

# TFG – Arquitectura de computadors i sistemes operatius

***FITA#04: Creació del dataset  
d'imatges per a entrenar i testejar el  
model de xarxes neuronals profundes.***

## ***Gestió del projecte a GitHub:***

*Branca del repositori:*

<https://github.com/UOC-Assignments/uoc.tfg.jbericat/tree/FITA%2304>

*Dashboard de seguiment de les tasques associades a la fita:*

<https://github.com/UOC-Assignments/uoc.tfg.jbericat/projects/6>

***Estudiant: Jordi Bericat Ruz***

***Professor col·laborador: Daniel Rivas Barragan***

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

***Versió: ESBORRANY\_v1***

# Index

4 - Creació del dataset d'imatges per a entrenar i testejar el model de xarxes neuronals profundes.	1
4.1 - Tasques de recerca i investigació	1
4.1.1 – Estructura del dataset	1
4.1.2 - Estudi de la API d'AirSim per a Python	1
4.1.3 - Recerca de projectes basats en AirSim	2
4.2 - Establiment dels objectius de classificació	3
4.3 - Definició de l'estructura d'etiquetat del classificador	6
4.3.1. Elaboració de l'estructura d'etiquetat	6
4.3.2 – Representació de l'etiquetat: Estructura de directoris i noms d'arxiu	8
4.3.3 – Quantitat de mostres necessàries per a l'entrenament i testeig del model	9
4.4 – Assignació de les zones definides a l'entorn per a l'obtenció d'imatges	10
4.5 - configuració dels paràmetres de captura de les imatges	13
4.6 – Modificació de l'script Python utilitzar per a la captura d'imatges	15
4.6.1 – Simulació de visió tèrmica nocturna tipus FLIR durant la obtenció d'imatges del dataset	15
4.6.2 – Automatització de la presa d'imatges i del seu etiquetat (labeling)	17
4.7 - Recol·lecció i etiquetat d'imatges (execució de l'script de captura d'imatges capture_ir_segment.py)	20
4.8 - Empaquetament i publicació del dataset d'imatges per a la realització de la PdC	23

## 4 - Creació del dataset d'imatges per a entrenar i testear el model de xarxes neuronals profundes

### 4.1 - Tasques de recerca i investigació

#### **4.1.1 – Estructura del dataset**

L'objectiu d'aquesta fita és el de generar un dataset d'imatges que ens permeti entrenar l'arquitectura NN de manera que:

1. Assoleixi el major rendiment possible (*accuracy*),
2. Es minimitzin les possibilitats de que es produeixi sobre-ajust (*overfitting*)
3. Presenti una estructura estandarditzada
4. Es defineixin clarament cadascuna de les classes
5. S'identifiquin totes les classes mitjançant el corresponent etiquetat de les imatges (*labeling*)

Després de realitzar un procés de recerca, es troba que la informació continguda al següent lloc web proporciona informació rellevant al respecte:

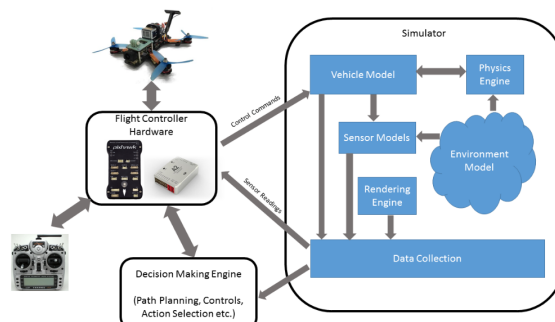
<https://levity.ai/blog/create-image-classification-dataset>

Per tant, **per la realització de les tasques relacionades amb aquesta fita es prendrà l'enllaç anterior com a referència principal** (s'aniran introduint noves referències bibliogràfiques en casos puntuals si és necessari).

#### **4.1.2 - Estudi de la API d'AirSim per a Python**

El paper que juga la API de *AirSim* en l'arquitectura del projecte és el de actuar com a interfície entre l'algorisme de DL/VC al cloud (en el cas que ens ocupa, l'entorn Python del PC de desenvolupament fa de cloud) i els *companion computers* dels drons del simulador (Edge Computing / IoT device + sensors device). Un cop revisada la documentació, es comprova que amb les funcionalitats que proporcionen les crides de la API podrem implementar la part dels requeriments de la PdC que correspon a la interacció amb els drons, concretament:

- 1) Podrem efectuar un control programàtic dels drons
- 2) Podrem obtenir les dades necessàries (imatges, posició GPS, dades sensors, etc) de cada dron



<https://www.microsoft.com/en-us/research/wp-content/uploads/2017/02/aerial-informatics-robotics.pdf>

Els següents enllaços proporcionen tota la informació necessària per a la seva correcta utilització (API versió 1.6.0):

- **Informació general al respecte de la API de AirSim**  
<https://microsoft.github.io/AirSim/apis/#airsim-apis>
- **Documentació del paquet “airsim” per a python**  
[https://microsoft.github.io/AirSim/api\\_docs/html/](https://microsoft.github.io/AirSim/api_docs/html/)

#### 4.1.3 - Recerca de projectes basats en AirSim

Els següents projectes d'exemple proporcionats juntament amb el codi font de la plataforma *AirSim* poden servir de partida per a implementar el codi de la secció 4.6:

- Visió tèrmica nocturna:
  - <https://www.microsoft.com/en-us/research/publication/airsim-w-a-simulation-environment-for-wildlife-conservation-with-uavs/>
  - <https://microsoft.github.io/AirSim/InfraredCamera/>

## **4.2 - Establiment dels objectius de classificació**

Abans de procedir a establir els objectius de classificació i el consegüent etiquetat de les imatges, primer caldrà definir de manera concreta el mode operacional en el que els drons realitzaran la seva tasca, de manera que es generaran imatges que simulin unes condicions d'incendi forestal concretes en funció de quina sigui la activitat que els drons hagin de realitzar. De partida, la intenció inicial era la d'implementar dos modes operacionals diferenciats; 1) Mode de detecció d'incendis a llarga distància i 2) mode d'extinció d'incendis a curta distància. Tanmateix, després de capturar les primeres imatges de prova en mode de detecció a llarga distància es determina que la quantitat d'escenaris diferents que s'haurien de generar per a obtenir una quantitat d'imatges suficient per a entrenar el model i assolir uns certs nivells de rendiment farien impossible realitzar la PdC en els terminis establerts. D'altra banda, la detecció d'incendis mitjançant algorismes d'intel·ligència artificial és un tema ja tractat en altres treballs de recerca [\[LINK NANO CANADIÀ DE LA BIBLIOGRAFIA HERE\]](#). Per tant, per a la PdC em centraré en implementar el **mode operacional 2): Extinció d'incendis a curta distància**.

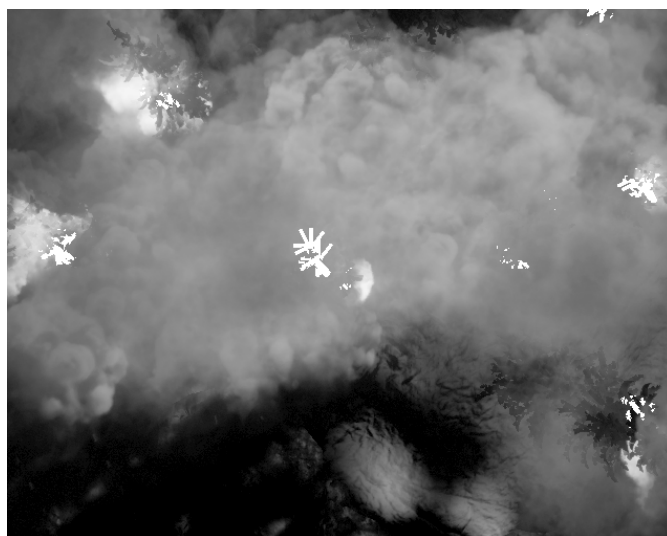
Fet aquest incís i havent deixat clar que les imatges que necessitem per a entrenar el model hauran de ser preses a curta distància, tot seguit es procedeix a establir de manera formal i amb exemples concrets les categories d'incendis (*classes*) que poden arribar ser diferenciades en aquestes condicions, tot tenint en compte les consideracions de disseny de l'entorn fetes a la [secció 3.2.1](#):

### **Classe #1: Incendis d'alta intensitat**



**Figura 4.x.x.x** – `arxiu /usr/datasets/v3.0/training+validation/high-intensity-wildfires/FLIR/FLIR_00021_-998.59998_-316.50000_-20_315_0_0.png`

## Classe #2: Incendis d'intensitat moderada



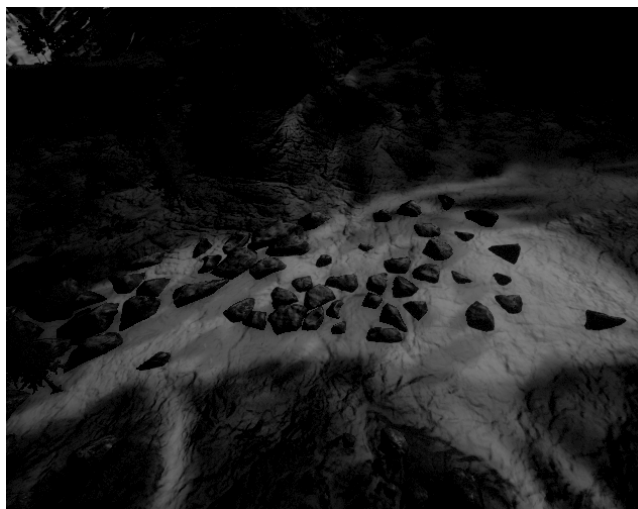
**Figura 4.x.x.x** – *arxiu* `uoc.tfg.jbericat/usr/datasets/v3.0//training+validation/medium-intensity-wildfires/FLIR/FLIR_00036_60.80000_-255.20000_50_270_0_0.png`

## Classe #3: Incendis de baixa intensitat



**Figura 4.x.x.x** – *arxiu* `uoc.tfg.jbericat/usr/datasets/v3.0/test/medium-intensity-wildfires/FLIR/FLIR_00024_-221.09999_36.50000_-20_270_0_0.png`

## Classe #4: Imatges sense incendis



**Figura 4.x.x.x** – *arxiu* `uoc.tfg.jbericat/ /usr/datasets/v3.0/training+validation/no-wildfires/FLIR/FLIR_00008_54.30000_47.60000_-20_315_0_0.png`

Més endavant, ([secció 4.6.1](#)) es donen detalls sobre com s'ha aconseguit simular la visió tèrmica nocturna.

Ara que ja hem definit les classes, podem determinar que els objectius de classificació passaran per determinar si la imatge generada per computador d'un incendi forestal amb condicions d'il·luminació reduïdes i capturada amb una càmera IR tèrmica simulada (FLIR), correspon a alguna de les 4 classes anteriors, que es resumeixen tot seguit:

- **Classe #1:** Incendis d'alta intensitat
- **Classe #2:** Incendis d'intensitat moderada
- **Classe #3:** Incendis de baixa intensitat
- **Classe #4:** Zones forestals sense incendis

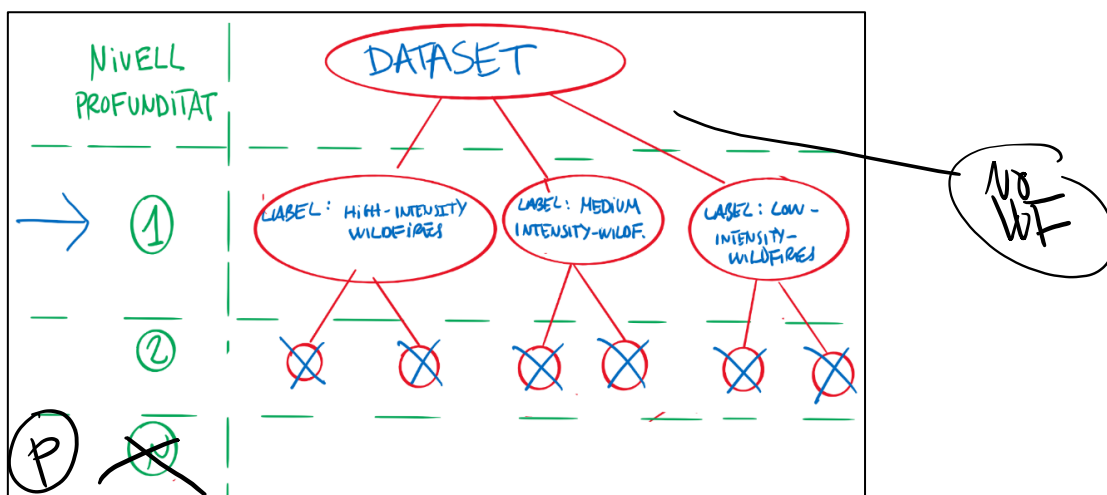
## 4.3 - Definició de l'estructura d'etiquetat del classificador<sup>1</sup>

### 4.3.1 - Elaboració de l'estructura d'etiquetat

Un cop establerts els objectius de classificació, el fet d'elaborar una estructura d'etiquetat d'acord amb aquests objectius és una tasca prèvia que cal dur sempre a terme abans de construir o generar qualsevol dataset d'imatges amb el propòsit d'entrenar un model de xarxes neuronals (si el que es pretén és assolir el màxim rendiment possible durant la explotació del model). Per a efectuar aquesta tasca estudiarem els tres aspectes clau següents:

#### 1. Nivell de granularitat implícit en cada etiqueta

El nivell de granularitat implícit en cada etiqueta defineix la profunditat de la capa de classificació, o dit amb altres paraules, la quantitat de trets diferencials de cada imatge que es tenen en compte per tal de determinar la existència o no de diferents subclasses dins una mateixa classe. En el cas de la PdC, tenim una granularitat que podríem considerar "gruixuda", ja que, com hem vist al punt anterior, la profunditat de la capa de classificació no distingeix entre diferents sub-classes d'incendis per a una mateixa classe (a tall d'exemple; tenim que no hi ha diferents classes d'incendis d'alta intensitat, és a dir que tots els incendis d'alta intensitat tindran la mateixa etiqueta). La següent estructura en forma d'arbre mostra el nivell de profunditat i la consegüent granularitat de cada etiqueta:



**Figura 4.x.x.x** – Profunditat de la capa de classificació (font: elaboració pròpia - ESBORRANY)

<sup>1</sup> Referències bibliogràfiques → <https://levity.ai/blog/create-image-classification-dataset>



## 2. Parts de cada imatge que poden ser representades per la etiqueta escollida

En primer lloc caldria determinar les característiques (*features*) de cada imatge que activaran el classificador del model. Concretament, hem de decidir si 1) etiquetarem les característiques implícites en cada classe per separat a mode de sub-classes, o bé; 2) tractarem el conjunt de les característiques com a un tot, de manera que quedin representades en una sola classe.

En el cas d'aquesta PdC, com que la classificació es realitza en base a imatges que simulen visió tèrmica nocturna, generades de manera sintètica mitjançant objectes geomètrics amb el motor gràfic *Unreal Engine*, aleshores les característiques que tindrem en compte de cada imatge seran:

1. La detecció de punts que emeten calor (píxels o amb valor #FFFFFF)
2. La forma (*shape*) que conformen aquests píxels per a determinar la classe de cada incendi.

Tanmateix, com que he ja hem determinat al punt 1 anterior que la granularitat de cada classe estableix una profunditat d'una sola capa, aleshores tenim que haurem de tractar tot el conjunt de característiques com a un bloc indivisible per tal determinar el resultat de la classificació. A tall d'exemple:

```
SI { Característica #1 == Conjunt de píxels #FFFFFF } I A MÉS { Característica #2 == Forma geomètrica X } → ALESHORES → { Classe = Alta intensitat }

SI { Característica #1 == Conjunt de píxels #FFFFFF } I A MÉS { Característica #2 == Forma geomètrica Y } → ALESHORES → { Classe = Intensitat moderada }

SI { Característica #1 == Conjunt de píxels #FFFFFF } I A MÉS { Característica #2 == Forma geomètrica Z } → ALESHORES → { Classe = Baixa intensitat }
```

**Figura 4.x.x.x** – Conjunt de característiques que defineixen cada classe

## 3. Nombre total d'etiquetes

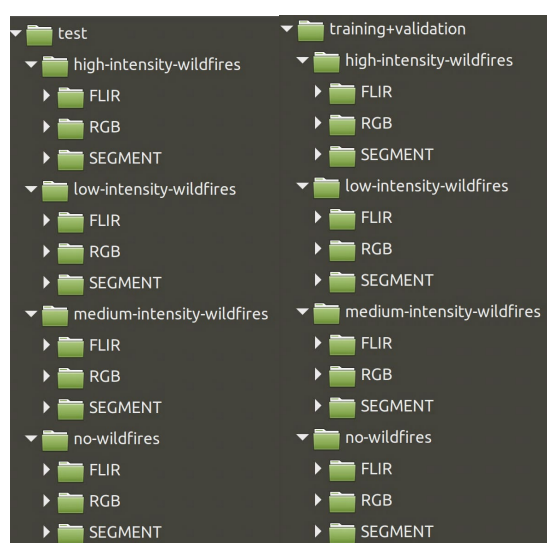
El nombre total d'etiquetes que obtindrem dependrà de la quantitat de classes definides per a cada nivell de profunditat. En el cas que ens ocupa, com que ja hem establert que la profunditat de la capa de classificació serà "superficial" (una sola capa), aleshores tindrem només una etiqueta per classe, quatre en total:

Classe	Etiqueta
#1	/high-intensity-wildfires/
#2	/medium-intensity-wildfires/
#3	/low-intensity-wildfires/
#4	/no-wildfires/

### 4.3.2 – Representació de l'etiquetat: Estructura de directoris i noms d'arxiu

#### Etiquetes = directoris

Un cop fetes les consideracions anteriors ja podem establir l'estructura de directoris del dataset. Concretament, aprofitarem que tant la estructura de classes que hem definit com una estructura de una directoris d'un sistema d'arxius tenen ambdues forma d'arbre, de manera que podem representar cada classe (node de l'arbre) com a un directori on s'hi recullin totes les imatges d'aquesta classe en particular. Tanmateix, com veurem a la [secció 4.5](#) i amb més detall a la [secció 5.2.3.2](#), també necessitarem fer una divisió del catàleg d'imatges del dataset en **dos subgrups, un per a l'entrenament i validació del model i un altre per al testeig**. Per tant, i de cara a formalitzar la estructura de directoris del dataset complet que s'utilitzarà per la PdC, s'estableix doncs el següent arbre de directoris, on s'hi contindran totes les imatges recollides amb el format que s'indica en el següent punt (metadades).



**Figura 4.x.x.x** – Estructura de directoris final del dataset

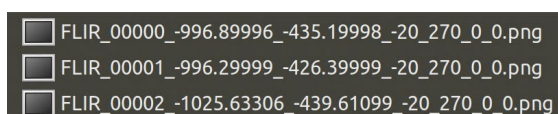
#### Metadades = noms d'arxiu

Pel que fa a d'altres metadades relacionades amb les condicions en les que s'han pres les imatges (definides amb més detall a la [secció 4.6.2](#)), aquestes s'inclouran en el nom d'arxiu de cada imatge amb la següent estructura:

`TYPE_ID_X-pos_Y-pos_HEIGHT_PITCH_ROLL_YAW.png`

Camp	Valor	Significat
TYPE	{ RGB, SEGMENT, FLIR }	Indica el tipus d'imatge: RGB → Imatge en color (3 canals) SEGMENT → Imatge segmentada en B/N (1 canal, valor de píxel binari) FLIR → Simulació de visió tèrmica, escala de grisos (1 canal)
ID	Integer	Identificador únic (seqüencial) de cada imatge
X-pos	Float	Indica la posició X en el pla de l'entorn Unreal on s'ha pres la imatge
Y-pos	Float	Indica la posició Y en el pla de l'entorn Unreal on s'ha pres la imatge
HEIGHT	Integer	Indica l'alçada de la càmera en el moment de prendre la captura
PITCH	Integer	Indica l'angle de la càmera (en graus) respecte del seu eix de costat a costat ( <i>side-to-side</i> ) <sup>2</sup>
ROLL	Integer	Indica l'angle de la càmera (en graus) respecte del seu eix frontal a darrer ( <i>front-to-back</i> )
YAW	Integer	Indica l'angle de la càmera (en graus) respecte del seu eix vertical

La següent captura mostra un subconjunt d'imatges d'una mateixa classe, on s'observen les metadades associades a la imatge incloses en el nom d'arxiu:



**Figura 4.x.x.x** – Metadades de posicionament geogràfic i angle de captura incloses al nom d'arxiu cada imatge

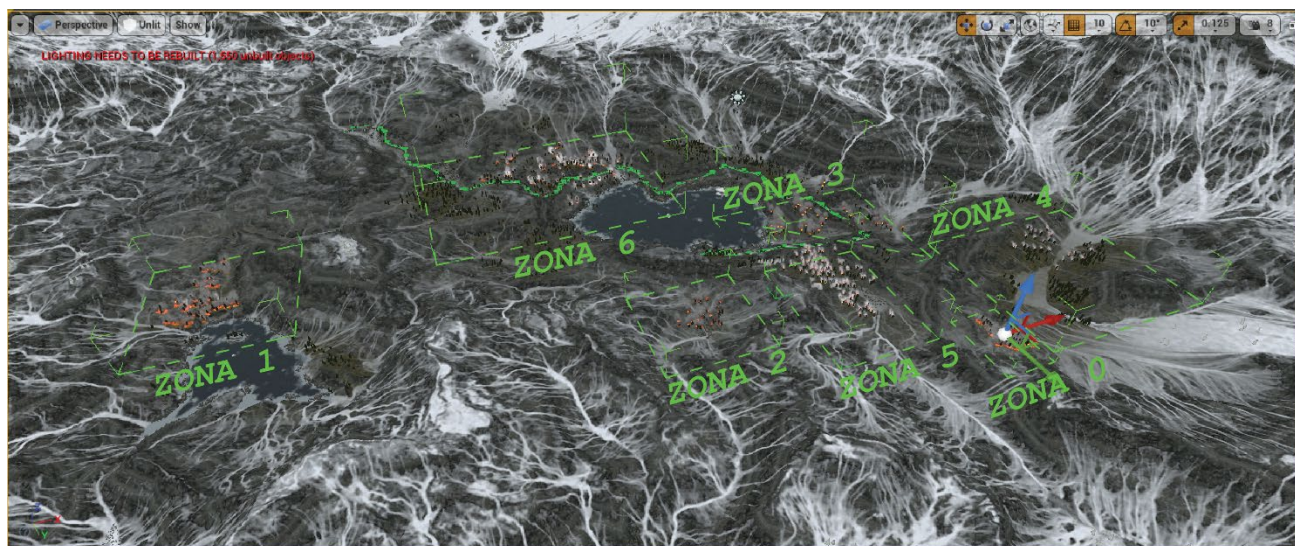
### 4.3.3 – Quantitat de mostres necessàries per a l'entrenament i testeig del model

Quan es tracta d'entrenar models DL és important disposar del màxim de mostres possibles per a cada classe per a incrementar al màxim el rendiment del model. Per norma general, per a cada classe definida durant la determinació dels objectius de classificació, com a mínim haurem d'establir unes 100 mostres, tot i que per a assolir rendiments alts de l'arquitectura s'aconsella incrementar aquest nombre. En el cas d'aquesta PdC, com que l'entrenament de l'arquitectura es realitzarà utilitzant un conjunt d'imatges amb característiques molt simplificades (veure [secció 4.3.1](#)), aleshores amb un conjunt d'entre 100 i 200 mostres per classes en tindrem prou per a realitzar l'entrenament. A la taula de la figura [4.x.x.x](#) de la [secció 4.7](#) es pot consultar la quantitat exacta d'imatges obtinguda per a cada classe.

<sup>2</sup> <https://howthingsfly.si.edu/flight-dynamics/roll-pitch-and-yaw>

## 4.4 – Assignació de les zones definides a l'entorn per a l'obtenció d'imatges

Donades les diferents zones establertes a l'entorn Unreal que s'ha creat específicament per a la realització d'aquesta PdC (veure **secció 3.3**), i que recordem a continuació:



**Figura 3.x.x.x** – Simulació d'incendis forestals corresponents a les 3 classes definides i distribuïts en zones clarament delimitades

Zona	Intensitat (Classe)	Coordenades				Intensitat	Extensió
		NW	NE	SW	SE		
#0	Alta (1)	(65250.0, 63170.0, 18410.0)	(71790.0, 65090.0, 17890.0)	(70720.0, 54600.0, 18380.0)	(65800.0, 55880.0, 18350.0)	Alta	Localitzada
#1	Alta (1)	(-52140.0, -10270.0, 17240.0)	(-44800.0, 11330.0, 17200.0)	(-61750.0, 9020.0, 17440.0)	(-64860.0, -9340.0, 19200.0)	Alta	Estesa
#2	Moderada (2)	(20597, 30507, 18861)	(31506, 33793, 19799)	(16080, 41923, 20045)	(27889, 45620, 19000)	Moderada	Localitzada
#3	Moderada (2)	(53784, 5943, 10176)	(57370, 8070, 11310)	(50020, 13850, 10270)	(54040, 15610, 10790)	Moderada	Estesa
#4	Baixa (3)	(96894, 33663, 18410)	(110697, 48665, 17836)	(86339, 37386, 17971)	(82762, 58568, 16764)	Baixa	Localitzada
#5	Baixa (3)	(45220, 29500, 16070)	(59410, 29670, 18200)	(45460, 50040, 19960)	(58970, 50770, 17870)	Baixa	Estesa
#6	Mixta (1, 2 i 3)	(-11495, -37985, 14289)	(34616, -47018, 19652)	(-8420, -14050, 17180)	(34253, -17952, 11687)	MIX	Estesa
#7	Nul·la (4)	N/A	N/A	N/A	N/A	Nul·la	N/A

D'una banda, es proposa l'assignació de les zones anteriors als tres subgrups en els que es dividirà el *dataset*<sup>3</sup> (entrenament, validació i testeig) per a generar les imatges que serviran per a entrenar, validar i testear el model:

- Zones seleccionades per a l'**entrenament** i **validació** del model → #1, #3, #5, i #7
- Zones seleccionades per al **testeig** del model → #0, #2 i #4, #7

Així doncs, s'utilitzaran les zones d'extensió més elevada per a l'entrenament i validació<sup>4</sup> del model, ja que d'aquesta manera disposarem d'una quantitat major d'imatges (sense tenir en compte l'aplicació **d'estratègies d'ampliació** que es veuran a la **secció 5.2.3.1**), fet que ens permetrà assolir millors nivells de rendiment del model un cop entrenat.

De l'altra banda, es realitza la següent assignació per tal d' "impermeabilitzar" el conjunt d'imatges que utilitzarem per a entrenar i testear el model, del conjunt imatges que obtindran el drons en temps real durant la explotació del model. A més, d'aquesta manera podem proporcionar un entorn on s'hi puguin trobar diferents instàncies de classe, de manera que sigui possible provar el classificador de l'arquitectura de xarxes neuronals implementada (veure **secció 5**):

- Zones reservades per als **experiments de PdC** → #6

## Observacions

- Aquesta segmentació de les mostres del *dataset* ens permetrà evitar (o si mes no, reduir les possibilitats) que es produeixi el fenomen del sobre-ajust o *overfitting*<sup>5</sup>, que en poques paraules implicaria una memorització durant l'entrenament de les característiques dels incendis corresponents a imatges que s'utilitzaran, o bé per al *testeig* del model després de cada *epoch* d'entrenament, o bé per a realitzar els experiments sobre el model. D'aquesta manera, aconseguirem que les mesures de rendiment del model proporcionin dades acurades sobre el seu grau de generalització un cop realitzat l'entrenament, com es veurà més endavant quan es realitzin el anàlisi de rendiment (benchmarking) del model dissenyat (veure **secció 6**).

---

<sup>3</sup> A la **secció 5.2.3.2** es concreten les motivacions i es donen més detalls al respecte de la divisió del *dataset* en subconjunts.

<sup>4</sup> Els subgrups d'entrenament i validació seran apropiadament separats en temps real mitjançant *pyTorch* / *Keras* durant el preprocesament de les imatges en el moment d'alimentar el model (veure **secció 5.2.3.1**). Tanmateix, el subgrup de testeig es separa amb antelació pel motius indicats a les observacions finals de la secció.

<sup>5</sup> <https://elitedatascience.com/overfitting-in-machine-learning>

- Cal tenir present que la decisió de realitzar la divisió del *dataset* en funció de les zones definides a la **secció 3.3** està majoritàriament motivada per les limitacions del motor gràfic Unreal Engine utilitzat per a desenvolupar aquesta PdC, concretament degut al fet de què generar un entorn “ric i divers” pel que fa a elements gràfics és temporalment molt costós, i per tant, hi ha un alt grau de similitud entre imatges d’una mateixa zona. A més, el sistema de presa automàtica d’imatges utilitzat (veure secció 4.6.2) pot arribar a prendre imatges molt similars en una mateixa zona si dos punts emissors de calor són molt a prop l’un de l’altre, ja que la càmera es posiciona automàticament a sobre d’aquests (en angles de 90° i 45°) per a prendre les captures.



## **4.5 - configuració dels paràmetres de captura de les imatges**

### **1 - Configuració del mode Computer Vision d'AirSim**

El mode *Computer Vision* (CV) d'*AirSim* ens proporciona una major flexibilitat a l'hora de fer ús de les seves característiques de captura d'imatges, tant de manera programàtica com de manera manual, i sense dependre del moviment de la càmera annexa al multirotor (dron) ni mostrar-lo en les imatges. Així doncs, utilitzarem el mode CV per a prendre les imatges necessàries per a generar els jocs de proves i entrenar la NN. Per a activar aquest mode només cal indicar el valor `computerVision` al paràmetre `simMode` de l'arxiu de configuració d'*AirSim*, `setting.json`, tal i com s'indica a la **figura 4.x.x.x**.

### **2 - Configuració de la mida de les imatges**

De partida es determina una mida de 640x512 per a totes les imatges del dataset pels motius indicats al final de la [secció 4.6.1](#). Tanmateix, cal recordar que la mida de les imatges sempre pot ser adaptada als requeriments d'una arquitectura de xarxes neuronals particular mitjançant la seva reducció en temps real amb els frameworks *pyTorch* o *TensorFlow+Keras* de python, just en el moment d'alimentar el model per al seu entrenament. Per a establir la mida al fitxer de configuració d'*AirSim* caldrà especificar-ho per a tots i cadascun dels tipus d'imatges que pot capturar *AirSim* (en el cas que ens ocupa; `scene/RGB = 0` i `segment = 7`) mitjançant els paràmetres `width` i `height`, tal i com s'observa a la **figura 4.x.x.x**.

### **3 – Arxiu de configuració d'AirSim: settings.json**

Per a configurar *AirSim* amb el mode CV, primer caldrà ajustar el seu fitxer de configuració de la següent manera:

```
{
  "SettingsVersion": 1.2,
  "CameraDefaults": {
    "CaptureSettings": [
      {
        "ImageType": 0,
        "Width": 640,
        "Height": 512,
        "FOV_Degrees": 90,
        "AutoExposureSpeed": 100,
        "MotionBlurAmount": 0
      },
      {
        "ImageType": 7,
        "Width": 640,
        "Height": 512,
        "FOV_Degrees": 90,
        "AutoExposureSpeed": 100,
        "MotionBlurAmount": 0
      }
    ]
  },
  "SimMode": "ComputerVision"
}
```

**Figura 4.x.x.x** - Fitxer de configuració ~/Documents/AirSim/settings.json

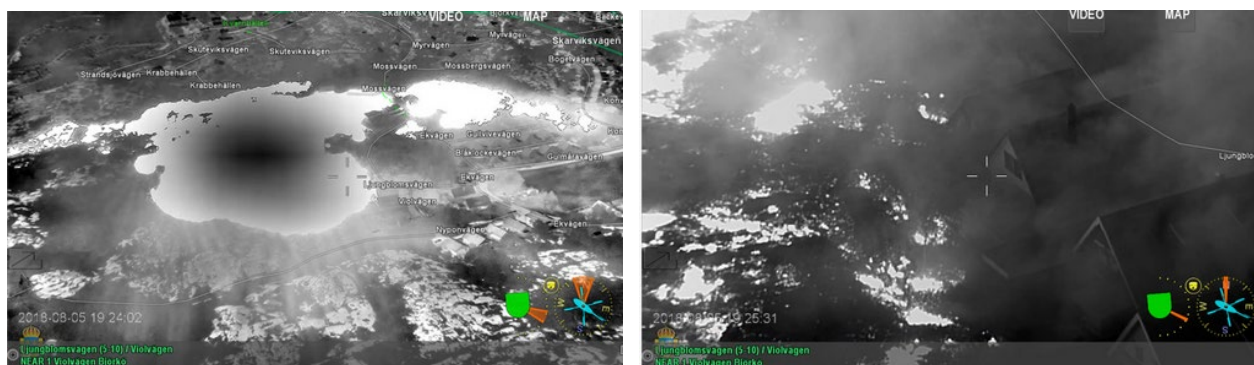


## 4.6 – Modificació de l'script Python utilitzar per a la captura d'imatges

### 4.6.1 – Simulació de visió tèrmica nocturna tipus FLIR durant la obtenció d'imatges del dataset

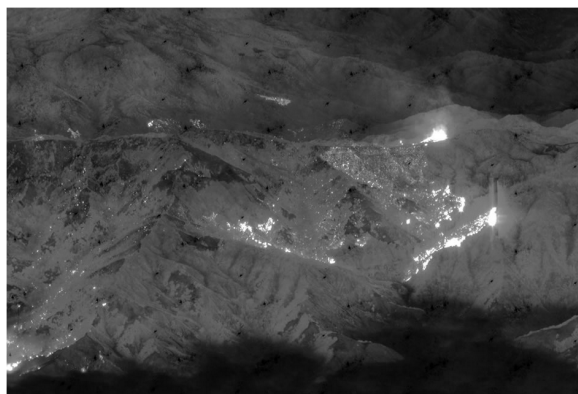
Per tal d'obtenir les imatges que ens permetran entrenar el model de xarxes neuronals de manera automatitzada (això és, sense haver de posicionar manualment la càmera davant d'un incendi cada cop que es vulgui obtenir una imatge), s'utilitzarà la funcionalitat de **seguiment per etiquetes (labels)** que implementa l'script `/src/AirSim/PythonClient/TFG-PoC/capture_ir_segmentation.py` de la distribució 1.6 d'AirSim utilitzada, i que s'anirà modificant a efectes de realitzar aquesta PdC.

Una de les modificacions realitzades que cal destacar es la realitzada a la implementació de visió IR tèrmica que proporcionava el codi original de `capture_ir_segmentation.py` a *AirSim 1.6*<sup>6</sup> per tal de reproduir de la manera més fidel possible la presa i anàlisi d'imatges aèries d'incendis en condicions de nocturnitat, tot tenint en comptes les limitacions del simulador. Concretament, després d'efectuar algunes cerques amb els cercadors d'Internet habituals s'ha pogut comprovar que en situacions reals<sup>7</sup>, la captura d'imatges nocturnes es realitza normalment mitjançant la utilització de càmeres *FLIR* amb tecnologia *MWIR* incorporades als vehicles aeris d'extinció d'incendis. Un exemple d'imatges capturades utilitzant aquesta tecnologia en situacions similars a la de la *PdC* d'aquest projecte són les següents:



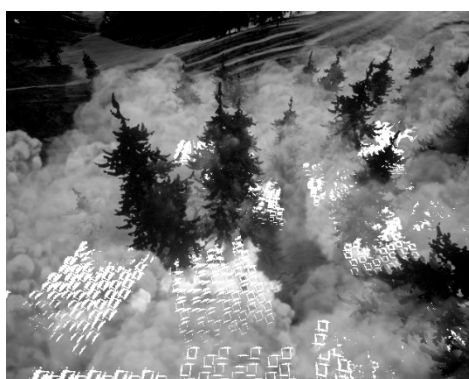
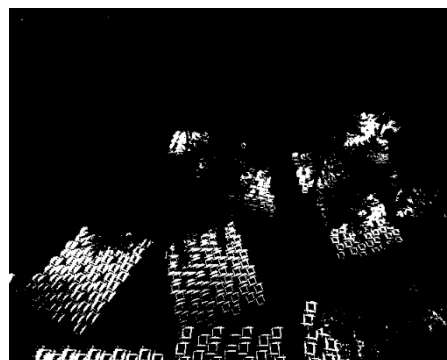
<sup>6</sup> <https://microsoft.github.io/AirSim/InfraredCamera/>

<sup>7</sup> <https://www.flir.co.uk/news-center/military/swedish-national-police-use-flir-in-key-fight-against-recent-swedish-wildfire/>



Imatges obtingudes de [https://www.strategypage.com/military\\_photos/military\\_photos\\_20080805232034.aspx](https://www.strategypage.com/military_photos/military_photos_20080805232034.aspx) i <https://www.flir.co.uk/news-center/military/swedish-national-police-use-flir-in-key-fight-against-recent-swedish-wildfire/>

Concretament, la modificació ha consistit en utilitzar la implementació de partida l'script `capture_ir_segmentation.py` per a prendre dues imatges del mateix escenari; una en RGB "estàndard" (que s'ha convertit a B/N) i una altra segmentada mitjançant la IR tèrmica *built-in* que inclou *AirSim*, on cada píxel en color blanc (valor "255,255,255" amb codificació RGB) forma part d'un objecte o element de l'entorn UE que genera calor en el rang del que és habitual en un incendi. Després només ha calgut buscar en quina posició de la matriu de píxels s'ha detectat calor en la imatge IR tèrmica per a seguidament projectar el valor d'aquests píxels a la imatge en color RGB, de manera que se superposen ambdues imatges aconseguint un efecte semblant als de les imatges de referència anteriors:



Nomes resta afegir que els objectes de UE que emeten calor s'assignen de manera automàtica mitjançant la modificació de l'script `src/AirSim/PythonClient/TFG-`

`PoC/create_ir_segmentation_map.py` associat (que s'ha d'executar primer), així com amb l'assignació de les etiquetes / nom d'objecte adequats a l'entorn LME de UE.

### Consideracions finals:

- Per a simular la visió tèrmica amb IR es podria haver utilitzat directament una imatge en RGB en condicions d'il·luminació convertida a B/N, però en aquests casos no seria possible simular la detecció de calor rere columnes de fum (per posar un exemple concret).
- La mida de les imatges (640x512) simulen les de **l'streaming de vídeo** d'una càmera FLIR comercial específica per a drons<sup>8</sup> i es defineix a l'arxiu de configuració de AirSim `settings.json` (veure **figura 4.x.x.x**, [apartat 4.5](#)).

### 4.6.2 – Automatització de la presa d'imatges i del seu etiquetat (labeling)

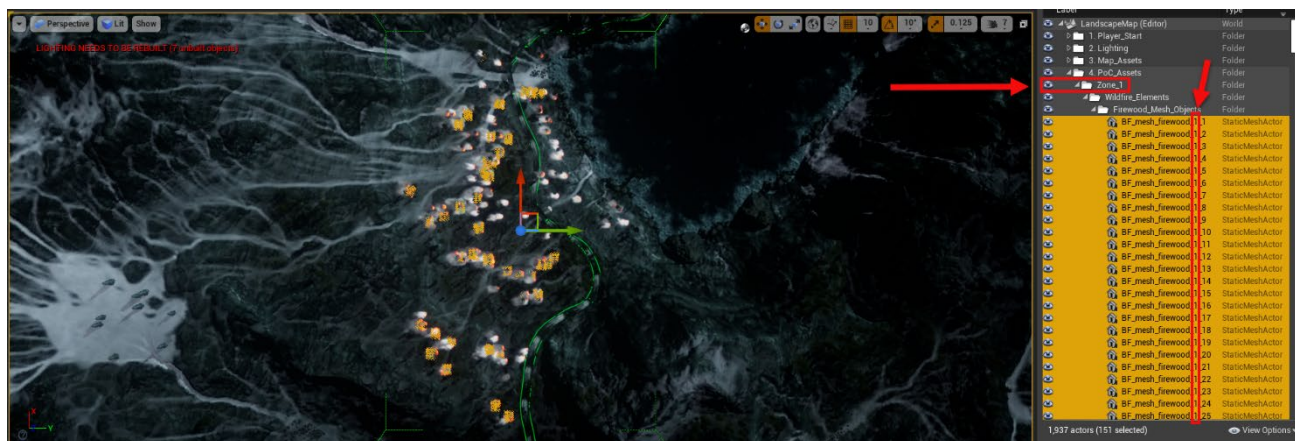
Un cop definides les zones de les quals obtindrem les imatges del dataset, així com establertes les classes d'incendi de cadascuna de les zones, podem procedir a adaptar l'script Python vist al punt anterior (`capture_ir_segmentation.py`) que s'utilitzarà per a la captura d'imatges, de manera que el modificarem per tal d'automatitzar la presa d'imatges en funció d'uns paràmetres específics:

- Classe de les imatges (incendis d'alta, moderada o baixa intensitat)
- Càmera del dron (davantera, laterals o posterior)
- Alçada del dron
- Angle de la càmera

Abans però, caldrà fer una modificació addicional a l'entorn Unreal creat per aquesta PdC de manera que serà possible localitzar els elements que simulen calor (objectes etiquetats com `BF_Mesh_Firewood_` i `BF_Grass_Mesh_`) en funció de la seva classe:

---

<sup>8</sup> <https://www.flir.com/products/vue-tz20-r/>



**Figura 4.x.x.x** - Etiquetat de classe als objectes de tipus StaticMesh

Concretament, cal notar el sufix que s'ha afegit a tots i cadascun dels nom d'objectes que simulen emissió de calor de cada zona. Prenent la captura anterior com a exemple on a la dreta s'observen aquests objectes **StaticMesh**, tenim que aquells que pertanyen a la zona 1 tindran el nom formatat amb el sufix **\_1\_xx**. De la mateixa manera, els objectes que pertanyen a la zona 2 tindran el sufix **\_2\_xx**, i així per a totes les zones.

Realitzat l'ajust anterior a l'entorn Unreal, seguidament caldrà redissenyar l'script **capture\_ir\_segmentation.py** de manera que:

1. Ens permeti especificar els paràmetres esmentats al paràgraf anterior mitjançant la entrada manual de dades per teclat (línies 377-419):

```
capture_ir_segmentation.py M X create_ir_segmentation_map.py
PythonClient > TFG-PoC > capture_ir_segmentation.py > ...
377 # Retrieve custom parameters from std input: Drone camera, height, pitch, roll, yall & UE4 environment zone
378 wrong_option=True;
379 while (wrong_option):
380     print("Specify the class of the simulated night/thermal vision wildfire images you want to generate:\n\n",
381           "Zone 0 (Class 1: high intensity wildfire images - small size area) = 0\n",
382           "Zone 1 (Class 1: high intensity wildfire images - big size area) = 1\n",
383           "Zone 2 (Class 2: medium intensity wildfire images - small size area) = 2\n",
384           "Zone 3 (Class 2: medium intensity wildfire images - big size area) = 3\n",
385           "Zone 4 (Class 3: low intensity wildfire images - small size area) = 4\n",
386           "Zone 5 (Class 3: low intensity wildfire images - big size area) = 5\n",
387           "Zone 6 (Class 1+2+3: PoC experiments zone = 6\n",
388           "Zone 7 (Class 4: images with no wildfires) = 7\n")
389     ue4_zone = int(input("Please choose an option (0-7 - Default = 1): ") or '1')
390     if ue4_zone==7:
391         print("\nERROR: Class not implemented yet.\n")
392         time.sleep(2)
393     elif ue4_zone==6:
394         print("\nERROR: Zone reserved to perform the PoC experiments (so we avoid overfitting the model by memorizing features).\n")
395         time.sleep(4)
396     elif ue4_zone<0 or ue4_zone>7:
397         print('\nERROR: Wrong option, try again.')
398         time.sleep(2)
399     else:
400         wrong_option=False;
401
```

**Figura 4.x.x.x.a** - Entrada manual de paràmetres a l'script **capture\_ir\_segmentation.py**



```

402 wrong_option=True;
403 while (wrong_option):
404     print("Choose the multicopter's camera you want to use to retrieve the images:\n\n",
405           "front_center=0\n",
406           "front_right=1\n",
407           "front_left=2\n",
408           "fpv=3\n",
409           "back_center=4\n")
410     camera = int(input("Please choose an option (0-4 - Default = 0):\n\n") or '0')
411     if camera<0 or camera>4:
412         print('\nERROR: Wrong option, try again.')
413         time.sleep(2)
414     else:
415         wrong_option=False;
416
417 height = int(input("Set the multicopter's height (negative integer value - Default = -20 -> lowest hight):\n\n") or '-20')
418
419 pitch = int(input("Set the camera's pitch angle (Integer degrees 180 > angle > 360 - Default = 270):\n\n") or '270')
420

```

**Figura 4.x.x.b** - Entrada manual de paràmetres a l'script `capture_ir_segmentation.py`

2. Ens permeti obtenir un llistat (estructura de dades) de tots els objectes que emeten calor a l'entorn Unreal pel tal de poder situar la càmera automàticament sobre les seves coordenades en el pla horitzontal (XY), tenint en compte la restricció de què s'ha de poder limitar la quantitat d'imatges obtingudes a les zones estrictament especificades per tal d'obtenir imatges d'una classe en particular. Per a realitzar aquesta darrera implementació només caldrà modificar les *regular expression* de les línies 431 i 432 de manera que "injectarem" l'id de zona al filtre *regex* fent que el llistat d'objectes `objectList` resti inicialitzat tan sols amb els nom d'objecte *Mesh* de la zona especificada en el moment d'executar l'script, tal i com es mostra a continuació:

```

capture_ir_segmentation.py M X create_ir_segmentation_map.py
PythonClient > TFG-PoC > capture_ir_segmentation.py > ...
420
421 # Look for objects with names that match a regular expression.
422 # On the case of this PoC, we're looking for objects that include
423 # the "firewood" and the "grass" strings on the UE4 env. objects
424 # that simulate heat emission (see the project's report, section 4.6.3).
425 #
426 # V4.6 -> Enabling the possibility of generating only specific image classes
427 # (see section 4.x.x of the project's report) by "injecting" the zone
428 # variable into the regex expression that filters the objects we want
429 # to take pictures of.
430
431 my_regex1 = r".*?mesh_firewood_" + str(ue4_zone) + r".*?"
432 my_regex2 = r".*?grass_mesh_" + str(ue4_zone) + r".*?"
433
434 objectList = client.simListSceneObjects(my_regex1)
435 objectList += client.simListSceneObjects(my_regex2)

```

**Figura 4.x.x.x** – Parametrització de la selecció / filtrat d'objectes a incloure a la estructura que inclourà tots els punts de captura d'imatges.

La versió final d'aquest script es pot trobar a la carpeta següent del repositori GitHub:

[https://github.com/UOC-Assignments/uoc.tfg.jbericat/blob/master/src/Air-Sim/PythonClient/TFG-PoC/capture\\_ir\\_segmentation.py](https://github.com/UOC-Assignments/uoc.tfg.jbericat/blob/master/src/Air-Sim/PythonClient/TFG-PoC/capture_ir_segmentation.py)

## 4.7 - Recol·lecció i etiquetat d'imatges (execució de l'script de captura d'imatges capture\_ir\_segment.py)

Per a la recol·lecció d'imatges dels datasets s'efectuaran diferents execucions de l'script `capture_ir_segment.py`, tot variant alguns dels paràmetres de captura (distància i angle de la càmera) per tal de maximitzar la quantitat de mostres total, fet que ens permetrà realitzar un entrenament més acurat del model.

A continuació es mostra un exemple d'execució:

- 1) En primer lloc caldrà executar els binaris de l'entorn, o bé carregar aquest a l'editor del Unreal Engine i iniciar l'acció de l'escenari (botó play):



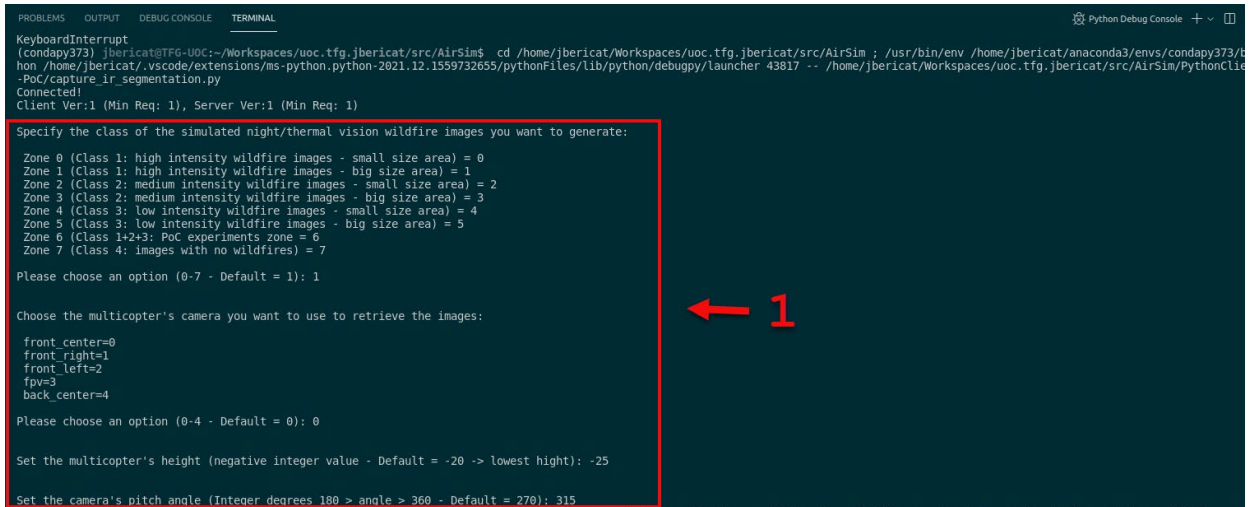
**Figura 4.x.x.x** – Inicialització dels actors de l'entorn des de l'editor Unreal Engine

- 2) En segon lloc cal realitzar la Inicialització del seguiment d'objectes emissors de calor mitjançant la execució de l'script `create_ir_segment.py`

```
(condapy373) jbericat@TFG-UOC:~/Workspaces/uoc.tfg.jbericat/src/AirSim$ /home/jbericat/anaconda3/envs/condapy373/bin/python /home/jbericat/Workspaces/uoc.tfg.jbericat/AirSim/PythonClient/TFG-PoC/create_ir_segmentation_map.py
Connected!
Client Ver:1 (Min Req: 1), Server Ver:1 (Min Req: 1)
camera_response.npy not found. Using default response.
```

**Figura 4.x.x.x** – Execució de `create_ir_segment.py`

### 3) Finalment, podem procedir a la recollida d'imatges a partir de la execució de l'script → `capture_ir_segment.py`



```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
KeyboardInterrupt
(condap373) jbericat@TFG-UOC:~/Workspaces/uoc.tfg.jbericat/src/AirSim$ cd /home/jbericat/Workspaces/uoc.tfg.jbericat/src/AirSim ; /usr/bin/env /home/jbericat/anaconda3/envs/condap373/bin/python3 /home/jbericat/.vscode/extensions/ms-python.python-2021.12.1559732655/pythonFiles/lib/python/debugpy/launcher 43817 -- /home/jbericat/Workspaces/uoc.tfg.jbericat/src/AirSim/PythonClient/PoC/capture_ir_segmentation.py
Connected!
Client Ver:1 (Min Req: 1), Server Ver:1 (Min Req: 1)

Specify the class of the simulated night/thermal vision wildfire images you want to generate:
Zone 0 (Class 1: high intensity wildfire images - small size area) = 0
Zone 1 (Class 1: high intensity wildfire images - big size area) = 1
Zone 2 (Class 2: medium intensity wildfire images - small size area) = 2
Zone 3 (Class 2: medium intensity wildfire images - big size area) = 3
Zone 4 (Class 3: low intensity wildfire images - small size area) = 4
Zone 5 (Class 3: low intensity wildfire images - big size area) = 5
Zone 6 (Class 1+2+3: PoC experiments zone = 6
Zone 7 (Class 4: images with no wildfires) = 7

Please choose an option (0-7 - Default = 1): 1

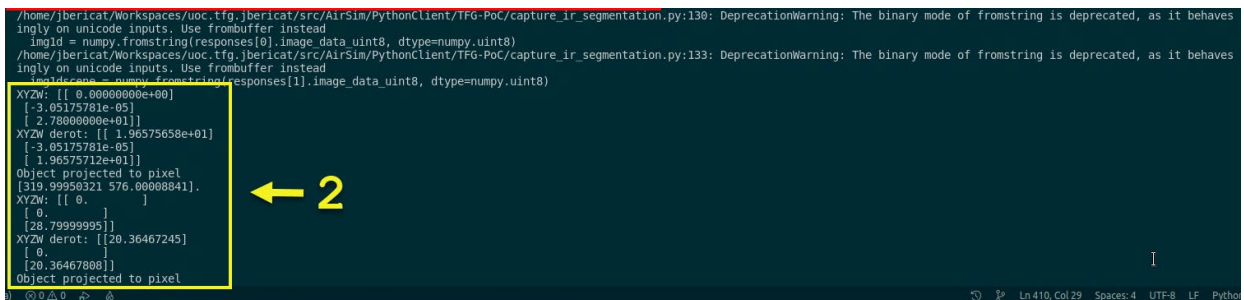
Choose the multicopter's camera you want to use to retrieve the images:
front_center=0
front_right=1
front_left=2
fpv=3
back_center=4

Please choose an option (0-4 - Default = 0): 0

Set the multicopter's height (negative integer value - Default = -20 -> lowest hight): -25

Set the camera's pitch angle (Integer degrees 180 > angle > 360 - Default = 270): 315
  
```

**Figura 4.x.x.x.a** – Execució de `capture_ir_segment.py`



```

/home/jbericat/Workspaces/uoc.tfg.jbericat/src/AirSim/PythonClient/TFG-PoC/capture_ir_segmentation.py:130: DeprecationWarning: The binary mode of fromstring is deprecated, as it behaves
ingly on unicode inputs. Use frombuffer instead
  imgId = numpy.fromstring(responses[0].image_data_uint8, dtype=numpy.uint8)
/home/jbericat/Workspaces/uoc.tfg.jbericat/src/AirSim/PythonClient/TFG-PoC/capture_ir_segmentation.py:133: DeprecationWarning: The binary mode of fromstring is deprecated, as it behaves
ingly on unicode inputs. Use frombuffer instead
  imgIdImage = numpy.fromstring(responses[1].image_data_uint8, dtype=numpy.uint8)

XYZW: [[ 0.00000000e+00]
 [-3.05175781e-05]
 [ 2.78000000e+01]]
XYZW derot: [[ 1.96575658e+01]
 [-3.05175781e-05]
 [ 1.96575712e+01]]
Object projected to pixel
[319.99950321 576.00008841].
XYZW: [[ 0. ]
 [28.79999995]]
XYZW derot: [[20.36467245]
 [ 0. ]
 [20.36467800]]
Object projected to pixel
  
```

**Figura 4.x.x.x.b** – Execució de `capture_ir_segment.py`

A les figures **4.x.x.x.a** i **4.x.x.x.b** s'observa com es pot realitzar la introducció manual de paràmetres d'una banda (1) i el la sortida per terminal del log de captura d'imatges realitzades (2).

La taula de la figura 4.x.x.x resumeix la quantitat d'imatges recollides per a cadascuna de les classes definides a la [secció 4.2](#):

Classe	Angle (Pitch)	Alçada (Height)	Subgrup d'imatges	total	Carpeta repositori GitHub
#1 (ALTA INTENSITAT)	270°	-20	Entrenament + validació	46	/usr/datasets/%VERSION_ID%/training+validation/high-intensity-wildfires/FLIR/
#1 (ALTA INTENSITAT)	270°	-20	Testeig	19	/usr/datasets/%VERSION_ID%/test/high-intensity-wildfires/FLIR/
#1 (ALTA INTENSITAT)	315°	-20	Entrenament + validació	45	/usr/datasets/%VERSION_ID%/training+validation/high-intensity-wildfires/FLIR/
#1 (ALTA INTENSITAT)	315°	-20	Testeig	19	/usr/datasets/%VERSION_ID%/test/high-intensity-wildfires/FLIR/
#2 (INTENSITAT MODERADA)	270°	50	Entrenament + validació	73	/usr/datasets/%VERSION_ID%/training+validation/medium-intensity-wildfires/FLIR/
#2 (INTENSITAT MODERADA)	270°	-20	Testeig	27	/usr/datasets/%VERSION_ID%/test/medium-intensity-wildfires/FLIR/
#2 (INTENSITAT MODERADA)	315°	50	Entrenament + validació	81	/usr/datasets/%VERSION_ID%/training+validation/medium-intensity-wildfires/FLIR/
#2 (INTENSITAT MODERADA)	315°	-20	Testeig	28	/usr/datasets/%VERSION_ID%/test/medium-intensity-wildfires/FLIR/
#3 (BAIXA INTENSITAT)	270°	-30	Entrenament + validació	91	/usr/datasets/%VERSION_ID%/training+validation/low-intensity-wildfires/FLIR/
#3 (BAIXA INTENSITAT)	270°	-20	Testeig	26	/usr/datasets/%VERSION_ID%/test/low-intensity-wildfires/FLIR/
#3 (BAIXA INTENSITAT)	315°	-30	Entrenament + validació	91	/usr/datasets/%VERSION_ID%/training+validation/low-intensity-wildfires/FLIR/
#3 (BAIXA INTENSITAT)	315°	-20	Testeig	24	/usr/datasets/%VERSION_ID%/test/low-intensity-wildfires/FLIR/

**Figura 4.x.x.x** – Imatges de cada classe recollides al dataset que s'utilitzarà per a la PdC



## **4.8 - Empaquetament i publicació del dataset d'imatges per a la realització de la PdC**

Amb el propòsit de formalitzar la publicació del dataset d'imatges i facilitar la seva consulta, es decideix utilitzar la plataforma { KADDLE / JUPYTER NOTEBOOKS }.

**DECIDIR I IMPLEMENTAR QUAN ES FACI EL WRAPPING-UP FINAL DEL TFG**