Assignment #4.2 Report
Computer Vision

No	Name	ID
1	عبدالرحمن السيد جاد السيد	19015894
2	عبدالعزيز محمد عبدالعزيز محمد	19015941
3	عمر خيرت محمد ابو ضيف	19016063

Google colab link: here
Output Videos: here

1. Problem Statement:

The project aims to address a fundamental aspect of both human and animal vision, emphasizing the crucial ability to track objects and individuals within our field of view. Tracking, integral to our daily experiences, is often taken for granted, whether it involves a predator pursuing its prey or an individual attempting to catch a moving object like a basketball. The task at hand involves the implementation of the Lucas-Kanade tracker, a renowned algorithm for object tracking in videos. Two distinct video sequences are provided for this purpose: one featuring a car on a road and the other highlighting a helicopter approaching a runway. To initiate the tracker, participants must define a template by delineating a bounding box around the target object in the first frame of the video. Subsequently, the tracker will continually update an affine transform, warping each frame to ensure alignment with the template established in the initial frame.

2. Used Libraries:

- **cv2**: It is used for reading and writing images, applying image processing operations, and drawing rectangles on frames.
- **numpy**: It is used for array operations and mathematical computations, particularly in the Lucas-Kanade tracker algorithm.
- **matplotlib**: It is used for creating plots and visualizing images with bounding boxes. In addition to drawing rectangles on images.
- **tqdm**: It is used for creating progress bars for iterations in loops.
- **google.colab**: It is used for mounting Google Drive and downloading files from Google Drive.

3. Preliminaries:

We define the mathematical framework of image transformations or warps, specifically focusing on the implementation of an affine transform denoted as $W(x;p)$, where $p = [p_1, p_2, p_3, p_4, p_5, p_6]^T$ represents the parameters associated with the warp. The code encapsulates this by providing functions for calculating the warp matrix (W(p)), applying the warp to an image, and handling gradient-related computations.

- The `**calculate_warp_matrix**` function computes the 3x3 affine warp matrix, and `**apply_warp**` function utilizes this matrix to warp the image.
- The `**apply_gradient_warp**` deals with applying the warp to gradient information, and `**calculate_delta_p**` computes the change in parameters based on the error and gradient information.

The code structures the affine transform as a 3x3 matrix, aligning with the mathematical representation mentioned in the preliminary section. These functions lay the foundation for the Lucas-Kanade tracker's forward additive alignment approach.

4. Lucas Kanade:

The Lucas-Kanade tracker implemented in the code aligns a sequence of images to a reference template using an affine warp $W(x;p)$. The following functions in the code contribute to the Lucas-Kanade algorithm's forward additive alignment:

- The `**run_tracker**` function initializes the tracker with the template information, image frames, and initial parameters. It iteratively refines the parameters using the forward additive form of the warp.
- We apply warp and calculate error using:

  IW_points, IW_intensity = apply_warp(image, W, template_points)
  error = template_intensity – IW_intensity
- Within the `**run_tracker**` function, the warp is applied to the current image using the `**apply_warp**` function, and the error is calculated by comparing the template intensity with the warped image intensity.
- The `**apply_gradient_warp**` function is responsible for applying the warp to the horizontal and vertical gradient information, contributing to the calculation of the change in parameters.
- The `**calculate_delta_p**` function computes the change in parameters based on the error, gradient information, and template points.
- Then we update parameters and repeat by $initial_p = initial_p + delta_p.T$
- The parameters are updated with the computed change (Δp), and the entire process is repeated iteratively until convergence, or the maximum number of iterations is reached.

The Lucas-Kanade algorithm outlined in the report leverages these functions to minimize the pixel-wise sum of squared differences, aligning the sequence of images to the reference template.

5. Tracking Implementation:

The tracking implementation section outlines the core functions responsible for detecting and tracking an object in a sequence of frames. The code provides a robust framework for initializing the tracker, detecting the object, and continuously refining the warp parameters to ensure accurate tracking.

- The `**run_tracker**` function serves as the entry point for the tracking process. It initializes the tracker with the template information, specifies the region of interest (ROI) using the provided rectangle coordinates, and begins the iterative tracking process.
- The Sobel operator is utilized to compute the horizontal (`**gradient_x**`) and vertical (`**gradient_y**`) gradients of the current image. These gradients are later employed in the Lucas-Kanade tracking algorithm.
- The iterative tracking loop refines the warp parameters by minimizing the pixel-wise sum of squared differences between the template and the warped image. It terminates either when the change in parameters falls below a specified threshold or when the maximum number of iterations is reached.

- Within the loop, the current warp matrix (W) is applied to the initial rectangle coordinates, transforming them into the coordinates of the tracked object in the current frame.
- In case the iterative tracking process reaches the maximum specified iterations without convergence, the function returns the initial safe warp matrix and parameters.
- If the tracking process converges, the final warp matrix and parameters are returned, representing the optimal local motion from frame I_t to frame I_{t+1} .

These components collectively contribute to the comprehensive tracking implementation, ensuring the accurate localization and tracking of the specified object in the video sequence.

6. Example Output Frames:

- Car

- Helicopter Landing