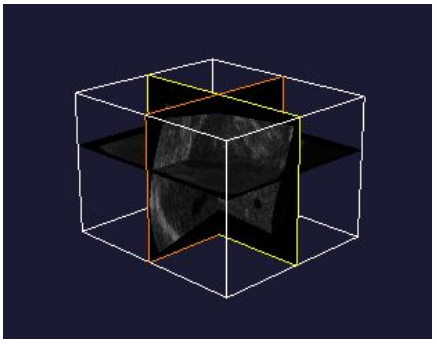
Clément Glédel
Quentin Dugne

TP Medical robotics : ultrasound dense visual servoing

L'objectif de ce tp est de simuler l'asservissement visuel automatique d'une sonde ultrasonore 2D. L'asservissement sera réalisé en utilisant directement les intensités de pixel présentes dans les images recueillies par la sonde.

Pour cela nous avons tout d'abord simuler le mouvement du patient en appliquant au volume 3D, représentant le corps du patient, un mouvement sinusoïdal sur ses 6 degrés de libertés. La période du mouvement sera fixée à 5 secondes, l'amplitude de translation à 0.005 meters/s et l'amplitude de rotation à 5 deg/s.

Images à la position initiale de l'objet :


représentation 3D initiale

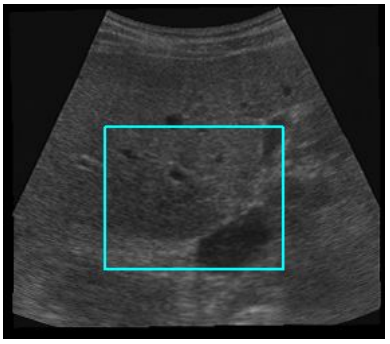
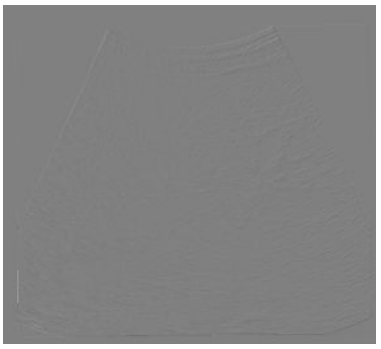
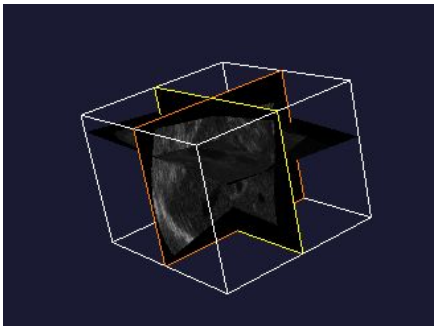


image courante


différence courante/désirée

Images de fin de mouvement sinusoïdal de l'objet :


représentation 3D fin de mvt

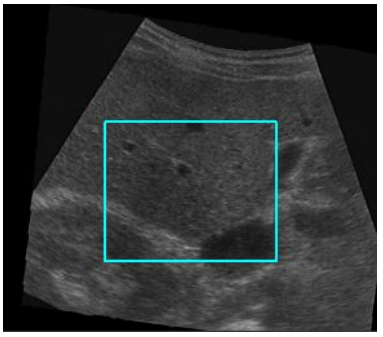
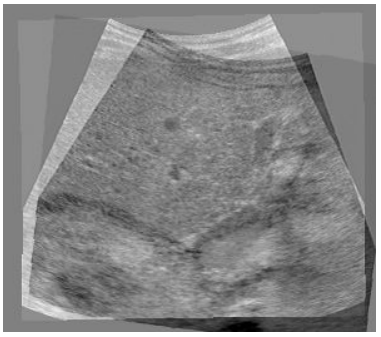
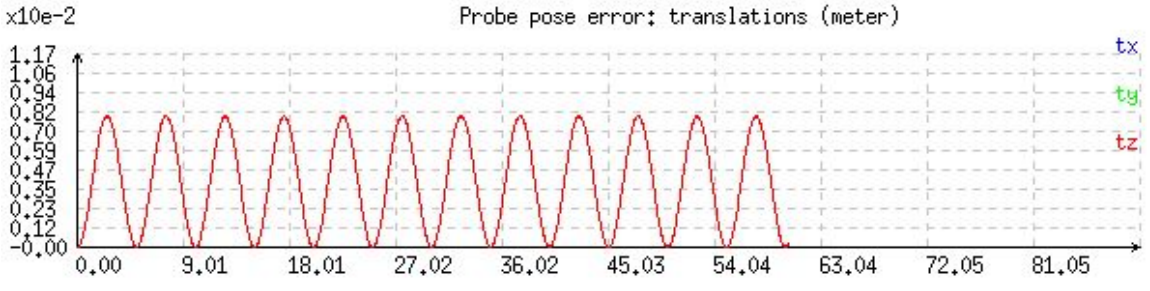


image courante


différence courante/désirée



Tout d'abord nous pouvons nous apercevoir que notre simulation de mouvement sinusoïdal fonctionne correctement. Comme notre plan de coupe (sonde 2D) reste fixe il est normale d'obtenir une erreur sous forme de sinusoïde grandissante puis diminuante que ce soit en translation ou en rotation sachant que lorsque la courbe d'erreur arrive à 0 le volume 3D est revenu à sa position initiale.

De plus, nous pouvons constater que l'image de différence entre l'image courante et l'image désirée permet de vérifier le suivi de la sonde au mouvement sachant que nous prenons l'image initiale comme image désirée. Dans notre cas, l'image de différence nous montre bien que la sonde ne suit pas le mouvement. Il faudra donc faire en sorte d'obtenir l'image de différence obtenue à la position initiale afin de valider notre asservissement.

Par la suite nous voulons, malgré le déplacement du patient, que l'image 2D courante soit raccord avec celle désirée. L'utilisateur du programme commence donc par définir une zone de pixel détourée par un cadre cyan. Celui-ci délimite les pixels à utiliser pour effectuer notre asservissement. Comme expliqué auparavant, l'asservissement visuel sera réalisé en utilisant l'intensité des pixels. Sachant cela et sachant le fait que l'asservissement visuel fait appel à une matrice d'interaction, nous utiliserons le gradient 3D de l'image initiale pour calculer celle-ci.

L'image initiale étant en 2D, il est facile d'obtenir le gradient 2D suivant les axes x et y. Cependant afin de calculer le gradient de la zone désirée suivant les 3 dimensions, nous utilisons un filtre dérivatif 3D sur 3 frames différentes (précédente, courante, future). Cela nous permettra d'obtenir le gradient suivant la profondeur Z.

Ce processus nous mène donc à la matrice d'interaction suivante :

$$L_s = [\, L_{I_{1,1}} \quad . \quad . \quad . \quad L_{I_{M,N}}]^T$$

avec $L_{I_{u,v}} = [\, \nabla I_x \quad \nabla I_y \quad \nabla I_z \quad y\nabla I_z \quad -x\nabla I_z \quad x\nabla I_y \quad -y\nabla I_x]$

Cette matrice d'interaction est calculée qu'une seule fois à la position initiale du volume car dans une tâche de compensation de mouvement, elle sera la même à n'importe quel instant t puisque nous corrigeons la position de la sonde pour obtenir l'image initiale (désirée) comme image courante.

Ainsi nous pouvons coder la loi de contrôle utilisant la matrice d'interaction précédente de la manière suivante :

vp = -lambda * Ls+ *(s(t)-sd)

avec **vp** la vitesse de la sonde, **lambda** un gain de contrôle, **Ls+** la pseudo inverse de notre matrice d'interaction. **s(t)** et **sd** représentent un vecteur contenant les pixels dans le cadre cyan de l'image courante et désirée. Par conséquent **s(t) - s** correspond à l'erreur visuelle.

Une fois la loi de commande implémentée, nous pouvons donc exécuter notre programme en fixant **lambda** à 0.4 afin de vérifier le bon fonctionnement de notre asservissement visuel. Ainsi nous obtenons les résultats suivant :

Images à la position initiale de l'objet :

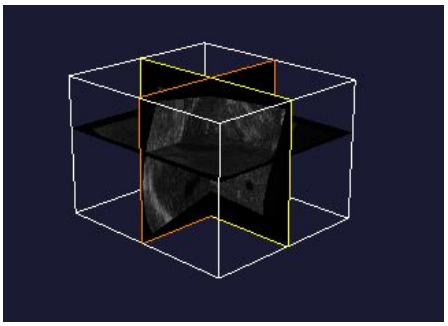
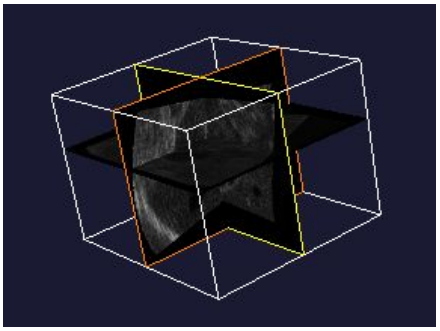

représentation 3D initiale


image courante


différence courante/désirée

Images à une position intermédiaire de l'objet :


représentation 3D interm.

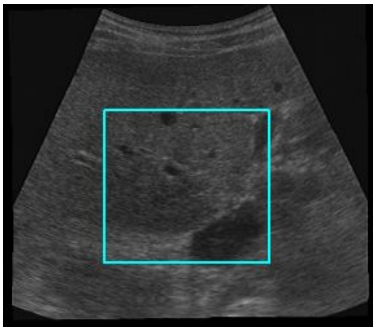
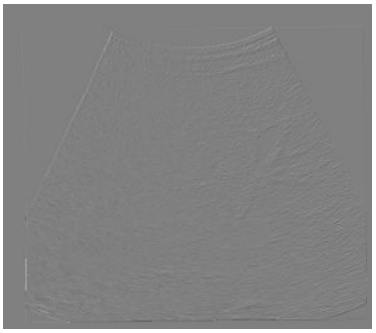
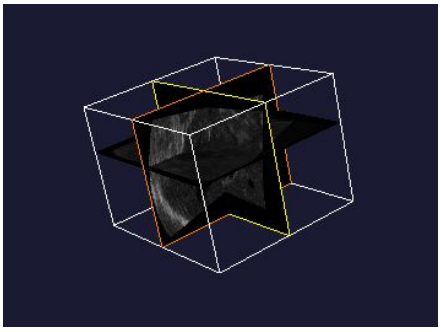


image courante


différence courante/désirée

Images de fin de mouvement sinusoïdal de l'objet :


représentation 3D fin de mvt

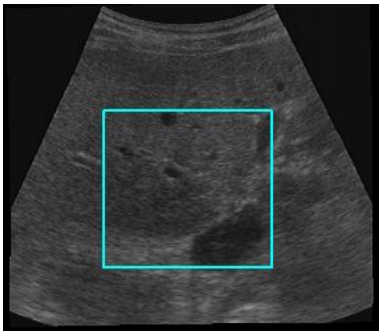
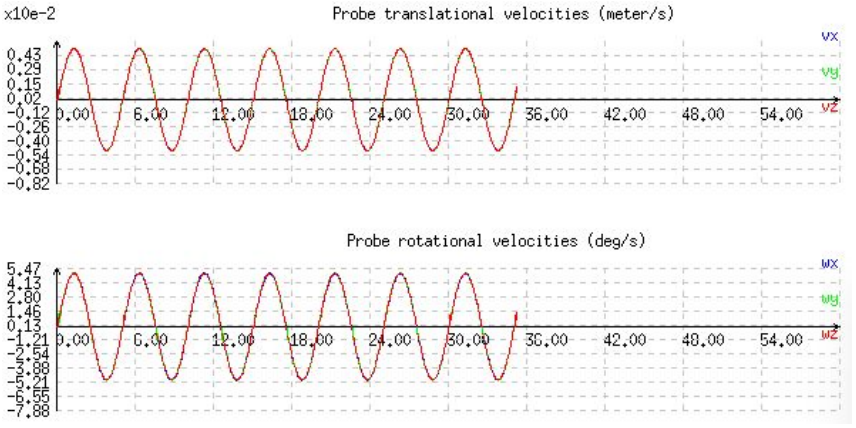


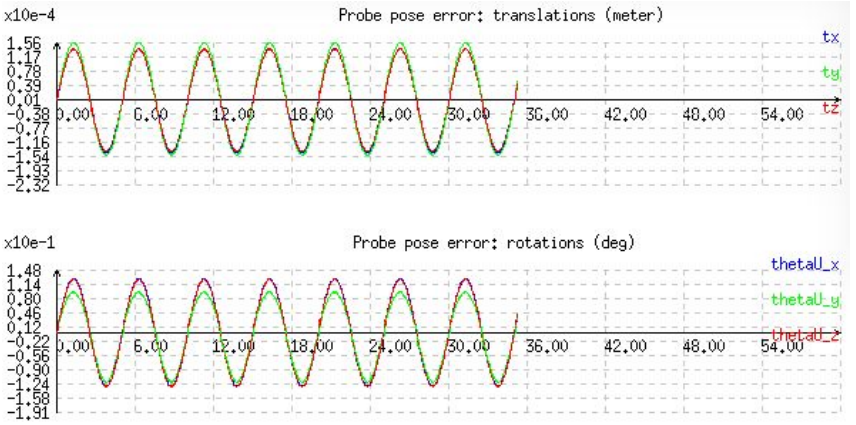
image courante


différence courante/désirée

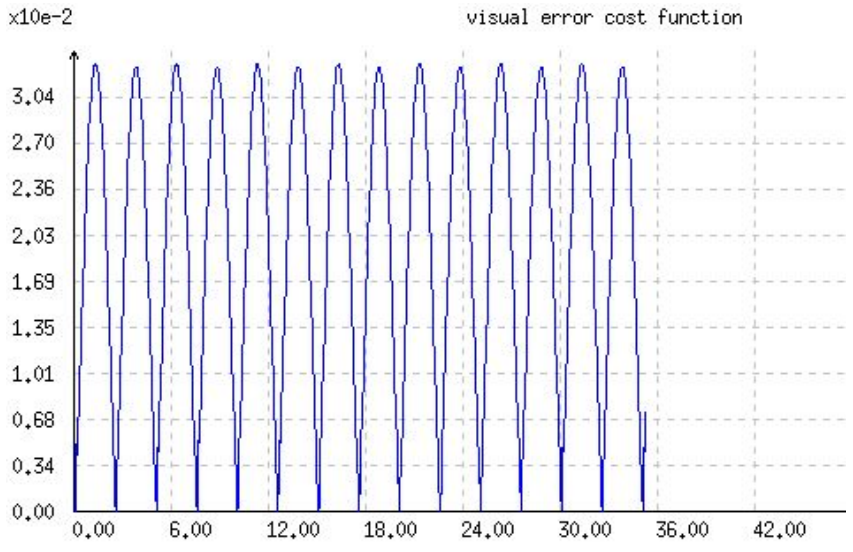
Courbes :



Vitesse de la sonde



Erreur de position de la sonde suivant les 6 degrés de liberté

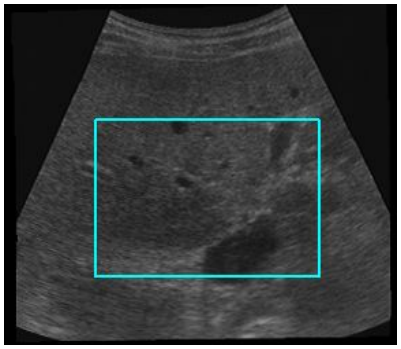
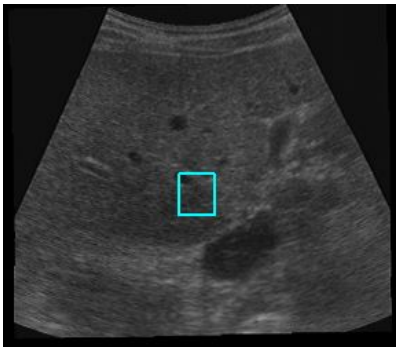
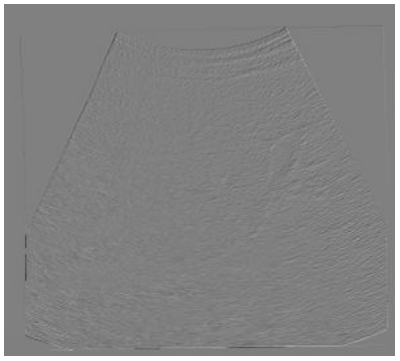
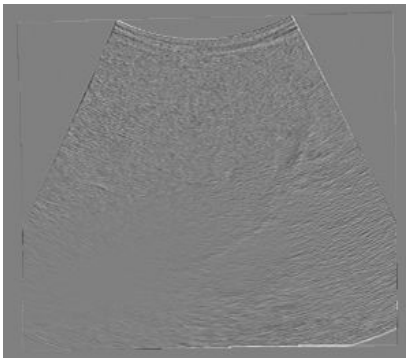
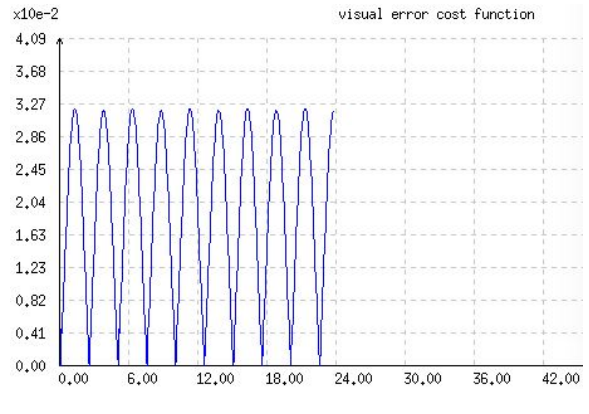
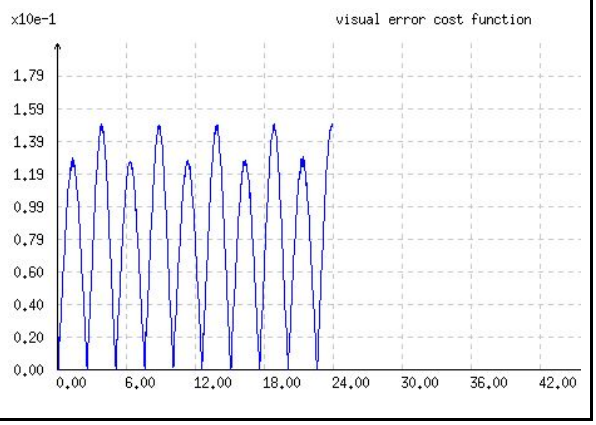


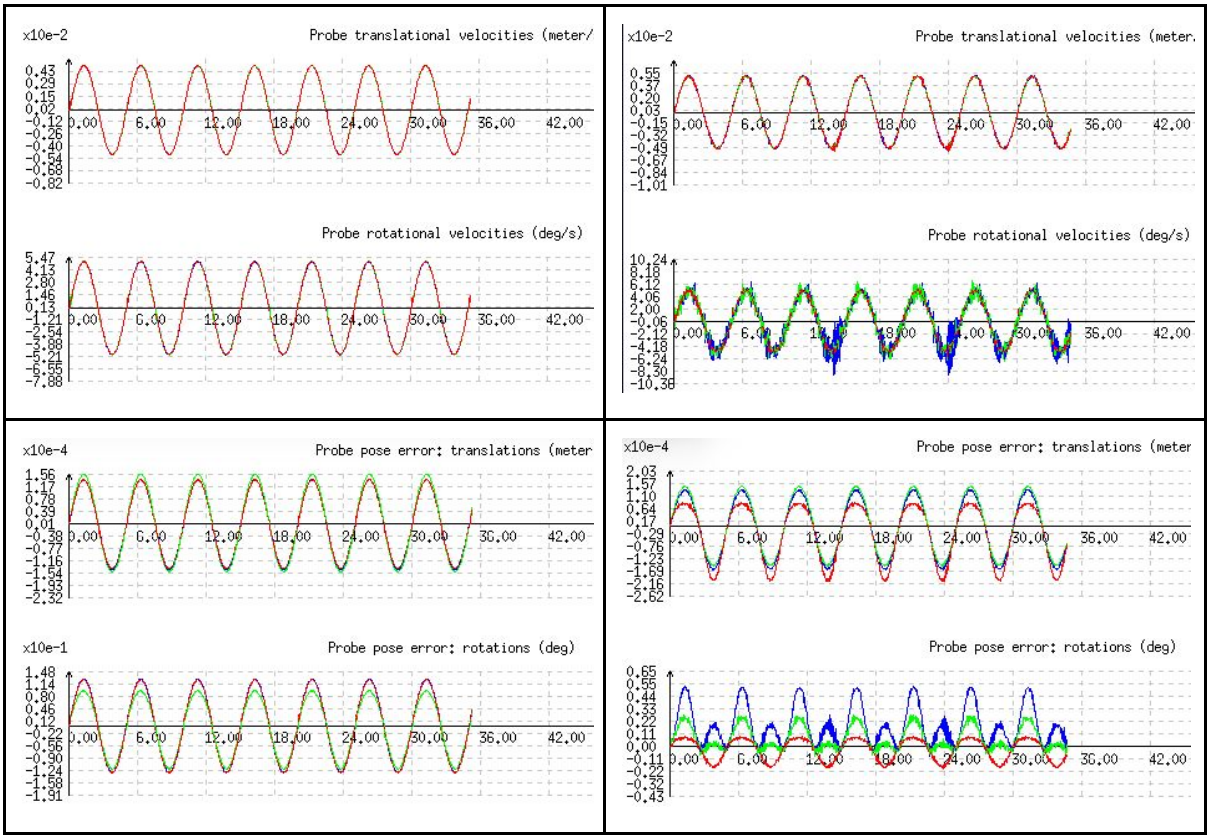
Coût visuel normalisé

Suite à nos résultats nous pouvons dans un premier temps constater que notre sonde suit le déplacement du volume correctement. Ceci est notamment visible sur les images obtenues à la position initiales ainsi qu'à la position de fin de mouvement. En effet, leur image de différence ne présente quasi aucune différence d'intensité. Cependant nous pouvons voir sur les images récupérées dans une position intermédiaire (en particulier l'image de différence) que lors du déplacement l'erreur est plus importante. Plus la vitesse de déplacement de l'objet augmente plus la sonde a du mal à compenser le mouvement

correctement. En effet, en superposant les courbes de *vitesse de la sonde* et de l'*erreur de position de la sonde suivant les 6 degrés de liberté,* nous pouvons voir les courbes d'erreur augmenter en concordance avec la vitesse. La courbe de *coût visuel normalisé* permet de s'en apercevoir clairement.

Ensuite nous avons testé notre programme avec plusieurs grandeurs de ROI (cadre cyan), donc tester notre asservissement avec plus ou moins de pixels. Nous avons organisé sous forme de tableau les résultats avec une première colonne correspondant à un grand ROI et la deuxième un petit ROI :

<table>
<tr>
<td>  </td>
<td>  </td>
</tr>
<tr>
<td>  </td>
<td>  </td>
</tr>
<tr>
<td> 
</td>
<td> 
</td>
</tr>
</table>



Les résultats obtenues nous montre l'importance d'utiliser un ROI assez grand. En effet, nous pouvons nous apercevoir que lorsque l'on utilise un ROI trop petit l'asservissement utilise que peu de pixel et donc on obtient une vitesse de la sonde très instable en particulier au niveau de la rotation dans notre cas. L'image de différence nous le confirme puisque nous pouvons voir que la différence dans certaines zones sont plus important que dans d'autres, ce qui doit être en partie dû à une mauvaise rotation du plan 2D de la sonde.

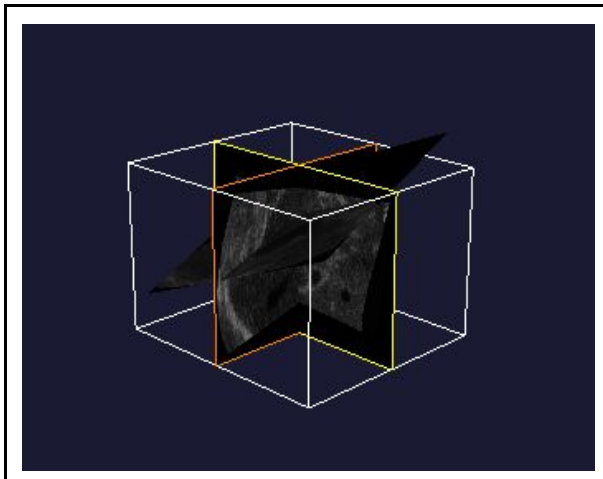
Nous remarquons aussi que la courbe d'erreur visuelle varie en fonction du sens du déplacement et détient des valeurs moins importante que lorsque l'on utilise un grand ROI, cependant, même si l'erreur sur les pixels présents dans le ROI est faible, ce n'est pas ce qui fait la qualité du suivi.

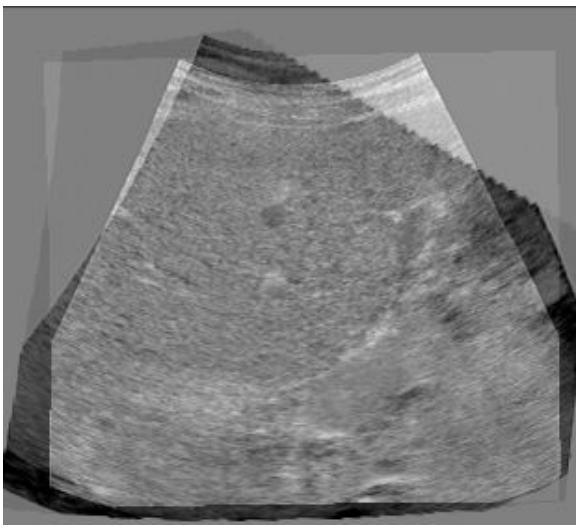
Dans notre cas, la rotation est plus instable que la translation car par rapport au centre de l'image, une mauvaise rotation aura peu d'impact sur le centre de l'image mais plutôt sur les bords. Il se trouve que notre cas utilise un ROI au centre de l'image, nous pouvons donc bien imaginer qu'en prenant un petit ROI sur un bord les résultats seront différents. De plus, notre ROI doit être positionné sur une zone relativement stable car la vitesse de translation semble être plutôt correcte.

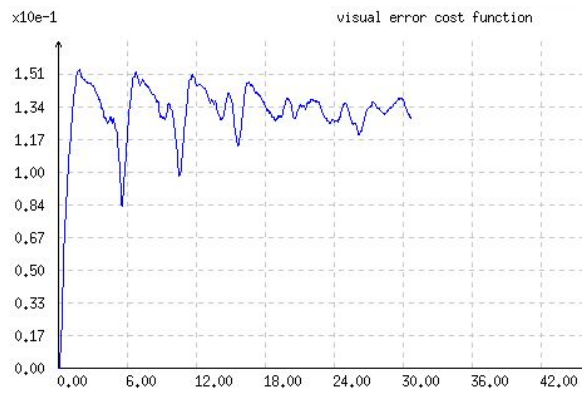
Après avoir regardé l'influence de la taille du ROI, nous souhaitons à présent tester différentes valeur de **lambda** dans notre loi de commande afin de constater le rôle de ce paramètre dans l'asservissement visuel.

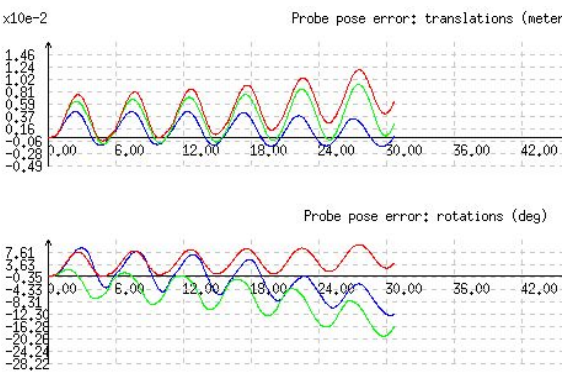
Pour cela nous avons testé plusieurs valeurs de **lambda** et avons obtenu les résultats suivant :

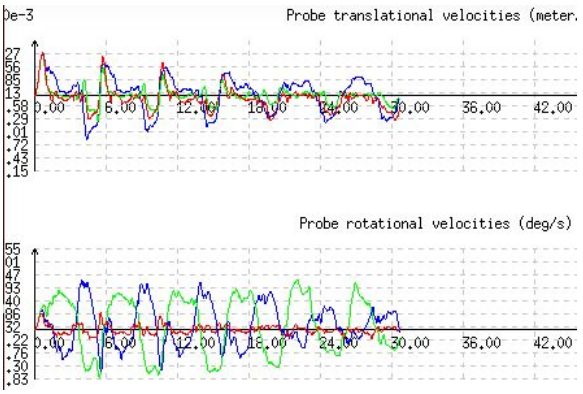
lambda = 0.1 :





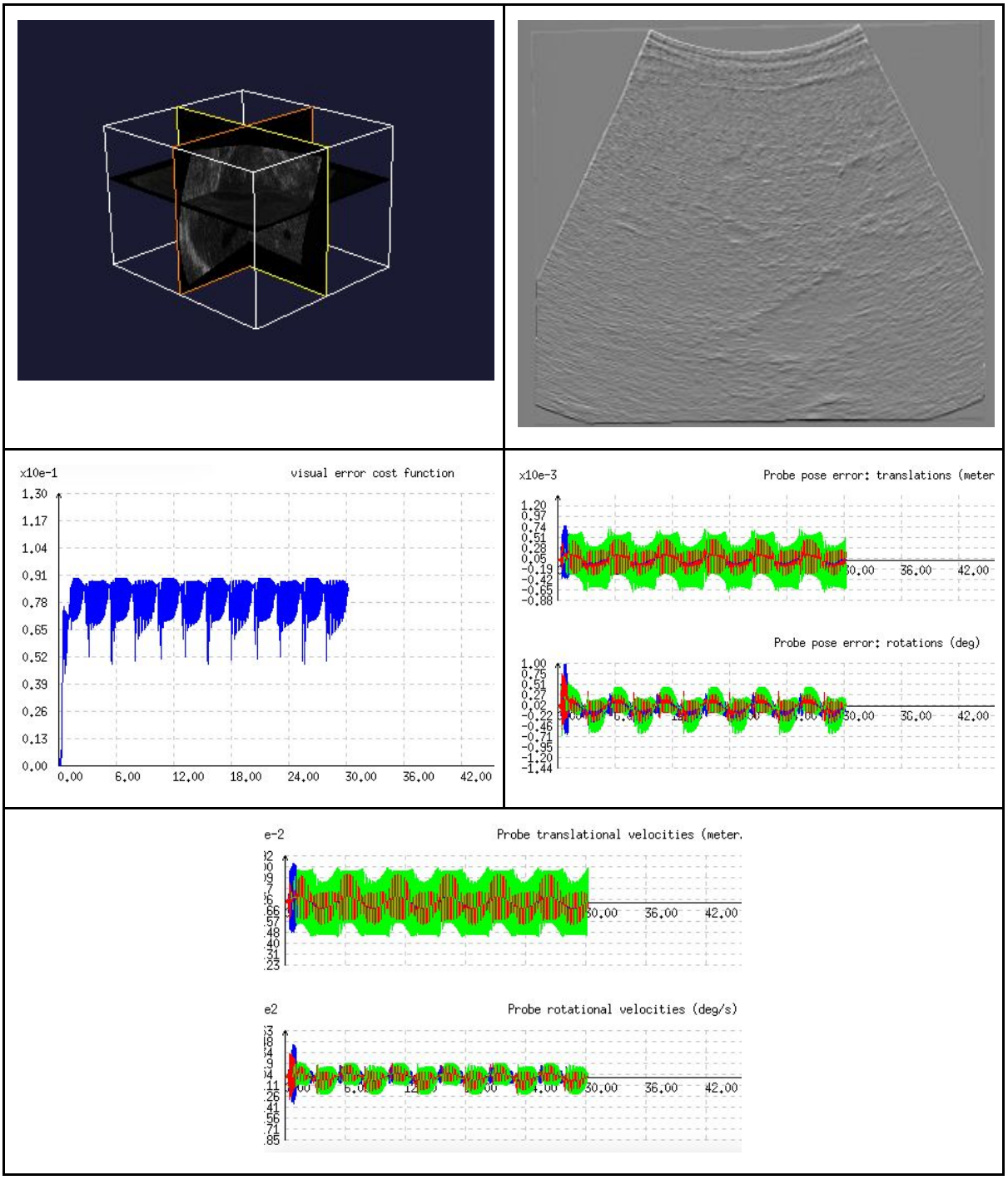






Lorsque l'on prend **lambda** égale à 0.1 (divisé par 4), nous pouvons rapidement constater que notre loi de commande ne corrige pas assez la position de notre sonde 2D et qu'elle finit par se perdre complètement (visible sur la représentation 3D et la différence courante/désirée). De même, les courbes d'erreurs et de la vitesse de la sonde ne sont pas du tout périodiques donc cela signifie que la sonde varie constamment de vitesse suivant tous ses degrés car elle ne sait pas où aller.

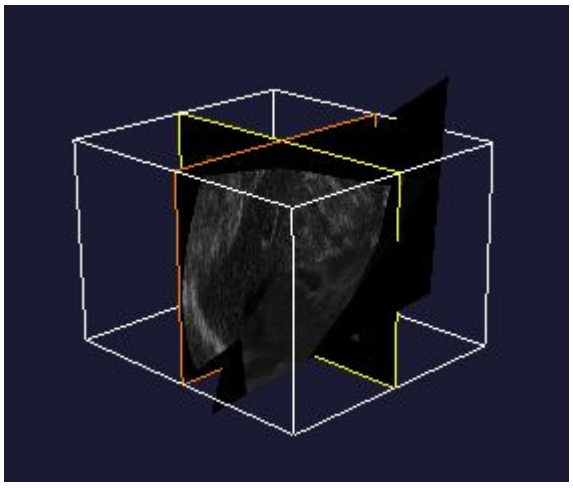
lambda = 0.8 :

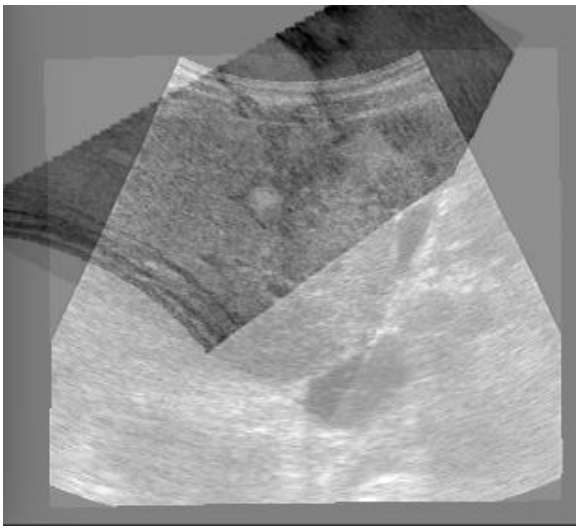


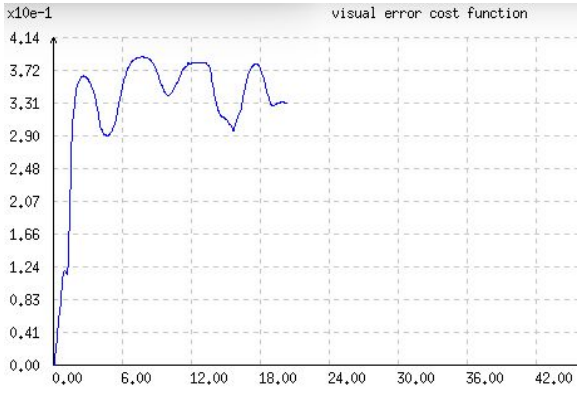
Lorsque l'on prend **lambda** égale à 0.8 (multiplié par 2), nous constatons que dans ce cas là, la sonde arrive à suivre le mouvement. Le problème (pas très visible sur la représentation 3D et la différence courante/désirée) est que l'on corrige la position de la sonde pas assez faiblement et donc se traduit par des tremblotements de la sonde. En effet, sur le graphique de vitesse nous pouvons voir que tous les degrés de liberté passe très rapidement du positif au négatif. La sonde corrige trop et donc ensuite doit corriger dans l'autre sens et ainsi de suite.

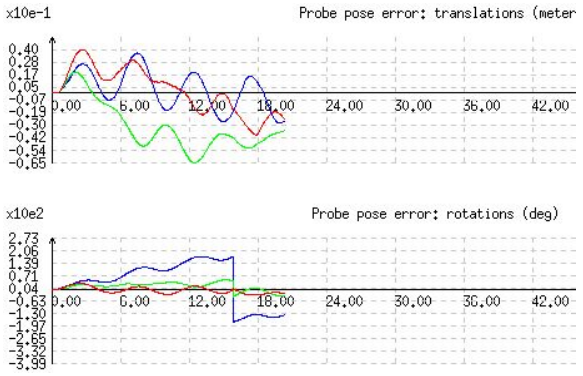
Après de nombreux essaie en faisant varier la valeur de **lambda** il semblerait que 0.5 soit le maximum fournissant une compensation "correct" de notre mouvement, s'il l'on souhaite aucun tremblement 0.46 semble être le maximum possible.

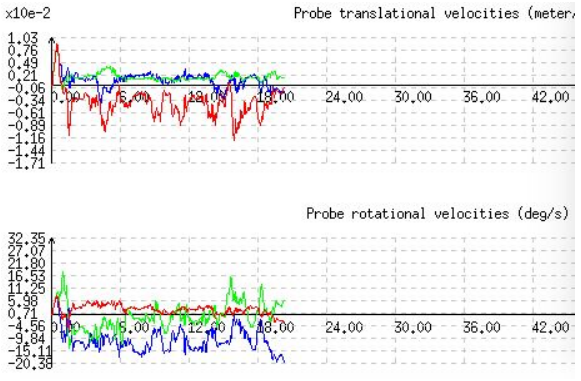
Enfin nous allons simuler un mouvement physiologique rapide du client en multipliant par 4 les amplitudes de translation et de rotation de la sinusoïde appliqué au volume.











Après avoir appliqué la multiplication de l'amplitude, nous nous apercevons que l'on se retrouve dans le cas où le **lambda** est trop faible et que la loi de commande ne compense pas assez la position de la sonde par conséquent la sonde fini par se perdre complètement. De la même façon que précédemment, nous pouvons le voir sur les images et graphiques ci-dessus, la courbe d'erreur ainsi que la courbe de coût ne sont pas du tout périodiques et ne représentent plus grand chose.

Afin d'essayer de compenser le mouvement nous avons donc essayé d'augmenter le lambda de nombreuses valeurs différentes mais n'avons pas réussi à compenser le mouvement en modifiant ce paramètre. Le problème vient en grande partie à cause du type

de mouvement que l'on applique au volume. En effet, un mouvement sinusoïdal de trop grande amplitude dispose d'une trop grande variance de vitesse (vitesse démarrant lentement, rapidement au niveau d'un sommet de la sinusoïde, puis re-lentement). Notre loi de commande configurée tel quel ne peut pas gérer trop de variance de vitesse.

De plus, nous pouvons ajouter que, dans notre cas, même s'il est possible de trouver un **lambda** permettant de compenser la plus grande vitesse du sinus, la sonde serait très instable lors de faible vitesse et risquerait de se perdre avant même d'arriver à la grande vitesse compensée.

Conclusion :

En conclusion, nous avons pu implémenter dans ce tp un système d'asservissement visuel de compensation de mouvement utilisant seulement les intensités de pixel d'une zone prédéfinie. Nous avons fait varier différents paramètres tels que la taille de la zone de pixel ou encore le lambda de notre loi de commande et avons pu constater leur influence sur celle-ci. Notre implémentation de la loi de commande n'est pas la plus complète et peut être amélioré afin de compenser une vitesse sinusoïdale de grande amplitude en ajoutant un autre terme que nous n'avons pas eu le temps d'implémenter.