Modèle de rétine neuromimétique basé sur l'utilisation de logiciels Open-source

Pierre Albiges[1], Laurent Perrinet[2]

[1]Aix Marseille Université, M1 NIC, Marseille, France [2]CNRS, INT, Marseille, France

Motivations du projet

Biomimétisme

Qu'est-ce que la biomimétique ?
La biomimétique consiste en la **reproduction de fonctions et de capacités des systèmes biologiques dans des systèmes artificiels** afin d'améliorer leurs performances.
L'objectif est donc de s'inspirer de ce que la sélection naturelle a permis d'obtenir au cours de l'évolution pour résoudre des problèmes d'ingénierie limitant les systèmes artificiels actuels : résistance et adaptabilité aux contraintes environnementales, collecte et utilisation de l'énergie dans l'environnement, autonomie dans les déplacements et les prises de décisions, ...

L'ingénierie biomimétique est donc souvent utilisé dans le développement de technologies de pointe comme la robotique, mais aussi dans la recherche car il permet de mieux comprendre les mécanismes biologiques étudies.

Dans le cadre de ce projet, c'est la **rétine et le système visuel animal** qui servent de modèle.

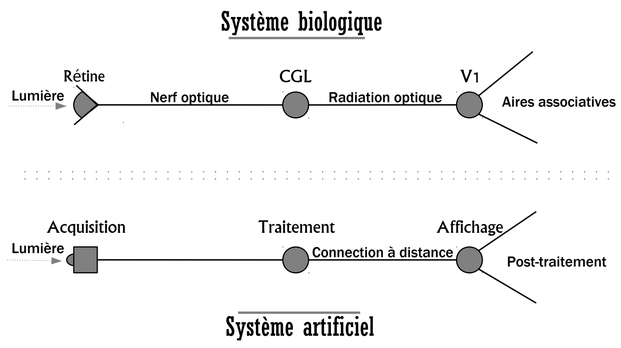

FIGURE 1 : Comparaison schématique du fonctionnement de OpenRetina et du système biologique équivalent

Low-Tech

L'une des limites actuelles de l'ingénierie touche à la **puissance de calcul** des systèmes utilisés.

L'ingenerie "high-tech" cherche à créer les systèmes les plus performants possibles sans se soucier de la consommation d'énergie et de puissance de calcul.
En réponse, l'ingénierie "low-tech" se concentre sur l'intégration de programmes dans des systèmes artificiels possédant une faible puissance de calcul ou une autonomie énergétique limitée, tels que les systèmes embarqués (drones, systèmes robotisés autonomes) ou répondant à des contraintes bien précises, tels que les rétines artifielles.

Dans le cadre de OpenRetina, c'est une **ingénerie low-tech et Open Source** qui est visée, afin de permettre son adaptation et son utilisation par le plus de systèmes possible.

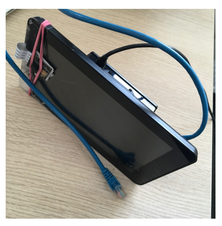

FIGURE 2 : Une RaspBerry Pi 2 et sa picamera, principal support physique utilisé pour OpenRetina

Méthodes computationnelles

Le projet OpenRetina est composé d'un ensemble de programmes majoritairement écrits dans le language de programmation Python.
Ces scripts réalisent une série de taches qu'on peut décrire en 4 catégories, ci-dessous :

Grab

Récupérer une image ou une vidéo à partir d'un fichier video, d'une caméra locale ou d'une caméra distante (connectée à une RaspBerry Pi).
Cette étape est proche d'une prise d'image classique mais sert surtout à récupérer les **informations enregistrées individuellement dans chaque pixel**.

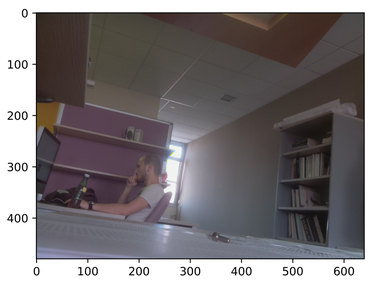

FIGURE 3 : Image brute capturée par une caméra distante, représentée sous forme d'un numpy array

Process

Réaliser un traitement sur chaque frame que comporte l'image, afin de **réduire la quantité d'informations transmises et d'accélerer la temporalité de la perception, sans perte de résolution**.

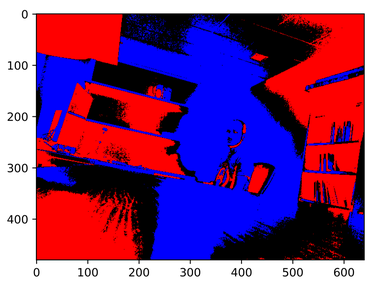

FIGURE 4 : Image après un traitement révélant les contrastes, représentée sous la forme d'un nympy array

Network

Transmettre les informations (requêtes ou données) entre les acteurs au travers un **protocole ssh** et l'utilisation de la **bibliothèque zeroMQ**.

Display

Afficher l'image nouvellement traitée afin de contrôler les résultats.

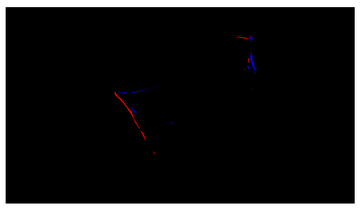

FIGURE 5 : Image après un traitement multi-couche ne révélant que les contours des objets en mouvement (extrait d'un affichage vidéo)

Avancées apportées au projet

— Récupération des informations captées par la caméra pour reconstruire un fichier vidéo permettant de réaliser des **traitements hors-ligne et à postériori**

— Réalisation d'**étalonnages** pour contrôler l'influence de certains paramètres sur les performances du système

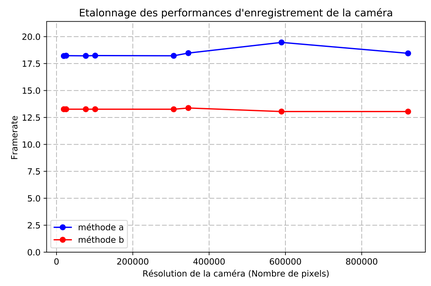

FIGURE 6 : Graphique issue de l'étalonnage comparant deux méthodes de calcul de la vitesse de capture

— Complexification du programme pour construire une **architecture en couches** : divise la charge de travail entre plusieurs acteurs, localise les erreurs et facilite la compréhension et les modifications

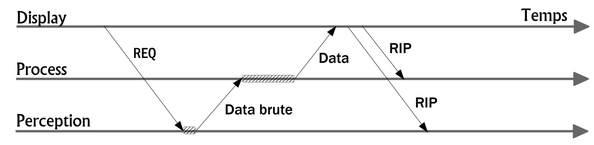

FIGURE 7 : Représentation schématique du fonctionnement dans le temps d'une architecture en 3 couches

Compétences acquises

— Methodologie : approfondissement des **connaissances en Python** (notamment en programmation orientée objet) et en LaTeX, utilisation et maintien d'un **logbook**, utilisation de **git** pour gérer efficacement des versions, communications à distance via les **protocoles ssh** et **zeroMQ**

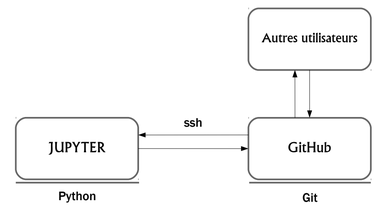

FIGURE 8 : Représentation schématique de la méthodologie utilisée pour OpenRetina

— Méthodes de traitement d'images et de vidéos

— Compétences d'ingénierie et de **reverse-engineering**.

— Connaissances neuroscientifiques dans les domaines de la perception et du traitement de l'information visuelle.

Présenté le 15 juin 2017 à l'Université d'Aix-Marseille, Marseille, France, dans le cadre de l'évaluation de l'unité d'enseignement ENSNEBU11