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).

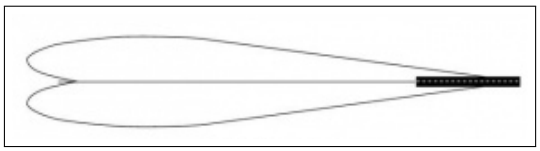

Fig. 1: Rà dio localitzador ILS

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).

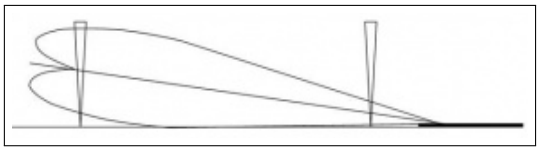

Fig. 2: Ràdio Glide-Scope ILS

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 PLANIFICACIÓ

En aquesta secció descriuré la planificació que vull seguir durant tot el projecte i el procés que he seguit per arribar-hi a través del mètode

3.1 Work Breakdown Structure

Començaré amb l'Estructura Detallada de Treball o Work Breakdown Structure (WBS). En la següent llista organitzaré les tasques que considero que són necessàries en una estructura jeràrquica de dos nivells:

1. Planificar el projecte de forma detallada.
2. Investigar possibles solucions pel problema plantejat.
   - 2.1. Recopilar una llista de possibles solucions (models de xarxa neuronal).
   - 2.2. Investigar pros i contres de cada possible solució
   - 2.3. Escollir models a implementar, poden ser més d'un si vull investigar-ne més d'un en profunditat
3. Implementar solucions escollides per a una única pista d'aterratge
   - 3.1. Crear datasets o simulació simple per a l'entrenament no supervisat.
   - 3.2. Implementar l'algoritme d'entrenament de les solucions escollides.
   - 3.3. Provar cada xarxa neuronal i iterar fins aconseguir la precisió desitjada.
4. Implementar una simulació complexa per provar el sistema.
   - 4.1. Investigar possibles formes de fer una simulació.
   - 4.2. Implementar la simulació de terreny.
   - 4.3. Implementar la simulació de l'avió model.
5. Provar el sistema sobre la simulació.
   - 5.1. Integrar lògica de control sobre la simulació.
6. Investigar integració arduino amb hardware típic d'avions radiocontrol.
7. Construir un avió model pathfinder.
   - 7.1. Investigar sobre construcció d'avions model.
   - 7.2. Construir estructura principal.
   - 7.3. Construir electrònica del model.
   - 7.4. Proves inicials de vol.
8. Construir avió model definitiu, de zero o extenent el model pathfinder.
   - 8.1. Construir estructura principal.
   - 8.2. Construir electrònica del model.
   - 8.3. Integrar el hardware de control autònom al model.
9. Implementar xarxa neuronal i controls en el model.
10. Provar el model sobre una pista d'aterratge real.
    - 10.1. Sobrevolar la pista controlant l'avió manualment i recopilar les dades necessàries per a poder analitzar les decisions del model després del vol.
    - 10.2. Provar l'estabilitat de vol en ràfegues curtes de vol autònom.
    - 10.3. Provar aproximacions autònomes i prendre control just abans de l'aterratge.
    - 10.4. Provar un aterratge complet.
11. Ampliar la xarxa neuronal perquè l'avió pugui aterrar en qualsevol aeroport.

3.2 Diagrama de Gantt

Un cop definides totes les tasques les organitzaré en un diagrama de Gantt per poder veure fàcilment la relació que tenen entre elles i el marge de temps que tindré per a realitzarles. La figura 3 mostra el diagrama de Gantt complet, podem veure que les tasques estan organitzades per setmanes, i hi ha 4 milestones marcades corresponents a les 4 entregues que hauré de fer. També indica el progrés de cada tasca, i per tant aniré actualitzant aquest diagrama a mesura que avanci el projecte.

Per comentar una mica l'organització de les tasques, podem veure que hi ha tres fronts principals que es poden fer en paral·lel: la implementació de la xarxa neuronal, la implementació d'una simulació completa on poder fer tests i la creació d'un model físic on poder integrar tot el sistema, i tots tres blocs s'ajunten en les tasques 5 i 9. Tot i que es fan en paral·lel prioritzaré la implementació en software, ja que és el que constitueix el meu Minimum Viable Product, i per tant he reservat més temps del que considero necessari per a la creació de l'avió model, de manera que hi puc dedicar menys temps setmanalment al principi del projecte.

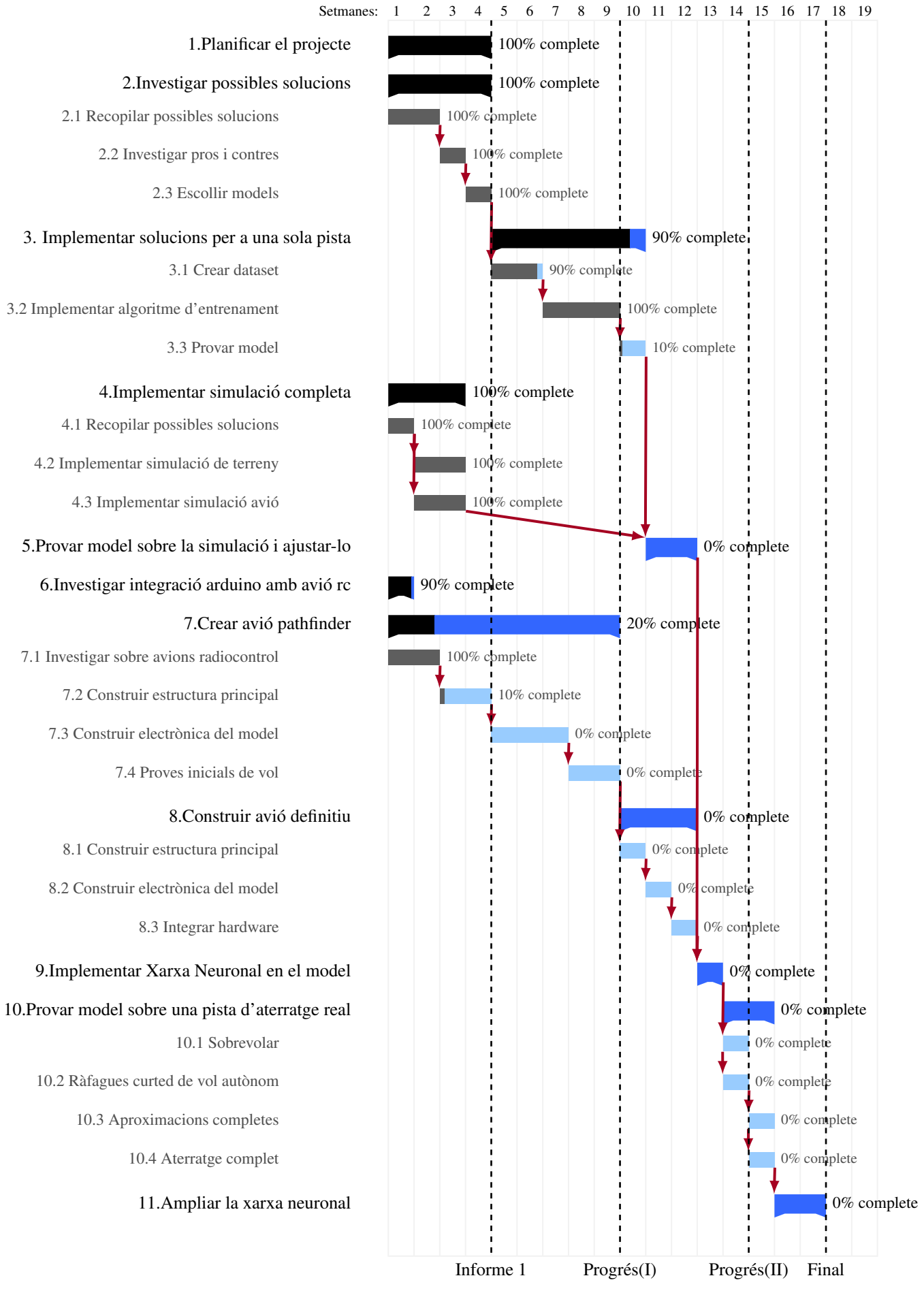

Fig. 3: Diagrama de Gantt del projecte

4 METODOLOGIA

En la metodologia hi ha hagut canvis respecte a l'informe anterior, principalment, he abandonat el jira per al control de tasques i temps ja que considero que per a un equip d'una persona no és necessari i semblava una càrrega més que una ajuda. Per tant les eines que segueixo utilitzant són:

- Github per al control de codi font. Tot el meu projecte estarà en aquest repositori de git: Aterratge Automàtic basat en visió, per a poder tenir un històric de tots els canvis que he fet, poder treballar des de diferents ordinadors i poder compartir el projecte fàcilment. L'únic aspecte del treball que segurament hauré d'excloure del git són els datasets per no superar el límit d'emmagatzematge de 10Gb, però els algoritmes de generació de datasets sí que hi seran.

Pel que fa a la metodologia de treball, tot el projecte ha quedat repartit en 17 setmanes, i està previst que tot el TFG ocupi al voltant de les 300 hores, per tant cada setmana hauré de dedicar unes 18 hores a treballar. Cada dia de la setmana podré dedicar unes dues hores després de la feina a fer el treball, i per tant els caps de setmana hauria de fer quatre hores cada dia. També he previst una hora de reunió amb el meu tutor cada setmana per comentar el progrés i assegurar que mantinc un bon ritme de treball i comentar la feina de la setmana.

5 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 5 han seguit el ritme previst. Les tasques de la 6 en endavant corresponen als objectius opcionals del treball així que no és preocupant que quedin una mica enrere, tot i que també mostren progrés. Algunes de les tasques mereixen una explicació en més profunditat ja que constitueixen la majoria del progrés, així que les detallaré a continuació:

5.1 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).

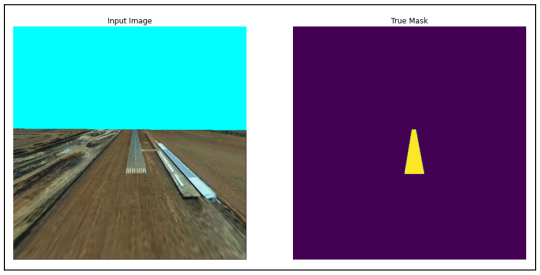

Fig. 4: Exemple d'una imatge i la predicció esperada

5.2 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.

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).

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.

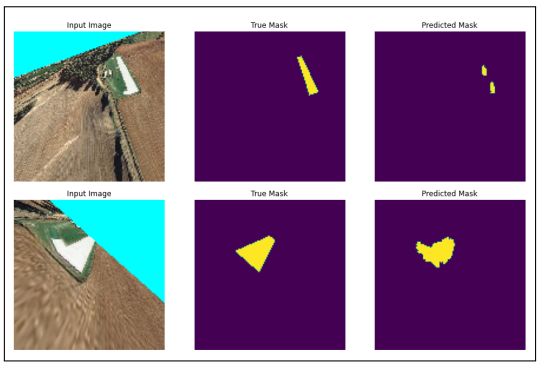

Fig. 5: Dues imatges, les seves prediccions esperades i les prediccions reals

5.3 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.

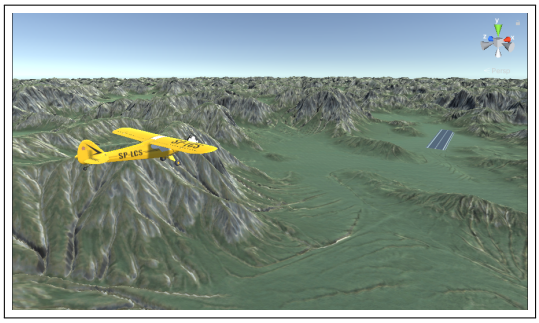

Fig. 6: Simulació de l'avió i la pista d'aterratge

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

5.4 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.