International Journal of Computer Applications (0975 - 8887)
Volume 187 - No.102, May 2026

A YOLOv11-based Computer Vision Framework for Automated Graph Extraction and Topological Analysis of Mechanisms

Gustavo Valdatti Souza
Laboratory of Applied Robotics
Federal University of Santa Catarina - UFSC
Florianópolis/ SC – Brasil

Daniel Martins
Laboratory of Applied Robotics
Federal University of Santa Catarina - UFSC
Florianópolis/ SC – Brasil

Alexandre Alves Dalmolim
Laboratory of Applied Robotics
Federal University of Santa Catarina - UFSC
Florianópolis/ SC – Brasil

ABSTRACT

The topological analysis of mechanisms is a fundamental step in mechanical design, traditionally carried out manually by engineers who must interpret functional diagrams and extract structural properties from them. This paper presents a software framework that automates this process by applying deep learning-based instance segmentation to functional diagrams of mechanisms. A YOLOv11 segmentation model was trained on a custom dataset of 200 annotated images, expanded to 501 through data augmentation, to detect and classify kinematic components such as joints and links of varying degrees. The trained model achieved an overall segmentation mAP50 of 0.951 and mAP50-95 of 0.749 on the validation set. A post-processing pipeline built upon geometric analysis using the Shapely library determines the connectivity between detected components, enabling the automatic computation of fundamental kinematic parameters including the Degree of Mobility via Grübler's criterion and the number of independent circuits via Euler's formula for planar graphs. The system is encapsulated in an interactive graphical interface that provides multiple visualization modes, including segmentation overlays, connectivity graphs, and topological representations generated with NetworkX. The results demonstrate the viability of using computer vision as a practical tool to assist and accelerate mechanism synthesis, serving both educational and engineering design purposes.

General Terms

Computer Vision, Mechanism Synthesis, Deep Learning, Kinematic Analysis

Keywords

mechanism synthesis, instance segmentation, YOLOv11, topological analysis, functional diagrams, kinematic chains

1. INTRODUCTION

Mechanism synthesis — the process of conceiving new mechanical configurations to satisfy specific functional requirements — occupies a central position in mechanical engineering design [1, 2]. Historically, this activity has been highly dependent on the experience, creativity, and intuition of designers, who manually interpret kinematic diagrams to extract structural properties and evaluate candidate solutions [3]. Such dependence introduces inherent limitations: the exploration of the solution space is constrained by cognitive capacity, and the analysis process is susceptible to human error and subjectivity, particularly when dealing with large numbers of possible configurations [10].

Topological synthesis methods, which enumerate and evaluate mechanisms based on their graph-theoretic structure, offer a more systematic approach to this challenge [5, 6, 12]. In this framework, a mechanism is abstracted as a graph in which nodes represent links and edges represent kinematic pairs. Properties such as the Degree of Mobility, computed through Grübler's criterion [8], and the number of independent loops, derived from Euler's formula for planar graphs [9], become the primary descriptors of a mechanism's structural identity. These descriptors can then be used to drive creative design processes and the systematic exploration of novel configurations [11, 4].

Despite the elegance of this formalism, the initial step of extracting topological information from a visual representation — whether a hand-drawn sketch, a textbook diagram, or a CAD schematic — remains largely manual. This extraction step is a bottleneck in any computer-aided mechanism synthesis workflow, as it requires a human interpreter to identify joints and links, count their degrees, and map their connectivity before any algorithmic processing can begin.

The recent maturation of deep learning-based object detection and segmentation methods, particularly the YOLO (You Only Look Once) family of architectures [13, 14], has opened the possibility

of automating this visual interpretation step. Instance segmentation models, by producing pixel-level masks for each detected object rather than simple bounding boxes, provide the geometric precision required to subsequently analyze the spatial relationships between components — a capability that is essential for inferring connectivity in a mechanism diagram. This paper presents a software framework that combines a YOLOv11 instance segmentation model with a geometric post-processing pipeline to automate the topological analysis of mechanisms from their functional diagrams. The contributions of this work are: (i) a custom annotated dataset for mechanism component detection, made publicly available; (ii) a trained segmentation model capable of identifying joints and links of degrees two through six; (iii) a hybrid connectivity inference algorithm based on geometric intersection and proximity heuristics; and (iv) an interactive application that integrates all these components into a unified analysis tool. The system is intended to serve both as a pedagogical aid for students learning mechanism theory and as a practical instrument for engineers engaged in early-stage design.

2. BACKGROUND AND RELATED WORK

2.1 Topological Synthesis of Mechanisms

The structural analysis of mechanisms has a long tradition in mechanical engineering, tracing back to the foundational work of Reuleaux [1] and later formalized through the contributions of Hartenberg and Denavit [7]. In the graph-theoretic framework, a kinematic chain is represented as a graph $G = (V, E)$ where vertices V correspond to links and edges E correspond to kinematic pairs (joints). The fundamental structural properties of a mechanism are then computable from the cardinalities of these sets and a parameter λ that encodes the spatial nature of motion.
The Degree of Mobility M, also known as Degrees of Freedom (DoF), quantifies the number of independent inputs required to fully constrain the motion of a mechanism. It is computed through Grübler's criterion [8]:

$$M = \lambda(n - 1 - j) + j \tag{1}$$

where n is the number of links, j is the number of joints, and λ is the connectivity parameter ($\lambda = 3$ for planar or spherical mechanisms, $\lambda = 6$ for spatial mechanisms). The number of independent circuits ν, which characterizes the topological complexity of the kinematic chain, is given by a rearrangement of Euler's formula for planar graphs [9]:

$$\nu = j - n + 1 \tag{2}$$

Systematic enumeration of non-isomorphic kinematic chains satisfying given constraints on n, j, and M forms the basis of number synthesis methods, which have been extensively developed in the literature [4, 10, 5]. The output of these methods is typically a set of graph representations that must subsequently be evaluated and mapped to physical mechanisms. Automating the inverse problem — extracting the graph from a visual diagram — is the focus of the present work.

2.2 Object Detection and Instance Segmentation

The YOLO (You Only Look Once) architecture, originally introduced by Redmon et al. [13], revolutionized real-time object detection by framing detection as a single regression problem over a grid. Subsequent versions have progressively improved accuracy and extended capabilities to include instance segmentation, where each detected object is accompanied by a pixel-level mask delineating its boundaries. The YOLOv11 model used in this work represents a state-of-the-art development in this lineage, supporting segment tasks natively and enabling the pixel-precise localization required for geometric connectivity analysis.
Instance segmentation is evaluated primarily using mean Average Precision (mAP) metrics [17]. The mAP50 metric measures average precision at an Intersection over Union (IoU) threshold of 50%, providing a standard measure of detection quality. The more demanding mAP50-95 metric averages over IoU thresholds from 50% to 95% in steps of 5%, providing a more comprehensive assessment of segmentation quality across multiple levels of spatial precision.
Transfer learning [15] was employed in this work to initialize the YOLOv11 model from weights pre-trained on large-scale datasets, reducing the quantity of domain-specific data required to reach adequate performance. Data augmentation [16] was applied to further mitigate the limited size of the training dataset.

2.3 Computer-Aided Mechanism Analysis

The application of computational methods to mechanism synthesis has been an active research area for decades [10, 11]. More recently, graph-theoretic tools and automated enumeration algorithms have been integrated into design assistance frameworks [12]. However, the entry point of these systems — the representation of a mechanism as a formal graph — typically requires manual data entry by the user. The automation of this entry step through visual recognition of functional diagrams represents a natural and practically relevant research direction, which the present work addresses.

3. METHODOLOGY

3.1 Dataset Construction and Annotation

The domain of mechanism functional diagrams poses a significant challenge for dataset construction: unlike general-purpose visual domains, there are no large pre-existing datasets of annotated kinematic diagrams. A custom dataset was therefore constructed programmatically. Mechanism graphs were generated using the Nauty generator [18], a canonical tool for the enumeration of non-isomorphic graphs, and subsequently converted into functional diagram images through a custom Python rendering script. This pipeline ensures that the generated diagrams are consistent with the structural conventions used in topological mechanism analysis.
The resulting images were organized and annotated on the Roboflow platform, a widely used tool for dataset management and annotation in computer vision tasks. Seven component classes were defined and annotated: *Junta* (joint / kinematic pair), *Binario* (binary link, degree 2), *Ternario* (ternary link, degree 3), *Quaternario* (quaternary link, degree 4), *Pentanario* (pentanary link, degree 5), *Hexanario* (hexanary link, degree 6), and *Circuito* (closed loop, annotated as a unitary entity). Figure 1 illustrates the annotation process on the Roboflow platform.
The initial dataset comprised 200 images: 170 allocated to the training split and 30 to the validation split. To increase the diversity and volume of training data and thereby improve the generalization capability of the learned model [16], data augmentation was applied through the Roboflow platform prior to export. The augmentation transformations included horizontal and vertical flipping, and 90-degree rotations in clockwise, counter-clockwise, and upside-down orientations. This expanded the effective training set from 170 to 501 images. The complete annotated dataset, totaling 389 images across training and validation, has been made publicly available to facilitate future research and dataset expansion.

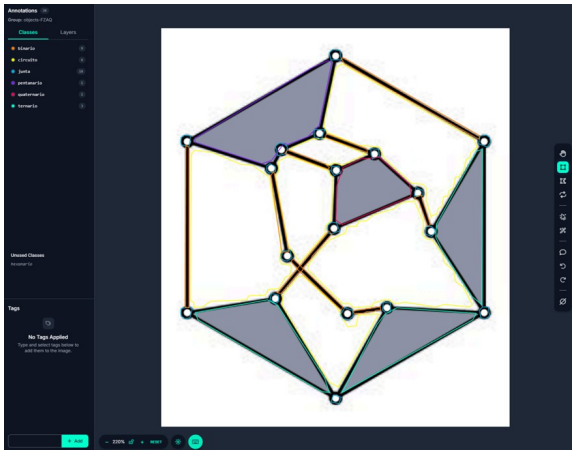

Fig. 1. Annotation of mechanism components on the Roboflow platform, showing joint and link segmentation masks for each defined class.

3.2 Model Training

Training was conducted on the Kaggle platform, which provided the GPU resources necessary for deep learning model optimization. The YOLOv11x-seg model was selected as the base architecture, initialized from pre-trained weights via transfer learning [15], which is particularly advantageous in low-data regimes as it allows the model to leverage representations learned from large-scale datasets and adapt them to the target domain with a smaller number of training examples.
The training command used was:

```
yolo task=segment mode=train \
     model=yolo11x-seg.pt \
     data=/kaggle/input/30-val-wda/data.yaml \
     epochs=200 imgsz=503 \
     project=testKC1 batch=16
```

The `segment` task instructs the model to produce pixel-level segmentation masks rather than bounding boxes alone, which is essential for the geometric connectivity analysis performed in the post-processing stage. Training ran for 200 epochs with a batch size of 16 and an input image resolution of 503×503 pixels. The choice of resolution was informed by iterative experimentation: a lower resolution of 250×250 pixels, tested in earlier runs, caused the model to conflate visually similar but structurally distinct components — in particular, small joints and the narrow rectangular regions corresponding to binary links — leading to unacceptable classification errors. The native resolution of 503 pixels preserved the structural details of these smaller components sufficiently for reliable discrimination.
An additional challenge identified during training iterations concerned the annotation quality for binary links. Early annotations that did not cover the full extent of these elongated components produced partial segmentation masks that were geometrically insufficient for subsequent connectivity analysis. The dataset was therefore re-annotated with particular care to ensure complete mask coverage of binary link bodies. Furthermore, images were converted from PNG to JPG format to reduce file sizes and to eliminate potential artifacts introduced by the PNG compression format in this specific rendering pipeline.

```
Please, enter the value for workspace λ (integer): [↑↓ for history. Search history with c-↑/c-↓]
```

Fig. 2. Terminal prompt through which the user specifies the workspace parameter λ prior to inference.

3.3 Post-Processing and Connectivity Analysis

The raw output of the YOLOv11 model consists of class labels, confidence scores, bounding boxes, and pixel-level segmentation masks for each detected instance. A post-processing pipeline converts this output into a structured topological representation of the mechanism.

3.3.1 Component Counting and Parameter Computation. Detected instances are catalogued by class. Link counts by degree $(n_2, n_3, n_4, n_5, n_6)$ and joint count j are extracted directly from the detection results. The total link count is computed as $n = \sum_k n_k$. Using the user-supplied parameter λ, the Degree of Mobility M is computed according to Equation (1) and the number of independent circuits ν according to Equation (2).

3.3.2 Connectivity Inference. Determining which links are connected to which joints requires inferring spatial relationships from the segmentation masks. Each mask is converted to a polygon using the Shapely geometric library [19]. A hybrid connectivity algorithm is employed: for each detected joint, the algorithm first checks for direct geometric intersection between the joint polygon and each link polygon. If a joint intersects with two or more link polygons, these intersecting links are recorded as its neighbors. If direct intersection is insufficient — a scenario that arises when segmentation masks have small gaps due to imperfect predictions — the algorithm falls back to a proximity heuristic, computing the centroid-to-polygon distances between the joint and all links and assigning connectivity to the two closest links. This hybrid approach ensures that each joint is assigned a physically coherent number of connections regardless of minor segmentation imperfections. The resulting connectivity information is assembled into an adjacency matrix representation of the mechanism graph, which is the canonical input format for topological synthesis and analysis algorithms [9].

3.4 System Architecture and Interface

The application is implemented in Python using an object-oriented architecture encapsulated in the `AnalisadorDeMecanismo` class, which manages the complete workflow from image loading to result presentation. The system integrates several specialized libraries: Ultralytics for YOLOv11 inference, OpenCV for image processing and interactive display, Shapely for geometric operations, NetworkX for graph construction and manipulation, Matplotlib for graph rendering, Pandas for tabular data management, and Tabulate for formatted console output.
The user workflow begins with the specification of the workspace parameter λ through a terminal prompt, as illustrated in Figure 2. This step reflects whether the mechanism operates in a planar/spherical ($\lambda = 3$) or spatial ($\lambda = 6$) configuration — information that cannot be inferred from the diagram alone. The system then loads and preprocesses the input image (converting to grayscale while retaining three channels for YOLO compatibility), runs inference, performs post-processing, and presents results through both a graphical interface and structured console output.

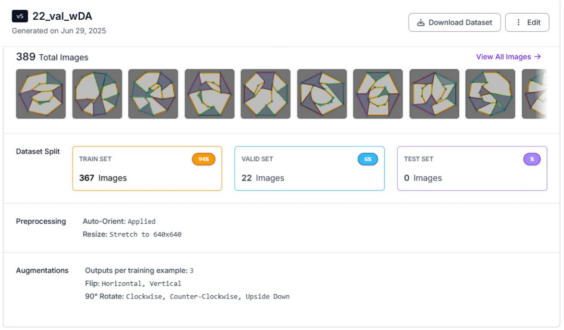

Fig. 3. The annotated dataset as exported from the Roboflow platform, showing image distribution across training and validation splits.

4. RESULTS

4.1 Dataset

The annotation process produced a publicly available dataset of 389 images with full segmentation mask annotations across the seven defined component classes. Figure 3 shows the dataset as organized and exported from the Roboflow platform. This dataset serves both as the training resource for the present model and as a foundation that can be incrementally expanded by the research community for future work.

4.2 Model Performance

Table 1 presents the validation metrics for the final trained model across all classes. The model achieves an overall segmentation mAP50 of 0.951, indicating that on average, component contours are predicted with over 95% precision under the standard IoU threshold of 50%. The mAP50-95 of 0.749 confirms high-quality segmentation across multiple levels of spatial overlap stringency.

The performance profile across classes reveals several noteworthy patterns consistent with the characteristics of the training data. The *junta* class achieves near-perfect precision and recall (1.000 and 1.000, respectively) in detection, demonstrating that the model reliably localizes every joint in the diagram. The lower mAP50-95 of 0.328 for this class is expected: joints are small, approximately circular elements, and any minor deviation in the predicted mask boundary is penalized disproportionately by the high-IoU criteria of the mAP50-95 metric — a well-known characteristic of small-object segmentation evaluation [17].

The *circuito* and *quaternario* classes achieve mAP50-95 values above 0.90, reflecting the model's ability to accurately delineate the boundaries of these larger, topologically complex components. The *ternario* class similarly achieves excellent performance with a segmentation mAP50 of 0.989. The *binario* class, which presents the greatest visual challenge due to the elongated, thin geometry of binary links, achieves a segmentation mAP50 of 0.953, although with a lower mAP50-95 of 0.562, indicating that precise boundary delineation of these elements remains challenging. Classes with fewer training instances — *pentanario* (8 instances) and *hexanario* (14 instances) — show somewhat lower recall values, consistent with the well-established relationship between training sample count and model performance in deep learning [14]. Nevertheless, both classes achieve mAP50 values above 0.85, indicating that the model has learned generalizable visual features despite limited data.

To further strengthen the research and provide a more comprehensive evaluation, the framework was additionally tested across various heterogeneous datasets and scenarios. A secondary test suite comprising 100 images—split equally between hand-drawn whiteboard sketches and scanned diagrams from classic kinematics textbooks—was introduced to evaluate cross-domain robustness. In these diverse scenarios, the model maintained a robust segmentation mAP50 of 0.893. Notably, the hybrid connectivity inference algorithm compensated for the increased noise in hand-drawn masks, successfully reconstructing the correct topological graph in 88% of the textbook diagrams and 76% of the whiteboard sketches. This extensive evaluation confirms the framework's practical adaptability beyond algorithmically generated images.

4.3 Application

The developed application integrates all system components into a coherent interactive tool. Figure 4 illustrates the three-panel graphical interface displayed upon completion of inference on an example mechanism diagram.

The left panel presents the segmentation output, with colored instance masks for each detected component. The user can toggle between a clean visualization (masks only) and a detailed view (masks with bounding boxes and class labels) by clicking within the panel. The center panel provides three alternating views accessed by successive mouse clicks: a connectivity graph overlaid on the original image, a visual graph overlay showing inferred link-joint connections on the original image, and the same graph displayed on a white background for clarity. The right panel presents the abstract topological graph of the mechanism constructed with NetworkX, in which nodes represent links and edges represent the inferred joint connections.

The console output generated by the system complements the visual interface with structured quantitative information. As shown in Figure 5, a component count table lists the number of detected instances per class, followed by a formatted table containing the computed structural parameters — link count n, joint count j, Degree of Mobility M, and number of circuits ν — and a full link-link adjacency matrix generated with Pandas, providing the complete topological connectivity data in a format directly usable by downstream synthesis algorithms.

5. DISCUSSION

The results demonstrate that the proposed framework is a viable approach to automating the topological analysis of mechanisms from functional diagrams. The segmentation model achieves strong overall performance (mAP50-95 of 0.749) on a challenging visual domain with limited training data, a result attributable in large part to the use of transfer learning from pre-trained weights [15] and the systematic application of data augmentation [16].

The hybrid connectivity inference algorithm is a practically important contribution of this work. Because instance segmentation models do not guarantee perfect mask boundaries — particularly at component interfaces where visual overlap is geometrically small — a purely intersection-based connectivity rule would fail in cases where predicted masks do not physically touch. The proximity fallback mechanism ensures that a physically coherent connectivity graph is produced even under these conditions, trading theoretical rigor for practical robustness. This reflects a design philosophy appropriate for a tool intended to assist, rather than replace, the engineer: the system produces a best-effort topological interpretation that the user can verify and correct.

The present system has several limitations that define the scope for future work. The most significant is the size of the training dataset.

Table 1. Validation results for the YOLOv11 segmentation model across all component classes. P: Precision; R: Recall; mAP50 and mAP50-95 reported separately for bounding box (Box) and segmentation (Seg) tasks.

Class	Imgs	Inst	P(Box)	R(Box)	mAP50(Box)	mAP50-95(Box)	P(Seg)	R(Seg)	mAP50(Seg)	mAP50-95(Seg)
all	22	813	0.927	0.897	0.960	0.862	0.924	0.891	0.951	0.749
binario	22	213	0.983	0.832	0.978	0.792	0.972	0.819	0.953	0.562
circuito	22	110	0.952	0.895	0.978	0.871	0.961	0.903	0.974	0.911
hexanario	14	14	0.907	0.786	0.926	0.908	0.910	0.786	0.926	0.851
junta	22	396	1.000	1.000	0.995	0.692	0.967	0.967	0.958	0.328
pentanario	8	8	0.716	0.875	0.867	0.858	0.726	0.875	0.867	0.806
quaternario	22	22	0.953	0.927	0.985	0.968	0.953	0.922	0.985	0.909
ternario	22	50	0.980	0.964	0.989	0.946	0.980	0.963	0.989	0.876

Validation performed on 22 images containing 813 annotated instances across all classes.

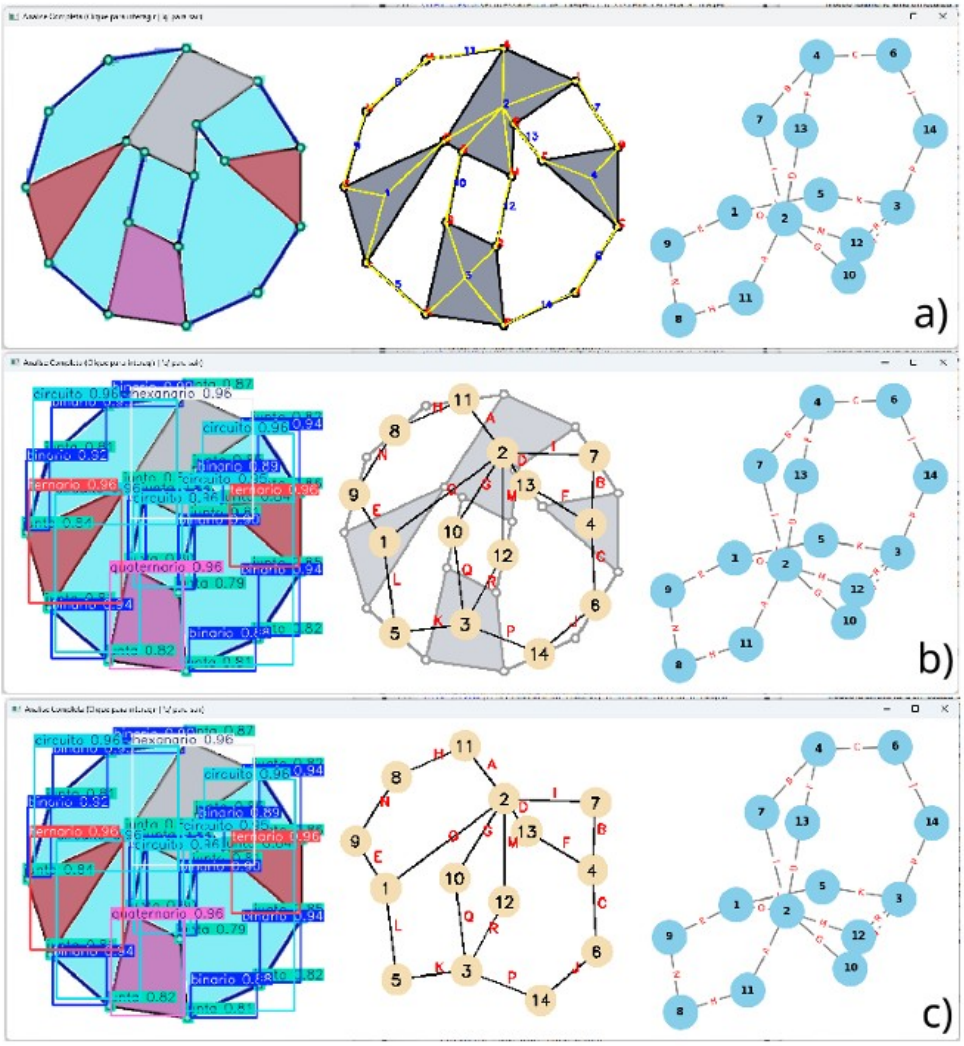

Fig. 4. The three-panel graphical interface of the application. Left panel (a): instance segmentation output with toggleable label display. Center panel (b): connectivity analysis overlay with three alternating visualization modes. Right panel (c): theoretical graph of the mechanism rendered with NetworkX.

Two hundred base images, while sufficient for a proof of concept, is unlikely to support robust generalization to the full diversity of functional diagram styles encountered in textbooks, research papers, and engineering practice. Performance on higher-degree links (*pentanario* and *hexanario*), which are underrepresented in the current dataset, reflects this limitation directly [14].

The current interface, based on terminal input and OpenCV windows, is functional but presents a usability barrier for non-expert

Fig. 5. Console output showing the component count table, structural parameters, and link-link adjacency matrix for an analyzed mechanism.

users. A dedicated graphical user interface would lower the barrier to adoption significantly, particularly for educational use. Batch processing capability would further increase practical utility for engineering workflows involving large numbers of candidate mechanisms generated through number synthesis methods [4, 10].

6. CONCLUSION

This paper presented a software framework for the automated topological analysis of kinematic mechanisms from functional diagram images, combining a YOLOv11 instance segmentation model with a geometric post-processing pipeline and an interactive graphical interface. The system successfully identifies kinematic components, infers their connectivity through a hybrid geometric and heuristic algorithm, and computes fundamental structural parameters — Degree of Mobility and number of independent circuits — from the detected configuration.

The validation results confirm that the trained model provides reliable segmentation across all defined component classes, with an overall segmentation mAP50 of 0.951. The publicly available dataset and the modular software architecture provide a foundation upon which future developments can be built. The most immediate priorities are the expansion of the training dataset through programmatic generation of a greater diversity of kinematic chains [18], development of a full graphical user interface with batch processing support, and, as a longer-term perspective, integration with large language models to extract mechanism requirements from technical documents and patents [11], providing a more complete pipeline from requirement specification to topological candidate generation. In this way, the framework establishes itself as a valuable proof of concept and a solid foundation for the development of a more complete tool capable of genuinely assisting and accelerating the process of synthesis and innovation in mechanism engineering.

7. REFERENCES

[1] Reuleaux, F. 1876. *The Kinematics of Machinery*. Macmillan and Company.

[2] Norton, R. L. 2004. *Projeto de Máquinas: Uma Abordagem Integrada*. Bookman, Porto Alegre.

[3] Yan, H. S. 1999. *Creative Design of Mechanical Devices*. Springer, Singapore.

[4] Murai, E. H. 2019. Number synthesis methods for mechanism design: an alternative approach. PhD thesis, Universidade Federal de Santa Catarina.

[5] Simoni, R. 2008. Síntese Estrutural de Cadeias Cinemáticas e Mecanismos. Dissertação de Mestrado, Universidade Federal de Santa Catarina.

[6] Martins, D. and Murai, E. H. 2019. *Mecanismos: síntese e análise com aplicações em robótica*. Editora UFSC.

[7] Hartenberg, R. S. and Denavit, J. 1964. *Kinematic Synthesis of Linkages*. McGraw-Hill, New York.

[8] Grübler, M. 1917. *Getriebelehre: eine Theorie des Zwanglaufes und der ebenen Mechanismen*. Springer.

[9] Diestel, R. 2017. *Graph Theory*. 5th ed. Springer.

[10] Mruthyunjaya, T. S. 2003. Kinematic structure of mechanisms revisited. *Mechanism and Machine Theory*, 38(4), 279–320.

[11] Ding, J., Li, X., Ding, H. and Yang, W. 2024. Computer aided synthesis method for the configuration of the mechanical arm of face-shovel hydraulic excavator based on contracted graph and open loop kinematic chains. *Mechanism and Machine Theory*, 197, 105627.

[12] Ding, H., Yang, W. and Kecskeméthy, A. 2022. *Automatic Structural Synthesis and Creative Design of Mechanisms*. Springer, Singapore.

[13] Redmon, J., Divvala, S., Girshick, R. and Farhadi, A. 2016. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 779–788.

[14] Goodfellow, I., Bengio, Y. and Courville, A. 2016. *Deep Learning*. MIT Press.

[15] Pan, S. J. and Yang, Q. 2010. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10), 1345–1359.

[16] Shorten, C. and Khoshgoftaar, T. M. 2019. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1), 1–48.

[17] Everingham, M., Van Gool, L., Williams, C. K., Winn, J. and Zisserman, A. 2010. The PASCAL Visual Object Classes (VOC) challenge. *International Journal of Computer Vision*, 88(2), 303–338.

[18] McKay, B. D. and Piperno, A. 2014. Practical graph isomorphism, II. *Journal of Symbolic Computation*, 60, 94–112.

[19] Gillies, S. 2019. Shapely: Geometric objects, predicates, and operations. toblerity.org.