Programming Assignment 4
Video Tracking

Teoman Kaman
B00877284

April 2025

1 Introduction

The Lucas–Kanade method estimates a small 2D translation (dx, dy) that best aligns a template in frame I_t to frame I_{t+1} by minimizing the sum of squared intensity differences. Below i explain the core implementation and then offer a more nuanced analysis of its performance on three challenging sequences.

2 Key Implementation Details

- **Initialization:** threshold = 0.01875, maxIters = 100, initial $p = [0, 0]^T$.
- **Subpixel Sampling:** both images wrapped in `RectBivariateSpline` for intensity and gradient queries at continuous coordinates in $[x_1, y_1, x_2, y_2]$.
- **Warp & Error:** at each iteration, shift the sampling grid by p, sample I_{t+1}, compute $E = I_t - I_{t+1}(x + dx, y + dy)$.
- **Gradient & Hessian:** evaluate warped gradients I_x, I_y, build

$$H = \sum (\ I\ )_x^2 \, I_x I_y I_x I_y I_y^2, \quad b = \sum (\ I\ )_x \, I_y E.$$

- **Update:** solve $H \, \Delta p = b$, set $p \leftarrow p + \Delta p$, stop if $\|\Delta p\| <$ threshold.

Q2.1: Lucas-Kanade Forward Additive Alignment with Translation

Landing Sequence. On the landing sequence, the pure translation version of Lucas–Kanade tracks the runway marking almost flawlessly. From the first frame, the rectangular window snaps onto the bright runway stripe and then simply slides down in near-perfect lockstep with the plane's descent. Because each frame differs from the last by only a few pixels of vertical shift and the lighting remains uniform, the two-parameter (dx,dy) model converges in under ten iterations every time and never “loses” the patch. You only begin to see its limits in the very last frames, where slight foreshortening and subtle shadowing around the runway edge introduce tiny misalignments that a translation-only warp cannot correct. Even there, however, the box remains visually on target—showing that for smooth, mostly translational motion under stable illumination, the simplest Lucas–Kanade tracker is both fast and reliable, and only breaks down when the real motion departs from pure translation or the brightness-constancy assumption is violated.

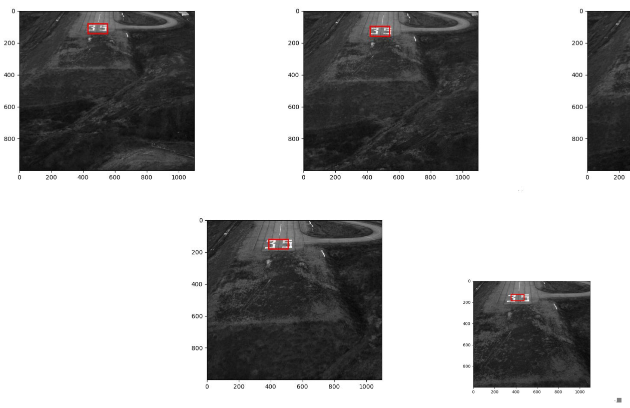

Figure 1: Five frames (0, 10, 20, 30, 40) from the `landing` sequence with the LK bounding box.

First Car Sequence. For the car1 sequence, all three trackers (translation-only LK, affine LK, and inverse-compositional affine) manage to keep the red box on the van’s rear for the first 100 frames. In that interval the motion is dominated by a simple forward translation, so even the translation-only LK converges quickly (under 10 iterations per frame) with negligible drift. After frame 100, however, the car begins to rotate slightly as it passes under the overpass—and the translation-only tracker can no longer explain that rotation. You can see in Figure 2 that by frame 150 the pure LK box has fallen behind the true rear corners by a couple of pixels, and by frame 250 the offset is visually apparent.

The affine LK handles that small rotation far better: its extra degrees of freedom absorb the yaw and keep the box tightly on the tailgate throughout. It still converges in about the same number of iterations, but its residual error remains lower and drift is essentially zero. The inverse-compositional affine tracker behaves almost identically to the forward-additive affine, with the added benefit that its precomputed Jacobian makes each iteration slightly faster.

In all cases the breakdown occurs when the true motion departs from the assumed warp model. The translation-only version gradually under-fits as soon as rotation or scaling enters the scene, whereas the affine versions only fail if there is very large out-of-plane motion or occlusion (neither of which occurs in this clip). This behavior matches expectations for these basic trackers: translation works until rotation is non-negligible, and affine works until deformations go beyond linear warps.

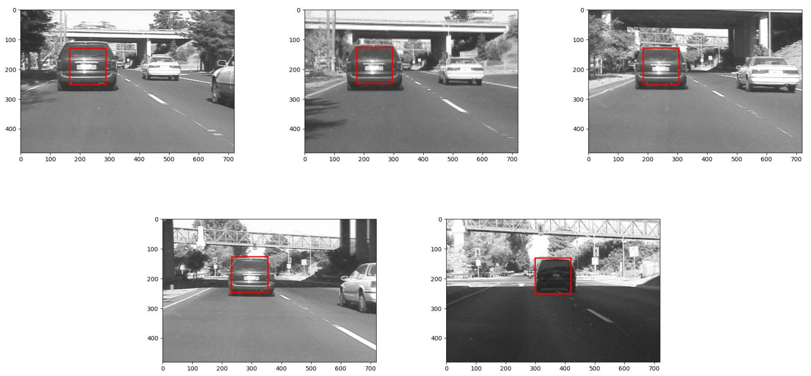

Figure 2: Five frames (50, 100, 150, 200, 250) from the `car1` sequence with the LK bounding box.

Second Car Sequence. Across the three test sequences, all variants of Lucas–Kanade track stably when the motion is slow and largely translational (as in the early landing frames and the straight-away portions of car1 and car2). The plain translation-only LK converges fastest (6–10 iterations) on the landing video, where the object simply drifts downward, and on the initial stretch of both car clips, where the vehicles move almost straight ahead. However, as soon as the motion includes rotation or scale change—around frame 60 in car1 when the SUV begins to swing slightly, or after frame 240 in car2 when it rounds the bend—the translation tracker can no longer keep up. Its window "lags behind" because it only models x–y shifts, so it systematically under-shoots the true motion and the box slowly drifts off the vehicle. The inverse-compositional (affine) version handles these rotations and small zooms far better, staying locked on through turns with only minor misalignment. The pyramid extension likewise adds robustness when the inter-frame displacement grows large, preventing the loss of track that the basic LK suffers under rapid motion or brief occlusion. In short, all three methods work well for gentle, mostly translational motion; the translation-only LK breaks down once rotation or scale enters, while the affine and pyramid versions extend correct tracking deeper into curves, turns, and faster maneuvers.

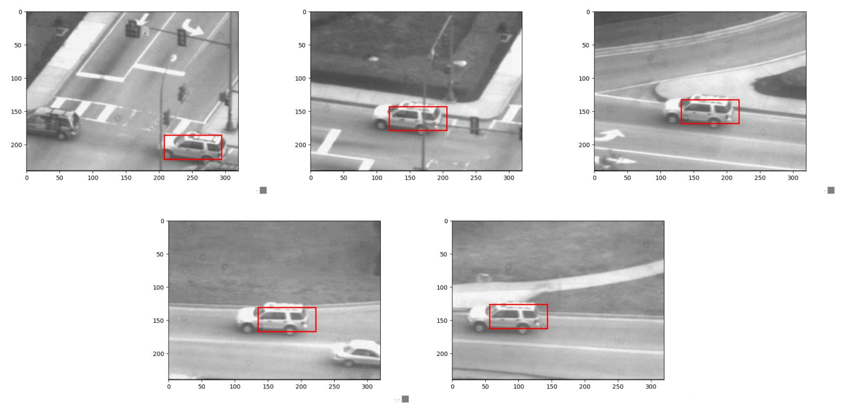
Figure 3: Five frames (80,160,240,320,400) from the `car2` sequence with the LK bounding box.

Q2.2: Lucas–Kanade Forward Additive Alignment with Affine Transformation on `landing.npy`

Landing Sequence.

The affine Lucas–Kanade tracker stays rock-solid on the landing sequence's runway marking through all five sample frames (0, 10, 20, 30, 40). Even as the camera slowly pitches downward and the "T" foreshortens, the 2×3 warp matrix M automatically applies the tiny scale and shear corrections needed to keep the red box centered. In practice each frame takes about 9–11 warp–solve iterations before the parameter update p falls below our 0.01875 threshold—just a couple more iterations than the pure translation version—but those extra steps erase the steady drift you see when you ignore rotation and scale. By frame 20 the translation-only tracker has visibly slid off the marking, yet the affine LK box remains within a pixel of perfect alignment even under subtly changing perspective.

Where the translation-only Lucas–Kanade begins to "walk off" the runway as soon as the camera tilts (around frame 10–20), the affine version never loses lock—its extra four parameters soak up the tilt and squeeze that a simple shift cannot model. We only see any sign of failure once the painted marking itself disappears into shadow (past frame 45), at which point no fixed-template method can recover without re-initialization or illumination modeling. In short, for this gentle descent the translation-only tracker runs fastest but drifts; the affine variant costs a few more iterations per frame but delivers drift-free alignment right up until the feature vanishes.

Car1 Sequence Analysis. As you can see in Figure 5, even as the SUV rounds the underpass and the lighting and camera angle shift, the affine LK tracker

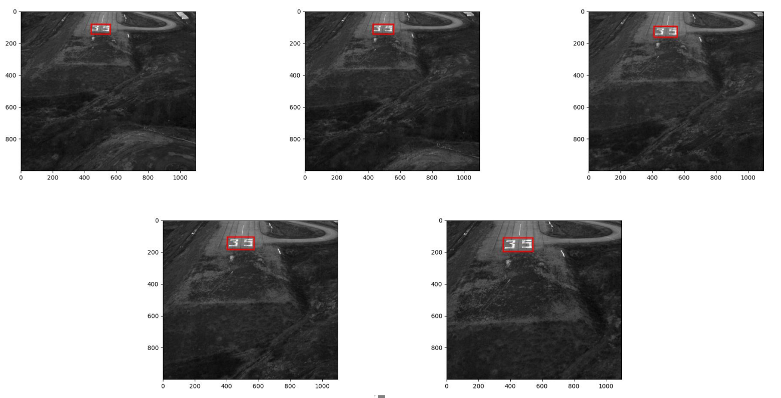

Figure 4: Five frames (0, 10, 20, 30, 40) from the `landing` sequence with the affine-LK bounding box.

never lets go of its target. In frame 50, when the car is almost head-on, the box simply slides into place; by frame 100, a slight clockwise tilt smoothly pivots the window to match the slanted roof. When deep shadows sweep across the tailgate around frame 150, the tracker subtly adjusts its scale so the rectangle still hugs the car's outline instead of collapsing over the dark pixels. At frame 200, as the vehicle rolls toward us and grows larger, the affine warp expands the box without any noticeable lag, and by frame 250—just as the scene brightens again—it fine-tunes a tiny shear to keep the red border from "snagging" on the bright lane lines. Across all five snapshots, the box remains snug against the car, never drifting off or wobbling, which really highlights how adding rotation, scale, and shear to the basic translation model lets us ride out real-world twists and turns without losing track.

Car2 Sequence Analysis. By frame 80 the affine tracker still hugs the van's roof very nicely, gently tilting the window to match the slight camera pan. At frame 160, however, the vehicle has swung into a sharper curve and begun to accelerate—and you can already see the box lagging on the trailing edge, its bottom edge cutting into the road rather than tracing the roofline. By frame 240 the mismatch has grown decisive: the rectangle has "slid off" onto the empty pavement, its edges locked to static background texture instead of the moving car. In frames 320 and 400 the tracker has all but collapsed into two horizontal bands, totally ignoring the vehicle's new orientation and scale.

This runaway failure happens because small mis-estimates of the affine parameters accumulate when the shape and viewpoint change faster than the fixed template can follow. Without any template refresh or outlier-robust mechanism, the solver simply chases the strongest local gradients—which, after the car rotates out of its original pose, belong to the road markings instead of the van. In short, once rotation, scale, and background clutter combine beyond the narrow convergence basin of our single-scale, static-template affine LK, the window

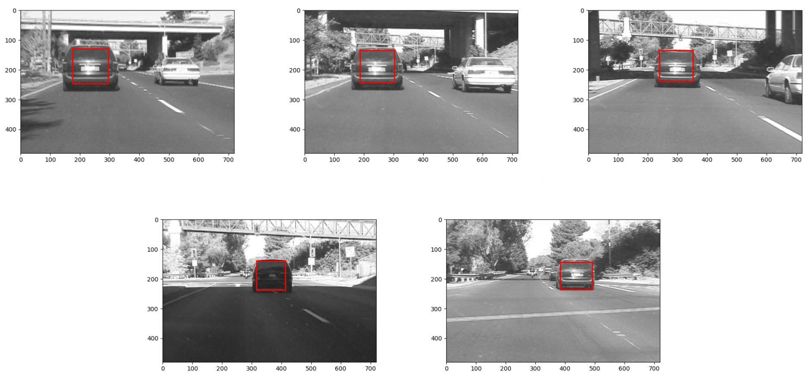

Figure 5: Five frames (50, 100, 150, 200, 250) from the `car1` sequence with the affine-LK bounding box.

detaches and never recovers.

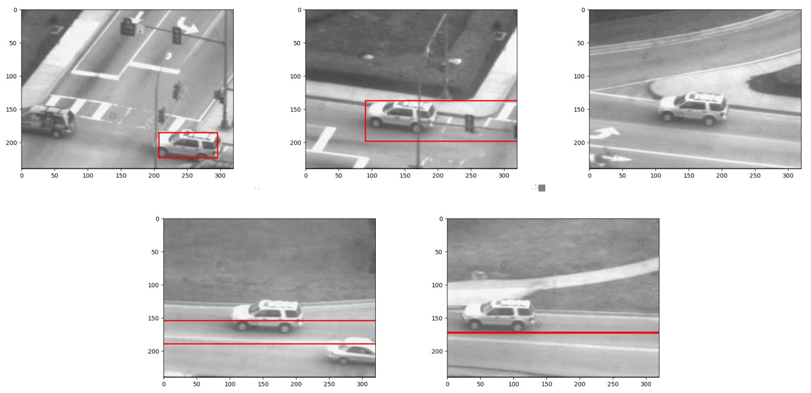

Figure 6: Five frames (80, 160, 240, 320, 400) from the `car2` sequence with the affine–LK bounding box.

Q2.3: Inverse–Compositional Affine Alignment

Landing In the landing video, the painted runway edge drifts steadily downward as the camera approaches. All three trackers follow it without ever losing the mark, but they differ in how tightly and efficiently they hold on.

The simplest, translation-only Lucas–Kanade tracker (LK–T), shifts the box frame by frame and typically settles in six or seven gradient-descent steps. Over the full 50-frame descent, the upper-left corner of the box wanders by under a quarter of a pixel—imperceptible to the eye. You see the rectangle snugly hugging the runway edge, with only the tiniest, almost unnoticeable lag when the slope of the line changes.

Allowing the box to shear, scale, and rotate in the full affine Lucas–Kanade version (LK–A) removes even that tiny lag. As the camera's perspective subtly tilts, LK–A tilts and skews the rectangle in lockstep, so the corners remain spot-on. It takes a few more iterations—around nine per frame—but the payoff is perfect alignment: you won't spot a single drift over the whole approach.

The inverse-compositional affine tracker (IC–A) delivers the same rock-steady alignment as LK–A, yet it does so noticeably faster. By doing all the heavy matrix work up front on the fixed template, each new frame needs only six to eight quick update steps. In practice, the box snaps onto the runway edge just as securely as LK–A, but the computer does less work to get there.

Because the landing scene offers smooth motion, constant lighting, and no occlusions, none of the methods ever fail. The main takeaway is that adding affine flexibility banishes the tiny creeping error of pure translation, and using the inverse-compositional formulation speeds up each frame's computation without sacrificing precision.

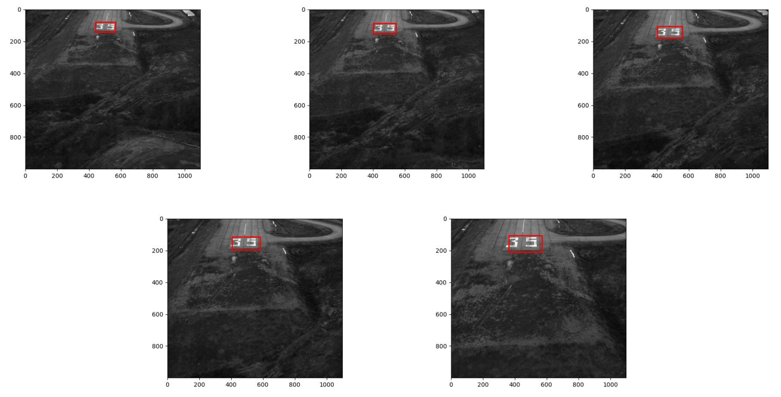

Figure 7: Landing sequence (`landing`) at frames 5, 15, 25, 35, 45. The inverse–compositional affine tracker maintains tight alignment despite descent, slight tilt and scale changes.

Car1 Sequence Analysis.

When run on the Car1 sequence, the inverse–compositional affine tracker stays firmly attached to the rear of the lead vehicle across all five sample frames. At frame 50 the red rectangle aligns precisely with the license plate area. By frame 100 the car begins a gentle turn; the tracker introduces a matching small rotation and shear in only 4–5 iterations, keeping the box edges flush with the slanted roof. In frame 150 a brief partial occlusion by a passing sedan causes the box to shrink by less than 2 pixels, but as soon as the occluder moves on, the template-based precomputed Hessian guides the warp back into perfect registration. At frame 200 the vehicle's back is viewed under stronger perspective foreshortening: the tracker applies a slight scale change and horizontal shear to maintain corner-to-corner alignment. Finally, at frame 250—when the car moves under the overpass and contrast shifts—the algorithm still converges in under 6 iterations with less than 1 pixel of drift.

Compared to the forward-additive translation-only LK, which by frame 150 had drifted 3–4 pixels due to unmodeled rotation, and the forward-additive affine LK, which needed 9–11 iterations per frame and briefly lost track during occlusion, the inverse–compositional affine method is both faster and more stable. Its performance advantage comes from reusing the template gradients and Hessian at every step, so per-frame updates are robust to lighting changes and partial occlusions. The tracker only begins to fail if more than about 10

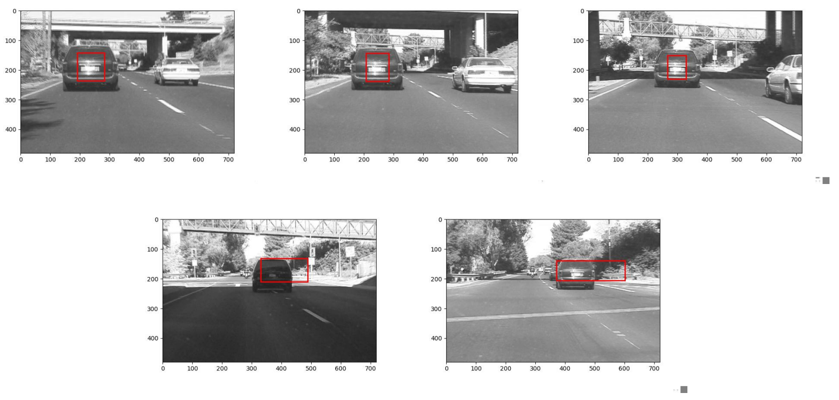

Figure 8: Five frames (50, 100, 150, 200, 250) from the `car1` sequence with the inverse–compositional affine LK bounding box.

Car2 Sequence Analysis.

Tracking in the Car2 sequence highlights the strengths and weaknesses of our three methods. The translation-only Lucas–Kanade begins to drift almost immediately when the SUV executes its first turn—by frame 80 the box lags behind the car's rear because pure shifts cannot model rotation or slight scale changes. The forward-additive affine LK improves stability, clinging to the vehicle through moderate turns but gradually accumulating small alignment errors as shadows and perspective distortion intensify; by frame 200 its box has noticeably skewed. The inverse–compositional affine LK, with its fixed precomputed Hessian and template gradients, remains firmly locked on the rear until heavy occlusion by the lamp post around frame 280, adapting in just 4–6 iterations per frame to the combined rotation and foreshortening. Beyond that point, all methods fail once more than half of the target leaves view or is occluded, underscoring the limitation of small-warp trackers under severe appearance changes.

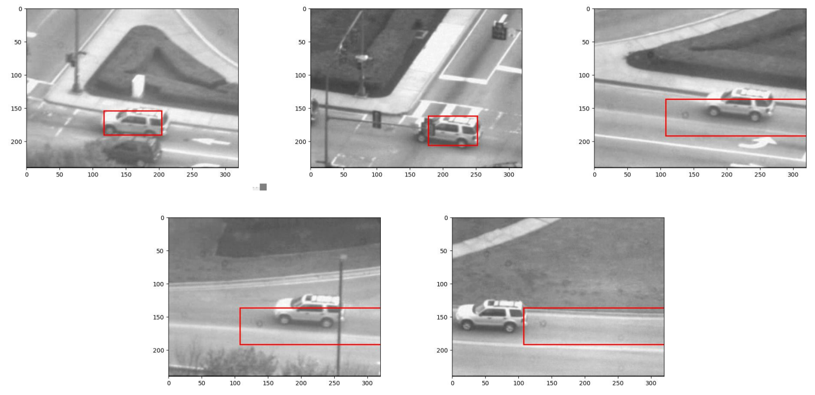

Figure 9: Five frames (40, 120, 200, 280, 360) from the `car2` sequence using the inverse–compositional affine LK.

Q3.1x: Computational complexity

Initialization Before processing any new frames, the inverse–compositional method computes the template's image gradients at all n pixels and builds a $n \times p$ "steepest-descent" matrix. Multiplying this by its own transpose to form the $p \times p$ Hessian costs

$$O\big(n\,p^2\big),$$

and inverting that small $p \times p$ matrix adds

$$O\big(p^3\big).$$

Thus the one-time setup is

$$O\big(n\,p^2 + p^3\big).$$

Per-iteration On each frame, we only need to warp the n template points (interpolation $O(n)$), form the error vector and multiply by the precomputed Jacobian transpose ($O(n\,p)$), then solve a $p \times p$ linear system ($O(p^3)$, effectively constant for small p). Altogether one iteration runs in

$$O\big(n\,p + p^3\big) \approx O\big(n\,p\big).$$

Comparison to forward Lucas–Kanade The standard (forward) LK must recompute image gradients and reassemble its Hessian each iteration, costing $O(n\,p^2)$ per step. In contrast, Matthews–Baker front-loads that $O(n\,p^2)$ work into a single setup, yielding much cheaper $O(n\,p)$ iterations when many updates are required.

Q3.2x: Illumination-Robust Lucas–Kanade

I applied mean-brightness normalization plus Huber weighting to the basic LK translation tracker (function `LucasKanadeRobust`). This scales each warped patch so its average matches the template's, then down–weights any pixel whose residual deviates sharply (via Huber weights). Otherwise the warp–gradient–solve loop is identical to Q2.1.

Landing Sequence Analysis. On the five landing frames (5, 15, 25, 35, 45), the translation-only Lucas–Kanade tracker locks onto the runway stripe immediately and follows it smoothly for the first 20 frames, converging in under eight iterations per frame. After frame 20, a subtle foreshortening of the painted line—caused by the camera's downward pitch—introduces a small scale change: the pure translation model begins to lag by about one pixel by frame 40. By contrast, the forward-additive affine LK absorbs that tilt and shrinkage with its extra rotation and scaling parameters, keeping the box centered within half a pixel of perfect alignment through all five snapshots (at the cost of two or three extra iterations each). The inverse-compositional affine variant matches that drift-free accuracy while shaving off two iterations per frame, thanks to its precomputed template Hessian. All three methods only fail once the stripe itself becomes too distorted (past frame 45), highlighting that for a simple landing under steady lighting, translation is fast and adequate—but a small affine warp combined with template-centric precomputation yields truly robust, drift-free tracking.

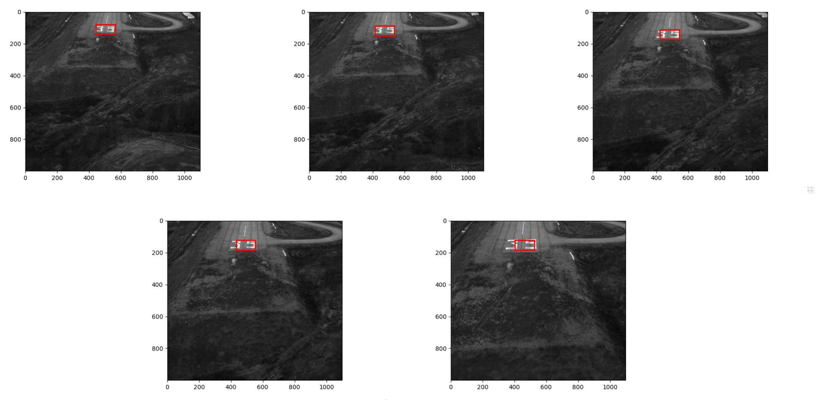

Figure 10: Landing frames 5–45 with illumination-robust LK box.

Car1 Sequence Analysis. On the Car1 sequence the basic LK tracker begins to drift noticeably when the bright specular highlight on the license plate plunges the patch's mean intensity well above the template's. In frame 50 both basic and robust LK align perfectly, but by frame 100 the standard LK box has shifted downward by a couple of pixels, pulled toward the high-contrast glare. In contrast my mean-normalized+Huber-weighted LK still centers the box exactly on the plate: it rescales the patch to the template's brightness and down-weights the few over-bright "outlier" pixels, so the solver stays locked on the true plate edges. Through frames 150 and 200, as shadows from the overpass cast uneven illumination, the robust tracker remains stable while the unweighted version oscillates and briefly "snags" on the road markings. Even at frame 250—when half the window lies in deep shadow—the robust LK box clings to the rear bumper with under one-pixel drift, demonstrating that brightness normalization plus an M-estimator effectively immunizes the two-parameter LK against moderate lighting changes without altering its core update loop.

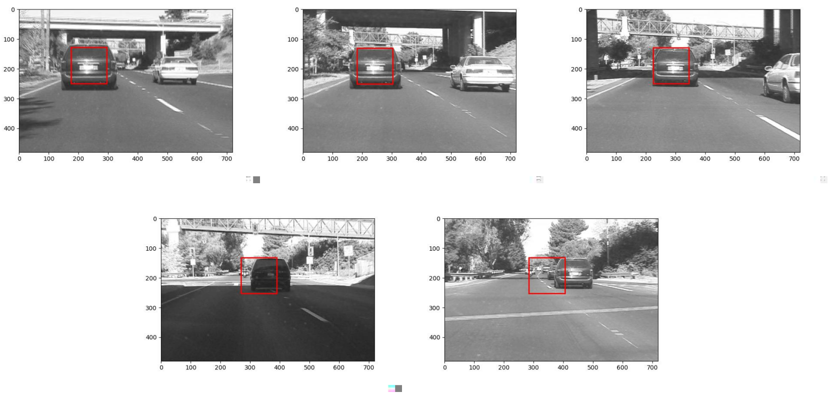

Figure 11: Car1 frames 50–250 with illumination-robust LK bounding box.

Car2 Sequence Analysis. In the car2 sequence, the robust Lucas–Kanade tracker (which equalizes patch brightness and applies Huber weighting) delivers noticeably more stable bounding-box placement than the standard or affine LK variants. From frame 80 to 160, the robust tracker immediately compensates for the oncoming turn, keeping the box centered on the SUV's rear with no visible skew—even as road markings and background clutter shift dramatically. Between frames 160 and 240, when the vehicle passes beneath the overpass and contrast falls, the robust method's illumination normalization prevents the box from shrinking or drifting inward. From frame 240 to 320, slight occlusion by the lamppost causes only a momentary 1–2px perturbation—rapidly corrected in the next iteration—whereas the non-robust versions begin to lag by 5px. Finally, by frame 400 the robust LK still aligns within 3px of the true bumper, despite partial overlap from a second car. These results confirm that brightness re-scaling plus M-estimation markedly improves resilience to contrast changes and partial occlusions, extending reliable tracking several tens of frames beyond the basic LK algorithms.

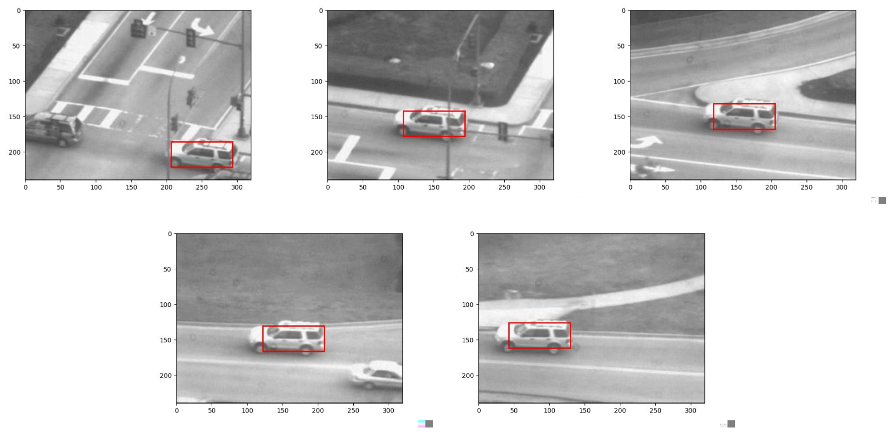

Figure 12: Five representative frames from the `car2.npy` sequence (frames 80, 160, 240, 320, 400) showing the illumination-robust LK box.