Boran Kılıç - 22103444

CS 484/555
Homework Assignment 3

1. Introduction

This assignment focuses on implementing two fundamental computer vision algorithms: Lucas-Kanade Forward Additive Alignment and the Kanade-Lucas-Tomasi (KLT) Tracker. The Lucas-Kanade method minimizes intensity differences between a template and a region in successive frames, using a translational motion model and iterative optimization. The KLT Tracker extends Lucas-Kanade by tracking multiple corner points across frames, combining corner detection (Harris Corner Detector) with motion estimation to update object positions. Unlike Lucas-Kanade, which aligns a single template, KLT estimates the motion of an entire bounding box, making it more robust for object tracking. To optimize performance, experiments were conducted with varying ϵ (convergence thresholds) and iteration limits. Fine-tuning these parameters ensures accurate tracking and computational efficiency, especially under challenging conditions like fast motion or low-contrast features.

2. Methodology

2.1 Lucas-Kanade Forward Additive Alignment

In that part, a simple Lucas - Kanade tracker with a single template is implemented. For this homework only translational warp function was dealed. The implantation is done by applying the following steps.

Starting with a rectangular neighborhood of pixels provided as templates in the homework documentation in frame $\mathbf{I_t}$ the aim was to find a displacement vector $\mathbf{p} = [\mathbf{p_x}, \mathbf{p_y}]^\mathbf{T}$ that minimizes the squared pixel difference between the corresponding region in the next frame $\mathbf{I_{t+1}}$ [1].

$$\mathbf{p}^* = \arg\min_{\mathbf{p}} \sum_{x \in \mathcal{N}} \|I_{t+1}(x + \mathbf{p}) - I_t(x)\|_2^2.$$

This objective involves aligning the region in I t+1 with the reference region in $\mathbf{I_t}$ by minimizing the pixel-wise intensity differences. The iterative process begins with an initial guess for **p** which set to $\mathbf{[0,0]^T}$.

Using a first-order Taylor expansion to approximate the intensity values the objective can be linearized as:

$$I_{t+1}(x' + \Delta\mathbf{p}) \approx I_{t+1}(x') + \nabla I_{t+1}(x') \frac{\partial W(x;\mathbf{p})}{\partial \mathbf{p}} \Delta\mathbf{p}.$$

Here, $\Delta\mathbf{p} = [\Delta p_x, \Delta p_y]^T$ represents the incremental update to the displacement vector, $\nabla I_{t+1}(x')$ represents the image gradients, and $W(x;\mathbf{p}) = x + \mathbf{p}$ denotes the warp function applied to pixel coordinates.
By combining this linearization into a matrix formulation, the optimal displacement update can be computed by solving:

$$\Delta\mathbf{p} = \arg\min_{\Delta\mathbf{p}} \|\mathbf{A}\Delta\mathbf{p} - \mathbf{b}\|_2^2$$

where **A** encodes the image gradients, and b represents the residual intensity differences. After calculating Δp he displacement is updated as $\mathbf{p} \leftarrow \mathbf{p} + \Delta\mathbf{p}$ and the process repeats until

convergence. In order to efficiently achieve convergence two parameters are defined: convergence threshold ϵ where the updating process stops when the difference is less than ϵ and maximum number of iterations (**max_iter**) where again the process is interrupted when the max number of iterations has reached. The values are determined experimentally by incrementally changing the values until a sufficient tracking output has reached.

For each dataset different parameters has been determined. Since the coordinates in the later frames are not provided in the homework documentation the success of the algorithms are assessed visually.

For initial values of ϵ and **max_iter**, 10^{-3} and 50 are determined as reasonable values respectively. For each video **max_iter** values are incremented untill convergence, this value should be as small as possible regarding the computational efficiency. However, ϵ was a bit different, when this value was too small or large, the output was not desirable. For this reason, an optimum spot is found for the ϵ values.

2.2 Kanade-Lucas-Tomasi (KLT) Tracker

In this part, similar to the previous one, the task was to perform tracking using Kanade-Lucas-Tomasi (KLT) Tracker algorithm. The KLT algorithm is an effective approach for tracking small templates or corner points across consecutive frames, relying on a straightforward translational motion model. The algortihm is implemented using the following steps [1].

2.2.1 Corner Detection

Corner points were identified in the first frame of the video sequence using the **Harris Corner Detection** algorithm. Only the corner points lying within the given bounding box of the region of interest (ROI) were selected for tracking. The bounding box coordinates were provided as part of the assignment. The Harris Corner Detection algorithm was implemented to compute corner points based on the eigenvalues of the image gradient matrix. A threshold was used to filter out weaker corners [2]. The mathematical formation of the Harris Corner Detection algorithm is provided below:
Compute the gradients of the input image I along the x and y axes using Sobel operators Sx and Sy.

$$I_x = I * S_x, \quad I_y = I * S_y$$

Where * denotes convolution.

Calculate the products of gradients:

$$I_x^2, \quad I_y^2, \quad I_x I_y$$

Smooth the gradient products using a Gaussian kernel G:

$$S_{x^2} = G * I_x^2, \quad S_{y^2} = G * I_y^2, \quad S_{xy} = G * (I_x I_y)$$

Form the structure tensor (second moment matrix) for each pixel:

$$M = \begin{bmatrix} S_{x^2} & S_{xy} \\ S_{xy} & S_{y^2} \end{bmatrix}$$

Compute the Harris corner response R using:

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

where:

$$\det(M) = S_{x^2}S_{y^2} - S_{xy}^2, \quad \text{trace}(M) = S_{x^2} + S_{y^2}$$

Threshold R values and perform non-maximum suppression to localize corners.

$$R_{\text{thresholded}} = \begin{cases} R & \text{if } R > \text{threshold}, \\ 0 & \text{otherwise.} \end{cases}$$

Retain the N strongest corners based on their response values.

2.2.2 Corner Tracking

For each selected corner point, its displacement was calculated between consecutive frames using the Lucas-Kanade method. The displacements of all corner points were averaged to estimate the new position of the bounding box in the next frame. To maintain a sufficient number of tracked points, new corner points were periodically reinitialized within the updated bounding box.

2.2.3 Iterative Updates and Parameter Tuning

The translational warp function was iteratively updated to refine the corner positions until convergence. Experiments were conducted with different values for the convergence threshold ϵ and the maximum number of iterations (**max_iter**) to balance accuracy and computational efficiency. The ϵ value determined when the iterative updates stopped based on the reduction of the error. Reasonable starting estimates for ϵ and **max_iter** values were chosen as 10^{-3} and 50 are determined respectively.

The performance of the tracker was sensitive to the initialization of parameters such as the window size, threshold for Harris corner detection, and the number of iterations for the optimization. Reinitializing corner points every few frames improved the stability of the tracker, especially when the object moved significantly between frames or when the number of valid tracked points reduced.

For fast-moving objects, such as the ball in the second video, a larger search window size and higher maximum iterations were necessary to ensure accurate tracking. Lowering the threshold for corner detection allowed more features to be tracked, but it also introduced noise, which was mitigated by filtering corners within the bounding box. Averaging the displacements of all corners improved bounding box stability.

3. Experimental Results

3.1 Lucas-Kanade Forward Additive Alignment

The results of different parameters are provided below. The results are discussed in the discussions part.

3.1.1 Video 1:

First, 3 different ϵ values are applied which are 10^{-1}, 10^{-3}, 10^{-6}:


Tracked Template Across Frames $\epsilon = 10^{-1}$, max iteration = 50
Frame 1
Frame 2
Frame 3
Frame 4
Frame 5
Frame 6
Frame 7
Frame 8


Tracked Template Across Frames $\epsilon = 10^{-3}$, max iteration = 50
Frame 1
Frame 2
Frame 3
Frame 4
Frame 5
Frame 6
Frame 7
Frame 8


Tracked Template Across Frames $\epsilon = 10^{-6}$, max iteration = 50
Frame 1
Frame 2
Frame 3
Frame 4
Frame 5
Frame 6
Frame 7
Frame 8

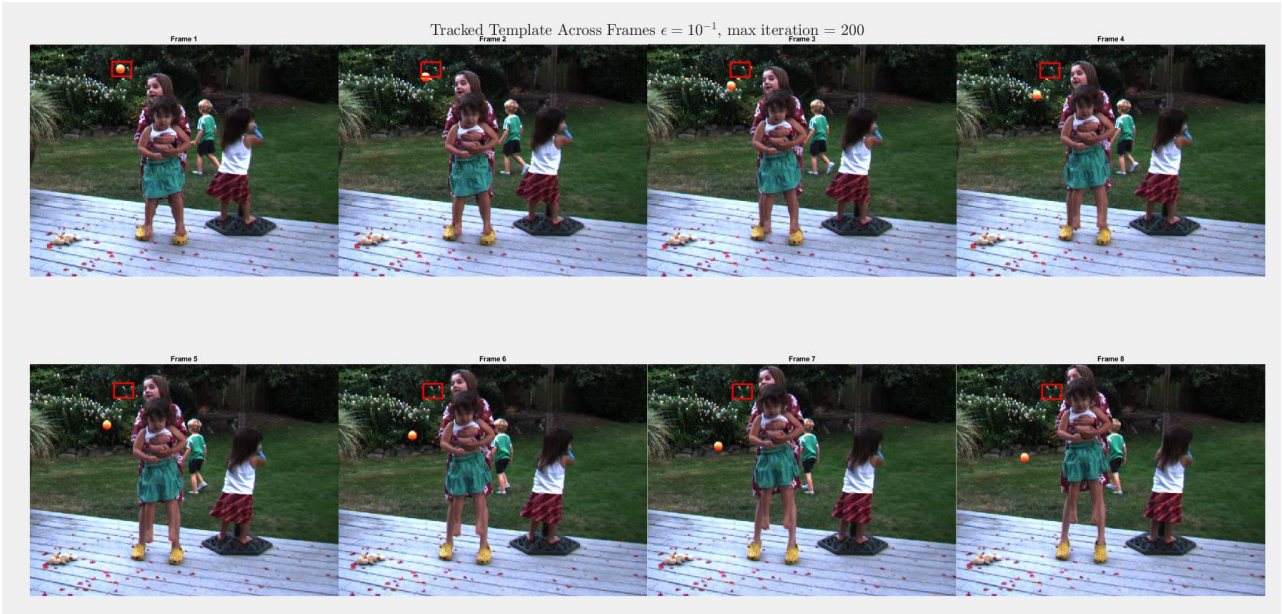
Most effective and efficient value was 10^{-1}.

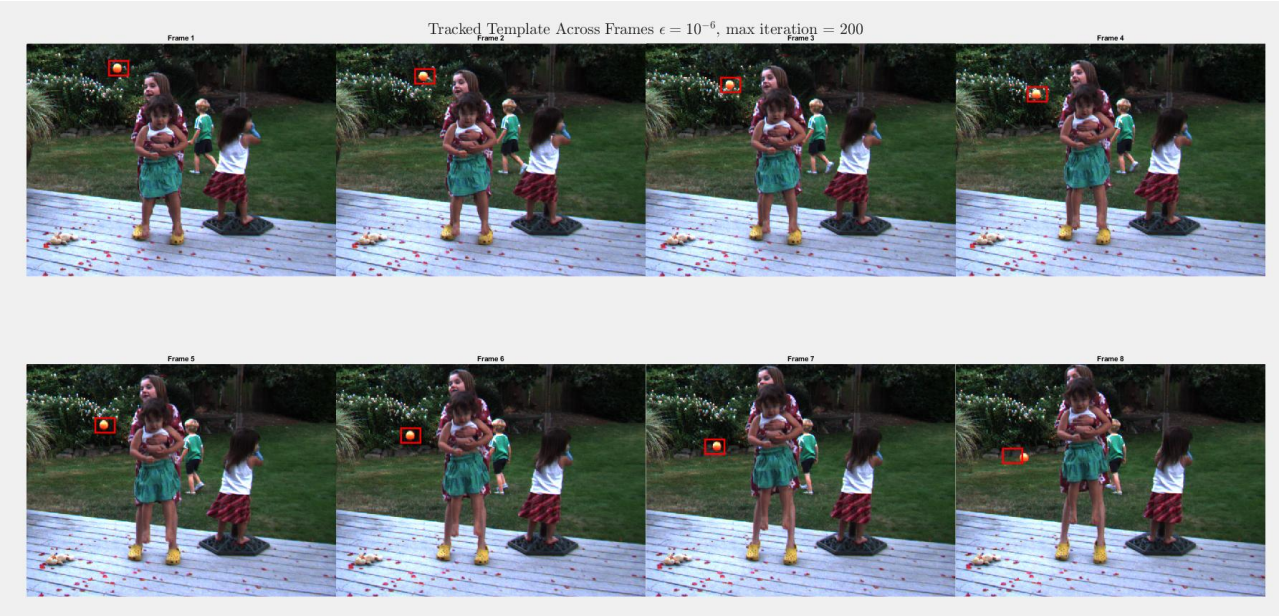
Then, 3 different maximum iteration values are applied 20, 50, 100:

Most effective and efficient value was 50.

3.1.2 Video 2:

First, 3 different ϵ values are applied which are 10^{-1}, 10^{-3}, 10^{-6}:





Most effective and efficient value was 10^{-3}.

Then, 3 different maximum iteration values are applied 50, 100, 200:


Tracked Template Across Frames $\epsilon = 10^{-3}$, max iteration = 50
Frame 1
Frame 2
Frame 3
Frame 4
Frame 5
Frame 6
Frame 7
Frame 8


Tracked Template Across Frames $\epsilon = 10^{-3}$, max iteration = 100
Frame 1
Frame 2
Frame 3
Frame 4
Frame 5
Frame 6
Frame 7
Frame 8

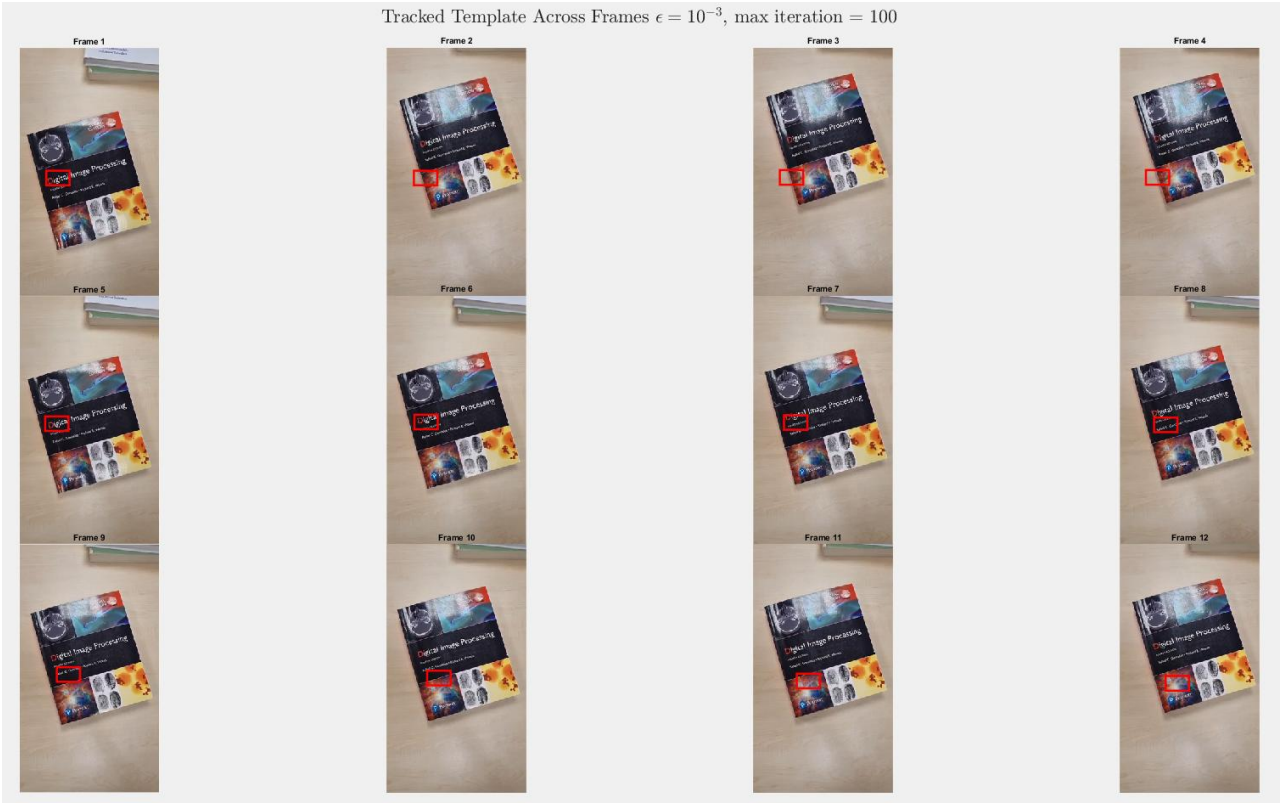
Most effective and efficient value was 200.

3.1.3 Video 3:

First, 3 different ϵ values are applied which are 10^{-1}, 10^{-3}, 10^{-6}:

Most effective and efficient value was 10^{-3}.

Then, 3 different maximum iteration values are applied 100, 200, 250:


Tracked Template Across Frames $\epsilon = 10^{-3}$, max iteration = 100
Frame 1
Frame 2
Frame 3
Frame 4
Frame 5
Frame 6
Frame 7
Frame 8
Frame 9
Frame 10
Frame 11
Frame 12


Plot
Tracked Template Across Frames $\epsilon = 10^{-3}$, max iteration = 200
Frame 1
Frame 2
Frame 3
Frame 4
Frame 5
Frame 6
Frame 7
Frame 8
Frame 9
Frame 10
Frame 11
Frame 12

Most effective and efficient value was 250.

3.2 Kanade-Lucas-Tomasi (KLT) Tracker

3.2.1 Harris Corner Detection

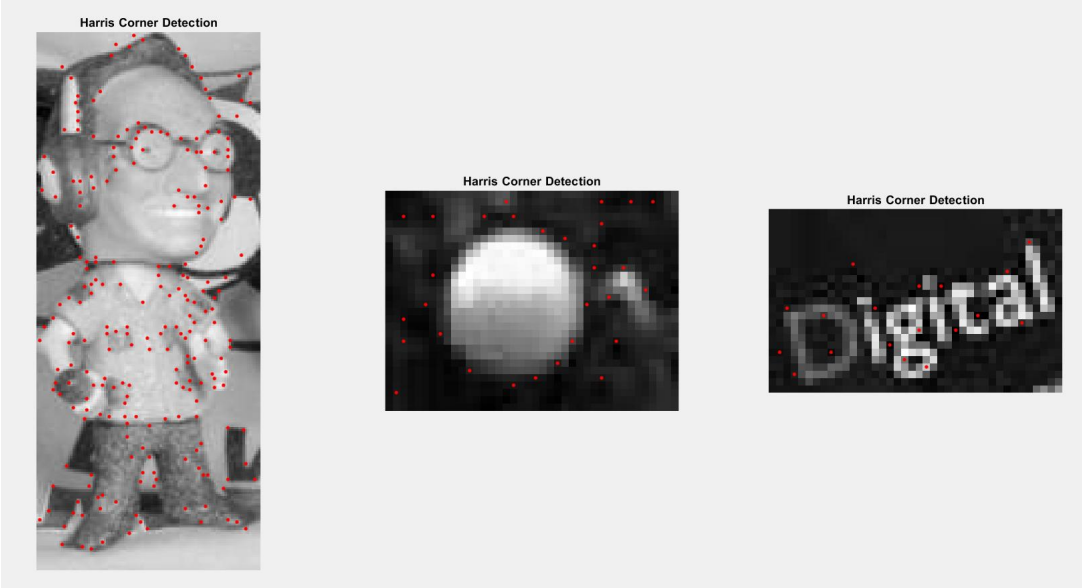
For KLT algorithm to work well the success of Harris Corner Detection was crucial for that reason the performance and parameter of the Corner detection algorithm was assessed using a separate test environment. The parameters for Harris Corner detection are:

Window size: This defines the size of the Gaussian window or neighborhood used for smoothing the gradient products. Larger window size results in more smoothing, which can reduce noise but may also blur finer details, potentially missing smaller or closely spaced corners. A smaller window size provides higher sensitivity to local variations but can be more susceptible to noise. Choosen as window_size = 4.

Sensitivity factor (k): The value of k determines the balance between detecting corners and edges. A smaller k (e.g., 0.02) makes the algorithm more sensitive to detecting corners. Larger k values (e.g., 0.04) may increase robustness but might miss some subtle corners. Choosen as k = 0.02.

N: This specifies the maximum number of strongest corners to retain after sorting based on the corner response R. If there are fewer than N corners that meet the threshold criteria, all detected corners will be included.
Adjusting N controls how many corners are considered for further processing or visualization. A higher value results in more corners being selected, while a lower value restricts it to the most prominent ones. Choosen as N = 250.

Detected corners for each image and corresponding template is given below:



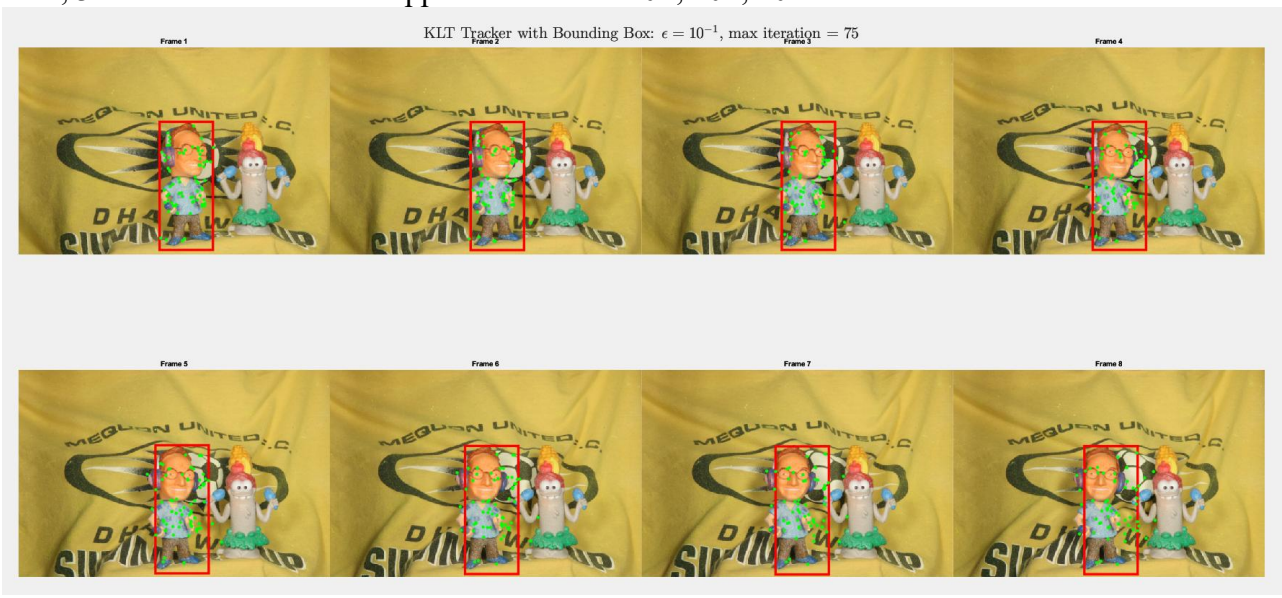
It is visible that the algorithm successfully captures the important corner points that can be used for object tracking later on.

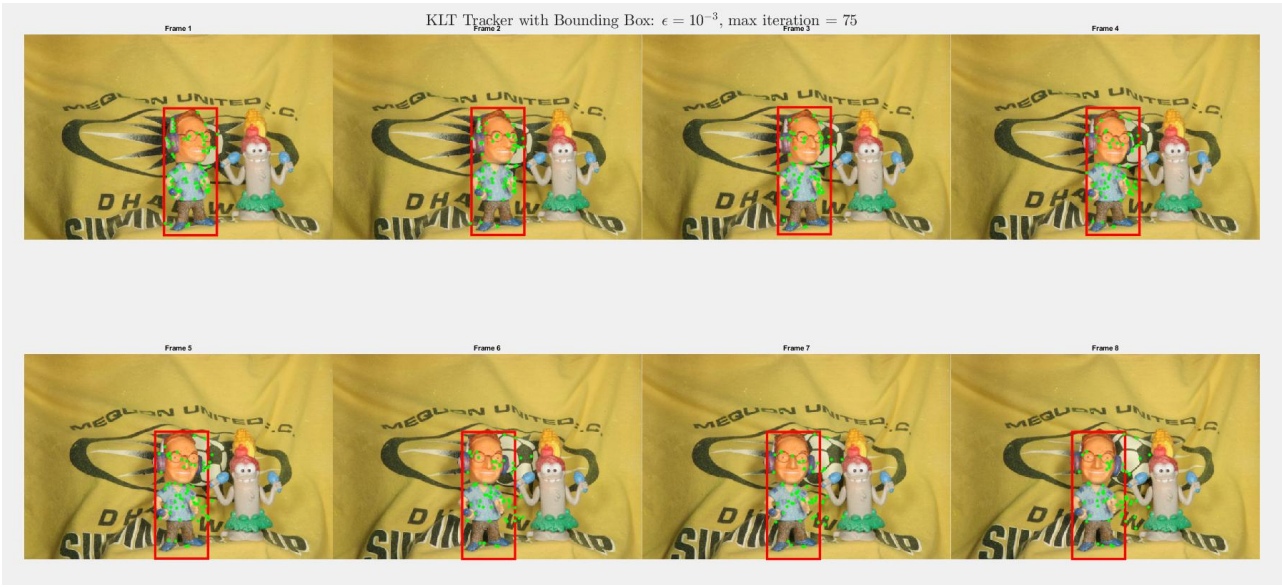
3.2.2 KLT Tracker

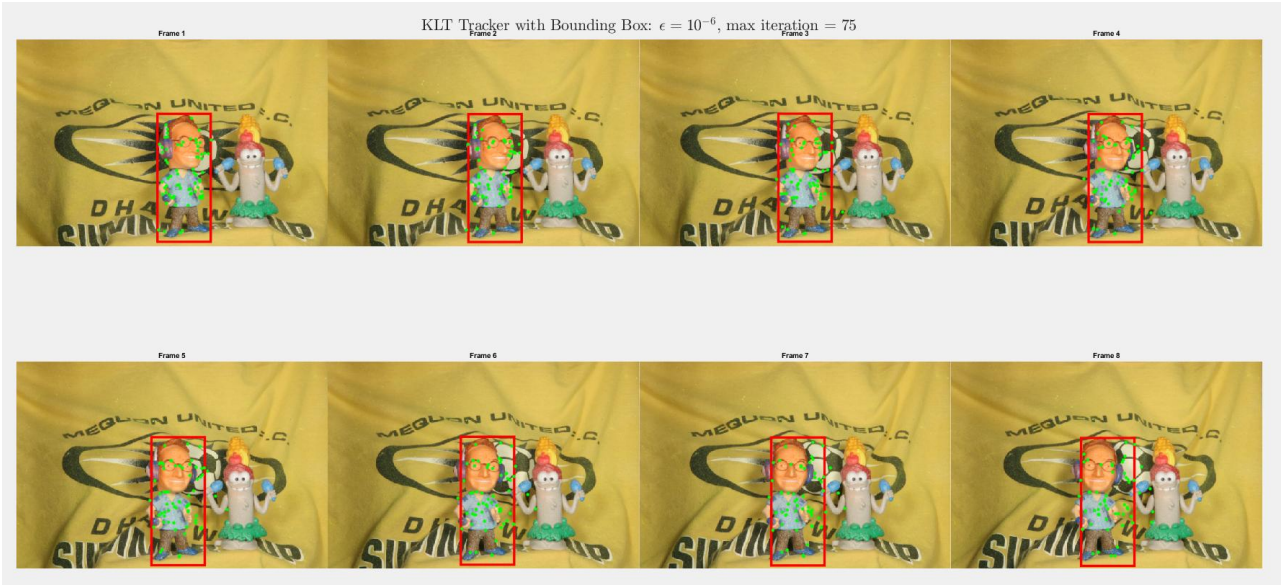
The results of different parameters are provided below. The results are discussed in the discussions part.

3.2.2.1 Video 1

First, 3 different ϵ values are applied which are 10^{-1}, 10^{-3}, 10^{-6}:

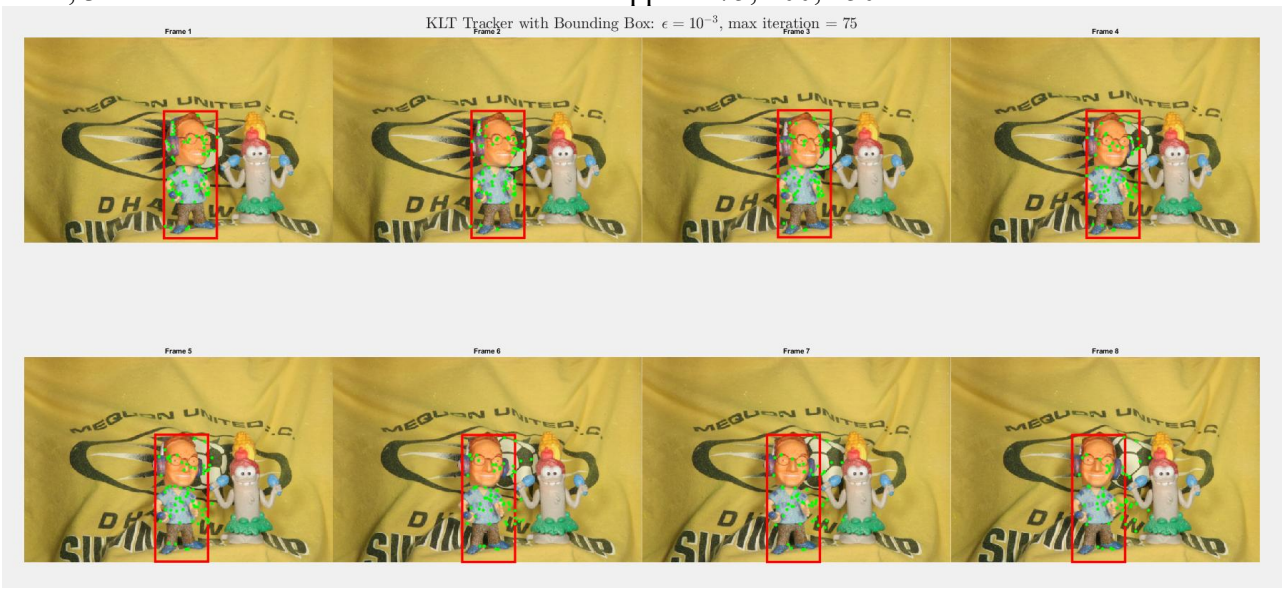


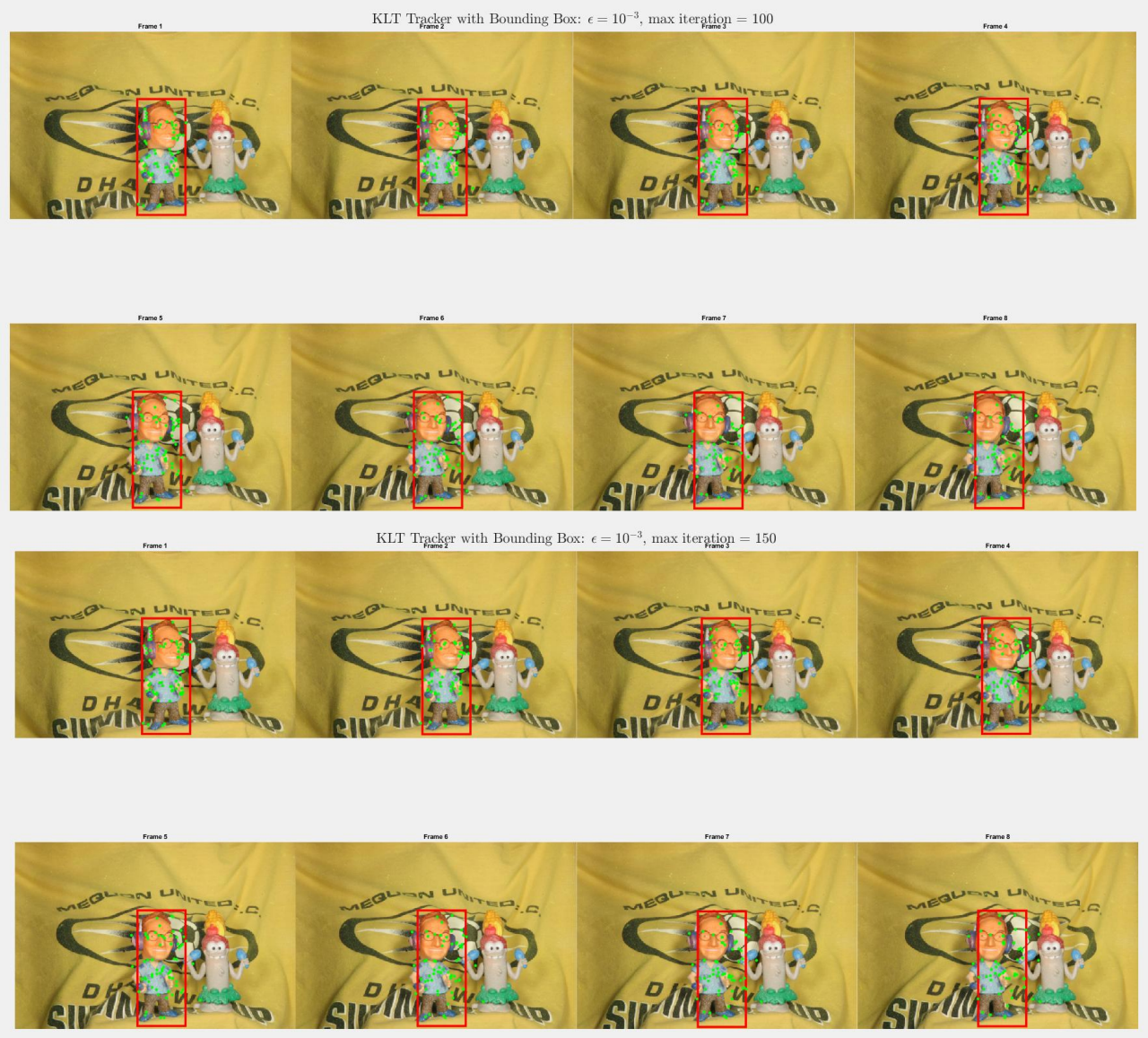




Most effective and efficient value was 10^{-3}.

Then, 3 different maximum iteration values are applied 75, 100, 150:



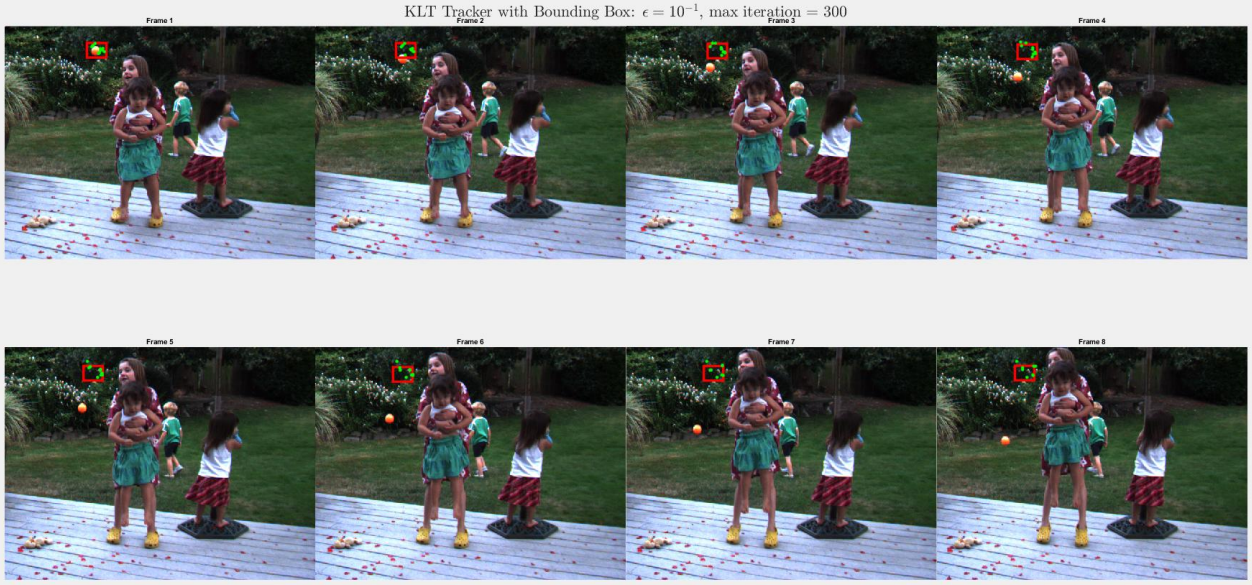


Most effective and efficient value was 75.

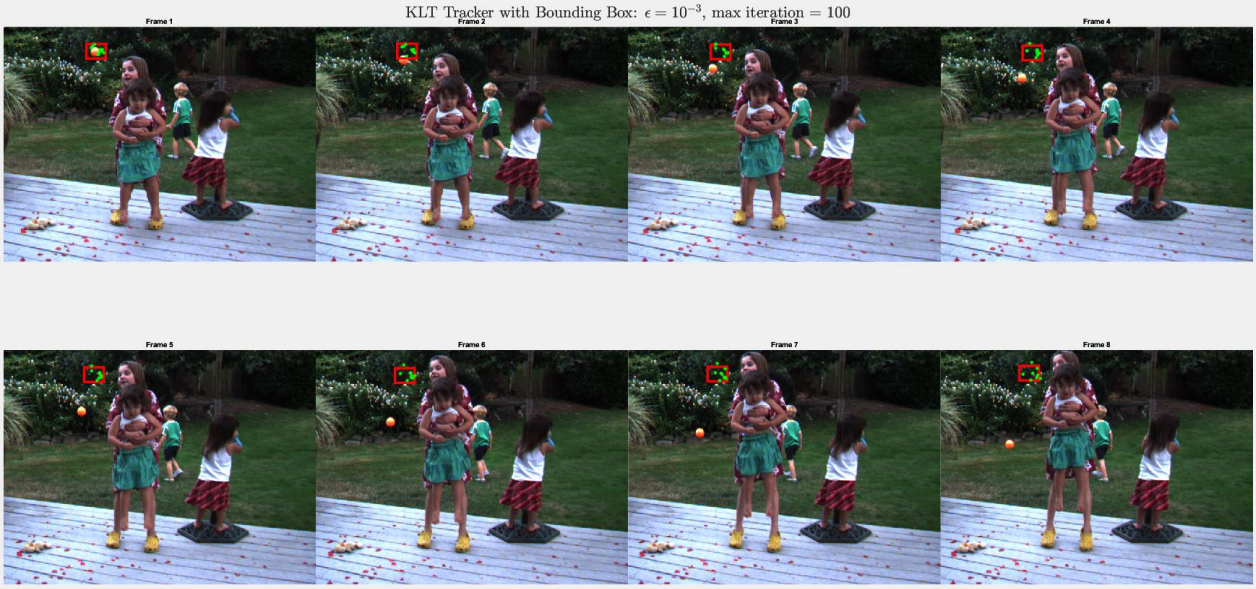
3.2.2.2 Video 2

First, 3 different ϵ values are applied which are 10^{-1}, 10^{-3}, 10^{-6}:

KLT Tracker with Bounding Box: $\epsilon = 10^{-1}$, max iteration = 300

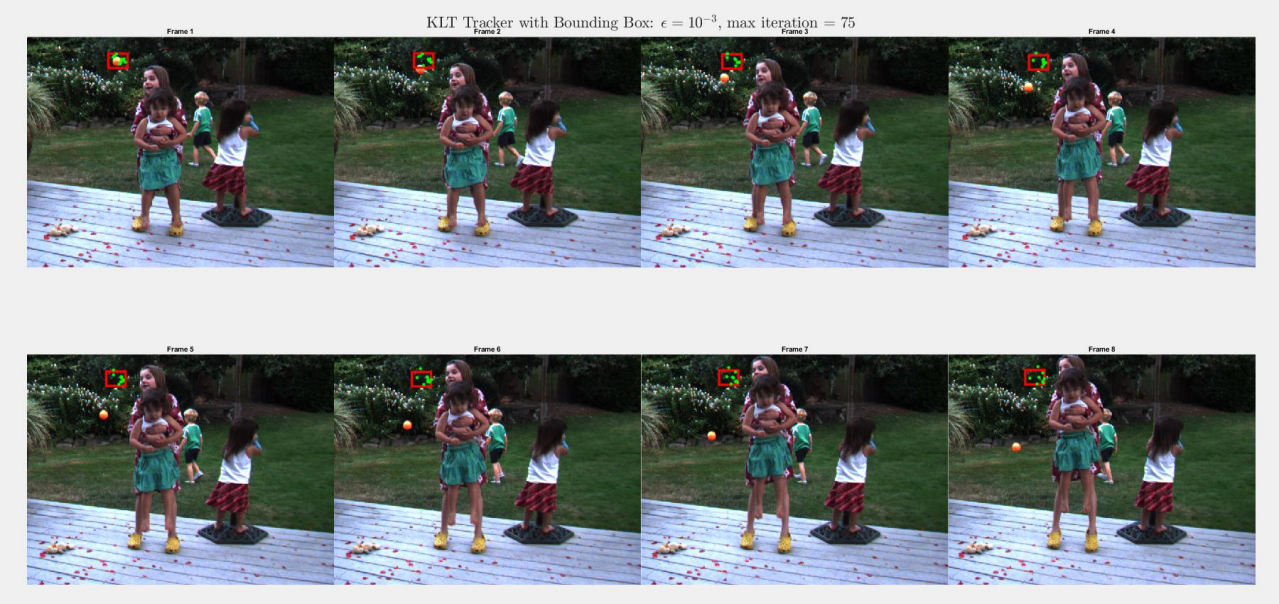


KLT Tracker with Bounding Box: $\epsilon = 10^{-3}$, max iteration = 100



Most effective and efficient value was 10^{-3}.

Then, 3 different maximum iteration values are applied 75, 100, 300:

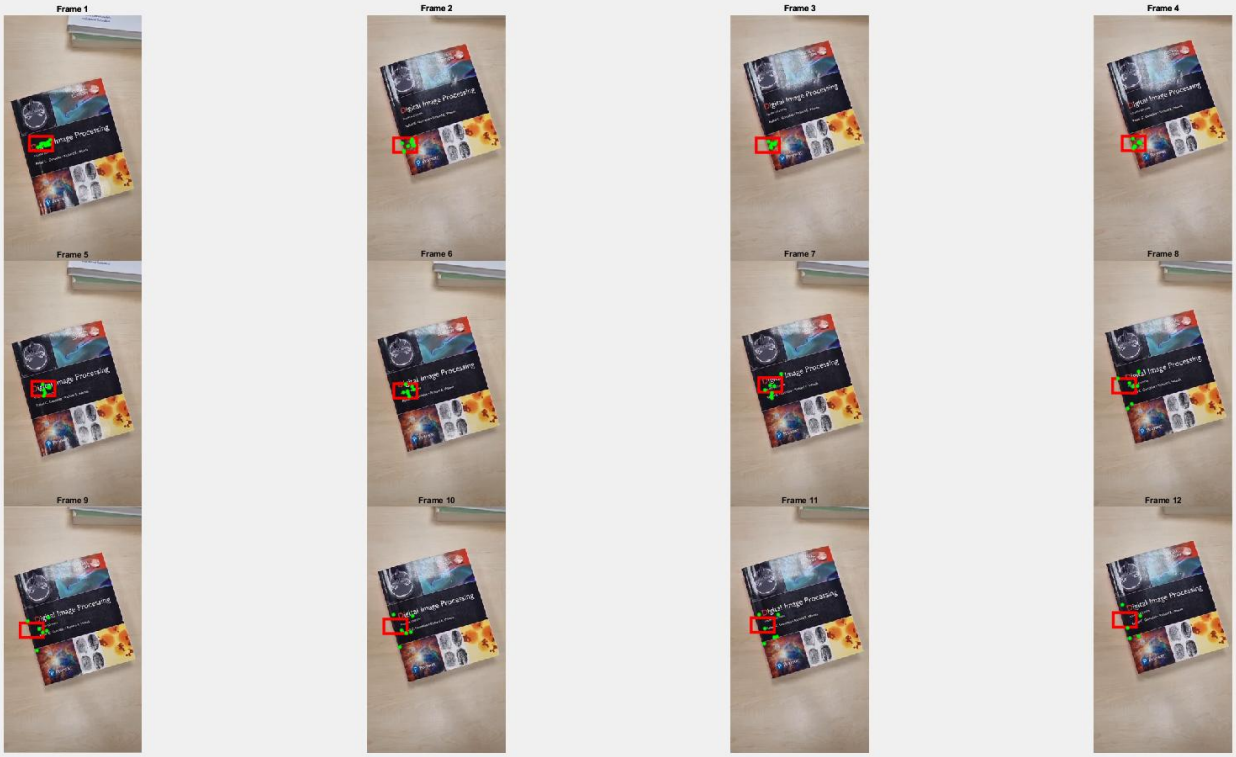


Most effective and efficient value was 300.

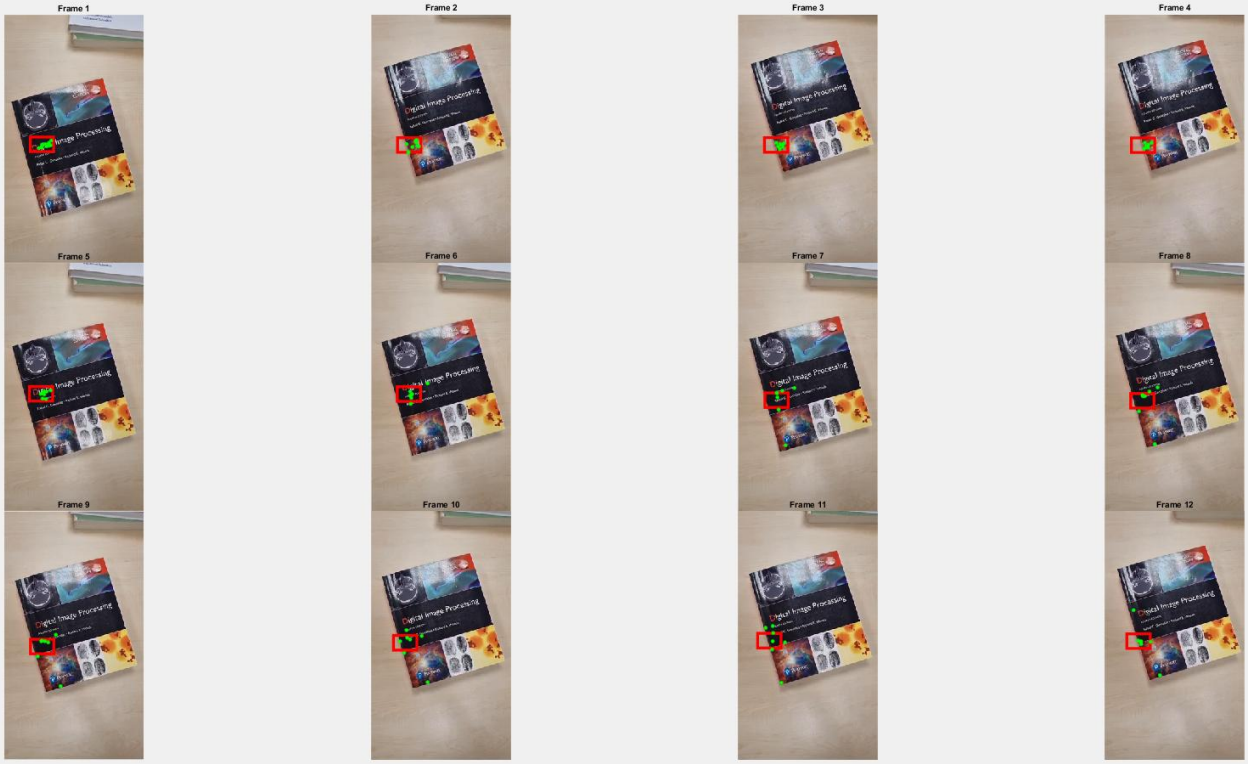
3.2.2.3 Video 3

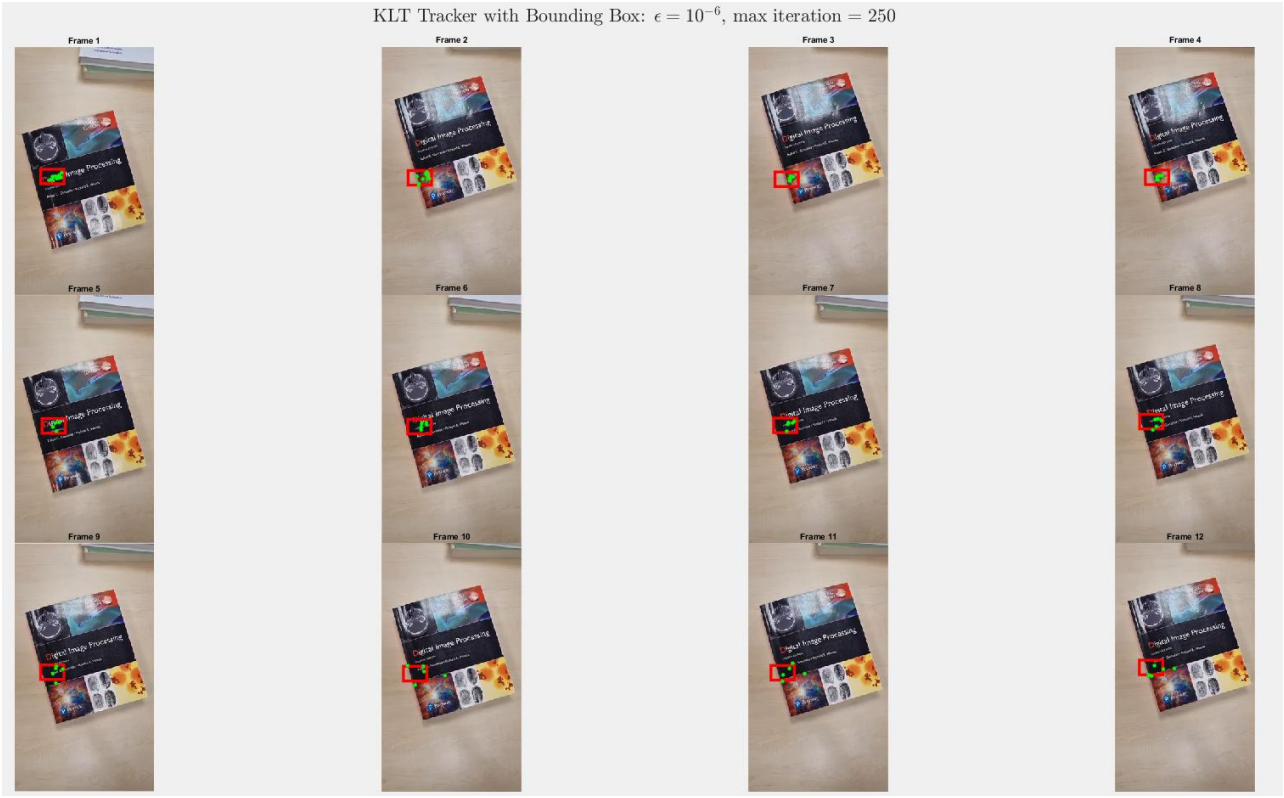
First, 3 different ϵ values are applied which are 10^{-1}, 10^{-3}, 10^{-6}:

KLT Tracker with Bounding Box: $\epsilon = 10^{-1}$, max iteration = 250



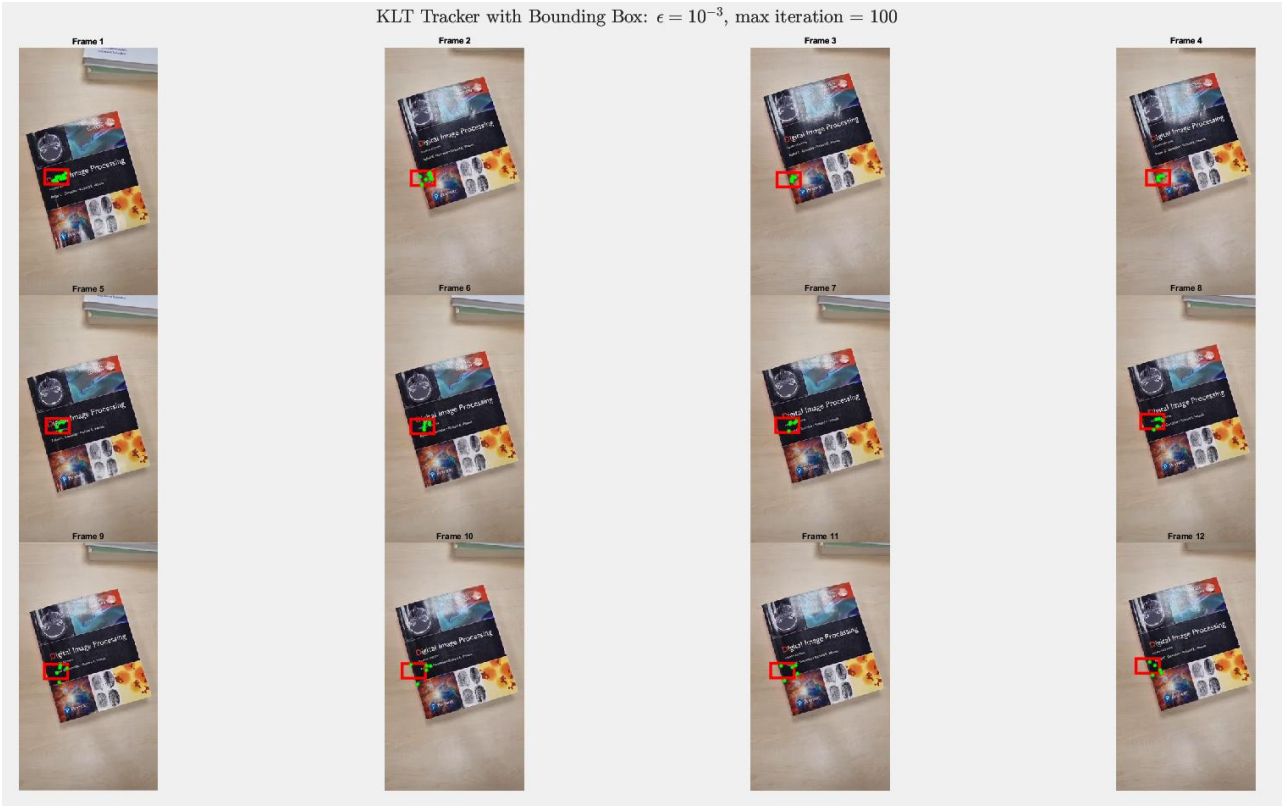
KLT Tracker with Bounding Box: $\epsilon = 10^{-3}$, max iteration = 250

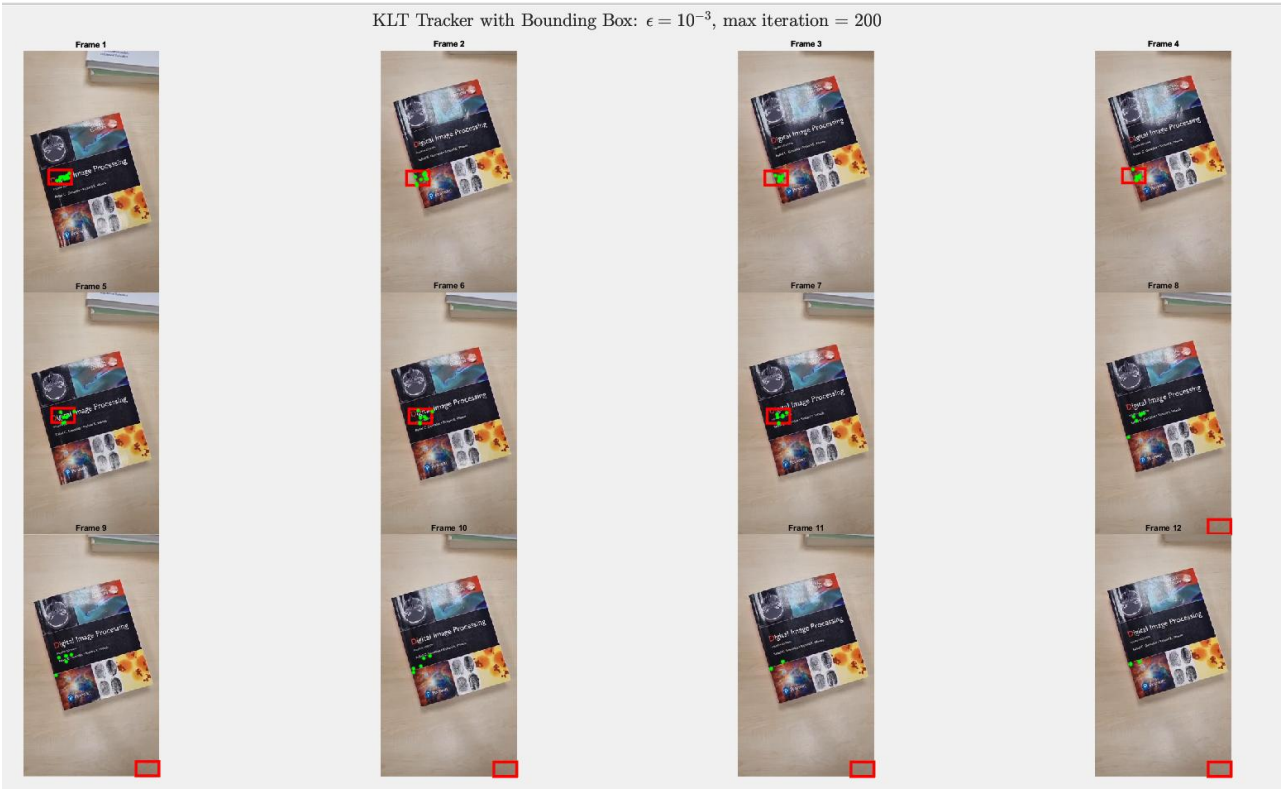




Most effective and efficient value was 10^{-3}.

Then, 3 different maximum iteration values are applied 100, 200, 250:





Most effective and efficient value was 250.

4. Discussions

Regarding the results of the Lucas-Kanade Forward Additive Alignment, the object tracking objective is achieved successfully for all the videos provided with the most effective and efficient values provided in the results section. However, for the 3 frames of the last video no matter how the values are changed I could not achieve perfection. The reasons for that issue might be variations in the object's texture, motion blur, or occlusions and this can be assessed by conducting experiments with more robust preprocessing or modifying the gradient computation methods. The affect of different ϵ values was evaluated in terms of convergence precision, where smaller values improved accuracy but increased computation time. The affect of different **max_iter** value was tested to balance computational efficiency and convergence, showing that higher values allowed for better alignment but with diminishing returns after a certain point.

Based on the results of the Kanade-Lucas-Tomasi (KLT) Tracker, the object tracking performance varies across different datasets and parameter configurations. For the first dataset, using $\epsilon = 10^{-3}$ and a maximum iteration value of 75, the tracker successfully maintained alignment for the majority of the frames. For the second dataset, where $\epsilon = 10^{-4}$ and the maximum iteration value was increased to 300, the tracking results were not successful. The tracker struggled with significant drift and inaccuracies, which can likely be attributed to the nature of the scene. The dataset contained inconsistent lighting conditions, reflections, and shadows, which made feature detection and tracking more challenging. These environmental factors likely degraded the quality of the extracted features, leading to poor convergence and inaccurate tracking results. To address these issues, preprocessing techniques such as contrast enhancement, histogram equalization, or illumination normalization could be applied to improve the robustness of the tracker in such conditions. Additionally, the performance can be reassessed by experimenting with more adaptive tolerance values and incorporating motion compensation techniques or multi-scale feature tracking to better handle rapid and irregular movements.

In the third dataset, with $\epsilon=10^{-3}$ and a maximum iteration value of 250, the tracker performed robustly even with significant object rotations and changes in perspective. The balance between precision and computational efficiency was well-maintained in this configuration, though slight drifting was occasionally observed in later frames.

These results highlight the importance of carefully tuning parameters like ϵ and the maximum iteration count, as well as the need for specialized techniques to address unique challenges such as occlusions, rapid motion, and lighting variations in complex datasets.

5. Conclusion

In conclusion, in this assignment, the implementation of the Lucas-Kanade Forward Additive Alignment and the Kanade-Lucas-Tomasi (KLT) Tracker demonstrated the impact of parameter tuning on tracking performance. Smaller ϵ values improved tracking precision but increased computation time, while higher iteration limits allowed better convergence but provided diminishing returns beyond a certain threshold. The Lucas-Kanade method performed reliably for object alignment across most frames, but challenges such as motion blur and occlusions affected its robustness in certain cases. For the KLT Tracker, lighting inconsistencies, shadows, and reflections in the second dataset highlighted the importance of preprocessing and robust feature extraction to enhance performance. Key takeaways include the need for balanced parameter optimization, adapting to environmental factors, and leveraging preprocessing techniques to handle challenging conditions, ensuring both accuracy and efficiency in real-world scenarios.

6. References

[1] M. O'Toole and K. Kitani, "Alignment and Tracking," Lecture Slides for CS 16-385, Carnegie Mellon University (CMU), Spring 2017 and Spring 2021. Adapted for CS 484/555, Fall 2024.

[2] C. Schmid and D. Lowe, "Local Feature Detectors," CVPR 2003 Tutorial. Slides adapted by M. Brown, Microsoft Research, and S. Seitz, University of Washington.