

CINE: Cinematic Intelligent Navigation Engine using B-spline trajectory generation and model-based perception

Daniel Kosukhin

William A. Shine Great Neck South High School
Great Neck, NY, USA
dkosukhintech@gmail.com

Abstract—Although autonomous unmanned aerial vehicle (UAV) cinematography has advanced significantly, existing systems force a trade-off between cinematic intent and computational tractability, because encoding aesthetic objectives into trajectory planning requires iterative optimization that combines the processing demands of collision avoidance and kinematic feasibility. Model Predictive Control (MPC) frameworks automate camera parameters but require global trajectory re-optimization, which is infeasible on the lightweight onboard processors that autonomous UAVs depend on for all computation during flight and produces no meaningful output when no object of focus is specified, since all extrinsic and intrinsic computations are defined relative to a tracked target. Gradient-based B-spline planners enable real-time replanning but optimize for execution time and collision avoidance, disregarding aesthetic constraints entirely. In this study, we present CINE, to our knowledge the first system to encode cinematographic intent as quantitative cost functions on a real-time B-spline planner designed to run entirely on edge hardware, achieved through the localized post-processing of control points rather than global trajectory re-optimization. CINE integrates Depth Anything 3 for monocular relative depth, YOLOv11n for object detection and rule-of-thirds framing, and a 2×128 multilayer perceptron (MLP) trained to mitigate visual-inertial odometry drift. Validation across 15 autonomous trials in kinematic simulation demonstrates that the MLP correction reduces mean RMSE positional error between estimated positions and setpoints from 0.25 m to 0.05 m and increases the proportion of corrected trajectory waypoints falling within a 10 cm positional tolerance from 25% to over 95%. Following domain-specific tuning on the HomeObjects-3K dataset, YOLOv11n reached a precision of approximately 0.71, recall of 0.60, mAP50 of 0.65, and mAP50-95 of 0.44, establishing sufficient detection accuracy for downstream framing constraints. Compared to baseline EGO-Planner trajectories, CINE reduces maximum object-relative depth deviation by 57% (0.72 m to 0.31 m), increases time within depth tolerance from 23.6% to 76.4%, and clamps peak velocity from 3.0 m/s to 1.2 m/s during active framing. Additionally, the perception and framing stage ran live on the physical platform, where YOLOv11n detection and the rule-of-thirds error computation executed on a Raspberry Pi 5 with a Coral Edge TPU and guided an operator who repositioned the grounded platform toward rule-of-thirds anchors, while in simulation the framework rotated the camera up to 110° in yaw to avoid detected light vectors. Ultimately, CINE provides a foundation for drone cinematography that is autonomous, accessible, and grounded in cinematic intent.

Index Terms—UAV cinematography, B-spline trajectory, model-based perception, edge computing, autonomous drones, rule-of-thirds, visual-inertial odometry

I. INTRODUCTION

A. Autonomous UAV cinematography forces a trade-off between cinematic intent and computational tractability

Unrestricted angular and vertical camera manipulation via unmanned aerial vehicles (UAVs) has become increasingly critical for modern aerial videography. Currently, the director of the shoot must coordinate with external pilots for access to cinematic navigation in any given environment and specialized sensing [1]. Pilots incorporate their understanding of cinematography and high-end kinematic adjustment technology, such as LiDAR combined with local obstacle avoidance algorithms and trajectory optimization algorithms that regulate proximity to surfaces and trajectory continuity, to not only achieve a visually pleasing effect similar to offline artistic planners [2], but also to preserve nearby safety. This approach relies heavily on manual piloting, introducing logistical bottlenecks and high costs due to the specialized equipment and expertise required [1]. Cost barriers create difficulties when coordinating shots, which often restricts directors from incorporating cinematographic techniques that require specific technology or pilot expertise. As a result, professional aerial cinematography remains accessible only to well-funded productions, locking out independent filmmakers, field journalists, and emerging creators.

To prevent holding back directors and to improve efficiency, CineMPC was developed, which couples model predictive control (MPC) with low-level **extrinsic** and **intrinsic** camera control [3], a hierarchical approach first formalized in the ‘Virtual Cinematographer’ paradigm [4]. **Extrinsics** refer to the position and orientation of the camera. **Intrinsics** refer to the internal parameters of the camera lens, such as focal length or optical center. The framework (as shown in Fig. 1) predicts drone kinematics and incorporates closed-loop feedback mechanisms to automate the extrinsics and intrinsics of the camera. Specifically, these values are computed relative to an object of focus and its surrounding background using YOLO, an object detection model that returns object bounding boxes [5]. Bounding boxes represent the object’s coordinate position relative to the camera. The approach improves intrinsic camera values by treating them as functions with velocity and acceleration using high-order derivation,

hence allowing for the incorporation of feedback loops into cameras rather than manual adjustment. Additionally, using the object's depth and the predicted camera outputs, extrinsics are calculated depending on the desired effect. Ultimately, both are encompassed by a Kalman filter, which iteratively fuses noisy sensor measurements with model predictions to refine estimates of camera position and orientation over time [6]. By doing so, CineMPC significantly improves the efficiency of cinematographic filming without compromising the accuracy of the recording and by reducing human input, allowing for pilotless flight. The framework completed shots roughly 2.77 times faster than professional pilots [3].

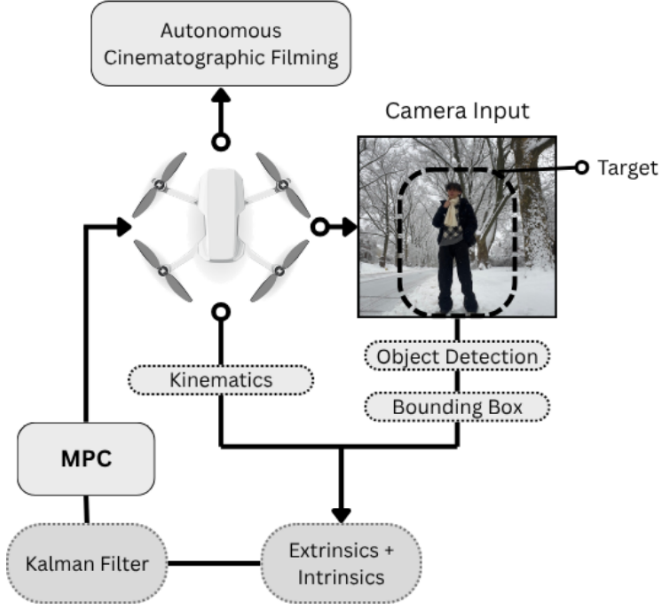


Fig. 1. Overview of the CineMPC autonomous cinematography pipeline. The system detects a target from the camera input, extracts a bounding box, and computes camera extrinsics and intrinsics relative to the object, which a Kalman filter fuses before the MPC loop issues closed-loop control to the drone for autonomous filming. Figure generated by student author using Canva in 2025.

Although CineMPC ultimately reduces dependence on professional pilots and greatly improves efficiency, the algorithm is ill-equipped for dynamic environment navigation without an object of focus.

Model predictive control reconfigures all previous waypoints at every planning cycle rather than adjusting segments independently, which means that the planner cannot incorporate live environmental changes, such as shifting lighting, new visible obstacles, or target movement, without recomputing the entire trajectory [7]. Furthermore, because each recomputation involves non-linear optimization over the full state space, the processing time exceeds what standard CPU-based onboard systems can deliver with a single control loop, making real-time planning unfeasible on edge hardware. Regarding extrinsic and intrinsic calculation, the framework collapses without a specified object of focus, since the algorithm was specifically designed to follow an object of interest with cinematic effect.

CineMPC takes minimal environment data into account [3], meaning that it has no general obstacle avoidance capabilities unless the obstructing object is recognized by the YOLO model.

Global trajectory reoptimization imposes a substantial computational burden, since MPC-based planners re-solve a constrained nonlinear optimization at every control step [3], [7]. Prior autonomous cinematography systems achieve real-time performance only by relying on external infrastructure or substantial onboard computing. Nageli et al. [8] demonstrated real-time planning over a receding horizon, but the system depended on an external motion capture system for pose estimation, restricting operation to instrumented spaces. Bonatti et al. [2], [9] achieved fully onboard real-time cinematography in unstructured environments, but required a LiDAR sensor, GPS tags, and an NVIDIA TX2 companion computer. Neither operates from a monocular camera on a low-power edge computer, which is the regime this study targets. Collectively, these efforts highlight a persistent gap, where no prior system achieves onboard, real-time trajectory optimization with explicit cinematographic constraints.

B. Existing approaches, specifically MPC and B-spline planners, each resolve only part of this trade-off

While early quadrotor navigation relied on minimum-snap trajectory generation [10], B-spline planners were introduced to address the computational shortcomings of global trajectory optimization by leveraging localized control and environmental sensing. Specifically, algorithms such as EGO-Planner control drones with a gradient of cost functions and environment depth mapping. Additionally, B-spline trajectories address MPC's shortcomings with a modular polynomial trajectory structure, which refers to a path composed of polynomial functions controlled by three properties. Firstly, the trajectory is constrained with the convex hull property, which states that the convex hull of the control points must be where the trajectory resides. Second, the knot vector controls the multiplicity of the trajectory by utilizing knots indexed by time. Finally, the control points of the spline itself allow for the modulation of local basis functions that make up the baseline of the spline. Because of these properties, B-splines possess local support, which states that

$$N(t) \text{ is affected by } P_i \Leftrightarrow t \in [t_i, t_{i+p+1}], \quad (1)$$

where $N(t)$ is the basis function, P_i is the control point at an index of i , t is the time parameter, and p is the degree of the B-spline function. This property is a necessity for real-time cinematographic planning [8]. Consequently, localized planners have the ability to adjust local segments of the trajectory and satisfy kinematic and dynamic constraints without the computational overhead of re-optimization.

C. A post-processing layer on B-spline trajectories can encode cinematographic principles as quantitative constraints

While B-splines introduce necessary features for dynamic navigation, standard planners optimize primarily for collision

avoidance and execution time, disregarding cinematic objectives. No prior system we identified encodes cinematographic intent as quantitative cost functions on a real-time B-spline planner designed for edge hardware, and this gap defines the contribution of this study. In this work, the perception and framing stage was validated on the target edge hardware against a live camera, while the trajectory-level cinematic results were validated in kinematic simulation. This study proposes a post-processing framework that adjusts EGO-Planner setpoints with cinematographic objectives. EGO-Planner produces a collision-free, dynamically feasible trajectory [11], [12] with depth and odometry inputs, which gets processed through a cinematic refinement step. By leveraging the local support property of B-splines, this step imposes cinematic constraints by adjusting nearby control points without triggering global replanning. CINE, the resulting system, integrates three components: model-based perception for environmental input, quantitative cinematographic analysis, and low-level correction for VIO drift.

D. Literature Review

Three prior systems define the closest comparison set. CineMPC [3] controls both camera extrinsics and intrinsics through a nonlinear MPC loop, re-optimizing the full trajectory at every timestep in response to scene changes. Because each step solves a constrained nonlinear optimization, the approach is computationally demanding, making real-time execution on a low-power onboard computer difficult. The framework also depends on a specified object of focus and does not perform general obstacle avoidance.

Bonatti et al. [2], [9] run fully onboard in real time but require LiDAR, GPS tags, and a TX2-class computer. Nægeli et al. [8] plan in real time but depend on an external motion-capture system for pose estimation. Among these, no system runs onboard, in real time, and respects cinematic intent.

Monocular depth estimation, once limited to coarse multi-scale convolutional architectures [13], has matured into models such as Depth Anything 3 [14], which predicts spatially consistent monocular depth from in-the-wild images at real-time rates while producing dense, fine-grained depth maps, with its smaller variants reaching real-time inference rates suitable for online use. Concurrently, object detection has converged on YOLO-family architectures: the nano-scale YOLOv11 model attains 39.5% mAP50-95 on the COCO benchmark with only 2.6 million parameters, running at roughly 1.5 ms per image on a T4 GPU and in real time on embedded accelerators [5]. Visual-inertial odometry, despite ongoing challenges in feature-sparse environments, now supports closed-loop pose estimation on quadrotors; recent work improves its robustness and accuracy under high-speed motion and low-texture conditions through IMU-based feature prediction and multi-camera fusion [15]. Together, these advances make it feasible to recover scene depth, object position, and camera pose from lightweight onboard sensing, motivating learned, low-cost corrections in place of expensive sensor upgrades.

These three threads converge to make CINE possible. The cinematographic gap identified in the first thread defines the problem; the local-support trajectory architecture from the second thread provides a planner that can accept localized adjustments without re-optimization; and the maturity of the perception primitives in the third thread supplies the real-time environmental signals on which cinematic constraints depend. By layering quantitative cinematographic objectives onto B-spline setpoints with model-based perception inputs, this study addresses the trade-off that prior systems could not resolve. The specific goals of this study are threefold:

- 1) To develop a post-processing framework designed for real-time operation, adjusting B-spline trajectory setpoints with cinematographic cost functions and using model-based perception for environmental input.
- 2) To validate the perception and error-correction pipeline, including object detection, monocular depth estimation, and MLP-based VIO drift correction, with the detection and framing stage executing on the target edge platform and the remaining stages evaluated in kinematic simulation.
- 3) To demonstrate that qualitative cinematographic principles, including depth stability, rule-of-thirds framing, glare avoidance, and motion-blur reduction, can be encoded as explicit mathematical constraints that meaningfully improve trajectory quality relative to time-optimized baselines.

II. FRAMEWORK ARCHITECTURE

A. CINE intercepts B-spline setpoints, applies cinematic constraints, and corrects for VIO drift before flight controller dispatch

This section details the CINE framework, a multi-stage post-processing architecture that enforces cinematic constraints on autonomous UAV trajectories. As illustrated in Fig. 2, the pipeline systematically intercepts generated trajectories, applies kinematic and spatial constraints, and outputs refined setpoints to the low-level flight controller.

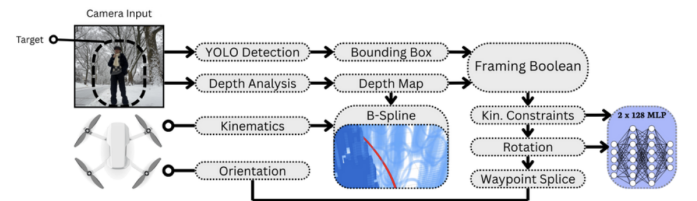


Fig. 2. Graphic visual of the multi-stage cinematographic adjustment algorithm. Figure generated by student author using Canva in 2025.

The framework operates in three sequential phases: first, a perception stack extracts environmental context through monocular depth estimation and object detection; second, a post-processing layer applies cinematic constraints to trajectory setpoints generated by EGO-Planner; and third, an MLP corrector compensates for VIO drift before setpoints are sent to the flight controller. Each phase is designed to execute within

the processing budget of a single-board companion computer, preserving the localized structure that B-spline generation provides rather than triggering global re-optimization.

The foundation of this framework relies on gradient-based local B-spline generation, initiated by EGO-Planner using odometry and depth inputs. For a degree- p uniform B-spline defined by a knot vector u , the trajectory position at time t is formulated as:

$$x(t) = \sum_{i=0}^n P_i N_{i,p}(t), \quad (2)$$

where P_i represents the control points and $N_{i,p}(t)$ are the B-spline basis functions. Using De Boor's algorithm [16], the framework extracts real-time position, velocity, and acceleration derivatives for localized constraint application.

Rather than executing global re-optimization, CINE modularizes this polynomial trajectory and adjusts local sections prior to execution. The environmental context required for these adjustments is provided by a perception stack. First, Depth Anything 3 processes monocular images to establish relative target depth. Concurrently, YOLOv11n extracts 2D bounding boxes to dictate screen-space framing. Finally, prior to dispatching the refined $x(t)$ setpoints via MAVLink, a 2×128 multilayer perceptron (MLP) applies spatial corrections to mitigate visual-inertial odometry (VIO) drift.

Through this pipeline, the framework sequentially enforces kinematic smoothing, orientation optimization for glare avoidance, and dynamic waypoint splicing to achieve visually pleasing, fully autonomous flight.

B. Three modular adjustment stages enforce depth, orientation, and kinematic constraints relative to the detected object

1) *Waypoint splicing maintains constant object-relative depth without global re-optimization:* To properly frame objects relative to the drone's pose, the planner uses **waypoint splicing**. **Waypoint splicing** removes stale trajectory setpoints and inserts dynamically generated waypoints to maintain a consistent depth relative to the target. Rather than overriding the global trajectory produced by EGO-Planner, this adjustment is applied locally to maintain computational efficiency. The splicing behavior is governed by a depth-based objective that encourages the drone to remain near a desired cinematic distance,

$$J_{depth} = (Z - d^*)^2 \quad (3)$$

where Z is the estimated depth of the object obtained from the depth analysis model and d^* is the desired focus distance. This objective promotes smooth orbit-like motion around the object while avoiding abrupt changes in relative distance. The planner only undergoes waypoint splicing when an object of focus is detected within a certain distance tolerance.

2) *Yaw and pitch are oriented toward the object of focus or away from detected light vectors:* In parallel with kinematic smoothing, CINE enforces orientation constraints to optimize aesthetic composition without strictly limiting translational freedom. Orientation control follows a simple two-stage logic. First, the planner publishes a boolean indicating whether the

drone is actively framing an object. When framing is active, the drone's yaw (and pitch, if necessary) is oriented toward the object of focus to maintain consistent visual emphasis. To encourage proper cinematic composition, the planner evaluates how far the detected object's screen-space position deviates from a desired rule-of-thirds anchor. This deviation is encoded as a framing cost,

$$J_{frame} = \|(u, v) - (u^*, v^*)\|^2 \quad (4)$$

where (u, v) is the detected object center in the image and (u^*, v^*) is the desired rule-of-thirds anchor point. Minimizing this cost encourages stable and visually balanced framing. When framing is inactive, or when strong illumination is detected, the planner instead prioritizes glare avoidance. This is achieved by penalizing alignment between the camera's forward direction and the estimated light direction, encouraging the drone to rotate away from visually disruptive light sources. Together, these orientation objectives enable both object-centric framing and glare-free navigation.

3) *Velocity, acceleration, and jerk penalties reduce motion blur during active framing:* Before visual refinement, the planner enforces kinematic smoothing and constraints to prevent large deviations from the expected trajectory and waypoint sequence, effectively intercepting setpoints sent to the low-level controller for post-processing. To minimize motion blur and ensure visual stability, the framework penalizes high velocity, acceleration, and jerk along the trajectory by minimizing the smoothing cost function:

$$J_{smooth} = \int (w_v \|x'(t)\|^2 + w_a \|x''(t)\|^2 + w_j \|x'''(t)\|^2) dt \quad (5)$$

In addition, when the drone is actively **framing** an object, the planner enforces a stricter velocity limit,

$$\|x'(t)\| \leq v_{max}^{cin} \quad (6)$$

which prioritizes visual stability over execution speed. **Framing** is defined as the segment of waypoints between a start anchor, where the drone begins emphasizing an object, and an end anchor, where framing concludes. During this interval, the planner biases the trajectory toward smooth, visually pleasing motion rather than time-optimal navigation.

III. METHODS

A. Hardware Configuration

The experimental UAV platform was constructed on a custom 5-inch TBS Source One frame. Low-level motor actuation was managed by a Matek H743-Mini flight controller paired with a SpeedyBee F405 35A ESC, while a Raspberry Pi 5 (8GB RAM) was selected as the central companion computer for onboard trajectory optimization and setpoint processing. Visual data was acquired via a wide-field-of-view Raspberry Pi Camera Module V3, configured with a grayscale filter to enhance visual-inertial odometry (VIO) contrast. To achieve the high inference rates required for real-time perception, all neural network processing was offloaded to a Google Coral

Edge TPU via USB 3.0. Power was supplied by 1300mAh 6S (22.2V) batteries, driving four EMAX ECO II Series 2207 1700KV motors with 3-blade propellers. This platform was physically built and used for the on-device framing test reported in Section IV; the trajectory results were generated in kinematic simulation modeling this configuration rather than on the platform itself.

B. Software Implementation and Model Training

The onboard software architecture was built on the Robot Operating System 2 (ROS2). Visual data acquired from the camera, filtered to grayscale to enhance feature tracking, was fused with IMU data to estimate UAV pose via visual-inertial odometry (VIO). Depth maps generated by Depth Anything 3 and object bounding boxes extracted by YOLOv11n were published as continuous ROS2 topics. These topics informed EGO-Planner’s gradient-based local B-spline generator. To mitigate systematic VIO drift, the planned state vectors $(x, y, z, \dot{x}, \dot{y}, \dot{z})$ were routed through the custom 2×128 MLP to compute positional corrections $(\Delta x, \Delta y, \Delta z)$. In the deployed design, the final, cinematographically refined setpoints are transmitted to the ArduPilot firmware at high frequency via MAVLink.

To optimize the perception stack for real-time indoor deployment, the models underwent household-specific fine-tuning. YOLOv11n was trained on the HomeObjects-3K dataset using an NVIDIA RTX 5080. Training was configured for up to 400 epochs with a batch size of 16, with early stopping halting training at 309 epochs, utilizing the AdamW optimizer to decouple weight decay from gradient updates [5]. The learning rate was dynamically modulated between 0.0025 and 0.01 via a cosine annealing scheduler. The 2×128 MLP was trained in two stages, both on simulated drift data from the kinematic simulation setup. An initial pretraining stage used residuals from a coarse drift model to establish a correction prior, and a second supervised stage refined the network on simulated trajectories with an 85%/15% train-validation split (4 batches per epoch, AdamW optimizer, mirrored cosine learning rate scheduler). The two stages were separated to stabilize convergence and prevent the refined weights from collapsing back toward the coarse prior. To meet the strict latency constraints of autonomous flight, all network weights were quantized to INT8 precision via TensorFlow Lite for hardware-accelerated inference on the Coral Edge TPU.

C. Experimental Setup and Validation Metrics

Validation was conducted across 15 autonomous trials in a kinematic simulation, in which EGO-Planner generated the trajectories with injected randomness and perturbations under ROS2 and RViz provided visualization; no physics engine was involved. The trial set was designed as a feasibility demonstration of the full pipeline targeting a single representative edge platform, rather than a generalization study across varying hardware or environments. A standing lamp positioned on a table served as the object of focus across all trials, placed

at a consistent location to allow direct comparison of depth-stability and framing metrics across trials. Each trial followed an identical pre-specified waypoint sequence beginning from a fixed starting pose, with a target duration of approximately 20 seconds per trial. Baseline trials using EGO-Planner alone and experimental trials using the full cinematographic adjustment framework were interleaved rather than run in separate batches, mitigating any ordering effects across the trial set. For glare-avoidance testing, an additional bright light source was introduced at a fixed location approximately 12 seconds into the trajectory to evaluate the framework’s orientation response to detected light vectors. The simulated vehicle state was reset between trials to ensure consistent initial conditions, and vehicle state and trajectory data were logged through ROS2 throughout each trial.

ID	Object Category	ID	Object Category
0	bed	6	laptop
1	sofa	7	wardrobe
2	chair	8	window
3	table	9	door
4	lamp	10	potted plant
5	tv	11	photo frame

12 categories · 2,285 training images · 404 validation images

Fig. 3. Household object categories used for YOLOv11n fine-tuning and validation. The twelve object classes from the HomeObjects-3K dataset are each assigned a numeric class ID (0–11) used during detection. The dataset comprises 2,285 training images and 404 validation images spanning common indoor furniture, fixtures, and electronics. Figure generated by student author using the Ultralytics HomeObjects-3K dataset in 2025.

During model testing, data analysis included the loss functions of the models. For YOLOv11n, these were dfl_loss , box_loss , and cls_loss [17]. For the 2×128 MLP, the primary loss was the mean squared error between predicted error and actual error [18]. Loss functions were modeled per step. The model was fine-tuned and validated on the twelve household object categories of the HomeObjects-3K dataset, listed in Fig. 3. Using cross-validation, confusion matrices were measured for the final fold. Precision, recall, F1-score, mAP50, and mAP50-95 were computed and modeled per epoch [17]. Precision is equivalent to

$$\text{Precision} = \frac{TP}{TP + FP}, \quad (7)$$

where TP is the number of accurate detections by the model and FP is the number of detections that were incorrectly classified as positive. Recall equals

$$\text{Recall} = \frac{TP}{TP + FN}, \quad (8)$$

where FN is the number of incorrectly identified detections identified as negative. F1 equals

$$F1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}, \quad (9)$$

which represents the harmonic mean of precision and recall. mAP50 is equivalent to

$$\text{mAP}_{50} = \frac{1}{N} \sum_{i=1}^N AP_{50,i}, \quad (10)$$

where N is the total number of categories in the dataset and $AP_{50,i}$ is the average precision for class i at an IoU threshold of 0.5, which quantifies the detection accuracy given an overlap condition. mAP50-95 is equivalent to

$$\text{mAP}_{50-95} = \frac{1}{NK} \sum_{i=1}^N \sum_{j=1}^K AP_{i,j}, \quad (11)$$

where K is the number of IoU thresholds considered (ranging from 0.5 to 0.95 with a step of 0.05) and $AP_{i,j}$ is the average precision for class i at a specific IoU threshold of j . Drift (meters) graphs for both VIO and VIO + MLP, where VIO is the control group, were measured over time (seconds). An initial trajectory in an empty and safe environment was graphed and compared to an optimized trajectory, which assumes the initial trajectory as the control group. Position and velocity were measured on the x, y, and z axes (meters), as well as the 3-dimensional graph of the trajectory within its environment, measured over time in seconds. Control points and waypoints help visualize the trajectory as well. The rotation (yaw, pitch, roll), in radians, of the drone was measured throughout the trajectory to show filming after light vector analysis. This was compared to the rotation of the drone without light consideration. The bounding box relative to the grid was measured (pixels), which was compared to the bounding box without the rule of thirds. The depth of field relative to the object of focus was modeled to visualize adjustments after object detection with different depths. Velocity was also modeled when an object of focus is detected by YOLO. Initial validation and MLP training were performed in kinematic simulation, and all 15 validation trials reported here were conducted in the same simulation environment. All comparative results are reported as means with standard deviation across the 15 interleaved trials; no formal significance testing was performed.

IV. RESULTS

A. The perception and error-correction pipeline achieved high prediction accuracy and reduced VIO drift

The results are organized to reflect the pipeline's logical dependencies. Because all downstream cinematic adjustments require successfully identifying and framing the object of interest, the perception and error-correction pipeline is validated first. Object detection accuracy and VIO drift correction establish the foundation on which trajectory evaluations rest. With the perception pipeline validated, the cinematographic

adjustment layer is then evaluated for measurable improvements in depth stability and velocity control relative to baseline EGO-Planner trajectories. Finally, the extrinsic and intrinsic composition constraints are assessed, including rule-of-thirds framing, glare avoidance, and velocity clamping during active filming.

1) *Targeted training on indoor data improved YOLO accuracy over the pretrained baseline:* The cinematographic adjustment framework is powered by the YOLOv11n model's ability to extract bounding box coordinates from its environment by extracting image features and accurately predicting the object's 2-dimensional position. Although YOLO models are pretrained and have capabilities generalized towards all objects, weights trained for household and indoor object detection were hypothesized to provide more accurate prediction results. In order to validate the model's sufficiency, sufficiency metrics and dynamic learning rate adjustment were modeled throughout training using Ultralytics' training framework [17]. Several sufficiency metrics were collected with Ultralytics' logging framework during cross-validation after each epoch: precision, recall, F1 score, mAP50, and mAP50-95. To validate the sufficiency of the model after training, the following metrics were analyzed: precision and recall (Fig. 4b); the F1 score, which is the harmonic mean of precision and recall (Fig. 4a); mean average precision relative to the bounding box, which is the area under the precision-recall graph at 50 and 50-95 IoU tolerances (Fig. 4c); and the loss functions, which include classification loss, box loss, and distribution focal loss (Fig. 4d). Metrics were retrieved with cross-validation by splitting the dataset into folds and training the model with unknown data, allowing the model to return accurate evaluations of its performance [5]. After training, the precision and recall graphs reached up to approximately 71% and 60%, respectively, and eventually plateaued with a saturating curve. When the model weights reached a level of sufficient adjustment, a metric plateau was observed from each graph.

To improve the likelihood of the success of the model during training, a cosine learning rate scheduler was used to stabilize the activation gradient for optimal and healthy learning. The scheduler dynamically changes the learning rate of three parameter groups, each one designated towards a specific subset of model weights. Specifically, parameter group 0 (Fig. 5a) controls the learning rate of convolutional layer weights, parameter group 1 (Fig. 5b) controls the bias weights, and parameter group 2 (Fig. 5c) controls batch normalization weights. Learning rate was recorded after a training cycle during each epoch.

2) *Confusion matrices showed concentrated true positives along the diagonal:* In order to successfully classify objects in an indoor environment, the YOLOv11n model must exhibit high accuracy for correct adjustment during cinematographic filming. The accuracy of the model refers to the proportion of correct classifications relative to total detections. Correct classifications included true positives, where the model accurately identified an object, and true negatives, where the model accurately identified a background feature. After completing

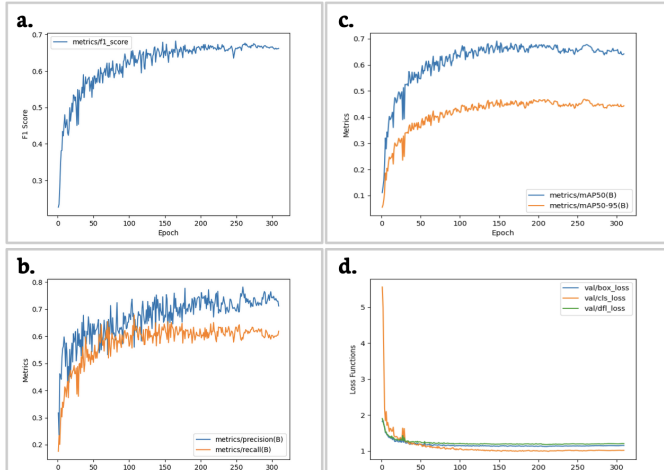


Fig. 4. Sufficiency metrics of the YOLO model during training demonstrating sufficient learning. (a) F1 score per epoch; (b) precision and recall curves per epoch; (c) mean average precision relative to the bounding box at IoU tolerances of 0.50 and 0.50–0.95 per epoch; (d) classification, box, and distribution focal loss per epoch. Metrics recorded with cross-validation after each epoch. Figure generated by student author using Matplotlib in 2025.

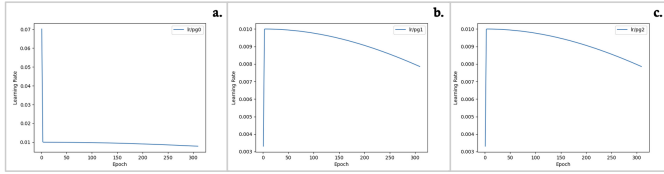


Fig. 5. The learning rates of the parameter groups under a cosine learning rate scheduler after each training cycle, for stable activation gradients. The learning rate of parameter group 0 (Fig. 5a) modulates the convolutional layer weights. Parameter groups 1 and 2 (Fig. 5b and 5c) modulate the bias and batch normalization weights, respectively. Figure generated by student author using Matplotlib in 2025.

all epochs, confusion matrices were generated after the final validation fold with the Ultralytics’ framework with regular and normalized detection data. A confusion matrix models the model’s number of true positives; true negatives; false negatives, objects the model classified as background; and false positives, background features that the model detected as an object. Row-normalized values were plotted to identify detection trends across object categories (Fig. 6a). Per-class recall averaged 67.7% across the twelve categories, ranging from 0.82 for photo frame and 0.81 for sofa down to 0.55 for bed, wardrobe, and door. The model performed strongest on visually distinct categories, including photo frame (0.82), sofa (0.81), table (0.77), and laptop (0.75), while categories with sparse or thin visual profiles were weakest: lamp, door, window, and wardrobe each missed between 34% and 40% of their true instances to the background class. The most frequent inter-class confusion was a bed misclassified as a sofa, which accounted for 41% of true bed instances and reflects their shared low, rectangular silhouette. Across all of the classes, an average of 23.7% of true objects were missed as background, indicating that missed detections rather than inter-class confusion were the dominant mode of error. These results

confirm sufficient accuracy for bounding box extraction while exposing reduced reliability for thin or low-contrast objects in cluttered indoor environments.

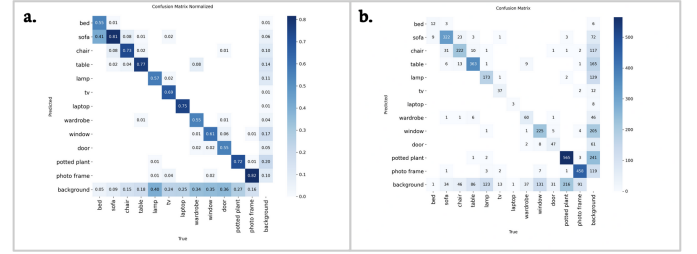


Fig. 6. Model accuracy for household objects modeled with a confusion matrix collected during cross-validation. (a) row-normalized confusion matrix and (b) confusion matrix without normalization after cross-validation. True positives appear on the principal diagonal; the right column holds false negatives (true object, predicted background); the bottom row holds false positives (true background, predicted object). The concentrated true positives in the normalized confusion matrix exemplify high statistical accuracy. Figure generated by student author using the Ultralytics training framework in 2025.

3) *MLP training reduced VIO RMSE from 0.25 m to 0.05 m and brought 95% of setpoints within 10 cm tolerance:* To validate the efficacy of the custom 2×128 MLP in mitigating visual-inertial odometry (VIO) drift, training convergence and spatial tolerances were analyzed over 300 epochs. As depicted in the training results (Fig. 7), the network demonstrated stable convergence without overfitting, with the overall validation root mean square error (RMSE) dropping steeply from an initial 0.25 meters to a stabilized 0.05 meters. Most notably, the model significantly improved spatial reliability during simulated flight. The percentage of corrected trajectory setpoints falling within a strict 10-centimeter tolerance increased from approximately 25% prior to training to over 95% upon convergence (Fig. 8). Furthermore, the maximum positional error experienced during execution was reduced from 0.80 meters to just 0.23 meters. These metrics confirm the MLP’s ability to successfully map noisy VIO state estimates to highly accurate, sub-decimeter positional corrections prior to flight controller dispatch.

B. Cinematographic adjustment stabilized depth and clamped peak velocity relative to the EGO-Planner baseline

1) *Maximum depth deviation dropped 57%, and time within depth tolerance more than tripled:* Although B-spline trajectory generation with EGO-Planner emphasizes environment navigation, it suffers when used for filming purposes, since it optimizes the time of execution rather than emphasizing features of the environment. As a result, EGO-Planner was compared with the proposed algorithm’s trajectories after processing. Using ROS2’s topic publishing system, trajectory positions after level 1 processing as the control group (EGO-Planner, Fig. 9a) and level 2 processing as the experimental group (cinematographic adjustment, Fig. 9b) were recorded at the drone’s position in time during 15 simulated trials. Both trajectories were tested with a given object, specifically a lamp on a table, to test the algorithm’s generation in reaction to an environment object.

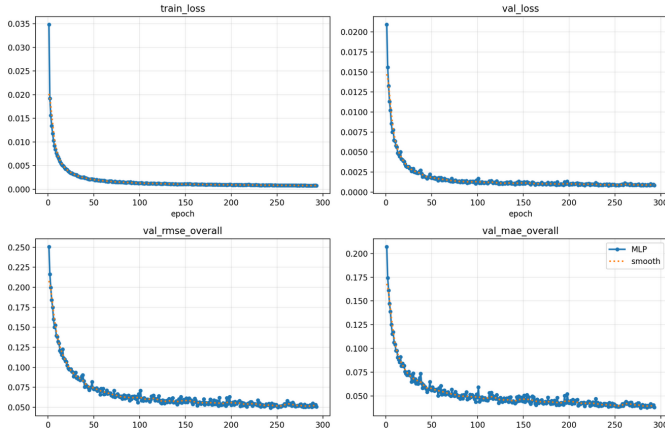


Fig. 7. Multilayer perceptron training demonstrates stable convergence and error reduction. The graphs illustrate the training loss, validation loss, and overall root mean square error (RMSE) over 300 epochs, showing a steep drop to a stabilized 0.05 meter error without overfitting. Figure generated by student author using Matplotlib in 2025.

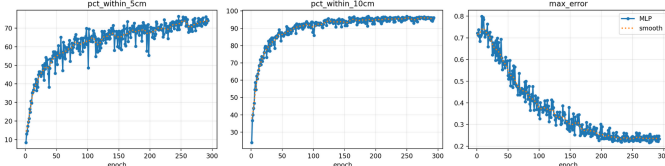


Fig. 8. Spatial tolerance metrics indicate reliable positional correction. The subplots track the percentage of corrected trajectory setpoints within 5-centimeter and 10-centimeter tolerances, alongside the maximum positional error over 300 epochs. Accuracy within a 10-centimeter tolerance exceeds 95% upon convergence. Figure generated by student author using Matplotlib in 2025.

When analyzing EGO-Planner's trajectory with no external adjustments, the planner simply wraps around an object without considering its presence relative to its surroundings. Additionally, the planner disregards controlled motion for optimal execution times [11]. Compared to EGO-Planner, the cinematographically adjusted trajectory features object avoidance with framing, which keeps the depth of the detected object constant while circulating around it. To validate the consistency of the object depth, the depth of the object relative to

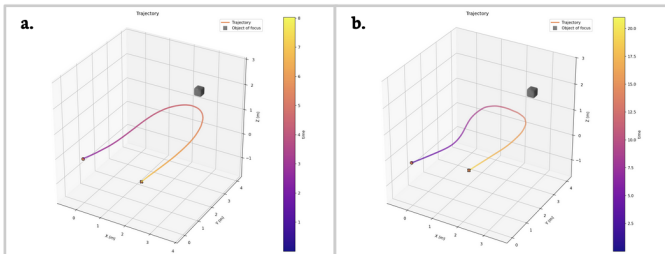


Fig. 9. Drone position trajectories aggregated across 15 simulated trials. (a) shows the trajectory under EGO-Planner alone, and (b) shows the trajectory under the cinematographic adjustment algorithm. The black sphere indicates the object of focus, and trajectory color represents elapsed time. Generated by student author using Matplotlib in 2025.

the drone was analyzed during framing with a ROS2 distance topic. The depth of the drone was compared in two different statistical analysis metrics: the max deviation from the desired depth for consistent distance and object framing (Fig. 10a), as well as the percentage of time points the planners spent within a 0.1 meter distance tolerance (Fig. 10b). The cinematographic adjustment reduced maximum depth deviation from approximately 0.72 m under EGO-Planner to 0.31 m, a reduction of 57% relative to the baseline. Additionally, the time spent within a 0.1 m depth tolerance increased from 23.6% under EGO-Planner to 76.4% under the cinematographic adjustment, more than tripling the baseline.

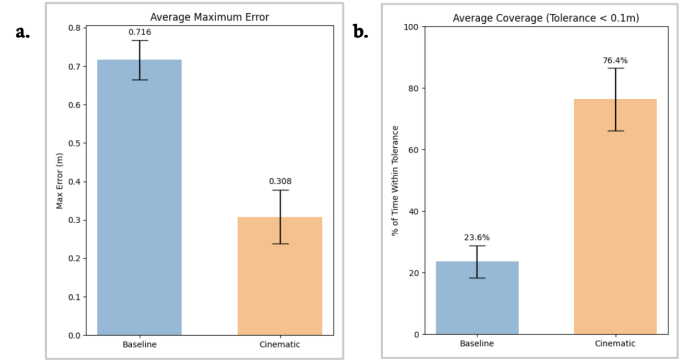


Fig. 10. Depth performance comparison between the EGO-Planner baseline and the cinematographic adjustment algorithm across 15 simulated trials. The blue column represents EGO-Planner's statistics and the orange column represents the cinematographic adjustment. (a) shows the maximum object depth error, which decreased from 0.72 m under baseline to 0.31 m, a 57% reduction. (b) shows the percentage of trajectory time spent within a 0.1 m depth tolerance, which increased from 23.6% to 76.4% across the 15 trials. Error bars represent ± 1 SD across the 15 trials. Generated by student author using Matplotlib in 2025.

Furthermore, time-optimized trajectories such as the B-spline feature less strict velocity and acceleration constraints during flight. To compare position and velocity groups, the planner used ROS2 and De Boor's algorithm for kinematic publishing. De Boor's algorithm computes the position and high-order derivatives of a B-spline trajectory. Position and velocity functions were statistically analyzed using standard deviation to suggest the planner's ability to optimize and constrain trajectories without scattered values. The position functions of the drone with EGO-Planner's trajectory (Fig. 11a) exemplify curves with unstable positioning. The velocity functions have significantly higher accelerations compared to the cinematographic agent when unconstrained (Fig. 11b). Unlike EGO-Planner's kinematics, the position in the z-axis is specifically smoothed out and kept relatively constant in the presence of an object of focus. Finally, the velocity functions of the cinematographic planner are significantly smoother, featuring object-based constraints and perturbations when an object of focus is detected. However, the cinematic velocities show overall smoothness but relatively small velocity changes.

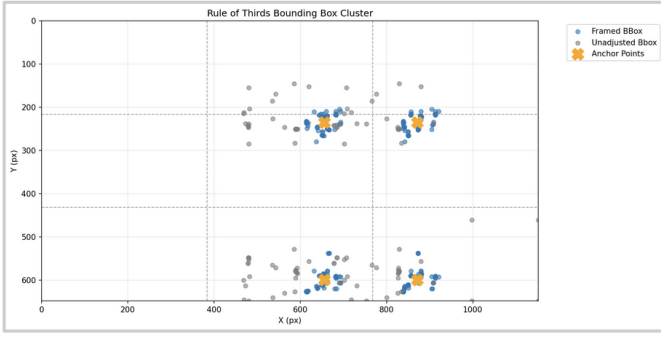


Fig. 11. Position and velocity functions across 15 simulated trials. (a) and (b) show baseline position and velocity under EGO-Planner. (c) and (d) show position and velocity under the cinematographic adjustment, with the z-axis clamped and velocity peaks dampened during framing. Shaded regions represent ± 1 SD across the 15 trials. Generated by student author using Matplotlib in 2025.

C. Extrinsic and intrinsic adjustments enforced cinematic composition in simulation and guided human-in-the-loop framing on device

1) *Bounding boxes clustered around rule-of-thirds anchors during framing:* The optimization algorithm extracts bounding box vertices and frames them relative to a rule-of-thirds grid, which allows for framework stability and adjustment using depth values and other intrinsic parameters. Unlike the trajectory results, this stage was validated on the physical platform itself. The drone did not fly during this test; the operator hand-held the grounded platform while the live camera feed, YOLOv11n detection, and the rule-of-thirds error computation ran onboard the Raspberry Pi 5 and Coral Edge TPU. The system issued a directional correction toward the target anchor, and the operator manually repositioned the platform to follow it; the platform did not actuate itself. Bounding box coordinates were recorded with and without the framing constraint, and a ROS2 bounding box node was recorded in pixel space for data collection. When analyzing the unframed control coordinates, the bounding box center exhibited a mean deviation of 127.6 pixels from the nearest rule-of-thirds anchor, with a standard deviation of 64.2 pixels across 33 recorded frames. After applying the framing constraint, the mean deviation fell to 32.5 pixels with a standard deviation of 9.8 pixels across 66 frames, a 74.5% reduction in mean deviation and a 97.7% reduction in positional variance. The framing condition yielded more recorded frames than the unframed control because the operator, following the system's corrections, kept the object within view, whereas the unframed control only incidentally captured the object in frame. All framed coordinates fell within 50 pixels of the target anchor, compared to only 3% of unframed coordinates. These values demonstrate a working on-device loop from detection to framing error to corrective guidance, with a human actuator closing it; they serve as a qualitative demonstration rather than a quantitative measure of the constraint layer. Because the operator had knowledge of the target anchors, these values cannot isolate the constraint

layer's contribution from operator compliance.

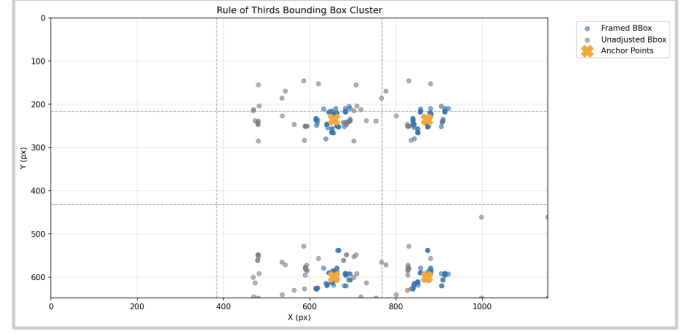


Fig. 12. Bounding box coordinate points relative to four rule-of-thirds anchors, collected from real camera frames during on-device bench testing with human-in-the-loop framing (33 unframed and 66 framed frames); the operator repositioned the grounded drone to follow the system's corrections. Coordinates were collected through a ROS2 bounding box vertices topic. Gray dots represent the bounding box coordinates without framing. Blue dots represent the framed bounding box coordinates. Orange Xs indicate the anchor points. Generated by student author using Matplotlib in 2025.

2) *Detected light vectors triggered yaw rotations of up to 110°:* The proposed approach also considers disruptions to the visual film itself, such as bright light from windows that puts the video at risk by blocking vision. Specifically, the drone rotates away from light and bright sources. To analyze the framework's ability to do so, the yaw, pitch, and roll were recorded throughout the 15 trajectories, as well as their orientation setpoints. The orientation was recorded using estimated odometry topics, which are published through ROS2. At about 12 seconds into the trajectory, a bright light vector is calculated and avoided by fully rotating away from it, especially in the yaw axis. Specifically, yaw is seen to rotate about 110 degrees to avoid film glare. Pitch is also seen to increase from 0 to 20 degrees after light detection. The gimbal angle remained near-constant throughout the trajectory.

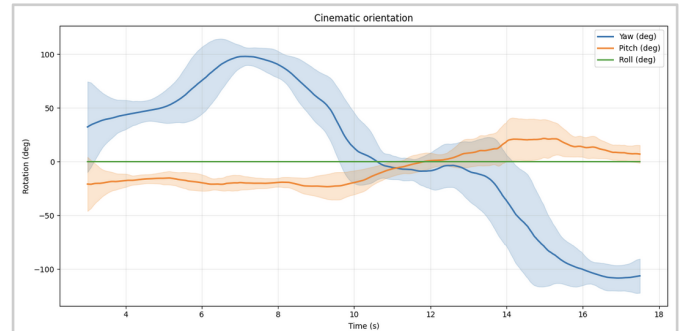


Fig. 13. Yaw, pitch, and roll functions modeled throughout the trajectory in the presence of a detected light vector, aggregated across 15 simulated trials. Orientation was recorded from ROS2 odometry topics. Dotted lines represent the orientation setpoints. The blue curve represents yaw (Z-axis), the yellow curve represents pitch (Y-axis), and the gray curve represents roll (X-axis). Shaded regions represent ± 1 SD across the 15 trials. Generated by student author using Matplotlib in 2025.

3) *Velocity dampening clamped peak framing speed from 3.0 m/s to 1.2 m/s:* To validate the framework's ability

to constrain velocities, the velocity with EGO-Planner and the adjustment algorithm were compared. Average speeds, in meters per second, throughout 15 simulated trials were collected with De Boor derivation and ROS2 subscriptions. Given an object of focus when generating a trajectory, EGO-Planner disregards framing and simply passes the object without changing velocities, but the proposed approach takes it into account and significantly constricts its velocity relative to its projected trajectory. Specifically, the unconstrained velocity goes to about 3 m/s in 3 seconds, while the constrained velocity function stays under 1.2 m/s at about 3.5 seconds and reaches a lower velocity during framing. Finally, the velocity function always stays under or hovers around 1.2 m/s due to constraints.

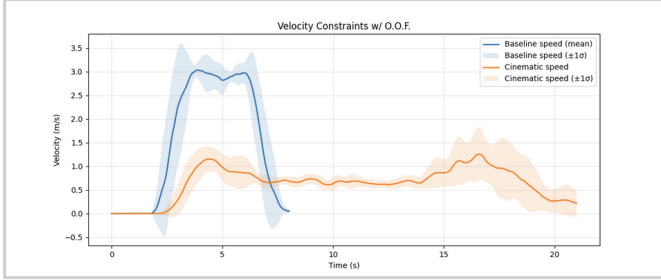


Fig. 14. Mean velocities plotted throughout the trajectory with and without cinematic constraints, aggregated across 15 simulated trials. The orange curve represents the velocity under cinematic constraints with active framing, and the blue curve represents the unconstrained baseline velocity. Shaded regions represent ± 1 SD across the 15 trials. Generated by student author using Matplotlib in 2025.

V. DISCUSSION

Although autonomous UAV cinematography has historically required either expert pilots or computationally heavy MPC frameworks, this study demonstrates that cinematographic principles can be encoded as quantitative cost functions on a real-time B-spline planner.

A. Domain-specific YOLO training was effective but mAP50-95 remained limited at strict IoU thresholds

Currently, the field of object detection with YOLO models inherits a training framework that emphasizes model generalizability, making the model universally viable. However, these pre-trained CNNs struggle to detect objects of specificity within a specific study or use case due to untailored weights [17]. Consequently, pre-trained models underperform in contexts such as indoor object identification. This study aimed to train a model for prediction sufficiency and accuracy for viable navigation during flight.

Using a training framework that validates the model with a cross-validation approach, the model was trained for 309 epochs to prevent overfitting and retain sufficient accuracy. After training, a healthy saturating curve was observed from the sufficiency metric plots, indicating that the model successfully identified objects throughout its environment with high accuracy and a positive learning curve (Fig. 4). Additionally,

the model training process ended early with a plateau at the end of the curve (initially set to 400 epochs), which signifies no overfitting during training and proper learning rather than memorization. Finally, the sufficiency metric curves signify proper training and promising results, but the mAP50 scores hovered around 0.65 and the mAP50-95 scores averaged 0.44 (Fig. 4c), indicating that the model struggles to accurately perform when tested with higher tolerances. This underperformance can cause issues in future flights or tests. Together, these observations indicate that while retraining for indoor environments improves detection reliability, higher IoU tolerances remain a limiting factor for downstream trajectory refinement.

After significant training and improvements in the weight values, the framework returned a confusion matrix that indicated the predicted and ground-truth values of the model. By comparing the predicted values to the ground-truth, the accuracy of the model can be determined with matrix normalization (Fig. 6a). The model indicated a concentrated pattern, specifically in the true positive diagonal, which shows the model's ability to predict objects in the environment accurately with minimal errors. False positives and false negatives were still observed though, with some matching the percentage of true positive predictions. This limitation can be observed for the wardrobe compared to the sofa, which had minimal false negatives or positives. As a result, the model suffers in environments that lack objects with high prediction accuracies.

Finally, a cosine learning rate scheduler was incorporated into the model's training to improve convergence for batch normalization and bias weights, as well as a stable gradient for convolutional layer weights (Fig. 5). The learning rate collected during training for all three parameter groups supports the healthy weight adjustment seen in the sufficiency metric curves.

These results demonstrate the benefits of training YOLOv11n with a weight adjustment and cross-validation framework for indoor applicability. Significant improvements in recall and precision can be observed in Fig. 4, as well as concentrated true positive predictions in Fig. 6a, validating the framework's feasibility for bounding box prediction within indoor environments.

B. Localized cinematic adjustments avoided global re-optimization while delivering measurable cinematographic gains

Existing systems for navigation with cinematographic intent rely on Model Predictive Control (MPC) to control camera intrinsics and extrinsics, which is computationally rigid and inefficient on edge platforms [3], [7]. The MPC planner globally recalculates the trajectory when flying, which makes navigation on constrained processors difficult and often unfeasible. Additionally, a lack of processing efficiency with MPC due to non-linear optimization leads to restrictions for autonomous flight without necessary computing, rendering flight within tight environments difficult. Although B-spline resolves this issue for tight navigation contexts, the generation system still

lacks cinematic intent. As a result, the study proposes a planner adjustment framework that incorporates cinematic intent into the trajectory at runtime. This improvement arises from localized adjustments to existing B-spline control points rather than globally replanning trajectories, allowing cinematic constraints to influence motion without incurring the computational cost of full trajectory optimization.

Visualizations of the path’s estimated trajectory throughout 15 simulated trials suggest the planner’s ability to navigate the environment with obstacle avoidance (Fig. 9). The trajectory shown in Fig. 9a, which showcases EGO-Planner’s approach to navigating the given waypoints, prioritizes dynamic planning without significant time loss. Specifically, the planner attempts to solve the time execution and constraint problem without sacrificing dynamic feasibility. EGO-Planner simply avoids the object without maintaining significant distance to ensure time execution reductions [11], meaning that the planner lacks cinematic considerations. However, the proposed framework preserves cinematic principles by reducing the planner’s emphasis on the time execution problem. Instead, the trajectory of the planner (Fig. 9b) exemplifies object framing relative to the environment. Fig. 10 suggests that the planner circulates the object for cinematographic framing behavior and keeps a stable depth to correctly visualize the environment with an emphasis on the object of focus. Maintaining a consistent object-relative depth stabilizes perceived object scale and reduces visual jitter, which is essential for producing smooth orbiting and tracking shots in autonomous cinematography. However, because depth was estimated from monocular imagery, the perception pipeline remained subject to scale ambiguity and sensitivity to lighting conditions.

EGO-Planner reports planning times on the order of 1 ms per cycle [11]. Because CINE adjusts existing control points locally rather than triggering global re-optimization, it adds only localized operations on top of this baseline; the on-device latency of the full post-processing loop was not measured in this study and is identified as future work. Regarding kinematic adjustment, both planners were analyzed in simulation to test **stability** and **motion blur**. **Stability** refers to a stable z-axis position, and **motion blur** in an extrinsic context refers to high accelerations and unclamped velocities during object framing. EGO-Planner’s kinematics (Fig. 11a and 11b) expose insufficient stability due to a continually changing z-axis position and unconstrained velocities, ultimately moving past the object without consideration for relative motion and position. Regarding velocity, our baseline trials reached peak velocities exceeding 3 m/s, and the cinematographic adjustment reduced peak framing velocity to 1.2 m/s, which translates to a 60% reduction relative to the unconstrained baseline. Additionally, the velocities present challenges to motion blur because of unconstrained values and significant bumps that reduce smoothness. Compared to EGO-Planner, the proposed framework stabilizes the drone position by clamping the z-axis at a consistent value (Fig. 11c), supporting object framing without disruptive perturbations. The planner emphasizes gradualism in positional change to prevent

motion blur by dampening the velocities and clamping their values (Fig. 11d), providing a cinematographically constrained trajectory that balances smoothness and feasible dynamics. Finally, in order to constrain the velocity further and prevent significant movement in all axes, a velocity clamp was applied during object framing (Fig. 14).

These results signify that the framework is able to cinematically restrict kinematics, specifically positional and velocity values, to alter visual characteristics of the resulting film. Additionally, the framework frames objects relative to a rule-of-thirds grid with stable depth, effectively highlighting the object relative to its environment.

Quantitative comparison to prior cinematographic systems further contextualizes these gains. CineMPC reported a 2.77 times faster shot execution time relative to professional pilots [3]. By contrast, CINE achieves comparable cinematic stability in kinematic simulation, measured through depth deviation reduction and tolerance time improvements, while its detection and framing stage runs live on a Raspberry Pi 5 with a Coral Edge TPU. Bonatti et al. [2], [9] demonstrated occlusion-free aerial cinematography that runs fully onboard in real time, but their framework required a LiDAR sensor, GPS tags, and a TX2-class onboard computer. All of this is hardware that is substantially heavier than the monocular, edge-only configuration that CINE targets.

C. Rule-of-thirds and glare-avoidance results demonstrate that cinematographic principles can be encoded as quantitative constraints

While existing MPC frameworks automate camera parameters, they often fail to explicitly quantify environmental aesthetics, resulting in inadvertent lens glare and unstructured target framing. To resolve this, the CINE framework explicitly encodes light vector avoidance and rule-of-thirds positioning as optimizable cost functions.

To properly quantify the cinematographic framing of the object relative to the camera’s position, a rule-of-thirds grid is modeled with anchor points to place the bounding box position accordingly. Without the framing constraint, target coordinates exhibited uncontrolled variance across the image plane. By introducing a rule-of-thirds constraint layer, CINE guided the target bounding box toward pre-defined cinematic coordinates in a human-in-the-loop bench test (Fig. 12). Ultimately, the results suggest that the grid can serve as a constraint layer for cinematic effect once autonomous actuation replaces the human operator. In order to film without obstructions, this framework incorporates a light avoidance approach that emphasizes the environment without restricting extrinsic flight. Furthermore, the framework’s orientation constraints enable the UAV to dynamically rotate its camera axis away from overexposed light vectors, eliminating visual glare without interrupting continuous translational flight (Fig. 13).

Ultimately, these results demonstrate that qualitative cinematographic principles, such as glare reduction and shot composition, can be successfully translated into rigid mathematical constraints within dynamic trajectory planners.

D. Applications

The cinematographic adjustment framework has direct implications for several use cases where professional aerial cinematography has historically been cost-prohibitive. Independent and small-budget filmmakers, who currently face hourly rates between \$200 and \$500 and per-project costs from \$1,000 to \$5,000 for professional pilot services [1], can leverage CINE to obtain cinematographically composed footage without hiring specialized aerial teams. Furthermore, field journalism and documentary capture in inaccessible or hazardous environments, including disaster response, remote ecological field sites, and conflict reporting, benefit from a system that produces visually stable footage without requiring an experienced human pilot. High-volume commercial applications such as real estate marketing, tourism content, and event coverage can also reduce per-shoot costs while maintaining cinematic quality. Beyond direct production use cases, the framework could generate cinematographically structured training data for downstream computer vision research, particularly for models that learn from video with consistent framing, motion, and composition.

E. Limitations

Although the proposed framework demonstrates strong improvements over baseline planners, several limitations constrain the scope of these results. First, depth estimation relied entirely on monocular imagery through Depth Anything 3, which produces relative rather than metric depth and remains sensitive to scale ambiguity and varying illumination conditions. Stereoscopic or LiDAR-based sensing would provide more reliable absolute depth at the cost of additional payload weight and computational overhead.

Second, the YOLOv11n model achieved an mAP50-95 of approximately 0.44, indicating reduced performance at stricter IoU thresholds. While this did not preclude reliable bounding box extraction during framing, downstream constraints that depend on precise object boundaries may inherit this imprecision. Additionally, training was performed exclusively on the HomeObjects-3K dataset, which limits generalization to outdoor or industrial environments without further fine-tuning.

Third, the cinematic cost function weights were manually tuned, which introduces subjectivity into trajectory generation and would likely require re-tuning for different shot types or environments. The framework also currently supports a single object of focus and does not handle dynamic or multi-object framing scenarios.

Fourth, the 15-trial validation set was scoped deliberately as a systems-level feasibility demonstration in kinematic simulation targeting a single edge configuration (Raspberry Pi 5 with a Google Coral Edge TPU), sufficient to establish measurable improvements over baseline EGO-Planner but not to characterize generalization across hardware tiers, larger trajectory sets, or outdoor environments. Broader validation across these axes is identified as future work in Section V-F. Furthermore, the 2×128 MLP drift corrector was trained entirely on simulated drift data from the kinematic setup, so

its corrections target simulated rather than physical drift and inherit any modeling error in the injected drift dynamics.

F. Future studies

1) *Short-term improvements:* In the immediate term, the perception pipeline could be strengthened by replacing monocular depth estimation with stereoscopic or LiDAR-based sensing, which would eliminate the scale ambiguity that currently constrains depth-stability accuracy. Additionally, validation should extend beyond the Raspberry Pi 5 and Coral TPU configuration to alternative edge platforms such as the NVIDIA Jetson Nano or Raspberry Pi 4, allowing characterization of the framework's performance across hardware tiers. Broader environmental validation is also necessary: testing in larger and texture-diverse spaces would address generalization concerns raised by the current simulation-only trial set. Finally, extending the framework to support multiple regions of interest would allow simultaneous framing of two or more objects, a capability the current single-object architecture cannot handle.

2) *Longer-term goals:* In the longer term, manual tuning of cinematic cost-function weights should be replaced with reinforcement learning, drawing on policies similar to those used in champion-level autonomous drone racing [19] to automatically discover effective weight configurations across shot types. Furthermore, the framework should be extended to support dynamic and human-subject framing, which requires predictive modeling of subject motion and intent. Productizing the system through a graphical user interface would also be necessary to make fully autonomous trajectory generation accessible to non-technical cinematographers. Ultimately, the cinematic-cost-function paradigm itself could generalize beyond cinematography to other autonomous aerial filming domains, including sports broadcasting, scientific imaging in remote field sites, and security-oriented surveillance.

VI. CONCLUSION

Although autonomous UAV cinematography has historically required either expert pilots or computationally heavy MPC frameworks, this study demonstrates that cinematographic principles can be encoded as quantitative cost functions on a real-time B-spline planner. This approach, which couples B-spline generation, depth analysis, visual-inertial odometry, object detection, and an MLP-based corrector, systematically constrained and adjusted trajectory setpoints for cinematographic films with drones. The proposed framework produced smoother, object-oriented trajectories with peak framing velocity clamped from 3.0 m/s to 1.2 m/s, object alignment relative to a rule-of-thirds grid achieved with operator guidance, and glare-free filming by rotating away from detected light vectors. These results demonstrate that visual and cinematic appeal can be modeled as explicit cost functions and model-based constraints on a B-spline planner, extending prior work from MPC-based cinematography and geometric B-spline navigation. Ultimately, CINE provides a foundation for drone cinematography that is autonomous, accessible, and grounded in cinematic intent.

ACKNOWLEDGEMENTS

I would like to thank Nicole Spinelli for her assistance in reviewing and helping me with the structure of this manuscript, as well as all contributors to EGO-Planner, ROS2, CineMPC, Ultralytics, and PyTorch.

REFERENCES

- [1] Z. Dukowitz, “Drone services pricing: A guide to pricing your drone work,” UAV Coach, 2025.
- [2] R. Bonatti, W. Wang, C. Ho, A. Ahuja, M. Gschwindt, E. Camci, E. Kayacan, S. Choudhury, and S. Scherer, “Autonomous drone cinematographer: Using artistic principles to create smooth, safe, occlusion-free trajectories for aerial filming,” in *2018 Int. Symp. Experimental Robotics (ISER)*, 2018. doi: 10.48550/arXiv.1808.09563.
- [3] P. Pueyo, J. Dendarieta, E. Montijano, A. C. Murillo, and M. Schwager, “CineMPC: A fully autonomous drone cinematography system incorporating zoom, focus, pose, and scene composition,” *IEEE Trans. Robot.*, vol. 40, pp. 1294–1311, 2024.
- [4] L. W. He, M. F. Cohen, and D. H. Salesin, “The virtual cinematographer: A paradigm for automatic real-time camera control and directing,” in *Proc. 23rd Annu. Conf. Comput. Graph. Interact. Tech. (SIGGRAPH)*, 1996, pp. 217–224.
- [5] R. Khanam and M. Hussain, “YOLOv11: An overview of the key architectural enhancements,” *arXiv preprint arXiv:2410.17725*, 2024.
- [6] R. E. Kalman, “A new approach to linear filtering and prediction problems,” *J. Basic Eng.*, vol. 82, no. 1, pp. 35–45, 1960.
- [7] M. Y. Arafat, M. M. Alam, and S. Moh, “Vision-based navigation techniques for unmanned aerial vehicles: Review and challenges,” *Drones*, vol. 7, no. 2, art. 89, 2023.
- [8] T. Nägeli, L. Meier, A. Domahidi, J. Alonso-Mora, and O. Hilliges, “Real-time planning for automated multi-view drone cinematography,” *ACM Trans. Graph. (TOG)*, vol. 36, no. 4, Art. 132, 2017. doi: 10.1145/3072959.3073712.
- [9] R. Bonatti, W. Wang, C. Ho, A. Ahuja, M. Gschwindt, E. Camci, E. Kayacan, S. Choudhury, and S. Scherer, “Autonomous aerial cinematography in unstructured environments with learned artistic decision-making,” *J. Field Robot.*, vol. 37, no. 4, pp. 606–641, 2020. doi: 10.1002/rob.21931.
- [10] D. Mellinger and V. Kumar, “Minimum snap trajectory generation and control for quadrotors,” in *2011 IEEE Int. Conf. Robot. Autom. (ICRA)*, 2011, pp. 2520–2525.
- [11] X. Zhou, Z. Wang, H. Ye, C. Xu, and F. Gao, “EGO-Planner: An ESDF-free gradient-based local planner for quadrotors,” *IEEE Robot. Autom. Lett.*, vol. 6, no. 2, pp. 478–485, 2021.
- [12] D. L. Christensen, “Real-time dynamically feasible B-spline trajectory generation for unmanned aircraft,” M.S. thesis, Brigham Young Univ., Provo, UT, 2024.
- [13] D. Eigen, C. Puhrsch, and R. Fergus, “Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture,” in *Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2014, pp. 2366–2374.
- [14] H. Lin, S. Chen, J. H. Liew, D. Y. Chen, Z. Li, G. Shi, J. Feng, and B. Kang, “Depth Anything 3: Recovering the visual space from any views,” *arXiv preprint arXiv:2511.10647*, 2025. doi: 10.48550/arXiv.2511.10647.
- [15] A. Minervini, A. Carrio, and G. Guglieri, “Enhancing visual-inertial odometry robustness and accuracy in challenging environments,” *Robotics*, vol. 14, no. 6, art. 71, 2025.
- [16] C. De Boor, “On calculating with B-splines,” *J. Approx. Theory*, vol. 6, no. 1, pp. 50–62, 1972.
- [17] L. He, Y. Zhou, L. Liu, et al., “Research on object detection and recognition in remote sensing images based on YOLOv11,” *Sci. Rep.*, vol. 15, art. 14032, 2025.
- [18] M. C. Popescu, V. E. Balas, L. Perescu-Popescu, and N. Mastorakis, “Multilayer perceptron and neural networks,” *WSEAS Trans. Circuits Syst.*, vol. 8, no. 7, pp. 579–588, 2009.
- [19] E. Kaufmann, L. Bauersfeld, A. Loquercio, M. Müller, V. N. Pappas, and D. Scaramuzza, “Champion-level drone racing using deep reinforcement learning,” *Nature*, vol. 620, no. 7976, pp. 982–987, 2023.