

ROB317 - TP1

Détection et Appariement de Points Caractéristiques

Compte-rendu

Juliette DRUPT - Caroline PASCAL

1 Format d'images et Convolutions

Q1 : La fonction utilisée pour la lecture d'images est la fonction OpenCV `cv2.imread`, qui prend en entrée le chemin d'accès - *path* - de l'image à lire et, de façon optionnelle, la manière dont l'image sera chargée (0 : image en valeurs de gris, 1 : image en couleurs,...). L'image lue est alors stockée dans un `numpy.ndarray` contenant la ou les valeur(s) associée(s) à chaque pixel. Dans le code fourni, l'image lue et également convertie en `float64` avant traitement.

Pour copier l'image, on fait ici appel à la fonction `cv2.copyMakeBorder`, qui copie l'image en y ajoutant éventuellement des bordures de façon artificielle. Ces bordures permettent par exemple de limiter les effets de bords lors de l'application de filtres de convolution. Il est possible de préciser la taille de chacune des bordures sur les quatres arêtes de l'image, ou de toutes les fixer à 0 pour réaliser une simple copie de l'image, comme c'est le cas dans le code fourni. L'option `BORDER_REPLICATE` signale que l'on souhaite répliquer les pixels situés sur les bords de l'image sur les bordures ajoutées.

Pour l'affichage, on utilise la fonction `matplotlib.pyplot.subplot` pour afficher plusieurs images dans une même fenêtre, la fonction `matplotlib.pyplot.imshow` pour indiquer l'image à afficher, et finalement la fonction `matplotlib.pyplot.show` pour afficher le tout.

L'appel de la fonction `imshow` peut être accompagné de nombreux arguments optionnels, parmi lesquels :

- `cmap` qui définit la manière dont les valeurs du `numpy.ndarray` décrivant l'image doivent être traduites en couleurs ;
- `vmin`, `vmax` qui renseignent les valeurs minimales et maximales de l'échelle de couleurs que doit couvrir `cmap`. Par défaut, les valeurs de `vmin` et `vmax` sont fixées selon les valeurs extrêmes contenues dans l'image à afficher.

Q2 : Le noyau de convolution utilisé en exemple est :

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad (1)$$

Ce noyau de convolution calcule la somme de l'image et de son Laplacien et permet le réhausser de contraste de l'image, comme on peut le voir sur l'image résultante, présentée en Figure 1.

En théorie, cet effet peut s'expliquer grâce aux propriétés du Laplacien d'une image. Supposons que l'image traitée soit représentée en niveaux de gris, sur une plage de valeurs allant de 0 à 255. Le Laplacien correspondant à la somme des dérivées secondes en *x* et en *y* de l'image, sa valeur sera nulle dans les zones où la teinte de l'image est constante. Dans ces zones, le noyau de convolution précisé ci-dessus n'aura pas d'effet.

A l'inverse, dans les zones où la teinte de l'image varie, la valeur du Laplacien sera négative du côté "sombre" et positive du côté "clair". En conséquence, en ajoutant l'image intiale à son Laplacien, les teintes claires vont être éclaircies (effet "positif", augmentant la valeur des pixels vers 255), et les teintes sombres assombries (effet "négatif", ramenant la valeur des pixels vers 0).

En somme, sur les zones où la teinte de l'image varie le contraste entre les parties sombres et claires sera accentué, tandis que les zones uniformes ne seront pas impactées : le contraste de l'image obtenue est bien réhaussé par rapport à celui de l'image originale.

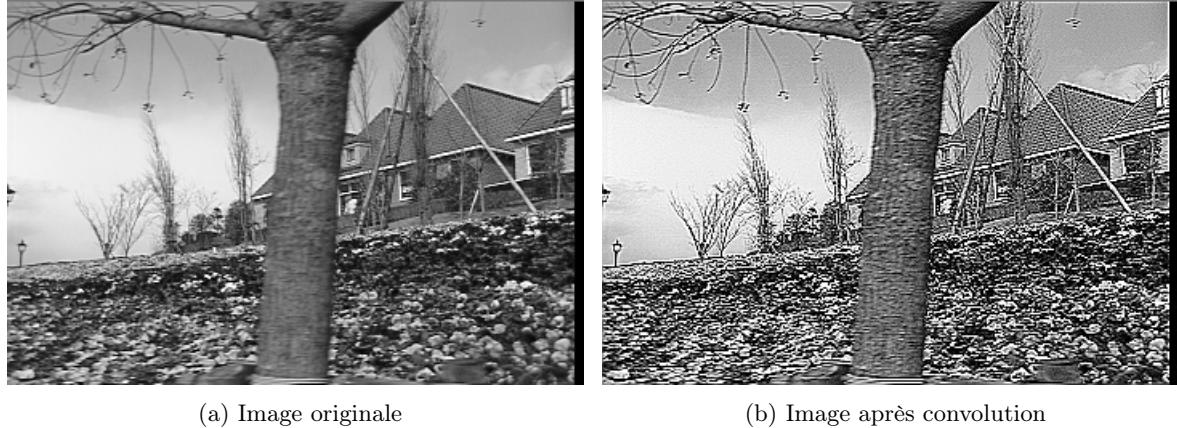


FIGURE 1 – Réhaussement de contraste

Q3 : On va approximer les dérivées partielles $I_x = \frac{\partial I}{\partial x}$ et $I_y = \frac{\partial I}{\partial y}$ à l'aide de filtres de Sobel. Les noyaux de convolutions correspondants sont donc :

$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad (2)$$

pour I_x et

$$\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \quad (3)$$

pour I_y . On peut ainsi calculer le module du gradient : $\|\Delta I\| = \sqrt{I_x^2 + I_y^2}$. La Figure 2 montre la dérivée partielle I_x et le module du gradient obtenus.

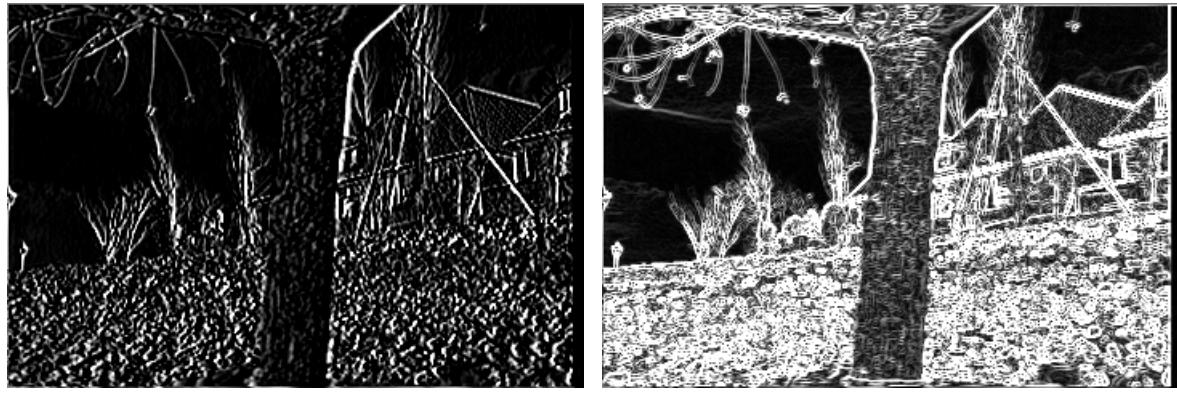


FIGURE 2 – Dérivée partielle I_x et gradient

Pour garantir un affichage correct avec la méthode directe, on doit prendre en compte l'indication d'OpenCV pour la représentation des images. Ainsi, la rangée de pixels supérieurs a une ordonnée de $x - 1$ (convention "matrice") et non de $x + 1$ (convention "repère $\{x, y\}$ "), et chaque pixel est d'abord indiqué par y , puis par x . Il est également indispensable de ramener les valeurs calculées pour chaque pixel à une valeur "correcte" et affichable de niveaux de gris, c'est-à-dire, comprise entre 0 et 255. Pour ce faire, on sature les valeurs obtenues lorsqu'elles sont plus grandes que 255 (saturation à la valeur 255), ou plus petites que 0 (saturation à la valeur 0).

2 DéTECTEURS

Q4 : La dilatation morphologique étend les zones claires de l'image dans un voisinage défini par un élément structurant. Ainsi, un pixel qui s'est éclairci suite à la dilatation est en fait voisin d'un pixel qui était plus clair que lui dans l'image d'origine. Ce pixel ne correspond donc pas à un maximum local dans l'image originale. C'est de cette manière que la dilatation morphologique participe au calcul des maxima locaux de la fonction d'intérêt **Theta** : elle permet d'éliminer de la liste des maxima potentiels les voisins des vrais maxima. Ceux-ci n'auraient en effet pas pu être simplement éliminés par seuillage, puisqu'ils peuvent avoir des valeurs très proches des maxima dont ils sont voisins.

Q5 : Théoriquement, la fonction d'intérêt de Harris est calculée en chaque pixel de l'image grâce à la relation suivante :

$$\Theta = \det(M) - \alpha \operatorname{Tr}(M)^2 \quad (4)$$

Où M est la matrice d'auto-corrélation de l'image, calculée au niveau du pixel considéré, et moyennée sur une fenêtre (dite fenêtre de sommation) dont la taille est caractérisée par un écart-type σ . Le paramètre α , appelé paramètre de Harris, est généralement fixé de manière empirique entre 0,04 et 0,06.

Les points anguleux situés sur l'image correspondent aux pixels pour lesquels la matrice d'auto-corrélation a deux grandes valeurs propres : λ_1 et λ_2 .

Ainsi, dans la base formée par les vecteurs propres de la matrice, l'équation 4 s'écrit sous la forme :

$$\Theta = \lambda_1 \lambda_2 - \alpha(\lambda_1 + \lambda_2)^2 \quad (5)$$

Nous noterons que ces deux valeurs propres sont positives, car la matrice d'auto-corrélation est une matrice réelle définie positive.

Nous pouvons alors distinguer trois cas de figure :

- $\Theta \ll 1$, dans ce cas là, les deux valeurs propres sont faibles, et le pixel est situé dans une zone homogène de l'image ;
- $\Theta < 0$, signale que l'une des valeurs propres est très grande devant l'autre. Dans cette situation, le terme $\alpha(\lambda_1 + \lambda_2)^2$, dans lequel l'effet des deux valeurs propres est décorrélé, est prépondérant sur le terme $\lambda_1 \lambda_2$, où leurs effets sont fortement corrélés. En pratique, ce genre de situation signale la présence d'un bord, ou d'une ligne dans la fenêtre de sommation, au niveau du pixel considéré ;
- $\Theta \gg 1$, dans cette situation, les deux valeurs propres sont toutes les deux grandes, et le terme corrélé prend le dessus sur le terme décorrélé. Lorsque les deux valeurs propres sont importantes, le pixel étudié est situé au niveau d'un coin situé dans la fenêtre de sommation.

Cette propriété de la fonction d'intérêt de Harris permet alors de détecter les pixels correspondants à des coins, ou à des points anguleux, en recherchant les maxima locaux de cette fonction sur l'ensemble de l'image.

En conséquence, la taille de la fenêtre de sommation et la valeur du paramètre α peuvent grandement impacter cette détection.

La taille de la fenêtre de sommation est directement reliée à l'échelle à laquelle les points d'intérêt seront détectés. En effet, si la fenêtre de sommation est trop grande, un maximum local de la fonction d'intérêt sera noyé par des informations globales parasites : présence de lignes, de coins dont la concavité est inversée,... En conséquence, peu de maxima locaux seront détectés, et la pertinence de ces derniers ne sera pas toujours assurée. A l'inverse, une fenêtre de sommation trop petite peut mener à un nombre de points détectés très

faible, car réduits à une échelle très petite. En somme, il est nécessaire de choisir la taille de la fenêtre de sommation en adéquation avec l'échelle des points caractéristiques à détecter.

L'effet du *paramètre* α est un peu plus subtil. Ce paramètre permet de quantifier la manière dont la décorrélation entre les valeurs propres est prise en compte lors du calcul de la fonction d'intérêt de Harris. Par exemple, si la valeur choisie pour α est très faible, le terme décorrélé de l'équation 4 aura peu de poids dans le calcul, et Θ aura donc plus tendance à prendre de grandes valeurs. En conséquence, il y a un risque de détecter des angles là où il n'y a en réalité qu'une ligne. A contrario, si α est très grand, le terme décorrélé aura un très grand poids dans le calcul, et Θ risque d'être souvent négatif. Dans ce cas, le risque est de confondre une ligne avec un angle, et donc de ne pas bien détecter les points anguleux de l'image.

La Figure 3 montre les résultats du détecteur de Harris pour différentes tailles de fenêtre de sommation, avec le paramètre de Harris $\alpha = 0.06$, constant, et l'écart-type $\sigma = 1$ constant également.

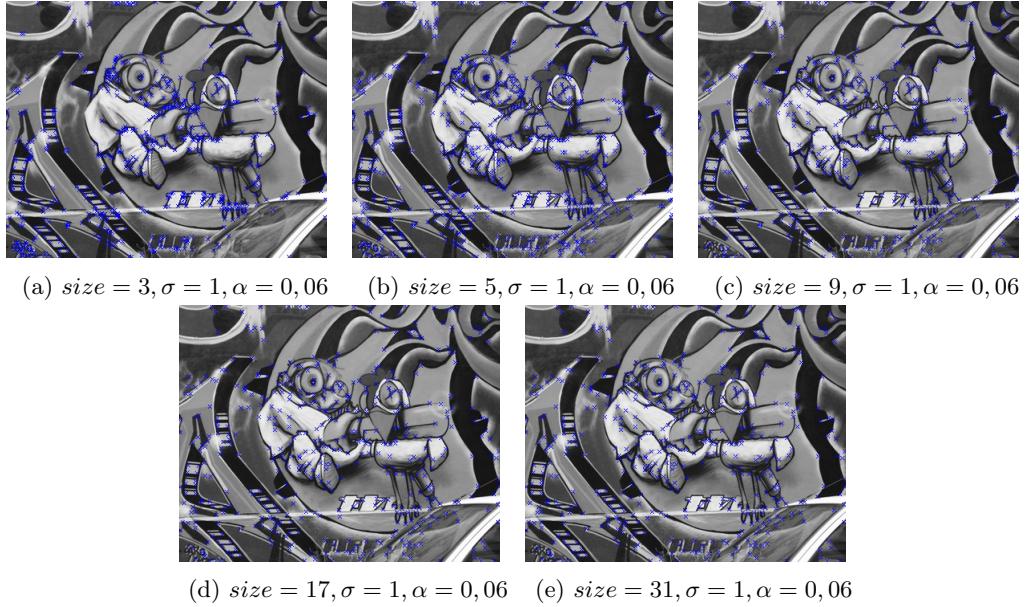
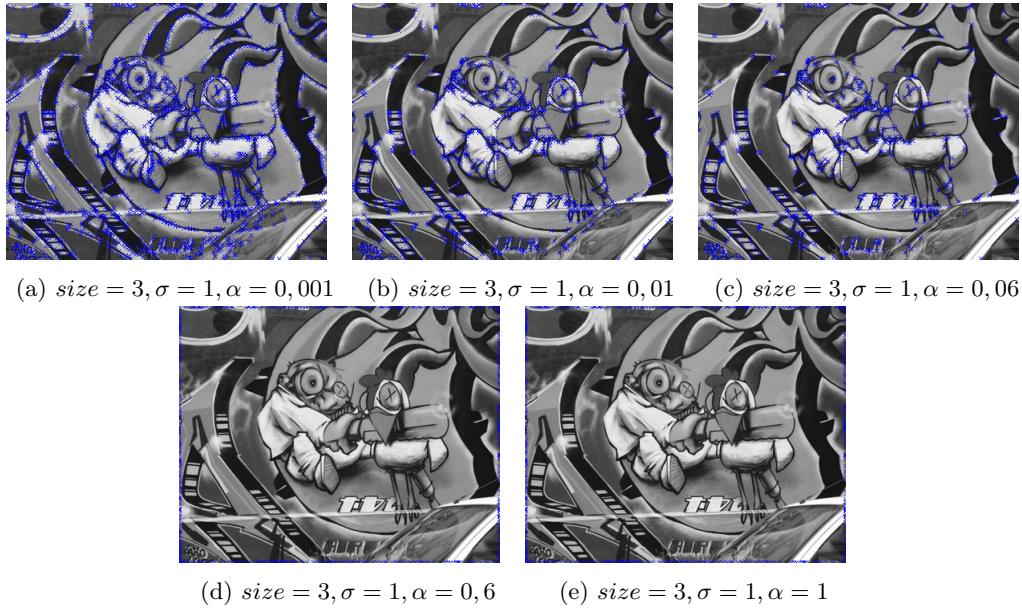


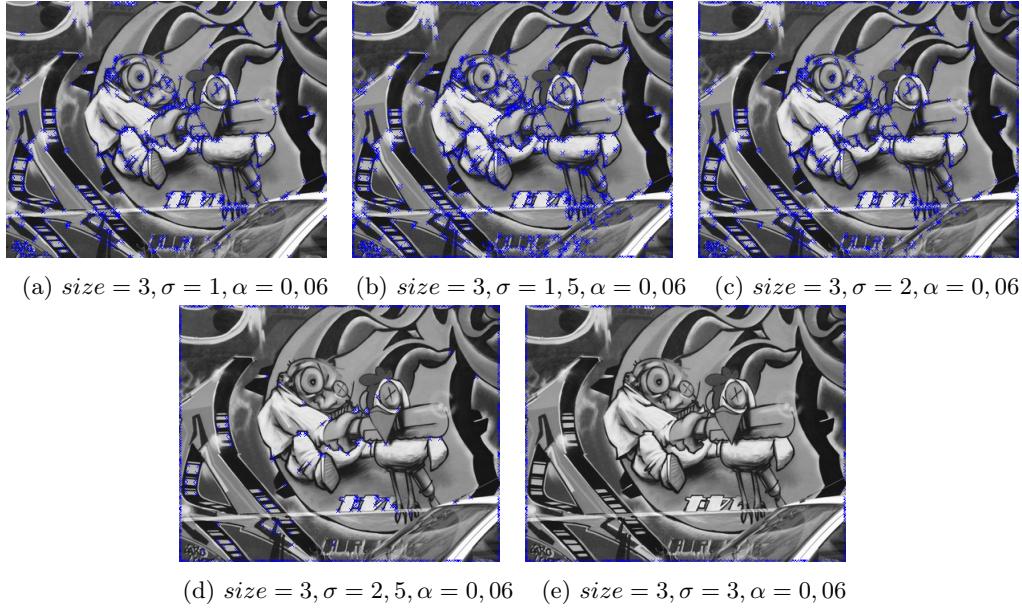
FIGURE 3 – Résultats du détecteur de Harris pour différentes tailles de fenêtre

On constate que l'augmentation de la taille de la fenêtre a pour effet de diminuer le nombre de points d'intérêt retenus.

La Figure 4 montre cette fois les résultats du détecteur de Harris pour différentes valeurs du paramètre de Harris α , avec une taille de fenêtre constante égale à 3 et l'écart-type $\sigma = 1$. On constate que plus α est petit et plus le nombre de points détectés est important. Toutefois, ces points se concentrent majoritairement sur les lignes de l'image. On vérifie donc bien qu'une faible valeur de α , en accordant un faible poids au terme décorrélé de l'expression de la fonction de Harris Θ , empêche de distinguer un point situé sur un coin d'un point sur une ligne. A l'inverse, lorsqu'on prend des valeurs de α trop grandes, les points ne sont pas correctement détectés : les points retenus se concentrent en effet sur les bords de l'image et non sur les véritables points anguleux.

FIGURE 4 – Résultats du détecteur de Harris pour différentes valeurs de α

On peut réaliser le calcul sur plusieurs échelles en faisant varier l'écart-type σ des noyaux gaussiens dérivés utilisés pour calculer les dérivées premières de l'image¹. La Figure 5 présente les résultats du détecteur de Harris pour différentes valeurs d'écart-type, avec des valeurs fixées pour α et la taille de la fenêtre de sommation.

FIGURE 5 – Résultats du détecteur de Harris pour différentes valeurs de σ

Comme nous pouvions l'intuiter suite à notre conclusion concernant l'échelle du calcul des points d'intérêts, plus l'échelle à laquelle sont calculées les dérivées augmente, moins le nombre de points anguleux détectés est important. En effet, si les dérivées sont calculées à une échelle trop grande, l'information nous permettant de retrouver les points anguleux à une échelle plus fine va être déteriorée. De ce fait les valeurs

1. Dans le cadre du calcul de la matrice d'auto-corrélation de l'image

propres de la matrice d'auto-corrélation seront plus faibles, et nous risquons de détecter une ligne, ou une zone homogène, là où il avait un coin.

Q6 : Comme illustré sur la figure 6, les résultats obtenus pour les détecteurs ORB et KAZE sont très différents. Les points d'intérêt détectés par ORB présentent globalement des tailles importantes ainsi que des orientations variées et sont majoritairement situés au centre de l'image. A contrario, les points d'intérêts détectés par KAZE sont de petite taille, ne sont pas orientés (orientation nulle pour tous les points d'intérêt), et sont répartis de façon bien plus dispersée sur l'image.



FIGURE 6 – Comparaison des points d'intérêts détectés par les détecteurs ORB et KAZE

Le **détecteur ORB**² est une extension du détecteur FAST et une très bonne alternative³ opensource aux détecteurs SIFT et SURF. Le détecteur FAST sélectionne les pixels dont le voisinage circulaire (pixels formant un cercle autour de ce pixel) contient assez de pixels (généralement, plus de la moitié) significativement plus clairs, ou significativement plus foncés que le pixel central.

Dans le cas du détecteur ORB, le détecteur FAST est utilisé pour détecter des points d'intérêt sur une pyramide multi-échelle de l'image. Cette pyramide est une représentation multi-échelle de l'image, qui contient une liste de versions de cette image à différentes résolutions. En pratique, chaque niveau de la pyramide contient une version de l'image sous-échantillonnée par rapport à celle du niveau précédent. Chaque point d'intérêt a ainsi une échelle caractéristique, ce qui permet de rendre le détecteur ORB invariant vis-à-vis de l'échelle.

De plus, afin de rendre le détecteur ORB invariant en rotation, une orientation est assignée à chaque point d'intérêt détecté (Oriented FAST). Cette orientation est définie par le vecteur reliant le point d'intérêt au barycentre d'intensité d'un patch circulaire de rayon r , centré sur le point d'intérêt. Généralement, ce même patch circulaire est utilisé par la suite pour le calcul du descripteur (cf. Q7).

Les paramètres propres au détecteur ORB ayant une influence sur la détection des points d'intérêt sont donc les suivants :

- Le *voisinage* définissant la taille du cercle de pixels (voisinage circulaire) utilisé par la méthode de détection FAST. En pratique, le détecteur ORB utilise un cercle de diamètre de 9 pixels, qui contient donc 16 pixels. La taille de ce voisinage impacte directement l'échelle à laquelle sont détectés les points d'intérêts : plus le cercle de pixels est grand, plus un point d'intérêt local pourra être perdu au milieu de variations d'intensité globales, et inversement ;
- Le *seuil* sur la différence entre l'intensité du pixel central et l'intensité des pixels situés sur son voisinage circulaire. Ce seuil permet de définir une tolérance sur la détection des points d'intérêts : plus le seuil est faible, et plus facilement un pixel sera interprété comme un point d'intérêt, et inversement ;

2. Oriented FAST and Rotated BRIEF

3. En matière de temps de calcul et de performance

- La *taille du patch* **edgeThreshold** utilisé pour calculer l'orientation des points d'intérêt. La valeur de l'orientation d'un point d'intérêt est donc très influencée par la taille de ce patch. Par exemple, si la taille de patch choisie est trop faible, l'orientation du point d'intérêt peut ne pas être définie !
- Le *nombre de niveaux* **nlevels** de la pyramide multi-échelle de l'image sur laquelle sont détectés les points d'intérêt ;
- Le *facteur d'échelle* **scaleFactor** de la pyramide multi-échelle, qui définit la manière dont la résolution de l'image est réduite (resp. augmentée) entre chaque niveaux de la pyramide.

L'effet des deux derniers paramètres sur la détection des points d'intérêt est couplé. Un facteur d'échelle faible (i.e. proche de 1), nécessite l'ajout de niveaux de pyramide supplémentaires afin de parcourir une certaine gamme d'échelles sur l'image, et sera donc plus coûteux en temps de calcul. A l'inverse, un facteur d'échelle plus grand permettra de parcourir la même gamme d'échelle en moins de niveaux de pyramide, mais entraînera une dégradation importante dans la détection des points d'intérêt. Aussi, il est indispensable de trouver un bon compromis entre temps de calcul et précision de la détection, de même qu'entre le facteur d'échelle et le nombre de niveaux de la pyramide multi-échelle.

Le **détecteur KAZE** est basé sur des techniques de diffusion anisotropique, qui visent à réduire le bruit présent sur une image sans en dégrader les zones significatives, qui contiennent les détails nécessaires à son interprétation.

Le principe de la diffusion anisotropique consiste à créer une famille d'images paramétrisées, dans laquelle chaque image est le résultat de la combinaison de l'image originale et d'un filtre qui dépend du contenu local de l'image d'origine. La diffusion anisotropique est donc une transformation non-linéaire et dépendante de la position, d'où l'adjectif *anisotropique*. Analytiquement, l'effet des filtres utilisés peut être décrit comme une généralisation de l'équation de *diffusion* de la chaleur, où le coefficient de diffusion, au lieu d'être une constante, est une fonction (matricielle) de la position dans l'image.

Les images ainsi filtrées préservent leurs structures linéaires (bords, coins,...), alors que les zones autour de ces structures seront lissées. En pratique, la famille d'images générée par diffusion anisotropique contient des versions de l'image d'origine lissées à des échelles d'autant plus grandes que la génération est avancée : c'est cette propriété que va exploiter le détecteur KAZE.

Avant de procéder au calcul du détecteur, une approche similaire à celle du détecteur SIFT est adoptée : l'espace d'échelle de l'image est découpé de manière logarithmique en une série d'octaves et de sous-niveaux. Cette discréétisation permet d'associer à chaque octave et à chaque sous-niveau une certaine échelle $\sigma_{o,s}$.

En utilisant ces échelles, il est alors possible de construire une famille d'images par diffusion anisotrope, aussi appelée espace d'échelle non-linéaire de l'image, où chaque image sera lissée à une échelle σ_i , définie par l'espace d'échelle de l'image.

Pour détecter les points d'intérêt, l'idée est de calculer le déterminant normalisé du Hessien de l'image à chaque échelle de l'espace d'échelle non-linéaire de l'image. La normalisation permet de contrer la diminution de l'amplitude des dérivées spatiales lorsque l'échelle à laquelle elles sont calculées augmente.

Une fois le déterminant du Hessien calculé à chaque échelle de l'espace d'échelle non-linéaire de l'image, ses valeurs extrêmales sont recherchées à chaque échelle, dans des fenêtres de taille $\sigma_i x \sigma_i$. Ces valeurs extrêmales permettront de localiser (de manière sub-pixélique) les points d'intérêt à chaque échelle caractéristique de l'image. En conséquence le détecteur KAZE est invariant par rapport à l'échelle, mais également par rapport à la rotation, le déterminant du Hessien faisant partie des invariants différentiels de Hilbert.

Les différents paramètres relatifs au détecteur KAZE sont listés ci-dessous :

- Le *seuil threshold* sur la valeur du déterminant du Hessien de l'image à partir duquel un pixel est accepté comme zone d'intérêt (le point d'intérêt est calculé de manière sub-pixélique par la suite). A l'instar du seuil sur l'intensité des pixels du détecteur ORB, ce seuil va permettre de fixer une tolérance sur la détection des pixels d'intérêt : plus ce seuil sera faible, plus un pixel sera facilement indiqué en tant que pixel d'intérêt, et inversement ;
- Le **nombre d'octaves nOctaves** selon lequel l'espace d'échelle de l'image est discréétisé ;
- Le **nombre de sous-niveaux nOctaveLayers** selon lequel l'espace d'échelle de l'image est discréétisé ;
- Le **type de diffusivité diffusivity** utilisé lors de la création de l'espace d'échelle non-linéaire de l'image par diffusion anisotrope. Ce paramètre influe la manière dont les images sont lissées aux différentes échelles de l'espace d'échelle de l'image. Plus la diffusion est importante, plus le lissage de

l'image sera marqué entre deux échelles successives. Si cet effet de lissage est trop important, le nombre de points d'intérêt détectés pour les grandes valeurs d'échelle risque de diminuer, et inversement pour un effet de lissage trop faible.

L'effet du nombre d'octaves et du nombre de sous-niveaux n'est pas particulièrement explicité dans [?]. Toutefois, nous pouvons supposer qu'un nombre important d'octaves et de sous-niveaux va permettre la prise en compte d'un plus grand nombre d'échelles, et donc mener à une meilleure détection des points d'intérêts, au prix d'un coût en temps de calcul plus important. Il s'agit possiblement du même genre de compromis que celui mis en avant par le facteur d'échelle et le nombre de niveaux de la pyramide du détecteur ORB.

Pour évaluer visuellement la répétabilité de chaque détecteur appliquée sur une paire d'images, on peut réaliser deux séries de détections : la première sur les images d'origine, et la seconde sur ces mêmes images, mais orientées différemment. Si le détecteur est répétable, les points d'intérêt détectés sur les images d'origine doivent être identiques à ceux détectés sur les images orientées.

3 Descripteurs et Appariement

Q7 : Le descripteur attaché au point ORB, est une version légèrement modifiée du descripteur BRIEF, rBRIEF (Rotation-Aware BRIEF).

Le descripteur BRIEF est un descripteur binaire, qui convertit les points d'intérêts trouvés par l'algorithme Oriented FAST en vecteurs de caractéristiques binaires. Ces vecteurs de caractéristiques sont des descripteurs constitués d'un mot binaire comprenant un nombre de bits pouvant aller de 128 à 512. Le fonctionnement de ce descripteur est assez simple : pour chaque point d'intérêt, BRIEF sélectionne une paire de pixels au hasard, dans un voisinage défini autour du point d'intérêt. En pratique, le premier pixel est sélectionné selon une distribution gaussienne centrée autour du point d'intérêt, et d'écart-type σ . Le second pixel est ensuite sélectionné selon une distribution gaussienne centrée sur le premier pixel, et d'écart-type 2σ . Si le premier pixel a une valeur plus importante (respectivement, plus faible) que celle du second pixel, le premier bit du vecteur de caractéristiques est assigné à la valeur 1 (respectivement, 0). Cette opération est ensuite répétée jusqu'à ce que tous les bits du vecteur de caractéristiques soit rempli, c'est-à-dire entre 128 à 512 fois selon la taille du vecteur, puis réitérée sur chacun des points d'intérêt de l'image.

Le descripteur BRIEF n'est pas invariant par rotation, ce qui ne permet pas de tirer profit de l'orientation assignée à chaque point d'intérêt par le détecteur Oriented FAST. Afin de remédier à ce manque, le descripteur rBRIEF affine la zone dans laquelle les pixels seront sélectionnés afin de calculer le descripteur BRIEF. Pour ce faire, rBRIEF construit pour chaque point d'intérêt une matrice contenant les coordonnées des pixels utilisés pour le calcul du descripteur BRIEF. Connaissant l'orientation θ donnée par le détecteur au point d'intérêt, il est alors possible de calculer une version "orientée" de cette matrice, en la multipliant par la matrice de rotation associée à l'angle θ . Les pixels utilisés pour le calcul du descripteur BRIEF seront alors choisis parmi les pixels de cette matrice orientée.

Cette première modification ne permet pas d'assurer une invariance par rotation satisfaisante pour les descripteurs calculés. rBRIEF résout ce problème en recherchant grâce à un algorithme glouton les couples de pixels menant à des comparaisons non-correlées, de variances et de moyennes proches de 0.5.

Le **descripteur attaché au point KAZE** fait appel au descripteur SURF, qui présente l'avantage de pouvoir s'adapter à l'échelle du point d'intérêt à décrire. Cette adaptabilité lui confère de ce fait une invariance par rapport à l'échelle.

Pour chaque point d'intérêt détecté à l'échelle σ_i , les dérivées premières de l'image, L_x et L_y , sont calculées sur une grille de taille $24\sigma_i \times 24\sigma_i$ autour du point d'intérêt. Cette grille est ensuite divisée en 4×4 sous-régions, sur lesquelles les dérivées sont pondérées par une gaussienne centrée sur le centre de la sous-région. Ces dérivées pondérées sont ensuite combinées pour former un vecteur-descripteur de la forme :

$$d_v = \left(\sum_{sous-rgion} L_x, \sum_{sous-rgion} L_y, \sum_{sous-rgion} |L_x|, \sum_{sous-rgion} |L_y| \right) \quad (6)$$

Par la suite, les vecteurs-descripteurs de chaque sous-région sont moyennés par un noyau gaussien de taille 4×4 , centré sur le point d'intérêt considéré.

Afin de rendre ce descripteur invariant par rotation, une partie de son calcul est dédiée à l'estimation de l'orientation principale du voisinage local du point d'intérêt traité. S'inspirant une fois de plus du descripteur SURF, cette orientation est déterminée, toujours à l'échelle σ_i , par rapport à un voisinage circulaire de rayon $6\sigma_i$, discrétilisé selon un pas de σ_i . Pour chaque pas de discrétilisation, les dérivées premières L_x et L_y sont calculées, et pondérées par une gaussienne centrée sur le point d'intérêt. Ces valeurs sont ensuite considérées comme des points dans un repère $\{x, y\}$, permettant de calculer les vecteurs (L_x, L_y) . Afin d'obtenir l'orientation principale, ces vecteurs sont sommés selon un masque de sommation cône d'une ouverture de $\frac{\pi}{3}$: l'orientation du cône aboutissant au plus grand vecteur correspond à l'orientation principale associée au point d'intérêt.

Une fois l'orientation principale du point d'intérêt calculée, les vecteurs-descripteurs sont tournés selon cette orientation. Ces vecteurs orientés, et le calcul des dérivées premières de l'image selon l'orientation principale, nous permettent finalement d'obtenir un vecteur descripteur global contenant 64 valeurs.

Les facteurs permettant de rendre les détecteurs et les descripteurs associés à ORB et à KAZE sont résumés dans les tableaux suivants :

ORB		
	Détecteur : Oriented FAST	Descripteur : Rotation-Aware BRIEF
Invariance d'échelle	Détection multi-échelle : Pyramide multi-échelle de l'image	Pyramide multi-échelle de l'image
Invariance de rotation	Ajout d'une orientation au point d'intérêt	Prise en compte de l'orientation du point d'intérêt Recherche des couples de pixels optimaux

TABLE 1 – Résumé des facteurs d'invariance pour ORB

KAZE		
	Détecteur : KAZE	Descripteur : SURF
Invariance d'échelle	Détection multi-échelle : Espace d'échelle non-linéaire de l'image	Espace d'échelle non-linéaire de l'image Adaptation du descripteur à l'échelle du point d'intérêt
Invariance de rotation	Calcul du déterminant du Hessien (invariant différentiel de Hilbert)	Calcul de l'orientation principale du point d'intérêt et prise en compte dans le calcul du descripteur

TABLE 2 – Résumé des facteurs d'invariance pour KAZE

Q8 : Dans les scripts *Features_Match_CrossCheck.py* et *Features_Match_RatioTest.py*, la stratégie d'appariement repose sur une mise en correspondance de force brute (*brute-force matcher*). On considère un par un les descripteurs de la première image. Le descripteur considéré est comparé à tous les descripteurs de l'image cible. Dans chaque cas, la distance entre les deux points est calculée. Le point d'appariement dans l'image cible est le point dont le descripteur est le plus proche de celui de la première image, à condition que la distance entre ces deux points demeure inférieure à un seuil pré-défini. Il faut ainsi effectuer $n * m$ comparaisons entre descripteurs, n étant le nombre de descripteurs de la première image et m celui de la seconde.

La méthode *cross-check* consiste à calculer les correspondances dans les deux sens (image 1 vers 2 et image 2 vers 1), et de ne conserver les appariements que si les résultats de ces deux calculs coïncident. Cette méthode permet ainsi de réduire les erreurs d'appariement.

La méthode *ratio-test* consiste à ne conserver que les paires de points des deux images ayant les plus faibles distances entre eux. Dans le script utilisé ici, on élimine toutes les paires séparées par une distance supérieure à 70% de la plus grande distance entre deux points d'une paire formée avec la méthode de force brute.

Enfin, le script *Features_Match_FLANN.py* s'appuie sur la bibliothèque FLANN (*Fast Library for Approximate Nearest Neighbor*). FLANN implémente une recherche des plus proches voisins reposant sur les kd-trees et k-moyennes hiérarchique. Le script *Features_Match_FLANN.py* utilise la méthode des kd-trees de la bibliothèque FLANN. Un kd-tree se construit à partir d'un ensemble de données en partitionnant cet ensemble en deux sous-ensembles à chaque noeud, en alternant les directions de coupe. Le choix de la valeur de coupe peut être lié à la répartition statistique des données, ou être aléatoire, comme dans le cas des kd-trees aléatoires, qui sont utilisés par FLANN. La recherche du plus proche voisin d'un point dans le kd-tree se fait dans un premier temps en descendant l'arbre jusqu'à trouver le plus proche voisin du point considéré dans la branche. On considère dans un second temps les feuilles intersectant la sphère du plus proche voisin calculé : on remonte alors ces feuilles pour aboutir au plus proche voisin final. Le problème de cette méthode de recherche est qu'on parcourt tout l'arbre, ce qui est coûteux en temps lorsqu'on travaille avec beaucoup de données. Dans le cas de FLANN, on réalise une recherche approximée : *Best Bin First Search*. On examine ainsi en premier les feuilles les plus proches du point, ce qui permet de réduire sensiblement le nombre de noeuds à explorer. Cette méthode est donc particulièrement intéressante lorsque la quantité de données à traiter est importante.

La Figure 7 montre les appariements résultants des 3 méthodes d'appariement testées sur les points ORB, et Figure 8 ceux des 3 mêmes méthodes pour les points ORB. Les résultats sont détaillés dans les Figure 9 pour ORB et 10 pour KAZE.

On constate que le cross-check est à première vue plus sensible à l'erreur que le ratio-test et FLANN. On peut en effet voir sur la Figure 7a que certains appariement sont faux avec cette méthode dans le cas d'ORB. On remarque également que ratio-test et FLANN ont validé très peu de points dans le cas d'ORB, problème qui ne se pose pas avec KAZE. Dans le cas du cross-check, on a demandé la génération du plus grand nombre possible de *features* inférieur ou égal à 500, donc cette case des tableaux en Figures 9 et 10 ne sert qu'à avoir un ordre de grandeur du nombre d'appariements maximal. Enfin, le temps de calcul du cross-check est environ le double de celui du ratio-test, ce qui est normal puisqu'on réalise deux fois le calcul des correspondances en force brute dans le cas du cross-check. Le temps de calcul avec FLANN est beaucoup plus élevé, ce qui est possiblement dû au fait que notre jeu de données soit de trop petite taille pour que FLANN devienne véritablement intéressant.

Dans le cas d'ORB, on a utilisé la distance de Hamming, et dans celui de KAZE, la distance L2. ORB repose en effet sur une variante des descripteurs BRIF, qui consistent, comme on l'a rappelé plus tôt, en un vecteur de caractéristiques binaires. La distance de Hamming semble alors un choix logique. Cette distance n'est en revanche pas du tout pertinente dans le cas des descripteurs associés à KAZE, qui s'appuient sur les dimensions de l'image - d'où la pertinence du choix de la distance L2.

ORB repose sur des descripteurs qui consistent en des vecteurs de valeurs binaires, ce qui est mal adapté aux kd-trees. Il est donc normal que les résultats de la stratégie FLANN ne soient pas bons. Dans le cas du ratio-test, on peut penser que le fait que les points utilisés pour la génération du descripteur rBRIF soient choisis de manière aléatoire peut augmenter la distance d'un point à son correspondant d'une image à l'autre, ou la réduire aléatoirement de manière importante. Ainsi, un certain nombre d'appariement peuvent être éliminés par ratio-test alors même qu'ils sont bons.

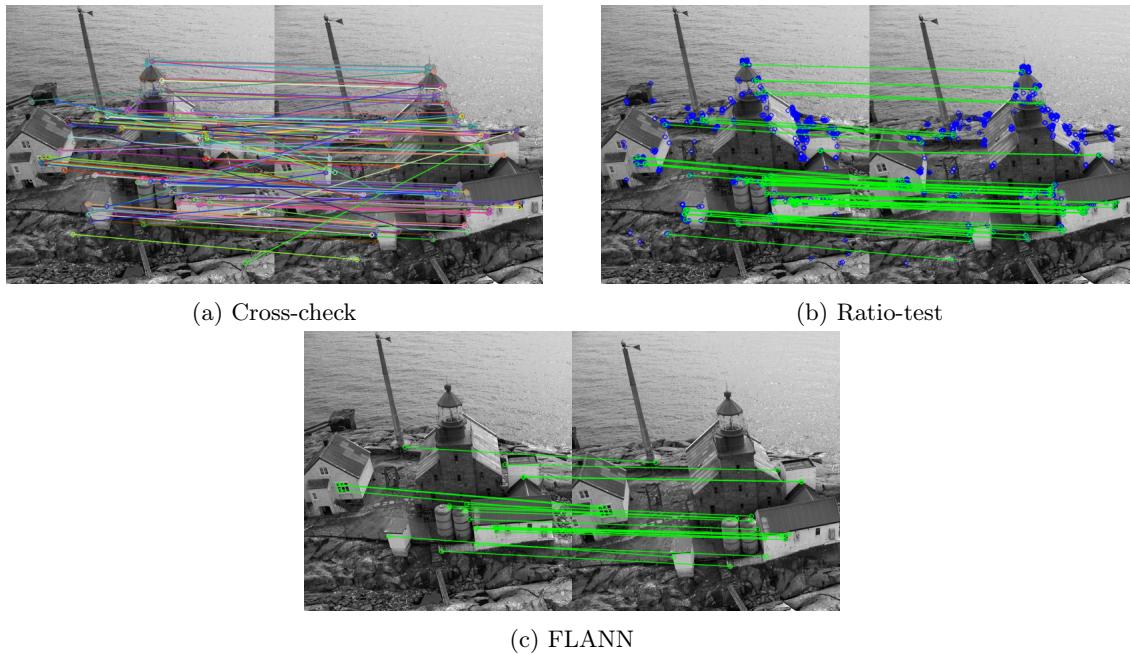


FIGURE 7 – Résultats des 3 stratégies d'appariement testées pour les points ORB

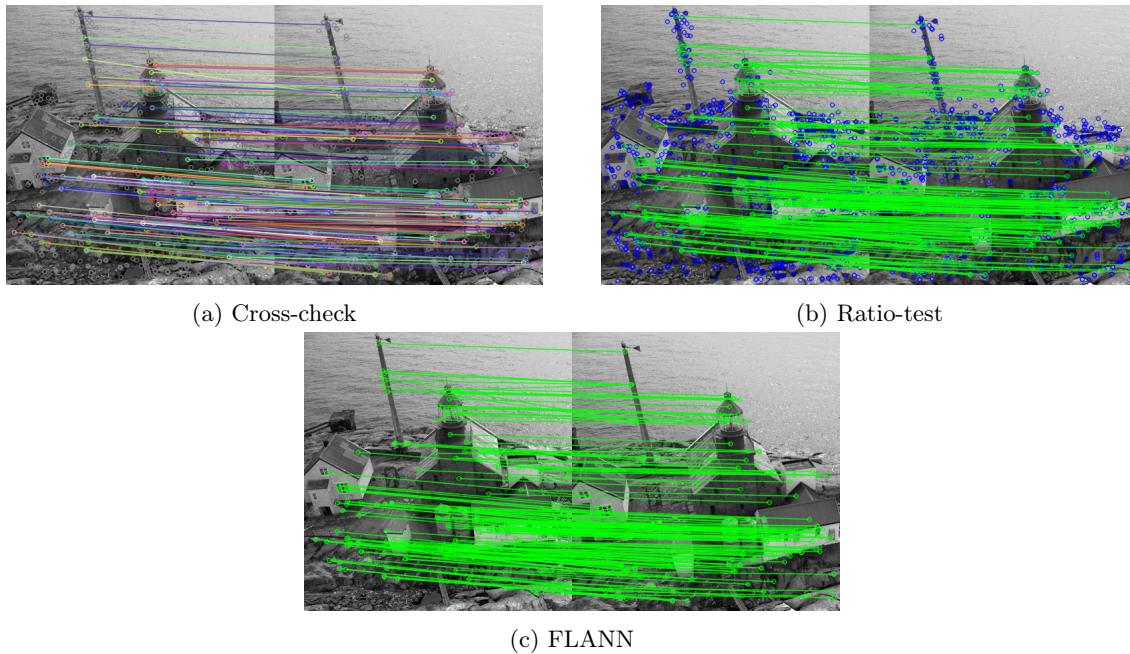


FIGURE 8 – Résultats des 3 stratégies d'appariement testées pour les points KAZE

Méthode	Erreurs évidentes ?	Appariements validés	Temps de calcul
Cross-check	Oui	< 200	1,7 ms
Ratio-test	Non	64	1,1 ms
FLANN	Non	22	12 ms

FIGURE 9 – Résultat des appariements des descripteurs ORB selon la stratégie d'appariement.

Méthode	Erreurs évidentes ?	Appariements validés	Temps de calcul
Cross-check	Non	> 200	4,5 ms
Ratio-test	Non	175	2,6 ms
FLANN	Non	175	20 ms

FIGURE 10 – Résultat des appariements des descripteurs KAZE selon la stratégie d'appariement.

Q9 : Afin d'évaluer quantitativement la qualité des différents appariements implémentés, nous avons mis en place la stratégie suivante :

- Création d'une image étalon à partir d'une transformation géométrique connue.** Pour ce faire, nous avons recours à la fonction `getRotationMatrix2D` d'OpenCV qui calcule la matrice de transformation représentant une rotation et un grossissement l'image. Cette transformation est donc parfaitement connue et entièrement définie par l'utilisateur (angle et centre de rotation, facteur de grossissement). La fonction `warpAffine` d'OpenCV nous permet par la suite d'appliquer cette transformation à l'image d'origine afin d'obtenir une image étalon, dont nous sommes capable de déterminer numériquement la position de chaque pixel.
- Calcul des points d'intérêt sur l'image étalon et appariement par rapport à l'image d'origine.** L'image étalon passe subit ensuite le même traitement que l'image d'origine : détection des points d'intérêt et calcul des descripteurs. A partir de là, la méthode d'appariement à tester est mise en œuvre pour apparier les points d'intérêt de l'image étalon à ceux de l'image d'origine.
- Calcul des positions théoriques des points d'intérêt sur l'image étalon.** L'idée clé de cette évaluation quantitative est de comparer les correspondances des points d'intérêts sur les deux images, calculé lors de l'appariement, aux correspondances théorique, que nous connaissons grâce à la transformation géométrique utilisée à l'étape 1. Avant de procéder à cette comparaison, il nous faut donc d'abord calculer la position théorique des points d'intérêt de l'image d'origine dans l'image étalon.
- Comparaison des deux listes de points et calcul d'erreurs.** Une fois que nous avons à disposition les deux correspondances, théorique et appariée, sous la forme d'une liste de points situés sur l'image étalon, il ne reste plus qu'à les comparer. Dans cette optique, nous avons décidé de calculer pour chaque point apparié la distance le séparant du point théorique correspondant, équivalent à une erreur d'appariement. A partir de ces erreurs, nous pouvons ensuite calculer l'erreur moyenne sur l'ensemble des points appariés, une première métrique permettant d'évaluer la qualité de l'appariement. Nous avons également décidé de tracer l'évolution du nombre d'appariements corrects par rapport à un certain seuil d'échelle que nous faisons varier de 10^{-3} pixels à 10 pixels. Cette seconde métrique nous permet d'évaluer la qualité de l'appariement en fonction de l'échelle de l'image la plus importante. Finalement, nous avons utilisé les points appariés sur l'image d'origine et l'image étalon afin de calculer une approximation⁴ de la matrice de transformation entre les deux images. Cette matrice approximée peut être ensuite facilement comparée à la matrice théorique calculée à l'étape 1, et l'erreur obtenue nous fournit une troisième métrique d'évaluation de la qualité de l'appariement.

Remarque : Tous les résultats exposés dans ce compte-rendu peuvent être retrouvés (ou testés pour la méthode d'évaluation quantitative des appariements) grâce aux scripts python disponibles sur le GitHub : <https://github.com/CarolinePascal/ROB317-TP1>

Références

- [1] Pablo Fernández Alcantarilla, Adrien Bartoli, and Andrew Davison. Kaze features. 10 2012.
- [2] C. Harris and M. Stephens. A Combined Corner and Edge Detector. In *Proceedings of the Alvey Vision Conference 1988*, pages 23.1–23.6, Manchester, 1988. Alvey Vision Club.
- [3] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. ORB : An efficient alternative to SIFT or SURF. In *2011 International Conference on Computer Vision*, pages 2564–2571, Barcelona, Spain, November 2011. IEEE.

4. En pratique, nous avons eu recours à la méthode des moindres carrés

- [4] Javier Sánchez, Nelson Monzón, and Agustín Salgado. An Analysis and Implementation of the Harris Corner Detector. *Image Processing On Line*, 8 :305–328, October 2018.