Scientific Research Publishing

Journal of Computer and Communications, 2026, 14(11), 240-263
https://www.scirp.org/journal/jcc
ISSN Online: 2327-5227
ISSN Print: 2327-5219

Camera-Assisted Recovery from False Data Injection Attacks in Cloud-Based Cyber-Physical Systems Using a CartPole Simulation

Mohammad Arafatur Rahman*, Imran Jahid Khan, Maliha Hossain, Fan Wu

Department of Computer Science, Tuskegee University, Tuskegee, USA
Email: *mrahman@tuskegee.edu

How to cite this paper: Rahman, M.A., Khan, I.J., Hossain, M. and Wu, F. (2026) Camera-Assisted Recovery from False Data Injection Attacks in Cloud-Based Cyber-Physical Systems Using a CartPole Simulation. *Journal of Computer and Communications*, **14**, 240-263.
https://doi.org/10.4236/jcc.2026.1411011

Received: October 31, 2026
Accepted: November 22, 2026
Published: November 25, 2026

Copyright © 2026 by author(s) and Scientific Research Publishing Inc.
This work is licensed under the Creative Commons Attribution International License (CC BY 4.0).
http://creativecommons.org/licenses/by/4.0/

Open Access

Abstract

In a cloud-based cyber-physical system (CPS), a remote controller relies on measurements sent over a network to determine how to act. If an attacker alters those measurements, the controller may choose the wrong action even when the reported values appear plausible. This study examines whether a trusted camera can help a controller detect and recover from False Data Injection (FDI) on its numerical sensor channel. We use a simulated CartPole system controlled by a deep Q-network (DQN). Image processing measures cart position and pole angle, and a Kalman filter combines these measurements with the commanded forces to estimate the full state. The controller replaces a numerical reading when its difference from the corresponding estimate exceeds a threshold set using attack-free trajectories. We evaluate five methods across ten independently trained runs, with 500 episodes per run and condition in the main evaluation. When attacks intermittently replace all four numerical state components, camera-assisted recovery keeps the pole balanced for an average of 9.87 s out of 10 s, compared with 4.17 s for the FDI-aware DQN, 3.22 s for Long Short-Term Memory (LSTM) reconstruction, and 2.73 s for the clean-trained DQN. Camera-only control reaches 9.67 s, and the baseline comparison does not establish a statistically significant gain from selective replacement. In a separate evaluation, camera-assisted recovery achieves 9.75 s mean survival with a 160 × 120 camera operating at 25 Hz and 40 ms latency. Severe pixel noise reduces both attack detection and survival. These findings show that a trusted camera can sustain control during numerical sensor attacks in simulation, provided its images remain accurate enough to guide the controller.

Keywords

Cyber-Physical Systems (CPS), False Data Injection (FDI), Deep Reinforcement Learning, Camera-Assisted Recovery, LSTM Reconstruction, Computer Vision

1. Introduction

In a cloud-based cyber-physical system (CPS), measurements travel from the plant to a remote controller, and control commands return to the actuators. This arrangement separates computation from the equipment being controlled, but it also places a communication path between the controller and its view of the plant. False data injection (FDI) exploits that path by replacing measurements with incorrect values [1] [2]. A range check offers little protection when the replacements remain within the accepted numerical limits. The problem has prompted research on attack detection, secure control, and recovery after compromise [3] [4].

Learned controllers are also affected by incorrect observations. A deep Q-network (DQN) selects actions from learned value estimates [5]; successful training does not ensure that those actions remain appropriate when its inputs are manipulated [6] [7]. Surveys of adversarial reinforcement learning describe both this vulnerability and the defenses developed to address it [8] [9]. Our earlier CartPole studies examined observation noise together with an inclined track [10], and observation noise under different reward functions [11]. Subsequent work considered attack-aware DQN training, feed-forward and LSTM reconstruction, blockchain records of verified states [12], and LSTM attack detection [13]. These methods use the numerical measurements or information retained from earlier measurements. When several current components are corrupted together, recovering the present state from that information becomes more difficult. Our blockchain study also found that prediction-based recovery for drift and FDI could worsen control, despite the value of verified historical states for other attacks [12].

A camera offers another way to observe a visible process. Cart displacement and pole orientation, for example, can be measured from an image without reading the numerical telemetry. If the camera and its communication path remain outside the attacker's control, the resulting estimate can be compared with the received sensor values. Disagreement then provides a reason to question a measurement and a possible source of replacement data. The usefulness of this approach depends on two conditions: the visual estimate must be accurate enough for control, and the camera must remain trustworthy.

We investigate these conditions in a CartPole simulation with a DQN controller and a separate visual observation channel. The numerical state is subjected to FDI on one component or on all four components. A fixed side-view camera provides images from which cart position and pole angle are measured; a Kalman filter estimates the remaining velocities using known control actions. A residual test

determines which numerical components to replace. The evaluation asks how this method compares with the implemented numerical-only defenses, whether selective replacement improves on camera-only control, and how performance changes as image quality and frame delivery deteriorate.

The work contributes the following experimental results and implementation:

1) A visual detection and recovery pipeline that supplies state estimates to an existing learned controller. Image-estimator calibration, filter tuning, and residual thresholds use clean trajectories without attack examples.

2) A comparison across ten independently trained runs, including clean-trained and FDI-aware DQNs, LSTM reconstruction, and a camera-only reference. The analysis reports survival, detection rates, uncertainty across runs, and the effect of intervention when no attack is present.

3) An 18-setting camera study covering pixel noise, resolution, frame rate, and latency, with two configurations that vary several factors together. These measurements identify conditions under which the method retains or loses its observed benefit.

Section 2 places the study in the context of secure estimation and visual control. Sections 3 and 4 describe the threat model and recovery method. The experimental protocol and results follow in Sections 5 and 6, with interpretation and limitations in Section 7 and conclusions in Section 8.

2. Related Work

False data injection can change a controller's view of the plant even when readings appear plausible. Liu *et al.* showed that an attacker who knows a power grid's measurement model can alter its state estimate without triggering a conventional bad-data alarm [2]. Mo and Sinopoli studied how replaying earlier measurements can mislead a controller. They introduced a random control signal to help distinguish recorded measurements from current ones [14]. Subsequent work examined integrity monitoring in SCADA systems [15], the influence of an attacker's knowledge and resources [16], and learning methods for attack detection in cyber-physical and industrial systems [17] [18]. After an alarm, the controller must still choose an action. Zhang *et al.* addressed this problem by computing recovery actions that return a plant to a safe region within a deadline [19].

Resilient state estimation asks how much a controller can infer from the measurements it can still trust. Fawzi *et al.* established conditions under which a linear system's state can be recovered despite sensor attacks [20]. Recovery depends on whether the remaining measurements reveal enough about the state. Pajic *et al.* examined how noise and modelling errors affect such estimates [21], while Ivanov *et al.* used measurement histories to combine information from multiple sensors [22]. In navigation, Gu *et al.* checked satellite data against estimates from cameras and inertial sensors [23]. Petro and Bauer used a single camera to check for GPS spoofing in aerial vehicles [24].

For a learned controller, an observation error can change the action the policy

selects. Huang *et al.* demonstrated this effect with small perturbations to the inputs of neural-network policies [6]. Zhang *et al.* modelled attacks on state observations and developed a training method to improve robustness [7]. Liang *et al.* accounted for worst-case observation errors without generating attacks during training [25]. Wu *et al.* considered attacks on actuators, where the disturbance affects the command applied to the plant [26].

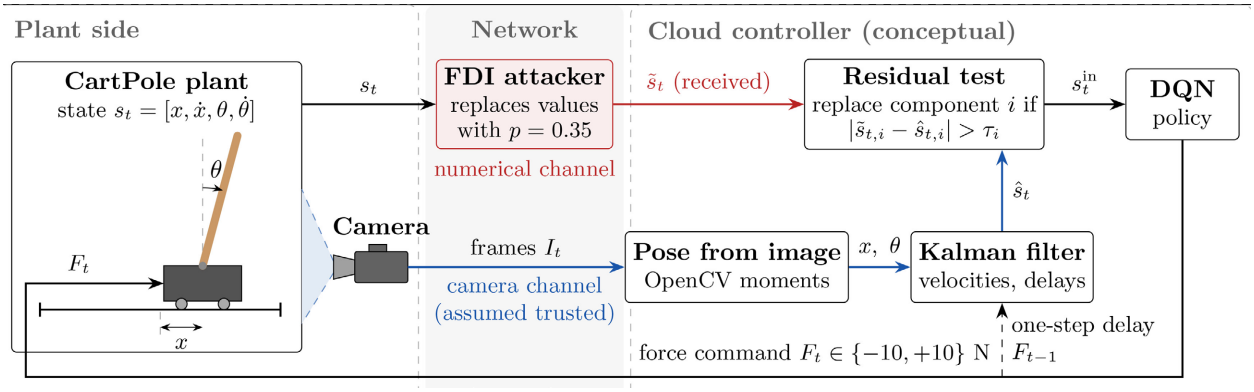


Figure 1. Simulated control architecture. At step t, the Kalman filter uses the camera measurements and the previous force command F_{t-1} to estimate the current state. The controller replaces any numerical component whose difference from the estimate exceeds its threshold. The DQN then selects the next force F_t. The cloud shows the proposed controller location.

Another approach is to reconstruct the current state from recent readings. The long short-term memory (LSTM) architecture introduced by Hochreiter and Schmidhuber can retain information across a sequence [27]. A related study examines reconstruction under Gaussian sensor noise and considers how noise levels, temporal information, and reward design affect performance. When several numerical components are replaced at once, a model that relies on those readings must infer the current motion from their history. A camera provides a separate observation of that motion. In CartPole, a single frame reveals cart position and pole angle, while successive frames and the commanded forces help estimate velocity.

Visual servoing uses image measurements to guide motion [28], and learned Kalman filters can estimate state from images [29]. An accurate visual estimate can correct an injected value, but an inaccurate one can cause the controller to discard an honest reading. The camera itself can also be a target. Physical changes to signs can mislead classifiers [30], and road-surface markings can disrupt automated lane centering [31]. Physical attacks extend to other vision applications [32] and to systems that combine cameras and LiDAR [33]. PercepGuard detects some visual errors by checking whether an object's motion is consistent with its assigned class [34]. Our study assumes a trusted camera. Comparing selective replacement with camera-only control lets us assess the value of retaining numerical readings as image quality declines or frames arrive less frequently or with a delay.

3. System and Threat Model

Figure 1 illustrates how the plant, controller, and observation channels are con-

nected. Numerical measurements and camera frames capture the same physical process separately, but the simulated attacker can modify only the numerical measurements. The cloud indicates where the controller would be hosted in the proposed architecture, although all experiments are run locally.

3.1. Plant and Controller

We use the CartPole balancing task [35] implemented as Gymnasium CartPole-v1 [36]. The numerical state s_t of the CartPole plant at time t is defined as:

$$s_t = \left[x_t, \dot{x}_t, \theta_t, \dot{\theta}_t \right] \tag{1}$$

where x_t, $\dot{x}_t$, θ_t and $\dot{\theta}_t$ represent cart position, cart velocity, pole angle, and pole angular velocity respectively. The controller applies either −10 or +10 N at intervals of 0.02 s. Failure occurs when $|x| > 2.4$ m or $|\theta| > 12^\circ$. Episodes that have not failed are stopped after 500 steps, giving a maximum recorded survival time of 10 s.

The DQN receives one state observation per control step and chooses the next force. Although the architecture represents remote control, no physical network or cloud service is implemented. The camera study in Section 6.4 introduces fixed frame delays in simulation; it does not measure transmission delay in a deployed system.

3.2. False Data Injection Model

At each observation, including the initial one, an attack occurs with probability $p = 0.35$. When an attack occurs, the selected numerical state components are replaced with values drawn independently from uniform distributions over their specified ranges:

$$\tilde{s}_t = s_t + b_t M \left(u_t - s_t \right), \quad b_t \sim \text{Bernoulli}\left(p \right), \quad u_t \sim \mathcal{U}\left[-r, r \right] \tag{2}$$

The diagonal mask M determines which components are replaced, and $r = \left(2.4\ \text{m}, 2.0\ \text{m/s}, 0.2095\ \text{rad}, 2.0\ \text{rad/s} \right)$ defines the replacement ranges. We evaluate five attack conditions: each of the four state components individually, and all four together.

A single Bernoulli draw determines whether an attack occurs at each observation, so all selected components are replaced simultaneously. When all four components are selected, 35% of observations are fully replaced in expectation, while the remaining observations are clean. The plant dynamics, rewards, and termination conditions use the true state. Neither the replacement values nor the attack timing is optimised against the policy or the detector.

3.3. Simulated Camera and Visual Estimation Assumptions

The visual channel generates synthetic side-view images using the cart, pole, and track geometry and colours of the Gymnasium renderer. Shapes are drawn with sub-pixel antialiasing to smooth their edges. Independent Gaussian noise is then

added to the colour channels, and pixel values are clipped to the valid range and quantised to 8 bits. The baseline configuration uses a resolution of 320 × 240 pixels, a pixel-noise standard deviation of 4, and one frame per control step. The horizontal field of view spans 5.6 m, giving approximately 57 pixels per metre and covering the full allowed range of cart positions.

We assume that the camera operates independently of the attacked numerical channel and that its images are protected from manipulation. The background, object appearance, and scene geometry are known, and each episode starts with a synchronised camera frame before the controller takes its first action. These assumptions support image-based state estimation in the simulated setting, but limit how directly the results apply to physical cameras.

The simulator uses the true state to render images and calculate offline targets and evaluation errors. During operation, however, the visual estimator receives only images and commanded actions; it has no direct access to the true state.

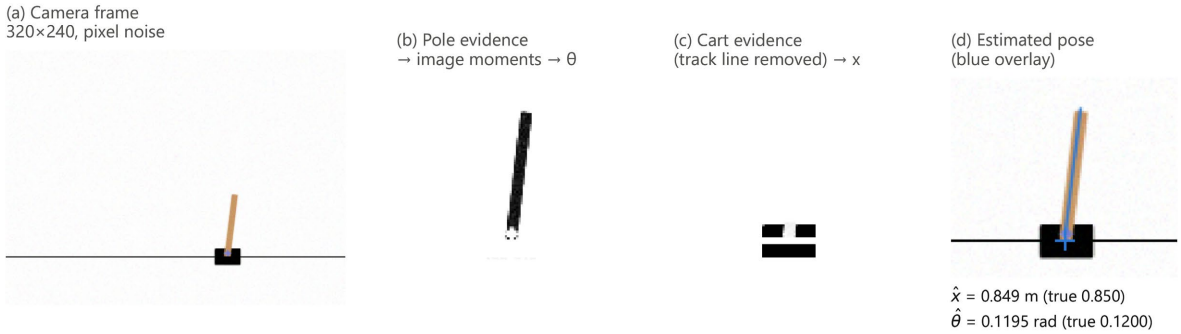


Figure 2. Extracting measurements from a noisy 320 × 240 camera frame. (a) The rendered image. (b) Pixel weights based on pole colour, used to estimate the pole angle. (c) Pixel weights based on cart darkness, with the track row removed. (d) The estimated cart position and pole orientation, shown in blue. The known scene geometry converts pixel measurements into physical position and angle.

4. Camera-Assisted Recovery

Camera-assisted recovery has three steps. First, image processing measures the cart position and pole angle from each camera frame. Then, a Kalman filter uses these measurements and the commanded forces to estimate the full state, including cart position, cart velocity, pole angle, and pole angular velocity. It also predicts the state when frames are missing or delayed. Finally, the controller compares each received numerical component with the corresponding estimate and replaces it if the difference exceeds a specified threshold. We use classical image processing because its measurements are easy to inspect and it can be calibrated without attack data. Figure 2 shows how measurements are extracted from an image.

4.1. Measuring Cart Position and Pole Angle

To identify the pole, each pixel receives a weight based on how closely its colour

matches the known pole colour: $w = \max(0, 1 - d/80)$, where d is the Euclidean distance between the two 8-bit colour vectors. Pixels with colours closer to the pole colour receive higher weights. Using continuous weights also preserves information from the smoothed edges of the pole.

OpenCV's moments function [37] calculates the second-order central moments μ_{20}, μ_{02}, and μ_{11}. These describe how the weighted pixels are distributed and are used to find the orientation of the pole's main axis:

$$\varphi = \frac{1}{2}\operatorname{atan2}(2\mu_{11}, \mu_{20} - \mu_{02}). \tag{3}$$

We choose the axis direction that points from the pivot towards the pole tip and express the pole angle relative to the vertical.

To locate the cart, we examine only the image band where the cart is expected to appear. The track row is excluded, along with adjacent rows when the image resolution allows. Each remaining pixel receives a darkness weight: $w = \min(1, \max(0, (160 - g)/96))$, where g is its mean colour-channel intensity. This gives more weight to dark cart pixels and reduces the influence of the background under moderate pixel noise.

The weights are summed by column to find the window with the strongest evidence of the cart. The weighted centre of this window gives the cart's horizontal pixel position u. We then convert it to physical displacement using $x = (u - W/2)/k$, where W is the image width and k is the number of pixels per metre.

Before updating the Kalman filter, the estimator rejects a frame if the evidence for either object is insufficient. The pole test rejects a frame when the total pole-colour weight, the zeroth image moment m_{00}, is less than one quarter of the pole's nominal area in pixels. The cart test rejects a frame when the summed darkness weight in the cart band, after removing the track rows, is less than one. A rejected frame supplies no measurement update; the filter instead relies on prediction as described in Section 4.2.

Neither rejection rule was triggered in this study. All 17,706 calibration frames in each of the 18 camera configurations produced accepted measurements. To assess rejection during evaluation, we replayed all 62,000 evaluation episodes that use the camera (camera-assisted recovery and camera-only control, in both studies) with a rejection counter. Episode lengths, detector counts, input RMSEs, and mean absolute pole angles matched the saved records exactly. No frames were rejected among the 17,350,585 processed in the main study or the 8,285,287 processed in the camera study. These 25,635,872 frames include the most severe tested settings, 20×15 pixels and pixel-noise standard deviation $\sigma = 64$. Thus, the observed failures under degraded imagery occur despite accepted measurements; the rejection rule does not screen out the inaccurate estimates responsible for those failures.

For the baseline camera, the raw measurement Root Mean Square Error (RMSE) is 2.7 mm for cart position and 0.0042 rad, or approximately 0.24˚, for pole angle. These errors describe the renderer and trajectories used for calibration.

4.2. Estimating Velocities and Handling Delayed Frames

A single image provides position and angle measurements. To estimate velocities as well, we use a linear Kalman filter [38]. Its motion model is obtained by linearising the CartPole dynamics around the upright position and applying the simulator's explicit-Euler update with a time step of 0.02 s. The filter also uses the commanded force F_t, which is known to the controller:

$$\begin{aligned} \hat{s}_{t+1|t} &= A\hat{s}_{t|t} + BF_t, \\ z_t &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} s_t + v_t. \end{aligned} \tag{4}$$

Here, A and B describe the motion model, and z_t contains the image-based measurements of cart position x and pole angle θ. The term v_t represents measurement noise. The filter predicts how the state changes under the commanded force, then corrects that prediction when a camera measurement is available.

Measurement-noise variances are estimated from the differences between the image measurements and the true calibration states. A grid search selects velocity process-noise standard deviations of 0.01 m/s and 0.02 rad/s per step by minimising velocity estimation error on the calibration trajectories. The process-noise variances for cart position and pole angle are both set to 10^{-8}.

The resulting calibration RMSEs are 1.3 mm for x, 0.012 m/s for $\dot{x}$, 0.0022 rad for θ, and 0.022 rad/s for $\dot{\theta}$. These are in-sample errors because the same trajectories are used to select the filter parameters.

Control begins after the first camera frame is received. This initial frame is delivered at reset without delay in every camera configuration, including those with reduced frame rates or delayed subsequent frames. The filter initializes cart position and pole angle from this image and sets both velocities to zero. In state order $\left(x, \dot{x}, \theta, \dot{\theta}\right)$, the initial mean and covariance are

$$\begin{aligned} \hat{s}_{0|0} &= \left(x_0^{\text{img}}, 0, \theta_0^{\text{img}}, 0\right)^{\mathrm{T}}, \\ P_{0|0} &= \operatorname{diag}\left(R_{xx}, 0.05^2, R_{\theta\theta}, 0.05^2\right). \end{aligned} \tag{5}$$

Here, R_{xx} and $R_{\theta\theta}$ are the calibrated measurement-noise variances, approximately $(0.0027\ \text{m})^2$ and $(0.0042\ \text{rad})^2$ for the baseline camera. The velocity prior has standard deviations of 0.05 m/s and 0.05 rad/s. Each simulator state component is initially drawn uniformly from $[-0.05, 0.05]$ in its corresponding units, so zero is its mean and the chosen velocity standard deviation exceeds the distribution's standard deviation, $0.05/\sqrt{3}$. A single frame cannot determine velocity; early estimates therefore depend on this prior and the motion model until later frames are fused. If the initial frame were rejected, position and angle would also start at zero, although this case did not occur in calibration or evaluation.

Numerical telemetry is never used to initialize or update the filter, so an injection at the first step cannot alter its initial estimate. The zero cart-velocity estimate can nevertheless cause an honest initial reading to be replaced if its magnitude

exceeds the baseline threshold of 0.039 m/s (Section 4.3). This startup effect contributes at most one cart-velocity replacement per episode.

When camera frames arrive with a fixed delay of d steps, the filter updates the estimate at the time represented by the frame. It then predicts forward to the current step using the forces commanded during the intervening steps. If no new frame is available, the filter predicts the state without a measurement update. This allows the camera to operate more slowly than the controller, although longer gaps make the estimate depend more on the motion model.

4.3. Detecting and Replacing Suspicious Measurements

The controller compares each received numerical component with the corresponding camera-assisted estimate. If the absolute difference exceeds that component's threshold τ_i, the received value is replaced by the estimate:

$$s_{t,i}^{\text{in}} = \begin{cases} \hat{s}_{t,i}, & \left|\tilde{s}_{t,i} - \hat{s}_{t,i}\right| > \tau_i, \\ \tilde{s}_{t,i}, & \text{otherwise.} \end{cases} \tag{6}$$

Here, $s_{t,i}^{\text{in}}$ is the value passed to the DQN. Values whose differences do not exceed the threshold are kept.

Each threshold is set to the empirical 99.5th percentile of the absolute differences observed during calibration without numerical attacks. The thresholds are 3.8 mm for cart position, 0.039 m/s for cart velocity, 0.0064 rad for pole angle, and 0.066 rad/s for pole angular velocity.

Calibration uses one trained policy. In half of the calibration episodes, 30% of actions are chosen randomly to cover a wider range of states. The corresponding 0.5% false-alarm rate is a calibration target for each component; it is not a guaranteed rate for other policies or trajectories.

The visual estimator never uses the received numerical measurements. Neither the estimator calibration nor the threshold selection uses attack examples.

4.4. Methods Used for Comparison

We compare camera-assisted recovery with four alternatives:

- **Clean DQN:** A policy trained on true states. During evaluation, it receives numerical observations directly, including any values changed by an attack.
- **FDI-aware DQN:** A separate policy trained with false data injection. Each training episode selects one of six conditions: no attack, an attack on one of the four state components, or an attack on all four components.
- **LSTM reconstruction:** An estimator with the same architecture. A single-layer LSTM [27] with 64 hidden units processes a window of eight numerical observations. At the first control step, the window contains eight copies of the first received observation. Each subsequent observation replaces the oldest entry, so the window contains eight distinct time steps once eight observations are available. Training uses the same padding by repeating the first received observation seven times before the episode's observation sequence. If the first

observation is attacked, its injected values fill the initial window in both training and evaluation. The LSTM output passes through a 64-unit ReLU layer and a four-unit output layer to reconstruct the current state. The reconstructed state is then given to the clean DQN. The estimator does not receive actions as inputs.

- **Camera-only control:** The clean DQN receives the full state estimate from the visual Kalman filter and ignores the numerical measurements. This comparison helps distinguish the benefit of adding a camera from the benefit of selectively replacing suspicious components.

In the reconstruction-assisted DQN, a separate DQN is trained on reconstructed states. Here, the LSTM output is passed to the already trained clean DQN, whose parameters remain fixed. Within each run, LSTM reconstruction, camera-assisted recovery, and camera-only control therefore use the same policy. Their performance differences reflect the state inputs supplied to that policy.

5. Experimental Setup

5.1. Training

We use ten seeds (42 - 51), training a clean DQN, an FDI-aware DQN, and an LSTM separately for each seed. Both DQNs use a 4-128-128-2 network with ReLU hidden activations, but with different training budgets, checkpoint selection, and time-limit handling. Each DQN trains for 3000 episodes using Adam with a learning rate of 10^{-3}, a discount factor of 0.99, Huber loss, gradient clipping at 10, batches of 64, and a replay buffer of 50,000 transitions. Exploration starts at $\varepsilon = 1.0$ and decreases by a factor of 0.995 per episode to a minimum of 0.05. Target-network weights are updated every 20 episodes.

Every 50 episodes, the policy is evaluated on 20 fixed validation episodes under its training condition. The checkpoint with the highest mean score is retained. All selected clean checkpoints reach 500 steps in every validation episode; mean scores for the selected FDI-aware checkpoints range from 252 to 458 steps.

The learning target removes the bootstrap term after a true termination but retains it at the 500-step time limit [39]. Although both DQNs receive the same episode budget, attacked episodes often end earlier. Clean and FDI-aware DQNs therefore average 540,925 and 266,561 training steps, respectively.

Each LSTM uses approximately 100,000 windows of eight observations collected with its corresponding clean DQN. The policy acts on true states, with a random-action probability sampled uniformly from 0 to 0.5 for each episode. Recorded observations are corrupted using randomly selected FDI masks, pairing corrupted histories with current true states and including trajectories near failure. Entire episodes are split approximately 90:10 between training and validation to prevent overlapping windows across the two sets. States are normalised by their component ranges. Training minimises mean squared reconstruction error using Adam at 10^{-3}, batches of 128, and 20 epochs. The checkpoint with the lowest validation error is retained.

5.2. Camera Calibration

Calibration uses 40 unattacked episodes from the first run's clean DQN (seed 42): half with greedy actions and half with 30% random actions. These data determine measurement noise, velocity process noise, and detection thresholds. The resulting baseline calibration is shared by all ten evaluated runs; the seed-42 run therefore uses a calibration collected with its own policy. Each camera-study configuration is calibrated once using the same procedure, and that calibration is shared across its ten runs. The camera study consequently measures performance with configuration-specific tuning.

5.3. Evaluation Protocol

The main evaluation uses 500 episodes per trained run under six conditions: no attack and the five attack masks. The four methods that use numerical observations are tested under every condition. Camera-only control is tested once per run because numerical attacks cannot affect it. This gives $10\times500\times\left(4\times6+1\right)=125000$ episode records. Initial-state seeds begin at 4,000,000 and are separate from training, validation, calibration, and earlier development tests. Methods share initial states and pseudorandom attack streams for paired comparisons.

The camera study covers 18 configurations: the baseline, individual changes to camera properties, and two combined changes. Each configuration is tested with camera-assisted recovery under no attack and all-component FDI, and with camera-only control. Ten trained runs and 50 episodes per condition give 27,000 additional records. This study uses separate initial-state seeds beginning at 5,000,000.

5.4. Metrics and Statistical Analysis

Mean survival time (MST) is the average episode duration, capped at 10 s. Success means reaching step 500 without termination. Of the 77,209 main-study episodes that execute 500 steps, 40 terminate on the final step and count as failures.

For each episode, input RMSE measures the difference between the state supplied to the policy and the true state before each action. Mean absolute pole angle (MAPA) averages $\left|\theta\right|$ over the same observations. Metrics are first averaged across episodes within each run; reported RMSE therefore averages episode-level RMSEs rather than pooling all errors.

The true-positive rate (TPR) is the fraction of injected component values flagged by the detector. We also report TPR for injected values that differ from the true value by more than 0.24 m, 0.2 m/s, 0.02095 rad, or 0.2 rad/s for the four components, respectively. These fixed cutoffs correspond to 10% of the replacement-range magnitudes and identify larger changes independently of the detector's thresholds. The false-positive rate (FPR) is the fraction of honest component values, those not injected at that step, that the detector flags. We report this rate for unattacked episodes and for honest values within attacked episodes, pooling false-positive and true-negative counts across the ten runs. Under all-component

FDI, honest values occur at steps without an injection. Under a single-component attack, they also include every value of the three components outside the attack mask.

Statistical analysis treats each independently trained run as one observation. We report the mean and sample standard deviation across the ten run means. All runs share a single camera calibration (Section 5.2), so the reported standard deviations, confidence intervals, and tests are conditional on that calibration. They reflect variability from policy training and evaluation episodes, but do not quantify variability from independently repeating calibration, which could change measurement-noise estimates, process-noise selection, and detection thresholds. The same qualification applies to each camera-study configuration.

Camera-assisted recovery is compared with each alternative using a two-sided paired t-test and a two-sided sign-flip test covering all 2^{10} sign assignments [40]. The sign-flip test assumes sign exchangeability under the null hypothesis. With ten nonzero paired differences, its smallest two-sided probability is 2/1024; tied runs reduce the number of effective sign assignments and raise this minimum.

Holm adjustment is applied across the 20 attacked-condition comparisons separately for each test type [41]. The no-attack family consists of two comparisons: camera-assisted recovery versus the clean DQN, and camera-assisted recovery versus camera-only control, both evaluated without attack. Holm adjustment is applied within this family of two, separately for each test type. Reported 95% confidence intervals are pointwise. A post hoc camera-study analysis compares camera-assisted recovery under all-component FDI with camera-only control using the same paired sign-flip test, with Holm adjustment across the 18 configurations as a separate family.

6. Results

6.1. Survival under FDI

Across all five attack conditions, camera-assisted recovery keeps mean survival between 9.87 and 9.98 s of the 10 s horizon (Table 1). The clean DQN is most vulnerable to attacks on pole angle and angular velocity, which reduce mean survival to 2.62 and 2.52 s, respectively.

Table 1. Mean survival time in seconds by attacked component. Entries are mean ± sample SD across ten runs, with 500 episodes per run and condition. The camera-only control row repeats one attack-independent evaluation per run.

Pipeline	None	x	$\dot{x}$	θ	$\dot{\theta}$	All
Clean DQN	10.00 ± 0.00	9.48 ± 1.35	9.48 ± 1.27	2.62 ± 2.32	2.52 ± 2.17	2.73 ± 2.07
FDI-aware DQN	9.21 ± 1.14	8.50 ± 1.06	7.89 ± 1.68	5.92 ± 2.52	5.52 ± 2.08	4.17 ± 1.25
LSTM reconstruction	9.65 ± 0.74	8.61 ± 2.09	8.56 ± 1.68	7.07 ± 3.12	6.50 ± 3.19	3.22 ± 1.89
Camera-assisted recovery	9.98 ± 0.05	9.98 ± 0.05	9.98 ± 0.05	9.94 ± 0.19	9.98 ± 0.07	9.87 ± 0.40
Camera-only control	9.67 ± 0.97	9.67 ± 0.97	9.67 ± 0.97	9.67 ± 0.97	9.67 ± 0.97	9.67 ± 0.97

When all four components are attacked, camera-assisted recovery reaches 9.87 s mean survival and 96.6% success (Table 2). The strongest numerical-only method, the FDI-aware DQN, averages 4.17 s. Camera-only control averages 9.67 s.

The mean episode-level angle RMSE is 0.0018 rad for camera-assisted recovery and 0.026 rad for LSTM reconstruction. Each error is measured along the trajectory of its own controller.

Table 2. All-component injection results. Survival and success are reported as mean ± sample SD across ten runs. Input RMSE and MAPA are run-averaged episode metrics over pre-action states.

Pipeline	MST (s)	Success (%)	Input RMSE θ (rad)	Input RMSE $\dot{\theta}$ (rad/s)	MAPA (rad)
Clean DQN	2.73 ± 2.07	7.5 ± 9.0	0.093	0.82	0.078
FDI-aware DQN	4.17 ± 1.25	8.2 ± 9.4	0.087	0.76	0.062
LSTM reconstruction	3.22 ± 1.89	8.9 ± 11.8	0.026	0.37	0.075
Camera-assisted recovery	9.87 ± 0.40	96.6 ± 10.5	0.0018	0.017	0.050
Camera-only control	9.67 ± 0.97	93.1 ± 20.6	0.0026	0.025	0.052

Performance varies across policies (Figure 3). With seed 48, mean survival is 8.73 s for camera-assisted recovery and 6.92 s for camera-only control. This seed accounts for about 93% of the overall 0.19 s difference between the two methods.

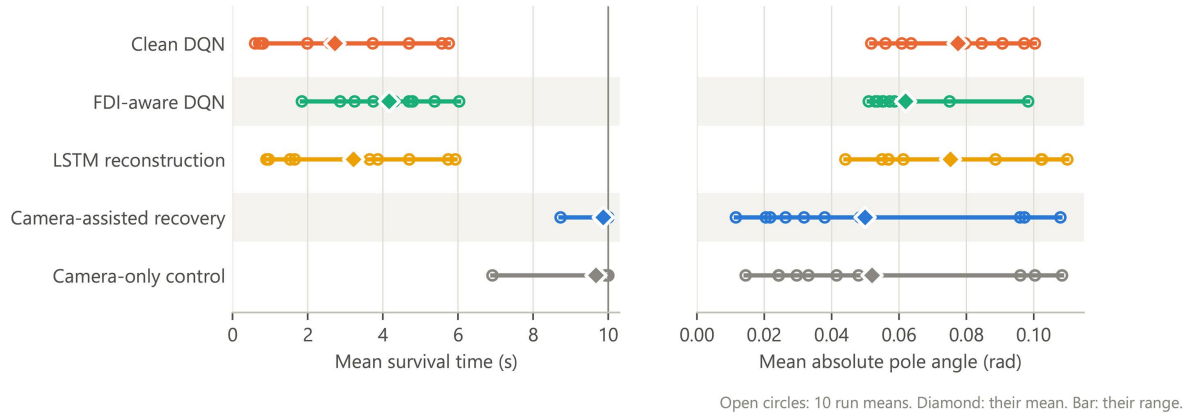


Figure 3. Variation across trained runs under all-component injection. Each circle represents a 500-episode run mean, the diamond is the mean across ten runs, and the bar spans their range. The panels show survival time and mean absolute pole angle. Survival is capped at 10 s.

Under angle, angular-velocity, and all-component attacks, camera-assisted recovery outlasts all three numerical-only methods in every run (Table 3, Holm-adjusted sign-flip $p = 0.039$). The adjusted paired t-tests agree except for the two comparisons with the LSTM under angular attacks. Gains are smaller for attacks on cart position and velocity. No adjusted main-study comparison establishes an advantage over camera-only control.

Table 3. Paired run-level survival differences, camera-assisted recovery minus each alternative. All tests use ten pairs. p-values are Holm-adjusted across 20 attack comparisons separately for each test family; 95% confidence intervals are pointwise.

Attack	Compared with	Difference (s)	95% CI	Positive runs	Sign-flip p	t-test p
x	Clean DQN	0.50	[−0.47, 1.47]	8/10	0.301	1.000
x	FDI-aware DQN	1.48	[0.71, 2.26]	10/10	0.039	0.024
x	LSTM	1.37	[−0.09, 2.83]	8/10	0.063	0.507
x	Camera-only control	0.31	[−0.35, 0.96]	5/10	0.375	1.000
$\dot{x}$	Clean DQN	0.50	[−0.41, 1.41]	6/10	0.375	1.000
$\dot{x}$	FDI-aware DQN	2.09	[0.89, 3.28]	10/10	0.039	0.041
$\dot{x}$	LSTM	1.42	[0.24, 2.60]	9/10	0.039	0.208
$\dot{x}$	Camera-only control	0.31	[−0.35, 0.97]	5/10	0.375	1.000
θ	Clean DQN	7.32	[5.70, 8.94]	10/10	0.039	<0.001
θ	FDI-aware DQN	4.01	[2.19, 5.84]	10/10	0.039	0.011
θ	LSTM	2.87	[0.71, 5.03]	10/10	0.039	0.148
θ	Camera-only control	0.26	[−0.30, 0.82]	5/10	0.375	1.000
$\dot{\theta}$	Clean DQN	7.46	[5.93, 8.99]	10/10	0.039	<0.001
$\dot{\theta}$	FDI-aware DQN	4.46	[2.95, 5.97]	10/10	0.039	0.001
$\dot{\theta}$	LSTM	3.47	[1.22, 5.72]	10/10	0.039	0.075
$\dot{\theta}$	Camera-only control	0.30	[−0.34, 0.95]	5/10	0.375	1.000
All	Clean DQN	7.14	[5.74, 8.55]	10/10	0.039	<0.001
All	FDI-aware DQN	5.70	[4.63, 6.77]	10/10	0.039	<0.001
All	LSTM	6.65	[5.39, 7.91]	10/10	0.039	<0.001
All	Camera-only control	0.19	[−0.21, 0.60]	5/10	0.375	1.000

Figure 4 shows how the recovered input tracks the true angle despite large injected readings. Among episodes with the longest survival under camera-assisted recovery, we chose the one whose clean and FDI-aware DQN survival times were closest in total to their respective medians. The LSTM also completes this episode.

6.2. Detection

With the baseline camera, detection rates range from 96.7% to 99.8% across components (Table 4). Every injection above the fixed physical cutoffs is detected in this evaluation.

All components except cart velocity exceed the 0.5% calibration target, most clearly angle (1.86%) and angular velocity (1.16%). Thresholds set using one policy did not yield the same false-alarm rates across the evaluation policies.

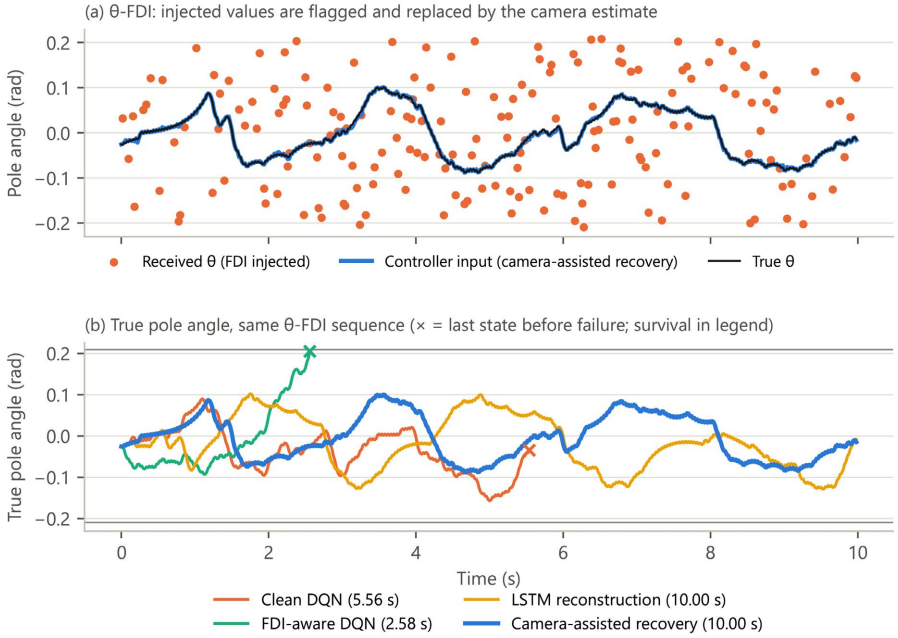


Figure 4. Selected angle-injection episode, seed 42 and episode 62. (a) Injected angle readings, recovered policy input, and true angle. (b) True angle along each pipeline's trajectory. Crosses mark the last plotted preterminal state, and solid grey lines mark ± 12˚. The clean DQN terminates at the cart-position limit. The LSTM and camera-assisted recovery policies both complete this episode.

Table 4. Baseline-camera detection rates pooled across ten runs. Each component has 863,580 injected observations under all-component FDI; large-injection counts use fixed physical cutoffs. FPR is the fraction of honest (non-injected) component values that are flagged. Its denominators are 2,495,465 values per component without attack and 1,603,601 within all-component FDI episodes.

Component	TPR, all injections	Large injections	TPR, large injections	FPR, no attack	FPR, all-component FDI
x	0.998	777,400	1.000	0.65%	0.69%
$\dot{x}$	0.980	777,576	1.000	0.36%	0.36%
θ	0.969	777,343	1.000	1.86%	2.29%
$\dot{\theta}$	0.967	776,555	1.000	1.16%	1.56%

Within all-component FDI episodes, the FPR on honest values is 0.69%, 0.36%, 2.29%, and 1.56% for x, $\dot{x}$, θ, and $\dot{\theta}$, respectively (Table 4). Pooling honest values across all five attack conditions gives 0.70%, 0.36%, 2.23%, and 1.46%. Relative to unattacked episodes, the angle and angular-velocity rates rise by about 0.3 - 0.4 percentage points, while the cart-component rates change little. Because the visual estimator never reads the numerical channel, an attack can affect its errors on honest values through the resulting control actions and trajectories, rather than by directly corrupting the estimate. The pooled rates also conceal variation between policies: under all-component FDI, angle FPR ranges from 0.43% to 5.10% across runs, compared with 0.42% to 3.82% without attack.

6.3. Cost without an Attack

Without attacks, camera-assisted recovery averages 9.98 s versus 10.00 s for the clean DQN and 9.67 s for camera-only control. Neither paired comparison is statistically significant after Holm adjustment within the two-comparison no-attack family. Relative to the clean DQN, the mean survival difference is −0.018 s (95% CI [−0.053,0.017] s), with adjusted $p = 0.54$ for the t-test and $p = 0.25$ for the sign-flip test. Recovery is lower in three runs and equal in seven; with only three nonzero differences, the smallest attainable two-sided sign-flip probability is 0.25. Relative to camera-only control, the mean difference is +0.307 s (95% CI [−0.351, 0.966] s), with adjusted $p = 0.54$ and $p = 0.125$, respectively. Recovery is higher in five runs and equal in five.

Across 5000 paired episodes, the clean DQN succeeds and recovery fails in 17 cases, with no cases in the reverse direction. Thus, the lack of a statistically significant mean difference does not imply that intervention is harmless: false replacements cause failures in some episodes.

In the separate camera study, mean survival without attacks falls to 6.44 s at pixel-noise standard deviation $\sigma = 32$ and 8.90 s at 40 × 30 pixels. In both settings, the controller can lose balance even when all numerical readings are accurate.

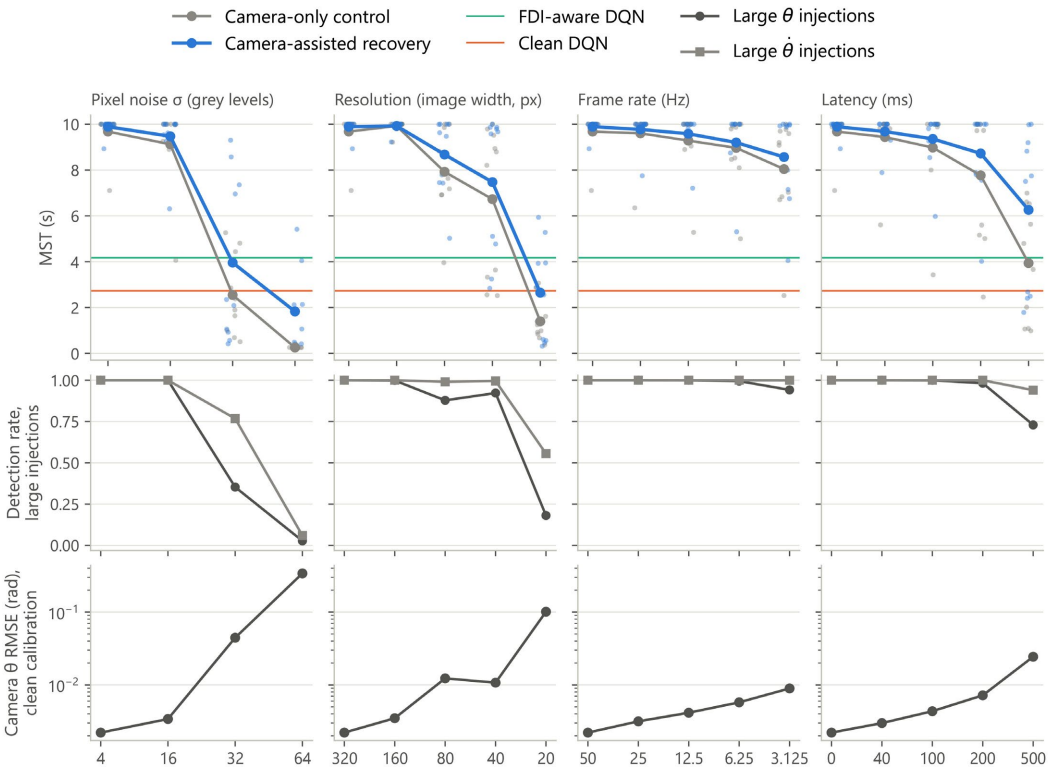


Figure 5. Effect of changing one camera property from the baseline (320 × 240 pixels, $\sigma = 4$, 50 Hz, zero latency). Top: survival under all-component FDI and the camera-only control reference. Dots show individual run means. Horizontal lines show main-study DQN means under the same attack on a different episode suite. Middle: detection rates for large angle and angular-velocity injections. Bottom: in-sample calibration angle RMSE on a logarithmic scale. Each setting uses ten runs and 50 episodes per run.

6.4. Camera Quality and Frame Delivery

Figure 5 shows how camera quality and frame delivery affect survival under all-component FDI after recalibration. Mean survival remains above 9 s at 160 × 120 pixels, 12.5 Hz, 100 ms latency, or $\sigma = 16$ when each factor is varied separately. It falls to 2.65 s at 20 × 15 pixels, 6.26 s at 500 ms latency, and 3.96 s at $\sigma = 32$.

Pixel noise also weakens detection. The fraction of large angle injections detected falls from 100% at baseline to about 35% at $\sigma = 32$ and 3% at $\sigma = 64$. Poorer images can make replacement estimates less accurate and widen the thresholds used to flag suspicious readings.

With resolution, frame rate, and latency set together to 160 × 120 pixels, 25 Hz, and 40 ms, camera-assisted recovery averages 9.75 s with 94.6% success at $\sigma = 4$ (Table 5). Increasing noise to $\sigma = 16$ reduces these figures to 9.34 s and 88.4%.

Across the 18 settings, lower angle error on the calibration trajectories generally corresponds to longer survival (Figure 6). This association applies to the tested simulator settings. It does not establish a minimum accuracy requirement for a physical camera.

Table 5. Baseline and joint camera configurations in the separate ablation suite. Each uses ten runs and 50 episodes per condition. Survival entries are mean ± sample SD; camera error is measured on the clean data used for calibration.

Setting	Camera θ RMSE (rad)	Recovery MST (s)	Success (%)	Camera-only control MST (s)	TPR, large θ injections	Highest FPR, no attack
Baseline: 320 × 240, 50 Hz, 0 ms, $\sigma = 4$	0.0022	9.89 ± 0.34	95.6	9.68 ± 0.91	1.000	1.9%
160 × 120, 25 Hz, 40 ms, $\sigma = 4$	0.0049	9.75 ± 0.56	94.6	9.56 ± 1.30	0.994	0.7%
160 × 120, 25 Hz, 40 ms, $\sigma = 16$	0.0079	9.34 ± 1.27	88.4	8.80 ± 1.88	0.971	0.8%

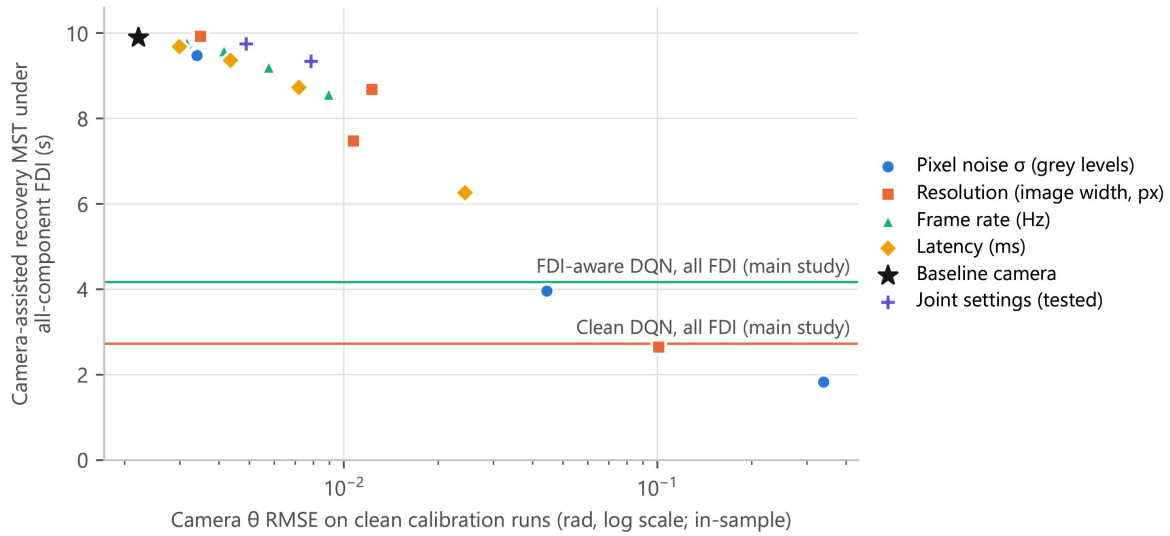


Figure 6. Camera-assisted recovery survival under all-component FDI versus in-sample calibration angle RMSE for 18 camera settings. Horizontal lines are main-study means for the numerical-only DQNs under the same attack type. They use a different episode suite and serve as descriptive references.

In a follow-up analysis, we compared camera-assisted recovery with camera-only control across all 18 settings. Camera-assisted recovery kept the system balanced longer in all ten runs at 500 ms latency and in all ten runs at pixel-noise level $\sigma = 64$. The mean gain was 2.32 s at 500 ms latency and 1.57 s at $\sigma = 64$. Only these two comparisons were statistically significant after Holm adjustment across the 18 settings ($p = 0.035$ for each). At $\sigma = 64$, both methods still failed quickly: mean survival was 1.83 s with camera-assisted recovery and 0.25 s with camera-only control.

6.5. Processing Time

Without attacks, the median of the per-episode mean processing times is 1.11 ms for camera-assisted recovery, 0.23 ms for LSTM reconstruction, and 0.035 ms for DQN inference alone. Eight single-threaded evaluation processes share the machine. Timing includes estimation, policy inference, and evaluation bookkeeping. Rendering, encoding, communication, and actuation are excluded. Across episodes, the 99th percentile of the per-episode 99th percentiles is 8.0 ms, and the largest is 115 ms. These simulation measurements do not establish a real-time deadline for deployment.

7. Discussion

7.1. Contribution of the Visual Channel

The main benefit is access to a measurement that the numerical-channel attacker does not control. During an injected step, the visual estimator can still observe cart position and pole angle, while its dynamic model carries information about velocity. The numerical-only methods must instead use the altered observation or infer the current state from earlier data. Under the tested all-component replacement process, those implementations lose substantially more survival time. This result is consistent with the importance of a trustworthy recovery reference in our earlier blockchain work [12], although the attacks, controllers, and recovery mechanisms differ.

Camera-only control already reaches 9.67 s under all-component FDI. Most of the baseline benefit therefore comes from access to a trusted measurement. The camera study shows where retaining numerical readings can help. The gain over camera-only control persists after adjustment at 500 ms latency and at $\sigma = 64$, although both methods perform poorly at that noise level. The failures without attack show the cost of false intervention. Replacing an accurate numerical reading can also disrupt control.

7.2. Fairness of the Comparison

The comparison concerns the implemented baselines, not all possible numerical defenses. LSTM reconstruction was among the strongest defenses, but that study used Gaussian noise and replay and trained a separate DQN on the reconstructed inputs. The weaker result here applies to this baseline under FDI and should not

be read as a result for the full reconstruction-assisted method. The LSTM receives no action history or explicit plant model, whereas the visual estimator uses both. An action-conditioned recurrent model or a model-based filter for the numerical stream could perform differently. The DQNs also have equal episode budgets but unequal interaction counts. A comparison matched on environment steps would require additional FDI-aware training. These differences should be considered when attributing the measured gains to the sensing modality.

7.3. Limitations

Several assumptions limit how broadly these findings apply.

- **Rendered images.** The background, object appearance, and geometry are known. Physical tests should assess how lighting changes, blur, distortion, frame loss, and clutter affect both image measurements and control.
- **Camera integrity and freshness.** We assume the attacker cannot alter or replay video. Either attack could compromise the camera estimate. Watermarking could help detect inconsistent or replayed measurements [15, 44], but we did not test it.
- **Nonadaptive injection.** Attack timing and replacement values follow fixed distributions. An adaptive attacker could exploit the policy, filter, or thresholds. A small residual indicates agreement with the camera estimate, but control errors could still accumulate over time.
- **Known dynamics.** The filter uses the simulator's parameters. Uncertainty about masses, friction, actuator response, or timing could make predictions less accurate between frames or during delays.
- **Calibration coverage and uncertainty.** Calibration was performed once per camera configuration using clean trajectories from the seed-42 policy. The reported uncertainty is conditional on these calibrations and does not quantify variation from independently repeating the procedure. The pooled angle FPR is 1.86% without attack and 2.29% on honest values within all-component FDI episodes, exceeding the 0.5% calibration target. Independent calibration sets, broader policy coverage, or online threshold adjustment would require further evaluation.
- **Image degradation.** Pole angle estimates become less accurate at $\sigma = 32$ and above, yet the visual rejection rule accepts every processed frame in the evaluated configurations. It therefore provides no effective protection against these inaccurate measurements. A measurement-quality check, denoising, or learned pose estimation could help, but each would need separate evaluation.
- **Evaluation scope.** The study covers one simulated plant, ten trained runs, fixed evaluation episodes, and a 10 s horizon. Longer runs are needed to assess stability. Performance on other plants and timing in physical cloud-based systems require separate testing.

7.4. Future Work

Future work will examine how attacks on the camera channel affect state estima-

tion and control. We will evaluate whether the controller can maintain balance when camera frames are manipulated, both when the numerical measurements are accurate and when they are also under attack.

8. Conclusions

A trusted visual channel substantially improved the tested DQN control loop under intermittent numerical false data injection. Image-based pose measurements, action-driven Kalman filtering, and componentwise replacement gave 9.87 s mean survival within a 10 s horizon under all-component FDI. The corresponding means were 4.17 s for FDI-aware DQN, 3.22 s for LSTM reconstruction, and 2.73 s for clean DQN. Camera-only control reached 9.67 s, showing that much of the benefit came from the additional state information. The smaller mean gain from selective replacement was not statistically established in the main study. Post hoc comparisons supported a gain at 500 ms latency and $\sigma = 64$ after correction across the 18 camera settings, although both pipelines failed quickly at $\sigma = 64$. Camera degradation reduced both detection and survival, while the directly tested 160×120, 25 Hz, 40 ms configuration retained high average performance. False intervention also caused failures without attack, with larger survival losses under some degraded cameras.

The practical value of camera-assisted recovery depends on whether the camera can supply a reliable estimate when the numerical readings are compromised.

Use of AI Tools

During preparation of the manuscript, the authors used OpenAI ChatGPT for grammar, spelling, style, and sentence-level clarity improvements throughout the manuscript; assistance with LaTeX formatting; generation and refinement of TikZ code used to produce **Figure 1**; and assistance in creating **Figure 2**. The research design, methodology, implementation, experiments, data analysis, results, interpretations, and conclusions were produced by the authors. All AI-assisted text, formatting, code, and figures were reviewed, revised, and verified by the authors, who take full responsibility for the manuscript.

Data Availability

Source code, trained models, calibration files, and episode-level evaluation records may be requested from the corresponding author.

Acknowledgements

This work is partially supported by the National Science Foundation under NSF Awards #2100134, #2209637, #2234911, #2417608, #2500121 and #2602903, and by the Google REACH GenAI Consortium Grant with Google Cloud Platform. Any opinions, findings and conclusions expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

Author Contributions

M.A.R. and I.J.K. conceived the study, designed the experiments, and implemented the attack model and camera-assisted recovery method. M.A.R., I.J.K., and M.H. ran the simulations, analysed the results, and prepared the figures. M.A.R. and I.J.K. validated the results. I.J.K. and M.H. wrote the original draft, and M.A.R. edited the manuscript. M.A.R. and F.W. supervised the research and secured funding for the project. All authors have read and agreed to the published version of the manuscript.

Conflicts of Interest

The authors report no conflicts of interest related to this work.

References

[1] Humayed, A., Lin, J., Li, F. and Luo, B. (2017) Cyber-Physical Systems Security—A Survey. *IEEE Internet of Things Journal*, **4**, 1802-1831. https://doi.org/10.1109/jiot.2017.2703172

[2] Liu, Y., Ning, P. and Reiter, M.K. (2011) False Data Injection Attacks against State Estimation in Electric Power Grids. *ACM Transactions on Information and System Security*, **14**, 1-33. https://doi.org/10.1145/1952982.1952995

[3] Segovia-Ferreira, M., Rubio-Hernan, J., Cavalli, A. and Garcia-Alfaro, J. (2024) A Survey on Cyber-Resilience Approaches for Cyber-Physical Systems. *ACM Computing Surveys*, **56**, 1-37. https://doi.org/10.1145/3652953

[4] Xing, W. and Shen, J. (2024) Security Control of Cyber-PHYSICAL Systems under Cyber Attacks: A Survey. *Sensors*, **24**, Article 3815. https://doi.org/10.3390/s24123815

[5] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., *et al.* (2015) Human-Level Control through Deep Reinforcement Learning. *Nature*, **518**, 529-533. https://doi.org/10.1038/nature14236

[6] Huang, S., Papernot, N., Goodfellow, I., Duan, Y. and Abbeel, P. (2017) Adversarial Attacks on Neural Network Policies. arXiv: 1702.02284.

[7] Zhang, H., Chen, H., Xiao, C., Li, B., Liu, M., Boning, D. and Hsieh, C.J. (2020) Robust Deep Reinforcement Learning against Adversarial Perturbations on State Observations. *Advances in Neural Information Processing Systems*, **33**, 21024-21037.

[8] Ilahi, I., Usama, M., Qadir, J., Janjua, M.U., Al-Fuqaha, A., Hoang, D.T., *et al.* (2022) Challenges and Countermeasures for Adversarial Attacks on Deep Reinforcement Learning. *IEEE Transactions on Artificial Intelligence*, **3**, 90-109. https://doi.org/10.1109/tai.2021.3111139

[9] Mohan, A. and Schön, T. (2026) Toward Robust Agents: A Survey of Adversarial Attacks and Defenses in Deep Reinforcement Learning. *IEEE Access*, **14**, 14481-14497. https://doi.org/10.1109/access.2026.3657855

[10] Khan, I.J., Hossain, M., Rahman, M. and Wu, F. (2026) Resilience Analysis of Cloud-Controlled Cyber-Physical Systems Using a Reinforcement Learning Cartpole Testbed. 2026 *IEEE World AI IoT Congress* (*AIIoT*), Seattle, 20-22 May 2026, 773-779. https://doi.org/10.1109/aiiot68874.2026.11569586

[11] Ren, S., Tasnim, T., Wu, B., Rahman, M. and Wu, F. (2025) Q-Learning Performance on the CartPole Environment under Observation Noise and Reward Variants. *Pro-*

ceedings of MODERN SYSTEMS 2025: International Conference of Modern Systems Engineering Solutions, Barcelona, 26-30 October 2025, 23-28.

[12] Khan, I.J., Hossain, M., Rahman, M. and Wu, F. (2026) Blockchain-Anchored Detection and Recovery for Resilient Reinforcement Learning-Based Cyber-Physical Systems. *Proceedings of the 7th National HBCU Blockchain, Fintech & AI Conference*, Nashville, 8-10 Novermber 2026, 1-15. https://doi.org/10.68414/omvn2247

[13] Rahman, M.A., Khan, I.J. and Wu, F. (2026) Detection of Integrity Attacks in Cloud-Based Cyber-Physical Systems (CBCPS) Using Long Short-Term Memory (LSTM) Models. *Proceedings of the* 2026 *9th International Conference on Information and Computer Technologies*, Honolulu, 11-13 March 2026, 533-538. https://doi.org/10.1145/3803291.3803303

[14] Mo, Y. and Sinopoli, B. (2009). Secure Control against Replay Attacks. 2009 *47th Annual Allerton Conference on Communication, Control, and Computing* (*Allerton*), Monticello, 30 September-2 October 2009, 911-918. https://doi.org/10.1109/allerton.2009.5394956

[15] Mo, Y., Chabukswar, R. and Sinopoli, B. (2014) Detecting Integrity Attacks on SCADA Systems. *IEEE Transactions on Control Systems Technology*, **22**, 1396-1407. https://doi.org/10.1109/tcst.2013.2280899

[16] Teixeira, A., Shames, I., Sandberg, H. and Johansson, K.H. (2015) A Secure Control Framework for Resource-Limited Adversaries. *Automatica*, **51**, 135-148. https://doi.org/10.1016/j.automatica.2014.10.067

[17] Zhang, J., Pan, L., Han, Q., Chen, C., Wen, S. and Xiang, Y. (2022) Deep Learning Based Attack Detection for Cyber-Physical System Cybersecurity: A Survey. *IEEE/CAA Journal of Automatica Sinica*, **9**, 377-391. https://doi.org/10.1109/jas.2021.1004261

[18] Kravchik, M. and Shabtai, A. (2022) Efficient Cyber Attack Detection in Industrial Control Systems Using Lightweight Neural Networks and PCA. *IEEE Transactions on Dependable and Secure Computing*, **19**, 2179-2197. https://doi.org/10.1109/tdsc.2021.3050101

[19] Zhang, L., Lu, P., Kong, F., Chen, X., Sokolsky, O. and Lee, I. (2021) Real-Time Attack-Recovery for Cyber-Physical Systems Using Linear-Quadratic Regulator. *ACM Transactions on Embedded Computing Systems*, **20**, 1-24. https://doi.org/10.1145/3477010

[20] Fawzi, H., Tabuada, P. and Diggavi, S. (2014) Secure Estimation and Control for Cyber-Physical Systems under Adversarial Attacks. *IEEE Transactions on Automatic Control*, **59**, 1454-1467. https://doi.org/10.1109/tac.2014.2303233

[21] Pajic, M., Weimer, J., Bezzo, N., Tabuada, P., Sokolsky, O., Lee, I., *et al.* (2014) Robustness of Attack-Resilient State Estimators. 2014 *ACM/IEEE International Conference on Cyber-Physical Systems* (*ICCPS*), Berlin, 14-17 April 2014, 163-174. https://doi.org/10.1109/iccps.2014.6843720

[22] Ivanov, R., Pajic, M. and Lee, I. (2014) Resilient Multidimensional Sensor Fusion Using Measurement History. *Proceedings of the 3rd International Conference on High Confidence Networked Systems*, Berlin, 15-17 April 2014, 1-10. https://doi.org/10.1145/2566468.2566475

[23] Gu, N., Xing, F. and You, Z. (2021) GNSS Spoofing Detection Based on Coupled Visual/Inertial/GNSS Navigation System. *Sensors*, **21**, Article 6769. https://doi.org/10.3390/s21206769

[24] Petro, P. and Bauer, P. (2024) Mono Camera-Based GPS Spoofing Detection for Aerial Vehicles. *IFAC-PapersOnLine*, **58**, 126-131. https://doi.org/10.1016/j.ifacol.2024.07.205

[25] Liang, Y., Sun, Y., Zheng, R. and Huang, F. (2022) Efficient Adversarial Training without Attacking: Worst-Case-Aware Robust Reinforcement Learning. *Proceedings of the* 36*th International Conference on Neural Information Processing Systems*, New Orleans, 28 November-9 December 2022, 22547-22561. https://doi.org/10.52202/068431-1638

[26] Wu, C., Pan, W., Staa, R., Liu, J., Sun, G. and Wu, L. (2023) Deep Reinforcement Learning Control Approach to Mitigating Actuator Attacks. *Automatica*, **152**, Article ID: 110999. https://doi.org/10.1016/j.automatica.2023.110999

[27] Hochreiter, S. and Schmidhuber, J. (1997) Long Short-Term Memory. *Neural Computation*, **9**, 1735-1780. https://doi.org/10.1162/neco.1997.9.8.1735

[28] Chaumette, F. and Hutchinson, S. (2006) Visual Servo Control. I. Basic Approaches. *IEEE Robotics & Automation Magazine*, **13**, 82-90. https://doi.org/10.1109/mra.2006.250573

[29] Buchnik, I., Revach, G., Steger, D., van Sloun, R.J.G., Routtenberg, T. and Shlezinger, N. (2024) Latent-KalmanNet: Learned Kalman Filtering for Tracking from High-Dimensional Signals. *IEEE Transactions on Signal Processing*, **72**, 352-367. https://doi.org/10.1109/tsp.2023.3344360

[30] Eykholt, K., Evtimov, I., Fernandes, E., Li, B., Rahmati, A., Xiao, C., *et al.* (2018) Robust Physical-World Attacks on Deep Learning Visual Classification. 2018 *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Salt Lake City, 18-23 June 2018, 1625-1634. https://doi.org/10.1109/cvpr.2018.00175

[31] Sato, T., Shen, J., Wang, N., Jia, Y.J., Lin, X. and Chen, Q.A. (2021) Demo: Security of Deep Learning Based Automated Lane Centering under Physical-World Attack. 2021 *IEEE Security and Privacy Workshops* (*SPW*), 11-13 August 2021, 3309-3326. https://doi.org/10.1109/spw53761.2021.00041

[32] Wei, H., Tang, H., Jia, X., Wang, Z., Yu, H., Li, Z., *et al.* (2024) Physical Adversarial Attack Meets Computer Vision: A Decade Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **46**, 9797-9817. https://doi.org/10.1109/tpami.2024.3430860

[33] Hallyburton, R.S., Liu, Y., Cao, Y., Mao, Z.M. and Pajic, M. (2022) Security Analysis of Camera-LiDAR Fusion against Black-Box Attacks on Autonomous Vehicles. *Proceedings of the* 31*st USENIX Security Symposium*, Boston, 10-12 August 2022, 1903-1920.

[34] Man, Y., Muller, R., Li, M., Celik, Z.B. and Gerdes, R. (2023) That Person Moves Like a Car: Misclassification Attack Detection for Autonomous Systems Using Spatiotemporal Consistency. *Proceedings of the* 32*nd USENIX Security Symposium*, Anaheim, 9-11 August 2023, 6929-6946.

[35] Barto, A.G., Sutton, R.S. and Anderson, C.W. (1983) Neuronlike Adaptive Elements That Can Solve Difficult Learning Control Problems. *IEEE Transactions on Systems, Man, and Cybernetics*, **13**, 834-846. https://doi.org/10.1109/tsmc.1983.6313077

[36] Towers, M., Kwiatkowski, A., Terry, J., et al. (2024) Gymnasium: A Standard Interface for Reinforcement Learning Environments. arXiv: 2407.17032.

[37] Bradski, G. (2000) The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, **25**, 120-125.

[38] Kalman, R.E. (1960) A New Approach to Linear Filtering and Prediction Problems. *Journal of Basic Engineering*, **82**, 35-45. https://doi.org/10.1115/1.3662552

[39] Pardo, F., Tavakoli, A., Levdik, V. and Kormushev, P. (2018) Time Limits in Reinforcement Learning. *Proceedings of the* 35*th International Conference on Machine*

Learning, Stockholm, 10-15 July 2018, 4045-4054.

[40] Good, P. (2005) Permutation, Parametric, and Bootstrap Tests of Hypotheses. 3rd Edition, Springer. https://doi.org/10.1007/b138696

[41] Holm, S. (1979) A Simple Sequentially Rejective Multiple Test Procedure. *Scandinavian Journal of Statistics*, **6**, 65-70.