Feature Matching Comparison with Limited Computing Power Device for Autonomous Driving

Xu Du
School of Computing
Montclair State University
Montclair, USA
dux3@montclair.edu

Weitian Wang*
School of Computing
Montclair State University
Montclair, USA
wangw@montclair.edu

***Abstract*—This research-to-practice full paper describes a study on the performance of feature-matching algorithms in constrained computational environments, focusing on autonomous driving using low-end hardware like the Raspberry Pi 4B. We evaluate algorithms such as ORB, AKAZE, BRISK, and SIFT, examining their efficiency, accuracy, and robustness under various conditions. While ORB offers speed, AKAZE and BRISK demonstrate more consistent performance. To mitigate the gap between theoretical analysis and practical application, we integrate these findings into a robotics course through a project-based learning (PBL) approach. The comparison analysis provides the instructor with the necessary insights to guide students, as the research setting closely mirrors the course project. This hands-on project not only deepens students' understanding of computer vision but also hones critical problem-solving skills essential for modern engineering challenges. Future work will extend this study to other single-board computers and explore advanced computational techniques like parallel computing and GPU acceleration.**

***Keywords*—Autonomous driving, feature matching, project-based learning, engineering education, computing education**

I. Introduction

Feature matching stands as a fundamental and essential element in the realm of computer vision. It equips computers with the capability to interpret and analyze visual information extracted from images or videos. This process is not merely about recognizing images; it is about understanding and pinpointing corresponding points across multiple images that depict the same portion of an object or scene. Such a capability is crucial in many applications, ranging from simple image recognition to complex real-world scenario analyses [1].

Project-based learning (PBL) is an educational approach in which students actively learn from real-world problems and challenges [9]. This method aligns with constructivist theories of education, which assert that learners construct knowledge via experiences and reflection. PBL encourages students to engage in hands-on projects that require them to apply what they have learned in practical, meaningful ways.

At Montclair State University, a robotics course designed for undergraduate and graduate students integrates PBL principles with advanced topics in robotics, including feature matching. The course provides students with a comprehensive learning experience in robotics, focusing on sensors, actuators, vision systems, and robotics programming. Through this course, students work on practical projects that deepen their understanding of computer vision and develop essential skills such as problem-solving, collaboration, and project management. One of the critical components of the course is a comprehensive project in which students work in teams to develop an autonomous mobile robot (AMR) using race cars (RC), Raspberry Pi 4, USB cameras, and Arduino MEGA2560 boards. This project is a capstone experience, challenging students to implement feature-matching algorithms on resource-constrained hardware to develop an autonomous vehicle capable of object recognition and human-following tasks. This hands-on experience simulates real-world autonomous driving scenarios, allowing students to apply theoretical knowledge to practical problems.

In feature matching, a "feature" refers to an image's specific, distinctive part that remains easily recognizable across various conditions, such as changes in viewpoint, scale, or lighting. These features play a critical role in numerous computer vision tasks, providing a means to effectively understand and compare different images.

A. Types of Features

Corners: Corners are points in an image where two edges meet, characterized by a change in intensity in two directions. These highly distinctive points maintain consistency across image transformations like rotation or scaling. The stability and distinctiveness of corners make them reliable anchor points in feature-matching [2].

Edges: Edges are defined by a sharp change in brightness and typically indicate the boundaries of objects within an image. They represent significant structural properties of objects, making them integral to feature matching. The primary goal of edge detection algorithms is to identify these vital points accurately [3].

Blobs: Blobs are regions in an image with unique properties, such as color or texture, distinct from surrounding areas. They are often used to represent and identify objects within an image. Blob detection methods are focused on effectively identifying these unique regions [4].

The selection of features is based on their distinctiveness and invariance to common transformations, such as scale, rotation, and perspective changes. This invariance is crucial in feature matching as it ensures that a feature remains recognizable under various conditions. The feature-matching process involves detecting these features in different images and establishing correspondences between them, where the effectiveness and accuracy are significantly influenced by the choice of features and the detection and description methods.

B. Feature Detection and Feature Description

Feature matching in computer vision has two core stages, including feature detection and feature description [5]. Feature detection concerns identifying distinctive points or 'features' within an image. These features are specific parts of the image that stand out due to their unique attributes, such as texture, color, edges, or corners. The primary goal of

feature detection is to locate these points consistently and reliably across different images. Various algorithms are designed to detect these salient features efficiently. The choice of features is critical and largely depends on their robustness to lighting, orientation, and scale changes. Following the detection stage, each feature undergoes a description process. This involves creating a 'descriptor', essentially a unique fingerprint for each feature. This descriptor encapsulates the key attributes of the feature in a compact format, typically represented as a vector of numbers. The process of feature description is crucial as it ensures that the same feature can be recognized across different images, despite variations in angle, scale, or illumination. The essence of feature matching lies in utilizing these descriptors to establish correspondence between features in different images. By comparing these descriptors, the algorithm identifies which features in one image correspond to features in another, thereby recognizing similar objects or scenes across various images.

C. Common Applications of Feature Matching

Feature matching is a pivotal technique in computer vision with a wide array of applications across diverse fields. Object recognition plays a crucial role in identifying specific objects within images or videos, finding uses in security for facial recognition systems, retail for product identification, and manufacturing for quality control. In the domain of 3D reconstruction, feature matching enables the creation of three-dimensional models of scenes or objects from multiple images taken at different angles, a technique extensively used in fields like archaeology, architecture, and computer-aided design (CAD) [6]. In the context of robotics and autonomous driving, it facilitates navigation and interaction with the environment by recognizing objects, obstacles, and pathways. In medical imaging, feature matching is instrumental in comparing and analyzing medical images over time, thus aiding in detecting and monitoring diseases [7]. Additionally, feature matching for geospatial analysis and remote sensing is crucial in aligning and comparing satellite imagery and aerial photography for environmental monitoring, urban planning, and disaster response [8]. These applications highlight the versatility of feature matching and underscore its critical role in advancing a broad spectrum of industries and research domains.

D. Project-Based Learning Theories and Practices

This work also implements PBL by integrating feature-matching techniques and autonomous driving simulations into the curriculum. Students develop a profound understanding of computer vision principles and their applications in autonomous systems via real-world projects. The emphasis on collaboration and iterative problem-solving in PBL helps students develop essential skills for their future careers. The key aspects of PBL integrated into this study include (1) Hands-On Learning: students work on practical projects that encourage them to implement theoretical knowledge to real-world problems, enhancing their understanding and retention of complex concepts [9], [10]; (2) Collaborative Learning: PBL promotes teamwork and collaboration, allowing students to develop learning skills such as communication, problem-solving, and project management [11]; (3) Reflective Practice: Students are encouraged to reflect on their experiences and identify areas for improvement, ensuring a deeper understanding of the subject matter [12]; and (4) Constructivist Theories of Education: PBL is grounded in constructivist theories which assert that learners construct knowledge via experiences and reflection. This educational approach aligns well with the principles of constructivism [13].

This work presents a comparative study of well-established feature-matching techniques, specifically emphasizing their performance in environments constrained by limited computational resources for autonomous driving. We delve into an extensive background on the evolution of these methodologies, highlighting their increasing sophistication and efficiency. At its inception, feature matching in computer vision was relatively rudimentary. The focus was primarily on identifying and aligning elementary features such as edges, corners, and blobs in different images. This simplicity, however, laid the groundwork for more complex and refined methodologies. As the discipline advanced, it introduced more sophisticated techniques that have since become cornerstones in the field. Notable among these advancements are the Oriented FAST and Rotated BRIEF (ORB) [14], Scale-Invariant Feature Transform (SIFT) [1], Accelerated-KAZE (AKAZE) [15], and Binary Robust Invariant Scalable Keypoints (BRISK) [16]. They offer increased accuracy and efficiency, making them invaluable in various applications that range from simple tasks to complex, real-world problem-solving scenarios. Furthermore, this study incorporates project-based learning (PBL) principles to enhance educational outcomes. Integrating these advanced feature-matching techniques into a hands-on curriculum provides students with practical, real-world experience in applying theoretical knowledge. This approach not only deepens their understanding of computer vision but also equips them with essential problem-solving and collaborative skills needed in autonomous driving.

While several comparative studies of feature-matching algorithms exist[5], [17], [18], they often lack a focus on performance under limited computing power conditions, such as those found on the Raspberry Pi 4B. Our comparison analysis addresses this gap, providing valuable insights into how these algorithms perform in resource-constrained environments. This analysis is particularly relevant for our course project, where students work in similar settings. The results of this comparison allow the instructor to offer informed advice on each method's strengths and limitations, helping students navigate the complexities of real-time image processing and make informed decisions when selecting and tuning algorithms for their projects.

II. Problem Statement

The increasing relevance of feature matching within constrained computing environments, particularly with low-cost single-board computers (SBCs), is becoming a crucial area of focus in computer vision research, especially in the context of the burgeoning Internet of Things (IoT) and Edge Devices. This paper addresses the critical need to test the efficiency and performance of existing feature-matching methods in such environments.

A. Efficiency

Efficiency is paramount in low-end hardware devices, which operate with limited power sources like batteries or solar panels. Feature-matching methods that demand excessive computational resources can increase energy consumption, which is a significant concern. The cooling requirements of these devices, typically equipped with passive cooling systems or minimal cooling fans, become a critical factor. Extended computing processes can lead to thermal throttling, and the power consumption of any cooling mechanism further contributes to overall energy usage. Therefore, it's essential to evaluate feature-matching algorithms for high efficiency with low energy consumption and minimal heat

generation, ensuring their effective deployment in energy-constrained and thermally sensitive environments.

B. Rotation, Perspective, and Scale Changes

Another aspect of paramount importance is testing feature-matching methods on low-end hardware against challenges like rotation, perspective, and scale changes. These changes are fundamental challenges in object recognition, as real-world scenarios often present objects in various orientations and distances, altering their appearance in images. Robustness to these variations is essential to ensure consistent and accurate object recognition across different viewpoints. Evaluating the performance of feature-matching methods under these conditions, particularly in resource-limited hardware environments, is a matter of scientific rigor and crucial for the practical advancement of computer vision technology.

C. Overview of Existing Feature Matching Methods

This study aims to balance multiple parameters, including efficiency and performance, particularly in handling image transformations such as skew, rotation, and scale changes. We have selected a suite of algorithms that are well-regarded in the field of computer vision and readily available in OpenCV, a prominent library for image processing and computer vision tasks. These algorithms, namely ORB, AKAZE, BRISK, FAST+ORB, and SIFT, are chosen for their unique characteristics and suitability for operations with low hardware requirements. Here's a brief overview of each approach:

1) ORB (Oriented FAST and Rotated BRIEF)

ORB, a fusion of the FAST keypoint detector and the BRIEF descriptor with additional enhancements, is known for its rotation invariance and partial scale invariance. It stands out for its speed and efficiency, making it apt for real-time applications. A notable application was in 2018, where researchers used ORB in a low-cost machine vision system for object shift determination in PCB board assembly [19].

2) AKAZE (Accelerated-KAZE)

An advanced version of the original KAZE algorithm, AKAZE offers accelerated performance while maintaining quality feature detection. It is robust against scale and rotation changes. Although more computationally intensive than ORB, AKAZE is known for its high-quality feature detection, with some studies achieving 98 fps at a resolution of 1024x768 on low-end hardware [20].

3) BRISK (Binary Robust Invariant Scalable Keypoints)

BRISK is designed as a robust method for keypoint detection, description, and matching. It performs well against rotation and scale variations and is relatively efficient, making it suitable for scenarios where computational efficiency is crucial. Studies have shown high-efficiency results with BRISK on low-end hardware [21].

4) FAST+ORB

This method combines the FAST keypoint detector's speed with the ORB descriptor's rotational invariance. This combination is highly efficient computationally, rendering it suitable for real-time applications.

5) SIFT (Scale-Invariant Feature Transform)

SIFT is a foundational algorithm in feature detection and matching, known for its robustness to scale and rotation changes, though it is partially invariant to perspective changes. While SIFT is more computationally intensive than the other methods, it offers high-quality features and robustness, making it a valuable control group in this study. Acceptable feature-matching speeds with SIFT have been achieved even with its computational intensity [22].

These methods were selected to balance computational efficiency and robustness to various image transformations. The choice of algorithm is guided by the objective of maintaining high efficiency and robustness in the face of skew, rotation, and scale changes with low-end hardware.

D. The Low-End Hardware

This study chose the Raspberry Pi 4B for its compatibility with OpenCV and its suitability for real-time image processing tasks in educational settings. Its connectivity and efficiency make it a practical choice for testing feature-matching algorithms in resource-limited environments[23], [24]. Despite its enhanced performance capabilities, the Raspberry Pi 4B is energy-efficient, requiring only a 5V/3A power supply, making it suitable for power-sensitive applications like embedded systems and remote monitoring. Furthermore, the Raspberry Pi community offers robust support and resources, which are invaluable for development and troubleshooting. The cost-effectiveness of the Raspberry Pi 4B, with its high performance-to-cost ratio, presents an accessible option for a wide range of purposes with budget constraints. However, we need to acknowledge that while capable of running the selected algorithms, the Pi4B's performance may not parallel that of more powerful computing systems, particularly for highly demanding tasks or high-resolution image processing, which aligns perfectly with the objectives of our study.

III. Methodology and Experimental Approaches

In this study, we undertake a detailed evaluation of various feature matching methods, focusing specifically on ORB, AKAZE, BRISK, a combination of FAST with ORB, and SIFT. These methods have been chosen due to their significance in computer vision and distinct characteristics, making them ideal candidates for comparative performance and efficiency analysis. We conducted two types of experiments. The main hardware components used in the experiments include (1) Raspberry Pi 4B: Selected for its cost-effectiveness and compatibility with OpenCV, it is ideal for representing real-time image processing in resource-constrained environments; (2) USB Camera: Used for capturing video, with resolution set to 640x480 (student can adjust) to match the processing capabilities of the Raspberry Pi; (3) Arduino MEGA2560: Controls the RC car's motors, receiving commands from the Raspberry Pi based on image processing results; and (4) Powerbank: Provides portable power to the system, enabling mobile applications.

The main software components used in the experiments include (1) OpenCV Library: Utilized to implement feature-matching algorithms like ORB, AKAZE, and BRISK, chosen for their efficiency on low-end hardware; and (2) Custom Python Scripts: Integrate OpenCV with motor control logic to enable the RC car to follow a target image by processing real-time video feeds.

All our experiments are carried out on the SBC-Raspberry Pi4B, chosen for its status as a cost-effective and low-end computing hardware that is widely used in embedded systems and edge computing applications. The OpenCV (Open Source Computer Vision Library) is this project's primary software library for image processing and feature matching. OpenCV provides a range of computer

vision tools compatible with the Raspberry Pi's ARM architecture, allowing for efficient real-time processing. This specific setup is integral to our study as it provides a realistic assessment of the performance and efficiency of the feature-matching methods on hardware that is commonly employed in embedded and real-time vision systems. The choice of Raspberry Pi4B ensures that our findings are applicable and relevant to the typical environments where these feature-matching methods would be deployed, e.g., in our course project, thereby enhancing the practical value of our research.

Template Matching Experiment: This experiment begins with template-matching exercises. Here, we compare the original template image with its varied versions to assess each method's response to scale, rotation, and perspective changes. This initial phase is critical in understanding how each feature-matching method performs under different image transformation conditions, setting the stage for more complex application-based testing. We measured the time and reprojection error for each combination.

Real-Time Video Testing and Autonomous Driving Experiment: In our comprehensive study, we conduct a two-step experiment starting with real-time video streaming tests to evaluate the proficiency of various feature-matching methods in accurately detecting features in live video footage, an essential aspect in real-world scenarios often characterized by background noise and dynamic conditions. Each frame in the streaming video presents a unique challenge, requiring the algorithms to identify and match features against variable and noisy backgrounds consistently. Methods that excel in these conditions are advanced to the second phase: autonomous driving simulation experiments. In this phase, we employ 1:10 scale remote control cars to simulate practical application scenarios, focusing on the methods that showed promising performance in the initial template-matching experiments. This simulation is designed to assess the selected feature-matching methods' applicability and robustness in a dynamic, real-world-like environment, mirroring the challenges inherent in autonomous vehicle navigation. The primary objective of these tests is to gauge the algorithms' ability to accurately and reliably identify and respond to environmental features in real-time, a critical requirement for the effective functioning of autonomous driving systems.

A. Data Collection and Analysis Methodology

1) Data Collection

We meticulously record several key metrics for each feature-matching method and image transformation. These metrics include reprojection error, time taken, memory usage, and estimated distance. This comprehensive data collection is crucial for evaluating the performance of each method under various conditions. The data gathered from the experiments are systematically compiled into a CSV file format. This structured approach allows for an organized and thorough analysis, enabling us to directly compare the performance and efficiency of the different feature-matching methods across a spectrum of image transformations.

2) BFMatcher

We use the Brute Force Matcher (BFMatcher) across all feature-matching methods for consistency and practicality in our comparisons. BFMatcher is known for its straightforward approach to finding the best matches between feature descriptors. It operates by exhaustively comparing each descriptor in one set against every descriptor in the other, ensuring no potential match is overlooked. For methods using binary descriptors, such as ORB, AKAZE, and BRISK, the BFMatcher is configured with the NORM_HAMMING norm type. This norm type is particularly suitable for binary descriptors and aids in finding the most accurate matches. Conversely, for the SIFT algorithm, which utilizes floating-point descriptors, the Euclidean (L2) norm is employed. The L2 norm is a mathematical way to measure the distance between two points in space, calculated as the square root of the sum of the squared differences between corresponding coordinates. This differentiation in the configuration of BFMatcher for different descriptor types is critical for obtaining reliable and accurate matching results.

The selection of BFMatcher aligns with our research objective to evaluate these feature-matching methods' inherent effectiveness and efficiency. By employing BFMatcher, we aim to minimize the influence of complex matching algorithms on our results, focusing instead on the intrinsic capabilities of each feature-matching method. This approach ensures that our findings reflect the actual performance of the methods rather than the matching algorithms' efficiency.

3) Evaluation of Timing Efficiency

We evaluate the efficiency of various feature-matching methods by measuring the time taken to identify a sufficient number of high-quality matches, from the initiation of the feature detection process to the point where the algorithm successfully meets the criteria set by Lowe's ratio test[1]. This test is crucial in our assessment, acting as a robustness filter to ensure only the most reliable matches are selected based on the proximity of the best match compared to the second-best for each keypoint. This approach allows us to gauge the speed of the matching process and the quality of the matches, which is particularly vital in practical applications where both speed and reliability are essential. We also establish a specific criterion for an 'adequate number of matches,' aligning with the requirements for subsequent homography calculations. This setting ensures a balanced evaluation of the quantity and quality of matches, reflective of real-world feature-matching scenarios and practical application needs.

4) Evaluation Metrics for Feature Matching Methods

Reprojection Error: This metric evaluates feature matching accuracy by calculating the average distance between corresponding points in the template image and their locations in the transformed image, following the application of the estimated homography. A lower reprojection error signifies greater precision in the feature matching, indicating a more accurate alignment between the original and transformed images.

Number of Good Matches: Utilizing Lowe's ratio test, this metric quantifies the number of reliable match pairs. It involves counting the pairs where the distance between the closest and second-closest matches falls below a set threshold, commonly 0.75. This test is instrumental in identifying and retaining the most robust matches, effectively filtering out less reliable or ambiguous ones.

Computation Time: This metric measures the time required to complete the feature-matching process between images. It directly indicates the algorithm's speed and operational efficiency, which is particularly vital in applications where time is a critical factor. The computation time helps assess the practicality of the feature-matching methods for real-time or time-constrained environments.

Maximum Detection Range: This metric gauges the ability of feature-matching methods to detect features over distances in real-time video streaming. It serves as a preliminary test for the autonomous driving simulation experiments. A feature-matching method with a limited detection range may necessitate a faster response time, which is crucial for real-time decision-making in scenarios such as autonomous driving. This metric reflects the method's suitability for applications requiring long-range perception.

Fig. 1. Base Image [25].

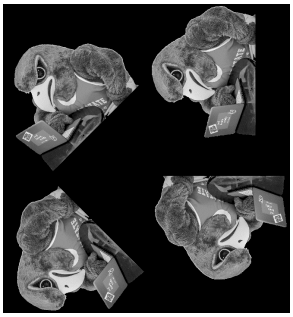

Fig. 2. Four Rotated Angles.

B. Experiment I: Template Matching

In Fig.1, our chosen base image for experimentation is Montclair State University's Red Hawk mascot, a particularly complex subject for testing the efficacy of feature-matching algorithms. This image was selected due to its unique attributes that pose significant challenges for feature matching. Notably, the mascot's distinctive fur coat adds complexity with its textured surface, presenting an array of similar features densely packed together. This characteristic of the fur coat heightens the likelihood of incorrect matches or false positives and demands that the algorithms be more adept at discerning unique features amidst a visually dense pattern. The fur texture notably increases the difficulty of the feature-matching process as the algorithms need to differentiate between subtle variations in the pattern, simulating the challenges found in natural and cluttered environments. In our experiment, we subject the base image to a series of transformations designed to replicate real-world scenarios, thus thoroughly assessing the robustness of each feature-matching method under various conditions. These transformations emulate typical variations encountered in practical applications and are categorized as follows:

Perspective Changes: The perspective transformations are applied to the base image to simulate different viewing angles and depth variations. This aspect of the test evaluates the algorithms' ability to adapt to changes in viewpoint and scale. The experiment incorporates four levels of yaw perspective transformations, set at parameters of 0.1, 0.2, 0.3, and 0.4, offering a graded scale of perspective alteration.

Rotations: The image is rotated at various angles to test rotational invariance. We use angles from 0° to 180° to provide a spectrum of rotational changes ranging from moderate to significant, as shown in Fig. 2. This step is crucial for determining how well the algorithms can maintain feature recognition despite the orientation of the image.

Scaling: The image undergoes resizing to different scales to evaluate scale invariance. The chosen scaling percentages for the experiment are 25%, 50%, 125%, and 150%, incorporating both downscaling and upscaling. This variation in image size tests the algorithms' capability to handle features in images of different dimensions.

These transformation tests are integral to our method, as they challenge the feature-matching algorithms with conditions that closely resemble those they would encounter in real-world applications. The outcome of these tests provides valuable insights into each method's adaptability and reliability across a range of common image transformations.

C. Experiment II-A: Real-time Video Streaming Test

In our experiment's real-time video streaming phase, we optimize the camera setup for the constraints of the SBC-Raspberry Pi4B. To manage the computing load, we set the camera resolution to 640x480. This resolution reduction is a strategic choice to decrease the overall computational intensity, making it more suitable for the capabilities of a single-board computer (SBC) like the Raspberry Pi4B, particularly in the context of autonomous driving applications.

We control the frame rate for this test at 2 frames per second. This rate is intentionally chosen to afford each feature-matching method ample time to process the frames and perform the necessary feature-matching tasks. It is a crucial adjustment to ensure that the limitations of the SBC do not unduly impact the effectiveness of the feature-matching methods being tested.

This setup compares the maximum detection range for a letter-sized printed base image. This approach allows us to assess each method's capability to detect and match features over various distances, an essential factor in real-world applications like autonomous driving.

The experimental code is designed to record critical data for each frame, including the time taken for feature matching, the number of suitable matches determined by Lowe's ratio test, and the estimated detection range. Additionally, we set the template size in this experiment to 50% of the resolution. This scale is a calculated decision to balance accuracy and computational time, ensuring that the feature matching is efficient and effective under the specified test conditions.

D. Experiment II-B: Autonomous Driving

We utilize a 1:10 scale race car for the autonomous driving test component, integrating a real-world application scenario into our study[9], [10]. The test setup involves placing a letter-sized printed picture in front of the RC car, which is the target for the feature-matching methods. As Fig. 3 shows, a camera is mounted on the front of the RC car, designed to capture the image of the printed picture as the car maneuvers.

The initial positioning of the RC car is determined by the maximum detection range identified for each feature-matching method from the previous real-time video streaming tests. This strategic placement ensures that each method is tested within its optimal operational range. The frame rate for this test is also adjusted based on the outcomes from the video streaming experiments, allowing us to maintain consistency in the testing conditions.

The primary objective of this phase is to assess each feature-matching method's practical application in an autonomous driving scenario. Upon starting the RC car, the

system should detect the printed picture using the selected feature-matching method. The car is then programmed to move forward towards the target. A critical aspect of the test is to evaluate the algorithm's capability to accurately detect the distance to the target and instruct the RC car to stop upon reaching a pre-set range from the printed picture.

Fig. 3. The RC with SBC and Camera.

This test is crucial in determining the real-world applicability of the feature-matching methods in scenarios that require precise object detection and distance estimation, such as in autonomous vehicle navigation. The ability of the system to detect the target and accurately respond by stopping at the correct range is a crucial indicator of the method's effectiveness and reliability in real-time applications.

IV. Feature Matching Result Analysis

A. Results of Experiment I: Template Matching

In Table I, detailing the baseline conditions for each feature matching method, it was observed that ORB exhibited the lowest memory usage, making it the most memory-efficient option. On the opposite end, SIFT recorded the highest memory consumption, indicating a relatively higher demand for system resources. In terms of processing speed, ORB also stood out as the fastest, while the FAST+ORB combination was the slowest. SIFT, despite its high memory usage, was the second slowest. AKAZE and BRISK displayed moderate performance, ranking them in terms of efficiency between the extremes of ORB and SIFT.

TABLE I. Baseline

Method	Skew	Rotate	Scale	Error	Time(s)	Memory(MB)
ORB	0	0	100	0	0.20	185.781
AKAZE	0	0	100	0	2.57	420.57
BRISK	0	0	100	0	3.04	400.48
FAST+ORB	0	0	100	0	12.57	406.77
SIFT	0	0	100	0	11.58	499.75

Table II showed that the FAST+ORB method showed poor performance with scale changes in our experiments, while SIFT struggled with upscaling and took longer processing times. Conversely, BRISK and AKAZE maintained stable performance across various scales. Notably, ORB emerged as the fastest method among those tested. Most of the methods have some level of scale-invariant, and the changes can also be explained with the image size.

TABLE II. Scale Change Results

Method	Scale(Percentage)	Error	Time(s)	Memory(mb)
AKAZE	25	25.27	1.42	247.17
AKAZE	50	13.47	1.55	291.45
AKAZE	125	5.56	3.56	285.16
AKAZE	150	5.16	4.72	263.70
BRISK	25	25.30	0.81	267.55
BRISK	50	4.18	1.31	275.09
BRISK	125	2.71	2.89	275.09
BRISK	150	2.97	3.55	281.99
FAST+ORB	25	114.92	1.10	279.47
FAST+ORB	50	101.05	3.65	279.54
FAST+ORB	125	14.52	15.15	280.61
FAST+ORB	150	238.79	22.33	281.17
ORB	25	3.68	0.13	157.82
ORB	50	12.80	0.16	170.54
ORB	125	1.38	0.24	170.54
ORB	150	1.88	0.31	170.54
SIFT	25	18.14	2.09	438.88
SIFT	50	6.25	3.23	470.75
SIFT	125	1.33	16.07	558.09
SIFT	150	1.20	20.37	537.70

As presented in Fig. 4, our experiments with rotated images observed that rotations at 0, 90, and 180 degrees resulted in similar processing times for all methods, indicating rotational invariance, except for the FAST+ORB combination. However, this invariance faced significant challenges with rotations at 45 and 135 degrees (odd multiples of 45 degrees). BRISK demonstrated the most robust scale invariance among the tested methods, indicated by the minor percentage change in processing time under these conditions. AKAZE maintained a stable error rate while showing a near 100% change in processing time for these odd-angle rotations. While AKAZE may take longer to process these rotations, its accuracy remains consistent. On the other hand, FAST+ORB struggled significantly, failing to calculate the error for almost 80% of the rotations, which points to a lack of comprehensive rotational invariance compared to the other methods. This limitation of the FAST+ORB method highlights its potential inadequacies in scenarios involving varied rotational transformations.

As shown in Table III, ORB emerged as the fastest method in our tests, but it struggled with significant perspective changes, a challenge that was also evident in other methods to varying degrees. BRISK displayed the most consistent performance when dealing with these types of transformations. Across all tests, ORB consistently used less memory than the established baseline, suggesting that converting color images to grayscale could be advantageous, as it may reduce memory consumption. This finding is particularly relevant for edge devices, indicating that using grayscale images or videos could lower memory costs in real-world applications.

TABLE III. Perspective Result

Method	Skew Rate	Error	Time (s)	Memory(MB)
ORB	0.1	1.10	0.19	161.61
ORB	0.2	15.93	0.18	161.61
ORB	0.3	95.63	0.20	159.77
ORB	0.4	358.21	0.18	161.61
AKAZE	0.1	4.09	2.66	296.70
AKAZE	0.2	13.03	2.64	296.95
AKAZE	0.3	33.70	2.47	288.04
AKAZE	0.4	165.75	2.43	256.69
BRISK	0.1	2.37	2.58	265.76
BRISK	0.2	6.62	2.54	265.76
BRISK	0.3	14.45	1.93	265.76
BRISK	0.4	60.06	1.49	265.76
FAST+ORB	0.1	1.94	8.84	271.30
FAST+ORB	0.2	13.20	7.46	271.29
FAST+ORB	0.3	117.63	5.67	270.92
FAST+ORB	0.4	483.77	3.14	270.71
SIFT	0.1	2.08	10.35	488.91
SIFT	0.2	3.88	8.61	452.46
SIFT	0.3	12.75	6.56	460.77
SIFT	0.4	88.14	3.82	450.42

By analyzing all the results of feature-matching against common variations, we determined that ORB, AKAZE, BRISK, and SIFT are potentially suitable methods for SBC

real-time streaming video tests. However, the FAST+ORB combination underperformed due to its inability to maintain rotational invariance, particularly with odd multiples of 45 degrees, which are common in real-world orientations. Additionally, its failure to calculate error rates for a significant portion of rotation tests indicates a lack of comprehensive scale and rotational invariance. It is crucial for real-time processing where rapid and accurate feature detection is required for decision-making in dynamic environments.

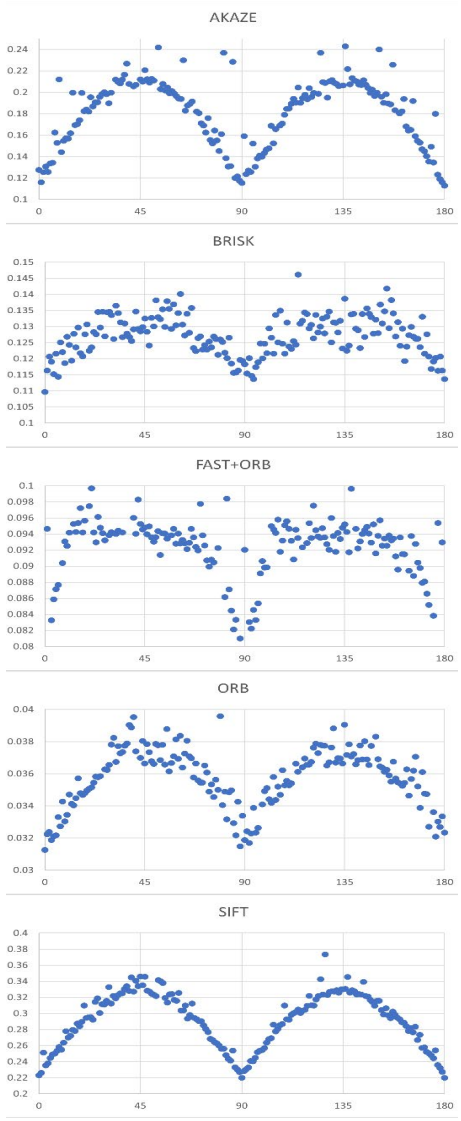

Fig. 4. Rotation Results.

B. Results of Experiment II-A: Real-time Video Streaming Test

We employed the same code used in the autonomous driving test for the real video stream experiment, with the speed parameter set to zero. We set up a threshold of 10 good matches as a benchmark for successful feature detection and adjusted the template resolution to approximately 50% of the camera's resolution to balance accuracy and computational demand. In this part of the experiment, reprojection error was deprioritized in favor of the more critical metric of object detection success. We gradually increased the distance of the RC car from the printed paper, monitoring the feature-matching process. The estimated distance at which the method no longer produced enough good matches to detect the object was then recorded. This distance effectively represents the maximum operational range for each feature-matching method in a real video streaming context.

Following the real video streaming test, it was determined that AKAZE and BRISK possessed the capabilities required to proceed to the autonomous driving test phase. ORB, however, did not yield a sufficient number of good matches to be considered adequate for this application. Meanwhile, although accurate, SIFT required approximately one second to complete its matching process using the brute force matcher, which may be too slow for real-time autonomous driving requirements where immediate decision-making is critical.

The analysis of data points near the threshold of detection loss revealed that, under the conditions of a 640x480 video resolution and a template scaled to 50%, AKAZE achieved a maximum detection range of 0.77 meters, while BRISK extended slightly further to 1.01 meters. Regarding detection speed, AKAZE managed an average detection time of 0.23 seconds, and BRISK was closely matched with an average of 0.25 seconds, demonstrating promptness suitable for real-time applications.

C. Results of Experiment II-B: Autonomous Driving

In the autonomous driving test, the printed picture was positioned at a slight angle to the RC car to simulate perspective change, adding another layer of complexity to the detection challenge. The car started approximately 0.7 meters from the target and was programmed to drive autonomously towards the printed picture. The system was designed to detect the range continuously, and once the calculated distance to the target was less than 0.3 meters, the RC car was instructed to stop automatically. This setup aimed to emulate real-world conditions where perspective changes are expected, testing the feature-matching method's ability to adapt and accurately gauge distances for safe navigation.

The outcomes of the autonomous driving test indicated that both AKAZE and BRISK, while capable at lower speeds(97 out of 180), failed to perfectly pass the test when the speed of the RC car was increased(100 out of 180). At reduced velocities, both methods successfully stopped the RC car before reaching the printed picture. However, at higher speeds, they were unable to stop in time. This failure can likely be attributed to several factors beyond detection speed alone, such as the camera's capture rate, video buffering delays, and the RC car's response time. These results underscore the necessity for quick response capabilities and more advanced or efficient hardware to meet the demands of autonomous driving, where timely and accurate detection and actuation are crucial.

V. Course Project Case: Autonomous Driving

To illustrate the practical application of feature-matching techniques in an educational setting, we present a comprehensive project within a robotics course at Montclair State University that challenges students to apply feature-matching algorithms on the Raspberry Pi 4. By tackling real-world autonomous driving scenarios, students gain practical experience that reinforces crucial theoretical concepts through project-based learning (PBL).

A. Project Overview

The objective of the project is to build and program an RC car that can navigate autonomously using feature-matching algorithms. The car has a Raspberry Pi 4B for image processing and an Arduino microcontroller for motor control. The Powerbank, Pi4, and camera are all integrated into the RC car with a custom-designed laser-cut board. The project involves some key components: (1) Image processing:

Students use OpenCV to implement feature-matching algorithms such as ORB, AKAZE, and BRISK. These algorithms enable the car to recognize and track objects or landmarks in its environment; (2) Servo/motor control with Arduino: The Arduino microcontroller is programmed to control the car's motors based on the input received from the Raspberry Pi. This involves writing code to interpret the feature-matching results and translate them into movement commands; and (3)Integration and testing: Students integrate the image processing and motor control components, testing the car's ability to navigate through a course with various obstacles and landmarks. They evaluate the performance of different feature-matching algorithms in real-time scenarios.

B. Implementation Steps

The implementation steps include (1) Hardware setup: Students assemble the RC car with a camera mounted on the front, connected to the Raspberry Pi, ensure the Raspberry Pi communicates with the Arduino, which controls the car's motors, and integrate the power bank, Pi4, and camera on the RC car using a custom-designed laser-cut board; (2) Software development: Students install OpenCV on the Raspberry Pi and write scripts to capture and process video frames, implement feature-matching algorithms and test their performance in detecting and tracking objects, and program the Arduino to control the car's movements based on the input from the Raspberry Pi. Then they develop codes for the RC car to follow the selected templated picture at a proper distance, move when the picture moves, stop when the picture stops, and turn when the picture turns. This simulates adaptive cruise control, a Level 1 autonomous driving feature; and (3) Testing and optimization: Students conduct experiments to evaluate the car's performance under different conditions, such as varying light levels and obstacle configurations. They also optimize the code for efficiency and accuracy, ensuring the car can navigate smoothly and respond quickly to environmental changes.

C. Results and Learning Outcomes

By completing this project, students gain hands-on experience with advanced computer vision techniques and their applications in autonomous systems. They learn to work collaboratively, solve complex problems, and apply theoretical knowledge in a practical context. The project-based approach enhances their understanding of feature matching, image processing, and embedded systems, preparing them for future challenges in the field of computer vision and robotics. Students benefit from:

1) Hands-On Learning

Students apply theoretical knowledge to real-world problems by assembling and programming the RC car, enhancing their understanding and retention of complex concepts. This hands-on experience solidifies their grasp of computer vision and embedded systems.

2) Collaborative Learning

Working in teams, students design, implement, and test their RC car, fostering a collaborative environment critical for professional settings.

3) Reflective Practice

Students reflect on their experiences, analyze their performance, and identify areas for improvement. Reflective sessions during and after each project meeting help them discuss successes and challenges, fostering a continuous improvement mindset.

4) Constructivist Theories of Education

Students build their understanding through the iterative process of designing, testing, and refining their Pi4-controlled RC car, reinforcing the connection between theoretical concepts and practical applications. Additionally, a simple sample code will be provided to the students to help them get started. Teachers will encourage students to optimize the code with proper feature-matching methods, functional programming, exception control, digital filtering, and other techniques.

Feature matching serves as the core method in this project, providing both a technical challenge and a valuable educational tool within the PBL framework. Students are encouraged to explore various feature-matching techniques, each with its trade-offs—some methods offer high accuracy but are slower. In contrast, others are faster but lack precision. The challenge lies in selecting a balanced approach or modifying the parameters of these methods to enhance the efficiency of slower techniques or increase the accuracy of quicker ones. The project's success is measured by the performance of the RC car, with students allowed to fine-tune their settings to find the optimal configuration for their chosen feature-matching method. The instructor's prior experiments with different feature-matching algorithms on the Raspberry Pi 4B provide a solid foundation of knowledge, ensuring that students understand how each method performs under resource-constrained conditions.

While all feature-matching methods may be capable of identifying a target image, detecting the image in real-time video streams with an acceptable response time is a complex task. This challenge is compounded by the need to design a practical algorithm for the RC car to follow the target image accurately. The final test, where another student or a teaching assistant moves the target image, requires the RC car to respond correctly in real-time. Optimizing the driving algorithm to achieve this requires significant trial and error and a deep understanding of the project's technical and practical aspects. This complexity ensures that the project is far from trivial, providing students with a rigorous, hands-on learning experience that helps them with real-world challenges in computer vision and robotics.

This project exemplifies how project-based learning can be effectively implemented in engineering and computing education, providing students with valuable skills and experiences directly applicable to their future careers. By engaging in this hands-on project, students deepen their understanding of the subject matter and develop essential soft skills such as teamwork, communication, and critical thinking. This holistic approach to learning ensures that students are well-equipped to tackle the challenges of the modern engineering and technology landscape.

VI. Conclusion and Future Work

This study establishes a framework for comparing feature-matching methods, offering substantial potential for enhancement through future research. This proposed framework not only evaluates current methodologies but also sets the stage for ongoing advancements in feature matching, particularly within low-end hardware environments. Such future works could extend the scope and depth of this research, leading to more refined and efficient feature-matching solutions adaptable to various real-world applications.

Enhancing the test environment to include a broader range of single-board computers (SBCs) is essential for the

future progression of this research. While effective, the current study's reliance on the Raspberry Pi 4 only represents a fraction of the rapidly diversifying SBC market. With the expected release of the Raspberry Pi 5 and the continuous introduction of new SBC models, it is crucial to evaluate feature-matching methods across different hardware platforms. Future research will focus on testing various SBCs with varying processing power, memory capacity, and graphical capabilities. This approach would shed light on how these differing hardware specifications impact the performance of feature-matching algorithms. Moreover, conducting comparative analyses between different generations of the same SBC, such as between the Raspberry Pi 4 and the forthcoming Raspberry Pi 5, could provide valuable insights into the effects of hardware technological advancements on the efficiency and accuracy of these algorithms.

This study also underscores the effectiveness of project-based learning (PBL) in enhancing students' understanding of complex concepts through hands-on projects, as demonstrated by the RC car project. Overall, this project-based approach ensures students are well-prepared for modern engineering and technology, combining technical knowledge with essential soft skills. Future work will continue to explore the integration of advanced computational techniques and a broader range of hardware platforms, further enhancing this research's practical and educational impacts.

ACKNOWLEDGMENT

This work is supported in part by the National Science Foundation under Grants CNS-2117308 and CMMI-2138351.

REFERENCES

[1] D. G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," *Int. J. Comput. Vis.*, vol. 60, no. 2, pp. 91–110, Nov. 2004, doi: 10.1023/B:VISI.0000029664.99615.94.

[2] R. Scherer, "Concluding Remarks and Perspectives in Computer Vision," in *Computer Vision Methods for Fast Image Classification and Retrieval*, vol. 821, in Studies in Computational Intelligence, vol. 821. , Cham: Springer International Publishing, 2020, pp. 137–137. doi: 10.1007/978-3-030-12195-2_6.

[3] M. D. Ansari, A. R. Mishra, and F. T. Ansari, "New Divergence and Entropy Measures for Intuitionistic Fuzzy Sets on Edge Detection," *Int. J. Fuzzy Syst.*, vol. 20, no. 2, pp. 474–487, Feb. 2018, doi: 10.1007/s40815-017-0348-4.

[4] K. T. M. Han and B. Uyyanonvara, "A Survey of Blob Detection Algorithms for Biomedical Images," in *2016 7th International Conference of Information and Communication Technology for Embedded Systems (IC-ICTES)*, Bangkok, Thailand: IEEE, Mar. 2016, pp. 57–60. doi: 10.1109/ICTEmSys.2016.7467122.

[5] S. A. K. Tareen and Z. Saleem, "A comparative analysis of SIFT, SURF, KAZE, AKAZE, ORB, and BRISK," in *2018 International Conference on Computing, Mathematics and Engineering Technologies (iCoMET)*, Sukkur: IEEE, Mar. 2018, pp. 1–10. doi: 10.1109/ICOMET.2018.8346440.

[6] J. Ma, X. Jiang, A. Fan, J. Jiang, and J. Yan, "Image Matching from Handcrafted to Deep Features: A Survey," *Int. J. Comput. Vis.*, vol. 129, no. 1, pp. 23–79, Jan. 2021, doi: 10.1007/s11263-020-01359-2.

[7] Z. Wang, H. Zhu, Y. Ma, and A. Basu, "XAI Feature Detector for Ultrasound Feature Matching," in *2021 43rd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*, Mexico: IEEE, Nov. 2021, pp. 2928–2931. doi: 10.1109/EMBC46164.2021.9629944.

[8] S. Karim, Y. Zhang, A. A. Brohi, and M. R. Asif, "Feature Matching Improvement through Merging Features for Remote Sensing Imagery," *3D Res.*, vol. 9, no. 4, p. 52, Dec. 2018, doi: 10.1007/s13319-018-0203-x.

[9] W. Wang and L. Paulino, "Instill Autonomous Driving Technology into Undergraduates via Project-Based Learning," in *2021 IEEE Integrated STEM Education Conference (ISEC)*, Princeton, NJ, USA: IEEE, Mar. 2021, pp. 284–287. doi: 10.1109/ISEC52395.2021.9763928.

[10] L. Paulino, M. Zhu, and W. Wang, "Learning Autonomous Driving in Tangible Practice: Development and On-Road Applications of a 1/10-Scale Autonomous Vehicle," in *2021 IEEE Frontiers in Education Conference (FIE)*, Lincoln, NE, USA: IEEE, Oct. 2021, pp. 1–4. doi: 10.1109/FIE49875.2021.9637402.

[11] D. Kokotsaki, V. Menzies, and A. Wiggins, "Project-based learning: A review of the literature," *Improv. Sch.*, vol. 19, no. 3, pp. 267–277, Nov. 2016, doi: 10.1177/1365480216659733.

[12] T. Morgan, "Enabling Meaningful Reflection Within Project-Based-Learning in Engineering Design Education," in *Design Education Today*, D. Schaefer, G. Coates, and C. Eckert, Eds., Cham: Springer International Publishing, 2019, pp. 61–90. doi: 10.1007/978-3-030-17134-6_4.

[13] N. F. Jumaat, Z. Tasir, N. D. A. Halim, and Z. M. Ashari, "Project-Based Learning from Constructivism Point of View," *Adv. Sci. Lett.*, vol. 23, no. 8, pp. 7904–7906, Aug. 2017, doi: 10.1166/asl.2017.9605.

[14] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," in *2011 International Conference on Computer Vision*, Barcelona, Spain: IEEE, Nov. 2011, pp. 2564–2571. doi: 10.1109/ICCV.2011.6126544.

[15] P. F. Alcantarilla, A. Bartoli, and A. J. Davison, "KAZE Features," in *Computer Vision – ECCV 2012*, vol. 7577, A. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato, and C. Schmid, Eds., in Lecture Notes in Computer Science, vol. 7577. , Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 214–227. doi: 10.1007/978-3-642-33783-3_16.

[16] S. Leutenegger, M. Chli, and R. Y. Siegwart, "BRISK: Binary Robust invariant scalable keypoints," in *2011 International Conference on Computer Vision*, Barcelona, Spain: IEEE, Nov. 2011, pp. 2548–2555. doi: 10.1109/ICCV.2011.6126542.

[17] O. Yakovleva and K. Nikolaieva, "RESEARCH OF DESCRIPTOR BASED IMAGE NORMALIZATION AND COMPARATIVE ANALYSIS OF SURF, SIFT, BRISK, ORB, KAZE, AKAZE DESCRIPTORS," *Adv. Inf. Syst.*, vol. 4, no. 4, pp. 89–101, Dec. 2020, doi: 10.20998/2522-9052.2020.4.13.

[18] M. Bansal, M. Kumar, and M. Kumar, "2D object recognition: a comparative analysis of SIFT, SURF and ORB feature descriptors," *Multimed. Tools Appl.*, vol. 80, no. 12, pp. 18839–18857, May 2021, doi: 10.1007/s11042-021-10646-0.

[19] R. Pramudita and F. I. Hariadi, "Development Of Techniques to Determine Object Shifts for PCB Board Assembly Automatic Optical Inspection (AOI)," in *2018 International Symposium on Electronics and Smart Devices (ISESD)*, Bandung: IEEE, Oct. 2018, pp. 1–4. doi: 10.1109/ISESD.2018.8605458.

[20] L. Kalms, K. Mohamed, and D. Göhringer, "Accelerated Embedded AKAZE Feature Detection Algorithm on FPGA," in *Proceedings of the 8th International Symposium on Highly Efficient Accelerators and Reconfigurable Technologies*, Bochum Germany: ACM, Jun. 2017, pp. 1–6. doi: 10.1145/3120895.3120898.

[21] E. Azimi, A. Behrad, M. B. Ghaznavi-Ghoushchi, and J. Shanbehzadeh, "A fully pipelined and parallel hardware architecture for real-time BRISK salient point extraction," *J. Real-Time Image Process.*, vol. 16, no. 5, pp. 1859–1879, Oct. 2019, doi: 10.1007/s11554-017-0693-4.

[22] K. Moren and D. Göhringer, "A framework for accelerating local feature extraction with OpenCL on multi-core CPUs and co-processors," *J. Real-Time Image Process.*, vol. 16, no. 4, pp. 901–918, Aug. 2019, doi: 10.1007/s11554-016-0576-0.

[23] J. -, M. Husna, and A. R. Lubis, "OpenCV Using on a Single Board Computer for Incorrect Facemask-Wearing Detection and Capturing," *J. Inform. Telecommun. Eng.*, vol. 5, no. 2, pp. 315–323, Jan. 2022, doi: 10.31289/jite.v5i2.6118.

[24] S. Brahmbhatt, *Practical OpenCV*. Berkeley, CA: Apress, 2013. doi: 10.1007/978-1-4302-6080-6.

[25] Montclair State University, *Redhawk Student Service*. 2023. Accessed: Jan. 04, 2024. [Png]. Available: https://images.app.goo.gl/pzcBxJjHqwEowLGh8.