FROM SKETCHES TO GRAPHS: A DEEP LEARNING BASED METHOD FOR DETECTION AND CONTEXTUALISATION OF PRINCIPLE SKETCHES IN THE EARLY PHASE OF PRODUCT DEVELOPMENT

Bickel, Sebastian;
Goetz, Stefan;
Wartzack, Sandro

Friedrich-Alexander-Universität Erlangen-Nürnberg

ABSTRACT

The digitalization trend is finding its way more and more into product development, resulting in new frameworks to enhance product engineering. An integral element is the application of new techniques to existing data, which offers an enormous potential for time and cost savings, because duplicate work in product design and subsequent steps is avoided. The reduction of costs can be further increased through the application as early as possible in the product development process. One solution is outlined in this publication, where the source of available data is principle sketches from engineering design. These represent the basic solution for technical products in a simplified way and are often deployed in the early stages of the development process. This representation enables not only a search of similar sketches but also other fields of interest such as product optimization or the search of CAD-geometries. To utilize this data in a practical way, a procedure is presented which recognizes the symbols of the sketches and subsequently converts them into graphs. An exemplary dataset from different gearbox layouts is used to present the application opportunities by performing similarity searches with multiple input formats.

Keywords: Early design phases, Machine learning, Artificial intelligence, object detection, deep learning

Contact:
Bickel, Sebastian
Friedrich-Alexander-Universität Erlangen-Nürnberg
Germany
bickel@mfk.fau.de

Cite this article: Bickel, S., Goetz, S., Wartzack, S. (2023) 'From Sketches to Graphs: A Deep Learning Based Method for Detection and Contextualisation of Principle Sketches in the Early Phase of Product Development', in *Proceedings of the International Conference on Engineering Design (ICED23),* Bordeaux, France, 24-28 July 2023. DOI:10.1017/pds.2023.198

1 INTRODUCTION

The digitalization influences many people in their daily lives, for example through the invention of smartphones, a new media landscape or working environment was established. This trend is also reflected in product design, as new tools and methods are constantly introduced to further improve product engineering and increase its efficiency. In addition to the development and conception of new methods, the use of existing data is an essential step for the implementation of digitalization. Especially in product development, the amount of virtual and physical data is enormous, but its automated utilization is often difficult.

Since drawings and sketches are common in the engineering of new components and usually represent the entire product in a condensed way, they are very well suited for automatic evaluation. To ensure that the existing data is effectively utilized, it must be recognized and automatically processed to assist the design task, for example through a search for similar sketches and the retrieval of linked data. The approach presented in this paper aims to solve this issue for principle sketches. These are applied very early in the product development process which leads to a high cost-saving potential compared to the subsequent phases.

An expansion of the symbol detection in technical sketches is presented in the following. Additionally, this paper introduces the transformation of the recognized symbols into graphs, which in turn can be used for text or image-based search. The whole procedure is demonstrated on a dataset of different gearbox layouts.

2 STATE OF THE ART

Symbols are a commonly employed way to quickly communicate specific domain expertise. In the field of engineering, each subdivision has developed its own symbols, which are understood by educated personnel, with examples of simple drawings with specific symbols given in Figure 1. The examples vary from illustrations of products from the field of mechanical engineering to flowcharts in computer science or architectural plans like the illustrated bedroom. The individual drawings are generated on a PC or by hand and therefore vary in size, format and complexity.

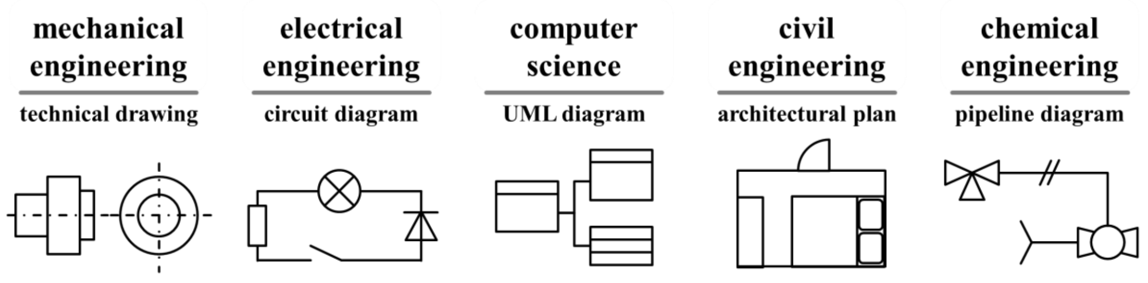

Figure 1: Overview of different types of engineering drawings

2.1 Symbols in engineering

In product development, symbols are often used in the early stages when the geometry is not yet finalized. In the development process, principle sketches are derived from the principle solution, which describes a rough and basic solution for a design task. They can offer solutions for the entire system or parts of it and serve as the basis for the embodiment phase and thus the creation of the product. Fundamental innovations or new designs are often developed with the help of principle sketches. (Roth, 2001)

The VDI 2222 (1997) lists various representation ways for principle solutions, e.g. standardized symbolic sketches, free line sketches, 3D freehand sketches or unscaled rough drafts. The difference between a principle sketch and a technical drawing depicts Figure 2. The sketch on the left shows a solution for the task of converting a translational motion into a rotational one. The technical drawing on the right illustrates an engineered connecting rod, which is marked in the sketch as component (a). In comparison, the principle sketch represents the solution of the problem more simplified, but offers the advantage of rapid creation of variants and different solutions.

https://doi.org/10.1017/pds.2023.198 Published online by Cambridge University Press

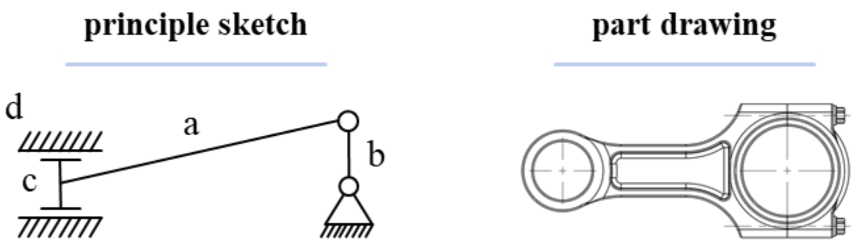

Figure 2: Example of a principle sketch and the resulting technical drawing of a designed part

Therefore, the goal is to recognize and transform principle sketches in an automated way, to use the available information. This conversion offers the possibility to quickly and automatically access existing amount of data and to sustainably improve the product development process.

2.2 Engineering drawings digitalization and contextualisation

Due to the complexity and variety of engineering drawings, the transformation into a digital environment is a difficult task but could provide a rich source of data and information. A framework for this problem statement was developed by Moreno-Garcia et al. (2017), which differs between the digitalization and contextualisation of engineering drawings. The first element of the framework focuses on the preparation of drawings, the detection of symbols and the assignment of the recognized symbols to class labels. In the second step, the contextualisation, the prepared drawings are further processed to extract information from them. The following state of the art section is organized according to this structure: first, the symbol detection is discussed in more detail, followed by the contextualisation.

2.2.1 Symbol detection

The idea of using computers to detect symbols in drawings reaches back to the 1980s when Okazaki et al. (1988) stated a new method to detect symbols in circuit diagrams. These older approaches often depend on heuristics for symbol recognition, such as Kasturi et al (1990). As a result, these methods are not flexible and difficult to adapt to new symbols or tasks.
In recent years, domain-specific solutions have been increasingly presented, which frequently rely on Deep Learning (DL) or Machine Learning (ML) for symbol recognition to address this issue. In the field of piping and instrument diagrams (P&ID), Elyan et al. (2018) have applied Convolutional Neural Networks (CNN) and other Machine Learning methods for detecting different symbols. In the same environment, Moon et al. (2021) presented a method that identifies different lines and arrows in P&ID drawings. The recognition of the respective line types is done by RetinaNet. Another approach focusing on valve and instrument symbol detection, was presented by Yun et al. (2020). This method utilizes a Regional Convolutional Neural Network (R-CNN) for symbol identification and deep adaptive clustering for the classification of the symbols.
For UML diagrams, which are widely used in the field of computer science, various approaches with Machine or Deep Learning were also developed. Ho-Quang et al. (2014) developed a procedure to evaluate the components of UML diagrams using classical line detection and then classify them by different ML classifiers. Another procedure was proposed by Shcherban et al. (2021), where different CNN architectures were compared using transfer learning to recognize different types of UML diagrams from a dataset. However, all Deep Learning methods have a disadvantage: the data set required for training. The data set is either too small, not publicly accessible or has to be prepared manually. According to Moreno-Garcia et al. (2017), one possibility to solve the problem is data augmentation, but this also requires a dataset with labelled data. A new approach for the mechanical engineering sketches was presented by Bickel et al. (2021), which generates a large synthetic dataset for the training of a principle sketch object detection algorithm. For the identification of tolerance symbols on technical drawings, Deng et al. (2020) presented a CNN approach that extracts the symbol from the drawing through image preprocessing and subsequently applies symbol and text segmentation. However, the pure recognition of the symbols does not necessarily provide much additional benefit, so the next chapter introduces the contextualisation of the detected symbols.

https://doi.org/10.1017/pds.2023.198 Published online by Cambridge University Press

2.2.2 Contextualisation

The procedures for contextualising the recognized symbols are specialized and, compared to symbol detection, fewer frameworks were published. An example from mechanical engineering is the Celesstin system by Vaxiviere and Tombre (1992). This system is able to divide a 2D technical drawing of shafts and bearings into different components by vectorization. These modules are then prepared and transferred to the CATIA CAD system, which generates a 3D geometry of the specific components.
In addition to this very specific solution system, Moreno-Garcia et al. (2017) also describe methods that transform the found symbols into a graph through a netlist. This approach was implemented by Yu et al. (1997) for piping diagrams and by Bailey et al. (1995) for electrical circuit diagrams. Another concept for piping diagrams and netlists was presented by Wen et al. (2016), which uses graphs to match 2D with 3D process plants.
For detecting hand-sketch and computer-generated diagrams, Fu and Kara (2011) created a method, where the user produces a synthetically enlarged training data. The results were transformed into a graph to derive Simulink or SimMechanics models. Another approach that takes advantage of graphs was presented by Mizanur Rahman et al. (2021). This method applies anomaly detection to the generated graphs to improve the detected symbols and to identify and correct possible errors.
The presented state of the art reveals that many frameworks and methods were developed for other domains, but in the early stages of the product development process, there is still no procedure for automatically recognizing principle sketches from product development and transferring them to a graph. Converting the image into a neutral format, such as a graph, facilitates and enables many possibilities that a plain image-based comparison would not allow. During the problem solving process, the specific properties of principle sketches must be taken into account, so they are represented in the resulting graph and are available for further applications like a similarity search or graph-based robust design. In the following, a procedure is presented and explained in detail, to solve the stated problem.

3 APPROACH EXTENSION

The first part of the method is built on previous work from Bickel et al. (2021) and is focused on training the Deep Learning model for symbol recognition. Compared to this previous approach, the synthetic data generation for the symbol detection was significantly improved for better applicability to principle sketches with background lines. The transfer of the sketches into graphs is a new process stage and therefore presented and investigated in this paper. In Figure 3 the general approach is depicted, with the novel steps highlighted in dotted frames.

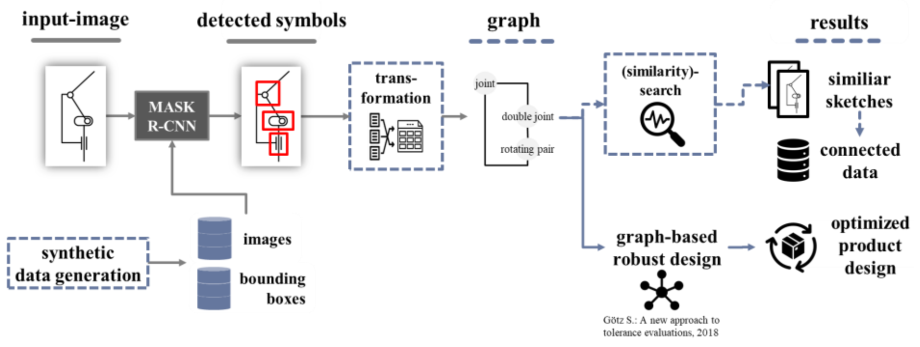

Figure 3: Overview of the general approach, based on Bickel et al. (2021)

Transferring sketches to graphs offers a wide range of applications. The conversion enables a similarity search with different input formats, for example text, images or graphs. All three search inputs can be compared with a database of graphs to find the most similar sketch. The retrieved sketches can be used to troubleshoot a current design task by quickly presenting multiple variants or to access linked data of the principle sketch and thus get CAD files, technical drawings or even the technical requirements. In addition to searching for the entire sketch, it is also possible to perform a query for sub-graphs, for example, if only certain elements in an assembly is relevant for the engineer. For this purpose, sub-graph matching can be utilized to find all relevant overall systems that contain the searched combination of elements.

https://doi.org/10.1017/pds.2023.198 Published online by Cambridge University Press

A necessary requirement for this transformation is a high classification accuracy of the different symbols, which calls for an adaption of the symbol recognition. The improvement of the existing procedure is explained first, followed by the new approach for the graph transformation.

3.1 Symbol detection

The pure recognition of symbols functions well, but a revision of the original procedure for data generation is necessary because it can lead to an incorrect classification of connecting lines. The straight lines of an enclosure for example can easily be interpreted as a symbol from the object detection model.
To fix this problem, a background generator was added to the training data generation. This inserts random lines and objects in any number and thickness as background. Subsequently, the objects including the bounding box information are placed over this background. The additional segment to the approach is displayed in Figure 4.

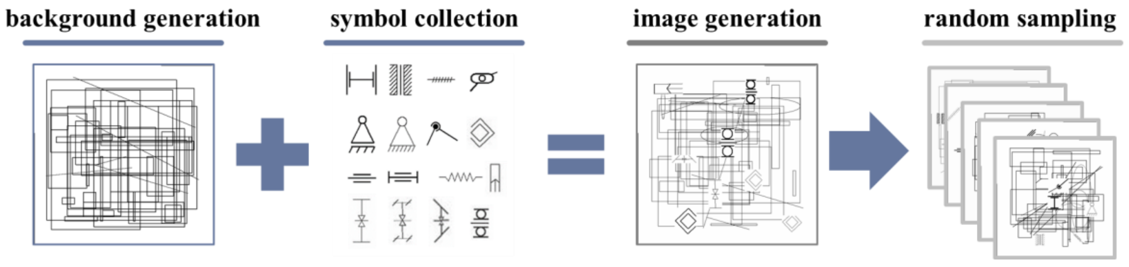

Figure 4: Augmentation of the training data generation method

Furthermore, new symbols are implemented into the data generation to extend the whole approach to a wider range of applications. In total 16 different symbols are generated in each training set. The new principle sketch icons are primarily from the field of gear theory and therefore they include different gears and bearings. The results of the new data generation are shown in Figure 4. After the enhancement of the data generation method, the following chapter explains the conversion of the results into a graph.

3.2 Graph transformation

The idea of the transformation is to capture the content of the principle sketch in the best quality possible. Therefore, the recognized symbols are supposed to be linked to each other, which requires the connecting lines on the drawing. These must be extracted and identified first and can then be combined with the bounding boxes to a resulting graph. The procedure is based on Fu and Kara (2011) and includes the first step of their method. The detected symbols including the associated bounding boxes serve as the starting point for the procedure. The entire process is displayed in Figure 5.

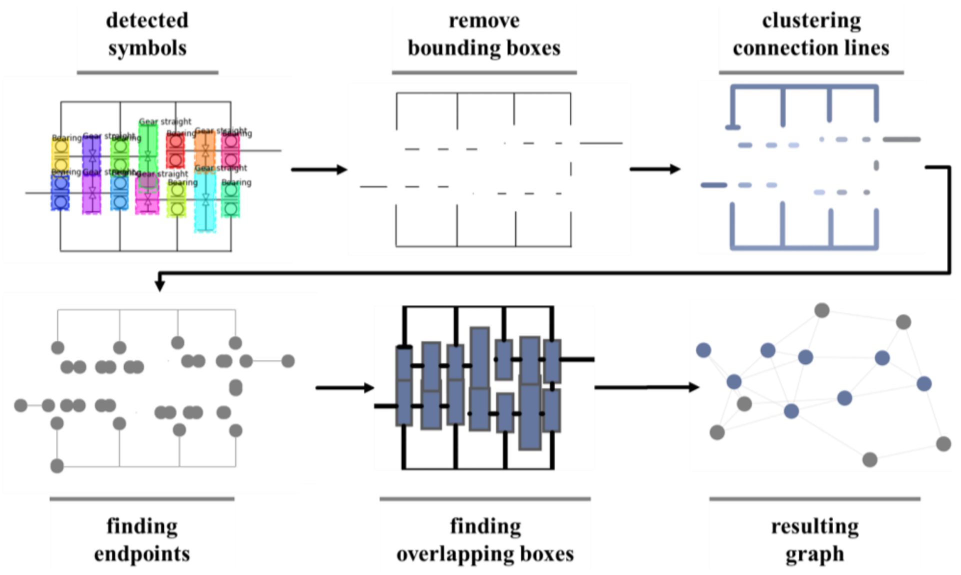

Figure 5: Different steps of the graph transformation procedure

https://doi.org/10.1017/pds.2023.198 Published online by Cambridge University Press

First, the bounding boxes are trimmed out of the sketch so only the connecting lines remain. These lines are then skeletonized to improve the subsequent detection of the strokes. A dbscan cluster algorithm (Ester et al. 1996) offers the possibility to filter the open lines through the number of necessary neighbour points and thus determines the recognized lines in the sketch. This allows to consider very small elements or to ignore them, depending on the use case. Subsequently, the open endpoints are determined for each cluster. This is performed by a hit or miss transformation, which is able to detect even multiple open ends of a line, which is a necessary requirement for principle sketches. Subsequently, the open endpoints are compared with the bounding boxes via collision detection to find connecting lines between the boxes. The size of the endpoints can be chosen as desired. A larger size leads to better collision detection but also to wrong connections. In addition, the overlap of the individual bounding boxes is examined and also used for the graph transformation. Based on these two results, a graph can be generated that represents the linkage of the symbols and the respective symbol type. The nodes represent the symbols including their class and the edges connect them, depending on the relationship in the sketch.

4 DEMONSTRATOR

To demonstrate the presented approach and its application possibilities, a collection of principle sketches of gearboxes is used to show the potential offered by a graph conversion. A total of 29 gearbox variants were developed, which can be roughly sorted according to their stage. Figure 6 shows five examples of different gearbox designs. The principle sketches consist of the symbols: bearing, straight-toothed gear, helical-toothed gear and bevel gear. Furthermore, the housing of the respective gearbox is illustrated in simplified form.

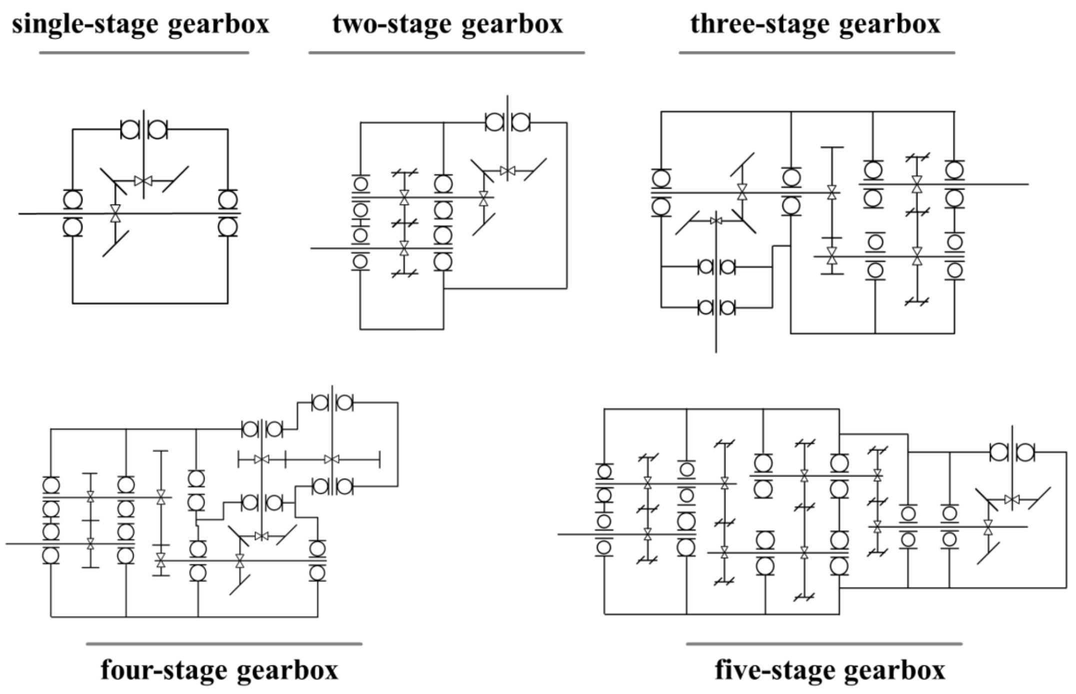

Figure 6: Example of different sketches for gearbox layouts

4.1 Comparison of different training datasets

The 29 drawings have been labelled by hand and then classified with two differently trained MASK R-CNN models, to test the new synthetic data generation. The backbone network and the starting weights are identical for both networks, using the resnet50 and the starting weights of the COCO dataset (Lin et al. 2014). Only the training data is varied, one with and one without the changing background, but both containing the same 16 different symbols, as shown in the symbol collection in Figure 4.

For the training of the detection model a workstation PC, with an Nvidia Titan V GPU, 32 GB RAM and an Intel Xeon W-2125 CPU was used. The training datasets consist of 200,000 images, generated with both approaches where each is split into 180,000 images for training and 20,000 for validation.

To compare the detection capabilities, the metrics of the COCO Dataset are considered, which consist of the intersection over union (IoU), the mean average precision (mAP) and the mean recall (mRc). The IoU value calculates the overlap between the predicted bounding box and the ground truth bounding box. If the IoU is above a certain threshold, which is stated behind the comparison metric (e.g. mAP 75 means a minimum of 75 % IoU) the detection is valid. Otherwise, the classified bounding box is not taken into account. For calculating the average precision (AP) a 101-point interpolated AP definition is used, which is computed at different IoU thresholds. The mean of all AP values is stored in the mAP variable. For the comparison of the different training datasets, the IoU threshold was set to 0.50 % and 0.75 %. The results are displayed in Figure 7.
The comparison of the average precision and the average recall shows very clearly that the new approach offers significant advantages. In all comparative metrics presented, it outperforms the previous one. The poor results of the old approach are not due to bad recognition of symbols in general, but to the confusion of the housing with a symbol. Since there is no background in the data, the trained model always tries to assign a symbol to the casing lines. In contrast, the model trained with the new data identifies the symbols more distinctly which results in a high mAP value. The decreasing precision values for higher IoU values in the new dataset model are partly due to the model itself, as it sometimes has problems with long symbols, e.g. gears, and the distinction between straight and helical gears. Another influence is the manually labelled data, where the bounding boxes are never positioned as accurately as synthetically generated data and therefore the overlap is never perfect.

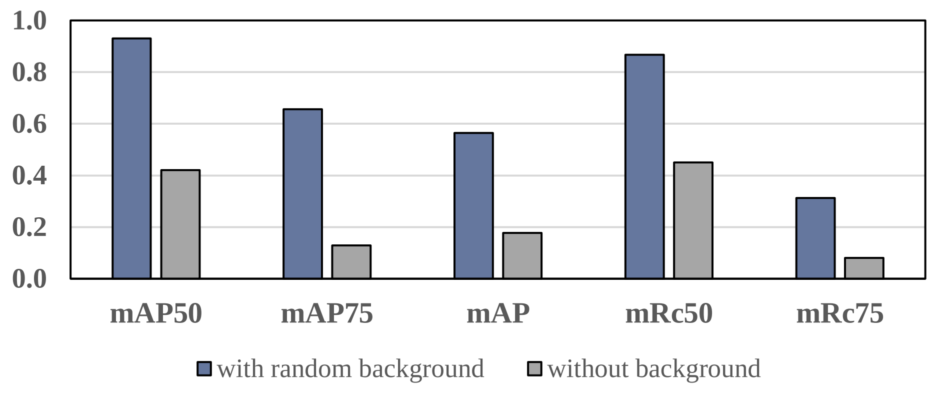

Figure 7: Comparing the results with different training datasets

4.2 Transformation to graphs

After the successful recognition of the symbols, all sketches were converted into graphs. The whole 29 principle sketches of the different gear setups are transformed, following the procedure from section 3.2. For the representation of the graphs, the networkX package in python was chosen. An example transformation is shown in Figure 8, where the different node colours represent the symbol classes. These nodes are connected to each other via edges. The transformed graph clearly displays the respective gear pairs, which are connected via the bearings. The resulting graph database offers many options for evaluation and analysis. In the following, potential implementations for different similarity searches or rule-based checks of the graphs are presented.

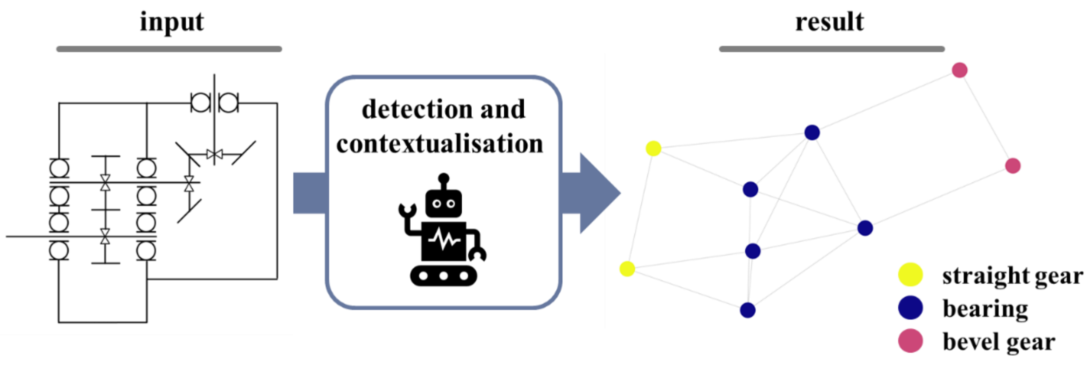

Figure 8: Transformation example of a sketch from the database

https://doi.org/10.1017/pds.2023.198 Published online by Cambridge University Press

4.2.1 Text-based search

The first way to perform a search is based on text input. By simply counting the individual nodes, it is possible to derive a table from the graphs and to search or filter for specific properties. A general categorization can be made, for instance, based on the number of bearings, the gear stages, and the number of specific gear types. In addition, a simple check of the imported data is feasible. In the context of the gearboxes, a comparison of the individual gears in contact would be possible, to determine whether both have the same gear type. If this is not the case, the identified graph is not included in the database or should be checked again manually by an engineer and adjusted if necessary. For the presented dataset of gearbox drawings, an example search query could be: gear stage equals five and the results are sorted by bevel and helical gears. Figure 9 represents the results of this search query in form of a dataframe. The most fitting result is listed as the first entry in the table, followed by the less similar results.

	Number Bearings	Gear Steps	Number Gears total	Number Gear straight	Number Gear helical	Number Gear bevel
27	11	5	10	0	8	2
28	10	5	10	0	8	2
25	11	5	10	4	4	2
23	11	5	10	8	0	2
24	10	5	10	8	0	2

Figure 9: The results for an example text-based search query

4.2.2 Graph-based search

Another way to use the similarity search is with a graph as a query input. This graph can be generated in various ways. The ideal graph itself can be created by setting the nodes and edges according to the current design. Furthermore, a principle sketch can be drawn and then converted as stated in the section 3.2. The presented method offers the possibility to recognize hand sketches as shown by the sketch query in Figure 10. The example image was only converted to a grayscale image using image thresholding (50%) and then passed to the trained model. The accuracy for this type could be improved significantly by taking hand drawn symbols into account in the data generation. Despite the error in the classification of the symbols the example of the similarity search in Figure 10 illustrates that the results retrieved match the query, since all outputs have the same gear stages. The third and fourth results have an equal similarity score, therefore there both ranked third.

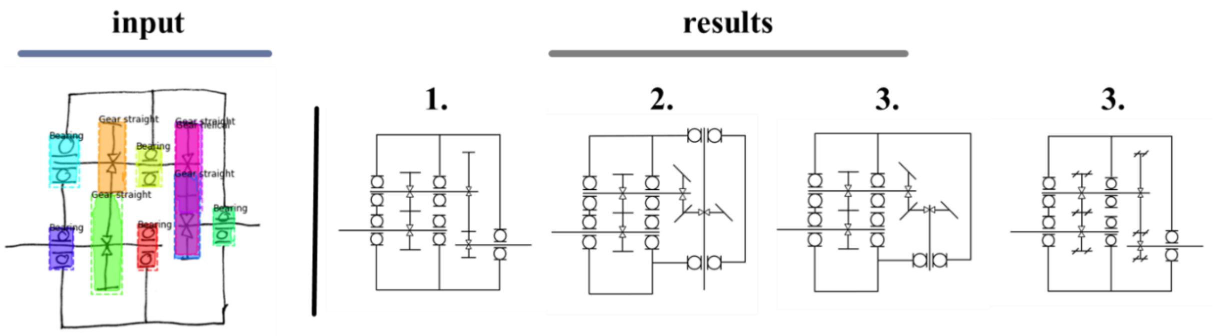

Figure 10: Example of a similarity search with a hand drawn sketch as input query

Independently from the creation of the search graph, many methods were developed for similarity search with a graph as input. The special characteristic of graphs derived from principle sketches are the attributes of the individual nodes. These are the class labels of the bounding boxes and must be taken into account in the search, which makes straightforward topology comparisons unsuitable. For this reason, the Graph Edit Distance (GED) was chosen as a similarity measure for this demonstrator (Abu-Aisheh et al. 2015). This metric can take into account the attributes of the graph and describes the smallest change path for two graphs to be isomorphic. It is based on the Levenshtein distance (Levenshtein, 1966), which describes the number of change operations for words. The costs for the different operations on the graph (node or edge, add or remove) can be freely chosen, but for this example, they have all been weighted equally.

https://doi.org/10.1017/pds.2023.198 Published online by Cambridge University Press

Another advantage of the graph-based search is that sub-graphs can be included in the search. The algorithms for this are often called sub-graph matching and test whether the search graph occurs in one of the database graphs. For the demonstrator, a VF2 algorithm (Cordella et al. 2001 and Cordella et al. 2005) was applied, which again considers the node attributes of the graphs.

4.2.3 Further applications and discussion

The previously mentioned examples mainly include the similarity search combined with the retrieval of additional information. But the converted graph is also suitable for other tasks. For example, it can be automatically extended with knowledge using ontologies. This requires product or domain specific ontologies, which recognize correlations and transfer them into the graph. One example is the system developed by Goetz et al. (2018), which combines robust design and tolerance management. For this system, the presented approach can be leveraged to achieve further automation of the process.

The potential of transferring a sketch to an independent format can be expanded. Another promising field of application is the transfer of other results of the product development process into graphs, which enables a format skipping similarity search and therefore time and cost savings. Especially suitable for this purpose is geometry in the form of assemblies, because they could be transferred into graphs by contact detection (Lupinetti et al. 2016) and component classification (Bickel et al. 2022). These assembly graphs could then be compared to sketch graphs using similarity metrics, which would enable sketch-based searches of geometries.

In addition to the broad application potential, the presented approach still has drawbacks, primarily caused by the high dependence on very good symbol recognition accuracy. During the application on the demonstrator, it was noticed that the detection of identical symbols arranged next to each other, e.g. two bearings, is sometimes difficult. This can lead to unidentified symbols or very large bounding boxes. The error may be corrected either through drawings with more space between the symbols or by increasing the variance of the symbols in the generated data. It also happens more rarely that the location of the symbols is correct, but the class label is incorrect, e.g. that helical and straight gears are mixed up. However, the problem of misclassified symbols can be filtered out easily with specific domain knowledge, as shown in section 4.2.1. Since it is not possible for two gears in mesh to have different tooth shapes, the error is quickly detected and corrected.

5 CONCULSION AND OUTLOOK

In summary, this paper has presented an approach that enables the transformation of principle sketches into graphs. Furthermore, the application potential of these graphs was shown through a demonstrator dataset. The method includes the detection of symbols in principle sketches using a Deep Learning model. The training data generation for the DL model is also part of the procedure since synthetic data is used for the model. Subsequently, the recognized symbols form the basis for a transformation into a graph, which is implemented with the help of computer vision algorithms. Afterwards, a demonstrator database with different gearbox sketches is provided to show the possibilities with the new graphs by performing similarity searches with different input formats.

The next step could be to build another demonstrator to show the versatility of the method, for example, a mechanical gripper or coin press machine. Additionally, the ability to recognize hand sketches could be investigated in more detail, for which studies with several samples would be necessary to test the applicability extensively. Finally, the graphs could be further developed for specific use cases through the implementation of domain-specific knowledge. This would create an expert system that combines the existing knowledge with the detected graph via ontologies.

ACKNOWLEDGMENTS

The authors would like to thank the NVIDIA Corporation and the academic GPU Grant Program for the donation of a Tesla GPU.

REFERENCES

Abu-Aisheh, Z., Raveaux, R., Ramel, J. Y., & Martineau, P. (2015). An exact graph edit distance algorithm for solving pattern recognition problems. In 4th International Conference on Pattern Recognition Applications and Methods 2015.

Bailey D, Norman A, Moretti G, North P (1995). Electronic schematic recognition. Massey University, Wellington, New Zealand

Bickel, S., Schleich, B., & Wartzack, S. (2021). Detection and classification of symbols in principle sketches using deep learning. Proceedings of the Design Society, 1, pp. 1183-1192.

Bickel, S., Schleich, B., & Wartzack, S. (2022). A New Projection Based Method for the Classification of Mechanical Components Using Convolutional Neural Networks. Proceedings of the Design Society, 2, pp. 1501-1510.

Cordella, L. P., Foggia, P., Sansone, C., & Vento, M. (2001). An improved algorithm for matching large graphs. In 3rd IAPR-TC15 workshop on graph-based representations in pattern recognition, pp. 149-159.

Cordella, L. P., Foggia, P., Sansone, C., & Vento, M. (2004). A (sub) graph isomorphism algorithm for matching large graphs. IEEE transactions on pattern analysis and machine intelligence, 26(10), pp. 1367-1372.

Deng, X., Li, T., Xu, Y., Cao, Y., Kong, C., Zhang, E., (2020). The Computer Vision-based Tolerancing Callout Detection Model. Procedia CIRP. 92, pp. 134-139. https://doi.org/10.1016/j.procir.2020.05.189

Elyan, E., Moreno-García, C., Jayne, C., (2018). Symbols Classification in Engineering Drawings. International joint conference on neural networks 2018 (IJCNN), 8-13 July 2018, Rio de Janeiro, Brazil https://dx.doi.org/10.1109/IJCNN.2018.8489087.

Ester, M., Kriegel, H. P., Sander, J., & Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In kdd, vol. 96, No. 34, pp. 226-231.

Fu, L., Kara, L., (2011). From engineering diagrams to engineering models: Visual recognition and applications. Computer-Aided Design. 43. pp. 278-292. https://doi.org/10.1016/j.cad.2010.12.011

Goetz S., Schleich B., Wartzack S. (2018). A new approach to first tolerance evaluations in the conceptual design stage based on tolerance graphs, Procedia CIRP, vol. 75, pp. 167-172 https://doi.org/10.1016/j.procir.2018.04.030.

Ho-Quang, T., Chaudron, M. R., Samúelsson, I., Hjaltason, J., Karasneh, B., & Osman, H. (2014). Automatic classification of UML class diagrams from images. In 2014 21st Asia-Pacific Software Engineering Conference, vol. 1, pp. 399-406. IEEE.

Kasturi R., Bow S. T., El-Masri W., Shah J., Gattiker J. R. and Mokate U. B. (1990). A system for interpretation of line drawings, in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 12, no. 10, pp. 978-992, https://dx.doi.org/10.1109/34.58870.

Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions, and reversals. In Soviet physics doklady, vol. 10, No. 8, pp. 707-710.

Lin T.-Y., Maire M., Belongie S., Hays J., Perona P., Ramanan D., Dollár P., Zitnick C. L. (2014). Microsoft COCO: Common Objects in Context. In: Fleet D., Pajdla T., Schiele B., Tuytelaars T. (eds) Computer Vision – ECCV 2014. vol. 8693. Springer, Cham. https://doi.org/10.1007/978-3-319-10602-1_48

Lupinetti, K., Giannini, F., Monti, M., & Pernot, J. P. (2016). Automatic extraction of assembly component relationships for assembly model retrieval. Procedia CIRP, 50, pp. 472-477.

Mizanur Rahman, S., Bayer, J., & Dengel, A. (2021). Graph-Based Object Detection Enhancement for Symbolic Engineering Drawings. In International Conference on Document Analysis and Recognition, pp. 74-90. Springer, Cham

Moon, Y., Lee, J., Mun, D., & Lim, S. (2021). Deep Learning-Based Method to Recognize Line Objects and Flow Arrows from Image-Format Piping and Instrumentation Diagrams for Digitization. Applied Sciences, 11(21), 10054.

Moreno-García, C. F., Elyan, E., & Jayne, C. (2019). New trends on digitisation of complex engineering drawings. Neural computing and applications, 31(6), pp. 1695-1712.

Okazaki A., Kondo T., Mori K., Tsunekawa S. and Kawamoto E. (1988). An automatic circuit diagram reader with loop-structure-based symbol recognition, in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 10, no. 3, pp. 331-341, https://dx.doi.org/10.1109/34.3898

Roth K. (2001). Konstruieren mit Konstruktionskatalogen: Band 2: Kataloge. 3., Aufl. 2000. Springer-Verlag Berlin Heidelberg https://doi.org/10.1007/978-3-642-17467-4

Shcherban, S., Liang, P., Li, Z., & Yang, C. (2021). Multiclass Classification of UML Diagrams from Images Using Deep Learning. International Journal of Software Engineering and Knowledge Engineering, 31(11n12), pp. 1683-1698.

Vaxiviere, P., & Tombre, K. (1992). Celesstin: CAD conversion of mechanical drawings. Computer, 25(7), pp. 46-54.

VDI 2223 (2004). Systematic embodiment design of technical products. Düsseldorf, VDI

Wen R, Tang W, Su Z (2016). A 2D engineering drawing and 3D model matching algorithm for process plant. In: Proceedings—2015 international conference on virtual reality and visualization, ICVRV 2015, pp 154–159

Yu, Y., Samal, A., & Seth, S. C. (1997). A system for recognizing a large class of engineering drawings. IEEE Transactions on Pattern Analysis and Machine Intelligence, 19(8), pp. 868-890.

Yun, D. Y., Seo, S. K., Zahid, U., & Lee, C. J. (2020). Deep neural network for automatic image recognition of engineering diagrams. Applied Sciences, 10(11), 4005.