Option Vidéo ET5 ESR

Polytech'Paris Saclay
Département Électronique et Systèmes Robotisés
Michèle Gouiffès
michele.gouiffes@universite-paris-saclay.fr

Les objectifs de l'option vidéo sont :

— d'acquérir des connaissances en traitement de vidéos et d'images multi-vues : soustraction de fond, flot optique, mise en correspondance de primitives visuelles pour le suivi et la stéréovision.

— de développer des algorithmes et de s'initier à l'utilisation de la librairie OpenCV, en se référant à la documentation [1] .

Le projet qui vous est fourni utilise `opencv4.3.0`. Il est constitué de 5 projets à ouvrir de préférence avec `Visual Studio 2017`. Les paramètres utiles, tels que les chemins d'images, les paramètres de certaines méthodes, bref les paramètres que vous êtes susceptibles de modifier, sont accessibles par fichier de configuration : `config_video0i.cfg`.

1. https ://docs.opencv.org/4.3.0/

Chapitre 1

Détection de mouvement

La première séance se focalise sur les méthodes de détection de mouvement. Typiquement, dans le cas où la caméra est fixe, cela consiste à construire une **image de référence** associée au *fond* sans mouvement (*Background*) et à la soustraire de l'image courante.

Il existe un grand nombre de méthodes qui diffèrent par les stratégies employées pour : 1) la construction de l'image de fond 2) le calcul de distance de l'image courante par rapport au fond permettant d'obtenir l'image des pixels en mouvement, appelée également image d'avant-plan (*Foreground*) et 3) éventuellement les primitives visuelles utilisées : même si ce sont généralement les valeurs d'intensité des pixels qui sont généralement utilisées, certains traitements se basent sur l'extraction des contours pour détecter le mouvement.

D'autre part, les méthodes s'avèrent plus ou moins robustes face aux différentes perturbations susceptibles de se produire dans les vidéos : changements d'illumination plus ou moins brutaux, bruit d'acquisition.

Dans la première partie du TP, nous analyserons un algorithme fourni dans la librarie *openCV*. Ensuite, nous mettrons en place notre propre méthode et ses améliorations, en complétant les fonctions du fichier `Motion.cpp`.

1.1 Étude de l'algorithme de soustraction de fond "MOG"

Questions

1) Exécutez le code fourni. Il affiche la vidéo initiale (précisée dans le fichier de configuration) et le résultat de la méthode MOG [1]. Les autres fenêtres n'affichent pas grand chose d'intéressant mais on va y remédier par la suite.

a- Que signifie MOG ?

b- Dans cette méthode, que modélise la Gaussienne ?

c- Cette méthode dépend-elle de paramètres ? Si oui quels sont-ils ?

d- Requière t'elle une image de référence (le fond).

2) Quels problèmes observez-vous et quelles sont leurs origines ?

L'algorithme MOG va nous servir de méthode de référence à laquelle vous comparerez vos propres algorithmes.

1.2 Détection par soustraction d'images

Ces approches supposent qu'une image du fond (que l'on appellera plus tard *background* $B(x)$) est disponible, c-à-d une image de la scène dans laquelle aucun objet ne bouge. Dans un premier temps, nous supposons que l'image de référence est la première image de la vidéo, notée I_0. Appelons I_t l'image courante (à l'instant t). Il est possible de détecter les pixels en mouvement (*Foreground*) $F(x)$ en tout point x de l'image à l'aide de l'équation suivante :

$$F(x) = \begin{cases} 1 & \text{si } |I_0(x) - I_t(x)| > Th \\ 0 & \text{sinon} \end{cases}$$

où Th est un seuil à définir.

Questions

1) Mettre en place cette approche en complétant la fonction `SimpleBackgroundSubtract` dans Motion.cpp.

2) Exécuter la méthode sur la séquence Hall et ajuster le paramètre Th (`motion_th` dans le fichier `config_video01.cfg`) de manière à obtenir un résultat de qualité comparable à celui obtenu par MOG.

3) Quels défauts peut-on identifier ?

4) Quels sont les avantages de MOG par rapport à la méthode mise en œuvre ?

1.3 Filtrage de l'image résultat

Ici, la fonction `SimpleBackgroundSubtract` doit être complétée pour y ajouter l'opération de morphologie mathématique de votre choix.

Questions

1) Choisir l'opération ou les opérations de morphologie mathématique le(s) plus appropriée(s) et la(les) mettre en œuvre en choisissant les paramètres adéquats.Vous pouvez utiliser les méthodes disponibles dans opencv : `morphologyEx` .

2) Comparer les résultats obtenus avec ceux qui ont été obtenus sans opérateur morphologique.

3) Comparer avec MOG.

1.4 Compensation des variations d'illumination

Quelques images de la séquence montrent des changements d'illumination importants (Walk1012.jpg et Walk1051.jpg). Pour remédier à ce problème, nous pouvons appliquer une normalisation de la luminance de la manière suivante :

$$J_t(x) = \frac{I_t(x) - \mu_t}{\sigma_t} \tag{1.1}$$

où $J_t(x)$ est l'image obtenue après normalisation, μ_t et σ_t sont respectivement la moyenne et l'écart-type d'intensité dans l'image I_t. Notons qu'ici, les images résultat ne sont pas filtrées par morphologie mathématique.

Questions

1) La normalisation a un impact sur l'histogramme. Quelles sont les transformations subies par l'histogramme ?

2) Illustrer par un exemple d'histogramme.

3) Quelles sont les caractéristiques du nouvel histogramme ?

4) Quels types de variations d'illumination la transformation (1.1) permet-elle de compenser ? Plus précisément, quelle est la forme de la transformation f telle que $J_t(x) = f(I_t(x))$ qui est compensée par (1.1).

5) Mettre en place cette normalisation (méthode `PhotometricNormalization` dans la classe `Color`).

6) Tester cette amélioration. Comparer les résultats obtenus avec ceux d'une égalisation d'histogramme. D'après vous, les histogrammes (après normalisation vs après égalisation) sont-ils identiques ? (il n'est pas demandé de les afficher).

7) Dans le cas où la vidéo montre des changements non-uniformes d'éclairage, la correction apportée par (1.1) peut s'avérer insuffisante. Proposer une solution (l'implémentation n'est pas demandée).

1.5 Mise à jour de l'image de référence

Précédemment, l'image de référence était tout simplement la première image de la vidéo I_0. Cette approche n'est pas viable dans la plupart des applications (vidéosurveillance par exemple). En effet, au cours du temps, à l'échelle d'une journée ou d'une année, les conditions d'acquisition et la scène peuvent évoluer. Il est alors nécessaire de mettre à jour continuellement l'image de référence. Nous allons nous intéresser à deux approches différentes.

A . La première méthode consiste à prendre comme image de référence courante $B_t(x)$ la moyenne de plusieurs images précédentes.

Questions

1) Écrire la formule de $B_t(x)$ qui correspond à la moyenne de N images.

2) Comment calculer l'image $B_t(x)$ de manière itérative à partir de $B_{t-1}(x)$?

3) Complétez la fonction `BackgroundUpdateMean` pour mettre en place cette première méthode (une ligne de code suffit).

4) Évaluez l'influence du ou des paramètres sur le résultat de la détection de mouvement.

B. La seconde approche consiste à calculer l'image médiane de quelques images précédentes.

Ici, nous considérons 3 images à trois instants différents (notées I_1, I_2, I_3 dans le code).

La première image de référence est calculée à partir de la troisième image acquise et utilisée pour détecter les pixels en mouvement à partir de la quatrième image. Elle sera ensuite mise à jour continuellement :

$$B_3(x) = median(I_1(x), I_2(x), I_3(x)) \text{ ou plus généralement :} \quad (1.2)$$

$$B_t(x) = median(I_{t-2}(x), I_{t-1}(x), I_t(x)) \forall t \quad (1.3)$$

Les images utilisées dans le calcul doivent être mises à jour à chaque instant, par exemple de la manière suivante :

$$I_1 = I_2;\ I_2 = I_3;\ I_3 = I_4(x) \text{ ou plus généralement :} \quad (1.4)$$

$$I_1 = I_2;\ I_2 = I_3;\ I_3 = I_t(x) \quad (1.5)$$

Questions

1) Compléter la fonction `BackgroundUpdateMedian` et tester.

2) Pour améliorer les résultats, nous pouvons ré-introduire l'image de fond précédente dans le calcul de l'image de fond actuelle, ce qui modifie (1.5) de la façon suivante :

$$I_1 = I_2;\ I_2 = I_3;\ I_3 = B_t(x)$$

3) Comparez les deux méthodes de mise à jour (A et B) de l'image de référence (qualité des résultats, ressources mémoire, temps de calcul).

Chapitre 2

Le flot optique

Le flot optique correspond au champ des vecteurs de vitesse apparente des pixels dans l'image. Ce mouvement est dit apparent car il ne correspond par forcément à un mouvement réel (dans la scène 3D) projeté sur le plan 2D du capteur. L'exemple le plus illustratif est sans doute celui d'une sphère (ou d'un cylindre) de couleur parfaitement uniforme qui tourne autour de son axe de symétrie. Le mouvement devient apparent dès que l'objet est texturé. Ainsi, la perception du mouvement dépend de la distribution spatiale des intensités lumineuses dans l'image.

Pour calculer le flot optique, il est nécessaire de poser des hypothèses quant au modèle de variation photométrique. Le plus simple est de considérer que l'intensité des points en mouvement est conservée au cours du temps et que les intensités de deux images successives d'une séquence sont reliées entre elles par l'expression suivante :

$$I_t(x, y) = I_{t-1}(x + u, y + v) \tag{2.1}$$

où I_t et I_{t-1} sont les images aux instants courant et précédent, u et v sont les coordonnées de translation du pixel et $\mathbf{v} = (u, v)$ est le vecteur vitesse. L'équation (2.1) est appelée **contrainte de conservation de la luminance**. Calculer le flot optique revient à calculer ce vecteur de déplacement en tout point [1].

En faisant l'hypothèse de **petits déplacements inter-image** et de différenciabilité spatio-temporelle de l'intensité lumineuse (c-a-d que les dérivées existent), la contrainte de conservation de l'intensité devient :

$$I_x u + I_y v + I_t = \nabla I.\mathbf{v} + I_t = 0 \tag{2.2}$$

Cette équation est une approximation, obtenue après développement limité de (2.1). Ici, I_x et I_y se réfèrent aux composantes du gradient spatial $\nabla I = (I_x, I_y)$ et I_t est le gradient temporel.

L'équation (2.2) permet de calculer la projection du vecteur vitesse dans la direction du gradient spatial (perpendiculaire aux contours). Ainsi, si le déplacement de l'objet est tangent à son contour, le mouvement n'est pas perceptible.

1. C'est un problème sous-estimé, car il faut déterminer 2 paramètres alors que nous n'avons que des informations de luminance.

2.1 Algorithme de Horn et Schunck [2]

Le champ de vitesse peut être calculé en combinant la contrainte de conservation de la luminance avec une régularisation globale dans l'image. $\mathbf{v}(\mathbf{p}) = (u, v)(\mathbf{p})$. Il s'agit alors de minimiser la fonction suivante :

$$\int_{(\mathbf{p})\in I} (\nabla I(\mathbf{p})\mathbf{v}(\mathbf{p}) + I_t(\mathbf{p}))^2 + \alpha^2(\|\nabla u(\mathbf{p})\| + \|\nabla v(\mathbf{p})\|)^2 d\mathbf{p}$$

α est un coefficient fixé par l'utilisateur pour donner plus ou moins de poids à la régularisation. Le terme de regularisation force la vitesse à être lisse dans l'image (force les dérivées de vitesse $\|\nabla u\|$ et $\|\nabla v\|$ à être faibles dans l'image). On minimise cette fonction de manière itérative à l'aide des équations suivantes :

$$u^{k+1} = \bar{u}^k - \frac{I_x(I_x\bar{u}^k + I_y\bar{v}^k + I_t)}{\alpha^2 + I_x^2 + I_y^2} \tag{2.3}$$

$$v^{k+1} = \bar{v}^k - \frac{I_y(I_x\bar{u}^k + I_y\bar{v}^k + I_t)}{\alpha^2 + I_x^2 + I_y^2} \tag{2.4}$$

où k sont les itérations, $\bar{u}$ et $\bar{v}$ sont les cartes de vitesses en $\mathbf{p}$ lissées par un filtre passe-bas pour réduire l'importance des artéfacts. Il est nécessaire de fixer une condition d'arrêt qui peut être soit un nombre d'itérations fixe, ou lorsque le vecteur de mouvement $\mathbf{v}$ n'évolue plus de manière significative d'une itération k à l'autre $k+1$.

2.2 Algorithme de Lucas-Kanade [3]

?? Lucas et Kanade ont proposé une méthode qui, au lieu de calculer un vecteur vitesse indépendant en chaque point, considère un voisinage $\mathcal{V}$. Il s'agit alors de minimiser la fonction suivante :

$$\sum_{\mathbf{p}\in\mathcal{V}} \mathcal{W}(\mathbf{p})[\nabla I(\mathbf{p},t)\mathbf{v} + I_t(\mathbf{p},t)]^2 \tag{2.5}$$

où $\mathcal{V}$ est un voisinage spatial de l'image contenant N pixels, $\mathcal{W}$ représente une fonction de fenêtrage permettant de donner plus d'influence aux pixels du centre plutôt qu'à sa périphérie. Typiquement, il s'agit un noyau Gaussien.

La solution de (2.5) est celle qui permet d'annuler la dérivée par rapport à $\mathbf{v}$. Cela se résume à calculer le produit matriciel suivant :

$$\mathbf{v} = \begin{bmatrix} \sum_{\mathbf{p}\in\mathcal{V}} W(\mathbf{p})I_x^2(\mathbf{p}) & \sum_{\mathbf{p}\in\mathcal{V}} W(\mathbf{p})I_x(\mathbf{p})I_y(\mathbf{p}) \\ \sum_{\mathbf{p}\in\mathcal{V}} W(\mathbf{p})I_x(\mathbf{p})I_y(\mathbf{p}) & \sum_{\mathbf{p}\in\mathcal{V}} W(\mathbf{p})I_y(\mathbf{p})I_y(\mathbf{p}) \end{bmatrix}^{-1} \begin{bmatrix} -\sum_{\mathbf{p}\in\mathcal{V}} W(\mathbf{p})I_x(\mathbf{p})I_t(\mathbf{p}) \\ -\sum_{\mathbf{p}\in\mathcal{V}} W(\mathbf{p})I_y(\mathbf{p})I_t(\mathbf{p}) \end{bmatrix} \tag{2.6}$$

La matrice 2×2 est appelée *tenseur de structure*. Il peut arriver qu'elle ne soit pas inversible ou mal conditionnée, ce qui rend le calcul du mouvement très sensible au bruit, instable. Pour

améliorer les résultats, il est possible de détecter au préalable les points pour lesquels la matrice est inversible. Il s'agit du détecteur de coins de Shi et Tomasi [4], disponible dans OpenCV sous le nom `goodFeaturesToTrack` [2]. On y reviendra plus tard dans la partie .

2.3 Travail à réaliser

Nous allons implémenter l'algorithme de Horn et Schunck. Ensuite, nous segmenterons les cartes de mouvement par un algorithme de croissance de régions.

2.3.1 Mise en œuvre de l'algorithme de Horn et Schunck

Objectifs : compléter la fonction `gradients_HS` (dans le fichier `ImgProcTools.cpp`) et la méthode `HornSchunck` (dans la classe `OptFlow.cpp`) pour calculer le flot optique.

2.3.2 Calcul des gradients

Pour mettre en œuvre l'algorithme de Horn et Schunck, il faut tout d'abord calculer les gradient spatiaux I_x et I_y et le gradient temporel I_t. Cela peut se faire comme sur l'exemple de la figure 2.1. On considère quatre points voisins dans l'image, disposés de la manière suivante :

$$\begin{array}{rclrcl} \mathbf{p}_{00} & = & (x,y) & \mathbf{p}_{01} & = & (x,y+1) \\ \mathbf{p}_{10} & = & (x+1,y) & \mathbf{p}_{11} & = & (x+1,y+1) \end{array}$$

Les composantes du gradient de l'image à l'instant t s'écrivent :

$$I_x^t = \frac{1}{2}(I(\mathbf{p}_{01}^t) - I(\mathbf{p}_{00}^t) + I(\mathbf{p}_{11}^t) - I(\mathbf{p}_{10}^t)) \tag{2.7}$$

$$I_y^t = \frac{1}{2}(I(\mathbf{p}_{10}^t) - I(\mathbf{p}_{00}^t) + I(\mathbf{p}_{11}^t) - I(\mathbf{p}_{01}^t)) \tag{2.8}$$

Le gradient à utiliser pour le calcul du flot optique est la moyenne des gradients des deux images successives, à t et $t+1$:

$$I_x = \frac{1}{2}(I_x^{t+1} + I_x^t) \tag{2.9}$$

$$I_y = \frac{1}{2}(I_y^{t+1} + I_y^t) \tag{2.10}$$

Le gradient temporel s'écrit comme la moyenne des quatre différences inter-image :

$$I_t = \frac{1}{4}\sum_{i=0}^{1}\sum_{j=0}^{1}\left(I(\mathbf{p}_{ij}^{t+1}) - I(\mathbf{p}_{ij}^t)\right)$$

2. http ://docs.opencv.org/modules/imgproc/doc/feature_detection.html#goodfeaturestotrack

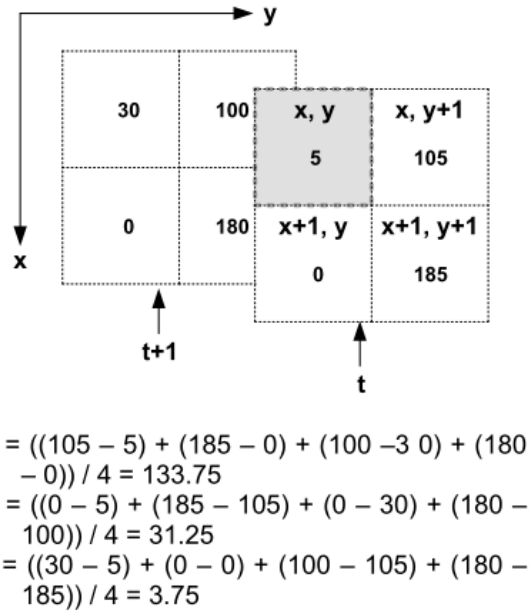


FIGURE 2.1 – Illustration du calcul du gradient spatial.

2.3.3 Minimisation

Pour k allant de 0 à `hs_itermax` (nombre d'itération maximal) et pour chaque point de l'image $\mathbf{p}$:

- on calcule en chaque point $\mathbf{p}$: u^{k+1} en fonction de u^k et v^{k+1} en fonction de v^k comme dans les équations (2.3) et (2.4) ;
- on convolue u^{k+1} et v^{k+1} par un noyau gaussien (filtre passe-bas).
- on met à jour la vitesse pour préparer l'itération suivante $u^k = u^{k+1}$ et $v^k = v^{k+1}$

2.3.4 Tests

Pour effectuer les tests, vous disposez de plusieurs séquences dans le répertoire `Data`. Les paramètres à utiliser pour chaque séquence sont donnés dans le fichier `Sequences.txt`.

1) Analyser l'impact du nombre d'itérations maximal `hs_itermax` (tester 1, 5, 10, 20, 100) pour `hs_alpha`=10.

2) Analyser l'impact du paramètre de régularisation `hs_alpha` (tester 1, 5,10, 100) pour `hs_itermax`=20.

Dans la suite, on prendra `hs_alpha`=10 et `hs_itermax`=20.

2.3.5 Segmentation du mouvement

Dans `video02.cpp`, vous décommenterez les 3 lignes de codes suivantes, qui permettront de segmenter la carte de mouvement, une fois que vous aurez choisi les paramètres adéquats.

```
of.MotionNormAngle(Vx,Vy, Vn, Va);
of.MotionSegment(Vn, Va, R);
```

```
imshow("HS Motion segmentation", R);
```

La méthode `MotionSegment` segmente l'image de mouvement par une méthode de croissance de régions. Les valeurs des étiquettes sont choisies aléatoirement [3]. Elle prend deux paramètres (seuils) à définir dans le fichier de configuration :

— `simi_angle_th` : seuil sur le critère d'homogénité calculé sur la norme du gradient

— `simi_norm_th` : seuil sur le critère d'homogénité calculé sur l'orientation du gradient.

1) Compléter la méthode `Similarity` dans `Motion.cpp`, qui calcule le critère de similarité entre un pixel et son voisin.

2) Tester l'algorithme sur les différentes séquences et choisir les paramètres.

2.3.6 Analyse du mouvement

Passons au projet `video03`. L'analyse du flot optique permet de réaliser des applications plus évoluées comme la reconnaissance d'activité. Dans cet exercice, nous disposons de vidéos représentant 6 activités simples : walking, running, jogging, boxing, handclapping, handwaving.

Pour chaque action, une vidéo va servir d'exemple (de référence). Pour chacune d'entre elles, un histogramme du flot optique (à 2 dimensions V_x, V_y) est calculé, en cumulant les occurrences sur l'ensemble de la vidéo.

Les autres vidéos montrent chacune une activité à identifier parmi les 6 possibles. Il s'agira d'identifier l'action représentée en comparant son histogramme avec les histogrammes des 6 actions de référence. L'action sera celle pour laquelle l'histogramme est le plus proche.

1) Compléter la distance inter-histogramme `distances[i]`. On implémentera la distance : somme des valeurs absolues des différences.

2) Visualiser les performances obtenues (matrice de confusion et le taux de reconnaissance). Le taux de reconnaissance correspond au nombre de bonnes reconnaissances sur le nombre total de tests. La matrice de confusion donne pour chaque classe d'action, quel est le type d'action reconnu.

3) Nous allons remplacer l'histogramme 2D par un histogramme pondéré, que vous compléterez dans `Motion.cpp` :

```
void Motion::histo_Angular( const Mat& N, const Mat& A, const Mat& Mask, Mat &H2D
, int &n, bool display)
```

Il prend en entrée la norme N et l'orientation A du mouvement. Pour chaque angle, on accumule les normes associés sur toute la vidéo. Ainsi cet histogramme nous indique les angles pour lesquels le mouvement est de plus forte amplitude.

4) Comparer les deux types d'histogrammes en termes de résultats.

5) La taille de l'histogramme a t'elle une influence sur les résultats ?

3. ce qui explique le côté psychédélique de l'affichage !

Chapitre 3

Mises en correspondance spatiales et temporelles

3.1 Introduction

Considérons un objet (ou des primitives visuelles telles que des points d'intérêt) détecté à l'instant t. Le suivi (ou **appariement temporel**) de cet objet consiste à retrouver sa position dans l'image à l'instant $t+1$. L'appariement temporel est facilité par :

1. le fait que la mise en correspondance se faisait entre des images acquises par la même caméra
2. le fait que le déplacement des objets est supposé faible entre deux images successives

Dans le cas de la **mise en correspondance spatiale**, ces hypothèses ne sont plus valides. Il s'agit de mettre en correspondance des images issues de capteurs différents, ou bien du même capteur mais à des instants très différents. Les distorsions sont alors plus importantes et les conditions d'acquisition peuvent avoir énormément changé. On ne peut donc plus s'aider de la cohérence temporelle. Les applications concernées sont la reconnaissance d'objets dans une base d'images, le calcul de déformations géométriques des objets, le calcul de cartes de profondeur par stéréovision. Par conséquent, ils est nécessaire de choisir des descripteurs suffisamment discriminants et invariants vis-à-vis des forts changements de conditions d'acquisition, même si leur calcul s'avère plus lourd.

Généralement l'objet est représenté par un ou plusieurs **descripteurs** ou encore par ses **caractéristiques cinématiques** (position, vitesse, accélération) dans le cas du suivi. Trouver la nouvelle position de l'objet revient à trouver la position permettant de maximiser une **mesure de similarité** (ou minimiser une distance) entre le descripteur calculé en cette position et le descripteur initial calculé précédemment. On parle alors d'**appariement** ou de **mise en correspondance**.

Il existe un grand nombre de détecteurs et de descripteurs. Leur choix dépend du type d'applications et du type d'objets à suivre. Le descripteur idéal possède les qualités suivantes, il doit être :

- invariant par rapport aux déformations géométriques de l'objet : changement d'échelle, mouvements non-rigides ;
- invariant par rapport aux conditions d'acquisition : bruit, changement d'éclairage ;

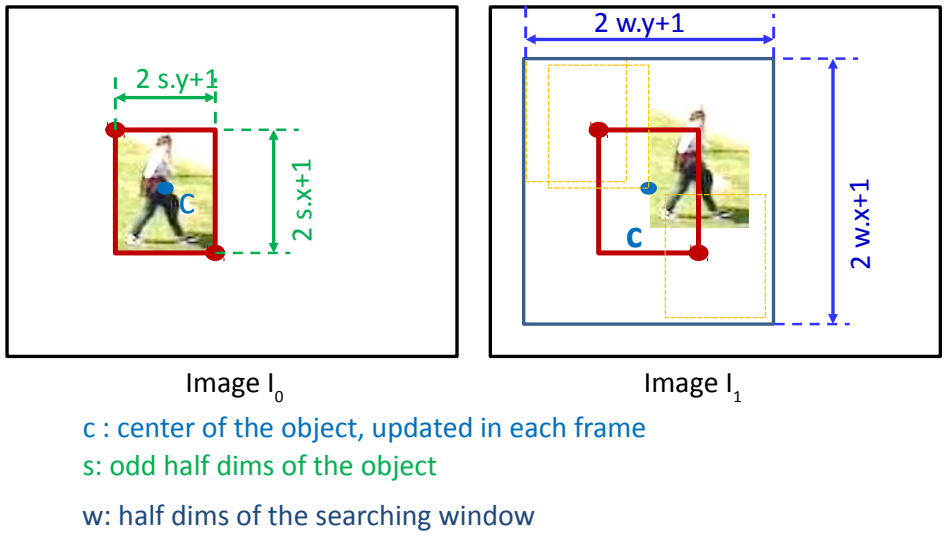


FIGURE 3.1 – Méthode de mise en correspondance par bloc matching

- robuste vis-à-vis des occultations partielles ou totales (disparition de l'objet) ;
- discriminant, c'est-à-dire que le descripteur doit être unique pour l'objet considéré. Dit autrement, il ne doit pas y avoir d'ambigüité d'appariement.
- peu gourmand en ressources mémoire et en temps de calcul, c'est-à-dire qu'il doit être compact et rapide à calculer.

3.2 Appariement d'image (*block-matching*)

Dans le cas du *block matching*, l'objet est directement représenté par son image ou bloc de couleurs. Ce type d'approche s'avère approprié dans le cas d'objets dont le mouvement est rigide, c'est-à-dire lorsque tous les pixels de la fenêtre ont le même déplacement.

Ces approches se distinguent entre elles par le type de distance/similarité utilisée et également par la stratégie de recherche de la nouvelle position. Les tableaux 3.1 et 3.2 donnent des exemples de mesures de distance et de similarité. La figure 3.1 illustre le mécanisme de recherche de l'objet.

En supposant un faible déplacement de l'objet entre deux instants différents, on peut affirmer que l'objet se retrouve dans un voisinage de taille w autour de sa position précédente.

3.3 Histogrammes de l'objet

Matching d'histogrammes.

L'objet peut également être représenté par son histogramme. Il peut s'agir par exemple d'histogrammes de luminance, de couleur, de gradients, de gradients orientés.

Sum of Absolute Differences	$SAD(u,v) = \sum_{(x,y)\in W} \lvert I_t(x,y) - I_{t-1}(x+u,y+v)\rvert$
Sum of Square Differences	$SSD(u,v) = \sum_{(x,y)\in W} \left(I_t(x,y) - I_{t-1}(x+u,y+v)\right)^2$
Distance de Kolmogorov-Smirnov	$KSD(u,v) = \max_{(x,y)\in W} \lvert I_t(x,y) - I_{t-1}(x+u,y+v)\rvert$

TABLE 3.1 – Exemples de mesures de distances entre deux imagettes. Avant le calcul de distance, il est possible d'effectuer une normalisation photométrique. Ces mesures s'appellent alors SADN, SSDN, KSDN

Variance of the differences	$\sigma_e^2(u,v)$ où $e(u,v) = \sum_{(x,y)\in W} \lvert I_t(x,y) - I_{t-1}(x+u,y+v)\rvert$
NCC : Normalized Cross Correlation	$\frac{\sum_{(x,y)\in W} I_t(x,y).J_{t-1}(x+u,y+v)}{\sum_{(x,y)\in W} I_t(x,y) \sum_{(x,y)\in W} I_{t-1}(x+u,y+v)}$

TABLE 3.2 – Exemples de mesures de similarité utilisées dans les approches par *block-matching*. Citons aussi ZNCC, qui est la mesure NCC après normalisation photométrique des deux imagettes.

En utilisant le même mécanisme de Block-matching que dans la section 3.2, la mesure de similarité des blocs est remplacée par une mesure sur les histogrammes. Le tableau 3.3 recense quelques mesures de distances ou similarité inter-histogramme entre deux histogrammes h_0 et h_1.

Mean-Shift.

Le matching d'histogramme n'est pas efficace lorsque la recherche de l'objet se fait de manière exhaustive dans toute l'image. Le Mean-Shift [7] et ses variantes consistent aussi à trouver la position de l'objet dont l'histogramme est le plus proche d'un histogramme de référence, mais cela est fait de manière itérative. Le calcul de la nouvelle position de l'objet est calculée de ma-

HAD	$d(h_0,h_1) = \sum_i \lvert h_0(i) - h_1(i)\rvert$
Battacharryya	$d(h_0,h_1) = \sum_i \sqrt{(h_0(i)h_1(i))}$
Intersection d'histogramme	$d(h_0,h_1) = \sum_{i=1}^{N} \min\left[h_0(i), h_1(i)\right]$
Chi-2 :	$d(h_0,h_1) = \sum_{i=1}^{N} \frac{\left(h_0(i) - h_1(i)\right)^2}{h_0(i) + h_1(i)}$

TABLE 3.3 – Mesures de distances/similarité inter-histogramme. L'histogramme peut être normalisé avant calcul.

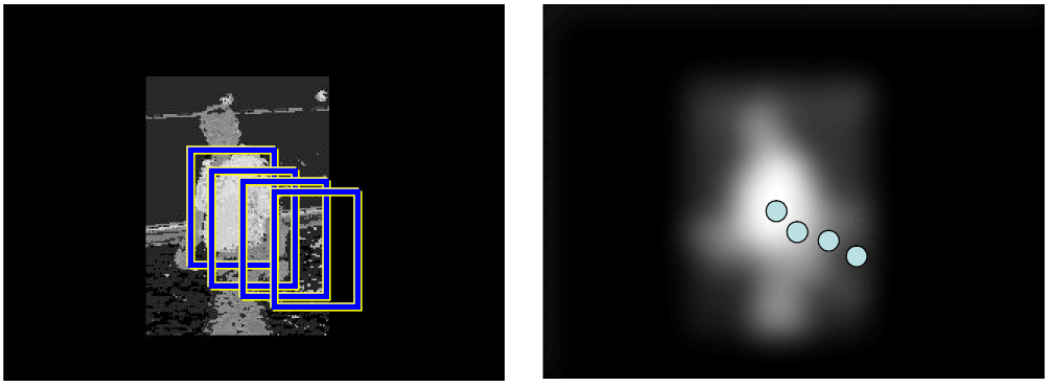


FIGURE 3.2 – Illustration de l'algorithme Mean-Shift. À gauche : image des poids w_i où la valeur est élevée (pixels clairs) lorsque la couleur du point a une forte probabilité d'appartenir à l'objet recherché (forte similarité avec l'histogramme précédent). À droite : convolution avec un noyau gaussien K pour affecter un poids plus fort aux pixels situés au centre. Le calcul de la nouvelle position est la moyenne des coordonnées x, y pondérées par ces poids.

nière suivante :

$$x_{k+1} = \frac{\sum_{i \in W} w_i x_i K(x_i)}{\sum_{i \in W} w_i K(x_i)} \tag{3.1}$$

$$y_{k+1} = \frac{\sum_{i \in W} w_i y_i K(x_i)}{\sum_{i \in W} w_i K(x_i)} \tag{3.2}$$

Ici, k correspond à l'itération. L'algorithme stoppe lorsque la nouvelle position à l'itération k est très proche de celle à l'itération $k-1$ (convergence). W est la boîte englobante de l'objet. La fonction K correspond à un noyau gaussien qui attribue un poids plus important au centre de la fenêtre précédente. w_i est une pondération dépendant de la probabilité d'appartenance du pixel à l'objet (voir 3.2). Son calcul dépend de la similarité entre les couleurs des objets à apparier. Dans l'algorithme initial, il s'agit de la distance de Bhattacharryya.

3.4 Mise en correspondance de points d'intérêt

Il existe également un grand nombre de méthodes de mise en correspondance de points qui se fondent toutes sur 3 étapes :

- la détection de points d'intérêt dans l'image
- la définition d'un descripteur de chaque point (histogramme par exemple)
- la mise en correspondance ou appariement entre les points de deux images par comparaison de leurs descripteurs

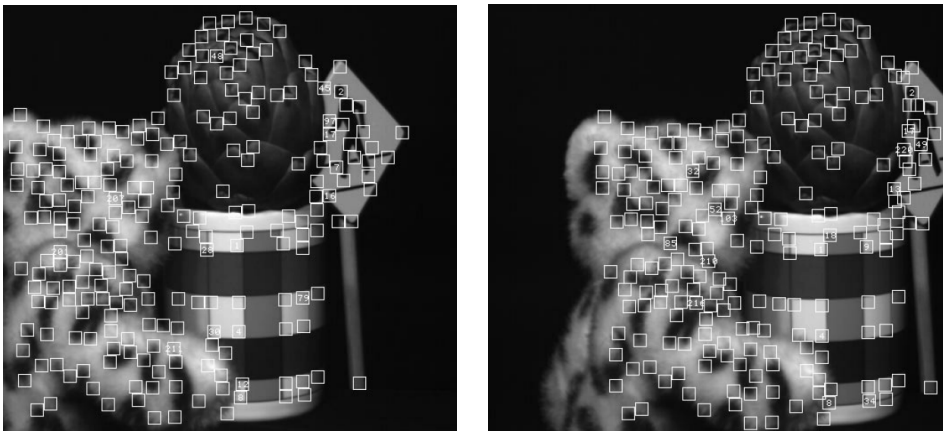


FIGURE 3.3 – Exemple de suivi KLT, issu de [5]. À gauche : les points détectés au début. À droite : les points correctement suivis à la fin de la séquence.

3.4.1 Détection de points d'intérêt

3.4.1.1 Good Features to Track.

Il existe un grand nombre de méthodes de suivi de points d'intérêt. La plus communément utilisée est l'approche KLT pour Kanade-Lucas-Tomasi [3–5] et ses nombreuses variantes. Cela se produit lorsque le point considéré est situé sur une zone pour laquelle l'un et/ou l'autre des gradients en x ou y est faible (sur les zones homogènes et sur les contours unidirectionnels). L'approche KLT consiste à détecter les points saillants pour lesquels le tenseur est inversible puis à calculer le mouvement en ce point (exemple dans la figure 3.3). Plus concrètement, on calcule les valeurs propres de la matrice λ_1 et λ_2. Si λ_1 et λ_2 sont faibles, cela correspond à une zone d'intensité homogène. Si l'une est élevée et l'autre faible, il s'agit d'un contour unidirectionnel. Les points sélectionnés sont ceux pour lesquels les deux valeurs propres sont élevées :

$$min(\lambda_1, \lambda_2) > \lambda$$

La librairie OpenCV fournit une implémentation de cet algorithme. `GoodFeaturesToTrackDetector` permet de détecter les points saillants et `calcOpticalFlowPyrLK` correspond à une approche KLT multi-échelle (Pyramidale).

3.4.1.2 Détecteur FAST (Features from Accelerated Segment Test).

FAST utilise un cercle autour du pixel central p_c (voir ci-dessous avec un cercle de 16 pixels).

			p_{16}	p_1	p_2			
		p_{15}				p_3		
	p_{14}						p_4	
	p_{13}			p_c			p_5	
	p_{12}						p_6	
		p_{11}				p_7		
			p_{10}	p_9	p_8			

Les pixels du cercle forment un vecteur $[p_1, p_2, ..., p_16]$. À chaque pixel p_i, on retranche la valeur du pixel central p_c : $[p_1 - p_c, p_2 - p_c, ..., p_16 - p_c]$. Ce vecteur est ensuite seuillé à l'aide de 2 seuils s_h (haut) et s_b (bas) pour obtenir une valeur parmi 3 :

— -1 si $p_i - p_c < s_b$

— 0 si $s_b < p_i - p_c < s_h$

— 1 si $p_i - p_c > s_h$

Un coin est ainsi détecté suivant les séquences de 0, de 1 ou -1 trouvés. Par exemple si p_c est sur un coin à angle droit, il y aurait 5 pixels p_i à 0.

3.4.2 Descripteurs de points

3.4.2.1 SIFT

Les descripteurs SIFT (pour Scale Invariant Feature Tranform) [8] sont actuellement les plus largement utilisés. Notons également la variante SURF (Speeded-Up Robust Feature) [9], qui se base sur les ondelettes de Haar pour accélérer le calcul des descripteurs. Ces descripteurs sont disponibles dans OpenCV. SIFT fonctionne selon les 6 étapes suivantes.

1. À partir de l'image initiale, une représentation *espace-échelle* (ou *scale-space* en anglais) est créée. Il s'agit d'un ensemble de transformations de l'image initiale en N images (appelées *octaves*) de taille de plus en plus réduite (voir figure 3.4). Chacune des N images est lissée avec M différents niveaux de lissage (filtrage gaussien avec des valeurs croissantes d'écart-type σ_m pour $m = 0...M - 1$).

2. Calcul du DOG (*difference of Gaussians*) (voir figure 3.4). Pour chaque octave, on calcule la différence entre l'image lissée par σ_m et celle lissée par σ_{m-1} avec $\sigma_m > \sigma_{m-1}$

3. Les *extrema* locaux du DOG sont détectés aux différentes échelles. L'échelle pour laquelle le maximum local est le plus élevé est considérée comme l'échelle du point d'intérêt.

4. Les points correspondant à des zones homogènes ou à des contours unidirectionnels sont écartés. Pour cela on calcule le tenseur de structure ou matrice Hessienne (évoqué dans la méthode du flot optique de Lucas-Kanade **??**). Si les deux valeurs propres sont élevées, le point est considéré comme saillant (gradient fort dans deux directions).

5. Les voisinages des points saillants sont analysés. La taille du voisinage dépend de l'échelle du point trouvée précédemment. On calcule l'histogramme des orientations du gradient (en abscisse : angles de 0 à 2 π, et en ordonnées : le nombre d'occurrences de l'angle). L'orientation majoritaire (qui correspond au maximum dans l'histogramme) est choisie comme l'orientation principale du point d'intérêt.

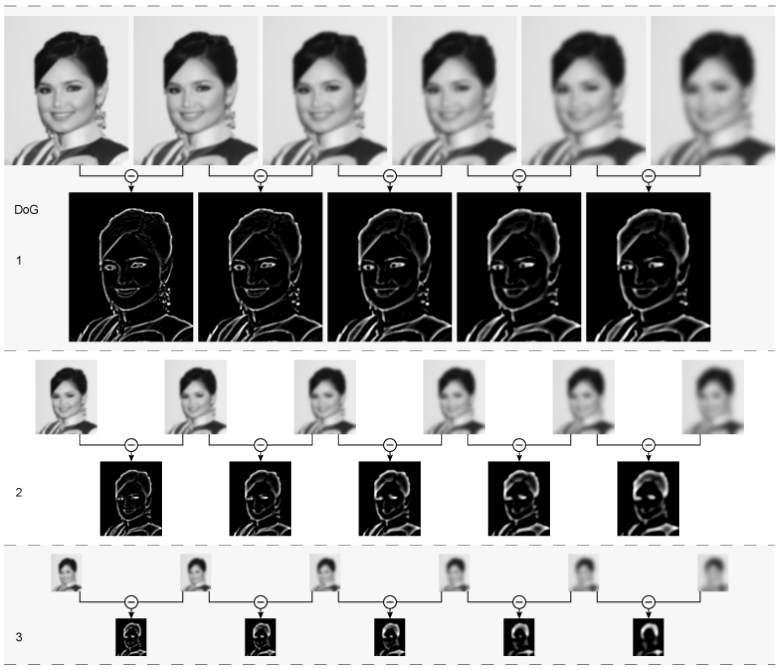


FIGURE 3.4 – Différences de Gaussiennes Par Siti_Nurhaliza.jpg : amrfumderivative work : Indif (talk), Siti_Nurhaliza.jpg, CC BY 2.0, https ://commons.wikimedia.org/w/index.php ?curid=11864726

6. Finalement, le point est décrit comme un ensemble d'histogrammes de gradients comme indiqué sur la figure 3.5. Les histogrammes sont normalisés de deux manières : 1) on applique un décalage des abscisses de manière à ce que l'orientation principale du gradient (calculée précédemment) se retrouve à l'origine. Ainsi, le descripteur est invariant en rotation ; 2) avant de calculer les gradients, on applique une normalisation photométrique (voir TP1, TP3).

3.4.2.2 KAZE

Le descripteur AKAZE (KAZE Features and Accelerated-Kaze Features) [10] se fonde sur des principes proches du SIFT et qui est plus performant à la fois en termes de performances de détection de points (, nombre de points détectés, répétabilité) et en termes de temps d'exécution. Au lieu d'un espace-échelle linéaire (gaussien), l'espace échelle est non-linéaire et permet de mieux conserver les détails (contours) tout en supprimant le bruit. Un autre avantage de ce descripteur est qu'il est open source, contrairement au SIFT qui est breveté.

3.4.2.3 BRIEF

Le descripteur BRIEF (*Binary Robust Independent Elementary Features* [11]) utilise le voisinage circulaire autour du point central. Il utilise un test binaire pour mesurer le signe des variations d'intensité entre deux points. Soit p_i l'intensité de l'image au point i :

$$f(p,i,j) = \begin{cases} 1 & si \quad p_i < p_j \\ 0 & \quad sinon \end{cases}$$

La liste des résultats de ces tests binaires est stockée dans un vecteur de n bits (créé à partir

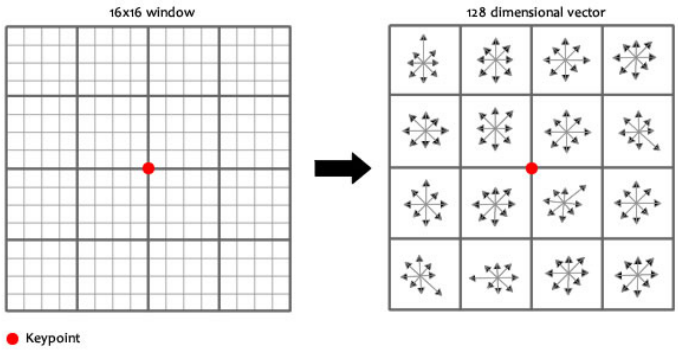


FIGURE 3.5 – Les points SIFT sont décrits par 16 histogrammes de gradients. Chaque histogramme contient 8 classes, une classe par direction.

des points comparés). Pour le descripteur ORB, n= 256.

$$u_n(p,i,j) = \sum_n 2^{i-1} f(p,i,j)$$

3.4.3 Estimation d'une transformation

Une fois la mise en correspondance effectuée entre deux images, comme sur l'exemple de la figure 3.6(a), il est possible de calculer la transformation géométrique entre les deux images. On considère n points SIFT dans l'image I_1 et m points SIFT dans l'image I_2

$$I_2 = \mathbf{H}(I_1)$$

Dans le cas où l'image est assimilée à un plan, la transformation **H** correspond à une **homographie**. Elle peut être estimée par des méthodes de type RANSAC (RANdom SAmple Consensus).

$$s \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = \mathbf{H} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix}$$

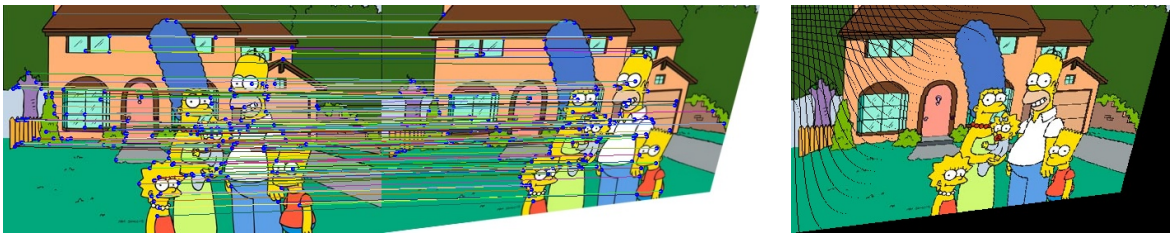
s correspond à l'échelle.

Cette méthode est également utilisée pour recaler deux images entre elles, pour créer une image panoramique par exemple, ou obtenir une image 3D.

3.5 Contours actifs

Cette méthode est également appelée **contours déformables** ou **snakes** [6]. L'objectif est de trouver le contour optimal d'une région dans l'image courante, par déformation du contour obtenu dans l'image précédente. Cette méthode est particulièrement appropriée pour suivre des objets déformables (voir figure 3.7). Cela se fait en minimisant une fonction d'énergie associée aux points du contour, calculée comme la somme de deux termes d'énergie :

$$E_{snakes} = E_{ext} + E_{int}$$



(a) (b)

FIGURE 3.6 – Exemple de mise en correspondance entre deux images. (a) : appariement et calcul de la transformation géométrique entre les deux images. (b) : on applique la transformation calculée précédemment.

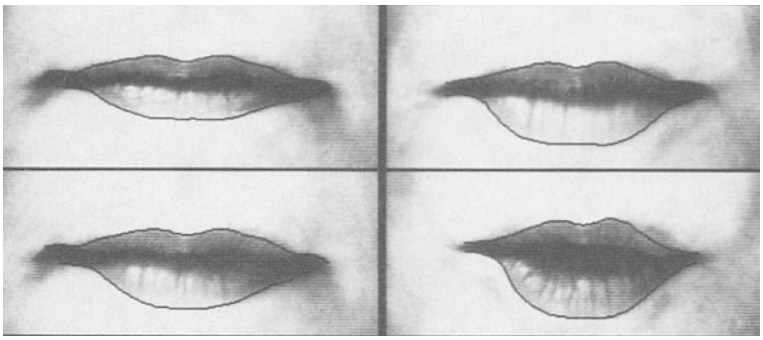


FIGURE 3.7 – Exemple de suivi de contours [6]

- E_{ext} est minimale lorsque le contour actif est positionné sur les frontières de l'objet (énergie faible lorsque le gradient est élevé).
- E_{ext} est minimale quand le contour actif a une forme pertinente par rapport à celle de l'objet recherché. Généralement, elle doit être minimale lorsque la forme est régulière et lisse et maximale lorsqu'elle contient des irrégularités et des courbures importantes.

3.6 Travail à réaliser

3.6.1 Mise en correspondance temporelle

1) Dans cette partie, vous testerez le projet `video04` sur différentes séquences (voir le fichier de configuration `config_video04.cfg`. Les méthodes de suivi implémentées sont les suivantes :

— Suivi de points (ORB et AKAZE) avec calcul d'homographie.
L'ORB est essentiellement une fusion du détecteur de points clés FAST (voir section 3.4.1) et du descripteur BRIEF (voir section 3.4.2.3), avec de nombreuses modifications pour améliorer les performances. Il utilise d'abord FAST pour trouver les points clés, puis applique la mesure de coin de Harris pour trouver les N premiers points parmi eux. Ensuite il utilise BRIEF pour décrire le point.

— Suivi Mean Shift

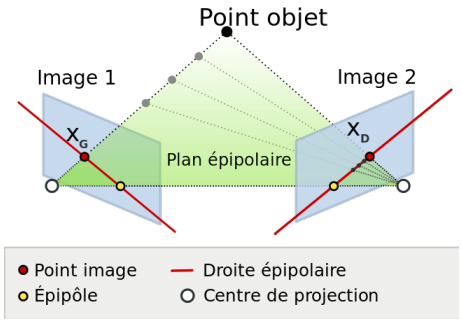


FIGURE 3.8 – Système stéréoscopique et lignes épipolaires.

2) Quelles sont les situations de mise en échec des deux méthodes ?

3.6.2 Mise en correspondance spatiale : stéréovision

Les explications concernant la stéréovision sont fournies en annexe .1.

1) Mettre en place l'appariement stéréoscopique en complétant `stereomatching` dans `stereo.cpp`. À partir d'une image de gauche `L`, et une image de droite `R`, cette méthode effectue la mise en correspondance de blocs de pixels de taille `2N+1` × `2N+1` pour obtenir une carte de disparité `D`. La mesure de similarité ou de distance est à spécifier dans `config_video05.cfg` (paramètre `type`).

Pour chaque pixel de l'image gauche, on cherche son homologue dans l'image droite sur la même ligne. On considérera une distance de recherche maximale `disp_max` (voir `configVideo.cfg`. On testera tout d'abord la distance SAD.

2) Tester les différentes mesures de similarité et comparer les résultats sur les 4 paires d'images de `test1`.

3) D'où viennent les erreurs d'appariement ?

4) Modifier le code `stereomatching` pour détecter et supprimer les faux appariements (en mettant à 0 dans la carte de disparité, les pixels mal appariés).

5) Choisir le critère qui vous paraît le plus performant sur les différentes images des séquences `test2` à `test4`.

Bibliographie

[1] Zoran Zivkovic. Improved adaptive gaussian mixture model for background subtraction. In Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on, volume 2, pages 28–31. IEEE, 2004.

[2] B.K.P. Horn and B.G. Schunck, "Determining optical flow." Artificial Intelligence, vol 17, pp 185-203, 1981

[3] B. D. Lucas and T. Kanade (1981), An iterative image registration technique with an application to stereo vision. Proceedings of Imaging Understanding Workshop, pages 121–130

[4] Shi and C. Tomasi. Good Features to Track. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 593-600, June 1994.ibitemLK81 B. D. Lucas and T. Kanade (1981), An iterative image registration technique with an application to stereo vision. Proceedings of Imaging Understanding Workshop, pages 121–130

[5] Carlo Tomasi and Takeo Kanade. Detection and Tracking of Point Features. Carnegie Mellon University Technical Report CMU-CS-91-132, April 1991.

[6] Michael Kass and Andrew Witkin and Demetri Terzopoulos. International Journal of Computer Vision, 321-331 (1988)

[7] Dorin Comaniciu. Mean shift : a robust approach toward feature space analysis. IEEE Transactions on Pattern Analysis and Machine Intelligence 24(5) 2002

[8] Lowe, D. G., "Distinctive Image Features from Scale-Invariant Keypoints", International Journal of Computer Vision, 60, 2, pp. 91-110, 2004.

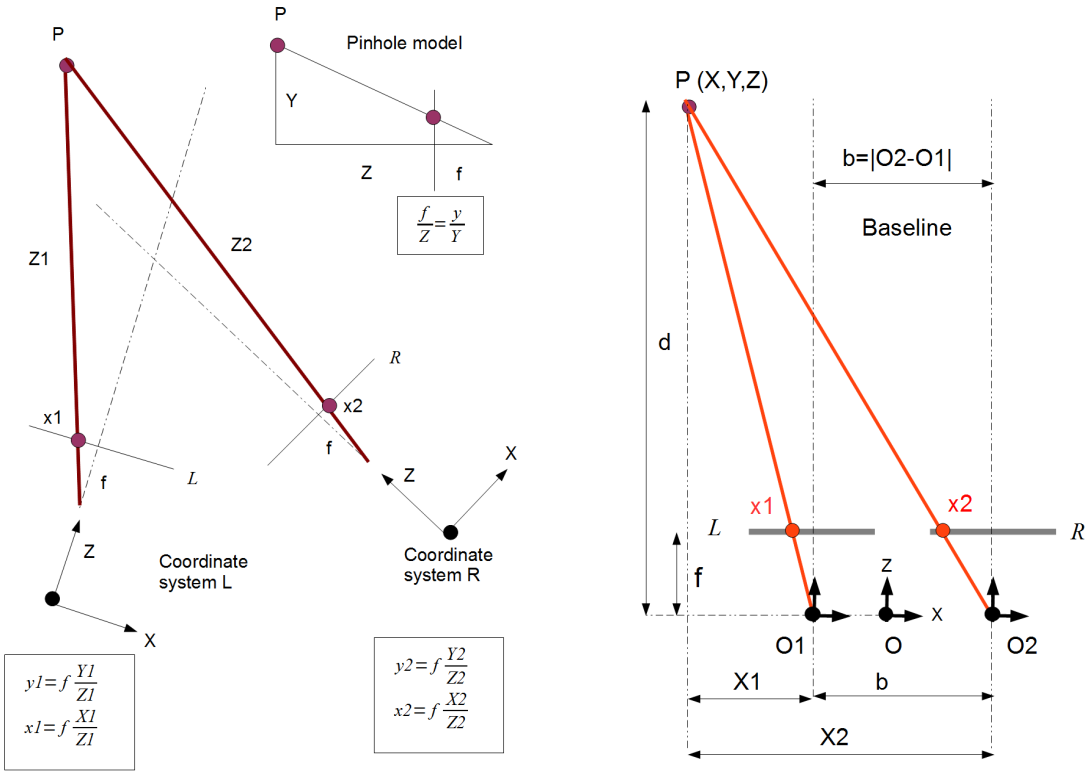
[9] Bay, H and Ess A and Tuytelaars T. and Van Gool L., "Speeded-Up Robust Features (SURF) ", Computer Vision and Image Understanding, 110(3), pp. 346-359, 2008

[10] KAZE Features. Pablo F. Alcantarilla, Adrien Bartoli and Andrew J. Davison. In European Conference on Computer Vision (ECCV), Fiorenze, Italy, October 2012.

[11] Calonder, M., Lepetit, V., Konolige, K., Bowman, J., Mihelich, P., Fua, P. : Compact Signatures for High-Speed Interest Point Description and Matching. In : International Conference on Computer Vision. (2009)

.1 Stéréovision

On considère deux capteurs qui observent la même scène. Nous allons voir que, par **triangulation**, il est possible de déterminer la distance de tout point de la scène par rapport aux capteurs en se basant sur les propriétés de géométrie 3D du système (voir figure 9(a)).



(a) (b)

FIGURE 9 – (a) Géométrie d'un système stréoscopique. (b) Géométrie d'un système stéréoscopique après calibrage et rectification.

Les coordonnées (X, Y, Z) correspondent aux coordonnées de points situés dans la scène réelle 3D, tandis que les coordonnées (x, y, z) correspondent aux projections de ces points dans le plan image. z est égal à la focale du capteur f. A l'aide d'un modèle simplifié de camera dit **modèle sténopé** [1] (ou *Pinhole model* en anglais), on aboutit à une relation entre les deux types de coordonnées.

On considère un point dans la scène de coordonnées X, Y, Z. Il se projette en (x, y) sur un plan image. Si l'on exprime les coordonnées XYZ dans le repère associé à la caméra, Z correspond à la distance entre le centre optique de la caméra et le point, et f est la **distance focale** du capteur :

$$x = f\frac{X}{Z} \text{ et } \quad y = f\frac{Y}{Z} \tag{3}$$

Pour caluler la distance Z il faudrait connaître f et X ou Y, ce qui n'est pas possible à partir d'une seule image (il y a plus d'inconnues que d'équations).

1. Le modèle sténopé est dérivé de la *camera obscura* qui signifie chambre noire. Il s'agit d'une boîte noire dans laquelle est fait un trou très fin laissant passer la lumière. Au fond de la boîte on place du papier photosensible. Après exposition du papier, on obtient une photographie.

Essayons avec deux images acquises par deux caméras. Le point se projette maintenant en (x_1, y_1) sur le plan image de gauche et en (x_2, y_2) sur la plan image de droite.

$$x_1 = f\frac{X_1}{Z_1};\;\; y = f\frac{Y_1}{Z_1};\;\; x_2 = f\frac{X_2}{Z_2};\;\; y_2 = f\frac{Y_2}{Z_2} \tag{4}$$

Malheureusement là encore, il y a plus d'inconnues que d'équations et on ne peut pas déduire la distance du point car les coordonnées X_1, Y_1, Z_1 et X_2, Y_2, Z_2 ne sont pas dans le même repère. Pour remédier au problème, il est possible de transformer toutes les coordonnées pour les ramener dans un seul et même repère. Cela nécessite une étape de calibrage.

Calibrage. En utilisant des mires réelles (des objets pour lesquels on connaît les coordonnées XYZ de plusieurs points comme sur la figure 10) et en acquérant les images de cette mire dans les deux capteurs, on peut calculer la **matrice fondamentale**. Cette matrice relie les deux repères caméra par un **vecteur de translation** et une **matrice de rotation** dans l'espace 3D. Ce sont les **paramètres extrinsèques** du système (qui dépendent de la position du capteur). On calcule également les **paramètres intrinsèques** (propres au capteur) : distance focale, distortions radiales dues à la lentille.

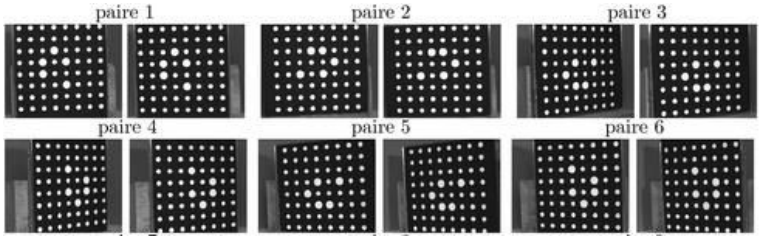


FIGURE 10 – Exemple de séquences d'images d'une même mire de calibrage.

Rectification. Une fois ces paramètres calculés, il est possible de transformer les images afin de simuler l'alignement des deux plans image et faire en sorte qu'ils partagent le même système de coordonnées tel que représenté par la figure 9(b). On appelle cette étape la **rectification**. Maintenant le point XYZ est à une même distance $d = Z_1 = Z_2$ de chaque plan image. Les plans image sont alignés et les axes optiques sont parallèles.

$$x_1 = f\frac{X_1}{d};\;\; x_2 = f\frac{X_2}{d}; \tag{5}$$

Calcul de la disparité. Il y a encore trop d'inconnues dans (5). Par contre, à partir du schéma 9(b), on peut exprimer $X_2 - X_1$ en fonction de la distance entre les deux axes optiques appelée b comme *baseline*.

$$b = X_2 - X_1 = O_2 - O_1$$

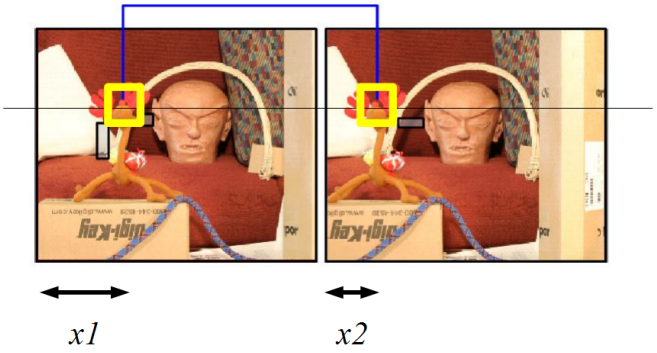


FIGURE 11 – Après rectification des deux images, la recherche des appariements se fait sur les lignes.

$$x_2 - x_1 = f\frac{X_2 - X_1}{d} = f\frac{b}{d}$$

Par calibrage, nous connaissons la baseline b. Ainsi, la profondeur peut être calculée :

$$d = f\frac{b}{x_2 - x_1}$$

Il ne reste plus qu'à calculer la distance $x_2 - x_1$ qui correspond à la **disparité**. Cela se fait comme indiqué dans la figure 11. Après rectification un point de l'image gauche se retrouve sur la **même ligne** dans l'image droite.

Définition : épipole. Une épipole est la projection dans l'image I_1 du centre optique de l'image I_2 (voir 3.8).

Définition : ligne épipolaire. Une ligne épipolaire est une ligne qui rejoint les deux épipoles. Après rectification les lignes épipolaires sont alignées.