Feature detection enhancement using FAST and Harris feature detectors with Temporal Smoothing

(Feature Detection Using Harris and FAST Operators for Real-Time Video Analysis)

Suryabhan Singh
School of Computer Science and Engineering
Lovely Professional University
Punjab, India
suryabhan.12114479@lpu.in

Chellingi S. S. D. Bhagavan
School of Computer Science and Engineering
Lovely Professional University
Punjab, India
chellingi.12114728@lpu.in

***Abstract*—This paper focuses on briefing of siginificance of features in Computer Vision field. Alongside, It also enlights feature analysis methods like Harris Corner Detector and FAST (Feature from Augmented Segment Test) feature detector method. Later on, It discusses a problem arising in case when these detectors and are used over a video, instead of image. And, what could be a possible solution for it.**

Keywords—Computer vision, FAST, Harris corner, feature detector, Machine learning

I. Introduction

In today's rapidly advancing technological landscape, artificial intelligence (AI) is transforming industries, often with machine learning (ML) as its core driving force. Despite widespread enthusiasm for these technologies, the underlying mathematical principles and mechanisms are less widely understood. OpenCV, a comprehensive computer vision library, integrates many of these mathematical foundations, making complex image processing and ML techniques more accessible. However, understanding the operational mechanisms of specific feature detection algorithms remains a challenge for many.

This paper focuses on two widely used feature detection algorithms: the Harris and FAST operators.

While effective, these methods can face challenges in real-time video analysis, particularly with feature consistency across frames, where features often diminish or shift, reducing the reliability of detected points. To address this, we conducted experiments incorporating Temporal Smoothing, a technique designed to enhance inter-frame feature stability, potentially offering a robust solution to mitigate feature loss across frames in dynamic video sequences.

II. Literature Study

A. *Features and it's significance in Computer Vision*

In machine learning, features act as the essential input for model training and testing, guiding the analysis and predictions. However, not all data qualifies as a feature. A feature, in a mathematical sense, is any distinct characteristic that differentiates an object from its surroundings, such as color, texture, edges, corners, and curves.

The challenge then becomes determining which features are most relevant for a given model. To address this, a range of algorithms have been developed to identify and extract specific features from raw data. In this study, we focus on extracting corners and other key features in image-based data, particularly exploring methods that enhance corner detection for improved model performance.

B. *Techniques for recognizing features from data*

Features such as corners and edges follow identifiable patterns that can be detected using mathematical operators like the Sobel operator. The Sobel edge detection method, a gradient-based approach, applies two linear convolution kernels across the image: one for detecting horizontal edges (Gx) and another for vertical edges (Gy). The kernels move pixel by pixel and line by line, calculating intensity changes to identify edges in both x and y directions, enabling effective feature extraction for further analysis [1].

a) But, What are different tectures of an image?

1. **Flat region:** A flat region is a window frame of any graphic that has very minor minor changes in image intensity.

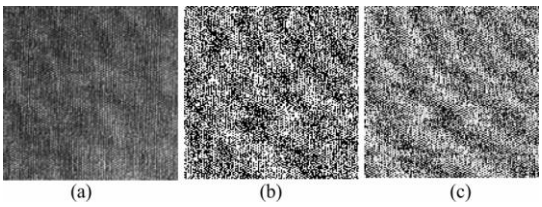

Fig 1: A flat surface in (a): The original image,(b): the RABGLD image,(c): the LBP image. [2]

2. **Edge Region:** An edge region is a window frame that has rapid changes in the image intensity but only in one direction [2].

Fig 2: An edge region

3. **Corner region:** A corner region is a window frame that has rapid changes in two direction [2].

Fig 3: A corner region

b) Sobel operator and it's working mechanism

Sobel operator works mainly on directional image gradients. An image gradient is the fundamental is the fundamental concept that describes the change in change of color or intensity of an image [1]. It is consists of two components :

- Vertical component (G_x) : This only keeps high intensity values in X-direction on a cartesian plane
- Horizontal component (G_y) : This only keep high intensity values in Y-direction on a cartesian plane.

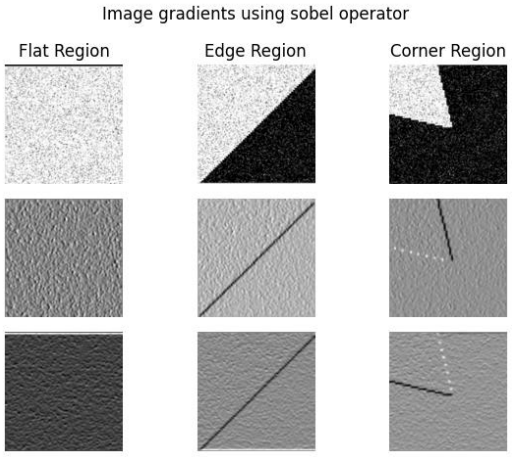

Fig 4: Different type of regions (Flat, Edge and Corner) under sobel operator

In the figure 4, sub-figure 5 and 8, It is clearly visible how edges are only seen within one direction.

Similarly, in sub-figure 6 and 9, It is clearly visible that both edges are colored differently as **according to sobel operator, positive or desired data is indicated with black line while negative is indicated with white dotted line, while rest are just gray.**

Overall, gradient of an image can be defined by this equation:

$$G = \sqrt{G_x^2 + G_y^2}$$

The sobel operator take a convulation product of the Sobel mask over the image to perform this task. The masks used are these:

$$G_{x_m} = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix}$$

$$G_{x_y} = \begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix}$$

Sobel masks for X and Y components

Now, Sobel operator made the task much easier to distinguish between an edge and a corner with this. And, that's how advanced systems like Harris and FAST operator work.

So, we can use it with built-in OpenCV Sobel module

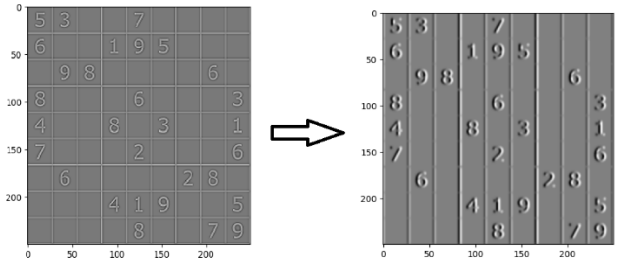

Fig 5 : Vertical sobel component indicating only X direction high pixels and ignoring others.

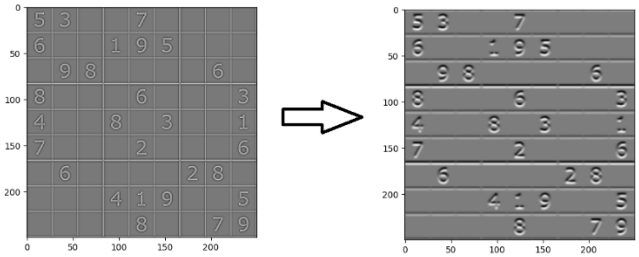

Fig 6 : Horizontal Sobel component indicating only Y direction high pixels and ignoring others

Through mathematical perspective, to distinguish the directional gradient distribution, we can the detected points and put inside an eclipse like this:

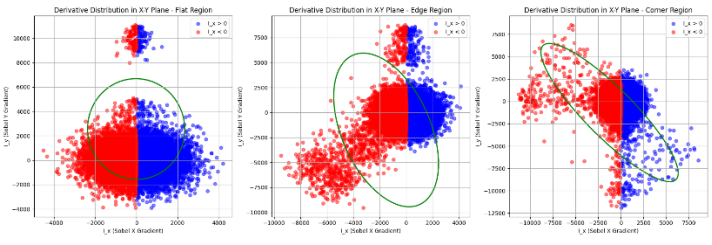

Fig 7 : Derivative distribution for in an eclipse for a) Flat region, b) edge region and c) cirner region

See, how direction of ring bent on covering the gradient distribution. One can draw an angle between midway and maxima of the ellipse and find out the angle. These critical angle works as defining and distinguishing the region for Sobel operator.

c) But, What is Harris corner detector?

Harris Corner Detection is an algorithm used in computer vision to identify corner points in an image, which are points where the image intensity changes sharply in multiple directions. A corner is typically defined as a point where two edges meet, and it exhibits a significant change in intensity in both the x and y directions [3].

- Gradient Computation: The first step in Harris Corner Detection is to compute the gradients of the image in the x and y directions, denoted as I_x and I_y. This is done using convolution with Sobel operators or other gradient filters.

$$I_x = \frac{\partial I}{\partial x}, \quad I_y = \frac{\partial I}{\partial y}$$

- Structure Tensor (Second Moment Matrix): Next, the structure tensor MMM is computed, which summarizes the gradients in both directions. It is defined as a 2x2 matrix:

$$M = \begin{matrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{matrix}$$

  Here, the elements of the matrix represent the sums of the squared gradients and the product of the gradients over a local window (usually a Gaussian window).
- Eigenvalues of the Structure Tensor: The eigenvalues λ_1 and λ_2 of the matrix MMM are computed. These eigenvalues give information about the local image structure. If both eigenvalues are large, it indicates that the intensity changes significantly in all directions, which is characteristic of a corner.
- Corner Response Function: The corner response function RRR is used to identify corners. It is defined as:

$$R = \det(M) - k \cdot \left(\text{trace}(M)\right)^2$$

  where det (M) is the determinant of the structure tensor, trace(M)\text{trace}(M)trace(M) is the sum of the diagonal elements (i.e.,$(I_x^2 + I_y^2)$), and k is a constant (typically set between 0.04 and 0.06).
- Thresholding**:** The final step is to apply a threshold to the response function RRR. If RRR exceeds a certain threshold, the point is considered a corner. The threshold is usually determined empirically.

$$R(x, y) > T$$

where (T) is the threshold value, and (x, y) are the coordinates of the pixel.

Formula:

$$R = \det(M) - k \cdot \left(\text{trace}(M)\right)^2$$

Where:

$$\det(M) = I_x^2 I_y^2 - \left(I_x I_y\right)^2, \quad \text{trace}(M) = I_x^2 + I_y^2$$

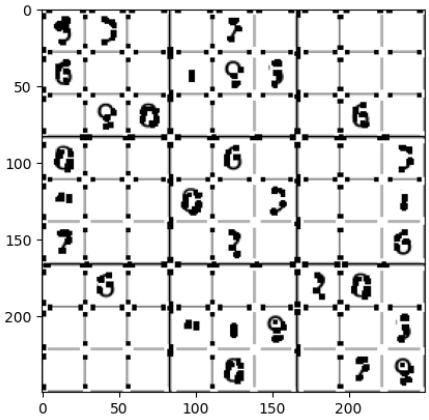

Fig 8: Corners of a checkboard detected using harris corner operator

d) And, What is FAST operator?

The FAST algorithm uses a binary test for corner detection. It works by analyzing the intensity changes of pixels around a central pixel. The central pixel is compared with its surrounding pixels in a circular neighborhood. If a sufficient number of surrounding pixels are significantly brighter or darker than the central pixel, the central pixel is considered a corner.

The main idea is to test whether a pixel is a corner based on whether the intensity of pixels within a circular region around it exceeds a certain threshold. If the intensities of at least a specified number of consecutive pixels in the region are either greater or less than the intensity of the central pixel by a certain threshold, the central pixel is marked as a corner.

Algorithm Steps:

1. Neighborhood Definition: A circular neighborhood of pixels around the central pixel p is considered, typically consisting of 16 pixels arranged in a circle. The surrounding pixels are indexed as $(p + \Delta p_i)$, where ($i = 0, 1, \ldots, 15$).

2. Intensity Thresholding: For each surrounding pixel $(p + \Delta p_i)$, we compute the intensity difference between $(I(p))$ (the intensity of the central pixel) and $(I(p + \Delta p_i))$ (the intensity of the surrounding pixel). A corner is detected if a sufficient number of surrounding pixels are significantly brighter or darker than the central pixel.

3. Binary Test: A threshold (t) is applied to the intensity differences. If the difference between the intensity of the center pixel and the surrounding pixels exceeds the threshold, the pixel is considered a corner [4] [5].

4. Corner Decision: The number of surrounding pixels that meet the condition is counted. If this number exceeds a predefined value, the central pixel is considered a corner [4].

Mathematical Formulation

1. Intensity Comparison for Corner Detection:

The corner detection criterion in FAST can be expressed mathematically as:

$$\text{FAST}(p) = \sum_{i=0}^{N-1} |I(p) - I(p + \Delta p_i)| > t$$

Where:

- (p) is the center pixel.
- (Δp_i) are the positions of the surrounding pixels in a circular pattern around the central pixel.
- ($I(p)$) and ($I(p + \Delta p_i)$) represent the intensity values of the central pixel and surrounding pixels.
- (t) is a threshold, typically set by the user, that determines the sensitivity of corner detection.
- (N) is the total number of surrounding pixels (usually 16).

2. Thresholding for Corner Decision:
For each surrounding pixel (Δp_i), a binary test is performed:

$$|I(p) - I(p + \Delta p_i)| > t$$

Where:

- If the intensity difference exceeds the threshold (t), the pixel is considered a part of a potential corner.
- The total number of such pixels is counted, and if the count exceeds a certain predefined number, (p) is classified as a corner.

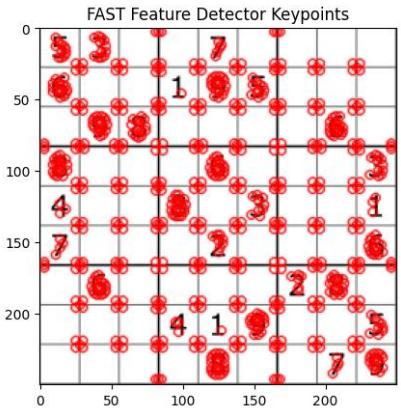

Fig 9 : Corners of a checkboard detected using harris corner operator

III. Related work

Corner detection is a critical aspect of image processing and computer vision, with numerous algorithms developed to enhance the accuracy and efficiency of feature extraction. The Harris corner detection algorithm, introduced by Harris and Stephens [6], remains one of the most widely used methods due to its effectiveness in identifying corner points based on local intensity variations. However, traditional implementations often suffer from issues such as corner information loss and migration due to the use of Gaussian smoothing filters, which can either oversmooth edges or introduce false corners when using smaller windows.

A) Harris with B-Spline: To address these limitations, various enhancements have been proposed. For instance, the work by Liu et al. [7] explores B-spline wavelet edge detection, which highlights the potential of B-spline functions in improving edge detection performance. This aligns with the findings of the current study, which integrates B-spline functions into the Harris corner detection framework to create a more adaptive and robust algorithm.

Additionally, methodologies that focus on adaptive thresholding and local maxima detection have been explored to refine corner detection results. The performance assessment of feature detection algorithms by Rocket [8] emphasizes the importance of parameter selection and the impact of threshold settings on detection accuracy. The proposed algorithm in this paper mitigates these issues by employing a new response function that reduces the randomness associated with parameter selection, thus enhancing the uniformity of detected corners.

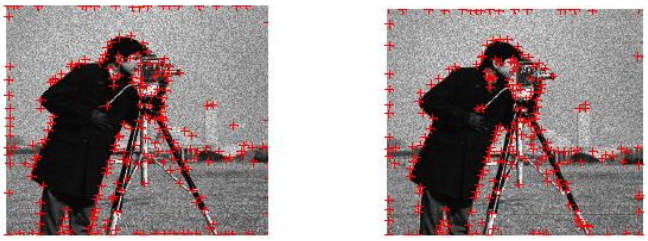

Fig 10: (a) Improved algorithm detect corners with B-Spline Harris

(b) Original algorithm detect corners with B-Spline Harris

B) luvHarris: The work by Vasco et al. [9] introduced a fast event-based Harris corner detection method that exploits the advantages of event-driven cameras, significantly improving detection speed. However, it still faces challenges in balancing event throughput and detection accuracy, particularly in high-speed scenarios. Similarly, Mueggler et al. [10] proposed a fast event-based corner detection approach, yet the reliance on batch processing can hinder real-time performance.

In contrast, the luvHarris [11] method presented in this paper introduces a novel event-surface compatible with the Harris algorithm, enabling asynchronous processing of event streams. This decoupling from the corner detection pipeline allows for continuous event processing, enhancing both speed and responsiveness in dynamic environments. The proposed method demonstrates a significant improvement in

performance, achieving over 2.6 times faster processing compared to existing solutions while maintaining high accuracy in corner detection [9] [10]

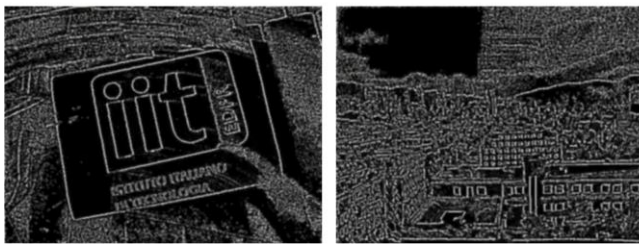

Fig 11 : (a) Improved algorithm detect corners with luvHarris

(b) Original algorithm detect corners with luvHarris

IV. The problem

In corner detection experiments on images, the input is usually static, meaning the detected features remain relatively stable throughout the process. However, in video sequences, the dynamic nature of the scene presents unique challenges. As objects move, corners may disappear temporarily due to occlusion, rapid motion, or variations in lighting, making them hard to track from one frame to the next. This results in a loss of detected features and compromises the consistency of feature tracking across frames.

Unlike static images, video sequences require corner detection algorithms to be more robust to these variations. The frame-to-frame discontinuity in feature presence can cause a breakdown in tracking, especially when rapid motion or changes in the scene occur. Corner points that appear in one frame may be lost entirely in the next, leading to inaccurate or incomplete feature tracking. In such cases, relying solely on image-based detection can lead to poor feature continuity, resulting in a lack of temporal consistency, which is critical for real-time video applications.

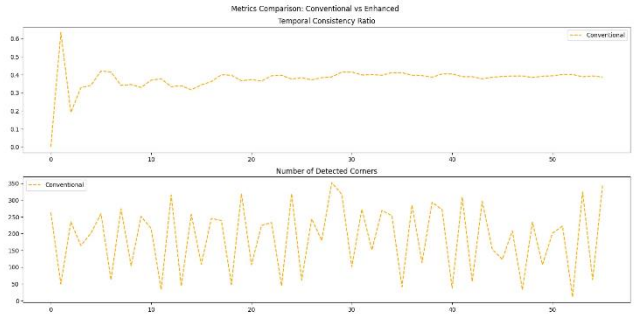

Fig 12 : Graph 1 signifying consistency of the frames While Graph 2 Signifying corners detected in each frames

In Figure 12, Graph 2, one can see how features drops to less than 30 feature from 300. That is near about 90% loss. However, it is not permamnent as it goes back to previous level in next frame but that may cost more resources to a model feed in for this fluctuation.

V. Possible solutions

Temporal smoothing offers an effective solution to this issue by utilizing information from multiple frames to stabilize feature positions over time. Instead of relying on the detection of corners in individual frames, temporal smoothing takes advantage of the temporal continuity inherent in video sequences. By averaging or smoothing corner positions across a series of consecutive frames, this approach mitigates the impact of occlusion, motion blur, or other disruptions that might cause features to be missed or incorrectly tracked. The temporal smoothing process can help maintain the presence of important features even when they are temporarily obscured or lost in individual frames, improving the overall stability and consistency of corner detection in video analysis.

By introducing temporal smoothing into the corner detection pipeline, features that might have been lost in a single frame can be recovered or stabilized, leading to more accurate and consistent results over time. This technique is particularly valuable in real-time video applications, where maintaining feature stability is crucial for tracking, motion analysis, and other video processing tasks. Thus, temporal smoothing is a vital enhancement for corner detection algorithms, especially when extended to dynamic video environments.

Temporal smoothing is a technique that mitigates this issue by incorporating information from multiple frames to stabilize the detected features. A typical smoothing method could be represented as a weighted average over consecutive frames:

$$C_t^{smooth} = \alpha C_t + (1 - \alpha) C_{t-1}$$

where (C_t) represents the detected corner positions at time (t), and (α) is the smoothing factor. By leveraging previous frame data, temporal smoothing helps maintain consistency across frames, minimizing the loss of detected features and improving the robustness of corner detection over time. This approach reduces the impact of noise or transient errors, ensuring that the feature points are more reliably tracked throughout the video sequence.

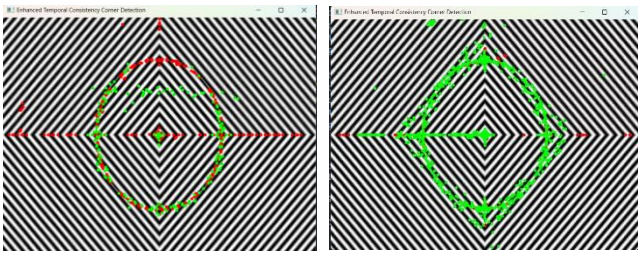

Fig 13 : The dots shows conventional Harris/FAST operator in consecutive frame 1 and 2 while green shows enhanced operator

As Figure 13 suggests, the red dots (conventional Harris/FAST) goes away on alternative frames while green one yet remains, though KNN messes up the cluster to distribute it inside the ring.

Now, Here is a comparison between both convetional and enhanced feature detectors

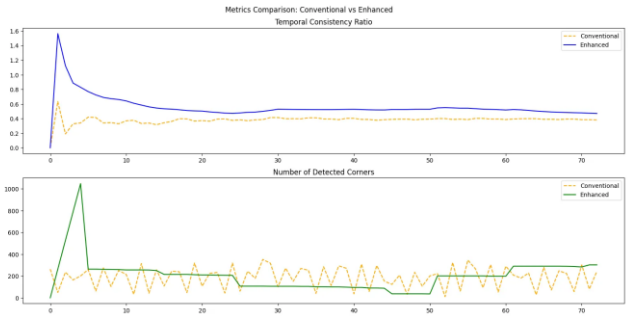

Fig 14 : Graph 1 showing consistency of the conventional and enhanced method
While Graph 2 shows number of detected corners

As Figure 14 indicates, the enhanced consistency is better than the conventional one and is more stable.
While, number of corners remains the same, even throughout the frame change. Although, The con here is, there is a significant amount of feature drop deetcted between frame 25 and 50 due to KNN issue but after that it again could be seen climbing back with stability.

VI. Comparison

Now, at this point, while keeping the conventional Harris and FAST as base, we can compare the all three models in three different studies –

Note: Data for this comparison is taken from the source papers [7] and [11]

Method	Average Precision Gain	Average Relative Recall	Average Relative F1 Score
Harris with B-Spline	+0.31	0.09	0.509
Harris with luvHarris	+0.16	0.14	0.661
Harris/FAST with Temporal Smoothing	+0.2198	0.8374	0.3390

Comparison table between performance of all 3 studies' analysis in this paper

Perhaps, There is an ask for improvement in this experiement as feature loss is significant besieds the stability. Yet, this could be a initiative for an improvement through this way.

VII. Conclusion

In conclusion, this paper demonstrates the utility and advantages of integrating temporal smoothing with the Harris and FAST feature detection algorithms for enhanced corner detection in real-time video analysis. The proposed temporal smoothing approach addresses key challenges posed by dynamic video sequences, such as feature loss and inconsistencies between frames, which are common with conventional methods when applied to video streams.

The study finds that incorporating temporal smoothing can improve the stability and accuracy of feature tracking by maintaining feature consistency across frames, even when individual frames contain occlusions, rapid motions, or lighting variations. This approach shows promising results in reducing the occurrence of feature dropouts and maintaining corner information over time, which is crucial for real-time applications that rely on robust and stable feature tracking, such as object recognition, tracking, and video-based motion analysis.

Through comparative analysis with advanced adaptations like the Harris with B-Spline and luvHarris algorithms, temporal smoothing demonstrates competitive advantages in maintaining feature stability while being computationally efficient. Although there is potential for further improvement, particularly in addressing specific issues related to feature drop and consistency under extreme dynamic conditions, the results indicate that temporal smoothing is a viable and effective enhancement for feature detectors used in dynamic video settings.

This research thus contributes a foundational approach to improving real-time video analysis, with implications for various computer vision applications that rely on robust feature detection under changing visual conditions. Future work could explore adaptive smoothing techniques that further optimize feature retention across highly variable video sequences and refine the balance between detection accuracy and processing efficiency.

Acknowledgment

We extend our sincere gratitude to our mentor, Dr. Usha Mittal, whose unwavering guidance, expertise, and encouragement were pivotal in the completion of this research. Her insightful feedback and commitment to excellence inspired us to approach this study with dedication and rigor. Additionally, we are profoundly grateful to Lovely Professional University for providing the infrastructure, resources, and a stimulating academic environment that facilitated our research. We acknowledge the support from the university's faculty and staff, whose contributions ensured a productive and enriching experience. This opportunity has not only enhanced our technical knowledge but has also strengthened our research and analytical skills, laying a foundation for our future academic and professional endeavors.

REFERENCES

[1] D. Alghurair, S. S. AI-Rawi, "Design of Sobel Operator using Field Programmable Gate Array," in Proc. International Conference on Technological Advances in Electrical, Electronics and Computer Engineering (TAEECE), pp. 589-594, May 2013.

[2] J. Yang and J. Guo, "Image Texture Feature Extraction Method Based on Regional Average Binary Gray Level Difference Co-occurrence Matrix," 2011 International Conference on Virtual Reality and Visualization, Beijing, China, 2011, pp. 239-242, doi: 10.1109/ICVRV.2011.20. keywords: {Feature extraction;Correlation;Entropy;Image texture;Joints;Statistical analysis;Brightness;texture feature;average binary gray level difference;regional co-occurrence matrix;texture feature parameters}.

[3] S. Guiming and S. Jidong, "Multi-Scale Harris Corner Detection Algorithm Based on Canny Edge-Detection," 2018 IEEE International Conference on Computer and Communication Engineering Technology (CCET), Beijing, China, 2018, pp. 305-309, doi: 10.1109/CCET.2018.8542206. keywords: {Corner detection;Filtering algorithms;Information filters;Image edge detection;Marine vehicles;Microsoft Windows;corner detection;canny algorithm;harris algorithm;mean-shift filter},

[4] Y. Biadgie and K. -A. Sohn, "Feature Detector Using Adaptive Accelerated Segment Test," 2014 International Conference on Information Science & Applications (ICISA), Seoul, Korea (South), 2014, pp. 1-4, doi: 10.1109/ICISA.2014.6847403. keywords: {Detectors;Feature extraction;Image segmentation;Life estimation;Computer vision;Image coding;Transform coding},

[5] L. Guo, J. Li, Y. Zhu and Z. Tang, "A novel Features from Accelerated Segment Test algorithm based on LBP on image matching," 2011 IEEE 3rd International Conference on Communication Software and Networks, Xi'an, China, 2011, pp. 355-358, doi: 10.1109/ICCSN.2011.6013732. keywords: {Feature extraction;Algorithm design and analysis;Lighting;Detectors;Image matching;Eigenvalues and eigenfunctions;Image segmentation;image matching;Local features;Local binary patterns (LBP);FAST (Features from Accelerated Segment Test);Feature description},

[6] Harris C and Satephens M J. A combined corner and edge detector. In Alvey Vision Conference, Manchester, 1988: 147-152.

[7] Liu Shuguang, Liu Mingyuan, He Yue, etc. On B-spline Wavalet Edge Detection Based on Canny Criteria [J]. Signal Processing, 2001, 17(5): 418-423.

[8] Peter I Rocket. Performance Assessment of Feature Detection Algorithms: A Methodology and Case Study on Corner Detectors [J].IEEE Tans. Image Processing, 2003

[9] V. Vasco, A. Glover, E. Mueggler, D. Scaramuzza, L. Natale, and C. Bartolozzi, “Fast event-based Harris corner detection exploiting the advantages of event-driven cameras,” in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst., 2016, pp. 4144–4149.

[10] E. Mueggler, C. Bartolozzi, and D. Scaramuzza, “Fast Event-based Corner Detection,” Brit. Mach. Vis. Conf., vol. 1, pp. 1–11, 2017.

[11] A. Glover, A. Dinale, L. D. S. Rosa, S. Bamford and C. Bartolozzi, "luvHarris: A Practical Corner Detector for Event-Cameras," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 44, no. 12, pp. 10087-10098, 1 Dec. 2022, doi: 10.1109/TPAMI.2021.3135635