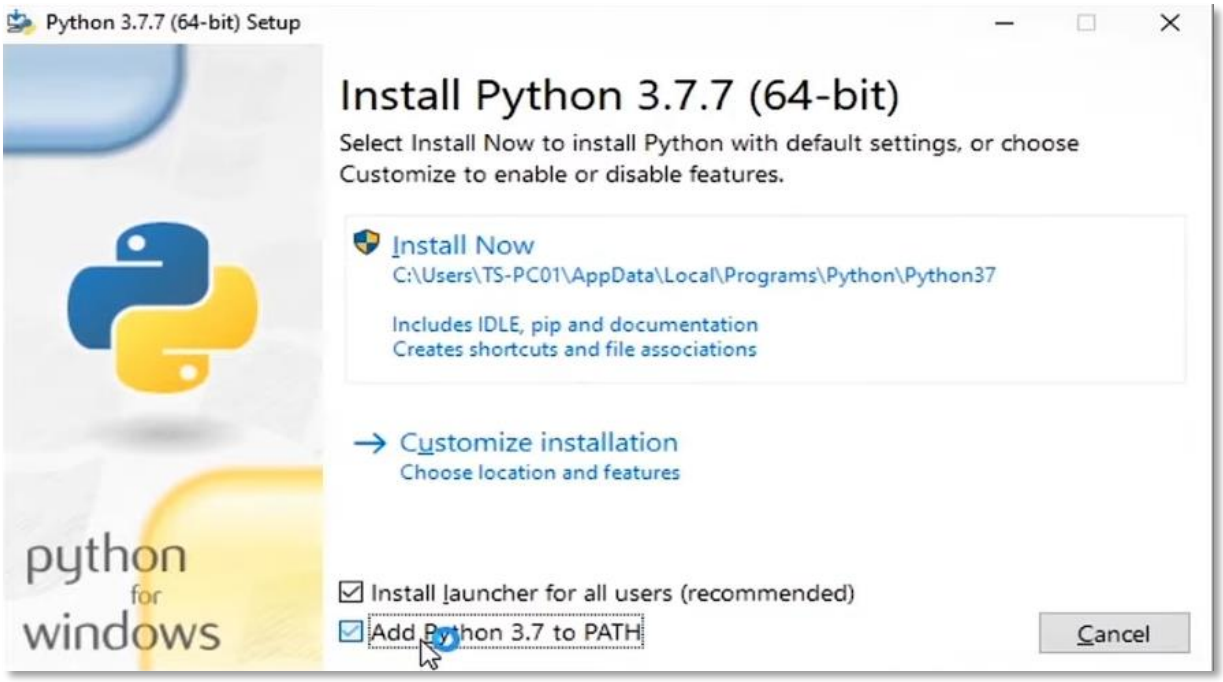
1. Installation de Python 3.7

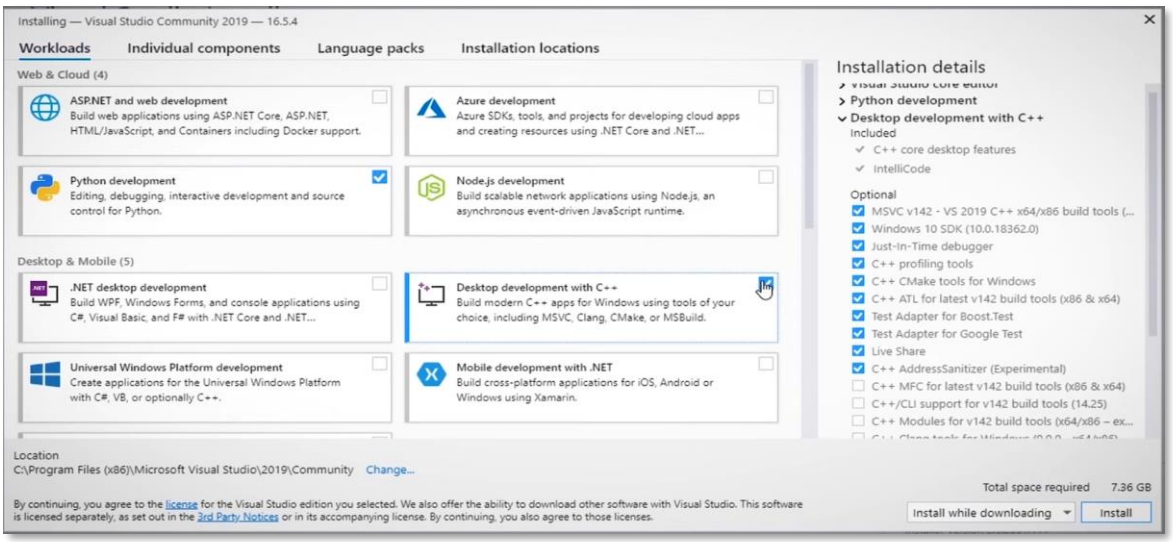
J'utilise la version 3.7.7, mais toute version de Python 3.7 fonctionnera.
Lors de l'installation, pensez à cocher la case « Add Python to path ».



Une fois installée, aller dans la console et installer Numpy avec « pip install numpy »

2. Installation des outils de compilation

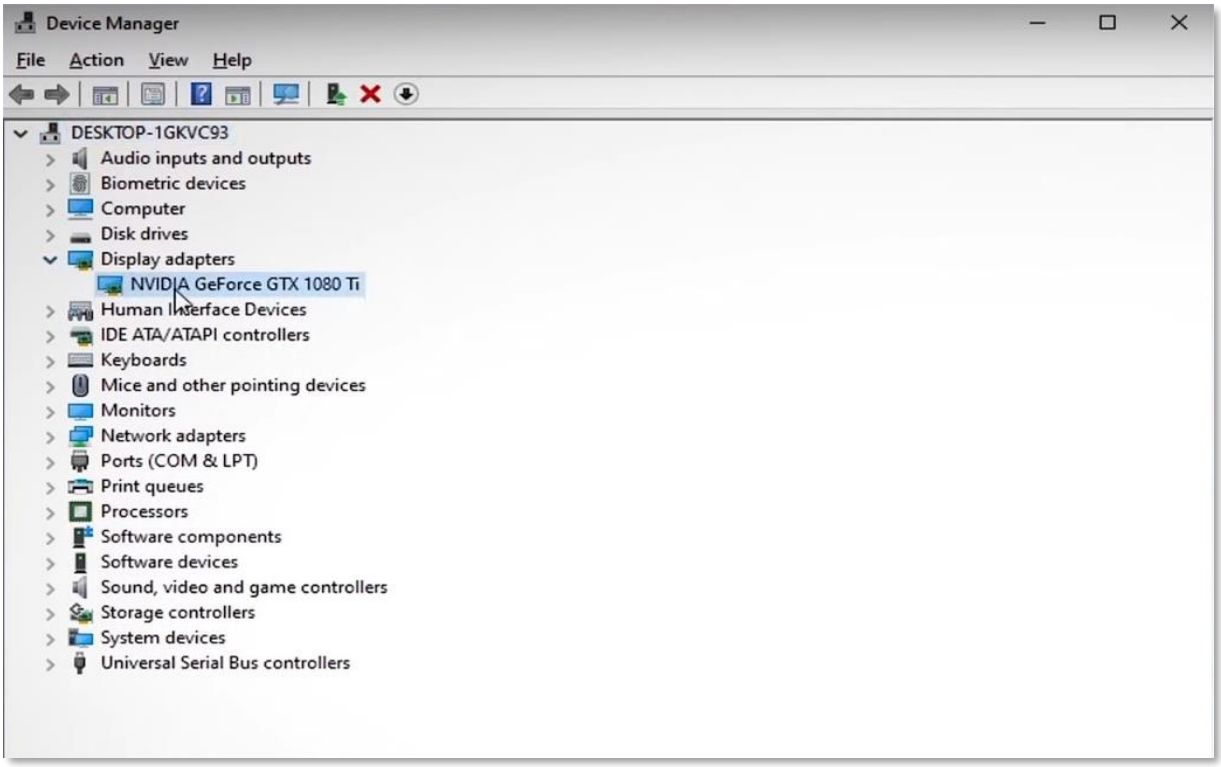
Téléchargez la dernière version de CMake et l'installer. Pendant l'installation, laissez tous les paramètres par défaut. Téléchargez Visual Studio Community 2019. Attendez la fin de l'installation et sur l'écran des outils de développement, sélectionnez « Python development » et « Desktop development with C++ ».



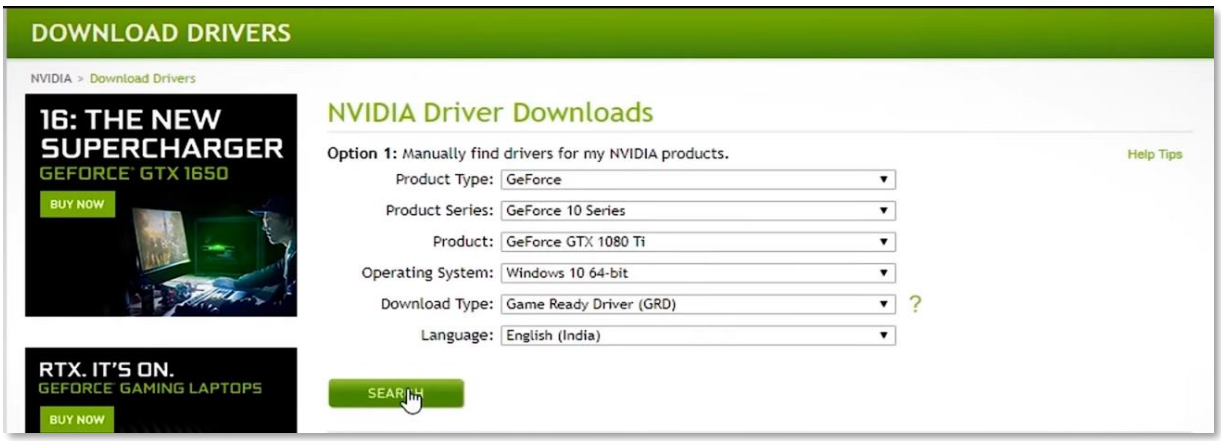
L'installation va prendre un certain temps. Redémarrez l'ordinateur une fois terminée.

3. Mise à jour des pilotes graphiques

Dans la barre de recherche Windows, tapez « gestionnaire de périphériques ». Ouvrez l'onglet « Cartes graphiques ». Il est nécessaire d'avoir une carte graphique NVIDIA pour pouvoir utiliser le plugin avec le GPU.



Notez le nom de la carte et se rendre sur la page web « Download drivers | NVIDIA ». Remplir les champs avec les informations propres à la carte graphique. Faites attention de bien sélectionner « Windows 10 64-bit » et pour le Download Type, « Game Driver Ready » est suffisant. Téléchargez et installez le pilote.



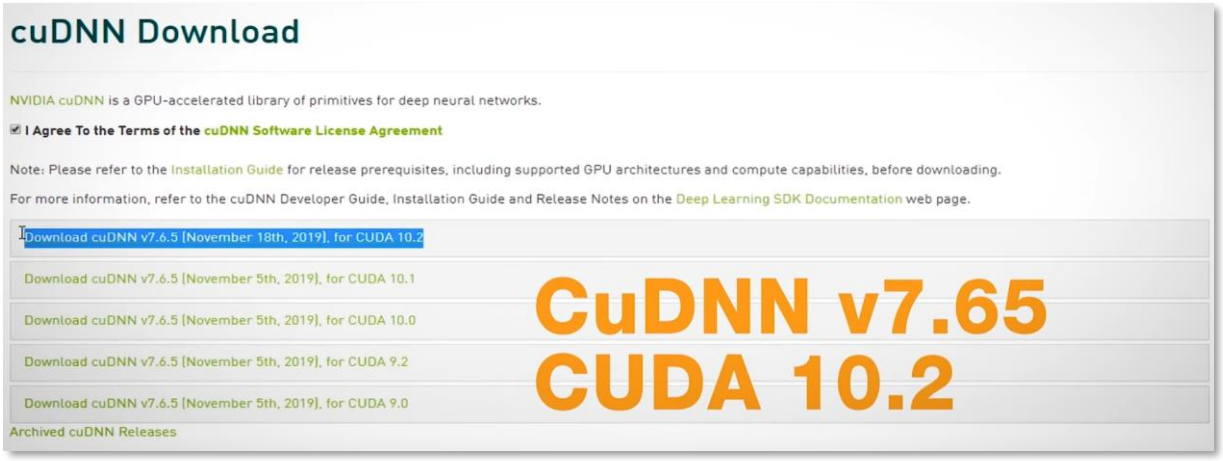
4. Installation de CUDA

Vérifiez sur « CUDA Support Wikipedia » que la carte graphique est compatible avec CUDA 3.0 au minimum. Recherchez « nvidia cuda 10.2 », cliquez sur le premier lien, sélectionnez la version de Windows correspondante et « exe (local) ». Téléchargez CUDA, cela va prendre un certain temps.
Lancez l'installer et changez l'extraction path en « C:\CUDA ».

Dans l'installer, tout laisser par défaut et laisser l'installation se faire.

5. Installation de cuDNN

Tapez « cudnn download » dans Google et allez sur la page « NVIDIA cuDNN | NVIDIA Developer ». Pour télécharger cuDNN, il vous faut créer un compte NVIDIA Developer, puis retournez sur la page et identifiez-vous. Choisissez la version « cuDNN v7.6.5 [November 18th, 2019], for CUDA 10.2 », et cliquer sur « cuDNN Library for Widnows 10 ».

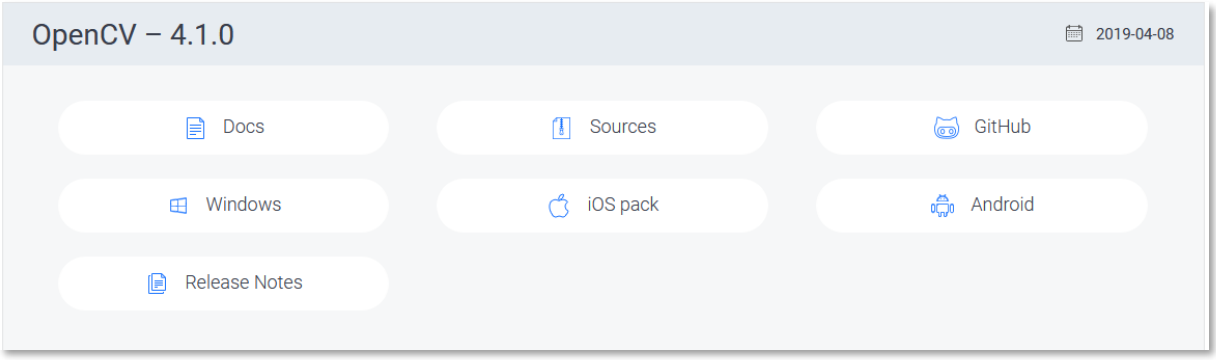


Extraire le fichier compressé.

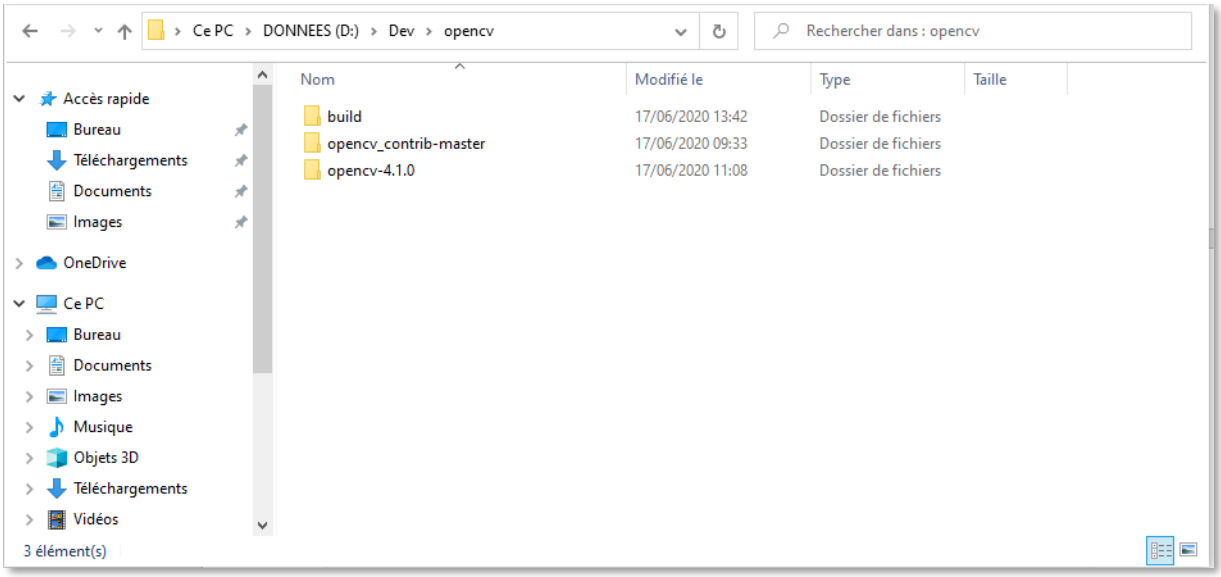
- Dans cuda/bin, copier « cudnn64_7.dll » et le coller dans « C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v10.2\bin ».
- Dans cuda/include, copier « cudnn.h » et le coller dans « C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v10.2\include ».
- Dans cuda/lib, copier « cudnn.lib » et le coller dans « C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v10.2\lib\x64 ».

6. Installation d'Open CV

Allez sur « opencv.org », puis dans l'onglet « Release », choisissez la version 4.1.0 et cliquez sur « Sources ».

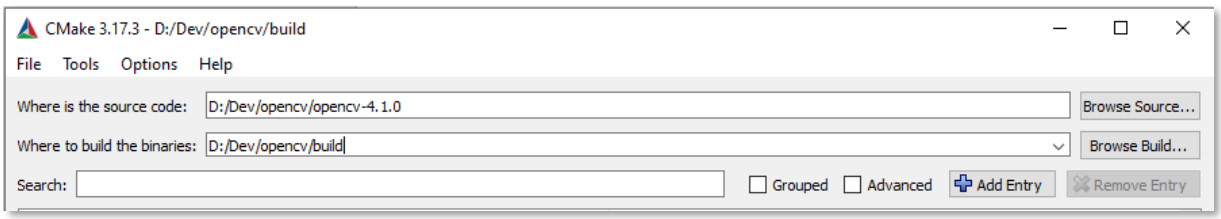


Allez sur « github.com/opencv/opencv_contrib » et téléchargez le zip. Sur le disque C, créez un dossier « opencv ». Dézippez les deux fichier « opencv-4.1.0 » et « opencv_contrib-master » et mettez dans le dossier « opencv ». Créez ensuite un 3ème dossier « build ».

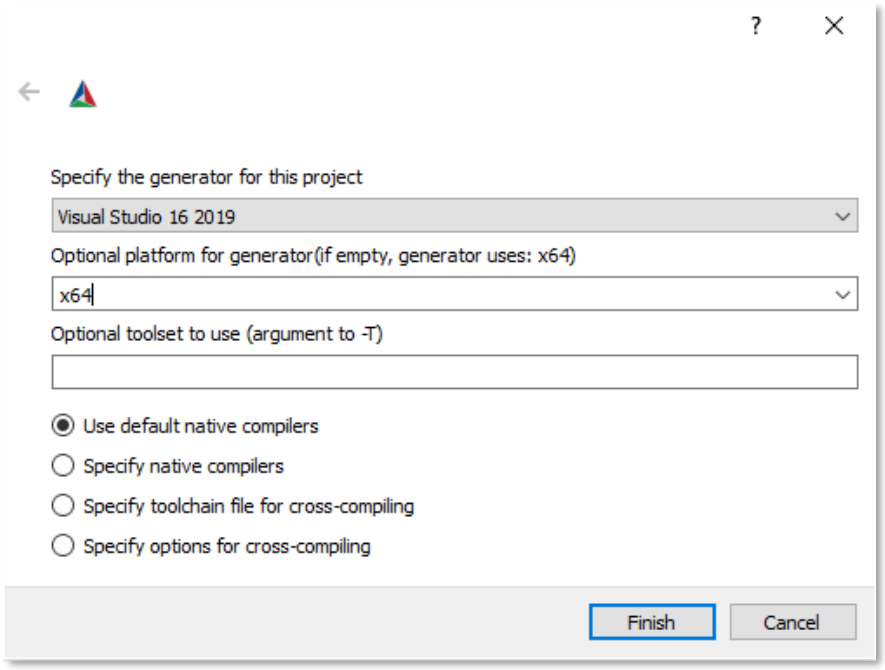


7. Configurer CMake

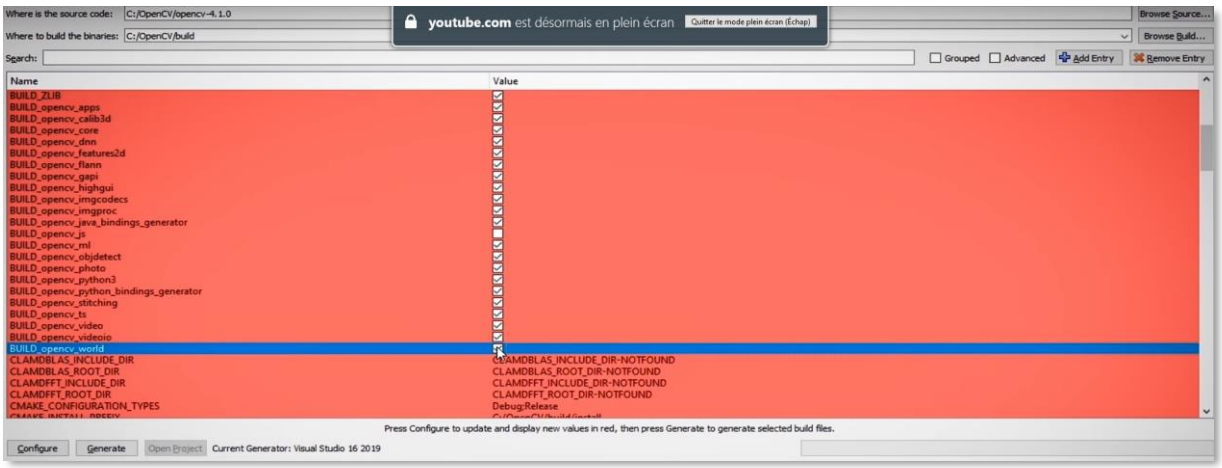
Ouvrez CMake et mettez comme chemin de code source le chemin du dossier « opencv-4.1.0 ». Pour le chemin de build, mettez le chemin du dossier « build » créé juste avant.



Cliquez ensuite sur le bouton « Configure » et choisissez « Visual Studio 16 2019 » ainsi que « x64 ».



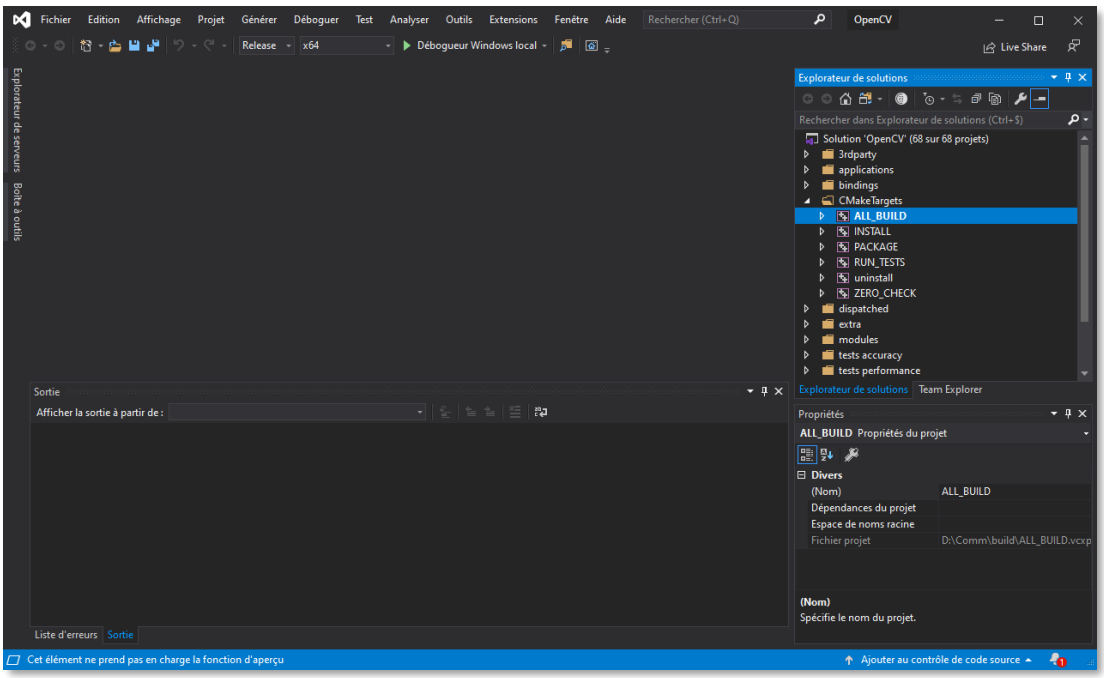
Cliquez sur « Finish ». Une fois la configuration terminée, cherchez dans les modules rouges « BUILD_opencv_world » et cocher la case à droite. Il vaut mieux aller vérifier que la version de Python utilisée est bien la 3.7 installée au début. Pour cela, cherchez dans les modules « PYTHON3_EXECUTABLE » qui aura pour valeur le chemin de la version de Python utilisée par OpenCV. Ce chemin peut être modifié mais il faudra faire de même avec les chemins des autres modules Python3.

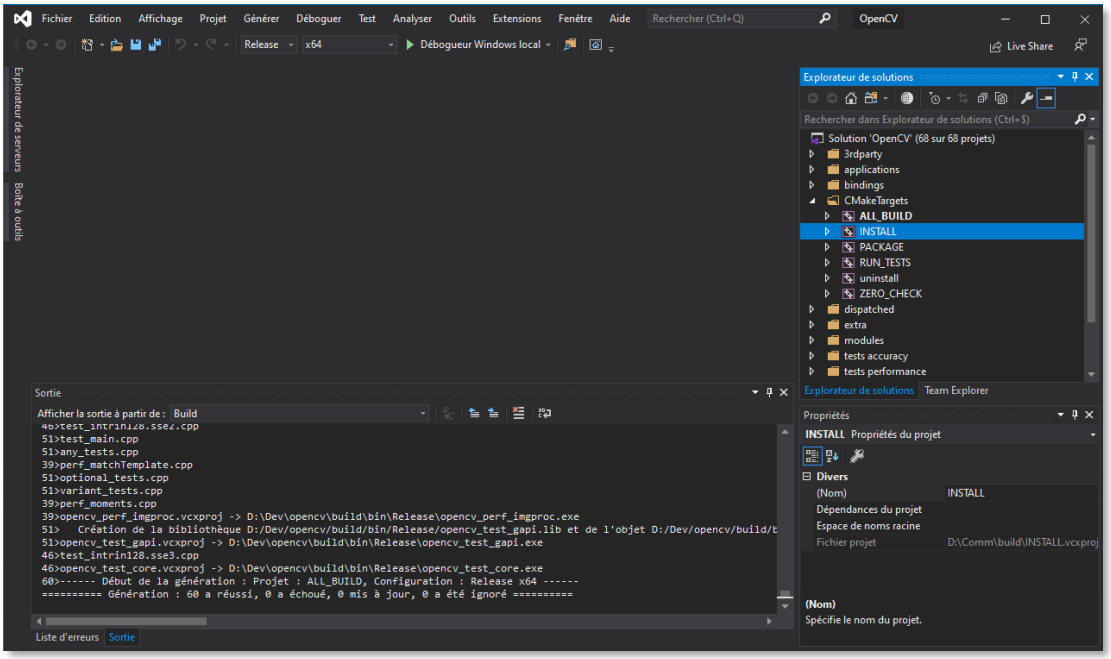


Cliquez ensuite sur « Configure » et si tous se passe bien CMake affichera « Configuring done, Generating done ».

8. Compiler OpenCV avec Visual Studio

Allez dans « opencv/build » et ouvrez « ALL_BUILD.vcxproj » avec Visual Studio. Attention, il faut lancer Visual Studio en tant qu'administrateur. S'il demande de créer un compte, connectez-vous avec un compte Microsoft ou créez-en un.
Mettez « Release » et « x64 » dans le menu en haut. Dans l'explorateur de solution, ouvrez « CMakeTargets », puis clic droit sur « ALL_BUILD » et choisissez « Générer ».





Le résultat doit être aucun échoués et 60 réussis. Générer de même le fichier « INSTALL ». Le résultat doit être aucun échoués, 2 réussis et 58 mis à jour. Si vous n'avez pas le même nombre de succès, pas de souci tant qu'il y a 0 échecs. Vérifiez si Open CV est installé, dans la console taper « python » et tapez « import cv2 ». Si aucune erreur, continuez avec « print(cv2.__version__) » et vérifiez que la

version affichée est bien la 4.1.0. Si oui, Open CV est maintenant correctement installé et compatible avec le GPU de la carte graphique.

9. Installation de Pupil Capture

Sur le site de Pupil Labs téléchargez l'installeur de Pupil Capture. Installez le logiciel et ouvrez Pupil Capture. Normalement, un dossier « pupil_capture_settings » est créé sur dans « C:\Users\{votre nom}\. S'il n'a pas été créé, modifiez des réglages dans Pupil Capture et vérifiez qu'un oculomètre est connecté à l'ordinateur. Lancer un enregistrement peut aider aussi. S'il n'apparait toujours pas il faut le créer à la main.

10. Ajout du plugin

Dans le dossier « pupil_capture_settings », ouvrez le dossier plugin (ou créez-le s'il n'existe pas).
Il faut maintenant coller le dossier « darknet » et le fichier « detection_plugin.py » dans ce dossier.
Si le plugin n'apparait pas à la prochaine ouverture de Pupil Capture, regardez le fichier « capture.log » dans le dossier « pupil_capture_settings » pour comprendre le problème.

11. Erreurs classiques

- Lors de la génération d'OpenCV, vérifiez bien que vous lancez Visual Studio en tant qu'administrateur, puis une fois lancé charger le projet « ALL_BUILD.vcxproj » présent dans le dossier « build ».

- Si lors du chargement du modèle dans pupil capture, une erreur apparait du type « can't find coco.names » ou tout autre fichier .names : aller dans le dossier build/darknet/x64 de darknet, puis dans cfg, ouvrez le fichier .names et dans le champs « names = » mettez le chemin absolu du fichier .names, qui se trouve dans x64/data/coco.names.

- Si vous obtenez l'erreur « try to set subdivisions to 64 », allez chercher le fichier .cfg du modèle que vous chargez (yolov4 pour Yolo 608) et modifiez la valeur du champ subdivisions à 64. Attention il y a des lignes commentées.

- De manière générale les fichiers de darknet que vous devrez modifier seront dans darknet/build/darknet/x64/.

- Si un modèle ne se charge pas, vous pouvez aller vérifier dans detection_plugin.py, dans la fonction load_model(), les valeurs de configPath, weightPath et metaPath et entrer les chemins correspondants aux fichier .cfg, .weights et .data correspondants. Attention pour le .data il faut prendre celui du dossier x64/cfg/.

- Enfin pour modifier la couleur des boites, dans le fichier du plugin, dans la méthode __init__(), vous pouvez changer en entrant une nouvelle valeur BGR (et pas RGB attention).