TFG – Arquitectura de computadors i sistemes operatius

FITA#07: *Execució de la PdC: Explotació del model CNN*

Gestió del projecte a GitHub:

Branca del repositori:
https://github.com/UOC-Assignments/uoc.tfg.jbericat/tree/FITA%2307

Dashboard de seguiment de les tasques associades a la fita:
https://github.com/UOC-Assignments/uoc.tfg.jbericat/projects/8

Estudiant: *Jordi Bericat Ruz*

Professor col·laborador: *Daniel Rivas Barragan*

Semestre: *Tardor 2021/22 (Aula 1)*

Versió: *ESBORRANY_v2*

Índex

7. Execució de la PdC: Explotació del model CNN

7.1 - Tasques de recerca i investigació

En aquesta secció les tasques de recerca i investigació es realitzen sobre la marxa per tal d'agilitzar la implementació del codi relacionat, el qual ja conté la documentació de recerca incorporada als comentaris.

7.2 – Preparacions finals per a la execució de la PdC

Per tal d'efectuar la prova de concepte, en primer lloc cal re-ajustar la configuració del simulador AirSim, així com realitzar uns petits canvis a l'entorn `Unreal Engine` personalitzat per a la PdC (veure secció 3).

7.2.1 – Adaptació de l'entorn UE4 "LME" per a la execució de la PdC

En primer lloc, per a situar el/s dron/s de manera que puguin seguir una ruta predefinida tot sobrevolant les zones #6A i #6B[1] de l'entorn desenvolupat a la secció 3, i que inclouen instàncies de les classes d'imatges "high-intensity-wildfires" i "low-intensity-wildfires" respectivament (veure secció 3.3), cal establir les següents coordenades als 4 objectes `PlayerStart` de l'entorn Unreal:

```
X = 34616.03125
Y = -47017.53125
Z = 21851.738281
```

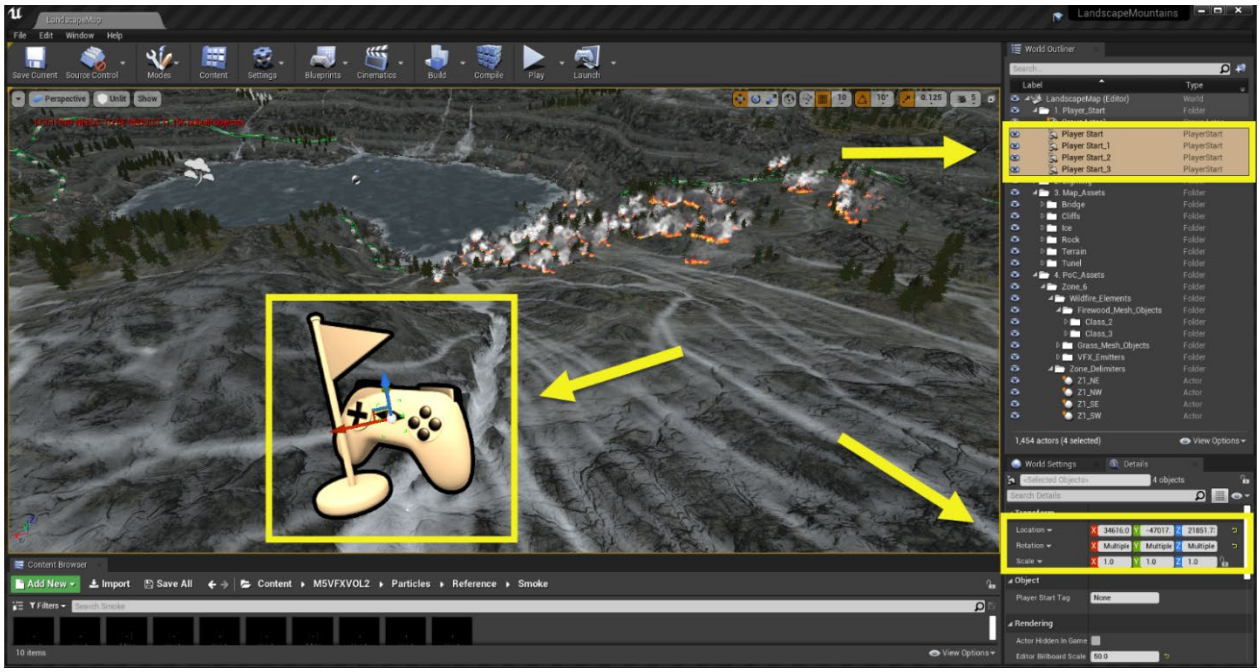

Figura 7.x.x.x *– Posicionament inicial dels drons per a la presa d'imatges durant la execució del model patrulla (`drone_patrol.py`)*

[1] La definició de la zona #6B correspon a un workaround efectuat a darrera hora, fet pel qual no resta documentada a la secció 3.x.x.x

D'altra banda, pel que fa la zona #6B s'ha reutilitzat la configuració d'una de les zones inicialment categoritzades com a `medium-intensity-wilfires` i que s'han acabat integrant com a classe `low-intensity-wildfires`. Per a evitar que es produeixi `overfitting` (memorització de característiques durant l'entrenament) s'ha modificat el posicionament dels objectes `StaticMesh` d'Unreal que simulen calor de la següent manera mitjançant la seva rotació:

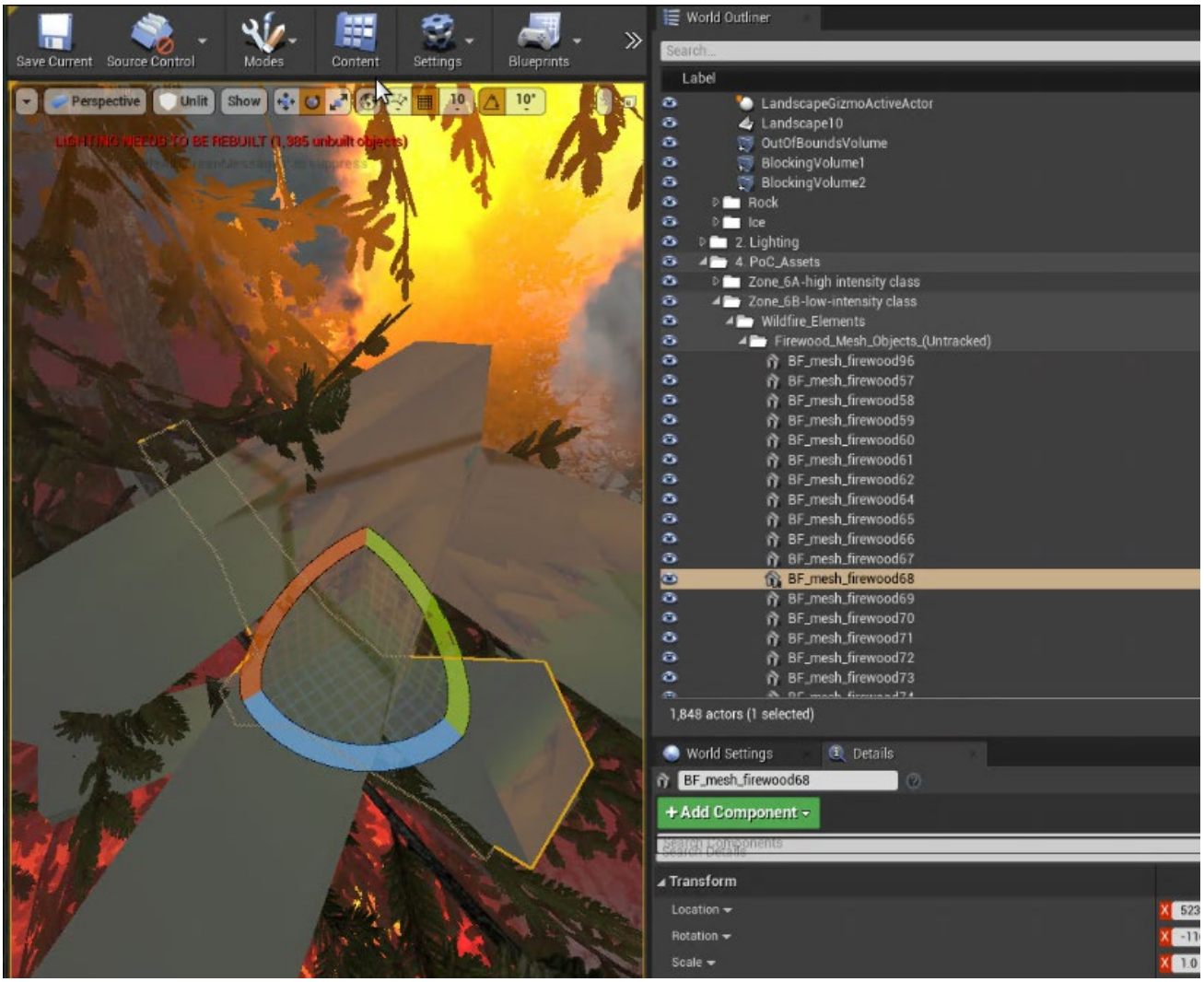

***Figura 7.x.x.x** – Rotació d'objectes `StaticMesh` reutilitzats per a evitar sobre-ajustament (`overfitting`)*

7.2.2 – Ajustos del fitxer de configuració d'AirSim

Pel que fa l'arxiu de configuració d'AirSim `settings.json`, haurem de fer canvis per tal que aquest resti configurat en mode `multirotor` (recordem que el vàrem configurar en mode `Computer Vision` per a construir el dataset d'imatges a la secció 4.x.x.x d'aquest document). D'altra banda, també haurem d'activar la presa d'imatges automàtica del dron (que simularà la gravació d'una càmera de vídeo FLIR tèrmica a 0.5 imatges / frames per segon). L'arxiu de configuració ha de restar com s'indica a la figura 7.x.x.x:

```
{
  "SettingsVersion": 1.2,
  "CameraDefaults": {
      "CaptureSettings": [
        {
          "ImageType": 0,
          "Width": 512,
          "Height": 512,
          "FOV_Degrees": 90,
          "AutoExposureSpeed": 100,
          "MotionBlurAmount": 0
        },
        {
          "ImageType": 7,
          "Width": 512,
          "Height": 512,
          "FOV_Degrees": 90,
          "AutoExposureSpeed": 100,
          "MotionBlurAmount": 0
        }
    ]
  },
  "Vehicles": {
      "Drone1": { "VehicleType": "SimpleFlight", "X": 4, "Y": 0, "Z": -2 }
  },
  "Recording": {
    "RecordOnMove": true,
    "RecordInterval": 0.5,
    "Folder": "/home/jbericat/Workspaces/uoc.tfg.jbericat/usr/PoC/in/",
    "Enabled": true,
    "Cameras": [
        { "CameraName": "3", "ImageType": 7, "PixelsAsFloat": false,  "VehicleName": "Drone1",
"Compress": true },
        { "CameraName": "3", "ImageType": 0, "PixelsAsFloat": false,  "VehicleName": "Drone1",
"Compress": true }
    ]
  },
  "SubWindows": [
    {"WindowID": 0, "CameraName": "3", "ImageType": 7, "VehicleName": "Drone1", "Visible": false,
"External": false},
    {"WindowID": 2, "CameraName": "3", "ImageType": 0, "VehicleName": "Drone1", "Visible": false,
"External": false}
  ],
  "SimMode": "Multirotor"
}
```

figura 7.x.x.x - Arxiu de configuració ~/Documents/AirSim/settings.json

Els dos arxius de configuració d'airsim es poden trobar a les rutes següents del repositori github:

1) Mode "Computer Vision" utilitzat durant la generació del dataset d'imatges:

- `/src/poc/dataset-generator/settings.json`

2) Mode "Multirotor" utilitzat durant la execució de la PdC:

- `/src/poc/cnn-deployment/settings.json`

7.2.3 - Re-estructuració dels arxius font del repositori del projecte

El fet que la gestió d'aquest projecte s'hagi efectuat amb una planificació de tipus "cascada" (tot i que s'han aplicat metodologies "scrum" durant la execució de les sub-etapes de cada fase o fita del mateix) ha promogut el desenvolupament no estructurat de les implementacions relacionades, en funció dels moments de recerca efectuats durant cadascuna de les fases. Aquesta "manera de treballar" però, ha sigut plenament deliberada donat el excessiu consum de temps que requereix fer una planificació amb una granularitat tant fina a les fases inicials del projecte. Així doncs, un cop re-estructurat el codi font en funcions i classes en funció de la seva finalitat, la nomenclatura dels arxius font també ha canviat, fet pel qual es considera oportú incloure el següent control de canvis per tal que les referencies realitzades en seccions anteriors d'aquest document no quedin "orfes".

La estructura final dels arxius del projecte es mostra a a la figura 7.x.x.x de la pàgina següent, mentre que el control de canvis es pot consultar a l'annex 2 (secció 7.7) d'aquest document.

```
src/poc/
        |
        |----- dataset-generator/
        |               |
        |               | -------  airsim_capture.py -> Implements gathering of dataset images
        |               |
        |               | ------- settings.json -> Airsim config file for Computer Vision Mode
        |               |
        |               | ------- airsim_start.sh -> Loads the custom Unreal environment with the Airsim plugin enabled
        |               |
        |               | ------- airsim_capture.sh -> Runs the dataset generator
        |
        |
        |----- cnn-training/
        |               |
        |               | ------- /data/ -> Stores the curated (not raw) datasets for training
        |               |
        |               | ------- hyper_parameter_calculator.wiris -> custom calculator to set hyper-parameters values -> https://calcme.com
        |               |
        |               | ------- pytorch_training.py -> pytorch CNN training implementation
        |               |
        |               | ------- pytorch_training.sh -> Runs the CNN Training
        |
        |
        |----- src/poc/cnn-deployment/
        |               |
        |               | ------- airsim_drone_survey.py (Implements the drone survey around the PoC's Unreal Environment)
        |               |
        |               | ------- cnn_deployment.py -> Implements the CNN model's inference
        |               |
        |               | ------- settings.json -> Airsim config file for "Multirotor" mode
        |               |
        |               | ------- airsim_start.sh -> Loads the custom Unreal environment with the Airsim plugin enabled
        |               |
        |               | ------- airsim_drone_survey.sh -> Runs the drone fly-by around the PoC's custom Unreal deployment environment
        |
        |
        |----- src/poc/lib/
                        |
                        | ------- pytorch.py -> Home-made library that contains auxiliar pytorch related functions
                        |
                        | ------- airsim.py -> Home-made library that contains auxiliar airsim related functions
```

Figura 7.x.x.x – Estructura final dels arxius del projecte

7.3 - Implementacions necessàries per a la execució de la PdC

7.3.1 - Optimització de la funció create_flir_img() per a la simulació d'imatges FLIR

La funció `create_flir_img()` de la llibreria `/src/poc/lib/airsim` que s'ha implementat anteriorment durant la execució de la secció 4.6.1 (originalment pertanyent a l'arxiu `capture_ir_segmentation.py`) presenta un problema o *flaw* important de disseny relacionat amb el rendiment que ofereix l'algorisme de simulació de visió tèrmica nocturna, ja que és molt ineficient en termes de consum de cicles de CPU (es tracta d'un doble loop que fa un recorregut per una matriu). Per tal desenvolupar una solució més orientada a la computació "HPC" i evitar així que aquesta funció es comporti com un coll d'ampolla durant la execució de la PdC, es planteja la idea d'aplicar el desenroscament de bucles niuats (*double "for" loops vectorization*), concepte adquirit durant el transcurs de l'assignatura "Arquitectures de computadors avançades". Concretament, després de realitzar una breu recerca a Internet[2] es troba que la llibreria `numpy` per a `python` incorpora la implementació de mètodes que faciliten en gran mesura aquest tipus d'implementació. Tot seguit es mostra "l'esquelet" de l'algorisme abans i després de realitzar la optimització:

7.3.1.1 - Versió 1: Algorisme sense optimitzar

```
#  looking for value 255 in arr and storing its index in i
for x in range(height):
for y in range(width):
    # getting the THERMAL pixel value.
    p = thermal_image[x,y]
    if (p==255)
        grayscale_image[x,y] = 255
```

Figura 7.x.x.x *– Algorisme que efectua el recorregut per una imatge (matriu de pixels) en cerca de píxels amb valor 255 mitjançant un doble loop*

[2] *Font: Python : For Loops X Vectorization. Make your code run 2000 X faster →* *https://pranoypaul.medium.com/replace-for-loops-in-python-with-vectorized-pandas-dataframes-and-numpy-arrays-e62cf8fbc72a*

7.3.1.2 - Versió 2: Algorisme optimitzat

```
#  Looking for value 255 in arr and storing its index in i
myTuple = np.where(thermal_image == 255)
if ( len(i[0]) ):
    for index in range(len(myTuple[0])):
        x = myTuple[0][index]
        y = myTuple[1][index]
        grayscale_image[x, y] = 255
```

Figura 7.x.x.x *– Versió optimitzada de l'algorisme de la figura anterior*

7.3.2 - Implementació de la funció de simulació d'imatges FLIR "flir_offline_batch_converter.py"

D'una banda, la motivació que m'ha dut a realitzar aquesta implementació de la conversió d'imatges RGB + IR segmentat a simulació de visió tèrmica nocturna "FLIR" **per lots** un cop acabada la recol·lecció d'imatges efectuada pels drons (en comptes de realitzar-la **en temps real** a mida que es van prenent les imatges) esta basada en problemes d'incompatibilitat de les llibreries de airsim 1.6 i pytorch +de cuda 11.4 per a python (versió vinclada a la distribució de Ubuntu utilitzada per al projecte, i que es requisit indispensable per al funcionament d'AirSim). Dit amb altres paraules, no es poden accedir a les llibreries airsim per a phyton de mentres que s'accedeix a les llibreries pytorch (són mutualment exclusives en aquest entorn concret de desenvolupament).

A més, de l'atra banda tenim el fet que la càrrega de pytorch + cuda a la memòria de la GPU requereix fins a 2Gb de la mateixa, mentre que la simulació airsim mitjançant el motor Unreal Engine també requereix gairebé 2Gb de RAM. Tenint en compte que la GPU NVIDIA GTX 1650 utilitzada durant aquest projecte disposa de 4Gb, tenim doncs que es impossible realitzar la inferència d'imatges amb el model CNN utilitzant la GPU mentre que a l'hora es realitza la simulació dels drons amb Unreal Engine. Una alternativa hauria sigut disposar d'una segona GPU treballant en paral·lel a la mateixa màquina de desenvolupament, fet totalment impossible de realitzar degut a les característiques de la placa base utilitzada (2 interficies PCIe 1.0 ja ocupades), però això no salva el problema d'incompatibilitat de llibreries.

La implementació final de la funció `flir_offline_batch_converter.py` un es pot trobar al següent arxiu del repositori git → `/src/poc/lib/airsim.py`

7.3.3 - Implementació de l'algorisme d'inferència del model CNN: "cnn_deployment.py"[3]

Un cop dissenyada i entrenada l'arquitectura de xarxes neuronals (veure secció 5 i secció 6 d'aquest document, respectivament), es el moment de realitzar la implementació de la **inferència** del model sobre un conjunt d'imatges que no pertanyi al datases creat exclusivament per a entrenar-lo (entenem per inferència l'acció de realitzar la predicció d'un conjunt d'imatges per part de la xarxa neuronal entrenada). De partida, aquesta implementació és pràcticament idèntica a la realitzada per tal de testejar el grau de precisió del model rere cada *epoch* d'entrenament (veure secció 5.3) i per tant no es comentaran els detalls de la implementació més enllà de remarcar que per a realitzar la inferència, és estrictament necessari configurar el model en mode d'avaluació mitjançant la directiva `model.eval()` de `pytorch`, ja que de no fer-ho es realitzaria el reajust dels paràmetres (pesos de les arestes) del model, és a dir, l'anomenat efecte de propagació cap al darrera o *gradient back-propagation*.

D'altra banda cal tornar a remarcar que la idea inicial era la de processar les imatges una a una en temps real, però aquesta implementació no ha sigut possible i per tant, la càrrega d'imatges al model es realitza per lots, de la mateixa manera com s'ha implementat a l'algorisme d'entrenament vist a la secció 5.3 mitjançant els mòduls `dataset` i `transforms` de la llibreria `torchvision` per a `python`.

7.3.4 – Implementació de la identificació de les classes detectades pel model mitjançant bounding boxes: funció "add_bounding_boxes()"[4]

En el cas de les dades obtingudes durant la explotació del model per part dels drons no disposarem de les etiquetes o *labels* que indiquen la classe **real** a la que pertany la imatge (com òbviament passa amb qualsevol aplicació de visió per computador durant la seva utilització). Per tant, per tal de poder realitzar la comprovació visual dels resultats de la inferència de les imatges ens caldrà establir algun mecanisme. Tanmateix, com que durant la execució de la secció 5 s'ha decidit implementar solament la classificació d'imatges, i no pas classificació amb localització o bé detecció d'objectes múltiples, aleshores ens caldrà imaginar una manera per a identificar cada

[3] *Fonts: https://towardsdatascience.com/how-to-train-an-image-classifier-in-pytorch-and-use-it-to-perform-basic-inference-on-single-images-99465a1e9bf5*

[4] *Fonts: https://docs.opencv.org/3.4/dc/da3/tutorial_copyMakeBorder.html*

imatge de manera fàcil, ràpida i inequívoca. En aquest sentit s'ha trobat que el framework `openCV`[5] per a python (entre d'altres) ofereix els recursos necessaris per a realitzar aquesta implementació. Abans però, definirem de quina manera identificarem cada predicció:

- Les imatges o *frames* identificats com a classe `no-wildfire` s'identificaran amb un marc (frame) de color **verd**
- Les imatges o *frames* identificats com a classe `low-intensity-wildfire` s'identificaran amb un marc (frame) de color **groc**
- Les imatges o *frames* identificats com a classe `high-intensity-wildfire` s'identificaran amb un marc (frame) de color **vermell**

La figura següent mostra un exemple de cada cas:

Figura 7.x.x.x – D'esquerra a dreta; identificació de les classes `no-wildfire`, `low-intensity-wildfire` i `high-intensity-wildfire` mtjançant marcs de colors implementats amb `openCV`.

Pel que fa la implementació de la funció add_bounding_boxes(), aquesta es troba integrada a l'arxiu on resideix el mateix algorisme d'inferència a la següent ruta del repositori git:

- `/src/poc/cnn-deployment/pytorch_deployment.py`

[5] *per a poder utilitzar openCV amb la versió de python 6.4 s'ha d'instal·lar amb pip:*

```
python -m pip install opencv-python
```

https://stackoverflow.com/questions/50909569/unable-to-import-cv2-module-python-3-6

7.3.5 - Creació d'scripts bash per a l'execució de codi Python

Finalment, pel que fa a les implementacions necessàries per a la execució de la prova de concepte només resta preparar tot un seguit d'scripts de shell `bash` per a cadascuna de les etapes de què consta. Cal observar que alguns d'aquests scripts es podrien haver desenvolupat durant seccions anteriors, fet que no s'ha realitzat degut a la no previsió de la seva necessitat. De fet, la motivació que m'ha dut a pensar en la creació d'aquests scripts ha sigut la de facilitar el canvi de context de l'entorn `conda`[6] necessari per a les diferents implantacions realitzades durant aquest projecte d'una manera àgil i automatitzada (que vinclen tot un seguit de versions de llibreries per a `python`, com s'ha comentat a les seccions 1.3.3.3 i 1.3.3.4 d'aquest document).

```
#!/bin/bash

source ~/anaconda3/etc/profile.d/conda.sh
conda activate condapy373
python3 airsim_capture.py
```

Figura 7.x.x.x – Execució del codi `airsim_capture.py` mitjançant l'script bash `airsim_capture.sh`

```
#!/bin/bash

source ~/anaconda3/etc/profile.d/conda.sh
conda activate py364_clone
python3 pytorch_training.py
```

Figura 7.x.x.x – Execució del codi `pytorch_training.py` mitjançant l'script bash `pytorch_training.sh`

```
#!/bin/bash

source ~/anaconda3/etc/profile.d/conda.sh
conda activate condapy373
python3 airsim_drone_survey.py
```

Figura 7.x.x.x – Execució del codi `airsim_drone_survey.py` mitjançant l'script bash `airsim_drone_survey.sh`

[6] *Font: https://stackoverflow.com/questions/60303997/activating-conda-environment-from-bash-script*

```
#!/bin/bash

source ~/anaconda3/etc/profile.d/conda.sh
conda activate py364_clone
python3 pytorch_training.py
```

Figura 7.x.x.x – Execució del codi `airsim_drone_survey.py` mitjançant l'script bash `airsim_drone_survey.sh`

Un altre detall que s'ha tingut en consideració a posteriori és la d'automatitzar la càrrega dels dos entorns `Unreal` creats per aquest projecte (el primer per a la obtenció del dataset d'imatges, i els segon per a la explotació del model CNN), fet que redueix el temps de càrrega a la meitat (de 6 o 7 minuts a 3 o 4 minuts). Per a tal efecte s'han creat els dos scripts `bash` seguents:

```
#bin/bash!

# Since it was impossible to create the Unreal Engine env. binaries, we'll be running
# AirSim by means of the Unreal Engine Editor itself:

UE4Editor /home/jbericat/Workspaces/uoc.tfg.jbericat/src/UnrealEnvironments/UE-wildfires-map_dataset-generator/LandscapeMountains.uproject
```

Figura 7.x.x.x – Càrrega de l'entorn Unreal per a la obtenció d'imatges del dataset mitjançant l'script bash `airsim_start.sh`

```
#bin/bash!

# Since it was impossible to create the Unreal Engine env. binaries, we'll be running
# AirSim by means of the Unreal Engine Editor itself:

UE4Editor /home/jbericat/Workspaces/uoc.tfg.jbericat/src/UnrealEnvironments/UE-wildfires-map_dataset-generator/LandscapeMountains.uproject
```

Figura 7.x.x.x – Càrrega de l'entorn Unreal per a la explotació del model CNN mitjançant l'script bash `airsim_start.sh`

7.4 - Execució de la PdC

Finalment, arribats a aquest punt només resta efectuar la execució dels scripts que s'han preparat a la secció 7.3.5 per tal de:

1. Carregar l'entorn Unreal personalitzat per a la realització de la PdC mitjançant l'script `/src/poc/cnn-deployment/airsim-start.sh`

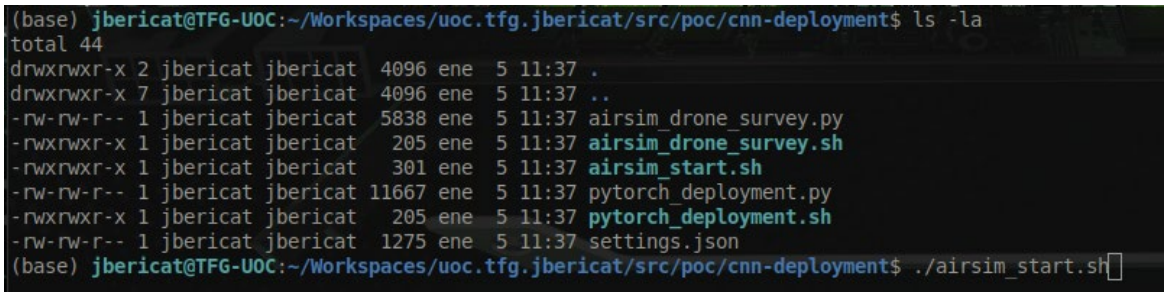

Figura 7.x.x.x – Càrrega de l'entorn Unreal per a la realització de la PdC

2. Iniciar el mode simulació de l'Unreal Engine mitjançant el botó "Play Simulator" de l'editor d'Unreal

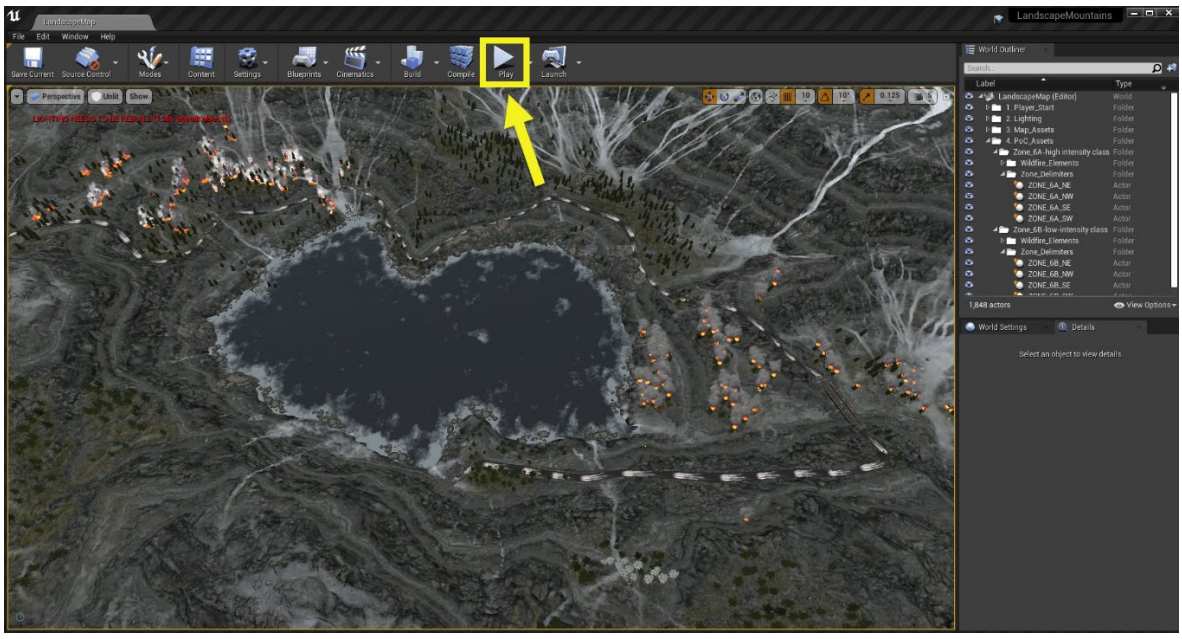

Figura 7.x.x.x – *Inicialització de la simulació Airsim*

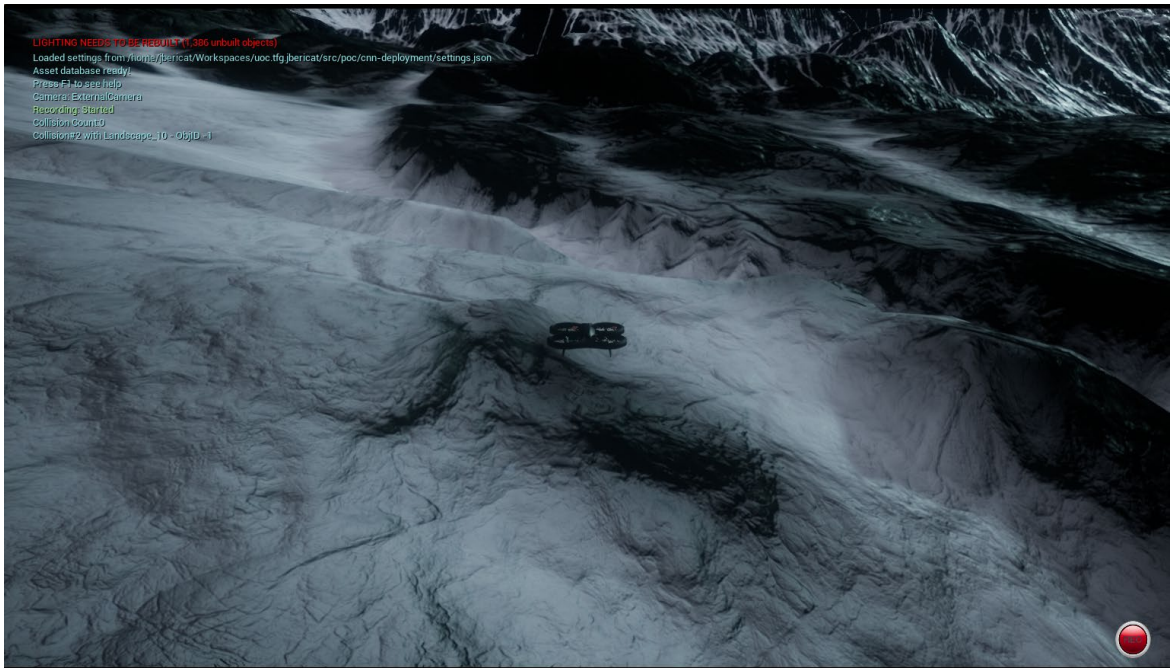

Figura 7.x.x.x – *Simulació airsim preparada i a la espera de rebre comandes per la API d'Airsim*

3. Iniciar el vol del dron mitjançant l'script `/src/poc/cnn-deployment/airsim-capture.sh`

```
(base) jbericat@TFG-UOC:~/Workspaces/uoc.tfg.jbericat/src/poc/cnn-deployment$ ls -la
total 44
drwxrwxr-x 2 jbericat jbericat  4096 ene  5 11:37 .
drwxrwxr-x 7 jbericat jbericat  4096 ene  5 11:37 ..
-rw-rw-r-- 1 jbericat jbericat  5838 ene  5 11:37 airsim_drone_survey.py
-rwxrwxr-x 1 jbericat jbericat   205 ene  5 11:37 airsim_drone_survey.sh
-rwxrwxr-x 1 jbericat jbericat   301 ene  5 11:37 airsim_start.sh
-rw-rw-r-- 1 jbericat jbericat 11667 ene  5 11:37 pytorch_deployment.py
-rwxrwxr-x 1 jbericat jbericat   205 ene  5 11:37 pytorch_deployment.sh
-rw-rw-r-- 1 jbericat jbericat  1275 ene  5 11:37 settings.json
(base) jbericat@TFG-UOC:~/Workspaces/uoc.tfg.jbericat/src/poc/cnn-deployment$ ./airsim_drone_survey.sh
```

```
Connected!
Client Ver:1 (Min Req: 1), Server Ver:1 (Min Req: 1)

arming the drone...
taking off...
camera_response.npy not found. Using default response.
make sure we are hovering at 10 meters...
None
flying-through to the ZONE-6A SW corner...
moveToPositionAsync result: None
flying the perimeter to the ZONE-6A NW corner...
moveToPositionAsync result: None
flying-through again to the ZONE-6A SE corner...
flying-through to the ZONE-6B SW corner...
flying-through to the ZONE-6B NE corner...
```

Figura 7.x.x.x – *Inici de la captura d'imatges per part del dron per a ser posteriorment inferides pel model CNN (1)*

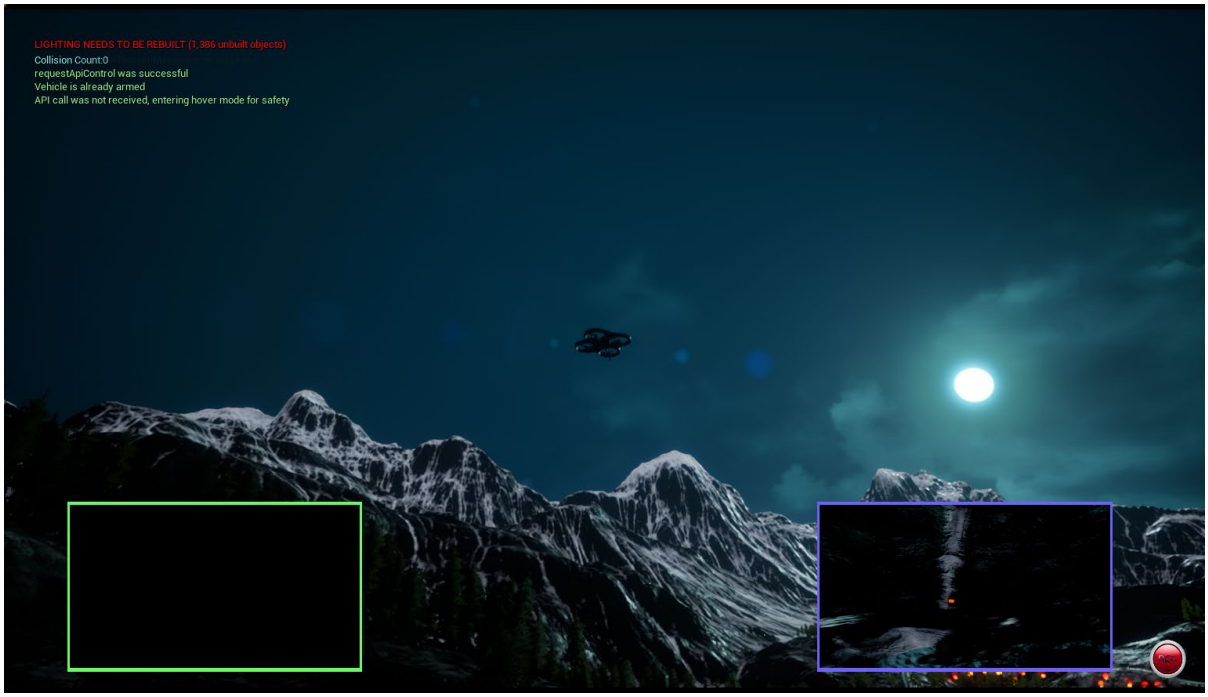

Figura 7.x.x.x – *Inici de la captura d'imatges per part del dron per a ser posteriorment inferides pel model CNN (1)*

4. Efectuar la inferència del model sobre les imatges capturades en el pas anterior mitjançant l'script `/src/poc/cnn-deployment/airsim-start.sh`

```
(base) jbericat@TFG-UOC:~/Workspaces/uoc.tfg.jbericat/src/poc/cnn-deployment$ ls -la
total 44
drwxrwxr-x 2 jbericat jbericat  4096 ene  5 11:37 .
drwxrwxr-x 7 jbericat jbericat  4096 ene  5 11:37 ..
-rw-rw-r-- 1 jbericat jbericat  5838 ene  5 11:37 airsim_drone_survey.py
-rwxrwxr-x 1 jbericat jbericat   205 ene  5 11:37 airsim_drone_survey.sh
-rwxrwxr-x 1 jbericat jbericat   301 ene  5 11:37 airsim_start.sh
-rw-rw-r-- 1 jbericat jbericat 11667 ene  5 11:37 pytorch_deployment.py
-rwxrwxr-x 1 jbericat jbericat   205 ene  5 11:37 pytorch_deployment.sh
-rw-rw-r-- 1 jbericat jbericat  1275 ene  5 11:37 settings.json
(base) jbericat@TFG-UOC:~/Workspaces/uoc.tfg.jbericat/src/poc/cnn-deployment$ ./pytorch_deployment.sh

********************************************************************************

   CNN Deployment: Model inference test for image classification WITHOUT localization
       (That is, we're just adding bounding boxes to identify the WHOLE images)

********************************************************************************

[INFO] - The number of images in a deploy set is: 621

[INFO] - The batch-size is: 128

[INFO] - Model deployed on cuda:0 device

[INFO] - Using the model to make inferences over the bulk data. This might take a minute or two...

[INFO] - The CNN model deployment has finished successfully. See the output .png files to visually check how accurate the predictions were.

[INFO] - OUTPUT FILES:

              - Inference summary: /home/jbericat/Workspaces/uoc.tfg.jbericat/usr/PoC/out/inference-predictions.log

              - Classification results: /home/jbericat/Workspaces/uoc.tfg.jbericat/usr/PoC/out/20220105-211606
```

Figura 7.x.x.x – *Execució de l'algorisme d'inferència del model CNN per tal d'efectuar prediccions sobre el conjunt d'imatges preses durant el vol del dron*

7.5 - ANNEX: Control dels canvis efectuats als fitxers de codi font durant la darrera etapa del projecte

Primera etapa:

```
src/poc/dataset-generator/
             |
             ------- create_ir_segmentation.py -> RENAME -> airsim_set_environment.py
             |
             ------- capture_ir_segmentation.py -> RENAME -> airsim_capture.py

src/poc/cnn-training/
             |
             | ------- /data/ <- CURATED DATASETS
             |
             | ------- hyper_parameter_calculator.wiris
             |
             | ------- pytorch_training.py


src/poc/cnn-deployment/
             |
             | ------- drone_patrol.py -> RENAME -> airsim_drone_survey.py
             |
             | ------- realtime_cv_PoC#1.py (PART I) -> NEW FILE -> airsim_nightvision_simulation.py
             |
             | ------- realtime_cv_PoC#1.py (PART II) -> NEW FILE -> pytorch_deployment.py
             |
             | ------- deploy_poc_one.sh

src/poc/lib/
             |
             ------- /pytorch/cnn_models.py
             |
             ------- /airsim/create_flir_image.py
             |
             ------- etc
```

Figura 7.x.x.x *– Control de canvis en la nomenclatura dels arxius del repositori git d'aquest projecte (primera etapa)*

Segona etapa:

```
src/poc/
        |
        |----- dataset-generator/
        |               |
        |               | ------- create_ir_segmentation.py -> MOVED TO -> src/poc/lib/airsim.py
        |               |
        |               | ------- capture_ir_segmentation.py -> RENAMED -> airsim_capture.py
        |               |
        |               | ------- settings.json -> NEW (Airsim config file for Computer Vision Mode)
        |               |
        |               | ------- airsim_start.sh -> NEW (Unreal Environment loading)
        |               |
        |               | ------- airsim_capture.sh -> NEW (Unreal Environment loading)
        |
        |
        |----- cnn-training/
        |               |
        |               | ------- /data/
        |               |
        |               | ------- hyper_parameter_calculator.wiris
        |               |
        |               | ------- pytorch_training.py
        |               |
        |               | ------- pytorch_training.sh
        |
        |
        |----- src/poc/cnn-deployment/
        |               |
        |               | ------- drone_patrol.py -> RENAME -> airsim_drone_survey.py
        |               |
        |               | ------- realtime_cv_PoC#1.py (PART I) -> NEW FILE -> airsim_nightvision_simulation.py -> MOVED TO -> /src/poc/lib/airsim
        |               |
        |               | ------- realtime_cv_PoC#1.py (PART II) -> NEW FILE -> pytorch_deployment.py -> RENAMED -> pytorch_inference.py
        |               |
        |               | ------- settings.json
        |               |
        |               | ------- airsim_start.sh
        |               |
        |               | ------- airsim_drone_survey.sh
        |
        |
        |----- src/poc/lib/
                        |
                        | ------- /pytorch/cnn_models.py -> MOVED TO -> pytorch.py
                        |
                        | ------- /airsim/create_flir_image.py -> MOVED TO -> airsim.py
                        |
                        | ------- setup_path.py -> MOVED TO -> airsim.py
```

Figura 7.x.x.x *– Control de canvis en la nomenclatura dels arxius del repositori git d'aquest projecte (segona etapa)*