Simulations for Vision-based Navigation and Collision Avoidance in Unmanned Aerial Systems

Howard Mahé, Mahmoud Ghoniem, Valentine Copin, Jennifer Vandoni, Julien Farjon

Simulations for Vision-based Navigation and Collision Avoidance in Unmanned Aerial Systems

Howard Mahé[1], Mahmoud Ghoniem[2], Valentine Copin[2], Jennifer Vandoni[1], and Julien Farjon[3]

[1] *Safran Tech, Rue des Jeunes Bois, 78114 Magny-les-Hameaux, France*
[2] *Safran Electronics and Defense, 100 av de Paris, 91344 Massy, France*
[3] *Safran Electronics and Defense, 21 av du Gros Chêne, 95610 Eragny, France*
Email: name.surname@safrangroup.com

KEYWORDS: simulation, image processing, navigation, localization, detect and avoid, perception, passive ranging, time-to-collision, neural network, AI

ABSTRACT:

The article presents two avionics applications of synthetic image rendering for the development of image processing algorithms and AI training for unmanned aerial systems (UAS). First, the design of vision-based navigation algorithms is explored and validated using a 6DOF simulator of a camera-equipped aircraft. Then, a mixed database of synthetic and real images is used to train new AI perception models dedicated to visual recognition and passive ranging from videos. These two applications show how simulation and synthetic images can be useful throughout the development cycle of image processing solutions, from optronic design to performance validation.

1. INTRODUCTION

The growing number of unmanned aerial systems (UAS) requires Detect & Avoid (DAA) and Navigation capabilities, such as localization or path planning, to prevent collisions in shared airspace. For manned aviation, Airborne Collision Avoidance Systems (ACAS) work by warning pilots of the presence of other aircraft or obstacles that may present a threat of collision. Commonly used technology relies on airborne radar or cooperative communications with other aircraft's transponders. In both cases, ACAS are bulky, expensive, and require active sensors emitting electromagnetic waves. A challenge is to operate these functions in a non-cooperative scenario i.e. without any information from other aircraft. Also, traditional navigation systems based on inertial measurement unit (IMU) and GPS hybridization are not always accurate in GPS-denied environments or low altitude flights. Camera sensors make it possible to explore new use cases for airborne systems beyond the limits of conventional navigation and perception methods while benefiting from an excellent trade-off in terms of size, weight, power, and cost.

A large database of images and video sequences is needed to develop, validate, evaluate navigation or perception algorithms, and ultimately assess whether they meet system requirements. Ideally, the entire design process is carried out using real data representative of the domain of use. However, the volume and variety of real data are often limited due to the cost and complexity of setting up acquisition campaigns. Additionally, some flight scenarios, e.g. near midair collisions [1] or certain geographical areas, are difficult, dangerous, or even impossible to acquire in real conditions. Thus, simulations are best suited to provide the required quantity and diversity of scenarios. Particular attention must be paid to the realism of the simulations. Depending on the algorithm, the requirements in terms of realism are not necessarily the same. The main research question we consider is: can we use the rich annotations and the amount of data provided by simulation:

(a) to design vision-based navigation algorithms?
(b) to train new AI perception models?

To address this question, related work in the literature is reviewed in Section 2. The criteria for selecting a simulator depend on the application and are detailed in Section 3. Then, the design of vision-based Navigation functions is covered in Section 4. Finally, a neural network is trained using synthetic images to solve a perception task for Detect & Avoid in Section 5.

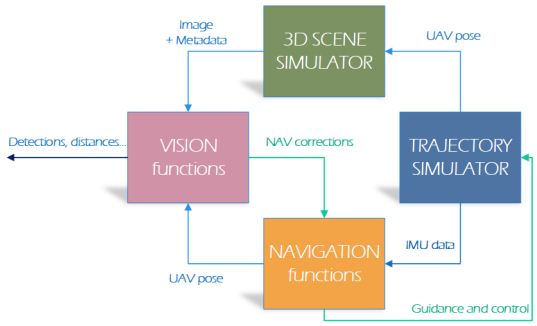

Figure 1: 3D scene and trajectory simulators help developing Computer Vision functions.

2. RELATED WORK

Recent works show the advantages of exploiting the photo-realistic and the endless diversity of modern simulation software and game engines to generate training datasets of images, metadata, and annotations.

Gaidon et al. [2] use a game engine to make the Virtual KITTI dataset, composed of non-photorealistic synthetic clones from real driving scenarios, and then explore challenging conditions. UnrealCV [3] and AirSim [4] are plugins for game engines that enable specific features such as communication protocoles [3], annotation generation or physics simulation, and control of a UAV with a flight controller [4]. Müller et al. [5] have developed Sim4CV for generic computer vision research, including object detection, object tracking, camera localization, navigation, etc. It provides three main features: automatic world generation, communication interfaces with common software, and applications for UAV target tracking and autonomous driving. Alvey et al. [6] describe a workflow for collecting photorealistic simulated data with associated metadata via open source tools to assist deep learning computer vision research for UAVs. In particular, Alvey et al. [6] introduce a data simulation workflow for monocular depth estimation in autonomous driving applications. First, the authors validate that a monocular depth estimation model trained on real-world KITTI dataset performs well on simulated data from a game engine. Then, a qualitative comparison assesses that the same model architecture trained from a mix of KITTI and Virtual KITTI datasets performs better than the former model.

In DAA systems, passive ranging is the process of estimating the distance between a camera sensor and detected targets aka. intruders. This task is often related to the ability to estimate the time before a collision might occur, also called time-to-collision. Some other works focus on the development of AI-based algorithms to solve this perception task using neural networks [7, 8, 9]. These works also exploit simulated data, but their objective is rather to optimize the neural network architecture, whilst we investigate the role of simulated data in AI training. Additionally, previous works usually refer to ideal scenarios such as clear sky, no clouds, and short distance ranges, leading to a simplified task for which the intruders are well contrasted and extend over more than 10 pixels in the images.

For UAV navigation, several datasets exist and provide a set of sequences to benchmark vision solutions on generic navigation scenarios. Fonder et al. present Mid-Air [10] a multi-purpose synthetic dataset for low altitude drone flight. Wang [11] present MineNav, an expandable synthetic dataset based on Minecraft for aircraft visual navigation taking advantage of the open-source contributions of the large community of the game in terms of features and plugins. Despite the quality and diversity of these datasets, they do not meet all our needs. In particular, the terrain overflown or the UAV trajectory are not sufficiently representative of reality. For example, in the case of power lines inspection mission, it is necessary to focus on scenarios with power lines of different natures in different configurations. Therefore, a simulator to generate the custom dataset that fits the study needs is a practical tool if not essential.

3. SIMULATORS

3.1. Simulation needs

Simulation benefits throughout the development of computer vision functions. However, the use of synthetic data varies across the process of technological maturation.

At the beginning of the process, there is seldom representative real data available to carry out the feasibility study and the definition of requirements. Thus, having a scene simulator is a real opportunity to create synthetic scenes corresponding to the use cases (flight speed and height, scene context, etc.). Only a few samples are needed at this stage. These samples are not very accurate and may lack realism. Indeed, it is mainly used for reducing the risk of the concept under consideration. Therefore, synthetic data can strongly contribute to establishing a fast proof-of-concept (POC) in the early stages of the development of a computer vision solution.

Once the concept is defined, the design stage can start. It includes system design, functional design, and optronic design. This covers the study of different architecture choices and their comparison on a small representative dataset. Compared to the POC stage, the evaluation and comparison of the different architectures studied require a larger amount of data. If the initial solution comes from a different domain, the design task begins by evaluating its behavior in this new domain. The simulation tool also contributes to the optronic dimensioning phase by testing a variety of sensor or lens parameters that are difficult to cover in reality at a reasonable cost. In addition, it facilitates the study of the optimal mechanical configuration of the UAV e.g. position and tilt.

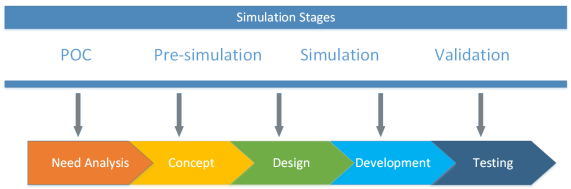

Figure 2: Development of technologies with simulation.

3.2. Task specifications

The democratization of small drones with compact onboard cameras made it possible to achieve vision-based navigation [12] as well as the perception of the surrounding environment. We focus on four vision applications, namely visual odometry, map registration, beacon registration, and passive ranging with different needs in terms of simulation.

Visual odometry. Visual odometry aims at measuring the relative pose of a vehicle thanks to the analysis of the image content along the time. State-of-the-art solutions [13] often rely on the strategy of detecting and tracking keypoints, then estimating the displacing by a bundle adjustment as described in Algorithm 1.

Algorithm 1 Visual odometry

Inputs: images
Output: 3x4 pose matrix

1: Detect and extract 2D keypoints on each frame
2: Keypoint matching with previous frames
3: Bundle adjustment to estimate
   a: The relative pose of the camera observing the scene
   b: The 3D position of the keypoints

Map registration. Map registration estimates the geographic position of the vehicle (Algorithm 2). Matching is performed between the current view of the onboard camera and an area extracted from georeferenced cartographic data. Once both images are registered, the exact pose of the camera is estimated by solving a Perspective-n-Point (PnP) problem.

Algorithm 2 Map registration

Inputs: images, camera poses, cartographic dataset
Output: camera pose

1: Extract the cartographic view from known (approximate) pose information (cartographic image)
2: Register current image and cartographic image:
   a: Extract image features
   b: Compute the geometric transformation between both images
   c: Estimate the camera pose using a PnP approach

Beacon registration. Beacon registration also provides an estimation of the geographic position of the camera, but without resorting to a cartographic image (Algorithm 3).

Algorithm 3 Beacon registration

Inputs: images, camera poses, beacons geodata
Output: camera pose

1: Detect beacons by image analysis
2: Project georeferenced beacons on the current image using the estimated camera pose
3: Match detected and projected beacons
4: Estimate the camera pose using a PnP approach

Passive ranging. Passive ranging consists in estimating the distance between an observer (camera) and an object (intruders) by using passive sensors. The estimated distance in addition to azimuth and elevation angle measurements allows 3D tracking capability of targets for DAA system. Our method uses a neural network trained for distance estimation using target size observation and priors on target dimensions. For a given forward-facing intruder, ignoring the sensor optical aberrations, the observed wingspan ℓ in pixels is defined as:

$$\ell \approx \frac{L}{Z \cdot \text{IFOV}} \tag{1}$$

where IFOV is the angle in radians subtended by a single imaging cell, L is the wingspan of the intruder in meters, and Z is the depth to the intruder in meters. Tab. 1 calculates this size for different configurations.

Table 1: Evolution of the target size in pixels for various imaging sensors and depth in range [1000m;5000m]

IFOV \Depth	1000 m	2000 m	3000 m	4000 m	5000 m
0.2 mrad	50 px	25 px	16 px	12 px	10 px
0.5 mrad	20 px	10 px	6 px	5 px	4 px
0.8 mrad	12 px	6 px	4 px	3 px	2 px

3.3. Simulator capabilities

Over the past few years, game rendering engines have proven their supremacy when it comes to photorealism. However, they do not simulate images outside of the visible spectrum, unlike the advanced physics-based rendering engine. Environment modeling capabilities (terrain, objects, light, sky) vary from simulator to simulator in terms of implementation, diversity, size, detail, and cost. For example, diversity is usually improved using procedural generation.

Computer vision functions are known to be quite sensitive to sensor modeling and after-effects such as optical aberrations or motion blur. Exporting metadata (geolocation, optical flow, depth maps, segmentation labels) should be simple.

Our use of simulators requires simplified management of scenarios (path following constraint, keyframe, etc.) using existing tools or a scripting interface. Some applications may also need to reproduce real environments (road, beacon) or real conditions (sunshine, weather). Some others have strong render speed or real-time constraints.

3.4. Simulation implementation

Fig. 3 shows our simulation pipeline for a DAA scenario. Some input data is provided to the simulator to define the scenario: carrier and intruder trajectories, intruder aircraft models, terrain models, atmosphere and temperature models. The sensors are also explicitly defined in terms of resolution, field of view, and spectral bands. The simulator then creates the scenario and produces the simulated images. The output data is a set of luminance images i.e. light intensity received by the sensor on a specific spectral band. A sensor model can also be applied to the final luminance image to mimic the physical properties of the sensor e.g. noise and point spread function.

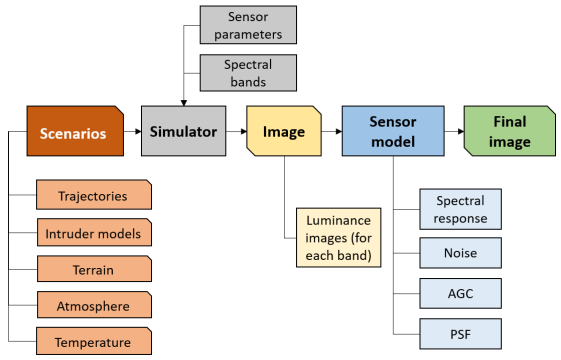

Figure 3: Image simulation pipeline

4. VISION-BASED NAVIGATION

The vision functions used for navigation are based on the analysis of the acquired scene to calculate navigation primitives such as position, speed, or attitude. Features of interest are terrain and objects above the ground such as roads, buildings, and trees. In this section, we will look at how simulators are used to develop a solution for visual odometry, map registration, or beacon registration.

4.1. Method

4.1.1. Scene modeling

The modeled scene must meet several levels of realism depending on the function considered. All functions require at least the scene to be realistic.

Realistic modeling. A scene is realistic when it looks like a real scene even if it does not exist in reality. At the early stage of development, geometric modeling of the terrain and some elements above the ground is a good starting point for a quick proof of concept. Then, a higher level of realism is achieved by taking into account physics and sensor performance for evaluation and validation. Indeed, the image is a representation of a scene acquired by a camera. The rendered image is highly dependent on the camera properties. In order to obtain a realistic image, it is necessary to reproduce the parameters of the camera and apply the reproduced processing chain (Fig. 3) before using the image for high-level functions such as vision-based navigation.
It is possible to use non-existing scenes (Fig. 4) to determine the effectiveness of a function in challenging conditions. Populating scenes with a collection of photorealistic models of elements above the ground (transmission towers, buildings, and trees) allows a variety of scenes to be simulated, aligned to the use domain but also covering more challenging conditions for stress testing vision functions.

Figure 4: Example of non-existent scene that could be used for beacon registration.

High-fidelity modeling. Having a realistic scene is a requirement for all vision functions. However, this is not enough for some specific tasks such as map registration. Not only must the scene be realistic, but it must also be a high-fidelity reproduction of an actual location on Earth.
For map registration, the main challenge is to match the aerial image of the scene overflown with an actual georeferenced reference image. Thus, in the case of aerial image simulation, the image registration step requires that the modeled scene be strictly faithful to reality. Otherwise, experiments based on low-fidelity scenes (Fig. 5) will lead to mismatches and erroneous conclusions.
Therefore, vision functions involving real georeferenced data require high-fidelity scene modeling in terms of geometric content and scene layout. A reasonable way to get a high fidelity render is to overlay an orthophoto instead of synthetic textures (Fig. 6).

Figure 5: On the left, the real scene. On the right a low-fidelity modeling of the same scene.

Figure 6: Real orthophoto used for a high-fidelity texture rendering.

4.1.2. Environment interaction

At this stage, the simulated image takes into account the elements on the ground and the characteristics of the camera. For aerial navigation, the carrier aircraft has a strong impact on images due to vibrations. Indeed, onboard the cameras undergo important vibrations which might make the images blurry or shaky. In addition, the weather is another source of image quality degradation. Indeed, haze, fog, or backlight reduce the observability of the scene. Modeling the daylight, the weather, and the season are key elements of any scene simulator. It is important to test the robustness of the vision function on simulated scenes that stick as closely as possible to real conditions with respect to physics.

4.1.3. Trajectory simulation

Besides the scene simulator, the path simulator is another major component of a complete simulator for vision applications. The kinematics of a fixed-wing aircraft or a multi-rotor aircraft are very different. More-

over, translational and rotational speeds vary during different phases of flight such as takeoff, maneuvering, and landing. The trajectory simulators make it possible to create different trajectories to test the behavior in all cases.

4.1.4. Generated data

The simulator offers the possibility to generate images covering a wide range of scene parameters (terrain, trajectories, weather, etc.) but it can also generate other data that are helpful to develop and validate vision applications.

Depth map. Generating the depth map corresponding to the images (Fig. 7), is a good practice for developing visual odometry solutions. Indeed, monocular solutions require an additional sensor such as a telemeter to address the scale estimation. This could be achieved using the depth map and the telemeter spot model. Additionally, the depth map is necessary to validate the estimate of the distance to the scene.

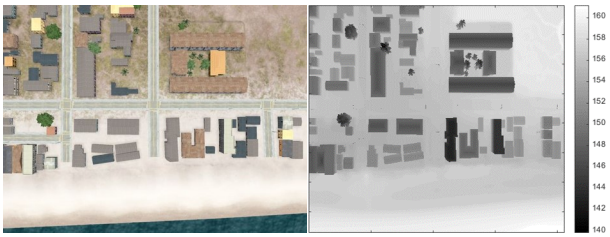

Figure 7: A synthetic image and associated depth map.

Segmentation labels. For beacons registration method, the ground truth of the beacons is needed to learn and evaluate the algorithm. Semantic segmentation of the scene could be generated using a label map corresponding to the scene elements of interest such as beacons.

Navigation data. As seen in the registration algorithms 2 and 3, the camera pose is an input. It is, therefore, necessary to know the position and the attitude of the aircraft as well as the relative position and orientation of the onboard camera. Nevertheless, the real pose differs from the pose estimated by the navigation block. Indeed, the navigation simulator must model the navigation sensors and their errors to create position and attitude data representative of the real sensor. An effective registration would compensate for errors in the input navigation data as shown in Fig. 8.

4.2. Results

4.2.1. Optronic dimensioning and camera positioning

During the development of a new vision system, the question of optronic dimensioning arises very early. It is important to choose optronic parameters such as focal length, image resolution, camera mounting angle

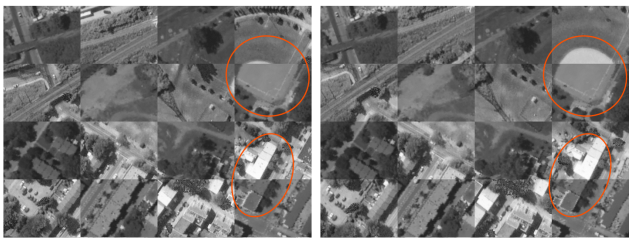

Figure 8: Map registration correcting the estimated position and attitude of a UAV on synthetic data. The image layout is a chessboard to compare the image generated from the navigation data with the ground truth. The image on the left shows the error of the input navigation data and on the right the registration result.

Figure 9: Camera oriented to the front (left) and to the nadir on the same scene (right). The optimal pitch depends on the application.

(Fig. 9), or baseline between cameras of a stereo system, for optimal observation of the scene. Testing multiple cameras with different settings and benchmarking these setups is extremely easy with the use of simulators. Fig. 10 shows the position and angular errors for different baselines and camera pitches for a visual odometry task. Simulators greatly contribute to the choice of the appropriate optronic configuration for a given need.

4.2.2. Testing terrains

During its mission, the aircraft must follow certain kinematic constraints. The trajectory simulator must generate trajectories representative of the in-flight behavior of the aircraft as shown in Fig. 11. Subsequently, these trajectories are used on different terrains to validate the behavior of the vision function in urban, semi-urban, or rural contexts.

4.2.3. Testing trajectories

The motion and trajectory of the aircraft are key parameters to generate representative sequences. If trajectories recorded at real flights are available, they can be used on different terrains. The scene simulator then allows the generation of synthetic sequences from real trajectories.

However, most of the time even the trajectories are not available and have to be generated using a trajectory simulator. This simulator must take into account the motion model of the aircraft. For the early stages of development, when the flight of the aircraft has not yet

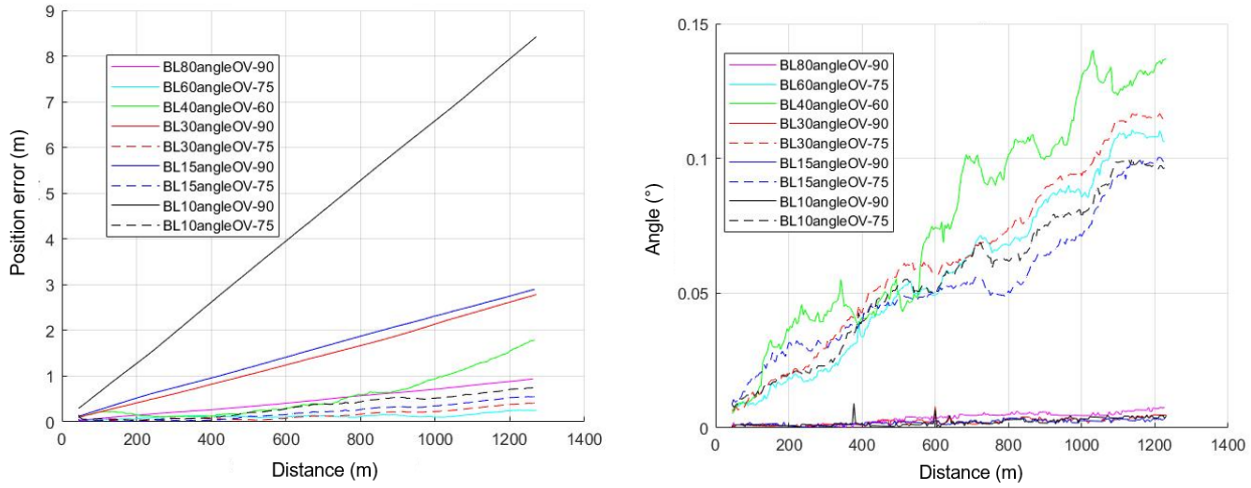

Figure 10: Benchmark of different baselines and camera pitches for a stereo vision system. **BL** *designates the baseline and* **AngleOV** *the pitch of the camera (−90° corresponds to the nadir).*

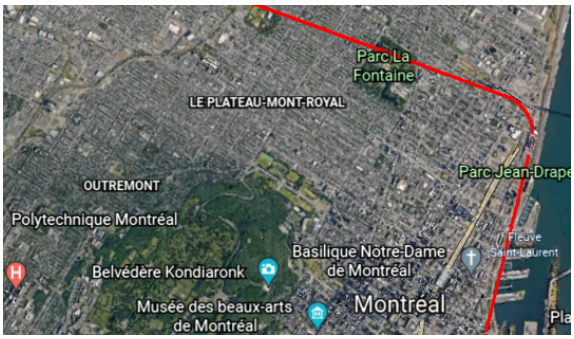

Figure 11: Examples of generated trajectories.

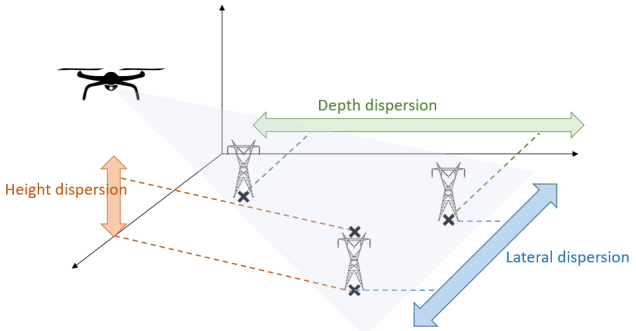

Figure 12: Pylons layout parameters for beacon registration.

been modeled, trivial trajectories, such as straight trajectories at a constant speed, make it possible to start the studies. Then, the complexity grows by adding accelerations and attitude changes.

4.2.4. Testing the influence of ground elements

The behavior of all vision algorithms strongly depends on the complexity of the observed scene. Indeed, all rely on the extraction of features from the scene in order to calculate navigation primitives. Simulators allow controlling the scene components, in particular artificial elements such as buildings, vehicles, or high power poles which we consider as beacons. Each simulator's community or publishers provide many models of common man-made components. For example, we can add buildings and control their height to test behavior with surfaces at different distances from the camera. For beacon registration, it is important to test several high power tower layouts as shown in Fig. 12.

4.2.5. Use domain

Determining the limits of the use domain takes a lot of time and often cannot be completely carried out in practice due to a lack of data for certain extreme cases. Such a problem can be mitigated with a simulator where the user has full control of the most impacting parameters. The main difficulty becomes the analysis of a large number of evaluation results. Fig. 13 highlights the limits of the use domain for visual odometry with respect to speed and flight height. The red disks correspond to unacceptable values of the position error while the green ones correspond to acceptable flight speed/height pairs.

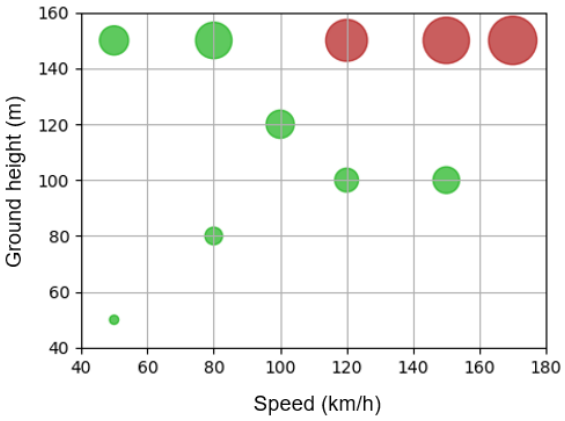

Figure 13: Evaluation of the position error at different configurations of speed and height of flight. The disk size and color depict to the position error.

5. DETECT & AVOID

This section describes our workflow for developing a passive ranging function as part of a DAA system for UAVs. Section 5.1 describes our engineering choices through simulation, data processing, and the design and training of deep neural networks. Section 5.2 and 5.3 present the performance evaluation of our algorithm in the visible and infrared domain, respectively.

5.1. Method

5.1.1. Image simulation

To evaluate the distance estimation neural network, we decided to simulate a large amount of synthetic images. Two datasets with increasing simulation quality were created.

Dataset Rev.1 contains images in the visible spectrum. The main concern for this dataset was to obtain simulated data easily and quickly, in order to get a first insight into the distance estimation problem under ideal conditions. The main characteristics of this dataset are: ideal atmospheric conditions (clear sky, maximum visibility, no clouds), straight deterministic forward-facing trajectories, a stationary camera, and 5 different types of intruders (including civil aircraft, light aircraft, and hot air balloons). The generated dataset is finally composed of approximately 750 different incoming trajectories for each type of intruder.

Dataset Rev.2 concerns both the infrared and the visible spectrum and is noticeably more complex. Indeed, it is composed of 7 different intruders with incoming, outgoing, and orthogonal trajectories. Seven different types of intruders are simulated under three different atmospheric conditions: good, average, and bad weather with decreasing range of visibility. Also, 3D clouds are added in some scenarios in the surrounding environment, only for testing (not training) purposes. Fig. 14 shows some examples of simulated IR images for different intruders.

Finally, we also collected real data for the final validation of our algorithms. The real data comes from acquisitions that have been made in the field. They were annotated with bounding boxes around the intruders for a preliminary detection task.

5.1.2. Sensor model

To make the simulated images more realistic, it is possible to add a sensor model. A sensor model applies to perfect luminance images and aims to model sensor properties and defects such as distortion, vignetting, point spread function, noise, and automatic gain correction. For our study, we designed a sensor model that consists of a succession of functions that model these properties of the sensor. Real images were used to adjust the parameters of the different functions so that they were representative of the real sensor used for the acquisitions. Fig. 15 shows an example of a simulated image before and after applying a sensor model.

Figure 14: Synthetic IR images simulated using a physics based rendering engine.

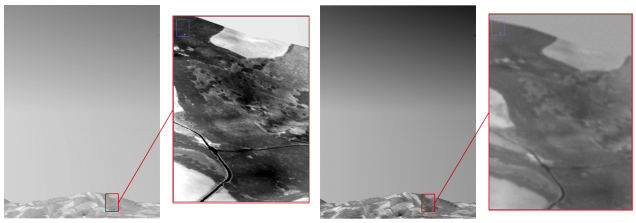

Figure 15: Simulated infrared image before (left) and after applying (right) sensor noise model.

5.1.3. Data preparation

Intruder detection is beyond the scope of this article. We assume that the bounding boxes detected around intruders are an input to our passive ranging algorithm. An image time series (Fig. 16) is created for each detected intruder and cut into short clips, denoted *tubes*, of shape (T, H, W, C) where H, W are the spatial resolution of the crop around the intruder in height and width respectively, T is the number of timestamps, C is the number of channels. Data standardization is performed as preprocessing step, along with basic data augmentation operations.

Figure 16: Image time series of an intruder aircraft.

5.1.4. Neural network architecture

The neural network is fed by a preprocessed and augmented tube. The neural network architecture is a 3D CNN that uses 3D convolutions to process spatio-temporal input tensors, using both spatial and temporal information simultaneously. The neural network infers, for one or more images of the tube, the distance from the onboard camera to the detected intruder aircraft. The estimated distance value is then provided as input to the collision avoidance algorithm. The evaluation of the passive ranging function aims to assess the quality of the predicted distance.

5.1.5. Training & evaluation procedure

The dataset is divided into three different splits according to the ratio 70:10:20 for the training, validation, and test sets respectively. The splits are created so that the different tubes extracted from a given sequence belong to the same set, in order to avoid overfitting and bias. At the same time, random sampling encourages diversity in appearance and trajectories within each set. The model is trained to estimate the distance $\widehat{d} \in \mathbb{R}^+$, minimizing a mean squared error (MSE) loss or an average relative distance (ARD) loss with the ground truth distance d using stochastic gradient descent. Two metrics (Eq. 2) are defined for the evaluation to measure the quality of the distance regression: mean absolute distance error (MADE) and mean relative ratio of the absolute distance error versus ground truth distance ($\%err_{dist}$), over a specified range interval $\mathcal{D}$ e.g. from 3000m to 3500m.

$$MADE = \frac{1}{|\mathcal{D}|} \sum_{i|d_i \in \mathcal{D}} \left| d_i - \widehat{d}_i \right|$$
$$\%err_{dist} = \frac{1}{|\mathcal{D}|} \sum_{i|d_i \in \mathcal{D}} \frac{\left| d_i - \widehat{d}_i \right|}{d_i} \quad (2)$$

5.2. Results in the visible domain

The method is first tested in the visible domain on **Dataset Rev.1** where imaging sensors generally benefit from good pixel resolution.

5.2.1. Training on simulated images

The training was conducted for 100 epochs. The best-performing model on the validation set is evaluated on the test set. Since the dataset is carefully balanced between the different aircraft types, global results are presented for all classes at once. Fig. 17 shows the distribution of estimated distance versus the ground truth distance. Most distance estimates are within 10% tolerance of the ground truth distance. Tab. 2 contains both distance metrics averaged over 500m range increments in a range from 3000m to 6000m. The absolute distance error is $63.9m$ on average and increases almost linearly with distance while remaining stable and less than 1.58% in terms of $\%err_{dist}$. This result on simulated RGB images shows the AI perception model is able to infer the distance from the input tube with sufficient accuracy in our simple simulation framework.

An independent experiment showed that incorrect distance estimate is correlated with incorrect aircraft class prediction using a distinct neural network, as shown by the blue and red crosses in Fig. 17. This result confirms the hypothesis that distinguishing between different aircraft classes helps the AI model to predict an accurate distance.

5.2.2. Sensitivity to sensor model

The point spread function (PSF) describes the diffraction of light in the focal plane of the camera for a point light source through a fine aperture. This function is strictly related to the resolution and blurring of an optical device. Therefore, we expect this effect to be non-negligible for the performance of the distance regression module. In the case of a circular aperture, the PSF kernel is a pill-box function (i.e. a cylindrical-shaped function). However, it has been shown that due to lens diffraction and other unmodelled characteristics of the optical system, the blur kernel is similar to the Bessel function. Following [14], to simplify our sensitivity study, we apply a Gaussian blur (Eq. 3) to the simulated RGB images using a 11×11 Gaussian kernel (Eq. 4) with variable standard deviation σ_G=0.2:0.3:2.0. For example, σ_G=1.7 corresponds to a full width at half maximum of 4 pixels. Then, an additive zero-mean Gaussian noise with standard deviation $\sigma_A \in \{0, 0.005, 0.010\}$ is applied.

The Gaussian blur is applied to an image I with integer values in $[0, 255]$ as follows:

$$I' = I \circledast K + \lfloor n \rceil \quad (3)$$

where $\circledast$ is the convolution operator, k=11 is the kernel size, $c = \frac{k-1}{2}$ is the center of the kernel, $n \sim \mathcal{N}(0; 255 \cdot \sigma_A)$ is Gaussian-distributed additive noise, $K_{x,y}$ are the Gaussian filter coefficients such that:

$$K_{x,y} = \frac{1}{2\pi\sigma^2} e^{-\frac{(x-c)^2+(y-c)^2}{2\sigma^2}}; x, y \in [0, k-1]^2 \quad (4)$$

Table 2: Absolute distance error on the set set of the simulated RGB dataset.

Dist. range	3000 - 3500m	3500 - 4000m	4000 - 4500m	4500 - 5000m	5000 - 5500m	5500 - 6000m
MADE	46.4m	58.0m	67.0m	74.2m	82.4m	88.0m
$\%err_{dist}$	1.41%	1.54%	1.58%	1.58%	1.57%	1.54%

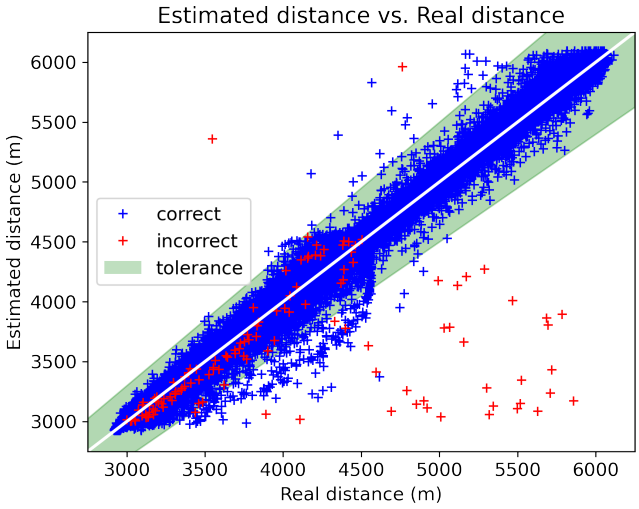

Figure 17: Plot of estimated distance versus actual distance. Blue and red cross marks indicate tubes where the intruder class would have been correctly and incorrectly (resp.) predicted by a separately trained image classifier. The green area represents an 10% error tolerance on the distance prediction.

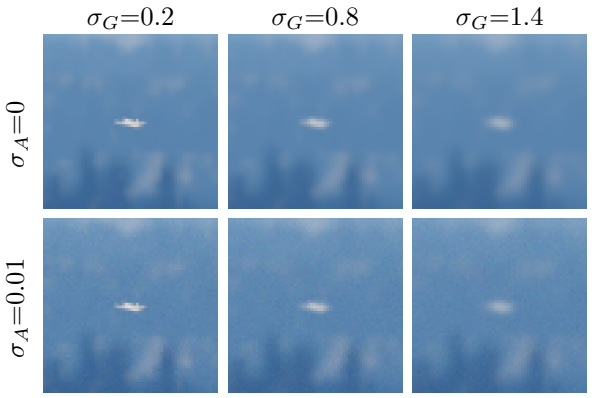

Figure 18: Simulating sensor noise on a synthetic RGB image of a Pilatus PC-9 at 4337m for IFOV=0.2mrad

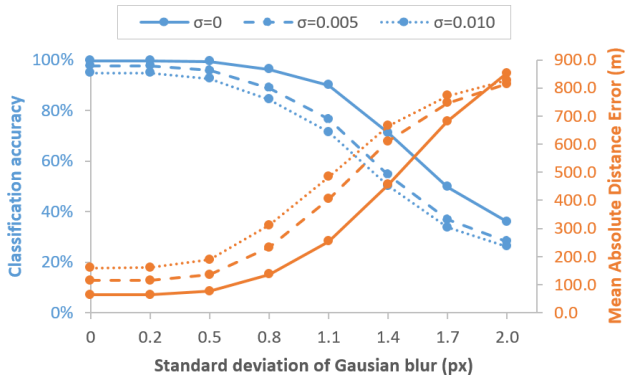

Figure 19: Plot of classification accuracy (blue) and MADE (orange) when Gaussian blur is applied to synthetic RGB images with varying noise amplitudes. The x-axis represents the standard deviation of the Gaussian blur σ_G. The line styles correspond to different values for the standard deviation σ_A of the Gaussian-distributed additive noise. The experiments are carried out on the entire test set and repeated 3 times.

Fig. 18 shows examples of a Gaussian blur with varying parameters applied to a synthetic RGB image. While the task is already difficult on its own, it clearly becomes even more difficult when sensor noise is applied, since the intruder aircraft loses most of its details e.g. at σ_G=1.4. Fig. 19 plots the performance impact when applying a simulated sensor noise model to synthetic RGB images. The absolute distance error slightly deteriorates from 63m to 159m with only additive noise. This shows the sensitivity of the model to sensor noise. For a not so improbable sensor noise magnitude of σ_G=0.8 and σ_A=0.005, the classification accuracy dropped to 88.9% and the MADE reached 232m. When the noise gets stronger, the neural network fails to predict an accurate distance estimate.

This experiment shows our passive ranging model is sensitive to the sensor noise model, which is identified as one of the causes of the domain gap between our simulation and real-world images. This encourages us to introduce sensor noise modeling in our data augmentation pipeline during training, as well as dig deeper into domain adaptation techniques to become less sensitive to the *sim2real* domain gap.

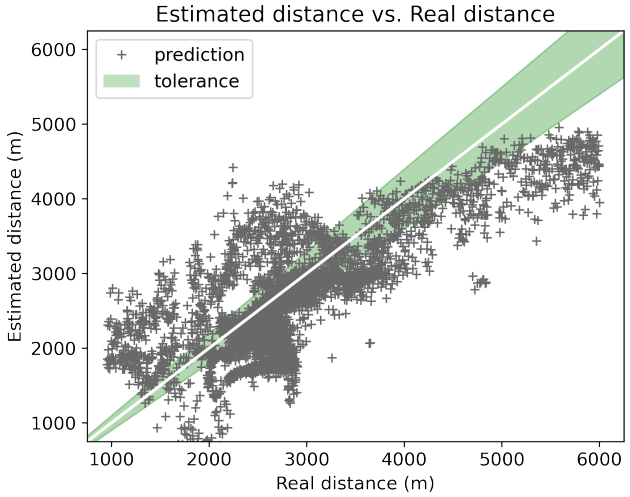

*Figure 20: Estimated distance versus real distance and a 10% tolerance area on the test set of **Dataset Rev.2**.*

5.3. Results in infrared domain

The results presented in this section come from experiments conducted on the IR part of **Dataset Rev.2**. The training was conducted for a hundred epochs. A few trials were done to optimize the training hyperparameters to get the best results on the validation set. On the test set, the best results are around 15% relative distance error across all seven intruder classes. Figure 20 shows the distance estimate versus the real distance. Distance estimation performance is not up to par with results of **Rev.1**. Indeed, our neural network is trained on IR data, not on visible data which benefits from better resolution. Also, **Dataset Rev.2** is more challenging than **Rev.1** in terms of weather and context.

Our best model trained on simulated IR images was then used as initialization for a fine-tuning experiment on real IR images. The goal was to determine if it performs better than random initialization. The results are not obvious. There is a decrease in relative distance error of only a few percent for medium to long distances. It seems that initializing on simulated images is irrelevant in our framework. The problem could be that the simulated images are too different from the real ones. Methods such as domain adaptation might provide better results. They have not been tried in this study but are planned for future work.

6. CONCLUSION

The use of simulation and synthetic images is a major asset for the development of vision-based navigation and perception functions. It can be used from the initial phase of the technology maturation process during proof-of-concept through to the validation phase. It is able to simulate edge scenarios that would be difficult to acquire in reality, and it can also generate massive amounts of data needed for AI training and performance evaluation at a reduced cost. As the technology matures, real data is always needed in addition to syn-

thetic data. Several strategies can be implemented to mix synthetic and real data to design image processing and AI algorithms, such as fine-tuning. Depending on the application, requirements for the similarity between synthetic and real data vary.

In our study, we have described several vision-based navigation algorithms, some of which are sensitive to the realism of the scene modeling, and others require a particular scene layout that preserves geographic data. For perception functions in Detect & Avoid systems, we found that AI models are sensitive to the domain gap between synthetic and real data, especially to the sensor noise modeling of the camera.

In conclusion, we believe it is important to improve the capabilities of the simulator as the technology matures in order to address the limitations identified by the validation phase sensitivity tests.

REFERENCES

[1] I. Borshchova and K. Ellis, "NMAC database," 2021. 1

[2] A. Gaidon, Q. Wang, Y. Cabon, and E. Vig, "Virtual worlds as proxy for multi-object tracking analysis," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4340–4349, 2016. 2

[3] W. Qiu and A. Yuille, "Unrealcv: Connecting computer vision to unreal engine," in *European Conference on Computer Vision*, pp. 909–916, Springer, 2016. 2

[4] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "Airsim: High-fidelity visual and physical simulation for autonomous vehicles," in *Field and service robotics*, pp. 621–635, Springer, 2018. 2

[5] M. Müller, V. Casser, J. Lahoud, N. Smith, and B. Ghanem, "Sim4cv: A photo-realistic simulator for computer vision applications," *International Journal of Computer Vision*, vol. 126, no. 9, pp. 902–919, 2018. 2

[6] B. Alvey, D. T. Anderson, A. Buck, M. Deardorff, G. Scott, and J. M. Keller, "Simulated photorealistic deep learning framework and workflows to accelerate computer vision and unmanned aerial vehicle research," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 3889–3898, 2021. 2

[7] Z.-Y. Huang and Y.-C. Lai, "Image-based sense and avoid of small scale uav using deep learning approach," in *2020 International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 545–550, IEEE, 2020. 2

[8] Y.-C. Lai and Z.-Y. Huang, "Detection of a moving uav based on deep learning-based distance estimation," *Remote Sensing*, vol. 12, no. 18, p. 3035, 2020. 2

[9] D. V. Navarro, C.-H. Lee, and A. Tsourdos, "Sense and avoid using hybrid convolutional and recurrent neural networks," *IFAC-PapersOnLine*, vol. 52, no. 12, pp. 61–66, 2019. 2

[10] M. Fonder and M. Van Droogenbroeck, "Mid-air: A multi-modal dataset for extremely low altitude drone flights," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 0–0, 2019. 2

[11] D. Wang, "Minenav: An expandable synthetic dataset based on minecraft for aircraft visual navigation," *arXiv preprint arXiv:2008.08454*, 2020. 2

[12] Y. Lu, Z. Xue, G.-S. Xia, and L. Zhang, "A survey on vision-based uav navigation," *Geo-spatial information science*, vol. 21, no. 1, pp. 21–32, 2018. 2

[13] C. Campos, R. Elvira, J. J. G. Rodriguez, J. M. M. Montiel, and J. D. Tardos, "Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam," *IEEE Transactions on Robotics*, vol. 37, p. 1874–1890, Dec 2021. 2

[14] F. Soulez and É. Thiébaut, "Joint deconvolution and demosaicing," in *2009 16th IEEE International Conference on Image Processing (ICIP)*, pp. 145–148, IEEE, 2009. 8