Differential Motion & Multi-Directional Optical Flow for Motion Analysis

Team YOLO - s19 (205, 355, 408, 420)

1. Introduction

Differential motion

Differential motion analysis is a simple yet effective technique for detecting motion between two images by computing the pixel-wise difference. It highlights regions where changes occur, making it useful for identifying moving objects.

Optical flow

Optical flow is the one method that can be used for detecting and analyzing the motion. It analyzes the pattern of apparent motion of objects within a sequence of images or video frames. This project focuses on implementing an optical flow method to estimate motion between two consecutive frames using a block-matching approach in multiple directions (horizontal, vertical, and diagonal).

2. Key Definitions

Absolute Difference - A pixel-wise operation that computes the absolute intensity difference between two grayscale images, used to highlight regions where motion has occurred.

Thresholding (Fixed and Adaptive) - A technique to convert grayscale difference images into binary form by separating pixels with significant change (motion) from the background, whereas adaptive thresholding adjusts dynamically to lighting variations.

Optical Flow - The motion of objects, surfaces, or edges, represented as a vector field where each vector indicates the direction and magnitude of motion at a pixel.

Block Matching - A technique to estimate motion by comparing small patches (blocks) of pixels between two frames to find the best match.

Flow Field - A 2D array where each element is a vector (u, v) representing the horizontal (u) and vertical (v) displacement of a pixel between two frames.

3. Project Steps

I. Problem Identification (1 & 3 tasks) & Project Planning
II. Data Collection
III. Algorithm Development and optimizing
IV. Testing with a couple of images
V. Report writing (3 tasks)

4. Methodology

Task 01

The differential motion analysis method is an effective technique for detecting motion between two images by computing the pixel-wise difference and can be enhanced to handle noise, illumination variation, and low-light conditions for more accurate results.

1. Image Acquisition - Capture or load two images with the same background and possible object motion. (In task one I used my owned image pairs)

2. Preprocessing -
    a. Convert images to grayscale.
    b. Apply CLAHE for contrast enhancement (especially in low light).
    c. Use Gaussian Blur to reduce noise.

3. Differencing - Compute the absolute difference between the two grayscale images to highlight changes.

4. Thresholding -
    a. Apply adaptive thresholding or a fixed threshold to extract motion regions.
    b. Invert the result for better visibility if needed.

5. Visualization - Display original images, motion mask, inverted result, and direction vectors using matplotlib.

Task 03

The methodology involved implementing a multi-directional block-matching algorithm to estimate optical flow.

1. 1D Flow Estimation - The function estimate_1d_flow() computes the optical flow in a specified direction (horizontal, vertical, diagonal1, or diagonal2)

2. Combining Flows from Multiple Directions - The function compute_optical_flow() estimates the flow in four directions and combines them using the average_flows() function.

3. Smoothing the Flow Field - The smooth_flow() function reduces noise in the flow field by averaging each pixel's flow vector with its 8 neighbors.

4. Visualization - The plot_flow_streamlines() function visualizes the flow field using streamlines.

5. Results and Discussion

Task 01

Normal Differential Motion Analysis Implementation - Figure 01 shows the Normal Differential Motion Analysis Implementation and this is the basic version of the algorithm. It computes the absolute difference between two grayscale images and applies a fixed threshold to highlight motion areas. It works well in clear, static lighting with minimal noise and a single moving object.

Differential Motion Analysis for Multiple Objects - The same algorithm is tested with multiple moving objects [Figure 02]. It successfully identifies all motion regions, indicating that the approach can generalize well to multiple motion sources as long as the background remains unchanged and the lighting is consistent.

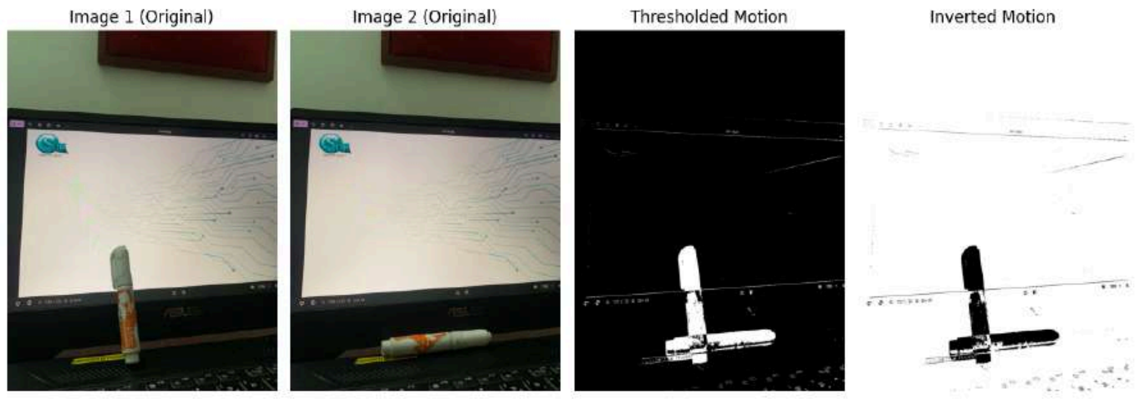


Figure 01 : Normal differential motion analysis implementation

With Noisy Images - Noise is introduced in the images [Figure 03], which causes false positives. The fixed thresholding is sensitive to pixel-level fluctuations, making the result less reliable. This highlights the algorithm's limitation under noisy conditions without any preprocessing like blurring.

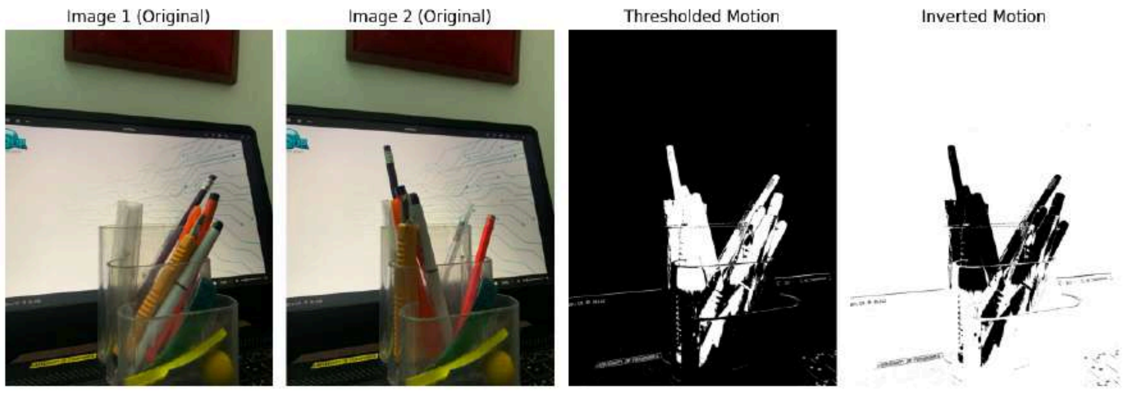


Figure 02 : Normal differential motion analysis implementation for multiple objects

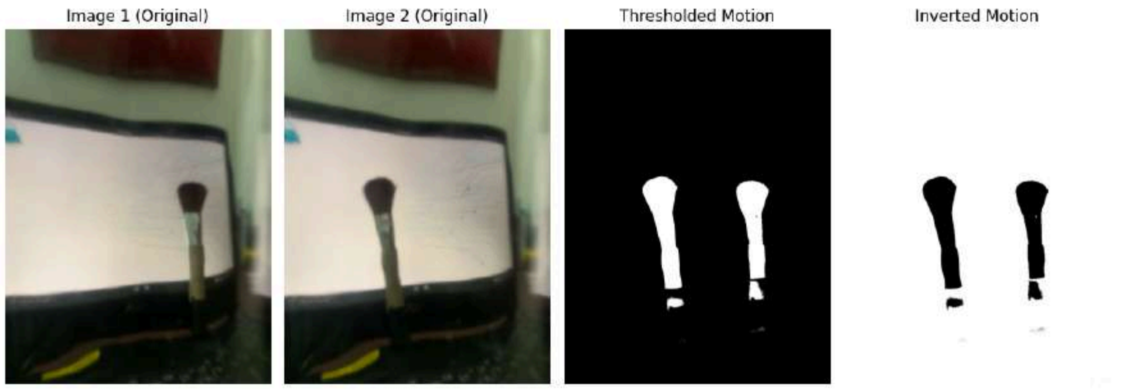


Figure 03 : Differential motion analysis implementation with noisy images

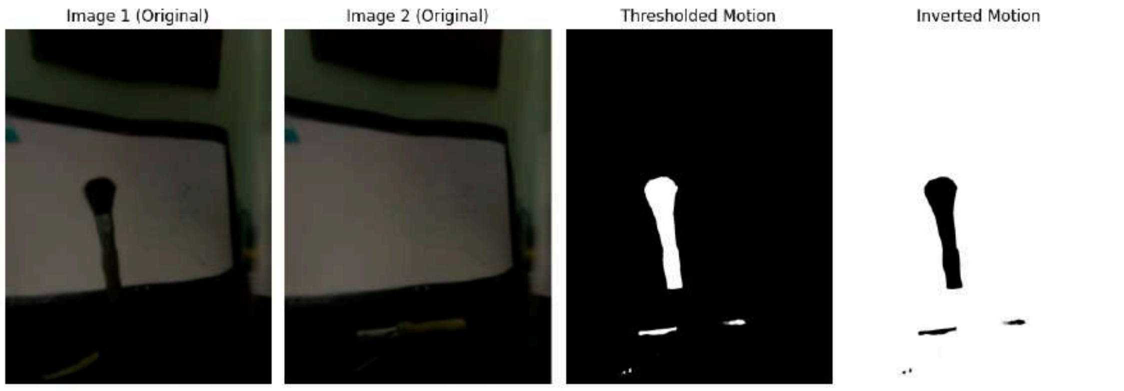


Figure 04 : Differential motion analysis implementation for low light conditions

For Low Light Conditions - In low-light images [Figure 04], the original method struggles to highlight motion clearly due to reduced contrast. Only partial motion is detected. This shows that without contrast enhancement, performance significantly drops under poor lighting.

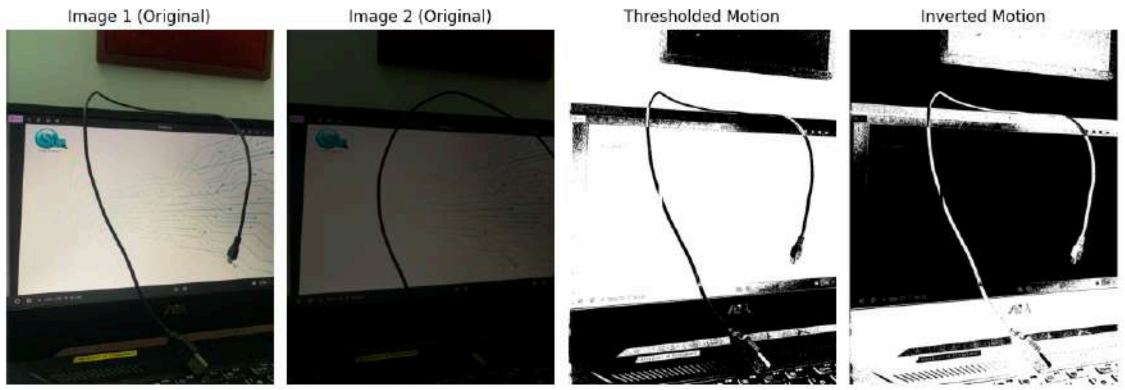


Figure 05 : With illumination changed images

With Illumination Changed Images - When lighting conditions vary between the two images, fixed thresholding causes over- or under-detection of motion areas. The performance becomes inconsistent, emphasizing the need for more adaptive techniques. [Figure 05]

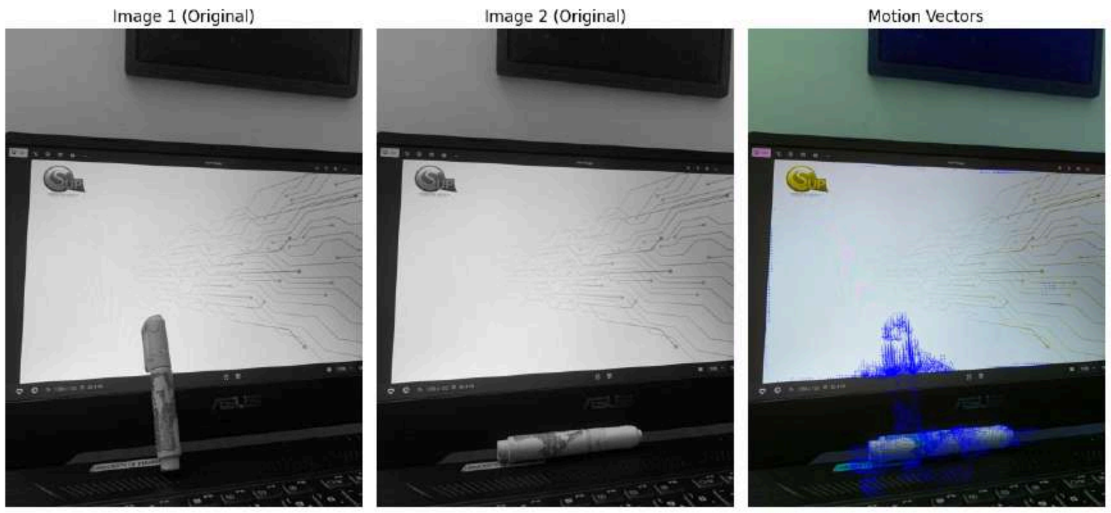


Figure 06 : Improved version with motion directions

Improved Version with Motion Directions - This version [Figure 06] integrates optical flow (Farneback method) to not only detect motion but also visualize its direction. Arrows represent the flow vectors, and a magnitude threshold filters out weak/noisy movements. This enhances interpretability and adds directional insight.

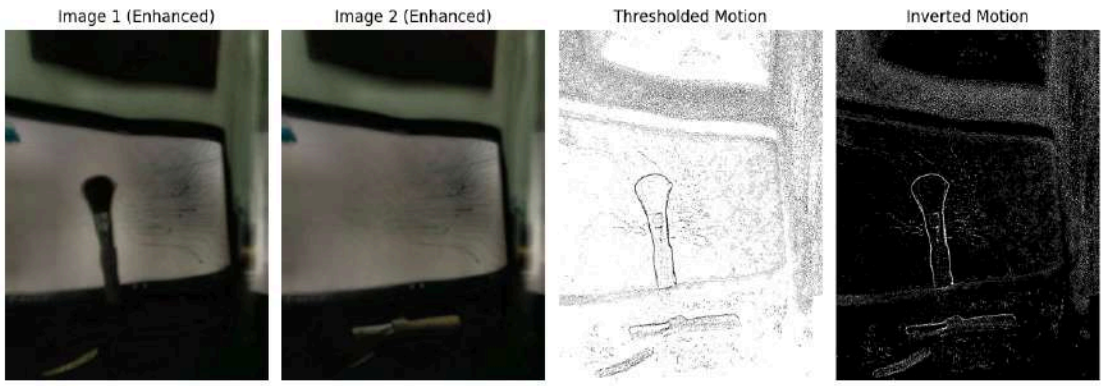


Figure 07 : Improved version for low light conditions

Improved Version for Low Light Conditions - This enhanced version applies CLAHE for contrast improvement and Gaussian Blur for noise reduction, followed by adaptive thresholding. These steps significantly boost detection accuracy in dark scenes, resolving the limitations observed in Figure 04.

Q1. What threshold values should be used and why?
The threshold value determines how sensitive the motion detection is,

i. Low threshold (< 20) - Captures small pixel changes, but may detect noise or illumination variations as motion.

ii. Medium threshold (30 - 50) - Best for general motion detection while ignoring small lighting changes.

iii. High threshold (> 70) - Detects only large movements, missing fine motion details.

For real-world applications, a value of 30-50 is a good starting point, adjustable based on the environment.

Q2. What improvements can be made to this method to include the direction of the motion?

To include motion direction, we can use,

i. Optical Flow (Lucas-Kanade or Farneback) - Tracks pixel movement frame-to-frame to get direction vectors.

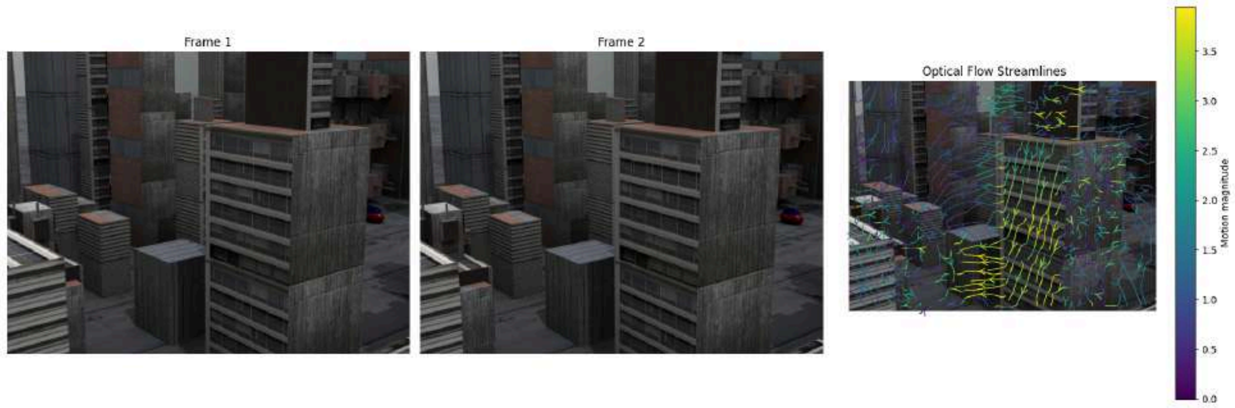
ii. Frame Differencing with Contours & Arrows - Finds motion bounding boxes and estimates direction using past positions.

Task 03

These images are used from https://vision.middlebury.edu/flow/data/

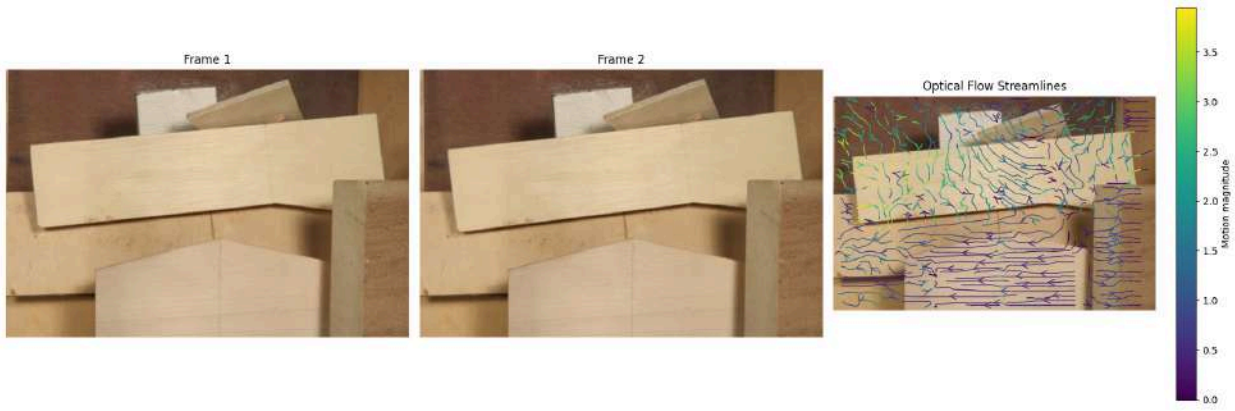
01. The result shows that the first frame, the second frame, and the optical flow streamlines. Streamlines show how motion has happened between the frames. color bar indicating the magnitude of motion.

We can see that the right side building has indicated more magnitude motion compared to the left part. And also arrows indicate the direction of each point.



Since the streamlines show the motion direction, it is very important to identify the correct flow fields for better visualization. According to the streamlines, we can say the right-hand side indicates more down motion than the left side.

02. The left side of the frames indicates comparatively more motion than the right. According to the arrow direction, motion has occurred toward the bottom. However, other motions show low magnitude.



According to the final streamlines, we can say that most of the motion has occurred at an average magnitude level. As mentioned above, identifying the correct block is very important to indicate the correct motion path.

Block matching is a technique used to find corresponding points between two frames by comparing small regions (blocks) of pixels. In the above example, most of the wood parts are identified as similar blocks. That might lead to the indication of more motions in the final result.

6. Task Breakdown

The project was divided into three main tasks,

Task	Team Members
Task 01	S19355
Task 02	S19205
Task 03	S19408
Presentation Creation & Presenting	S19420

7. Conclusion

Task 01

We tested the differential motion analysis algorithm on various images, including single and multiple object shifts, noise, and illumination changes. While it performed well, it struggled in low-light conditions, detecting only partial motion areas.

To improve low-light performance, we applied CLAHE for contrast enhancement, Gaussian Blur to reduce noise and adaptive thresholding for better detection under varying lighting conditions. These adjustments significantly improved accuracy.

For clearer motion direction, we integrated OpenCV's Farneback optical flow, plotting motion vectors as arrows. A magnitude threshold was applied to filter out small, irrelevant movements, making directional changes more visible and precise.

Task 03

This project successfully implemented motion analysis using an optical flow algorithm using a multi-directional block-matching approach. Developed block-matching algorithms can identify the small patches of pixels in four directions - horizontal, vertical, and two diagonals. The implementation was carried out using Python with libraries such as OpenCV, NumPy, and Matplotlib.

The process is extracting patches from the first frame (img1) and then searching for the best match in the second frame (img2) within a defined search range. The final results were visualized through streamlines as we mentioned earlier. It effectively illustrated the motion patterns between the frames.

However, we noticed some challenges such as handling occlusions and computational efficiency. For enhancements, we can use advanced techniques like optical flow methods such as Lucas-Kanade or Horn-Schunck.

8. References

1. https://vision.middlebury.edu/flow/data/
2. https://en.wikipedia.org/wiki/Lucas%E2%80%93Kanade_method
3. Szeliski, R. (2010). *Computer Vision: Algorithms and Applications*. Springer.
4. University of Peradeniya. (2025). *CSC4082_2025_Motion Analysis* [Lecture notes].
5. Kadam, P., Fang, G., & Zou, J. J. (2022). Object tracking using computer vision: A review. *Sensors*, *22*(13), 4781. https://doi.org/10.3390/s22134781
6. Pinto, A. M. G., Moreira, A. P., Costa, P. G., & Correia, M. V. (2021). Revisiting Lucas-Kanade and Horn-Schunck. *INESC TEC, Department of Electrical and Computer Engineering, Faculty of Engineering of the University of Porto*, Portugal.