Vision robotique

Travaux pratiques séance 2

Yuhan DOU

I. Objectifs

Cette séance illustre plusieurs méthodes d'analyse du mouvement. On considère une séquence d'environ 20 s / 600 images prise par une caméra.

L'objetif de ce TP est de détecter les voitures en mouvement. Au début, on va le réaliser par la détection du fond dont il n'y a pas de mouvement. Après, on va utiliser l'analyse du flot optique.

II. Détection du fond

1. Une décision basée sur (μ, v) pour classifier tous les pixels.

Dans le premier temps, on calcule le moyenne et le écart-type pour chaque pixel dans la séquence. Puis on définit un seuil, et détermine si un pixel apparaît au objet ou au fond par la formule suivant :

$$I(i,j) = \begin{cases} 0 \text{ (fond), si } | I(i,j)\text{-moyenne} | \leqslant \text{seuil} \\ 1 \text{ (objet), sinon} \end{cases}$$

C'est-à-dire que si la différence entre la valeur du pixel et la moyenne est inférieure que le seuil, on le considère comme le pixel du fond, car l'aspect du fond ne change pas en général.

Code correspondant :

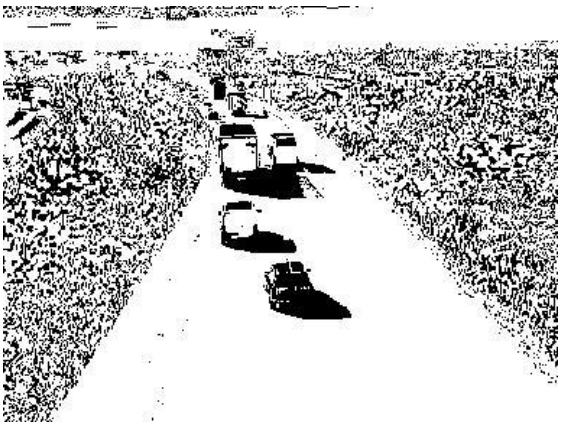
```
moyenne=mean(frame_buffer,3);     %calcul du moyenne de tous les pixels de tous les frames
ecart_type=std(frame_buffer,0,3); %calcul de la variance de tous les pixels de tous les frames

for i=1:N
    diff=abs(frame_buffer(:,:,i)-moyenne);
    objet_fond(:,:,i)=255*(diff<lambda*ecart_type); %comparaison au seuil
    %si la difference entre sa valeur et la moyenne est inferiere que le seuil, on le considere comme le fond
end

for i=1:N
    imshow(objet_fond(:,:,i),[])
    pause(0.033);
end
```

Ici, le seuil est calculé par "seuil = lamda * écart-type" où "lamda" est un paramètre du seuil que on peut faire le choix.

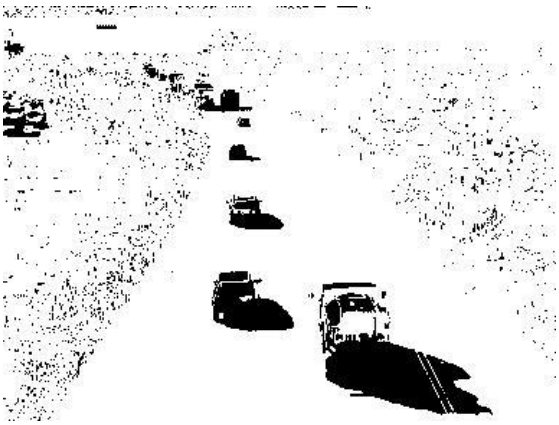
Résultat (Ici on discute sur deux frames typiques):

frame 1 (lamda = 0.1)

frame 2 (lamda = 0.1)



frame 1 (lamda = 1)



frame 2 (lamda = 1)

frame 1 (lamda = 2)

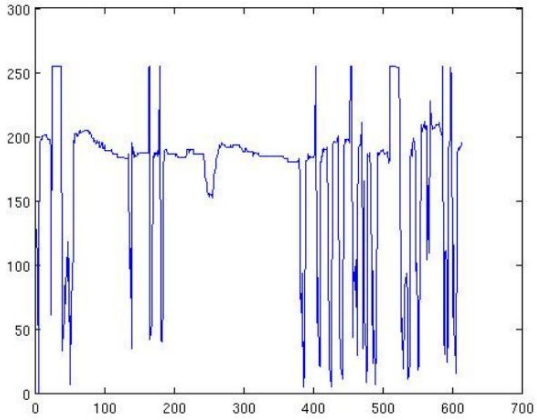
frame 2 (lamda = 2)

Selon la formule pour calculer le seuil ci-dessus, si la valeur du paramètre “lamda” augmente, il y aura plus de pixels qui sont classifiés dans le fond (indiqués comme des pixels blans ici). En comparant les trois valeurs 0.1, 1 et 2, on peut voir que si “lamda” est égale à 1, le résultat est le plus meilleur.

Par ailleurs, dans le cas avec lamda = 1, le faible mouvement des arbres est aussi représenté, qui n’est pas ce que on souhaite. C’est parce que cette méthode pour détecter l’objet n’est pas assez précise.

2. Une méthode pour identifier des plateaux dans les niveau de gris de pixels, pour estimer (μ, v) plutôt en s'appuyant sur ces endroits.

La variation du niveau de gris du pixel (200, 150) est représentée dans la figure ci-dessous :



On se rend compte que le calcul de (μ, v) est trompeur. Parce que sur quelques certaines séquences parmi de toutes les 614 frames du vidéo (par exemple, [50, 120], [190, 250], [260, 380]), le niveau du gris de ce pixel (200, 150) est presque invariant, donc on va le considérer comme le fond. Mais sur autres séquences, son niveau du gris change fortement, donc il faut le classifier comme l'objet. Ainsi, on ne peut pas décider si ce pixel (200, 150) appartient à un objet ou au fond.

3. L'algorithme $\sum\Delta$

Selon l'article, on choisit l'algorithme "Basical $\sum\Delta$" pour optimiser la classification. On définit deux estimateurs M et V. Ses valeur initiale sont respectivement la moyenne et le écart-type de tous les pixels dans la séquence. Les valeurs de ces deux estimateurs sont mettent à jour dans chaque frame par la comparaison avec la valreur courante du pixel I_t et la valeur courant de différence absolue $O_t = |M_t - I_t|$. En fin, on décide le label pour chaque pixel en comparant sa valeur V et O.

Code correspondant :

Algorithm 1: Basical $\Sigma\Delta$

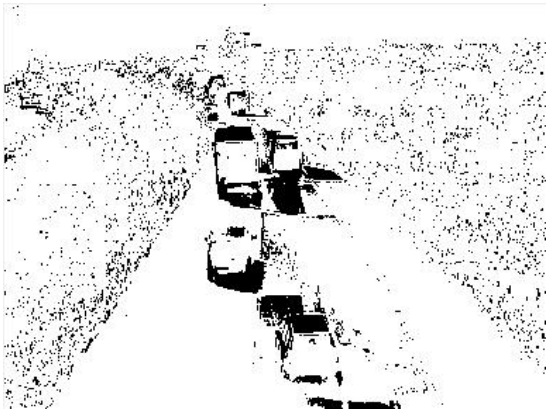
foreach *pixel* x **do** [step #1: M_t estimation]
  if $M_{t-1}(x) < I_t(x)$ **then** $M_t(x) \leftarrow M_{t-1}(x) + 1$
  if $M_{t-1}(x) > I_t(x)$ **then** $M_t(x) \leftarrow M_{t-1}(x) - 1$
  otherwise $M_t(x) \leftarrow M_{t-1}(x)$
foreach *pixel* x **do** [step #2: O_t computation]
  $O_t(x) = |M_t(x) - I_t(x)|$
foreach *pixel* x **do** [step #3: V_t update]
  if $V_{t-1}(x) < N \times O_t(x)$ **then** $V_t(x) \leftarrow V_{t-1}(x) + 1$
  if $V_{t-1}(x) > N \times O_t(x)$ **then** $V_t(x) \leftarrow V_{t-1}(x) - 1$
  otherwise $V_t(x) \leftarrow V_{t-1}(x)$
  $V_t(x) \leftarrow max(min(V_t(x), V_{max}), V_{min})$
foreach *pixel* x **do** [step #4: $\hat{E}_t$ estimation]
  if $O_t(x) < V_t(x)$ **then** $\hat{E}_t(x) \leftarrow 0$ **else** $\hat{E}_t(x) \leftarrow 1$

```
moyenne=mean(frame_buffer,3);
v=std(frame_buffer,0,3);

for i=1:N
    moyenne=moyenne-(moyenne>frame_buffer(:,:,i))+(moyenne<frame_buffer(:,:,i)); %mise a jour du estimateur M
    O(:,:,i)=abs(frame_buffer(:,:,i)-moyenne);
    v=v+(v<(N_para*O(:,:,i)))-(v>(N_para*O(:,:,i))); %mise a jour du estimateur V ("N_para" est un param que on choisit)
    v=max(min(v,VMAX),VMIN);                         %correction l'effet de sur-exposition (VMAX=255, VMIN=2)
    objet_fond(:,:,i)=255*(O(:,:,i)<v);              %decision de pixel
end

for i=1:N
    imshow(objet_fond(:,:,i),[])
    pause(0.033);
end
```

Ici, les deux paramètres “VMAX” et “VMIN” sont des parametres pour la correction du debordement dans le cas de sur-exposition. Typiquement, “VMIN” prend 2. Pour “VMAX”, on choisit 255 car il y a 8 niveaux du gris.

En plus, “N_para” est un paramètre d’amplification pour la variance V. En général, il prend la valeur entre 1 et 4. On teste avec plusieurs valeurs, et trouve que quand N_para = 2.5, le résultat est le plus meilleur.

Résultat :

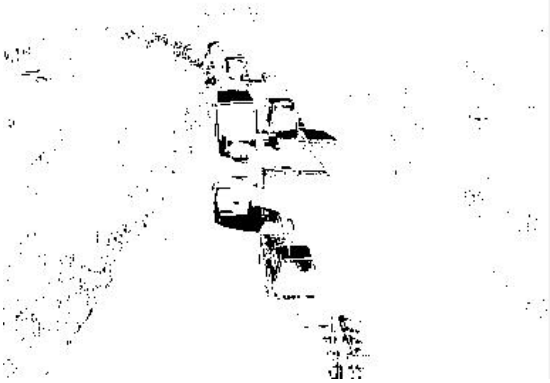


frame 1 (N_para = 1)

frame 2 (N_para = 1)

frame 1 (N_para = 2.5)

frame 2 (N_para = 2.5)



frame 1 (N_para = 4)

frame 2 (N_para = 4)

On peut voir que si la valeur de "N_para" augmente, il y aura plus de pixels qui sont classifiés dans le fond (indiqués comme des pixels blans ici). En comparant les trois valeurs 1, 2.5 et 4, on peut voir que quand "N_para" est égale à 2.5, le résultat est le plus meilleur.

Évidemment, cet algorithme est meuilleur que la méthode précédente pour identifier les voitures quand il y a de faible mouvement des arbres.

III. Flot optique

Maintenant, on détecte l'objet dans la séquence d'images en utilisant l'analyse du flot optique, qui est un champ de déplacement pour une transition entre deux images. On fait l'hypothèse de persistance de l'intensité lumière dans le cas à deux instant proche et pour un mouvement faible. Donc il y a une équation comme ci-dessous :

$$E(x,y,t) \approx E(x+u, y+v, t+dt)$$

En développement de Tayler, on a :

$$E(x,y,t) \approx E(x,y,t) + E_x \cdot u + E_y \cdot v + E_k \cdot dt \Rightarrow E_x u + E_y v + E_k = 0$$

1. Calcul de E_x, E_y et E_k

Pour déterminer E_x, E_y et E_k, on les estime par la valeur moyenne des différences de ses 8 voisins comme ci-dessous (selon la Section 7 (P5) dans l'article):

$$E_x = \frac{1}{4} \cdot \{(E_{i,j+1,k} - E_{i,j,k}) + (E_{i+1,j+1,k} - E_{i+1,j,k}) + (E_{i,j+1,k+1} - E_{i,j,k+1}) + (E_{i+1,j+1,k+1} - E_{i+1,j,k+1})\}$$

$$E_y = \frac{1}{4} \cdot \{(E_{i+1,j,k} - E_{i,j,k}) + (E_{i+1,j+1,k} - E_{i,j+1,k}) + (E_{i+1,j,k+1} - E_{i,j,k+1}) + (E_{i+1,j+1,k+1} - E_{i,j+1,k+1})\}$$

$$E_k = \frac{1}{4} \cdot \{(E_{i,j,k+1} - E_{i,j,k}) + (E_{i+1,j,k+1} - E_{i+1,j,k}) + (E_{i,j+1,k+1} - E_{i,j+1,k}) + (E_{i+1,j+1,k+1} - E_{i+1,j+1,k})\}$$

Remarque :

E_x est la moyenne de la différence sur la direction "j";

E_y est la moyenne de la différence sur la direction "i";

E_k est la moyenne de la différence sur la direction "k";

Code correspondant :

```
EiJk=[Eijk(:,2:col,:),Eijk(:,col,:)]; %deplacement du cube (3D) pour la utilisation suivante de E(i,j+1,k)
EIJk=[EiJk(2:lin,:,:);EiJk(lin,:,:)]; %E(i+1,j+1,k)
EIjk=[Eijk(2:lin,:,:);Eijk(lin,:,:)]; %E(i+1,j,k)
EijK=cat(3,Eijk(:,:,2:N),Eijk(:,:,N));%E(i,j,k+1)
EiJK=[EijK(:,2:col,:),EijK(:,col,:)]; %E(i,j+1,k+1)
EIJK=[EiJK(2:lin,:,:);EiJK(lin,:,:)]; %E(i+1,j+1,k+1)
EIjK=[EijK(2:lin,:,:);EijK(lin,:,:)]; %E(i+1,j,k+1)

Ex=0.25*(EiJk-Eijk+EIJk-EIjk+EiJK-EijK+EIJK-EIjK); %approximation de Ex
Ey=0.25*(EIjk-Eijk+EIJk-EiJk+EIjK-EijK+EIJK-EiJK); %approximation de Ey
Ek=0.25*(EijK-Eijk+EIjK-EIjk+EiJK-EiJk+EIJK-EIJk); %approximation de Ek
```

2. Schéma itératif

Il faudrait chercher le champ inconnu dans une équation globale pour résoudre des difficultés dans les zones homogènes, et dans une grosse minimisation qui pénalise les écarts par rapport à l'équation du flot optique. Selon la proposition de Horn-Schunck dans l'article, on va minimiser la fonction suivante :

$$E(u,v)=\min\iint(E_x u+E_y v+E_k)^2+\alpha^2({u_x}^2+{u_y}^2+{v_x}^2+{v_y}^2)dxdy$$

Les équations d'Euler-Lagrangé donnent :

$$\begin{cases}(E_x u+E_y v+E_k)E_x+\alpha^2\nabla^2 u=0\\(E_x u+E_y v+E_k)E_y+\alpha^2\nabla^2 v=0\end{cases}$$

On peut montre que $\begin{cases}\nabla^2 u=u_{xx}+u_{yy}\approx u-\tilde{u}\\\nabla^2 v=v_{xx}+v_{yy}\approx v-\tilde{v}\end{cases}$, où

$$\begin{cases}\tilde{u}_{i,j,k}=\frac{1}{6}[u_{i-1,j,k}+u_{i,j+1,k}+u_{i+1,j,k}+u_{i,j-1,k}]+\frac{1}{12}[u_{i-1,j-1,k}+u_{i-1,j+1,k}+u_{i+1,j+1,k}+u_{i+1,j-1,k}]\\\tilde{v}_{i,j,k}=\frac{1}{6}[v_{i-1,j,k}+v_{i,j+1,k}+v_{i+1,j,k}+v_{i,j-1,k}]+\frac{1}{12}[v_{i-1,j-1,k}+v_{i-1,j+1,k}+v_{i+1,j+1,k}+v_{i+1,j-1,k}]\end{cases}$$

sont calculés par les voisins pour chaque frame (P6 dans l'article).

Le calcul itératif est fait par la méthode Gauss-Siedel (P9 dans l'article):

$$\begin{cases}u^n=\tilde{u}^{n-1}-E_x(E_x\cdot\tilde{u}^{n-1}+E_y\cdot\tilde{v}^{n-1}+E_k)/(\alpha^2+{E_x}^2+{E_y}^2)\\v^n=\tilde{v}^{n-1}-E_y(E_x\cdot\tilde{u}^{n-1}+E_y\cdot\tilde{v}^{n-1}+E_k)/(\alpha^2+{E_x}^2+{E_y}^2)\end{cases}$$

Remarque :

Ici, le paramètre α^2 ne joue un rôle important que dans les zones où le gradient de luminosité est faible, ce qui empêche tout ajustement aléatoire du mouvement occasionnée par le bruit dans les dérivées estimées. Ce paramètre doit être égal au bruit attendu dans l'estimation de ${E_x}^2+{E_y}^2$.

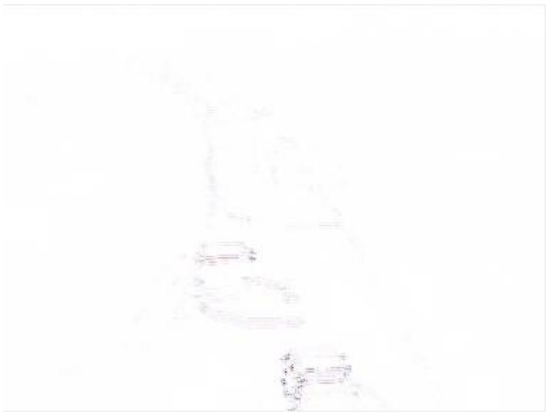
Code correspondant :

```
for frame=1:N
    Uk=zeros([lin,col]); %initialisation
    Vk=zeros([lin,col]);
    for i=1:1
        Ui_j=[Uk(1,:);Uk(1:lin-1,:)]; %deplacement de matrice pour la utilisation suivante de U(i-1,j,k)
        UiJ=[Uk(:,2:col),Uk(:,col)];       %U(i,j+1,k)
        UIj=[Uk(2:lin,:);Uk(lin,:)];       %U(i+1,j,k)
        Uij_=[Uk(:,1),Uk(:,1:col-1)];      %U(i,j-1,k)
        Ui_j_=[Uij_(1,:);Uij_(1:lin-1,:)]; %U(i-1,j-1,k)
        Ui_J=[UiJ(1,:);UiJ(1:lin-1,:)];    %U(i-1,j+1,k)
        UIJ=[UiJ(2:lin,:);UiJ(lin,:)];     %U(i+1,j+1,k)
        UIj_=[Uij_(2:lin,:);Uij_(lin,:)];  %U(i+1,j+1,k)

        Um=(1/6)*(Ui_j+UiJ+UIj+Uij_)+(1/12)*(Ui_j_+Ui_J+UIJ+UIj_); %calcul de la moyenne de "U" locale

        Vi_j=[Vk(1,:);Vk(1:lin-1,:)]; %meme operation sur V
        ViJ=[Vk(:,2:col),Vk(:,col)];
        VIj=[Vk(2:lin,:);Vk(lin,:)];
        Vij_=[Vk(:,1),Vk(:,1:col-1)];
        Vi_j_=[Vij_(1,:);Vij_(1:lin-1,:)];
        Vi_J=[ViJ(1,:);ViJ(1:lin-1,:)];
        VIJ=[ViJ(2:lin,:);ViJ(lin,:)];
        VIj_=[Vij_(2:lin,:);Vij_(lin,:)];

        Vm=(1/6)*(Vi_j+ViJ+VIj+Vij_)+(1/12)*(Vi_j_+Vi_J+VIJ+VIj_); %calcul de la moyenne de "U" locale

        Uk=Uk-Ex(:,:,frame).*((Ex(:,:,frame).*Um+Ey(:,:,frame).*Vm+Ek(:,:,frame))./(Ex(:,:,frame).^2+Ey(:,:,frame).^2+lambda));
        Vk=Vk-Ey(:,:,frame).*((Ex(:,:,frame).*Um+Ey(:,:,frame).*Vm+Ek(:,:,frame))./(Ex(:,:,frame).^2+Ey(:,:,frame).^2+lambda));
    end

    img=flowToColor(Uk,Vk);
    imshow(img,[]);
    pause(0.033)
end
```

①Dans le premier temps, on fait les tests sur le nombre d'itération.

frame 1 (itération=2, α^2=100)

frame 2 (itération=2, α^2=100)



frame 1 (itération=20, α^2=100)

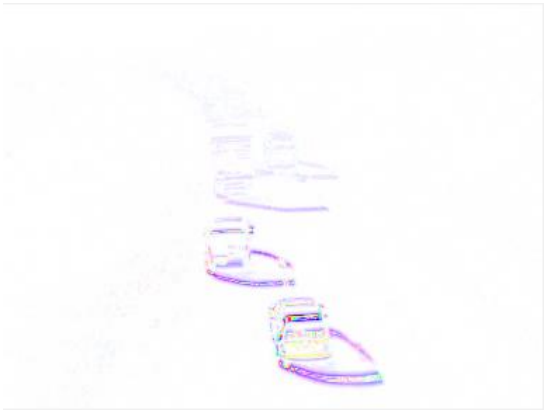
frame 2 (itération=20, α^2=100)

On peut voir que quand on fait le calcul itératif deux fois, le résultat est assez bon. Si le nombre d'itération est trop grand, la représentation n'est pas très évidente.

②Maintenant, on fixe le nombre d'itération à 2, et teste sur la valeur du paramètre "α^2".

frame 1 (α^2 = 1)

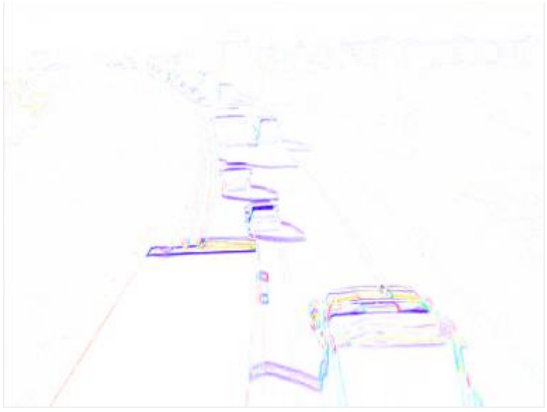
frame 2 (α^2 = 1)



frame 1 (α^2 = 100)

frame 2 (α^2 = 100)

frame 1 (α^2 = 10000)



frame 2 (α^2 = 10000)

Comme on peut voir dans les figures, si α^2 est petit, trop beaucoup de pixels seront

estimés comme dans le fond. Donc l'objet ne sera pas évident. C'est parce que α^2 ne joue un rôle important que dans les zones où le gradient de luminosité est faible, ce qui empêche tout ajustement aléatoire du mouvement occasionnée par le bruit dans les dérivées estimées. Qaund α^2 est grand, il est possible de confondre le bruit avec l'objet. Ici, le résultat du cas α^2 =100 et celui du cas α^2 =10000 ne sont pas très différente parce qu'on applique un filtre de Gauss au début.

IV. Discussion

Pendant ce TP, on a essayé plusieurs méthodes pour détecter l'objet en mouvement.

Dans un premier temps, on commence à partir de la classification des pixels. Comme on l'aspect du fond ne change pas, la disperison des valeurs du fond soit petit. Alors celle d'objet soit plus grand parce que l'objet est en mouvement. Ensuite, on optimiser cette méthode en utilisant l'algorithme $\sum\Delta$ discutée dans l'article.

Après, on utilise l'analyse du flot optique pour détecter l'objet en mouvement. Le problème de l'estimation du flot optique est un mal-posé problème comme le nombre de inconnues est supérieur que celui d'équations. La solution n'exista pas. Donc il faut trouver la solution au sens des moindres carrés. Ici on exécute le calcul itératif pour obtenir les champs de déplacement horizontaux et verticaux.