Abubaker Shabbir, Idris Dodwell, Wong Shean
EPM02 Object Detection Via OPENCV on Python Coursework 2

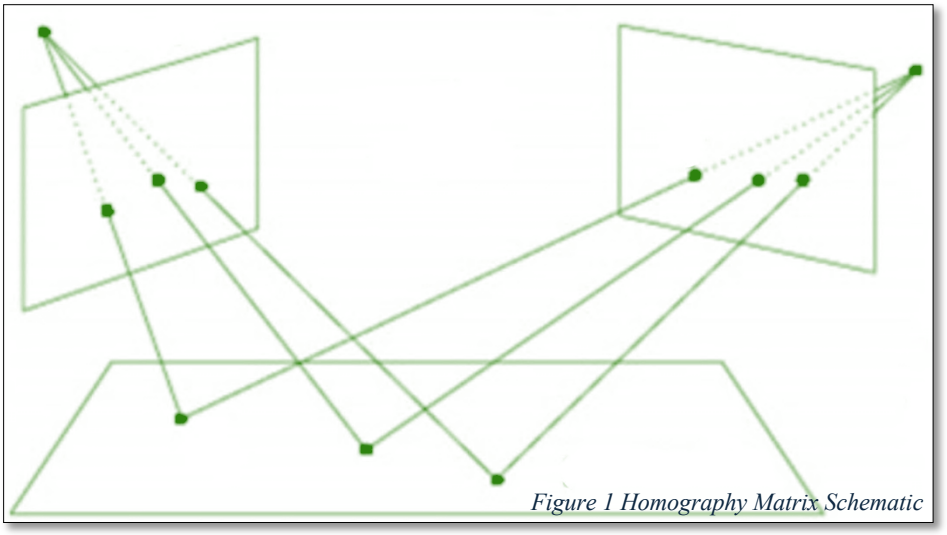


Figure 1 Homography Matrix Schematic

Nomenclature

RANSAC –
A robust algorithm for estimating a model (e.g., homography) while rejecting outliers.

cv.findHomography() –
OpenCV function to compute the homography matrix from feature matches.

cv.perspectiveTransform() –
OpenCV function to apply the homography matrix to transform points or bounding boxes.

Bounding Box (BB) –
A rectangular box that defines the spatial extent of an object in the image.

SIFT –
Feature detection & description algorithm that is robust to scaling & rotation.

ORB –
An efficient feature detection & description algorithm was used in OpenCV.

Descriptors –
Numerical representations of key points for matching purposes.

Training Image –
The image containing the known objects to detect in query images.

Query Image –
The image in which the objects from the training images are to be detected.

Outlier Rejection –
Process of discarding mismatched points during feature matching.

Warp Perspective –
Transforming an image or points using a homography matrix.

Planar Assumption -
The assumption that objects are approximately flat, allows for homography-based transformations.

H –
Homography matrix

x, y –
Coordinates of a point in the image plane

x', y' –
Transformed coordinates of a point after applying the homography

h_{ij} –
Elements of the homography matrix H

Feature Point –
Distinctive pixel/region in the image, used for correspondence matching

Keypoints –
Detected points of interest in an image via detection algorithms

Abstract

Our project centres on object detection utilising OpenCV and Python. Object detection is a critical domain within computer vision and image processing, concentrating on the identification of objects or instances belonging to specified categories (such as humans, animals, or flowers) in digital photos and videos. This domain is utilised in several applications, including surveillance & security, video conferencing, automated industries, drones, and navigation. The major objective of this project is to precisely identify an object within an image, encircle it with a bounding box utilising homography, and subsequently categorise it using Python and object-orientated programming principles. Computers analyse images differently than humans, rendering object detection particularly hard due to large variations in elements such as the item's size, orientation, position, and perspective within the image, which can alter the image data.

Introduction

The paper's project aimed to employ The paper's project aimed to employ feature-matching algorithms to compute homography for object detection. We achieved this by using a bounding box and a label. The first section will explain the coding principles, the initial problems, and how HSV yielded better trial results. It will then introduce SIFT and ORB key point/feature detection, explain their functions, and explain why the project chose SIFT over ORB. We will compare feature matches and explore the rationale behind the use of FLANN. Finally, moving on to RANSAC and PROSAC, homography, and the final results and conclusion.

The project started off by importing OpenCV in Python and loading the training images that had the object of interest in them. Next, we used an image-matching algorithm to extract the features through key point detection and descriptor computation. We then applied a bounding box to these areas of interest, ensuring the algorithm only focused on these features and not the background features. Following this, we created a mask for the training image using the bounding box coordinates.

The system has successfully detected the right object in the image.

After finding matches in the query image, we compute a homography to map the bounding box of the object in the training image to the object in the query image. This process locates the objects in the query image. We then visualise the matches by drawing lines between the corresponding images and the bounding box, which we also make visible, before displaying the results.

Python served as the foundation for the code, utilising the principles of object-orientated programming (OOP). OOP presents itself as a reliable way to maintain large code bases (Chekakta 2024); in this project, it allowed great readability and locality of behaviour (Gross 2020). The code grew larger than anticipated, and the implementation of separate classes for each element of the project's workflow (e.g., detector, matcher, etc.) facilitated the easy modification of individual aspects or parameters within a single section. Each class had its own operation; for example, the SIFT Detector class detects key points and describes their features, which keeps the code clean, readable, and maintainable.

HSV/HSI -

The problem with the typical red, green, and blue channels is that they have relatively high differences (Zhang et al. 2012); differences in exposure or lightening may give the same object different RGB (or BGR) numbers in different photos (Yun et al. 2022). This can lead to issues with object detection; one solution is to utilise Hue, Saturation, and Value (or Intensity) HSV/HSI, which enhances the robustness of colour detection (Zhang et al. 2012). In the project itself, this did create better matches and a homography matrix/bounding box. Figure 2 allows us to visualise the HSV/HSI colour spectrum, where hues are represented as colours, saturation as colour intensity, and value or luminosity as black-to-white intensity.

The hue below Fig. 2. demonstrates the 2-D circumference of the cone, while the saturation refers to the area of the 2-D area on the cone, with the outer edges having the highest saturation and the centre point having the lowest. The value, or intensity, is on the black-to-white scale. HSV (2022).

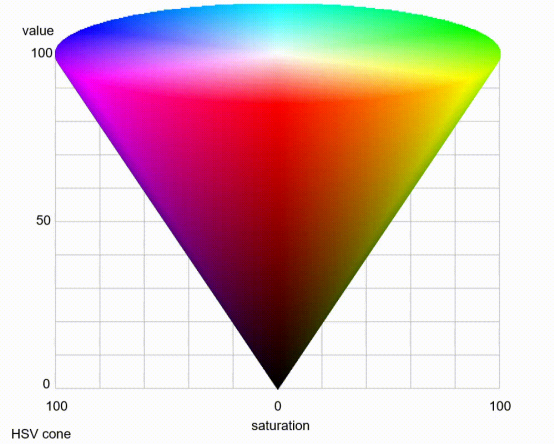


Figure 2 The colour spectrum in HSV/HSI.

RANSAC -

Promising 2023 employs RANSAC, a stochastic sampling method, to eliminate outliers and enhance the precision of the homography matrix. It formulates a hypothesis based on random samples and validates it against data (Choi et al. 2009). Fischler & Bolles (1981) introduced it into the image processing domain as a substitute for the least squares method. From the data, it randomly selects a subset and the minimum number of required points, which in this case is 2. Subsequently, it enumerates the inliers within a specified threshold (delta or e), associated with the inlier noise statistics, across multiple iterations (n), with a designated probability (p) of identifying the greatest number of inliers (Argyros 2024). Figure 1 illustrates this equation, allowing us to calculate the required number of iterations (n = log(1-p)/log(1-(1-e)s). Figure 2 shows a visualisation of one step of the process; the minimum sampled points are in red, and the inliers from this sample are in green. The process then repeats this step n times, yielding the optimal model as an output. RANSAC is efficient and resilient, lacking complex algorithms or significant memory needs because it only uses a portion of the data (Choi et al. 2009). A limitation of RANSAC is that it does not model local matching. We may regard it as a black box, producing provisional correspondences (Chum & Matas 2005). In robotics, RANSAC may also make it harder to process real-time data because of the randomness of its features, which is linked to how precise it is (Jeon et al. 2015).

PROSAC –

Progressive Sample Consensus enhances RANSAC by utilising ranking correspondences. The process initiates with high-confidence matches and gradually incorporates lower-confidence ones, thereby enhancing computational efficiency and augmenting model correctness (Chum & Matas, 2005). The study by Chum & Matas (2005) demonstrated a two-fold enhancement in addressing non-trivial problems alongside equivalent worst-case performance between RANSAC & PROSAC.

Jeon et al. (2015) advocated it as a robust alternative to RANSAC for robotics applications. Therefore, Prosac (RHO in OpenCV) was included in the project. We utilised the distance feature (Fig.3a) to sort the matches, thereby facilitating the discernment required by Prosac. Figure 3. The code snippet illustrates the sorting of matches following the Lowe ratio. Sorting allows Prosac to sort via ranking. This did indeed result in a more accurate homography.

```
# Sort matches by distance
good_matches = sorted(good_matches, key=lambda x: x.distance)
```

Figure 3 Sampling in RANSAC. Argyros(2024)

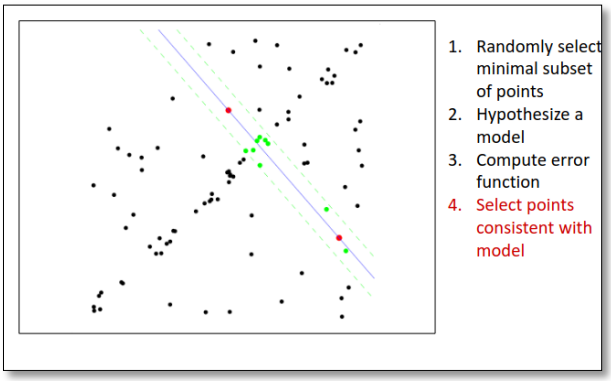


Figure 4a RANSAC SAMPLING

$$1 - (1 - (1 - e)^s)^N = p$$

e – probability that a point is an outlier.

s – number of sample points

N – number of samples

p – desired probability

Figure 3b Ransac equation. Argyros (2024)

Abubaker Shabbir, Idris Dodwell, Wong Shean
EPM02 Object Detection Via OPENCV on Python Coursework 2

Homography

Homography is a projective transformation that connects two planes. This connection is crucial in computer vision as it allows for the analysis of the same scene from multiple perspectives. Numerous applications extensively utilise these techniques. Homography operates in homogeneous coordinates, transforming 2D points into 3D space for computing efficiency (DeTone et al. 2016). In the following example matrix, we can see that there are 8 points (the last being a fixed unity value of 1). Within the code, we have the following values ($\mathbf{h_{ij}}$)**:**

```
[[ 8.41488838e-01  9.47421253e-01 -1.21818230e+02]
 [ 2.20294312e-01 -3.97849604e-02  5.35928101e+02]
 [ 1.71033433e-03 -2.56609987e-04  1.00000000e+00]]
```

There are eight unknowns, which means we need four inlier pairs to compute the matrix, as each of them gives two equations (Zoghlami et al. 1997). Using sampled points, we need a minimum of 4 points to compute a homography matrix. - (x & x' are corresponding points) (3x3 transformation matrix). Once we obtain the homography matrix, we use it to convert the coordinate system of one image to another (Premsingh, 2023). The project utilised this to calculate the transform for the bounding box on the query image.

$$s\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \mathbf{H}\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Figure 5 H-matrix from Gava & Bleser (2017)

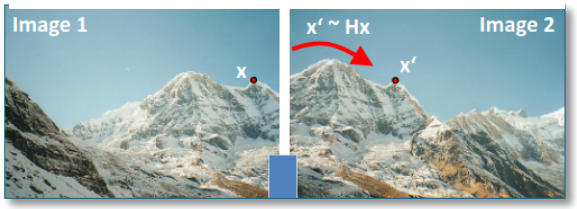


Figure 6 H-matrix applied to 2 images. From Gava & Bleser (2017).

Methods Implemented

The project's codebase consists of two primary modules: the hub code and the polygon coordinate extractor.

(1) The hub code serves as the principal control mechanism, enabling the selection of feature detection techniques (SIFT, ORB) and feature matching methods (FLANN, Brute-Force). This modular design facilitates adaptation & expansion, allowing for the incorporation of additional analytical methods while preserving simplicity.

- (2) The polygon coordinate extractor functions as an initial module, which extracts and processes object coordinates for subsequent input into matching detectors.

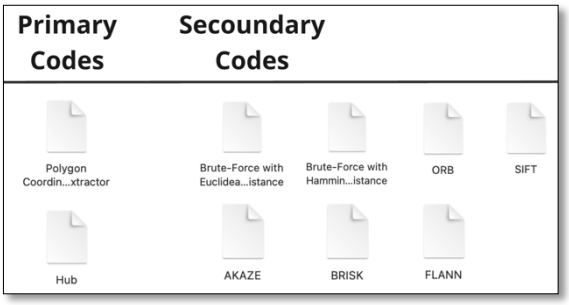


Figure 7 Coding Structure

(1) The hub code serves as the principal control mechanism, enabling the selection of feature detection techniques (SIFT, ORB) and feature matching methods (FLANN, Brute-Force). This modular design facilitates adaptation & expansion, allowing for the incorporation of additional analytical methods while preserving simplicity.

- (2) The polygon coordinate extractor functions as an initial module, which extracts and processes object coordinates for subsequent input into matching detectors.

The following methods were chosen to be implemented for feature detection & description, due to the major comparisons which can be established:

1) *SIFT*

2) *ORB*

Alongside these methods, here are the corresponding *feature-matching algorithms:*

1) *SIFT – FLANN*

2) *ORB - Brute-Force with Hamming Distance*

Feature detection methods such as SIFT and ORB are approaches employed to locate unique key points and their associated descriptors inside an image.

These key points denote visually distinct regions (such as corners or edges) that remain unchanged despite transformations (including rotation, scaling, or variations in illumination). The descriptions encapsulate the visual characteristics of the area surrounding each key point, facilitating effective picture comparison.

Feature matching is the procedure of identifying corresponding key points between two images (e.g., a query image & a training image) according to their descriptors. Methods such as brute-force matching (utilising Euclidean or Hamming distances) and FLANN employ these descriptors to calculate similarity scores by aligning features between images. This alignment facilitates object localisation, homography estimation, and subsequent transformations for applications such as object identification and tracking.

The Polygon Extractor

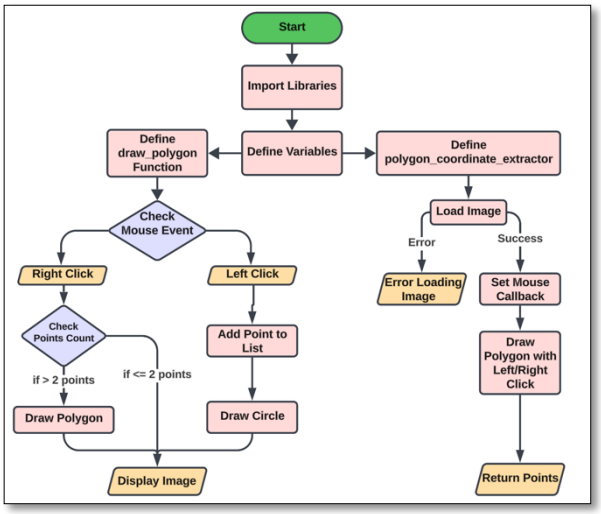


Figure 7 The Polygon Extractor Flowchart

The polygon coordinate extractor is essential for the manual identification and extraction of item boundaries in the training image. Utilising this tool, users engage with the image to accurately delineate the areas of interest by clicking along the edges of each object. This project will utilise the tool to delineate polygonal borders for the four objects in the training image: the controller, tablets, AirPods, & currency.
Upon completion of the polygon rendering for these items, the tool generates the coordinates of the polygon vertices. We will subsequently process the coordinates to create bounding boxes for each object, represented by four principal corner points. We will use the bounding box coordinates as inputs for feature detection & matching techniques, such as SIFT, ORB, BRISK, & AKAZE, as demonstrated in the feature detection sections.

By applying a mask around the bounding box, each feature detection approach uses the bounding box to focus on the region of interest, extracting only the object's features and ignoring extraneous portions of the image. We will compare the derived features with those from query photos to facilitate precise identification and localisation of objects within the query images. This method guarantees accurate feature matching by focusing only on pertinent object regions in the training image, as articulated by Lowe (2004) in the SIFT framework for object recognition and matching applications.

FLANN

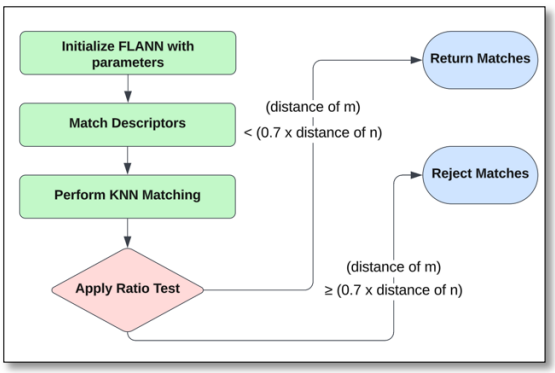


Figure 8a The Flann Flowchart

Designed to find the nearest neighbours between feature descriptors extracted from images, the FLANN matcher is a robust and efficient algorithm. It is particularly well-suited for high-dimensional data & works via approximating matches instead of searching for exact matches, thereby significantly improving computational efficiency. FLANN uses indexing structures, such as KD-Trees, for organising the data points and performing approximate searches to identify the closest

descriptors in terms of similarity. (Muja, M. & Lowe, D.G., 2009. FLANN)

This project employs the FLANN matcher in conjunction with the SIFT feature detection method to match feature descriptors between the training and query images. The matching process incorporates the Lowe's ratio test to eliminate unreliable matches, guaranteeing the utilisation of only dependable matches for subsequent processing, such as homography computation.

This technique is considered essential for accurately mapping the bounding boxes of objects in the training image to their respective locations in the query images. By leveraging FLANN, we can efficiently identify key matches that contribute to precise object detection for the four objects: the controller, tablets, AirPods, & the coin, even in high-dimensional descriptor spaces. FLANN Flowchart Figure 4 illustrates the process of the FLANN matcher, a crucial component in feature matching for object detection in this project. The process starts by initialising FLANN using appropriate indexing parameters, like KD-Trees, optimised for efficient searches in high-dimensional spaces. Descriptors from the training and query images are subjected to K-Nearest Neighbours (KNN) matching, identifying the two nearest matches for each descriptor. A critical step in this workflow is Lowe's ratio test, which discards unreliable matches by evaluating the distance of the nearest match relative to the second nearest. Only matches that pass the ratio test are retained for subsequent processing, including homography computation. Figure 4 illustrates the exclusion of rejected matches, which guarantees the contribution of only reliable correspondences to accurate object mapping. Lowe, D. G. (2004)

This method plays a fundamental role in detecting the four objects—the controller, tablets, AirPods, and coins—by mapping their bounding boxes from the training image to the query image.

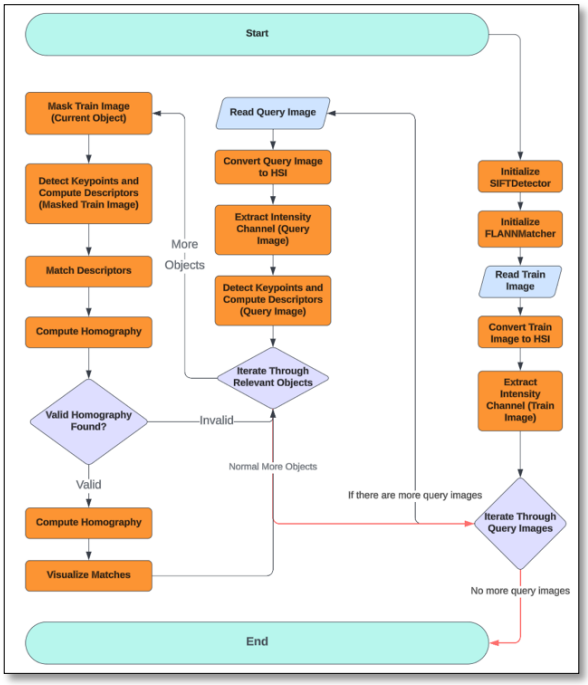


Figure 9 Sift Flowchart

SIFT

SIFT is a robust method for feature recognition and description aimed at identifying and representing critical points in images that remain invariant to scale, rotation, and illumination variations. SIFT operates by identifying stable regions, such as edges & corners, across various scales & produces distinctive descriptors for these regions that facilitate comparison between pictures. This project employs SIFT to identify four distinct objects (Controller, Tablets, AirPods, & Coin) in the training image & to locate them in the query images.

The essential elements of the workflow include initialising the detector (SIFT) and matcher (FLANN), followed by HSI preprocessing that strengthens feature detection by separating the intensity channel. This preprocessing step alleviates the impact of colour and illumination discrepancies. We extract descriptors from the masked regions of the training image for each object and compare them with descriptors from the query images. The FLANN matcher utilises Lowe's ratio test to verify the dependability of matched features.

The procedure also involves the homography

computation for valid correspondences, which is employed to project each object's bounding box from the training image onto the query images. This iterative method guarantees precise location and display of objects like the controller, tablets, AirPods, and coins, even across diverse query images and situations. As shown above, the flowchart successfully emphasises the algorithm's iterative characteristics and its proficiency in managing several objects and images well.

Brute-Force with Hamming Distance

Brute force with Hamming distance juxtaposes each descriptor from the training image with every descriptor from the query image. The Hamming distance, which quantifies the number of different bits across binary descriptors, serves as the similarity metric. This method is especially appropriate for binary feature descriptors, such as those produced by ORB detectors. Although it is known for its computationally demanding nature, it guarantees precise matches by conducting a comprehensive search for the nearest correspondences. (Skanti Hamming Brute Force) This project will utilise brute force with Hamming distance to match feature descriptors, hoping to produce reliable object identifications across diverse situations.

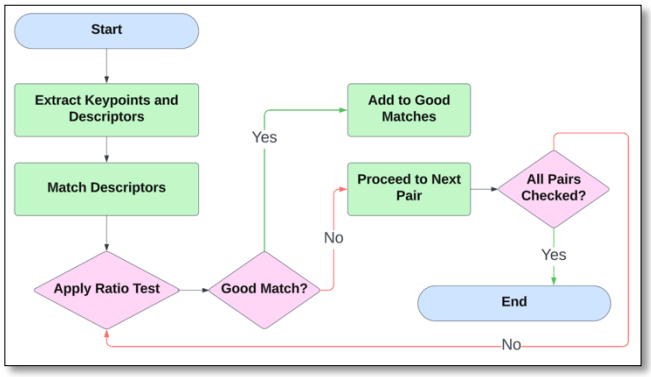
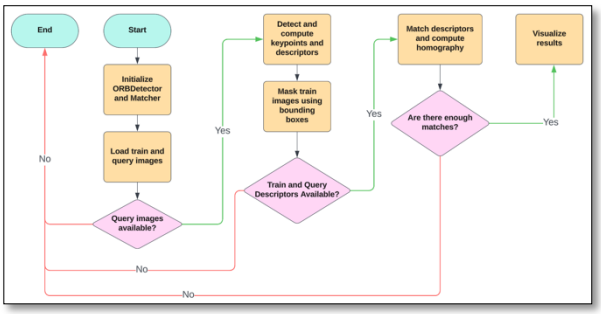


Figure 10 Brute-Force With Hamming Distance

Figure 10 illustrates the brute-force matching procedure with the Hamming distance. This method initiates the extraction of key points and binary descriptors from the training and query images. We evaluate the descriptors using Hamming distance to quantify bitwise discrepancies. We subject matches to a ratio test to eliminate faulty correspondences, ensuring that only high-quality matches contribute to object localisation. We continue iterations until we evaluate every descriptor pair, which produces reliable matches necessary for accurate homography estimates and object detection. The diagram below illustrates the operational framework of ORB in this program. After initialising the ORB detector and matcher, the training and query images undergo processing to extract descriptors and key points. We obscure the training image with bounding boxes, focusing on specific elements. The ratio test determines valid matches and then uses them to calculate homography, which enables precise item location and visualisation. This organised approach guarantees effective and accurate feature matching for all inquiries.

ORB

Orientated Fast Robust Brief, a feature detector developed by Lowe(2001). Uses a combination of a Fast Keypoint detector with a centroid for orientation and a modified Brief, a binary feature detector, that can deal with perspective changes (Lowe 2001).



Sift Result

This section presents the outcomes of the SIFT feature detection and matching method. We implement the approach using a single training image that includes all four items (AirPods, tablet, coin, and controller) and test it on three query images of increasing complexity.

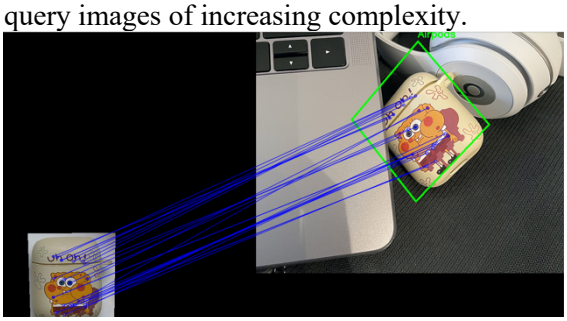


Figure 11. Query 1 - Sift Result

Query 1, the most elementary scenario, comprises solely the AirPods, facilitating uncomplicated identification and matching. Ultimately, Query 3 poses the most difficult situation, incorporating all four elements from the training image. This investigation aims to illustrate the robustness and precision of the SIFT approach in recognising and localising items across different levels of difficulty in the query photos.

Examination of Inquiry 1: AirPods Recognition

The findings for Query 1 (Figure 12) underscore SIFT's efficacy in detecting and localising the AirPods case within a straightforward query image featuring little distractions. The system effectively identified distinct characteristics on the AirPods case, correlated them with the training image, and rendered an exact bounding box on the query image by homography transformation. These findings confirm SIFT's dependability in processing texture-dense, feature-laden objects.

Principal Insights:

• High Accuracy: The blue lines in Figure 18 depict precise feature correspondences between the training and query photos, and the green bounding box is impeccably aligned with the AirPods case. This accuracy arises from SIFT's capacity to extract and match unique textures and edges, including cartoon patterns and case outlines.

• Preprocessing Enhancements: HSI preprocessing, which isolates the intensity channel, alleviated the impact of light variability, hence enhancing feature recognition. Masking the training image further removed extraneous background elements, enabling the system to concentrate solely on the AirPods.

Query 2 provides an added layer of complexity with the incorporation of the table.
Robustness: Lowe's ratio test guaranteed that only dependable matches were included in the homography computation, facilitating precise bounding box projection and accurate object localisation.

Constraints:

• Query Simplicity: Although Query 1 presented an optimal situation with minimal complexity, SIFT's dependence on unique key points may encounter difficulties in more congested or obstructed settings.

Examination of Query 2—Correlation between Tablets and AirPods

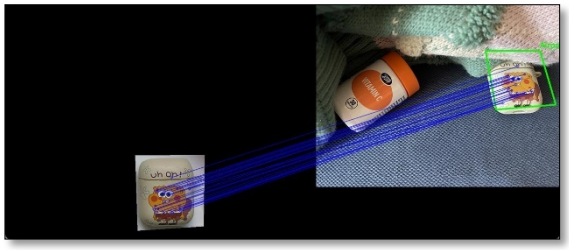


Figure 12. Query 2 - Sift Result Airpods

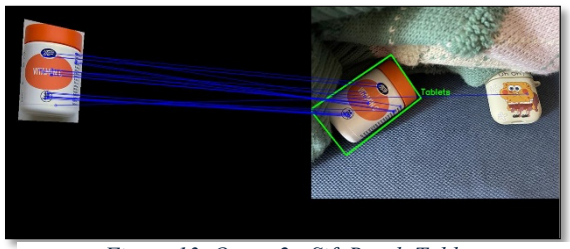


Figure 13. Query 2 - Sift Result Tablets

Tablet Correspondence (Figure 13);
The identification and localisation of the tablets in Query 2 underscore SIFT's proficiency in managing items with generally homogenous textures and few distinguishing characteristics. Notwithstanding the subtle distractions posed by the soft fabric background, the system effectively identified elements from the training image and precisely drew the bounding box for the tablets.

Key observations:

Feature Precision: The green bounding box precisely corresponds with the tablets, with distinct feature correspondences indicated by the blue lines, confirming SIFT's capacity to emphasise nuanced characteristics like edges and printed text. The soft fabric background caused slight feature discrepancies; however, preprocessing techniques such as masking and intensity channel extraction alleviated possible influence.

AirPods Synchronization (Figure 14)

In Query 2, SIFT demonstrated strong performance by accurately recognising critical spots on the AirPods case and effectively projecting a precise bounding box in the query image. The results are analogous to Query 1, indicating consistency in identifying texture-dense objects.

Key observations:

• Consistency in Detection: SIFT reliably matched the AirPods even with the presence of an additional object (tablets) in the scene,

underscoring its dependability in somewhat complicated environments. The bounding box projection was seamless, so confirming the efficacy of Lowe's ratio test and outlier rejection in achieving precise matches.
• The findings for Query 2 highlight SIFT's efficacy in managing moderate complexity through precise detection and localisation of items. While the added distractions slightly increased the chances of mismatches, preprocessing and feature-matching improvements ensured reliable detection of both the tablets and AirPods. This highlights the algorithm's ability to handle environments with multiple objects effectively.

Examination of Inquiry 3 - Identification of all four entities

Query 3 posed a difficult situation with all four items (AirPods, Coin, Tablets, and Controller) situated in a disordered setting. Figures 15–18 illustrate the advantages and disadvantages of SIFT in intricate scenarios.

Figure 15, Query 3 - Airpods Detection

AirPods Detection (Figure 15):
• ***Strength***: Precise feature correspondence, evidenced by the thick and accurately aligned blue lines. The bounding box was accurately delineated. The unique textures and cartoon designs on the AirPods case enabled accurate key point identification.
• ***Flaw***: Minor discrepancies arose from background clutter; however, preprocessing effectively reduced the majority of noise.

Coin Detection (Figure 16):

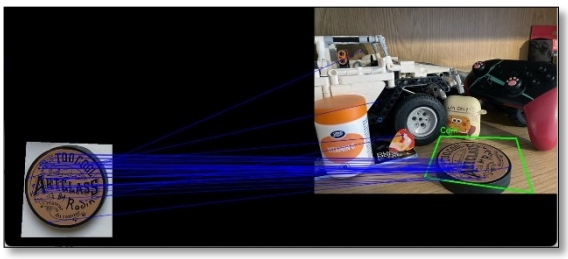


Figure 16. Query 3 - Coin Detection

• **Strength**: Elevated feature density on the coin facilitated precise alignment, resulting in the bounding box closely conforming to the item. This is due to the coin's complex texture, resulting in numerous characteristics for comparison.
• **Flaw**: Certain discrepancies were noted because of the coin's analogous hues with surrounding items, such as the wooden backdrop.

Tablets Detection (Figure 17):

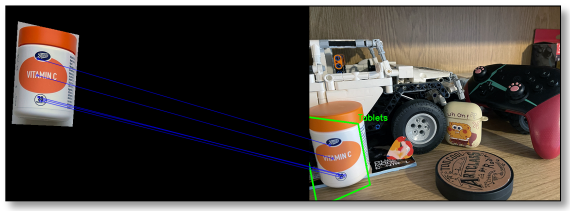


Figure 17. Query 3 - Tablets Detection

• **Strength**: Accurate bounding box placement with minor feature discrepancies, demonstrating resilience in multi-object detection; preprocessing with HSI segregated the intensity channel, hence augmenting feature clarity.
• **Flaw**: Minor discrepancies in features arose from the resemblance in shapes and colours of adjacent objects.

• ***Controller Detection (Figure 18):***

Figure 13. Query 3 - Sift Controller Detection

• **Strength**: The program effectively identified the controller, although it's substantial dimensions and intricate texture demonstrated SIFT's robustness against differences in scale and texture; the controller's different buttons and edges provide unique key points for alignment.
• **Flaw:** Overlapping objects resulted in a slight rise in mismatches; however, Lowe's ratio test efficiently eliminated faulty matches.

ORB Results

The procedure remains identical to ORB, except for the segment when ORB is utilised in place of SIFT for feature recognition. The blue lines denote the matches identified between the left training image and the right query image. Each line links a key point identified by the algorithm as common between the two photos. The green bounding box in the query image represents the object of interest that has been successfully spotted, obtained from the homography matrix calculated using matched key points.

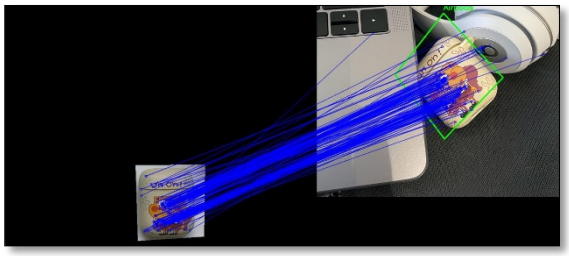


Figure 14. Query 1 - Airpods Matching

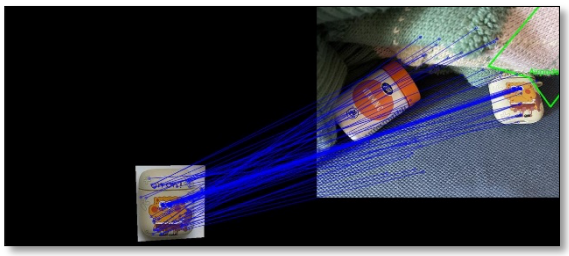


Figure 15, Query 2 - Airpods Matching

In Query 1, figure 19 demonstrates that ORB successfully matched the bulk of critical points immediately, with minimal incorrect matches, due to the limited number of items in the query image. In the latter image, it is evident that as the complexity of the query image increases, the results become increasingly erroneous due to the characteristics of ORB. While we experience a higher incidence of matching mistakes compared to SIFT, the outcome remains rather acceptable.

Figure 20 has more objects in the query image, and hence, it is noticeable that the result has more complications during matching common key points and hence, more errors are seen here. Some prominent features were matched correctly, and a lot of the key points were mismatched.

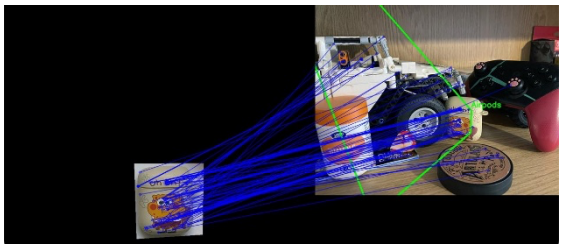


Figure 16. Query 3 - Airpods Matching

As the analysis moves to Figure 21, it is noticeable that the complexity increases, and so do the errors, this, compared to the results archived using SIFT, shows that orb is less accurate when there is a higher difficulty and a more complex query image.

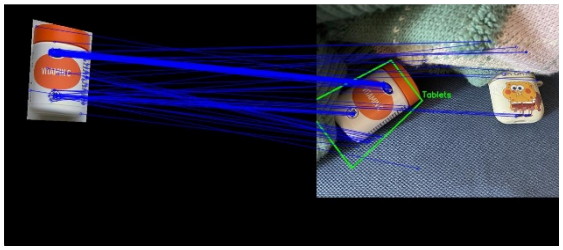
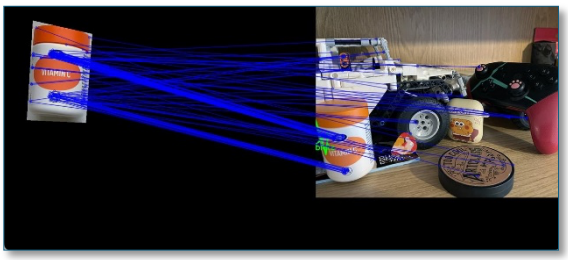


Figure 17. Query 2 - Tablets Matching



The figure 18, Query 3 - Tablets Matching

In Query 2, figure 22, it can be observed that the query image is quite simple, and hence quite clean and accurate results are achieved with a few mismatched key points, but this changes as it proceeds to Figure 23, wherein the query image is more complex with more angles and different objects with different point of view and hence the algorithm is working harder to get accurate results. Eventually, it comes up with a result that is quite usable but has a lot more errors in it as well, since there are more objects in the query image, it has more variety of shapes and colours, which means an increased probability of miss-matched key points, which is reflected in this result as well.

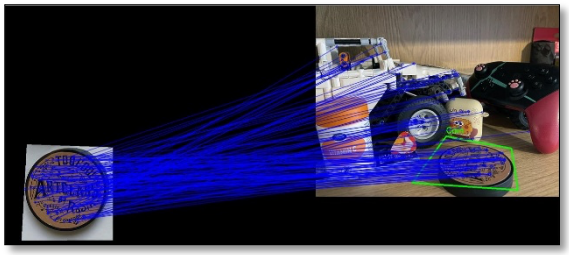


Figure 19. Query 3 - Coin Matching

In Query 3, the query image is quite complex, and the algorithm struggles with the right feature matching It does get the important features matched but also gets quite a lot of mismatched key points. This makes it evident that ORB is meant for scenarios where accuracy is not the key factor and where a simpler query image will be used.

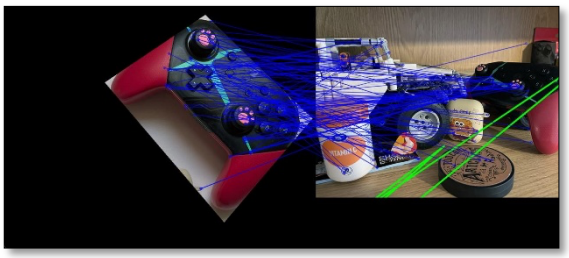


Figure 20. Query 3 - Controller Matching

In Query 4, the limitation of ORB is prominent since it really struggles to keep up and get accurate results due to the training image, as well as the query image being quite complex, due to this, it is visible that the majority of the key point matches are miss matched and it was only able to match a few prominent key points right. This enforces the fact that ORB is not ideal for situations where accuracy is a needed factor.

Figure 26 shows that while S is more spurious (e.g. detecting non-interesting features on the table) but more evenly spread out, resulting in better matches and so better homography, which the results agree with.

Conclusion

Initial tests focused on ORB vs SIFT, optimising & fixing the values. The parameters were derived from a combination of trial & error & from the official OpenCV documentation (OpenCV). Looking at the matches from Figures 5 & 6, SIFT gives a better good-to-bad match ratio compared to ORB, but ORB's features were more accurate & focused, while SIFT's were more spurious, picking up non-relevant features with low contrast on the table. Although the SIFT matches appear more spread across the object, potentially explaining the better ratio. The initial tests showed that SIFT was a prima facie better option.

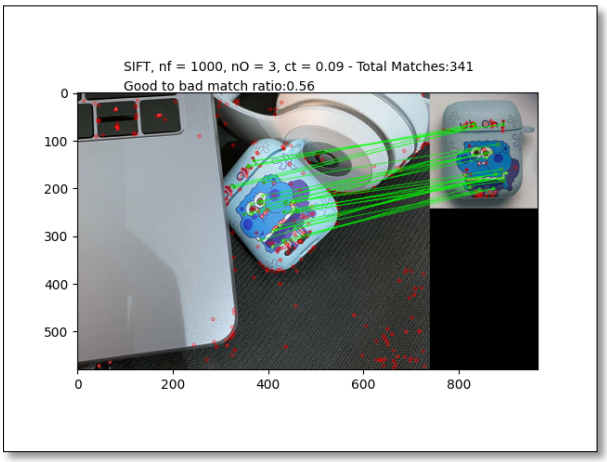


Figure 26 SIFT Parameters. A better Good to Bad match ratio of 0.56, compared to Orb's by 0.51.

This experiment evaluated the efficacy of SIFT and ORB feature identification and matching techniques using three query photos of varying complexity. The objective was to assess their robustness and precision in identifying and locating four specific objects (AirPods, tablets, coins, and controllers) among differing degrees of clutter and complexity. SIFT exhibited exceptional accuracy and resilience, especially in convoluted situations, owing to its capacity to identify high-dimensional, distinguishing features, such as complex textures and edges. Its dependence on Lowe's ratio test and outlier elimination guaranteed accurate homography and object localisation. Moreover, preprocessing techniques such as HSI conversion and masking enhanced feature clarity and diminished noise, hence augmenting its performance. Nonetheless, SIFT's computing demands and

intermittent difficulties with background noise, particularly in densely populated environments, underscored its constraints. Conversely, ORB, with its binary descriptor matching using Hamming distance, provided expedited processing and adequate outcomes in uncomplicated situations, as demonstrated by Query 1. To be fair, ORB's accuracy decreased as environments got more complicated. This was because it couldn't handle cluttered or overlapping objects well, and its feature representation was limited to two dimensions. Consequently, ORB is appropriate for real-time applications where rapidity is essential; however, it is less suitable for tasks requiring high precision. The findings indicate that SIFT excels in intricate, feature-dense contexts necessitating high precision, but ORB is preferable in situations prioritising computational economy when slight mistakes are acceptable.

One problem was that the code used for detecting the objects applied to each one equally, which is not ideal. In the future, improvements would include better use of OOP principles such as inheritance. Each object to be detected would be its separate child class and inherit from a parent class (i.e. Object Detector would be the parent and Controller would be the child). This would make the code more maintainable by separating each object's ideal parameters. For example, by merging all the classes into one and altering the constructor function, if a particular parameter is better suited for detecting one object over the other (like features or contrast threshold), the child class can make use of that without affecting the matches of another, thereby optimising object detecting.

Aspect	**SIFT**	**ORB**
Feature Descriptor	Float-based, high-dimensional	Binary, low-dimensional
Accuracy	High, especially in complex scenario	Moderate, drops significantly in complexity
Performance in Query 1	Excellent, precise matches and bounding boxes	Good, minimal mismatches
Performance in Query 2	Accurate with moderate complexity	Struggles with added objects and angles
Performance in Query 3	Robust but minor errors in cluttered settings	High errors due to increased complexity
Processing Speed	Slower due to float-based computations	Faster due to binary matching
Preprocessing Impact	Beneficial, enhances feature clarity and accuracy	Limited, preprocessing had minimal effect
Ideal Use Case	Accurate object detection in complex environments	Real-time detection in simpler scenarios

GitHub Link:

https://github.com/idrisdodwell00711/EPM_102_Project_2/tree/main/idris_contribs

References

[1] Introduction OpenCV. Available at: https://docs.opencv.org/3.4/d9/dab/tutorial_homography.html (Accessed: 05 December 2024).

[2] Nam, D.,Aouf, N. (2017) "Automated mosaicing for improving vehicle situational awareness in real time," 2017 IEEE Intelligent Vehicles Symposium (IV), Los Angeles, CA, USA,pp. 1146-1151.

[3] Choi, S. Kim, T. Yu., W. (2009) Performance Evaluation of RANSAC Family. BMVA. Paper 355.

[4] Jeon, K., H. Jeong, M., J. Lee, Y., K. 2015 An implementation of the real-time panoramic image stitching using ORB & PROSAC. International SoC Design Conference (ISOCC), Gyeongju, Korea (South), 2015, pp. 91-92.

[5] Argyros, A. (2024) Slides from Lecture 16. Online resource: http://users.ics.forth.gr/~argyros/cs472_spring24/16_CV_RANSAC_LMEDS_2903.pdf Accessed on: 20/11/2024.

[6] Primsingh, P., (2023) Image Stitching using OpenCV — A Step-by-Step Tutorial. Online resource: https://medium.com/@paulsonpremsingh7/image-stitching-using-opencv-a-step-by-step-tutorial-9214aa4255ec. Accessed on: 12/11/2024

[7] Zoghlami, I. Faugeras, O. Deriche, R, (1997) Using geometric corners to build a 2D mosaic from a set of images, in Proc. EEE Computer

[8] Society Conference on Computer Vision & Pattern Recognition, SanJuan, Puerto Rico, USA. pp. 420-425

[9] Gava, C. Bleser, G. (2017) 2D projective transformations (homographies). Online resource: https://web.archive.org/web/20171226115739/https://ags.cs.uni-kl.de/fileadmin/inf_ags/3dcv-ws11-12/3DCV_WS11-12_lec04.pdf Accessed on: 08/11/2024

[10] Datahacker. (2020) How to create a panorama image using OpenCV with Python. Online resource: https://datahacker.rs/005-how-to-create-a-panorama-image-using-opencv-with-python/ Accessed on: 15/11/2024

[11] Fischler, A., M.,Bolles, C., R. 1981. R&om sample consensus: a paradigm for model fitting with applications to image analysis & automated cartography. Commun. ACM 24, 6 (June 1981), 381–395.

[12] Chum, O., Matas, J. (2005). Matching with PROSAC - progressive sample consensus. Proc. of CVPR. 1. 220 - 226 vol. 1.

[13] Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision, 60(2), 91–110. https://en.wikipedia.org/wiki/Scale-invariant_feature_transform

[14] Muja, M. & Lowe, D.G., 2009. FLANN - Fast Library for Approximate Nearest Neighbors. Available at: https://www.cs.ubc.ca/research/flann/

[15] Chekakta, Z. (2024). Lecture 6: Intro to OOP [Lecture]. EPM102: Engineering Programming. City, London University. November 24.

[16] Gross, C. (2020). Locality of Behaviour (LOB). Online resource: https://htmx.org/essays/locality-of-behaviour/ Accessed on: 7/12/2024

[17] Zhang, Z.B., Liu, Z., Song, Y.S., Wang, H.Y., Tang, G.J., 2012. A License Plate Recognition System Based on HSV Space in Natural Lighting. AMR 590, 421–426

[18] Yun, J. Park, S. Yoo, S. (2022). Infusion-Net: Inter- and Intra-Weighted Cross-Fusion Network for Multispectral Object Detection. Mathematics. 10. 3966. 10.3390/math10213966.

[19] HSV Color Space. (2022) Online Resource:

https://www.psy.ritsumei.ac.jp/akitaoka/HSV_color_space.html Accessed on: 08/12/2024

[20] Skanti Skanti/HammingBruteForce: A Fast 64-bit brute-force method to match 256-bit binary descriptors by the closest Haming distance, GitHub. Available at: https://github.com/skanti/HammingBruteForce