Towards the certification of vision based systems: modular architecture for airport line detection

Esteban Perrotin, Matthieu Roy, Ariane Herbulot, Michel Devy, Fabrice Bousquet

Towards the certification of vision based systems: modular architecture for airport line detection

Esteban Perrotin[12] Matthieu Roy[2] Ariane Herbulot[2]
Michel Devy [2] Fabrice Bousquet[1]

[1] AIRBUS Operation
[2] LAAS-CNRS

esteban.perrotin@airbus.com

Abstract

This paper addresses the certification issues for vision based systems embedded on civil aircraft to assist pilots during the ground navigation phase. We propose a design methodology, through the example of a modular architecture to detect ground mark lines in a color image. Our detection method is based on the combination of classical image processing algorithms, deep learning methods and a particle filter algorithm. We argue that the main interest of this architecture is to ease the certification problem when compared to an end-to-end neural network. We discuss about difficulties around certification and propose some arguments.

1 Context

Computer vision applications have made considerable progress in recent years, with applications to many fields, from healthcare to autonomous vehicles. In this context, the civil aeronautic is considering using vision based systems to support pilots in their operations during the ground navigation tasks. For instance, functions such as obstacle detection, axis keeping, runway detection and others are currently studied. However, due to the complexity of such functions, it is challenging today to achieve the certification of systems largely based on computer vision algorithms in civil aeronautic field.

Actually, all functions embedded in civil aircraft have to be compliant to standard design processes defined by the authorities. These processes are described in documents such as Certification Specification [1], the ARP-4754a [2] and DO-178-C [3]. These documents define complete guidelines to design, develop and implement functions to be embedded on civil aircraft. This whole process is called certification. The standard certification process follows a V-model. It consists in designing, developing, implementing, verifying and validating a system by following specific steps described in guidelines. An example of V-model is shown in Figure 1. More details are provided in the section 2.1.

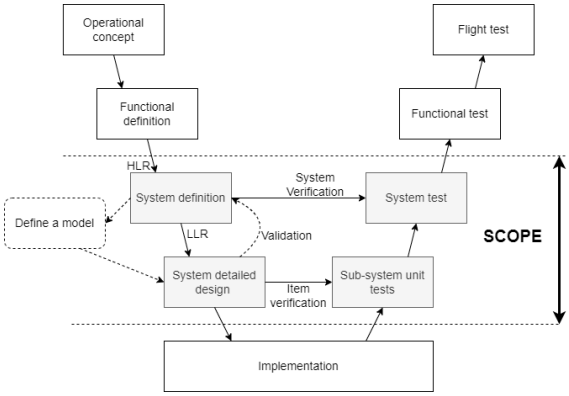

Figure 1: General V-model and our scope.

Some functions embedded on civil aircraft are already using vision based systems (VBS). For example, the A380 from AIRBUS is equipped with a camera integrated in the aircraft fin. The video input can be sent to both passengers and pilots. The pilot can use the video output as an help to drive the aircraft during the taxi phase. The image is directly used and interpreted by the pilot without any algorithm used for its interpretation. However, actual certified functions based on vision systems let the pilot as the principal actor to extract and interpret information represented

in each image. New functions, such as line detection, motion estimation or obstacle detection are using complex algorithms to extract and process features from the image in order to improve the pilot assistance. The certification of such algorithms has never been done in the European civil aeronautic field.
That is why, previous studies [4] highlighted the fact that the design of computer vision-based systems opens a new paradigm in certification. These difficulties are mostly due to the complexity of algorithms. For example, most of recent computer vision algorithms are based on machine learning (ML). Machine learning algorithms learn through examples how to analyze images in order to achieve a desired task. This approach is not compliant with standard design process [5]. Actually, current standards require an explicit description of detailed requirements. This is the opposite of ML systems which they learn the operational behaviors through examples and not from requirements. In order to prepare a new standard, the European Union Aviation Safety Agency (EASA) is working on guidelines [6] [7] to safely design such systems. There are also many public research projects on this topic. The DEEL project is also currently working on this problematic and and have published a White Paper [8]. They address many questions on the risks, challenges and potential solutions of using ML in systems submitted to certification constraints. However, vision based algorithms are not limited to machine learning.

In this paper, we present our position with regards to VBS certification issues. In order to illustrate how to solve these problems, we will take the example of line detection on airport runways/taxiways and describe our methodology to design the modular architecture of a specific line detection algorithm. Line detection is a useful function that could make easier the ground navigation for the aircraft pilot. Due to its simplicity, it is probably one of the first task using vision based algorithms which could be certified. However, most of recent methods repose on end to end deep learning architecture [9]. Because of the certification aspects, we desire to construct a functional architecture without using end to end deep learning methods. Figure 2 shows an example of system architecture that could be used to guide an aircraft on a taxiway, and/or to display information to the pilot. We will focus our discussions on the design and tests of this line detection function architecture. We will not discuss the hardware implementation or

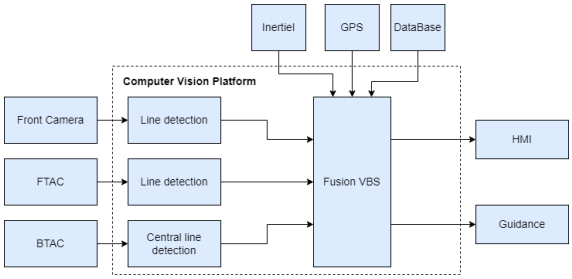

Figure 2: Example of visual based functions integrated in a civil aeronautic system

interface with other systems. The input of the function is an image provided by the Fin Taxi Aid Camera (FTAC) of the aircraft. We make the assumption that this image gives a correct representation of the real world.

The paper is organized as follows: Section 2 presents difficulties that could be encountered in the certification of computer vision based systems. Section 3 presents our proposed architecture in the specific line detection application. Section 4 explains our arguments to help certification of this architecture. Then the last section concludes.

2 General aspect of the certification of Computer Vision-based systems

Computer vision-based algorithms can be summarized in two main parts: extract from each image discriminant information (often called features) and then process this information to achieve the desired operation. Extracting pertinent information is a difficult task. An image can be seen as a 2D signal and the undesired information can be seen as a structured noise. In our application, we aim at extracting features that characterize lines. Hence, we desire to remove all other elements of the image.
In order to answer such problems, complex algorithms of computer vision are used. We will show in the following section that standard guidelines used for the development of civil avionic function are difficult to be applied on complex computer vision items.

In this section, we briefly present general aspect of certification, then we expose steps that could lead to difficulties in the certification of vision based algorithms and expose some elements from the state of art.

2.1 General aspects of certification

The principal guidelines of the civil aeronautics certification is provided by standards such as ARP-4754A [2] and DO-178C [3]. To design a new avionic high level function, the first point of the design process consists in analyzing the impact of this function in case of potential hazards. This step is called functional hazard analysis (FHA). For instance, if a failure of the function could lead to endanger the aircraft or passenger safety, the function will probably be classified as "catastrophic". It will be necessary to prove that the probability for the failure of this function during a flight hour is less than 10^{-9}. This FHA step will drive requirements that should be verified by the implementation of the function.

Then, this high-level function is refined into lower level functions and a preliminary functional architecture is proposed. This architecture is submitted to a preliminary system safety assessment (PSSA). It verifies that requirements from the FHA are fulfilled by the architecture. If requirements are not fulfilled the architecture should be consolidated or a new one should be proposed.

Once a correct architecture is proposed, for each item's architecture a Deasign Assurance Levels (DAL) is addressed. It drives hardware objectives for the specific item and software guideline activities to develop the function. The highest critical level is of DAL A and the lowest is of DAL E. For instance, in case of a DAL A item, the software will be executed on different hardwares reducing the risk of hardware failure and ensuring the availability of the function at every time.

During the design process, multiple tests have to be defined to guarantee and verify that the implementation of items is correct and fulfill requirements at each level. Tests are driven by the system safety assessment (SSA) to ensure the availability and the performance of the function in all identified scenarios.

This safety design process is used to develop high level safety functions for civil avionic system.

2.2 Difficulties around certification for VBS algorithms

The certification process ensures that the proposed high-level function is correctly designed and implemented following specific guidelines depending on the level of criticality of the function. Each sub-level item should verify specific requirements. However, in the case of computer vision systems some aspects of the guidelines seem complicated to satisfy, in particular in systems using machine learning. The authors in [5] and [10] discuss on this problematic for adaptive systems. In [4] the authors detail the certification challenges on a specific VBS dedicated for two applications: visual odometry and obstacle detection.
Considering this previous work and with regards to the difficulties we encountered during our development, we precise some points that seem difficult to answer considering the current certification standards:

- **Comprehensible requirements** have to be written. In computer vision applications writing high level requirements - HLR (*e.g.* detect a specific object) could follow the usual process. However, writing lower level requirements could be harder (e.g, explicit evidences that characterize a given object in all kind of environments).

- **Verifiable requirements**: In order to verify that a computer vision algorithm is correctly working, we have to present different inputs in the system and verify that the outputs correspond to the desired ones. Ideally, we would like to test all potential inputs. But, due to the high dimensionality of images ($255^{1024\times768\times3} \approx 10^{4718592}$ combinations for 1024px $\times$768px color images) it is not realistic to test all possibilities. Most of these images are pure noise or not relevant. A solution could be to provide a minimal dataset, with a distribution of images close to the real application. Furthermore, the desired outputs have to be known for each image tested. Manual labelling on a huge dataset seems impossible and it raises an issue on the accuracy on the labelling process. When using automatic or semi-automatic labelling one has to prove that the algorithm used for labelling is robust with regard to the performance for the high-level function. Validating an automatic labelling process raises similar certification activities. Section 2.3 contain more details about the acquisition of a test data set.

- **Robustness**: The recent advances in computer vision (in particular with deep learning methods) have greatly enhanced performance for many applications. However,these

performances must be defined and evaluated in particular in degraded conditions (fog, rain, low illumination, etc.). In such conditions with environmental hazards algorithms could perform poorly. Avionic systems have to operate correctly despite abnormal inputs and conditions.

In addition the stability of the results have to be considered. For a small variation in the input image, computer vision algorithms have to output the same value. There is no evidence that computer vision algorithms achieve this stability. In particular this case appears with deep learning methods and adversarial attack [11].

- **Traceability**: During the design process, the requirements of a given level are broken down into one or more requirements of the next level. It has to be shown that each low level requirement (LLR) corresponds to a requirement of the higher level (HLR).

- **Interpretability**: This point is not essential for the process of certification but it argues about trustworthiness of the system and helps to convince authorities about the viability of the system. In order to trust a functionality, each step of a function has to be explained and must have its proper purpose based on rationals. This is not true for machine learning based applications. Machine learning learn to extract and process features from the image through examples. Many works have been done to explain which features are extracted, how an interpretation is generated from these features, especially for deep learning methods [12]. However, this aspect should be accepted for classic computer vision algorithms.

2.3 Related works

To address these difficulties, many works have been done, for example to ensure robustness of algorithms, to discuss and propose methods for the creation of test data sets and develop solutions to interpret complex algorithms such as convolutional neural networks.

To ensure robustness of algorithms (accuracy and confidence on the output), methods have been proposed to compute bounces of confidence [13]. These methods could be based either on the demonstration of a formal proof that gives a guarantee about envelope of the algorithm output, or on the execution of enough tests to cover all potential use cases.

In computer vision, due to the complexity of algorithms, the large variety of input images and the lack of precise requirements, it is difficult to develop formal proofs for applications. However, this is already the case in many avionic systems. This is why functions are tested on large data tests. The construction of these test data sets raises many interrogations. Since it is not feasible to test every images, it is necessary to select pertinent scenarios to test algorithms. But algorithms have to be tested in many corner cases, so that acquiring real data for every desired scenario is impossible. A solution consists in generating synthetic images from simulators. Many works have been done to provide image generators useful to test computer vision algorithms for given applications. For instance, we use OKTAL-SE simulator. It generates color or infrared images based on a physical model. However, there is a gap between their simulated image and real image, in particular in the level of detail for the textures representation.

On the other hand, Zendel et al. [14] have proposed a system of guide-words to quantify and qualify hazards that could appear in computer vision applications. Their works provided an answer to "Which situations should be covered by the test data and have we tested enough to reach a conclusion?". By using CV-HAZOP, in [15] and in [16] they propose a standard procedure devised by the safety community to validate complex systems.

In the next section, we describe our line detection algorithm, so that we could illustrate how to deal with the verification of the properties summarized here above.

3 Modular architecture of our line detection method

As said previously, constructing an argument of certification for VBS is a difficult task. This section describes our methodology to construct our line detector. We start by modeling the problem using prior information. Then a general architecture solving this problem. Finally we describe each item in this architecture. Our principal objective is to ease the certification process by diminishing the complexity of verification and validation on each item of our architecture.

3.1 Problem modeling

As said in the previous section, an important point to follow in the certification process is the definition of requirements. The objectives of the function (high level requirements - HLR) should be as clear as possible. In our application, we aim at detecting lines (ground marks). That is why, we define the objectives of our function as the following example: "The function has to detect ground marks that define lines for aircraft ground navigation. The input of the function is an RGB image acquired by an embedded camera (typically in the fin or the cockpit as shown in Figure 3) and parameters set from operational concept. Internal and external parameters of the camera are known. Lines have to be detected until a desired distance to the camera. Output lines will be defined by a set of points.".

Figure 3: Input RGB image from the fin camera.

We start by constructing a model of the lines that will help to fulfill HLR. We construct this model by following some key points:

- Use the maximum of prior information to select important regions of interest (ROI) in the image. These regions have a high probability of containing desired information (presence or absence of ground marks). A ROI is constructed by using the parameters of the camera and the desired maximum range. In practice, it will mask unwanted elements like the aircraft (if the image is from the fin camera) or the sky.
- Construct it in order to facilitate broken high level requirements into lower level requirements. Construct a set of information that represents the object and its evolution (spatial and/or temporal). In our case, the HLR specifies the output: a set of points. Thus, we consider this set of points as the spatial propagation of points that describe a line. So, we have to detect a first point, define a dynamic law that describes the propagation of points, and precise an end criteria.
- Find and write most of the properties verified by the desired object in the image (size, structure, texture, etc.). These properties will help to verify that extracted information is correct. These properties could also help in writing an end criteria.

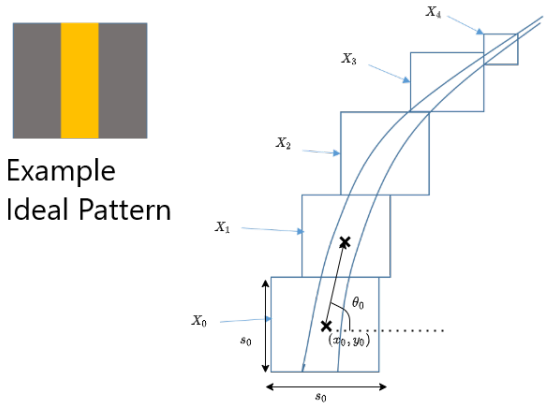

Figure 4: Model of a line.

Following these points, we model a line as a spatial repetition of a pattern (Figure 4). The pattern is considered known. The pattern will repeat himself in the presence of some noise (degradation, orientation, illumination variation, deformation, etc.). This repetition is done in respect to some properties provided by the International Civil Aviation Organization (ICAO) [17]: the curve made by central points of the patterns must satisfy constraints. In addition, depending on the illumination, the color of the painting on each pattern should be close to yellow [18]. The width of lines is known.

To describe a pattern in a thumbnail, we use a state vector $X_k = \{(x_k, y_k), \theta_k, s_k\}$. (x_k, y_k) is the position of the center of the thumbnail in the image, θ_k is the orientation of element in the thumbnail and s_k is the size of the thumbnail (considered as a square).
The evolution between state vectors is described by a function g, where g uses the current state vector X_k to update the next state vector X_{k+1}. The update depends also on external factor noted in an unknown noisy term U_k:

$$X_{k+1} = g(X_k, U_k) \tag{1}$$

This function is unknown and we consider a simplified model. We use the following equations to update the variables inside the dynamic model:

$$x_{k+1} = x_k + s_k cos(\theta_k) \quad (2a)$$
$$y_{k+1} = y_k + s_k sin(\theta_k) \quad (2b)$$
$$\theta_{k+1} = \theta_k \quad (2c)$$
$$s_{k+1} = min(max(s_k, a), b) \quad (2d)$$

Where the constants a and b determine the minimal and maximum size of thumbnails. It models the line's trajectory as a linear trajectory and supposes the two variables s_k and $theta_k$ are not evolving. This model is not quite precise. But the particle filter will correct the evolution of these parameters according to the observations from the image. In practice it is good enough to catch simple line patterns. In the future, we plan to add other variables to capture and describe environmental perturbations (such as the illumination, the principal color, etc.) and also enhance the dynamic model to improve performances.

3.2 General architecture

Using this model, we split our specification into three parts: find potential starting points of the line, recognize a pattern of the line from a thumbnail described by a state vector and predict the evolution of the pattern in the image. Each item has its own requirements driven from HLR and its own unit tests. The coverage of the HLR by LLR is ensured by the model. Figure 5 shows our design. Once the interface between modules is defined, it is important to note that each part can be developed independently.

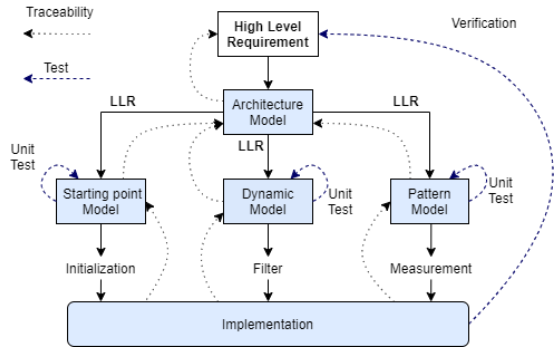

Figure 5: Design and test of our model.

Following the key points expressed above and using our model, we propose a modular architecture to detect lines from an image. The architecture of the system is described in Figure 6. The main point of our architecture is to separate prediction from a model to information extracted from the image (since the image can be considered as a complex noisy signal). The proposed algorithm works as follows:

1. The image is acquired by the camera and considered as a representative data of the reality.

2. Using the image, an initialization based on simple assumptions (such as gradient, color and position) will propose some initial conditions (potential starts of lines) as an initial state vector X_0.

3. Using the initial condition and the dynamic model of lines a filter will predict the next state of the line $\hat{X}_{k+1}$.

4. The prediction will be sent to an observation function that will attribute a measure corresponding to the probability of the correctness of the prediction $\hat{X}_{k+1}$ with regards to the image and modeled feature of the desired pattern.

5. Using the measures, the filter will update his predictive model: $\hat{X}_{k+1} \rightarrow X_{k+1}$.

6. Until an end criterion is verified, the program will repeat from step (3).

This algorithm detects one line. In practice, we applied this algorithm many times to detect many lines in the image. Some heuristics are used to discard some elements like the ridges of the lines (see Figure 3).

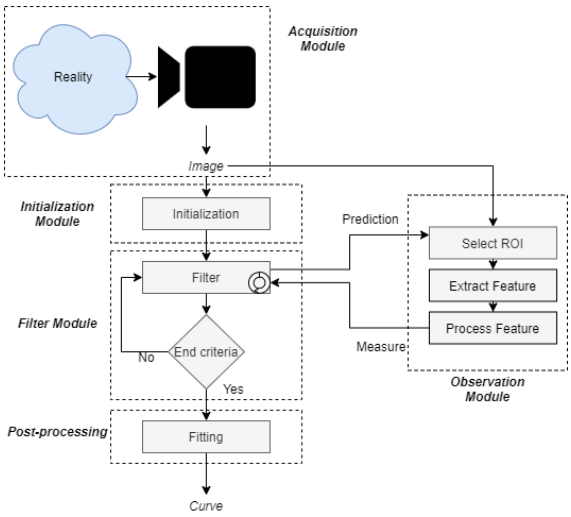

Figure 6: Simplified schema of our modular line detection architecture.

3.3 Description of modules

Now, we detail and explain our choice for each module.

Initialization module. The initialization function uses the image as input. It has to propose potential starting points of a ground line. To detect these starting points we developed an algorithm based on classical image processing, using gradient and color prior from the model of line. This algorithm is not detailed in this paper. The robustness of this function is low. However, in case of failure (wrong starting point), the proposed point might be discarded by the filtering and observation function. That is why we favor recall over to precision.

Filtering module. To predict the next state of the line, we have to estimate the solution from the equation 1. To solve this equation many filters can be used such as the Kalman filter. However, in practice, that model is limited due to the noisy term U_k. If the distribution of U_k was Gaussian, the Kalman filter would be a suitable solution. In our case, we choose to use a particle filtering algorithm similar to the one presented in [19]. Firstly because it can solve this problem without knowledge on the distribution of U_k. Secondly, to provide a first analysis on problems encountered by the certification process of particle filters. These filters are useful for navigating, positioning and tracking [20], they could be used in more applications in the future. Since our project is to put on the table new technologies, we made the choice to construct our line detector using a particle filter.

The particle filters are a set of Monte Carlo algorithms. The objective is to compute the posterior distribution of a stochastic process. The first step is to create particles $\{X_k^n\}_{n=1}^N$ from an initial prior distribution. Each particle corresponds to an estimation of the state vector. A weight w_k^n is associated with each particle. This weight corresponds to the confidence in a particle to represent the actual real state. The closer the particle is to the real state the higher its weight is. In most systems using particle filters the weight is computed by using a measure provided from external sensors (such as GPS or inertial sensor in the case motion estimation). In our case, we define a sub-item called "observation module" that attributes a weight to a particle using as input the image and the variables contained in the particle. This observation module is detailed below. Once all particles have an associated weight, the particle filter algorithm will re-sample particles using the distribution created by $\{(X_k^n, w_k^n)\}_{n=1}^N$. More information about particle filtering can be found in [21][22] and more about this particle filtering method are provided in a french paper [23].

The principal advantage of particle filter algorithms is their ability to solve nonlinear problems tainted by a noise without prior knowledge about this noise. The principal inconvenience of these algorithms is the computing time. The more a problem is complex, the more it will require particles to compute a good estimation. We don't cover computation time problematic in our approach. However, it could be possible to cover this issue by adapting the amount of particles depending on the confidence in the observation module [24].

Observation module. Using a state vector provided by the filter (through a particle), it selects a thumbnail in the image (between 15 and 51 pixels depending on the variable s_k in the state vector), resize the thumbnail to 33×33 pixels and attributes it a score. This score is between 0 and 1 and evaluates the pertinence of components in the state vector in regard to the thumbnail. The score is the product of the output of tree sub-items:

- The first one is a binary classifier. It scores the presence or absence of ground marks in the thumbnail. After a preliminary study [23], we selected a small convolutional neural network (CNN) as classifier. This choice is also motivated to provide a real application to use CNN in a restrained context controlled by other elements. By using a CNN at this part of the line detection method, it permits to achieve correct result in the line detection without using end-to-end CNN. To train this CNN we constructed a dataset of 4000 thumbnails with manual annotation (50% positive and 50% negative). The score used is directly the output of the classifier. More details about the creation of the CNN are provided in [23].

- The second item measures correlation between the orientation θ_k in the state vector and the principal orientation in the thumbnail. We compute the orientation of the element in the thumbnail using a method based

on the Gabor filter.

- The last item measures the centering of the line in the thumbnail using classical image processing techniques such as gradient, color processing, Otsu's binarization and Hough transformation. It extracts the principal straight yellow lines in the thumbnail. Then it computes a score (between 0 and 1) depending on the projection of the center of the thumbnail on the line equation.

The product of these scores produces the final score that will serve as the weight for the particle in the filter.
In the next section, we discuss the verification and validation of these modules.

4 Certification Arguments

This section presents our arguments and thoughts about the feasibility of the certification for this architecture. This architecture proposes to reduce the amount of tests required to verify the system by using unit tests on each item. The principal assumption is: "If each item works correctly the function works correctly". This assumption holds because high level requirements are covered by lower level requirements. Each item has a simple and comprehensive task. The next parts develop discussion and arguments about verifying whenever each item is doing its task correctly.
This modular architecture should make the certification easier. The more an architecture is decomposed on items the more we can hope to reduce the complexity of each item and facilitate the creation of tests to verify performance of each item. It also helps to implement monitoring systems. Each item can have its own monitoring system and improve confidence on its results. In addition, when a module is changed (or improved) the certification should be done for this module only. It can significantly reduce certification cost when updating a system.

4.1 Filtering module

From the point of view of certification, Monte Carlo methods are often used to test algorithms but are generally discarded in embedded avionic systems. This is partially due to the fact that these algorithms use random number generation and one requirement for certification is repeatability. However, it is possible to generate offline a set of distributions that verify desired prior distribution. In addition, some particle filter algorithms use fewer random steps. A survey of recent advances in particle filtering can be found in [25].

Errors in the particle filter could occur depending on three conditions. Firstly, if the dynamic model is incorrect and does not match the real trajectory of the line. Secondly, if the prior distribution is not realistic (*e.g.* if the distribution has a huge bias). Thirdly, if the measure provided by the observation module is wrong and does not bring information about the evolution of the dynamic system. Figure 7 shows these potential errors.

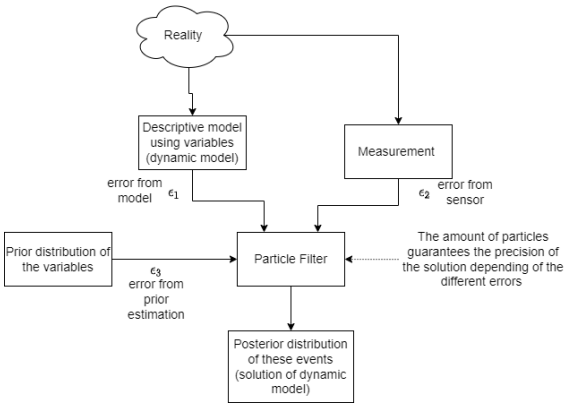

Figure 7: Potential errors of the particle filter

To test and validate performances of the filter, we build a generator of lines. An example of the generated line is shown in Figure 8. In this generator, it is possible to compute the perfect measure (it is computed by checking if the position of the particle is on the line or not and if the orientation of the particle is correct or not). We selected a prior distribution as a Gaussian distribution, where the variance is fixed in regards to the width of the line. These tests allow validating that within our dynamic model and prior distribution this method can follow line until a certain curve and depending of the amount of particles.
Furthermore, authors have been conducting research to prove the convergence of the particle filter [26] or computing bound of errors in particle filtering algorithms depending on incorrect model assumption [27]. Also, they have proposed solutions to enhance the algorithm in presence of noise and bias on the measure [28]. In regards to the state of the art, it should be possible in some situations to use formal proof to guarantee confidence of the particle filter. Otherwise, the feasibility of tests ensure the possibility to compute performance on such algorithms.

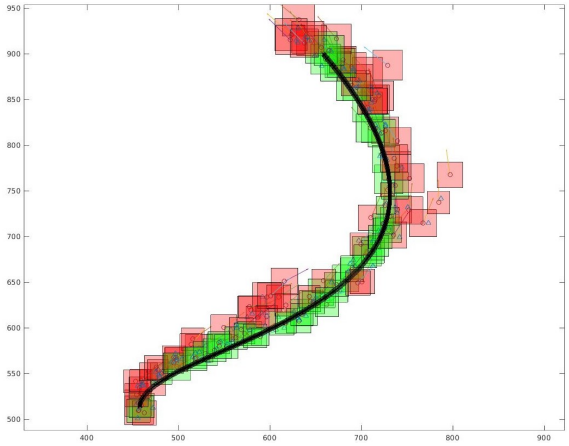

Figure 8: Example of test to validate the particle filter under the proposed dynamic model. In white the generated line. In green, particles with high weight (positioned on the line). In red particles with low weight (not on the line).

4.2 Observation module

The observation module obtains as input a state vector from the filtering module and an image from the camera acquisition. It has to judge the pertinence of the state describing a line pattern. The first step of this module is to create a thumbnail by selecting a specific region of interest in the image using information in the state vector. Then it attributes a measure by combining the result of three sub functions. The first one is a classifier that predicts the presence or absence of the desired pattern in the thumbnail. The second one is a function that estimates the principal orientation of elements in the thumbnail. And the last is a function that estimates if the element is centered in the thumbnail.

Classification. We provide here a minimal scope to use deep learning methods. Instead of using them to achieve complex tasks in high dimensional images, we reduce the problem to a simple binary classification of small thumbnails. In addition, the output of the classifier does not require a very high integrity. As said previously, the filtering module accepts a part of error from the measure. It is acceptable for the classifier to make some wrong prediction.

In this context, it is still not possible to test the classifier on every possible image ($256^{3\times33\times33} \approx 10^{7843}$). At the same time it should be possible to compute confidence bounds for the classifier by using methods proposed in the state of art. For example, the Pac-Bayes theory [29], adversarial method [11] or bounce generalization[30] seem promising.
This part is still ongoing and at the moment the only confidence in our classifier comes from a data test consisting of approximately 4000 thumbnails (50% positive and 50% negative). This dataset is constructed by manual annotation and we lack arguments in the confidence of this test. Figure 9 shows examples of thumbnails of the dataset. Our convolutional neural network achieved 88.3% of accuracy on training and 87.2% on the test data set.

Figure 9: First line shows thumbnails considered as positive. Second line shows thumbnails considered as negative.

Orientation and center measure. The orientation measure defines the general orientation of a line in the thumbnail. The orientation is found by filtering the thumbnail using the Gabor filter on every possible orientation. The maximum response of the filter appears when the direction of the line is in the same orientation that the filter. In the case of absence of line in the thumbnail, the response of the filter is not predictable. This function is checked from a dataset similar to the one built for the classifier. In addition, because of the pattern of line, we can ensure that the Gabor filter has one of the best response when the line is the principal element in the thumbnail.
Another measure consists in determining whenever the line is or not in the center of the thumbnail. This is done by geometric image processing without much difficulty. It is verified on the same data set as for orientation measure.

4.3 Initialization

The initialization extracts features that lines should verify: color, gradient and position assumptions. From camera parameters we defined the region of interests where the starting points of the lines can be. In each region, we use color processing to separate "yellow" elements from others. Then we use the morphological operator and skeleton algorithm to select potential starting points. Due to the use of color processing, environmental hazards impact the result. To im-

prove performance and robustness against illumination variation and other hazards, the defined color "yellow" is adapted in each region by using a method proposed in [18]. In case of a false starting point, other modules should detect and discard this point (absence of ground marks, length of line, etc.). It is still possible to miss a line because the minimal requirements are not fulfilled (illumination, camera resolution, etc.).

4.4 Verification

The final verification is done qualitatively on real images from different scenarios. Figure 10 shows results to illustrate the function. Points of the same color correspond to the result of the filter at each iteration. Red curves correspond to the quadratic regression of these points. Currently, the system is not designed to be robust to occlusion. As we can see, the initialization step captures only three lines. Lines far away from the camera are not detected because of this step. The architecture is designed to reduce false positives (erroneous detection).

Figure 10: Results with our architecture.

5 Conclusion and future works

This paper focuses on the design methodology of vision based algorithms. We proposed a modular architecture for line detection applications designed to ease the certification process. This proposition avoids the use of end to end deep learning methods. They seem to have better accuracy but they are not compliant with the actual certification process. The main interest of decomposing vision architecture into smaller parts is to facilitate the generation of tests. Each part of the architecture has its own specifications and its own test data set. In addition, it provides a minimal restrained context where complex algorithms, such as deep learning methods, could be studied. Also, it should be easier to update and improve components one by one.
To build this architecture, we propose a general method describing the spatial or temporal dynamic of the visual object and decompose the architecture into two parts. The first one will predict features that should describe the object. The second one will check in the image if this feature corresponds to reality. This architecture should work for object tracking.
We are currently working to build larger test data sets and a monitoring system using temporal information (video processing). The question about test data sets is still ongoing. The use of guidelines such as CV-HAZOP helps the identification and construction of potential hazardous scenarios. However, data from these scenarios are not easily acquired and we have to base tests from simulators. It raises the question about the certification of such simulators.

ACKNOWLEDGMENT

This work was supported by AIRBUS Operations. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views AIRBUS Operations. We would like to thank the OKTAL-SE company for providing the simulation tool, used to build synthetic images.

References

[1] EASA, *Certification Specifications and Acceptable Means of Compliance for Large Aeroplanes CS-25*. 2007.

[2] SAE., *ARP4754a - Aerospace Recommended Practices (Development of Civil Aircraft and Systems.)*. 2010.

[3] RTCA, *DO-178C, Software Considerations in Airborne Systems and Equipment Certification.* 2008.

[4] F. Boniol, A. Chan-Hon-Tong, A. Eudes, S. Herbin, G. Besnerais, C. Pagetti, and M. Sanfourche, "Challenges in the certification of computer vision-based systems," 09 2020.

[5] S. Bhattacharyya, D. Cofer, D. Musliner, J. Mueller, and E. Engstrom, "Certification considerations for adaptive systems," *2015 International Conference on Unmanned Aircraft Systems, ICUAS 2015*, 2015.

[6] EASA and Daedalean, "Concepts of design assurance for neural networks (codann) ii," tech. rep., 5 2021.

[7] EASA, "Easa concept paper: First usable guidance for level 1 machine learning applications," tech. rep., 2021.

[8] H. Delseny, C. Gabreau, and A. G. et al., "White paper machine learning in certified systems," *CoRR*, vol. abs/2103.10529, 2021.

[9] Z. Wang, W. Ren, and Q. Qiu, "Lanenet: Real-time lane detection networks for autonomous driving," 2018.

[10] E. Mirzaei, C. Thomas, and M. Conrad, *Safety Cases for Adaptive Systems of Systems: State of the Art and Current Challenges*, pp. 127–138. 09 2020.

[11] Y. Lin, H. Zhao, X. Ma, Y. Tu, and M. Wang, "Adversarial attacks in modulation recognition with convolutional neural networks," *IEEE Transactions on Reliability*, vol. 70, no. 1, pp. 389–401, 2021.

[12] C. Molnar, *Interpretable Machine Learning*. 2019.

[13] P. Germain, A. Lacasse, F. Laviolette, M. March, and J.-F. Roy, "Risk bounds for the majority vote: From a pac-bayesian analysis to a learning algorithm," *Journal of Machine Learning Research*, 2015.

[14] O. Zendel, M. Murschitz, M. Humenberger, and W. Herzner, "Cv-hazop: Introducing test data validation for computer vision," in *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015.

[15] O. Zendel, K. Honauer, M. Murschitz, M. Humenberger, and G. Fernandez Dominguez, "Analyzing computer vision data - the good, the bad and the ugly," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[16] O. Zendel, M. Murschitz, M. Humenberger, and W. Herzner, "How good is my test data? introducing safety analysis for computer vision," *International Journal of Computer Vision*, vol. 125, pp. 1–15, 12 2017.

[17] ICAO, *AERODROMES: aerodromes design and operations*. International Civil Aviation Organization (ICAO), 2018.

[18] C. Meymandi-Nejad., E. Perrotin., A. Herbulot., and M. Devy., "Colorimetric space study: Application for line detection on airport areas," in *VEHITS*, 2021.

[19] C. Meymandi-Nejad, S. E. Kaddaoui, M. Devy, and A. Herbulot, "Lane detection and scene interpretation by particle filter in airport areas," in *VISAPP*, 2019.

[20] F. Gustafsson, "Particle filter theory and practice with positioning applications," *IEEE Aerospace and Electronic Systems Magazine*, vol. 25, no. 7, pp. 53–82, 2010.

[21] B. Ristic, S. Arulampalam, and N. J. Gordon, "Beyond the kalman filter: Particle filters for tracking applications," 2004.

[22] A. Doucet and A. M. Johansen, "A tutorial on particle filtering and smoothing: Fifteen years later," 2008.

[23] E. Perrotin, C. Meymandi-Nejad, A. Herbulot, M. Devy, and F. Bousquet, "Détection des lignes aéroportuaires par méthode de filtrage particulaire: Évaluation de fonctions d'observations," in *ORASIS 2021*.

[24] X. Zhang, J. Peng, W. Yu, and K.-C. Lin, "Confidence-level-based new adaptive particle filter for nonlinear object tracking," *International Journal of Advanced Robotic Systems*, vol. 9, no. 5, p. 199, 2012.

[25] X. Wang, T. Li, S. Sun, and J. Corchado Rodríguez, "A survey of recent advances in particle filters and remaining challenges for multitarget tracking," *Sensors*, vol. 17, p. 2707, 11 2017.

[26] X.-L. Hu, T. B. Schon, and L. Ljung, "A general convergence result for particle filtering," *IEEE Transactions on Signal Processing*, vol. 59, no. 7, pp. 3424–3429, 2011.

[27] N. Vaswani, "Bound on errors in particle filtering with incorrect model assumptions and its implication for change detection," in *2004 IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2004.

[28] F. Abdallah, A. Gning, and P. Bonnifait, "Box particle filtering for nonlinear state estimation using interval analysis," *Automatica*, vol. 44, no. 3, pp. 807–815, 2008.

[29] K. Pitas, M. Davies, and P. Vandergheynst, "Pac-bayesian margin bounds for convolutional neural networks," 2018.

[30] P. M. Long and H. Sedghi, "Size-free generalization bounds for convolutional neural networks," *CoRR*, vol. abs/1905.12600, 2019.