PFE ENSTA-Bretagne
Mémoire de Master SDS 2024

DVS-Straight : Impact du masquage sur la performance des asservissements visuels directs

---

Auteur : Gwendal Crequer,
fise24 spé. ROB at ENSTA Bretagne

Encadrants : Antoine André
Guillaume Caron
permanent researchers at CNRS-JRL

Résumé

L'asservissement visuel photométrique est une technique d'asservissement qui utilise l'intensité de chaque pixel d'une image pour calculer une commande, sans passer par des transformations variées ou du tracking de features. L'objectif est de minimiser l'erreur entre 2 images, en la faisant répondre à une équation différentielle du 1^{er} ordre. L'utilisation de caméras 360° a permis d'augmenter considérablement le domaine de convergence de ces méthodes, ainsi que de prendre en compte des environnements dans leur intégralité. Cependant, ce nouveau champs de vision implique de voir le robot lui-même, dont la forme change en fonction de la configuration, qui peut alors impacter le calcul d'erreur et dégrader les performances de l'asservissement. Nous proposons une méthode pour masquer des zones de l'image qui ne sont pas pertinentes pour le calcul de l'erreur, et étudions l'impact de ce masquage sur les-dites performances. Nous avons testé notre méthode sur un manipulateur mobile équipé d'un caméra dual-fisheye, et seront rapidement en mesure de montrer l'amélioration des performances.

Table des matières

1 Introduction **5**
1.1 Le Joint Robotics Laboratory 5
1.2 Le projet DVS-Straight 5
1.3 Objectif du stage et inscription dans le projet 5

2 De l'asservissement visuel au MPPVS **6**
2.1 Les 2 philosophies de l'asservissement visuel 6
2.2 L'asservissement visuel 6
2.3 Le PVS, le PGMVS et le MPPVS 7

3 Estimation du masque **9**
3.1 Jumeau numérique 9
3.2 Environnement graphique 10
3.3 Alignement du masque avec l'image réelle 10

4 Stratégies de masquage **12**

5 Expériences et Résultats **13**
5.1 Critères de performance et expériences prévues 13
5.2 Petits déplacements 14
5.3 Grands déplacements 15
5.4 Pose identique, configuration articulaire différente 16

6 Axes de travail actuel et perspectives d'amélioration **18**

Annexes **I**

Liste des figures **II**

Bibliographie **III**

Glossaire

JRL Joint Robotics Laboratory

CNRS Centre National de la Recherche Scientifique

AIST National Institute of Advanced Industrial Science and Technology

SLAM Simultaneous Localization and Mapping

VS Visual Servoing

IBVS Image Based Visual Servoing

IVS Indirect Visual Servoing

DVS Direct Visual Servoing

PVS Photometric Direct Visual Servoing

PGM Photometric Gaussian Mixture

PGMVS Photometric Gaussian Mixture BasedVisual Servoing

MPP Mixture of Photometric Potentials

MPPVS Mixture of Photometric Potentials Based Visual Servoing

ROS Robot Operating System

UTBM Université Technologique de Belfort-Montbéliard

MIS Modélisation, Information, Systèmes

CIAD Connaissance et Intelligence Artificielle Distribuée

VIO Visual Inertial Odometry

1 Introduction

1.1 Le Joint Robotics Laboratory

Le Joint Robotics Laboratory (JRL) est un laboratoire franco-japonais, collaboration entre le Centre National de la Recherche Scientifique (CNRS) et le National Institute of Advanced Industrial Science and Technology (AIST). Les sujets de recherchent couvrent un large spectre de la robotique, de la plannification à l'interaction homme-robot[1], en passant par une expertise reconnue en perception. Dans l'équipe de perception, supervisée par Guillaume Caron, sont abordés aussi bien les questions de modélisation des capteurs visuels [2] que d'asservissement visuel [3], de Simultaneous Localization and Mapping (SLAM)[4] ou d'observateurs visuels [5].

1.2 Le projet DVS-Straight

Le projet *DVS-Straight* est une collaboration franco-japonaise, avec coté japonais l'AIST, et coté français l'UTBM (Université Technologique de Belfort-Montbéliard), le MIS (laboratoire Modélisation, Information, Systèmes - Amiens) et le CIAD (laboratoire Connaissance et Intelligence Artificielle Distribuée - Université Bourgogne-Franche Comté).

L'objectif de ce projet est de maitriser les trajectoire des méthodes de Direct Visual Servoing (DVS) (développé en 2.1). Le projet a permis jusque là de proposer des filtrages étendant le domaine de convergence et la robustesse de ces méthodes [6], ainsi que de proposer une extension significative des domaines de convergence par l'utilisation de la vision omnidirectionnelle [3].

1.3 Objectif du stage et inscription dans le projet

Le stage s'inscrit dans le projet *DVS-Straight*, et a pour objectif de proposer une solution à un problème apparu avec l'utilisation de la vision omnidirectionnelle dans le cadre de l'asservissement visuel.

Maintenant que l'ensemble de l'environnement est observé, le robot à manipuler devient en permanence visible lors du contrôle. L'objectif de *DVS-Straight* étant notamment de maitriser la position et l'orientation (pose) de cette caméra, celle-ci ne doit pas être perturbée par la présence, et la configuration du robot, pouvant être différente pour une même pose de la caméra. L'objectif de ce travail de recherche est donc d'évaluer l'impact de la présence du robot sur les performances de l'asservissement en position (en terme de vitesse de convergence, de domaine de convergence et de précision), en rendant invisible le robot pour la loi de contrôle.

2 De l'asservissement visuel au MPPVS

2.1 Les 2 philosophies de l'asservissement visuel

L'asservissement visuel, Visual Servoing (VS) en anglais, est une technique de contrôle des robots basée sur l'information visuelle [7]. Il existe plusieurs philosophie d'asservissement visuel, qui se distinguent par la manière dont l'information visuelle est utilisée pour calcul :

— l'Indirect Visual Servoing (IVS), qui repose sur l'extraction de caractéristiques (dites features) éparses, comme des points d'intérêt ou des textures, à des fin de tracking, d'asservissement ou d'estimation de position relative d'un objet observé [8],

— le DVS qui, au sens large, utilise l'information visuelle calclable à partir de l'ensemble des intensités de chaque pixel de l'image pour calculer une commande.

Bien que le principe initial du DVS, appelé Photometric Direct Visual Servoing (PVS), utilise l'intensité de chaque pixel de l'image pour définir la loi de contrôle [9], il définit au sens large les méthodes utilisant une information dense basée sur l'intensité, il peut alors s'agir d'informations dérivées, comme pour le Kernel-based visual servoing [10], l'asservissement visuel basé sur les moments photométriques [11] ou encore celles basées sur une variate du PVS en utilisant des mixtures [6].

Dans les projets robotiques, les méthodes IVS sont souvent privilégiées, car développées depuis plus longtemps et étant adaptées au tracking, à la localisation, à l'asservissement en position... Avec pour avantage jusqu'ici d'avoir un domaine de convergence (*i.e.* la distance maximale entre la position désirée et courante des features, permettant de garantir que les 2 images s'aligneront) plus important que les méthodes DVS. Cependant, les méthodes DVS ont l'avantage de ne pas nécessiter d'étape de tracking, et de pouvoir être utilisées dans des environnements à luminosité et texture variables, tout en étant plus précises (sous réserve de convergence). De plus, les progrès successifs en DVS étendent de plus en plus le domaine de convergence de ces méthodes.

2.2 L'asservissement visuel

Que ce soit pour le IVS ou le DVS, le principe reste le même [7] :

1. Considérer un ensemble de features, géométriques et selectionnées (points d'intérêt, lignes, courbes, surfaces...) ou non (features denses), $f(t)$,
2. Définir une erreur, qui mesure la différence entre les features désirées et observées,

$$e(t) = f(t) - f^* \tag{1}$$

   Avec $f(t)$ les features observées à l'instant f^* les features désirées,

3. Relier la variation des features (Intensité, position, phase ou autre en fonction de la méthode utilisée) à la variation des commandes, pour obtenir une loi de contrôle,

$$\dot{f}(t) = \dot{e}(t) = \mathbf{L}_f \mathcal{V}_c(t) \tag{2}$$

   Avec $\mathcal{V}_c(t) = \left[v_c^T, \omega_c^T\right]^T$ la commande à appliquer, et $\mathbf{L}_f(t)$ la matrice jacobienne des features, communément appelée matrice d'interaction,

4. Calculer la commande à appliquer, afin que l'erreur $e(t)$ soit solution de l'équation différentielle $\dot{e}(t) = -\lambda e$:

$$\mathcal{V}_c(t) = -\lambda \mathbf{L}_f^+ e(t) \tag{3}$$

Avec λ un coefficient de gain, et $\mathbf{L}_f^+$ la pseudo-inverse de $\mathbf{L}_f$.

L'enjeu est d'estimer correctement la matrice d'interaction, qui dépendra des features auquelles on s'intéresse, de la camera utilisée, des conditions d'éclairage etc. En effet, à supposer que l'on ait estimé $\widehat{\mathbf{L}_f}$. En se basant sur (3) on peut calculer la commande comme étant :

$$\mathcal{V}_c(t) = -\lambda \widehat{\mathbf{L}_f}^+ e(t) \tag{4}$$

Ce qui revient à suivre l'équation différentielle :

$$\dot{e}(t) = -\lambda \mathbf{L}_f \widehat{\mathbf{L}_f}^T e(t) \tag{5}$$

Dans l'idéal, on souhaiterait que $\mathbf{L}_f \widehat{\mathbf{L}_f}^T = \mathbf{I}$. Le critère $\mathbf{L}_f \widehat{\mathbf{L}_f}^T > 0$ peut d'ailleurs être retrouvé par l'analyse de la stabilité avec la théorie de Lyapunov[12].

2.3 Le PVS, le PGMVS et le MPPVS

Le set de features considéré $f(t)$ n'est pas une information géométrique dans l'image, il peut aussi être son spectre [13], son histrogramme [14], ou encore ses moments photométriques [11].

Dans le cas du PVS, on considère l'intensité de chaque pixel de l'image, et on cherche à ce que l'intensité de chaque pixel de l'image (ou bien un set soignseusement selectionné, une forme de décimation pour éviter une trop grande complexité) observée soit égale à celle de l'image désirée. La formulation de l'erreur est alors :

$$e(t) = \mathcal{I}(t) - \mathcal{I}^* \tag{6}$$

Avec $\mathcal{I}(t)$ l'intensité de l'image observée, et $\mathcal{I}^*$ l'intensité de l'image désirée. En utilisant la relation 5, on obtient la loi de contrôle :

$$\mathcal{V}_c(t) = -\lambda \mathbf{L}_\mathcal{I}^+ \left(\mathcal{I}(t) - \mathcal{I}^*\right) \tag{7}$$

Avec $\mathbf{L}_\mathcal{I}$ la matrice d'interaction pour l'intensité.

Photometric Gaussian Mixture BasedVisual Servoing (PGMVS)

Le PGMVS [6] est une extension du PVS, qui applique une mixture de gaussiennes (Photometric Gaussian Mixture (PGM)) sur l'intensité de l'image observée et désirée (cf. figure 1), tel que en un point m de l'image, sa mixture s'exprime :

$$\mathbf{g}_\lambda(\mathcal{I}(m,t), m) = \sum_{n \in Image} \mathcal{I}(m,t) e^{-\frac{1}{2}\left(\frac{distance(m,n)}{\lambda}\right)^2} \tag{8}$$

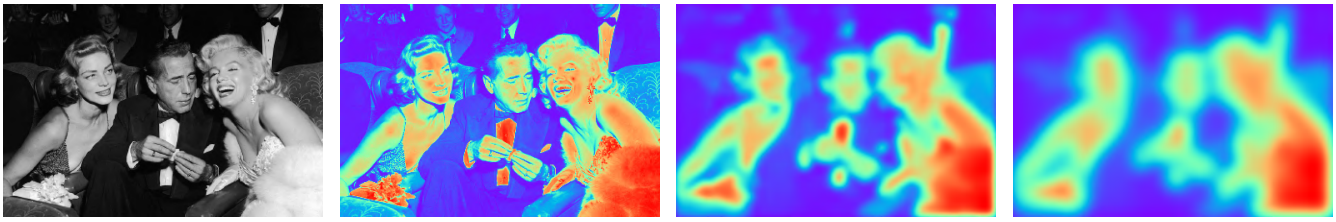
où $distance(m,n)$ est la distance entre les points m et n, dans le plan image (euclidienne), ou sur une sphere (distance géodésique) si la vision est omnidirectionnelle et l'image est projetée sur une sphère. λ est un paramètre de variance de la gaussienne, qui détermine la taille de la zone d'influence de chaque pixel.

Mixture of Photometric Potentials Based Visual Servoing (MPPVS)

Le MPPVS suit la même logique que le PGMVS, mais en utilisant une mixture de potentiels photométriques [15], qui s'exprime comme suit :

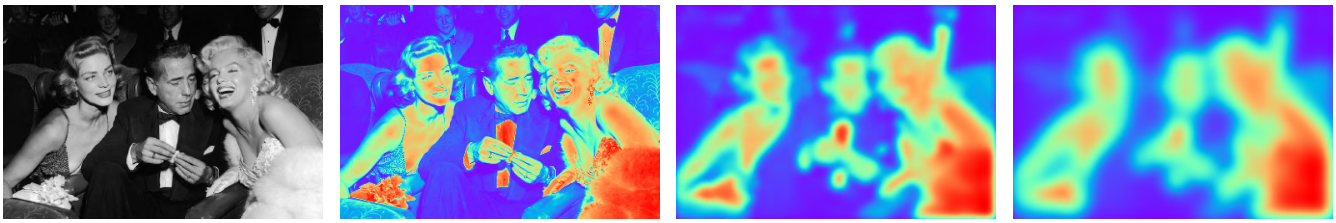
$$\mathbf{g}_\lambda(I(m,t),m) = \sum_{n \in Image} \overline{\mathcal{I}}(m,t) \frac{1}{\sqrt{(2\pi\lambda^2)^n}} e^{-\left(\frac{distance(m,n)}{\lambda}\right)^2} \tag{9}$$

Avec n le nombre de dimensions spatiales (2 sur une image plane, 3 sur un sphere), $\overline{\mathcal{I}}(m)$ le potentiel photométrique du pixel m, tel que $\sum_{n \in \text{Image}} \overline{\mathcal{I}}(m) = 1$.



(a) $1 \to 255$ (b) $1. \to 255.$ (c) $536 \to 3.8 \times 10^4$ (d) $2.1 \times 10^3 \to 1.5 \times 10^5$

FIGURE 1 – Exemple de mixture de gaussiennes sur une image. (a) Image originale, Mixtures (b) $\lambda = 0.1$, (c) $\lambda = 5$ (d) $\lambda = 10$, et leur plage de valeurs. Source : [6]



(a) $1 \to 255$ (b) $2 \times 10^{-6} \to 5 \times 10^{-4}$ (c) $4 \times 10^{-7} \to 3 \times 10^{-5}$ (d) $4 \times 10^{-7} \to 3 \times 10^{-5}$

FIGURE 2 – Exemple de mixture de potentiels photométriques sur une image. (a) Image originale, (b) $\lambda = 0.1$, (c) $\lambda = 5$ (d) $\lambda = 10$, et leur plage de valeurs.

L'utilisation de la MPP, de part sa forme normalisée, est plus proche de l'approche mathématique naturelle que l'on fait de l'usage des gaussiennes (lois normales). Cela permet notamment de paramétrer plus directement les performances que l'on souhaite obtenir, en terme de temps de réponse ou d'étalement.

Les mixtures ont pour intérêt, d'une part, de pouvoir filtrer les textures, et d'autre part de fournir une formulation continue et dérivable de l'erreur, afin de pouvoir estimer la matrice d'interaction. L'utilisation de ces mixtures a permis d'augmenter considérablement le domaine de convergence des méthodes DVS, de les rendre plus robustes aux variations de luminosité et de texture, au prix d'une dégradation de la précision. Que ce soit pour filtrer ou pour modéliser des

features denses, leur formulation analytique peremet de les utiliser sur des variétés différentes du plan, comme la sphère $\mathbb{S}^2$ pour une vision omnidirectionnelle [15]. C'est ce qui est utilisé dans le cadre de mon stage.

3 Estimation du masque

L'objectif de ce stage est de proposer une méthode pour masquer des zones de l'image qui ne sont pas pertinentes pour le calcul de l'erreur, et ainsi améliorer les performances de l'asservissement. Dans le cas de la vision à 360°, la présence du robot dans l'image, mobile et étant plus essentiel au déplacement de la caméra que volontairement dans la scène, peut dégrader les performances de l'asservissement.

Nous avons testé notre méthode sur un manipulateur mobile UR5e de Universal Robots, avec la caméra 360° Insta360 One X2 (cf. figures 3 et 4).

FIGURE 3 – Robot UR5e

FIGURE 4 – Caméra Insta360 One X2

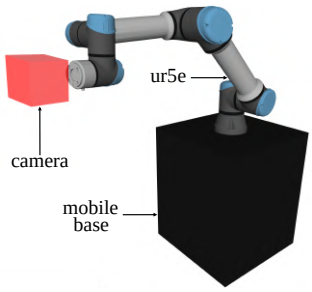


FIGURE 5 – Modèle Mujoco du montage

FIGURE 6 – Montage réel

3.1 Jumeau numérique

Afin d'estimer le point de vue de la caméra, nous avons utilisé un jumeau numérique du robot, qui permet de reproduire les mouvements du robot réel et de la caméra.

Les contraintes de conception de ce jumeau numérique sont les suivantes :

- Le jumeau numérique doit être capable de reproduire les mouvements du robot réel.
- Le jumeau numérique doit prendre le minimum de ressources possibles, les calculs d'asservissement visuel étant en quasi-temps réel et étant déjà très consommateurs.
- Le jumeau numérique n'a pas besoin de reproduire la physique du robot, mais seulement sa géométrie.
- La caméra doit être capable de renvoyer une ou plusieurs images permettant de représenter l'environnement à 360°.
- Le jumeau numérique doit être capable de renvoyer une image de l'environnement vu par la caméra, en fonction de la pose de la caméra.

Contraintes liées à la caméra

La nécessité de voir l'environnement virtuel à 360° nous a naturellement orienté vers l'utilisation d'environnements graphiques intégrant la possibilité d'utiliser des *shaders*, qui permettraient de projeter l'image de l'environnement sur une sphère ou bien de venir ou de reproduire les distorsions des 2 caméras de la caméra 360°.

Unity3D , UnrealEngine, Blender ou autres étant des logiciels lourds et nécessitant des compétences en modélisation 3D, il a été choisi d'utiliser plutôt des bibliothèques rattachée directement à OpenGL, ici GLFW.

Contraintes liées au robot

Les contraintes de ressources nous ont orientés vers des solutions déjà existantes pour déplacer et obtenir la pose du robot et de la caméra. Nous nous sommes tournés vers le simulateur Mujoco, pour sa légèreté, sa simplicité d'utiliser et pour le micro-service de visualisation de scène qu'il propose, rudimentaire, léger et basé sur glfw (cf. figure 5).

Contraintes liées au programme d'asservissement

Le programme utilisé pour la tâche d'asservissement est basé sur LibPer, une bibliothèque développée au MIS, qui permet de faire de l'estimation de pose et de l'asservissement visuel. Elle permet notamment d'accueillir des fonctions pour estimer des features et leurs mixtures sur $\mathbb{S}^2$, en prenant en image d'entrée :

— soit une projection equirectangulaire de l'environnement,

— soit 2 images hemisphériques avec leur position et champs de vision respectifs.

L'ensemble des tâches d'acquisition de l'image, de génération de la commande et de communication avec le robot est géré via ROS1. Le masque généré doit donc être publié sur un topic ROS dédié.

3.2 Environnement graphique

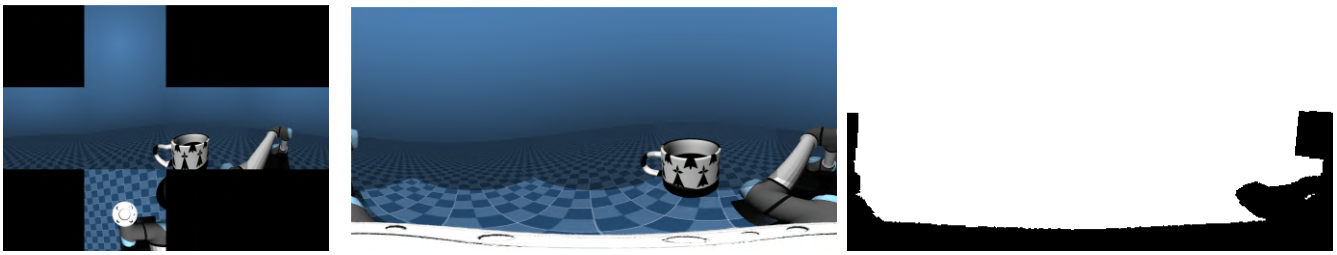
L'utilisation d'un environnement graphique dédié a permis de générer une scène binaire, sur fond blanc et robot noir (figure 7). L'acquisition n'a pas été faite sur base d'une seule camera et d'un shader par manque de connaissance et de temps.

L'alternative trouvée a été d'utiliser 6 caméras, disposées à 90° les unes des autres, de 90° de champ de vision afin de couvrir l'ensemble de l'environnement. Il est ainsi possible de récupérer la cubemap de l'environnement (figure 7a), transformable en image equirectangulaire (figure 7b).

Le temps de calcul pour transformer 6 images 128×128 en une image equirectangulaire 256×512 étant conséquent, ce calcul a été parallélisé sur GPU avec CUDA.

3.3 Alignement du masque avec l'image réelle

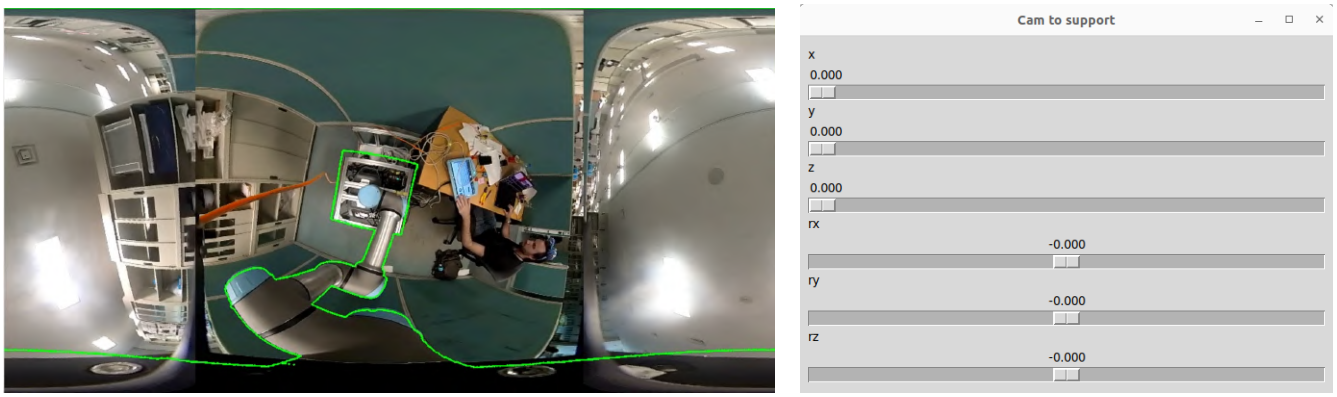
Les articulations du robot réel sont récupérées via ROS grâce à un driver fourni par Universal Robots. Un noeud est dédié à la mise à jour des articulations, à l'acquisition des 6 images et à la publication du masque.



(a) cubemap couleur (b) projection equirectangulaire (c) masque (autre configuration)

FIGURE 7 – Différentes étapes du calcul du masque

Afin de positionner correctement la caméra dans l'environnement réel, et afin de pouvoir travailler avec d'autres caméras, un algorithme de calibration manuelle permet de mettre à jour les paramètres extrinsèques de la caméra dans le fichier de configuration de l'environnement virtuel (figure 8a). Une fois la calibration faite, le masque est aligné avec l'image réelle, aux imperfections près dans les images fournies par les caméras.



(a) contours du masque (en vert) (b) Fenêtre de réglage

FIGURE 8 – Etape d'alignement du masque avec l'image réelle

4 Stratégies de masquage

Le masquage des features impacte le calcul de l'erreur différemment selon la stratégie employée. On s'intéressera ici à comparer 3 stratégies de masquage :

— Masquage naïf : le masque est appliqué sur l'image avant d'être porté sur la sphère et que la mixture soit calculée (cas ② figure 9)

— Masquage de la mixture : le masque est appliqué sur l'image, et et la mixture des features cachées vaut 0 (cas ③ figure 9)

— Retrait de la feature masquée : la feature bien est portée sur la sphère par souci de compatibilité avec les fonctions de la librairie, mais n'est pas prise en compte dans le calcul de la mixture, et n'apparait pas dans le vecteur des erreurs $g_\lambda(\mathcal{I}(t)) - g_\lambda(\mathcal{I}^*)$ (cas ④ figure 9)

Une illustration avec 3 features en 1D, dans le cas de l'application d'une Mixture of Photometric Potentials (MPP), permet de mieux visualiser les différences entre ces stratégies.

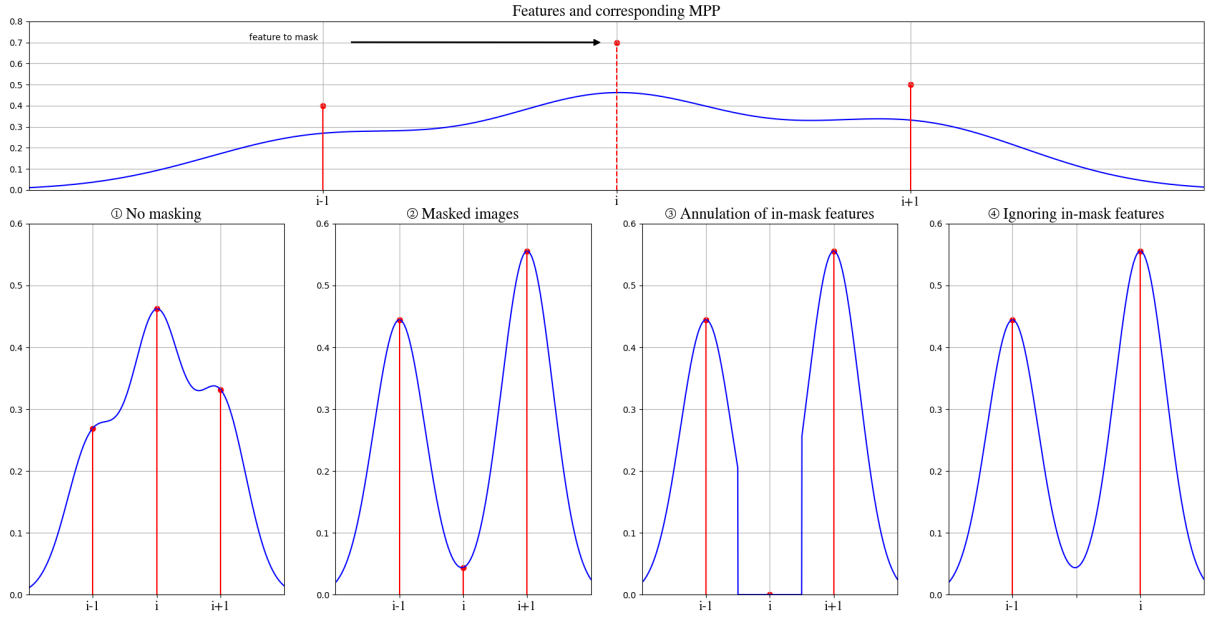


FIGURE 9 – Illustration des 3 stratégies de masquage

Dans le cas purement photométrique ces 3 méthodes sont identiques, il est toutefois possible de montrer que même dans le cadre des mixtures, les méthodes ③ et ④ sont équivalentes (Annexe 6). Ce résultat peut s'intuiter par le fait que des composanted d'un vecteur d'erreur forcés à 0, dans le cas d'un calcul de commande basé sur des intensités, n'apporte pas d'information.

Seuls les cas ② et ③ sont donc comparés, en terme de précision et de vitesse de convergence dans un premier temps, puis en terme de domaine de convergence à l'avenir. Le masque a été dilaté de 5 pixels pour couvrir l'intégralité du robot et masquer les imperfections dans les projections equirectangulaires des images.

Les programmes de génération de masque sont disponibles sur github.

5 Expériences et Résultats

5.1 Critères de performance et expériences prévues

Les cas ①, ② et ③ sont comparés dans différentes situations, afin d'isoler les critères de performance. Ces expériences sont toujours en cours et prennent du retard, notamment à cause de la difficulté à trouver des configurations dans lesquelles l'UR5e ne se met pas en sécurité pour des problèmes d'auto-collision ou de cinématique inverse. Les situations sont les suivantes :

— Petites translations, petites rotations, afin de comparer la vitesse de convergence avec et l'erreur statique.

— Grandes translations, grandes rotations, afin de comparer les potentiels dépassements (à faire : ces expériences sont les plus difficiles à réaliser sur le robot actuel).

— Pose identique avec une configuration articulaire différente, afin de tester la sensibilité de l'algorithme de contrôle à la présence du robot.

— Petites translations, petites rotations, dans des configurations articulaires complètement différentes (à faire),

— Ensemble de tests aléatoires, afin d'une part d'estimer numériquement un domaine de convergence, mais aussi de faire ressortir des métriques plus générales (erreur statique moyenne, trajectoires...), à faire en simulation.

Les métriques observées sont les suivantes :

— La distance cartésienne entre la pose désirée et la pose observée,

— L'erreur angulaire entre l'orientation désirée et l'orientation observée,

— Le coût photométrique, à savoir la norme de l'erreur $e(t) = g_\lambda(\mathcal{I}(t)) - g_\lambda(\mathcal{I}^*)$.

Chaque expérience sera réalisés 5 fois dans chaque cas, afin d'être lissée statistiquement. Le gain du contrôleur peut être ajusté pour chaque mise en situation, mais sera fixé pour toutes les expériences d'une même situation.

(a) Image equirectangulaire

(b) Image projetée sur la sphère

FIGURE 10 – Exemple de vue de la caméra sur l'UR5e

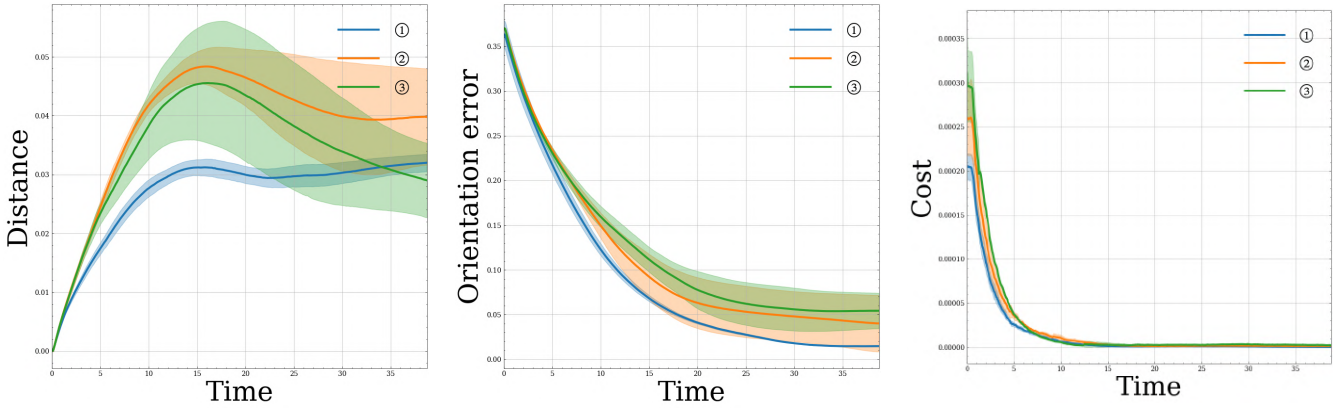
Il est important de souligner les soucis de répétabilité des expériences : les conditions d'éclairage naturel peuvent être amenées à changer au cours du temps, impactant la commande générée. Bien que ceci n'impacte pas le comportement général de l'asservissement, les comportements peuvent être différents d'une expérience à l'autre et peuvent mener le robot à suivre des trajectoires qui risquent de le bloquer.

Certaines expériences ont été menées sous lumière uniquement artificielle, afin de limiter ces variations, mais ce n'est pas le cas de toutes. La lumière artificielle amène aussi à un autre problème encore très mal palié : la spécularité sur l'image dûes aux sources lumineuses, non modélisée ici mais qui peut fausser les estimations de la matrice d'interaction.

5.2 Petits déplacements

petite rotation

Pour une rotation pure de la caméra, de 20° autour de l'axe z, et un gain statique $\lambda = 0.3$ (temps caractéristique de $3s$) les résultats sont les suivants :



(a) Distance cartésienne (b) Erreur angulaire (c) Coût photométrique

FIGURE 11 – Rotation pure de 15°. Les marges d'erreurs sont les intervalles de confiance à 95%

Malgré une forte variance, on peut analyser les courbes, et par curve-fitting extraire quelques critères de performance (table 1) :

	Distance cartésienne		Erreur angulaire		coût photométrique	
	τ	ϵ_s	τ	ϵ_s	τ	ϵ_s
①	×	×	2^{nd} ordre	3%	$2.31s$	0.5%
②	×	×	2^{nd} ordre	11%	$3.s$	0.7%
③	×	×	$9.52s$	16%	$3.s$	0.7%

TABLE 1 – Critères de performance pour une rotation pure de 15°

Cet exemple d'expérience ne permet pas de conclure directement sur l'efficacité de telle ou telle méthode de masquage, mais la variance très faible pour la méthode ① laisse penser que cette

méthode est plus stable que les autres. Il est toutefois nécessaire de réaliser plus d'expériences pour confirmer cette hypothèse.

Le temps de convergence est, pour toutes les méthodes, autour de $3s$, ce qui est cohérent avec le temps caractéristique du contrôleur.

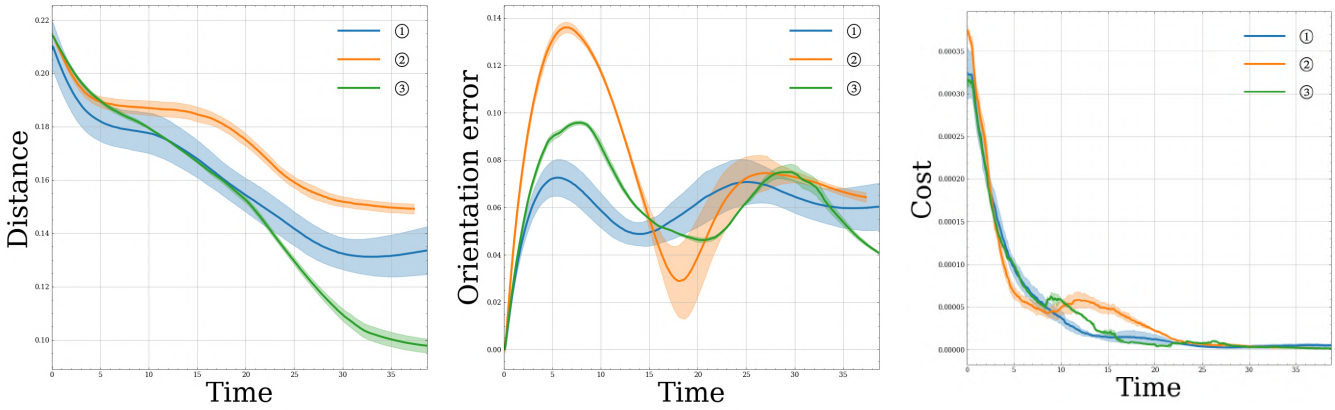
petite translation

Pour l'heure, il n'y a pas de résultats pour les petites translations, les expériences étant en cours.

5.3 Grands déplacements

grande translation

Pour un déplacement de $20cm$ en translation pure, et un gain statique $\lambda = 0.2$ (temps caractéristique de $5s$) les résultats sont les suivants :



(a) Distance cartésienne (b) Erreur angulaire (c) Coût photométrique

FIGURE 12 – Translation pure de $20cm$. Les marges d'erreurs sont les intervalles de confiance à 95%

Cette expérience n'a pas eu le temps de converger, du fait que la trajectoire adoptée forçait le robot à se mettre en sécurité par risque d'auto-collision (d'où le sursaut du coût photométrique). Il est donc nécessaire de retrouver une trajectoire plus sûre pour réaliser cette expérience, et de réduire le gain du contrôleur pour éviter les oscillations.

les résultats sont donc à prendre avec un regard critique, mais on peut tout de même extraire quelques critères de performance (table 2) :

grande rotation

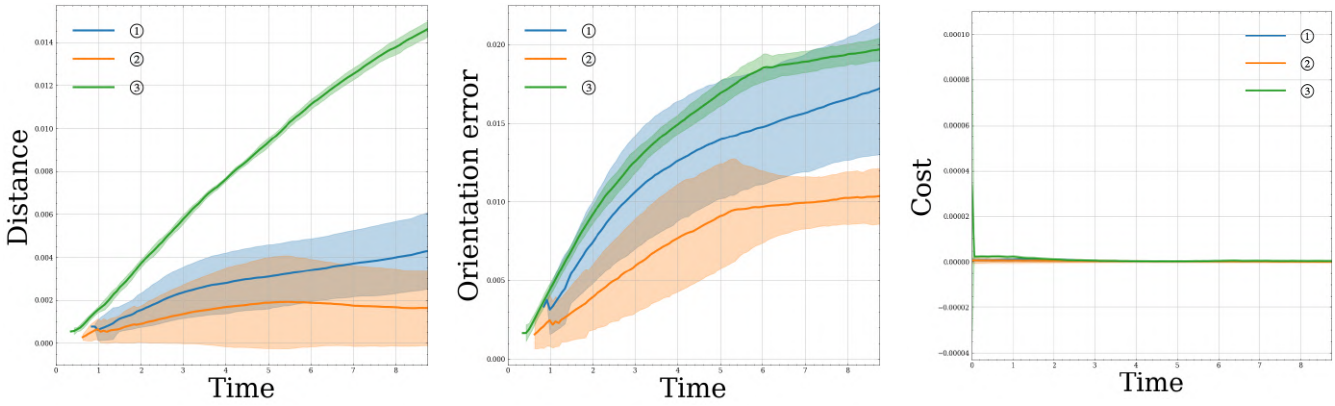
Pour l'heure, il n'y a pas de résultats pour les grandes rotations, les expériences étant en cours.

	Distance cartésienne		Erreur angulaire		coût photométrique	
	τ	ϵ_s	τ	ϵ_s	τ	ϵ_s
①	×	×	×	×	$3.9s$	1.4%
②	×	×	×	×	$3.2s$	0.4%
③	×	×	×	×	$4.2s$	0.5%

TABLE 2 – Critères de performance pour une rotation pure de 15°

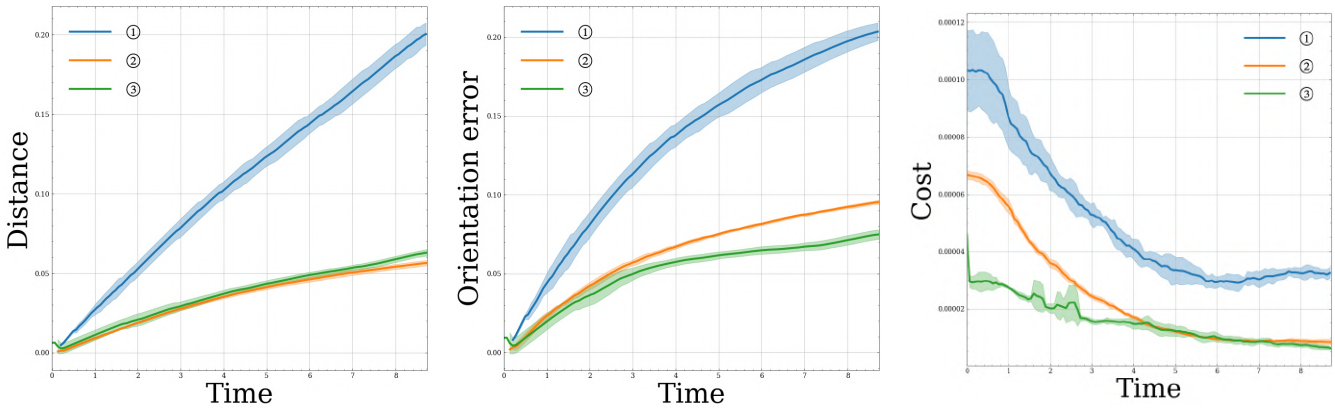
5.4 Pose identique, configuration articulaire différente

Ce résultat est le plus stable du set, et permet de vérifier l'intérêt de retirer les features masquées au lieu de seulement masquer le robot.



(a) Distance cartésienne (b) Erreur angulaire (c) Coût photométrique

FIGURE 13 – Pose identique, configuration articulaire articulaire identique. Ce résultat sert de référence.



(a) Distance cartésienne (b) Erreur angulaire (c) Coût photométrique

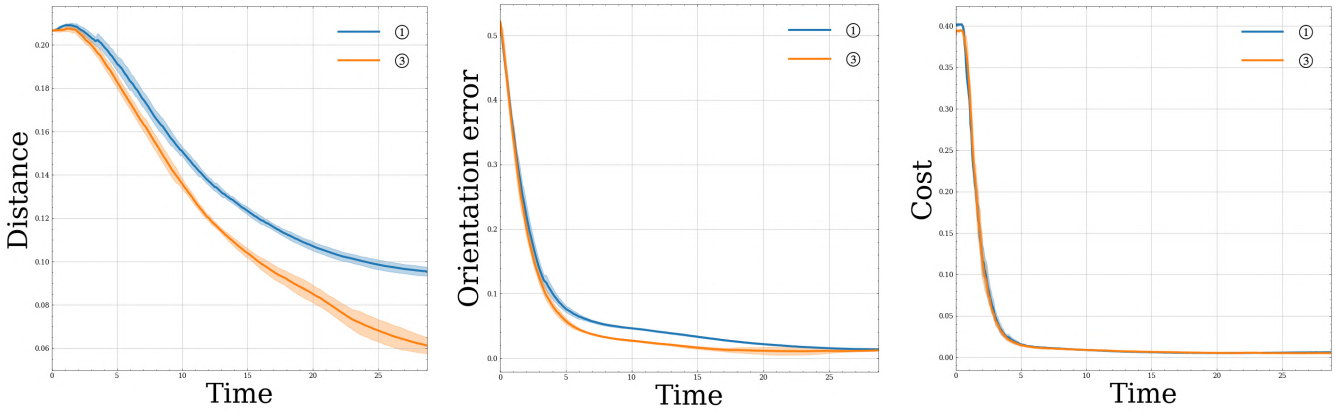
FIGURE 14 – Pose identique, configuration articulaire différente.

Dans les 2 cas, le robot commence à dériver. Il est toutefois intéressant de remarquer différents points :

— Les méthodes ① et ② dérivent moins lorsque la configuration articulaire est identique. En effet, le robot étant plus proche de la caméra, lors d'un petit déplacement dû à une perturbation (accumulation d'erreurs dans le calcul ?), il défilera plus vite que l'arrière plan, et la différence d'intensité dans cette zone a un effet attractif sur le robot.

— La méthode ③ est celle qui dérive le moins lorsque la configuration articulaire est différente. En effet, tandis que la différence d'intensité dûe à la non superposition des masques génère une erreur, donc une commande dans les cas ① et ②, cette différence d'intensité n'est pas prise en compte dans le calcul de la commande, et n'a donc pas d'effet sur le robot dans le cas ③.

Les autres expériences sont en cours, les prochains objectifs de pouvoir trouver des situations dans lesquelles on peut assurer la répétabilité des expériences, de trouver des conditions de lumière et de trouver des paramètres de mixture adaptés à l'environnement de travail. Il est notamment possible, dans des conditions encore à caractériser, d'obtenir des comportements très propices à la caractérisation, comme figure 15.



(a) Distance cartésienne (b) Erreur angulaire (c) Coût photométrique

FIGURE 15 – Résultat issu d'un test de fonctionnement (15° de rotation, 20cm de translation).

6 Axes de travail actuel et perspectives d'amélioration

Les résultats actuels ne permettent pas encore de quantifier significativement l'impact du masquage sur les performances du contrôle. Les tests les plus intéressants à faire sont les grands déplacements et les petits en s'arrangeant pour que les configurations finales diffèrent. L'objectif à court terme est de faire toutes ces expérimentations, pour une soumission SII .

Sur le plus long terme, il serait intéressant de :

— étendre les tests à d'autres configurations de caméra, et à d'autres robots,

— tester les méthodes de masquage sur des environnements virtuels,

— intégrer le masquage comme une fonctionnalité dans la bibliothèque libPeR utilisée.

— porter l'ensemble des programmes sur ROS2, voir utiliser d'autres frameworks comme mc-rtc.

La possibilité d'intégrer des masques dans des méthodes d'asservissement visuel denses est une piste de recherche intéressante, qui pourrait être étendue à :

— une plus grande variété de masques, comme des masques d'inpainting génératif [16] pour minimiser son impact sur la commande, ou de détection d'objet [17], afin de limiter l'information visuelle uniquement à celle que l'on souhaite,

— une implémentation dans des observateurs visuels, comme un gyroscope visuel [5] afin de limiter les sources de décrochage, de dérive ou peut-être d'augmenter le domaine de convergence,

— une implémentation dans des méthodes de SLAM pour limiter les erreurs de cartographie, ou dans des méthodes de Visual Inertial Odometry (VIO) pour limiter les erreurs de localisation.

Enfin, les masques pourraient guérir le cancer, permettre de voyager dans le temps et de tondre la pelouse en moins de 5 minutes.

Démonstration de l'équivalence entre les stratégies ③ et ④

La matrice d'interaction $L_{\mathbf{I}}$ est définie tel que :

$$\dot{e}(t) = L_{\mathbf{I}}\mathcal{V}_c \tag{10}$$

Avec e le vecteur des erreurs, $\mathcal{V}_c$ la commande $\left[v_c^T, \omega_c^T\right]^T$, et $\dot{e}$ la dérivée de l'erreur.

Afin de faire converger l'erreur vers 0, on choisit $\mathcal{V}_c$ tel que $\dot{e} = -\lambda e$. Ainsi il faut :

$$L_{\mathbf{I}}\mathcal{V}_c = -\lambda \mathbf{e}(u) \tag{11}$$

e est de taille $(n, 1)$, $\mathcal{V}_c$ de taille $(6, 1)$, $L_{\mathbf{I}}$ de taille $(n, 6)$., avec n le nombre de features considérées.

Soit $\mathbf{S} \subset \{1, ...n\}$ tel que $|\mathbf{S}| = k$ (k étant le nombre de features masquées), et $\forall i \in \mathbf{S}, e_i = 0$. $e_i = 0$ est une conséquence du masquage, ainsi tant qu'il y a statistiquement suffisamment de features représentatives dans la scène (hypothèse plus forte que la condition de matric ede rang plein, supposée valide dans le cas des images d'environnements réels pouvant être modélisés par des distributions aléatoires), $e_i = 0$ est considéré comme n'ayant pas d'influence sur $\mathcal{V}_c$, ainsi la contrainte s'exprime :

$$\begin{array}{rrcl} \forall \mathcal{V}_c, e_i = 0 & \Rightarrow L_{\mathbf{I}}^i \mathcal{V}_c & = & 0 \\ & \Rightarrow L_{\mathbf{I}}^i & = & [\mathbf{0}]_{1,6} \end{array} \tag{12}$$

Les lignes $L_{\mathbf{I}}$ correspondantes à $e_i = 0$ doivent alors être nulles.

En déplaçant $L_{\mathbf{I}}$ et les e correspondants à $e_i = 0$ sur les dernières lignes de la matrice, On peut décomposer l'équation avec des matrices par blocs sans modifier le système d'équations associé :

$$\begin{bmatrix} \tilde{L}_{\mathbf{I}} \\ \mathbf{0}_{k,6} \end{bmatrix} \mathcal{V}_c = -\lambda \begin{bmatrix} \tilde{e} \\ \mathbf{0}_{k,1} \end{bmatrix} \tag{13}$$

avec $\tilde{e} = \left[e_i, i \notin \mathbf{S}\right]$ and $\tilde{L}_{\mathbf{I}} = \left[L_{\mathbf{I}}^i, i \notin \mathbf{S}\right]$

Ainsi, le système d'équations linéaires correspondant s'ecrti :

$$\left\{ \begin{array}{rl} \tilde{L}_{\mathbf{I}}\mathcal{V}_c = & -\lambda\tilde{e} \\ \mathbf{0}_{k,6}\mathcal{V}_c = & \mathbf{0}_{k,1} \end{array} \right. \tag{14}$$

Ce qui équivaut à :

$$\tilde{L}_{\mathbf{I}} v = -\lambda \tilde{e} \tag{15}$$

Une feature masquée n'a donc pas d'impact sur le calcul de la commande, et peut être retirée de la matrice d'interaction. Les cas ③ et ④ sont donc équivalents.

C'est la vérification expérimentale quasi-systématique qui a mené à tenir cette reflexion, afin de gagner en temps d'expérimentation.

Table des figures

1 Exemple de mixture de gaussiennes sur une image. (a) Image originale, Mixtures (b) $\lambda = 0.1$, (c) $\lambda = 5$ (d) $\lambda = 10$, et leur plage de valeurs. Source : [6] 8
2 Exemple de mixture de potentiels photométriques sur une image. (a) Image originale, (b) $\lambda = 0.1$, (c) $\lambda = 5$ (d) $\lambda = 10$, et leur plage de valeurs. 8
3 Robot UR5e 9
4 Caméra Insta360 One X2 9
5 Modèle Mujoco du montage 9
6 Montage réel 9
7 Différentes étapes du calcul du masque 11
8 Etape d'alignement du masque avec l'image réelle 11
9 Illustration des 3 stratégies de masquage 12
10 Exemple de vue de la caméra sur l'UR5e 13
11 Rotation pure de 15°. Les marges d'erreurs sont les intervalles de confiance à 95% . 14
12 Translation pure de $20cm$. Les marges d'erreurs sont les intervalles de confiance à 95% 15
13 Pose identique, configuration articulaire articulaire identique. Ce résultat sert de référence. 16
14 Pose identique, configuration articulaire différente. 16
15 Résultat issu d'un test de fonctionnement (15° de rotation, $20cm$ de translation). . . 17

Références

[1] R. Cisneros, A. Dallard, M. Benallegue, K. Kaneko, H. Kaminaga, P. Gergondet, A. Tanguy, R. P. Singh, L. Sun, Y. Chen, C. Fournier, G. Lorthioir, M. Tsuru, Ś. Chefchaouni-Moussaoui, Y. Osawa, G. Caron, K. Chappellet, M. Morisawa, A. Escande, K. Ayusawa, Y. Houhou, I. Kumagai, M. Ono, K. Shirasaka, S. Wada, H. Wada, F. Kanehiro, and A. Kheddar, "A cybernetic avatar system to embody human telepresence for connectivity, exploration, and skill transfer," *International Journal of Social Robotics*, January 2024.

[2] E. Goichon, G. Caron, P. Vasseur, and F. Kanehiro, "On camera model conversions," in *IEEE International Conference on Robotics and Automation*, (Yokohama, Japan), May 13-May 17 2024.

[3] N. Crombez, J. Buisson, A. André, and G. Caron, "Dual-hemispherical photometric visual servoing," *IEEE Robotics and Automation Letters*, vol. 9, May 2024.

[4] E. Goichon, F. Kanehiro, P. Vasseur, and G. Caron, "Evaluation de localisation et cartographie simultanées de robot mobile par caméra couleur-profondeur embarquée," in *Congrès des Jeunes Chercheurs en Vision par Ordinateur*, (Carqueiranne, France), May 22-May 26 2023.

[5] A. André and G. Caron, "Photometric visual gyroscope for full-view spherical camera," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, (New-Orleans, LA, USA), pp. 5232–5235, IEEE, June 19-June 20 2022.

[6] N. Crombez, E. M. Mouaddib, G. Caron, and F. Chaumette, "Visual servoing with photometric gaussian mixtures as dense features," *IEEE Transactions on Robotics*, vol. 35, pp. 49–63, February 2019.

[7] F. Chaumette and S. Hutchinson, "Visual servo control. i. basic approaches," *Robotics & Automation Magazine, IEEE*, vol. 13, pp. 82 – 90, 01 2007.

[8] E. Marchand and F. Chaumette, "Feature tracking for visual servoing purposes," *Robotics and Autonomous Systems*, vol. 52, no. 1, pp. 53–70, 2005. Special issue on "Advances in Robot Vision", D. Kragic, H. Christensen (Eds.).

[9] C. Collewet and E. Marchand, "Photometric visual servoing," *IEEE Transactions on Robotics*, vol. 27, no. 4, pp. 828–834, 2011.

[10] V. Kallem, M. Dewan, J. P. Swensen, G. D. Hager, and N. J. Cowan, "Kernel-based visual servoing," in *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1975–1980, 2007.

[11] M. Bakthavatchalam, F. Chaumette, and E. Marchand, "Photometric moments : New promising candidates for visual servoing," in *2013 IEEE International Conference on Robotics and Automation*, pp. 5241–5246, 2013.

[12] S. Sastry, *Lyapunov Stability Theory*, pp. 182–234. New York, NY : Springer New York, 1999.

[13] E. Marchand, "Direct visual servoing in the frequency domain," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 620–627, 2020.

[14] Q. Bateux and E. Marchand, "Histograms-based visual servoing," *IEEE Robotics and Automation Letters*, vol. 2, no. 1, pp. 80–87, 2017.

[15] G. Caron and F. Morbidi, "Spherical visual gyroscope for autonomous robots using the mixture of photometric potentials," in *IEEE International Conference on Robotics and Automation*, pp. 820–827, May 21-May 26 2018.

[16] J. Yu, Z. Lin, J. Yang, X. Shen, X. Lu, and T. S. Huang, "Generative image inpainting with contextual attention," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[17] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "Overfeat : Integrated recognition, localization and detection using convolutional networks," 2013. cite arxiv :1312.6229.