Comparative Analysis of Harris Corner Detector and Fast Detector

Shawn Cameron
Dept. of Science
Toronto Metropolitan University
Toronto, Canada
shawn.cameron@torontomu.ca

***Abstract* —This report examines the speed and accuracy of the Harris Corner Detector and the Fast Feature Detector algorithms. This study involves the analysis on each of the algorithms' performance in corner detection, accuracy in detecting true corners, processing time, memory usage and effectiveness on noisy images and real world objects. The results have found that the Harris corner detector is more capable in correctly detecting corners within a smaller time frame, than the Fast feature detector. The Fast Feature detector may be much slower than the Harris detector, however it is more robust to noise. Real world factors both play a role in negatively affecting the algorithms' results. Overall, the Harris corner detector is a more ideal choice as it performs better in general, and its problems can be mitigated with image preprocessing.**

Source Code: https://github.com/Shawn-Cameron/CPS843-Comp-Vision/tree/main/FinalProject

I. Introduction

In the present programmers, people in business, and students must always decide which algorithms they must choose for their projects. Sometimes they may choose an algorithm with better computational complexity at the expense of quality and accuracy, or they may take a more complex, slower algorithm for better results. This project aims to make this decision easier by comparing the speed and accuracy of two corner detection algorithms, the Harris Corner Detector and the Fast Feature Detector. Both these algorithms are crucial in computer vision as they are widely used for feature detection. They are required for tasks like image recognition and object tracking. By learning which of these algorithms is better in speed and accuracy, we would be better equipped to decide which to use in a project.

II. Method

Images for this experiment will either be created or sourced from Google. The images will be analyzed to determine the number of corners, and then the number of correct corners the algorithm found will be used to determine its accuracy. The accuracy, processing time and resource consumption will be evaluated and compared for this experiment. The algorithms will be applied to the same set of images. The results will be compared to the accepted value to determine the accuracy, and they will also be compared with the other algorithms.

The analysis of the Harris Corner detector vs. the Fast Corner detector will come from running the experiment on different images. For the experiment, the detection accuracy and speed will be collected and compared, to determine which algorithm is more effective. To evaluate the memory usage and computation time, the memory_profiler and time python modules will be used.

III. Harris Corner Detection Algorithm

Harris corner detector is an effective method for identifying feature points in images. The algorithm measures the intensity change between one pixel, and the pixels in different directions [1]. This concept is known as the local auto-correlation. When the local auto-correlation is high, it indicates that there are intensity variations in multiple directions, which are determined to be corners [1].

The algorithm works by first computing the image derivatives or gradients, then optimizing the results using a 2x2 auto-correlation matrix, and finally using a corner response function to isolate the corners. Afterwards, thresholding and non-maximum suppression is then used to determine the corners. The image derivatives in both the x and y directions are calculated using the Sobel filter.

Sobel Gx

-1	0	1
-2	0	2
-1	0	1

Sobel Gy

1	2	1
0	0	0
-1	-2	-1

Fig. 1. Sobel Filters

The gradient derivatives are not sufficient to detect corners all corners for images with different problems. Including noisy images and alignment issues. The following function is used to reduce multiple errors that can occur in the algorithm [1].

$$E(x, y) = Ax^2 + 2Cxy + By^2 \quad (1)$$

where

$$A = dx2 \otimes w \quad (2)$$
$$B = dy2 \otimes w \quad (3)$$
$$C = (dx * dy) \otimes w \quad (4)$$

Applying this function to the image derivative allows the algorithm to detect corners with a sub-pixel accuracy. This function allows it to account for small shifts in an image that can occur when the image taken is not correctly aligned with the objects in the scene.

The resulting image from applying function (1) can be calculated using the equation.

$$E(x,y) = (x,y)\, M\, (x,y)^T \quad (5)$$

where M is the following 2x2 auto-correlation matrix

$$\boldsymbol{M} = \begin{bmatrix} \boldsymbol{A} & \boldsymbol{C} \\ \boldsymbol{C} & \boldsymbol{B} \end{bmatrix}$$

Afterwards, a Gaussian filter is used to reduce the amount of noise in the image, as the results of the previous operations can result in a noisy image [1].

Finally, the corner response function is used to determine the corners in the image. The response size is used to select isolated corner pixels and thin the edge pixels [1]. This function uses the trace and the determinant of M to compute the corner response R.

$$Tr(M) = A + B \quad (6)$$

$$Det(M) = AB - C^2 \quad (7)$$

$$R = Det(M) - k * Tr(M)^2 \quad (8)$$

Afterwards applying thresholding will remove lower levelled detected edges and non-maximum suppression will select the more relevant edges by removing duplicates or close edges that are not as strong [1].

IV. Fast Corner Detection Algorithm

The Fast corner detection algorithm is a simple corner detection algorithm that uses a circle of pixels to determine if a pixel is a corner or not. The algorithm works by taking the intensity of the current pixel p and the surrounding pixels. Pixel p is considered a corner if there are n pixels in the circle of 16, that are all brighter than the intensity of p plus a threshold or all darker than p minus a threshold [2]. More simply the intensity is calculated using the following equations.

$$I_i > I_p + t \quad (9)$$

$$I_i < I_p - t \quad (10)$$

The circle of the surrounding pixels is as follows.

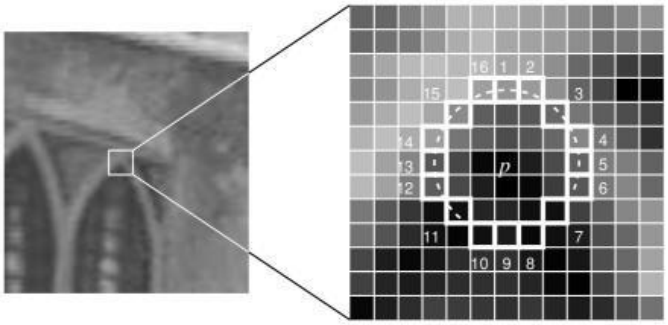


Fig. 2. Fast corner detection circle of used pixels

A faster variation of this algorithm only checks pixel numbers 1 and 9, then 5 and 13. If 3 out of the 4 pixels are all lighter or all darker, then p is a corner [2]. Analysis of this faster variation will not be included in this experiment due to the theoretical inaccuracy relative to the original version.

The variable n will be changed for the duration of this experiment to allow the detection of different types of corners that may not be detected with a set value. Only the results of the lowest value will be shown since the information from other values of n will either be redundant or cause the algorithm to fail to detect corners.

V. Experiment Figures

For this experiment multiple figures will be used for to analyses the effectiveness and results of the algorithms. To begin simple images will be used to determine the algorithms effectiveness of detecting simple corners.

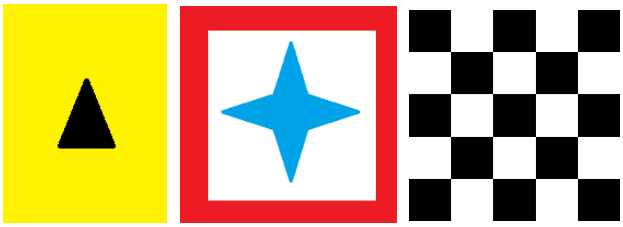


Fig.3. Triangle Fig.4. Star Fig. 5. Checkerboard

These images will test the algorithms effectiveness on corners that contain acute, obtuse or right angles and on different colours.

Afterwards real-world images will be used to test the corner detectors effectiveness on real world objects and scenes. These objects will test the algorithms when there is difference in lighting, or when the object is not directly facing the camera. Noise will also be added to fig. 4 to test the algorithms effectiveness on noisy images. The real-world images that will be used in the experiment are as follows.

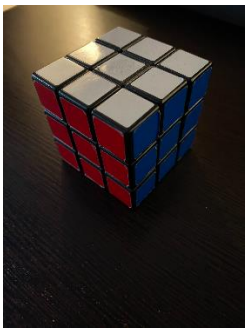


Fig.6. Rubik Cube

Fig. 7. Picture frame.

VI. Experiment Results

A. *Results of algorithms with fig 3*

TABLE I. RESULTS WITH FIG. 3

	Harris	Fast (n = 11)
Number of corners detected	7	4
Number of corners in image	3	3
Accuracy (correct detections)	100%	100%
% error	133.33%	33.33%
Speed(ms)	4	810
Memory Usage(Mib)	74.7	73.7

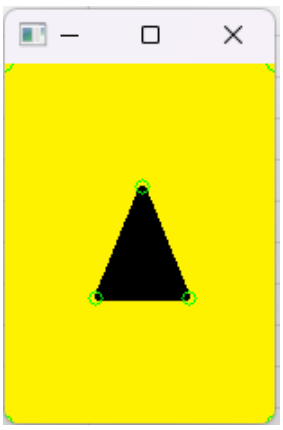


Fig.8. Harris results of fig.3

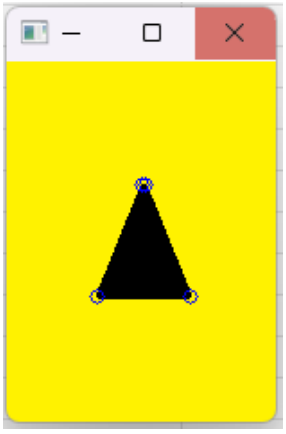


Fig.9. Fast results of fig.3

The image, Fig. 3 used for this round is a small image with the resolution of 150x200px. Its simplicity serves as a base line to analyze that the algorithms function correctly.

The results show that the Harris corner detection and the Fast corner detection algorithms were both able to successfully detect the corners in this run. The Harris detection seems to detect the corners of the full image and includes the points in the results. This is the result of using image derivatives in the computation. Each algorithm used a similar amount of memory to complete the task however the Fast detection algorithm took over 200x longer than the Harris algorithm.

B. *Results of algorithms with fig 4*

TABLE II. RESULTS WITH FIG. 4

	Harris	Fast (n = 9)
Number of corners detected	16	4
Number of corners in image	12	12
Accuracy (correct detections)	100%	100%
% error	33.33%	150%
Speed(ms)	6	2589
Memory Usage(Mib)	77.2	75.5

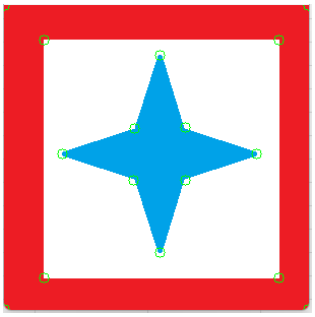


Fig.10. Harris results of fig.4

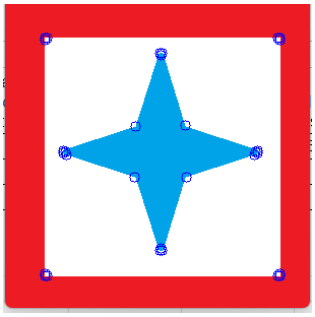


Fig.11. Fast results of fig.4

The image, Fig. 4 used for this round is a square image with the resolution of 300x300px. It was used to test the algorithms effectiveness on different angles of corners.

Both algorithms were effective at detecting the correct corners of the image. However, like the first image, the Fast algorithm took much longer, and it detects many extra corner points.

C. Results of algorithms with fig 5

TABLE III. RESULTS WITH FIG. 5

	Harris	Fast (n = 8)
Number of corners detected	80	144
Number of corners in image	80	80
Accuracy (correct detections)	100%	80%
% error	0%	80%
Speed(ms)	84	40057
Memory Usage(Mib)	133.5	131.9

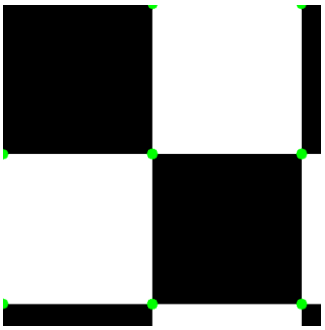


Fig.12. Harris results of fig.5

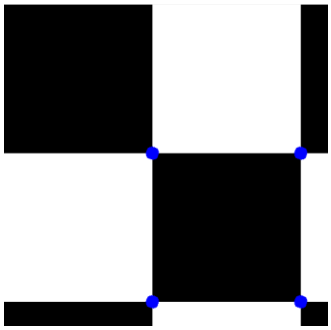


Fig.13. Fast results of fig.5

The image, Fig. 5 used for this round is a checkered patterned image with the resolution of 1200x1200px. For this image each box is considered to have 4 detectable corners for each square, except or the corners. This is because each square can be considered as a distinct object of which the corner detection algorithm can detect the 4 adjacent corner pixels.

The results show that the Harris corner detection algorithm was able to accurately detect all the corners in the image. Since the corners of the image were black, and the algorithm uses zero padding for its calculations, the 4 corners of the image were not detected as corners, which is expected. It was able to complete its task much faster than the Fast corner detector with more accuracy.

D. Results of algorithms with fig 6

TABLE IV. RESULTS WITH FIG. 5

	Harris	Fast (n = 12)
Number of corners detected	116	237
Speed(ms)	251	87832

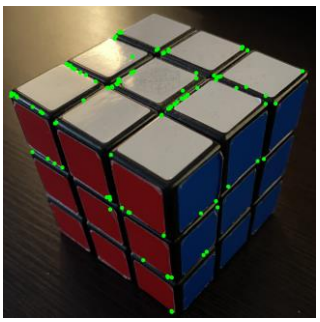


Fig.14. Harris results of fig.6

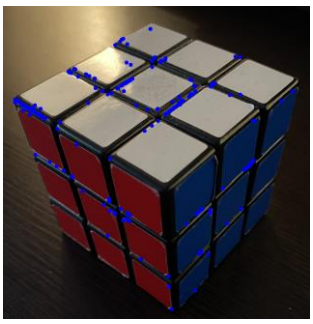


Fig.15. Fast results of fig.6

The image, Fig. 5 used for this round is a checkered patterned image with the resolution of 1536x2048px. Based on the results of the first real world image, it can be said that both algorithms struggled with images where there is less light where there is no well-defined corners. Each of the algorithms were able to detect some of the corners on the image, but they mostly detected edges. The key points that they detected were fairly like each other, which indicates that the algorithms are working as expected. These results can be explained by the nature of the object, the lighting and the resolution. The High resolution and the rounded corners on the image would have impacted the algorithms' ability to detect key points. This is because the low intensity level corner point would still be next to another point with a similar intensity. This indicates that lighting would play a key role in these types of images.

E. Results of algorithms with fig 7

TABLE V. RESULTS WITH FIG. 5

	Harris	Fast (n = 9)
Number of corners detected	706	1313
Speed(ms)	94	39982

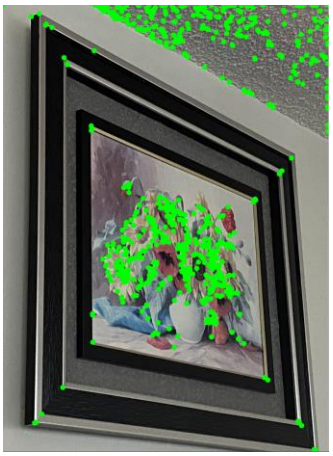


Fig.16. Harris results of fig.7

Fig.17. Fast results of fig.7

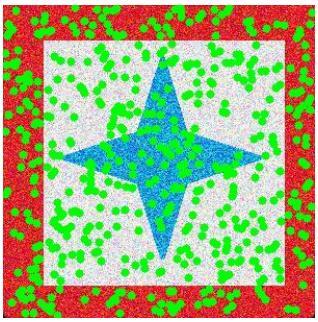
Both algorithms were able to detect some of the corners in the image, however they were not able to detect all the points in the image. This could indicate that other factors such as orientation, can greatly affect the algorithms. Despite

this limitation, the algorithms were able to detect many edges in the flower picture and was able to detect the lighter corners on the frame.

F. Results of algorithms with noisy fig 4

TABLE VI. RESULTS WITH FIG. 5

	Harris	Fast (n = 12)
Number of corners detected	519	66
Speed(ms)	6	2621



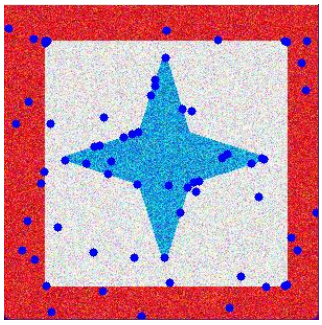


Fig.18. Harris results of noise Fig.19. Fast results of noise

Based on the results of the algorithms it appears that Fast feature was more capable in detects the corner points when there is noise in the image. The Harris algorithm was very susceptible to noise as it detected to many false feature points because of it. The Fast algorithm was able to detect nearly all the corner points in the image. This shows that the fast algorithm is more robust to noise and will be a superior choice on noisy images.

VII. Conclusion

The comparative analysis reveals that while both algorithms are capable of corner detection, they exhibit significant differences in speed and sensitivity to noise. The Harris Corner detector offers a much higher performance with better accuracy in both simulated and real-world images. It can accurately detect all the points in simulated images. The Fast Feature detector is at least 200x slower and is not as accurate as the Harris corner detector. However, it is more robust to noise, which makes it better when image processing must be done on noisy images. Despite the differences, both algorithms uses the same amount of memory. Based on the results of this experiment, it can be said that the Harris Corner detector is a superior algorithm to use in many situations.

REFERENCES

[1] C. Harris and M. Stephens. "A Combined Corner and Edge Detector," The Plessy Company plc, 1988.

[2] OpenCV Team. "FAST Feature Detector." OpenCV 4.x documentation, [Accessed: December 4, 2023]. Available: https://docs.opencv.org/4.x/df/d0c/tutorial_py_fast.html