# **在進行前請先確保TrafficTrackerGUI基本版本(CPU)已經建置成功並可運行**

GUIv3.9.0 - 2023/08/28

GUIv3.10.0 - 2024/09/12 build all processes in windows11 RTX4060 cuda-11.1 success.

# PART 1 :在windows 安裝 opencv with cuda

參考網址 : <https://thinkinfi.com/install-opencv-gpu-with-cuda-for-windows-10/>

並根據所需額外修改細部步驟

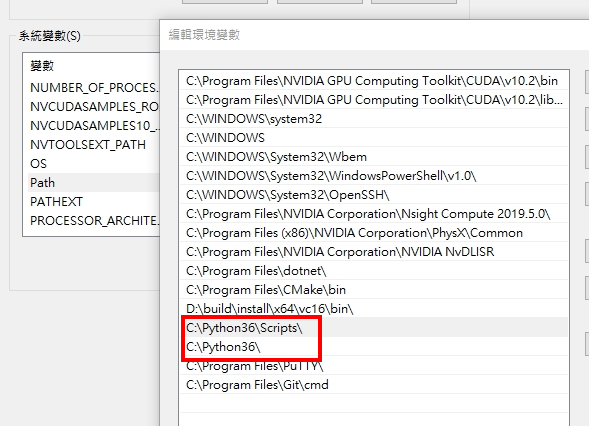
### Step1 | Uninstall anaconda or python and install fresh python**3.6.7** for all user

Follow this step otherwise there can be path issue later.

如版本為3.6.7，且對之後cmake編譯、VS編譯時的Python路徑有把握可以略過

**1.1.** Uninstall python or anaconda whatever you are using and install python a fresh python **3.6.7**.

**1.2.** Check the “system environment variables” make sure older path has been removed and current path is there

****

### Step2 | Install “numpy” and uninstall “opencv-python”, “opencv-contrib-python”

Before compiling make sure “**numpy**” is installed. **numpy一定要裝.**

Make sure that “opencv-python” and “opencv-contrib-python” is uninstalled and will never be installed again using “pip” in this environment again

**2.1.** Install “numpy” (pip install numpy)

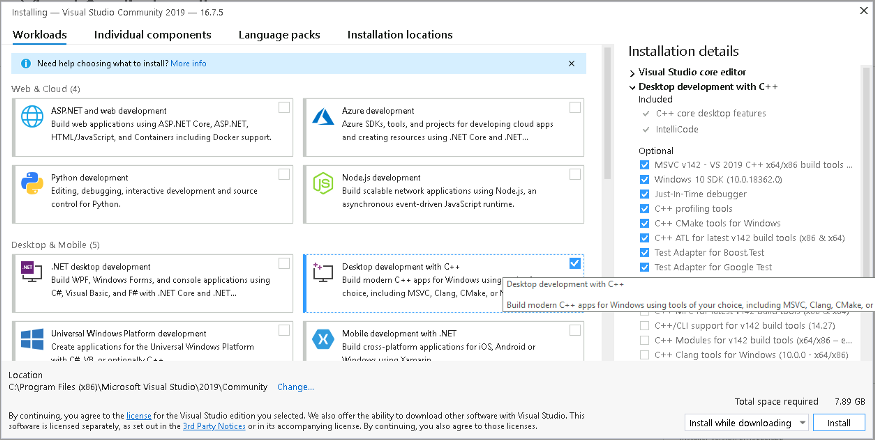
**2.2.** Uninstall opencv if you installed anaconda instead fresh Python (pip uninstall opencv-python)

**2.3.** Uninstall “opencv-contrib-python” if you installed anaconda instead fresh python (pip uninstall opencv-contrib-python)

### Step3 | Download & install Visual Studio

**3.1.** Download Community edition Visual Studio, In my case, I have downloaded Visual Studio 2019: <https://visualstudio.microsoft.com/downloads/>

**3.2.** Check on “Desktop development with C++”, and Continue with defaults and click on install



### Step4 | Download and install CUDA Toolkit according to your GPU

請根據之後要安裝yolo時欲使用之cuda版本**(10.2 RTX10系 或11.1 RTX30系)**，cuda10.2有兩個update請記得安裝

<https://developer.nvidia.com/cuda-toolkit-archive>

**4.3.** Make sure your **CUDA\_PATH** & **CUDA\_PATH\_V1X\_X** is there, if not then add those paths

### Step5.1 | Download cuDNN according to CUDA

To download cuDNN you need to register on the NVIDIA website, then you can download cuDNN: <https://developer.nvidia.com/rdp/cudnn-archive>

**5.1.1.** Download and extract cuDNN (version which corresponds to your suitable Cuda version)

### Step5.2 | Install cuDNN in windows

**5.2.1.** Find CUDA installation folder, In my case: **C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v10.2**

**5.2.2.** Open downloaded and extracted cuDNN folder

**5.2.3.** Copy below files from cuDNN folder and paste on CUDA installation folder

1. Copy all files (one file in my case) from CuDNN **bin folder** and paste inside CUDA (installation folder) **bin folder**
2. Copy all files (one file in my case) from CuDNN **include folder** and paste inside CUDA (installation folder) **include folder**
3. Copy all files (one file in my case) from CuDNN **lib/x64 folder** and paste inside CUDA (installation folder) **lib/x64 folder**

By doing that the installation of cuDNN is now finished.

### Step6 | Download & extract Opencv-**4.5.0** Source from Github

Before compiling make sure you have installed the numpylibrary.

**6.1.** Follow this link: <https://opencv.org/releases/>

**6.2.** Click on sources to download OpenCV **source**

**6.3.** Extract the downloaded folder

### Step7 | Download & extract Opencv-contrib-4.5.0 from Github

We need one extra package which needs to be used along with opencv. This package is called Opencv-contrib.

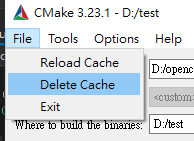
**7.1.** Goto link <https://github.com/opencv/opencv_contrib/tree/4.5.0> then click on **code > download zip**

**7.2.** Extract the downloaded folder

### Step8 | Download & install CMake

**8.1.** Click this link to download CMake: <https://cmake.org/download/>

<<提醒 : 如有發生需重新編譯CMake的情況記得Delete Cache再進行全新的編譯，舊資料有可能會造成編譯失敗或是使用到舊設定。>>



### Step9 | Make a new folder

This is the folder where we will compile and save the object code. **I am giving this folder name as “build”**

### Step10 | Make changes in CMake file “OpenCVDetectPython.cmake”

If we try to compile OpenCV without any changes by default it will search for the **python2 compiler**. We should change the **“OpenCVDetectPython.cmake”** code so that it detects the **python3 compiler by default**.

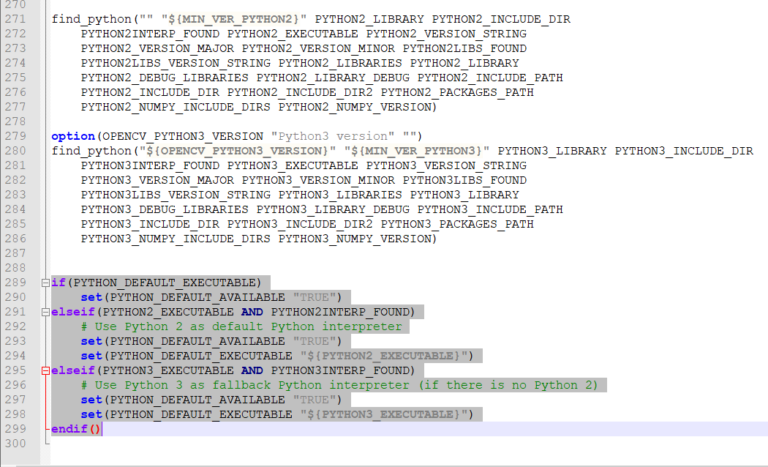
詳情可以上參考網站上看<https://thinkinfi.com/install-opencv-gpu-with-cuda-for-windows-10/>

理論上只要沒有python2應該就沒事，不過保險起見還是改一下

**10.1.** Go inside extracted “opencv-4.5.0” folder

**10.2.** Find and open file “**opencv-4.5.0\cmake\OpenCVDetectPython.cmake**“

**10.3.** Replace last code block of that Cmake file with below code:



**Previous code:**

if(PYTHON\_DEFAULT\_EXECUTABLE)

set(PYTHON\_DEFAULT\_AVAILABLE "TRUE")

elseif(PYTHON2\_EXECUTABLE AND PYTHON2INTERP\_FOUND)

# Use Python 2 as default Python interpreter

set(PYTHON\_DEFAULT\_AVAILABLE "TRUE")

set(PYTHON\_DEFAULT\_EXECUTABLE "${PYTHON2\_EXECUTABLE}")

elseif(PYTHON3\_EXECUTABLE AND PYTHON3INTERP\_FOUND)

# Use Python 3 as fallback Python interpreter (if there is no Python 2)

set(PYTHON\_DEFAULT\_AVAILABLE "TRUE")

set(PYTHON\_DEFAULT\_EXECUTABLE "${PYTHON3\_EXECUTABLE}")

endif()

**Replace with this code:**

if(PYTHON\_DEFAULT\_EXECUTABLE)

set(PYTHON\_DEFAULT\_AVAILABLE "TRUE")

elseif(PYTHON3INTERP\_FOUND)

# Use Python 3 as default Python interpreter

set(PYTHON\_DEFAULT\_AVAILABLE "TRUE")

set(PYTHON\_DEFAULT\_EXECUTABLE "${PYTHON3\_EXECUTABLE}")

elseif(PYTHON2INTERP\_FOUND)

# Use Python 2 as fallback Python interpreter (if there is no Python 3)

set(PYTHON\_DEFAULT\_AVAILABLE "TRUE")

set(PYTHON\_DEFAULT\_EXECUTABLE "${PYTHON2\_EXECUTABLE}")

endif()

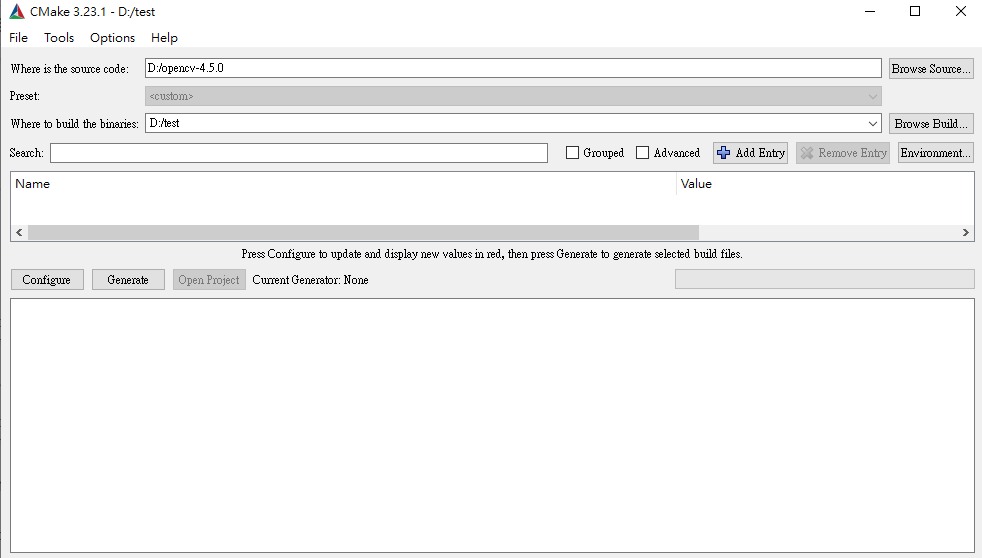
### Step11 | Configure OpenCV and Opencv-contrib using Cmake

Now we need to configure OpenCV by providing paths of CUDA, cuDNN.

**11.1.** Open Cmake app which we have already installed in **Step 8**

**11.2.** Provide path of OpenCV source code which we have already downloaded in **Step 6**

**11.3.** Provide path of ‘build’ folder which we have already created in **Step 9**

****

**11.4. Click on “Configure” button**

**11.5. Then in Configure window select optional platform as x64**

**11.6. Click on “finish” button**

**11.7. Now configure following variables by searching and checking those variable in search tab:**

**1.BUILD\_opencv\_world - Check it**

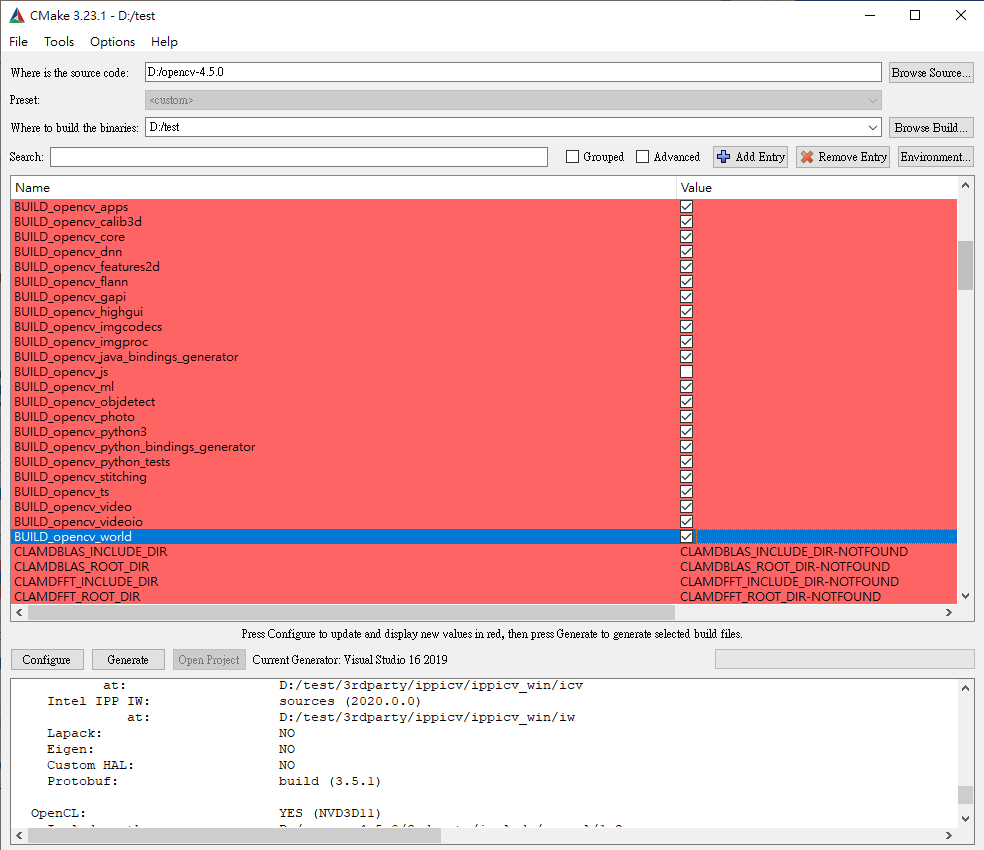
2.ENABLE\_FAST\_MATH — Check it

3.OPENCV\_DNN\_CUDA — Check it

4.WITH\_CUDA — Check it

5.OPENCV\_EXTRA\_MODULES\_PATH — Provide path of **“modules”** directory from “opencv-**contrib**-4.5.0” directory(Step 7)

In my case: "D:\opencv-**contrib**-4.5.0**\modules**"

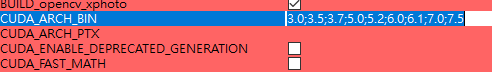


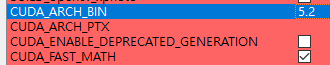
**11.8.** Hit configure button again wait for the “**configuration done**” output

**11.9.** Now we need to configure some more variables

CUDA\_FAST\_MATH — Check it

CUDA\_ARCH\_BIN — 5.2 (This is strictly for my GTX980)



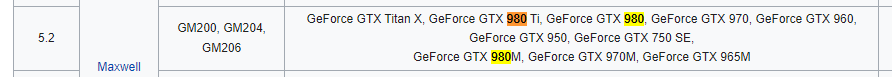


You can visit below Wikipedia page to know your GPU’s **arch\_bin**

Wikipedia link: <https://en.wikipedia.org/wiki/CUDA>

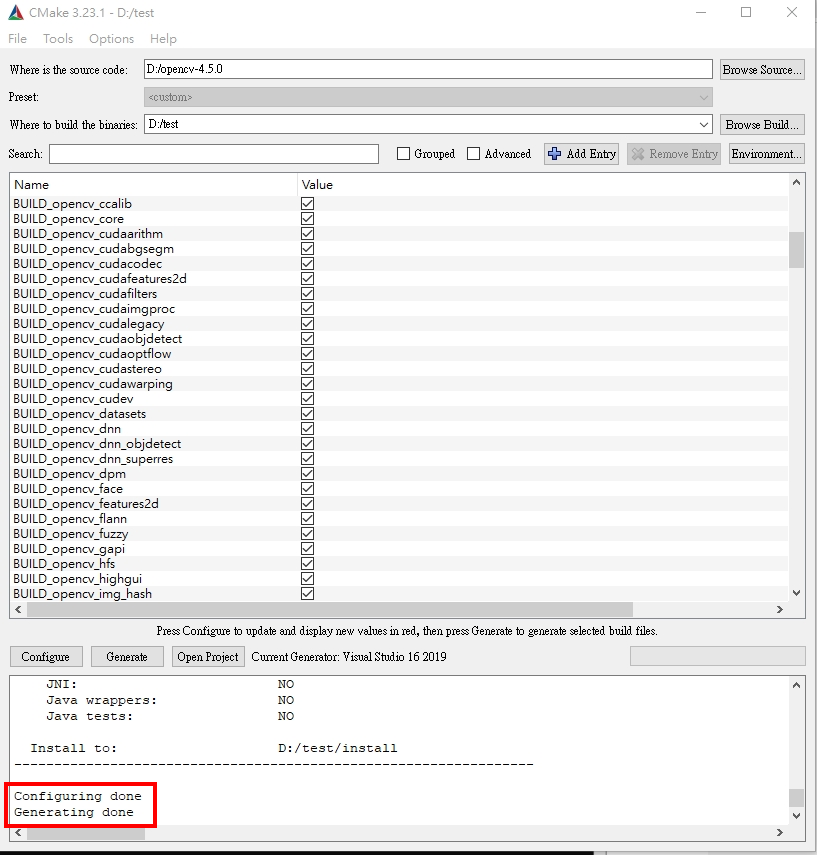
You just visit that page and search for your graphics card model number (Check System Requirement section of this page). **My Graphics card model is GTX 1050 Ti**.

So on that Wikipedia page, I will search for GTX 1050 Ti, and in that row **first column value is CUDA\_ARCH\_BIN** value for my graphics card (GTX980). Which is **5.2**

****

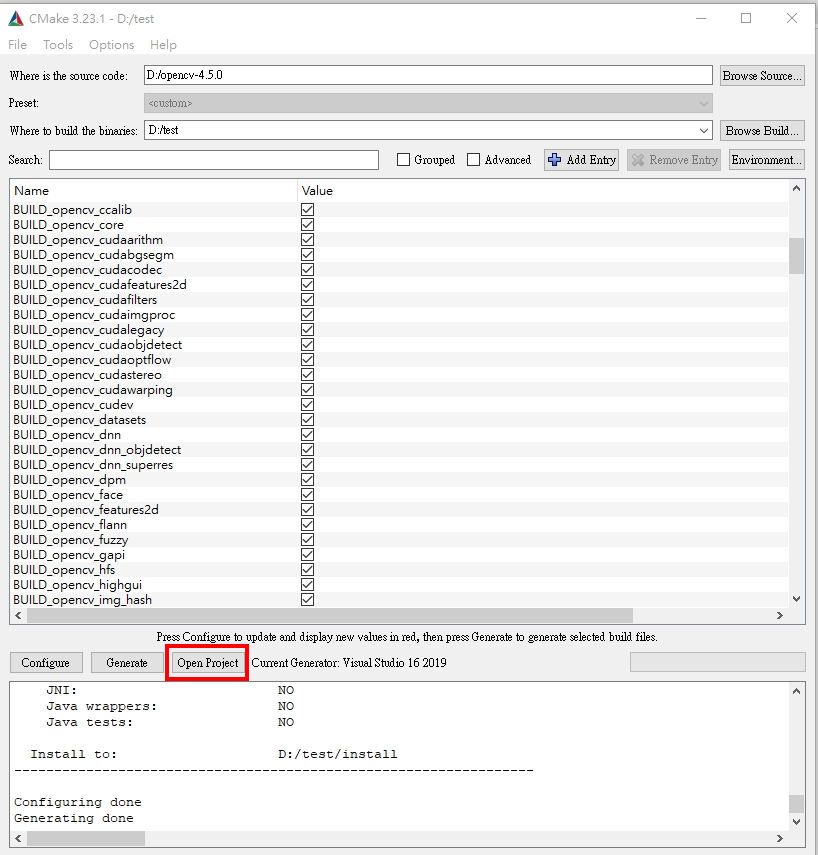
**11.10. Click configure button again and wait for “configuration done” output**

**11.11. After that click the Generate button and wait for “Generating done” output**



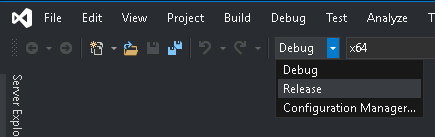
### Step12 | Build The project created by Cmake with Visual Studio

12.1 點擊"Open Project”



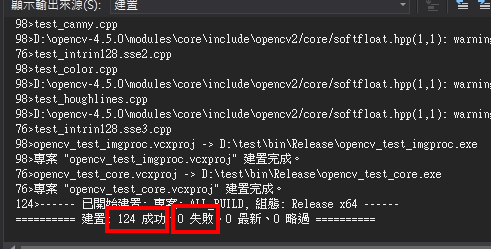
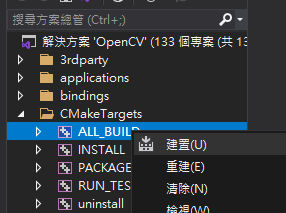
**12.2** Now inside Visual Studio **change “debug” mode to “release” Mode**

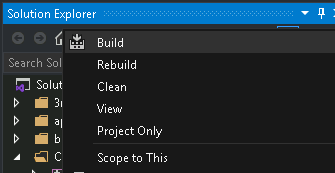
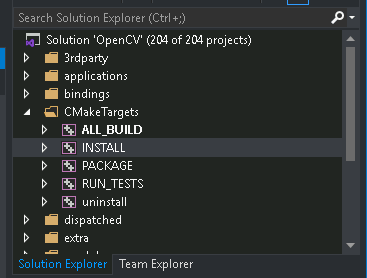
**記得換，不然要從CMake設定開始重新做**

****

**12.3 Inside Visual Studio expand “CMakeTargets” (Located at right)**

**12.4 右鍵 “ALL\_BUILD”並點擊"建置，This may take around 30 minutes to complete.**



**12.7.** Now right-click “**INSTALL**” (from same “CMakeTargets”) and click **build**. 

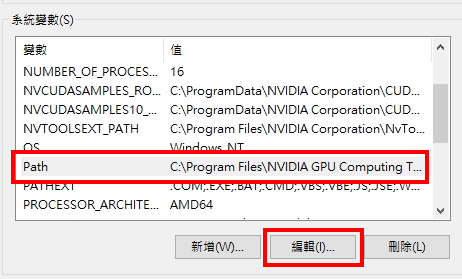


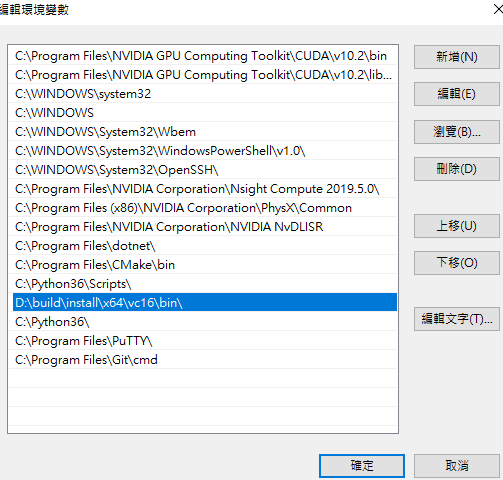
### Step13 | 環境變數

編譯資料夾中的install是opencv with cuda的本體。

確保環境變數 ->系統變數中有加入自己編譯的openCv，其路徑為D:\你自己取的資料夾名\**install**\x64\vc16\bin\

如無請自行添加





### Step14 | cmd測試opencv版本與cuda

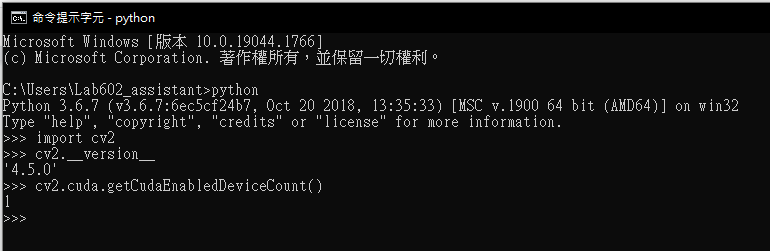
在cmd中輸入 :

python

import cv2

cv2.\_\_version\_\_

cv2.cuda.getCudaEnabledDeviceCount()



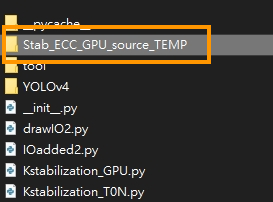
# PART 2 : 在本機環境編譯可用的執行檔

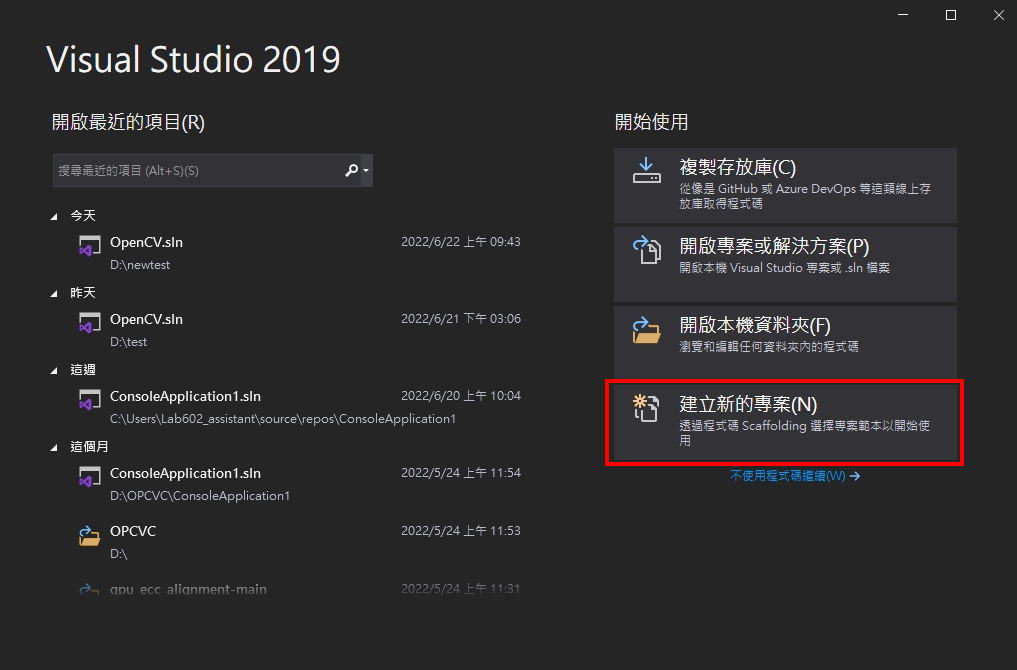
### Step15 | 創建全新的VS C++專案並複製程式檔案

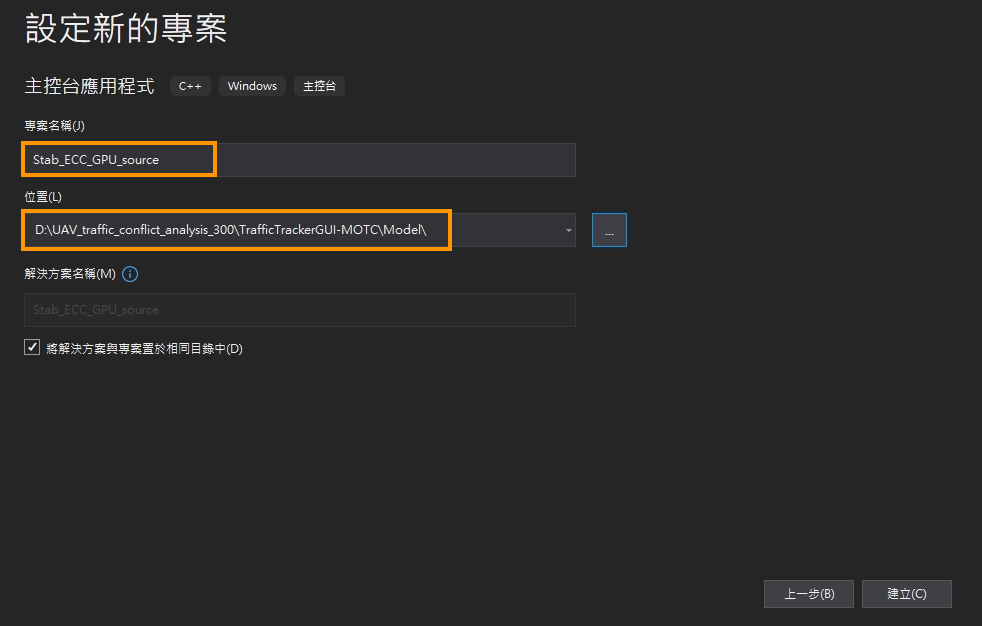
相關檔案會在TrafficTrackerGUI-MOTC/Model/GPU\_Stab\_source/

#### **15.1** 專案名稱請指定為"Stab\_ECC\_GPU\_source"

由於會遇到當前資料夾有相同名稱的問題，請先暫時將存放source code之資料夾改名

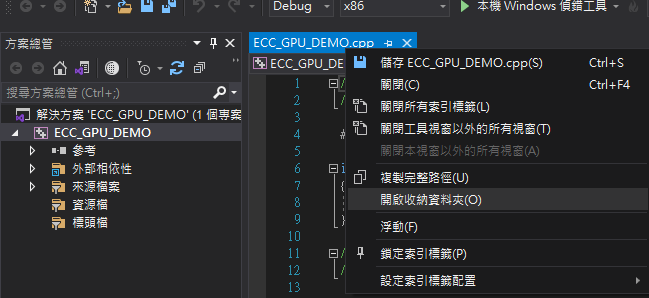






專案名稱 : Stab\_ECC\_GPU\_source  
專案路徑 : UAV\_traffic\_conflict\_analysis\_300\TrafficTrackerGUI-MOTC\Model\

#### 15.2 開啟收納資料夾



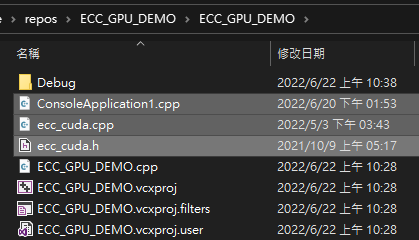
#### 15.3 將以下三個程式檔案複製過去

預設三個檔案位置會在/Model/GPU\_Stab\_source/

**1.eccgpu\_sourceCode.cpp**

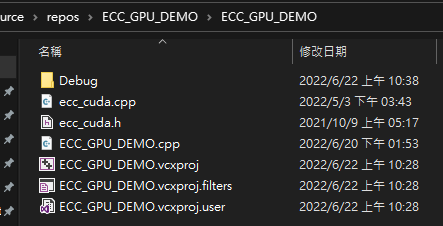
**2.ecc\_cuda.cpp**

**3.ecc\_cuda.h**

****

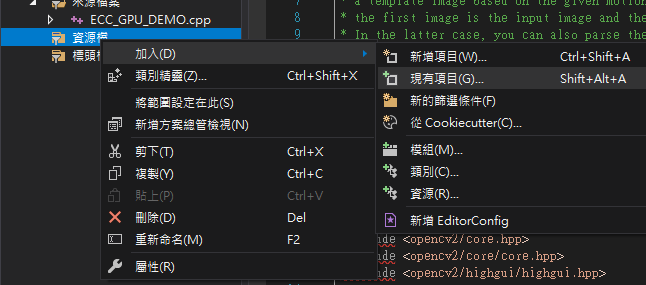
#### 15.4 將eccgpu\_sourceCode.cpp作為main檔

**<方法一>將舊main檔(ECC\_GPU\_DEMO.cpp)刪除後把eccgpu\_sourceCode.cpp更名為ECC\_GPU\_DEMO.cpp，重啟VS專案**

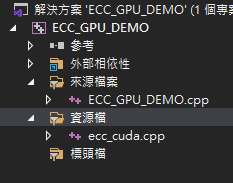
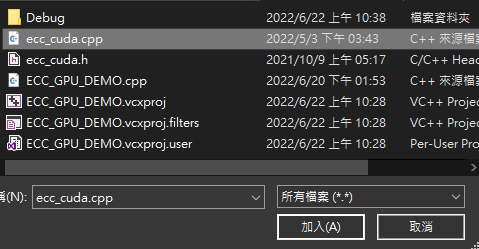
**<方法二>將eccgpu\_sourceCode.cpp程式碼複製並全選取代貼上到ECC\_GPU\_DEMO.cpp**

#### 15.5 資源檔中加入ecc\_cuda.cpp

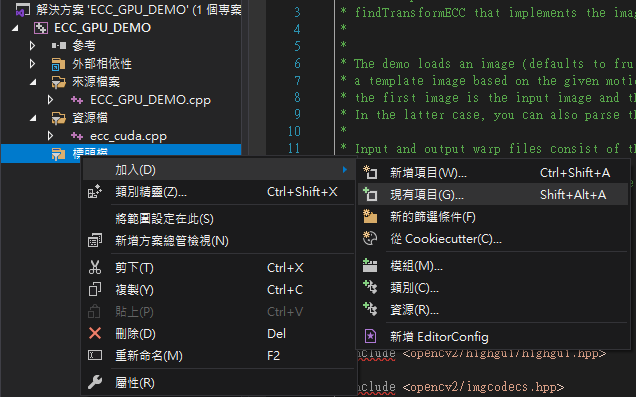
左側欄位 -> 右鍵資源檔 ->加入 -> 現有項目

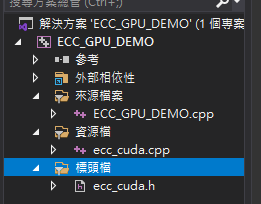
****

**選取 ecc\_cuda.cpp**

****

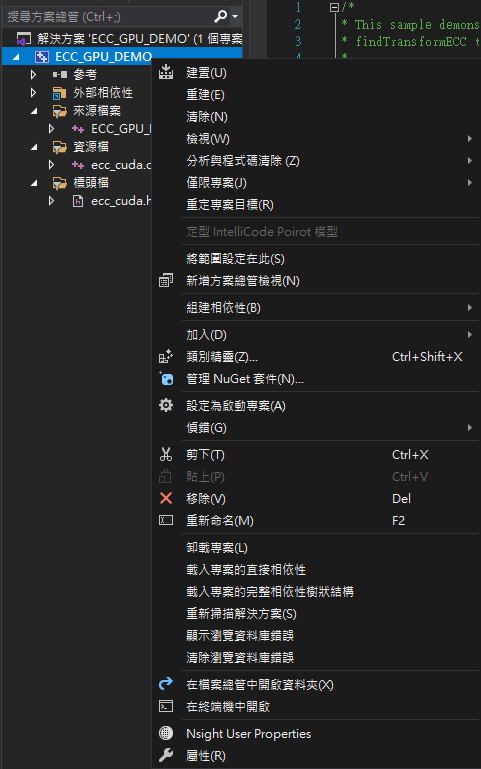
#### 15.6 標頭檔中加入ecc\_cuda.h

****

****

### Step16 | 設定include、link路徑

#### 16.1 右鍵專案 -> 屬性

****

#### 16.2 設定include目錄

**VC++目錄 -> include目錄(左鍵->小箭頭->編輯) -> 加入以下3個路徑 -> 確定**

1. cuda/include

2. opencv with cuda\install\include

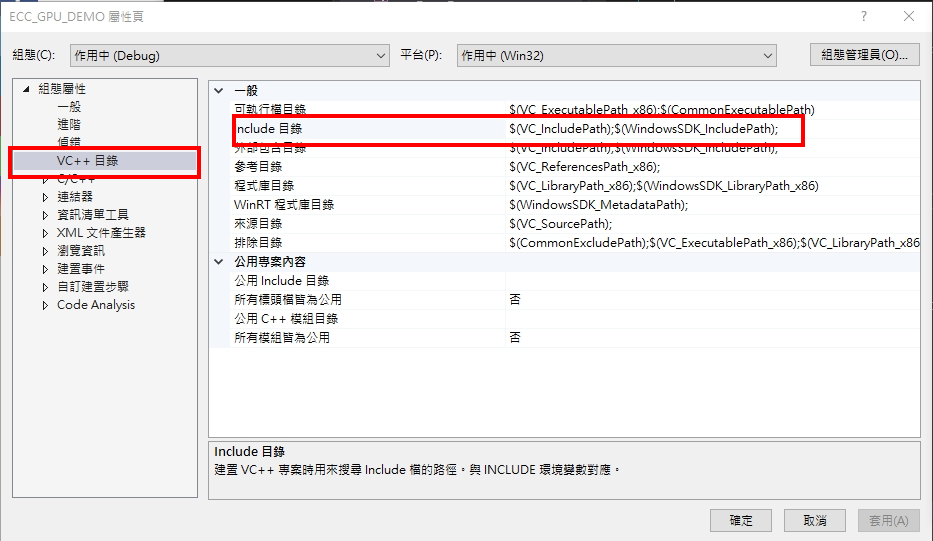
3. opencv with cuda\install\include\opencv2

以我為例分別為

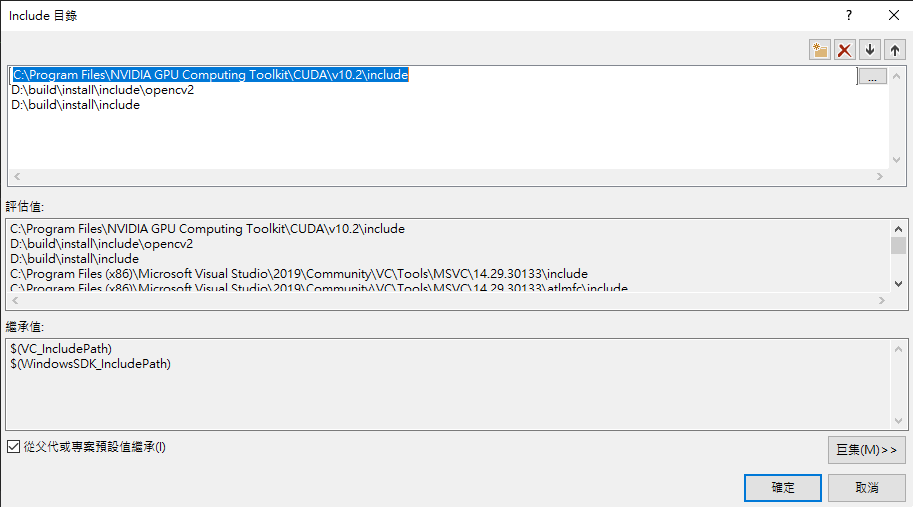
1. C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v10.2\include

2. D:\build\install\include

3. D:\build\install\include\opencv2

****

****

****

#### 16.3 設定程式庫目錄

**VC++目錄 -> 程式庫目錄 -> 編輯 -> 加入以下2個路徑 -> 確定**

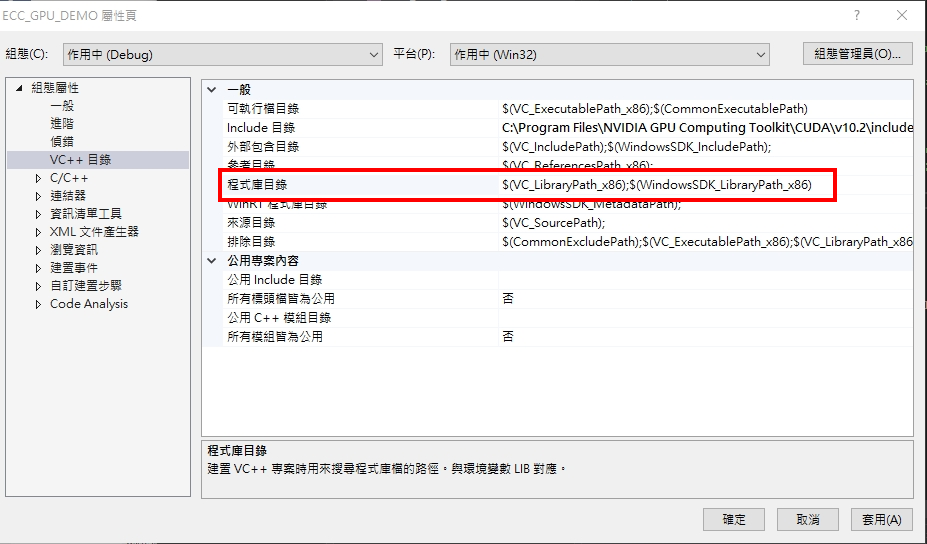
1. cuda/lib/x64

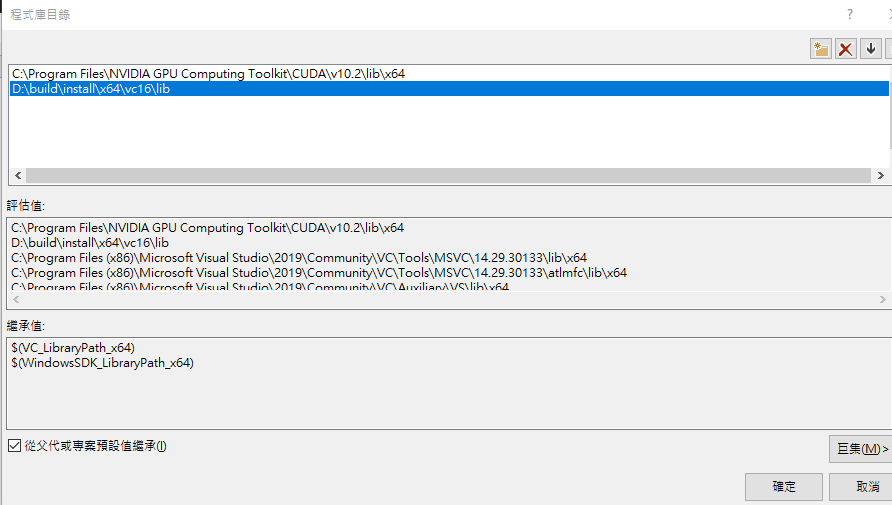
2. opencv with cuda\\install\x64\vc16\lib

以我為例分別為 :

1. C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v10.2\lib\x64

2. D:\build\install\x64\vc16\lib

****

****

#### 16.4 設定"其他相依性"

**連結器 -> 輸入 -> 其他相依性 -> 編輯 -> 將以下lib加入 -> 確定**

**將以下清單複製貼上**

**###由於cuda 11.1** nppicom.lib 已不再使用，因此請將其刪除於清單中###

opencv\_world450.lib

opencv\_img\_hash450.lib

cuda.lib

cudnn.lib

cudart.lib

cublas.lib

nppitc.lib

nppial.lib

nppicc.lib

nppicom.lib

nppidei.lib

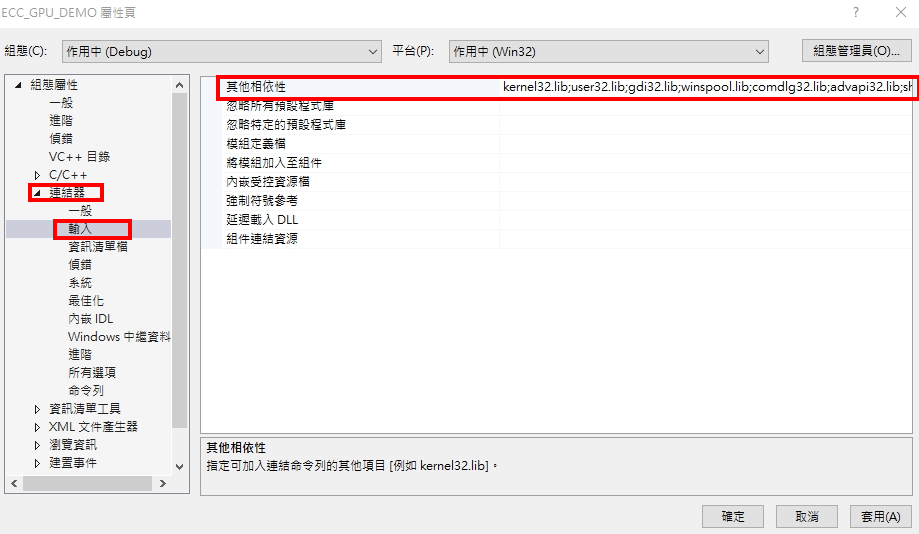
nppif.lib

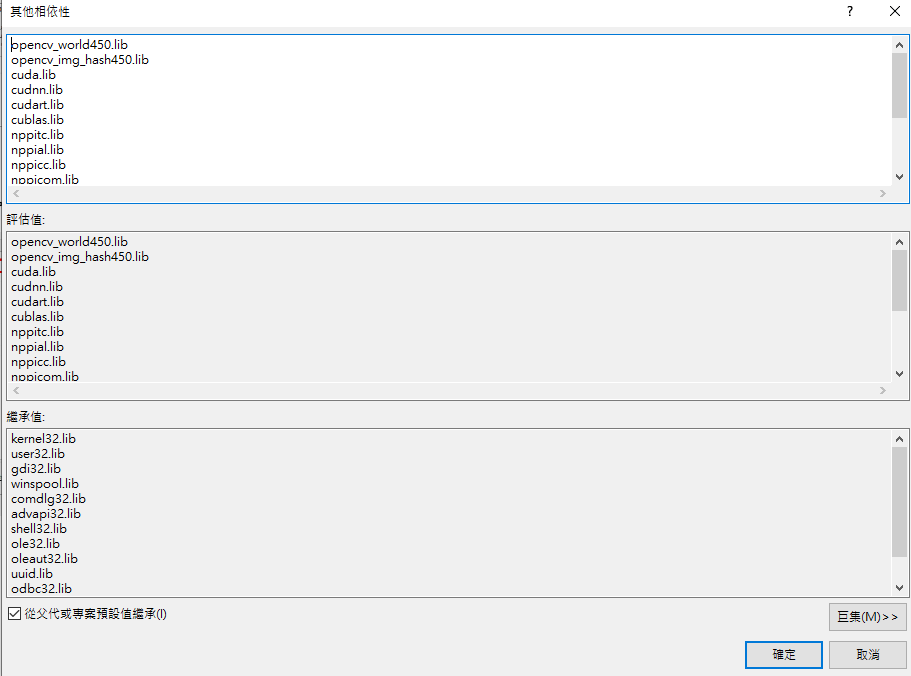
nppig.lib

nppim.lib

nppist.lib

nppisu.lib

****

****

#### 16.5 產生不依賴dll的exe執行檔案

組態屬性 > C/C++ > 程式碼產生 > 執行階段程式庫 > 選擇 多執行緒(/MT)

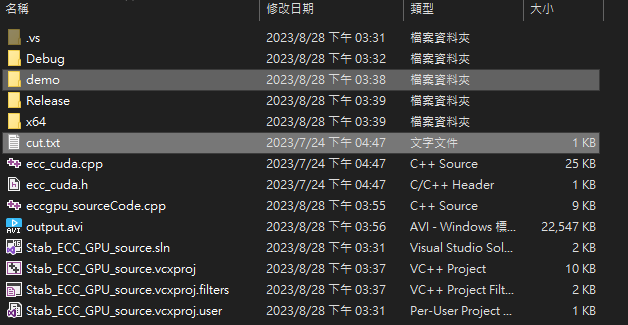
這可以降低exe檔案對於dll動態函示庫的依賴，可能可以直接把exe移植給其他電腦使用而不需重新編譯。

### 

### Step17 | 編譯exe檔

預設會將demo的影片以cut.txt的規格跑一次穩定，並輸出名為output.avi的穩定影片

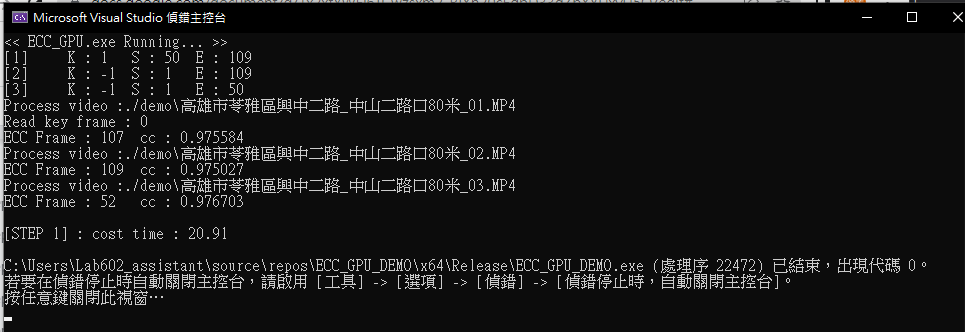
#### 17.1將demo資料夾、cut.txt放入收納資料夾中



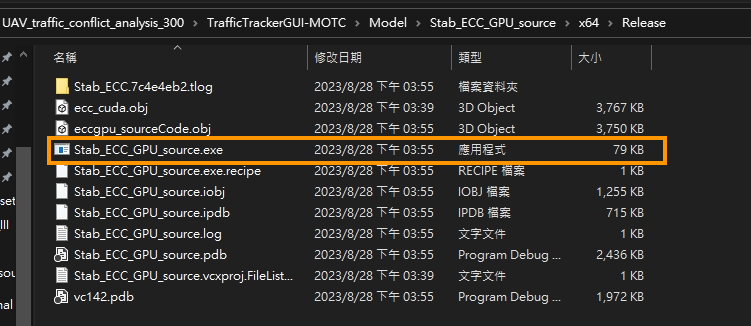
#### 17.2 將組態設為Release x64，點擊本機偵錯

使用Debug可能會因為IDE對於編碼解析外部符號與LINK的嚴謹性而報錯





#### 17.3 確認資料檔案結構



確保編譯出的exe檔位於TrafficTrackerGUI-MOTC\Model\Stab\_ECC\_GPU\_source\x64\Release 當中

Kstabilization\_GPU.py 將會根據該路徑來呼叫GPU\_ECC

執行完成後，其目錄結構大致會呈現如下

=======================================

Model\Stab\_ECC\_GPU\_source\x64\Release

├─ [檔案] cut.txt

├─ [檔案] eccgpu\_sourceCode.cpp

├─ [檔案] ecc\_cuda.cpp

├─ [檔案] ecc\_cuda.h

├─ [檔案] output.avi

├─ [檔案] Stab\_ECC\_GPU\_source.sln

│ ├─ [目錄] .vs/

│ ├─ [目錄] Debug/

│ ├─ [目錄] demo/

│ │ ├─ [檔案] FullDataDemo\_ID-1.mp4

│ │ ├─ [檔案] FullDataDemo\_ID-2.mp4

│ │ ├─ [檔案] FullDataDemo\_ID-3.mp4

│ │ ├─ [檔案] FullDataDemo\_ID-4.mp4

│ │ └─ [檔案] FullDataDemo\_ID-5.mp4

│ ├─ [目錄] Release/

│ ├─ [目錄] x64/

│ │ ├─ [目錄] Release/

│ │ │ ├─ [檔案] Stab\_ECC\_GPU\_source.exe

=======================================

### Step18 | 完成!!

現在可以從原本的專案中使用GPU版本的穩定影片步驟了。