Automatic landing on unprepared zone

Guillaume Anoufa[12], Christophe Garcia[1], Stefan Duffner[1], Nicolas Bélanger[2]

[1]LIRIS
[2]Airbus Helicopters

Abstract

Landing is arguably the most complex and most dangerous phase of flight for an aircraft. As a consequence, automatic landing on unprepared zone is a fundamental function and a key challenge for unmanned aerial vehicles. Many obstacles such as a pylon or any object on the helipad can hinder the descent. Cameras and lidars can be disturbed or deceived by a smoke cloud, low luminosity or obstacles with low reflectance. Therefore, using a combination of different types of sensors for the landing zone clearance diagnosis improves the robustness of the system.
Although trajectory generation and optimal control are already widely spread, fewer methods deal with Automatic Decision Making. At Airbus Helicopters we have designed a demonstrator to prove our ability to enable a drone to make its own decision. A drone equipped with an autopilot will be given an order to find the closest helipad and land on it only if it is safe. Otherwise, the drone will fly to the next helipad.

1 Introduction

Automatic landing implies very accurate detection of obstacles around the helipad. Best performing single-camera object detectors have all been based on deep learning for several years [19, 13]. Deep Convolutional Neural Networks (CNNs) [18] are a special type of Neural network that have shown impressive performances in several applications of computer vision such as segmentation, object detection, image classification etc...

Models that achieve top accuracy in competitions have deep architectures that incur high computation cost and require large amounts of memory. These state-of-the-art models are generally not suited for real time processing especially on an embedded device. However, as more and more practical use of deep learning models emerged in recent years, there is a growing interest in building architectures that are both accurate and computationally efficient.

After detailing the demonstration that was built at Airbus Helicopters we will highlight the limitations of a purely single camera based system for the landing zone clearance. The recent affordability of 3D sensors and the performances of 3D detectors now more accurate and fast than ever made adding a lidar an attractive choice for our system.

2 Demonstrator

To simulate the automation of an actual helicopter we equipped a 2 meters long helicopter drone (Figure 1) with a low-power calculator (Jetson Xavier Nx), a fixed camera (ATC-HZ5530W-LP, 1080p, 61.2°*36.1°), a solide-state lidar (Livox Avia, 70.4°*77.2°) a GPS and multiple IMUs. We integrated a 4g router to the aircraft enabling it to receive orders from anywhere in the world and transmit a high quality H265 encoded video with very low latency and without any range limit. The drone is able to automatically take-off and follow a route consisting of waypoints then land on a helipad. At any moment during the operation the pilot can take back control of the drone.

Seeing the drone's vision in real time is essential during the data gathering phase to control what data are recorded and during the test phase to ensure that the demonstration is carried out correctly.

Figure 1: Vario Helicopter drone used for the demonstration

As illustrated by figure 2, the drone will be given a pre-set route (in red). It will take-off and follow the route. At some point during its course we will send to the drone the emergency order to land on the nearest available helipad. Thanks to its GPS and a database of geo-referenced landing zones the drone will re-route itself to land on the nearest helipad.

GPS position of an helipad is often imprecise especially in the case of offshore platforms. A vision-based helipad detector algorithm was developed [22] as part of a previous project at Airbus. It is capable of detecting an helipad from a grazing angle of incidence ($> 7°$) from as far as a 1500m away with the camera used. This algorithm is triggered once the drone approaches the helipad's GPS position. If the helipad is not immediately detected the drone performs a search pattern to scout the area for an helipad. From the 2D helipad detections, the camera parameters and the drone GPS and IMU we can estimate the 3D position of the helipad. This estimated position becomes more and more precise as the drone gets closer to the helipad. If the helipad is detected by our vision-based detector the drone is re-routed towards the computed 3D position of the detected helipad. Once the drone is less than 100 meters away from the helipad the systems checks if the drone can land. An helipad clearance algorithm will look for obstacles around the landing zone. Thanks to the precise detection of the helipad, we can limit the search

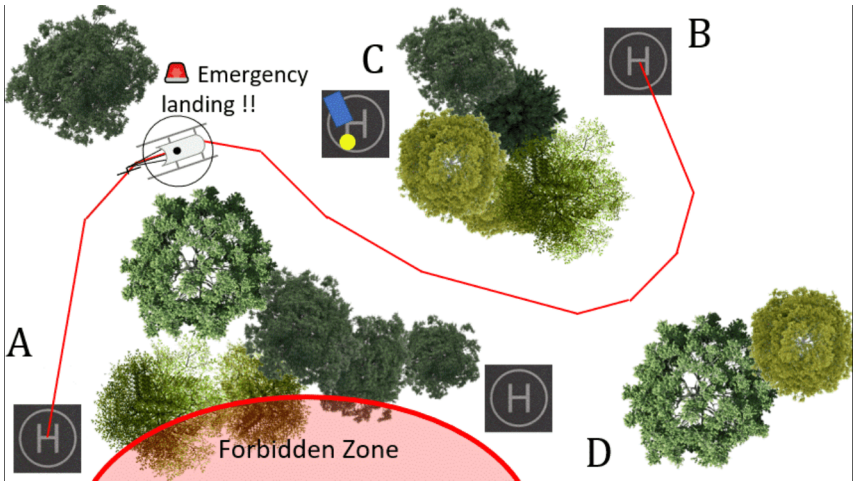

Figure 2: Course of the demonstration

for possible obstacles to the relevant area around the helipad in the image. Depending on the output the drone will try to land or change its mission to go to the closest helipad. In this paper we focus on the challenging phase of automatic landing and specifically on helipad clearance diagnosis.

3 Helipad clearance

3.1 Data acquisition

Most selected methods for helipad clearance are based on neural networks and therefore require large amounts of data representative of the use case. In most cases these data must be labelled by a human. Although these methods perform very well when representative data are available, in practice acquiring data is usually the hardest part of building a good object detector. For autonomous aircrafts, data are much more costly to obtain in contrast to autonomous driving.

3.1.1 Real data

In february 2022 we performed a flight test with the helicopter drone to gather data. Due to the complexity of having enough variability in the obstacles around the helipad during test flight we chose three obstacles of distinct shapes and sizes. These three obstacles, a small mast, a cardboard box and a van placed around the landing area3 will serve as indicator for the performances on real data of the evaluated methods. We recorded data from all the available sensors (camera, Lidar, GPS and IMU) during several approaches to the helipad from different angles. These data have only been used as a test set given that labelling 3D point clouds is a laborious task.

Figure 3: Flight data

3.1.2 Synthetic data

We are relying on computer generated data to train the models. Using a game engine we generate lidar point clouds along with camera images with the complex pattern of the Livox Avia. 3D models of various shapes are placed on or near an helipad. By changing the models placed around the helipad and the scene (terrain, lights, etc...) we created rich datasets representative of the many possible real life situations where the detector must identify obstacles.

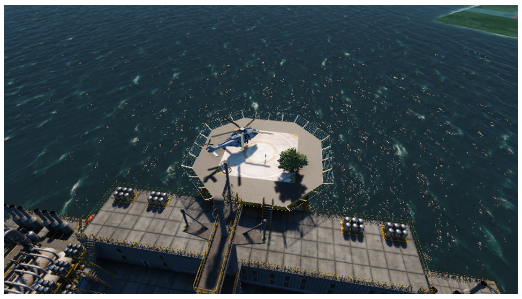

Figure 4: Synthetic Image from Unigine

3.2 Vision based approach

At first, we tried to tackle the detection problem with a single camera attached to the front of the aircraft slightly tilted towards the ground. There are several difficulties the detector needs to overcome for a successful helipad clearance:

1. Obstacles can have various shapes and colors making them difficult to be precisely detected in the image with common CNNs without a large amount of training data. However, there are few representative data available.

2. Every obstacle must be detected due to the criticality of the clearance.

3. The detector must run at least at 10fps on the selected embedded platform (Jetson NX Xavier).

Classifying detected objects is not an absolute necessity for our use case. The diversity of objects needing to be detected and the scarcity of training data led us to consider a more data-efficient approach. The detection problem is viewed as an **anomaly detection** problem where anomalies are any obstacles that could interfere with the landing. There are two main categories of anomaly detection models:

1. Reconstruction-based methods rely on the idea of training a model which is usually an AutoEncoder (AE) [6], VAE [17] or a GAN [16, 4] to reconstruct images. The model is trained only on images without anomalies. As images containing anomalies are different from the training data, their reconstruction diverges from the original image. Anomalies can thus be spotted by detecting the areas where the reconstruction is the most different from its original counterpart.

2. Distribution-based methods usually do not perform any training but instead use a pre-trained deep feature extractor on a huge database such as ImageNet[12]. These methods establish a distribution of meaningful features extracted from a training data base of images without any anomaly. At test time, features extracted from the tested image are compared with the previously calculated distribution. An anomaly score is computed based on the distance resulting from the comparison [28, 9, 11].

Methods from both categories are ranked in the top ten methods of MVTec Anomaly Detection leaderboard [5, 2] which is the baseline ranking for anomaly detectors. The main benefit of distribution-based methods is that very good results can be obtained with only a few dozen images. However, these methods suffer the most from domain shift as no learning is performed.

DFR [27] takes the best of both worlds and uses an AE to reconstruct extracted features from a pre-learned deep feature extractor and achieves very good results on MVTec. As we did not have any real data before building the platform, first results were obtained using simulated data only. We adapted the anomaly detection model based on the DFR architecture and obtained very satisfying results on synthetic images (see Fig. 5). Simulated data were generated with Unigine game engine as illustrated in Figure 4. Results obtained on synthetic images did not transpose well to real images. There is a so-called "reality gap" when testing to real images.

Several approaches exist to adapt the trained model to real images, e.g. CycleGAN [33]. We tried to use CycleGAN to transform synthetic images into more photorealistic looking images to reduce the reality gap. Training on transformed images did not improve the results when inferring on real images. However, tests on other similar applications suggest that training on synthetic data and fine-tuning on real data gives significant improvements but this is still on-going work and we need to collect more real images.

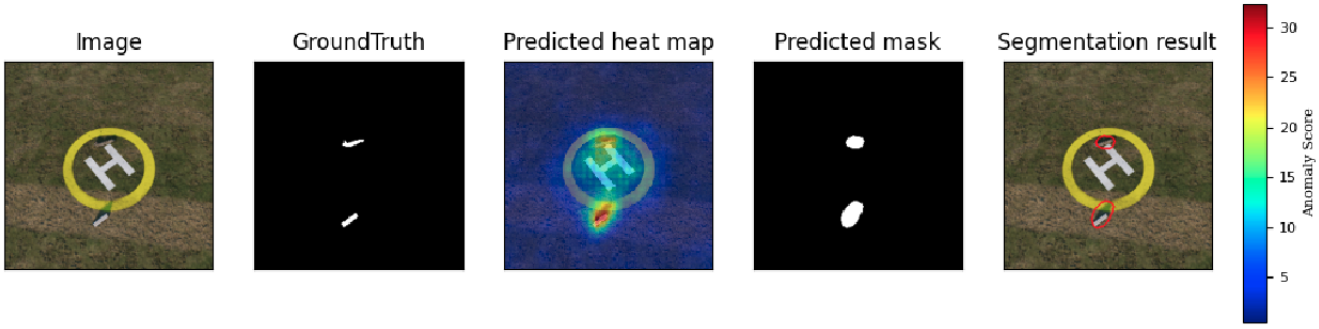

Figure 5: Anomaly detection on a synthetic image

3.3 Lidar

Although 2D vision-based methods can achieve decent results, the variability of possible real-life situations remains an issue. A camera only captures 2D light intensity and can therefore be easily deceived by a smoke cloud, a painted pattern or illumination.

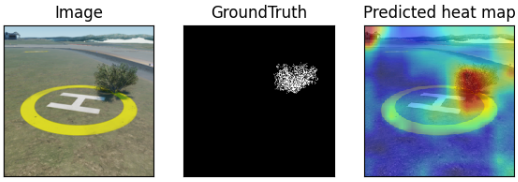

Figure 6: Background false detection on a synthetic image

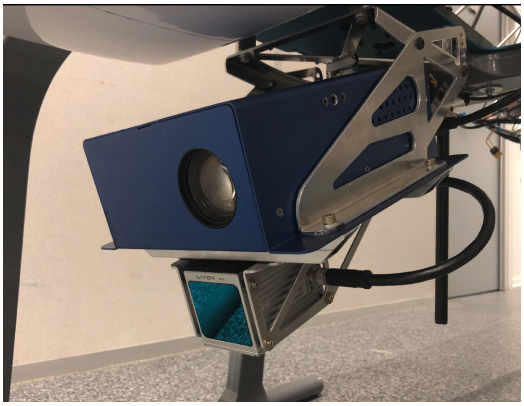

Figure 7: Embedded platform mounted on the drone

To disambiguate the presence of an obstacle we chose to add a 3D sensor. The high cost of lidars has largely prevented them from being used for autonomous driving and UAV until recently. However, lidar industry is rapidly growing and mass-produced low cost lidar sensors have become available. 3D detection using a lidar is nearly unaffected by variations in environmental conditions and is much more reliable. Usually a lidar outputs array of points containing 3D coordinates and reflectance value at a fixed frequency. A great benefit of lidar data is that it can be simulated fairly easily with only a small gap between simulated and real data. Every lidar scans the area with a specific pattern that needs to be accurately reproduced in the simulation. The main difficulty is to generate synthetic reflectance values which depend on a multitude of factors such as color, material or angle of reflection. We chose to use the average of R, G and B values which is a very rough estimate.

The Lidar used for the demonstration is the DJI Livox Avia lidar because its field of view is 70.4°*77.2°, it is lightweight, relatively precise and has a very long maximum detection range of 400 meters (Figure 7).

3.3.1 Calibration

To merge sensor data from lidar and camera, the field of view of the lidar should preferably be close to the field of view of the camera. Extrinsic and intrinsic calibration procedures are needed to interpret point clouds with images on the same reference. Intrinsic calibration of the camera was done by estimating the camera focal length and optical center using a given reference checkerboard in the image. Extrinsic calibration of the lidar and the camera was done thanks to the official implementation of ACSC [10] which an automatic calibration procedure for solid-

state lidars like the Livox Avia. This procedure requires to capture images and point clouds of a specific checkerboard from different views and distances. The checkerboard should be without edges and is therefore easily detectable in the image as well as in the point cloud. Extrinsic calibration must be redone each time the camera moves relative to the lidar.

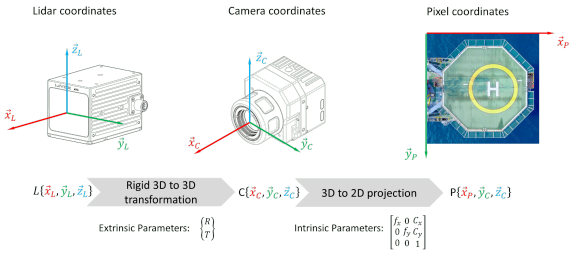

Figure 8: Point cloud to pixel projection

Figure 9: Picture of one of the scenes used for the calibration showing the point cloud projected on the image

4 Lidar based clearance

We considered different **lidar based approaches** for helipad clearance. Most evaluated methods are designed for ground vehicles where the sensors are placed at a fixed height and are parallel to the ground. Instead of adapting these methods for point clouds captured from aircrafts, the problem can be reduced to the usual case by transforming the point clouds. As a preprocessing the main ground plane is estimated thanks to a RANSAC[14] algorithm. The point cloud is then rotated and translated so that the majority of ground points are on the xy-plane.

4.1 Binary classifier

Instead of detecting the objects in 3D a simpler idea is to apply a 3D binary classifier on parts of the point cloud labelled as anomalies in the image. Each smaller point cloud is classified as either containing an object or not. This method has a low computational cost which is very valuable as several algorithms will be running on the same low power processor. However, this method assumes that the anomaly detector actually detects every obstacles and would only clear false positives. Given the inconstancy of anomaly detection this method is not satisfying and an independent method is preferable.

4.2 3D Object Segmentation

Semantic segmentation is not mandatory for our task as any obstacle of any type could hinder the approach. The idea of these methods is to identify clusters of points above ground forming objects.

A ground segmentation can be beneficial to reduce the number of points. If the terrain is bumpy a model like GndNet[21] can be trained to segment the ground. GndNet estimates the ground elevation by regressing a 2D grid of height. GndNet requires labelled ground truths in the form of grid height maps to be trained. We tried training a GndNet model using point clouds of randomly generated bumpy terrains with their associated ground truth height maps. Despite the satisfying results the extra computing time of 20 to 30ms is not negligible. We can reasonably assume that the ground around a helipad is plane. In that case a much faster and already computed during preprocessing RANSAC[14] algorithm is generally enough to identify the ground plane and segment ground points from the rest.

Once most ground points have been removed from the point cloud a **clustering algorithm** will exhibit objects from the remaining points.

The most straightforward methods use directly the point cloud in the 3D domain. For instance, Euclidean clustering uses Euclidean distance to justify if two points belong or not to the same objects. The shape and reflectance of ob-

jects and background, and the distance between the sensor and where the beam are reflected, affect the shape and sparsity of the point cloud. As a result it can be difficult or impossible to find the right euclidean distance threshold to isolate objects from the background. After several tests we concluded that this method is unreliable for our use case. Recent point cloud segmentation method Scan-line Run[30] take advantage of the very different circular pattern of rotating 360 Lidar and is not usable with the selected lidar.

Other methods compute a range image from the point cloud and perform segmentation on this range image [7], [20]. Testing this group of techniques is an ongoing work and could be promising because the range image has a stable high density while maintaining neighborhood information.

4.3 3D Object Detector

4.3.1 Sensor fusion for 3D object detection

Several public datasets containing both 2D and 3D data acquired from different sensors are used as benchmark for recent 3D object detectors. The most notable ones are NuScenes [8] and KITTI[15]. NuScenes and KITTI are large collections of scenes for autonomous driving made of data from cameras, lidars and radars sensors.

Intuitively there is two main categories of methods to leverage Mono Camera and Lidar fusion at signal level:

Projecting the point cloud to the image (RGBD images) methods makes it possible to use 2D image object detection models which are more advanced than 3D detectors while adding the depth information to the image. However point clouds give sparse information therefore a depth completion algorithm must be used to have an accurate depth information for each pixels. Generating a sharp depth map from point cloud data in real time is a tough task[3] that requires pixel-level depth ground truth which is hard to obtain.

Projecting the image to the point cloud (Colored point clouds) is the most common way as it maintains the valuable spatial information of the point cloud data at the cost of semantic information from the image because information is a lot denser on the image. Results show that adding color to the point clouds improves mAP by a few points on KITTI for object detection tasks.

State-of-the-art methods generally leverage sensor fusion to achieve the best results. Recent approaches are based on augmenting the point cloud with camera features while trying to preserve semantic information from the images. The top results on NuScenes combine a well performing 3D-object detector with a good fusion strategy. For instance CenterPoint-Fusion combines CenterPoint [29] with PointPainting [26] to merge the data from the lidar and the camera. PointPainting works by running a semantic segmentation model on the image to obtain pixelwise segmentation scores to decorate the point clouds. These type of approaches are computationally costly as they require to run multiple models on the data.

Using colored point clouds for real life applications is challenging for multiple reasons:

1. The delay between camera output and lidar output needs to be taken into account.
2. Calibration between the sensors must be near perfect and the sensors must remain fixed relative to each other.
3. The Livox Avia has an almost square field of view whereas the camera has a 16/9 field of view therefore a significant part of the point cloud can't be colored by the image.
4. Color is more sensible to domain shift making results on synthetic data less reliable and less transferable to real data.

4.3.2 Evaluated 3D detectors

The evaluation of an object detector is based on two metrics:

1. **Precision**: the ability of a model to identify only the relevant objects.
2. **Recall**: the ability of a model to identify all the relevant objects.

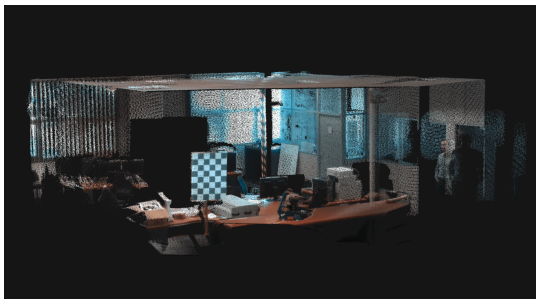

Figure 10: Coloured point cloud

Average precision (AP) is the most popular evaluation metric for 2D and 3D object detectors. AP is defined as the precision averaged across all values of recall between 0 and 1. The quality of a predicted bounding box is measured by its **intersection over union (IoU)** with the corresponding ground truth box (if it exists). Usually a threshold of 0.5 for IoU is chosen to discriminate between valid and bad detections. For our application the exactness of the predicted boxes are not too important as the boolean output of the algorithm should not be affected by slight changes in the locations and sizes of the detected objects. Therefore we chose to measure AP at 0.3 IoU (AP@.3).

3D object detectors are based on already proven 2D detectors. 3D object detection has seen tremendous progress in recent years thanks to lidars growing popularity for autonomous driving. Aerial scenes are usually sparser than driving scenes therefore easier to interpret. Objects stand out from the point cloud as cluster of points as long as the point clouds are dense enough. Intuitively, detecting objects of various shapes should be easier on 3D data than on images. Like 2D detectors they can be categorized as single stage detectors (SSD) or two-stage detectors. To reduce to computational load only the points in a relevant area around the helipad are processed.

Papers tend to show inference times on high end GPUs and we plan on running the models on a much less powerful GPU. The Livox Avia outputs around 24000 points every 100ms. Ideally we would want to stack points from two lidar outputs to form a lidar frame with enough points. Currently, even in stationary mode, the drone is not stable enough to stack multiple lidar outputs. The model should be able to run on every lidar frame therefore its inference time should be under 100ms. Using the average precision ranking on Kitti[1] we evaluated the performances of CIA-SSD[31], PV-RCNN[23], SE-SSD[32] and PV-RCNN++[24]. These models are at the state of the art in term of AP and should run fast enough for our application according to the authors.

PV-RCNN[23] and PV-RCNN++[24] are two-stage detectors ranked respectively 10th and 7th on the Kitti cars moderate ranking as of the 01/08/2022. The main difficulty for 3D detection is to handle the sparsity of point clouds compared to images. To this end most 3D detectors use voxelization to discretize the point cloud into a 3D grid of voxels. Voxelization, although way more efficient than using raw points, incurs a loss of spatial information. PV-RCNN uses both voxel-based efficient encoding and uses raw points (efficiently) to preserve spatial information making it both fast and accurate. PV-RCNN++ adds a much faster point feature encoding technique and a more efficient feature aggregation method. PV-RCNN++ achieves slightly better performances than PV-RCNN while being up to two times faster. CIA-SSD and SE-SSD are single-stage detectors that achieve similar results as PV-RCNN++. During our experiments we focused on PV-RCNN++ due to time constraints but further tests on single-stage detectors could be valuable because they are usually faster.

4.3.3 Experiments

Tests on a Jetson Xavier Nx confirmed that PV-RCNN++ is fast enough to handle a 10Hz lidar in real time. Implementations for our experiments are based on OpenPCDet[25], a codebase and open source project for Lidar 3D detection that provides optimized Cuda operators.

For our experiments, the models were trained only on computer generated lidar frames. Each lidar frame is an array of 30000 to 100000 points (3D coordinates + Reflectance). Two datasets of 1000 lidar frames of a helipad placed of several grounds and taken from different view angles were generated to train and test the model:

1. Set A contains only objects from classes

tree, vehicle (cars, trucks, vans) and pylon.

2. Set B contains more than 500 diverse objects such as a wheel, a construction cone, or a roll of hay.

Table 1 shows results for four different trainings. PV-RCNN++ is designed to recognize different classes of objects. Therefore it could work better when trained on specific classes of objects. As a **first test** we generated a dataset of frames containing three classes of objects: trees, vehicles (cars, trucks, vans) and pylons (set A). The model was then trained on a dataset where every object is labelled with its class.

The **second test** was to train the models without specifying the class of objects with the same dataset. Objects of the same class usually have reflectance and shape similarities. This test should theoretically be easier than the first one but separating objects by class could help the training even for the detection. Results showed that labelling objects with their class does not change the results.

For the **third test** the model was trained on a more diverse dataset (set B). This second dataset is the most representative of real data. Table 2 shows that using a more diverse dataset significantly improves results on real data.

As a **last test** we tried adding color to the point clouds on set B to simulate image projection on the point cloud. R, G and B values are used as features as well as 3D coordinates and reflectance. Using color as a feature improves AP@0.3 by almost 10 points on synthetic datasets. This suggests that adding color by projecting the camera image may be very valuable for the detection. However experiments on real frames showed a negative impact of the use of color. Most of the limitation of coloring point clouds enumerated in Sec4.3.1 can't be overcome easily. For the final demonstration we chose to keep camera and lidar usages separated.

Achieving higher average precision value is difficult due to the nature of the labelling. The labelling of our synthetic data uses the 3D bounding box of the 3D models used as obstacles. These bounding box are often imprecise and too large. With every training, PV-RCNN++ achieved near perfect recall for the synthetic datasets with the exception of a few objects that were mostly occluded.

Results on real data showed on Table 2 were computed on 20 different frames from test flight. Every frame contains all three obstacles, sometimes placed differently. A predicted box is considered valid for the evaluation as long as the box covers most of the obstacle and only the obstacle. Regardless of the method and the training set the van was always perfectly detected. Few lidar points intersected the mast making it fairly difficult to detect.

Lidar based 3D detector approach is independent on the anomalies detected on the image. It creates redundancy which adds a layer of safety to the diagnosis but requires trading off some accuracy for less complexity to ensure a reasonably low computing speed.

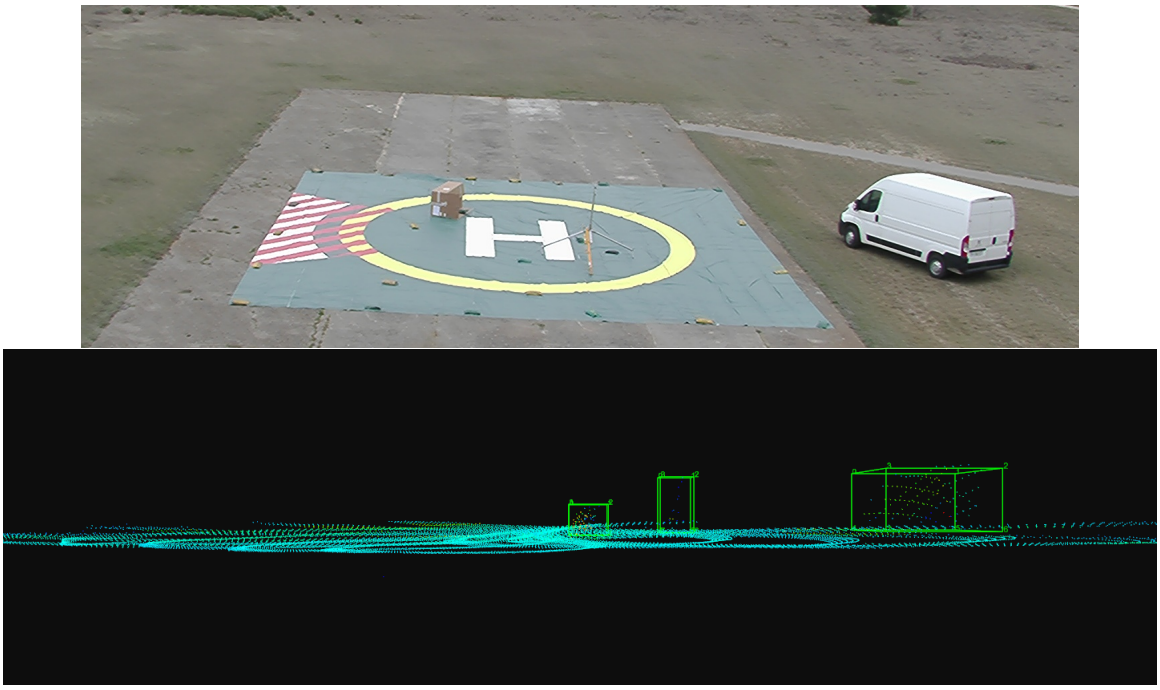

Figure 11: Detection on real point cloud from test flight. Obstacles from left to right: cardboard box, mast, van. Image from the camera for reference.

Methods	AP@.3	Recall@.3
Set A Labelled	53.21	99.4
Set A Classes Unlabelled	53.34	99.4
Set B	58.36	99.8
Set B + color	66.46	99.8

Table 1: Performance comparison for PV-RCNN++ on synthetic data.

Methods	Mast	Box	Van
Set A Labelled	2/20	8/20	20/20
Set A Classes Unlabelled	2/20	8/20	20/20
Set B	12/20	19/20	20/20
Set B + color	1/20	11/20	20/20

Table 2: Performance comparison for PV-RCNN++ on real data. Performances on real data are only indicative because we only have a few unlabelled real data.

5 Selected method(s) for landing zone clearance

Using 2D methods on a monocular camera is very challenging. Anomaly detection methods are sensitive to changes in the scene and training a standard 2D detector would require large amounts of labelled training data. Regarding lidar based methods state of the art 3D detectors stand out as surprisingly data efficient. 3D detectors can learn from even a small amount of computer generated data and seem to perform well on real data. PV-RCNN++ has only been tested on real data from a single test flight with only a few obstacles. Further tests must be conducted on real situations before drawing conclusions but this seems promising. Capturing more real data and labelling real 3D data could also further improve the model by using fine-tuning. The main drawback of 3D methods is the short

range lidars making the diagnosis only possible within 100meters of the target.

6 Conclusion

In this paper we present and assess methods to enable an unmanned aerial vehicle to autonomously establish a contingency plan to land safely. We focus on sensors, strategies and methods to accurately detect obstacles around an helicopter landing zone. We show that lidar is an efficient sensor for the object detection task. Several recent 3D detectors are very accurate and fast enough to run on a low power hardware that can be carried by even a small drone. It is possible to train these models with synthetic data only and still obtain very high performances on real data which alleviates the expensive cost of capturing substantial amounts of real data. The final demonstration is planned for Q4 of 2022.

References

[1] https://paperswithcode.com/sota/3d-object-detection-on-kitti-cars-moderate.

[2] https://paperswithcode.com/sota/anomaly-detection-on-mvtec-ad.

[3] https://paperswithcode.com/sota/depth-completion-on-kitti-depth-completion.

[4] Samet Akcay, Amir Atapour-Abarghouei, and Toby P. Breckon. Ganomaly: Semi-supervised anomaly detection via adversarial training, 2018.

[5] P. Bergmann, M. Fauser, D. Sattlegger, and C. Steger. Mvtec ad — a comprehensive real-world dataset for unsupervised anomaly detection. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9584–9592, 2019.

[6] Paul Bergmann, Sindy Löwe, Michael Fauser, David Sattlegger, and Carsten Steger. Improving unsupervised defect segmentation by applying structural similarity to autoencoders. *Proceedings of the 14th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*, 2019.

[7] Igor Bogoslavskyi and Cyrill Stachniss. Fast range image-based segmentation of sparse 3d laser scans for online operation. pages 163–169, 10 2016.

[8] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving, 2020.

[9] Niv Cohen and Yedid Hoshen. Sub-image anomaly detection with deep pyramid correspondences, 2021.

[10] Jiahe Cui, Jianwei Niu, Zhenchao Ouyang, Yunxiang He, and Dian Liu. Acsc: Automatic calibration for non-repetitive scanning solid-state lidar and camera systems, 2020.

[11] Thomas Defard, Aleksandr Setkov, Angelique Loesch, and Romaric Audigier. Padim: a patch distribution modeling framework for anomaly detection and localization, 2020.

[12] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[13] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman.

[14] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, jun 1981.

[15] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013.

[16] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

[17] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2014.

[18] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989.

[19] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context, 2015.

[20] Frank Moosmann, Oliver Pink, and Christoph Stiller. Segmentation of 3d lidar data in non-flat urban environments using a local convexity criterion. In *2009 IEEE Intelligent Vehicles Symposium*, pages 215–220, 2009.

[21] Anshul Paigwar, Özgür Erkent, David Sierra González, and Christian Laugier. Gndnet: Fast ground plane estimation and point cloud segmentation for autonomous vehicles. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020.

[22] Zoppitelli Pierre, Sébastien Mavromatis, Jean Sequeira, Guillaume ANOUFA, Nicolas Belanger, and François-Xavier FILLIAS. Embedding intelligent image processing algorithms : the new safety enhancer for helicopters missions. In *44th European Rotorcraft Forum - ERF 2018*, DELFT, Netherlands, September 2018.

[23] Shaoshuai Shi, Chaoxu Guo, Li Jiang, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. Pv-rcnn: Point-voxel feature set abstraction for 3d object detection, 2019.

[24] Shaoshuai Shi, Li Jiang, Jiajun Deng, Zhe Wang, Chaoxu Guo, Jianping Shi, Xiaogang Wang, and Hongsheng Li. Pv-rcnn++: Point-voxel feature set abstraction with local vector representation for 3d object detection, 2021.

[25] OpenPCDet Development Team. Openpcdet: An open-source toolbox for 3d object detection from point clouds. `https://github.com/open-mmlab/OpenPCDet`, 2020.

[26] Sourabh Vora, Alex H. Lang, Bassam Helou, and Oscar Beijbom. Pointpainting: Sequential fusion for 3d object detection, 2020.

[27] Jie Yang, Yong Shi, and Zhiquan Qi. Dfr: Deep feature reconstruction for unsupervised anomaly segmentation, 2020.

[28] Jihun Yi and Sungroh Yoon. Patch svdd: Patch-level svdd for anomaly detection and segmentation, 2020.

[29] Tianwei Yin, Xingyi Zhou, and Philipp Krähenbühl. Center-based 3d object detection and tracking, 2021.

[30] Dimitris Zermas, Izzat Izzat, and Nikolaos Papanikolopoulos. Fast segmentation of 3d point clouds: A paradigm on lidar data for autonomous vehicle applications. In *IEEE International Conference on Robotics and Automation*, 2017.

[31] Wu Zheng, Weiliang Tang, Sijin Chen, Li Jiang, and Chi-Wing Fu. Cia-ssd: Confident iou-aware single-stage object detector from point cloud, 2020.

[32] Wu Zheng, Weiliang Tang, Li Jiang, and Chi-Wing Fu. Se-ssd: Self-ensembling single-stage object detector from point cloud, 2021.

[33] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks, 2020.