

Efficient Self-Supervised Neuro-Analytic Visual Servoing for Real-time Quadrotor Control

Sebastian Mocanu¹ Sebastian-Ion Nae^{1*} Mihai-Eugen Barbu^{1*} Marius Leordeanu^{1, 2, 3}

¹National University of Science and Technology POLITEHNICA Bucharest, Romania

²Institute of Mathematics "Simion Stoilow" of the Romanian Academy, Romania

³NORCE Norwegian Research Center, Norway

{sebastian.mocanu@, sebastian.ion.nae@stud.filis., mihai.eugen.barbu@stud.acs.}upb.ro, leordeanu@gmail.com

Abstract

This work introduces a self-supervised neuro-analytical, cost efficient, model for visual-based quadrotor control in which a small 1.7M parameters student ConvNet learns automatically from an analytical teacher, an improved image-based visual servoing (IBVS) controller. Our IBVS system solves numerical instabilities by reducing the classical visual servoing equations and enabling efficient stable image feature detection. Through knowledge distillation, the student model achieves 11× faster inference compared to the teacher IBVS pipeline, while demonstrating similar control accuracy at a significantly lower computational and memory cost. Our vision-only self-supervised neuro-analytic control, enables quadrotor orientation and movement without requiring explicit geometric models or fiducial markers. The proposed methodology leverages simulation-to-reality transfer learning and is validated on a small drone platform in GPS-denied indoor environments. Our key contributions include: (1) an analytical IBVS teacher that solves numerical instabilities inherent in classical approaches, (2) a two-stage segmentation pipeline combining YOLOv11 with a U-Net-based mask splitter for robust anterior-posterior vehicle segmentation to correctly estimate the orientation of the target, and (3) an efficient knowledge distillation dual-path system, which transfers geometric visual servoing capabilities from the analytical IBVS teacher to a compact and small student neural network that outperforms the teacher, while being suitable for real-time onboard deployment.

1. Introduction

Quadrotor applications have expanded from remote controlled aircraft to autonomous systems for aerial photography, mapping, and delivery, typically relying on GPS,

IMU, and 9-DOF sensor fusion [1]. While effective outdoors, these sensors suffer from degraded performance indoors due to signal interference, multipath effects, and calibration uncertainties [28, 35]. Vision-based sensors are cheaper and address these limitations through direct environmental perception and immediate visual feedback, enabling real-time feature extraction and spatial awareness for autonomous navigation in GPS-denied environments.

Applications requiring spatial coordination between quadrotors and target objects, such as automated warehouse inspection systems [39] and dynamic cinematography [21], require robust visual servoing methodologies. In warehouse maintenance scenarios, quadrotors must maintain specific distances [12] and viewing angles relative to infrastructure components while compensating for environmental disturbances [39]. Similarly, cinematic applications demand continuous pose adjustment to maintain desired framing as subjects move through complex trajectories [6].

Visual servoing uses a closed-loop feedback to compare desired image features with current measurements, using feature coordinates and geometric properties as direct control inputs without explicit 3D pose reconstruction [7, 9]. However, classical Image-Based Visual Servoing (IBVS) methods often suffer from numerical instabilities due to singularities in the interaction matrix and conditioning issues during large camera motions [4]. While object tracking and pose alignment tasks typically use fiducial markers such as ArUco [5, 10, 20] or AprilTags [2, 23, 44] that provide easily detectable features with predefined geometric properties, these approaches still face computational challenges and marker dependencies that limit deployment in dynamic, unstructured indoor environments.

Our work presents an efficient marker-free approach utilizing a self-supervised analytical IBVS teacher model that addresses numerical instabilities through reduced classical equations and enhanced image feature detection, coupled with a lightweight student neural network that achieves su-

*These authors contributed equally to this work.

terior real-time performance. The proposed self-supervised teacher method integrates a two-stage neural network approach: first, a YOLO [29, 30, 42] instance segmentation network for object detection, then a second stage that splits the generated mask in half to estimate the orientation (front/back segmentation) for a toy car target, enabling precise drone positioning without fiducial markers. This knowledge distillation framework transfers the analytical stability of the improved IBVS teacher to a more efficient, low-cost student network while achieving better control performance. System validation follows a progressive methodology, beginning with simulation-based training and subsequently incorporating real-world data for fine-tuning.

The experimental setup requires minimal hardware infrastructure, consisting only of a standard laptop computer and wireless communication link to the quadrotor platform, demonstrating practical feasibility for indoor autonomous operations while also targeting future edge deployment.

2. Visual-Based Servoing

Visual servoing represents a fundamental control paradigm for mobile robots utilizing camera-based feedback. Classical approaches are categorized into Position-Based Visual Servoing (PBVS) and Image-Based Visual Servoing (IBVS). PBVS reconstructs 3D pose from image features for Cartesian space control[26, 34], while IBVS directly employs image features as feedback signals, circumventing pose estimation uncertainties[4, 14, 40]. For real-time robotic applications, IBVS is predominantly adopted due to its computational efficiency and robustness [13, 41].

The IBVS control law is formulated as:

$$\mathbf{v} = -\lambda \mathbf{L}_s^+ (\mathbf{s} - \mathbf{s}^*) \quad (1)$$

where $\mathbf{v} = [v_x, v_y, v_z, \omega_x, \omega_y, \omega_z]^T$ represents the camera velocity screw (linear and angular velocities), λ is the control gain, \mathbf{s} are current image features, and \mathbf{s}^* are desired features.

The interaction matrix \mathbf{L}_s (image Jacobian) relates feature motion to camera motion:

$$\dot{\mathbf{s}} = \mathbf{L}_s \mathbf{v} \quad (2)$$

For each point at pixel coordinates (u, v) with depth Z , the interaction matrix is:

$$\mathbf{L}_s = \begin{pmatrix} -\frac{f}{Z} & 0 & \frac{\bar{u}}{Z} & \frac{\bar{u}\bar{v}}{f} & -\frac{f^2 + \bar{u}^2}{f} & \bar{v} \\ 0 & -\frac{f}{Z} & \frac{\bar{v}}{Z} & \frac{f^2 + \bar{v}^2}{f} & -\frac{\bar{u}\bar{v}}{f} & -\bar{u} \end{pmatrix} \quad (3)$$

where $\bar{u} = u - u_0$ and $\bar{v} = v - v_0$ are the pixel coordinates relative to the principal point, while f is the focal length expressed in units of pixels.

The pseudo-inverse \mathbf{L}_s^+ is computed as:

$$\mathbf{L}_s^+ = (\mathbf{L}_s^T \mathbf{L}_s)^{-1} \mathbf{L}_s^T \quad (4)$$

Feature detection represents a critical component for IBVS implementation. Marker-based approaches predominate due to their reliability and implementation simplicity. Inria demonstrated ArUco marker tracking using a Parrot Bebop 2 for moving target applications[18] and PID-based tracking of moving targets[38]. Similar marker-based systems have been deployed for autonomous drone landing and surface detection tasks[11, 32], establishing their efficacy in real-world scenarios. However, marker-based methods face practical limitations including sensitivity to environmental conditions such as variable lighting, partial occlusion, and marker degradation. Additionally, marker installation and maintenance requirements limit applicability in dynamic or inaccessible environments [16].

Recent advances in deep learning have introduced alternative feature detection approaches. Neural networks as keypoint detectors have been employed for specific object tracking, as demonstrated by[33], where a CNN was trained to detect keypoints on rectangular objects under varying illumination and occlusion conditions. Other works focus on bounding box detection and object center tracking during aggressive maneuvers[27]. The YOLO family of models has been extensively adopted for robotics applications[15, 43], including instance segmentation[19, 36], object detection[32], and image classification[45], demonstrating the versatility of deep learning approaches for visual servoing feature extraction [25].

Recent approaches integrate neural networks with visual servoing algorithms to enhance robotic control. Liu et al.[17] developed a two-stream network for robotic arm manipulation, directly outputting velocity commands to achieve desired poses without explicit visual servoing computations. Knowledge distillation techniques have shown promise in lightweight servo networks, as demonstrated by[32], where a student network learns optimal grasp point identification for tree branch manipulation.

Our work extends these concepts by distilling knowledge from YOLO segmentation networks combined with IBVS algorithms, eliminating fiducial marker requirements while maintaining computational efficiency. The proposed lightweight network directly generates velocity commands from single camera frames. Additionally, we address the simulation-to-reality gap through continuous learning, initially training in simulation and subsequently fine-tuning with real-world data to ensure robust performance across deployment environments.

3. Our Approach

Our method consists of a teacher-student framework for vision-based quadrotor control, both teacher and student are

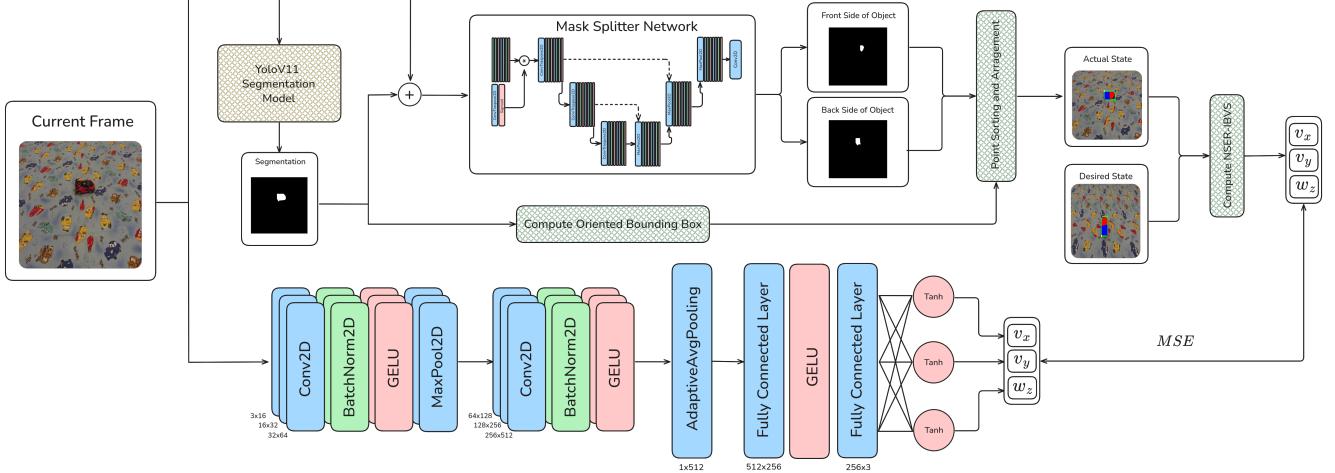


Figure 1. Overview of the proposed Teacher-Student Self-supervised Neuro-Analytic model. Top row illustrates the NSER-IBVS (analytical) Teacher path, which uses YOLO for object segmentation and a mask splitter network to estimate object orientation relative to the drone camera. The bottom one represents the Student path, which learns by minimizing the MSE cost between its output drone commands (v_x, v_y, ω_z) and the Teacher’s output.

tested in a digital-twin of the real-world environment.

The teacher employs a two-stage neural network processing system that is combined with an image-based visual servoing algorithm. In the first stage, a small YOLOv11 Nano (2.84M parameters) instance segmentation network is used. This network is initially trained on synthetic data from the digital-twin and subsequently fine-tuned using real images. This stage has two main objectives: to compute an oriented bounding box (OBB) of the car and to provide its mask to the second stage of our system, as the OBB alone is insufficient for accurate pose inference. The second stage is a specialist neural network trained to split the YOLO masks in half, generating masks that represent the front and back regions of the target vehicle. Knowing the position of these front and back masks and having the OBB, we can rearrange the keypoints to generate four ordered points (two per region) that serve as stable visual features for IBVS. The image Jacobian and control velocities are computed using these features to minimize the error and achieve the desired relative state. The desired orientation keypoints are established by capturing the target pose at the beginning, applying segmentation, and extracting the corresponding feature points. Then we preprocess these keypoints for stability enhancement, detailed in Section 3.1. Our method then rectifies its pose with respect to the desired reference pose.

This framework constitutes the teacher model, which is distilled into a compact convolutional neural network trained via regression to output quadrotor control velocities from camera input RGB images detailed in Section 3.2.

3.1. Numerically Stable Efficient Reduced IBVS

The proposed method begins by training a YOLOv11 Nano instance segmentation model using simulation-generated data, forming the foundation of a two-stage segmentation pipeline. The first stage segments the entire vehicle, while a secondary network splits the segmented region into anterior and posterior components, as illustrated in Figure 2. While effective for segmentation, the first stage generates bounding boxes that may include parts not belonging to the object. To address this issue, we compute a bounding box around the segmented object mask. This produces an unstable point ordering at inference time, introducing inconsistencies for the analytical IBVS algorithm. By determining the positions of the frontal and rear masks, we can rearrange these points to create an oriented bounding box that better represents the car pose and provides more stable features points, as shown in Figure 3.

The mask splitting network employs a U-Net [31] architecture with attention mechanisms [22], accepting a 4-channel input tensor that comprises the original RGB image and the binary vehicle mask from YOLOv11. The encoder-decoder structure features skip connections and progressively downsamples through three stages with channel dimensions of 32, 64, 128, and 256. An attention mechanism applied to the mask channel generates spatial weights that modulate image features to focus on vehicle regions. The decoder reconstructs spatial resolution through transposed convolutions, producing a 2-channel output that represents anterior and posterior segmentation masks. The network contains approximately 1.94M parameters and incorporates dropout regularization in deeper layers preventing overfit.

The splitter loss function partitions a binary segmenta-

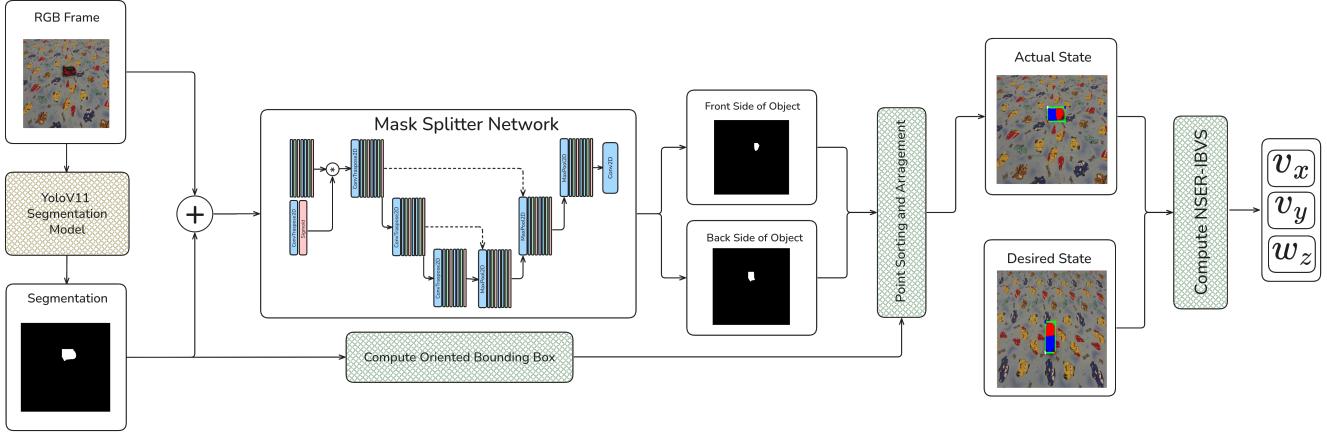


Figure 2. Overview of the proposed Numerically Stable, Efficient, Reduced Image Based Visual Servoing (NSER-IBVS) architecture. The pipeline takes an RGB frame, generates segmentation, and feeds them as input to the Mask Splitter Network. From the original segmentation, bounding box coordinates are determined and used together with the divided front and back masks to rearrange coordinates positions. These coordinates serve as visual features for the analytical model to compute the velocity commands (v_x , v_y , w_z).



Figure 3. Comparison of different bounding box approaches derived from segmentation results: (left) regular bounding box including parts that do not belong to the object, (middle) oriented bounding box that may vary in orientation between frames, and (right) oriented bounding box using a mask-splitting network to separate anterior and posterior vehicle components for improved ordering stability.

tion mask into complementary front and back regions by reconstructing the original input mask. Multiple constraints enforce proper partitioning: individual binary cross-entropy losses ensure that each predicted mask matches its ground truth target, a partition constraint penalizes deviations when the sum of both predicted masks differs from the original mask, and overlap penalties discourage simultaneous occupancy of the same pixels. The coverage loss ensures accurate reconstruction by penalizing both excess predictions beyond the original mask and the missed areas within it. Loss weights are dynamically scheduled during training, emphasizing individual mask accuracy, then gradually shifting toward stronger partition and overlap constraints.

The resulting segmentation masks are processed to compute an oriented bounding box by fitting the minimum enclosing rectangle to the object boundary. The four corner

coordinates of this bounding box serve as visual features of the IBVS framework. To ensure keypoint stability and consistent ordering, we employ a preprocessing algorithm that assigns points to anterior or posterior regions based on their proximity to the respective mask centroids. The points are subsequently ordered clockwise to maintain temporal consistency across frames, thereby preventing feature correspondence ambiguity that could destabilize the IBVS control loop.

Control velocity is computed by comparing current keypoint coordinates to reference features from the desired pose. Given the quadrotor’s underactuated dynamics, our efficient approach computes only essential velocity components from the classical IBVS controller by eliminating angular velocities ω_x , ω_y , as well as the linear velocity v_z , since our UAV operates at fixed altitude.

Starting from Equation 3, the reduced interaction matrix takes the form:

$$\begin{pmatrix} \dot{u} \\ \dot{v} \end{pmatrix} = \begin{pmatrix} -\frac{f}{Z} & 0 & \bar{v} \\ 0 & -\frac{f}{Z} & -\bar{u} \end{pmatrix} \begin{pmatrix} v_x \\ v_y \\ \omega_z \end{pmatrix} \quad (5)$$

that we use to stack the Jacobians corresponding to the keypoints from the oriented rectangle.

3.2. Teacher-Student Model Distillation

To enable real-time deployment, we employ knowledge distillation [8] to transfer the visual servoing capabilities from the teacher IBVS pipeline to a lightweight, low-cost student network. The student network is designed as a 1.7M-parameter CNN that directly regresses control velocities from RGB images using mean squared error (MSE) loss.

A key component of this architecture is the normalization of the target labels (the velocities commands) and their respective de-normalization at inference time. To train this student network, we require both RGB images and their corresponding control velocities. We analyzed the values from the experiments conducted on both real-world and digital-twin environments and determined the minimum and maximum bounds for each command across all scenes, as shown in Table 1. The normalization of target labels improves training stability by ensuring all velocity commands operate within similar numerical ranges, preventing certain outputs from dominating the loss function due to scale differences. This approach also enhances gradient flow during backpropagation and enables the use of bounded activation functions like tanh, which naturally constrains the network outputs to physically meaningful control ranges while maintaining differentiability throughout the training process.

	ν_x		ν_y		ω_z	
	min	max	min	max	min	max
Sim	-24	16	-9	9	-27	40
Real	-30	30	-30	30	-40	40

Table 1. Minimum and maximum command values for translational velocities ν_x and ν_y and rotational velocity ω_z in digital-twin (Sim) and real-world (Real) collected from NSER-IBVS evaluation runs. Used to determine the normalization of the target labels for the self-supervised method.

The student architecture consists of a convolutional feature extractor followed by fully connected regression layers. The convolutional backbone progressively increases channel dimensions from 16 to 512 through six convolutional blocks, each incorporating batch normalization and GELU activation functions. Max pooling is applied after the first two blocks for spatial downsampling, while the final layers use kernel sizes of 3x3 to capture fine-grained spatial features. An adaptive global average pooling layer reduces spatial dimensions to 1x1, followed by two fully connected layers (512→256→3) that output final velocity commands.

Given the limited availability of real-world data, training is performed in two stages. The first stage is pretraining on simulated trajectories from the digital-twin. The second stage is fine-tuning on real-world data. For the real-world fine-tuning stage, we implement targeted data augmentation strategies to enhance dataset diversity and improve generalization to unseen conditions.

The distillation process transfers knowledge from the NSER IBVS teacher to the neural student, enabling direct end-to-end control from visual input while maintaining the characteristics of the classical visual servoing approach, thereby yielding similar trajectories and behavior.

3.3. Data Generation and Simulation Environment

The experimental environment is replicated using a simulator based on the Parrot Anafi drone platform with its accompanying Sphinx simulation framework [24]. This simulation represents a digital-twin for our real-world testing environment, featuring accurate measurements, object structure and poses. The physical test setup consists of a 5 × 4-meter carpet with a centrally positioned toy car target and the quadrotor positioned at varying poses from the object. The carpet is necessary because the bunker-like environment has a non-Lambertian floor, which poses challenges for our UAV to maintain a fixed position.

Simulated environment creation uses Unreal Engine 4 for level design and asset placement and Blender [3] to create the meshes for most of the assets. Since the target vehicle was harder to reproduce in Blender, it was digitized using Hunyuan3D-2 [37] to generate a .fbx mesh file, which is then scaled to match the dimensions of the physical objects in the real-world for accurate simulation fidelity.

Data generation involves flight sequences within both the digital-twin and real-world environments using varying gimbal camera angles (30°, 45°, 60°, 75°, 90°) for training and 45° for validation, since that is the angle we want to test our method. For the simulated environment, we used two rendering configurations: low and high quality that affect lighting and mesh fidelity to ensure generality and a wide-range of FOVs. With this approach, we created a dataset of 14693 frames for training and 1123 for validation from the digital-twin environment, and 13760 frames for training and 1084 frames for validation from the real-world dataset. The frames were subsequently augmented to enhance generalization capability.

The collected data serves as input for both the YOLO segmentation and Mask Splitter pipelines employed for object segmentation and keypoint orientation reassignment. For the YOLO model, we annotated the data using polygons over 9302 frames for simulated environment and 8637 for the real one. After training the segmentation model, for the Mask Splitter we implemented a simple labeling tool which divides the car segmentation based on the user interaction. The algorithm first calculates the centroid of the car mask using image moments and then prompts the user to click around the front portion of the vehicle. Using the centroid as reference point, it creates a direction vector from the center to the user-selected front point and projects all mask pixels onto this vector using dot product calculations. Pixels with positive dot products lie in the same direction as the front point relative to the centroid and are assigned to the front mask, while the negative dot products are assigned to the back mask, effectively creating a linear split that separates the vehicle into front and rear segments.

4. Experiments

4.1. Experimental Setup

To verify our system’s performance, we systematically sampled 100 simulation runs for each viewpoint across eight distinct drone poses, yielding 800 total evaluations (see Figure 4). We applied both teacher and student methods to these drone poses, generating the same number of evaluations for each approach.

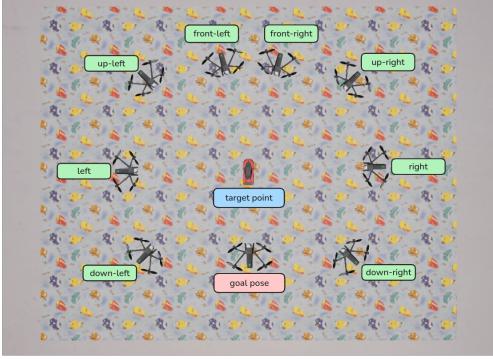


Figure 4. Environment configuration showing 8 starting poses around the target car for the flight tests. The drone must navigate from each start pose to the goal pose exactly behind the car, while keeping the car within the camera frame. The up-left, up-right, down-left and down-right poses are oriented at $\pm 45^\circ$ relative to the car, while front-left and front-right poses are at $\pm 14^\circ$.

The student model was trained on just 30 simulation runs across each of the presented drone poses, totaling 240 training run samples (74307 frames). We applied data augmentation using a multiplication factor of 5 (4 augmented + 1 original), creating synthetic diversity through randomized saturation, brightness shifts, and salt-and-pepper noise at each epoch while preserving one set of original frames. The velocity commands in the dataset were normalized across v_x , v_y , and ω_z according to the data distribution (Table 1), and images were resized to 224×224 pixels. Since our objective is to output velocity commands, this resizing does not affect the method’s performance. Training was performed using Adam optimizer with a learning rate of 0.001 and mean squared error as the loss function. We monitored validation loss using an early stopping mechanism with a patience of 3 and $\delta = 10^{-4}$.

4.2. Mission Termination Conditions

We implemented three termination conditions, all using frames of size 640×360 pixels. First, a hard timeout of 75 seconds triggered automatic landing if the drone failed to correct its pose within this time frame. Second, a soft condition was activated when the median of the last 5 errors was ≤ 80 pixels and all velocity commands remained at 0

for 3 consecutive seconds, indicating acceptable error levels but potential entrapment in a local minimum. Third, a hard condition is triggered when the median of the last 5 errors was ≤ 40 pixels for 3 consecutive seconds, signifying highly acceptable error levels where NSER IBVS generated minimal commands, producing negligible movement. The error is computed as the L2 norm for the vector that contains the raw pixel values.

To independently evaluate the student model, without requiring IBVS and the two neural networks (YOLOv11 and the Mask Splitter), we empirically derived the student termination condition by analyzing the statistics from teacher method evaluations. We examined the velocity command values within the last three seconds of each evaluation run. This analysis indicated that velocity commands were typically within the absolute range of 0 to 1 for most of our tests. Therefore, we established this as our termination condition for the simulated student: if the absolute value of the last 5 commands was ≤ 1 for 3 consecutive seconds, the drone would land and terminate the testing process.

Real-world testing used a similar approach but with reduced data samples. We conducted 10 flight sequences per pose (80 total evaluations, 43963 frames). The student model was pre-trained on 240 simulated runs, then fine-tuned on real-world data using the same termination conditions as the simulator.

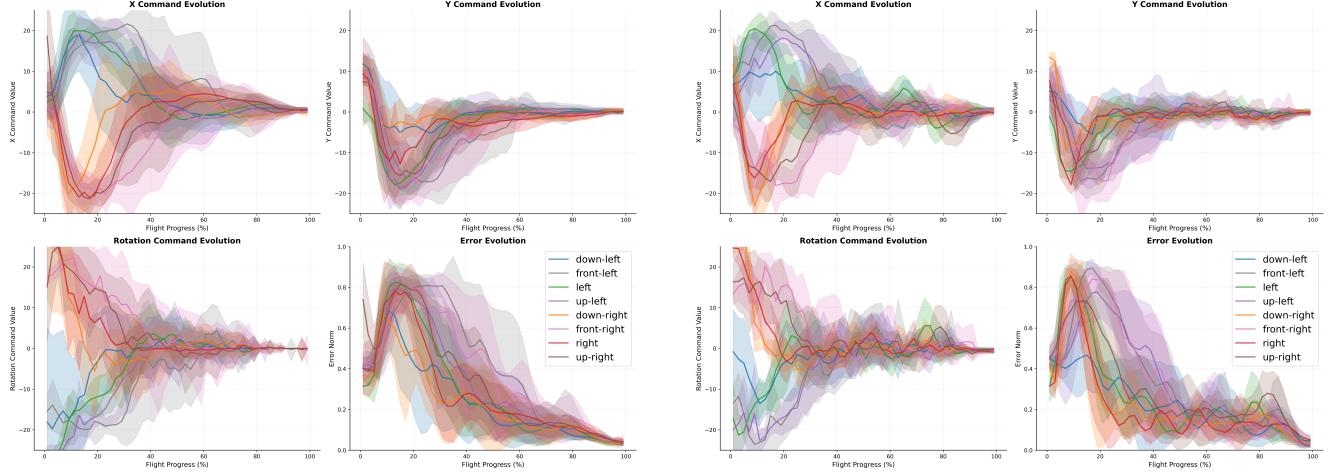
4.3. Evaluation Metrics

We compared the performance of teacher and student models using several metrics. The primary metrics were **error norm** and **Intersection over Union** (IoU) between the goal points and current points in the image during the last three seconds before landing. Furthermore, we measured the total **flight duration** and **distance** traveled from the point where tracking began (drone take-off with camera angled 45 degrees downward) until the point of landing. For the student method, these metrics were computed offline.

4.4. Results and Discussion

The results demonstrate that the student model achieves significantly improved tracking accuracy compared to the teacher method. In simulation, the student model achieves a mean error norm of 14.261 pixels compared to the teacher 29.756 pixels, representing a 52% improvement in tracking precision. The IoU metric shows similar improvements, with the student achieving 0.752 compared to the teacher 0.522, indicating better object coverage and localization 2.

The trajectory evolution analysis reveals that both methods successfully converge toward the target, but with different characteristics. The teacher method shows more conservative command profiles with gradual convergence, while the student method demonstrates more direct approaches with faster error reduction rates shown in Figure 6.



(a) Real-world analytical teacher (NSER IBVS): evolution of drone control commands and tracking errors across flight trajectory.

(b) Real-world small ConvNet student, which distills the analytical teacher: evolution of drone control commands and tracking errors across flight trajectory.

Figure 5. Comparison of drone control command and error evolution on novel test sequences for (a) teacher (NSER IBVS) and (b) student (Self-Supervised Neuro-Analytical) across flight trajectories from 8 different starting points (see Fig. 4). Solid lines represent mean values; shaded areas indicate variability across runs. All control commands and errors converge toward zero, indicating robust trajectory tracking and control convergence from various initial conditions. Note the striking similarity in drone control and distance error to target between the complex analytical teacher and the small ConvNet student.

		Flight SIM (distance(m) / time(s)) ↓	Norm Error SIM (px) ↓	IoU SIM ↑	Flight (distance(m) / time(s)) ↓	Norm Error (px) ↓	IoU ↑
Up-Left	Teacher	5.312 / 23.466	29.256	0.530	5.798 / 36.721	30.134	0.620
	Student	5.193 / 24.164	13.319	0.759	5.735 / 43.334	28.600	0.6263
Up-Right	Teacher	5.675 / 24.226	31.800	0.503	5.622 / 41.581	31.499	0.621
	Student	6.064 / 28.298	13.172	0.766	5.716 / 45.885	22.802	0.6919
Front-Left	Teacher	6.196 / 27.315	30.706	0.517	6.493 / 37.535	28.54	0.611
	Student	6.041 / 27.917	13.430	0.758	6.490 / 47.238	33.981	0.560
Front-Right	Teacher	6.846 / 32.358	32.608	0.488	6.197 / 37.166	33.15	0.658
	Student	7.043 / 35.535	18.028	0.718	6.316 / 46.363	31.035	0.627
Left	Teacher	4.177 / 20.228	31.453	0.519	4.559 / 29.921	28.015	0.629
	Student	4.089 / 20.481	13.243	0.762	5.065 / 43.841	30.853	0.6482
Right	Teacher	4.317 / 19.637	31.137	0.494	4.831 / 41.409	32.423	0.612
	Student	4.518 / 21.987	13.798	0.759	4.811 / 57.245	43.672	0.5
Down-Left	Teacher	2.779 / 15.988	28.473	0.518	4.384 / 31.622	28.00	0.611
	Student	2.777 / 14.900	13.257	0.763	4.326 / 41.044	39.531	0.5253
Down-Right	Teacher	2.893 / 13.667	22.618	0.606	4.137 / 33.523	27.89	0.654
	Student	3.145 / 17.035	15.839	0.728	3.938 / 38.001	36.195	0.5478
Mean	Teacher	4.774 / 22.111	29.756	0.522	5.253 / 36.185	29.956	0.627
	Student	4.859 / 23.790	14.261	0.752	5.300 / 45.369	33.334	0.591

Table 2. Teacher-student comparison across different starting positions. The left side shows results in the simulator; the right side shows results on flights in the real world. Metrics include total flight distance/time, final norm error in pixels (L2 norm of the error vector for all 4 corner points combined), and final IoU (L2 norm and IOU are computed in the last 3 secs of the flight). The teacher is slightly faster in flight time. Student is 11× faster in computation time (see Tab. 3). The student is slightly more accurate (lower errors at destination than the teacher) in simulation, where it was trained more, but it is slightly less accurate in real-world, where it was trained on far fewer flights.

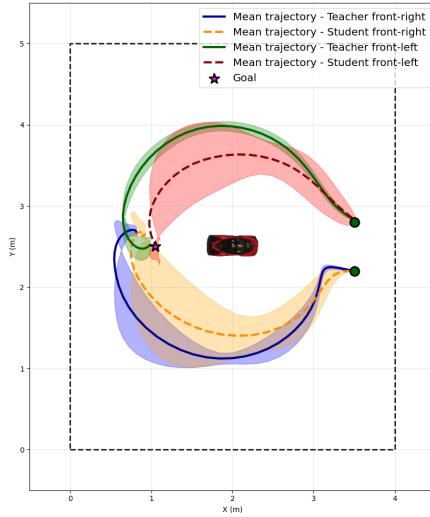


Figure 6. Trajectories of teacher and student simulation flights, with mean and standard deviation for 2 starting poses (green circles: front-left and front-right). The solid blue and green lines represent mean paths of teacher NSER IBVS method, while dashed orange and red lines show student paths. Shaded regions indicate trajectory variability across runs. The star represents the goal pose. Note that the student displays more path variation, but it has a shorter average path than the teacher.

Table 3. Computation times (in milliseconds) over 30 trials. The small 1.7M params student ConvNet is 11× faster than the teacher.

Evaluator	Avg	Std	Med	Min	Max	FPS
NSER IBVS	20.69	7.63	24.56	6.45	82.55	48.30
Student	1.85	0.93	1.84	1.79	235.64	540.8

To compare the computational efficiency of our proposed student network with and without segmentation against the baseline NSER IBVS, we conducted a runtime benchmarking experiment on a sequence of 640×360 resolution real-world frames. Each evaluator processed the same set of input frames independently, and we measured per-frame inference time over 30 repeated trials to account for variability due to system load and memory effects. For each trial, we performed warm-up iterations to avoid initialization bias and explicitly released memory between runs. We reported inference times in milliseconds and the FPS rate. The results are summarized in the Table 3.

The results show effective sim-to-real transfer, though with some performance degradation as expected. In real-world scenarios, the student model maintains competitive tracking accuracy while the teacher method shows more consistent performance across different environments. The fine-tuning approach using 80 real-world scenes proves effective for domain adaptation, enabling successful deployment in practical scenarios.

5. Final Remarks and Conclusions

This work introduces the first self-supervised neuro-analytic system for visual servoing of drones, through efficient knowledge distillation: a small ConvNet student learns from the proposed, numerically stable analytical teacher (NSER-IBVS) to fly the drone from different start poses to an end pose, which fully cover the relative pose difference on the trigonometric circle. The system is fully unsupervised, eliminating any human intervention, supervised pretraining or external sensors, unlike any other method in the literature. The student net, which is 1.7M parameters and easily deployable on embedded devices, is 11× computationally faster (529.77 FPS) than the teacher pathway (48.3 FPS), while remaining equally accurate and fast in flight, in both simulation and real world tests. Learning on real-world cases is limited to only a few flights (80), to further reduce the costs, through an efficient transfer between pretraining in the digital twin and the real-world scene.

Our novel two-stage segmentation pipeline, combining YOLOv11 with a specialized U-Net-based mask splitter, provides robust anterior-posterior vehicle segmentation without requiring explicit geometric models or fiducial markers. The self-supervised nature of this approach, validated through simulation-to-reality transfer learning and testing on a Parrot Anafi drone platform, demonstrates the practical feasibility of fully autonomous indoor operations with minimal hardware infrastructure requirements.

Experimental validation across distinct drone poses fully covering the trigonometric circle confirms the effectiveness of our unsupervised approach. While the student network exhibits slightly higher pixel errors in the real world (due to very limited real training flights) when compared to the teacher (mean error of 29.956 vs 33.334 pixels), it maintains acceptable IoU performance (0.627 vs 0.591) and achieves faster convergence times. Notably, in the digital twin where more extensive self-supervised training occurs, the system demonstrates superior convergence times with reduced errors, validating the cost-effectiveness of simulation-based training compared to expensive real-world data collection. Computational efficiency gains make our approach particularly valuable for real-time performance applications on resource-constrained platforms, establishing a new paradigm for autonomous visual servoing without human supervision or marker dependencies.

Acknowledgments: This work is supported by projects "Romanian Hub for Artificial Intelligence - HRIA", Smart Growth, Digitization and Financial Instruments Program, 2021-2027 (MySMIS No. 334906), European Health and Digital Executive Agency (HADEA) through DIGITWIN4CIUE (Grant No. 101084054), and "European Lighthouse of AI for Sustainability - ELIAS", Horizon Europe program (Grant No. 101120237).

References

- [1] Luis Arreola, A Montes De Oca, Alejandro Flores, J Sanchez, and Gerardo Flores. Improvement in the uav position estimation with low-cost gps, ins and vision-based system: Application to a quadrotor uav. In *2018 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1248–1254. IEEE, 2018. [1](#)
- [2] Sandeep Banik, Jinrae Kim, Naira Hovakimyan, Luca Carloni, John Thomas, and Nancy G Leveson. Integrating vision systems and stpa for robust landing and take-off in vtol aircraft. In *AIAA SCITECH 2025 Forum*, page 1324, 2025. [1](#)
- [3] Blender Online Community. Blender, 2024. [5](#)
- [4] Peter I Corke et al. *Visual Control of Robots: high-performance visual servoing*. Research Studies Press Taunton, UK, 1996. [1, 2](#)
- [5] Anton Domini Sta Cruz, Jayson Osayan, Luis Rafael Moreno, and Rob Christian Caduyac. Performance of aruco detector in aruco marker detection using non-gps drone. In *Advancing Sustainable Science and Technology for a Resilient Future*, pages 63–67. CRC Press, 2024. [1](#)
- [6] Quentin Galvane, Julien Fleureau, Francois-Louis Tariolle, and Philippe Guillotel. Automated cinematography with unmanned aerial vehicles. *arXiv preprint arXiv:1712.04353*, 2017. [1](#)
- [7] Nicolas Guenard, Tarek Hamel, and Robert Mahony. A practical visual servo control for an unmanned aerial vehicle. *IEEE Transactions on Robotics*, 24(2):331–340, 2008. [1](#)
- [8] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015. [4](#)
- [9] Seth Hutchinson, Gregory D Hager, and Peter I Corke. A tutorial on visual servo control. *IEEE transactions on robotics and automation*, 12(5):651–670, 1996. [1](#)
- [10] Michail Kalaitzakis, Brennan Cain, Sabrina Carroll, Anand Ambrosi, Camden Whitehead, and Nikolaos Vitzilaios. Fiducial markers for pose estimation: Overview, applications and experimental comparison of the artag, apriltag, aruco and stag markers. *Journal of Intelligent & Robotic Systems*, 101:1–26, 2021. [1](#)
- [11] Azarakhsh Keipour, Guilherme AS Pereira, Rogerio Bonatti, Rohit Garg, Puru Rastogi, Geetesh Dubey, and Sebastian Scherer. Visual servoing approach to autonomous uav landing on a moving vehicle. *Sensors*, 22(17):6549, 2022. [2](#)
- [12] Alexandros Kouris and Christos-Savvas Bouganis. Learning to fly by myself: A self-supervised cnn-based approach for autonomous navigation. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–9. IEEE, 2018. [1](#)
- [13] Yogesh Kumar, Bassam Pervez Shamsi, Sayan Basu Roy, and Sujit PB. Tracking a planar target using image-based visual servoing technique. *IEEE Transactions on Intelligent Vehicles*, 9(3):4362–4372, 2024. [2](#)
- [14] Daewon Lee, Hyon Lim, H Jin Kim, Youdan Kim, and Kie Jeong Seong. Adaptive image-based visual servoing for an underactuated quadrotor system. *Journal of Guidance, Control, and Dynamics*, 35(4):1335–1353, 2012. [2](#)
- [15] Weijie Li, Weicheng Huang, Yinhua Li, and Li Wang. Design of joint control system for target detection uav and ground vehicle based on yolo algorithm. In *Proceedings of the 2023 11th International Conference on Computer and Communications Management*, pages 70–75, 2023. [2](#)
- [16] Xiao Liang, Guodong Chen, Shirou Zhao, and Yiwei Xiu. Moving target tracking method for unmanned aerial vehicle/unmanned ground vehicle heterogeneous system based on apriltags. *Measurement and Control*, 53(3-4):427–440, 2020. [2](#)
- [17] Jingshu Liu and Yuan Li. An image based visual servo approach with deep learning for robotic manipulation. *arXiv preprint arXiv:1909.07727*, 2019. [2](#)
- [18] E. Marchand, F. Spindler, and F. Chaumette. Visp for visual servoing: a generic software platform with a wide class of robot control skills. *IEEE Robotics and Automation Magazine*, 12(4):40–52, 2005. [2](#)
- [19] Eslam Mohamed, Abdelrahman Shaker, Ahmad El-Sallab, and Mayada Hadhoud. Insta-yolo: Real-time instance segmentation. *arXiv preprint arXiv:2102.06777*, 2021. [2](#)
- [20] Eduard Mráz, Jozef Rodina, and Andrej Babinec. Using fiducial markers to improve localization of a drone. In *2020 23rd International Symposium on Measurement and Control in Robotics (ISMCR)*, pages 1–5. IEEE, 2020. [1](#)
- [21] Tobias Nägeli, Lukas Meier, Alexander Domahidi, Javier Alonso-Mora, and Otmar Hilliges. Real-time planning for automated multi-view drone cinematography. *ACM Transactions on Graphics (TOG)*, 36(4):1–10, 2017. [1](#)
- [22] Ozan Oktay, Jo Schlemper, Loic Le Folgoc, Matthew Lee, Matthias Heinrich, Kazunari Misawa, Kensaku Mori, Steven McDonagh, Nils Y Hammerla, Bernhard Kainz, Ben Glocker, and Daniel Rueckert. Attention u-net: Learning where to look for the pancreas, 2018. [3](#)
- [23] Edwin Olson. Apriltag: A robust and flexible visual fiducial system. In *2011 IEEE international conference on robotics and automation*, pages 3400–3407. IEEE, 2011. [1](#)
- [24] Parrot Drones SAS. Parrot Sphinx. Software, 2023. Simulation tool; Online; accessed 2025-06-20. [5](#)
- [25] Pranjali Pathre, Gunjan Gupta, M Nomaan Qureshi, Mandyam Brunda, Samarth Brahmbhatt, and K Madhava Krishna. Imagine2servo: Intelligent visual servoing with diffusion-driven goal generation for robotic tasks. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 13466–13472. IEEE, 2024. [2](#)
- [26] Chao Qin, Qiuyu Yu, HS Helson Go, and Hugh H-T Liu. Perception-aware image-based visual servoing of aggressive quadrotor uavs. *IEEE/ASME Transactions on Mechatronics*, 28(4):2020–2028, 2023. [2](#)
- [27] Chao Qin, Qiuyu Yu, H. S. Helson Go, and Hugh H.-T. Liu. Perception-aware image-based visual servoing of aggressive quadrotor uavs. *IEEE/ASME Transactions on Mechatronics*, 28(4):2020–2028, 2023. [2](#)
- [28] Sairam VC Rebabpragada, Pranoy Panda, and Vineeth N Balasubramanian. C2fdron: Coarse-to-fine drone-to-drone detection using vision transformer networks. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6627–6633. IEEE, 2024. [1](#)

- [29] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016. 2
- [30] Mohammad Faisal Riftiarraysyid, Ford Lumban Gaol, Haryono Soeparno, and Yulyani Arifin. Suitability of latest version of yolov11 in drone development studies. In *2024 7th International Seminar on Research of Information Technology and Intelligent Systems (ISRITI)*, pages 504–509. IEEE, 2024. 2
- [31] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015. 3
- [32] Zahra Samadikhoshkho and Michael G. Lipsett. Visual servoing for aerial vegetation sampling systems. *Drones*, 8(11), 2024. 2
- [33] Shayan Sepahvand, Niloufar Amiri, and Farrokh Janabi-Sharifi. Deep visual servoing of an aerial robot using key-point feature extraction. In *2025 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 37–43. IEEE, 2025. 2
- [34] Jingtao Sun, Yaonan Wang, and Danwei Wang. Towards real-world aerial vision guidance with categorical 6d pose tracker. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2025. 2
- [35] Jian Tang, Zhiyuan Deng, Hao Zhang, Tao Jiang, Chenyang Wang, and Zhiwu Ke. Control barrier function-based quadrotor interception control using visual servoing. In *2024 IEEE International Conference on Unmanned Systems (ICUS)*, pages 319–324. IEEE, 2024. 1
- [36] Jian Tang, Zhiyuan Deng, Hao Zhang, Tao Jiang, Chenyang Wang, and Zhiwu Ke. Visual servoing for quadrotor uavs: A survey. In *2024 IEEE International Conference on Unmanned Systems (ICUS)*, pages 325–330. IEEE, 2024. 2
- [37] Tencent Hunyuan3D Team. Hunyuan3d 2.0: Scaling diffusion models for high resolution textured 3d assets generation, 2025. 5
- [38] Celine Teuliére, Laurent Eck, and Eric Marchand. Chasing a moving target from a flying uav. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4929–4934. IEEE, 2011. 2
- [39] Lukas Wawrla, Omid Maghazei, and Torbjørn Netland. Applications of drones in warehouse operations. *Whitepaper. ETH Zurich, D-MTEC*, 212:1–12, 2019. 1
- [40] Kun Yang and Quan Quan. An autonomous intercept drone with image-based visual servo. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2230–2236. IEEE, 2020. 2
- [41] Kun Yang, Chenggang Bai, Zhikun She, and Quan Quan. High-speed interception multicopter control by image-based visual servoing. *IEEE Transactions on Control Systems Technology*, 2024. 2
- [42] Ziyi Yang, Xin Lan, and Hui Wang. Comparative analysis of yolo series algorithms for uav-based highway distress inspection: Performance and application insights. *Sensors*, 25(5):1475, 2025. 2
- [43] Changhao Zhang, Zengqi Xiu, Boxuan Hu, and Yuyi Pan. A vision-only drone formation waypoint navigation method. In *2024 4th International Conference on Industrial Automation, Robotics and Control Engineering (IARCE)*, pages 473–479. IEEE, 2024. 2
- [44] Yongwei Zhang, Yangguang Yu, Shengde Jia, and Xiangke Wang. Autonomous landing on ground target of uav by using image-based visual servo control. In *2017 36th Chinese Control Conference (CCC)*, pages 11204–11209. IEEE, 2017. 1
- [45] Hongyu Zhao, Zezhi Tang, Zhenhong Li, Yi Dong, Yuancheng Si, Mingyang Lu, and George Panoutsos. Real-time object detection and robotic manipulation for agriculture using a yolo-based learning approach. In *2024 IEEE International Conference on Industrial Technology (ICIT)*, pages 1–6. IEEE, 2024. 2