GLO-4001/GLO-7021 Introduction à la robotique mobile
TP1 Version complète 1.0
Date de remise : 23 octobre 2017 à 23h55

En équipe de 1 à 2.

7 octobre 2017

Attention ! N'oubliez pas d'attacher le code de toutes les questions dans la remise .zip de votre travail. Si les codes sont manquants, nous pourrons retirer jusqu'à 20% de la note.

Pour tous les étudiants, vous devez fournir un rapport en un seul document (format pdf) et les fichiers matlab ou Python zippé. Pour toutes les questions, n'oubliez pas de mettre le détail des calculs.

Pour les étudiants en GLO-7021, veuillez noter qu'une présentation déficiente dans le rapport (manque de clarté, orthographe et grammaire, police de caractère illisible sur figure matlab, etc) pourra entraîner une pénalité allant jusqu'à 10 % de la note. Le rapport doit aussi obligatoirement être formaté avec Latex.

1 Carte 2D à partir d'un gyroscope et d'un capteur de distance (15 pts)

Un robot (simulé) possède 1 gyroscope à taux bruité et un capteur de distance (calibré, Équation 1) placé exactement au centre du robot. La fonction du capteur de distance, avant bruit, est :

$$z(d) = 1/d, \text{ en Volt.} \tag{1}$$

où d est en mètres (m). Le bruit sur la mesure de voltage est une distribution normale non-biaisée $\sigma_z = 0.025$ Volt. La portée maximale du capteur est autour de 5 m. Si le capteur ne détecte pas un obstacle, il retourne une valeur de 0 Volt exactement. Il vous faut donc vous assurer de tester cette condition quand vous inverserez la fonction de capteur à l'Équation 1 pour passer d'une mesure en Volt vers une mesure estimée de distance en m.

Le gyroscope donne la vitesse angulaire du robot, en rad/s. Le gyroscope a un bruit non-biaisé suivant une distribution normale avec $\sigma_g = 0.03$ rad/s. Le fichier `Q1Donnees.mat` contient les données `z`, `g` échantillonnées à 25 Hz, ainsi que le temps `t`, pour un essai de robot simulé. Lors de cet essai simulé, le robot était immobile pendant les premiers instants (courbe de voltage du gyroscope relativement plate), puis a effectué un tour complet (2π rad), avant de s'immobiliser de nouveau pendant quelques instants.

1.1 Carte 2D locale (5 pts)

Tracez une carte en 2D (un nuage de points), avec l'option `'ro'` dans la commande `plot` dans matlab (ou fonction équivalente dans python), à partir des mesures `g` du gyroscope et les mesures `z` du capteur de distance, comme dans la *Partie 6 - Création d'une carte de l'environnement* du laboratoire sur les gyroscopes. Vous avez donc besoin de retrouver $\theta(t)$ et la distance $d(t)$. Pour cette carte (dite *locale*), faites l'hypothèse que le robot est à (0,0) et $\theta(0) = 0$ (donc la première mesure est aligné sur l'axe des x). Assurez-vous de convertir les données en coordonnées cartésiennes. Stockez les coordonnées cartésiennes des points dans une matrice `P` de taille $2 \times n$, où n est le nombre de points que vous avez choisis de conserver pour faire la carte. N'utilisez-pas de commande de tracés en coordonnées polaires (comme la commande `polar` dans matlab), mais plutôt `plot` pour tracer les points contenus dans votre matrice `P`. Incluez cette carte dans votre rapport.

1.2 Localisation du robot dans la carte globale (10 pts)

La matrice `Carte` contient un tracé des murs de l'environnement (Fig. 1) dans lequel le robot se trouvait lors de la prise du scan 2D. Pour charger en mémoire cette carte et la tracer, faites les commandes suivantes dans matlab :

```
load Carte.mat
plot(Carte(1,:),Carte(2,:));
axis equal;
```

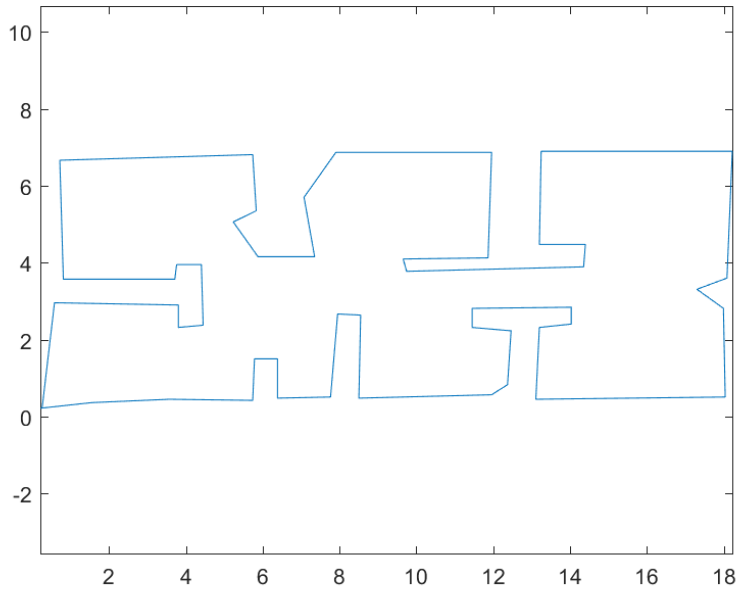


FIGURE 1 – Carte de l'environnement dans lequel le scan 2D a été effectué. Les lignes bleues représentent les murs dans l'environnement. Les coordonnées sont dans le repère global. La position du robot lors du scan est inconnue.

Votre tâche consiste à trouver par essai et erreur la pose globale initiale $[x_g, y_g, \theta_g]^T$ du robot où le scan 2D a été effectué (donc au début du scan), en alignant le mieux possible la carte locale de points trouvée à la section 1.1 sur la carte globale. Vous devez donc trouver la matrice de rotation $\mathbf{R}$ et de translation $\mathbf{T}$ qui permettra de superposer raisonnablement bien le nuage de points de la carte locale sur la carte globale. Pour faire la conversion de coordonnées, modifiez votre matrice `P` en coordonnées homogènes `PHomogene`. La conversion des coordonnées locales en globales sera donc :

$$\texttt{PGlobal} = \mathbf{T} * \mathbf{R} * \texttt{PHomogene}$$

Dans votre réponse, indiquez la pose initiale du robot $[x_g, y_g, \theta_g]^T$ ainsi que les matrices $\mathbf{R}$ et $\mathbf{T}$ correspondantes. Incluez la figure montrant la carte globale, le nuage de points alignés `PGlobal` superposé sur cette carte globale, et la position du centre du robot que vous marquerez par un cercle vert (`'go'`). Ne mettez pas les données invalides du capteurs sur la carte. Gardez en tête que les points les plus éloignés du robot sont les plus bruités.

*Note : Ce problème s'appelle "calage de nuage de points" (point-cloud registration). Nous verrons un algorithme (*ICP : Iterative Closest Point*) permettant de faire cette tâche automatiquement (mais*

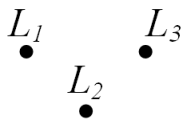
sans garantie d'être la bonne solution) dans la deuxième moitié du cours. L'exercice ici est de vous familiariser avec la problématique et aussi avec la difficulté de trouver cette pose globale initiale.

2 Modèle de caméra et positionnement par caméra

Ces exercices serviront à vous familiariser à la fois au processus de génération d'images et de localisation par caméra. Dans un premier temps, vous aller devoir faire le code permettant la génération d'images pour un monde simplifié. Dans un deuxième temps, vous allez utiliser du code pour vous localiser par triangulation. Finalement, à l'aide de nombreuses simulations, vous allez être à même de voir l'impact de l'incertitude des mesures sur l'incertitude de la position. Pour simplifier le problème, nous allons toujours utiliser les coordonnées du référentiel de la caméra. La distance focale de la caméra est de $f = 1200$ pixels. Notez que ces questions sont fortement inspirées du laboratoire sur les caméras.

2.1 Génération d'une image (5 pts)

Vous avez trois points de repère, situé au coordonnées *caméra* suivantes (en m) :
— L_1 : [-0.25, 0 , 1.25]
— L_2 : [0, 0 , 1]
— L_3 : [0.25, 0, 1.25]
L'axe optique de la caméra est parallèle au plancher, et l'axe des y pointe vers le bas. Vue de haut, l'axe des x pointe vers la droite. Le diagramme de la Figure 2 illustre le problème.



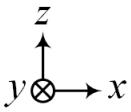


FIGURE 2 – Diagramme illustrant la position des points de repères par rapport au repère de la caméra.

Faites une fonction qui prend la photo, c'est-à-dire qui calcule la positions des trois points de repère L_i dans les coordonnées u et v du plan image de la caméra. Notez que selon la convention choisie, v pointe vers le bas et u vers la droite. Incluez dans votre rapport les coordonnées obtenues.

2.2 Estimation de la pose de la caméra à partir des angles α et β (5 pts)

Reprenez le code du laboratoire pour calculer les angles α et β à partir de l'image, et retrouver la position de la caméra. Faites l'équivalence avec une carte 2D où les points de repères L_i y sont situés, avec $x_{carte} = x$ et $y_{carte} = z$. Ainsi votre fonction retournera les coordonnées de la caméra dans cette carte 2D.

2.3 Impact du bruit sur l'estimation des repères (10 pts)

En réalité, la détection de point de repères dans une image ne se fait pas parfaitement. Ainsi, il y aura une petite erreur entre la position estimée et la position réelle, en pixel, de ces repères. Vous

allez simuler cela en ajoutant du bruit sur la coordonnée en u de ces repères. Nous n'ajoutons pas de bruit en v car nous ne sommes intéressés que par la position horizontale de ces repères dans l'image. Ce bruit suivra une distribution gaussienne, avec écart-type $\sigma_p = 2$ pixel.

Montrez la distribution des estimés de pose de la caméra sur la carte 2D (x_{carte},y_{carte}), en traçant un point par estimé trouvé par simulation. Chaque simulation consiste simplement à piger les bruits aléatoirement, de les additionner au valeurs véritable des u, d'estimer les angles α et β sur ces valeurs bruitées, puis de calculer la pose de la caméra. Répéter cette routine mais pour des nouvelles poses de caméras en la reculant de 1 à 7 m, par incrément de 1 m. Pour simuler le déplacement de la caméra d'un mètre vers l'arrière par exemple, simplement additionner 1 m aux coordonnées en z des repères avant la prise de photo. Conserver toujours les mêmes coordonnées x_{carte} et y_{carte} pour les points de repères L_i, de sorte à voir cette caméra reculer dans la carte 2D.

Pour chaque pose, faites 1000 simulations bruitées et marquez d'un point sur un graphique la pose trouvée, afin de visualiser approximativement cette distribution. Par exemple, la Figure 3 montre un ensemble hypothétique de distributions illustrée de la manière demandée, pour des poses comparables à l'exercice demandé.

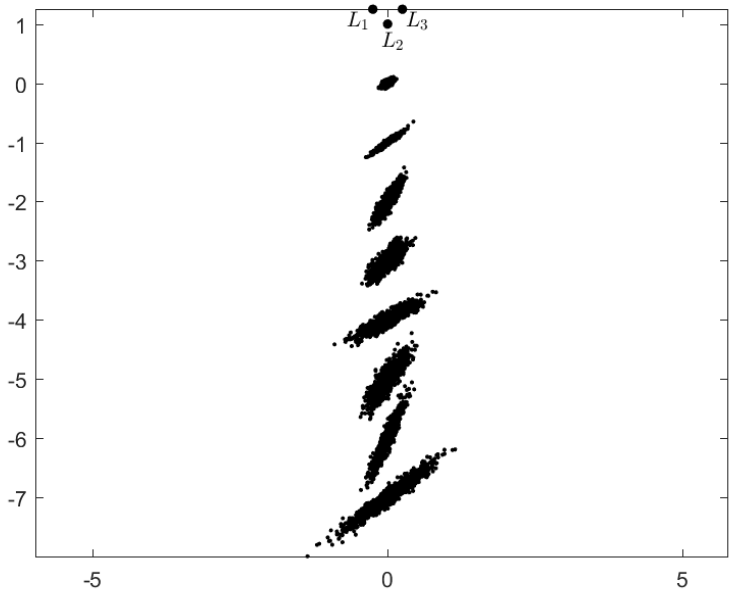


FIGURE 3 – Exemple de distributions hypothétiques pour différentes poses de la caméra. Notez la position fixe des repères L_i, tel que demandé dans l'exercice, et le centre des distributions qui reculent, selon y_{carte}.

2.4 Discussion sur la forme de la distribution (5 pts) GLO-7021 seulement

Calculez la covariance empirique de cette distribution pour chacune des poses. Que remarquez-vous sur la forme de ces distributions, en particulier les termes en dehors de la diagonale de cette matrice de covariance ? À quoi attribuez-vous cela ?

Tracez une figure montrant les écart-types σ_{xcarte} et σ_{ycarte}, en fonction des huit poses réelles (de 0 à 7 m). Discutez.

2.5 Calcul de l'orientation du robot (5 pts) GLO-7021 seulement

Comment pouvez-vous retrouver l'orientation du robot à partir de ces images, pour un cas générique ? Détaillez votre réponse, en incluant un diagramme et les équations nécessaires.

3 Imagerie stéréo (15 pts)

Voici une paire d'images en stéréo à la Fig 4 pour une scène composée de 4 tours parfaitement verticales. Le baseline b entre les caméra est de 5 cm. La tour verte fait 15 cm de longueur et est située à une distance en z (selon l'axe optique) de 95 cm. L'axe des x de la caméra pointe vers la droite dans les images. Le point principal est $(0, 0)$.

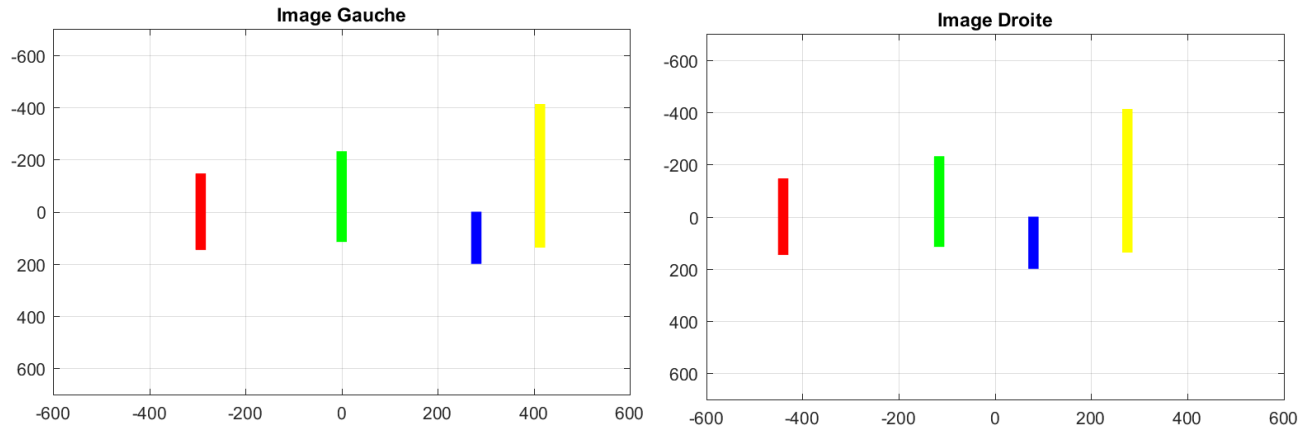


FIGURE 4 – Paire d'images en stéréo. Les coordonnées sont en pixels.

3.1 Distance focale f (4 pts)

Estimez la distance focale f (en pixel) de la caméra, en utilisant la tour verte dans l'image de gauche. Donnez le détail de votre calcul.

3.2 Estimation de la distance A_z de chaque tour LEGO (6 pts)

En mesurant la disparité d entre les centres des 3 autres tours (bleu, rouge et jaune) trouvez leur coordonnée A_z respective. Bien indiquer vos mesures et calculs dans votre rapport, et présentez votre réponse finale dans un tableau comme la Table 1.

TABLE 1 – Distance en z des centres des faces des colonnes.

tour	A_z
rouge	
bleu	
jaune	

3.3 Estimation de la coordonnée A_x de chaque tour LEGO (5 pts)

Retrouvez la coordonnées en X de chacun des objets par rapport à la caméra de gauche. Faites l'hypothèse que l'axe X de la caméra gauche pointe vers la droite, et que le point principal est situé au centre de l'image.

TABLE 2 – Coordonnée en x des centres des faces des colonnes.

tour	A_x
rouge	
vert	
bleu	
jaune	

4 Extracteur de coin FAST (13 pts)

4.1 Fonction d'extraction des coins FAST (7 pts)

Codez une fonction d'extraction de coins basée sur la méthode FAST, pour des cercles tels que montré à la Figure 5. Notez que vous cherchez un coin ayant 12 pixels continus sur l'arc qui satisfassent le critère. La fonction a le prototype suivant :

```
[IsFastCorner IntensiteCoin] = DetectionCoinFAST(Image, Centre, Seuil)
```

où les arguments en entrée sont :

- `Image` une image (en noir et blanc) ;
- `Centre` un vecteur donnant la coordonnée x-y pour laquelle tester la présence d'un coin dans l'`Image` ;
- `Seuil` le seuil t, tel que spécifié dans les équations de l'acétate 134 de `03-VisionII.pdf` ;

et les sorties sont :

- `IsFastCorner` indique la présence ou l'absence d'un coin (valeur binaire) ;
- `IntensiteCoin` donne l'intensité du coin représenté par la somme V, telle que spécifiée à l'acétate 135 ;

		16	1	2		
	15				3	
14						4
13			C			5
12						6
	11				7	
		10	9	8		

FIGURE 5 – Position des pixels à tester pour le détecteur de coin FAST. Les numéros sont à titre indicatif.

4.2 Test de votre fonction `DetectionCoinFAST` sur une image réelle (6 pts)

Vous allez maintenant tester votre détecteur de coin une image réelle, à partir d'un jeu de données qui a été capturé sur l'île de Devon dans l'arctique canadien par un robot mobile de l'équipe du Prof. Barfoot (U. Toronto). Cette île est reconnue mondialement pour sa similarité avec le sol martien, et sert donc souvent de lieu d'essai pour la robotique interplanétaire. L'image en question est

`bw-rectified-left-022146small.png`. Parcourez toute cette image (sauf la bordure à l'intérieur de 8 pixels [1]) et trouvez tous les coins, avec la valeur de `Seuil` de 10 [2]. Pour chaque coin trouvé, marquez sa position à l'aide d'un petit cercle rouge (option `` `ro' ``). Attention ! Dans matlab, la fonction `imshow` intervertie les axes x-y. Pour tracez un cercle rouge qui indique qu'un coin se situe autour du pixel `Image(X,Y)`, vous devez faire :

```
imshow(Image)
hold on
...
plot(Y,X,'ro');
```

Combien de coins trouvez-vous dans cette image ? Quel pourcentage des pixels sont donc considérés comme des coins ? Pourquoi cette scène génère-t-elle ce nombre de coins ? Quelles régions de l'images contiennent plus de coins ? Moins de coins ?

Incluez aussi dans votre rapport l'image avec les coins trouvés.

1. Car il ne sera pas possible d'extraire les features BRIEF de la question suivante pour les bordures.
2. Les intensités dans l'image sont entre 0 et 255, car codé en entier non-signé de 8 bits (`uint8`)

5 Descripteur BRIEF (17 pts pour GLO-4001, 27 pts pour GLO-7021)

Le descripteur binaire BRIEF est de plus en plus utilisé pour décrire des points de repères naturels dans des problèmes de localisation par caméra. Cette popularité grandissante est en partie dûe à sa facilité de codage, sa robustesse et sa rapidité de calcul. Dans cette question, vous allez explorer l'utilisation de ces descripteurs BRIEF, extraits autour de coins FAST. Pour vous aider, référez-vous au besoin à l'article original du BRIEF : `https://www.robots.ox.ac.uk/~vgg/rg/papers/CalonderLSF10.pdf` .

5.1 Fonction calculant un descripteur BRIEF (3 pts)

En matlab ou python, écrivez une fonction `ExtractBRIEF(ImagePatch,BriefDescriptorConfig)` qui accepte une patch d'image noir et blanc de SxS pixels avec S=15, ainsi qu'une structure de données `BriefDescriptorConfig`. Cette fonction retourne le descripteur utilisant `BriefDescriptorConfig`, dans un format de votre choix. Cette fonction ne devrait être que quelques lignes de code. Plus d'information sur `BriefDescriptorConfig` est disponible à la question suivante.

5.2 Pipeline d'extraction de features sur une paire d'images réelles (9 pts pour GLO-4001, 11 pts pour GLO-7021)

L'extraction de features visuels est en général une opération coûteuse du point de vue calcul. Ainsi, il est préférable de ne se concentrer que sur des points intéressants dans les images, les *keypoints*. Dans la section 4, vous avez justement codé une fonction permettant l'extraction de keypoints. Un pipeline d'extraction ressemble typiquement au diagramme de la Figure 6.

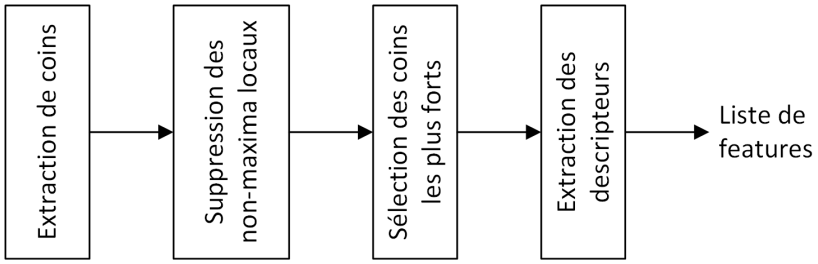


FIGURE 6 – Pipeline d'extraction des features, utilisé lors de la recherche de points de repères naturels dans des images.

Implémentez ce pipeline, en utilisant vos deux fonctions `DetectionCoinFAST` et `ExtractBRIEF` pour la première et quatrième étape du pipeline. Pour les étudiants en GLO-4001, la suppression des non-maxima locaux est optionnelle. Pour les GLO-7021, implémenter une version approximative de cette suppression, de votre choix. La sélection des coins sera basée sur la valeur de leur intensité `IntensiteCoin` : vous ne conserverez que ceux dans le 90ème percentile (autrement dit, les 10 % les plus forts)[3] . Pour les descripteurs, utilisez `numberOfBits`=256 bits. Dans ce programme, la structure de données `BriefDescriptorConfig` qui décrit les `numberOfBits` paires de pixels pour lesquelles les tests sont faits y est initialisée une seule fois. Pour générer ces paires de pixels (obligatoirement de

3. Une manière facile pour identifier ces coins est de trier la liste des coins selon l'intensité, et de ne conserver que un dixième de cette liste.

manière aléatoire), utilisez une distribution uniforme (`rand()`) sur SxS, avec S=15. Ne vous préoccupez pas des doublons possibles sur ces paires de pixels, pour vous sauver du temps de codage. Assurez-vous que ces paires sont des entiers, via la fonction `ceil()`. Le descripteur doit être extrait dans une fenêtre centrée sur le coin FAST.

5.3 Appariement features image gauche-droite (5 pts)

Appliquez votre pipeline sur les images suivantes :
— `bw-rectified-left-022146small.png` et
— `bw-rectified-right-022146small.png`,
disponibles dans le répertoire inclus avec ce TP. Ces images sont issues d'une caméra stéréo. Pour chaque descripteur de l'image de gauche, trouvez le feature dans l'image de droite qui a la plus petite distance de Hamming, donc le plus semblable. Ceci constituera l'appariement. Dans dans votre rapport, mettez une image représentant ces matches. Le fond sera l'image de gauche, et chaque ligne verte reliera le feature de gauche au feature de droite associé, comme dans l'image 7. Si vous avez trop de lignes, choisissez-en un nombre raisonnable de manière aléatoire (une centaine) pour affichage.

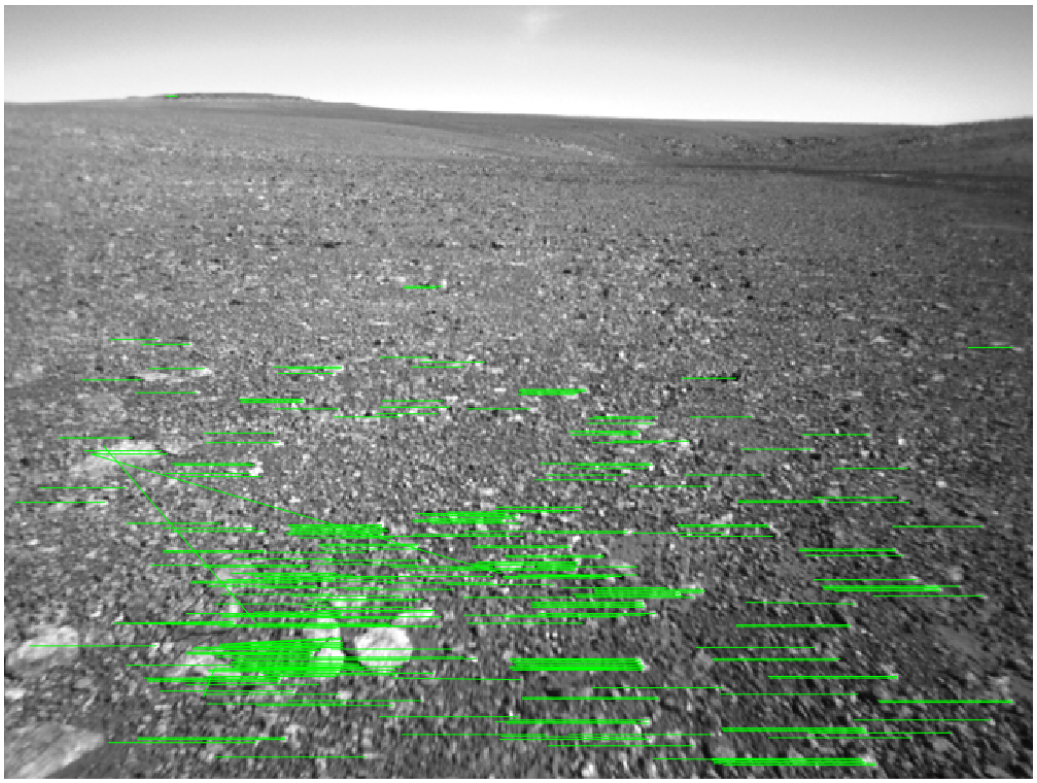


FIGURE 7 – Image montrant quelques appariements entre les features de l'image gauche (en fond) vers les features de l'image droite. Notez que j'ai épuré ces matches, afin de ne conserver que de très bons. Vous devriez avoir plus de lignes diagonales dans votre résultat.

5.4 Amélioration de la qualité des matchs GLO-7021 Seulement (8 pts)

Vous devriez constater que vous avez beaucoup de faux matches dans l'image. Tentez d'améliorer la qualité de ces matches, en utilisant des stratégies de votre choix. Commentez sur l'amélioration apportée par chacune de vos stratégies.