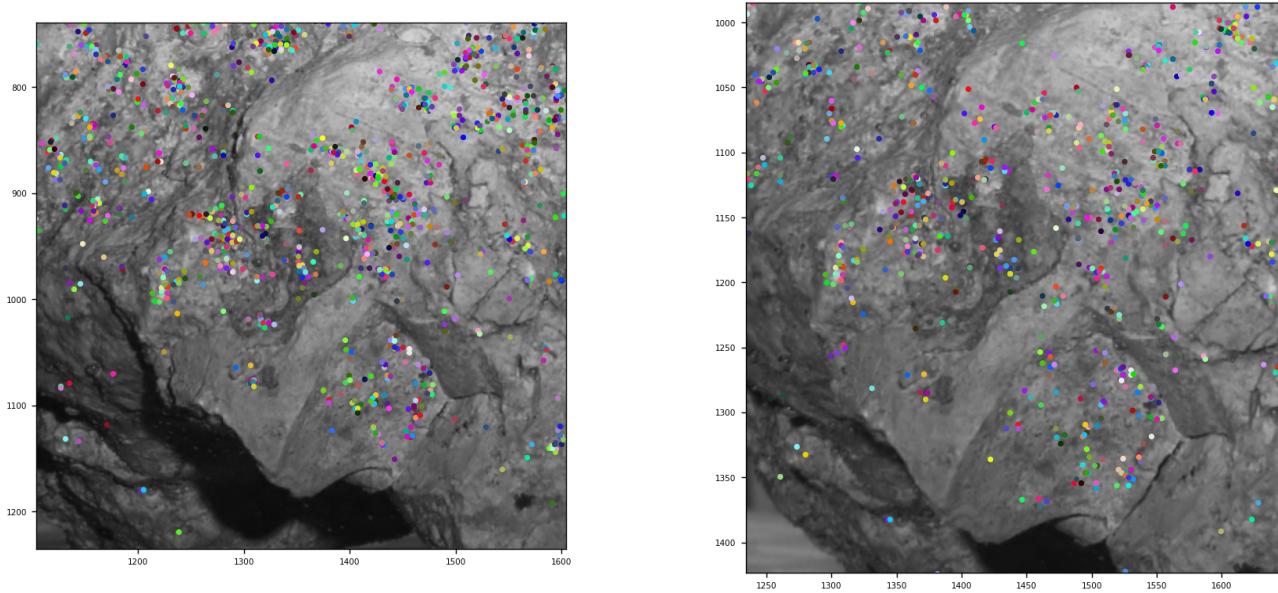


Figure 6: Zoom sur des paires tirées aléatoirement parmi celles proposées par HM sur images réelles.

Résultats sur images réelles

Les images réelles ont comme résolution 800×1500 après recadrage. *HM* a fonctionné avec un préfiltrage de 70%, le temps de calcul est de 4 heures 30 par image et 5 heures 10 pour l'appariement sur une puissante machine de 20 coeurs.

On voit clairement ici la limite de notre algorithme, peu parcimonieux par nature et implémenté en *Python*. Quant aux résultats, les zooms en Figure 6 montrent de nombreux déchets, malgré de visiblement bonnes paires dans les zones contrastées. On voit tout d'abord que les images de synthèses sont sensiblement différentes de celles réelles, en terme d'angle entre caméras, de variation de contraste, et de résolution. On comprend ici la faiblesse de *HM* sur ces dernières qui semble observer un voisinage fixe, trop petit ici pour encapsuler un relief topologique caractéristique du point, car ces images sont de haute résolution et possèdent globalement peu de relief, donc les variations notables de celui-ci se font sur une plus grande zone que 15×15 .

On trouvera en Annexes sur la Figure 10 les résultats de *SIFT* sur la même zone. Sans surprise, ceux-ci sont très bons, avec très peu de déchets flagrants, il faudrait certes zoomer plus pour vérifier la validité des paires au pixel près, mais il reste que les performances de *HM* restent bien en deçà de celles de *SIFT*. S'il faut néanmoins trouver un avantage à *HM*, on fera remarquer que celui-ci a trouvé quelques visiblement bonnes paires, non identifiées par *SIFT*, par exemple la paire bleue en bas à gauche de la Figure 6. Mais évidemment, l'algorithme *HM* n'a pas les résultats attendus.

Discussion

Une lacune importante de la méthodologie adoptée est que les différents paramètres n'ont pas été testés avec d'autres valeurs pour s'assurer que celles utilisées sont les plus pertinentes. C'est par exemple le cas pour l'écart-type du filtre gaussien lors du traitement d'image, influant sur l'équilibre détail - bruit; la résolution, les dimensions des rochers, l'éclairage et l'angle de caméras des images de synthèse, pas assez proches de ceux des images réelles; le pourcentage de points à garder pour l'appariement; ou plus important encore, la taille du voisinage pour le descripteur.

Ce dernier point semble important car l'algorithme *HM* repose sur l'hypothèse de pouvoir trouver assez de bons points avec une taille fixe de voisinages. Or celle-ci n'a pas fait l'objet d'un travail spécifique de détermination, et la valeur retenue de 15 est arbitraire et trop petite pour la résolution des images réelles.

Une méthode pour pallier cela serait d'abord d'ajuster, comme évoqué juste avant, les paramètres des images de synthèse, afin d'avoir des scènes virtuelles plus fidèles à la réalité, ensuite observer la variation du taux de bonnes paires trouvées par *HM* en fonction de la taille du descripteur. C'est au final une tâche d'optimisation. Il est fort probable que cette relation soit une fonction croissante, il faudrait sûrement introduire une pénalité ou une limite supérieure sur la taille du descripteur. En plus de cela, l'étape de préfiltrage est grossière: il serait pertinent d'inclure la norme du gradient en plus des courbures. Cette remarque vaut aussi pour l'étape

d'appariement par distance entre descripteurs. Cette distance peut en effet être pondérée pour augmenter l'importance de l'information sur les gradients par rapport à celle des courbures. Il serait aussi envisageable, une fois la taille de descripteur fixée, de s'inspirer encore de *SIFT* et d'ajouter une étape de détection d'extrema locaux sur des différences de gaussiennes. A la différence qu'on se limiterait à un nombre restreint d'échelles, choisies en fonction de la taille du descripteur. On pourrait même incorporer la phase d'interpolation des extrema dans l'espoir d'ajouter de la précision.

Enfin, comme présenté en introduction, plusieurs méthodes très efficaces ont été développées ces dernières années à la suite de *SIFT*, comme *SURF* et *DAISY*. Elles n'ont pas été utilisées pour ce travail mais semblent pertinentes.

Remerciements

Nous tenons à remercier tous les encadrants du trimestre *Data Sophia* pour leur aide et leurs conseils: Philippe Blanc pour ses cours sur le traitement d'image, Sébastien Travadel pour ses cours sur l'apprentissage statistique et son encadrement de notre travail, Luca Istrate pour son aide informatique, Frank Guarnieri et Hadrien Verbois pour son cours sur les séries temporelles.

Annexes

$$\begin{aligned}
 dxx &= I(x+1, y) + I(y, x-1) - 2I(x, y), \\
 dyx &= I(x, y+1) + I(y-1, x) - 2I(x, y), \\
 dxy &= \frac{(I(x+1, y+1) + I(x-1, y-1) - I(x-1, y+1) - I(x+1, y-1))}{4}.
 \end{aligned} \tag{4}$$

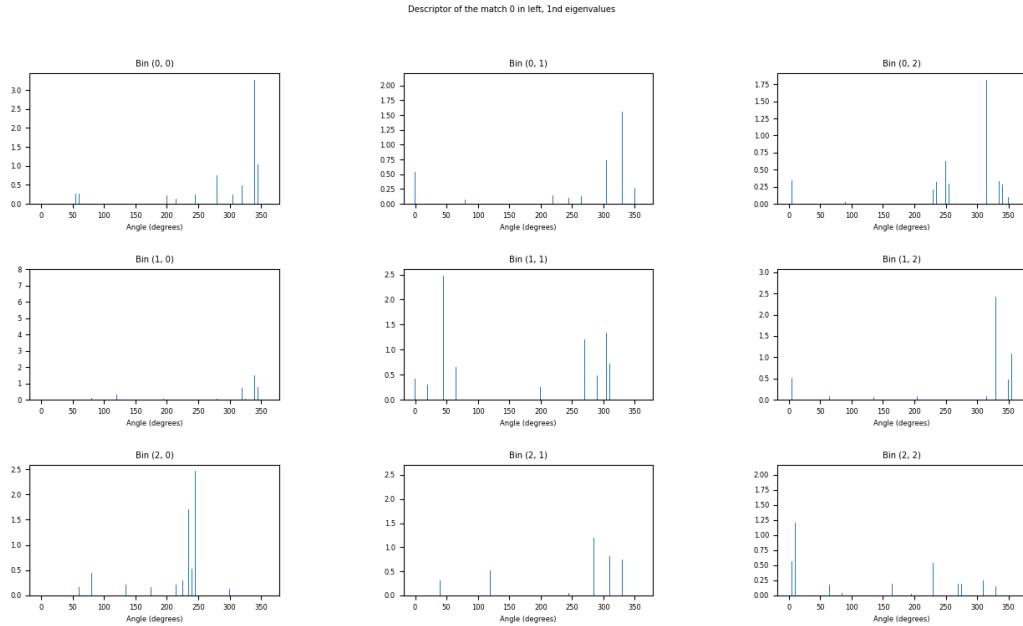


Figure 7: Exemple du tableau des histogrammes de la courbure principale 2, servant à construire le descripteur d'un point. Les bins désignent les différentes cellules, la (0,0) est en haut à gauche de la (1,1), contenant en son centre le point pour lequel le descripteur est calculé.



Figure 8: Comparaison des histogrammes spatiaux moyens de la coubure principale 1 des bons points trouvés par HM (à gauche, ~ 2500 points) et des mauvais points (à droite, ~ 70000 points) sur image Blender.

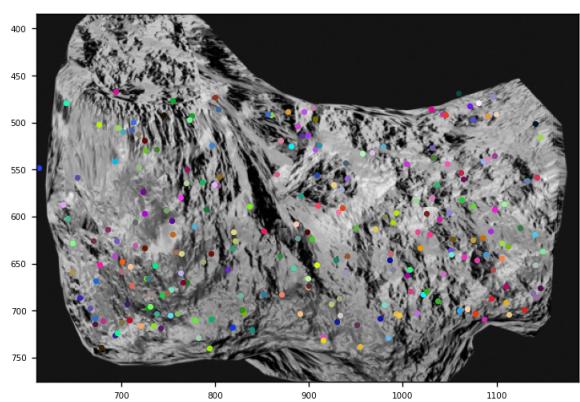
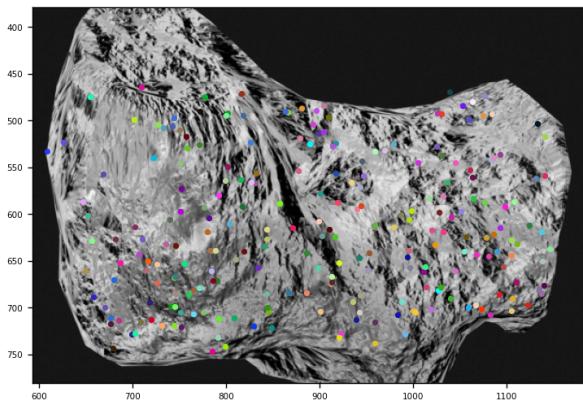


Figure 9: 250 paires tirées aléatoirement parmi celles proposées par SIFT.

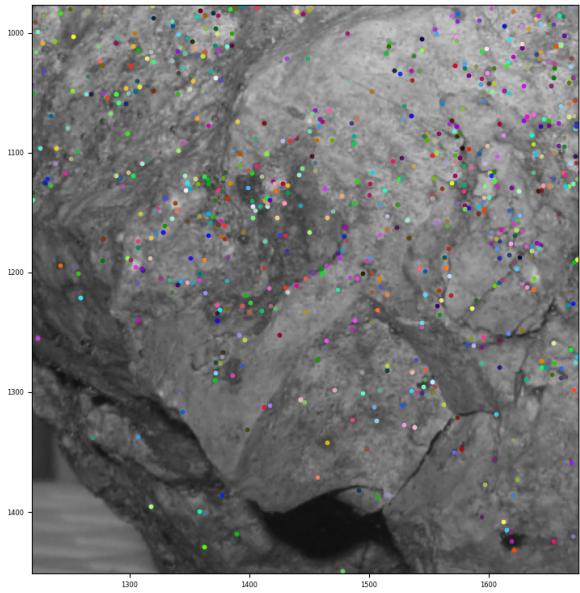
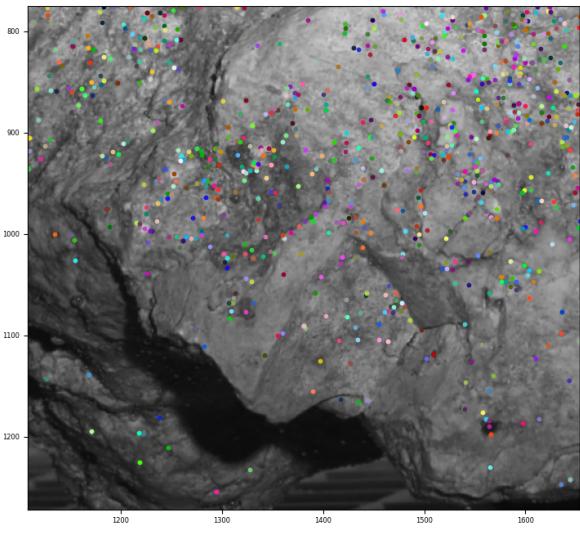


Figure 10: Zoom sur des paires tirées aléatoirement parmi celles proposées par SIFT.

References

- [1] David G. Lowe. "Distinctive Image Features from Scale-Invariant Keypoints". In: *International Journal of Computer Vision* 60 (2004), pp. 91–110. doi: 10.1023/B:VISI.0000029664.99615.94.
- [2] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. "SURF: Speeded up robust features". In: *Computer Vision-ECCV 2006* 3951 (July 2006), pp. 404–417. doi: 10.1007/11744023_32.
- [3] P. Fua E. Tola V. Lepetit. "A fast local descriptor for dense matching". In: *Conference on Computer Vision and Pattern Recognition, Alaska, USA* (June 2008). doi: 10.1109/CVPR.2008.4587673.