Learning Based Lukas-Kanade Tracking through Motion Vector Predictions

Jiachen Huang
Mingze Sun

Department of Computing Science
University of Alberta

April 2024

Abstract

This project report introduces an enhanced object tracking method that uses estimated motion vectors to guide the Lukas-Kanade (LK) tracker. We employ a Convolutional Neural Network (CNN) to estimate motion vectors in real-time from selected regions of interest (ROI). Our method improves the tracking accuracy of the LK tracker to some extent, particularly in challenging conditions involving blurred objects or objects high-speed movements.

1 Introduction

The accurate tracking of objects in video sequences is a fundamental task in the field of computer vision, with applications spanning surveillance, autonomous driving, augmented reality, and more. Among the various algorithms developed for this purpose, the LK tracker stands out due to its efficiency and simplicity. However, the traditional LK tracker operates under the assumption that object motion between consecutive frames is small, which limits its effectiveness in scenarios involving fast-moving objects.

Recent advancements in deep learning, particularly in the use of CNN, have opened new avenues for enhancing optical flow methods. These models offer good performance in capturing complex motion patterns a shortcoming of gradient-based approaches like the LK method.

This report proposes an enhancement to the LK tracking method by incorporating a CNN-based predictive model that estimates the motion vectors of tracked objects. Our approach addresses the limitations of the traditional LK tracker in handling fast motion and also introduces an error-handling mechanism. This mechanism ensures tracking performance by verifying the predicted motion vectors against a template before updating the tracker's position, thereby minimizing the risk of tracking failures due to erroneous predictions.

- We introduce a CNN-based model to predict motion vectors for the Lukas-Kanade tracker, enhancing its ability to handle large and rapid movements of objects across frames.
- An error handling mechanism is developed to assess the accuracy of predicted motion vectors by comparing the sum of squared differences (SSD) between the predicted and actual object positions.
- We present a comparative analysis of our enhanced tracker against the traditional LK tracker, demonstrating improvements in tracking accuracy and robustness in some environments.

The remainder of this report is organized as follows: Section 2 details the methodology of our enhanced tracking framework. Experimental results are presented in Section 3, followed by a discussion Finally, Section 4 concludes the paper and outlines directions for future research.

2 Methodology

We propose an overall pipeline starting from preprocessing the CIFAR-100 dataset by applying motion blur kernels to 1000 random selected images. The CNN classifier is trained on original images and blurred images. The well trained CNN classifier can be used in real time to estimate the motion vector (u, v) of the ROI in a frame. The tracker position is updated according to the motion vector in the next frame and then apply the Lukas-Kanade algorithm to the updated tracker.

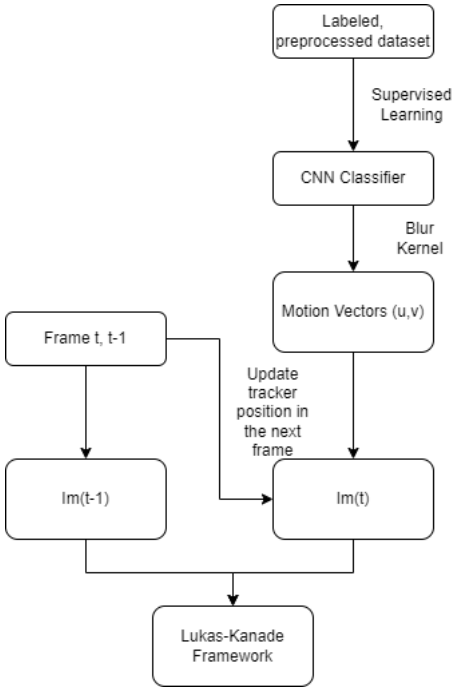

Figure 1: Pipeline

2.1 Dataset Preprocessing

It is necessary to make a blurred dataset of different objects in order to learn the CNN and enhance the robustness of the selected candidate model during the tracking. Thus, our desired objects are intentionally blurred and the training dataset is convolved with a predefined blur kernel. [3] We generate 217 different motion kernels based on the motion direction θ and magnitude l. The range of motion kernel magnitude is discretized into 13 samples $l = 1$ and from $l = 2$ to 25 with the interval of 2, and the range of motion angle was discretized into 19 samples from $\theta = 0°$ to 180°, and since 0° and 180° are equivalent, $n_\theta = 18$. Therefore, we have 217 number of motion kernels. If the regular non-blurred

image and Gaussian blur kernel are l_d and k_v respectively, then the blurred image l_b can be constructed by their convolution [2]

$$l_b = l_d \otimes k_v \tag{1}$$

The process of blurring the image is illustrated in Figure 2.

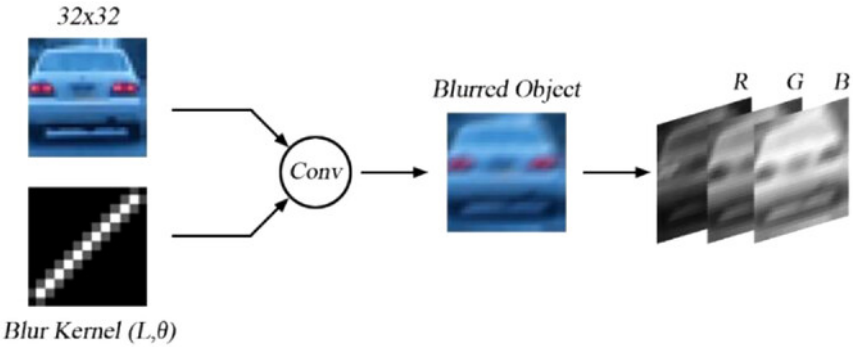

Figure 2: Blurring process

2.2 Training

In the proposed blur parameter estimation framework, useful features for angle and length prediction have been learned by CNNs. For the best performance of the blur object tracking, a CNN was designed that could predict the probabilities of motion kernels. [3] The general architecture of the CNN is depicted in Figure 3.

The CIFAR-100 [4] dataset was used, consisting of 60,000 32 × 32 colour images. This dataset is available on http://www.cs.toronto.edu/ kriz/cifar.html. We randomly selected 1,000 images and applied the motion kernels that we got in Section 2.1 to each image. Thus, our constructed dataset contains 217,000 blurred images. 173,600 of them were used as training set and 43,400 were used as validation set. And we stack the original image and the blurred image together, resulting in a 6 channel input (RGBRGB), this allows the CNN to learn patterns related to the blurred information.

The construction of the CNN is as follows:

- The first one is convolution layer C1 using 32 (5 x 5) filters, which produces 28 x 28 pixel feature maps followed by ReLU non-linear transform $R(z) = \max(z, 0)$.
- The second one is the sub-sampling layer (Max-pooling) S2 with stride 2, which uses a receptive field of size 2 x 2 and produces 14 x 14 pixel feature maps.
- The third one is the convolution layer C3 using 64 (5 x 5) filters, which produces 10 x 10 pixel feature maps, followed by ReLU.

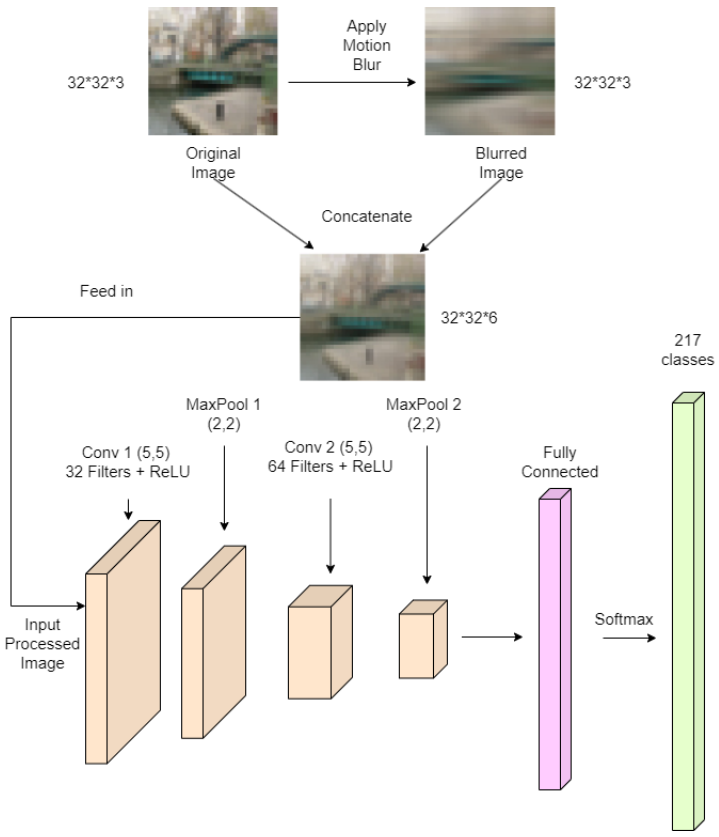

Figure 3: CNN Architecture

- The fourth one is the sub-sampling layer (Max-pooling) S4 with stride 2, which is exactly the same as layer S2, uses a (2 x 2) receptive field, and produces 5 x 5 pixel feature maps.
- The fifth one is a fully connected neural network layer FC5 with 128 input neurons. Regarding the desired number of classes, there exist 217 neurons in the output of the network.
- The sixth one is a soft-max S6, which determines 217 classes in the output of the network, and each class corresponds to a label of a motion blur kernel candidate.

2.3 The Model Guided Lukas-Kanade Framework

We will be applying our model to the Lukas-Kanade(LK) Tracker [1]. The Lukas-Kanade (LK) method is a widely used differential technique for optical flow estimation developed by Bruce D. Lukas and Takeo Kanade. It assumes

that the flow is essentially constant in a local neighborhood of the pixel under consideration, and solves the basic optical flow equations for all pixels in that neighborhood, by the least squares criterion. This results in an over-determined system of linear equations which can be solved for each pixel to obtain the estimates of the optical flow velocities. The basic assumption in the Lukas-Kanade method is the brightness constancy of a point over time, expressed as:

$$I(x, y, t) = I(x + dx, y + dy, t + dt) \tag{2}$$

where $I(x, y, t)$ is the image intensity at position (x, y) and time t, and dx, dy are the displacements in x and y directions over the time interval dt. Assuming the motion is small, a Taylor series expansion approximates this equation as:

$$I(x, y, t) + \frac{\partial I}{\partial x}dx + \frac{\partial I}{\partial y}dy + \frac{\partial I}{\partial t}dt = I(x, y, t) \tag{3}$$

Simplifying the equation, we obtain the optical flow constraint:

$$\frac{\partial I}{\partial x}dx + \frac{\partial I}{\partial y}dy + \frac{\partial I}{\partial t} = 0 \tag{4}$$

To solve for the optical flow vectors $\mathbf{u} = [dx, dy]^T$ across all pixels in a window, the equations are expressed in matrix form and solved using the least squares method:

$$A^T A\mathbf{u} = -A^T\mathbf{b} \tag{5}$$

where A consists of the spatial gradients $\left[\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}\right]$ and $\mathbf{b}$ is the vector of temporal gradients $\frac{\partial I}{\partial t}$. The solution is then given by:

$$\mathbf{u} = (A^T A)^{-1}A^T\mathbf{b} \tag{6}$$

In the domain of object tracking, the displacement between consecutive frames is quantified by the global displacement vector $\mathbf{p}$. This vector is pivotal in updating the tracker's position from one frame to the next. Specifically, for frames $t-1$ to t, the displacement vector $\mathbf{p}$ is applied to adjust the tracker's initial position, effectively shifting the ROI at frame $t-1$ by the vector $\mathbf{p}$. This adjustment aligns the tracker with the new position of the object within the scene, and the tracking algorithm proceeds to analyze this updated segment of the ROI to ascertain further movements.

The global displacement vector $\mathbf{p}$ is iteratively updated by integrating the local displacement vector $\mathbf{u}$, which is computed at each frame. The mathematical representation of this updating process is as follows:

$$\mathbf{p}_t = \mathbf{p}_{t-1} + \mathbf{u}_{t-1} \tag{7}$$

where $\mathbf{u}_t$ denotes the displacement measured between frames t and $t+1$. This iterative updating is critical for maintaining accurate and continuous tracking of the object across the sequence, enabling the tracking system to adapt dynamically to the object's movements. In our proposed model, a key enhancement

has been implemented to predict the trajectory of the tracked object more accurately. The model is designed to estimate the motion vector, denoted as $\mathbf{V}_t$, which predicts the potential endpoint of the tracker at frame t based on the data from frame $t-1$. This motion vector is crucial for guiding the tracker towards the predicted position, thereby optimizing the tracking accuracy and efficiency.

The computation of the motion vector $\mathbf{V}_t$, which guides the tracker's movement from frame $t-1$ to frame t, is informed by the kernels predicted by our CNN model. Each ROI is dissected into 32×32 patches, and the CNN applies to each patch to yield a predicted kernel. The vector components, u_t and v_t, representing the horizontal and vertical components of the motion, respectively, are calculated as follows: [5]

$$u_t = l_{t-1} \cdot \cos(\theta_{t-1}) \tag{8}$$

$$v_t = l_{t-1} \cdot \sin(\theta_{t-1}) \tag{9}$$

where l_{t-1} and θ_{t-1} represent the length and the angle of rotation of the kernel in frame $t-1$, respectively. These parameters are derived from the CNN's output for each patch.

To calculate the overall motion vector $\mathbf{V}_t$ for the entire ROI, we average the motion vectors (u_t, v_t) obtained from all the patches. This aggregation is performed component-wise across all patches to ensure that both direction and magnitude of the motion are accurately represented. The resultant motion vector $\mathbf{V}_t$ is given by:

$$\mathbf{V}_t = \left(\frac{1}{N} \sum_{i=1}^{N} u_{t,i}, \frac{1}{N} \sum_{i=1}^{N} v_{t,i} \right) \tag{10}$$

where N is the total number of patches, and $u_{t,i}$ and $v_{t,i}$ are the horizontal and vertical components of the motion vector derived from the i-th patch.

The motion vector $\mathbf{V}_t$ is then used to update the tracker's position:

$$\mathbf{p}_t = \mathbf{p}_t + \mathbf{V}_t \tag{11}$$

By applying these components, the model proactively adjusts the tracker's path, enhancing its accuracy and responsiveness to the object's dynamics.

The integration of this predictive capability into the tracking framework represents a substantial advancement over traditional methods, which typically rely solely on reactive mechanisms based on observed displacements. By incorporating predictions of future movements, the model facilitates a more dynamic and anticipatory tracking strategy, crucial for handling high-speed objects or objects that change their trajectory abruptly.

2.4 Error Handling in Motion Prediction

The robustness of our object tracking approach is contingent upon the accuracy of the motion predictions provided by our CNN model. While the CNN

efficiently predicts motion vectors, the inherent uncertainty in any predictive model necessitates the implementation of an error handling mechanism. This is critical to ensure that occasional inaccuracies in the model's outputs do not lead to a cumulative tracking error, thereby causing the tracker to deviate from the target object.

To address this, we introduce an intermediary verification step prior to updating the tracker's position based on the predicted motion vector. Specifically, the process involves the following steps:

1. The predicted motion vector $\mathbf{V}_t$ is initially used to hypothesize a new position for the tracker. This position shift leads to a new region of interest (ROI) from which image data is extracted.

2. The new ROI is then used to apply the LK framework, the resulting image I_{new} is used to compare with the predefined template. This comparison is quantitatively assessed using the sum of squared differences (SSD), providing a measure of similarity between the template and the ROI:

$$\text{SSD}_{\text{new}} = \sum_i (I_{\text{new}}(i) - T(i))^2 \tag{12}$$

   where I_{new} is the intensity of the pixels of the ROI with displacements, and T is the template intensity.

3. Concurrently, the SSD is also calculated for the current position of the tracker applying to the LK framework, serving as a control measure:

$$\text{SSD}_{\text{current}} = \sum_i (I_{\text{current}}(i) - T(i))^2 \tag{13}$$

4. The position corresponding to the lower SSD value is selected as the more probable location of the object. If the new position proves to be less similar to the template than the current position, the tracker does not update its location based on the predicted motion vector.

By implementing this checking mechanism, the tracker only updates its position when there is a high likelihood that the predicted motion vector accurately reflects the object's displacement. This method significantly mitigates the risk of the tracker losing the object due to erroneous motion predictions, thereby enhancing the overall stability and reliability of the tracking system. The integration of this verification step effectively makes the tracker more resilient to potential inaccuracies in the motion prediction model, ensuring continuous and accurate tracking even in scenarios where the model's predictions are suboptimal.

3 Experiment

To evaluate the effectiveness of our proposed advancements to the Lukas-Kanade (LK) tracker, we conducted a series of experiments comparing the performance of our enhanced tracker with that of the traditional LK tracker. The

results are visually represented in the 4, 5 and 6, where our tracker is denoted in blue, and the traditional LK tracker is represented in green.

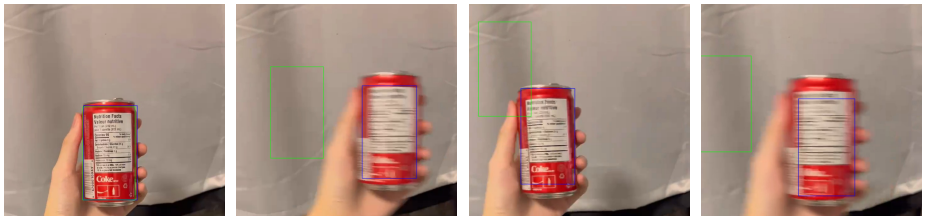

Figure 4: Experiment 1

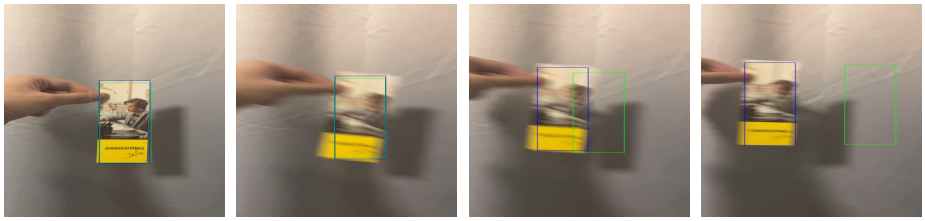

Figure 5: Experiment 2

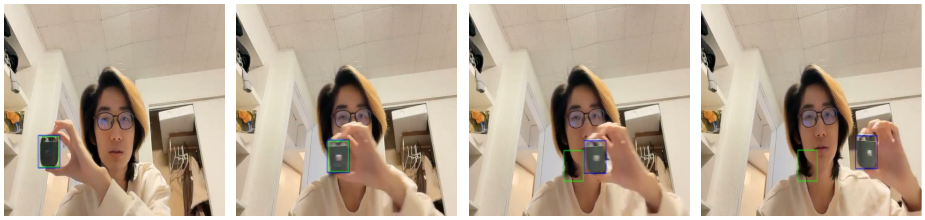

Figure 6: Experiment 3

3.1 Experimental Setup and Observations

Both trackers initially perform adequately under conditions of slow object motion. However, as the velocity of the object increases, distinct differences in tracking accuracy become apparent. Notably, the traditional LK tracker begins to lose track of the object. This degradation in performance can primarily be attributed to the following reasons:

1. **Assumption of Small Motion:** The traditional LK tracker is based on the assumption that motion between consecutive frames is small. This assumption allows the tracker to use a linear approximation to solve the optical flow equations. However, when the object moves rapidly, this

assumption no longer holds, leading to significant errors in motion estimation.

2. **Sensitivity to Large Displacements:** The LK method employs gradient-based optimizations that are not well-suited for handling large displacements. The tracker fails to capture significant object movements across frames, which exceeds the spatial coherence that the LK algorithm can handle.

3.2 Advantages of the Proposed Tracker

In contrast, our enhanced tracker demonstrates robust performance even when the object undergoes fast motion. The key enhancements that contribute to this superior performance include:

1. **Predictive Motion Vector Adjustment:** By incorporating a predictive model that estimates the motion vector based on previous movements, our tracker is able to anticipate larger shifts in position. This anticipation allows for more accurate adjustment of the tracking window in each frame.

2. **Error Handling Mechanism:** The implementation of an intermediary checking step, as described earlier, allows our tracker to validate the predicted motion vector against the current tracking scenario. This step significantly reduces the likelihood of erroneous updates, thus maintaining tracking accuracy even under rapid object movement.

These enhancements enable our tracker not only to keep up with the object but also to maintain accuracy and reliability in scenarios where the traditional LK tracker fails. The results clearly demonstrate the practical benefits of integrating predictive capabilities and error handling into the tracking process, providing a robust solution for real-world applications involving fast-moving objects.

3.3 Limitations of the Proposed Tracker

While the enhancements introduced in our tracker improve tracking accuracy, particularly in challenging conditions involving fast motion, it is imperative to consider the trade-offs between performance improvements and computational efficiency. Below, we discuss the advantages and limitations of both the traditional LK tracker and our CNN-based enhanced tracker.

1. **Increased Computational Load:** The integration of CNN models for predicting motion vectors significantly increases the computational demands. Each patch of the ROI must be processed through the neural network, which can slow down the tracking process, particularly when large ROIs are used to enhance prediction accuracy.

2. **Requirement for Larger ROIs:** Our tracker requires larger regions of interest to allow the CNN to make more accurate predictions. However, processing larger ROIs incurs additional computational costs and can further reduce the tracker's operational speed, potentially limiting its applicability in real-time scenarios.

In conclusion, while our enhanced tracker offers notable improvements in tracking accuracy and adaptability, these benefits come at the cost of increased computational requirements and reduced speed. This trade-off needs to be carefully considered, especially in applications where real-time processing is essential. Future work could focus on optimizing the CNN architecture and streamlining the tracking process to mitigate these drawbacks while preserving the enhancements in tracking performance.

4 Conclusion

The suggested tracking method for a blurred object consists of two main phases. In the first phase, a CNN is trained and used to estimate the blur length and angle of the motion kernels. This helps determine the magnitude and direction of motion vectors. The purpose is to predict the blur kernels from blurred image inputs. To achieve this, a large set of blurred images was created from the CIFAR-100 dataset by convolving original images with various angles and lengths kernels. In the second phase, with the estimated blur kernels by CNN, we can derive the horizontal and vertical motion vectors of ROI and update the position of the LK tracker accordingly. Our scheme conquers the problem of object tracking during the sudden and fast motion or the shakes of the camera to some extent.

References

[1] Simon Baker and Iain Matthews. "Lucas-Kanade 20 Years On: A Unifying Framework". In: *International Journal of Computer Vision* 56 (Feb. 2004), pp. 221–255. DOI: 10.1023/B%3AVISI.0000011205.11775.fd.

[2] Jianwei Ding et al. "Severely Blurred Object Tracking by Learning Deep Image Representations". In: *IEEE Transactions on Circuits and Systems for Video Technology* 26.2 (2016), pp. 319–331. DOI: 10.1109/TCSVT.2015.2406231.

[3] Karim Faez Iman Iraei. "A motion parameters estimating method based on deep learning for visual blurred object tracking". In: *IET Image Processing* (2021).

[4] Alex Krizhevsky, Geoffrey Hinton, et al. "Learning multiple layers of features from tiny images". In: (2009).

[5] Jian Sun et al. *Learning a Convolutional Neural Network for Non-uniform Motion Blur Removal*. 2015. arXiv: 1503.00593 [cs.CV].