Projet 2A - Département Information & Systèmes

Rapport final

Lampe Robot

BARBILLON Stanislas - JULIO Marjorie - STAB Aurélien

Sommaire

Introduction

Durant cette 2ème année à l'Ecole des Mines de Nancy, nous avons réalisé un projet en cohérence avec notre formation au département Information & Systèmes. Le sujet choisi pour ce projet est celui de la Lampe Robot. Celui-ci est réalisé en partenariat avec le laboratoire Loria et s'inscrit dans le cadre d'un projet plus large qui s'intitule "PsyPhINe". Cette appellation est l'abréviation de "Psychologie, Philosophie, Informatique, Neuroscience" dont la thématique est l'étude des interactions entre l'Homme et les machines ou les robots. En effet, les machines deviennent de plus en plus présentes dans notre environnement et cette évolution est extrêmement rapide au sein de notre société. Les questions qui se posent face à cette thématique sont donc nombreuses. Comment percevons-nous la machine dans notre environnement ? Peut-on lui prêter des émotions humaines ? C'est le but de répondre à ces questions que nous participons au projet de la Lampe Robot. Ce projet se déroule autour d'un unique objet : une lampe robotisée. Une lampe fait généralement partie de la vie quotidienne et se fond dans notre environnement. C'est pourquoi, une lampe a été équipée de moteurs pour lui permettre d'effectuer des mouvements. Ainsi, on peut se demander comment un utilisateur va réagir face à une lampe robotisée auquelle on donne des comportements. L'idée est de voir si on peut humaniser un robot et lui donner des traits d'intelligence. C'est donc dans le cadre de cet ambitieux projet, que nous avons travaillé sur la Lampe Robot.

I. Présentation du projet

Mission demandée

La Lampe Robot est utilisée pour réaliser des expériences en direct, en interaction avec un humain. L'objectif de ces expériences est de faire interagir l'humain avec la lampe robotisée. Ainsi, le contenu de l'expérience peut être soit de faire faire à la personne des jeux en présence de la lampe d'observer son interaction avec celle-ci soit de donner directement à la lampe la capacité d'interagir avec la personne. Ainsi, la lampe robotisée n'a d'intérêt que si on peut la rendre dynamique, c'est-à-dire lui affecter des comportements ou des émotions qui se rapprochent de ce que l'homme fait de manière naturelle.

Notre mission est donc d'apporter à cette lampe des solutions techniques pour faciliter son utilisation et rendre l'échange avec une personne humaine plus dynamique et plus interactif. Ainsi, l'objectif est de permettre à la lampe de réagir de manière instantanée, il est également intéressant pour l'expérience, de donner à la lampe des attitudes naturelles qui se rapprochent le plus possible de celles d'un homme. Enfin un dernier objectif est de rendre cette lampe autonome, c'est à dire qui ne nécessite pas l'intervention d'un homme pour la contrôler/piloter.

Expérience du projet Psyphine

Gestion du projet

Afin d'organiser au mieux le projet, nous avons établi le mode de fonctionnement qui nous a semblé le plus favorable. Ce fonctionnement repose sur l'alternance régulière entre le TechLab des Mines Nancy et les locaux du Loria.

En effet, cela nous permet de travailler de façon autonome et essayer de nous aider mutuellement lorsque nous rencontrons des problèmes, lorsque nous sommes sur le campus d'Artem. L'objectif est de nous inciter à avancer sur le projet en autonomie. Les visites au Loria permet de poser des questions à nos encadrants pour éviter de rester bloquer inutilement sur des points techniques. C'est également l'occasion de faire le point sur l'avancement avec nos encadrants et de fixer les nouveaux objectifs à venir. D'autre part, la lampe robotisée n'est présente que sur le site du Loria, mais nous pouvons utiliser le logiciel de simulation VREP, pour tester notre code sans la lampe.Outils techniques utilisés

Logiciel de simulation

Comme précisé dans le paragraphe précédent, nous utilisons un logiciel de simulation, VREP, ce qui permet de tester notre code et ainsi observer la réponse du robot aux comportements commandés. Ceci a également l'avantage d'éviter que les moteurs surchauffent du robot réel à cause d'une utilisation en continue lors des essais du code.

Il faut néanmoins faire attention de légères différences entre le robot réel et le robot représenté sur VREP, à cause de leur configuration. En effet les moteurs ne pas orientés dans le même sens et les offsets sont également différents.

Bibliothèques python

Lorsque nous nous sommes emparé du projet la lampe avait déjà une interface de bas niveau pour contrôler les moteurs de la lampe. La structure utilisait la bibliothèques pypot pour contrôler les moteurs du robots sous python, une petite interface graphique avait été faite pour sauvegarder des mouvements ou les lancer.

Nos encadrants nous ont conseillés d'implémenter la structure de poppy, une bibliothèque python de plus haut niveau que pypot qui s'occupe de la communication avec différents logiciel annexes comme Vrep.

Gestion du code

Pour nous permettre de programmer simultanément, nous utilisons un repository privé sur git nommé “p2a_lampe_robot”. Le travail effectué cette année a ainsi pu être largement facilité par cet outil qui gère les versions. Il s’agit aussi d’un moyen propre et facile de transmettre le code à nos encadrants.

II. Actions réalisées au 1er semestre

Le code existant permettait de contrôler le robot sur ses moteurs mais était très peu robuste au lancement de multiple primitive et sa communication avec Vrep était très archaïque. Notre premiere tâche a été de passer sur une architecture de type Poppy. Ce choix était motivé pour centraliser les ordres donnés aux moteurs mais surtout avoir une communication efficace avec notre simulateur.

Construction de la nouvelle architecture

Nous nous sommes alors peu à peu documenté sur les bibliothèque Poppy et Pypot. La bibliothèque Poppy était assez peu documenté et nous dûmes nous inspirer des codes existants, principalement le code d'ergo-jr (un bras robot sous Poppy) pour constituer notre propre extension de Poppy.

Une fois l'architecture de base construite nous nous sommes mis à la tâche de passer l'ensemble des fonctions précédemment construites sous l'architecture pypot sous notre nouvelle architecture.

Nous avons eu quelques difficultés sur le format des fichiers qui n'étaient pas tout à fait définie et nous avons dû traiter à la fois des fichiers de type pickles ainsi qu'un format propre au projet (assez proche d'un format CSV). Finalement nous sommes resté sur ce dernier format car plus lisible et donc plus facilement éditable à la main ce qui semblait important pour nos encadrants.

Définition des mouvements

Une fois la structure des fichiers choisie, il à fallu trouver une solution simple à mettre en place et reproductible pour que les mouvements soit facilement applicables à la lampe. Nous avons donc fait le choix des primitives Pypot qui semblait les objets les plus proches de ce que l'on souhaitait faire.

Lorsque les anciens mouvements ont ainsi été transformés en primitives, le choix du type de primitives s'est posé (Loop, Posture, primitive centralisé…). Nous avons exploré la possibilité de les transformés sous la forme de Move (un autre type de primitive) mais il y avait quelques doutes sur ce que faisait réellement les bibliothèques et par manque

d’informations et de contrôle nous avons décidés avec nos encadrants de rester sous la forme classique des primitives de type Posture et Loop.

Stockage des primitives

Une fois cette décision prise, il fallait trouver une solution pour permettre de stocker et d’ajouter des primitives à celles déjà existantes. Les débuts de ce qui sera le futur ORM ont alors été ajoutés. Un fichier de sauvegarde au format JSON à été créé et il contenait les primitives sous forme (clé: nom de la primitive, valeur: lien vers le fichier de description des mouvements). Les fichiers de description des mouvements sont ceux décrit précédemment, il s’agit de fichiers ayant une structure proche de CSV et qui contiennent les positions des moteurs à des temps donnés.

Ce fichier servait lors de l'initialisation de la lampe, pour qu’elle puisse savoir quelles primitives lui étaient associées. Ce concept était intéressant et assez puissant, d’autant qu’il était facile d’ajouter des primitives au chargement. Nous avons donc décidé de continuer de l’améliorer avec l’ORM et de créer de nouvelles fonctionnalité avec le gestionnaire de primitive lors du second semestre.

III. Actions réalisées au 2nd semestre

Gestionnaire des primitives

Contexte

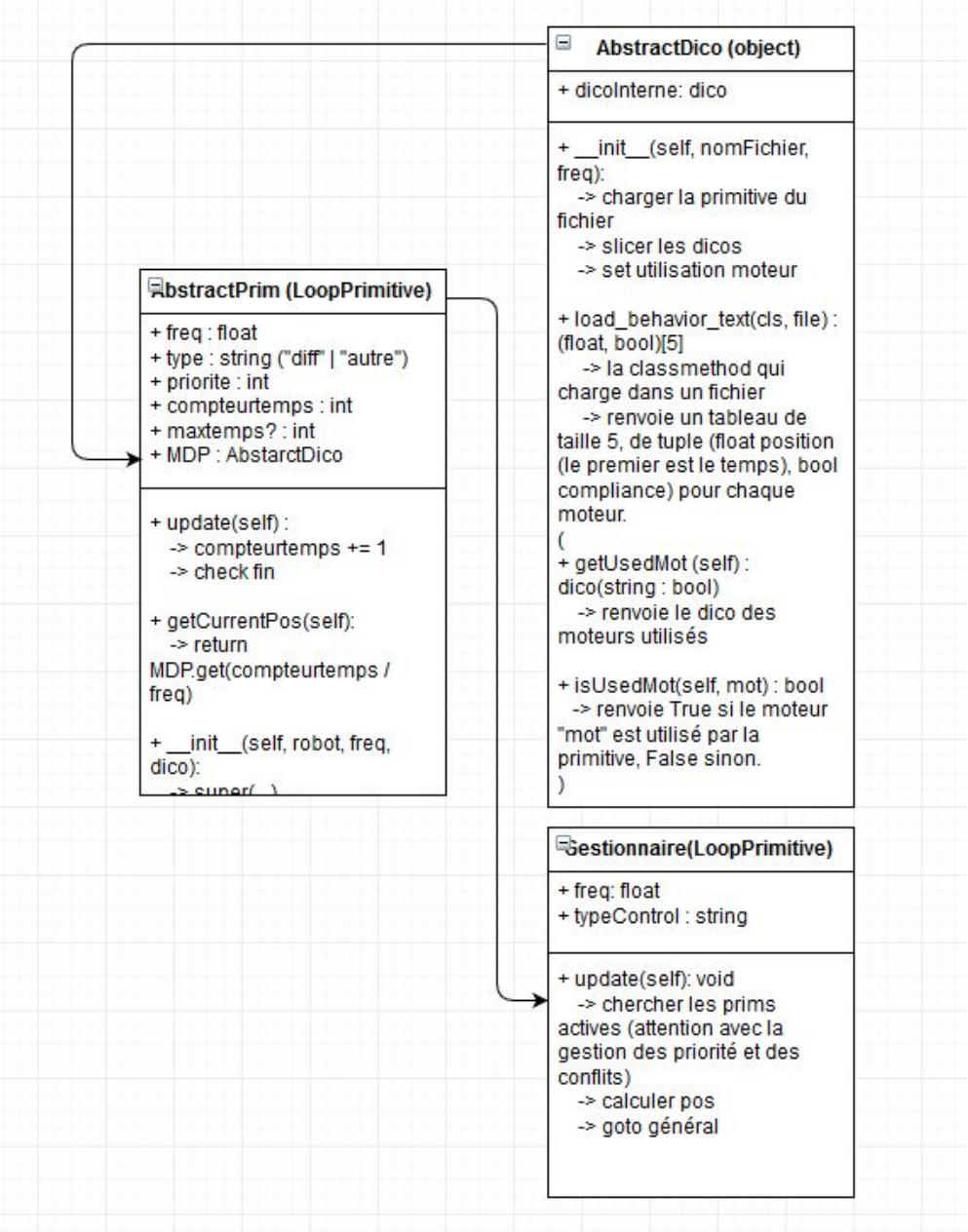
Avec l'ajout du suivi de visage nous nous sommes rendu compte d'un problème : notre robot ne pourrait rien faire d'autre tant qu'il suivait un visage. Ce qui pose réduit grandement sa possibilité d'interagir. Nous nous sommes donc fixé comme objectif de le rendre multitâche. D'où la création d'un gestionnaire centralisé de primitives.

En effet pypot permet déjà de superposer les primitives mais d'une manière qui ne nous satisfait pas, nativement pypot fait la moyenne de tous les ordres envoyés. Dans notre cadre nous voulons que les actions se fassent dans leur ensemble ou pas du tout, un mouvement qui représente la peur ne représente plus grand chose si des mouvements parasites viennent se greffer dessus.

Nous voulions ainsi un gestionnaire centralisé pour gérer l'addition de mouvement et la priorité des différentes primitives. Ce gestionnaire pourrait aussi permettre de donner une dynamique particulière aux mouvements (gérer l'accélération et décélération via les controlleur de pypot) ce qui pourrait la rendre plus vivante : mouvement lent avec peu d'accélération lorsqu'elle est triste à l'inverse très réactive lorsqu'elle est joyeuse.

Implémentation

Voici la structure que nous avons décidé d'adopter :



Le dictionnaire abstrait permet de gérer les liens avec le gestionnaire de fichier, les abstracts primitives sont des primitives fantômes qui envoi au gestionnaire ce qu'elles souhaitent pour le robot. Le gestionnaire se charge lui de s'occuper de la priorité et de savoir quelles primitives peuvent s'additionner ensembles sans poser de problèmes.

Résultats

Suite à la définition clair de la structure l'implémentation s'est fait avec peu de problème. Le seul problème notable a été la fréquence d'envoi d'ordre aux moteurs, une fréquence trop

élevés donnait des mouvements saccadés et nous avons dû plutôt envoyer les ordres suivant une liste de “checkpoints”.

ORM

L’ORM a été créé suite au travail effectué avec le fichier de sauvegarde des primitive. Il en suit les idées et se comporte de la même manière mais dispose de fonctionnalités additionnelles et permet d’ajouter et de modifier plus facilement les différentes primitives de la lampe.

Le fichier de sauvegarde s’est vu modifié pour pouvoir inclure les nouvelles fonctionnalités des primitives du gestionnaire. Comme avant, une primitive se stocke par son nom, mais désormais la valeur est un dictionnaire contenant le lien vers le fichier de comportement (avec la clé “file”), la priorité de la primitive (avec la clé “prior”), et peut contenir la liste des moteurs utilisés par la primitive (avec la clé “usedmot”). Toutes ces options dans le fichier de configuration sont nécessaires au bon fonctionnement des AbstractPrimitives pour le gestionnaire et en plus permettent d’ajouter efficacement de nouvelles fonctionnalité à une primitives car l’on peut stocker différentes options dans le fichier d’auto-chargement de l’ORM.

L’ORM est également indépendant de la lampe, ce qui permet de pouvoir créer de nouvelles primitives et les ajouter via une fonction de classe sans avoir besoin de créer un robot lampe.

Suivi du visage

Contexte

L'objectif initial était de donner au robot des fonctionnalités et comportements qui lui permettrait d'interagir avec une personne. Ainsi, le suivi de visage représente une grande opportunité pour rendre la lampe dynamique et surtout lui donner la capacité de réagir instantanément à la personne en face d'elle.

Concrètement, la lampe possède une tête (ampoule) avec une webcam. L'objectif est que la lampe se déplace de façon à toujours garder en visuel la personne devant elle. La lampe instaure ainsi une relation avec la personne.

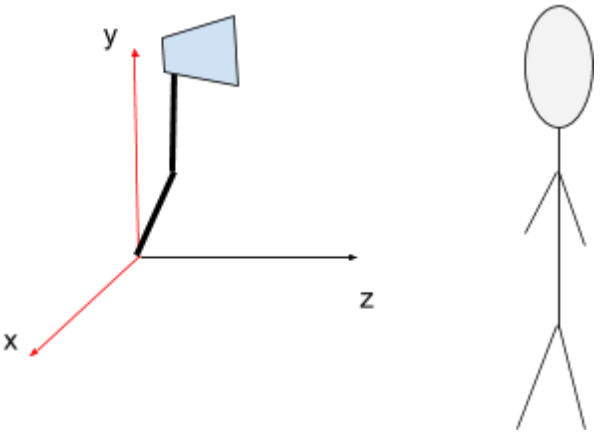
Mise en place

La 1ère étape de la mise en place du Face Follower est la mise en place d'une reconnaissance de visage. A partir, d'un retour image (c'est à dire l'affichage de ce qu'observe la webcam), nous devons être capable de trouver un visage. D'autre part, cela ça doit s'effectuer non pas seulement sur une image mais sur une vidéo (retour image en continu). Pour répondre à nos contraintes, nous avons choisi d'utiliser OpenCV et son module Face Recognition using Haar Cascades.

Globalement, ce module permet la reconnaissance de pattern spécifique au visage, en particulier ici les yeux. En effet, on retrouve sur tous les yeux des contrastes caractéristiques à cette zone. On en déduit ensuite la position de la tête. La méthode a été d'abord testé avec nos webcam personnelles. Les résultats étant satisfaisant nous avons opté pour celle-ci. Il est important de noter que la détection de visage n'est pas parfaite. Ainsi, un visage incliné peut ne pas être détecté, de même un visage peut être détecté sur une zone où il n'y a rien (ex: sur une prise électrique).

Ensuite, il a fallu choisir la façon dont on voulait commander la lampe ou plutôt quels mouvements on voulait donner à la lampe. Plusieurs possibilités de difficultés différentes s'offraient à nous. En effet, on a d'abord envisagé la solution de la cinématique inversée, c'est-à-dire donner un point dans l'espace atteignable de la lampe est lui demander de s'y positionner grâce aux différents moteurs. Cependant, la lampe est capable de se déplacer dans l'espace, c'est- à-dire en 3D. Or, pour détecter le visage, nous utilisons une image, qui est donc seulement 2D. Cela rend donc difficile le choix du point à atteindre dans l'espace 3D, en plus de la nécessité de mettre en place la cinématique inversée.

Pour résoudre ce problème, nous avons choisi de diviser les mouvements du robot selon uniquement deux directions : horizontal et vertical. On oublie ainsi la composant “profondeur” par rapport à la personne en face de la lampe.



Ainsi on utilise un moteur de la tête (MOT_head) et un moteur de la base (MOT_bas_rot) pour régler le mouvement horizontal selon x et un moteur de la tête (MOT_head_arm) et du corps (MOT_arm) pour le mouvement vertical selon y.

Enfin, le dernier choix à faire pour réaliser le FaceFollower est celui des commandes envoyées aux moteurs. En effet il faut lui donner des déplacements selon deux axes (x et y). On a donc fait le choix de calculer le vecteur entre le centre de l’image et le centre du visage. Ce vecteur est ensuite normalisé et donnant en entrée pour les modifications des positions des moteurs. Globalement, pour obtenir un mouvement le plus naturel possible, on fait bouger en priorité les moteurs de la tête, de même qu’un humain bougerait a priori la tête pour suivre un petit déplacement. Puis lorsque le déplacement est sur les positions extrêmes des moteurs on utilise le déplacement du “corps” de la lampe.

Résultats

Le résultat du suivi de visage est satisfaisant et la lampe arrive, pour des mouvements assez lents. En fait les déplacements se font de manière plutôt naturelle, et sans grand à-coup. La vitesse de réponse de la lampe est également satisfaisante Cependant, il convient de remarquer que pour une utilisation optimale du suivi de visage, la lampe doit être légèrement surélevée pour ainsi que la webcam puisse percevoir correctement et centrer le visage en face d’elle.

Concernant les pistes pour améliorer la fonction FaceFollower de la lampe, il est possible d’envisager de modifier la répartition des déplacements sur les moteurs pour que le mouvement de suivi ait l’air encore plus “humain”. Ensuite, il peut être intéressant de donner la possibilité à la lampe de se stabiliser quand le visage est presque au centre de l’image de la webcam. Cela permet à la lampe d’éviter d’osciller autour de la position, étant donné

qu'un humain n'est pas figé entièrement même lorsqu'il n'effectue a priori aucun mouvement.

Enfin, une fois le visage hors de la webcam, la lampe est incapable de savoir quelle direction prendre. Il pourrait donc être utile de soit lui faire reprendre sa position initiale (ou position de repos), soit lui donner la possibilité de continuer son mouvement pour éventuellement retrouver le visage plus loin.

Difficultés rencontrées

Différents points techniques nous ont posé des problèmes lors de la réalisation de ce FaceFollower. Tout d'abord, la différence entre les deux robots (réel et simulé), même en étant infime, était problématique dans l'optique d'une commande précise de la lampe. Nous avons dû trouver des solutions pour que ces différences matérielles n'aient aucun impact sur le software et que quelque soit le robot que l'on utilisait les commandes restaient semblables. C'est une des raisons qui nous à poussé à avoir deux fichiers de configuration différents pour les deux robots.

Ensuite, lorsque l'on donne aux moteurs un petit déplacement (exemple : position initiale x ➡position finale x+△x), ceux-ci peuvent avoir des difficultés à l'atteindre. Ainsi le moteur MOT_arm n'effectuait aucune commande envoyée car le déplacement était trop petit pour être réalisable. Nous avons donc ajuster le coefficient devant le déplacement à faire pour que celui-ci soit plus grand et donc réalisable par le moteur.

Un autre problème majeur rencontré est celui de la webcam utilisée par la bibliothèque open-cv. En effet nous avons eu quelques soucis pour détecter la caméra de la lampe sur plusieurs ordinateurs. En outre ce problème n'étant pas majeur nous avons décidé de ne pas pousser l'investigation plus loin, préférant travailler sur des tâches plus importantes.

Conclusion

La lampe a pour finalité d'être utilisée dans des expériences mettant en scène la lampe et une personne volontaire. Ainsi, le but de notre travail sur ce projet était de réaliser des fonctionnalités qui permettent de rendre les expérimentations plus simples à réaliser et de facilement ajouter de nouveaux mouvements à la lampe. C'est dans cette optique que le suivi de visage a été créé, afin de permettre aux expérimentateurs de suivre le visage des volontaires de manière plus simple que ce qui était fait. Nous nous sommes également axés sur la partie automatisation des différents chargement afin d'augmenter l'ergonomie tout en séparant les différents modules les uns des autres pour les rendre plus robustes.

La lampe nous a permis de nous confronter à quelques problèmes des systèmes concrets comme le temps réel ou le passage de la simulation au robot réel mais aussi des problèmes plus généraux de l'informatique comme la standardisation (des fichiers de configurations ainsi que le stockage des mouvements).

Ce projet nous a permis de nous familiariser avec les notions clés de la robotique, tout en gardant une application concrète dans le cadre du projet Psyphine. En plus des connaissances techniques acquises durant cette année, nous avons également fait l'expérience du travail en groupe, et donc de la répartition des tâches, de l'organisation des séances de travail,etc. Ainsi, le projet nous a apporté à la fois des compétences techniques et un apprentissage sur la gestion de projet.

Annexe 1 : Fiche de présentation du projet

Développement d'une architecture de contrôle pour une lampe robotisée

Projet Mines 2A

Laboratoire : Loria **Equipes** : Larsen/Biscuit
Encadrants : Alain Dutech (`alain.dutech@loria.fr`), Yann Boniface, Amine Boumaza.

Résumé

Dans le cadre du projet Psyphine [1] nous disposons de deux lampes robotisées inspirées du projet Pinokio [2]. L'objectif du projet est de participer au développement d'une architecture de contrôle pour commander ces lampes. Les outils préconisés seront les bibliothèques poppy et openCV.

Sujet

Le groupe Psyphine réuni des chercheurs de différentes disciplines (Philosophie, Psychologie, Sciences Du Langage, Intelligence Artificielle, Neurosciences et Anthropologie) pour explorer la possibilité d'interaction entre des humains et des créatures artificielles. Pour conduire ses expériences, le groupe dispose de deux lampes robotisées, inspirées du projet Pinokio. A ce stade, ces lampes sont télécommandées et le groupe souhaite les rendre autonomes. Le but de cette proposition est de participer au développement des éléments logiciels qui permettront cette autonomie.

Ce travail se décompose en plusieurs étapes :

- Prise en main logicielle et matérielle de la lampe Psyphine. En particulier il faudra s'approprier la bibliothèque Pypot [3] qui permet de communiquer avec les moteurs (Dynamixel) des robots.
- Faire de cette lampe une "créature" façon "Poppy" [4].
  - Le projet Poppy propose une infrastructure logicielle de contrôle robotique.
  - L'interface "créature" permet de modéliser un robot et de disposer ainsi de commandes de haut-niveau.
- Proposer un module de commande par cinématique inverse
  - Adapter et traduire le modèle 3D du robot au format Poppy
  - Tester et valider cette cinématique inverse
- Proposer un module de "commandes miroir"
  - La mapulation manuelle commande la seconde lampe
- Proposer un module de suivi de visages
  - S'approprier la bibliothèque OpenCV [5]
  - Détecter un visage en utilisant les filtres et les fonctions proposés par OpenCV.
  - Proposer un algorithme de commande pour le suivi de visage.
- Proposer un module de détection d'émotions
  - A partir des visage détectés, extraire certains traits caractéristiques de comportements ou d'émotions (mouvement des sourcils, forme de la bouche, ...)
  - Associer ces caractéristiques à des comportements (frustration, joie, ...)

Cadre de travail

Ce travail sera principalement encadré par les informaticiens du LORIA du groupe Psyphine et se fera en interaction avec les autres membres du groupe. Le développement se fera au LORIA avec Linux + python + C++.

Liens

1 https ://psyphine.hypotheses.org/
2 https ://vimeo.com/53476316
3 http ://poppy-project.github.io/pypot/
4 https ://www.poppy-project.org/en/
5 http ://opencv.org/