INP-ENSEEIHT

Projet long
-
Bibliographie

Puech Pierre
Pigneux Alice
Dary Jean-léo
Baudinaud Luc

Promo 2022

3 ème Année 2021/2022 — Semestre 9

Table des matières

1 Introduction

1.1 Présentation du projet

Le projet tourne autour d'un robot innovant nommé ARU [7] qui a pour but de pouvoir exécuter différentes actions très diversifiées. ARU n'a pas de "direction" privilégiée, à l'inverse d'un humain par exemple, qui est conçu pour "avancer". Actuellement, ARU est doté d'une unique caméra stéréo-photométrique propriétaire, ce qui limite ses capacités. Le but du projet est de concevoir un système de plusieurs caméras stéréos à placer autour de Aru pour lui donner un champ de vision à 360° et fusionner les images et nuages de points obtenus. Il y a un enjeu de performance CPU (ou GPU) à garder en mémoire au cours de la réalisation du projet puisque ce sont des algorithmes qui tournent en embarqué.

1.2 Présupposés

- On dispose de la position et de l'orientation des caméras.
- On dispose des paramètres de calibration et de distorsion de la caméra.

1.2.1 Définition des contraintes

- Synchronisation d'un réseau de caméras
- Enjeu de performance CPU (ou GPU) à garder en mémoire au cours de la réalisation du projet puisque ce sont des algorithmes qui tournent en embarqué.
- Reconstruction 3D dense à 360°

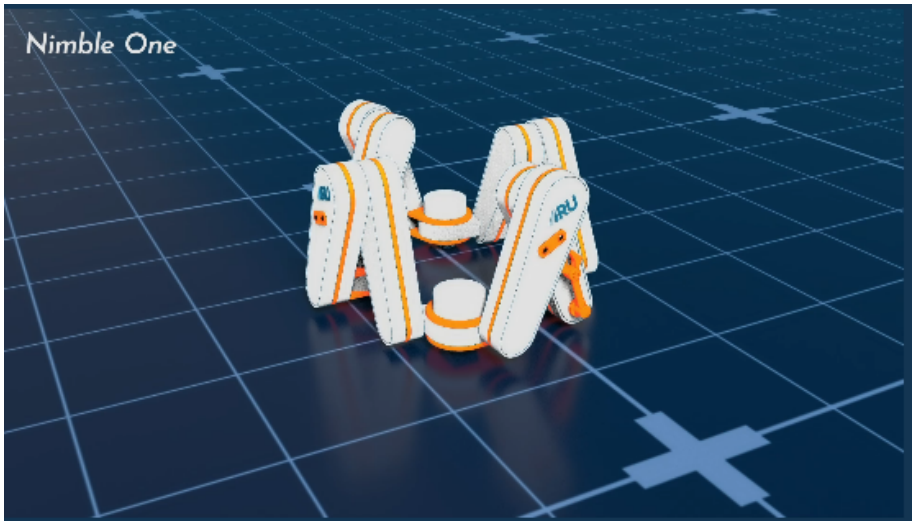

Figure 1: Présentation du robot ARU [7]

2 Mise en place d'un réseau de caméra

Le but de cette partie du projet est de mettre en oeuvre un réseau de caméra permettant de faire de la reconstruction 3D. On peut décomposer le problème en plusieurs points:

- Avoir un réseau de caméra. Il en faut assez pour recouvrir les 360° du champ de vision du robot. Cela nécessite de gérer plusieurs flux vidéo d'une dizaine de caméra.
- Pouvoir faire de la reconstruction 3D stéréoscopique. Il faut que les caméras soient synchronisées, 2 à 2 pour effectuer le traitement de chaque couple stéréoscopique mais aussi de manière générale pour pouvoir fusionner les nuages de points.

De cette recherche bibliographique, il en est ressorti 2 idées différentes :

- synchroniser de manière électronique les caméras.
- synchroniser les caméras connecter sur un réseau local en utilisant un protocole de synchronisation.

2.1 Synchronisation hardware

Le principe est d'utiliser la même horloge interne sur les 2 caméras. L'acquisition des 2 capteurs est donc synchronisée. Il faut ensuite récupérer les flux vidéos en gardant cette synchronisation. Une solution proposée par Arducam [3] est de fusionner les flux vidéo et il sera donc utilisé comme un seul flux par l'unité de traitement (RPI) comme montré sur la figure 2. Cette méthode a l'avantage d'être native à la conception du système.
Les caméra sont synchronisées à la perfection et le flux étant unique, une frame se suffit à elle même pour effectuer de la reconstruction. En revanche, on ne peut pas généraliser cette méthode pour obtenir un réseau d'une dizaine de caméra. Les solutions existantes permettre d'aller jusqu'à 4 caméras mais pas plus.

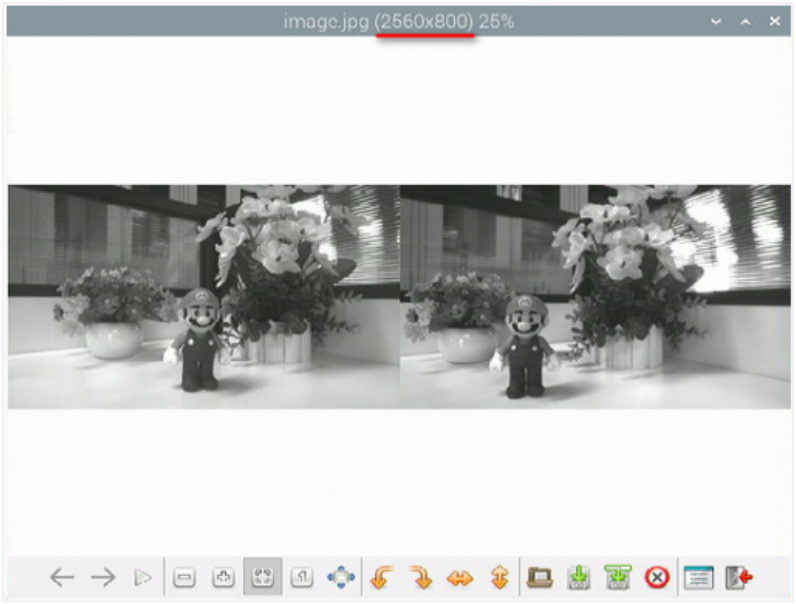

Figure 2: Exemple de fusion de flux vidéo de 1280x800

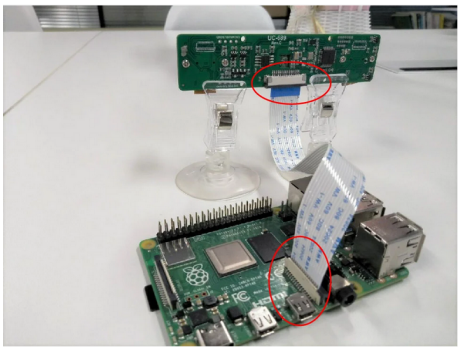

Figure 3: Module de caméra stéréoscopique de Arducam

2.2 Synchronisation de caméra sur un réseau LAN

Le principe se repose sur un protocole Ethernet de synchronisation d'horloge, le *Précision Time Protocol*. Les caméra seront synchronisées via la pll interne d'une unité de traitement (Raspberry Pi par exemple) sur le signal fourni par ce protocole. L'acquisition des flux vidéo est ainsi synchronisée et pourra être récupérer pour le post-traitement, la reconstruction 3D.

Chaîne de traitement:

- (camera board) acquisition de la frame
- (camera board) encodage de la frame
- (Ethernet) envoie de la frame à travers le reseau
- (stitcher board) alignement temporel de la frame
- (stitcher board) décodage de la frame
- (stitcher board) traitement de la frame (reconstruction)

2.2.1 Protocole PTP

Ce protocole fonctionne sur le principe de maître-esclave. L'horloge maître sert de référence temporelle. Pour garantir une synchronisation, il faut corriger:

- La dérive. Pour cela, deux signaux sont envoyés à la suite
  - $SYNC$ qui contient une estimation de l'heure d'émission les propriétés de l'horloge
  - $FOLLOW\_UP$ qui contient l'heure exacte d'émission qui servira à déterminer l'offset en soustrayant heure de réception à l'heure exacte.
- Le délais de transmission. On a ici aussi un procédé en 2 temps répété toutes les 2 à 30 secondes.
  - $DELAY\_REQ$ L'esclave envoie au maître une demande de délai. A l'émission de ce message, l'esclave note l'heure d'émission (t3) tandis que le maître note l'heure de réception (t4).
  - $DELAY\_RESP$ contenant l'heure à laquelle il a reçu la requête de délai (t4). Ainsi l'esclave peut calculer le délai moyen: $ds2m = t4 - t3$

2.2.2 Mise en oeuvre

Ici est présenté un exemple de mise en oeuvre de ce genre de synchronisation entre plusieurs Raspberry pi utilisant les modules RPI camera v2. Il faut modifier le schéma classique d'acquisition des frames d'une caméra en utilisant une pll pour aligner la fréquence d'acquisition sur l'horloge interne ($sysclk$ sur le schéma 4) de linux qui a lui été synchronisé via le protocole PTP.

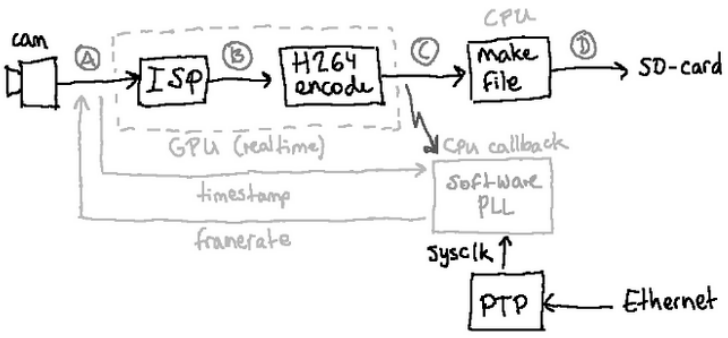

Figure 4: Schéma montrant la mise en oeuvre de la synchronisation de l'acquisition d'une caméra sur un protocole PTP dans le cadre d'une fusion 360 de flux vidéo [6]

2.2.3 Performances

D'après [1], on peut s'attendre théoriquement à des performance de synchronisation de l'ordre de la microseconde. De la latence peut apparaître dans le traitement a posteriori de ce protocole mais cela devrait rester assez fiable si le réseau n'est pas trop encombré. Peu de détails ou d'étude sont proposés pour savoir à quelle point le trafic réseau peut impacter le fonctionnement de ce protocole. Des stress tests sont donc à prévoir.

Un point négatif de cette méthode est quelle nécessite une unité de traitement telle qu'une Raspberry Pi pour chaque caméra. La gestion des paires d'images va aussi être une complexité de plus a prendre en compte pour s'assurer qu'un mélange des différentes frames n'apparaisse pas.

2.3 Bilan de la partie matériel

Si l'on dresse un bilan, on a une solution qui permet d'avoir des caméras synchronisées à la perfection car utilisant la même horloge interne. Mais cette méthode n'est pas généralisable à un réseau aussi grand que celui de notre application.
Dans un deuxième temps, on a une autre solution ou le nombre de caméra n'est pas un facteur limitant mais une unité de traitement est nécessaire par caméra.

On peut donc envisager de fusionner ces méthodes en utilisant un réseau de couple de caméra stéréoscopique. Chaque unité de calcule du réseau serait équipée d'une paire de caméras synchronisées au niveau hardware comme expliqué dans la première méthode. Chaque unité serait ensuite synchronisée via le protocole PTP pour que la capture des différents couples se fasse en même temps. La chaîne de traitement devient alors :

- (camera board) acquisition de la frame
- (camera board) traitement de la frame pour obtenir un nuage de points
- (Ethernet) envoie du nuage à travers le réseau
- (stitcher board) alignement temporel du nuage
- (stitcher board) fusion du nuage avec ceux des autres couple stéréoscopique

Dans cette situation, le calcul de la reconstruction est réparti sur chaque unité de traitement ce qui permet de pleinement les utiliser et seul les nuages de points sont envoyés sur le réseau pour être fusionnés par la suite. Le protocole PTP permet de synchroniser la capture des couples de caméra stéréoscopique et ainsi horodater les nuages de points. Lorsqu'ils seront récupérés par l'unité centrale, cette horodatage sera utilisé pour fusionner les nuages de points correspondant entre eux.

3 Reconstruction 3D

3.1 Modèle géométrique

[4]

Les méthodes adaptées à la reconstruction 3D à partir de plusieurs caméras avec des positions différentes sont les méthodes de stéréovision. Ces méthodes utilisent toutes un modèle géométrique similaire pour modéliser une caméra :

La caméra est modélisée par le **modèle sténopé**, constitué d'un centre optique, d'un axe optique, et d'un plan image sur lequel sont projetés points 3D.

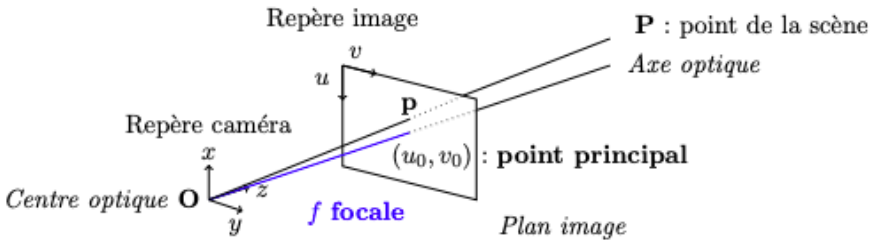

Figure 5: Modèle sténopé

Pour calculer la position de la projection d'un point 3D sur le plan image, on utilise la **projection perspective** :

$$\boxed{\lambda(u\ v\ 1)^T = K(x\ y\ z)^T} \tag{1}$$

Avec x, y, z les coordonnées du point 3D dans le repère caméra, et u, v les coordonnées de sa projection dans le plan image.

K est la matrice de calibration, supposée connue :

$$K = \begin{pmatrix} f_x & 0 & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{pmatrix} \tag{2}$$

Les matrices de rotation et de translation de la caméra par rapport à un repère monde étant connues, on peut également donner la relation en fonction des coordonnées X, Y, Z dans le repère monde, ce qui permet d'utiliser plusieurs caméras facilement:

$$(x\ y\ z)^T = R(X\ Y\ Z)^T + t \tag{3}$$

d'où

$$(x\ y\ z\ 1)^T = A(X\ Y\ Z\ 1)^T \tag{4}$$

avec :

$$A = \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{5}$$

Ce qui nous donne la relation finale :

$$\boxed{\lambda(u\ v\ 1)^T = M(X\ Y\ Z\ 1)^T} \tag{6}$$

avec :

$$M = (K0_{3\times1})A \tag{7}$$

3.2 Contrainte épipolaire

La contrainte épipolaire correspond au fait que le point d'une deuxième image correspondant à un certain point p dans une image, se trouve forcément sur une droite particulière appelée droite épipolaire.

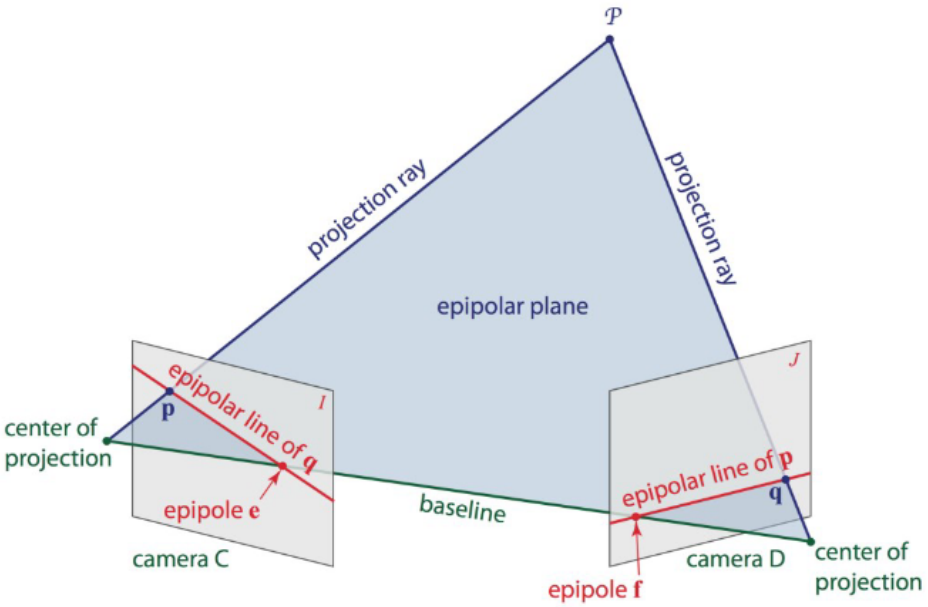

Figure 6: Géométrie épipolaire

La droite épipolaire permet donc d'accélérer la recherche de correspondances entre deux images, puisqu'il ne faut chercher que le long de la droite épipolaire correspondant au point de la première image.

Pour accélérer les calculs, les méthodes de stéréovision ont recours à une rectification épipolaire [2], qui consiste à appliquer une transformation géométrique aux deux images de sorte que les droites épipolaires soient toutes parallèles et horizontales et que deux pixels correspondant soient sur la même ligne dans les deux images, ce qui facilite considérablement la recherche de correspondances.

Pour cela, on commence par appliquer des rotations de sorte que les deux plans images soient parallèles à la droite joignant les deux centres optiques, puis on applique une deuxième rotation afin de ramener l'axe des x de chaque caméra sur la droite joignant les centres optiques.

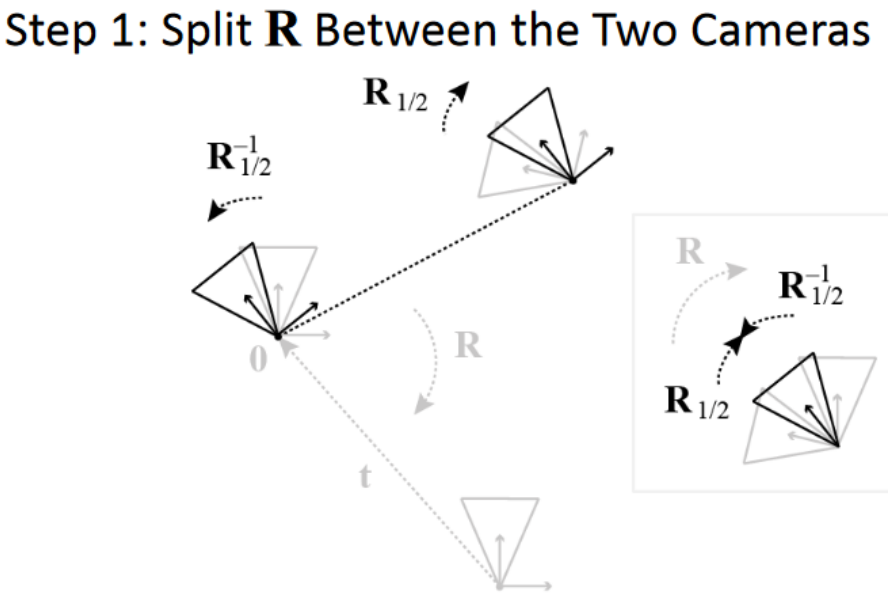

Figure 7: Rectification épipolaire 1

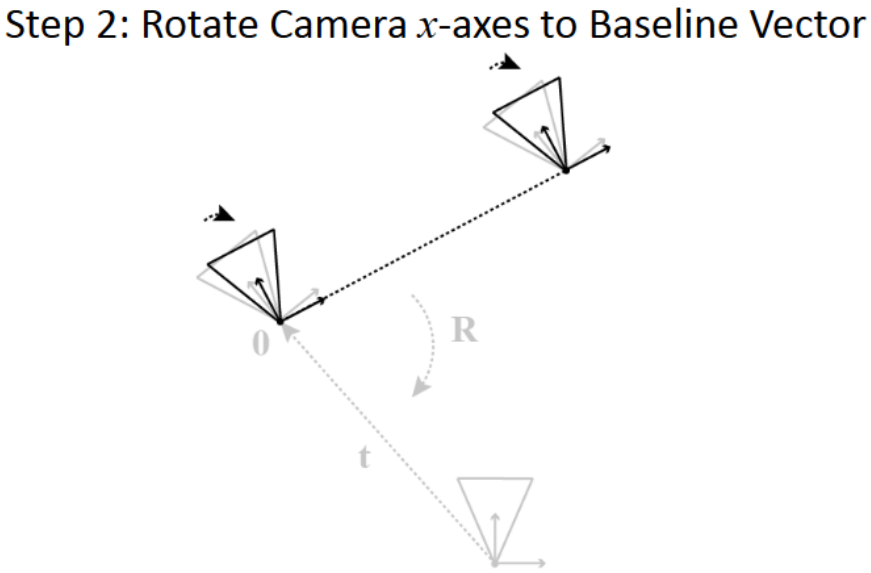

Figure 8: Rectification épipolaire 2

Il faut également appliquer une rectification pour enlever la distorsion crée par la caméra. Pour cela on utilise des paramètres estimés lors de la calibration de la caméra.

Une fois ces différentes rectifications effectuées, on peut finalement passer à la mise en correspondance.

3.3 Mise en correspondance

3.3.1 Présentation

La mise en correspondance est le premier pars vers la reconstruction. On veut retrouver dans deux images les points homologues, c'est-à-dire les paires de points ayant la même coordonnée 3D dans le repère monde.

Dans le cas d'images auxquelles on a appliqué une rectification épipolaire, le problème de la mise en correspondance revient à calculer une fonction de disparité pour chaque pixel $p_g^{i,j}$ de l'image, c'est à dire l'écart entre les abscisses de deux pixels homologues, leurs ordonnées étant identiques grâce à la rectification:
Si p_g et p_d sont des points homologues dans les images de gauche et de droite, alors :

$$d(p_g) = v_{p_d} - v_{p_g} \tag{8}$$

En calculant cette disparité pour chaque pixel de l'image de référence, on obtient une carte de disparités, comme l'exemple de l'image ci-dessous. Les niveaux de gris correspondent à la disparité de chaque pixel.

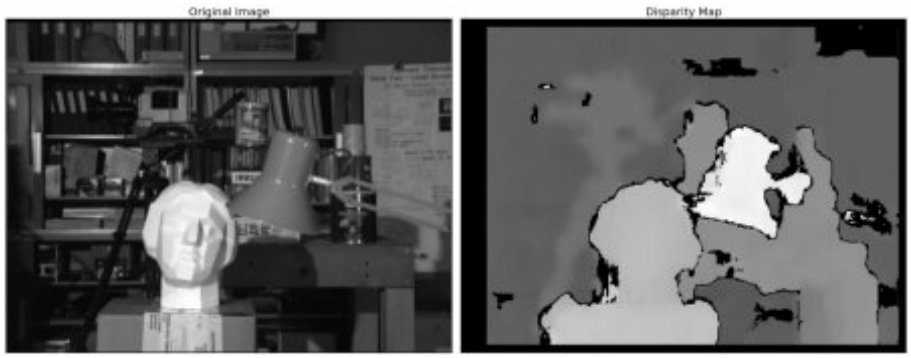

Figure 9: Exemple de carte de disparités

Il existe plusieurs méthodes de mise en correspondance : les approches **locales** et les approches **globales**. Cependant l'ensemble des méthodes suivent les étapes suivantes [8]:

1. **Calcul du coût de l'appariement** (correspondance) : Ce coût permet de déterminer si deux pixels sont bien associés au même point 3D dans une scène. Ce coût est calculé pour chaque pixel de l'image avec tous les autres pixels de l'autre image. La contrainte épipolaire expliquée plus haut est importante car elle permet de faire une recherche unidimensionnelle horizontale pour chaque pixel.
2. **Agrégation des coûts** (support) : Pour un pixel donné dans l'image 1 on prend une fenêtre 3x3 par exemple et on cherche dans l'image 2 la fenêtre qui minimise le coût entre les deux fenêtres de pixels considérées. Puis on choisit le pixel au centre de la fenêtre de l'image 2 répondant à ce critère de minimisation.

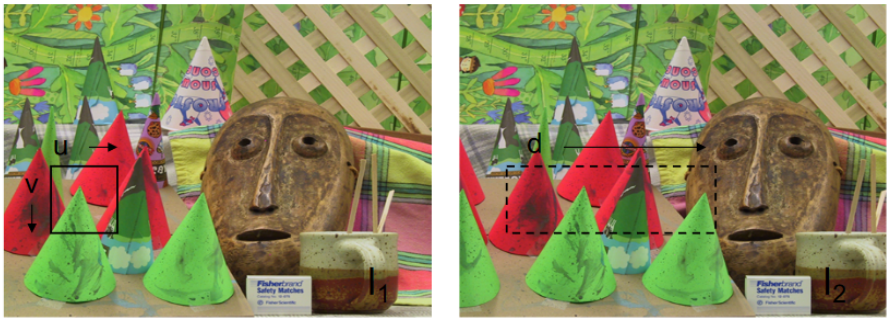

Figure 10: Agrégation des coûts par région

3. **Calcul et optimisation de la disparité** (diffère selon les approches) :
   - Approches locales :
     - Choix de la mesure de corrélation pour le calcul d'erreur entre les pixels
     - Détermination de la zone d'agrégation
     - Choix de la méthode de recherche exhaustive
   - Approches globales : A la différence des méthodes locales, les méthodes globales font intervenir l'ensemble des pixels des images dans la procédure d'optimisation. La plupart des méthodes gloables utilisent l'optimisation par programmation dynamique.
4. **Raffinement de la disparité** : En général, l'étape de raffinement consiste à régulariser et à remplir ou interpoler des occlusions.

3.3.2 Différentes méthodes

Passons maintenant à la présentation de différentes méthodes d'appariement de points (mise en correspondance) [8].

Approches locales

- Méthode de mise en correspondance par blocs: On cherche quelle région de la seconde image minimise l'erreur avec la région de l'image de référence. Il existe différentes métriques pour le calcul d'erreur :

Match metric	Definition
Sum of Absolute Differences (SAD)	$\sum_{u,v} \lvert I_l(u,v) - I_r(u+d,v) \rvert$
Sum of Squared Differences (SSD)	$\sum_{u,v} (I_l(u,v) - I_r(u+d,v))^2$
Normalized Cross-Correlation (NCC)	$\frac{\sum_{u,v}(I_l(u,v) - \bar{I}_l) \cdot (I_r(u+d,v) - \bar{I}_r)}{\sqrt{\sum_{u,v}(I_l(u,v) - \bar{I}_l)^2 \cdot (I_r(u+d,v) - \bar{I}_r)^2}}$
Rank	$\sum_{u,v} (I'_l(u,v) - I'_r(u+d,v))$ $I'_k(u,v) = \sum_{m,n} I_k(m,n) < I_k(u,v)$
Census	$\sum_{u,v} HAMMING(I'_l(u,v), I'_r(u+d,v))$ $I'_k(u,v) = BITSTRING_{m,n}(I_k(m,n) < I_k(u,v))$

Figure 11: Différentes métriques pour le calcul d'erreur

- Méthode du gradient : Les méthodes basées sur les gradients estiment le mouvement par l'analyse des différences de luminosité entre les régions analysées. Ces variations sont modélisées par des équations différentielles représentées par des gradients spatiaux et temporels. On définit l'hypothèse suivante : la luminosité d'un pixel est la même sur chaque image.
- Méthode de mise en correspondance des caractéristiques : Il existe deux types de mise en correspondance par caractéristiques
  - Mise en correspondance par hiérarchie: il faut détecter des éléments remarquable à appareiller. La plus connue [9] utilise 4 éléments : surfaces, ligne, sommets et bords. On apparie d'abord les surfaces (le plus facile puis les les lignes sommet et bords dans cet ordre)
  - Mise en correspondance par segmentation : [11] on peut pré-segmenter l'image pour mettre en correspondance que des objets entre eux par transformation affine des régions segmenter entre les 2 images.

La méthodes sont plus ou moins adaptées. En effet la méthode par gradient est sensible aux discontinuités de profondeur ce qui peut être facilement le cas dans une maison. Les 2 autres méthodes donnent de meilleurs résultats mais ces 2 dernières nous intéressent moins car elles

ne mettent en correspondance que certains points. Cependant cela pourrait être un point de départ pour ensuite faire une reconstruction dense.

Approches globales
[10]

Une méthode est dites globale lorsque la fonction de coût est évaluée sur l'ensemble de l'image. On part du principe que des points voisins dans un image seront voisins dans l'autre image également, et on cherche à rendre la fonction de disparité la plus constante possible.
Pour cela on l'énergie suivante sur l'image, considérée comme un graphe (V, E), avec des arêtes entre les pixels voisins (8-voisinage ou 4-voisinage):

$$E(D) = \sum_{p \in V} C_p(D_p) + \sum_{(p,q) \in E} S(D_p, D_q) \tag{9}$$

où $C_p(d)$ représente le coût d'appariement du pixel p pour une disparité d, et $S(D_p, D_q)$ est un terme qui pénalise de grands écarts de disparité entre pixels voisins.

Ce problème est NP-complet, il est donc in-envisageable de le résoudre de manière exacte. Diverses méthodes heuristiques sont utilisées pour résoudre ce problème, telles que des méthodes utilisant des coupures de graphes ou la propagation de convictions. Cependant ces méthodes donnent de très bons résultats pour la reconstruction mais sont trop lentes pour êtres utilisées dans notre application.

Approches semi-globales

Afin d'utiliser les contraintes de continuité des approches globales pour du temps réel, il existe des approches semi-globales, qui cherchent à imposer la continuité uniquement dans certaines directions. Ces méthodes utilisent la programmation dynamique.

- Contrainte uniquement sur les lignes[9]: On cherche le chemin "optimal" dans la matrice des appariements possible entre les points de deux lignes correspondantes dans les deux images. On calcule le coût local en chaque points (par les mise en correspondance locales). On pénalise les occultations lorsqu'on assigne un groupe de pixels à un seul autre pixel en pénalisant le coût global du chemin. En procédant de cette manière, on assure que l'ordre des pixels homologues est le même dans les deux images. Voici un schéma ici de la thèse [9]:

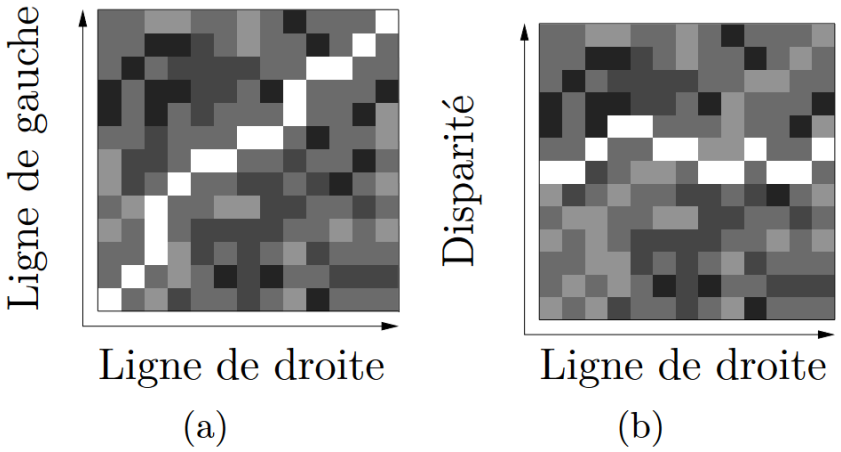

Figure 12: exemple d'une représentation de l'espace des disparités en représentant les lignes droite-gauche (a), ou en utilisant la représentation ligne droite-disparité (b). L'intensité représente le coût de la mise en correspondance potentielle le long de la ligne de recherche, un pixel blanc représentant un faible coût.

Cependant cette méthode ne prend pas en compte la continuité entre des lignes différentes, ce qui crée des défauts dans la reconstruction.

- L'algorithme de Semi-Global Matching [5] cherche à pallier à ce problème en assurant cette continuité dans un certain nombre de directions différentes :
  La régularisation prend cette forme :

$$S(d,d') = \begin{cases} 0 & si\ d = d' \\ P1 & si\ |d - d'| = 1 \\ P2 & sinon \end{cases} \tag{10}$$

Cette forme permet de mettre une pénalisation moindre aux endroits ou la disparité varie peut, qui correspondent probablement à des pente. Le fait d'avoir une pénalisation fixe si on a un saut de disparité permet de ne pas trop pénaliser les occultations, qui correspondent à des sauts de profondeur et donc de disparité.

On définit ensuite le coût en un pixel suivant la direction **r** (haut, gauche, haut-gauche, etc)

$$L_r(p,d) = C_p(d) + \min_{d' \in D}(L_r(p-r,d') + S(d',d)) \tag{11}$$

avec D l'ensemble de valeurs possible de disparité.

On calcule ces coûts en chaque pixel, pour chaque valeur de disparité, et pour chaque direction, en faisant une traversée de notre image de haut en bas et de gauche à droite, et une traversée dans le sens inverse, puis on les somme pour avoir le coût total de chaque pixel. On prend ensuite pour chaque pixel la disparité qui donne le coût total le plus faible.

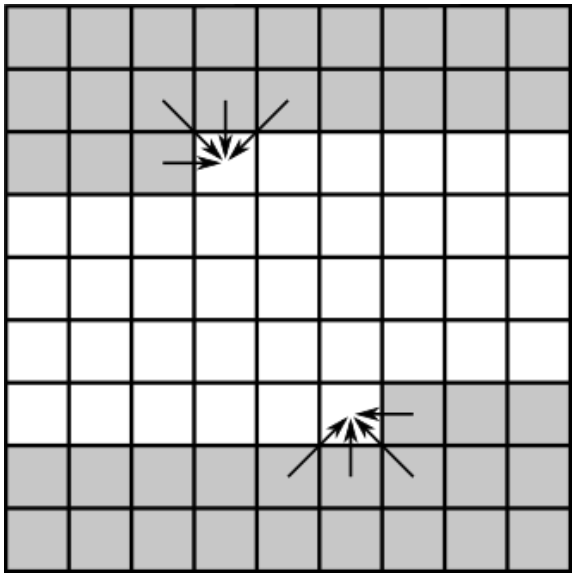

Figure 13: Illustration du calcul des coûts en deux passes sur l'image

3.4 Reconstruction 3D

Nous avons maintenant des paires de pixels homologues. On va donc chercher à trouver les points 3D de chaque paire homologue obtenue. Ce qui permettra à la fin d'obtenir la reconstruction 3D désirée.
Tout d'abord, on va chercher à résoudre l'équation suivante au sens des moindres carrés pour chaque paire de pixels :

$$(\widehat{Z_1}, \widehat{Z_2}) = argmin_{(Z_1,Z_2)}||Z_2 w_2 - R_{1->2}(Z_1 w_1) + t_{1->2}||^2 \tag{12}$$

- Les matrices $R_{1->2}$ et $t_{1->2}$ sont connues car on connaît les positions des caméras dans le repère monde.
- w_1 et w_2 sont les coordonnées des pixels dans les repères images R_1 et R_2 qui sont les repères caméra par rapport aux repères monde.

La solution de ce problème nous donne la paire de points 3D les plus proches situés sur les deux lignes de visée. La méthode de triangulation dite du point milieu consiste à positionner le point Q recherché au centre du segment défini par ces deux points 3D. Les coordonnées de Q dans le repère R2 s'écrivent donc :

$$Q_2 = \frac{1}{2}[\widehat{Z_2} w_2 + R_{1->2}(\widehat{Z_1} w_1) + t_{1->2}] \tag{13}$$

4 Conclusion

Nous avons donc vu que l'algorithme global de reconstruction est commun à toutes les méthodes, mais que de nombreuses possibilités existent pour la mise en correspondance. Les méthodes globales étant trop lentes pour notre application, nous allons commencer par utiliser une méthode locale par bloc, en testant différentes métriques, puis, si les résultats ne sont pas assez bons, nous utiliserons la méthode de Semi-Global Matching pour affiner la reconstruction.

Sur le plan matériel, plusieurs méthodes sont aussi possibles mais une combinaisons des méthodes est possible, pour combiner les avantages de chacune. L'idée est donc de concevoir un réseau de couples de caméras stéréoscopique qui fonctionneraient indépendamment comme un capteur de profondeur. Ces capteurs sont synchronisés au seins de ce réseaux pour pourvoir fusionner leurs mesures sans problème de décalage temporelle entre les nuages de points.

References

[1] ALBEDO. "Precision Time Protocol (PTP) explained". In: (). DOI: https://www.albedotelecom.com/src/lib/WP-PTP.pdf.

[2] Alessandro Verrin Andrea Fusiello Emanuele Trucco. "A compact algorithm for rectification of stereo pairs". In: *Machine Vision and Applications (2000)* ().

[3] *Camarray – Arducam 1MP*2 Stereo Camera MIPI Module.* URL: https://www.arducam.com/docs/cameras-for-raspberry-pi/multi-camera-adapter-board/1mp-stereo-camera-global-shutter-mipi-arducam/.

[4] Sylvie Chambon. "Mise en correspondance stéŕeoscopique d'images couleur en pŕesence d'occultations". In: *Doctorat de l'Universit e Paul Sabatier – Toulouse II* (). DOI: http://thesesups.ups-tlse.fr/5/1/Chambon_Sylvie.pdf.

[5] Heiko Hirschmüller. "Stereo Processing by Semiglobal Matching and Mutual Information". In: *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE* ().

[6] Vincent Jordan. *360° Video Stitching With Raspberry Pi Camera and OpenCV*. 2020. URL: https://medium.com/swlh/360-video-stitching-with-raspberry-pi-and-opencv-30549b37acbc.

[7] NimbleOne. *ARU service robot.* URL: https://nimbleone.io/.

[8] Roberto Pisapia. "Disparity map extraction for a low cost 3D sensor". In: *Università degli Studi di Salerno - Université de Lorraine* (2015), pp. 33–56.

[9] Christophe Rabaud. "Une nouvelle approche de mise en correspondance stéréoscopique dense par méthodes possibilistes". In: *Université Montpellier II - Sciences et Techniques du Languedoc* (2005), pp. 31–48. DOI: https://tel.archives-ouvertes.fr/tel-00262287/document.

[10] Richard Szeliski et al. "A Comparative Study of Energy Minimization Methods for Markov Random Fields with Smoothness-Based Priors". In: *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. 30* ().

[11] Birchfield et Tomasi. "Multiway Cut for Stereo and Motion with Slanted Surfaces". In: *nt'l Joint Conf. Artificial Intelligence,* (1999), pp. 2–3. DOI: https://users.cs.duke.edu/~tomasi/papers/birchfield/birchfieldIccv99.pdf.