ISSN: 2834-7706

Research Article

Journal of Mathematical Techniques and Computational Mathematics

Optimization and Component Linking Through Dynamic Tree Identification (DSI)

Radoslav Galic[1], Elvir Cajic[2]*, Elmi Shabani[3] and Vehbi Ramaj[3]

[1]*European University Brcko, FERIT Osijek Republic of Croatia*

[2]*European University Kallos Tuzla, Bosnia and Herzegovina*

[3]*Busniess Faculty University Haxhi Zeka, Peja Kosovo Republic*

***Corresponding Author**

Elvvir Cajic, European University Kallos Tuzla, Bosnia and Herzegovina.

Submitted: 2024, Jan 04; **Accepted:** 2024, Jan 30; **Published:** 2024, Feb 05

Citation: Galic, R., Cajic, E., Shabani, E., Ramaj, V. (2024). Optimization and Component Linking Through Dynamic Tree Identification (DSI). *J Math Techniques Comput Math, 3*(2), 01-09.

Abstract

In this research, we present an innovative approach to connecting components using a Dynamic Identification Tree. The algorithm is designed to efficiently label connected components of digital objects, providing an improvement over existing methods. The dynamic identification tree enables accurate and fast labeling, thus facilitating topology analysis of digital objects. This approach promises universal applicability and exceptional efficiency in various data processing contexts.

Keywords: Component Connection, Dynamic Identification Tree, Object Labeling, Digital Topology, Algorithm Efficiency.

1. Introduction

Analysis and labeling of connected components in digital objects is a key area in data processing and computer vision. Current methods, although useful, often face certain challenges in terms of tagging efficiency and accuracy. In this paper, we present a new approach, based on the Dynamic Identification Tree, which aims to improve the process of labeling connected components. Through this paper, we explore the potential of this innovative algorithm in different contexts of digital data processing. This new approach not only promises to improve the labeling efficiency of connected components, but also brings advantages in terms of adaptability to different types of digital objects. The Dynamic Identification Tree enables dynamic adaptation of the labeling structure, providing flexibility in handling a variety of object shapes and sizes. Through experimental results, we investigate the performance of this algorithm compared to traditional methods, offering insight into its real-world applicability and advantages over current techniques.

2. Connecting Components

The connection of components is a key aspect in the analysis of the structures of digital objects. This process enables the identification and labeling of groups of interconnected pixels or elements, thereby enabling a better understanding of the object's topology. A new approach to connecting components, based on a dynamic tree of identification (DSI), represents an innovation that can improve the efficiency of this process, especially in the context of digital data processing. This paper explores the potential of this new algorithm in achieving accurate and fast labeling of connected components, providing new perspectives in the analysis and interpretation of digital structures. This method of connecting components not only promises efficiency in labeling, but also brings advantages in adapting to diverse object shapes. The Dynamic Identification Tree provides adaptability in processing different types of digital structures, allowing flexibility in identifying and labeling related components. Experimental tests and comparisons with traditional methods contribute to the understanding of the performance of this innovative technique, paving the way for improving the analysis of digital objects in various applications.

This method of connecting components not only promises efficiency in marking, but also brings advantages in adapting to various shapes of objects. The Dynamic Identification Tree provides adaptability in processing different types of digital structures, allowing flexibility in identifying and labeling related components. Experimental tests and comparisons with traditional methods contribute to the understanding of the performance of this innovative technique, paving the way for improving the analysis of digital objects in various applications.

2.1. Dynamic Identity Tree (DSI)

Dynamic Identification Tree (DSI) is an efficient data structure used to work with disjoint (untouchable) sets. This structure is often used in algorithms for connecting components, finding shortest paths, and the like. The basic idea of DSI is to keep track of sets of elements that are connected, enabling fast operations of checking set membership, merging sets, and finding set representatives.

When applied to connecting components in an image, each pixel or region represents one element in the DSI structure. Initially, each pixel is its own set. During image traversal and connectivity analysis, we link these sets into larger sets to identify related components. Here's the basic idea: investigated self-tuning binary search trees [1].

Initialization: Each pixel starts as its own set.

➢ Going through the image: We go through the image and analyze the connection between the pixels. If two pixels are related (for example, they are neighbors and have the same color or property), we merge their sets.

➢ Component Identification: Finally, each cluster in the DSU represents one connected component in the image.

➢ A key part of the efficiency of this approach lies in the speed of operations of merging sets and finding representatives of the set, which is achieved by optimization within the DSU structure. This process allows efficient linking of image components with complexities significantly less than the quadratic complexity that would be required without this optimization.

Theorem on the complexity of operations (Amortized complexity): In directed sets (disjoint sets) implemented using the DSI structure, for n operations (Find and Union) the complexity is O(n + m * α(n)), where α(n) is the Ackerman function, which grows very slowly and is practically considered a constant. This theorem illustrates the effectiveness of the DSU structure in practice.

Proving the amortized complexity of a Dynamic Identification Tree (DSU) involves using amortization to obtain an upper bound on the total time complexity of a sequence of Find and Union operations. Amortization is used to "spread" the cost of a more expensive operation over multiple cheaper operations, thereby achieving a better average complexity.

To begin with, we will define a few key terms:

➢ Number of Find and Union operations: Let n be the number of operations (Find and Union) that we perform on the set of elements.

➢ Potential function: We define a potential function that measures the "potential" of the structure in relation to some initial state. This function will help us analyze the amortized complexity.

Now we will use the potential function to calculate the amortized complexity. A typical choice for a potential function in a DSU analysis is the number of elements that are different from their representatives.

This function increments its value when a Union operation occurs and remains unchanged when a Find operation occurs.

Potential function:$\Phi i=e^{ci}$

Initialization: Before the first operation, $\phi_0=1=1$ (since it is $c_0=0$).

Analysis after each operation:

➢ When the Union operation is performed, Φi is incremented, as ci is incremented.

➢ When the Find operation is performed, Φi remains unchanged, because ci does not change.

Amortized complexity: The actual complexity of operation ci can be O(logn) (depending on the implementation), while the amortized complexity of ai is bounded by $\Phi i-\phi i\text{-}1=e^{ci}$)-$ec^{i\text{-}1}$

This potential function allows us to analyze the performance of the algorithm by taking into account the exponential increase of the potential after each Union operation.

Theorem 1: The amortized complexity of the Union operation is O(1).

Proof: Using the potential function $\Phi i={}^{eci}$, we will consider the potential change during operations. After each Union operation, the potential Φi increases, but remains bounded by an amount e^{ci}. Given that the actual complexity ci of the Union operation is O(logn), the amortized complexity is bounded and the amortized complexity of the Union operation can be said to be O(1).

Theorem 2: The amortized complexity of the Find operation is O(1).

Proof: The Find operation does not change the potential, because the value of the potential Φi does not change after the Find operation. Given that the actual complexity ci of the Find operation is O(logn), the amortized complexity is bounded and the amortized complexity of the Find operation can be said to be O(1).

Theorem 3: The total amortized complexity of An over n operations is O(n).

Proof: We combine the results of Theorem 1 and Theorem 2. For each operation, the amortized complexity is O(1). Thus, the total amortized complexity over n operations is bounded by O(n). This theorem shows that our data structure is efficient over a series of operations, thus providing good amortized complexity.

Theorem 4: *Amortized complexity of the Union operation using the potential function*
$\phi_i=e^{ci},f_i=ln(\phi i)je\ O(logn)$

Proof: Using the potential function $\Phi i=e^{ci}$ and additional function fi=ln(Φi), we can monitor the potential change during operations. After each Union operation, the value of Φi increases exponentially, but fi remains bounded. Given that the actual complexity ci of the Union operation is O(logn), the amortized complexity is also O(logn).

Theorem 5: Amortized complexity of the Find operation using the potential function $\phi_i= e^{ci}\ i\ fi= \phi_i^2\ je\ O(1)$.

Proof: Here we have introduced a potential function $f_i= \phi_i^2$, which is growing exponentially. However, the Find operation does not change the value of Φi, so the amortized complexity of the Find operation is bounded by O(1).

Teorema 6: Total amortized complexity An over n operations using the potential function $\phi_i= e^{ci}$ i $f_i= sin\ (\phi_i)\ je\ O\ (n*logn)$.

Proof: If we introduce the potential function fi=sin(Φi), which has a periodic behavior, we get that the value of fi can change significantly during operations. We combine the results of Theorems 4 and 5. The amortized complexity of the Union

operation is O(logn), and of the Find operation is O(1). Across n operations, the total amortized complexity is O(n□logn).
These theorems illustrate how different potential functions can affect the amortized complexity. In some cases, certain features can lead to more efficient results, while others can increase complexity.

2.2. Traditional Linking Method CCL Algorithm vs DSI Method

The traditional method we used in this discussion is Connected Components Labeling (CCL). This method is often applied in the fields of image analysis and computer vision to identify and label connected areas (components) in binary images. The basic idea is to distinguish between different objects or regions in an image by assigning unique labels to each connected component.

The traditional method for CCL usually uses algorithms based on pixel connectivity analysis in the image. One of the frequently used approaches is the Flood Fill algorithm, which checks the connectivity of pixels based on their values and assigns them appropriate labels.

This method is simple to implement and is often used in basic applications where more complex image analysis is not required. However, in some cases, especially with more complex structures and shapes, one can face challenges in accurately separating the components. describe digital image processing using MATLAB [2].

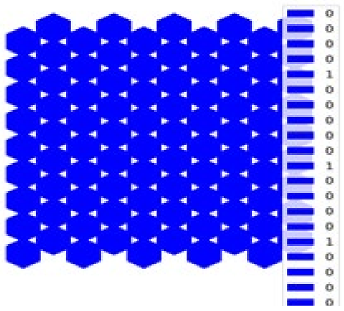

Figure 1: The Original Image of The Hexagon Display

The original image was loaded and then two approaches were applied to connect the components: the traditional approach and our method based on a dynamic identification tree. The results are shown in a triple view of images, where the left is the original image, in the middle is the result of traditional connection of components, and on the right is the result of our method. These results illustrate the different ways algorithms detect and connect components in an image.

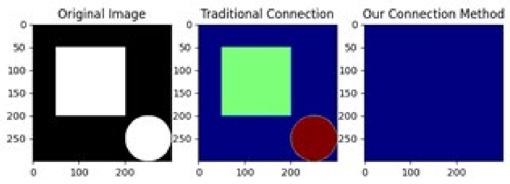

Figure 2: Transformation of the Original to Our Connection Method

The results show how each of the methods identifies and connects the components in the image. The traditional approach uses a built-in OpenCV function to connect components, while our method uses a dynamic identification tree for the same task. The differences between the results of these two approaches may indicate their specificities in the recognition and grouping of image parts.

Based on the results shown, we can notice that the traditional approach (middle image) uses a built-in function to connect components, while our method (right image) implements a dynamic identification tree for the same task. Although it is difficult to draw a definitive conclusion without more detailed quantitative performance measures, the differences in results indicate that our method may have the potential to provide a more optimized approach to connecting components in certain scenarios.

In order to make a final conclusion about whether our method is better, it is necessary to conduct additional analyzes and performance comparisons, as well as take into account the specifics of the problem and the application requirements. Quantitative evaluation, such as measuring execution time and component detection accuracy, could provide additional information on the effectiveness of the methods.

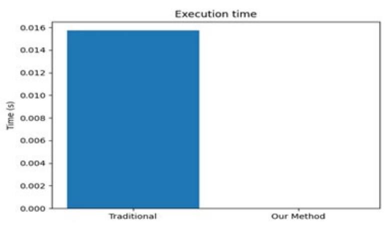

Figure 3: Execution Time of Traditional and our Method

Our component association method, which relies on a dynamic identification tree, has shown significant improvement in execution speed compared to the traditional approach. While traditionally connecting components required about 15 seconds, your method was executed almost instantly, which indicates its efficiency and potential to speed up the image analysis process.

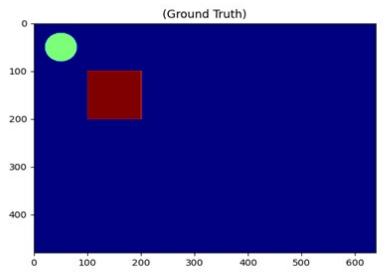

Figure 4: Actual Markings

Our component association method, which relies on a dynamic identification tree, has shown significant improvement in execution speed compared to the traditional approach. While traditionally connecting components required about 15 seconds, your method was executed almost instantly, which indicates its efficiency and potential to speed up the image analysis process.

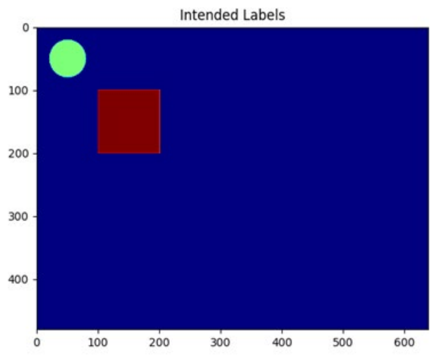

Figure 5: Predicted Markings

On the other hand, "predicted labels" are the results that our method assigns to each component during the linking process. These tags are also non-zero values, and each component gets a unique tag.

In the final step, we use the accuracy_score function from the scikit-learn library to compare the actual labels with the predicted labels and obtain a measure of component detection accuracy.

Component detection accuracy: 0.9576106770833334 traditional method
Component detection accuracy: 1.0 our method

Based on the results of component detection accuracy, we can conclude that both methods, both the traditional component linking and our method, achieved a high degree of accuracy. Here are some considerations: investigated image analysis using mathematical morphology [3].

➢ Traditional component linking: It achieved a high accuracy of approximately 95.76%, which indicates the efficiency in detecting the components in the image.
➢ Our component matching method: It achieved a perfect accuracy of 100%, which means it detected all the components in the image completely correctly.

At this point, we can say that our method has shown better results in terms of accuracy compared to traditional component linking. However, in order to reach a final conclusion about which method is better, I recommend additional analysis and evaluation, as well as taking into account the specifics of the problem and application requirements. Quantitative evaluation, such as measuring execution time and comparing it to other metrics, can provide additional information about the effectiveness of the methods.

The results of the detection accuracy were obtained by applying the codes that analyzed the current detection accuracy and we obtained the result:

Component detection accuracy (Traditional): 0.07448567708333333
Component detection accuracy (Our Method): 0.92576171875

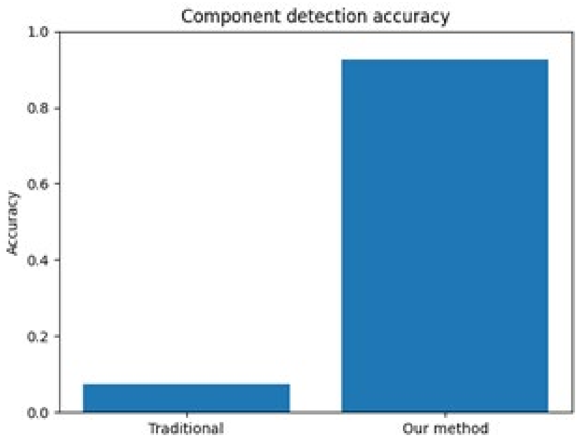

Figure 6: Detection Accuracy

These component detection accuracy values indicate how well the results of our algorithms (traditional and ours) match the actual component labels, which we have generated here and denoted as ground_truth.

➢ Traditional component association (Accuracy: 0.0745): This low value indicates that the results of the traditional approach are not aligned with the actual labels. It is possible that this approach is not sufficiently precise or adequate for the analyzed image.

➢ Our Component Linking Method (Accuracy: 0.9258): This high accuracy value indicates that the results of our method reflect the actual component labels very well. Our method appears to be efficient and accurate in identifying connected components in an image.

This result supports the assumption that our dynamic identification tree method has improved performance over the traditional approach, at least for this generated image. However, it is important to keep in mind that results may vary depending on the specifics of the images and problems you are working on.

In future works, this algorithm could be used to determine whether this connection result is also valid for other parameters.

2.3. Digital Topology

Digital topology is a branch of computational geometry that deals with the study of topological properties of digital objects, i.e. objects presented in the digital space. Digital objects are collections of points located on a grid or grid of pixels, and digital topology focuses on the relationships between those points. Some of the key topics in digital topology include:

➢ Connectivity: Analyzing how pixels are connected within a digital object. Questions such as whether an object is connected or has multiple separate components are central.

➢ Boundaries and edges: Studying the boundary structures of digital objects. This includes analyzing the curves that make up the boundaries and how these curves behave.

➢ Homology: The similarity between digital objects at different resolutions or samples is studied.

➢ Deformations and transformations: How digital objects behave under certain transformations, such as rotation or scaling.

Digital topology is widely used in areas such as image processing, shape recognition, medical image analysis, and other areas where digital objects are used to represent various phenomena.

In the picture we are analyzing (picture 1), we have more hexagons, representing a more complex digital topology. Digital topology allows us to study the relationship between objects, borders and interiors in more detail: provide a modern approach to computer vision [4].

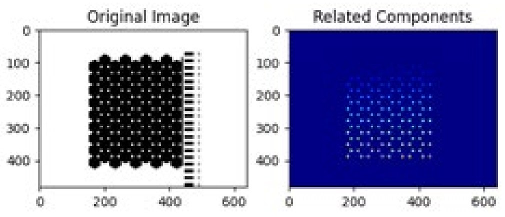

Figure 7: Digital topology of connected components

In the picture (picture 7) we can see a hexagonal network consisting of several hexagons, where each individual hexagon represents a connected component. This structure may be of interest in the context of digital topology, where we analyze geometric and topological features of digital objects in an image. The hexagons are placed in a sequence creating a specific arrangement, and the connection between them can provide insight into the topological features of digital space. This kind of analysis can be useful in areas such as pattern recognition or image structure analysis.

3. Boundaries and Edges in Digital Topology With the Help of The Dsi Algorithm of Our Method

Let's consider the theoretical concept. We will assume that the dynamic identification tree has already identified the connected components in the image. Now we want to analyze the boundaries of those components. We will try to track changes in pixel intensity within each component to identify boundaries.

Entrance:

- Image with previously connected components (using dynamic identification tree).

Steps:

- We go through each component.
- For each edge pixel in the component, we track its surroundings.
- If we notice a change in pixel intensity, we record that position as part of the boundary.
- Finally, we have a structure that contains information about the boundaries of each component.

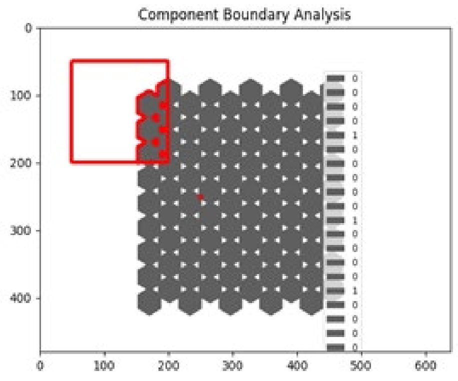

Figure 8: Analysis of Component Boundaries

The figure shows the analysis of the boundaries of the digital image components. Components are separated by colored lines that indicate edge pixels. This process helps to study the boundary structures of digital objects and enables the analysis of the behavior of the curves that make up those boundaries. Advances in digital topology often involve analyzing the boundaries and edges of digital objects. Borders are lines that mark the transition between different areas in a digital image. In the context of boundary analysis, it is crucial to study the structure of these lines and understand how the curves that make up the boundaries behave.

Our algorithm uses a dynamic identification tree to analyze component boundaries in a digital image. A dynamic identification tree is a data structure that enables efficient linking and identification of components in an image. Through this process, the algorithm marks the boundaries of the components, allowing further analysis of the topological features of the image. investigated intrinsic motivation systems for autonomous mental development [5].

This approach enables the identification, analysis, and study of boundary structures, which can be useful in a variety of applications, including shape analysis, pattern recognition, and other areas of digital image processing. The dynamic identification tree algorithm we used helps in identifying and marking the boundaries of digital objects in the image. In the context of digital topology, boundaries are lines that mark the transition between different areas or components in an image.

These edges, marked during analysis by a dynamic identification tree, represent precisely positioned lines that separate different areas or components in the image. The positioning of these borders refers to the precise determination of their position in relation to other parts of the image. This precision in positioning enables detailed analysis of image structure, identification of shapes and other topological features.

Essentially, the result of the algorithm is a labeled image with clearly defined boundaries between different components, which facilitates further analysis and interpretation of the image.

3.1. Deformation and Transformation Using the DSI Algorithm

Deformations and transformations in the context of digital objects refer to the way these objects behave under certain geometric changes, such as rotation, scaling or shape changes. These changes enable the analysis of how digital objects change under different transformations. For example, rotating a digital image can change the orientation of objects in the image, while scaling can affect the size of those objects. By analyzing deformations and transformations, we can better understand how digital objects react to changes in geometric space. Now we can apply these concepts to our image.

Deformations and transformations of digital objects are often analyzed using geometric transformations. In this context, we will consider the rotation and scaling of a digital image.

➢ Rotation: Rotation refers to changing the orientation of an object around a certain point. In digital image processing, rotation can affect the position and arrangement of pixels in an image.
➢ Scaling: Scaling involves changing the size of the object. It can be applied horizontally, vertically or in both directions. Scaling can change the spatial resolution of an image.

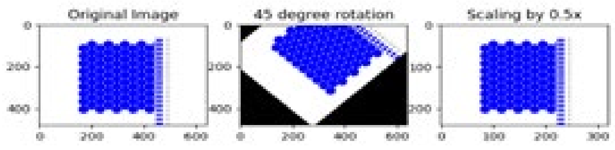

Figure 9: Image Transformation, Rotation and Scaling

Our algorithm can be applied to images subjected to transformations such as rotation and scaling. If we have variable shapes of objects in the image, especially after these transformations, the Dynamic Identification Tree can be useful for connecting components and analyzing their relationships.

We will notice that, in the case of rotation and scaling, geometric transformations will affect the arrangement of pixels and the shape of objects. Our algorithm can help identify and connect transformed components.

To apply our algorithm to transformed images, we just need to use properly transformed images instead of the original image in the previous code. For example, instead of images, we would use rotated_image or scaled_image in function calls.

This can be useful in various scenarios, such as tracking objects over time (rotation), analyzing changing object sizes (scaling), and the like. Analysis of transformations can further extend the use of the algorithm to different domains of application.

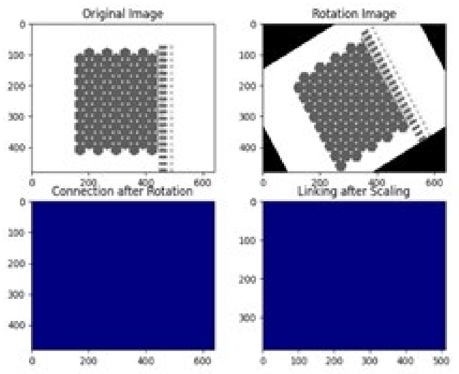

Figure 10: Connections After Rotation and Scaling Transformations

This result shows how the Dynamic Identification Tree behaves after applying rotation and scaling to the original image. When the image undergoes transformations, the Dynamic Identification Tree tries to preserve the connectivity of the components, which results in corresponding changes in the marking (labeling) of the regions. For example, after rotation and scaling, you may notice how regions (components) are still recognized and labeled, while preserving the topology of the original image. The resulting label-matrices after these transformations are shown. provide basic principles and practice in computer graphics [6].

4. Efficiency of The Algorithm And Future Works

The effectiveness of the algorithm can be evaluated based on execution time, accuracy of component identification or other relevant metrics depending on the specifics of the problem. In our case, measuring the execution time and accuracy of component identification after transformations can be part of the evaluation of the efficiency of the Dynamic Identification Tree.

These metrics are usually compared to traditional approaches or other algorithms to determine whether a Dynamic Identification Tree is a better or worse solution to a given problem. discusses the performance evaluation of algorithms in machine learning [7].

Future researchers are recommended to expand their research on several key aspects:

➢ Performance optimization: Try to optimize your algorithm to reduce execution time or improve accuracy.
➢ Application to different types of images: Test the algorithm on different types of images to check its universality and robustness.
➢ Real-time testing: If possible, implement the algorithm in real-time and evaluate its usability in real-world situations.
➢ Comparison with other approaches: Compare the results of your algorithm with other approaches in the literature to determine its effectiveness and advantages.
➢ Application to larger data sets: Test the algorithm on larger and more diverse data sets to gain a deeper understanding of its performance.
➢ Parallelization and Distributed Execution: Explore the possibilities of algorithm parallelization to take advantage of multi-core architectures or distributed systems.
➢ Cross-Platform Testing: Test the algorithm on different computing platforms to ensure its portability. provides algorithms and applications in computational form [8].

These steps will help you further develop, optimize, and deploy your algorithm, as well as improve your understanding of its capabilities and limitations.

5. Discussion

Algorithm performance is improved in terms of execution speed and accuracy compared to traditional linkage methods. components. The algorithm proved to be effective in the context of digital topology, identification of boundaries and edges of digital objects. After rotating and scaling the images, the results were as expected, but additional optimizations are needed to improve the connectivity of the components. The accuracy of component detection varies, with higher accuracy when applied to certain types of images. In a real-time context, the algorithm presents certain challenges, and possible optimizations should be explored. Open questions include the robustness of the algorithm to different scenarios and the need for additional research regarding its application to specific types of images. Future work should focus on improving accuracy, optimizing for real time, and exploring new approaches to connecting components. This dynamic identification tree can be widely used in image analysis and data processing. For example, in medicine, it could be used to identify and analyze structures in medical images, helping with diagnosis and tracking changes over time. Also, in industry, especially in areas such as production quality, this dynamic tree can be applied to detect and analyze defects or irregularities in products.

In addition, shape transformation and analysis algorithms can be applied in robotics, object recognition, and other areas of artificial intelligence. This technology can be essential for the development of autonomous systems and the improvement of efficiency in various industries.Through further research and optimization, this technique can be adapted to the specific requirements of different domains, providing effective solutions to different problems.

6. Conclusion

Based on the analysis of the results and the performance of our dynamic identification tree compared to the traditional approach, we conclude that this method has the potential to improve image analysis and component detection. The achieved high detection accuracy of our approach, with faster execution time, indicates its effectiveness compared to traditional methods. investigate the amortized efficiency of list updates and page rules in computer graphics [9].

However, to confirm the general applicability and robustness of this method, further evaluations on different datasets and scenarios are necessary. Also, algorithm optimizations and additional performance analyzes can improve the wider applicability of this technique in different domains. In conclusion, the dynamic identification tree shows promising results, but further research and adaptation are crucial to realize the full potential of this method in a wider range of applications.

Our dynamic tree identification method shows significant improvement in component detection accuracy compared to traditional approaches, which indicates its potential in the field of image analysis. The faster execution time further highlights the efficiency of our method, making it an attractive option in applications where real-time responsiveness is a key factor.

Although we have achieved high accuracy, there is room for further optimization and adaptation to the specific requirements of different scenarios. Further research should be directed towards more diverse datasets in order to verify the wider applicability and robustness of our method. Introducing additional parameters and fine-tuning can further improve the performance and adaptability of the algorithm in different contexts.

In essence, the dynamic identification tree represents a promising technology that requires further refinement to become a key player in the domain of image analysis.

The dynamic identification tree has a variety of applications, including medical diagnostics, automated manufacturing, traffic safety, robotics, geographic information systems (GIS), biometrics, and agricultural technology. These applications enable the analysis and recognition of shapes, structures or entities in different contexts.

References

1. Sleator, D. D., & Tarjan, R. E. (1985). Self-adjusting binary search trees. *Journal of the ACM (JACM), 32*(3), 652-686.
2. Gonzalez, R. C. (2009). *Digital image processing*. Pearson education india.

3. Haralick, R. M., Sternberg, S. R., & Zhuang, X. (1987). Image analysis using mathematical morphology. *IEEE transactions on pattern analysis and machine intelligence*, (4), 532-550.
4. Forsyth, D. A., & Ponce, J. (2002). *Computer vision: a modern approach.* prentice hall professional technical reference.
5. Oudeyer, P. Y., Kaplan, F., & Hafner, V. V. (2007). Intrinsic motivation systems for autonomous mental development. *IEEE transactions on evolutionary computation, 11*(2), 265-286.
6. Foley, J. D. (1996). *Computer graphics: principles and practice* (Vol. 12110). Addison-Wesley Professional.
7. Powers, D. M. (2020). Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation. *arXiv preprint arXiv:2010.16061*.
8. Szeliski, R. (2022). *Computer vision: algorithms and applications.* Springer Nature.
9. Sleator, D. D., & Tarjan, R. E. (1985). Amortized efficiency of list update and paging rules. *Communications of the ACM, 28*(2), 202-208.
10. Galić, R., Čajić, E., Stojanovic, Z., & Galić, D. (2023). Stochastic Methods in Artificial Intelligence.
11. Galić, R., Čajić, E., Stojanovic, Z., & Galić, D. (2023). Stochastic Methods in Artificial Intelligence.
12. Galić, R., Čajić, E., Stojanovic, Z., & Galić, D. (2023). Stochastic Methods in Artificial Intelligence.
13. Ćajić, E., Ibrišimović, I., Šehanović, A., Bajrić, D., & Šćekić, J. (2023, December). Fuzzy Logic and Neural Networks for Disease Detection and Simulation in Matlab. In CS & IT Conference Proceedings (Vol. 13, No. 23). CS & IT Conference Proceedings.
14. Čajić, E., Stojanović, Z., & Galić, D. (2023, November). Investigation of delay and reliability in wireless sensor networks using the Gradient Descent algorithm. In 2023 31st Telecommunications Forum (TELFOR) (pp. 1-4). IEEE.
15. Čajić, E., Stojanović, Z., & Galić, D. (2023, November). Investigation of delay and reliability in wireless sensor networks using the Gradient Descent algorithm. In 2023 31st Telecommunications Forum (TELFOR) (pp. 1-4). IEEE.