CS484 2024 Fall
HW3 Report

Lucas-Kanade Forward Additive Alignment and Kanade-Lucas-Tomasi (KLT) Tracker

Ahmet Bera Özbolat
ID: 21902777
Department: Electrical and Electronics Engineering

Bilkent University
December 2024

Contents

1 Introduction

This report presents the implementation and analysis of two fundamental algorithms in image alignment and feature tracking: the Lucas-Kanade Forward Additive Alignment and the Kanade-Lucas-Tomasi (KLT) Tracker. These methods are widely used in applications such as video stabilization, motion estimation, and object tracking in computer vision.

The **Lucas-Kanade Forward Additive Alignment** focuses on tracking a given rectangular template across frames by minimizing the pixel intensity differences between the template and the corresponding region in subsequent frames. This is achieved using a translational warp model and iterative optimization. The goal is to estimate the displacement vector $\mathbf{p} = [p_x, p_y]^\top$ that aligns the template with its new location. The algorithm leverages image gradients and a local linearization of the pixel intensity changes to iteratively refine the estimated displacement.

In contrast, the **Kanade-Lucas-Tomasi (KLT) Tracker** extends this approach by focusing on the detection and tracking of corner points, rather than a fixed rectangular template. Using the Harris Corner Detection algorithm, the KLT Tracker identifies optimal features within the bounding box of interest in the initial frame. These corner points are then tracked across frames using the Lucas-Kanade method. The bounding box is iteratively updated based on the displacements of these tracked corners.

One of the key differences between the two algorithms lies in their focus:

- The Lucas-Kanade algorithm is template-based and requires an initial rectangular region of interest, whereas the KLT Tracker is point-based and relies on identifying strong features (corners) for tracking.
- The KLT Tracker dynamically adjusts to changes in the region of interest by re-estimating the bounding box based on the tracked points, providing greater flexibility in scenarios with significant motion or deformation.

Both algorithms rely on parameter tuning, particularly the convergence threshold (ϵ) and the maximum number of iterations, to balance computational efficiency and tracking accuracy. Experimentation with these parameters plays a crucial role in achieving optimal performance. This report details the implementation of both algorithms, highlights the challenges faced, and discusses the impact of parameter selection on their performance.

2 Methodology

2.1 Lucas-Kanade Forward Additive Alignment

The **Lucas-Kanade Forward Additive Alignment** algorithm is an iterative approach for template matching and optical flow estimation. Below is a detailed explanation of its mathematical formulation and implementation steps.

Mathematical Formulation

The algorithm minimizes the error between the template $T(x)$ and the warped image $I(W(x;p))$, where $W(x;p)$ is the warping function parameterized by p. The key steps include:

1. **Warped Image:** Compute the warped image using the warping function $W(x;p)$:

$$I(W(x;p))$$

2. **Error Image:** Compute the error image as the difference between the template and the warped image:

$$[T(x) - I(W(x;p))]$$

3. **Gradient:** Compute the gradient of the image I with respect to x:

$$\nabla I(x')$$

where x' are the coordinates of the warped image.

4. **Jacobian:** Evaluate the Jacobian of the warping function W with respect to the parameters p:

$$\frac{\partial W}{\partial p}$$

5. **Hessian Matrix:** Compute the Hessian matrix H, which is defined as:

$$H = \sum_x \left[\nabla I \frac{\partial W}{\partial p}\right]^\top \left[\nabla I \frac{\partial W}{\partial p}\right]$$

6. **Parameter Update:** Compute the parameter update Δp using:

$$\Delta p = H^{-1} \sum_x \left[\nabla I \frac{\partial W}{\partial p}\right]^\top [T(x) - I(W(x;p))]$$

Update the parameters iteratively:

$$p \leftarrow p + \Delta p$$

This iterative process continues until the parameter update Δp is smaller than the convergence threshold ϵ, or the maximum number of iterations is reached. Figure 1 summarizes the key steps of the Lucas-Kanade algorithm.

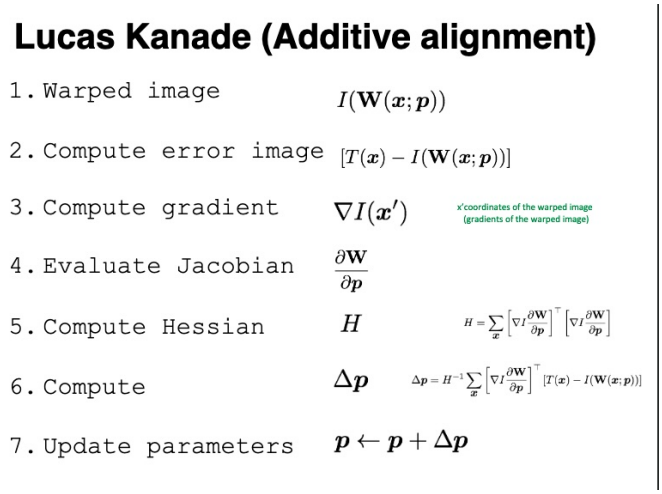

Figure 1: Lucas-Kanade Forward Additive Alignment algorithm steps.

Parameter Influence and Challenges

The performance of the Lucas-Kanade method depends on several key parameters:

- **Convergence Threshold (ϵ)**: Smaller values of ϵ improve accuracy by continuing iterations until minute changes occur. However, this increases computation time.
- **Maximum Iterations**: Higher iteration limits allow the algorithm to converge for challenging inputs but may introduce unnecessary computations when the solution is already stable.
- **Pyramid Levels**: The Pyramidal Lucas-Kanade method improves robustness for large displacements by iterating at progressively finer resolutions. Starting at a coarser level reduces the sensitivity to large motion and noise.

By experimenting with these parameters, I balanced precision and efficiency to achieve stable and accurate results.

2.2 Kanade-Lucas-Tomasi (KLT) Tracker

The **Kanade-Lucas-Tomasi (KLT) Tracker** extends the Lucas-Kanade optical flow method by incorporating optimal feature selection and pyramidal processing for robust feature tracking across video frames.

Mathematical Formulation

The KLT tracker begins with identifying **corner-like features** suitable for tracking, using the **structure tensor** of the image gradients I_x and I_y within a window W around a point:

$$G = \sum_{x \in W} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}.$$

Here:

- I_x and I_y are partial derivatives (gradients) of the image in the x- and y-directions, respectively.

The eigenvalues λ_1 and λ_2 of G determine whether the point is a good feature to track:

- **Corner (Good Feature):** Both eigenvalues are large ($\lambda_1, \lambda_2 >$ threshold).
- **Edge:** One eigenvalue is large, the other is small ($\lambda_1 \gg \lambda_2$).
- **Flat Region:** Both eigenvalues are small ($\lambda_1, \lambda_2 \approx 0$).

The Harris Corner Response R, used to filter these features, is given by:

$$R = \det(G) - k \cdot (\text{trace}(G))^2,$$

where k is an empirical constant (typically $k = 0.04$).

Lucas-Kanade Optical Flow for Corner Tracking

Once features are selected, the **Lucas-Kanade method** tracks them between two consecutive frames $I(t)$ and $I(t+1)$. The method assumes the **brightness constancy constraint**:

$$I(x, y, t) = I(x + u, y + v, t + 1),$$

where (u, v) is the displacement (motion vector) of a feature.

Expanding I using a Taylor series and ignoring higher-order terms gives:

$$I(x + u, y + v, t + 1) \approx I(x, y, t) + u \cdot I_x + v \cdot I_y.$$

The goal is to solve for (u, v) by minimizing the error over a window W centered around the feature:

$$\epsilon = \sum_{x \in W} \left[I(x + u, y + v, t + 1) - I(x, y, t) \right]^2.$$

This yields the following linear system:

$$G \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} \sum_{x \in W} I_x \cdot I_t \\ \sum_{x \in W} I_y \cdot I_t \end{bmatrix},$$

where I_t is the temporal gradient between consecutive frames. G is the same structure tensor defined earlier.

Pyramidal Lucas-Kanade Method

To handle **large displacements**, the Lucas-Kanade method is extended using a **Gaussian image pyramid**:

1. The input images are downsampled into multiple resolutions (coarse-to-fine levels).
2. At the coarsest level, the displacement (u, v) is estimated and refined iteratively.
3. The refined displacement is propagated as an initial guess to higher-resolution levels.

This coarse-to-fine strategy ensures:

- Faster convergence for large motions.
- Improved robustness to tracking errors caused by rapid or large displacements.

Implementation Summary

In my implementation:

1. **Corner Detection:**
   - The Harris Corner Detector is applied to detect strong features in the current frame.
   - A threshold is set to filter out weak features based on their eigenvalue response.
2. **Tracking via Lucas-Kanade:**
   - For each detected corner, a small **template window** around the corner is extracted:

$$W(x, y) = I(x, y) \quad \text{for} \quad x \in [x_c - w/2, x_c + w/2],$$

     where w is the window size.
   - The Pyramidal Lucas-Kanade method computes the displacement of the window in the next frame iteratively, ensuring convergence to an accurate motion vector.
3. **Bounding Box Update:**
   - The new bounding box is estimated based on the **mean position** of the tracked corners:

$$\bar{x} = \frac{1}{N} \sum_{i=1}^{N} x_i, \quad \bar{y} = \frac{1}{N} \sum_{i=1}^{N} y_i,$$

     where N is the number of tracked corners.
   - The bounding box is centered around the new mean position while maintaining its original size.

Parameter Influence and Challenges

The quality of the KLT tracker depends on:

- **Corner Detection Threshold:** Determines the minimum eigenvalue response for feature selection. A high threshold improves robustness but may reduce feature density.
- **Window Size:** Controls the size of the region used for local motion estimation. Larger windows improve noise resistance but may blur motion details.
- **Pyramid Levels:** Higher pyramid levels allow for larger displacements but increase computational cost.
- **Convergence Criteria (ϵ) and Iteration Limit:** These parameters ensure the iterative solver for (u, v) reaches an accurate solution within a reasonable time.

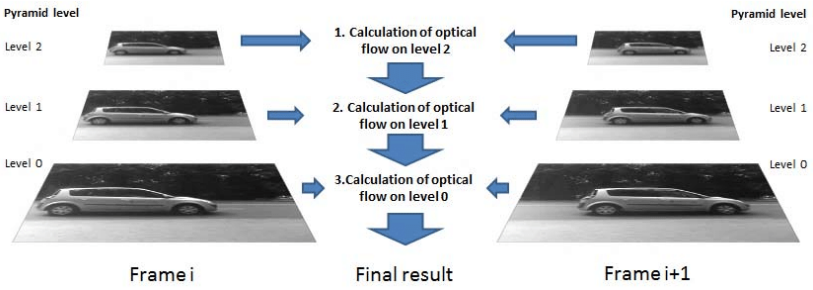

Figure 2: Pyramidal Lucas-Kanade method for coarse-to-fine optical flow estimation. [1]

This approach ensures accurate, robust feature tracking across video frames, even in scenarios involving large displacements or motion blur.

3 Experimental Results

In the implementation, I experimented with different parameters for both the Lucas-Kanade Tracker (LKT) and the Kanade-Lucas-Tomasi (KLT) Tracker. The results showcasing the best-performing parameters for each algorithm are presented in the figures below. Results for additional parameter settings can be found in the Appendix.

3.1 Lucas-Kanade Tracker (LKT)

The following figures show the tracking results of the Lucas-Kanade Tracker using the parameters that provided the best performance on three different videos:

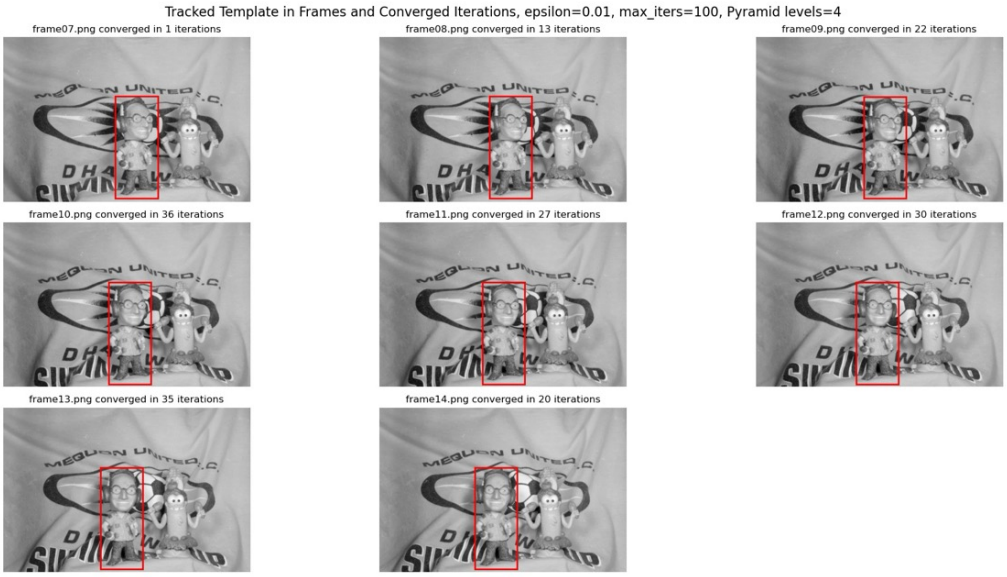

Figure 3: LKT: Tracking results on Video 1 with the best-performing parameters.

Figure 4: LKT: Tracking results on Video 2 with the best-performing parameters.

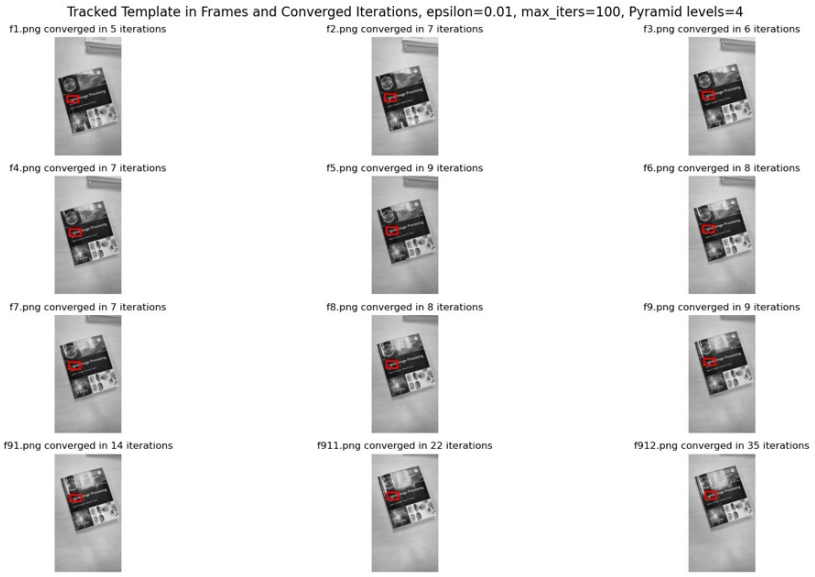

Figure 5: LKT: Tracking results on Video 3 with the best-performing parameters.

The following table presents the average number of iterations required for convergence under different values of ϵ for the Lucas-Kanade Tracker:

Table 1: LKT: Average Number of Iterations for Convergence under Different ϵ Values

ϵ	**Video 1**	**Video 2**	**Video 3**
1e−2	25	10	15
1e−4	110	123	27
2e−5	150	140	35

3.2 Kanade-Lucas-Tomasi (KLT) Tracker

The following figures show the tracking results of the Kanade-Lucas-Tomasi Tracker using the parameters that provided the best performance on three different videos:

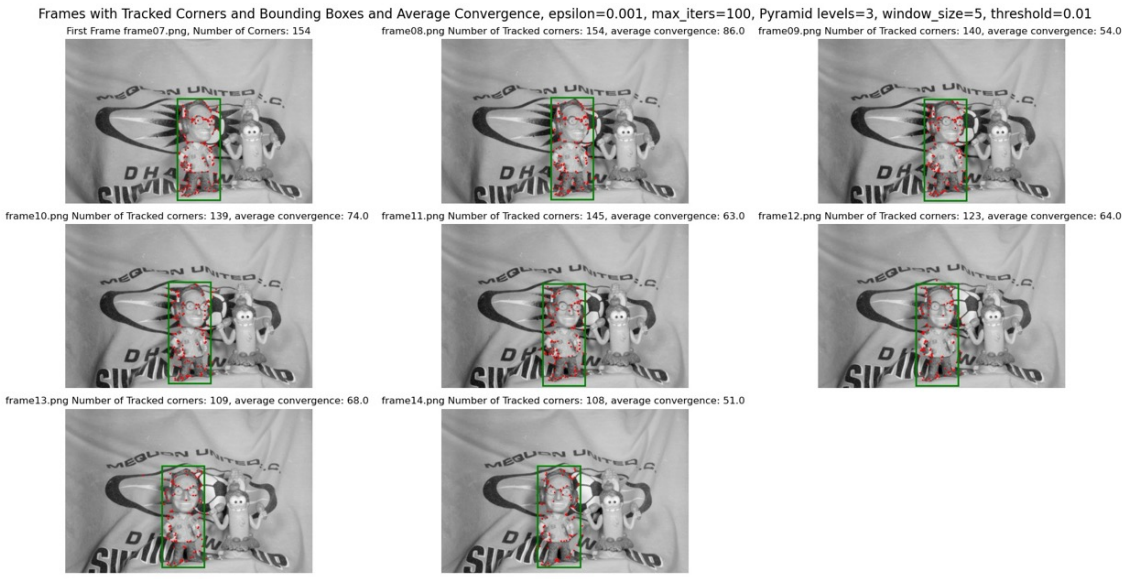

Figure 6: KLT: Tracking results on Video 1 with the best-performing parameters.

Figure 7: KLT: Tracking results on Video 2 with the best-performing parameters.

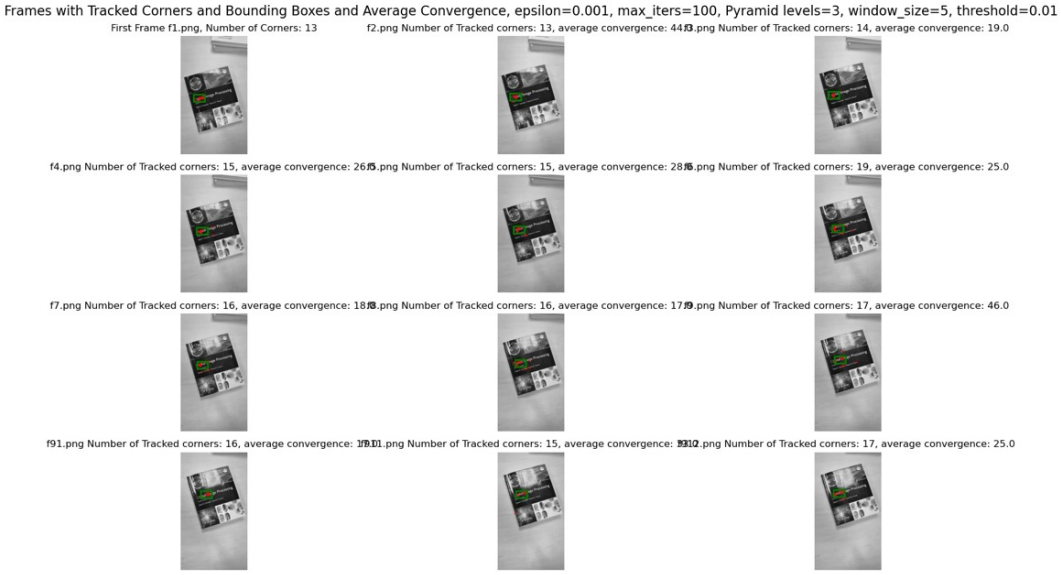

Figure 8: KLT: Tracking results on Video 3 with the best-performing parameters.

The following table presents the average number of iterations required for convergence under different values of ϵ for the Kanade-Lucas-Tomasi Tracker:

Table 2: KLT: Average Number of Iterations for Convergence under Different ϵ Values

ϵ	**Video 1**	**Video 2**	**Video 3**
1e−2	35	21	25
1e−3	67	33	30
2e−5	110	67	45

The figures illustrate how well each tracker performed across different videos using optimized parameters, while the tables summarize the impact of varying ϵ values on the convergence behavior. The detailed discussion on the results and parameter influence is provided in the next section.

3.3 Effect of Parameters on Convergence and Accuracy

Effect of ϵ on Iterations: The convergence threshold (ϵ) significantly influences the number of iterations required for convergence in both LKT and KLT algorithms. As ϵ decreases, the algorithms become stricter in determining convergence, leading to an increased number of iterations. This effect was consistently observed across all videos and both algorithms.

- **LKT:** The Lucas-Kanade Tracker converged faster than the KLT Tracker across most experiments, particularly in Video 1 and Video 3, where it demonstrated higher accuracy with fewer iterations.
- **KLT:** The Kanade-Lucas-Tomasi Tracker required more iterations on average to converge due to its reliance on additional corner detection steps and tracking refinement.

Algorithm-Specific Observations:

- **LKT Performance:** The LKT achieved better accuracy in Videos 1 and 3 compared to KLT. However, both algorithms struggled to track the ball in Video 2 beyond a few frames. Specifically:
  - In Video 2, the LKT failed to track the ball after Frame 3 due to large displacements or abrupt changes in motion that exceeded the algorithm's assumptions of small, linear movement.
- **KLT Performance:** The KLT performed better than LKT in Video 2, successfully tracking the ball until Frame 4. This highlights the KLT's robustness in identifying strong corner features and initializing tracking. However, it ultimately failed to sustain tracking beyond this point, likely due to occlusions, corner drift, or a lack of detectable features.

3.4 Impact of Additional Parameters

In the KLT Tracker, additional parameters beyond ϵ also influenced the tracking results:

- **Harris Corner Detection:** Parameters such as the Harris response threshold, non-maxima suppression, and the constant k were critical in determining the quality and quantity of corner features selected for tracking.
  - A higher threshold reduced noise but resulted in fewer features, which sometimes led to tracking failures in low-texture regions.
  - A lower threshold provided more features but occasionally introduced unstable points that drifted over time.

  For consistency, these parameters were experimentally optimized and fixed for all experiments.
- **Pyramid Levels:** Both LKT and KLT utilized a multi-resolution approach with Gaussian pyramids to handle larger displacements. The number of pyramid levels was fixed at 3 across all experiments to balance computational efficiency and accuracy. Increasing the number of levels improved robustness to large motions but added computational overhead.

3.5 Summary of Results

- The Lucas-Kanade Tracker (LKT) demonstrated faster convergence and better accuracy in Videos 1 and 3.
- The Kanade-Lucas-Tomasi (KLT) Tracker outperformed LKT in Video 2 but required more iterations for convergence overall.
- Both algorithms failed to maintain tracking in Video 2 after a few frames, indicating limitations when dealing with large or non-linear motions.
- Key parameters, including ϵ, Harris corner detection settings, and pyramid levels, were found to significantly impact the results. These parameters were tuned to ensure fair comparisons across experiments.

4 Discussion

This section analyzes the strengths and limitations of the Lucas-Kanade Tracker (LKT) and the Kanade-Lucas-Tomasi (KLT) Tracker based on experimental results. Additionally, suggestions for improvements and alternative approaches are presented to address the observed challenges.

4.1 Strengths and Limitations of LKT and KLT

Lucas-Kanade Tracker (LKT): The LKT algorithm demonstrated faster convergence compared to KLT, particularly for Videos 1 and 3. This performance can be attributed to the following:

- **Efficiency:** LKT directly applies the Lucas-Kanade method without the overhead of feature detection. The iterative minimization process for optical flow estimation converges faster when the initial conditions are well-posed.
- **Sensitivity to ϵ:** As seen in the results, decreasing ϵ increases the accuracy by enforcing stricter convergence criteria. However, this also increases the number of iterations required, leading to longer computation times and reduced performance in real-time applications.

Despite its efficiency, LKT has limitations:

- **Assumption of Small Motions:** LKT assumes small, linear displacements between frames. Large or abrupt motions (e.g., Video 2) violate this assumption, causing tracking failures after a few frames.
- **Sensitivity to Noise:** The algorithm is susceptible to noise or distracting features in the frame, which can degrade tracking performance.

Kanade-Lucas-Tomasi Tracker (KLT): The KLT algorithm extends LKT by incorporating corner detection using the Harris operator, which allows it to identify features for tracking. Its strengths include:

- **Feature Selection:** KLT can identify robust features (e.g., corners) that are more stable for tracking, improving its resilience in textured regions.
- **Better Handling of Complex Scenes:** In Video 2, KLT initially outperformed LKT, as it successfully tracked the ball for one additional frame (Frame 4). This demonstrates its ability to detect and track reliable corner features even in cluttered environments.

However, KLT's limitations are evident:

- **Dependency on Corner Detection Parameters:** The performance of KLT is highly sensitive to Harris corner detection parameters, including the response threshold, non-maxima suppression, and the constant k. Improper parameter tuning can lead to unreliable feature detection.
- **Higher Computational Cost:** The additional step of feature detection, combined with iterative tracking, results in longer computation times compared to LKT.

4.2 Impact of Parameters on Performance

Effect of ϵ: As observed in the experiments, the convergence threshold ϵ plays a critical role in balancing accuracy and efficiency. Lower ϵ values increase accuracy by ensuring smaller residual errors in optical flow estimation. However, this comes at the cost of additional iterations, which increases computation time and decreases real-time applicability.

Sensitivity to Noise and Distractors: Both LKT and KLT struggled in Video 2 due to the presence of distracting features (e.g., flowers), which created noise for the algorithms. In particular:

- LKT failed after Frame 3 as it incorrectly tracked regions similar in appearance to the ball.
- KLT failed after Frame 4, likely because the detected corner features around the flowers caused tracking drift.

To verify the impact of these distractors, an additional experiment was conducted where the flowers were manually removed from the frames. The results, shown in Figure 9, demonstrate that:

- LKT successfully tracked the ball across the entire video when the flowers were removed.
- KLT did not achieve significant improvement, indicating a higher dependency on initial corner selection and less adaptability to challenging scenarios.

Figure 9: LKT successfully tracks the ball after removing flowers

4.3 Suggestions for Improvement

Based on the observed results, the following improvements and alternative approaches are suggested:

- **Preprocessing to Reduce Noise:** Techniques such as background subtraction, filtering, or masking can be applied to eliminate distracting features (e.g., flowers) before tracking begins. This preprocessing step would reduce the noise and improve tracking performance.
- **Adaptive Corner Detection (for KLT):** The KLT algorithm's dependency on fixed Harris corner detection parameters could be improved by adaptive methods that tune the threshold and k values based on the frame content. For example, dynamic thresholding could prioritize features with the highest responses.
- **Hybrid Approaches:** Combining LKT and KLT methods with object detection techniques (e.g., using deep learning-based YOLO or SSD models) could provide better initialization and improve robustness to large motions or cluttered environments.

4.4 Summary

In summary, the LKT algorithm demonstrated superior efficiency and accuracy in simple scenarios (Videos 1 and 3), while the KLT algorithm performed better in cluttered scenes (Video 2). Both algorithms are sensitive to parameters such as ϵ and external noise. By incorporating preprocessing steps, adaptive parameter tuning, and hybrid approaches, the performance of these algorithms can be further enhanced for robust tracking.

5 Conclusion

In this study, the Lucas-Kanade Tracker (LKT) and Kanade-Lucas-Tomasi (KLT) Tracker were implemented and evaluated under varying convergence thresholds ϵ and iteration values. The results demonstrated the strengths and limitations of both algorithms, while highlighting the impact of algorithm parameters on accuracy, efficiency, and robustness.

Key Findings:

- The **LKT algorithm** proved to be faster and more efficient, particularly in Video 1 and Video 3, where it achieved accurate tracking with fewer iterations. However, it failed to track the ball beyond Frame 3 in Video 2 due to the presence of distracting features (flowers).
- The **KLT algorithm**, while slower due to its reliance on corner detection, initially performed better in cluttered environments (Video 2), where it successfully tracked the ball until Frame 4. However, its sensitivity to the Harris corner detection parameters limited its performance.
- Lowering the convergence threshold ϵ improved accuracy for both algorithms but increased the number of iterations required for convergence, which negatively impacted computational efficiency.

Challenges Faced: The project presented significant challenges that required careful problem-solving and experimentation. Initially, considerable time was spent implementing the LKT algorithm correctly. After successfully building both LKT and KLT algorithms, significant effort was devoted to tracking the ball in Video 2, where distracting features caused both algorithms to fail. This issue was particularly challenging, as the algorithms became "stuck" on features resembling the ball (e.g., flowers).

An additional idea, proposed and conducted in collaboration with a classmate, involved removing the flowers from the frames to test the hypothesis that the flowers were introducing noise. This approach successfully enabled the LKT algorithm to track the ball throughout the video, proving that preprocessing to reduce noise can significantly enhance tracking performance. However, the KLT algorithm did not show any notable improvement, likely due to its higher dependency on initial corner selection.

Reflections and Takeaways: The experiments revealed several key takeaways:

- The **pyramid method** played a crucial role in improving tracking robustness for LKT. Without the pyramid approach, tracking was only successful in Video 1, where the ball motion was relatively small.
- Both algorithms are highly sensitive to parameter tuning (ϵ, pyramid levels, and Harris parameters for KLT). Achieving optimal performance requires careful experimentation and fine-tuning of these parameters.
- Preprocessing techniques, such as noise removal or background subtraction, can significantly improve tracking results, as demonstrated in the flower-removed experiment.

Conclusion: In conclusion, the LKT algorithm showed superior speed and accuracy for simpler videos, while the KLT algorithm initially performed better in cluttered environments. However, both algorithms struggled with noise and abrupt motions, particularly in Video 2. The challenges faced during this assignment, including algorithm implementation and parameter optimization, provided valuable insights into the strengths, limitations, and practical considerations of implementing feature-tracking algorithms. Future improvements, such as adaptive parameter tuning and preprocessing steps, can further enhance the robustness and accuracy of these algorithms in challenging environments.

References

[1] V. Gaidash and A. Grakovski, "Mass Centre Vectorization Algorithm for Vehicle's Counting Portable Video System," *Transport and Telecommunication*, vol. 17, no. 4, pp. 289–297, Nov. 2016. [Online]. Available: https://www.researchgate.net/publication/310817996_Mass_Centre_Vectorization_Algorithm_for_Vehicle%27s_Counting_Portable_Video_System. [Accessed: Dec. 18, 2024].

[2] K. Kitani, "Kanade-Lucas-Tomasi (KLT) Tracker," *16-385 Computer Vision*, Carnegie Mellon University, Pittsburgh, PA, USA, 2017. [Online]. Available: https://www.cs.cmu.edu/~16385/s17/Slides/15.1_Tracking__KLT.pdf. [Accessed: Dec. 18, 2024].

[3] "15.1 Tracking - KLT," *16-385 Computer Vision*, Carnegie Mellon University, Pittsburgh, PA, USA, 2017. [Online]. Available: https://www.cs.cmu.edu/~16385/s17/Slides/15.1_Tracking__KLT.pdf. [Accessed: Dec. 18, 2024].

[4] Stanford Robotics, "Tracking Algorithms," *CS223B: Computer Vision*, Stanford University, 2004. [Online]. Available: http://robots.stanford.edu/cs223b04/algo_tracking.pdf. [Accessed: Dec. 18, 2024].

[5] "Pyramids in OpenCV," *OpenCV Documentation*, 2018. [Online]. Available: https://docs.opencv.org/3.4/d4/d1f/tutorial_pyramids.html. [Accessed: Dec. 18, 2024].

Appendix

This appendix presents additional results for each video and algorithm under different ϵ values. These figures show the impact of varying convergence thresholds on the performance of the Lucas-Kanade Tracker (LKT) and Kanade-Lucas-Tomasi (KLT) Tracker.

Lucas-Kanade Tracker (LKT)

Video 1 Results:

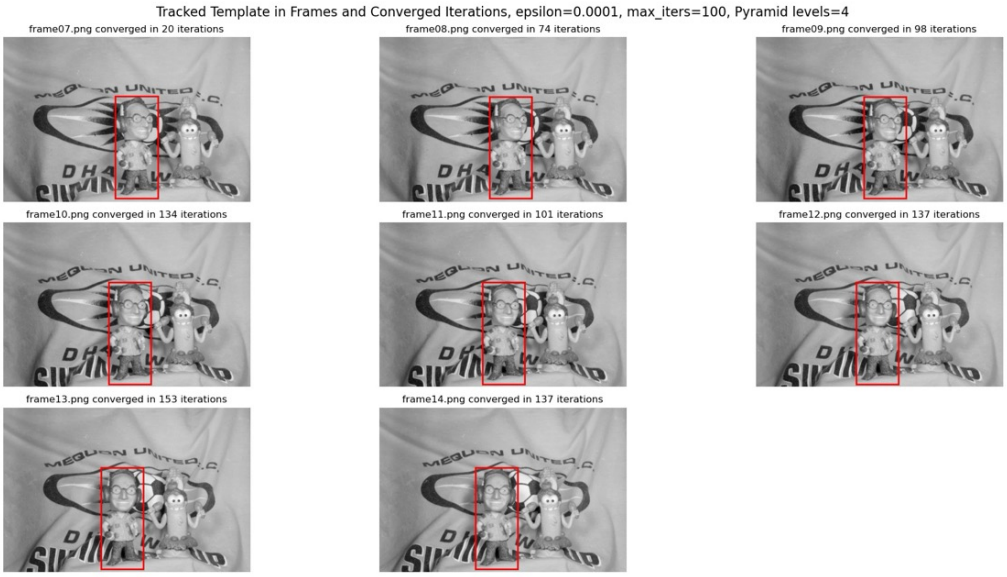

Figure 10: LKT: Video 1 tracking results with ϵ = 1e-4.

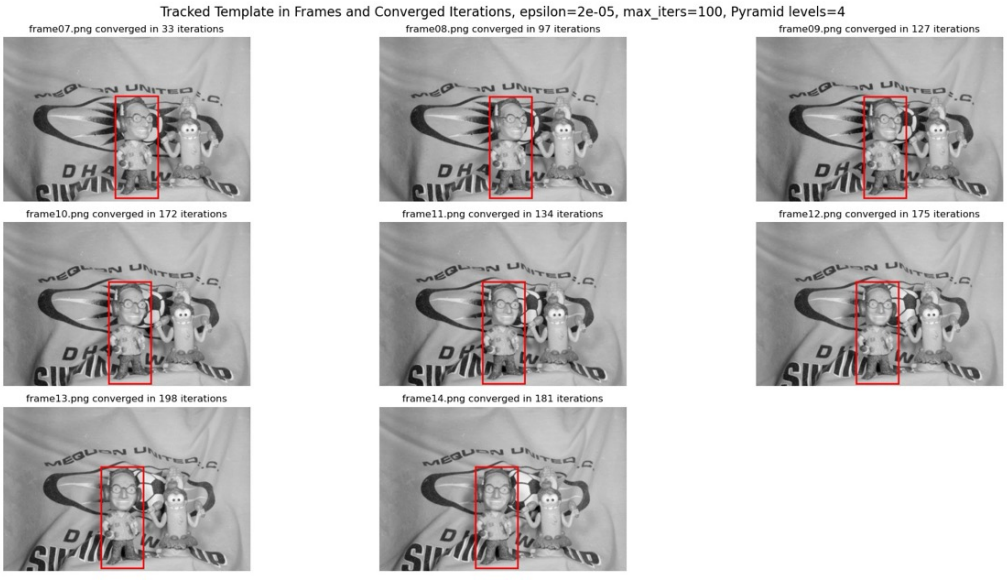

Figure 11: LKT: Video 1 tracking results with ϵ = 2e-5.

Video 2 Results:

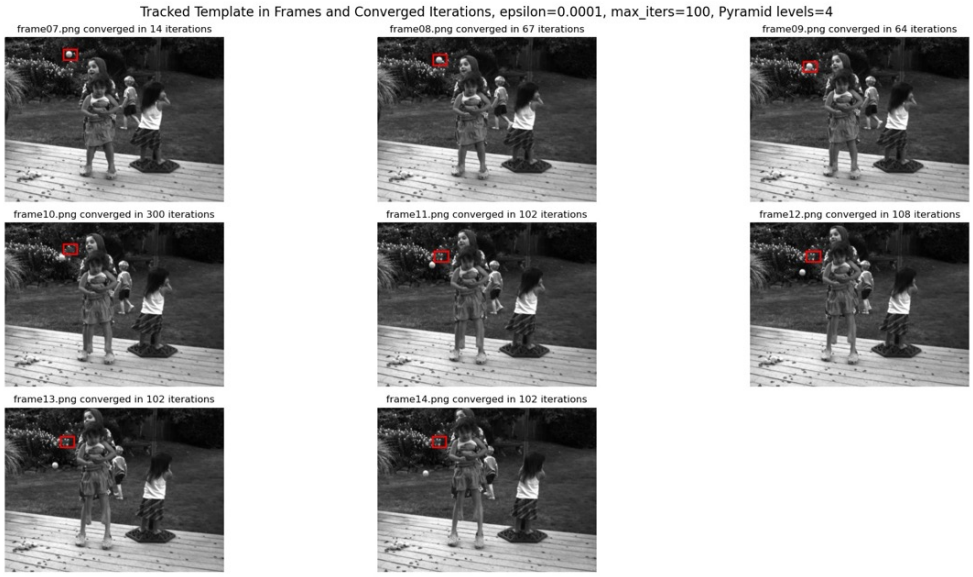

Figure 12: LKT: Video 2 tracking results with ϵ = 1e-4.

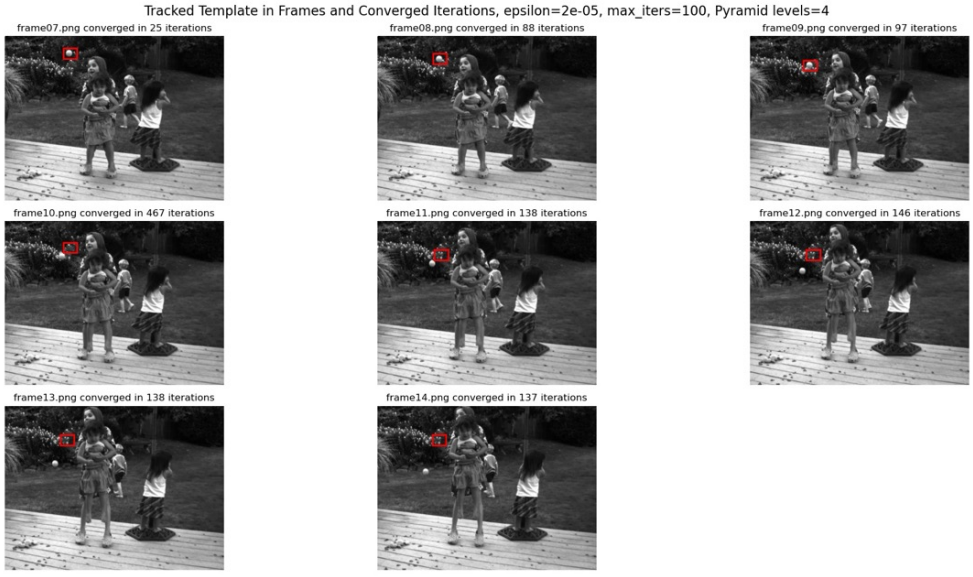

Figure 13: LKT: Video 2 tracking results with ϵ = 2e-5.

Video 3 Results:

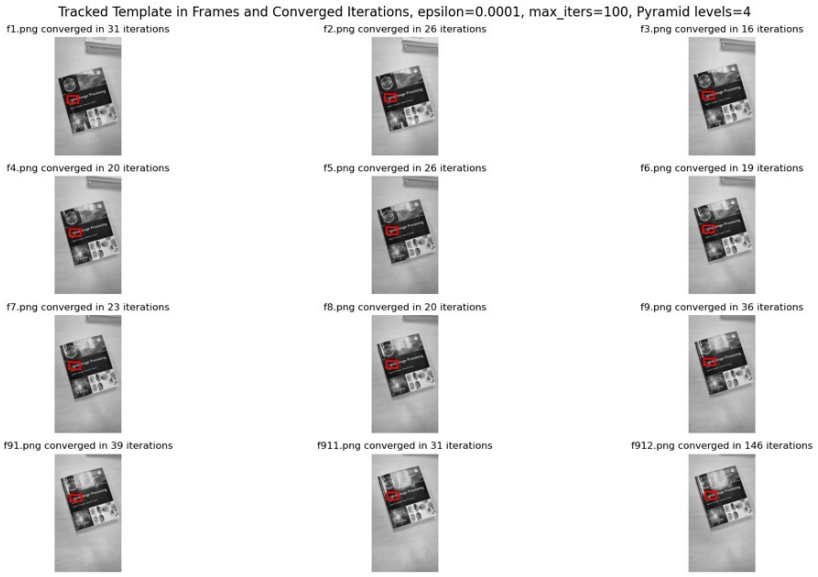

Figure 14: LKT: Video 3 tracking results with ϵ = 1e-4.

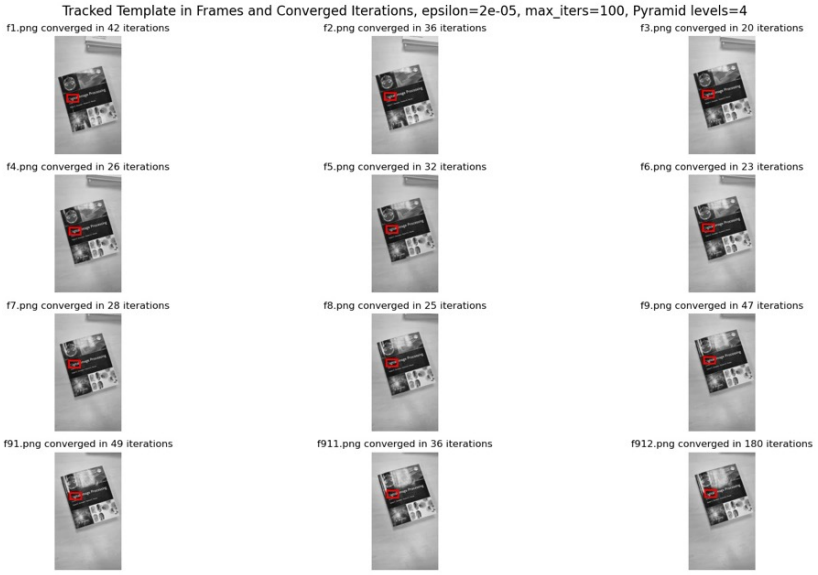

Figure 15: LKT: Video 3 tracking results with ϵ = 2e-5.

Kanade-Lucas-Tomasi Tracker (KLT)

Video 1 Results:

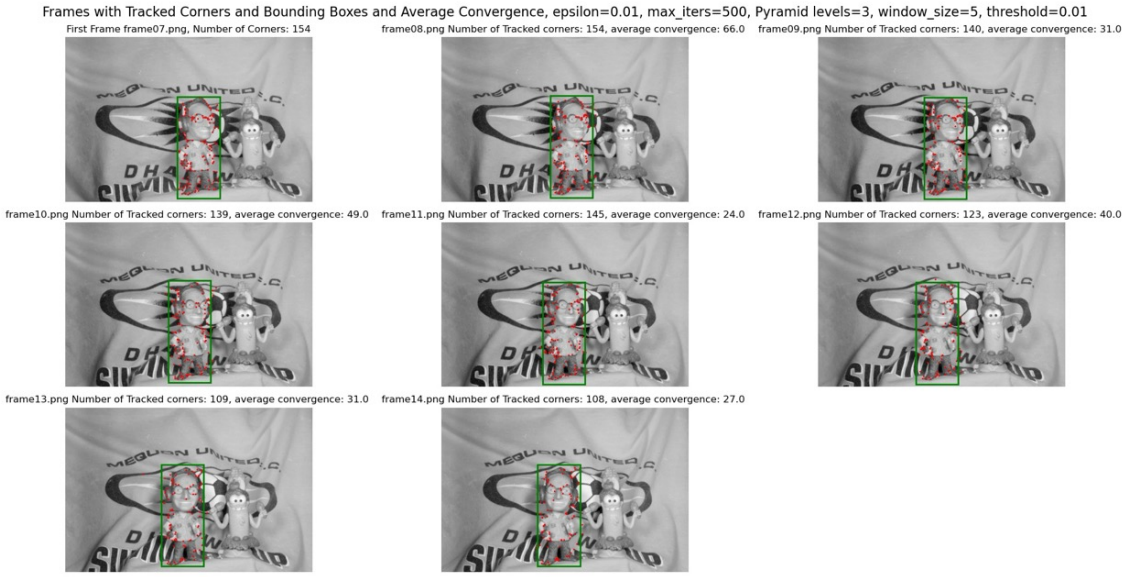

Figure 16: KLT: Video 1 tracking results with ϵ = 1e-2.

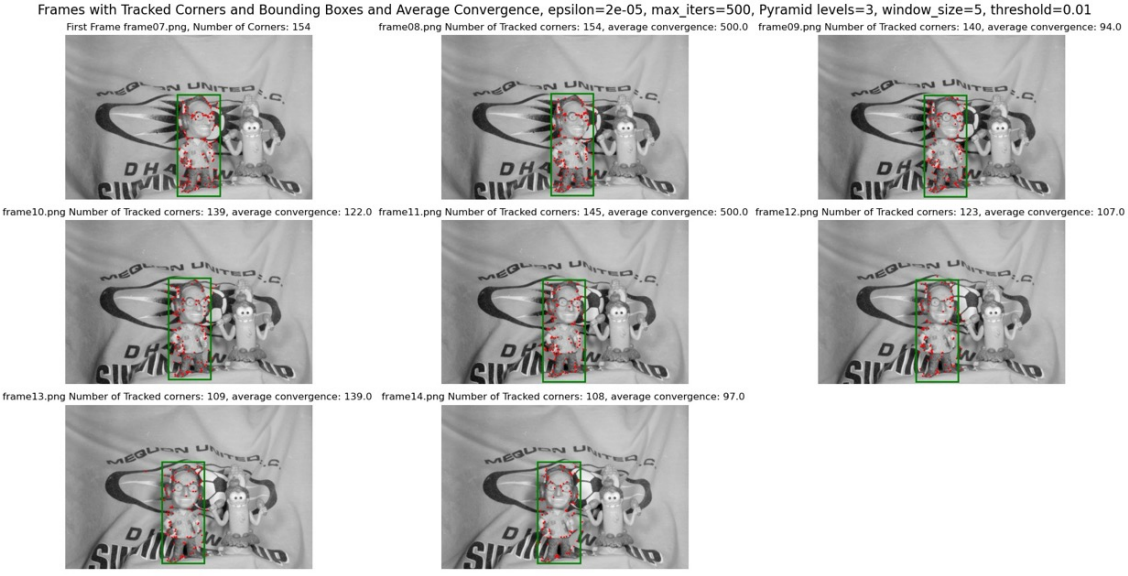

Figure 17: KLT: Video 1 tracking results with ϵ = 2e-5.

Video 2 Results:

Figure 18: KLT: Video 2 tracking results with ϵ = 1e-2.

Figure 19: KLT: Video 2 tracking results with ϵ = 2e-5.

Video 3 Results:

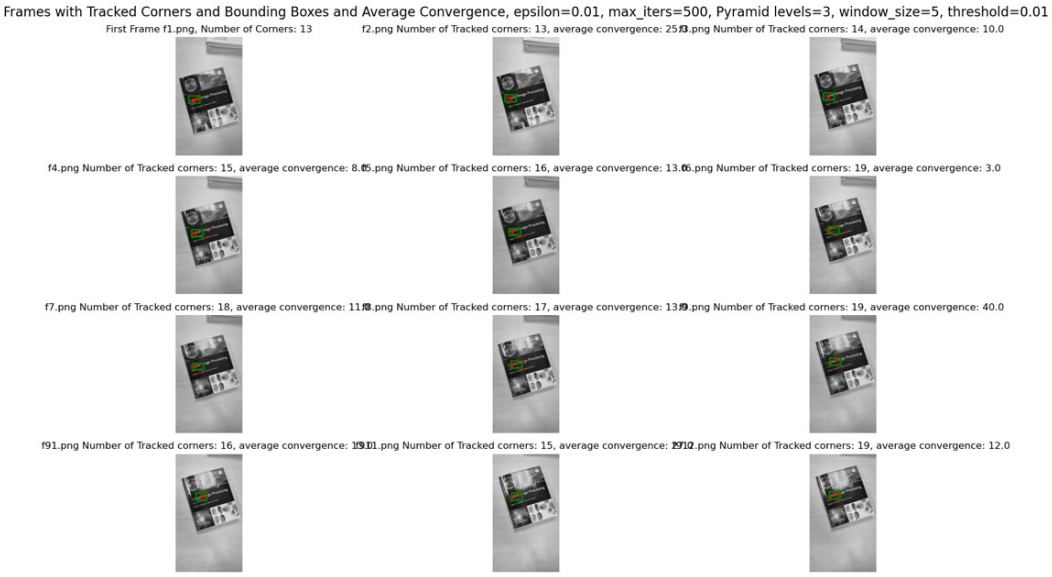

Figure 20: KLT: Video 3 tracking results with ϵ = 1e-2.

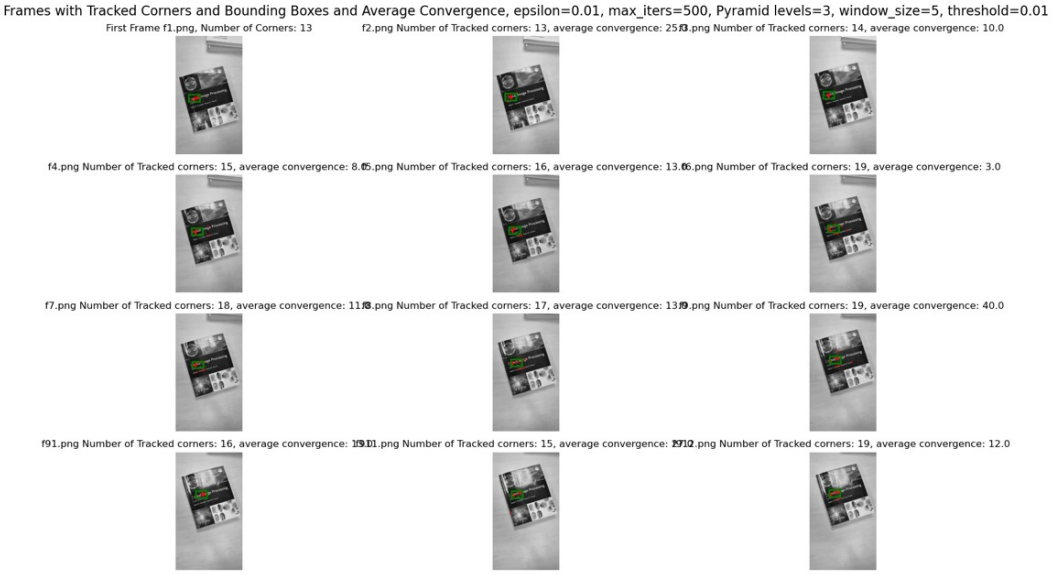

Figure 21: KLT: Video 3 tracking results with ϵ = 2e-5.