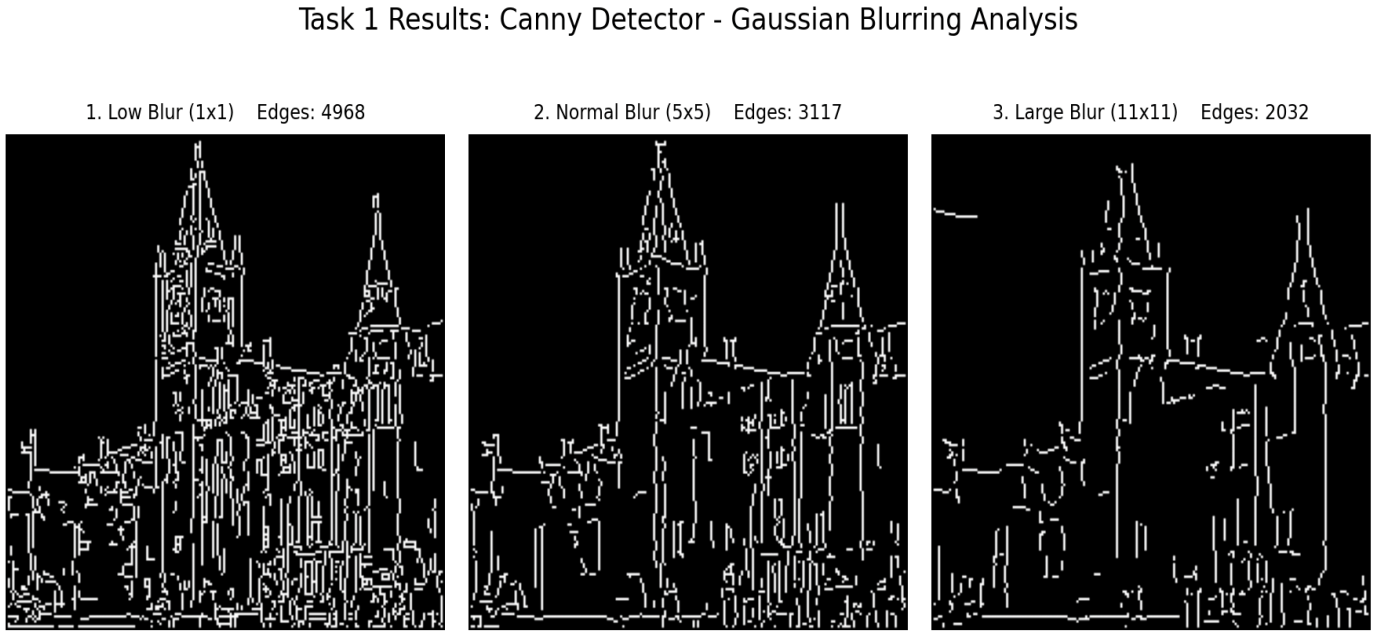
Task 1.2: Testing and visualising implemented results

We will be analysing the effects of the Canny Edge Detector's crucial parameters like Gaussian Smoothing (Blur) and Double Thresholding on the final output, runtime, and edge continuity, this section evaluates the robustness and integrity of the detector.

The Role of Gaussian Smoothing (Noise Reduction):

Before calculating the gradient, the first step of Gaussian smoothing is essential for reducing high-frequency noise. By adjusting the Gaussian kernel size, the test examples amply illustrate the trade-off between noise suppression and detail retention. The kernel sizes which were tested and visualised are (1,1), (5,5) and (11,11).



Observation table:

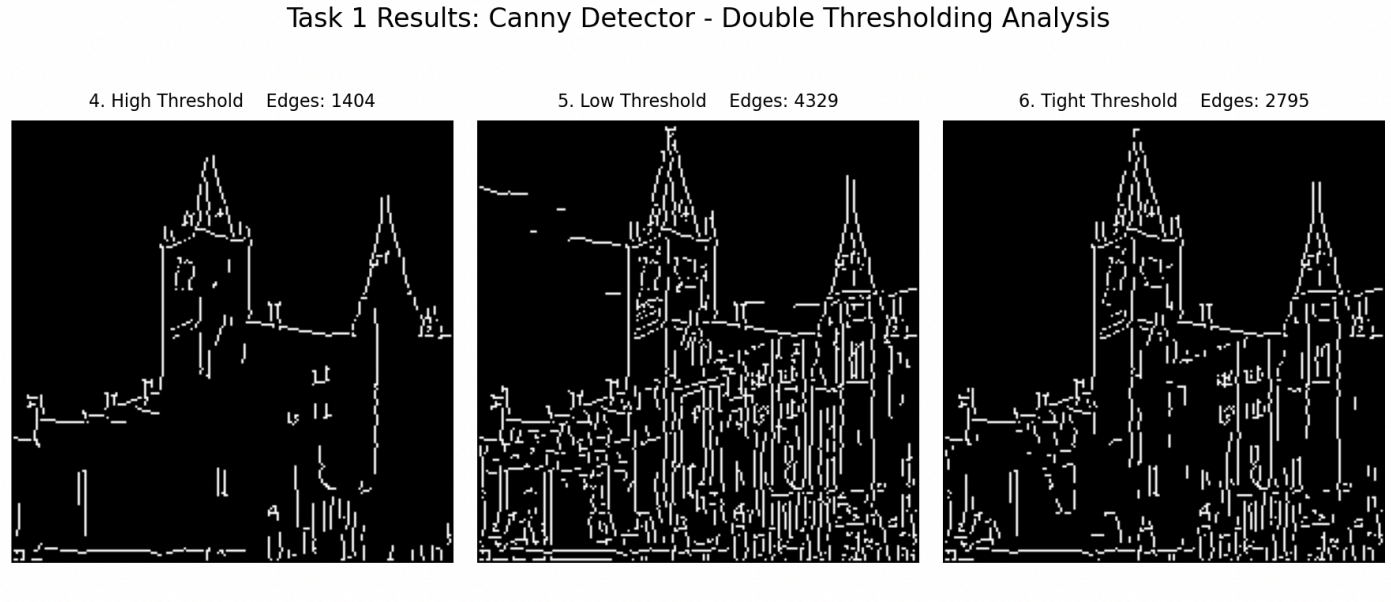
Test case	Kernel size	Number of edges	Observation
Low blur	(1,1)	4968	Highest level of detail, yet a busy and thick outline is produced because noise and little textures are mistaken for edges.
Normal blur	(5,5)	3117	Produces smooth, continuous primary edges by suppressing the majority of little noise while maintaining important structural details.
Large blur	(11,11)	2032	Completely eliminates fine details. Although it is highly successful against noise, it might result in discontinuity amongst object borders and significantly reduces real edges.

Summary:

The results show that gaussian smoothing is necessary to remove unnecessary noises from the image which otherwise generates false edges. By going from a (1,1) kernel size to (5,5) kernel size the number of edges decreases which confirms that a considerate amount of edges has been removed which would allow for the efficient functioning of Sobel and Non-maximum suppression processes.

The Role of Double thresholding and hysteresis:

Double Thresholding divides possible edges into Strong, Weak, and Suppressed categories after gradient computation and Non-Maximum Suppression (NMS). The Hysteresis process then determines the final edge. The (5, 5) Gaussian kernel size was used for all of the tests listed below.



Observation table:

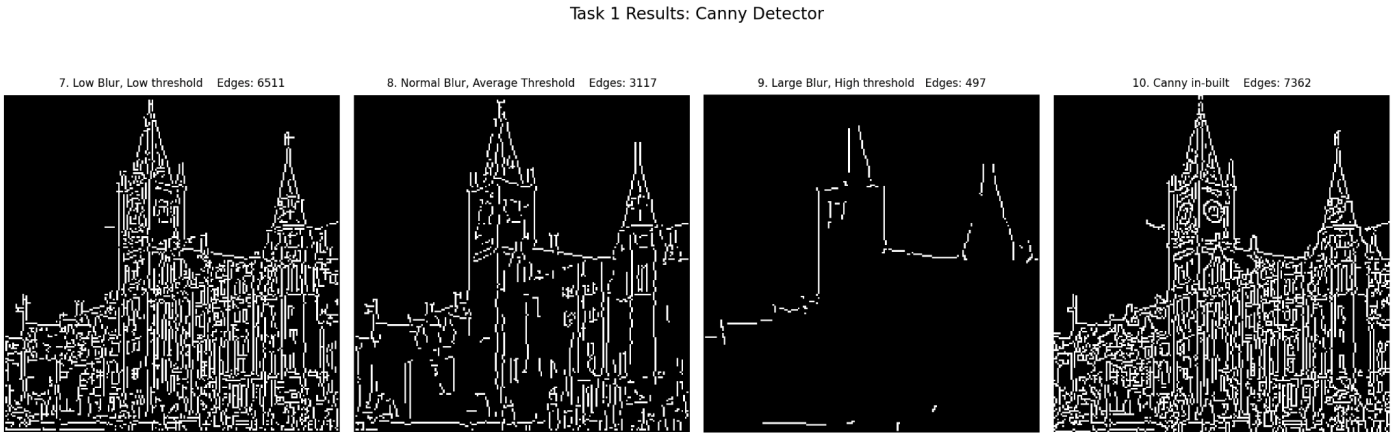
Threshold settings	High ratio	Low ratio	Number of edges	Observation
High threshold	0.30	0.10	1404	Fragmented edges: The only gradients that are kept are the steepest ones, such as the main building outlines. Many true edges are initially suppressed, resulting in broken contours and gaps.
Low threshold	0.10	0.01	4329	Noise inclusion: The low high ratio and lower low ratio allow weaker gradient paths to be linked, potentially including subtle texture changes or noise that survived the (5,5) blur, making the result thicker.
Tight threshold	0.15	0.12	2795	Discontinuity risk: The narrow band between the high and low ratio makes the outcome extremely delicate. Even if they should link to strong edges, many prospective weak edges fall into the suppressed zone, which causes contours to terminate too soon.

Summary:

In order to balance the trade-off between noise and fragmentation, the Double Thresholding and Hysteresis mechanism which uses lower low ratios and greater high ratios is essential. Adjacent weak edge segments are strategically linked by strong edges, which serve as anchors. This procedure guarantees maximal edge continuity and noise suppression, producing a final edge map that is clean and topologically accurate.

Task 1.3: Comparing custom canny detector implementation with OpenCV built-in canny function:

The differences between the built-in OpenCV's cv2.Canny() function and the custom Canny detector implementation are covered in this part along with a comparison of the numerical and visual findings and an analysis of performance (running time).



Observation table:

Test case	Kernel size	High ratio, Low ratio	Number of edges	Observation
Low blur, Low threshold	(1,1)	(0.10), (0.01)	6511	Most Detailed: Captures a lot of tiny, possibly noisy edges. In densely populated places, edges appear thick or double-lined.
Normal blur, Average threshold	(5,5)	(0.15), (0.05)	3117	Balanced: Well-balanced noise reduction and detail retention. Significant structural detail is still there, but the edges are cleaner than in Low blur, Low threshold test.
Large blur, high threshold	(15,15)	(0.40), (0.15)	497	Least Detail: A sparse, high-level structural outline is produced via extremely aggressive noise removal. Numerous important structural edges are overlooked.
Built-in Canny	(3,3) Sobel	Absolute values: (100), (40)	7362	Highly Detailed : Almost every discernible line is captured in this incredibly detailed output. Compared to the Low Blur, Low Threshold test, it seems even denser and possibly louder.

Quantitative Performance and Numerical Results:

The custom implementation's time varies between 0.037s and 0.050s, while the optimized OpenCV call is consistently faster at 0.0272s. There are notable variations in numerical output and performance between the OpenCV built-in function and the bespoke Canny implementation. The OpenCV result (7362 edges) outperformed even the bespoke "Low Blur, Low Threshold" scenario (6511 edges) in terms of sensitivity and edge count, indicating better algorithmic design for edge preservation. The OpenCV function employed absolute intensity values (100 / 40) in conjunction with a default (3*3) Sobel kernel, whereas the custom implementation used dynamic thresholds based on the maximum gradient magnitude (e.g., High Ratio 0.15 / Low Ratio 0.05). Importantly, the running time validates the benefit of library optimisation: the highly optimised, C/C++-backed OpenCV call was consistently faster at 0.0272s, achieving a 1.36 times speedup over the best custom run, while the custom implementation showed variable processing times between

0.037s and 0.050s. Although the custom implementation is functionally correct, this performance gap highlights the efficiency gains offered by the lower-level optimisation techniques, like SIMD instructions, used in professional libraries like OpenCV, making it the preferred choice for real-time applications.

Task 2.1: Feature Detector Comparison (SIFT vs. SURF vs. ORB)

Finding distinct, stable spots (keypoints) in an image and producing a compact signature (descriptor) for them that is resilient to geometric and photometric changes is the aim of feature identification.

1. Summary of Feature Detector Architectures

A. SIFT (Scale-Invariant Feature Transform)

SIFT is the foundational method, prioritizing **accuracy** and **robustness**.

Detection: By identifying the maxima and minima of the Difference of Gaussians (DoG) function at different scales, keypoints are localised in scale space.

Descriptor: Based on weighted gradient histograms, it generates a 128-dimensional floating-point vector that is extremely resilient to scale and rotation changes.

Matching: Makes use of the slower but more accurate L2 Norm (Euclidean distance) for comparison.

Key Takeaway: Although it is the slowest method computationally, it offers the most stability against significant viewpoint and scale changes.

B. SURF (Speeded-Up Robust Features)

SURF uses effective data structures and approximations to achieve speed.

Detection: Uses Box Filters to approximate the DoG, and Integral Images are used to compute the convolution results instantaneously. The determinant of the Hessian matrix in scale space is used to locate keypoints.

Descriptor: Produces a 64-dimensional (default) or 128-dimensional (extended) floating-point vector using Haar Wavelet responses in both horizontal and vertical directions.

Optimisation: By examining the upright flag, it provides Upright SURF (U-SURF). If this is the case, it expedites the process even more for non-rotational activities like panoramic stitching by omitting orientation calculation. Additionally, it improves speed by rapidly rejecting pairings during matching using the sign of the Laplacian (Trace of Hessian).

Key Takeaway: Considerably quicker than SIFT while retaining high accuracy; nonetheless, it still depends on L2 matching and floating-point descriptors.

C. ORB (Oriented FAST and Rotated BRIEF)

ORB is the modern, open-source solution optimized for **speed** and **real-time performance**.

Detection: Locates keypoints using the quick FAST (Features from Accelerated Segment Test) algorithm. The Intensity Centroid technique is then used to determine the local orientation in order to apply rotation invariance.

Descriptor: Employs Steered/Rotated BRIEF, a modified form of BRIEF. The 256-bit binary string is created by rotating the pixel-pair sampling pattern in accordance with the determined orientation of the keypoint.

Matching: Makes matching instantaneous by using the very effective Hamming distance (counting the number of distinct bits) rather than the L2 norm.

Key Insight: Extremely quick and unpatented, making it perfect for mobile and embedded applications, yet slightly less resilient to significant scale/viewpoint shifts than SIFT/SURF.

TABULATED DIFFERENCE

FEATURE	SIFT	SURF	ORB
Matching metric	L2 Norm	L2 Norm	Hamming Distance
Descriptor type	Float vector	Float vector	Binary string
Descriptor size	128 dim	64 dim (128 dim if extended)	256 bits
Detection method	DoG maxima/minima	Hessian matrix	Intensity Weighted Centroid
License	Patented	Patented	Open source
Speed	Slowest	Faster than SIFT	Fastest
Accuracy	Highest	High Accuracy with approximation	Least accurate
Number of keypoints detected	Least	More than SIFT	Most

Summary:

SIFT (Highest accuracy): Because it employs a complicated, high-dimensional floating-point descriptor (128-dim) that necessitates the resource-intensive L2 Norm for matching, SIFT is the slowest approach. On the other hand, this description offers the best stability against significant geometric changes.

ORB (Highest speed): Designed for real-time systems, ORB is the fastest algorithm. The compact 256-bit binary descriptor, which employs the computationally inexpensive hamming distance for instantaneous matching and the fast FAST detector are used to achieve its speed.

SURF (Intermediary): This intermediate method uses integral images to approximate Gaussian blurring, improving performance over SIFT while maintaining high accuracy using a floating-point descriptor.

In essence, the choice of detector is driven entirely by the application: **SIFT** for maximum quality, **ORB** for production-level speed and **SURF** for a balanced approach.

Task 2.2: Extracting ORB keypoints

Below attached image shows the implementation of extracting ORB keypoints as it correctly loads both the images, initialises the ORB detector, finds the keypoints of both the images and finally displays the result.

victoria1.jpg: ORB Keypoints

victoria2.jpg: ORB Keypoints

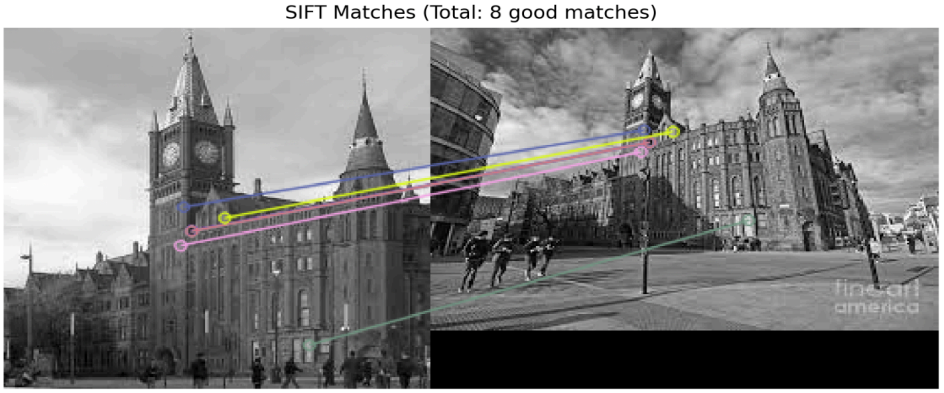
Task 2.3: SIFT vs. ORB Feature Matching Comparison

Implementation Summary:

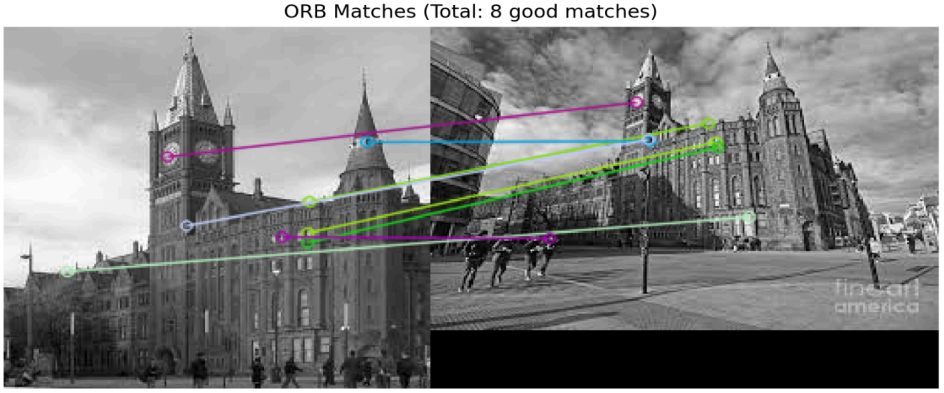
The performance of two well-known local feature descriptors—Scale-Invariant Feature Transform (SIFT) and Orientated FAST and Rotated BRIEF (ORB)—in matching keypoints between the two supplied images (victoria.jpg and victoria2.jpg) is compared in this experiment.

Task 2.3: SIFT vs. ORB Feature Matching Comparison(Nfeatures = default, ratio threshold = 0.75):

SIFT Matches (Total: 8 good matches)



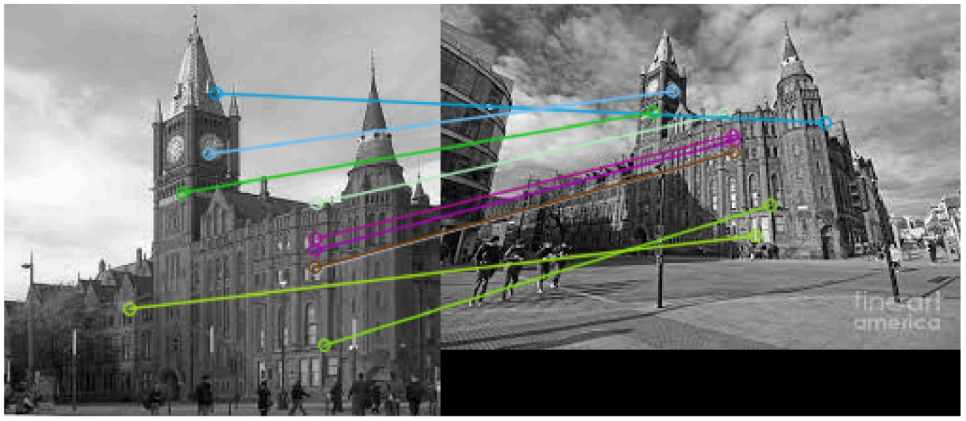
ORB Matches (Total: 8 good matches)



Task 2.3: SIFT vs. ORB Feature Matching Comparison(Nfeatures = 2000, ratio threshold = 0.75):

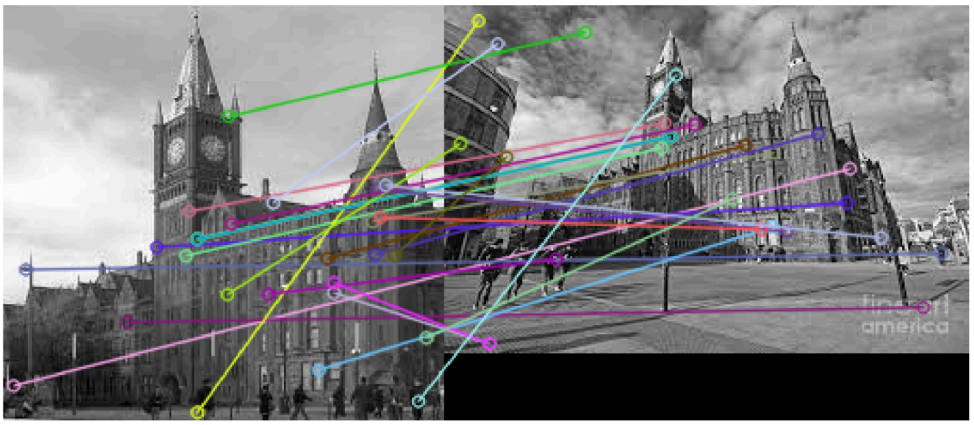
SIFT Matches (Total: 8 good matches)

ORB Matches (Total: 10 good matches)

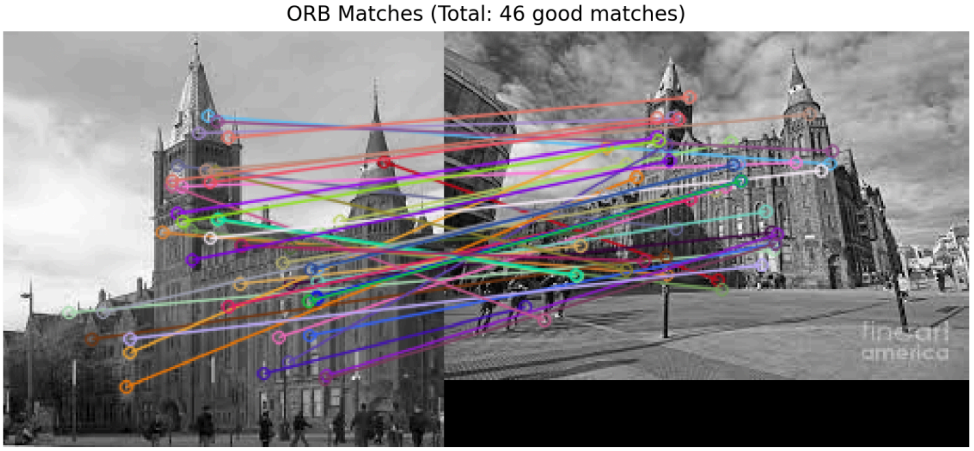


Task 2.3: SIFT vs. ORB Feature Matching Comparison(Nfeatures = 5000, ratio threshold = 0.85):

SIFT Matches (Total: 28 good matches)



ORB Matches (Total: 46 good matches)



Observation table:

N_features	Ratio threshold	Total SIFT Matches	Total ORB Matches	Observation
Default (500)	0.75	8	8	Low match count due to strict ratio test and small keypoint pool.
2000	0.75	8	10	Although expanding the pool somewhat benefits ORB but 0.75 is still too stringent for SIFT.
2000	0.80	13	23	More matches become available as the ratio test is loosened, also ORB's advantage emerges.
5000	0.85	28	46	By accepting a looser match quality and optimising the keypoint pool, ORB's superior match quantity is reinforced, resulting in maximum performance.

This table indicates that ORB consistently discovers a bigger amount of matches than SIFT across optimised parameters and clearly illustrates the crucial trade-off between quantity (nfeatures) and quality (ratio threshold).

Efficiency and Density of ORB Descriptors:

Massive Descriptor Generation: The binary descriptors from ORB are incredibly light and quick to extract. Even with SIFT's complexity, ORB can produce a denser, more useful set of points in the two photos when (nfeatures) is set to 5000.

Optimal for Hamming Distance: The loose ratio test and ORB's binary descriptors work very well together. When the image modification is small, as it is between victoria.jpg and victoria2.jpg, the Hamming distance works exceptionally well in identifying a high number of trustworthy correspondences. It is also computationally very straightforward.

Robustness and Limitation of SIFT Descriptors:

SIFT's Conservatism: SIFT's 128-dimensional floating-point descriptor is more conservative by nature. SIFT still requires a certain degree of uniqueness in the gradient orientations even after the ratio test was relaxed to 0.85. The algorithm's final match count is lower than ORB's because it finds it difficult to identify 5000 genuinely unique and stable keypoints across all scale levels in a building with repeating features.

Computational Cost: SIFT's extraction and matching procedure (using L2-Norm or Euclidean distance) is significantly slower than ORB's (using Hamming distance), even with a large match count.

Conclusion:

When ORB (Orientated FAST and Rotated BRIEF) and SIFT (Scale-Invariant Feature Transform) feature descriptors are compared, it becomes clear that ORB is the better option for this particular feature-matching job, particularly when density is optimised. ORB consistently produced a higher number of matches (46 excellent matches compared to SIFT's 28) by adjusting the number of features to 5000 and relaxing the

ratio threshold to 0.85. This shows that ORB can extract a denser, more useful set of keypoints even with an enlarged keypoint pool size. Essentially, ORB has a substantial computational advantage over SIFT's 128-dimensional floating-point descriptors since it uses effective binary descriptors and the Hamming distance for matching. For common feature-matching applications where image variations are small, ORB is the superior, highly efficient option because it is significantly faster, highly scalable, and offers excellent, reliable correspondence, while SIFT maintains its theoretical advantage in extreme robustness to large scale and rotation changes.

Thank You….