UNIVERSITÉ DU MANS

TOM DE PASQUALE

Rapport final

Reconnaissance de motif et localisation 2D en temps réel

Encadrant : K. Raoof

LE MANS, MARS 2023

Table des matières

1 Introduction

Ce projet est un projet de dernière année en école d'ingénieur·es à l'ENSIM. L'objectif était de se familiariser avec les systèmes d'intelligence artificielle (IA) embarqués sur micro-contrôleurs. Ces systèmes étant de plus en plus utilisés dans des domaines tel que le transport ou l'industrie, il est intéressant de découvrir comment concevoir de tels systèmes, voir quels sont les contraintes et comment les résoudre. Plus précisément, ce projet se base sur la découverte du micro-contrôleur *Arduino Nicla Vision* [5] qui intègre – en plus d'un micro-processeur – une caméra intégrée. C'est donc pourquoi la problématique de ce projet est la suivante : **Comment créer une IA embarquée sur l'*Arduino Nicla Vision*, capable de reconnaître et de situer en temps réel un motif présent dans une image ?**
Pour atteindre cet objectif, il a fallu tout d'abord rechercher quelles étaient les techniques existantes et possibles à intégrer sur un micro-contrôleurs. Dans un second temps, il a fallu créer l'IA pour qu'elle réponde le mieux possible aux objectifs souhaités malgré les différents contraintes qui existaient. Enfin, il a fallu intégrer cette IA sur le micro-contrôleur pour la voir fonctionner en temps réel.

Ce projet a donc permit – outre la découverte de l'IA embarqué et de la détection de motifs – la création d'une IA fonctionnelle capable de détecter un motif en temps réel et de le situer dans l'image.

2 Contexte

2.1 Cadre

Ce projet s'inscrit dans le cadre d'un projet d'étude en cinquième année à l'ENSIM. Celui-ci se pose en tant que projet majeur de fin d'études et donne la possibilités aux étudiant·es de s'ouvrir et d'approfondir sur un nouveau sujet. Il a été fait en parallèle d'un semestre en ERASMUS et a été proposé et encadré par M. Raoof. L'enjeu du projet consistait à découvrir ce qu'il était possible de faire en mélangeant intelligence artificielle et micro-contrôleurs. Étant étudiant en informatique et ayant suivi l'option ASTRE (Architecture des Systèmes en Temps Réel Embarqués) il était tout naturel pour moi de réaliser ce projet.

2.2 Besoins

Les micro-contrôleurs font partie intégrante de la vie quotidienne. On peut les trouver partout, de l'industrie jusqu'à dans les jouets pour enfants. Ils sont utilisés dans notre vie de tous les jours et souvent invisibilisés à l'intérieur des objets. Il est donc stratégique de savoir comment les utiliser mais également de se tenir au courant des nouvelles innovations de ce domaine. Depuis quelques années, il existe une nouvelle tendance dans l'industrie appelée "l'usine 4.0". Dans cette usine, chaque étape de la chaîne de production est étroitement contrôlée à l'aide de capteurs et de technologies communicantes. De cette façon, on peut connaître à tout moment l'état du processus de création. Cela nous permet de contrôler la qualité des produits et d'intervenir si un accident à lieu.

Une autre partie de l'innovation qui s'est grandement développée ces dernières années est la reconnaissance d'objets par des machines ou des ordinateurs (*machine vision* en anglais). Celle-ci est devenue un élément très important de l'industrie. L'objectif est d'être capable d'observer, d'apprendre de ce que l'on voit et de déclencher les réactions correspondantes. Grâce à la puissance de traitement rapide intégrée aux micro-contrôleurs, nous pouvons désormais utiliser des techniques d'IA pour faire de la détection en temps réel et embarquée dans des petits systèmes. Ce projet s'intègre donc dans la continuité des tendances d'innovation actuelles.

2.3 Objectifs

Au terme de ce projet, l'objectif est d'être capable de détecter un motif avec la caméra embarquée et de connaître ses coordonnées dans l'espace de l'écran (2D). La détection devrait être possible à distance variable et selon différents angles. Compte tenu de la résolution de la caméra, une distance de quelques mètres est possible. L'angle à lequel le motif peut être détecté doit rester raisonnable et s'apparente à une marge d'erreur. Toute la détection devra être faite grâce à une intelligence artificielle fonctionnant sur le micro-contrôleur. L'intégralité du système doit pouvoir fonctionner de manière isolée. Le motif ainsi que la technique d'IA utilisée est à définir au cours du projet.

En plus de cela, deux rapports (dont celui-ci) sont fourni indiquant l'avancement du projet. L'un à mi-temps et l'autre au terme de celui-ci. Une présentation du travail est effectuée à ces moments là. L'objectif est de rendre compte du travail effectué et de récupérer des avis constructifs sur la voie que prend le projet.

3 Organisation

3.1 Plan de travail

Étant à Barcelone pendant la grande partie du déroulement de ce projet, je n'ai pas pu accorder beaucoup de temps à celui-ci durant cette période. Cependant, finissant mon semestre à l'étranger fin janvier, j'ai pu libérer du temps à cette période là pour finir et compléter ce qu'il fallait pour que ce projet aboutisse et puisse être présenté.

Ce projet a été initialement divisé en plusieurs tâches qui ont permises d'atteindre le résultat souhaité. Celles-ci sont listées sans prendre en compte l'aspect temporel. La temporalité du projet a été définie dans un diagramme de Gantt.

— Réaliser une revue de la littérature sur l'état de l'art dans le domaine.
— Prendre en main le micro contrôleur
— Optimiser le choix du modèle 2D
— Choisir l'algorithme et les outils pour la partie IA
— Implémenter en temps réel sur le processeur ARM de la famille H7
— Tester le système pour évaluer ses performances.

3.2 Méthodologie

La méthode de réalisation du projet a évoluée au fur et à mesure de l'avancement de celui-ci. Dans un premier temps, les tâches étant définies, il a fallut les effectuer chronologiquement. Puis, les tâches pouvant être effectuées en parallèle, il a fallut effectuer en priorité certaines parties et dégager ce qui était le plus important.

Ainsi, chaque tâches est regroupée en carte dans un tableau *Kanban* [7] comportant trois colonnes :

— TODO
— DOING
— DONE

Au cours de la réalisation de la tâche, la carte est déplacée dans la colonne correspondante. Cela permet deux choses : la première est que chaque tâche doit être définie avec précision. En effet, pour qu'une tâche soit considérée comme finie, il faut savoir à l'avance les condition de validation de la tâche. La seconde est que les tâches sont effectuées une par une. Ainsi, on s'assure que le projet avance à chaque étape. Il n'existe pas de tâches en attente pendant que l'on travail sur une autre.

De cette façon, le projet a pu arriver à son terme et un livrable fonctionnant tel que souhaité initialement a été créé. Il existe des pistes pour continuer le projet et améliorer l'IA. Celles-ci seront évoqués à la fin du rapport.

4 État de l'art

4.1 Micro-contrôleurs

L'évolution de l'IA embarquée sur micro-contrôleurs a connu une croissance considérable ces dernières années. Cette tendance est motivée par les avantages de l'IA embarquée, tels que la latence réduite, l'autonomie accrue, la sécurité améliorée, la réduction de la bande passante et l'amélioration de la confidentialité. Les micro-contrôleurs adaptés à ces applications sont généralement des processeurs à faible consommation d'énergie, capables de gérer de grandes quantités de données et offrant des fonctionnalités de traitement d'IA.
Voici quelques micro-contrôleurs en vogue dans l'IA embarquée :

— Les micro-contrôleurs ARM Cortex-M sont largement utilisés dans les applications embarquées d'IA en raison de leur puissance de traitement suffisante et de leur disponibilité à un prix abordable. Ils sont souvent utilisés pour les tâches de classification d'images en temps réel, la détection de mouvement, la reconnaissance de formes et la surveillance de la qualité des images.

— Les micro-contrôleurs Raspberry Pi Pico sont également de plus en plus populaires en raison de leur capacité à prendre en charge des langages de programmation courants tels que Python, C et C++.

— Le NVIDIA Jetson Nano est un autre micro-contrôleur populaire utilisé dans les applications d'IA embarquée en raison de son GPU intégré, ce qui le rend capable d'exécuter des tâches d'IA complexes.

— Le STM32MP1 est également un choix courant pour les tâches d'IA en raison de sa faible consommation d'énergie et de sa prise en charge de nombreux protocoles de communication.

— Le BeagleBone AI est un autre choix populaire pour les projets d'IA embarquée en raison de son processeur ARM Cortex-A15 et de son coprocesseur C66x DSP pour les tâches de traitement d'IA.

— L'ESP32-S2 est un micro-contrôleur WiFi basé sur un processeur Tensilica LX7 à double cœur et doté d'une faible consommation d'énergie, ce qui en fait un choix idéal pour les projets nécessitant une connectivité WiFi intégrée.

— Le STMicroelectronics STM32H7 est également un micro-contrôleur hautes performances prisé dans les applications d'IA embarquée grâce à sa puissance de traitement élevée, sa faible consommation d'énergie et sa connectivité intégrée. C'est celui qui est utilisé dans l'Arduino Nicla Vision.

En somme, l'IA embarquée sur micro-contrôleurs est un domaine en constante évolution, offrant de plus en plus de possibilités pour des applications pratiques. Les micro-contrôleurs ARM Cortex-M, Raspberry Pi Pico, NVIDIA Jetson Nano, STM32MP1, BeagleBone AI, ESP32-S2 et STMicroelectronics STM32H7 sont tous des choix courants pour les projets d'IA embarquée en raison de leur puissance de traitement, de leur faible consommation d'énergie et de leurs fonctionnalités de communication avancées.

4.2 Arduino Nicla Vision

4.2.1 Présentation

L'Arduino Nicla Vision est un micro-contrôleur qui convient parfaitement aux projets d'IA embarquée nécessitant de la reconnaissance de motifs sur des images prises en temps réel, en raison de ses caractéristiques techniques avancées. Tout d'abord, le micro-contrôleur est équipé d'un processeur ARM Cortex-M7, ce qui offre une grande puissance de traitement pour les tâches d'IA en temps réel. En outre, l'Arduino Nicla Vision est équipé d'une caméra capable de capturer des images, ainsi qu'un microphone.

L'Arduino Nicla Vision dispose également d'une bibliothèque logicielle d'IA optimisée pour les applications de reconnaissance d'images, qui comprend des algorithmes de traitement d'images, tels que la segmentation d'image, la détection de contours et la classification d'images. Cette bibliothèque permet de développer rapidement des applications d'IA embarquées et qui peuvent être utilisées pour la reconnaissance de motifs en temps réel.

De plus, le micro-contrôleur Arduino Nicla Vision dispose d'un grand nombre de capteurs et d'interfaces de communication, tels qu'un port USB, des interfaces WiFi et Bluetooth, et des entrées/sorties analogiques et numériques, ce qui permet une intégration facile dans des systèmes plus larges et une connectivité accrue pour les applications IoT.

Enfin, l'Arduino Nicla Vision est également facile à programmer et à utiliser grâce à son environnement de développement intégré (IDE) open source : OpenMV. Celui-ci offre une interface graphique intuitive pour la programmation et le débogage. Cela permet de développer rapidement de nombreuses applications dans le domaine du Machine Vision.

En somme, l'Arduino Nicla Vision est une plateforme de développement idéale pour les projets d'IA embarquée nécessitant de la reconnaissance de motifs sur des images prises en temps réel, en raison de sa puissance de traitement élevée, de sa bibliothèque logicielle d'IA optimisée, de sa connectivité et de sa facilité d'utilisation et de programmation.

4.2.2 Caractéristiques techniques [6]

Boards	Name SKU	Nicla Vision ABX00051
Microcontroller	STM32H757AII6 Dual Arm Cortex M7/M4	
Pins	LED built-in	1 RGB LED (I2C)
	Digital I/O Pins	10 (of which 2 are shared with I2C and 4 are shared with SPI)
	Analog input pins	2, both shared with PWM
	PWM pins	12 (of which 2 are shared with analog, 2 are shared with I2C and 4 are shared with SPI)
	External interrupts	12
Connectivity	Bluetooth Wi-Fi Secure element	Bluetooth 4.2 (Murata 1DX module) 11b/g/n (Murata 1DX module) NXP SE050C2 Crypto chip
Communication	UART I2C SPI	YES 1 1
Power	Microcontroller operating voltage Board Power Supply (USB/VIN) Supported battery Battery connector DC Current per I/O pin	1.8V translated to 3.3V on external pins 5V Li-ion/Li-Po Single Cell, 3.7V JST 3-pin 1.2mm pitch 4.7mA
Clock speed	Processor (M7) Processor (M4)	480MHz 240MHz
Memory	ST STM32H757AII6 QSPI flash	2MB Flash, 1MB RAM 16MB
Dimensions	Weight Width Length PCB Thickness	2g 22.86mm 22.86mm 1mm

TABLE 1 – Documentation technique de l'Arduino Nicla Vision

4.3 Machine vision

4.3.1 Algorithmes classiques

Les algorithmes de machine vision (ou vision par ordinateurs) permettent de traiter des images afin d'en extraire des informations utiles pour des applications telles que la reconnaissance de formes, la détection d'objets, la segmentation d'images ou encore la reconnaissance faciale.

Parmi les algorithmes de machine vision couramment utilisés, on peut citer le seuillage, la convolution, le filtrage spatial, la détection de coins, la transformée de Hough et le pyramidage d'images. Le seuillage est utilisé pour la segmentation d'images en deux classes, tandis que la convolution est utilisée pour la détection de bords et de contours dans les images. Le filtrage spatial permet de supprimer le bruit d'une image, tandis que la détection de coins permet de trouver des points d'intérêt dans une image. La transformée de Hough est utilisée pour la détection de formes géométriques telles que les cercles, les lignes et les ellipses, tandis que le pyramidage d'images permet de faciliter l'analyse d'images à différentes échelles.

Ces algorithmes peuvent être utilisés pour de nombreuses applications, telles que la reconnaissance de formes, la détection d'objets dans les images médicales, la reconnaissance de caractères pour la reconnaissance optique de caractères (OCR), la reconnaissance faciale pour les applications de sécurité et la détection de mouvements dans les vidéos de surveillance. En somme, ces algorithmes de machine vision offrent des solutions efficaces pour l'analyse d'images, sans nécessiter l'utilisation de l'intelligence artificielle.

4.3.2 Intelligences artificielles

Les techniques d'Intelligence Artificielle sont devenues indispensables pour la réalisation de tâches complexes. Parmi ces techniques, les réseaux de neurones convolutionnels (CNN) sont largement utilisés pour la classification d'images, la segmentation d'images et la détection d'objets. Les CNN sont des réseaux de neurones qui utilisent une architecture spéciale pour extraire les caractéristiques des images. Ils prennent en entrée une image et la transforment en une série de cartes de caractéristiques qui sont utilisées pour identifier des objets, des formes et des motifs dans l'image.

Les réseaux de neurones récurrents (RNN) et les réseaux de neurones à mémoire à court terme (LSTM) sont utilisés pour des tâches nécessitant la prise en compte de séquences d'images, telles que la reconnaissance de gestes ou de mouvements. Les RNN et les LSTM sont des réseaux de neurones qui peuvent prendre en compte les états précédents pour faire des prédictions sur les images suivantes. Ces réseaux sont couramment utilisés pour la reconnaissance de la parole et la traduction automatique, où ils sont utilisés pour capturer la structure temporelle des données d'entrée.

Les réseaux de neurones adversaires génératifs (GAN) sont une technique d'IA qui permet de générer des images réalistes à partir de données d'entrée. Les GAN sont composés de deux réseaux de neurones : un générateur et un discriminateur. Le générateur crée des images à partir de données d'entrée, tandis que le discriminateur détermine si les images sont réelles ou non. Les GAN ont de nombreuses applications, comme la génération d'images de visages synthétiques, la synthèse de textures, la restauration d'images endommagées et la modification de vidéos.

Enfin, les réseaux de neurones siamois sont utilisés pour comparer des images et détecter des similitudes entre elles. Les réseaux siamois sont composés de deux branches de réseaux de neurones identiques, chacune prenant une image en entrée et produisant un vecteur de caractéristiques en sortie. Ces vecteurs de caractéristiques sont ensuite comparés pour déterminer la similitude entre les deux images. Les réseaux de neurones siamois ont des applications dans la recherche d'images similaires dans une grande base de données, l'authentification d'identité, et la surveillance de la qualité de fabrication.

En somme, les techniques d'IA ont permis des avancées significatives dans le domaine de la vision par ordinateur, offrant des solutions efficaces pour des tâches complexes et diverses, telles que la reconnaissance faciale, la reconnaissance d'objets dans des environnements complexes, la détection de défauts sur des surfaces et la détection d'anomalies. Ces techniques sont devenues des outils incontournables pour la vision par ordinateur, offrant une grande flexibilité pour s'adapter aux nouvelles données et répondre aux besoins de diverses industries.

4.4 Outils

4.4.1 OpenMV

OpenMV[4] est un logiciel libre dédié à la vision par ordinateur embarquée. Il a été développé dans le but de rendre la vision par ordinateur accessible à tous. OpenMV permet aux utilisateurs d'utiliser des micro-contrôleurs compatibles avec Python pour acquérir et traiter des images en temps réel, sans nécessiter de matériel supplémentaire coûteux.

OpenMV est basé sur Python, ce qui facilite l'écriture de scripts pour les tâches de traitement d'image. Il offre une variété de fonctionnalités telles que la reconnaissance de forme, la détection de mouvement, la reconnaissance de couleurs et la détection de caractéristiques. Ces fonctionnalités peuvent être utilisées pour des tâches allant de la reconnaissance de gestes à la surveillance de la qualité des produits, en passant par la détection de personnes et de véhicules. Le logiciel est livré avec un IDE intégré qui permet aux utilisateurs de développer et de tester facilement des programmes pour la vision par ordinateur embarquée.

4.4.2 Edge Impulse

Edge Impulse[3] est une plateforme en ligne qui permet aux utilisateurs de développer, de déployer et de gérer des modèles d'apprentissage automatique pour les appareils IoT et les micro-contrôleurs. La plateforme est conçue pour aider les développeurs à créer rapidement des modèles d'IA à partir de données collectées à partir de capteurs tels que les accéléromètres, les gyroscopes et les capteurs de température.

Le logiciel permet aux utilisateurs de télécharger leurs données de capteurs et de les nettoyer automatiquement pour les préparer à l'entraînement des modèles. Les utilisateurs peuvent ensuite sélectionner le type de modèle d'apprentissage automatique qu'ils souhaitent entraîner, tels que les réseaux de neurones, les SVM ou les arbres de décision, et ajuster les paramètres de l'algorithme pour obtenir les meilleures performances.

Edge Impulse fournit également des outils pour visualiser les données de capteurs, les performances du modèle et les résultats de l'inférence. Les modèles entraînés peuvent être déployés sur des appareils IoT ou des micro-contrôleurs à l'aide de bibliothèques de code générées automatiquement par la plateforme.

5 Projet

5.1 Choix du motif

Dans le domaine du machine vision, les codes-barres 2D sont des motifs couramment utilisés pour la détection et l'identification d'objets.

(a) ARuco code (b) QR code (c) AprilTag

FIGURE 1 – Différents types de motifs utilisés

Parmi ces motifs, trois des plus populaires sont le QR code, le code ARuco et l'AprilTag (Figure 1) :

— Le code ARuco est un code de réalité augmentée utilisé pour la détection et le suivi de la position et de l'orientation des objets dans des environnements en trois dimensions. Il se compose d'un ensemble de carrés noirs et blancs disposés de manière à former une grille carrée. Chaque code ARuco a un identifiant unique qui peut être utilisé pour suivre la position et l'orientation de l'objet.

— Le QR code est un type de code-barres 2D qui peut stocker une grande quantité d'informations, telles que des URL, des numéros de téléphone et des adresses électroniques. Il est constitué de modules carrés disposés dans un modèle régulier sur un fond blanc. Le QR code est facilement identifiable en raison de sa structure régulière, ce qui facilite sa détection par les algorithmes de vision par ordinateur.

— L'AprilTag est un autre type de code de réalité augmentée similaire au code ARuCO, mais avec des caractéristiques uniques qui le rendent plus facilement identifiable. L'AprilTag utilise une grille de carrés noirs et blancs avec des bordures épaisses et un motif unique de bits noirs et blancs à l'intérieur de chaque carré. Cette conception unique permet à l'AprilTag d'être rapidement identifié par les algorithmes de vision par ordinateur, même dans des conditions d'éclairage difficiles.

En ce qui concerne la détection par une IA, chaque motif a ses propres avantages. Le QR code est facilement identifiable et peut stocker une grande quantité d'informations, tandis que les codes ARuco et AprilTag peuvent être utilisés pour suivre la position et l'orientation des objets en 3D. Le code ARuco est connu pour sa précision et sa robustesse dans des environnements variés, tandis que l'AprilTag est reconnu pour sa rapidité et sa fiabilité.

Dans le contexte du projet en question, l'AprilTag a été utilisé en raison de sa rapidité et de sa fiabilité dans des environnements en temps réel. En effet, ce projet implique la reconnaissance de motifs sur des images prises en temps réel, et l'AprilTag a été choisi pour sa capacité à être rapidement identifié par les algorithmes de vision par ordinateur, même dans des conditions d'éclairage difficiles. De plus, sa conception unique permet de minimiser les erreurs de détection et d'assurer une haute précision.

5.2 Prise en main du micro-contrôlleur

Il est crucial de prendre en main les outils logiciels nécessaires et de les tester avant d'essayer d'atteindre l'objectif fixé pour plusieurs raisons.

Dans le cadre de ce projet, l'outil logiciel OpenMV sera utilisé pour programmer l'Arduino Nicla Vision. Il est donc d'autant plus important de se familiariser avec cet outil pour comprendre son fonctionnement et s'assurer qu'il est correctement configuré. Tester OpenMV permettra également d'identifier les éventuels problèmes techniques ou les limitations de cet outil spécifique.
En outre, il est important de comprendre les capacités et les limites d'OpenMV pour s'assurer qu'il convient à l'objectif que l'on souhaite atteindre. Cela permettra également d'optimiser son utilisation pour répondre aux besoins spécifiques du projet.
Enfin, tester OpenMV permettra de maximiser son efficacité en comprenant son fonctionnement et ses capacités. En configurant cet outil de manière optimale, il sera possible de répondre aux exigences du projet de manière plus efficace.

Prendre en main et tester l'outil logiciel OpenMV est une étape cruciale pour la réussite de ce projet d'intelligence artificielle embarquée. Cela permettra de comprendre le fonctionnement de cet outil, de réduire les risques d'erreur, d'identifier les limitations et les problèmes techniques et d'optimiser son utilisation pour répondre aux exigences spécifiques du projet.

Tous les programmes OpenMV ont une architecture commune :

1. Chargement des bibliothèques : on importe les bibliothèques Python qui vont permettre de manipuler les différents éléments du micro-contrôleur. Notamment les plus importantes : la caméra et le timer.
2. Initialisation des variables : on initialise les paramètres de la caméra
3. Boucle principale : on prend une image et on l'affiche dans l'IDE

Voilà à quoi ressemble un programme avec OpenMV :

```
import sensor, image, time

# Reset and initialize the sensor.
sensor.reset()
# Set pixel format to RGB565 (or GRAYSCALE)
sensor.set_pixformat(sensor.RGB565)
# Set frame size to QVGA (320x240)
sensor.set_framesize(sensor.QVGA)
# Wait for settings to take effect.
sensor.skip_frames(time = 2000)
# Create a clock object to track the FPS.
clock = time.clock()

while(True):
    clock.tick()             # Update the FPS clock.
    img = sensor.snapshot()  # Take a picture and return the image.
    print(clock.fps())
```

FIGURE 2 – Code de base d'un programme avec OpenMV

5.3 Création d'une base de données

Composer une base de données est une étape essentielle dans la plupart des projets de machine learning et de vision par ordinateur. Une base de données permet de stocker les données d'entraînement et de test nécessaires pour entraîner et évaluer un modèle d'IA.

Cette étape est inévitable car la qualité et la quantité des données sont cruciales pour la réussite d'un modèle d'IA. En effet, les performances d'un modèle dépendent en grande partie de la qualité et de la variété des données utilisées pour l'entraînement. Par conséquent, il est nécessaire de collecter ou de générer une base de données suffisamment grande et représentative du domaine d'application pour obtenir des résultats précis et fiables.

Dans le cadre de ce projet, le choix a été fait de générer une base de données avec un programme pour plusieurs raisons. Tout d'abord, la génération de données permet de créer une base de données plus variée et contrôlée, en faisant varier des variables pertinentes. Ici, on peut varier la taille du motif dans l'image, sa position ainsi que son orientation.

En partant d'une base de données d'image d'intérieurs [1], le programme permet de placer le pattern dans cet environnement. Cela permet d'obtenir une grande quantité d'intérieurs contenant le motif ce qui permettra à notre IA de généraliser le résultat dans un plus grand type d'environnements.

La figure 3 présente la *pipeline* permettant de traiter chaque image et ainsi de créer la base de données qui servira pour l'entraînement du modèle et la figure 4 présente deux exemples de cette *pipeline*.

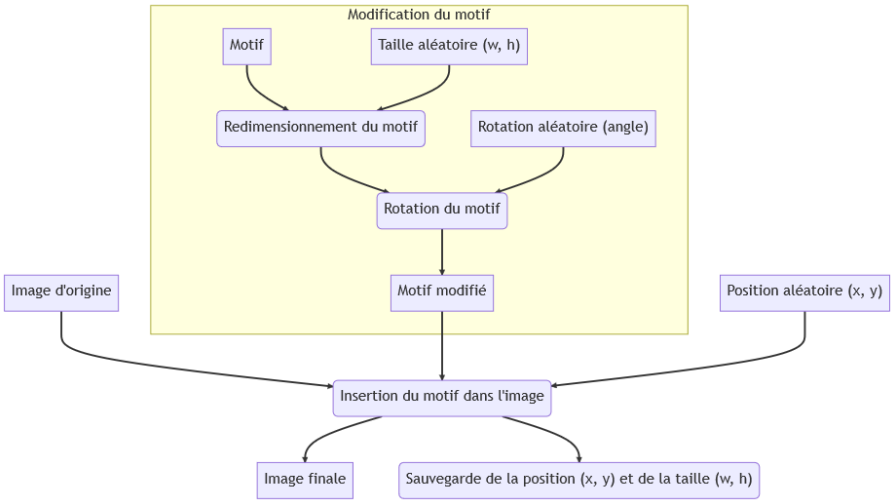


FIGURE 3 – *Pipeline* du traitement des images

(a) Airport-inside-001

(b) Airport-inside-002

FIGURE 4 – Exemples d'images générées

5.4 Entraînement de l'IA

L'entraînement d'un modèle de machine learning avec Edge Impulse consiste à apprendre à une IA à reconnaître un ou plusieurs objets spécifiques à partir d'images. Ici notre motif. Pour cela, on utilise l'ensemble d'images représentant contenant notre motif que nous avons généré précédemment. Edge Impulse va alors analyser ces images pour apprendre les caractéristiques distinctives du motif. Il va générer des *features* propres au motif et à l'environnement dans lequel il est.

L'ensemble d'images, appelé jeu de données, est divisé en deux parties : une partie pour l'entraînement du modèle et une autre pour la validation. L'entraînement consiste à apprendre à l'IA à reconnaître les objets en utilisant le jeu de données d'entraînement. La validation consiste à évaluer la précision du modèle en utilisant le jeu de données de validation.

Edge Impulse propose deux modèles entraînés à l'avance pour la détection d'objet [2] :

— MobileNetV2 SSD FPN

— FOMO

Après avoir testé les deux modèles, l'utilisation de MobileNetV2 SSD FPN semble donner de meilleurs résultats. Une fois le modèle entraîné, il est ensuite affiné et optimisé pour son intégration dans le micro-contrôleur.

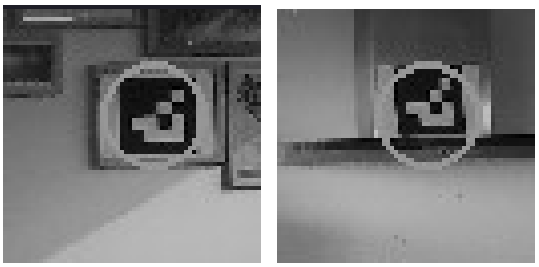
5.5 Intégration sur le micro-contrôleur

Pour intégrer le modèle au micro-contrôleur, il y a deux façon. L'une est de charger le modèle depuis l'espace mémoire de la carte. Cependant, l'Arduino Nicla Vision ne comporte qu'une mémoire flash. Il va donc falloir modifier le *bootloader* de la carte afin de charger directement le modèle dedans. De cette façon, le programme n'a pas besoin de charger depuis un stockage externe. Pour modifier le *bootloader* de l'Arduino Nicla Vision, une manipulation relativement simple est possible. En créant un *fork* du projet OpenMV sur Github, il est possible de modifier le modèle interne du *bootloader*. A chaque mise à jour de ce *fork*, un programme va compiler le *bootloader* pour toutes les plateformes et donc pour l'Arduino Nicla Vision. Il suffit alors de charger le *bootloader* sur la carte.

5.6 Résultats

Pour obtenir le résultat, il suffit de lancer le programme sur l'Arduino Nicla Vision. La carte charge le modèle interne (celui comprit dans le *bootloader*) et l'utilise. On fixe le seuil de validation à 40% ce qui signifie que si le modèle est "sûr" à plus de 40%, alors le motif sera détecté. On peut alors récupérer la position du motif dans l'image et l'utiliser. Ici, on affiche simplement un cercle avec pour centre la position du motif.

Afin de s'approcher au plus près du résultat souhaité, il faut obtenir une image proche de celles avec lesquels nous avons entraîné le modèle auparavant. Le modèle ayant été entraîné avec des images de 48x48 pixels, on réduit la taille de l'image. On utilise également une image avec des niveaux de gris. L'on peut ainsi voir deux exemples de résultats sur la figure 5.



(a) Résultat 1 (b) Résultat 2

FIGURE 5 – Exemples de résultats

Il existe cependant des limitations à notre IA. Son entraînement s'appuie sur un ensemble d'images d'intérieurs variés. En fonction de l'environnement dans lequel il faudra détecter le motif, les résultats peuvent varier. De même, l'IA ayant été entraînée avec des motifs d'une certaine taille, elle ne s'adapte pas toujours à tous les cas de figure. Pour régler ces problèmes il faudrait définir plus précisément les besoins de cette IA, son environnement et les résultats qui doivent être obtenus.

En somme, l'IA pourrait être améliorée pour répondre plus précisément à un besoin (chaîne de production, détection de panneaux de signalisation etc). Cependant, dans un contexte général, son efficacité est avérée. L'IA traite en temps réel les images prises par le micro-contrôleur, et donne un résultat cohérent. L'on peut ainsi savoir si l'image contient notre motif et à quelle position il est situé dans l'image. Les objectifs fixés au début du projet ont donc été atteint avec succès.

5.7 Code

Voici donc le programme final :

```
import sensor, image, time, tf, math

sensor.reset()                         # Reset and initialize the sensor.
sensor.set_pixformat(sensor.GRAYSCALE) # Set pixel format to GRAYSCALE
sensor.set_framesize(sensor.B64X64)    # Set frame size to QVGA (64x64)
sensor.set_windowing((240, 240))       # Set 240x240 window.
sensor.skip_frames(time=2000)          # Let the camera adjust.

min_confidence = 0.4

# Load built-in pattern detection model
labels, net = tf.load_builtin_model("pattern_detection")

colors = [(255,   0,   0)]

clock = time.clock()
while(True):
    clock.tick()

    img = sensor.snapshot()

    for i, detection_list in enumerate(net.detect(img,
    thresholds=[(math.ceil(min_confidence * 255), 255)])):
        if (i == 0): continue # background class
        if (len(detection_list) == 0): continue # no detections for this class?

        print("********** %s **********" % labels[i])
        for d in detection_list:
            [x, y, w, h] = d.rect()
            center_x = math.floor(x + (w / 2))
            center_y = math.floor(y + (h / 2))
            print(f"x {center_x}\ty {center_y}")
            img.draw_circle((center_x, center_y, 12), color=colors[i-1], thickness=2)

    print(clock.fps(), "fps", end="\n")

```

FIGURE 6 – Programme final

L'entièreté du code est disponible dans le *repository* suivant : https://github.com/TomDep/Projet5A. Celui-ci contient également le code permettant d'insérer le motif dans les images d'entraînement. Le code source du projet est sous licence GPL et donc parfaitement réutilisable sous contrainte de la licence.

6 Recul et perspectives

6.1 Recul

Ce projet a été une découverte de l'IA intégré sur micro-contrôleurs. J'ai donc pu découvrir comment créer et intégré des IA spécialement pour ces systèmes. J'ai pu apprendre à faire des concessions sur le choix du modèle afin de le rendre utilisable. Il a donc fallu s'adapter pour réduire la taille du modèle pour qu'il tienne dans la mémoire flash et l'entraîner avec relativement peu de données pour réduire les temps de calculs.

Cela a aussi été pour moi l'occasion d'améliorer mes connaissances en intelligence artificielle. Bien que n'étant pas spécialiste du domaine, réussir à construire une IA de la sorte constitue pour moi une réussite personnelle.

Cependant, réaliser ce projet en parallèle des études à l'étranger s'est avéré une tâche compliquée. La majorité du travail a été effectué sur les mois de janvier et février. Le cahier des charges ne comportant pas d'applications précise à aussi été une source de difficulté. En effet, il a fallut construire une IA pouvant généraliser dans beaucoup d'environnement différents. Et, bien qu'ayant restreint le champs d'application aux environnements d'intérieurs, fixer un contexte d'utilisation aurait permit de meilleurs résultats.

6.2 Perspectives

Ce projet pourrait constituer une bonne base pour utiliser de l'IA embarquée dans beaucoup de contextes différents. En choisissant un contexte d'utilisation plus précis, l'on pourrait réellement mettre en pratique cette IA et toute sa potentialité. Parmi toutes les possibilités, voici quelques perspectives qui pourraient s'avérer intéressantes pour ce projet :

— Détection de panneaux de signalisation : intégrée à un véhicule (autonome), cette IA pourrait prendre connaissance des différents panneaux et ainsi mettre à disposition de nouvelles informations pour donner des instructions de conduite au véhicule.

— Reconnaissance de visages : utilisée dans un contexte embarqué, une reconnaissance de visage pourrait assurer la sécurité d'un système en s'assurant de l'identité de son utilisateur. Bien que comportant de nombreux problèmes éthiques, cela pourrait s'avérer être une bonne poursuite de projet.

— Contrôle qualité : intégré à une chaîne de production, ce micro-contrôleur pourrait garantir de la conformité de pièces produites. En fonction des résultats, une action autonome pourrait être prise en ainsi garantir l'efficacité de la production sans intervention humaine.

La liste n'est pas exhaustive et il pourrait y avoir de nombreuses autres utilisations pour un système comme celui là. Cependant, il semblerait pertinent pour de futurs étudiant·es de travailler sur ce système pour l'améliorer et l'utiliser dans un projet. Ce micro-contrôleur n'étant qu'un support pour une IA, il pourrait être intéressant de l'utilisant en addition d'autres micro-contrôleurs de travailler cette fois-ci sur les différents protocoles de communication disponibles et d'étudier la façon dont un ensemble de sous-systèmes pourraient former un système autonome et intelligent.

7 Conclusion

Ce projet avait pour but de mener un travail de recherche autour du micro-contrôleur Arduino Nicla Vison et d'y intégrer une intelligence artificielle. Au cours de ce projet, nous avons donc étudié le différents micro-contrôleurs ainsi que les algorithmes majoritairement utilisés utilisés dans ce domaine. Nous avons également recherché quels outils pourraient être utiles pour programmer l'Arduino Nicla Vision et intégrer l'IA. C'est donc grâce à ce travail préliminaire que la création de l'IA a pu commencer.

Nous avons dans un premier temps prit en main les différents outils à notre disposition. Et, suite à des résultats convaincants, la création de l'IA a pu démarrer. Un jeu de données à donc été généré grâce à un programme afin de créer une grande diversité de données d'entraînement. Ces données ont permises de construire notre modèle de reconnaissance d'objets sur l'outil Edge Impulse.

Enfin, nous avons implanté le modèle dans le micro-contrôleur grâce à OpenMV et réalisé des tests dans différents contexte. L'IA est capable de détecté le motif choisi et de le positionner dans l'image. Le programme tourne en temps réel. Les objectifs du projet ont donc été atteints.
Bien qu'étant une réussite, ce projet pourrait être continué et se tourner vers des cas d'utilisations concrets. Les objectifs de recherches autour des possibles qu'ouvre ce micro-contrôleur semblent avoir étés atteints.

Table des figures

Liste des tableaux

Références

[1] A.Torralba et A. Oliva A. Quattoni. Indoor scene recognition. URL : http://web.mit.edu/torralba/www/indoor.html.

[2] Edge Impulse. Object detection (documentation). URL : https://docs.edgeimpulse.com/docs/edge-impulse-studio/learning-blocks/object-detection.

[3] Edge Impulse. Website. URL : impulse.com/.

[4] OpenMV. Website. URL : https://openmv.io/.

[5] Arduino Website. Arduino nicla vision. URL : https://store.arduino.cc/products/nicla-vision.

[6] Arduino Website. Arduino nicla vision, documentation. URL : https://docs.arduino.cc/hardware/nicla-vision.

[7] Wikipedia. Kanban board. URL : https://en.wikipedia.org/wiki/Kanban_board.