ORB-Based Map Fusion with Position Transformation for Enhanced Pairwise Connection

Lucas Alexandre Zick[1,2], Dieisson Martinelli[1,2], Andre Schneider de Oliveira[1] and Vivian Cremer Kalempa[2]

[1]*Graduate Program in Electrical and Computer Engineering, Universidade Tecnológica Federal do Paraná (UTFPR), Curitiba, Brazil*

[2]*Department of Information Systems, Universidade do Estado de Santa Catarina (UDESC), São Bento do Sul, Brazil*

Keywords: Map Fusion, Robot Localization, ORB Algorithm, Pairwise Map Alignment.

Abstract: This paper discusses developing and evaluating a map fusion algorithm based on ORB (Oriented FAST and Rotated BRIEF) feature matching, designed to improve the integration of robotic occupancy grids. The algorithm effectively merges maps generated by multiple robots, accommodating map size and orientation variations. A key aspect of its functionality is the ability to accurately position robots within the fused map, even when the overlap between maps is minimal. Comprehensive testing demonstrated the algorithm's effectiveness in identifying correlations between different map pairs and aligning them accurately, as well as its capability to assess the success of the merging process, distinguishing between successful merges and those with inaccuracies. The findings indicate that this approach significantly enhances the capabilities of multi-robot systems, improving navigation and operational efficiency in complex environments.

1 INTRODUCTION

The presence of multi-robot systems in robotics has significantly expanded the range of possible solutions, primarily improving task execution speed (Gautam and Mohan, 2012). Due to the growing ease of implementing these technologies, multi-robot systems have become increasingly common in academic research and industrial settings (Arai et al., 2002; Vaščák and Herich, 2023). This collaborative approach allows tasks to be performed more quickly and efficiently and facilitates the resolution of complex problems that would be challenging for robots operating independently.

In addition to efficiency, using multiple robots also provides additional advantages, such as the ability to perform simultaneous tasks at different points within an environment (Gautam and Mohan, 2012). This is particularly useful in scenarios such as exploring unknown environments, monitoring vast areas (such as space exploration), and conducting search and rescue operations (Chakraa et al., 2023). With advancements in communication and sensing technologies, robots can share information in real time, enabling better coordination and a more accurate understanding of the environment in which they operate.

However, this growing reliance on multi-robot systems brings specific challenges to the forefront, particularly in map fusion and the integration of sensory information (Stathoulopoulos et al., 2023). The ability of different robots to merge their mapping data, especially in situations where the maps may be disconnected or incomplete, is crucial for ensuring functional autonomous navigation and operational effectiveness (Mukhopadhyay et al., 2023). Therefore, research into robust and scalable map fusion techniques becomes essential for the success of multi-robot systems in real-world applications (Ahmed Jalil and Kasim Ibraheem, 2023).

Among the most promising solutions to address these challenges is the ORB (Oriented FAST and Rotated BRIEF) algorithm, which is widely used in robotics and computer vision applications (Ahmed et al., 2024). ORB is characterized by its fast and efficient detection of keypoints and its robustness in recognizing visual patterns, even in situations with variations in orientation or scale (Karami et al., 2017). Its application in map fusion allows for the identification of correspondences between different maps generated by robots of varying models in dynamic environments, facilitating the combination of visual data and the construction of a unified map. Addition-

Zick, L. A., Martinelli, D., de Oliveira, A. S. and Kalempa, V. C.
ORB-Based Map Fusion with Position Transformation for Enhanced Pairwise Connection.
DOI: 10.5220/0013670500003982

In *Proceedings of the 22nd International Conference on Informatics in Control, Automation and Robotics (ICINCO 2025) - Volume 2*, pages 185-193
ISBN: 978-989-758-770-2; ISSN: 2184-2809

ally, ORB provides an efficient solution for handling large-scale maps, demonstrating good scalability regarding processing time and memory usage (Sabry et al., 2024).

In this context, this work aims to develop an approach for map fusion based on ORB, focusing not only on the combination of the maps themselves but also on the precise transformation of the positions of multiple robots within the final map. By effectively integrating the mapping data from two robots, it becomes possible to construct a unified map of the environment and accurately transform the robots' positions within the resulting map, ensuring more efficient, safe, and coordinated navigation. Additionally, the ORB algorithm utilized in this approach can assess the likelihood of overlap between the provided maps. This capability allows the system to identify scenarios where the maps are disconnected or have minimal overlap, avoiding unnecessary processing for non-connected maps and optimizing data fusion only when relevant. Thus, the proposed solution is robust enough to handle maps of varying sizes and scenarios where the robots lack complete environmental information, providing a more efficient and adaptable solution for multi-robot systems.

The contributions of this work include the development of a scalable map fusion technique based on the ORB algorithm, designed to handle partial overlap or disconnection between maps, and implementing a mechanism to correctly transform and align the positions of robots within the final map. The experimental validation of the approach was conducted exclusively in simulated scenarios, using randomly generated maps, which allowed for controlled testing of the technique's robustness and effectiveness. Although the results demonstrate the potential of ORB-based map fusion in virtual multi-robot systems, continuous real-time application in physical robots has not yet been explored. This represents the next step to expand the research, validating the approach in real-world and challenging environments, where the variations of the physical world can further demonstrate the technique's feasibility in practical situations.

The main contribution of this work is an integrated framework for map fusion that addresses the practical challenge of aligning maps with partial or disconnected overlaps while ensuring the correct transformation of robot positions into the unified reference frame. While foundational techniques like ORB are well-established, our work focuses on their application within a system that can reliably determine when to merge maps and how to accurately place each robot within the resulting shared environment. This approach is validated through a quantitative error metric that enhances the system's reliability by discarding low-quality fusions. The experimental validation was conducted exclusively in simulated scenarios, which allowed for controlled testing of the technique's robustness. Although these results are promising, real-time application on physical robots has not yet been explored and represents the next step to validate the approach in challenging real-world environments.

The structure of the paper is as follows: Section *Proposed Strategy* provides a detailed presentation of the algorithm's development. Section *Experiments and Validation* introduces the experiments and validations conducted, along with the obtained results. Finally, Section *Conclusion* offers concluding remarks and discusses potential future work.

2 RELATED WORK

In the field of map fusion for multi-robot systems, various approaches have been proposed to enhance the integration of mapping data from multiple robots. One notable work is by Dieisson Martinelli, who developed a method that requires standard references within the mapped environments (Martinelli et al., 2023). While Martinelli's approach effectively facilitates the merging of maps when such references are available, it presents limitations in scenarios where robots operate in disconnected or unreferenced environments. In contrast, the proposed ORB-based algorithm does not necessitate standard references between the maps, allowing for greater flexibility and applicability across diverse environments.

Moreover, one of the key advantages of our approach is its ability to identify non-overlapping maps rather than forcing a potentially inaccurate overlap. This capability enhances the algorithm's robustness in dealing with disconnected environments and prevents the introduction of erroneous data into the final fused map.

However, it is important to acknowledge that while our method demonstrates advantages regarding reference independence and accurate identification of map relationships, it has yet to be validated in real-world scenarios. The reliance on simulated environments raises questions about the algorithm's robustness and effectiveness in dynamic and unpredictable settings typically encountered in practical applications.

3 ORB-BASED MAP FUSION

This section discusses the proposed methodology to address the challenges of map fusion and robot position transformation using the ORB algorithm. The proposed approach focuses on accurately detecting overlapping areas between maps and efficiently merging them while ensuring the correct transformation of the robots' positions in the final unified map. The following subsections outline the key components of our method, including map overlap detection, the fusion process, and position transformation.

3.1 Overview of the ORB Algorithm

The ORB (Oriented FAST and Rotated BRIEF) algorithm is a robust feature detection and description method widely used in computer vision. Developed to overcome the limitations of traditional feature detectors like SIFT and SURF, ORB is both computationally efficient and invariant to rotations and scales, making it ideal for real-time applications (Karami et al., 2017; Yan et al., 2023).

The first step in the ORB algorithm is key point detection using the FAST (Features from Accelerated Segment Test) method. FAST identifies corners in an image, which are potential key points. Each detected key point is then oriented based on the intensity centroid, ensuring that the features remain consistent under rotation (Karami et al., 2017; Yan et al., 2023).

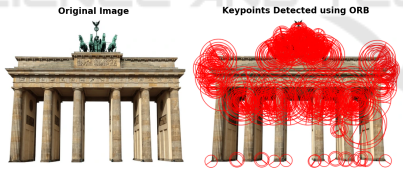

Figure 1: Keypoints detected by the ORB algorithm overlayed on the original image.

Once the keypoints are detected, the ORB algorithm describes these points using the BRIEF descriptor, which compares the intensity of pixels around the keypoint. This results in a binary string representing the key point's feature, enabling rapid matching between different images (Wu, 2023).

The benefits of using ORB include its invariance to rotation and scale, allowing features to be recognized regardless of the image's orientation or size. Additionally, the algorithm offers high computational efficiency compared to other methods, enabling real-time operations. As illustrated in Table 1, ORB operates significantly faster than SIFT and SURF, with execution times of just 0.02 seconds compared to 0.25 seconds for SIFT and 0.08 seconds for SURF, while achieving a match rate that is close to its counterparts (Karami et al., 2017).

Furthermore, the results in Table 2 highlight ORB's robustness against noise, achieving a match rate of 54.48%, slightly higher than SIFT's 53.8% which demonstrates ORB's ability to maintain performance even in challenging conditions. Additionally, ORB shows its effectiveness in varied scenarios, as indicated in Table 3, where it maintains competitive matching rates across different rotation angles. This combination of speed and accuracy makes ORB an ideal choice for applications requiring reliable performance under diverse and complex conditions (Karami et al., 2017; Cong, 2024).

Table 1: Results of comparing the image with its scaled version. *Reproduced from Karami et al. (2017).* (Karami et al., 2017)

Algorithm	Time(s)	Kpnts1	Kpnts2	Matches	Match Rate (%)
SIFT	0.25	248	1210	232	31.8
SURF	0.08	162	581	136	36.6
ORB	0.02	261	471	181	49.5

Table 2: Results of the image matching by adding 30% of salt and pepper noise. *Reproduced from Karami et al. (2017).* (Karami et al., 2017)

Algorithm	Time (s)	Kpnts1	Kpnts2	Matches	Match Rate (%)
SIFT	0.115	248	242	132	53.8
SURF	0.059	162	385	108	39.48
ORB	0.027	261	308	155	54.48

Table 3: Matching rate versus the rotation angle. *Reproduced from Karami et al. (2017).* (Karami et al., 2017)

Angle	0	45	90	135	180
SIFT	100%	65%	93%	67%	92%
SURF	99%	51%	99%	52%	96%
ORB	100%	46%	97%	46%	100%

In multirobot mapping scenarios, ORB facilitates effective map fusion by providing robust feature matching between disconnected maps. This capability of detecting and describing features in real time can enhance the efficiency of collaborative robotic systems (Singéis, 2024).

In conclusion, the choice of the ORB algorithm is fundamental for the proposed map fusion strategy, as it ensures accurate detection and description of keypoints while maintaining computational efficiency. Furthermore, ORB excels in handling the dynamic nature of map sizes, essential in robotics applications where varied and complex environments may be explored (Cong, 2024).

3.2 Map Generation and Patch Extraction

In this work, an algorithm was developed for the dynamic and random generation of structured maps, ensuring that each created map is unique. The function

`generate_structured_map(size)` generates a map represented as a 2D matrix of dimension $size \times size$, where the value zero (0) represents free spaces and the value one (1) represents obstacles. The default size of the map is 550×550 pixels.

The map creation involves adding various geometric shapes, such as rectangles, circles, and ellipses, using auxiliary functions that allow the inclusion of varied elements in the environment. Below, we describe some of these functions and the associated calculations:

- `add_rectangle(x, y, width, height, angle)`: This function inserts rectangles into the map, potentially applying a rotation according to the specified angle. The coordinates of the rectangle's vertices are calculated from (x,y), where x and y represent the position of the top-left corner. The dimensions of the rectangle are given by the parameters *width* and *height*. The calculation of the rotated coordinates is done using the rotation matrix:

$$R = \begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix}$$

  where θ is the rotation angle.

- `add_circle(x, y, radius)`: For the insertion of a circle, the function calculates the points that satisfy the equation, where (x,y) are the coordinates of the center of the circle and (X,Y) are the coordinates of any point within the circle:

$$(X-x)^2 + (Y-y)^2 \leq \text{radius}^2$$

- `add_ellipse(x, y, r1, r2, angle)`: With this function, ellipses can be added to the map. The equation of the ellipse is given by:

$$\frac{(X-x)^2}{r1^2} + \frac{(Y-y)^2}{r2^2} \leq 1$$

  where (x,y) represents the center of the ellipse and $r1$ and $r2$ are the radii of the ellipse along the X and Y axes, respectively.

Randomness is a crucial aspect of the algorithm. The function `add_large_areas()` creates a non-fixed number of large areas, where the size is randomly determined within a variable range.

The function `add_corridors()` generates a non-fixed number of corridors, where the width of the corridor is also a random value, and the length is equal to the size of the map. The starting coordinates for these corridors are selected randomly, ensuring a varied distribution of obstacles.

The function `extract_patch(map, patch_size)` is responsible for extracting patches from the generated map. The patch is randomly rotated, and the coordinates for extraction are calculated using the function `get_random_patch_coordinates(map_size, patch_size)`, which ensures that the starting coordinates (x,y) of the patch are within the map limits. The coordinates are obtained through:

$$x_{start} = \text{random.randint}(0, map\_size - patch\_size[0])$$
$$y_{start} = \text{random.randint}(0, map\_size - patch\_size[1])$$

This ensures that the extracted patch does not exceed the dimensions of the map. An example of generated map and patches is presented in Figure 2

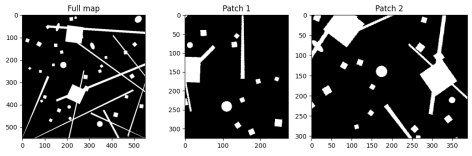

Figure 2: Example of generated map and patches.

The algorithm for generating structured maps and extracting patches provides a diverse and dynamic testing environment, allowing for practical and comprehensive experiments in map fusion and pattern recognition.

3.3 Map Fusion Process

The map fusion process is crucial for integrating information from different sources, especially in scenarios where maps may have varying sizes and resolutions (Martinelli et al., 2023). Fusion begins with detecting and describing features in the maps using algorithms such as ORB (Oriented FAST and Rotated BRIEF). Once key points and their descriptors are extracted, the next step is to find matches between the descriptors of the two maps. Using BFMatcher, the correspondences that indicate which features of one map correspond to features of the other are identified. An example can be seen in Figures 3 and 4.

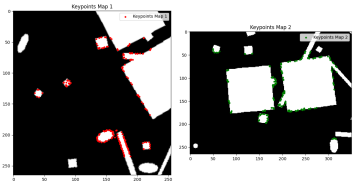

Figure 3: Keypoints extracted from a pair of maps.

After identifying the correspondences, it is necessary to calculate the transformation that aligns the maps. This transformation may include translation and rotation and is obtained through the function `cv2.estimateAffinePartial2D`, which uses

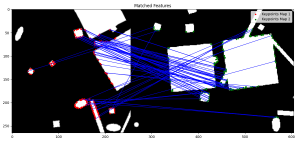

Figure 4: Lines illustrating similarities on connections between the maps, represented side-by-side.

the RANSAC method to minimize the impact of erroneous correspondences. The resulting transformation is described by:

$$\begin{pmatrix} 1 & 0 & dx \\ 0 & 1 & dy \end{pmatrix}$$

where dx and dy are the translations along the x and y axes, respectively. The rotation angle θ is calculated from the parameters of the transformation matrix. For example, if the obtained transformation matrix is:

$$M = \begin{pmatrix} \cos(\theta) & -\sin(\theta) & dx \\ \sin(\theta) & \cos(\theta) & dy \end{pmatrix}$$

the angle can be calculated as:

$$\theta = \arctan\left(\frac{M[1,0]}{M[0,0]}\right)$$

Here, the notation $M[i, j]$ refers to the element in the i-th row and j-th column of matrix M, following zero-based indexing conventions common in programming languages such as Python.

Once the transformation is determined, the next step is to apply the rotation to the second map and overlay the two maps using the function `overlay_maps`. The merged map for the maps on the Figure 3 can be seen in Figure 5

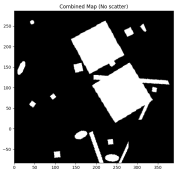

Figure 5: Final result of the map merging process.

Finally, considering the scalability of the process, the fusion should be capable of handling maps of different sizes. For this, interpolation techniques and scaling adjustments can be applied to ensure the information is correctly combined, even when the maps are captured at different resolutions or dimensions. This modular approach makes map fusion scalable and robust, making it applicable to a wide range of scenarios, from robotics applications to geospatial mapping.

3.4 Overlap Precision Detection

Before applying the transformation and overlaying the maps, evaluating the accuracy of the fusion performed is essential. For this, we calculate the mean absolute error between the overlapping areas of the maps, which provides a quantitative measure of the quality of the fusion. The following formula defines the error:

$$\text{Error} = \frac{1}{N}\sum_{i=1}^{N} |I_1[i] - I_2[i]|$$

where I_1 and I_2 are the pixel values in the overlapping areas, and N is the total number of pixels in the overlap region. A low mean error indicates that the maps have been successfully fused and that the information is aligned accurately.

This error metric is used purely for evaluating the quality of the alignment after the transformation is estimated, not for optimization. The Mean Absolute Error was chosen for its straightforward interpretation in occupancy grids, where it directly represents the average difference in pixel values, providing a clear and computationally inexpensive measure of fusion quality.

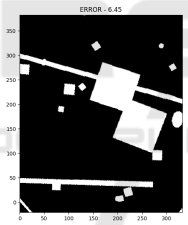

Figure 6: Good alignment with a low error of 6.43 (mean pixel difference).

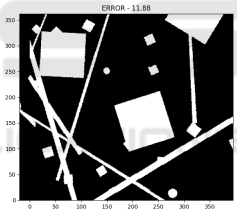

Figure 7: Minor misalignment with an error of 11.88.

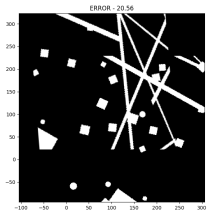

Figure 8: Noticeable misalignment with an error of 20.56, nearing the discard threshold.

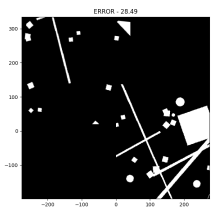

Figure 9: Poor alignment with a high error of 28.49, indicating a failed fusion.

To better illustrate the impact of the error metric, Figures 6, 7, 8, and 9 show examples of map overlays with different mean absolute error values. These figures provide visual insight into how the increasing error affects the accuracy of the fusion:

- In **Figure 6**, the error is relatively low (6.43), indicating that the maps have been well aligned, and the overlap area shows minimal discrepancies between the pixel values.
- As the error increases, as seen in **Figure 7** (error = 11.88), slight misalignments between the maps begin to appear. While the fusion is still somewhat accurate, the overlapping areas exhibit minor differences, which remain nearly imperceptible to the human eye.
- In **Figure 8** (error = 20.56), the discrepancies become more noticeable and start to affect the precision of the fusion significantly. At this point, the error reaches a value where the alignment is no longer reliable, and the map should be considered for discarding.
- Finally, in **Figure 9** (error = 28.49), the misalignment becomes more pronounced, with substantial differences between the overlapping areas. From this point onwards, the error reaches a threshold where the maps must be discarded due to poor alignment and unreliable fusion results.

Thus, the error remains nearly imperceptible up to around 20. Beyond this threshold, the fusion quality deteriorates significantly. This provides a clear metric for determining when to consider or discard maps based on their mean absolute error.

In addition to the mean absolute error, feature detection and the quality of the matches between the descriptors are crucial for the success of the fusion. The effectiveness of the matching algorithm, such as BF-Matcher, and the precision in identifying features directly influence the quality of the calculated transformation. If a significant number of erroneous matches are identified, it is possible that the fusion result may not be satisfactory, even if the mean absolute error is low.

To enhance the robustness of the fusion process, additional techniques are employed, such as cross-validation of matches and the application of filters to eliminate outliers. These methods ensure that only the most reliable matches are included in the final fusion. This comprehensive approach improves the accuracy of the map fusion and increases the system's resilience to noise and variations in the input data.

3.5 Robot Position Transformation

When merging two maps from different robots, it is essential to accurately transform the robot positions to the unified map (Martinelli et al., 2023). This process ensures that each robot's position is correctly placed relative to the newly aligned map. In this work, the transformation of robot positions is achieved by using the same homography matrix used for map alignment.

Given a robot's initial position (x_r, y_r) in its local map, the transformed position in the unified map (x'_r, y'_r) is calculated by applying the homography matrix:

$$\begin{bmatrix} x'_r \\ y'_r \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_r \\ y_r \\ 1 \end{bmatrix}$$

This transformation ensures that the robot's position in the unified map reflects the correct orientation and scale of the newly fused environment. An example can be seen in Figure 10, with robots being represented by the colored dots.

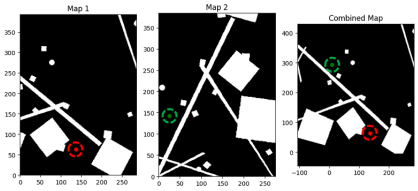

Figure 10: Robots positions transformation.

Ensuring the accurate transformation of robot positions is critical for several reasons. First, it allows for precise coordination between the robots within the shared environment, enabling them to communicate and make decisions based on their relative locations (Wang et al., 2023). Furthermore, in navigation and obstacle avoidance tasks, having correct position information is crucial to avoid collisions and plan efficient paths (Martinelli et al., 2023). Lastly, this process guarantees that any subsequent actions or updates in the environment (placing landmarks or interacting with objects) are executed in the correct spatial context, preserving the integrity of the merged map.

3.6 Computational Efficiency

The computational efficiency of the proposed map merging algorithm is driven by its ability to optimize the processing of large-scale maps. Despite processing the entire map initially, the algorithm is designed to reduce overall computation time by halting the process when certain conditions indicate that the resulting map alignment would be suboptimal. This feature allows the system to avoid unnecessary processing when there is no clear benefit to refining poor matches further.

One key factor contributing to this efficiency is the algorithm's ability to reject maps that do not meet quality thresholds before completing a full merge. In cases where the matching quality is insufficient, the algorithm terminates early, preventing further waste

of computational resources. This early termination mechanism ensures that computational power is only spent on map alignments, likely contributing to an accurate final map.

Additionally, by efficiently handling cases of poor overlap or misalignment, the algorithm ensures that high-quality maps are not replaced or corrupted by low-confidence merges. Figures 11 and 12 show examples of a successful merge versus a case where early termination was beneficial, visually emphasizing the effectiveness of the selective merging process. This selective merging process improves reliability and significantly reduces the time spent on unnecessary calculations, particularly valuable in real-time operations or when working with extensive map datasets.

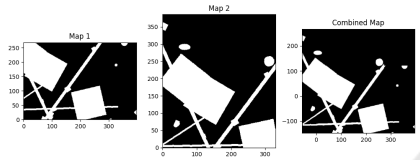

Figure 11: Succesful merge, with an error of 5.47.

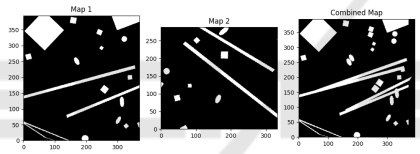

Figure 12: Unsuccesful merge, with an error of 33.78.

In summary, the algorithm's efficiency lies in its ability to streamline processing by avoiding the full computation of maps unlikely to improve the overall result. This targeted approach reduces the computational load while maintaining the integrity of well-matched sections, thus balancing speed and accuracy in large-scale map fusion tasks.

4 EVALUATION AND ANALYSIS

Two distinct experiments were conducted to assess the performance of the proposed map fusion algorithm based on ORB. The first experiment aims to evaluate the algorithm's ability to detect the quality of the map merging, particularly focusing on its accuracy in identifying false positives. This test will also inform future algorithm adaptations into a cyclical system for continuous operation and real-time map updates. The second experiment visually compares the results of the automatic map fusion performed by the algorithm with manually merged maps, providing a qualitative analysis of its performance.

4.1 Comparison of Automatic and Manual Merging

This experiment aims to evaluate the qualitative performance of the proposed ORB-based map fusion algorithm by comparing its results with those of manually merged maps. The goal is to visually assess how well the algorithm aligns the maps regarding structural consistency, continuity, and overall accuracy.

For this comparison, several pairs of maps were selected, each with varying levels of complexity, rotation, and translation. The chosen maps included features such as walls, corridors, and obstacles commonly found in robotic occupancy grids. Each map pair was processed using two methods:

1. **Automatic Fusion:** The proposed algorithm was applied to merge the maps without manual intervention. The algorithm used ORB feature matching to detect key points and estimate the transformation required for alignment.
2. **Manual Fusion:** A single human operator conducted a manual fusion process in parallel. The manual fusion involved identifying overlapping regions between the two maps and aligning them by visually adjusting their rotation and translation.

The results of both methods were displayed side by side to facilitate a direct comparison. Key qualitative aspects were analyzed visually, including:

- **Alignment Precision:** The visual degree to which the structural elements of the maps, such as walls and obstacles, appeared to be correctly aligned.
- **Continuity of Features:** The smoothness and consistency of the merged map, particularly at the boundaries between the original maps, are assessed through visual inspection.
- **Visual Accuracy:** The merged map's overall coherence, focusing on preserving critical map features without distortion, evaluated based on visual assessment.

Figures 13 and 14 show the visual outputs of the automatic and manual fusion, respectively. As observed in this example, the automatic fusion demonstrated the ability to understand the connection between the maps even with limited overlapping areas, indicating its robustness in handling sparse data. However, this particular example exhibited a slight imprecision due to the lack of information. This comparison is one of several tests conducted using various map pairs to evaluate the algorithm's performance under different scenarios.

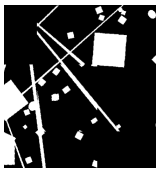

Figure 13: Automatic fusion.

Figure 14: Manual fusion.

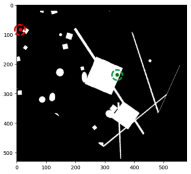

Figure 17: Final merged map with estimated robot positions.

4.2 Robot Positioning Accuracy Test

In this section, we evaluate the capability of the proposed map fusion algorithm to position robots within the generated map accurately. The positioning accuracy is crucial for effective navigation and operation of robotic systems in real-world environments. This test assesses how well the algorithm can align the merged maps with the actual positions of the robots based on randomly assigned coordinates.

The experimental setup involved several pairs of patches extracted from a larger map, with robot positions randomly assigned within each patch. The steps taken during the testing process were as follows:

1. **Patch Extraction:** Randomly extract two patches from a larger map, each containing a robot placed at a random position within the patch.
2. **Map Fusion:** Apply the proposed algorithm to merge the extracted patches into a single map.
3. **Position Estimation:** After merging, estimate the positions of the robots within the fused map based on the algorithm's output.
4. **Visual Assessment:** Compare the estimated positions visually in the merged map to evaluate the algorithm's accuracy.

The results of the robot positioning accuracy test are illustrated in Figures 15, 16, and 17. The first two figures display the individual patches with randomly assigned robots. In contrast, the third figure shows the final merged map and the calculated positions of both robots. The accuracy of the algorithm can be visually assessed in the merged map. In all conducted tests, the algorithm demonstrated exceptional performance in accurately transforming the positions of the robots within the merged maps.

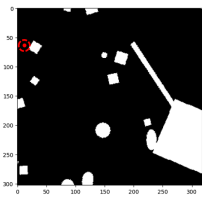

Figure 15: Patch 1 with robot in red.

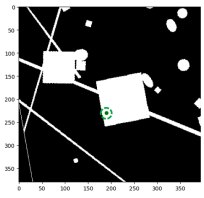

Figure 16: Patch 2 with robot in green.

5 CONCLUSION

This study presented an algorithm for map fusion utilizing ORB feature matching, specifically designed to merge robotic occupancy grids. The algorithm demonstrates significant potential for improving the reliability and efficiency of robotic navigation systems.

One of the key advantages of the proposed algorithm is its capability to operate with maps of varying sizes and orientations effectively. This flexibility is particularly beneficial in multi-robot systems, where different robots may generate maps under different conditions. By accurately merging these maps, the algorithm enables robots to collaborate and navigate within a unified environment representation, enhancing their ability to perform coordinated tasks.

Moreover, the algorithm can estimate robot positions within the fused maps, even with limited overlapping areas. This positioning accuracy is crucial for runtime decision-making and navigation in dynamic environments.

The efficiency of the ORB feature matching method further distinguishes this algorithm from conventional approaches. Its computational speed allows for real-time processing, making it suitable for dynamic applications where timely responses are essential.

Overall, the results indicate that the proposed algorithm is a valuable tool for enhancing the capabilities of autonomous robots. The successful integration of maps is vital for enabling these systems to operate effectively in complex environments.

All experimental data, including generated maps and test results, can be found in a public repository [1]. This repository is a resource for further exploring and validating the proposed method in robotic mapping and navigation tasks.

Future work on the proposed map fusion algorithm could focus on several enhancements and practical applications. One significant improvement would be to adapt the algorithm to handle occupancy grids with values ranging from -1 to 255 rather than

[1] https://github.com/LucasZick/map_fusion

being limited to binary values of 0 and 1. This extension would allow the algorithm to incorporate more nuanced information about the environment, such as varying degrees of uncertainty or likelihood of obstacles.

Additionally, it would be beneficial to implement the algorithm on real robotic systems operating in real-world environments. Testing in practical settings would provide valuable insights into the algorithm's performance under dynamic conditions and with actual sensor data. This approach could lead to further refinements and optimizations, enhancing the algorithm's robustness and applicability in diverse robotic applications.

Exploring these avenues would contribute to developing more sophisticated multi-robot systems, ultimately improving their efficiency and effectiveness in navigating complex environments.

ACKNOWLEDGEMENTS

The project is supported by the National Council for Scientific and Technological Development (CNPq) under grant number 407984/2022-4; the Fund for Scientific and Technological Development (FNDCT); the Ministry of Science, Technology and Innovations (MCTI) of Brazil; Brazilian Federal Agency for Support and Evaluation of Graduate Education (CAPES); the Araucaria Foundation; the General Superintendence of Science, Technology and Higher Education (SETI); and NAPI Robotics.

REFERENCES

Ahmed, M. F., Frémont, V., and Fantoni, I. (2024). Active collaborative visual slam exploiting orb features. *arXiv preprint arXiv:2407.05453*.

Ahmed Jalil, B. and Kasim Ibraheem, I. (2023). Multi-robot slam using fast lidar odometry and mapping. *Designs*, 7(5):110.

Arai, T., Pagello, E., Parker, L. E., et al. (2002). Advances in multi-robot systems. *IEEE Transactions on robotics and automation*, 18(5):655–661.

Chakraa, H., Guérin, F., Leclercq, E., and Lefebvre, D. (2023). Optimization techniques for multi-robot task allocation problems: Review on the state-of-the-art. *Robotics and Autonomous Systems*, page 104492.

Cong, Y. (2024). Image stitching technology for police drones using an improved image registration method incorporating orb algorithm. *Informatica*, 48(2).

Gautam, A. and Mohan, S. (2012). A review of research in multi-robot systems. In *2012 IEEE 7th international conference on industrial and information systems (ICIIS)*, pages 1–5. IEEE.

Karami, E., Prasad, S., and Shehata, M. (2017). Image matching using sift, surf, brief and orb: performance comparison for distorted images. *arXiv preprint arXiv:1710.02726*.

Martinelli, D., Kalempa, V. C., and de Oliveira, A. S. (2023). Map merge and accurate localization in multi-robot systems in real environments. In *Iberian Robotics conference*, pages 26–38. Springer.

Mukhopadhyay, S., Umari, H., and Koirala, K. (2023). Multi-robot map exploration based on multiple rapidly-exploring randomized trees. *SN Computer Science*, 5(1):31.

Sabry, E. S., Elagooz, S., El-Samie, F. E. A., El-Bahnasawy, N. A., and El-Banby, G. M. (2024). Sift and orb performance assessment for object identification in different test cases. *Journal of Optics*, 53(3):1695–1708.

Singéis, R. (2024). Check for performance analysis of orb-slam in foggy environments rita singéis, sedat dogru (), and lino marques institute of systems and robotics, department of electrical and computer. In *Robot 2023: Sixth Iberian Robotics Conference: Advances in Robotics, Volume 1*, volume 1, page 209. Springer Nature.

Stathoulopoulos, N., Koval, A., Agha-mohammadi, A.-a., and Nikolakopoulos, G. (2023). Frame: Fast and robust autonomous 3d point cloud map-merging for egocentric multi-robot exploration. In *2023 IEEE international conference on robotics and automation (ICRA)*, pages 3483–3489. IEEE.

Vaščák, J. and Herich, D. (2023). Map merging for multi-robotic applications. In *2023 IEEE 21st World Symposium on Applied Machine Intelligence and Informatics (SAMI)*, pages 000021–000026. IEEE.

Wang, M., Cong, M., Du, Y., Liu, D., and Tian, X. (2023). Multi-robot raster map fusion without initial relative position. *Robotic Intelligence and Automation*, 43(5):498–508.

Wu, K. (2023). Creating panoramic images using orb feature detection and ransac-based image alignment. *Advances in Computer and Communication*, 4(4):220–224.

Yan, H., Wang, J., and Zhang, P. (2023). Application of optimized orb algorithm in design ar augmented reality technology based on visualization. *Mathematics*, 11(6):1278.