Université de La Rochelle

Département Informatique

Stage de fin d'études

Émilien BARBAUD

du 23/04/2019 au 12/06/2019

Laboratoire MIA

MATHEMATIQUES IMAGE APPLICATIONS
UNIVERSITÉ DE LA ROCHELLE

Enseignants tuteurs et référents :

M. Laurent MASCARILLA

M. Renaud PÉTERI

Rapport consultable
Année universitaire 2018-2019

Table des matières

Introduction

Le traitement de l'image est une discipline impliquant l'informatique et les mathématiques appliquées. Après l'acquisition et la numérisation, l'enjeu est d'automatiser l'étude d'images afin d'en extraire de l'information. Le traitement de l'image est un domaine en pleine expansion, on peut citer le milieu industriel avec l'automatisation des contrôles qualités, ou encore les futures voitures autonomes, se basant en grande partie sur l'interprétation des images fournies par des caméras autour du véhicule. Le traitement de l'image est un secteur de l'informatique qui m'intéresse. Le parcours « Vision pour les Objets Connectés » m'a donné l'opportunité d'effectuer mon stage dans ce domaine avec le MIA[1], un laboratoire de recherche de l'université de La Rochelle. Mes deux enseignants référents font partie de ce laboratoire, il s'agit de Laurent MASCARILLA et de Renaud PÉTERI.

Mon stage est axé sur l'utilisation d'une caméra intelligente nommée « JeVois[2] », et il est séparé en deux projets. Le premier consiste à réaliser une application facilitant la découverte et la programmation de cette caméra pour les futurs élèves. Le deuxième est la réalisation d'un système permettant de jouer à un jeu de plateau, le Puissance 4, contre une intelligence artificielle, la caméra « JeVois » permettant la reconnaissance des jetons via sa capture vidéo, mais aussi d'anticiper les coups à jouer grâce à son processeur relativement puissant. Ce deuxième projet est réalisé en collaboration avec un collègue stagiaire, Rémi DUJARDIN, et un enseignant, Bernard BESSERER, qui s'occupent de toute la partie mécanique du projet. Ce jeu fera office de démonstration lors des journées portes ouvertes de l'université.

Pour travailler, j'ai à ma disposition les ordinateurs de l'université, mais pour des raisons de simplicité, je préfère utiliser mon ordinateur personnel : les logiciels dont j'ai besoin ont déjà été installés dessus pendant ma formation. Je développe les deux projets en Python3[3], ce qui facilite le portage du code sur la caméra connectée qui possède un interpréteur, autrement dit, pas de problème de compilation croisée. J'utilise bien entendu divers modules Python, notamment OpenCV, dont l'utilisation avec la caméra « JeVois » est très optimisée.

[1] Le laboratoire MIA, Mathématiques, Image et Applications, site web : mia.univ-larochelle.fr
[2] JeVois Smart Machine Vision Camera, site web : jevois.org
[3] Langage de programmation, site web : www.python.org

Laboratoire MIA

Présentation

Le MIA est un laboratoire pluridisciplinaire qui regroupe des enseignants-chercheurs de quatre sections CNU (« Mathématiques », « Mathématiques appliquées et applications de mathématiques », « Informatique » et « Génie informatique, automatique et traitement du signal »)

Si l'activité scientifique du MIA s'appuie sur des travaux théoriques tels qu'on les imagine traditionnellement dans un laboratoire de mathématiques, sa structuration transverse est pensée pour répondre aux grands enjeux applicatifs actuels et ce au travers de deux domaines d'expertise : « Mathématiques et Image Numérique » et « Mathématiques Environnement et Science de la Vie ».

Membres

Voici les quelques membres du MIA avec qui je suis en relation durant mon stage :

Laurent MASCARILLA, Maître de Conférences (HDR)

Laboratoire MIA et département d'Informatique

Tél. 05.46.45.87.43. Courriel : laurent.mascarilla@univ-lr.fr

Site web : https://sites.google.com/site/laurentmascarilla/home

Renaud PÉTERI, Maître de Conférences

Laboratoire MIA et département d'Informatique.

Tél. 05.46.45.72.19. Courriel : renaud.peteri@univ-lr.fr

Site web : http://pageperso.univ-lr.fr/rpeteri/

Bernard BESSERER, Maître de Conférences (HDR)

Laboratoire MIA et département d'Informatique.

Tél. 05.46.45.82.32. Courriel : bernard.besserer@univ-lr.fr

LinkedIn : https://fr.linkedin.com/in/bernard-besserer-ba64807/fr

La liste complète des membres du MIA : http://mia.univ-larochelle.fr/content/view/75/95

Caméra JeVois

La caméra « JeVois » est en réalité un micro-ordinateur équipé d'une caméra, permettant à la fois la capture et le traitement des images, sa puissance de calcul est un peu supérieure à un Raspberry Pi 3 (cf. annexe 1). Elle est petite et légère, elle peut se loger presque partout : 17 g pour 28 cm^3. Comme évoqué précédemment, la « JeVois » est au cœur de mon stage, c'est un élément principal pour mes deux projets.

Spécifications techniques

Figure 1 Fonctionnalités et caractéristiques de la caméra "JeVois"

Processeur et processeur graphique

Une « JeVois » possède un CPU[4] quadri-cœur cadencé à 1,34 GHz, ainsi qu'un GPU[5] double cœur permettant de rendre le traitement sur les images plus rapide. La caméra possède une version d'OpenCV optimisée pour utiliser au mieux le processeur graphique.

Optique

Différents capteurs optiques peuvent être installés sur la caméra, comme par exemple un capteur à obturation globale, permettant de s'affranchir des déformations sur les objets en mouvement.

Les caméras que j'ai à disposition possèdent le capteur basique de 1,3 MP, donnant accès à ces standards vidéo[6] :

- SXGA (1280 x 1024) jusqu'à 15 i/s
- VGA (640 x 480) jusqu'à 30 i/s
- CIF (352 x 288) jusqu'à 60 i/s
- QVGA (320 x 240) jusqu'à 60 i/s
- QCIF (176 x 144) jusqu'à 120 i/s
- QQCIF (88 x 72) jusqu'à 120 i/s

Figure 2 Caméra "JeVois"

[4] Central Processing Unit, le processeur.
[5] Graphic Processing Unit, le processeur graphique.
[6] Liste de tous les standards vidéo : www.gouvenelstudio.com/homecinema/resolution.htm

Flux vidéo via USB

Une « JeVois » a été conçue pour pouvoir générer facilement un flux vidéo via son port USB, le même flux vidéo que produirait une webcam par exemple. Cela a l'avantage de ne pas nécessiter de pilotes spéciaux pour être reconnu sur un ordinateur. De cette manière, on peut facilement récupérer le flux vidéo de la caméra sur n'importe quel ordinateur.

Communication série

La caméra peut communiquer en série via son USB ou via un micro-connecteur série supportant une logique de 3,3V ou 5,5V, ce qui permet de la connecter sur n'importe quel microcontrôleur avec UART[7].

Le micro-connecteur série possède 4 broches : i/oREF, GND, RX, TX. Attention, bien que le fil i/oREF soit de couleur rouge, il est impossible d'alimenter la caméra de cette manière. i/oREF et GND servent seulement à indiquer la tension pour la logique de la communication série.

La caméra « JeVois » communique en TLL-level, et n'est pas compatible RS-232.

Les baudrates[8] supportés sont : 110, 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 38400, 57600, 115200, 230400, 460800, 921600, 1000000, 1152000, 1500000, 2000000, 2500000, 3000000, 3500000, 4000000.

JeVois Inventor

Les concepteurs de la « JeVois » ont également créé un environnement de développement pour la caméra, il permet d'envoyer le code d'un simple clic et offre aussi la possibilité de faire du débogage, d'accéder à la carte micro SD via l'USB, de créer de nouveau modules, etc. Vous trouverez des captures d'écran du logiciel en annexe (cf. annexe 5).

Ce logiciel est cependant en phase de développement, il est actuellement disponible en version 0.4.0. De nombreux bogues sont présents et l'interface n'est pas toujours très intuitive, mais ce qui manque essentiellement deux choses qu'un EDI[9] se doit d'avoir aujourd'hui : l'auto-complétion et l'auto-indentation.

Pour pallier à ce problème je développe la plupart du temps sur PyCharm[10], et fais une copie de mon code pour l'envoyer sur la « JeVois ».

[7] Universal Asynchronous Receiver Transmitter : émetteur-récepteur asynchrone universel
[8] Nombre de bits par secondes transmis lors de la communication série.
[9] Environnement de Développement Intégré, IDE en anglais
[10] EDI développé par JetBrains, site web : www.jetbrains.com/pycharm

Projet 1) Interface JeVois – PC

Ce premier projet consiste en la création d'un logiciel facilitant la prise en main de la caméra « JeVois » par les étudiants novices. Afin qu'ils puissent facilement y apporter des modifications, le programme sera codé en Python3, ce langage étant entré au programme de la licence. Ce logiciel permettra de tester les différentes fonctionnalités de la caméra connectée avec une interface graphique.

Vous trouverez en annexe des captures d'écrans du logiciel fini (annexes 2 et 3).

Conception

Maquette

Dans un premier temps, j'ai réalisé une maquette[11] de la disposition des éléments graphiques sur l'écran. Cela me permet de savoir vers quoi je me dirige lors de la programmation.

Figure 3 Maquette de l'application d'interface avec la "JeVois"[12]

L'écran sera séparé verticalement en deux. Du côté gauche, nous pourrons voir le rendu en direct de la caméra. Du côté droit, nous pourrons ajuster certains paramètres en fonction du module choisi, par exemple, pour un module de seuillage, ajuster la valeur de ce seuil. Nous aurons aussi la possibilité de voir le journal de bord[13] de l'application, qui indiquera par exemple, si la caméra s'est bien connectée. Une barre de menu se situera en haut de l'interface pour notamment, choisir le périphérique d'entrée et le module à charger.

[11] On trouve souvent le terme anglais « wireframe » en UX Design
[12] Maquette réalisée sur : wireframe.cc
[13] Plus communément appelé par son anglicisme « log »

Diagramme de classes

Voici un diagramme de classes simplifié pour comprendre le fonctionnement du système de modules :

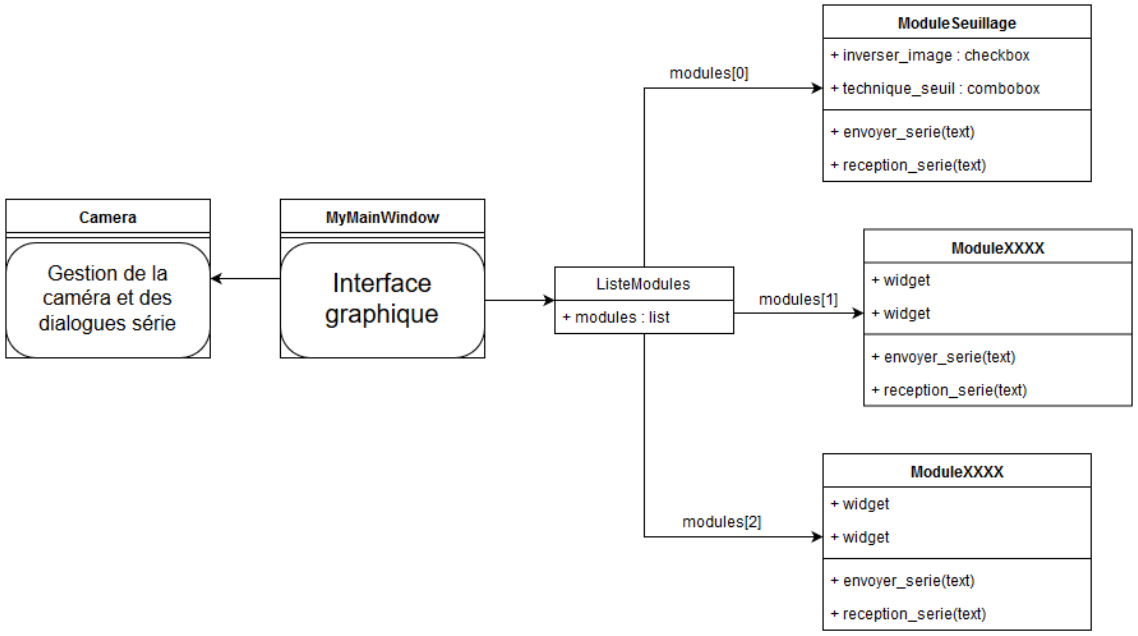


Figure 4 Diagramme de classes simplifié

Les étudiants pourront, à leur guise, ajouter une classe de module pour créer le leur. Ils n'auront pas à s'occuper de l'interfaçage avec la caméra « JeVois », il leur suffira de déclarer les éléments de l'interface et de renseigner le protocole série à utiliser. (cf. Annexe 4)

Le système a été pensé pour que les modules soient le plus indépendants de l'application possible. Ainsi, seuls quelques attributs et méthodes sont obligatoires, pour chaque module.

Attributs requis :

- `layout` : la disposition générale des éléments dans le module
- `jevois_resolution` : la taille et la fréquence des images définies sur la « JeVois »
- `name` : le nom du module
- `widget_list` : la liste de tous les éléments graphiques utiles pour le module

Méthodes requises :

- `send_serial(text)` : le « slot » pour envoyer un message série à la « JeVois »
- `serial_received_jevois(text)` : le « signal » émis lorsqu'un message est reçu.

Technologies utilisées

PyQt5 / Qt5

PyQt permet de lier Python3 aux bibliothèques de Qt[14]. Pour ce projet, Qt est utilisé pour gérer l'interface graphique, ainsi que pour les threads[15].

Cette bibliothèque implémente un système de « signaux » et de « slots », qui sont bien pratiques pour gérer l'évènementiel et l'asynchrone. Ils permettent de communiquer entre les objets. Les « signaux » sont émis par un objet lorsque son état change et que cela intéresse un autre objet, ce sont des évènements. Les « slots » sont des fonctions appelées lorsqu'un « signal » est émis. La liaison entre « signaux » et « slots » est établie grâce à la fonction « connect ». Attention tout de même, pour que cela fonctionne, il est nécessaire que les objets concernés héritent de la classe « QObject ».

Qt implémente son propre système de thread, utilisable grâce à la classe « QThread ». Il est nécessaire d'utiliser les threads de Qt, car l'interface graphique ne prends pas en compte les threads de base de Python, sinon on risque de se retrouver avec une interface graphique gelée à la moindre action demandant des ressources.

Communications séries

Le logiciel lancé sur un ordinateur communique avec la caméra « JeVois » grâce à une liaison série passant par l'USB. Le module Python « serial » est utilisé sur l'ordinateur. Sur la caméra, un module Python spécifique nommé « libjevois » inclut les fonctions nécessaires à la communication série.

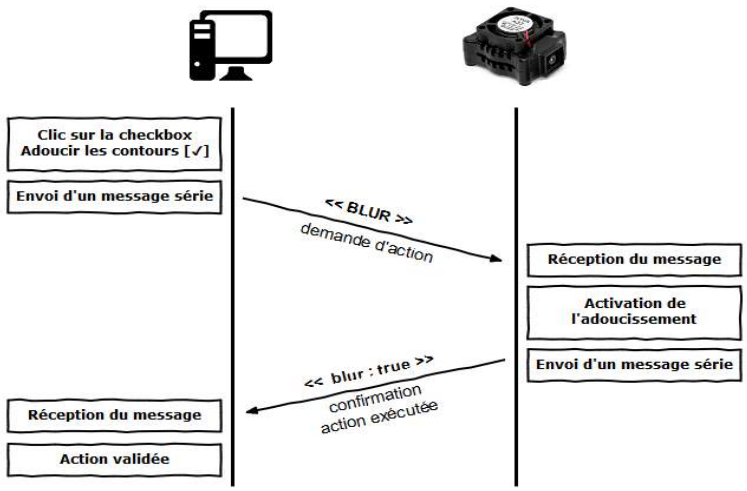


Figure 5 Diagramme de séquence d'une communication série entre le logiciel et la "JeVois"

[14] Qt est une API orientée objet développée en C++, site web : www.qt.io
[15] Anglicisme, souvent traduit par « processus »

Projet 2) Puissance 4

Présentation

La seconde partie de mon stage est consacrée à l'automatisation d'un jeu de plateau : le Puissance 4. Ce projet est réalisé en collaboration avec un collègue stagiaire, Rémi DUJARDIN, et un membre du laboratoire MIA, Bernard BESSERER. Il consiste à réaliser un mécanisme permettant de jouer au Puissance 4 contre un robot intelligent. Ce projet pourra être montré en démonstration lors des journées portes ouvertes par exemple, mettant en œuvre deux des mineures proposées à l'université de La Rochelle : une portée sur la « computer vision » et l'autre sur les microcontrôleurs.

Le jeu

Puissance 4 (Connect 4 en anglais) est un jeu de stratégie combinatoire abstrait. Le but du jeu est d'aligner une suite de 4 pions de même couleur sur une grille comptant 6 rangées et 7 colonnes. Chaque joueur dispose de 21 pions d'une couleur. Tour à tour, les deux joueurs placent un pion dans la colonne de leur choix, le pion coulisse alors jusqu'à la position la plus basse possible dans ladite colonne à la suite de quoi c'est à l'adversaire de jouer. Le vainqueur est le joueur qui réalise le premier un alignement (horizontal, vertical ou diagonal) consécutif d'au moins quatre pions de sa couleur. Si, alors que toutes les cases de la grille de jeu sont remplies, aucun des deux joueurs n'a réalisé un tel alignement, la partie est déclarée nulle.

Source : Wikipédia – le 28/05/19

Figure 6 Le jeu de plateau Puissance 4, par Hasbro

Répartition des tâches

Pour ce projet, les tâches ont été réparties comme suit : je m'occupe de la partie détection des jetons et intelligence artificielle, et mon collègue travaille sur la partie mécanique du jeu avec Bernard BESSERER. Cette répartition fait sens, car nous travaillons chacun dans le domaine de notre mineure.

Schéma général du jeu

Un robot placeur de jetons, fabriqué spécialement pour cet usage, sera installé au-dessus du plateau de jeu. Il sera muni d'une tête mobile qui viendra déposer les jetons au bon endroit. Nous veillerons à ce que cette tête se décale hors du plateau pour laisser le joueur humain placer son pion. Ce robot sera commandé par un microcontrôleur.

Une caméra « JeVois » sera placée du côté opposé au joueur afin de pouvoir capter l'image du plateau et d'en déduire les pions qui y sont présents. Après analyse, l'intelligence artificielle décidera d'envoyer une commande au microcontrôleur afin que celui-ci place un pion au bon endroit sur le plateau.

Quelques LEDs RGB seront présente à côté de la caméra. Elles serviront d'éclairage pour le plateau, mais pourront aussi indiquer au joueur que c'est à son tour de jouer, via un système de couleur.

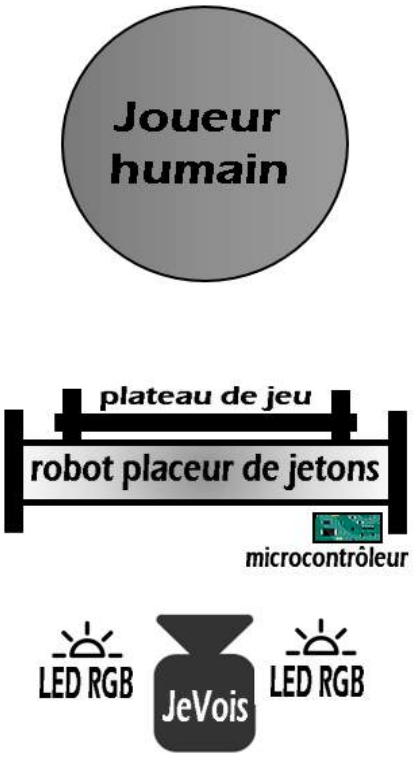


Figure 7 Schéma du système, vue de dessus

Robot placeur de jetons

Comme évoqué précédemment, le robot est réalisé par Bernard BESSERER et est programmé par mon collègue Rémi DUJARDIN. Le support de base est un rail muni d'une tête mobile, tiré d'une ancienne imprimante. Le système pour prendre un jeton est réalisé avec un ancien disque dur. Cela a l'avantage d'avoir un coût très faible, mais aussi de recycler des vieux appareils qui seraient partis à la déchèterie.

Le robot est piloté par un microcontrôleur. Pour le prototype, il s'agit actuellement d'un MSP430G2253[16] développé par Texas Instrument[17]. La version finale du projet prévoit plutôt un MSP430F169[18], car l'université en possède un plus grand nombre.

[16] Fiche technique : www.ti.com/product/MSP430G2253
[17] Entreprise mondiale de semi-conducteurs, site web : www.ti.com
[18] Fiche technique : www.ti.com/product/MSP430F169

Traitement de l'image

OpenCV

OpenCV (pour Open Computer Vision) est une bibliothèque graphique libre, initialement développée par Intel, spécialisée dans le traitement d'images en temps réel.

Il existe une implémentation d'OpenCV en Python nommée OpenCV-Python. Tout le code n'a pas été traduit en python, il s'agit plutôt d'un « emballage » Python autour du code original en C++. Lors d'un appel d'une fonction OpenCV en Python c'est en réalité le code C++ qui est exécuté en arrière-plan. Cela présente deux avantage : le code est très lisible car écrit en Python mais il est aussi rapide à exécuter que du code C++.

OpenCV sera utilisé dans ce projet pour effectuer les traitements sur les images.

Acquisition

La caméra « JeVois » est positionnée en face du plateau de jeu Puissance 4, elle capte la scène avec une résolution de 640 × 480 pixels (VGA), à une fréquence de 20 images par secondes maximum. Cette résolution a été choisie car elle offre un bon compromis entre performance et qualité de l'acquisition. En effet, prendre plus de pixels aurait surchargé le processeur de la caméra, et en prendre moins aurait réduit la capacité à détecter les jetons.

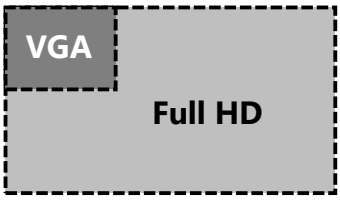


Figure 8 Taille d'une image en VGA comparé à une image en Full HD

Détection du plateau

Tout d'abord, la caméra va essayer de trouver le plateau de jeu s'il est devant elle. Pour cela, on se sert des trous : en effet, tous les plateaux de Puissance 4 ont quarante-deux trous (7 colonnes × 6 lignes). Afin de détecter ces trous, on utilise la transformée de Hough circulaire, plutôt efficace, elle est implémentée dans OpenCV.

Si la caméra arrive à trouver 42 cercles rangés à peu près en grille, elle en déduit que le plateau est bien positionné devant elle. De par la position de tous les cercles on peut ensuite déduire la position exacte du plateau.

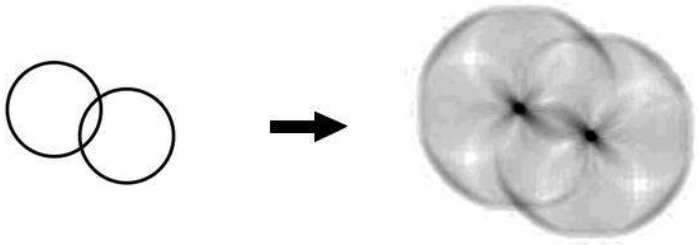


Figure 9 Transformée de Hough circulaire sur deux cercles

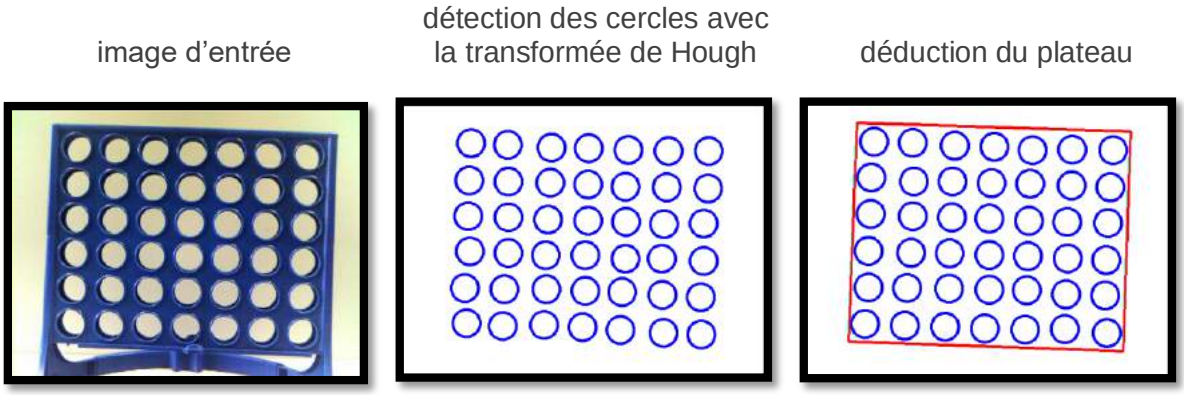


Figure 10 Étapes de détection du plateau de jeu

Correction de perspective

Bien que la caméra ait pour but d'être fixée en face du plateau, il est en réalité presque impossible de la placer parfaitement. Une fois le plateau de jeu détecté, il est alors possible de redresser l'image afin que celle-ci soir plus facile à traiter.

Pour redresser l'image, on peut effectuer une correction de perspective. OpenCV propose une telle fonction, il suffit de lui donner les coordonnées des quatre coins du plateau, que nous avons récupéré à l'étape précédente.

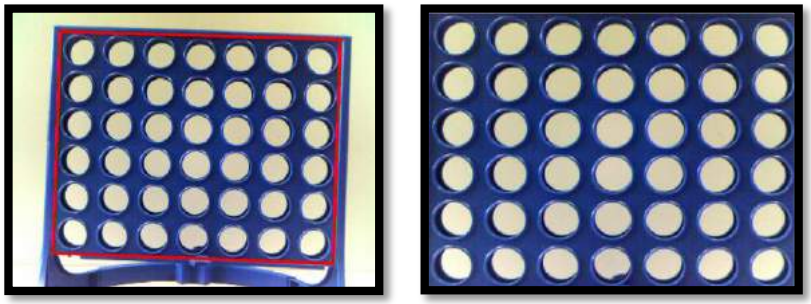


Figure 11 Image avant et après correction de perspective

ROI des cases

Grâce à l'image corrigée, nous allons pouvoir définir des zones d'intérêts (ROI) pour la recherche des jetons. Partant du principe que tous les trous sont de taille identique et qu'ils sont répartis sur une grille régulière, il suffit de quadriller le plateau avec des cercles, afin que ceux-ci deviennent les ROI des jetons potentiels.

Cette méthode permet d'éviter les imprécisions de la transformée de Hough, ici les ROI sont presque parfaites.

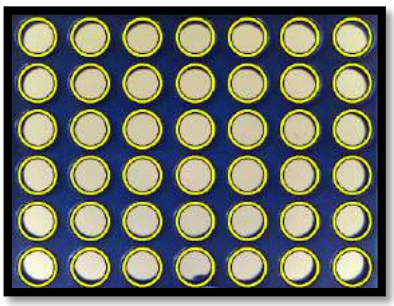


Figure 12 ROI des jetons (entourées en jaune)

Détection des jetons

Maintenant que nous avons toutes les ROI des cases, il faut trouver un moyen de savoir s'il y a un pion dedans et, si oui, sa couleur. Afin de tester si un paramètre (par exemple la teinte, la luminosité…) est discriminant, il est possible de le tester sur un plateau prévu à cet effet avec à la fois des cases vides, des jetons rouges et des jetons jaunes.

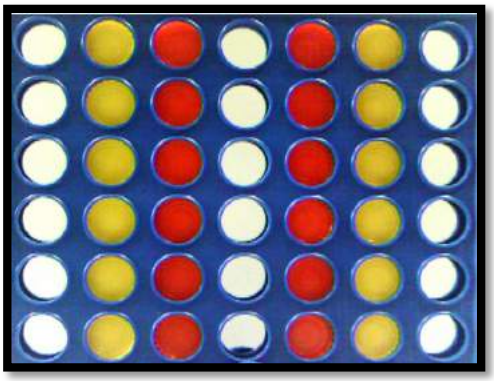


Figure 13 Plateau de test

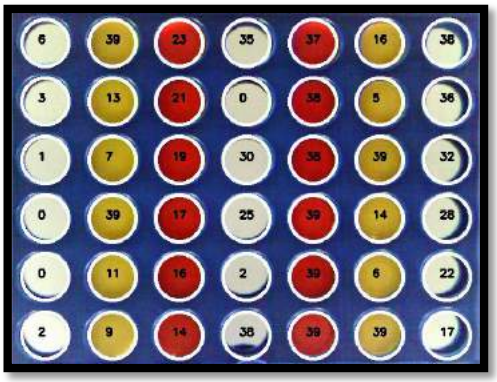


Figure 14 Affichage des valeurs pour un paramètre, celui-ci n'est pas discriminant

Cette méthode de test permet rapidement de se rendre compte si un paramètre est intéressant ou non. Il ne faut pas oublier de faire varier l'arrière-plan pour simuler des conditions réelles.

N. B. les clichés de ce rapport ont tous été pris avec un arrière-plan clair et uni afin d'avoir de belles images. Cela ne reflète pas la réalité, où l'arrière-plan est très diversifié.

Filtre par contraste

Un pion est toujours de couleur unie. On peut alors savoir qu'une case est vide si l'on observe un fort contraste dans sa ROI. Pour effectuer ce test, on calcule l'écart type de la valeur des pixels en niveau de gris. S'il est supérieur à un certain seuil, la case est vide.

Attention tout de même, on ne peut pas conclure directement à la présence d'un pion si l'écart type est faible. En réalité il peut très bien y avoir un mur de couleur unie à l'autre bout de la pièce derrière le plateau.

Filtre par saturation et luminosité

Un pion est très coloré : sa valeur de saturation est proche du maximum. On peut donc rejeter les ROI qui contiennent des pixels peu colorés, on sait que ce sont des cases vides.

De même avec la luminosité : si une ROI est très sombre, c'est que la case lui correspondant ne contient aucun pion.

Sélection par teinte et saturation

Les pions étant de couleur rouge ou jaune, il est tout à fait possible de les différencier par leur teinte. D'une part, les pions sont de couleur unie, donc on peut s'attendre à avoir un écart type sur la teinte faible. On peut ensuite effectuer une moyenne de la teinte sur la ROI pour différencier un pion rouge d'un pion jaune. Cependant, nous devons faire face à deux contraintes techniques :

- Les variations de lumière entrainent intrinsèquement une variation de la teinte. En effet, en fonction du type d'éclairage, des ombres ou du ciel dehors, les composantes de la lumière ne sont pas les mêmes.
- Dans l'espace HSV[19] le rouge se situe aux bornes de la roue chromatique, la valeur de la teinte peut donc être très différente pour deux rouges similaires à l'œil.

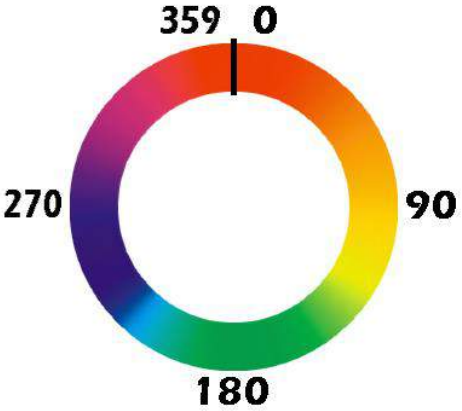


Figure 15 Roue chromatique, valeurs de teinte de 0 à 359

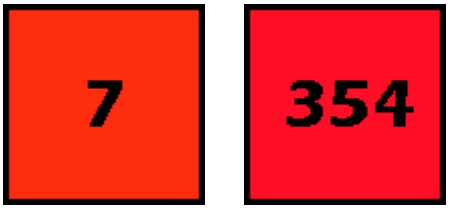


Figure 16 Deux rouges semblables visuellement, avec des valeurs de teintes très différentes

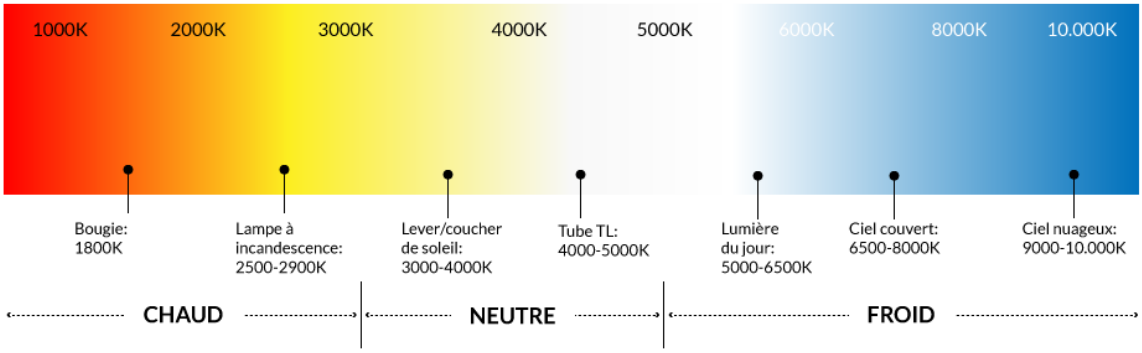


Figure 17 Variation de la température de couleur en fonction du type d'éclairage[20]

[19] HSV : Hue, Saturation, Value (Teinte, Saturation, Luminosité). Espace colorimétrique.
[20] Source du visuel : www.ledsky.be/shop/fr/temperature-couleur

Afin de s'affranchir du problème de passage d'un côté à l'autre de la roue chromatique, on utilise une fonction spéciale simulant un cercle chromatique pour le calcul de la moyenne et de l'écart type[21]. De plus, cette fonction pondère la teinte avec la saturation, afin d'être moins sensible aux changements de luminosité.

Le problème lié à la température de couleur de l'éclairage devrait être résolu sur le projet final, lorsque les LEDs seront installées pour éclairer le plateau.

Détection générale

Voici un diagramme reprenant l'ensemble des étapes de détection d'un jeton :

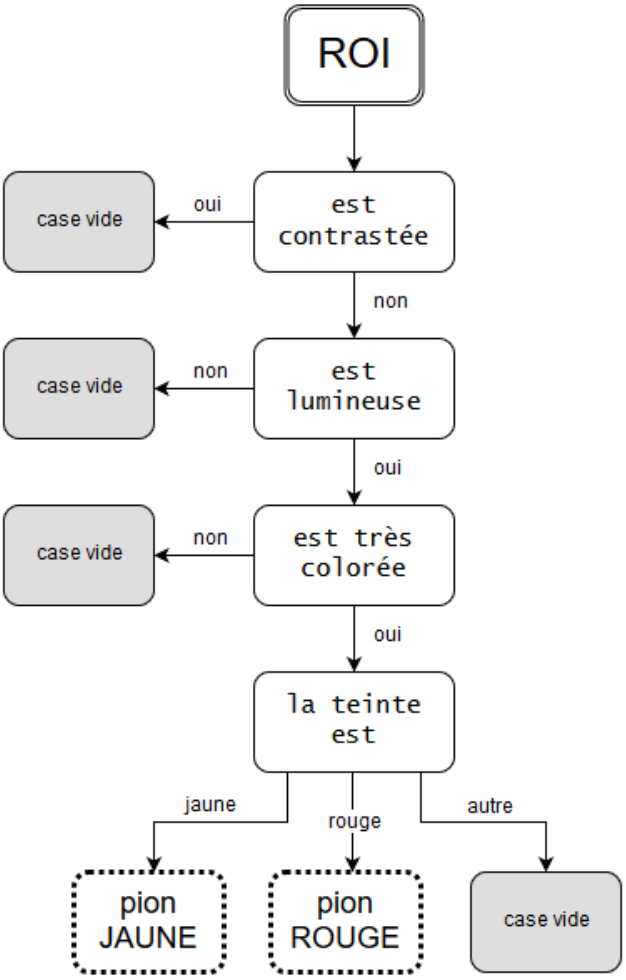


Figure 18 Diagramme schématisant la détection d'un jeton selon plusieurs critères

Vous trouverez en annexe des exemples de détection sur différents plateaux de jeu.

(cf. annexe 6, annexe 7)

[21] Source du code : github.com/jhamrick/python-snippets/blob/master/snippets/circstats.py

Simulation du plateau

Structure de données

Le plateau est enregistré en mémoire avec une matrice de dimension 6 par 7. La valeur 0 a été choisie pour une case vide, la valeur 1 pour un pion rouge, et la valeur 2 pour un pion jaune. Cela permet d'avoir une structure de données très légère et facile à manipuler.

```
[[0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 2. 0. 0.]
 [0. 0. 0. 1. 2. 1. 0.]
 [0. 0. 2. 2. 2. 1. 0.]
 [0. 0. 1. 2. 1. 1. 2.]
 [0. 0. 2. 1. 1. 2. 1.]]
```

Figure 19 Plateau de jeu sauvegardé en mémoire

Visuel de synthèse

Pour développer ce projet, il fallait un retour visuel sur le plateau. Il était possible d'afficher la matrice de façon textuelle comme ci-dessus, mais il était difficile de s'imaginer le plateau. D'où la création de ce plateau virtuel :

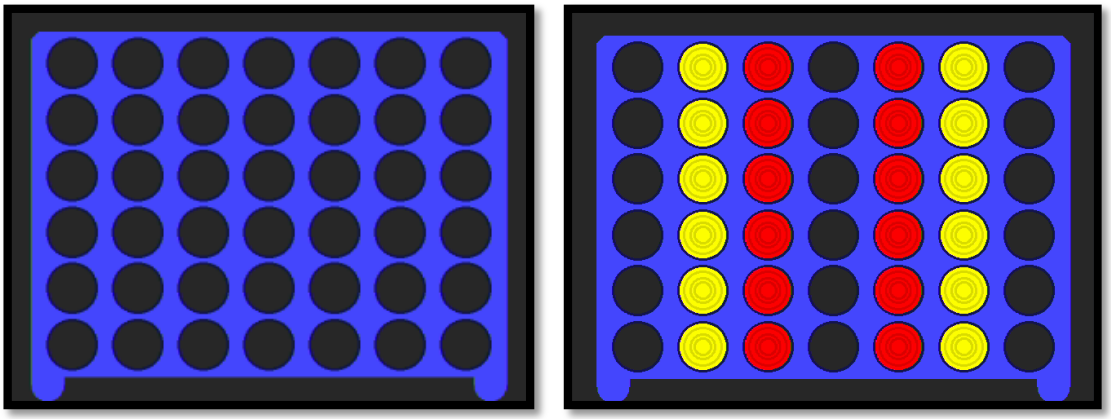


Figure 20 Plateaux virtuels générés par la "JeVois". Plateau vide à gauche, plateau de test à droite.

Ses formes plutôt simples lui permet d'être généré rapidement par la « JeVois », il offre un bon retour visuel sur le plateau et permet de voir facilement si la détection des jetons fonctionne correctement.

Intelligence artificielle

Minimax

L'algorithme minimax[22] est un algorithme qui s'applique à la théorie des jeux pour les jeux à deux joueurs à somme nulle et à information complète[23], consistant à minimiser la perte maximum, c'est-à-dire dans le pire des cas.

Dans le cas du Puissance 4, l'algorithme va anticiper les coups pouvant être joués en créant un grand arbre, où chaque coup est un nœud.

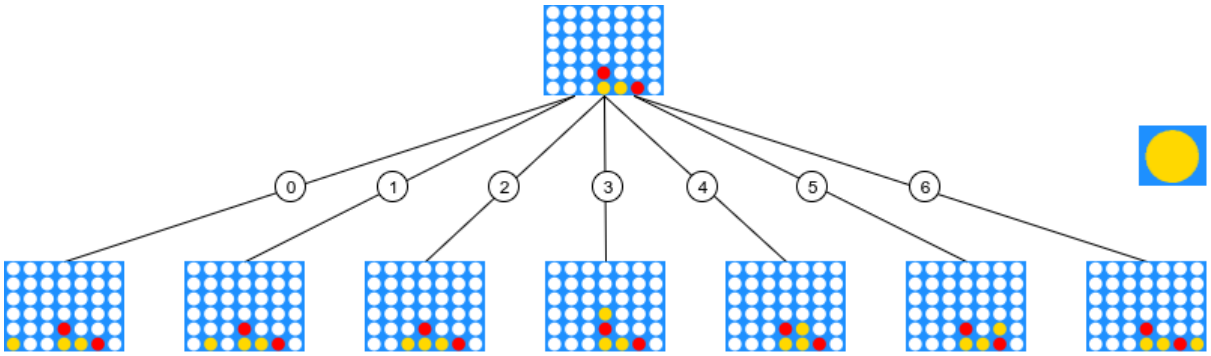


Figure 21 Simulation de coup joué par le joueur Jaune

À chaque coup anticipé, l'algorithme va attribuer un score permettant de savoir si ce placement est avantageux ou non : si le coup avantage l'adversaire, le score sera négatif, si le coup nous avantage, le score sera positif.

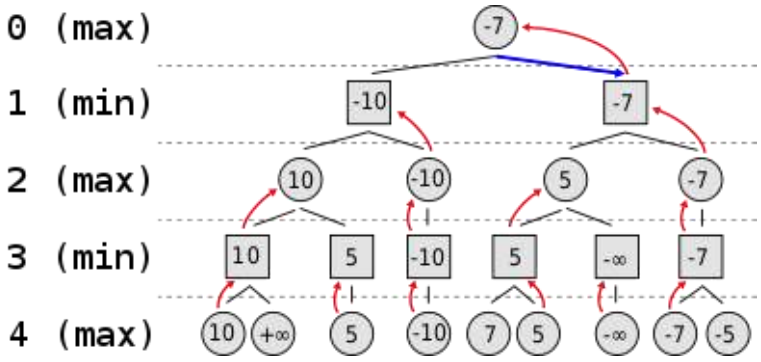


Figure 22 Exemple d'arbre minimax provenant d'un autre jeu

Sur l'illustration ci-dessus, le meilleur coup à jouer possède un score de -7, cela signifie que ce n'est pas une situation avantageuse : même en jouant le meilleur coup possible, on approche l'adversaire de la victoire.

[22] Pour plus de détails : fr.wikipedia.org/wiki/Algorithme_minimax

[23] Jeux à somme nulle et information complète, voir : fr.wikipedia.org/wiki/Théorie_des_jeux

L'algorithme du minimax est récursif. Si aucune limite lui est donné il peut anticiper le jeu jusqu'à la fin de la partie. Cependant, le code devra être exécuté sur le petit processeur de la « JeVois » et nous ne pouvons pas espérer simuler l'entièreté de l'arbre.

On peut faire une rapide estimation du nombre de possibilités à simuler en fonction de la profondeur de l'algorithme :

Tours d'avance	*Profondeur de l'arbre*	*Calcul*	*Possibilités*
1	2	7 choix, 2 itérations -> 7^2	49
2	4	7 choix, 4 itérations -> 7^4	2 k
3	6	7 choix, 6 itérations -> 7^6	117 k
4	8	7 choix, 8 itérations -> 7^8	5,7 M
5	10	7 choix, 10 itérations -> 7^{10}	282 M

Le tableau ci-dessus nous montre les limites de cet algorithme : la taille de l'arbre évolue de façon exponentielle par rapport aux coups d'avance.

Élagage alpha-béta

L'élagage alpha-béta est une technique permettant de réduire le nombre de nœuds évalués par l'algorithme minimax, et donc d'optimiser sa vitesse d'exécution et la quantité de mémoire utilisée.

L'algorithme minimax effectue une exploration complète de l'arbre de recherche jusqu'à un niveau donné. L'élagage alpha-béta permet d'optimiser grandement l'algorithme minimax sans en modifier le résultat. Pour cela, il ne réalise qu'une exploration partielle de l'arbre. Lors de l'exploration, il n'est pas nécessaire d'examiner les sous-arbres qui conduisent à des configurations dont la valeur ne contribuera pas au calcul du gain à la racine de l'arbre. Dit autrement, l'élagage alpha-béta n'évalue pas des nœuds dont on est sûr que leur qualité sera inférieure à un nœud déjà évalué.

Dans le jeu du Puissance 4, alpha est le meilleur score courant sur le chemin vers la racine, pour nous. Béta est aussi le meilleur score courant vers la racine mais pour l'adversaire.

Les valeurs alpha et bêta sont mises à jour chaque fois que l'algorithme voit une nouvelle valeur des enfants. Les alpha et béta sont transmis aux autres enfants, si on trouve une valeur supérieure à notre bêta actuel, ou inférieure à notre alpha actuel, nous pouvons supprimer le sous-arbre entier, puisque nous sommes sûrs que le chemin optimal ne passera pas par là.

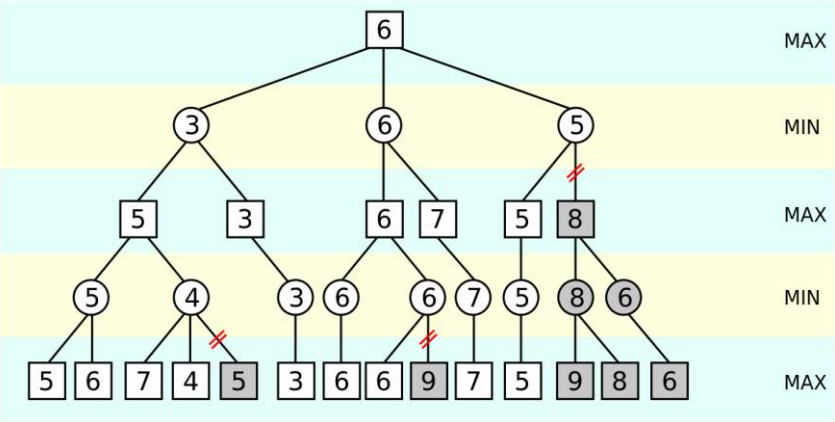


Figure 23 Exemple d'élagage alpha-béta sur un autre jeu

Tests réels

Le nombre de possibilités étant drastiquement réduit avec l'élagage alpha-béta, il devient beaucoup plus rapide à exécuter. Il est temps de tester l'algorithme sur la caméra « JeVois ».

Après plusieurs tests effectués, une récursivité limitée à une profondeur de trois, semble être un bon compromis entre temps de calcul et réussite. Cela signifie que l'algorithme anticipe trois coup sur l'adversaire. Cela peut paraitre peu, mais cela suffit à battre un joueur humain dans la plupart des cas, pour un temps de calcul de quelques dixièmes de secondes.

Suite du projet

Ce projet est n'est pas encore terminé, il reste à connecter la « JeVois » au robot placeur de jetons. Actuellement la caméra affiche à l'écran l'emplacement où elle souhaite jouer, c'est à nous de placer son jeton (cf. Annexe 8). Voici comment devrait se dérouler la suite du projet :

Communications série

Toute la partie communication entre la caméra et le microcontrôleur qui gère le robot reste à programmer. Les échanges série seront des petites chaînes de caractères classiques. On peut imaginer la caméra envoyant « START » et « STOP » pour le début et la fin de partie, puis un chiffre indiquant dans quelle colonne le robot doit placer son jeton. Le microcontrôleur reverra un « feedback » à la caméra pour lui confirmer que les actions se sont bien déroulées. On pourrait aussi communiquer entre chaque tour, afin d'avoir un contrôle plus précis, notamment sur les LEDs RGB, qui par un jeu de couleur pourraient orienter le joueur.

Finalisation

Il restera ensuite à finaliser le projet, faire les derniers ajustements nécessaires. Peut-être aussi faire quelques améliorations et optimisations. Ce projet me semble être sur la bonne voie, il devrait être terminé à temps pour la fin du stage.

Conclusion

Ces quelques semaines de stage furent ma deuxième expérience professionnelle dans le domaine de l'informatique. Cela m'a permis de découvrir le laboratoire MIA de l'université de La Rochelle mais aussi de mettre mes connaissances au service de la collectivité.

Tout au long du stage, j'ai eu l'occasion d'approfondir mes connaissances en « Computer Vision », notamment pour le traitement de l'image. J'ai pu mettre en pratique les bases apprises durant mon année à l'université. Cette expérience n'a pas été enrichissante uniquement sur le plan des connaissances, elle m'a également fait prendre de l'autonomie et de la rigueur dans mon travail.

Ce stage s'inscrit pour moi dans la continuité de mon cursus, de par mon parcours en informatique embarquée que j'ai suivi à l'IUT, mais aussi avec la mineure « Vision pour les Objets Connectés » suivie cette année. Cela m'encourage à poursuivre dans cette voie pour la suite de mes études.

Sources

Programmation

Algorithme optimisé pour l'IA du Puissance 4, par Gilles Vandewiele :
towardsdatascience.com/creating-the-perfect-connect-four-ai-bot-c165115557b0

Algorithme optimisé pour l'IA du Puissance 4 avec code source, par Keith Galli :
github.com/KeithGalli/Connect4-Python

Fonction pour calculer une moyenne correcte sur la teinte, par Jessica B. Hamrick :
github.com/jhamrick/python-snippets/blob/master/snippets/circstats.py

OpenCV

Site officiel :
opencv.org

Documentation Python :
opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_setup/py_intro/py_intro.html

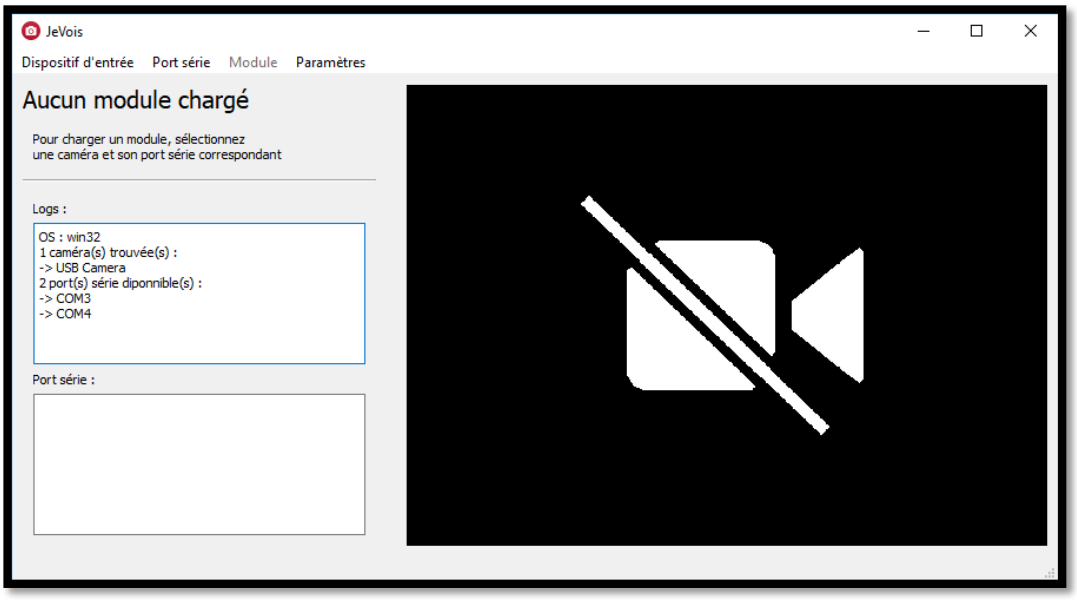
JeVois

Site officiel :
jevois.org

Annexes

	JeVois A33	Raspberry Pi 3 Model B
CPU	Quad-core ARM Cortex A7	Quad-core ARM Cortex A53
CPU clock	1.35 GHz stable under full load.	Up to 1.2 GHz depending on load and CPU temperature.
GPU	Dual-core MALI 400	VideoCore IV
DRAM	256 MB DDR3-1600	1 GB LPDDR2-900
Used RAM while running typical app	~70 MB RAM used while running Visual Attention Demo.	~280 MB RAM used while running Visual Attention Demo.
Storage for O.S. & data	MicroSDHC slot. Supports 50 MHz mode (up to 25 MByte/s).	MicroSDHC slot
Camera	Built-in 1.3MP. Supported resolutions: SXGA (1280 x 1024): up to 15 fps VGA (640 x 480): up to 30 fps CIF (352 x 288): up to 60 fps QVGA (320 x 240): up to 60 fps QCIF (176 x 144): up to 120 fps QQVGA (160 x 120): up to 60 fps QQCIF (88 x 72): up to 120 fps Supported camera pixel types: YUYV, Bayer, RGB565. Frame rate continuously adjustable in 0.01 fps increments. Standard camera controls supported, including manual exposure, gain, color.	Not included. Optional USB webcam (about $10 or more) or Raspberry Pi camera ($25 or more).
USB video streaming to a host computer	Up to full isochronous USB 2.0 bandwidth of 24 Mbyte/s. Supported resolutions and fps: Any up to USB bandwidth limit. Supported pixel types: YUYV, Bayer, RGB565, Grey, MJPEG.	Not supported.
Ports & connectors	USB 2.0, Serial (3.3V & 5V compatible)	USB 2.0 x 4, Ethernet, WIFI, HDMI, Audio, Camera, Display, GPIO
Boot time	5 seconds.	33 seconds.
Power (max)	3.5 Watts (includes about 1 Watt for fan) running CPU + GPU + Camera + USB streaming + NEON + integer + floating point burn test	3.75 Watts without fan.
CPU temperature & clock under burn test	Max. 71 C stable at constant 1.34 GHz under full load. No thermal throttling.	Thermal trip at 80 C reached within minutes under load without fan. CPU speed reduction (throttling) activated to maintain 80 C under load.
Performance	Visual Attention demo: 70 fps @ 320x240, no dropped frames w/camera at 60fps.	Visual Attention demo: 56 – 62 fps @ 320x240 (with fan), dropping frames.
Size	39 x 31 x 23 mm (1.5 x 1.2 x 0.9 in) – 28 cc including case and fan.	86 × 57 x 18 mm (3.4 × 2.2 x 0.7 in) – 88 cc Bare board only, not including optional case and fan.
Price	$49 Includes computing unit, camera, case, cooling fan, 4-pin serial port cable.	$35 for computing board Optional: webcam ($10), case ($10), heatsink/fan ($10)

Annexe 1 Tableau comparatif entre une "JeVois" et un Raspberry Pi 3

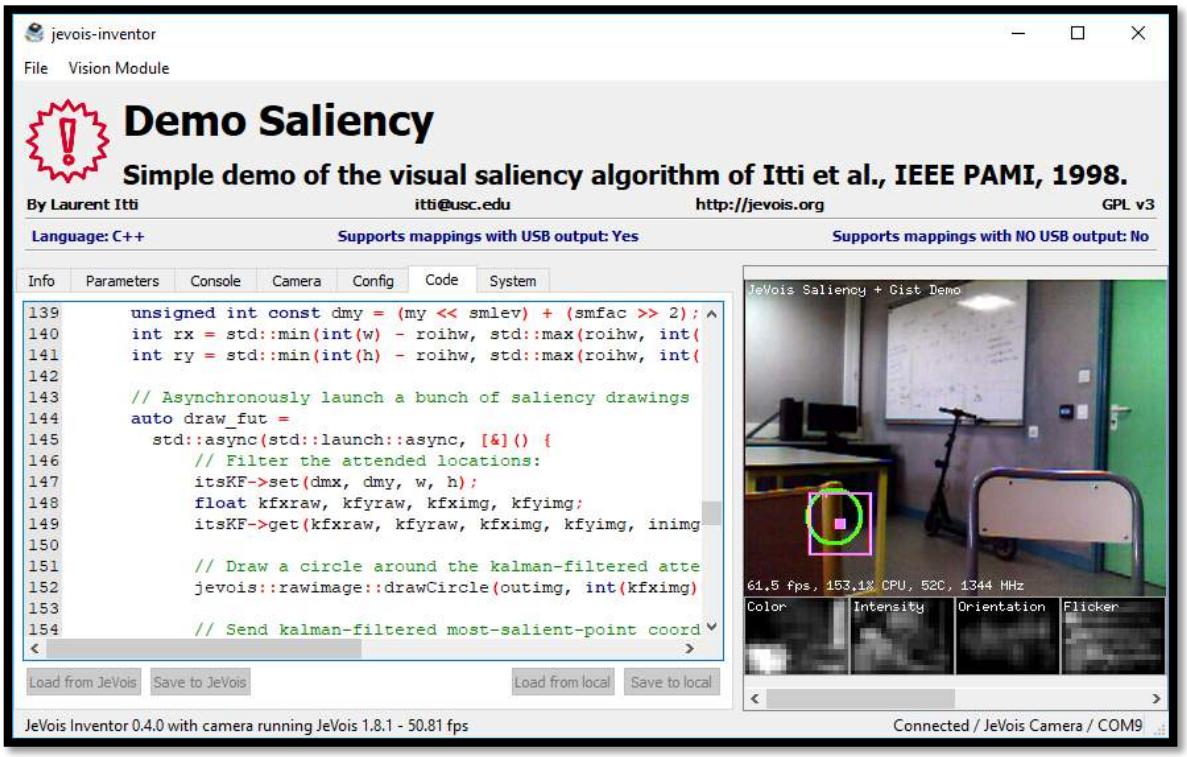


Annexe 2 Logiciel pour l'interfaçage de la "JeVois" au lancement

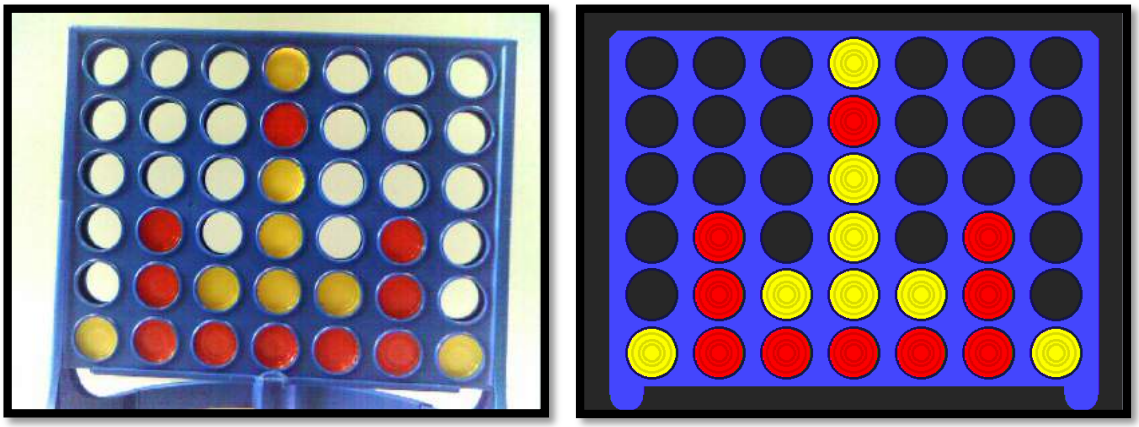
Annexe 3 Logiciel pour l'interfaçage de la "JeVois" avec le module « Seuillage » chargé

```
class JeVoisThreshold(QObject):
    serial = pyqtSignal(str)

    def __init__(self):
        # Basic required attributes
        super().__init__()
        self.layout = QVBoxLayout()
        self.jevois_resolution = (640, 480, 22)
        self.name = "Seuillage"
        self.widget_list = []

        # Specific attributes
        self.slider = QSlider(Qt.Horizontal)
        self.widget_list.append(QLabel("Valeur du seuil :"))
        self.widget_list.append(self.slider)
        self.slider.valueChanged.connect(self.slider_changed)
        self.slider.setMaximum(255)
        self.slider.setValue(127)
        self.widget_list.append(QLabel("Méthode :"))
        self.cb_mode = QComboBox()
        self.cb_mode.currentTextChanged.connect(self.mode_changed)
        self.cb_mode.addItem("BINARY")
        self.cb_mode.addItem("BINARY_INV")
        self.cb_mode.addItem("OTSU")
        self.widget_list.append(self.cb_mode)
        self.is_blurred = QCheckBox("Adoucir l'image")
        self.is_blurred.stateChanged.connect(self.blur_changed)
        self.widget_list.append(self.is_blurred)

    def slider_changed(self):
        self.send_serial(str(self.slider.value()))

    def mode_changed(self):
        self.send_serial(self.cb_mode.currentText())
        if self.cb_mode.currentText() == "OTSU":
            self.slider.setDisabled(True)
        else:
            self.slider.setEnabled(True)

    def blur_changed(self):
        self.send_serial("BLUR")

    # Required, emit signal to send with serial
    def send_serial(self, text):
        self.serial.emit(text)

    # Required, slot for jevois serial return
    def serial_received_jevois(self, text):
        pass
```

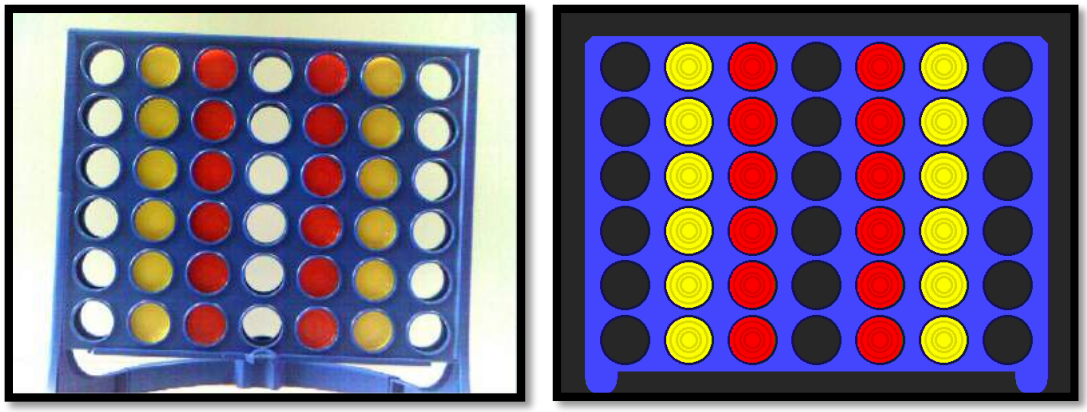
Annexe 4 Code Python du module de seuillage pour l'interfaçage avec la "JeVois"



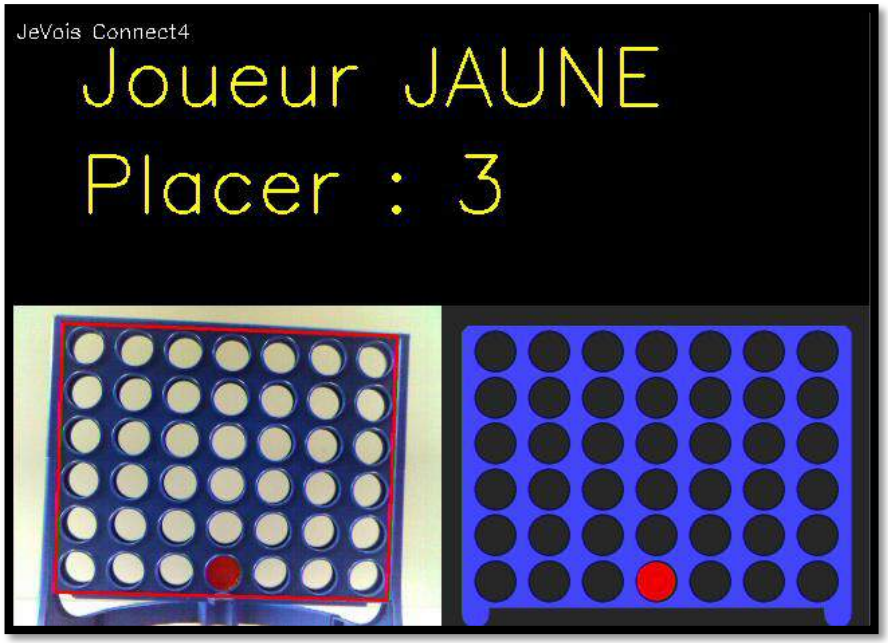
Annexe 5 Logiciel JeVois Inventor avec le module "Demo Saliency" chargé



Annexe 6 Plateau filmé par la "JeVois", suivi de l'image générée grâce à la détection des jetons



Annexe 7 Plateau de test filmé par la "JeVois", suivi de l'image générée grâce à la détection des jetons



Annexe 8 L'IA indique la colonne où placer le jeton