Real-time visual pose estimation: from BOP objects to custom drone - A journey

Thomas Rey, Julien Moras, Alexandre Eudes, Antoine Manzanera

Mechatronics 109 (2025) 103339

Contents lists available at ScienceDirect

Mechatronics

journal homepage: www.elsevier.com/locate/mechatronics

Real-time visual pose estimation: from BOP objects to custom drone — A journey

Thomas Rey [a,*], Julien Moras [a], Alexandre Eudes [a], Antoine Manzanera [b]

[a] DTIS, ONERA, Université Paris-Saclay, 6 Chemin de la Vauve aux Granges, Palaiseau, 91120, France
[b] U2IS, ENSTA, Institut Polytechnique de Paris, 828 Boulevard des Maréchaux, Palaiseau, 91120, France

ARTICLE INFO

Keywords:
Drone swarms
Far range
Pose estimation
Monocular
Computer Vision

ABSTRACT

Pose estimation plays a crucial role in robotics for prehension tasks or in augmented-reality application, yet its application on real-world far-range estimation has not been thoroughly studied. This study aims to evaluate pose estimators on a custom drone at distances from 0.5 m to 10 m, which is beyond the scope of existing datasets, that only contain objects close to less than 2 m. We created synthetic and real databases specific to our drone and compared various RGB pose estimators, evaluating their performance across different distances. PViT-6D, being one of the SoTA methods on the classic [0,2] m interval, also outperforms others estimators at greater distances, and proves robust with respect to detection inaccuracy. The results demonstrate the potential of PViT-6D to be used on a real time application embedded in the drone platform. This work aims to evaluate the potential of pose estimators for mutual perception and communication within a drone swarm.

1. Introduction

The study of object pose estimation has made significant progress in recent years due to advances in artificial intelligence techniques and large datasets. Pose estimation, which involves predicting an object's 3D position and orientation relative to the camera, is a crucial component in various robotics applications, including drone navigation and swarm formation.

The BOP Challenge [1], driven by multiple contributors, has established a unified framework for benchmarking pose estimation algorithms. The use of large datasets has enabled the training of deep neural network models that demonstrate impressive results. However, traditional pose estimation datasets, such as YCBV [2] or LMO [3], have focused on everyday objects captured over limited ranges of distance and orientations, which is not representative of the complex dynamics involved in drone swarms. Indeed, most existing studies have used short-range scenarios (up to 2 m), whereas real-world drone applications often involve far-range interactions among multiple drones.

The proposed study aims to bridge this gap by evaluating the impact of visual pose estimation in the context of drone navigation using a complex-shaped object (a drone) as the target. By creating new datasets and comparing state-of-the-art pose estimator architectures, the research seeks to advance our understanding of how to accurately estimate the pose of drones in far-range scenarios, which is critical for the development of robust and efficient drone swarms.

This paper is organized as follows: The first step will be to present the state of the art in pose estimation models. We will then present our contributions, ranging from dataset creation to the comparison of state-of-the-art pose estimator architectures. Finally, we will present our results and extended ablation study. Our contributions are, in sum:

- The range limitation of traditional pose estimation datasets is evidenced.
- New, extensive long-range pose estimation datasets (up to 10 m) are created, utilizing our drone model and traditional objects.
- A comparative analysis of three pose estimators is performed.
- A real-time implementation of the chosen pose estimator from RGB input to pose prediction in ROS is presented.
- An in-depth ablation study examines the camera calibration's impact on pose estimator performance and its robustness to detection inaccuracy.

2. Related works

2.1. Object detection

The prediction of an object's 3D position and orientation from a single image is a complex task. Most current pose estimators use a

* Corresponding author.
E-mail addresses: thomas.rey@onera.fr (T. Rey), julien.moras@onera.fr (J. Moras), alexandre.eudes@onera.fr (A. Eudes), antoine.manzanera@ensta.fr (A. Manzanera).

https://doi.org/10.1016/j.mechatronics.2025.103339
Received 29 December 2024; Received in revised form 14 April 2025; Accepted 28 April 2025
Available online 17 May 2025

two-stage pipeline. As most of them are working within a region of interest (RoI) around the object, an object detector has to be used.

The object detection task has been one of the first tasks where neural network were used, and remains a very active research domain. The early methods [4] are based on hand-crafted features, usually very slow and poorly performing. Those approaches have been replaced by deep learning methods. The goal is twofold: localize and classify the object. The research in this domain splits in two: two-stages detectors and one-stage detectors. The former separate the localization from the classification whereas the latter use one single neural branch for both. At the beginning, the two-stages detectors [5,6] were well performing as each subtask was done with an expert neural network. The one-stage detectors [7,8] were not as effective but were faster.

A well-known architecture is the You Only Look Once (Yolo) [9], which is a one-stage detector. By doing both the localization and classification of the object at once, those architectures are designed to work on low-capacity processors, such as those equipped in drones. Despite extensive research in object detection, many challenges remain. As noted by [9], object detectors often struggle to recognize objects that are farther away than those encountered during training, a limitation that becomes apparent when these detectors are used as off-the-shelf solutions. The object detection being the first step of the pose estimation pipeline, the pose estimator uses this detected RoI to estimate the 6D pose of the object.

2.2. Pose estimator

Pose estimators can be classified into four classes. The correspondence methods are the oldest ones; they try to find the 2D position of at least 4 known 3D points of the object. Using more pairs of points, the Perspective-n-Point (PnP) – used to find the 6D pose – is more robust to occlusion. One of the first deep learning correspondence methods is [10], which tries, using a VGG [11], to predict the 2D position of the 8 corners of the 3D bounding box. Symmetries of the object are a big challenge for this method. Then, to simplify the problem, the authors train the model on a limited range of rotation for symmetrical object. So as to predict every possible rotation, a separate classifier is then used to classify the range of rotation. This technique refines predictions using object-specific information combined with depth data, resulting in improved accuracy. However, the substantial increase in computational demand precludes its use in real-time scenarios. [10] is a sparse correspondence method; however the literature shows that dense methods such as [12,13], tend to have better performance and be more robust to occlusion. The SoTA technique ZebraPose, proposed in [13], utilizes simple-to-learn descriptors. Specifically, it generates a 16-bit code for each vertex of the CAD model through an iterative process that involves recursively dividing each section into two parts, with a 2-means clustering method for each iteration. So, in this descriptor, a less-significant-bit represents a finer part of the object surface. The neural network, a ResNet-32 [14], is trained to predict this 16-bit code of each part of the object. Specifically, each one of the last 16 layers of the neural network predicts one bit of the code. The loss function calculated on these descriptors then allows a coarse-to-fine estimation, which helps the training process. By matching all the 16-bit codes with the 3D positions, the PnP can be used to retrieve the 6D pose. As the PnP is a slow process, these methods still fall short in terms of speed.

Currently, the fastest paradigm is the regression one, as these methods directly output the 6D pose. Moreover, loss functions can be calculated in the pose space, whereas correspondence methods are limited to the descriptors they used. One of the first regression methods is [15], which still considers the rotation estimation as a classification problem. This loss of continuity explains why, at the beginning, correspondence-based methods were more efficient. However, with the rapid evolution of neural network architectures, the rotation estimation started to be considered as a regression problem. GDR-Net [16] use a mixed model with a backbone to create correspondence-like features – a correspondence between 2D and 3D points, a visible mask of the object and a surface region attention map – and a neural network which plays the role of a differentiable PnP. The visible mask is used to isolate distinctive object features, while the 2D-3D correspondence map provides critical primitive information utilized by the differentiable PnP, and surface region attention maps resolve symmetries ambiguities. By integrating a differentiable PnP with correspondence and regression methods, an end-to-end training framework is established. This enables the optimization of loss functions in both image space and 3D space, optimizing correspondence features from the 6D pose. The SoTA regression architectures are still based on this idea. One of the variants of this mixed model architecture is GDRNPP, which improves performance through the employment of a better encoder and optimized training hyper-parameter settings. One of the first Transformer [17] based methods is the PViT-6D [18], which tries to use this well-performing architecture for a new task: pose estimation. The literature [19,20] highlights the importance of separating rotation and translation prediction for accurate 6D pose regression. Stapf et al. [18] address this challenge by introducing learnable tokens that selectively capture features required for each prediction. The Transformer architectures produces features based on these tokens, which are used by two separate convolutional neural networks to estimate the full 6D pose. This method is able to run at about 100 Hz on consumer hardware.

NeRF [21] is a neural network capable of generating new images from a learned scene and camera pose. [22] had the idea of inverting that concept to estimate the pose of the object. The Render-and-Compare paradigm is an iterative process, where the pose is refined at each iteration using a photogrammetric loss. However, NeRF are very slow to render a single image, so there embedded usage on a drone is still difficult. Some works, [23,24] tried to use the coarse output and not the refined output, others tried to use Gaussian Splatting [25], as [26], an architecture that is way faster to render an image, as it uses 3D Gaussian distributions to represent the scene.

Finally, recent works [27] try to use diffusion models to estimate the pose. The idea is close to the correspondence paradigm as it tries to denoise the 2D position of 3D known points and then uses the PnP. However, as this concept is insufficiently explored, it is distinguished from traditional correspondence methods.

2.3. Pose parametrization for regression models

In current regression architectures, pose estimators work within the RoI, so an essential step involves parametrizing the 6D pose within this region. Early works [8,10] show that the estimation of the rotation is very complex. So as to simplify this problem, the authors see the rotation estimation as a classification task. This representation is not continuous and has a limited precision. [19,20] worked on finding the best property of the mapping function from a Euclidean $\mathbf{R}^n$ space of the model's output to the special orthogonal group SO(3), where rotations lie.

The usual rotation parametrization, e.g. Euler angle or even quaternions, have an ambiguity like any parametrization with less than 5 parameters. [20] creates the parametrization that is used by a lot of new architectures, a parametrization with 6 dimensions. The model has to predict two of the three columns of the rotation matrix, the third one is retrieved with Gram–Schmidt orthonormalization. This parametrization respects all the properties from [19] and is also very easy to implement. Moreover, so as to have a representation independent of the object translation, the authors suggest to use an allocentric representation.

As the pose estimator works within the RoI, its goal is to refine the information given by the RoI's center (o_x, o_y) and its size (s) in pixels. Most of the approaches use a scale-invariant translation parametrization (SITP) created by [28]. This parametrization – using the object centroid (c_x, c_y) in pixels, the ratio between the images's

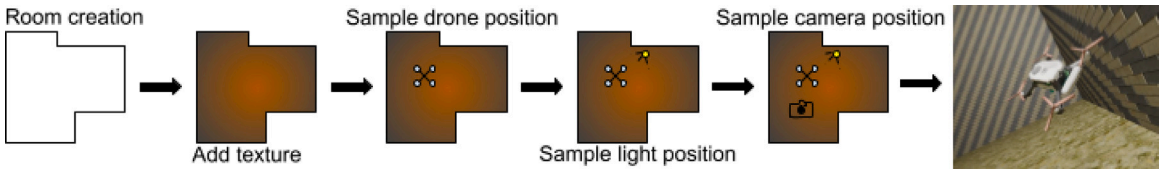


Fig. 1. Creation pipeline of our synthetic datasets.

size and the RoI's size r and t_z the distance between the object and the camera in meters – aims to predict the offsets $[\gamma_x, \gamma_y, \gamma_z]^T$ given by:

$$\begin{cases} \gamma_x = (c_x - o_x)/s \\ \gamma_y = (c_y - o_y)/s \\ \gamma_z = t_z/r \end{cases} \quad (1)$$

However, [18] shows that this parametrization does not enable the model to distinguish between a small object close to the camera and a large object farther away. The authors suggest a size invariant Z parametrization (SIZP), where r is not anymore the ratio between the images's size and the RoI's size but is equal to :

$$r = f \cdot s_{obj}/s \quad (2)$$

where f the focal distance, s the RoI's size and s_{obj} the object's size, that is computed as the diagonal of the 3D box of the CAD model. This new parametrization stabilizes the training, as all the offset values have the same order of magnitude. Finally, the translation between the object and the camera is calculated using the inverse projection model of the camera.

3. Contributions

3.1. Acquisition of data

3.1.1. Synthetic data

One of the most important aspects of neural network is the database that is used to train the model. A representative dataset is important so as to have a performing neural network. In our case, where we want to estimate the pose of our specific drone with relative distance up to 10 m, there is no existent database. The BOP Challenge [29], gives access to more than 12 different datasets. Nevertheless, those are usually toy datasets, using everyday life objects such as cans or plastic toys. The setup of the camera is usually the same, where all objects are not farther than 2 m from the camera, and for some datasets, such as YCBV and LMO, with a small to moderate positive elevation angle, these distributions can be seen on the Fig. 2(a). As no BOP dataset fulfills our case study, we had to create our own. We use BlenderProc [30] – a method to create Physically-Based Rendered (PBR) synthetic data using the open source 3D creation suite: Blender – to create multiple synthetic photorealistic datasets.

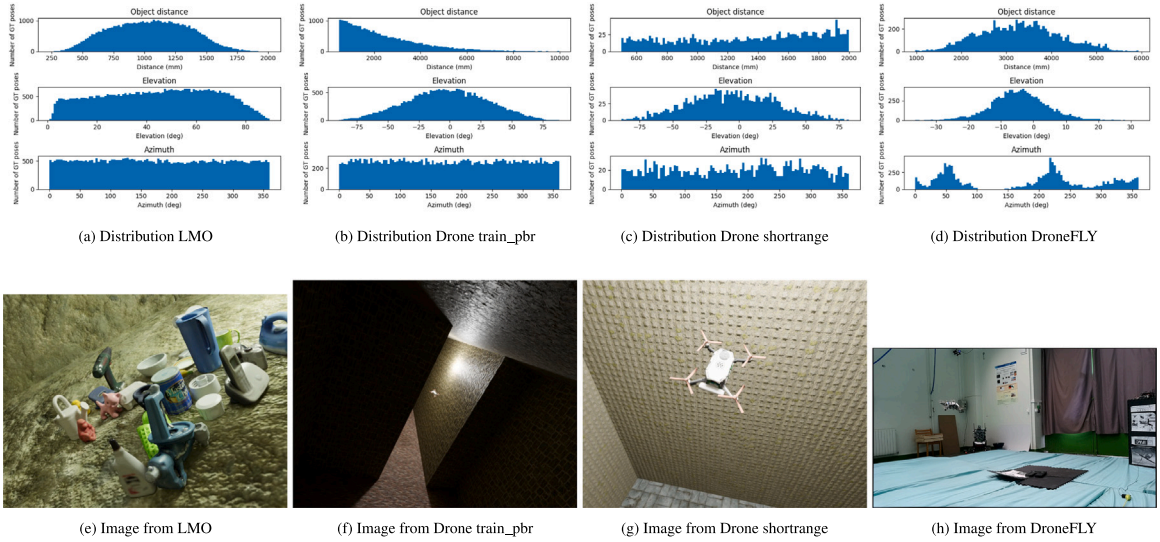
We created all synthetic images following a specific pipeline, shown on Fig. 1. With BlenderProc, we can create a non-rectangular room with different textures on the roof, walls and floor. A random tunnel-like texture is sampled so as to reflect our case study. Five drone positions are sampled in the room. For each drone position, spherical coordinates for 1 light position and 5 camera positions are sampled, where the azimuth and elevation angles and the minimum and maximum radius can be parametrized. All the intrinsic camera parameters are also parametrized. At the end, a dataset complying with BOP format is created, with an RGB image, a depth map, a mask for each class, the 2D detection and the 6D pose ground truths. To ensure that every image of our dataset is useful, we have multiple sanity checks. The first one checks if the object is visible from the camera, as the room can have positive or negative extrusion. To do so, we use Ray-tracing on the 8 corners of the 3D bounding box, and require more than 4 of them to be visible to consider the drone as visible. A check is done on every position sampled so as to ensure that each of them are in the room, an in-polygon function and a height interval are used. The key characteristics of the datasets used are summarized in Table 1, which includes relevant details such as image quantity, resolution, and focal length. Fig. 2 provides an example of images taken from these datasets, along with a visual representation of their distance and angular distributions.

Table 1
Overview of all the datasets used in this study; for LMO and YCBV the number of images is measured for a specific object only on the synthetic data.

Dataset name	type	nb images	resolution	focal (pixels)
LMO	synthetic	43k	640 × 480	573
LMO_longrange	synthetic	25k	640 × 480	573
YCBV	synthetic	38k	640 × 480	1067
train_pbr	synthetic	28k	640 × 512	370
test	synthetic	7k	640 × 512	370
train_pbr+	synthetic	43k	640 × 512	370
test+	synthetic	5k	640 × 512	370
drone_shortrange	synthetic	18k	640 × 512	370
Background	synthetic	1000	640 × 512	370
droneGRDcam1	real	2150	640 × 512	370
droneGRDcam2	real	1766	1280 × 720	1034
droneFLYcam1	real	4376	640 × 512	370
droneFLYcam2	real	4077	1280 × 720	1034
CAMBase	real	7k	640 × 512	370
CAMOak	real	7k	768 × 480	457

Our efforts to ensure representative samples ultimately led us to confront the difficulty of obtaining far-sampled data. As a result, we observed an unbalanced distance distribution in Fig. 2(b). To address this issue, we developed a novel approach to sampling distances, combining uniform volume sampling (increasing sample density at greater distances) with uniform distance sampling at different distance intervals. By implementing this new method, we created two datasets: train_pbr+ and test+. Notably, train_pbr+ exhibits an almost uniform distance distribution in the interval [2, 10] m, with still a bias for the [0,2] m distance interval. For the test+ dataset, the distance distribution has been corrected but faces still a bias for the [0,2] m and [4,6] m distance intervals. Furthermore, to create the test+ dataset, we modified the textures used for the floor, ceiling, and walls to introduce unseen patterns. This change aims to further challenge the pose estimation models and provide a more realistic testing scenario.

Finally, to underscore the intricacies of long-range pose estimation, we have adapted our creation pipeline to generate a short-range version of our drone dataset, which aligns with the traditional BOP datasets' configuration. This short-range drone dataset operates within a [0,2]m interval. The drone_shortrange dataset preserves all parameters from train_pbr, except for one critical modification: the maximum radius used to sample spherical coordinates around the drone position has been set at 2 m. This dataset is divided into a training set and test set. On the Fig. 2(c), we can see a nearly uniform distance distribution of this short-range dataset. This achievement represents a significant milestone in the creation of a dataset with such property, validating our efforts in dataset creation. To ensure a thorough comparison with traditional datasets, we created a long-range version of the LMO for a specific object, the toy ape. We built upon the procedure from the LMO train_pbr presented in [30,31], but modified it to eliminate occlusions and focus on a single object. This adaptation allows us to compare our long-range data with traditional datasets more effectively. In fact, the



(a) Distribution LMO (b) Distribution Drone train_pbr (c) Distribution Drone shortrange (d) Distribution DroneFLY

(e) Image from LMO (f) Image from Drone train_pbr (g) Image from Drone shortrange (h) Image from DroneFLY

Fig. 2. Distributions and example images from some datasets used in this paper.

complexity of our dataset lies in the extented range setup, which we try to better understand in this study.

3.1.2. Real data

Even if our synthetic data are photorealistic, real images are always more cluttered, with more occlusions and more natural lighting conditions. Multiple real datasets were created either by holding the drone (GRD) or by manually piloting it (FLY). The scene was captured by two cameras, modeled as pinhole cameras once the distortion is corrected with a radial-tangential distortion model. ROS is used as an acquisition system to produce a rosbag with all the mandatory topics to create a BOP-like dataset, containing the ground truth 6D pose and the ground truth bounding boxes. The ground truth of the 6D pose is built for each dataset from the pose – measured by a mocap system – of the drone and the observing camera. Some additional calibrations are needed to allow to project the CAD of the drone in the acquired image. The camera calibration (intrinsic and extrinsic w.r.t. the motion capture reference frame) is recovered through an automatic procedure from kalibr [32] and the CAD origin of the drone in mocap marker frame is manually estimated. The final measured pose was checked by visually assessing the CAD reprojection in images. To calculate the 2D bounding box ground truth, we use a similar way as BOP, which projects all the visible CAD points on the image and calculates the bounding box, but as we do not create a depth map, no visibility score can be calculated. On the Table 1, an overview of all the different datasets created can be seen.

3.1.3. Object detector dataset

The synthetic pipeline of the pose estimator dataset is not really adapted to train an object detector, since only one centered drone was visible on every image with quite unified background. We created a specific object detection dataset where 5 drones are placed at every 2 m interval from the camera. A real image is alpha-blended with the 5 drones image, to have the maximum variety in the background. As the state-of-the-art in object detection grows at a very fast pace, we choose to use a renown method, Yolov8. We fine-tuned multiple models on either the full synthetic data or on real datasets or on a mix of both. The output of this object detector will be used to test the full pipeline of our drone pose estimator, from the full RGB image to the RoI and then to the 6D pose.

Table 2
Important characteristics of SoTA methods.

	Type of method	Nbparam	Frequency(Hz)
ZebraPose	Correspondence	28.9M	4 Hz
PViT-6D	Regression	31.8M	50 Hz
GDR-NPP	Regression	28.8M	30 Hz
iNeRF	Render and Compare	X	0.1 Hz
6D-Diff	Diffusion	X	5 Hz

3.2. Comparison of the three different methods

Among methods presented in Section 2.2, we selected for this study the ones that are able to run on embedded system close to real-time. Only two paradigms are available: regression and correspondence. Render-and-Compare are far too slow and high memory demanding to be used directly on the drone. Moreover, the diffusion-based methods are still at their early stages and the code is not available. We chose to pick state-of-the-art methods in both remaining paradigms: ZebraPose [13] for correspondence, PViT-6D [18] for regression and GDR-NPP [33] as it is one of the foundation methods on which many leaderboard methods are based. On the Table 2, important information about different approaches can be seen.

3.3. Metrics

The BOP Challenge introduced a standardized test pipeline with usual metrics from robotic grasp tasks, based on distance between surfaces. The Average Distance of Model Points (ADD), the most used in the literature, is the proportion of 3 000 sampled CAD points with a Euclidean distance lower than 10% of the diameter of the object. However, in the context of robotic applications, the physical error is more adapted to qualify the ability to be used by downstream applications, so we will be using a classic metric Π_{th}, based on a threshold th on the Euclidean distance and on the geodesic distance, which gives the minimum angle between two rotation matrices. Three different thresholds are used: 1, 2 and 5 cm and 1, 2 and 5° respectively.

$$\Pi_{th} = \frac{\#\{\text{Predictions with error } \leq th\}}{\#\{\text{Predictions}\}} \tag{3}$$

3.4. Full pipeline

We implement this architecture into a real-time pipeline using ROS middleware composed of two nodes. The first one in charge of the

Table 3
Mean rate, latency and loads when the full pipeline is used.

	rate (Hz)	latency (ms)	CPU load (%)	GPU load (%)	RAM load (%)
bbox	33.1	20	46.72	30.06	39.32
transformation	32.8	40			

object detection, and the second of the pose estimation using the RoI and the RGB image. The first node publishes the detected bounding box (bbox) around the object, which is used by the second node to publish a transformation so as to track the position of the drone relative to the camera. Moreover, this implementation enables us to measure – on a similar hardware as the drone – the rate and latency at which this transformation and the bbox are published. Each node uses OnnxRuntime as inference engine, with both model converted to ONNX format from pytorch. We setup OnnxRuntime to run on Cuda EndPoint. We tested our pipeline on a rosbag, recorded during our campaign of real data's acquisition, played at real time rate. The system used is composed of an Intel(R) Xeon(R) W-2123 CPU @ 3.60 GHz with 16Go of DDR4 and a NVIDIA GeForce GTX 1080 Ti. The mean rate, latency and the CPU and GPU charges are shown on the Table 3, highlighting a good performance achieved on this hardware.

4. Results

4.1. Object detection

At first, we fine-tuned a yolov8n on our train-pbr+ dataset for 100 epochs with a learning rate of 10^{-2} and a mini-batch size of 16; we also froze the first 10 layers of the network, this setup of hyperparameters will be the same for all the next fine-tuning. However, as this pose estimation dataset was not intended to train an object detector, every drone being in the center of the image, a specific dataset, named Background, was created. This dataset has more diverse backgrounds and contains 5 drones on each image. Table 4 shows the performance of multiple object detectors on various real data, in which the drone is flying. The performance of the model trained on the background dataset is not significantly different from the train-pbr+ one. So as to have a performing object detector, we decided to fine-tune on real data directly and on a mix of real and synthetic data, using all the GRD data to train and all the FLY data to test. Fine-tuning the model using a mix of both real and synthetic data, as shown in Table 4, results in improved precision without affecting other performance metrics.

The performance remains low due to frequent scale changes for the object of interest. As we can see on Fig. 3, the predicted RoI scales vary a lot, deviating by an average of 5.2% from the ideal scale, and yielding a mean normalized center shift error of 0.113.

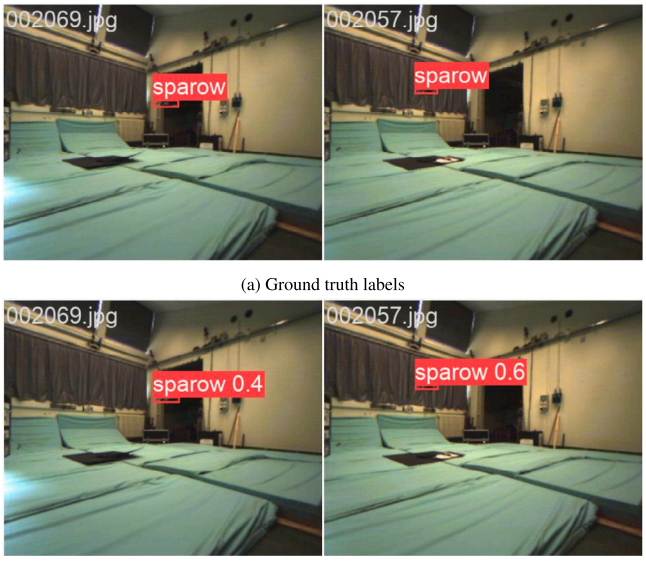
Table 4
Test on the concatenation of droneFLYcam01 and droneFLYcam02 benchmark of a Yolov8n trained on different datasets.

Train dataset	train-pbr+	Background	GRD	MIX
Precision	0.85	0.52	0.75	**0.87**
Recall	0.36	0.40	**0.54**	0.52
mAP50	0.43	0.4	**0.60**	0.59
mAP50–95	0.15	0.13	0.17	**0.19**

4.2. Pose estimator

4.2.1. Experiments setup and implementation details

The hyperparameters used during training are presented in Table 5. Unless specified otherwise, we employed the same parameters, learning rates and schedulers as those described in the original papers. The "scale bbox" column shows the percentage of dynamic zoom-in augmentation, inspired by [34]. This technique applies a shift and scale



(a) Ground truth labels

(b) Predictions

Fig. 3. Label and predictions of our drone, 'sparow' seen in flight by the yolov8n trained on mixed data.

transformation to the RoI input into the pose estimator, enhancing its robustness to detection errors and reducing dependence on the detector used.

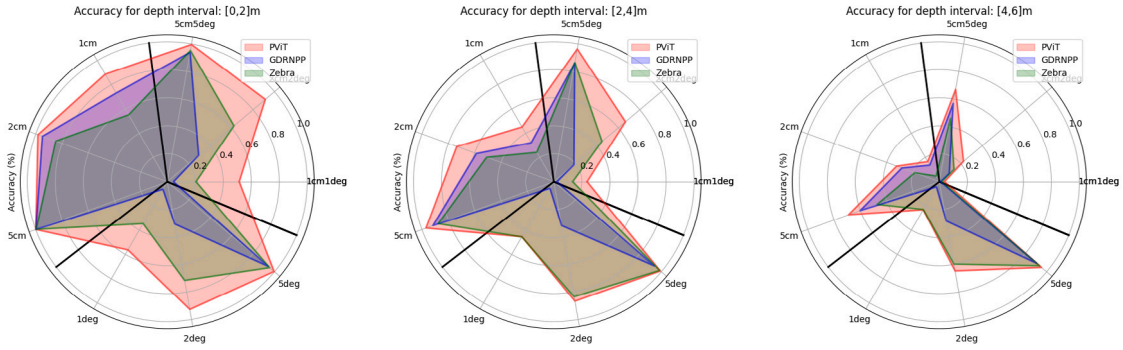
Table 5
Comparison of all the hyperparameters used to train the three pose estimators. The ZebraPose counts in iterations, so the epoch is the ratio seen from the size of our synthetic dataset PBR. The "metric used" column is to specify on which criterion the best model on the validation set has been chosen.

	nb epoch	batch size	metric used	scale bbox	optimizer
PViT-6D	150	30	ADD	20%	Adam
ZebraPose	~400	30	ADD	25%	Adam
GDR-NPP	150	30	ADD	25%	Ranger

When evaluating three different methods, it becomes apparent that the loss functions employed vary significantly. For correspondence-based methods such as ZebraPose and in a way GDR-NPP, the loss function is computed directly from the descriptors used: a 16-bit code for ZebraPose and a 2D-3D correspondence map and surface region attention map for GDR-NPP. In contrast, directly calculating the loss function on the 6D pose for ZebraPose would significantly impede training speed. The computation of the 6D pose requires the use of the PnP algorithm, which is a computationally intensive process that slows down the training. This is why the 6D pose is only computed during validation. On the other hand, regression-based methods like PViT-6D and GDR-NPP use a loss function calculated on the 6D pose directly, without affecting the training pace.

4.2.2. Three methods on classic BOP datasets

On Table 6, we compare the performance of three studied methods on two classic BOP datasets: LMO and YCBV. The discrepancy between our results and those reported in the article can be attributed to several factors. Firstly, some articles employ time-consuming techniques that iteratively refine the initial pose estimate, leveraging additional information such as depth. In contrast, we chose not to use refinement techniques due to our real-time goal. Secondly, some methods do not disclose all the hyperparameters used to train their models, which can affect comparability. Despite these differences, there is a consistent overall order of performance among the three methods: ZebraPose outperforms PViT-6D, while GDR-NPP falls short in terms of performance. This observation highlights the 2-year performance gap between SOTA approach and older methods.



(a) Distance interval: [0,2]m (b) Distance interval: [2,4]m (c) Distance interval: [4,6]m

Fig. 4. Performance of the three SoTA methods on our synthetic test set for three different distance intervals. The [0,2] m interval represents the usual case study of traditional BOP datasets.

Table 6
Comparison of three pose estimators precision, trained on two classic BOP datasets, only using the synthetic data to train and the real data to test.

	LMO		YCBV	
	ADD (article)	ADD (ours)	ADD (article)	ADD (ours)
PViT-6D	**58.6**	53.50	27.4	**55.8**
ZebraPose	**57.9**	52.07	**62.6**	59.49
GDR-NPP	**46.8**	44.36	**41.5**	13.40

Table 7
Comparison of PViT-B precision, trained on either train-pbr or train-pbr+ on the test dataset.

	ADD	5cm	5°	5cm5°	2cm
train-pbr	92.33	92.60	98.12	91.81	78.80
train-pbr+	**93.72**	**93.92**	**98.84**	**93.42**	**81.05**
	2°	2cm2°	1cm	1°	1cm1°
train-pbr	85.12	71.60	62.08	46.91	34.39
train-pbr+	**87.46**	**74.68**	**64.51**	**51.57**	**38.44**

4.2.3. Three methods on our synthetic dataset

We trained the 3 methods on LMO and YCBV synthetic data, two known BOP datasets, to guarantee that our training process achieves comparable results to those reported in existing literature.

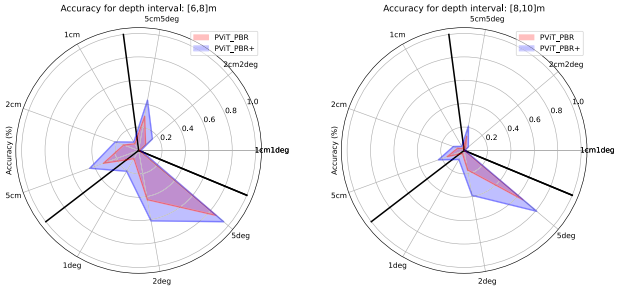
As our case study is far from those usually addressed by the BOP datasets, we have to test the 3 models on our synthetic dataset, which we decompose in 80% train set and 20% test set. Even if our dataset does not contain as many occlusions as LMO or YCBV, the challenge that it represents is still under-explored. On Fig. 4, we compare the precision of the 3 methods for Euclidean distance, geodesic distance and both of them combined for the 3 thresholds, as explain in Section 3.3. The Fig. 4(a) has similar setup as in the BOP challenge, where objects are not farther than 2 m away from the camera. In the three different distance intervals, PViT-6D consistently outperforms both ZebraPose and GDR-NPP. Interestingly, as the threshold decreases, PViT-6D's performance advantage over the other two methods becomes more pronounced. Furthermore, PViT-6D is also significantly faster than the other two methods, with a speedup of more than 2 times for GDR-NPP and more than 6 times for ZebraPose. The performance disparity between PViT-6D and the other pose estimators can be attributed to their distinct architectures. The use of attention mechanisms in PViT-6D enables it to tap into global features from the start, whereas ZebraPose and GDR-NPP rely on convolutional architectures that necessitate multiple layers to access these global features. This fundamental difference in architecture has a profound impact on their performance and efficiency. Moreover, the gap between the ZebraPose performance and the PViT-6D's one could be explained by the fact that as our CAD models of the drone is very complex, containing more than 1.6 millions points, the 16 bits quantization seems not precise enough to encode its geometry. This question will be studied in Section 5.3. Furthermore, as the drone moves farther from the camera, the less semantic information we have, which leads to a decrease in the model performance.

4.2.4. Train-pbr vs train-pbr+

Due to visibility checks, our initial synthetic database named train-pbr was biased, many of the most distant drones being filtered out. A new dataset, named train-pbr+ with an almost uniform distance distribution was then created. Training on a uniform distance distribution is indeed a better practice, as a boost on every metrics can be seen on the Table 7. As the metrics is stricter, the boost of performance is greater, and the model trained on the train-pbr+ is more precise for farther drones.

4.2.5. Test on a challenging dataset

Our synthetic test dataset, like train-pbr, suffers from a non-uniform distance distribution. To overcome this limitation, we created an entirely new test set, called test+, that has an almost uniform distance distribution. Furthermore, to further challenge the pose estimation models, we incorporated novel textures into this test set that have never been seen during training. For more detailed information on the construction of this synthetic test dataset, please consult Section 3.1.1.



(a) Distance interval: [6,8]m (b) Distance interval: [8,10]m

Fig. 5. Comparison of PViT precision, trained on either train-pbr or train-pbr+ on the test+ dataset for the last two distance intervals.

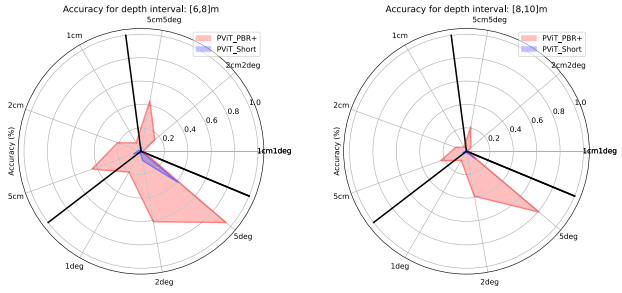
Fig. 5 presents the performance of two PViT-6D trained on either train_pbr or train_pbr+ for the two last distance intervals on this new test dataset: test+. As seen in Section 4.2.4, we observe that training on the more uniformly distributed pbr+ dataset leads to improved

performance compared to training on the non-uniformly distributed pbr dataset. Although we only show results for the last two distance intervals, our analysis suggests that PViT trained on pbr+ performs better across all five distance intervals, even in the [0,2]m interval. This outcome emphasizes the importance of using a uniformly distributed training set to achieve optimal performance. Furthermore, this experiment demonstrates the good generalization capabilities of pose estimators when dealing with unseen materials used for backgrounds, showcasing their ability to handle diverse scenarios.

It is worth noting that even though the test dataset has a non-uniform distance distribution, its overall performance across all five distance intervals is not significantly impacted by this issue. Only the overall performance can be questioned as there is an overrepresentation of the [0,2]m interval compared to the [6, 8]m one.

4.2.6. Short-range drone dataset

To highlight the necessity of a long-range dataset, we developed a short-range variant of our drone dataset. Utilizing the same hyperparameters presented in Section 4.2.1, we trained a PViT-6D model on the drone_short-range dataset. A comparison was made between the performance of this newly trained model and the pre-existing PViT-6D model, which had been previously trained on the normal range drone dataset, train_pbr+. The precision of both models is compared in Fig. 6 on the test+ datasets, with a focus on their performance across two distinct distance intervals: [6,8]m and [8,10]m interval, for which only one model was trained, as the short-range dataset only has sample as far as 2 m from the camera.



(a) Distance interval: [6,8]m (b) Distance interval: [8,10]m

Fig. 6. Comparison of PViT precision, trained on either train-pbr+ or short-range and tested on the test+ dataset for two distance intervals.

The Fig. 6 demonstrates that the model trained on a limited distance interval ([0,2]m) falls short in terms of performance, as it struggles to accurately localize drones within the scene. When challenged with estimating the pose of drones positioned at distances significantly farther than those encountered during training (more than 3 times farther), we are effectively testing its capacity for out-of-distribution generalization - a topic of ongoing research interest in the field.

4.2.7. Long-range LMO dataset

Building upon the previous section's idea, we compared the performance of PViT models trained on two variants of the LMO dataset: the original and a long-range version. Details about these datasets can be found in Section 3.1.1. The key differences between them lie in their range: the classic LMO dataset spans [0,2] m, whereas its long-range counterpart extends up to 10 m. On the Table 8, we observed that while PViT trained on the original dataset can generalize somewhat to distances between [2,4] m, its accuracy drops beyond 4 m. Both models demonstrate outstanding performance across the traditional BOP range, with a subtle difference due to the lack of occlusion in the long-range training set, which contrasts with the more complex scenario of the LMO dataset, heavily affected by occlusions. The model trained on the long-range dataset exhibits superior performance up to a distance of 10 m. Interestingly, as the distance interval increases, precision metrics do indeed decrease due to the diminishing semantic information available in the images.

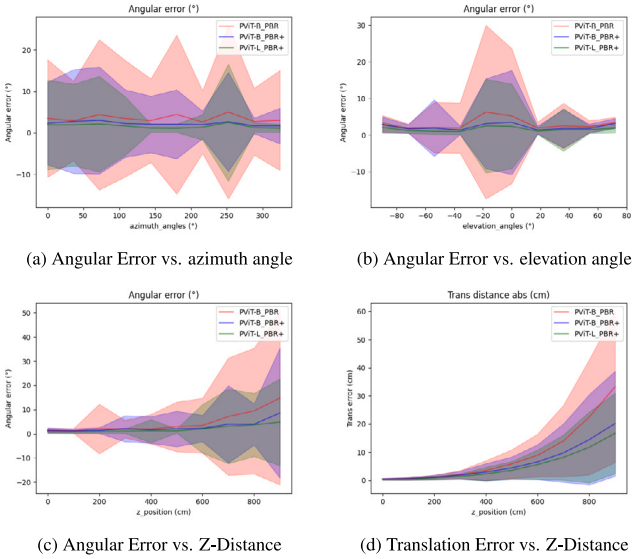
Table 8
Comparison of PViT-B precision, trained on either the classic LMO dataset or our long-range version of it, on the long-range test dataset. The performance are shown only for a single object, the ape.

	[0,2] m		[2,4] m		[4,6] m
	5cm	5°	5cm	5°	5 cm
Classic	99.7	98.94	56.74	73.26	0.38
Long-range	**100**	**99.24**	**93.79**	**97.68**	**48.38**

	[4,6] m	[6,8] m		[8,10] m	
	5°	5 cm	5°	5 cm	5°
Classic	2.67	0	0.1	0	0.1
Long-range	**79.48**	**17.93**	**39.63**	**12.18**	**15.98**

4.2.8. PViT-B vs PViT-L

Since PViT-6D seems to be the best choice for online pose estimation we compared two versions of the model, the basic one, referred to as PViT-B, and a version with a larger backbone, called PViT-L. We also compare the PViTs trained on the train-pbr and the train-pbr+. The use of a larger backbone, by creating finer features, boosts the performance of the model, especially on the 1 cm and 1° precision metrics. Moreover, the gap between the basic and the large backbones was greater for more distant objects: on the [6,8] m interval the large backbone doubles the performance in the 1 cm and 1° metrics, even if the performance remains low. The finer features mostly helps the rotation estimation. However, this boost of performance has a cost: the memory footprint and the inference time. With 4 times more parameters, the inference time increases by 70%.



(a) Angular Error vs. azimuth angle (b) Angular Error vs. elevation angle

(c) Angular Error vs. Z-Distance (d) Translation Error vs. Z-Distance

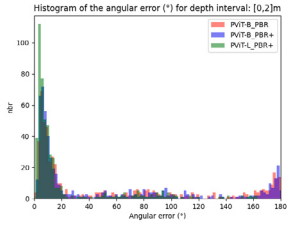
Fig. 7. Comparison of the errors of three different PViT models on the test+ dataset.

On Fig. 7, we can see the mean and the standard deviation of the angular and translation errors depending on either the azimuth or the elevation angle or the distance to the camera. We experimented various iterations of the PViT, with either a basic (PViT-B) or a large encoder (PViT-L), we also compared the PViTs trained on the train-pbr and the train-pbr+. When the drone faces the camera (azimuth angle equals to 180°), the angular error is minimal. When the camera is on the side, the model has a hard time to predict the rotation, as the standard deviation peaks. The translation error is less impacted by the azimuth angle. As the drone is not as large as it is long, there is a small rise in the standard deviation when the drone is seen from the side. As we can see on Fig. 7(b), the elevation angle has a high influence on the estimated pose, impacting both the angular and translation errors. An elevation angle

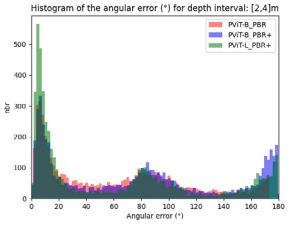
close to 0° is a challenging position for the model. As the elevation angle moves away from zero, a larger surface of the drone becomes visible, making it easier to estimate the pose. As seen on Figs. 7(c) and 7(d), the further the drone is from the camera, the greater the angular and translation errors. However, using a large encoder helps reduce the mean error and the standard deviation.

4.2.9. Real data test

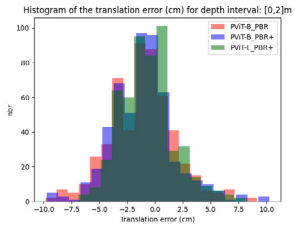
As the performance of the pose estimator is good on synthetic data, we test the PViT-6D on real data, with the perfect RoIs. We show on Fig. 8, the histogram of angular and translation errors of the three different PViT, for two distance intervals.



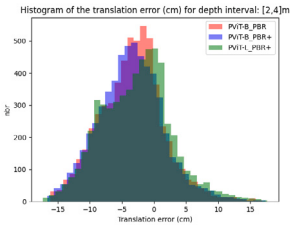
(a) Angular errors for the distance interval: [0,2]m



(b) Angular errors for the distance interval: [2,4]m



(c) Translation errors for the distance interval: [0,2]m



(d) Translation errors for the distance interval: [2,4]m

Fig. 8. Histograms of errors on the concatenation of droneFLYcam01 and droneFLYcam02 benchmark, for the three PViT models and two distance ranges.

As seen on Figs. 8(a) and 8(b), the farther the drone from the camera, the more incorrect the estimated pose. However, we can see peaks at 90° and 180°, which shows that there is an ambiguity caused by the symmetry of the quadcopter, which is worsened by the distance. In fact, when the camera worn by the observed drone is not seen or self-occluded, the model has a hard time estimating the rotation, as most of the features that provides the rotation come from that camera. When the drone is close to the viewing camera, i.e. in the first distance interval, the drone's estimated pose is mostly right or wrong with a 180° error. No symmetry-related errors were observed in the synthetic dataset, but the real images exhibited motion blur.

5. Ablation study

5.1. Agnosticity of camera

Estimating, with a neural network, the depth of an object without prior information is camera specific. Creating a camera agnostic neural network is still a research topic. However, as the PViT-6D works within the RoI which is resized into a fixed input size, the predicted pose does not depend on the camera. Moreover, as the full 6D pose is found with the SIZP parametrization, presented in Section 2.3, any kind of camera can be utilized. In this case, a single model can be used on diverse drone platforms.

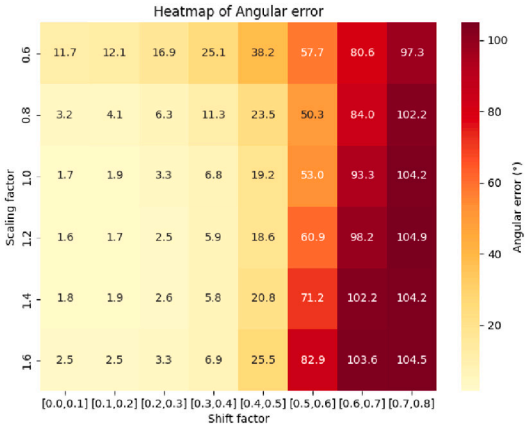
To test the agnosticity with respect to the camera, we created two twin datasets CamBase and CamOak, with the pipeline presented in Section 3.1.1 shown on Fig. 1. CamBase uses the same intrinsic parameters as in the rest of the paper. CamOak has a smaller focal length (457 pixels) and a higher resolution (768 × 480). The Table 9, displaying the performance of the PViT-L on the two cameras, shows that a higher resolution camera boosts the performance. As the PViT relies on semantics to regress the 6D pose, more information improves the pose estimation, specifically the rotation estimation is far more precise.

Table 9
Comparison of the precision of the PViT-L on two cameras.

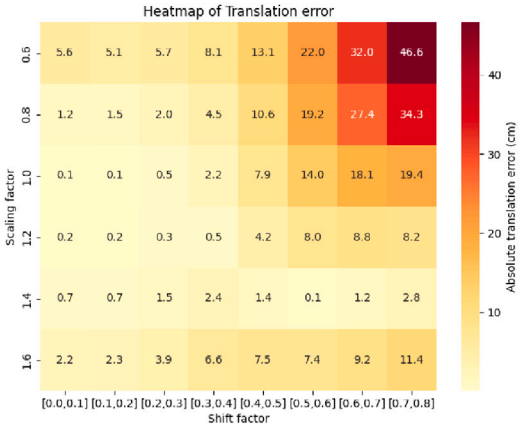
	1 cm	1°	1 cm1°	2 cm	2°	2 cm2°
cambase	23.0	44.6	13.2	40.8	76.5	37.9
camoak	**27.5**	**55.3**	**20.7**	**47.3**	**86.0**	**45.0**

5.2. RoI study

Since detected RoIs are never perfect, most training pipelines consider this problem and use data augmentation, such as scaling or shifting the RoI. In our study, where the drone can be far, we aim to evaluate the robustness of our pose estimator with respect to detection uncertainties. We test the PViT-B on our synthetic data, with scaling factor from 0.4 to 3.5 with a 0.1 step and by sampling uniformly a shifting factor in 8 intervals from [0.0,0.1] to [0.7,0.8].



(a) Angular error



(b) Absolute translation error

Fig. 9. Heatmap of the mean angular and mean translation error for different scale and shift values.

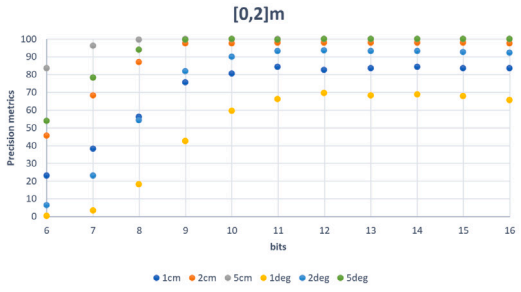
The performance of the PViT-B is not much impacted by the scaling of the RoI when it remains within [0.8,1.6]. As the nominal value used in the PViT-B is 1.2, it shows that the model is able to perform a good estimation of the 6D pose even with an error of 33% in the size of the RoI. On Fig. 9, we can see that if the scale imprecision is below 33%, a shift of the RoI's center has a low impact on either the angular

or translation error, as it stays under 40%. Above this threshold, the angular error rises, as almost half of the drone is not visible. However, if the scale exceeds 1.2, the shift factor induces a lesser error. This study shows the remarkable robustness of the PViT architecture.

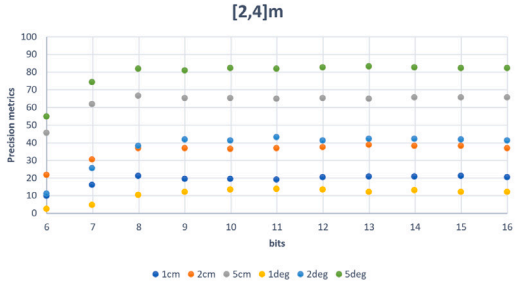
5.3. Importance of the bit in ZebraPose

The authors of ZebraPose [13] argue that if an object is small or is further away from the camera, multiple finer bits could match an unique image pixel, being then redundant. In their paper, they demonstrate the evolution of ADD metrics across varying bits used to do the correspondence. However, their study's scope is limited to the LMO dataset, which has restricted range. We replicated this study using our own precision metrics and applied them to both the long-range version of LMO and our drone dataset.

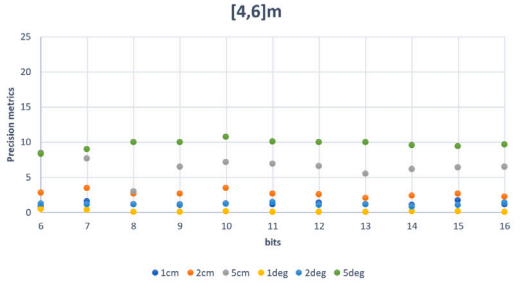
Fig. 10 illustrates the effect of varying the number of bits on the precision metrics used. As the authors of [13], we suggest that a fixed number of bits may not be universally applicable for all scenarios, depending on range and on the object. For certain pose estimation applications one might want to focus on the rotation estimation rather than the translation one or the inverse, the best number of bits for each application might be different as we can see on Fig. 10. Moreover, as the object is farther away from the camera, the precision metrics are less impacted by the number of bits used.



(a) Distance interval: [0,2]m



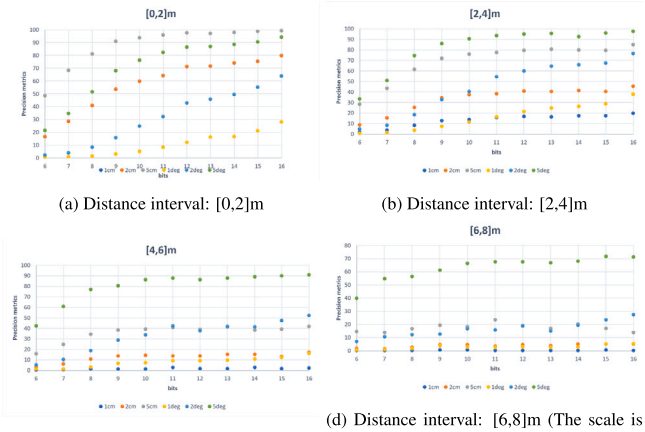
(b) Distance interval: [2,4]m



(c) Distance interval: [4,6]m (The scale is not up to 100%)

Fig. 10. Comparison of the precision of ZebraPose by the number of bits used to make the correspondence on the test part of long-range LMO dataset.

Fig. 11 shows the same experiment as Fig. 10 but applied to the test set of the train_pbr dataset. In the LMO setup, using the full 16bits is not necessary, instead when this study is applied to our drone setup, using the maximum number of bits gives the best performance. As in the LMO setup, the impact of the number of bits used decrease as the distance rise up, but in short range case we observe a continuous improvement. This difference could be explained by the complexity of our object. Usually, an object's mesh has to be upsampled to have enough points to create a 16-bit encoding, with our model this step is not needed. For complex objects, utilizing 16-bit code yields the optimal performance. However, for less complex models like the LMO object, using fewer bits might offer a computational complexity reduction.



(a) Distance interval: [0,2]m

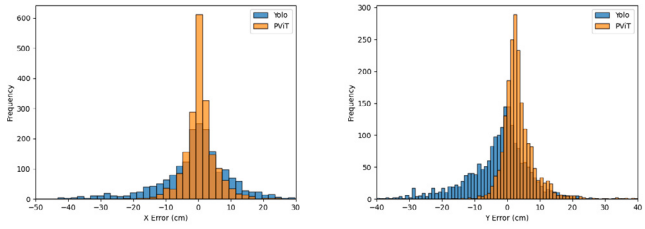
(b) Distance interval: [2,4]m

(c) Distance interval: [4,6]m

(d) Distance interval: [6,8]m (The scale is not up to 100%)

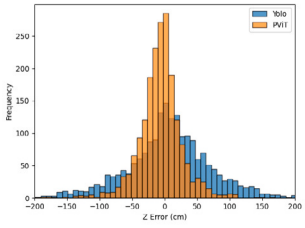
Fig. 11. Comparison of the precision of ZebraPose by the number of bits used to make the correspondence on the test part of train-pbr.

5.4. Pose estimation on detected bbox



(a) X error

(b) Y error



(c) Z error

Fig. 12. Histogram of errors on each axis of the estimated 3D position on the droneGRDcam1 dataset. In blue: by using only the detected RoI from YOLO, in orange: by using 6D pose provided by PViT-6D.

Using a prior on the size of our object, a simple object detector could be used to get a rough estimation of the 3D position of our drone, using the size of the RoI and its center. The goal of pose estimators is to refine this first position estimated and regress the 3D rotations, so as to have the full 6D pose. As shown on Fig. 12, the use of the pose estimator improves the estimation of the 3D position of the drone, which is clearly visible by the reduction of the standard deviation. Moreover, the scale and shift error made by the yolov8-n are in the same robustness intervals as in the synthetic data, which shows that our study scales well with real data.

6. Conclusion

Developing an estimator capable of accurately predicting the 6D pose of distant objects would significantly advance the automation of industrial robots and drone swarms. We presented a pipeline to train a pose estimator from scratch using custom real and synthetic datasets, covering dataset creation and annotation. Additionally, we designed a comprehensive test pipeline that takes input from full RGB images to estimate the 6D pose. Our results demonstrate that the PViT-6D model offers the best balance of speed and accuracy, even in challenging scenarios where the object, such as a drone, is as far as 10 m. Since the pose estimator operates within a region of interest (RoI), we addressed the challenges of training object detectors to handle a wide range of scales. The PViT-6D model has shown robustness against significant RoI errors, withstanding up to a 33% scale error, and is agnostic to the camera used for capturing the images. Unlike other methods, our approach does not rely on any refinement techniques, which means that additional improvement can still be expected. Moreover, until now our models have not been quantified or pruned, so better efficiency should be achieved on embedded hardware. Futures studies for real application should investigate the impact of the motion of the observing camera and also the influence of the rolling shutter to capture a highly dynamic structure. Furthermore, the full pipeline should benefit from tracking-based pose prediction, or posterior time-filtering of the pose sequence. Then, one should explore the addition of temporal information to those types of architectures, either directly as input or used as a post-process.

CRediT authorship contribution statement

Thomas Rey: Writing – review & editing, Writing – original draft, Software, Methodology, Investigation, Data curation, Conceptualization. **Julien Moras:** Writing – review & editing, Supervision, Methodology. **Alexandre Eudes:** Writing – review & editing, Supervision, Methodology. **Antoine Manzanera:** Writing – review & editing, Supervision, Methodology.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Thomas REY reports financial support was provided by French National Research Agency. Thomas REY reports a relationship with French National Research Agency that includes: funding grants. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work has been partially supported and carried out in the framework of the VORTEX project (ANR-23-IAS2-0005-01).

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.mechatronics.2025.103339.

Data availability

The authors do not have permission to share data.

References

[1] Hodaň T, Michel F, Brachmann E, Kehl W, Buch AGlent, Kraft D, Drost B, Vidal J, Ihrke S, Zabulis X, Sahin C, Manhardt F, Tombari F, Kim TK, Matas J, Rother C. BOP: Benchmark for 6D object pose estimation. In: European conference on computer vision. ECCV, 2018.

[2] Xiang Y, Schmidt T, Narayanan V, Fox D. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. CoRR 2017. arXiv:1711.00199.

[3] Brachmann E, Krull A, Michel F, Gumhold S, Shotton J, Rother C. Learning 6d object pose estimation using 3d object coordinates. In: Computer vision–ECCV 2014: 13th European conference, zurich, Switzerland, (2014) 6-12, proceedings, part II 13. Springer; 2014, p. 536–51.

[4] Wu X, Sahoo D, Hoi SC. Recent advances in deep learning for object detection. Neurocomputing 2020;396:39–64.

[5] Girshick R, Donahue J, Darrell T, Malik J. Region-based convolutional networks for accurate object detection and segmentation. IEEE Trans Pattern Anal Mach Intell 2015;38:142–58.

[6] Ren S, He K, Girshick R, Sun J. Faster r-cnn: Towards real-time object detection with region proposal networks. IEEE Trans Pattern Anal Mach Intell 2016;39:1137–49.

[7] Redmon J, Divvala S, Girshick R, Farhadi A. You only look once: Unified, real-time object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2016, p. 779–88.

[8] Liu W, Anguelov D, Erhan D, Szegedy C, Reed S, Fu CY, Berg AC. Ssd: Single shot multibox detector. In: Computer vision–ECCV 2016: 14th European conference, amsterdam, the netherlands, (2016) 11–14, proceedings, part i 14. Springer; 2016, p. 21–37.

[9] Vijayakumar A, Vairavasundaram S. Yolo-based object detection models: A review and its applications. Multimedia Tools Appl 2024;1–40.

[10] Rad M, Lepetit V. Bb8: A scalable, accurate, robust to partial occlusion method for predicting the 3d poses of challenging objects without using depth. In: Proceedings of the IEEE international conference on computer vision. 2017, p. 3828–36.

[11] Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition. 2014, arXiv preprint arXiv:1409.1556.

[12] Park K, Patten T, Vincze M. Pix2pose: Pixel-wise coordinate regression of objects for 6d pose estimation. Proc the IEEE/ CVF Int Conf Comput Vis 2019;7668–77.

[13] Su Y, Saleh M, Fetzer T, Rambach J, Navab N, Busam B, Stricker D, Tombari F. Zebrapose: Coarse to fine surface encoding for 6dof object pose estimation. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2022, p. 6738–48.

[14] He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2016, p. 770–8.

[15] Kehl W, Manhardt F, Tombari F, Ilic S, Navab N. Ssd-6d: Making rgb-based 3d detection and 6d pose estimation great again. 2017, arXiv:1711.10006.

[16] Wang G, Manhardt F, Tombari F, Ji X. Gdr-net: Geometry-guided direct regression network for monocular 6d object pose estimation. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2021, p. 16611–21.

[17] Vaswani A. Attention is all you need. Adv Neural Inf Process Syst 2017.

[18] Stapf S, Bauernfeind T, Riboldi M. Pvit-6d: Overclocking vision transformers for 6d pose estimation with confidence-level prediction and pose tokens. 2023, arXiv preprint arXiv:2311.17504.

[19] Brégier R. Deep regression on manifolds: a 3d rotation case study. In: 2021 international conference on 3D vision (3DV). IEEE.; 2021, p. 166–74.

[20] Zhou Y, Barnes C, Lu J, Yang J, Li H. On the continuity of rotation representations in neural networks. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2019, p. 5745–53.

[21] Mildenhall B, Srinivasan PP, Tancik M, Barron JT, Ramamoorthi R, Ng R. Nerf: Representing scenes as neural radiance fields for view synthesis. Commun ACM 2021;65:99–106.

[22] Yen-Chen L, Florence P, Barron JT, Rodriguez A, Isola P, Lin TY. Inerf: Inverting neural radiance fields for pose estimation. In: 2021 IEEE/RSJ international conference on intelligent robots and systems. IROS, IEEE.; 2021, p. 1323–30.

[23] Claessens L, Manhardt F, Martin-Brualla R, Siegwart R, Lerma CDC, Tombari F. Robust and efficient edge-guided pose estimation with resolution-conditioned nerf. BMVC; 2023, p. 543.

[24] Csehi ÁI, Józsa CM. Bid-nerf: Rgb-d image pose estimation with inverted neural radiance fields. 2023, arXiv preprint arXiv:2310.03563.

[25] Kerbl B, Kopanas G, Leimkühler T, Drettakis G. 3D gaussian splatting for real-time radiance field rendering. ACM Trans Graph 2023;42:1–139.

[26] Sun Y, Wang X, Zhang Y, Zhang J, Jiang C, Guo Y, Wang F. Icomma: Inverting 3d gaussians splatting for camera pose estimation via comparing and matching. 2023, arXiv preprint arXiv:2312.09031.

[27] Xu L, Qu H, Cai Y, Liu J. 6D-diff: A keypoint diffusion framework for 6d object pose estimation. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2024, p. 9676–86.

[28] Hoque S, Arafat MY, Xu S, Maiti A, Wei Y. A comprehensive review on 3d object detection and 6d pose estimation with deep learning. IEEE Access 2021;9:143746–70.

[29] Sundermeyer M, Hodaň T, Labbé Y, Wang G, Brachmann E, Drost B, Rother C, Matas J. BOP challenge 2022 on detection, segmentation and pose estimation of specific rigid objects. In: Conference on computer vision and pattern recognition workshops. CVPRW, 2023.

[30] Denninger M, Winkelbauer D, Sundermeyer M, Boerdijk W, Knauer M, Strobl KH, Humt M, Triebel R. Blenderproc2: A procedural pipeline for photorealistic rendering. J Open Source Softw 2023;8:4901. http://dx.doi.org/10.21105/joss.04901.

[31] Hodaň T, Sundermeyer M, Drost B, Labbé Y, Brachmann E, Michel F, Rother C, Matas J. Bop challenge 2020 on 6d object localization. In: Computer vision–ECCV 2020 workshops: glasgow, UK, (2020) 23–28, proceedings, part II 16. Springer; 2020, p. 577–94.

[32] Rehder J, Nikolic J, Schneider T, Hinzmann T, Siegwart R. Extending kalibr: Calibrating the extrinsics of multiple imus and of individual axes. In: 2016 IEEE international conference on robotics and automation. ICRA, IEEE; 2016, p. 4304–11.

[33] Liu X, Zhang R, Zhang C, Fu B, Tang J, et al. Gdrnpp. 2022, https://github.com/shanice-l/gdrnpp_bop2022.

[34] Li Z, Wang G, Ji X. Cdpn: Coordinates-based disentangled pose network for real-time rgb-based 6-dof object pose estimation. In: Proceedings of the IEEE/CVF international conference on computer vision. 2019, p. 7678–87.