Name	ID
Gamal Abdel Hamid Nasef Nowesar	19015550
Ahmed Mahmoud Abdelhay Elemary	19015359
Zyad Samy Ramadan	19015720

Overview:

It's required to implement the Lucas-Kanade tracker, where two video sequences are provided: a car on a road, and a helicopter approaching a runway link. To initialize the tracker we need to define a template by drawing a bounding box around the object to be tracked in the first frame of the video. For each of the subsequent frames the tracker will update an affine transform that warps the current frame so that the template in the first frame is aligned with the warped current frame.

Lucas-Kanade:

A Lucas Kanade tracker maintains a warp W(x; p) which aligns a sequence of images It to a template T. We denote pixel locations by x, so I(x) is the pixel value at location x in image I. For the purposes of this derivation, I and T are treated as column vectors (think of them as unrolled image matrices). W(x; p) is the point obtained by warping x with a transform that has parameters p. W can be any transformation that is continuous in its parameters p. Examples of valid warp classes for W include translations (2 parameters), affine transforms (6 parameters) and full projective transforms (8 parameters). The Lucas Kanade tracker minimizes the pixel wise sum of square difference between the warped image I(W(x; p)) and the template T. In order to align an image or patch to a reference template, we seek to find the parameter vector p that minimizes L, where:

$$L = \sum_{x} [T(x) - I(W(x;p))]^2$$

In general this is a difficult non-linear optimization, but if we assume we already have a close estimate p of the correct warp, then we can assume that a small linear change Δp is enough to get the best alignment. This is the forward additive form of the warp. The objective can then be written as:

$$L = \sum_x [T(x) - I(W(x; p + \Delta p))]^2$$

Expanding this to the first order with Taylor Series gives us:

$$L \sim \sum_x [T(x) - I(W(x; p)) - \Delta I(x) \frac{\mathrm{d}W}{\mathrm{d}p} \Delta p]^2$$

Where:

$$\Delta I(x) = [\frac{\mathrm{d}I(x)}{\mathrm{d}u}, \frac{\mathrm{d}I(x)}{\mathrm{d}v}]$$

which is the vector containing the horizontal and vertical gradient at pixel location x.
Rearranging the Taylor expansion, it can be rewritten as a typical least squares approximation:
$\Delta p^* = \text{argmin}_{\Delta p} ||A\Delta p - b||^2$

$$\Delta p^* = argmin_{\Delta p} \sum_x [\Delta I(x) \frac{\mathrm{d}W}{\mathrm{d}p} \Delta p - \{T(x) - I(W(x; p))\}]^2$$

This can be solved with:

$$\Delta p^* = (A^T A)^{-1} A^T b$$

Where:

$$(A^T A) = H = \sum_x [\Delta I(x) \frac{\mathrm{d}W}{\mathrm{d}p}]^T [\Delta I(x) \frac{\mathrm{d}W}{\mathrm{d}p}]$$

$$A = \sum_x [\Delta I(x) \frac{\mathrm{d}W}{\mathrm{d}p}]$$

$$b = T(x) - I(W(x; p))]$$

Once ∆p is computed, the best estimate warp can be updated p = p + ∆p, and the whole procedure can be repeated again, stopping when ∆p is less than some threshold.

Tracker Implementation:

Affine Transformation Functions:

- The `affine_warp` function:
  creates a 2x3 affine transformation matrix given parameters `p`.
- The `update_affine_parameters` function:
  updates affine parameters using the Lucas-Kanade algorithm.

  ### Input Parameters:
  - `coordinates`: Original coordinates of the template window.
  - `template`: Template frame.
  - `img`: Image frame.
  - `x1, x2, y1, y2`: Bounding box coordinates.
  - `p`: Affine parameters.
  - `Ix, Iy`: Gradients of the image frame.

  ### Coordinate Transformation:
  - `v` is a 3x4 matrix containing the original coordinates of the bounding box.
  - The affine transformation matrix is calculated using `affine_warp(p)`.
  - The original bounding box coordinates `v` are then transformed using this affine matrix to get `new_coordinates`.

  ### Boundary Adjustment:
  - The `new_coordinates` are adjusted to ensure they lie within the image boundaries.

  ### Warp Image:
  - Using the adjusted `new_coordinates`, a new image array `img_arr` is obtained by sampling the image frame.

  ### Residual Calculation:
  - `b` is the residual or the difference between the template and the warped image.

  ### Gradient Calculation:
  - Gradients `sx` and `sy` are computed using the Sobel operator on the normalized image.

Matrix A Construction:

- Matrix `A` is constructed by stacking the gradients and their products with the original coordinates.

Hessian and Parameter Update:

- The Hessian matrix is calculated using `A.T @ A`.
- The pseudo-inverse of the Hessian is computed (`Hessian_inverse`).
- The update to the parameter vector `delta_p` is obtained using the Lucas-Kanade formula.

Norm Calculation:

- The Euclidean norm of `delta_p` is calculated.

Parameter Update:

- The parameter vector `p` is updated by adding `delta_p`.

Return Values:

- The function returns the updated parameters `p`, the norm of `delta_p` (`p_norm`), and the transformed coordinates `v`.

Lucas-Kanade Object Tracking Function:

- The `Lucas_Kanade` function implements the Lucas-Kanade object tracking algorithm.
- It takes input frames, a template, bounding box coordinates, and affine parameters `p`.
- It performs iterative updates using the Lucas-Kanade algorithm until convergence or a maximum number of iterations is reached.
- The result is visualized by drawing a polyline on the output frame.

Input Parameters:

- `img_frame`: Current frame of the video.
- `template_frame`: Template frame for the object.
- `x1, y1, w, h`: Bounding box coordinates and dimensions.
- `p`: Affine parameters.

Initialization:

- `x2` and `y2` are calculated as the opposite corners of the bounding box.
- `norm_frame` is calculated by normalizing the current frame based on the mean intensity of the template frame.

Template Window Coordinates:

- A 3xN matrix (`template_window_coordinates`) is created to represent the coordinates of the template window.

- Coordinates are generated for the template window based on the bounding box.

Extract Template Frame Intensities:
- Intensities of the template frame are extracted at the template window coordinates.

Gradient Calculation:
- Gradients `Ix` and `Iy` are calculated using the Sobel operator on the normalized image.

Lucas-Kanade Iterative Update:
- The function enters a while loop that continues until convergence (defined by `threshold`) or a maximum number of iterations (`max_iterations`).
- The `update_affine_parameters` function is called in each iteration to update the affine parameters.

Parameter and Result Handling:
- If the updated parameter norm is smaller than the original parameter norm, the parameter vector `p` is updated.
- The updated parameter `p` is used to draw a polyline (`cv2.polylines`) on the current frame, forming the tracked object.

Return Values:
- The function returns the updated parameters `p` and the output frame `out`.

Object Tracking in a Video Function:

- The `track_object` function takes a video, initial bounding box coordinates, output video path, frame size, frame rate, and number of frames.
- It initializes a video writer and iterates through frames, applying the Lucas-Kanade algorithm.
- The resulting frames are written to the output video.

  ### Input Parameters:
  - `video`: The video sequence as a NumPy array.
  - `rectangle`: Initial bounding box coordinates (x, y, width, height).
  - `output_video_path`: Path to save the output video.
  - `frame_size`: Size of the video frames (width, height).
  - `frame_rate`: Frame rate of the output video.
  - `num_frames`: Number of frames to process.

  ### Video Writer Initialization:
  - `fourcc` is used to specify the codec for the video writer.
  - `outVideo` is a video writer object initialized with the specified parameters.

Template Initialization:

- The template is set as the first frame of the video.

Bounding Box and Parameter Initialization:

- Bounding box coordinates (x, y, width, height) are extracted from the input rectangle.
- Initial affine parameters `p` are set to zeros.

Video Processing Loop:

- The function iterates over the specified number of frames (`num_frames`).
- For each frame, a copy of the transposed frame is obtained (`curr_frame`).
- The `Lucas_Kanade` function is called to track the object in the current frame, using the template, bounding box coordinates, and affine parameters.
- The updated parameters and the resulting frame are obtained.

Video Writing:

- The output frame (`out`) is written to the output video.

Video Release and Move:

- The video writer is released after processing all frames.
- The `shutil.move` function is used to move the output video file, though it appears to be moving the file to the same location (`output_video_path` to `output_video_path`), which seems unnecessary.

Object Tracking for "car" and "landing" Videos:

- The code loads two video sequences, 'car2.npy' and 'landing.npy', from the specified paths.
- It defines bounding box coordinates for the initial position of the object in each video.
- Calls `track_object` for each video with the specified parameters.

Screenshots from output videos:

Code Link: Tracking-Objects.ipynb

Output Videos Link: cv_assig_5