Projet final de DEC

REQUIS CLIENT

Buts:

* Au moins un circuit imprimé de caméra avec une image sur un Nvidia Jetson (préférablement un Nano Devkit).
* Détection d’objets arbitraire à partir de l’image reçue sur un Nvidia Jetson (préférablement un Nano Devkit).
* Commandes moteur fonctionnelles.
* Mode manuel : contrôle du bras.
* Mode manuel : feedback de la caméra.
* Rapport.

Bonus :

* Capacité à cibler, attraper et déplacer des objets arbitrairement placés.
* Auto-build et auto-deploy.
* ROS (Robot Operating System).
* Docker containers.
* Simulation sur Gazebo.
* Simulation d’apprentissage visuomoteur par renforcement.
* Apprentissage visuomoteur par renforcement dans le monde réel.
* Supporte Bash sur Ubuntu Linux.
* Bash sur Ubuntu Linux sur WSL2 (écrire une procédure d’installation pour Windows 10 afin que n’importe qui avec un GPU sur x86\_64 puisse utiliser le code).
* Bash sur Brew pour MacOS.

EN TOUT TEMPS :

* Accumuler des outils de conception de circuit imprimé haute-vitesse.
* Accumuler des outils de simulation
* Accumuler de la documentation.
* Prendre 15 minutes chaque jour pour évaluer ce qui pourrait avoir fonctionné ou pas et porter attention au déficit d’attention pour rectifier rapidement lorsque ça arrive. Noter les succès et les échecs pour le rapport et pour changer de stratégie régulièrement.
* Demander l’approbation du projet et prendre des notes des rétroactions des professeurs.
* Poser des questions à qui je peux pour maximiser l’apprentissage ultra-rapide.
* Tester tout ce qui génère des doutes (surtout ce qui est relié à la conception haute-vitesse).
* Sauvegarder tout partout en tout temps.

SEMAINE 1 :

* Écrire requis client (buts vs bonus). (fait)
* Activer VNC dans Docker sur ARM64v8 et sauver un écran, clavier et souris! (fait)
* Réparer bug de udev (fait)
* Planifier les prochaines semaines. (en cours)

SEMAINE 2 :

* Tester le container pour uARM dans Docker avec ROS sur Nvidia Jetson Nano. (fait, deadline lundi)
* Choisir/définir matériel et commander/emprunter le matériel. (fait, deadline mardi)
* Planifier les prochaines semaines avec des chemins alternatifs où il reste des risques et officialiser dans Trello. (en cours, deadline mercredi)
* Schématiser l’architecture matérielle dans DRAWIO et produire Bill Of Materials. (en cours, deadline jeudi)
* Tests de build/run en utilisant le même code sur ARM64v8 que sur x86\_64, avec émulation optionnelle en Docker avec QEMU. Ne pas dépenser plus de 3h sur ça, parce que ça fonctionne déjà sur ARM64v8. Ferait gagner du temps de build. (deadline samedi)

SEMAINE 3 :

* Schématiser l’architecture système dans DRAWIO. (deadline lundi)
* Schématiser l’architecture logicielle dans DRAWIO. (deadline lundi)
* Conception de la schématique du circuit imprimé. (deadline mardi)
* Description préliminaire des algorithmes utilisés (attendre avant de mettre trop de détails). (deadline mercredi)
* Schémas préliminaires des algorithmes utilisés. (deadline jeudi)
* Ajouter la caméra sur ROS. (deadline mercredi)
* Simuler le circuit électrique. (deadline jeudi)
* Étudier conception haute-vitesse (voir Texas Instruments application notes). (deadline vendredi)

SEMAINE 4 :

* Début du plan A (architecture logicielle) : tracking par sélection du canal de couleurs. (deadline lundi)
* Officialiser en réécrivant la publication (très brouillon, simple mais détaillé pour sauver du temps plus tard). (deadline mardi)
* Empreintes terminées. (deadline mercredi)
* La réception du matériel devrait débuter. (deadline jeudi)
* Tester l’entrée de la caméra. (deadline vendredi)

SEMAINE 5 :

* Continuation du plan A.
* Début de l’installation d’OpenAI Gym sur ROS. (deadline lundi)
* Début du plan B (architecture logicielle) : installer jetson-inference sur ROS et tester. (deadline mercredi)
* Conception de la disposition du circuit imprimé. (deadline vendredi)

SEMAINE 6 :

* Continuation du plan B.
* Début de l’écriture de la définition de l’environnement OpenAI Gym. (deadline lundi)
* Complétion du plan A. (deadline mercredi)
* Les pièces devraient être toutes arrivées (deadline vendredi).

SEMAINE 7 :

* Début du plan C (architecture logicielle) : étudier la conception d’environnement sur OpenAI Gym en écrivant un crypto\_bot ou stocks\_bot simple qui vend et achète automatiquement en apprenant sur le tas par renforcement. NE PAS prendre plus de 2 jours pour perfectionner le bot. Le but est d’apprendre les conventions de la librairie; pas de faire de l’argent. (deadline lundi)
* Complétion du plan B. (deadline mardi)
* Remise des fichiers de production du circuit imprimé. (deadline mercredi)
* Commander au moins 2 circuits imprimés (avoir pièces prêtes au moins en double). (deadline mercredi)

SEMAINE 8 :

* Continuer plan C : écrire environnement OpenAI Gym pour le uARM dans le Docker ROS.
* Souder un seul circuit imprimé sur 2. (deadline vendredi)

SEMAINE 9 :

* Début de la réécriture de FERM avec l’environnement OpenAI Gym pour le uARM dans container Docker sur ROS. (deadline lundi)
* Entraîner réseaux neuronaux et tester. (deadline vendredi)

SEMAINE 10 :

* Valider les réseaux neuronaux. (deadline mercredi)
* S’assurer de la cohésion matériel/logiciel avec le plan A et B. (deadline jeudi)
* Complétion du plan C. (deadline vendredi)
* Procéder aux derniers tests matériels. (deadline samedi)

SEMAINE 11 :

* Continuation et complétion de tout.
* Débogage, réglages, perfectionnement. (no deadline)
* Perfectionner publication. (deadline vendredi)

SEMAINE 12 :

* Semaine réservée aux imprévus.
* Sauvegardes partout!!!
* Temps en extra pour finaliser et nettoyer.

Code à date:

<https://github.com/abstractguy/jetson-containers.git>

<https://github.com/abstractguy/ferm.git>

Références matérielles:

https://github.com/realizator/stereopi-kicad-schematic/

<https://www.raspberrypi.org/documentation/hardware/camera/schematics/rpi_SCH_Camera2_2p1.pdf>

https://www.raspberrypi.org/documentation/hardware/camera/

<https://www.raspberrypi.org/documentation/hardware/camera/schematics/rpi_SCH_HQcamera_1p0.pdf>

<https://github.com/RidgeRun/NVIDIA-Jetson-IMX477-RPIV3>

<https://www.buyapi.ca/product/raspberry-pi-8mp-camera-board-v2/>

https://www.uctronics.com/arducam-1mp-2-stereoscopic-camera-bundle-kit-for-raspberry-pi-nvidia-jetson-nano-xavier-nx-two-ov9281-global-shutter-monochrome-camera-modules-and-camarray-stereo-camera-hat.html

Références logicielles:

https://github.com/JetsonHacksNano/CSI-Camera

<https://github.com/DrYerzinia/RaspberryPiCamerav21>

<https://developer.ridgerun.com/wiki/index.php?title=Jetson_Nano/Development/Building_the_Kernel_from_Source>

http://wiki.ros.org/ROS/Tutorials

<http://wiki.ros.org/ROS/Tutorials/WritingPublisherSubscriber%28python%29>

<https://www.arducam.com/docs/camera-for-jetson-nano/multiple-cameras-on-the-jetson-nano/using-arducam-camarry-on-ros/>

<https://github.com/ArduCAM/MIPI_Camera>