Asservissement visuel

Travaux pratiques

1 Rappels et compléments du cours

Les techniques d'asservissement visuel consistent à élaborer des lois de commande sur le mouvement d'une caméra afin de réaliser des tâches de positionnement ou de suivi d'objets mobiles. Les informations visuelles utilisées dans la boucle de commande, notées $\mathbf{x}$, doivent atteindre une valeur désirée $\mathbf{x}^*$ et le problème de la commande se ramène à générer les mouvements de la caméra permettant de minimiser $(\mathbf{x} - \mathbf{x}^*)$ dans l'image. Une telle loi de commande en boucle fermée est donnée par :

$$\mathbf{v}_c = -\lambda \, \mathbf{L}_\mathbf{x}^+ (\mathbf{x} - \mathbf{x}^*) \tag{1}$$

où :

- $\mathbf{v}_c = (v_x, v_y, v_z, \omega_x, \omega_y, \omega_z)$ est le torseur cinématique de la caméra ;
- λ est le gain qui règle la rapidité de convergence de la loi de commande ;
- $\mathbf{L}_\mathbf{x}$ est la matrice d'interaction associée à $\mathbf{x}$ (définie par $\dot{\mathbf{x}} = \mathbf{L}_\mathbf{x} \, \mathbf{v}_c$ et de dimension $k \times n$ où k est le nombre d'informations visuelles utilisées et n le nombre de degrés de liberté de la caméra) ;
- $\mathbf{L}_\mathbf{x}^+$ est la pseudo inverse de $\mathbf{L}_\mathbf{x}$ (de dimension $n \times k$) ;

On considèrera par la suite que la caméra a six degrés de liberté ($n = 6$).

2 Introduction

L'objectif de ce TP est de simuler plusieurs tâches d'asservissement visuel. Pour cela vous vous aiderez de la bibliothèque ViSP accessible depuis http ://www.irisa.fr/lagadic/visp. Cette librairie est diffusée en tant que logiciel libre sous license GPL par l'équipe Lagadic d'INRIA Rennes - Bretagne Atlantique.

Vous pourrez récupérer la trame du TP à l'adresse suivante :
`http://people.irisa.fr/Eric.Marchand/`

3 Réalisation d'un démonstrateur d'asservissement visuel

L'objectif de ce TP est de faire un démonstrateur permettant de simuler des tâches d'asservissement visuel. Votre travail sera de comparer (et de commenter) le comportement de lois de commande 2D et de lois de commande 3D.

Vous pourrez vous aidez de la trame de code qui vous est fournie et la compléter.

3.1 Changement de repère et projection

Étant donné les coordonnées d'un point ${}^{w}\mathbf{X}$ exprimé dans le repère de la scène, vous écrirez une fonction permettant de calculer la position de ce point dans l'image vue depuis une position de la caméra donnée par la matrice homogène ${}^{c}\mathbf{T}_{w}$ (qui désigne la position de la caméra par rapport à la scène $\mathcal{F}_w$).

Les coordonnées $\mathbf{X}$ d'un point 3D sont représentées par des vecteurs à trois composantes $\mathbf{X} = (X, Y, Z)^\top$ en utilisant le type `vpColVector`.

Question 1 *Compléter la fonction* `changeFrame(...)` *qui permet d'obtenir les coordonnées d'un point exprimer dans le repère $\mathcal{F}_a$ en fonction de ses coordonnées exprimées dans le repère $\mathcal{F}_b$ et la matrice de transformation rigide ${}^{a}\mathbf{T}_{b}$. Pour mémoire ${}^{a}\mathbf{T}_{b}$ est définie par :*

$$ {}^{a}\mathbf{T}_{b} = \begin{pmatrix} {}^{a}\mathbf{R}_{b} & {}^{a}\mathbf{t}_{b} \\ \mathbf{0} & 1 \end{pmatrix} $$

Les coordonnées $\mathbf{x}$ d'un point 2D sont représentées par des vecteurs à deux composantes $\mathbf{x} = (x, y)^\top$ en utilisant le type `vpColVector`.

Question 2 *Compléter la fonction* `project(...)` *qui permet d'obtenir les coordonnées d'un point 2D par projection perspective à partir des coordonnées d'un point 3D exprimées dans le repère de la caméra $\mathcal{F}_c$.*

3.2 Asservissement visuel sur un point 2D

L'idée est ici de compléter le code de la fonction `tp2DVisualServoingOnePoint(...)` qui vise à mettre en œuvre un asservissement visuel sur un point. Une trame vous est donnée, les fonctions d'affichage des courbes et des images simulé aussi. mais la loi de commande est entièrement à écrire.

Conditions experimentales

1. L'objectif est de voir le point au centre de l'image soit $\mathbf{x}^* = (0, 0)^\top$.
2. Les coordonnées ${}^{w}\mathbf{X}$ du point qui nous intéresse dans le repère monde $\mathcal{F}_w$ sont données par : ${}^{w}\mathbf{X} = (0.5, 0.2, -0.5)^\top$
3. On placera initialement la camera en

$$ {}^{c}\mathbf{T}_{w} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix} $$

Cette matrice homogène peut se créer grace à la commande :

```
vpHomogeneousMatrix cTw (0,0,1,  0,0,0) ;
```

`cTw` est une matrice 4×4.

Question 3 *Donner la taille et la forme de la matrice d'interaction* $\mathbf{L}_\mathbf{x}$ *associée à un point 2D. Compléter la fonction* `computeInteractionMatrix(...)` *en conséquence.*

Question 4 *Mettre en œuvre la loi de commande permettant de déplacer la caméra afin d'observer le point à la position* $\mathbf{x}^* = (0,0)^\top$.

Question 5 *Vous analyserez les résultats obtenus en particulier la norme globale de l'erreur* $\| \mathbf{x} - \mathbf{x}^* \|$, *l'erreur pour chaque primitive* $\mathbf{x}_i - \mathbf{x}_i^*$, *la vitesse* $\mathbf{v}$ *de la caméra, la position* ${}^c\mathbf{T}_w$ *de la caméra, la position* $\mathbf{x}$ *du point dans l'image.*

3.3 Asservissement visuel sur quatre points 2D

Conditions experimentales

1. Nous placerons quatre points en carré aux positions suivante dans le repère $\mathcal{S}$: ${}^w\mathbf{X} = ((-L, L, 0), (L, L, 0), (L, -L, 0), (-L, -L, 0))$, Nous choisirons $L = 0.5m$.
2. On placera initialement la camera en

$$ {}^{c^*}\mathbf{T}_w = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix} $$

3. les positions initiales de la caméra pour lesquelles vous mènerez les tests seront données par :

```
vpHomogeneousMatrix cTw (0, 0, 1.3,  0, 0, 0) ;
vpHomogeneousMatrix cTw (-0.2, -0.1, 1.3,
                         vpMath::rad(10), vpMath::rad(20), vpMath::rad(30) ) ;
```

Question 6 *Modifier la loi de commande de la section précédente (à mettre dans la fonction* `tp2DVisualServoingQuatrePoints(...)` *pour réaliser cette tâche.*

Vous analyserez les résultats obtenus en particulier la norme globale de l'erreur $\| \mathbf{x} - \mathbf{x}^* \|$, l'erreur pour chaque primitive $\mathbf{x}_i - \mathbf{x}_i^*$, la vitesse $\mathbf{v}$ de la caméra, la position ${}^c\mathbf{T}_w$ de la caméra, la position $\mathbf{x}$ du point dans l'image. En particulier :

Question 7 *Comparer plusieurs lois de commande : avec la matrice d'interaction évaluée à la position désirée* $\mathbf{L}_{\mathbf{x}|\mathbf{x}=\mathbf{x}^*}$ *; avec la matrice d'interaction évaluée à la position courante* $\mathbf{L}_{\mathbf{x}|\mathbf{x}=\mathbf{x}(t)}$. *Commenter les résultats.*

Question 8 *Vous ferrez aussi les tests pour les deux positions suivantes de la caméra correspondants à une simple rotation de 90 degrés (sur l'axe z) par rapport à la position désirée. Que constatez vous. Même question avec un rotation de 180 degrés. Commentez.*

```
vpHomogeneousMatrix cTw (0, 0, 1,  0, 0, vpMath::rad(90)) ;
vpHomogeneousMatrix cTw (0, 0, 1,  0, 0, vpMath::rad(180)) ;
```

3.4 Asservissement visuel 3D

L'objectif est de mettre en oeuvre une loi de commande 3D. Pour memoire celle ci vise à minimiser l'erreur entre la position courante ${}^{c}\mathbf{T}_{w}$ et désiré ${}^{c^*}\mathbf{T}_{w}$ de la caméra.

Question 9 *Comment calculer l'erreur* ${}^{c^*}\mathbf{T}_{c}$ *et le vecteur d'erreur associée ? Quelle est la matrice d'interaction associée ?*

Vous implementerez les fonctions `computeError3D` et `computeInteractionMatrix3D`.
Question 10 *Modifier la loi de commande de la section précédente (à mettre dans la fonction* `tp3DVisualServoing(...)` *pour réaliser cette tâche.*

Nous reprendrons les mêmes conditions expérimentales que dans la section précédente et vous testerez donc les **quatre** positions initiales précédentes. Vous analyserez les résultats obtenus en particulier la norme globale de l'erreur, l'erreur pour chaque primitive, la vitesse $\mathbf{v}$ de la caméra, la position et la trajectoire ${}^{c}\mathbf{T}_{w}$ de la caméra.

Question 11 *Que constatez vous ? Pourquoi ?*

3.5 Poursuite

On supposera dans cette partie que la cible est maintenant en mouvement.

Question 12 *Réecrivez les fonctions décrites dans les questions 3.3 en supposant que l'objet est animé d'un mouvement de translation de 1cm/s suivant l'axe x dans le repère $\mathcal{F}_w$.*

Question 13 *Modifiez le programme permettant de simuler les lois de commande 2D en conséquence. Lancez une simulation. Que constatez vous ? (si vous ne constatez rien, augmentez la vitesse de la cible...)*

Question 14 *Proposez une méthode pour résoudre les éventuels problèmes observés.*