Travaux pratiques
Séance 1 : mise en correspondance Stéréo

Traitement d'images et vision - M2 E3A SAM - UEVE

L'objectif de cet exercice est de découvrir les principaux défis auxquels doivent faire face les algorithmes de mise en correspondance stéréo. Nous partirons d'un algorithme très intuitif et tenterons de proposer des pistes d'amélioration.

- Des couples/séquences d'images stéréoscopiques associées à des cartes de disparité de vérité terrain sont disponibles sur le site de MiddleBury (http ://vision.middlebury.edu/stereo/).
- Afin d'évaluer la précision de votre algorithme, vous pourrez calculer l'erreur MRE (*Mean Relative Error*) entre la carte de disparité obtenue et la carte de disparité de vérité terrain.

$$MRE = \frac{1}{N} \sum_{x,y} \frac{|D_{estime}(x,y) - D_{GT}(x,y)|}{D_{GT}(x,y)}$$

- Un autre type d'évaluation à l'aveugle consistera à calcul l'erreur moyenne de reprojection. Connaissant la disparité d en un point, il suffit de vérifier ensuite que :

$$|I_1(x,y) - I_2(x,y-d)| \sim 0$$

- ll vous sera aussi possible de comparer vos résultats avec ceux obtenus par la fonction **disparityBM** déjà existante dans Matlab. La description de la liste des paramètres en entrée de cette fonction pourra vous donner quelques pistes.
- Il est possible d'initialiser une variable x avec une valeur égale à l'infinie (en pratique la valeur maximale du type considéré) en utilisant : x=**inf**.
- Les points pour lesquels la disparité n'a pas pu être calculée (non valide) auront une disparité égale à **NaN**. Afin de sommer les valeurs d'une image comportant des **NaN**, il est possible d'utiliser la fonction **nanSum**.
- La disparité calculée pourra être affichée sous la forme d'une image en niveaux de gris puis en changeant de palette de couleur (utiliser **colormap** jet).

Exercise 1 : Implémentation d'un algorithme de type *Block Matching*

1. Charger et afficher les images : "venus1.ppm" et "venus2.ppm". Dans la suite, nous travaillerons avec une version en niveaux de gris de ces images. La fonction **rgb2gray** permet de faire la conversion.
2. A votre avis, quelle est la configuration particulière des caméras qui ont servi à acquérir ces images ? Vérifier vos constatations en utilisant la fonction **stereoAnaglyph**. Est-il nécessaire de rectifier ce couple d'images ?
3. Afficher le profil d'une ligne donnée pour les deux images. Conclure.
4. Utiliser la fonction **iminfo** afin d'estimer visuellement la disparité minimum et la disparité maximum dans l'image.
5. Ecrire une fonction qui détermine pour chaque point de l'image gauche son correspondant dans l'image droite. La fonction devra renvoyer une carte de disparité codée sur 8 bits. La liste des paramètres en entrée de la fonction sont :

— I1, I2 : images en entrée en niveaux de gris
— blocSize : taille du bloc, en pratique on utilisera la demi taille de bloc : blocSize2 = **fix**(blocSize/2) ;.
— dispMax : la disparité maximum (permet de fixer l'intervalle de recherche)
— seuilContraste : le seuil en dessous duquel un bloc est considéré comme homogène. Ce paramètre doit être compris entre 0 et 1.

Vous pourrez éventuellement compléter le code suivant sans que cela soit une obligation :

```
[h,w] = size(I1);
D=zeros(h,w);
for i1=1+BlocSize2:h-BlocSize2
    for j1=1+BlocSize2:w-BlocSize2
        ...
        BL1=I1(i1-BlocSize2:i1+BlocSize2,j1-BlocSize2:j1+BlocSize2);

        ...

        i2=i1;
        for j2=j1:min(w-BlocSize2,j1+dispMax)

            BL2=I2(i2-BlocSize2:i2+BlocSize2,j2-BlocSize2:j2+BlocSize2);

            %calculer ici la mesure de similarite
            % stocker les mesures dans un vecteur

            %tester si la mesure courante est la meilleure
            % ou le faire une fois la ligne parcourue en dehors du for.

        end
        % calcul de la disparite (meilleure association trouvee)

        end

    end
end
```

.

6. Discuter l'influence des différents paramètres : taille du bloc, disparité maximum et seul de contraste.
7. Tester [1] plusieurs mesures de similarités : distances (SAD, SSD) ou corrélations (coefficient de corrélation normalisé). Conclure.

$$SAD(x,y,d) = \sum_{(x,y)\in w} |I_1(x,y) - I_2(x,y-d)|$$

$$SSD(x,y,d) = \sum_{(x,y)\in w} (I_1(x,y) - I_2(x,y-d))^2$$

$$NNC(x,y,d) = \frac{\sum_{(x,y)\in w} I_1(x,y) I_2(x,y-d)}{\sqrt{\sum_{(x,y)\in w} I_1^2(x,y) \sum_{(x,y)\in w} I_2^2(x,y-d)}}$$

8. Tester [1] plusieurs mesures permettant d'éviter de prendre en compte les blocs homogènes. On testera d'abord une mesure de contraste simple :

$$c_{bloc} = \frac{max\{I(x,y)\} - min\{I(x,y)\}}{max\{I(x,y)\}}$$

1. Chaque test fera l'objet de comparatif en utilisant les critères d'évaluation décrits en introduction

qui a l'avantage d'être comprise entre 0 et 1. On tentera ensuite d'éviter les blocs dont le module de gradient est inférieur à un seuil (utiliser SOBEL).

9. Proposer une méthode pour boucher les trous de la carte de disparité. Implémenter l'approche proposée (suggestion : **imdilate**).
10. Expérimenter [1] différentes techniques pour accélérer et améliorer votre implémentation. Par exemple :
    - Vérifier la consistance des appariements : droite gauche puis gauche droite (les appariements retenus sont ceux qui sont identiques dans les deux sens).
    - Assouplir la contrainte d'appartenance de la primitive à la même ligne image.
11. Afin de tenir compte des disparités estimées dans le voisinage, les techniques basées sur la relaxation ou sur la programmation dynamique sont des approches bien adaptées. Dans les deux cas, l'idée est de formuler l'estimation de la disparité comme un problème d'optimisation où l'énergie à minimiser s'écrit pour chaque pixel :

$$E(d) = E_d(d) + \lambda E_s(d)$$

$E_d(d)$ est le terme d'attache aux données (basé sur le critère de similitude) tandis que $E_s(d)$ est le terme de lissage (permet aux pixels voisins d'obtenir des disparités proches). Dans notre cas, nous pouvons choisir $E_d(d) = \sum_{(x,y)} C(x, y, d(x, y))$ c'est-à-dire la distance (SAD, SSD, etc.) entre deux blocs. Pour ce qui concerne le terme de lissage, nous testerons deux approches :

1. L'approche par **relaxation** : dans cette méthode, $E_s(d) = \sum_{(p,q)\in \mathbf{w}} V(d_p, d_q)$ où $\mathbf{w}$ est le voisinage 4-connexité ou 8-connexité du pixel analysé et $V = |d_p - d_q|$ est la distance L_1 entre les disparités en p et en q.
2. L'approche par **programmation dynamique** : l'idée est d'optimiser le long d'une ligne pour trouver les meilleurs associations au niveau global. $D(x, y, d)$ est l'erreur à minimiser avec :

$$D(x, y, d) = C(x, y, d) + \min_{d'}\{D(x - 1, y, d') + \lambda|d - d'|\}$$

Implémenter chacune des approches ci-dessus et commenter les résultats. Quelles pistes d'amélioration suggérez-vous en guise de perspectives à ce travail ?

FIN