Proceedings of the ASME 2022 International Design Engineering Technical Conferences &
Computers and Information in Engineering Conference
IDETC/CIE 2022
August 14-17, 2022, St. Louis, USA

DETC2022/CIE-91043

DATA AUGMENTATION OF ENGINEERING DRAWINGS FOR DATA-DRIVEN COMPONENT SEGMENTATION

Wentai Zhang Quan Chen Can Koz Liuyue Xie Amit Regmi
Soji Yamakawa Tomotake Furuhata Kenji Shimada Levent Burak Kara*
Carnegie Mellon University, Pittsburgh, PA, 15213, USA

ABSTRACT

We present a new data generation method to facilitate an automatic machine interpretation of 2D engineering part drawings. While such drawings are a common medium for clients to encode design and manufacturing requirements, a lack of computer support to automatically interpret these drawings necessitates part manufacturers to resort to laborious manual approaches for interpretation which, in turn, severely limits processing capacity. Although recent advances in trainable computer vision methods may enable automatic machine interpretation, it remains challenging to apply such methods to engineering drawings due to a lack of labeled training data. As one step toward this challenge, we propose a constrained data synthesis method to generate an arbitrarily large set of synthetic training drawings using only a handful of labeled examples. Our method is based on the randomization of the dimension sets subject to two major constraints to ensure the validity of the synthetic drawings. The effectiveness of our method is demonstrated in the context of a binary component segmentation task with a proposed list of descriptors. An evaluation of several image segmentation methods trained on our synthetic dataset shows that our approach to new data generation can boost the segmentation accuracy and the generalizability of the machine learning models to unseen drawings.

INTRODUCTION

2D engineering part drawings are a common, systematic medium for encoding design and manufacturing requirements. In such drawings, major components are the contour lines, dimension lines, extension lines, and measurements (Fig. 1(a)). Contour lines, which can be straight lines, curves, or circles, represent the outer shape and internal structures of the part. Other elements in the drawing quantify the corresponding dimensions, the basis for measurement and manufacturing requirements. The collective information from these elements serves as the basis of various engineering tasks including content-based part indexing [1, 2], cost estimation [3] and process planning [4]. While such drawings are often trivial for humans to interpret, it remains prohibitively challenging to automate a computer interpretation

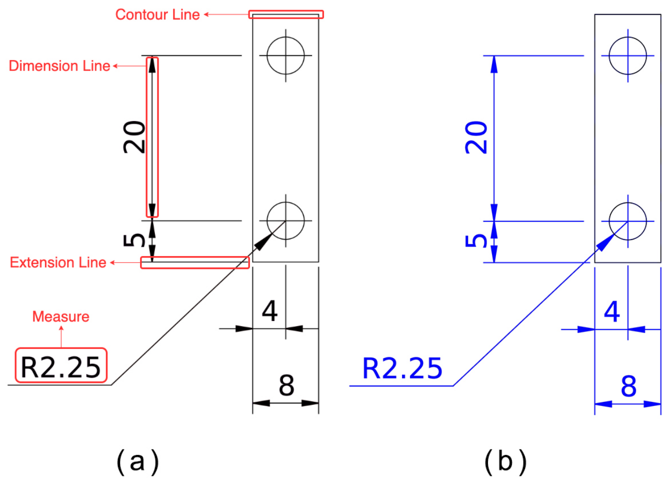

FIGURE 1. (a) Common component types in an engineering drawing. (b) Our target binary segmentation. **Dimension sets** are shown in blue. **Contour lines** are shown in black.

*Address all correspondences to lkara@cmu.edu of them, severely restricting part manufacturers' bandwidth to fulfill the tasks mentioned above.

The majority of part drawings utilized in the industry are encoded in the form of vector or raster images [5]. The drawings are usually constructed in vector format with modern CAD systems and converted to raster images during information exchange. While computerized interpretation is much less challenging with vector data, raster data in the form of black and white images or PDF files still take up the majority in industrial settings. According to a survey on projects and issues in Japan's manufacturing industry [5], 84% of the customers use 2D raster-based drawings in PDF, paper or fax format when placing an order for manufacturing.

In this work, we focus on the challenge of interpreting raster-based part drawings. Specifically, we aim to develop methods to automatically extract component-level information from raster-based part drawings. As a first step toward this goal, we focus on a binary segmentation of a drawing into a group of contour lines versus the remaining dimensional elements (Fig. 1(b)). Our current work excludes other critical elements of technical part drawings including tolerance, material and surface finish specs, cross-sectional views, assembly drawings, tabulated data, and mixed 2D / 3D views.

While trainable computer vision techniques such as object detection [6–8], semantic segmentation [9–11], and visual question answering [12–14] have shown great promise in analogous image interpretation tasks, a lack of adequately labeled part drawing data severely restricts the adoption of such methods. In particular, the acquisition of labeled data for such drawings is a major challenge as it requires humans with requisite technical training to take part in the labelling process.

To address this challenge, we present a new method to automatically generate a large corpus of labeled 2D raster drawings by parsing and generatively randomizing the dimension sets of a few, labeled vector drawings. The initial labeled vector drawings are obtained by manually creating a limited set of part drawings in the vector-based Drawing Exchange Format (DXF format). From these limited number of DXF files, we separate the dimensional elements from the object contour lines, then expand the drawing set by generating new drawings wherein new dimensional elements are generated and placed in novel configurations. Finally, each of the new part drawings is rasterized to strip them off any vector information, thereby producing a large set of labeled rasterized training data.

The synthetically generated data augments the existing data to expand the amount of raster-based training data significantly. To ensure the validity of the generated dimension sets, we present two geometric constraints that prevent overlapping and improper dimensioning. We test the performance of our data expansion approach using three ML models for binary segmentation. We also present a set of twelve features we use to facilitate segmentation as part of the evaluation. These features are extracted from a vectorization applied to the input raster images. We note that while our featurization for ML involves the vectorization of the input raster images; this vectorization is entirely unrelated to how the expanded dataset is generated from DXF files. As such, the input to the ML models are raster images, consistent with the stated goal of this work.

Our results indicate that the synthetic expansion of the labeled training set enables a 30% increase in the binary segmentation accuracy on the validation set over a training set that does not utilize data expansion.

Our main contributions include:

- A method to synthesize new labeled 2D part drawings based on constrained dimension randomization of existing examples.
- An algorithm to extract features as representations for data-driven methods from engineering drawings at the component level.
- A practical system to construct a large dataset for training a data-driven model in the task of component segmentation with a handful of labeled data using the above two ideas.

RELATED WORKS

In this section, we review the background of our work in terms of data augmentation methods on images and data-driven methods for image semantic segmentation. Additionally, we also highlight recent advances in the methods of general diagram recognition.

General Data Augmentation Methods in Computer Vision Tasks

As data-driven methods, especially deep learning methods perform remarkably well in modern computer vision tasks, various data augmentation methods are introduced to improve the generalizability of the models as well as to avoid overfitting. In early works, a common strategy was to conduct a series of image manipulations. Despite the most frequently used geometric transformations such as flipping, cropping, rotation, translation and homographic warping, other manipulations include kernel filtering [15], random erasing [16], color space transformation [17] and mixing images [18]. These manipulation methods are shown to be effective in the majority of computer vision tasks including image semantic segmentation. Unfortunately, most of these manipulations will result in invalid image data in the context of engineering drawings. For example, the drawings are incomplete after random cropping, making it impossible to read, even for human experts.

In other scenarios like autonomous driving, robotic control, and chemical/physics experiments, the data acquisition is limited because the necessary studies are often unsafe, expensive, and/or impractical. Instead, simulations are utilized as an alternative to experiments for generating large amounts of data with numerous experiment conditions. For instance, an agent can be trained within a simulated environment to drive a car (CARLA [19], Udacity [20]) or control a robotic arm (Kuka [21]). Image clips or 3D scenes are fed to the agent model as perception inputs. Simulations also serve as surrogate models to provide training data for predicting a physical process such as fluid dynamics [22,23] and weather forecasting [24]. The simulated data can be generated with flexible experiment conditions in a reasonably short time. Following a similar approach, we aim to create a parametric drawing generator that can synthesize a pool of new drawings in a simulated manner with a handful of existing drawing examples.

Data-driven Methods for Image Semantic Segmentation

Early works in semantic segmentation combined feature extraction (Edge, HOG, SIFT and etc) with weak classifiers. Classifiers like Thresholding [25], SVM [26,27] and KMeans [28,29] are shown to be effective in classifying the detected features. To further improve the results, a Markov Random Field (MRF) can be introduced to the task as a post-processing step [30–34], assuming that the object/class exists as a continuous smooth shape.

With rapid advances in machine learning approaches for computer vision in recent years, a large body of work proposes end-to-end learning frameworks that directly map an input image to a label map. Long et al. [9] introduce the first deep learning framework that yields hierarchies of features in image segmentation. Ronneberger et al. [10] proposed a novel network architecture with skip connections and demonstrated its effectiveness on a biomedical image segmentation task. In most works on deep learning-based methods, CNNs are the best option to serve as an effective automatic feature extractor. In our task, the raster space is sparse in terms of color and textural information. The local features in the pixel level cannot guarantee enough evidence for predicting the component type. An alternative that considers connections between different features at the component level is necessary. As such, we build on these frameworks on image segmentation and extend it to the segmentation of graph of components using data driven models.

Methods for General Diagram Recognition

Similar to engineering drawings, diagrams and flow charts are also common media in electrical engineering, mechanical engineering and many other scenarios to indicate the structure of a system design or the steps of a series of processes. Electric circuit diagram recognition has been explored with finite state machines [35], dynamic programming [36] and various template matching methods. Schäfer et al. [37] propose Arrow R-CNN for flow chart recognition. Other applications include floor plans [38], vibratory mechanical systems [39].

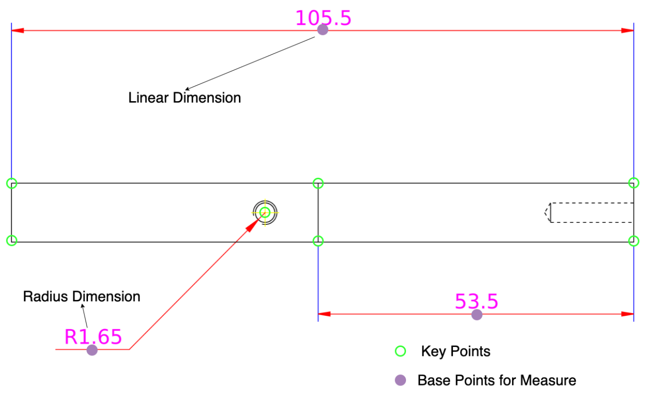

FIGURE 2. Control points for the proposed dimension set generation. All possible key points are marked as green circles. The base points for each dimension measure are marked as purple dots.

Among these works on diagram recognition, template matching methods can usually yield good performance as the symbols and diagram styles are consistent across the same application domain. However, in technical drawings, lines and text are the primary essential elements that comprise different shapes and annotations. The individual recognition of unified symbols cannot provide a full interpretation of the drawings, but it can be used for feature extraction.

TECHNICAL APPROACH

In this work, a handful of engineering drawings in DXF format are utilized as the base for synthetic generation. Note that all the drawings should be constructed with consistent styles and layers in the CAD tool. Then, the component-level information can be parsed and saved in a compact format (JSON) for the convenience of further editing. In engineering drawings, many features are independent of the shape of the part itself, such as dimension label placement, part location on the page, and overall style. Therefore, we develop a data augmentation method by randomizing the dimension set generation without changing the contour shapes. Eventually, a synthetic dataset is generated with our proposed method for the task of component segmentation. To overcome the issue of sparse information space and lack of contextual indication, a component-based representation is introduced with vectorization and feature engineering.[1]

DXF Parsing

For a given DXF drawing, the parametric information of each component is defined and stored in a unified data format

[1]The repository of our method is available at `https://github.com/teddyz829/Data-Augmentation-Engineering-Drawing`

using existing CAD software. A common way to modify this information requires manual inspection and operations in the same software environment, which makes it impossible for the use of an automatic drawing synthesis process. As such, our goal is to directly access and parse the geometric and categorical information stored in the DXF files and generate a lightweight intermediate data format (JSON) to enable a convenient modification of the existing drawings.

In the DXF reader module, we focus on four component types: lines, circles, arcs and dimension sets. For lines, the x and y coordinates of the start and end points are written as the parameters in the JSON file. The line type (solid/dashed) is also recorded. Circles are saved as center coordinates and radius. As a supplement, two perpendicular center lines are also recorded at the center of the circle. Similarly, the arcs are stored with the center coordinates, radius, start angle and end angle. All three types mentioned above are commonly labeled as contour lines. For the dimension sets, linear dimensions and diameter dimensions are considered in our current scope of the work. The linear dimensions are recorded based on two key points for dimension, an orientation and a base point for text measure, see Fig. 2. The diameter dimensions are attached to a circular object with the same center and radius and a base point for text measure.

Through this parsing process, each component's geometric and categorical information is read and saved in the JSON file with only a few parameters. Then a generation module is designed to create new parameter sets as the dimension sets based on given examples. We also implement a rendering module, in which a parametric JSON file is parsed and rendered back to an colored engineering drawing.

Dimension Sets Randomization

For a given drawing, all parametric component information is extracted and saved in JSON format through the reader module. Then a randomization algorithm is introduced to synthesize new dimension sets in the parametric space. In this work, the end points of each line and the centers of the circles are considered as key points for the generation of dimensions sets (Fig. 2).

To maintain a similar data complexity in the dimensioning, the total number of linear dimension sets is randomly chosen within the range of ±20% of the number in the original drawing. During our experiments, 5%, 10% and 20% are explored and compared through visual inspections. As a result, 20% is selected to ensure adequate diversity in the generated drawings. In an iterative manner, a linear dimension set is generated with a randomly selected pair of key points. The orientation is randomly set (horizontal/vertical) if two key points are not aligned horizontally or vertically. Here, we compare the distance from the key points to the bounding box of the entire contour shape and place the dimension on the near side (Fig. 3). This algorithm is detailed in Alg. 1. Similarly, the circular dimensions are

Algorithm 1 Dimension Sets Randomization Algorithm for Linear Dimension sets(DSR)

```
1:  **Input:** a Json file containing all the information read from the original DXF file
2:  **Parameters:** $N_{dg}$, the total number of generated dimension sets. $N_{do}$, the total number of dimension sets in the original drawing. $Pool_{KP}$, the pool of key points read from the original DXF file.
3:  **Output:** a synthetic DXF file with newly generated dimension sets
4:  **procedure** DSR($N_{dg}, N_{do}$)
5:      $N_{dg} \leftarrow$ **Int(random(**$0.8N_{do}, 1.2N_{do}$**))**
6:      **while** $i \leq N_{dg}$ **do**   ▷ iteratively generate a new dimension set
7:          $K_1^i, K_2^i \leftarrow$ Sample($Pool_{KP}$)   ▷ randomly pick two key points from the pool
8:          randomly pick an orientation (Vertical/Horizontal)
9:          calculate the two distances for two placements: $d_{p1} = d(K_1^i, C_{contour})$, $d_{p2} = d(K_2^i, C_{contour})$
10:         Select the near side as the direction for the dimension set $argmin(d_{p1}, d_{p2})$.
11:         Sample a distance $d_b^i \in [0, d_{max}]$, $d_{max}$ is determined by the key points location and page size. The base point for measure $P_b^i$ is determined given the key points, orientation and direction. The bounding box $B_b^i$ of text measure for $P_b^i$ is recorded as $[P_b^i, h_f, H_i]$, where $h_f$ is the font size, $H_i$ is the horizontal distance between $K_1^i, K_2^i$.
12:         Check if $B_b^i \cap B_b^m \neq \emptyset$, where $B_b^m$ is the bounding box of any dimension set in the hashset $S_b$. If yes, go to 11.
13:         Check if $P_b^i \in C_{contour}$. If yes, go to 11.
14:         Save $K_1^i, K_2^i, P_b^i$, orientation(binary) to the Json file. Save $B_b^i$ to the hashset $S_b$.
```

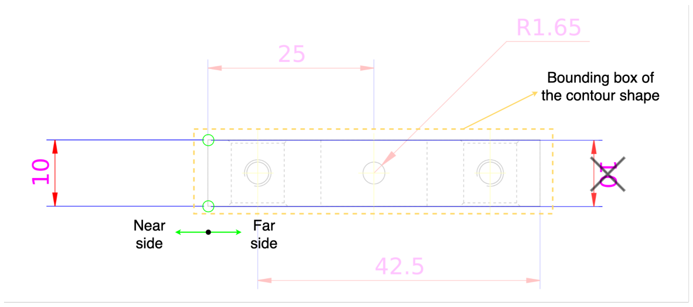

FIGURE 3. An example of our dimension placement strategy. For two chosen key points (shown as green circles), there are two directions to generate a linear dimension set. Based on the bounding box of the contour shape (shown in yellow dashed lines), the left one is generated since it lies along the near side.

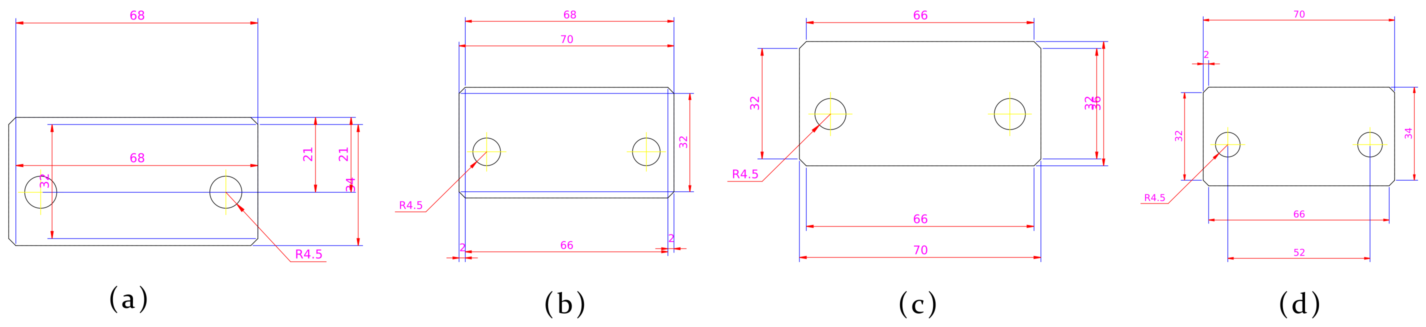
FIGURE 4. Comparison between synthetic data generated under four different conditions: (a) Unconstrained, (b) C1 only, (c) C2 only, (d) Fully constrained

generated with a selected radius center and base point. Although it provides substantial flexibility in the synthesis, the generated drawing can still be too invalid or unrealistic if no constraints are applied. So, we also define a series of rules adaptive to the original labeled data to ensure the quality of the synthetic drawings.

In real drawings, the overlap between different text labels should be avoided to prevent confusion for human reading. Therefore, the first constraint (C1) is that there should not be overlapping among the generated dimension sets. In the iterations of dimension randomization, the bounding box of existing dimension digits is stored in a hashset. Each time a new dimension set is generated, it must pass a conflict check with the existing bounding boxes in the hashset. If a conflict is identified, a new dimension set will be regenerated until it passes the check.

Apart from the overlapping among the dimension sets, human designers usually avoid dimension sets within the contour shape. Most of the dimension sets are placed around each view of the part. Therefore, another constraint (C2) is added to our synthesis algorithm that the dimensions should locate outside of the contour shape if possible. In the implementation, the bounding box of the contour shape is extracted from the DXF file and serves as a restriction area for dimension generation. A visual comparison of sample synthetic drawings under different constraints is demonstrated in Fig. 4.

Another rule of dimensioning is to place the longer dimensions outside of all intermediate dimensions to avoid the crossing between extension lines and dimension lines. However, in preparing the datasets for training a data-driven model, we don't enforce this rule as a constraint; neglecting this rule produces more challenging cases in the training set while preserving the readability. From a computational perspective, this rule is stricter than the previous two constraints because it requires correct placements for a series of dimension sets, which prolongs the runtime needed to synthesize a valid drawing.

Synthetic Dataset Generation

With the strategy mentioned above, we are able to synthetically generate 100 drawings with various dimension set layouts out of each existing DXF drawing. This number can be adjusted based on the need for training. A parametric study is also conducted to explore the effect of this number on the final model performance. A sample synthesis result is shown in Fig. 7. More results are demonstrated in Appendix A. In our work, 32 sheet metal part drawings are given as exemplar shapes, which are split into 25/7 as train/validation set. After the synthesis process, 2500 synthetic drawings are generated for training. The remaining 700 synthetic drawings and 7 original drawings with different contour shapes are utilized separately in the validation. Next, the generated DXF files are converted to PNG files with two colors to indicate the ground truth type of the components (Fig. 6a). For consistency, all the drawings are resized and placed in the middle of a 600×600 white canvas with an offset $[a, b]$, where a and b are randomly sampled from $[-30, 30]$. These files are served to automatically obtain the labels for the synthetic data. Another set of grayscale PNG files is consecutively generated as the input data for data-driven models.

Component Feature Extraction

Due to the differences between engineering drawings and natural images mentioned in the introduction, it is extremely

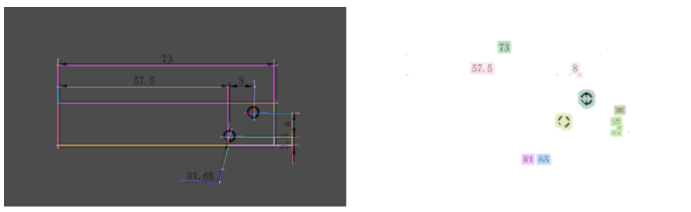
FIGURE 5. Sample vectorization results for Hough line detection (Left) and island detect (right).

TABLE 1. List of features proposed in our feature extraction process.

Index	Symbol	Notion
1	X_1	x coordinate of the upper left corner points of the bounding box.
2	Y_1	y coordinate of the upper left corner points of the bounding box.
3	X_2	x coordinate of the lower right corner points of the bounding box.
4	Y_2	y coordinate of the lower right corner points of the bounding box.
5	L	Diagonal length of the bounding box of the component. The length is normalized by the diagonal length of the image.
6	r	Aspect ratio of the bounding box. length (x range)/height (y range) is used for consistency.
7	P_b	Percentage of black pixels within the bounding box.
8	P_{bp}	Percentage of black pixels in the projection of the components. The components are projected along the axis with smaller range.
9	D_a	Average distance of the 4 nearest neighboring components.
10	D_{std}	Standard deviation distance of the 4 nearest neighboring components.
11	COV	Coefficient of variation. The standard deviation of the distances from the black pixels in a component to its center of gravity. This feature is introduced to indicate the symmetry.
12	M_Z	Zernike Moments of the components. 8 degrees are utilized to generate 25 response features. These features are able to indicate the local gradient orientation of the components.

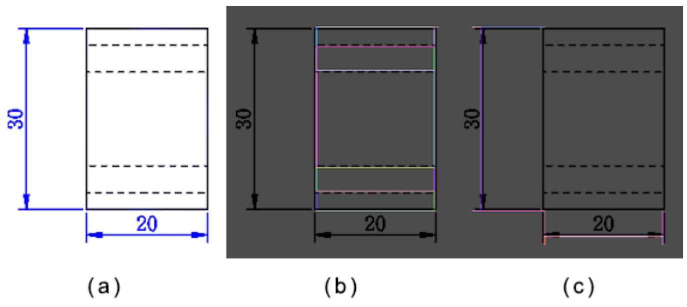

FIGURE 6. The process to assign the ground truth label for each obtained vector. By indexing the color information in the same region from a generated drawing image (a), contour vectors (b) and dimension set vectors (c) are labeled by majority voting.

challenging to get both component and contextual information from the pixel space using an end-to-end framework like CNNs. In this work, we are inspired by the way human designers read such drawings. First, the lines, circles and text information are detected as the basic elements that form different components. Then a set of descriptors are constructed based on the location, size and information from neighboring elements. Our proposed feature extraction process contains vectorization and featurization.

Vectorization: In engineering drawings, line segments are the most frequently used basic elements that form contour lines, extension lines and dimension lines, etc. Therefore, our vectorization method mainly focuses on machine vision-based line detection for all the line elements. A fine-tuned Hough line detector from OpenCV is utilized to find straight lines in the drawing. During parameter tuning, two critical parameters, the minimum line length and the maximum line gap, are adjusted to avoid vectorizing the strokes of the text. The parameters of our implemented line detector are detailed in Appendix B. The resulting detected lines are then filtered based on size heuristics ($1/150$ of the page size) to remove the isolated lines that are either redundant or too short. During this step, we also combine the line

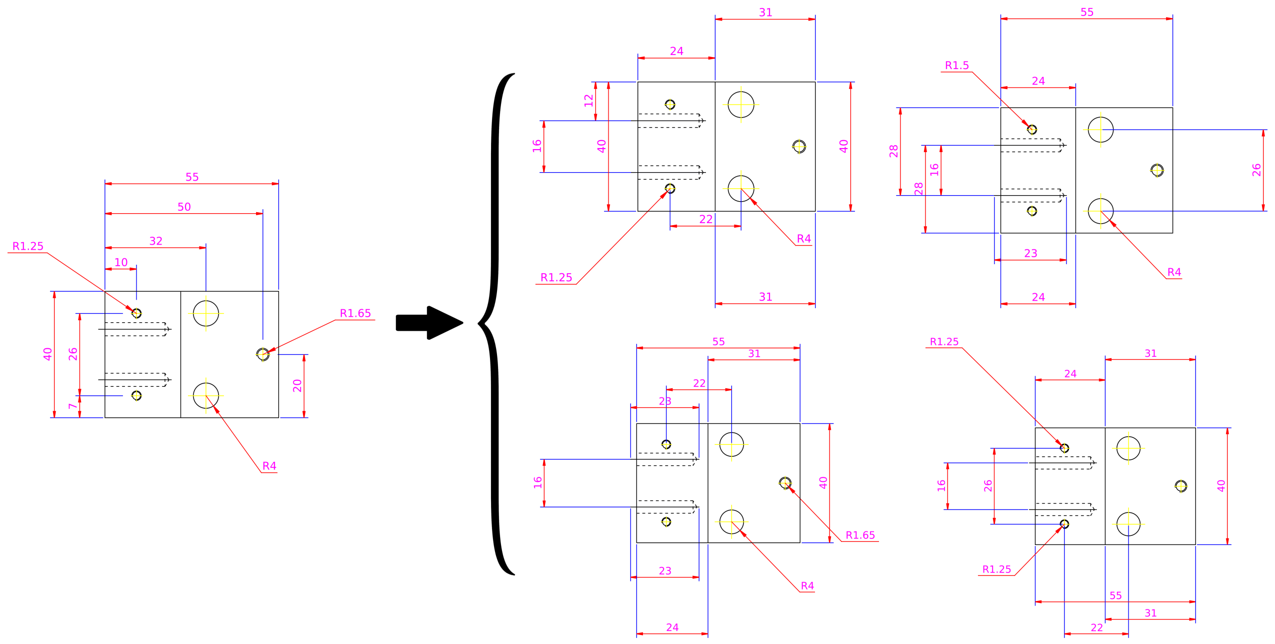

FIGURE 7. Sample data results generated from our proposed constrained data synthesis method. Left: A given drawing in DXF format. Right: 4 synthetic drawings with dimension set randomization.

segments that lie on the same underlying straight line and split a line when there is an intersection point. Eventually, we can get the line segments in the drawing (See Fig. 5 left). Then, all the detected line segments are removed by coloring the line region with white so that the non-line elements can be extracted as isolated islands in the remaining pixel space (See Fig. 5 right). In the scope of our current dataset, the islands include circles, arcs, arrow heads, text information and line segments that are missing in the previous detection step. The islands are detected using a contour detector based on a flood-filling algorithm from OpenCV. The obtained contours are then filtered and grouped by a size and distance heuristic similar to the line detection. Finally, each detected line segment or non-line island is treated as a basic component in the drawing. The ground truth label for each component is determined by indexing the color of the same region in the original drawing (See Fig. 6).

Featurization: After vectorization, pixel information in the original drawing is converted to either line segments or non-line islands. Inspired by [40–42], a set of features are specifically designed according to our vectorization results for providing the basic geometric and contextual information. The feature list is detailed in Tab. 1. In summary, features 1 to 6 are introduced to capture the basic geometric information like the location, size and orientation of the component. Features 7 and 8 provide the density information of the component, which can usually indicate a dashed line. Features 9 and 10 are based on the nearest neighbors of each component so that the context information is taken into account. Lastly, the coefficient of variation (COV) and Zernike Moments are introduced to demonstrate the symmetric and local gradient information so that the circles, arcs and text information can be easily distinguished.

RESULTS

This section illustrates the effect of our proposed drawing synthetic algorithm with multiple common classification models. To further validate the necessity of our designed constraints, we also demonstrate the results using various constraint conditions. Finally, a parametric study is conducted to show the effect of the number of synthetic drawings on the model performance.

Component Segmentation Performance Using Common Classifiers

In Tab. 2, the validation accuracy on two test sets using MLP (Multi-layer Perceptron), DT (Decision Tree) and RF (Random Forest) models are demonstrated. For the MLP model, a single hidden layer fully connected neural network with 100 hidden nodes and ReLU activation is utilized. The model is trained with Adam [43] optimizer and learning rate of 0.001. An early stopping strategy is adopted with the maximum number of iterations $10,000$. For the DT model, a decision tree model with 10 as maximum depth is trained using Gini impurity as the metric. The minimum number of sample split is 3. For the RF model,

TABLE 2. Comparison between our classification methods in two test scenarios. **MLP**: Multi-layer Perceptron. **DT**: Decision Tree. **RF**: Random Forest. The test performance is measured on **unseen original**: seven unseen original drawings in the test set and **unseen synthetic**: 700 synthetic drawings from the seven real drawings.

Validation Accuracy %	MLP	DT	RF
Unseen Synthetic	76.84	86.29	87.52
Unseen Original	74.72	82.71	83.78

the basic tree model is the same as the DT model. The maximum number of features selected for each tree is the square root of the total number of features. 40 such tree models are trained and combined in a bootstrap manner. Finally, the performance of all trained models is evaluated by the accuracy of predicted labels for each component in the test drawing sets.

As shown in Tab. 2, the tree-based methods yield better results than the simple MLP model. The difference may result from the fact that some of the numerical features are close to categorical features in the drawings. For example, the aspect ratio of the component bounding box is highly quantized since over 80% of the components are either horizontal lines (r close to 0) or vertical lines (r close to infinity). Tree-based methods are known to handle categorical/discrete features better in classification and regression tasks. Additionally, we can conclude that the performance on the unseen synthetic dataset is better than on the unseen real dataset as expected. Still, this demonstrates the effect of our synthetic methods on improving the generalizability of any model to a broader set of unseen data with only a handful of labeled examples.

Model Performance Using Datasets with Different Constraint conditions

From Tab. 2, it can be concluded that the RF model yields the best performance out of three methods. Therefore, it is used in the extensive experiment to analyze the effect of our designed constraints for data synthesis. Five different constraint conditions are analyzed in the experiment. The baseline model is trained only on the real labeled drawings to provide accuracy if we do not conduct any synthesis for data augmentation. Then, another model is trained from the synthetic dataset generated without any constraints. Two other comparison models trained from partially constraint conditions (only C1 or only C2) are also introduced to illustrate the influence of each constraint individually.

Tab. 3 summarizes the validation accuracy of models trained from the aforementioned data synthesis constraints. A major improvement (like 87.52 vs 58.19 for RF) can be seen when our proposed synthesis method is introduced. Note that the accuracy on the unseen validation set is only slightly better than the random baseline (50%) if we only train with several real drawings with the tree-based methods. The change in accuracy demonstrates that our proposed synthetic data augmentation greatly improves the diversity of the dataset and the generalizability of the trained model to unseen new data.

As we compare the accuracy of models using fully constrained, partially constrained and unconstrained datasets, it can be concluded that both C1 and C2 contribute to a marked improvement by regularizing the random dimension sets with valid prior assumptions. Specifically, C1 results in a larger increase in accuracy compared to C2. A potential reason is that the overlapping region between the dimension sets usually leads to poor vectorization results.

Parametric Study on the Number of Necessary Drawings to Synthesize

Theoretically, an infinite number of drawings can be generated using our method with each given example. However, the variation of the synthetic dataset is still upper bounded by the number of drawings being used since the contour shapes remain unchanged. To explain the relationship between the number of synthetic drawings and the resulting model performance improvement, a parametric study is conducted with models trained from four synthetic datasets. The distinctions are the number of synthetic drawings generated from each real given drawing $(5, 20, 50, 100)$. Each model is trained five times with different random initialization, and the mean and standard deviation of the accuracy are shown with lines with error bars on each data point.

For Fig.8, it can be inferred that all three models share a very similar trend in accuracy as the number of drawings increases. This means that the variation of the synthetic dataset has a consistent influence across different classification models. In the figure, the rate of the increase in accuracy gradually levels out as more synthetic drawings are generated. The model performance sees negligible improvement beyond 50 synthetic drawings. Another interesting finding is that the standard deviation of tree-based methods is much less than MLP models since the split of the tree-based models usually only depends on a small subset of critical boundary data during classification. The standard deviation of the MLP models decreases as more drawings are generated, which aligns with our expectation that more drawings help improve the accuracy.

DISCUSSIONS AND FUTURE WORK

In our current work, the effectiveness of our synthesis method is validated in the task of binary component segmentation. As an initial effort, it is beneficial to understand the contour shape and relevant dimensions with our current framework.

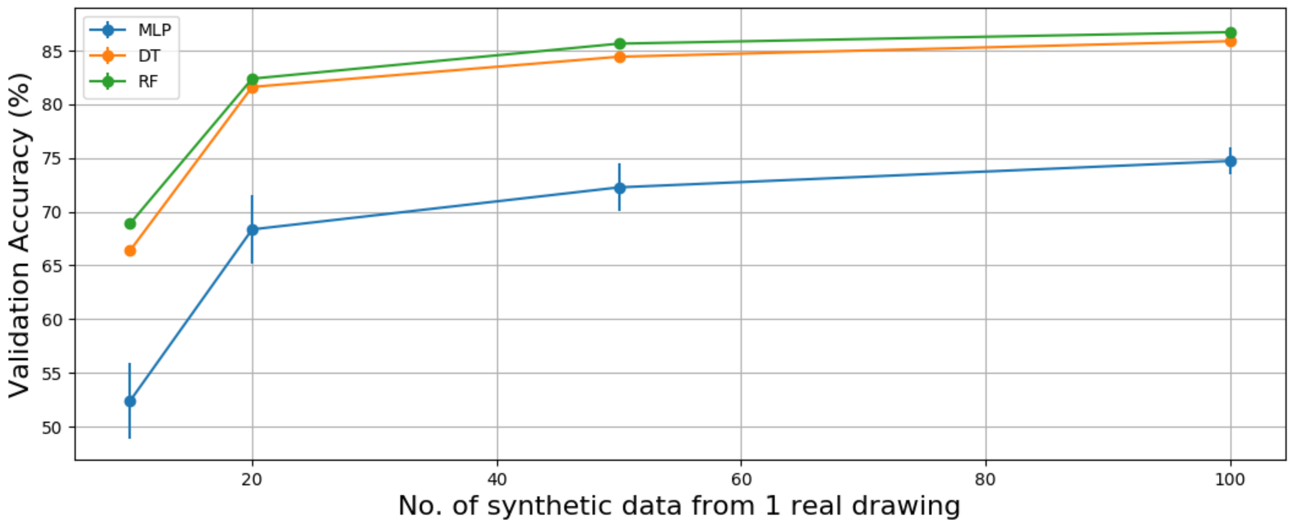

FIGURE 8. Effect of number of synthetic drawings (derived from a single example) on the validation accuracy

TABLE 3. Comparison between the model trained from synthetic datasets under different constraints. The test performance is measured on unseen synthetic drawings.

Validation Accuracy %	RF	DT	MLP
No synthesis	58.19	56.50	45.74
Unconstrained	66.56	64.12	58.03
C1 only	82.12	81.31	70.65
C2 only	80.27	77.84	67.49
C1+C2(fully constrained)	87.52	86.29	76.84

For a broader practical use, tolerance, material and surface finish specs should also be taken into consideration because they usually carry critical information about manufacturing requirements. This will be our immediate next step in further study.

From the results section, it can be concluded that the validation accuracy is bounded by a maximum no matter what type of classifiers are used. The maximum is determined mainly by the variation in the given examples and the accuracy of the preprocessing (vectorization) step. Since our framework is sequential, the overall accuracy is upper-bounded by the weakest module. The Hough line detector we currently use yields much worse results when there exist many large circles and arcs in the drawing. To address this issue, a better vectorization method should be used, taking into account the straight lines as well as curves.

Finally, the components/elements we extracted in the vectorization step are still treated as independent entities, although some of the neighborhood information is considered with two designed features. A more effective way is to construct a component graph based on the connectivity and distance among the components and analyze the graphs to better understand the entire drawing.

CONCLUSIONS

In this work, we present a novel method to synthetically generate a large amount of engineering drawing images for the purpose of training a data-driven model on drawing component segmentation. The method relies on the randomization of the dimension sets in existing vector-based drawings under two self-defined constraints. Results show that the capacity of the trained model to generalize to unseen new geometries is considerably improved when the proposed synthesis method is used for data augmentation. The model performance is enhanced logarithmically as the number of synthetic drawings increases and converges to an upper bound determined by the content diversity of the original data. A similar synthesis framework can also be applied to the training of other data-driven models on engineering drawings. It is possible to extend our proposed method to various tasks such as similarity search, drawing classification, drawing indexing and manufacturing process identification as an effective way of data augdmentation.

ACKNOWLEDGEMENT

The authors would like to thank MiSUMi Corporation for their provision of a contemporary engineering problem, guidance on the applicability of developed methods, and financial support.

REFERENCES

[1] Fonseca, M. J., Ferreira, A., and Jorge, J. A., 2005. "Content-based retrieval of technical drawings". *International Journal of Computer Applications in Technology,* ***23***(2-4), pp. 86–100.

[2] Kasimov, D. R., Kuchuganov, A. V., and Kuchuganov, V. N., 2015. "Individual strategies in the tasks of graphical retrieval of technical drawings". *Journal of Visual Languages & Computing,* ***28***, pp. 134–146.

[3] Sajadfar, N., and Ma, Y., 2015. "A hybrid cost estimation framework based on feature-oriented data mining approach". *Advanced Engineering Informatics,* ***29***(3), pp. 633–647.

[4] Kulkarni, P., Marsan, A., and Dutta, D., 2000. "A review of process planning techniques in layered manufacturing". *Rapid prototyping journal.*

[5] Mitsubishi UFJ Research & Consulting Co., L., 2019. "A survey on projects and issues in Japan's manufacturing industry". `https://www.meti.go.jp/meti_lib/report/2020FY/000066.pdf`.

[6] 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, pp. 779–788.

[7] He, K., Gkioxari, G., Dollár, P., and Girshick, R., 2017. "Mask r-cnn". In Proceedings of the IEEE international conference on computer vision, pp. 2961–2969.

[8] Wang, J., Sun, K., Cheng, T., Jiang, B., Deng, C., Zhao, Y., Liu, D., Mu, Y., Tan, M., Wang, X., et al., 2020. "Deep high-resolution representation learning for visual recognition". *IEEE transactions on pattern analysis and machine intelligence,* ***43***(10), pp. 3349–3364.

[9] Long, J., Shelhamer, E., and Darrell, T., 2015. "Fully convolutional networks for semantic segmentation". In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 3431–3440.

[10] Ronneberger, O., Fischer, P., and Brox, T., 2015. "U-net: Convolutional networks for biomedical image segmentation". In International Conference on Medical image computing and computer-assisted intervention, Springer, pp. 234–241.

[11] Chen, L.-C., Zhu, Y., Papandreou, G., Schroff, F., and Adam, H., 2018. "Encoder-decoder with atrous separable convolution for semantic image segmentation". In Proceedings of the European conference on computer vision (ECCV), pp. 801–818.

[12] Li, L. H., Yatskar, M., Yin, D., Hsieh, C.-J., and Chang, K.-W., 2019. "Visualbert: A simple and performant baseline for vision and language". *arXiv preprint arXiv:1908.03557.*

[13] Jiang, H., Misra, I., Rohrbach, M., Learned-Miller, E., and Chen, X., 2020. "In defense of grid features for visual question answering". In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 10267–10276.

[14] Wang, W., Bao, H., Dong, L., and Wei, F., 2021. "Vlmo: Unified vision-language pre-training with mixture-of-modality-experts". *arXiv preprint arXiv:2111.02358.*

[15] Kang, G., Dong, X., Zheng, L., and Yang, Y., 2017. "Patchshuffle regularization". *arXiv preprint arXiv:1707.07103.*

[16] Zhong, Z., Zheng, L., Kang, G., Li, S., and Yang, Y., 2020. "Random erasing data augmentation". In Proceedings of the AAAI conference on artificial intelligence, Vol. 34, pp. 13001–13008.

[17] Chatfield, K., Simonyan, K., Vedaldi, A., and Zisserman, A., 2014. "Return of the devil in the details: Delving deep into convolutional nets". *arXiv preprint arXiv:1405.3531.*

[18] Inoue, H., 2018. "Data augmentation by pairing samples for images classification". *arXiv preprint arXiv:1801.02929.*

[19] Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., and Koltun, V., 2017. "Carla: An open urban driving simulator". In Conference on robot learning, PMLR, pp. 1–16.

[20] Smolyakov, M., Frolov, A., Volkov, V., and Stelmashchuk, I., 2018. "Self-driving car steering angle prediction based on deep neural network an example of carnd udacity simulator". In 2018 IEEE 12th International Conference on Application of Information and Communication Technologies (AICT), pp. 1–5.

[21] Lukač, D., 2018. "Simulation of a pick-and-place cube robot by means of the simulation software kuka sim pro". In 2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), pp. 0846–0849.

[22] Ummenhofer, B., Prantl, L., Thuerey, N., and Koltun, V., 2020. "Lagrangian fluid simulation with continuous convolutions". In International Conference on Learning Representations.

[23] Kashefi, A., Rempe, D., and Guibas, L. J., 2021. "A point-cloud deep learning framework for prediction of fluid flow fields on irregular geometries". *Physics of Fluids,* ***33***(2), Feb, p. 027104.

[24] Rasp, S., Dueben, P. D., Scher, S., Weyn, J. A., Mouatadid, S., and Thuerey, N., 2020. "Weatherbench: A benchmark data set for data-driven weather forecasting". *Journal of Advances in Modeling Earth Systems,* ***12***(11), Nov.

[25] Kohler, R., 1981. "A segmentation system based on thresholding". *Computer Graphics and Image Processing,* ***15***(4), pp. 319–338.

[26] Wang, X.-Y., Zhang, X.-J., Yang, H.-Y., and Bu, J., 2012. "A pixel-based color image segmentation using support vector machine and fuzzy c-means". *Neural Networks,* ***33***, pp. 148–159.

[27] Yang, Y., Hallman, S., Ramanan, D., and Fowlkes, C. C., 2011. "Layered object models for image segmentation".

IEEE Transactions on Pattern Analysis and Machine Intelligence, ***34***(9), pp. 1731–1743.

[28] Chen, C. W., Luo, J., and Parker, K. J., 1998. "Image segmentation via adaptive k-mean clustering and knowledge-based morphological operations with biomedical applications". *IEEE transactions on image processing,* ***7***(12), pp. 1673–1683.

[29] Dhanachandra, N., Manglem, K., and Chanu, Y. J., 2015. "Image segmentation using k -means clustering algorithm and subtractive clustering algorithm". *Procedia Computer Science,* ***54***, pp. 764–771.

[30] Shotton, J., Winn, J., Rother, C., and Criminisi, A., 2009. "Textonboost for image understanding: Multi-class object recognition and segmentation by jointly modeling texture, layout, and context". *International journal of computer vision,* ***81***(1), pp. 2–23.

[31] Krähenbühl, P., and Koltun, V., 2011. "Efficient inference in fully connected crfs with gaussian edge potentials". *Advances in neural information processing systems,* ***24***, pp. 109–117.

[32] Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K., and Yuille, A. L., 2014. "Semantic image segmentation with deep convolutional nets and fully connected crfs". *arXiv preprint arXiv:1412.7062*.

[33] Zheng, S., Jayasumana, S., Romera-Paredes, B., Vineet, V., Su, Z., Du, D., Huang, C., and Torr, P. H., 2015. "Conditional random fields as recurrent neural networks". In Proceedings of the IEEE international conference on computer vision, pp. 1529–1537.

[34] Chandra, S., and Kokkinos, I., 2016. "Fast, exact and multi-scale inference for semantic image segmentation with deep gaussian crfs". In European conference on computer vision, Springer, pp. 402–418.

[35] Lakshman Naika, R., Dinesh, R., and Prabhanjan, S., 2019. "Handwritten electric circuit diagram recognition: An approach based on finite state machine". *International Journal of Machine Learning and Computing,* ***9***(3).

[36] Feng, G., Viard-Gaudin, C., and Sun, Z., 2009. "On-line hand-drawn electric circuit diagram recognition using 2d dynamic programming". *Pattern Recognition,* ***42***(12), pp. 3215–3223.

[37] Schäfer, B., Keuper, M., and Stuckenschmidt, H., 2021. "Arrow r-cnn for handwritten diagram recognition". *International Journal on Document Analysis and Recognition (IJDAR),* ***24***(1), pp. 3–17.

[38] Delalandre, M., Valveny, E., Pridmore, T., and Karatzas, D., 2010. "Generation of synthetic documents for performance evaluation of symbol recognition & spotting systems". *International Journal on Document Analysis and Recognition (IJDAR),* ***13***(3), pp. 187–207.

[39] Kara, L. B., Gennari, L., and Stahovich, T. F., 2008. "A sketch-based tool for analyzing vibratory mechanical systems". *Journal of Mechanical Design,* ***130***, pp. 101101–1.

[40] Yun, X.-L., Zhang, Y.-M., Ye, J.-Y., and Liu, C.-L., 2019. "Online handwritten diagram recognition with graph attention networks". In International Conference on Image and Graphics, Springer, pp. 232–244.

[41] Ye, J.-Y., Zhang, Y.-M., and Liu, C.-L., 2016. "Joint training of conditional random fields and neural networks for stroke classification in online handwritten documents". In 2016 23rd International Conference on Pattern Recognition (ICPR), IEEE, pp. 3264–3269.

[42] Van Phan, T., and Nakagawa, M., 2016. "Combination of global and local contexts for text/non-text classification in heterogeneous online handwritten documents". *Pattern Recognition,* ***51***, pp. 112–124.

[43] Kingma, D. P., and Ba, J., 2014. "Adam: A method for stochastic optimization". *arXiv preprint arXiv:1412.6980*.

Appendix A: More Results on Drawing Synthesis

As a supplement to the synthetic drawings shown in the Results section, more generated drawings by our augmentation method are demonstrated in Fig. 9.

Appendix B: Parameters of the Hough Line Detector

The line detection process consists of three major steps. First, a gaussian blur with kernal size 5 is applied to the drawing to smooth the fringe of the lines. Then a Canny edge detector with low threshold 50 and high threshold 150 is utilized to locate possible edges for lines. Finally, the lines are identified using a Hough line detector with distance resolution $\rho = 1$, angular resolution $\theta = 1^\circ$, minimum line length 30 and maximum line gap 15 as hyperparameters.

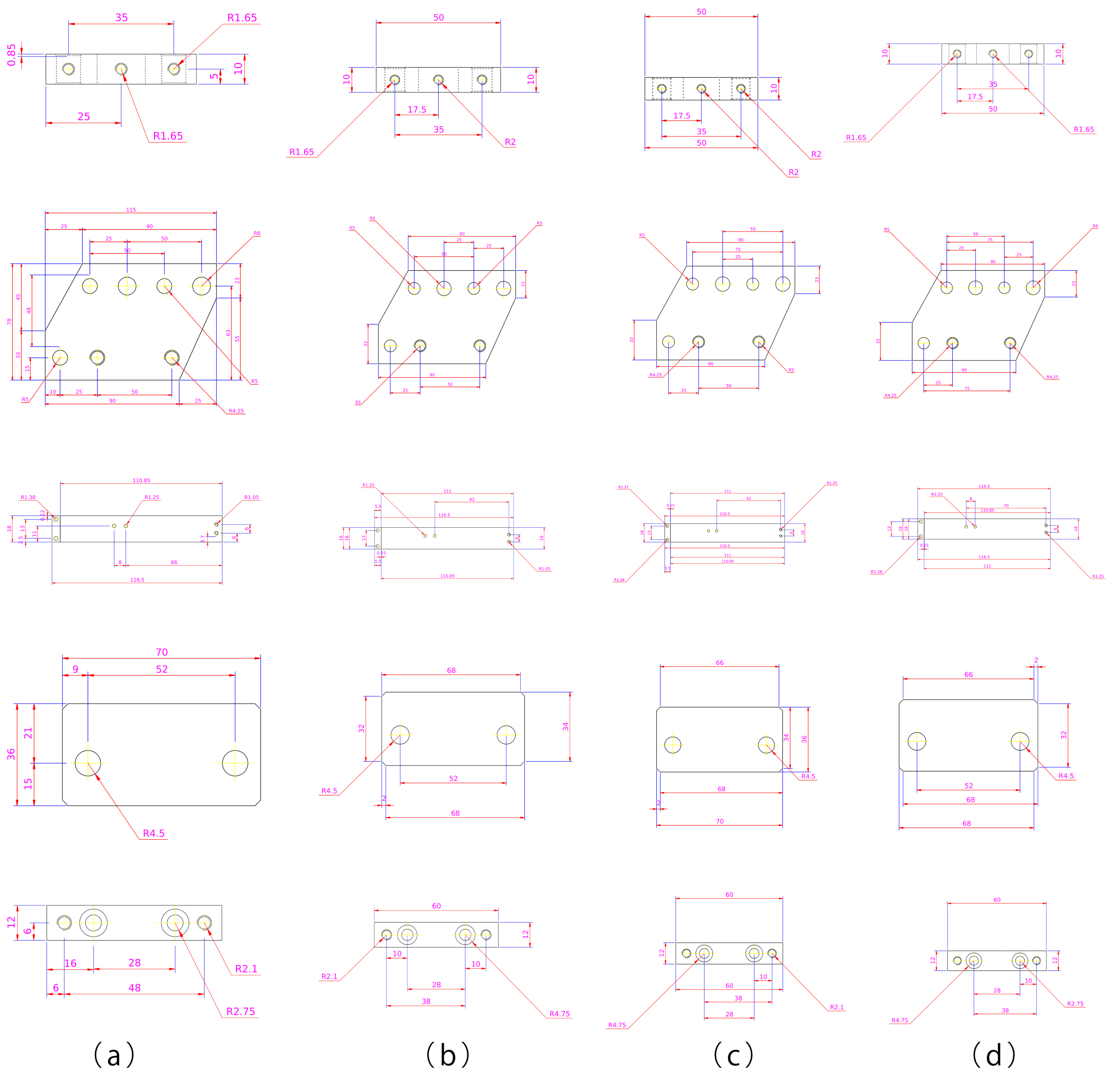

FIGURE 9. Sample synthetic drawings from given examples. Existing drawings are shown in (a), three synthesis results are shown in (b), (c) and (d).