Aterratge autònom d'aeronaus d'ala fixa basat en visió

Narcís Nogué Bonet

Introducció– Com indica el títol, el meu Treball de Final de Grau consisteix a crear un sistema de control autònom que sigui capaç d'aterrar un avió en una pista d'aterratge utilitzant únicament una càmera i altres sensors bàsics com acceleròmetres i giroscopis. Actualment la majoria de sistemes d'aterratge autònom necessiten modificacions substancials de la pista d'aterratge per instal·lar un sistema ILS (Instrument Landing System), dissenyat per permetre a una aeronau aterrar de nit o en baixa visibilitat. Tot i això, hi ha un subgrup important dels aeroports que segueixen les normes VFR (Visual Flight Rules), on només es pot aterrar de dia i quan la visibilitat sigui suficient, ja que l'única informació que té el pilot és el contacte visual directe de la pista d'aterratge. La majoria d'aeroports petits, aeròdroms i pistes de muntanya cauen en aquesta categoria, i per tant l'aterratge autònom per mètodes convencionals hi és de moment impossible. La solució que proposo deriva directament d'aquesta restricció: si la majoria de pistes d'aterratge estan pensades i dissenyades per a vol visual, un sistema d'aterratge autònom ha de ser capaç d'aterrar de forma purament visual, i sense confiar en cap input des de la pista d'aterratge, per a poder-se considerar plenament autònom i genèric.

1 Objectius

Per la naturalesa del projecte, els objectius del meu Treball de Final de Grau poden augmentar en complexitat molt ràpidament, i per tant els dividiré en dues seccions: els objectius necessaris per tenir un MVP (Minimum Viable Product), i la resta d'objectius opcionals per seguir expandint el projecte més enllà.

Objectius per a un MVP:

- Dissenyar i implementar un algoritme de control capaç d'aterrar un avió model si sap on és la pista d'aterratge.
- Dissenyar una simulació prou acurada d'un cas genèric d'aterratge, sobre la qual poder provar els algoritmes de control i de detecció de la pista.
- Dissenyar i implementar una xarxa neuronal capaç de reconèixer qualsevol pista d'aterratge sobre la qual hagi estat entrenada directament.

Objectius addicionals:

- Construir un avió model capaç d'aterrar de forma autònoma a una pista d'aterratge.
- Dissenyar i implementar una xarxa neuronal capaç de reconèixer qualsevol pista d'aterratge que no hagi vist prèviament.

• E-mail de contacte: nnogue4@gmail.com
• Menció realitzada: Computació
• Treball tutoritzat per: Felipe Lumbreras Ruiz (Department of Computer Science)
• Curs 2020/21

2 Estat de l'art

En la introducció ja he parlat una mica de com funciona l'aterratge autònom avui en dia, en aquesta secció entraré més en detall sobre els sistemes ILS i donaré una ullada a altres projectes similars al meu i com han resolt els problemes que se'm presenten.

2.1 El sistema ILS

El sistema ILS (Sauta, Shatrakov, Shatrakov, & Zavalishin, 2019), anomenat Instrument Landing System o Sistema d'Aterratge Instrumental es considera un sistema d'ajuda per als pilots en situacions de baixa visibilitat, i només algunes categories d'ILS permeten aterratge automàtic a través d'un sistema Autoland. Els sistemes ILS es poden classificar en tres categories: CAT I, CAT II i CAT III, en funció de la precisió que proporcionen en el posicionament de l'aeronau, i només les categories II i III es consideren suficients per a aterratges automàtics.

Pel que fa al funcionament, un ILS consisteix en dos transmissors de ràdio situats a la pista d'aterratge. Un és el localitzador o *localizer* (LOC), que indiquen la direcció de la pista (en la figura 1 es mostren la pista i la senyal de ràdio vistes des de sobre).

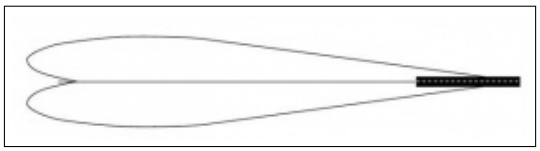

Fig. 1: Rà dio localitzador ILS

L'altra ràdio és la de pendent de descens o *Glide-Scope* (GS), que permet a l'aeronau controlar la ràtio de descens durant l'aproximació. (la figura 2 mostra la pista d'aterratge i la senyal de ràdio vistes de perfil).

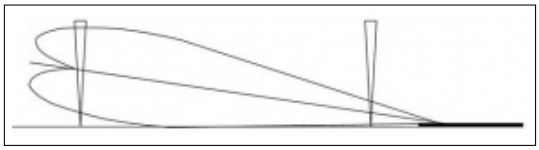

Fig. 2: Ràdio Glide-Scope ILS

El sistema ILS funciona bé i és molt robust, però necessita que les antenes de ràdio a la pista estiguin instal·lades i funcionin correctament, i avui en dia només els aeroports i aeròdroms amb més trànsit solen tenir aquest sistema, les pistes més petites i els aeròdroms en llocs remots solen quedar fora de l'equació pel que fa a aterratges amb ILS.

3 DESENVOLUPAMENT

He actualitzat el diagrama de Gantt (figura 3) per mostrar el progrés del projecte fins al moment, i s'hi pot veure que les tasques de la 1 a la 6 estan completades. Tenint en compte que les tasques entre la 1 i la 5 representen el desenvolupament complet del meu Minimum Viable Product tal i com està estipulat en els meus objectius, considero que el meu progrés en les últimes setmanes ha estat molt adequat i que el meu TFG es podria donar per finalitzat en el seu estat actual de forma satisfactòria, si així fos necessari.

En les pròximes seccions detallaré el progrés que he fet fins al moment. En aquells punts que ja vaig comentar en l'informe actualitzaré la informació on sigui necessari, i afegiré els punts nous que he desenvolupat desde llavors.

3.1 Tasca 2: Investigar possibles solucions

Al principi del projecte es van plantejar varis models d'aprenentatge per a la detecció de la pista, i sobretot se'n van discutir més en profunditat tres:

- Detectors de característiques de l'estil de Sift o Surf (Khan, McCane, & Wyvill, 2011), són molt ràpids però tenen dificultats detectant característiques complexes en totes les condicions possibles.
- Detectors de segmentació semàntica, principalment U-Net (Ronneberger, Fischer, & Brox, 2015), pot ser més lent però és flexible i pot donar prediccions fiables en entorns molt complexos.
- Entrenament End-to-End (Bojarski et al., 2016), on la intenció és que una sola xarxa neuronal resolgui el problema de visió i el de control alhora. Pot donar molt bons resultats però és més difícil d'entrenar correctament.

Finalment he continuat el projecte amb un detector de segmentació semàntica, que consisteix en una xarxa neuronal que rep com a input una imatge i retorna una màscara que separa les parts de la imatge que són pista d'aterratge i les que no ho són (figura 4).

3.2 Tasca 3: Implementar la solució escollida

Aquest apartat és el que ha vist més progrés en les últimes setmanes. Un cop escollit el model d'entrenament em calia trobar una forma de generar la gran quantitat d'imatges requerides per a entrenar un model de segmentació semàntica. Volia un sistema d'aprenentatge no supervisat que pogués generar models i datasets d'inici a fí sense input humà, perquè el temps que tinc és limitat i no em puc dedicar a anotar imatges, i perquè l'escalabilitat d'un sistema no supervisat és molt més senzilla i directa.

He montat un sistema que genera homografies d'imatges satèl·lit per generar una sensació de perspectiva que considero que serà suficient per entrenar un model, i només necessita les coordenades de les quatre cantonades de la pista d'aterratge per fer-ho (de l'estil de les imatges en la figura 4). I com que conec les cordenades de la pista d'aterratge i les seves cantonades puc generar la màscara automàticament també.

Un cop tinc la imatge del terreny generada he afegit un procés que distorsiona la imatge només a la part que correspon al terra, amb la intenció d'evitar que una xarxa neuronal pogués aprendre a reconèixer les homografies de les pistes utilitzades en el dataset, i d'aquesta manera evitar l'overfitting de la xarxa. Afegir aquest pas ha donat molt bons resultats i les xarxes que he pogut entrenar generalitzen molt millor quan es troben amb imatges del món real.

Tot i així, només puc generar una imatge cada 2 segons més o menys, ja que cada imatge fa unes 100 crides asíncrones al servei d'imatges satèl·lit i estic limitat pel major temps de resposta, per tant també he implementat un sistema d'augmentació de dades que fa zoom en diferents posicions de cada imatge i extreu unes 120 imatges noves per a cada imatge original.

Un cop s'han generat les imatges es passen automàticament per l'entrenament de la xarxa neuronal, amb resultats adequats al punt de progrés en que em trobo actualment(figura 5).

En les últimes setmanes he experimentat amb models que puguin reconèixer el cel i la pista, en comptes de només la pista, amb resultats que no han estat ben bé els esperats. Els models funcionen bé, però no queda clar encara que aportin millora respecte els models que detecten només la pista, i per l'arquitectura i algoritme de control que he implementat no és necessaria la detecció del cel.

Tot l'entrenament l'he fet en Google Colab perque m'ha donat millors resultats, així que el codi a Github no està sempre actualitzat. Es pot veure el codi més recent en el que estic treballant aquí.

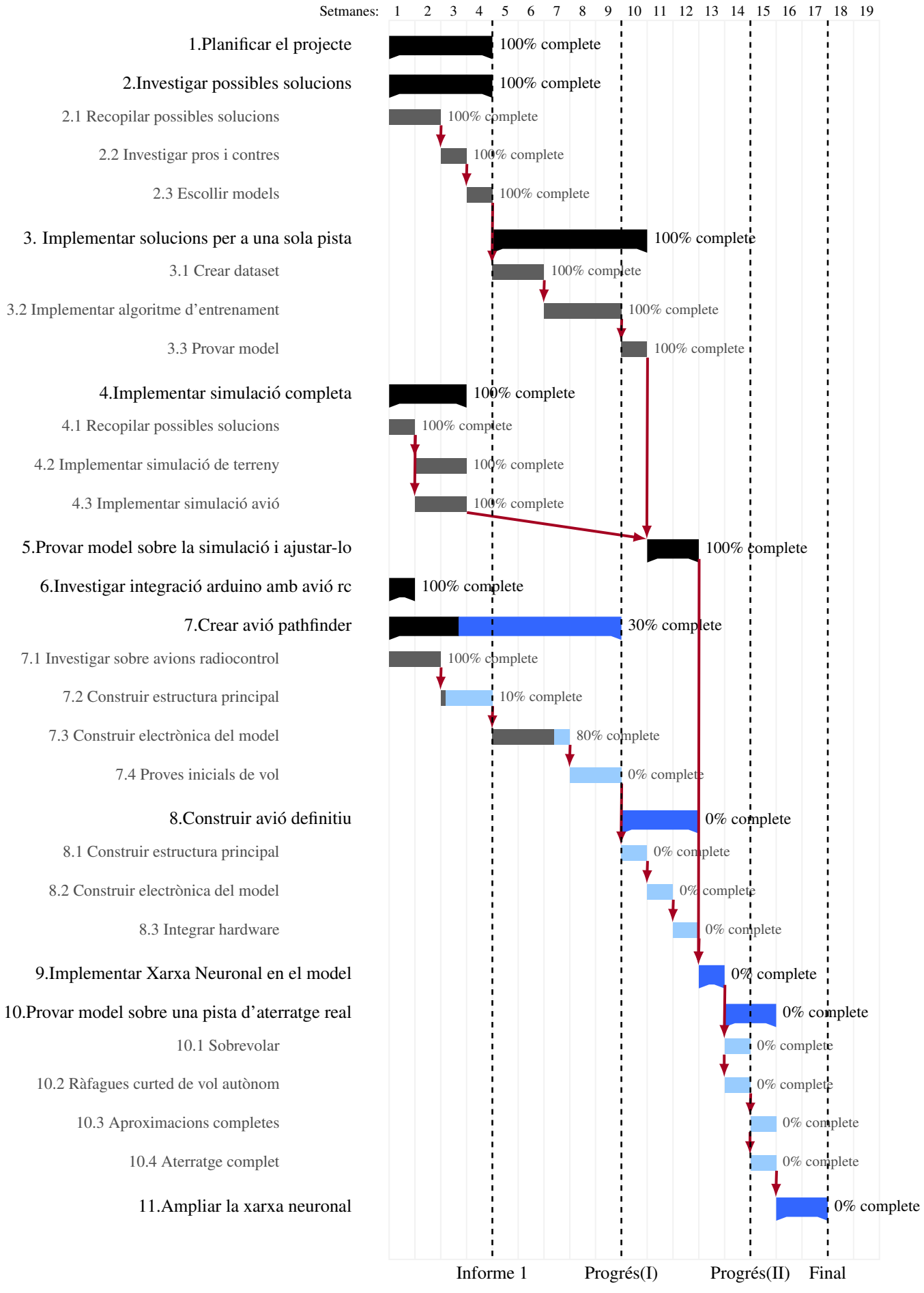

Fig. 3: Diagrama de Gantt del projecte

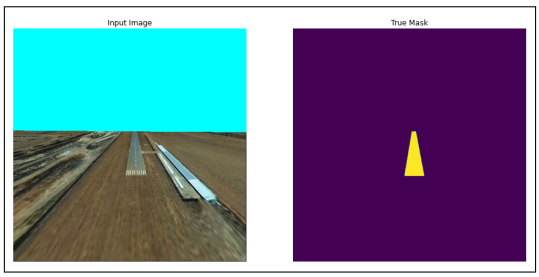

Fig. 4: Exemple d'una imatge i la predicció esperada

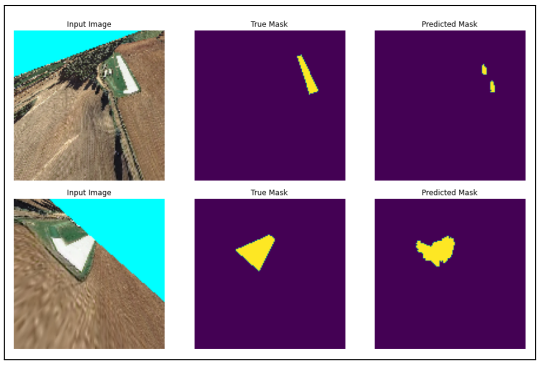

Fig. 5: Dues imatges, les seves prediccions esperades i les prediccions reals

3.3 Tasca 4: Implementar una simulació completa

Per a aquesta tasca vaig decidir utilitzar el motor de creació de videojocs Unity per a fer la simulació d'un avió, un terreny i una pista d'aterratge (figura 6). També vaig implementar un motor físic per a simular el moviment de l'avió segons la pisició de les seves vàries superfícies de control.

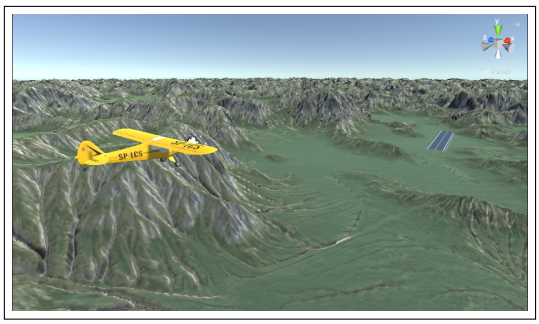

Fig. 6: Simulació de l'avió i la pista d'aterratge

Per a provar el meu sistema de control autònom sobre el simulador necessitava una forma de controlar Unity amb Python, i com que Unity no suporta Python directament he montat un programa de control en Python que es comunica amb el controlador de l'avió a Unity per Sockets. La idea és que pugui rebre la imatge que captura la càmera de l'avió simulat (figura 7), processar-la i enviar les posicions de les superfícies de control que considera necessàries.

Fig. 7: Simulador a l'esquerra i la imatge rebuda pel controlador a la dreta

3.4 Tasca 5: Integració del model i la simulació i programar el control de l'avió

Un cop implementat el simulador i una xarxa neuronal que generalitzi bé he dedicat les últimes setmanes a integrar-los tots dos, de manera que el servidor en python pugués veure l'estat del simulador, analitzar-lo utilitzant els models entrenats prèviament i diferents tècniques d'anàlisi d'imatges, i enviar una resposta al simulador per poder controlar l'avió, tot tant en temps real com fos possible. Desde instal·lar i provar diferents configuracions de drivers de Nvidia i versions de les diferents llibreries que utilitzo fins a utilitzar multithreading per separar la comunicació amb el simulador i l'anàlisi pròpiament dit, finalment tot el bucle funciona 10 vegades cada segon, que és més que suficient per el que necessito.

Per analitzar una imatge el primer que faig és fer una predicció de segmentació semàntica amb la xarxa neuronal per obtenir una aproximació del que és la pista i el cel, després extrec només una màscara per la pista i la erosiono i la dilato seqüencialment per eliminar el soroll. Com que normalment no s'elimina tot el soroll assumeixo que el bloc més gran és la pista i en busco el punt mig i els punts màxims i mínims per trobar les cantonades. Finalment busco el punt mig de l'aresta de la pista més propera a l'avió, ja que aquest punt és el que servirà de referència per al controlador a l'hora de dirigir l'avió.

Per la banda del controlador, llavors, cada dècima de segon arriba un punt representat per dues coordenades, i que representa el centre de l'aresta més pròxima de la pista en el pla en dues dimensions que veu la càmera frontal de l'avió. Com que conec la posició de la càmera i l'angle actual de l'avió (recordo que l'avió incorpora un giroscopi), puc traduïr aquestes coordenades en un vector que indica la direcció i el sentit cap a la pista desde l'avió. Aquest vector és el que es seguirà fins que arribi el següent frame i es repeteixi el procés. Per eliminar soroll, cal remarcar que en realitat el vector es calcula amb la mitjana dels deu últims punts que han arribat, ja que com que el moviment de l'avió no és brusc un delay de aproximadament un segon no impossibilita un bon resultat i ajuda molt en reduïr moviments bruscs per punts que no han estat ben detectats en algun frame.

3.5 Tasca 6: Investigar integració arduino amb radiocontrol

Aquesta és la última secció que vull destacar. El meu tfg també inclou en els seus objectius opcionals construir un avió a escala i aconseguir que aterri automàticament, així que he estat investigant la manera de construïr un avió que tingui la programabilitat suficient per a donar-me la flexibilitat que vull, la potència de càlcul suficient per a fer anar la xara neuronal i tots els càlculs necessaris a un framerate suficient i que es pogués integrar amb un transmissor i un receptor radiocontrol convencionals.

Finalment he acabat amb un sistema que consisteix en un arduino micro com a controlador dels servos i el motor, una raspberry pi zero o una Jetson Nano per al reconeixement de la pista i el control autònom, un Taranis Q X7 per al transmissor radiocontrol i un Taranis R-XSR per al receptor radiocontrol, tots dos de la marca FrSky. He escollit Taranis perquè la majoria dels productes són de codi obert i permeten comunicació bidireccional, de manera que els podria programar per rebre telemetria de l'avió directament al transmissor i saber a temps real l'estat dels algoritmes autònoms i la confiança del controlador en un bon aterratge.

REFERÈNCIES

Bojarski, M., Del Testa, D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., ... others (2016). End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*.

Khan, N. Y., McCane, B., & Wyvill, G. (2011). Sift and surf performance evaluation against various image deformations on benchmark dataset. In *2011 international conference on digital image computing: Techniques and applications* (pp. 501–506).

Ronneberger, O., Fischer, P., & Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In *International conference on medical image computing and computer-assisted intervention* (pp. 234–241).

Sauta, O., Shatrakov, A., Shatrakov, Y., & Zavalishin, O. (2019). Instrumental landing systems. In *Principles of radio navigation for ground and ship-based aircrafts* (pp. 65–71). Springer.