

Visualització, creació i millora de terrenys 3D

Gerard Martínez Espelleta

23 de gener de 2022

Resum– Durant els últims anys, la visió de mapes en 3d ha avançat moltissim des de l'aparició de Google Maps l'any 2005, el qual va popularitzar i fer disponible al públic general aquest servei. No obstant les noves tecnologies com la visió de relleu només s'apliquen a zones molt poblades deixant de banda una gran part de la població.

En aquest treball crearem un visor 3D que permeti veure tota l'àrea de catalunya amb el seu relleu sense deixar de banda cap població, i farem ús de diverses xarxes de *deep learning* que ens permetrà millorar les imatges mostrades.

Paraules clau– Visor 3D, Deep Learning, GAN, Generative Adversative Network, Mapa, Superresolució, Web

Abstract– During last years, the view of 3D maps has advanced greatly since Google Maps appeared in 2005, which popularized and made available to the general public this service. However, this new technologies such as relief vision are only applied to heavily populated areas, leaving a large part of the population aside.

In this work we will create a 3D viewer that allows you to see the whole area of Catalonia with its relief. without neglecting any population, and we will use various *deep learning* networks that will allow us to improve the images shown.

Keywords– 3D Visor, Deep Learning, GAN, Generative Adversative Network, Map, Superresolution, Web

1 INTRODUCCIÓ - CONTEXT DEL TREBALL

Un dels avantatges del que disposem avui en dia en comparació amb fa uns anys és la possibilitat d'utilitzar mapes online per situar-nos, obtenir indicacions, o simplement investigar el terreny.

De fet, desde la aparició del primer mapa online, aquest servei ha anat millorant any rere any, permetent aplicar diferents capes a la vista com imatges satèl·lit, de relleu, del trànsit o fins i tot en els últims anys permetent la visualització del relleu sobre imatges satèl·lit en 3D.

Tot i així, visors com el de *Google Earth* o *Apple Maps* només ofereixen una bona representació del terreny en zones que les companyies consideren més interessants. És a dir, en zones d'alta afluència de persones, com podrien ser grans ciutats o zones universitàries. Si sortim d'aquests límits, les imatges es comencen a veure borroses i sense relleu visible.

En aquest projecte crearem un visor 3D accessible de ma-

nera online que serà capaç de mostrar tot el mapa de Catalunya mitjançant crides a un servidor, d'on anirà descarregant les imatges i els valors del relleu per posteriorment ajuntar-les en un objecte tridimensional i col·locar-lo en un punt de l'espai, creant així un mapa.

1.1 Objectius

L'objectiu principal d'aquest treball és el de crear una aplicació que ens permeti, a partir d'un conjunt d'imatges satèl·lit, afegir-hi un relleu i crear un entorn tridimensional pel quan ens poguem desplaçar. És a dir, crear un mapa d'alta resolució i en relleu que no només estigui disponible en ciutats i llocs molt poblats sinó que ho estigui a tot arreu. Per tal de definir millor el projecte, l'hem dividit en els següents objectius

Aquests serien:

1. Obtenció d'imatges satèl·lit
2. Obtenir els valors del relleu

3. Creació d'una xarxa que ens permeti fer super-resolució
4. Crear entorn 3D

1.2 Metodologia

Per tal de dur a terme aquest projecte d'una manera ordenada i rigurosa es farà ús de la metodologia de *Kanban* [1]. Aquesta metodologia és considerada una metodologia "*agile*" que ens obliga a portar un control precís del que s'ha fet, el que s'està fent i el que queda per fer mitjançant la repartició de la feina en tres columnes diferents que representen aquests estats.

Per tal de poder objectiu esmenat anteriorment, es farà ús d'una eina anomenada *Trello* [2] i es crearàn tres columnes: una pels objectius complerts, una pels objectius en els que estem treballant i una última pels objectius que encara no s'han fet.

1.3 Planificació

Per explicar la planificació del treball i alhora, per ajudar a l'organització d'aquest, s'ha realitzat un diagrama de Gantt [3] organitzat per setmanes. Aquest permet obtenir una vista general dels objectius programats, de manera que resulta molt senzill saber quines feines s'han de fer, quant temps s'ha de dedicar a cada tasca i, per tant, quan ha d'estar acabada cada una d'elles.

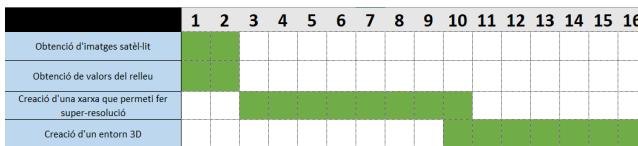


Fig. 1: Diagrama de Gantt

Com es pot observar en la imatge superior, els apartats als que dediquem més temps són aquells referents a la creació de xarxes neuronals, ja que a part del disseny, s'haurà de dedicar temps a entrenar-les i testejar-les. A més també s'haurà de dedicar una quantitat considerable de temps al visor 3D, ja que es destinarà molt temps a revisar la documentació de l'eina seleccionada.

Al dossier, es podrà trobar informació més detallada sobre com s'ha distribuït el temps de cada sub-objectiu.

2 ESTAT DE L'ART

Aquest projecte compta amb dues parts importants. La primera és la generació d'imatges satèl·lit d'alta resolució a partir d'imatges de baixa resolució. Això es durà a terme mitjançant una tècnica anomenada tècnica de super-resolució.

Aquesta és molt important en el món de la visió per computador i s'utilitza principalment en el món mèdic o en el món de la vigilància, ja que en aquests entorns es necessita una molt millor qualitat d'imatge de la que permeten oferir els dispositius usats.

Per tal de poder obtenir les imatges en alta qualitat, s'ha optat per fer ús de tècniques de *Deep Learning*, concretament d'una xarxa neuronal del tipus *Unet*.

Una *UNet* és una xarxa neuronal del tipus convolucional i s'utilitza principalment per la segmentació d'imatges, és a dir, per poder diferenciar diferents elements dins d'una sola imatge. No obstant, també pot ser usada per realitzar super-resolució d'imatges, com es pot observar en els casos següents:

2.1 Runet [4]

Es tracta d'una implantació d'una UNet però afegint-hi sumes després de cada "block" de la xarxa, creant així una estructura de *residual blocks*.

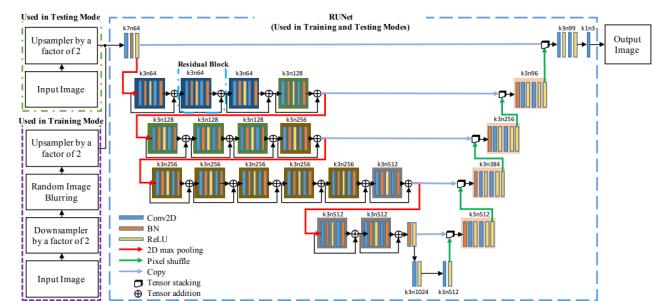


Fig. 2: Estructura d'una RUNet

2.2 Dense Unet [5]

La Dense Unet combina les característiques de la Unet i de la Dense-Net. És a dir que cada block no només rep informació del block anterior sinó de tots els blocks que l'han precedit

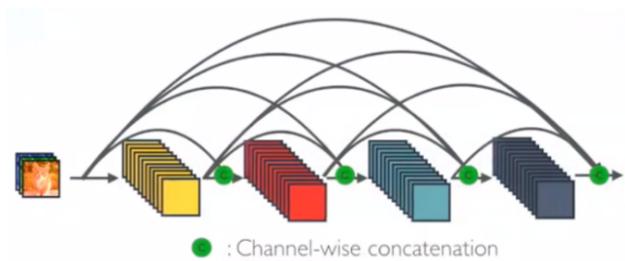


Fig. 3: Estructura d'una connexió densa

Un cop combinada una Unet amb una Dense net s'obté el següent esquema:

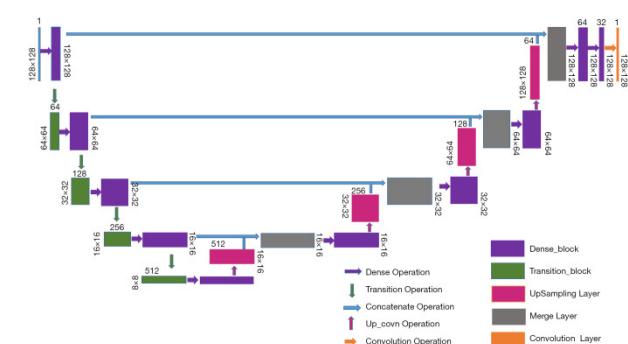


Fig. 4: Estructura d'una Dense-Unet

Per altra banda a l'hora de realitzar el visor 3D, hem pogut observar que també existeixen entorns que realitzen les funcions que nosaltres volem desenvolupar, tot i que tenen les limitacions anteriorment mencionades en la introducció.

2.3 Google Maps

Google Maps és un servei web de visió de mapes oferit per Google que ofereix imatge per satèl·lit, visió aèria, visió de carrer, imatges 360°, indicacions per viatjar juntament amb l'estat del trànsit i horaris de transport públic i fins i tot imatges dels interiors d'alguns edificis.

És usada per més d'un bilió de persones d'arreu del món cada dia i ha estat desenvolupada en javascript i crides AJAX (la part del client)



Fig. 5: Visor de mapes Google Maps

2.4 Google Earth

Google Earth és un altre servei de visió de mapes que superposa les imatges i el relleu sobre un globus terraqui, donant així una sensació més realista. Aquesta plataforma té imatges del 98% del globus terraqui, a més de contenir dos globus més; un de la Lluna i l'altre de Mart.

També compta amb un visor d'estrelles i un simulador de vol que et permet volar per terreny realista i ubicacions reals. A part, també inclou guies que et permeten visitar llocs turístics o monuments sense sortir de casa.



Fig. 6: Visor de mapes Google Earth

2.5 Apple Maps

Apple Maps és un visor de mapes creat per Apple per a ser usat en els seus dispositius, tot i que també pot ser usat a través del motor de cerca *DuckDuckGo* el qual l'utilitza com a servei de mapes.

Aquest visor, a part de mostrar mapes amb el seu relleu, inclou eines que ofereixen indicacions a l'hora de desplaçar-se.

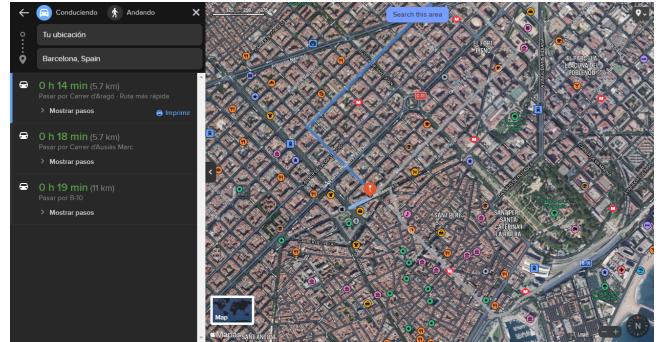


Fig. 7: Visor de mapes Apple Maps

3 DESENVOLUPAMENT DEL TREBALL

3.1 Obtenció de dades

El primer pas per tal de dur a terme el projecte és una bona elecció de la font que utilitzarem per aconseguir les imatges.

És necessari trobar una font que no només ens ofereixi imatges satèl·lit, sinó d'on també es puguin aconseguir dades del relleu i imatges amb una major qualitat per poder entrenar correctament la xarxa de super-resolució.

Un cop feta la recerca, s'han trobat les següents fonts d'on és possible descarregar les dades requerides:

3.1.1 Sentinel2

Sentinel2 [8] és una missió d'observació del grup *Copérnico*. Aquesta compta amb dos satèl·lits que permeten la captura d'imatges multi-espectrals (de 13 bandes), d'infrarojos i de l'espectre electromagnètic.

Afegidament, compta amb una API bastant senzilla d'utilitzar i que ofereix resultats amb un molt baix temps de resposta.

No obstant, s'ha decidit no fer ús dels serveis que ens ofereix Sentinel ja que la API que s'utilitza per demanar les imatges és de pagament i es preferix buscar alguna plataforma gratuïta.

3.1.2 NASA

La NASA [9] és l'agència d'exploració espacial dels Estats Units, i permet fer ús de la seva API de *web map service* d'una manera gratuïta amb un simple registre.

Tot i que amb aquesta API es poden obtenir imatges de tot el globus i de forma gratuïta, es pot notar com les imatges rebudes són d'una qualitat considerablement menor que les que es podien obtenir amb el satèl·lit Sentinel. A més, és bastant casual que aquestes continguin núvols que evitin que es pugui veure la superfície correctament.

Finalment, es va descartar aquesta font perquè a part de tenir imatges de no massa bona qualitat, el temps de resposta era superior a 10 segons, la qual cosa perjudicaria l'experiència de l'usuari que utilitzés el visor 3d.

3.1.3 Google Earth

Google Earth [10] es un visor online de mapes en 3d.

La seva API permet obtenir dades del relleu d'un punt només indicant-hi les coordenades. Aquesta API és gratuïta i fàcilment accessible mitjançant un compte de Google.

No obstant, aquesta plataforma va ser descartada quan es va decidir que es faria ús de mapes d'altura per calcular el relleu.

3.1.4 IGN

L'Institut Nacional de Geografia [11], és una entitat pública encarregada d'investigar i organitzar les dades geogràfiques del país.

Aquest compta amb una API gratuïta que permet obtenir imatges satèl·lit d'una manera ràpida i senzilla. Tot i així, no s'ha fet ús d'aquesta API ja que no oferia dades sobre el relleu del territori.

3.1.5 ICGC

L'Institut Cartogràfic i Geològic de Catalunya [12] és també una entitat pública encarregada d'investigar i emmagatzemar dades geogràfiques, però en aquest cas, dades exclusivament del territori català.

Aquesta API també ofereix la opció de descarregar mapes d'altures on es veu fàcilment el relleu del terreny o imatges de major qualitat capturades des d'un avió. En el cas de les imatges satèl·lit, aquestes són imatges obtingudes de Sentinel2 i arreglades per la pròpia organització per eliminar núvols i millorar-ne la qualitat.

És per aquesta sèrie de motius que s'ha decidit fer ús d'aquesta font d'imatges pel projecte.

Un cop escollida la font (i degut a que la nostra aplicació necessita moltes imatges per funcionar de manera correcta), s'ha creat un script que permet descarregar-les de manera automàtica només passant-li les coordenades desitjades.

Aquest està escrit amb python i compta amb dues funcions principals:

- 1. Conversor de sistema de coordenades:** La API de ICGC fa ús d'un sistema de coordenades anomenat '*EPSG:25831*', mentre que normalment en aplicacions com Google Maps es fa ús d'un altre sistema anomenat '*EPSG:4326*'.

Per tant, per poder introduir les coordenades desitjades d'una manera més còmode, aquesta funció converteix les dades d'un sistema a l'altre.

- 2. Obtenció de les imatges:** Un cop obtingudes les coordenades en el sistema desitjat, només s'ha de realitzar la sol·licitud d'imatge al servidor i emmagatzemar la resposta d'aquest en cas que no retorni un error.

A part també es va crear un script de revisió de dataset, el qual comprobava que les imatges descarregades fossin adequadues (mitjançant tècniques com el percentatge de píxels negres o el tamany de la imatge) i en cas de que no ho fossin les eliminava.

3.2 Creació de la xarxa de super-resolució del terreny

Com ja s'ha comentat anteriorment, la xarxa que s'utilitzarà per tal d'aconseguir la super-resolució és una Unet.

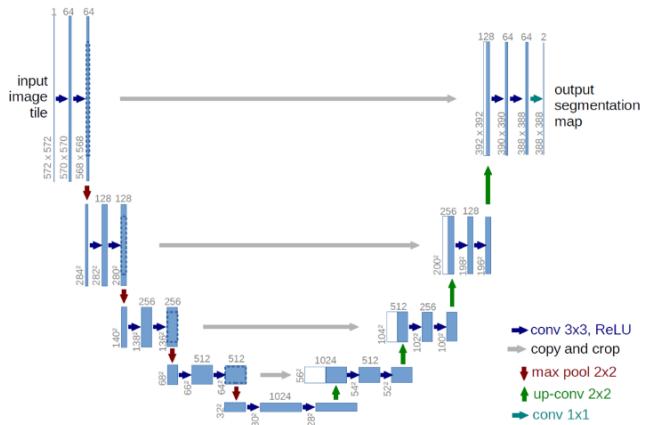


Fig. 8: Estructura d'una Unet [13]

Tal com es pot observar a la figura superior, aquesta xarxa és totalment convolucional, és a dir, les úniques operacions que es realitzen (a part d'incrementar o reduir les seves característiques) són convolucions.

3.2.1 Creació d'un dataset

Una xarxa neuronal necessita moltes imatges per poder ser entrenada i que doni uns resultats acceptables. És per això que es va decidir crear un Dataset propi amb imatges d'alta resolució i de baixa.

Per aconseguir això es van utilitzar dos scripts; Un primer que es baixava imatges d'alta qualitat de Catalunya i un altre que "netejava" aquest dataset, eliminant així imatges en negre (fora de la zona del territori) o arxius d'errors.

Un cop executats els dos arxius, es va aconseguir un dataset amb mes de 70.000 imatges en alta resolució. El següent pas doncs, va ser disminuir la resolució d'aquestes per simular l'input esperat.

Això es va realitzar d'una manera senzilla aprofitant l'algoritme de redimensionament d'imatges que utilitzà la llibreria **opencv**. Aquesta feia ús de nearest neighbour com a mètode de selecció de valor de píxel a l'hora de fer una redimensió, per tant, només es va haver de disminuir el tamany de les imatges i després augmentar-lo per poder aconseguir una imatge pixelada i amb una baixa qualitat.

3.2.2 Creació de la xarxa

Per tal de crear la xarxa es va fer ús de **pytorch** [14], una llibreria de python dedicada a la creació, entrenament i ús de xarxes neuronals.

Com s'ha mencionat anteriorment, s'ha triat crear una Unet degut principalment a la seva senzillesa; com que és una xarxa totalment convolucional, el tamany de la imatge que sigui introduïda, serà exactament igual que el tamany de la imatge que retorna la xaxa. A més, s'ha pensat que

permetria suavitzar els contorns dels píxels en les imatges pixelades de manera que quedarien mes suaus, donant un efecte de millor resolució.

La xarxa consta de tres parts, una part de codificació, una part intermitja i una part de descodificació. Cada una d'aquestes parts està formada per una estructura a la que anomenarem *Block*.

Cada *Block* està format per dues convolucions i dues funcions d'activació de tipus *ReLU* agrupades de la següent manera:

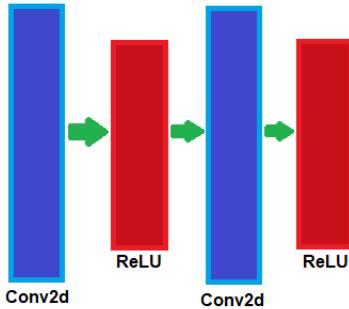


Fig. 9: Estructura d'un Block de la UNet

A la part de codificació l'objectiu de la xarxa és extreure el major número de característiques que li sigui possible de la imatge.

Per aconseguir aquest objectiu anirà unint una sèrie de *blocks* amb operacions de *max pooling* de 2x2. Això farà que alhora que el número de característiques augmenta, el tamany de la imatge va reduint-se. En el cas d'aquest treball, la part de codificació compta amb 4 *blocks* de profunditat.

La part intermitja només comptarà amb un *block*, el qual augmentarà un cop més les característiques i passarà les dades a la part de descodificació.

La part de descodificació, amb el mateix tamany que la de codificació és l'encarregada de generar la imatge que retorna la xarxa. En aquest cas, els *blocks* estaran units amb convolucions verticals, o el que és el mateix operacions de convolució transposades. Aquesta operació permetrà reduir el número de característiques, tornant la imatge a la seva mida original. Afegidament, cada *block* que formi aquesta descodificació, rebrà també les característiques que hem extret a la fase de codificació per poder formar una imatge final satisfactòria.

3.3 Creació d'una GAN

Un cop valorats els resultats obtinguts amb la UNet visibles a l'apartat *Resultats de la superresolució amb UNet* es va observar que per intentar aconseguir uns millors resultats es podia fer ús d'una *Generative Adversative Network* o *GAN*.

Una GAN [16] és un model de xarxa neuronal capaç de generar un conjunt de dades de sortida amb les mateixes qüalitats que el conjunt de dades d'entrada. Aquesta consisteix en dues xarxes neuronals que competeixen l'una

contra l'altra de manera que només pot guanyar una de les dos (el *gain* d'una és el *loss* de l'altra).

La mecànica principal d'aquest tipus de xarxa és la de l'entrenament "indirecte" mitjançant un discriminador (una altra xarxa capaç d'identificar si un *input* s'assembla al resultat esperat), de manera que la xarxa no s'entrena per treure una imatge en concret, sinó que s'entrena per a enganyar al discriminador.

Tal i com s'ha comentat una GAN consta de dues parts:

3.3.1 Discriminador

És l'encarregat de decidir si la imatge que se li passa com a input és una imatge en HD o no.

Està format per:

- Una convolució d'entrada
- Una ReLU
- Una sèrie de blocks
- Dues funcions Lineals, per a convertir les dades a un sol valor
- Una funció sigmoid

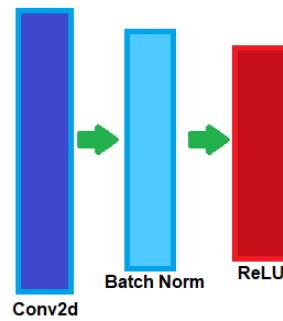


Fig. 10: Estructura d'un Block del Discriminator

3.3.2 Generador

És la xarxa encarregada de generar les imatges. Aquesta està formada per:

Una convolució inicial seguida d'una funció ReLU

Una estructura mitjana: La qual consisteix en un número de *Blocks* seguit d'una convolució, un batch normalització i una suma de la *x* inicial amb la *x* resultant al final de l'estructura.

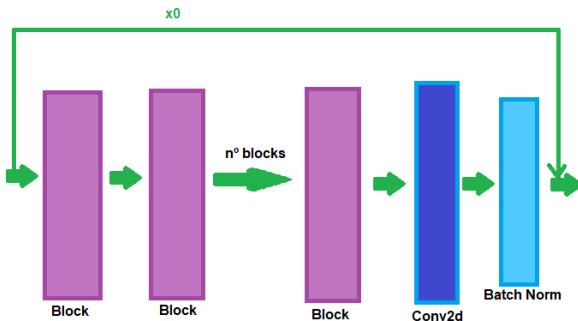


Fig. 11: Estructura mitjana

A continuació es mostra l'estructura d'un dels blocks mencionats anteriorment. Podem notar com també es fa la suma de característiques al final de cada un d'ells.

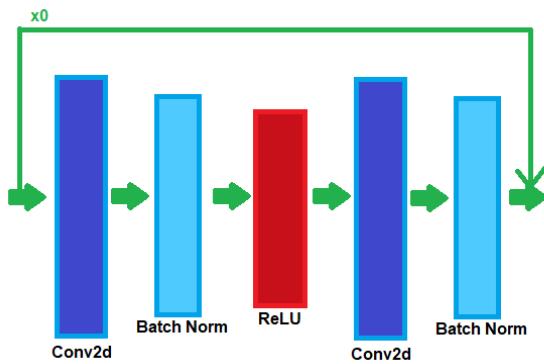


Fig. 12: Estructura d'un Block del Generator

Una estructura final d'upsampling: Aquesta estructura consta d'una convolució, un pixel shuffle (el qual reordena els píxels permetent així una convolució per subpíxels) i una ReLU final.

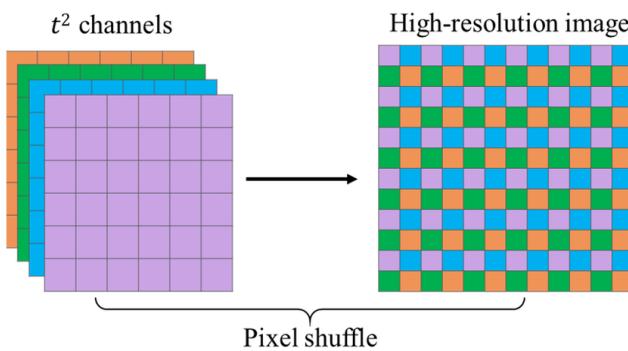


Fig. 13: Demostració de Pixel Shuffle

Una última convolució

3.4 Creació de l'entorn 3D

Per tal de poder fer visibles els resultats obtinguts en els anteriors apartats, s'ha de comptar amb un entorn tridimensional on no només puguem veure les imatges del terreny, sinó també la representació del seu relleu.

Per tal d'aconseguir això, s'ha discutit entre diferents motors gràfics, concretament entre *Unity* [17], *Unreal Engine* [18], *Godot* [19] i *Three js* [20].

TAULA 1: COMPARACIÓ DELS DIFERENTS ENTORNS

Motor	Unity	Unreal Engine	Godot	Three js
Característica				
Codi obert	No	Si	Si	Si
Llenguatge	C#	C++	C#	Javascript
Visor	Aplicació	Aplicació	Aplicació	Navegador

Tot i la popularitat dels tres primers, i que ofereixen un bon conjunt d'eines així com càlculs de físiques i renderitzats, s'ha optat per la última opció ja que, a part de ser una llibreria de codi obert, al ser una aplicació en línia ofereix una major integració amb la resta de components que ja han sigut creats en altres llenguatges, a part de permetre un accés a l'eina molt més senzill i lliure d'instal·lacions per part de l'usuari final.

3.4.1 Three.js

Three.js és una llibreria de javascript que permet crear i mostrar gràfics i animacions en 3D en un navegador Web.

Per tal de que aquest motor funcioni, s'ha de crear un arxiu html que tingui importada la llibreria de Three.js i tots els arxius que continguin el codi. En el cas de la nostra aplicació, aquesta s'estructura en quatre arxius javascript:

3.4.1.1 scene.js

Es tracta de l'arxiu principal del projecte, en aquest es carregarà l'escena, així com la càmera i el bucle de l'animació, el qual és el bucle principal del motor. Des d'aquí es realitzaran les crides a tots els altres arxius.

3.4.1.2 checkload.js

Aquest arxiu conté les funcions que seran usades per *scene.js* per comprovar si el terreny que està a punt de trepitjar l'usuari està carregat o no. En cas de que no estigui carregat, es farà una crida al servidor per obtenir les imatges del terreny i dibuixar-les.

Aquest últim pas el farem mitjançant una eina de javascript anomenada *promises*. Aquesta ens permet executar una funció sense esperar al seu resultat i, un cop s'obtingui el resultat, utilitzar-lo. D'aquesta manera la càrrega de terreny no aturarà el bucle principal de la pàgina sinó que es realitzarà en segon pla.

3.4.1.3 terrain.js

Aquest arxiu és l'encarregat de dibuixar el terreny en pantalla. Compta amb dues funcions principals:

- **LoadTerrain:** A aquesta funció se li passa una imatge (una de les imatges inicials) i el seu relleu i ho dibuixa per pantalla a la posició que se li indiqui.
- **LoadTerrainBinary:** Aquesta funció rep el valor de la imatge retornada pel servidor quan li fem una crida

des de l'arxiu **checkload.js**. Un cop rebuda, la descodifica (ja que la imatge vén en hexadecimal) i torna a fer una crida al servidor per a obtenir el seu relleu. Un cop rep el relleu, el descodifica i dibuixa el terreny a la posició que se li indica.

Tal i com hem comentat ambdues funcions anteriors operen amb una imatge del relleu i amb una del terreny.



Fig. 14: Imatge del terreny i imatge del relleu

Per tal d'extreure el relleu de la imatge fem ús d'una funció la qual analitza el valor de cada píxel de la nostra imatge de relleu i l'enmagatzema en una llista. Després se li assigna un valor més alt o més vaig a la ubicació dels vèrtexs del nostre mapa segons el valor enmagatzemat a la llista.



Fig. 15: Exemple de terreny amb el relleu generat

3.4.1.4 **memorysaver.js**

Aquest arxiu és l'encarregat de gestionar que el nostre visor 3D no ocupa més memòria de la necessària, ja que si fós així el visor es tornaria lent i complicat de fer servir. Això ho aconsegueix mitjançant una funció que és cridada cada cop que es genera terreny nou.

Aquesta comprova si les caselles que s'han generat anteriorment es troben encara dintre del rang de visió de l'usuari, i en cas de que no sigui així les elimina i actualitza les variables de posició.

3.4.2 Estructura física de l'entorn

Per poder rebre les dades tractades, l'aplicació web necessita connectar amb un servidor per tal que aquest realitzi les operacions necessàries. Per aquest motiu, s'ha creat la següent estructura:

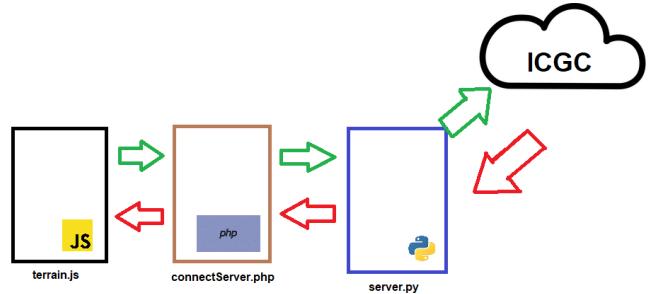


Fig. 17: Estructura física de l'entorn

Tal com podem veure a la figura superior, quan l'arxiu de càrrega de terreny noti que necessita dades noves, contactarà amb l'arxiu *connectServer.php* mitjançant una crida assíncrona d'*AJAX*. Quan aquest rebi la crida amb les dades necessàries per obtenir les imatges, es connectarà al servidor *server.py* mitjançat l'ús de sockets, i aquest, un cop rebi la connexió, demanarà les imatges mitjançant un *HTTPRequest* al servidor del **ICGC**. Un cop hagi obtingut les imatges, les tractarà si així s'escau i les retornarà a l'arxiu *php* mitjançant sockets, el qual respondrà la crida *AJAX* retornant les dades de la imatge desitjada en hexadecimal al nostre arxiu de càrrega de terreny, el qual haurà de descodificar la imatge per poder-la afegir com a textura al nostre relleu.

3.4.3 Processat de les dades

Quan al servidor li arriba una ordre de càrrega d'imatge, a aquest se li indica si cal processar-la o no. En cas positiu, un cop ha obtingut la imatge del servidor del **ICGC**, la descodifica i la divideix en 36 sub-imatges. A continuació carrega la xarxa neuronal prèviament entrenada i aplica el tractament de les imatges passant-les per la xarxa. Un cop acabat el procés les imatges són novament ajuntades i enviades a l'arxiu *.php* mitjançant sockets tal i com s'ha explicat en l'àpartat anterior.

4 RESULTATS OBTINGUTS

4.1 Resultats de la superresolució amb UNet

Un cop creada la xarxa, aquesta va ser entrenada amb el nostre dataset, fent servir un 70% d'entrenament i un 30% de validació. Va ser executada en un *jupyter notebook* a la web de kaggle [15], ja que aquesta plataforma oferia 40 hores gratuïtes d'accés a les seves GPUs a diferència d'altres plataformes com *Google* o *Amazon* que no permeten grans capacitats de càlcul o t'obligaven a pagar pel servei.

Un cop entrenada la xarxa, es va poder observar que, tot i que l'accuracy entre la imatge resultant i la buscada fós només d'un 3%, els resultats obtinguts eren bastant prometedors, tal i com es pot observar a la figura 16.

No obstant, un cop analitzats amb deteniment, es va poder notar com tot i que la xarxa feia una millora de la imatge i es podia veure molt més clar el seu contingut que en la imatge original, els resultats no acabaven de ser els esperats, ja que la imatge resultant es veia borrosa i falta de qualitat.

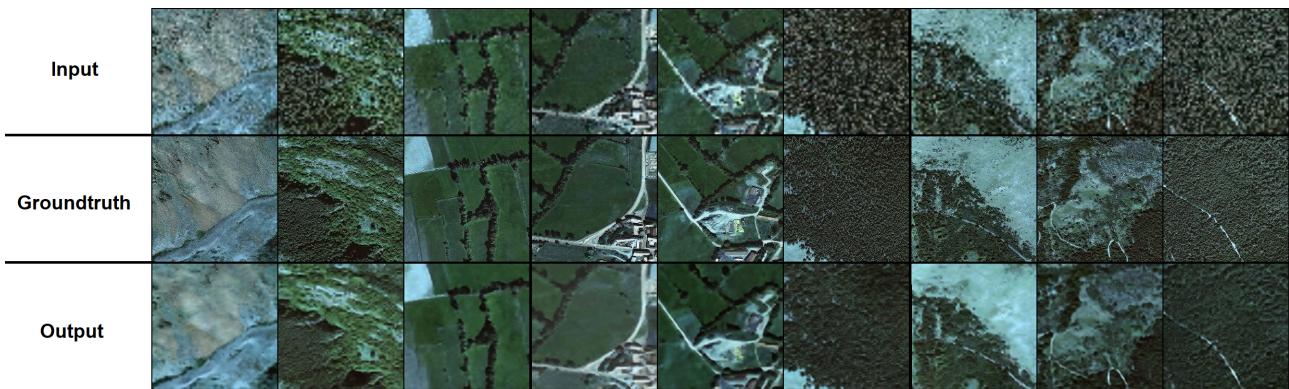


Fig. 16: Resultats obtinguts

4.2 Resultats de la superresolució amb la GAN

Resultats no obtinguts actualment.

4.3 Resultats del visor 3D

Estic esperant els resultats de la GAN per acabar aquest punt.

5 CONCLUSIONS

6 AGRAÏMENTS

REFERÈNCIES

- [1] Laia Gilibets. (2020, November 11). Qué es la metodología Kanban y cómo utilizarla. Retrieved November 11, 2021, from Thinking for Innovation website: <https://www.iebschool.com/blog/metodologia-kanban-agile-scrum/>
- [2] Trello. (2019). Retrieved November 11, 2021, from Trello.com website: <https://trello.com/>
- [3] Waelput, B. (2021, August 18). ¿Qué es y para qué sirve un diagrama de Gantt? Retrieved October 6, 2021, from España website: <https://www.teamleader.es/blog/diagrama-de-gantt>
- [4] Hu, X., Naiel, M., Wong, A., Lamm, M., & Fieguth, P. (n.d.). RUNet: A Robust UNet Architecture for Image Super-Resolution. Retrieved from https://openaccess.thecvf.com/content_CVPRW_2019/Resolution_CVPRW_2019_paper.pdf
- [5] Lu, Z., & Chen, Y. (n.d.). Dense U-net for single image super-resolution with shuffle pooling layer. Retrieved from <https://arxiv.org/pdf/2011.05490.pdf>
- [6] Cui, Y., Liao, Q., Yang, W., & Xue, J.-H. (n.d.). RGB GUIDED DEPTH MAP SUPER-RESOLUTION WITH COUPLED U-NET. Retrieved from <https://discovery.ucl.ac.uk/id/eprint/10129650/1/YingjieCui-ICME2021-final.pdf>
- [7] IEEE Xplore Full-Text PDF: (2021). Retrieved November 14, 2021, from Ieee.org website: <https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9380364>
- [8] Sentinel. (n.d.). SentinelHub API. Sentinel-Hub API. Retrieved September 24, 2021, from <https://www.sentinel-hub.com/develop/api/>
- [9] NASA. (n.d.). NASA Open APIs. NASA APIs. Retrieved September 24, 2021, from <https://api.nasa.gov/>
- [10] Google. (n.d.). Google Earth Engine —. Google Developers. Retrieved September 24, 2021, from <https://developers.google.com/earth-engine>
- [11] Instituto Geográfico Nacional. (2021). Instituto Geográfico Nacional. Retrieved November 14, 2021, from Geoportal oficial del Instituto Geográfico Nacional de España website: <https://www.ign.es/web/ign/portal/ide-area-nodo-ide-ign>
- [12] Codi obert i prototips. Institut Cartogràfic i Geològic de Catalunya. (2014). Retrieved October 2, 2021, from Igce.cat website: <https://www.icgc.cat/Applicacions/Desenvolupadors/Codi-obert-i-prototips>
- [13] Implementación de UNET en TensorFlow usando la API de Keras - Idiot Developer. (1970). Retrieved November 14, 2021, from ICHI.PRO website: <https://ichi.pro/es/implementacion-de-unet-en-tensorflow-usando-la-api-de-keras-idiot-developer-206935901719145>
- [14] PyTorch. (2021). Retrieved December 19, 2021, from Pytorch.org website: https://pytorch.org/papers/WICHI-RUNet-A-Robust-UNet-Architecture_for_Image_Super-Resolution.pdf
- [15] Kaggle Code. (2021). Retrieved December 19, 2021, from Kaggle.com website: <https://www.kaggle.com/gerymaligne/xarxargb>
- [16] Wikipedia Contributors. (2022, January 14). Generative adversarial network. Retrieved January 23, 2022, from Wikipedia website: https://en.wikipedia.org/wiki/Generative_adversarial_network
- [17] Unity Technologies. (2020). Unity - Unity. Retrieved October 9, 2021, from Unity website: <https://unity.com/es>

- [18] Unreal Engine — The most powerful real-time 3D creation tool. (2019). Retrieved October 9, 2021, from Unreal Engine website: <https://www.unrealengine.com/en-US/?sessionInvalidated=true>
- [19] Godot Engine. (2021). Free and open source 2D and 3D game engine. Retrieved October 9, 2021, from Godot Engine website: <https://godotengine.org/>
- [20] Three.js – JavaScript 3D Library. (2021). Retrieved October 9, 2021, from Threejs.org website: <https://threejs.org/>