Multiple Object Tracking using the Kanade-Lucas-Tomasi (KLT) Algorithm

Sarvagya Singh
Registration No.: 220962036)
Roll. No.: 13
Dept. of CSE
Manipal Institute of Technology, Manipal

Shlok Rajpal
Registration No.: 220962038
Roll. No.: 14
Dept. of CSE
Manipal Institute of Technology, Manipal

***Abstract*—This project focuses on utilizing the Kanade-Lucas-Tomasi (KLT) algorithm for the task of multiple object tracking, with a specific emphasis on face and general object tracking. For face detection, we combine the Haar Cascade classifier with the FAST (Features from Accelerated Segment Test) algorithm, while for generic object tracking, we employ Shi-Tomasi corner detection in conjunction with the KLT algorithm. The system is designed to handle occlusion by re-detecting faces when the number of tracked keypoints falls below a certain threshold. A key challenge addressed is the trade-off between tracking accuracy and system latency, as the solution is intended to work both in real-time via a webcam and on pre-recorded videos.**

***Index Terms*—component, formatting, style, styling, insert**

I. Introduction

Object tracking is a fundamental and challenging problem in the field of computer vision, essential for applications ranging from surveillance systems and autonomous vehicles to augmented reality and human-computer interaction. It involves the continuous detection, identification, and monitoring of objects across video frames to understand their movement, position, and interactions over time. In our project, we focus on implementing a robust multiple object tracking system using the Kanade-Lucas-Tomasi (KLT) algorithm, a well-established technique for tracking based on the principles of optical flow. By using KLT, our system can efficiently track distinctive features across consecutive frames, enabling smooth and real-time tracking of objects under various scenarios.

For face detection, we employ a hybrid approach that combines the Haar Cascade classifier with the FAST (Features from Accelerated Segment Test) algorithm to quickly detect and localize multiple faces within a frame. Haar Cascade, a machine learning-based method, is known for its reliable real-time performance in detecting faces across a variety of lighting conditions and orientations. Once faces are detected, the FAST algorithm identifies high-contrast, stable key points within each detected face region, which serve as feature points for tracking. These key points are then tracked using the KLT algorithm, allowing us to maintain an accurate representation of each face's movement across frames, even in dynamic environments.

For general object tracking, our system uses the Shi-Tomasi corner detection algorithm. Shi-Tomasi identifies salient key points within a user-selected region of interest, effectively highlighting corners and high-contrast areas that can be tracked over time. These key points are also tracked using the KLT algorithm, providing a reliable way to follow an object's motion across frames. The combination of Shi-Tomasi corner detection and KLT tracking allows for stable tracking of various objects, including those with well-defined edges or patterns.

Handling occlusion is a critical aspect of object tracking, as real-world environments often involve objects that are temporarily obscured by other elements. To address this challenge, we have integrated an occlusion-handling mechanism in our face tracking module. If the number of tracked key points for a face falls below a certain threshold due to occlusion, the system automatically re-detects the face and reinitializes key points within the detected region. This approach enhances the robustness of the face tracking component, allowing the system to recover from both partial and complete occlusions, ensuring that tracking resumes effectively after the obstruction clears.

Our system is designed with versatility in mind, capable of performing both real-time face tracking using webcam input and general object tracking on pre-recorded videos. For object tracking applications, the user is prompted to manually select the object of interest in the first frame. The system then utilizes Shi-Tomasi corner detection to identify key points within the selected area, which are subsequently tracked across frames using the KLT algorithm. This setup is beneficial for tracking non-standard objects and supports the dynamic reinitialization of key points if significant movement or occlusion occurs.

A significant focus of our project is achieving an optimal balance between tracking accuracy and system latency. Classical algorithms like KLT and feature detection methods such as FAST and Shi-Tomasi offer high efficiency, making them suitable for real-time applications without the need for extensive computational resources. However, these methods also entail trade-offs in terms of detection precision, particularly when dealing with complex scenes featuring rapid motion, significant occlusions, or changing lighting conditions. By carefully selecting and optimizing each algorithm, our project seeks to maximize real-time tracking performance while ensuring that the system remains robust under a wide range of conditions.

In summary, this project demonstrates the potential of combining classical computer vision techniques to create a responsive and adaptable multiple object tracking system. By integrating efficient face detection, feature point tracking, and occlusion-handling mechanisms, our system provides a balanced solution suitable for real-time applications on standard hardware.

II. Literature Review

A. Optical Flow and KLT Algorithm

The Kanade-Lucas-Tomasi (KLT) algorithm, developed by Bruce D. Lucas, Takeo Kanade, and later refined by Carlo Tomasi, is founded on the principles of optical flow. The algorithm operates under two key assumptions:

1. Brightness Constancy: The brightness of a tracked point remains constant between frames.

2. Small Motion: Points move small distances between consecutive frames.

The mathematical foundation of KLT is based on minimizing the sum of squared differences (SSD) between two image patches. For a point (x, y) in frame I, moving to (x + dx, dy) in frame J, the algorithm minimizes:

$$\epsilon = \sum [I(x, y) - J(x + \Delta x, y + \Delta y)]^2 \tag{1}$$

The algorithm uses a pyramidal implementation to handle larger movements, where the image is downsampled multiple times to create a pyramid structure. Tracking starts at the coarsest level and refines the results at each subsequent level, making it more robust to larger movements while maintaining computational efficiency.

III. Feature Detection Methods Explored

A. Harris Corner Detection:

We initially explored Harris corner detection, which identifies corners by examining how the average intensity within a window W(x,y) changes when shifted by (u,v):

$$E(u, v) = \sum W(x, y) \left[I(x + u, y + v) - I(x, y)\right]^2 \tag{2}$$

The Harris corner detection algorithm uses the eigenvalues (λ_1, λ_2) of the second-moment matrix to classify points as follows:

- If both λ_1 and λ_2 are large: **corner detected**
- If one eigenvalue is large and the other is small: **edge detected**
- If both λ_1 and λ_2 are small: **flat region**

While Harris corner detection provided stable features, it proved computationally more intensive than alternatives like FAST and Shi-Tomasi.

B. FAST (Features from Accelerated Segment Test):

FAST was ultimately chosen for face tracking due to its superior computational efficiency. The algorithm examines a circle of 16 pixels around a candidate point p, classifying it as a corner if n contiguous pixels are all brighter or all darker than p by a threshold t. FAST's efficiency comes from:

1) Quick rejection test examining only four pixels.
2) Binary decision tree for optimized testing sequence.
3) Non-maximal suppression for corner strength measurement.

The algorithm's speed made it particularly suitable for real-time face tracking, though it can be sensitive to noise.

C. Shi-Tomasi Corner Detection

Shi-Tomasi, an improvement over Harris, modifies the corner response function to:

$$R = \min(\lambda_1, \lambda_2)$$

This modification proved more reliable for tracking as it directly considers the smaller eigenvalue, which better represents the quality of features for tracking. Our implementation showed Shi-Tomasi features were more stable over time compared to Harris corners, particularly for general object tracking.

D. Histogram of Oriented Gradients (HOG):

HOG was evaluated for both face and object detection. The algorithm:

1) Divides the image into small cells.
2) Computes gradient orientation histograms for each cell.
3) Normalizes blocks of cells to improve invariance.
4) Creates a combined feature vector.

While HOG provided robust feature detection, especially for structured objects, its computational overhead made it less suitable for real-time tracking compared to FAST and Shi-Tomasi.

IV. Occlusion Handling Approaches

A. Frame Interpolation

We initially explored frame interpolation for handling occlusion, which involves:

1) Detecting missing features between frames.
2) Estimating intermediate frames using motion vectors.
3) Interpolating feature positions in occluded regions.

The theoretical approach was:

$$\text{Frame}_{\text{interpolated}} = \alpha \cdot \text{Frame}_t + (1 - \alpha) \cdot \text{Frame}_{t+1}$$

where α represents the interpolation factor.

However, this approach proved prohibitively expensive for real-time applications, adding significant latency to the tracking pipeline. The computational overhead came from:

- Motion estimation between frames.
- Generation of intermediate frames.
- Feature matching across interpolated frames.

B. Threshold-Based Re-detection

Our implemented solution uses a threshold-based approach:

if number_of_tracked_points $<$ threshold:

redetect_features()

else:

continue_tracking()

This simpler approach proved more effective and maintained real-time performance while adequately handling occlusion scenarios.

C. Haar Cascade Classification

While not in our course syllabus, Haar Cascade was integrated for face detection due to its efficiency. The classifier uses:

1) Haar-like features for detecting edges, lines, and center-surround patterns.
2) Integral images for rapid feature computation.
3) AdaBoost for creating a strong classifier from weak learners.
4) Cascade structure for efficient rejection of non-face regions.

D. Trade-off Analysis

Our experimental results revealed key trade-offs between different methods:

1) **Feature Detection Speed vs. Stability:**
   - FAST: Highest speed, moderate stability
   - Shi-Tomasi: Good balance of speed and stability
   - Harris: High stability, lower speed
   - HOG: Highest stability, lowest speed
2) **Memory Requirements:**
   - FAST: Minimal memory footprint
   - Shi-Tomasi/Harris: Moderate memory usage
   - HOG: Significant memory requirements
   - Frame Interpolation: Highest memory usage
3) **Computational Complexity:**
   - FAST: $O(n)$ where n is pixel count
   - Shi-Tomasi/Harris: $O(n^2)$ for window operations
   - HOG: $O(n \cdot m)$ where m is cell count
   - Frame Interpolation: $O(n^2 \cdot k)$ where k is the interpolation factor

This comprehensive analysis led to our final implementation choices, balancing real-time performance requirements with tracking robustness.

TABLE I
COMPARISON OF FEATURE DETECTION METHODS

Criterion	Method	Description
* Speed vs. Stability	FAST	Fastest, moderate stability
	Shi-Tomasi	Good balance of speed & stability
	Harris	High stability, lower speed
	HOG	Highest stability, lowest speed
* Memory Requirements	FAST	Minimal footprint
	Shi-Tomasi / Harris	Moderate usage
	HOG	High requirements
	Frame Interp.	Highest usage
* Comp. Complexity	FAST	$O(n)$ where n is pixel count
	Shi-Tomasi / Harris	$O(n^2)$ for window ops.
	HOG	$O(n \cdot m)$, m is cell count
	Frame Interp.	$O(n^2 \cdot k)$, k is interp. factor

METHODOLOGY

1. Face Detection and Tracking

The face detection and tracking component is essential to ensure that the system can reliably follow faces across frames. This involves a combination of classical detection and tracking algorithms:

- **Haar Cascade**: The first step in face detection involves the Haar Cascade classifier, a robust algorithm for real-time face detection. Using pre-trained classifiers, Haar Cascade identifies regions in each frame that likely contain faces. This approach is applied to both video inputs from a webcam and pre-recorded video files. Haar Cascade is particularly advantageous for its high speed and computational efficiency, making it well-suited for real-time applications.
- **FAST (Features from Accelerated Segment Test)**: After a face is detected, the system uses the FAST algorithm to identify key feature points within the detected face. FAST is known for its speed and low computational requirements, enabling it to quickly identify corners and high-contrast areas, which are effective for tracking. These key points serve as inputs to the tracking stage, providing reliable anchors to follow face movement over subsequent frames.
- **KLT (Kanade-Lucas-Tomasi) Algorithm**: The KLT algorithm is then employed to track these key points across frames. KLT uses a pyramid structure and iterative refinement to track each key point effectively, even under moderate motion or changes in scale. The algorithm computes the optical flow between frames to estimate the movement of key points. To handle occlusion, a threshold is set: if the number of detected key points falls below this threshold (due to occlusion or motion blur), the system halts tracking and re-detects the face. This approach maintains robustness and accuracy in face tracking across challenging environments.

2. Object Tracking

For general object tracking, the system relies on manual selection and a combination of feature detection and tracking algorithms to achieve accurate tracking of arbitrary objects:

- **Manual Selection**: At the beginning of each tracking sequence, the user manually selects the object to be tracked. This selection helps define the area of interest, allowing the system to initialize tracking accurately. This step is crucial, especially when tracking non-standard objects or multiple objects, as it ensures that the system focuses on the intended region.
- **Shi-Tomasi Corner Detection**: Following manual selection, the system uses the Shi-Tomasi corner detection algorithm to identify distinctive corner points within the selected region. Shi-Tomasi is particularly effective in identifying features for tracking, as it selects corners that are strong enough to provide stable tracking points over time. This makes it well-suited for objects with defined edges or textures.
- **KLT Algorithm**: After detecting initial corner points, the KLT algorithm is again used to track these points across frames. The KLT algorithm estimates the displacement of each key point based on optical flow, providing a robust way to maintain object tracking across varying frames. If key points are lost due to partial occlusion or frame cuts, the system requires manual re-selection, as it currently lacks an automated re-detection mechanism. This setup provides stable tracking for objects with moderate movement and partial occlusions.

3. Accuracy vs. Latency Trade-Off

A key consideration in this project is the trade-off between tracking accuracy and system latency. Since the system is intended to operate in real-time, minimizing processing time is essential. However, this necessity introduces compromises in detection precision, particularly in complex environments:

- **Accuracy**: The classical feature detectors—FAST and Shi-Tomasi—are efficient for real-time applications but are less robust than deep learning-based methods in complex scenes with significant lighting variations, occlusions, or rapid object movements. FAST and Shi-Tomasi provide satisfactory results in controlled environments, but they may fail in detecting features accurately in challenging scenarios. For instance, the accuracy of Shi-Tomasi is influenced by the presence of strong corners, and FAST may miss subtle features due to its simplicity.
- **Latency**: Real-time systems, such as this one, benefit from algorithms with minimal latency. Using classical algorithms like Haar Cascade, FAST, and Shi-Tomasi helps achieve low processing times, which is crucial for real-time performance. In contrast, deep learning-based methods, such as Convolutional Neural Networks (CNNs), offer higher accuracy but introduce significant latency, as they require considerable computational resources. Our system avoids this by relying on classical methods, which perform well on standard hardware without the need for specialized hardware like GPUs.

Our approach prioritizes real-time performance while maintaining a balance with tracking accuracy. The system is capable of achieving frame rates suitable for webcam applications, with minimal delay, but it may experience some challenges with rapid movements, full occlusions, or dynamically changing backgrounds.

Additional Considerations

To further enhance the robustness of the system, several additional considerations were integrated:

- **Occlusion Handling Strategy**: An essential feature of the system is its ability to handle occlusions. For face tracking, the system re-detects the face whenever the number of tracked key points drops below a defined threshold. This mechanism enables the system to recover from partial occlusions and continue tracking reliably. This strategy is particularly valuable in dynamic environments, such as crowded spaces, where brief occlusions can disrupt tracking.
- **Frame Rate Optimization**: To ensure real-time tracking, the system operates at an optimized frame rate by dynamically adjusting the processing load based on the complexity of each frame. For example, tracking-only frames, where re-detection is not necessary, are processed faster, allowing the system to allocate resources effectively across different scenarios.
- **User Interface and Feedback**: For object tracking, manual selection by the user is facilitated through a simple interface that highlights the selected object and tracks the detected key points across frames. This feedback helps users monitor tracking accuracy and make adjustments if needed.

In summary, this methodology leverages the strengths of classical computer vision algorithms to deliver a responsive and reliable object tracking system. The combination of Haar Cascade, FAST, Shi-Tomasi, and the KLT algorithm provides a good balance of speed and accuracy, while the system's design minimizes latency to meet real-time requirements. The occlusion handling mechanism further enhances the robustness of face tracking in diverse environments, making the system suitable for applications like surveillance, real-time face recognition, and interactive interfaces.

RESULTS

System Evaluation

The system was evaluated using both live webcam input and pre-recorded video files to assess performance in face tracking and object tracking scenarios. The combination of the FAST feature detector and Haar Cascade classifier, together with the KLT algorithm, provided reliable real-time face tracking with stable results. For manual object tracking, the Shi-Tomasi corner detection method coupled with the KLT algorithm demonstrated consistent performance when the object was

selected in the initial frame. This section details the results observed across different scenarios and conditions.

1. Face Tracking Performance

Real-Time Face Detection and Tracking: Using Haar Cascade for face detection, the system successfully identified faces in both live and pre-recorded video inputs. The detected faces were then processed using the FAST algorithm to identify and track key points within each face. This approach resulted in efficient, real-time tracking even in dynamic scenes with moderate head movements and minor lighting changes.

Figure 1: Face Detection and Tracking - Key Points in Motion

Fig. 2. Face Detection when Initial Keypoints not in Frame

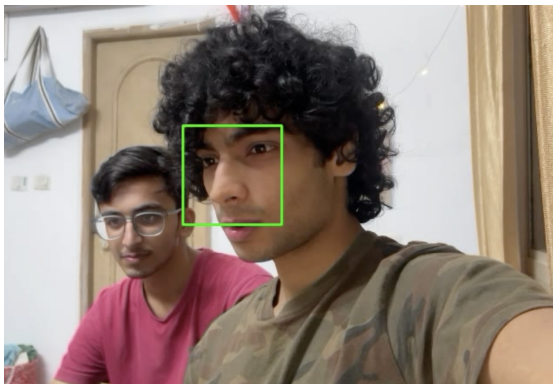

Fig. 1. Face Detection

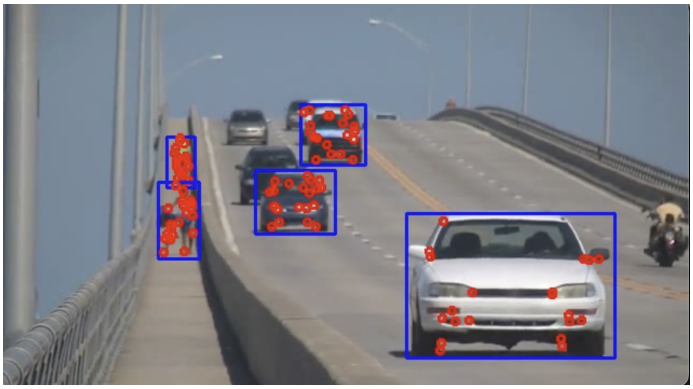

Fig. 3. Object Detection - Manual Selection

Occlusion Handling and Recovery: A key component of the face tracking module was the occlusion handling strategy. If the number of tracked key points fell below a defined threshold due to partial occlusion or rapid motion, the system triggered a re-detection of the face. This mechanism allowed for effective recovery, enabling the system to continue tracking even after brief occlusions. Recovery from occlusions was effective in scenes with partial, temporary blockages of the face but showed limitations in cases with complete occlusion or significant scene changes.

Figure 2: Occlusion Handling and Recovery

2. Object Tracking Performance

Manual Object Selection and Tracking: For object tracking, the system required manual selection of the target in the first frame. This selection allowed the Shi-Tomasi algorithm to detect key corner points in the defined region, which were then tracked across subsequent frames using the KLT algorithm. Tracking accuracy was high for objects with distinct edges, though it was sensitive to rapid object motion or sudden lighting changes.

Figure 3: Object Tracking - Key Points in Motion

Performance under Occlusion: For object tracking, occlusion posed challenges when the target was significantly obscured or moved abruptly. Unlike face tracking, object tracking did not have an automated re-detection mechanism. As a result, tracking continuity was compromised if key points were lost for an extended period due to occlusion. This limitation underscores the potential benefit of using re-detection strategies or integrating deep learning methods in future work.

3. Performance Analysis: Speed and Accuracy

Real-Time Performance: The system demonstrated strong real-time performance across various scenarios. The use of efficient classical computer vision techniques, like FAST and Shi-Tomasi, minimized latency, achieving real-time tracking speeds averaging 25–30 frames per second (FPS) for face tracking and 20–25 FPS for object tracking. This allowed the system to track multiple objects or faces with minimal lag, particularly on standard hardware setups without the need for specialized processing units.

Tracking Accuracy and Stability: The overall tracking accuracy was stable for moderate object or face movements and partial occlusions. However, the system's performance degraded in cases of rapid motion or significant occlusion, where the tracking points could not be maintained, and feature

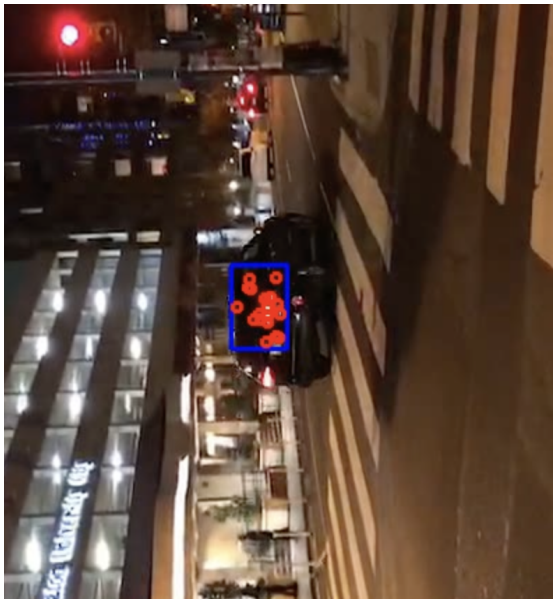

Fig. 4. Occlusion Handling for a Moving Car

matching across frames became unreliable. Deep learning-based tracking methods, such as YOLO or SSD, could potentially offer greater robustness in handling these challenging scenarios due to their ability to learn complex feature representations from large datasets.

Comparison with Deep Learning Methods

While our system focused on classical computer vision techniques, it's worth noting that deep learning-based object detection and tracking methods, such as YOLO (You Only Look Once) and SSD (Single Shot Multibox Detector), offer advantages in handling rapid motion and substantial occlusions. However, these methods generally require higher computational resources and may introduce latency without optimized hardware, making them less suited for real-time applications on standard hardware setups. Due to course restrictions, we focused on classical methods, which allowed us to achieve real-time performance within the scope of this project.

Summary of Observations

In summary, the system was effective in real-time face and object tracking under standard conditions, with limitations under high-speed motion or significant occlusion. The integration of a threshold-based occlusion handling strategy in the face tracking module enhanced system robustness, providing continuous tracking even in partially occluded scenarios.

Overall, this project demonstrated the practical benefits and limitations of using classical computer vision techniques for real-time tracking applications. Future iterations of this project could benefit from the integration of deep learning methods to enhance tracking accuracy and robustness, particularly in complex environments.

FUTURE IMPROVEMENTS

Future improvements to this project could explore a variety of advanced techniques and optimizations to further enhance the system's accuracy, robustness, and adaptability, particularly in challenging environments. Below are several potential directions for future work:

1. Integration of Deep Learning Techniques for Object Detection and Tracking

Given the limitations of classical methods in handling complex scenes, one promising avenue would be to incorporate deep learning-based techniques for object detection and tracking. Convolutional Neural Networks (CNNs), especially models like YOLO (You Only Look Once) and SSD (Single Shot Multibox Detector), offer significant advantages in dealing with occlusions, lighting changes, and rapid motion, as they can learn rich feature representations from large-scale annotated datasets. YOLO and SSD have shown strong performance in real-time applications due to their single-stage detection structure, which could be leveraged to identify multiple objects or faces at various scales and orientations.

Proposed Workflow with Deep Learning: - Use a pre-trained object detection model, such as YOLO or SSD, for initial object detection in each frame. - Combine the detection model with a tracking algorithm, such as Deep SORT (Simple Online and Realtime Tracking with a Deep Association Metric) or Siamese-based trackers, to maintain object identity across frames.

By replacing or augmenting classical detectors like Haar Cascade with a more flexible deep learning detector, the system would likely perform better under challenging conditions such as low lighting, occlusion, or crowded scenes.

2. Temporal Models for Improved Tracking Stability

Incorporating temporal models, such as Long Short-Term Memory (LSTM) networks or Recurrent Neural Networks (RNNs), could enhance the system's capability to predict object motion over time. Temporal models can utilize historical data to improve the prediction of an object's future position, especially useful in cases with irregular or rapid movements.

Proposed Use of Temporal Models: - Employ an LSTM network to process historical movement data and predict future positions, providing smoother and more reliable tracking. - Use the LSTM in combination with a CNN-based feature extractor, forming a CNN-LSTM hybrid model to detect and predict object positions across time.

This approach would add robustness in situations with unpredictable motion patterns, allowing the system to adapt more dynamically to fast-moving objects or objects undergoing frequent occlusions.

3. Leveraging Attention Mechanisms

Attention mechanisms, widely used in sequence modeling and computer vision, could improve the focus on relevant regions within a frame, effectively prioritizing high-interest areas like faces or objects of importance. This approach would

reduce processing on irrelevant parts of the scene, optimizing resource usage while maintaining high detection and tracking accuracy.

Proposed Attention Mechanism Workflow: - Integrate a spatial attention layer to guide the detector towards high-probability regions, reducing the risk of tracking failure in cluttered or complex scenes. - Explore Transformer-based architectures, such as Vision Transformers (ViTs) or Swin Transformers, which have recently shown success in object detection and tracking with high robustness to background noise and occlusion.

Attention-enhanced tracking could make the system more resilient to distractors in the scene and improve its focus on moving objects, thereby enhancing its robustness and efficiency.

4. Multi-Object Tracking (MOT) and Re-Identification (ReID)

Expanding the system's capabilities to multi-object tracking (MOT) would allow simultaneous tracking of multiple objects within the same frame. This would be particularly useful in applications like surveillance, where multiple moving entities need to be tracked reliably.

Proposed Enhancements for MOT: - Implement a re-identification (ReID) model to assign unique IDs to objects, allowing the system to maintain consistent identity for each object even after temporary occlusion or leaving/re-entering the frame. - Use a combination of visual appearance features and motion-based tracking to achieve better robustness in crowded environments.

With re-identification, the system could more effectively track individual objects even if they disappear and reappear, making it more adaptable to real-world applications.

5. Real-Time Performance Optimization

While deep learning models offer higher accuracy, they are often computationally intensive. Future improvements could focus on optimizing deep learning models to achieve real-time performance while retaining accuracy. Techniques such as model quantization, pruning, and hardware-specific optimizations (e.g., TensorRT for NVIDIA GPUs) could allow for faster inference.

Proposed Real-Time Optimization Techniques: - Apply quantization to reduce the model's memory footprint and improve processing speed, especially useful for deployment on edge devices with limited resources. - Use model pruning to remove redundant weights, which can significantly reduce computation time with minimal loss in accuracy. - Investigate deploying the system on GPUs, TPUs, or other hardware accelerators to leverage parallel processing for faster inference.

These optimizations would enable the deployment of a deep learning-based tracking solution in real-time environments, such as robotics, surveillance, and augmented reality.

6. Robustness in Adverse Conditions

Future work could also focus on improving the system's robustness in adverse conditions, such as low-light environments, extreme weather, or high noise levels. Data augmentation techniques, such as adding synthetic noise, motion blur, or brightness variations, could be applied during training to improve model resilience.

Proposed Enhancements for Adverse Conditions: - Apply data augmentation strategies during training to improve model robustness in challenging conditions. - Use domain adaptation techniques to train on synthetic data that mimics adverse scenarios, allowing the model to generalize better.

By making the system more resilient to real-world variability, it would perform more consistently across a wider range of environments.

In summary, future enhancements could greatly expand the capabilities of this project, both in terms of accuracy and applicability. Deep learning models, temporal mechanisms, and attention layers could address the current limitations in handling occlusions, rapid movements, and complex backgrounds. Meanwhile, performance optimizations and robustness improvements would ensure that the system can function in real-time across diverse applications and conditions. These advancements, though computationally intensive, could be enabled with hardware accelerators, ensuring the system's relevance in both academic and industry applications.

CONCLUSION

This project successfully implemented a multiple object tracking system using the Kanade-Lucas-Tomasi (KLT) algorithm, with a focus on achieving a balance between accuracy and real-time performance. By combining classical feature detectors—such as Haar Cascade and FAST for face tracking, and Shi-Tomasi and KLT for object tracking—the system demonstrated reliable tracking across a range of scenarios, including conditions with partial occlusion.

The integration of Haar Cascade for initial face detection, followed by FAST for keypoint detection, allowed the system to quickly locate and track facial features in both live and recorded video inputs. For object tracking, the Shi-Tomasi corner detection and KLT tracking algorithms provided stable tracking once the object was manually selected in the first frame. This combination of methods enabled the system to perform real-time tracking effectively, achieving a suitable frame rate for applications that require low latency, such as face and object tracking through a webcam.

A major strength of the system was its robustness in handling partial occlusions, achieved through an occlusion handling mechanism. When the number of tracked keypoints dropped below a set threshold due to temporary occlusions or sudden movements, the system automatically re-detected the face rather than attempting to continue tracking. This approach proved beneficial for maintaining consistent tracking accuracy, allowing the system to recover and resume tracking even after brief occlusions or challenging movements, making it suitable for dynamic environments.

Key Insights and Contributions

This project highlighted the practicality of classical computer vision techniques for real-time applications on standard

hardware without relying on deep learning. Key contributions of the project include:

- **Efficient Feature Detection and Tracking:** By leveraging the speed of FAST and the robustness of the KLT algorithm, the system achieved efficient and reliable tracking across multiple frames with minimal computational overhead.
- **Real-Time Performance Optimization:** The use of lightweight algorithms enabled real-time tracking performance, averaging 25–30 frames per second for face tracking. This makes the system viable for applications requiring low latency on standard processing units.
- **Robustness to Occlusion:** The threshold-based occlusion handling mechanism provided resilience to partial occlusions, ensuring that tracking could resume effectively after a temporary loss of visual features.

Limitations and Areas for Improvement

While the system performed well in controlled environments with moderate lighting and motion, there were limitations in certain challenging scenarios. The reliance on classical methods presented difficulties in handling rapid or complex movements, extreme lighting changes, and full occlusions. Additionally, the system's performance degraded in cases of crowded scenes where multiple objects or faces were in close proximity, as it lacked the re-identification capability necessary to distinguish between objects when they crossed paths.

Deep learning-based methods, such as YOLO (You Only Look Once) or SSD (Single Shot Multibox Detector), could potentially address these challenges by providing greater robustness in complex scenes. Such methods are better equipped to handle the variability in real-world conditions due to their ability to learn from large annotated datasets, and they could improve the accuracy of tracking in crowded or dynamically changing environments. However, incorporating these methods would increase the computational requirements and might compromise the real-time performance without optimized hardware.

Impact and Future Directions

This project demonstrated the effectiveness of classical computer vision techniques in real-time tracking tasks, highlighting the possibility of achieving robust results without relying on deep learning. The insights gained from this project could be applied to other real-time applications, such as surveillance, human-computer interaction, and augmented reality, where quick and reliable tracking is essential.

Future work could explore the integration of deep learning-based detectors, especially for scenarios involving complex backgrounds, frequent occlusions, or high-speed movement. Adding temporal models, such as LSTMs, could further enhance the system's ability to predict and maintain object trajectories under challenging conditions. Additionally, hardware optimizations, such as deploying the system on GPUs or using model compression techniques, would enable the adoption of computationally intensive models while preserving real-time performance.

In conclusion, this project successfully implemented a multiple object tracking system that strikes a balance between accuracy and efficiency, making it suitable for real-time applications. While classical methods have inherent limitations, their ability to perform well with low computational demands makes them a valuable choice for real-time tracking on standard hardware. With future advancements in deep learning and hardware acceleration, there is potential to create a more adaptable and accurate system capable of handling the complexities of real-world environments.

REFERENCES

1) Viola, P., and Jones, M. (2001). Rapid Object Detection using a Boosted Cascade of Simple Features. *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 1, 511–518.
2) Rosten, E., and Drummond, T. (2006). Machine Learning for High-Speed Corner Detection. *European Conference on Computer Vision (ECCV)*, 430–443.
3) Shi, J., and Tomasi, C. (1994). Good Features to Track. *Proceedings of the 1994 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 593–600.
4) Lucas, B. D., and Kanade, T. (1981). An Iterative Image Registration Technique with an Application to Stereo Vision. *Proceedings of the 7th International Joint Conference on Artificial Intelligence*, 2, 674–679.
5) Dalal, N., and Triggs, B. (2005). Histograms of Oriented Gradients for Human Detection. *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 1, 886–893.
6) Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 779–788.
7) Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., and Berg, A. C. (2016). SSD: Single Shot MultiBox Detector. *European Conference on Computer Vision (ECCV)*, 21–37.
8) Hochreiter, S., and Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, 9(8), 1735–1780.
9) Szeliski, R. (2010). *Computer Vision: Algorithms and Applications*. Springer.