Image to CAD: Feature Extraction and Translation of Raster Image of CAD Drawing to DXF CAD Format

Aditya Intwala(✉)

Symbiosis Institute of Technology, Pune, India
adityaintwala@yahoo.co.in

Abstract. A CAD drawing has various drawing features like entity lines, dimensional lines, dimensional arrows, dimensional text, support lines, reference lines, Circles, GD&T symbols and drawing information metadata. The problem of automated or semi-automated recognition of feature entities from 2D CAD drawings in the form of raster images has multiple usages in various scenarios. The present research work explores the ways to extract this information about the entities from 2D CAD drawings raster images and to set up a workflow to do it in automated or semi-automated way. The algorithms and workflow have been tested and refined using a set of test CAD images which are fairly representative of the CAD drawings encountered in practice. The overall success rate of the proposed process is 90% in fully automated mode for the given sample of the test images. The proposed algorithms presented here are evaluated based on F1 scores. The proposed algorithm is used to generate user editable DXF CAD file from raster images of CAD drawings which could be then used to update/edit the CAD model when required using CAD packages. The research work also provides use cases of this workflow for other applications.

Keywords: Feature extraction · CAD · Reverse engineering · Machine learning · OCR · Feature recognition

1 Introduction

Even though most of the manufacturing industry uses 3D CAD models to represent the parts, the scope and usage of 2D CAD drawings is still highly prevalent in large section of the industry. Also, there is a huge amount of legacy CAD data locked in the form of 2D drawings either as raster images or hard copies. Also in many situations, like, say manufacturing subcontracting, or even at a manufacturing station, using CAD software to visualize, interpret and use is not possible. In such cases, 2D CAD drawing images prove to be a default mode of representation of the part. Such images could be in the form of the camera snapshot of a hard copy drawing or they could be in the form of exported raster images or PDF files from the CAD system.

It becomes very much essential to process the information contained in a 2D CAD drawing in automated way in many cases to increase the productivity [5], e.g. generation of path plan for inspection or manufacturing, perform incremental modifications in the existing drawings for different versions of the part, identify and plan manufacturing processes in automated way for a given part, generation of 3D CAD model automatically or semi-automatically from multiple view images of 2D CAD drawing, etc.

The Current work is divided in three stages as shown in Fig. 1, first stage being the identification of individual geometric and text entities through image processing, the second stage involves processing the extracted information to find the correlation between the identified entities and dimensional information while the third stage is generating a user editable DXF file from the processed information. Thus, if one can get a blueprint of the product then it can be used to extract features and generate a user editable model for the same. This can be used to modify features of the competitor's/company's model and generating a new updated and improved model for same with fewer efforts.

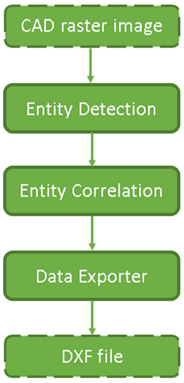


Fig. 1. Stages of proposed workflow

The implementation and experimentation have been performed using Open CV 3.0 library through Python 3. Test data for candidate CAD drawing images are taken from the book Machine Drawing [10].

2 Proposed Algorithm

The input to our algorithm may be a multi-channel colored or single-channel grayscale image of the CAD drawing. If the image is a multi-channel, then it is first converted to single-channel grayscale image and then converted to a binary form by threshold.

Let t be threshold value, q_i be the probabilities of two histogram bins separated by threshold t; μ_i be the mean of histogram bin and σ_i^2 be the variances of two bins and $x(i)$ be the value of center intensity of histogram bin i,

$$q_\mathrm{i}(t) = \sum_{0}^{t} f_\mathrm{g}(i) \tag{1}$$

$$\mu_\mathrm{i}(t) = [\sum_{0}^{t} f_\mathrm{g}(i) * x(i)]/q_\mathrm{i} \tag{2}$$

$$\sigma_\mathrm{b}^2(t) = q_1(t) * q_2(t) * [\mu_1(t) - \mu_2(t)]^2 \tag{3}$$

$$f_\mathrm{t} = T(f_\mathrm{g}) = \begin{cases} 255, & f_\mathrm{g}(i,j) \geq ThresholdValue \\ 0, & f_\mathrm{g}(i,j) < ThresholdValue \end{cases} \tag{4}$$

Where i, j represents the row and column of pixel of the image.

2.1 Entity Detection

After pre-processing the input image to threshold image, we detect various entities like dimensional arrow head, dimensional text, lines, circle, GD&T symbols etc. at this stage of the algorithm.

Dimensional arrow heads from the image are detected using the morphological operations [7] such as Black hat and White hat in order to filter out non arrow like objects from the image [9]. The approach is able to detect solid and line type arrow heads as shown in Fig. 2.

$$T_\mathrm{bh}(f_\mathrm{t}) = f_\mathrm{t} \bullet s(x) - f_\mathrm{t} \tag{5}$$

Where, T_bh is a Black Hat function, $s(x)$ is 2×2 kernel and $\bullet$ is closing operator.

$$T_\mathrm{wh}(f_\mathrm{t}) = f_\mathrm{t} - f_\mathrm{t} \circ s(x) \tag{6}$$

Where, T_wh is a White Hat function, $s(x)$ is 5×5 kernel and $\circ$ is opening operator.

CAD drawings contain multiple textual entities in form of dimensions, GD&T symbols [11], Bill of Material etc. This information is required to process the drawing features. The approach is a systematic combination of region based techniques and morphological separation techniques. Here we first search for regions where the textual entities are present. Initially using the region based approach, we filter out most non-textual entities from the image and in a second step we apply a morphological separation. A two pass connected component filtering is applied to pixel labelling. In the first pass, each pixel of the binary image is iterative scanned and checked if it's a foreground pixel. If the pixel is a foreground pixel, then the neighboring pixels are checked if there is any other foreground pixels connected to the current pixel. If no neighboring foreground pixels are found, then a new label is assigned to the current pixel or else the label of the neighboring foreground pixel is assigned to the current pixel. In the second

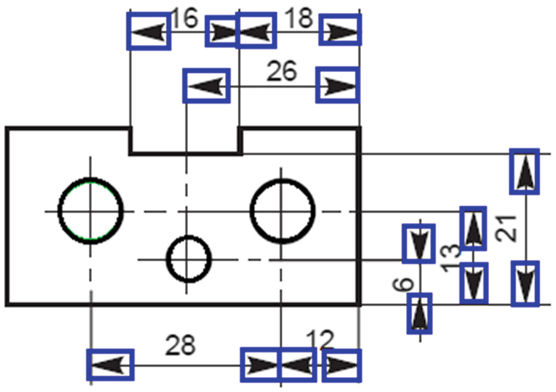


Fig. 2. Dimensional Arrow heads detection

pass the labelled image is again scanned in similar fashion and new label is given based on smallest equivalence of neighboring pixel labels. The algorithm uses the union-find data structure which provides excellent performance for keeping track of equivalence relationships. Union-find store labels which correspond to the same blob in a disjoint-set data structure.

CC_{labels} is a set of connected components which is represented by 2 dimensional array $f_{\text{CC}}(r, c)$ where,

$$f_{\text{CC}}(r,c) = \begin{cases} 1, if(r,c) \in CC_{\text{labels}} \\ 0, if(r,c) \in CC^c_{labels} \end{cases} \tag{7}$$

Where r, c represents the row and column of pixel of the image.

These sorted labelled pixel list, CC_{labels} is returned along with the new connected component image f_{CC}.

In morphological separation step, a set of morphological opening operations is performed in order to differentiate the text area from the background. Morphology Opening of the image is erosion followed by dilation using the same structuring element. It smooths the contours of an image, breaks down narrow bridges and eliminates thin protrusions. Thus, it isolates the object which may just touch another object for effective segmentation. This helps in getting rid connections between dimensional lines and the text or text and pixel representation a drawing entity.

$$f_{\text{o}} = f_{\text{CC}} \circ B = (f_{\text{CC}} \ominus B) \oplus B \tag{8}$$

Where f_{CC} is connected components $\circ$ is Opening operator, $\ominus$ is Erosion operator, $\oplus$ is Dilation operator and B is a rectangular structuring element.

In the next step, a bounding box for each of the candidate region is found. A set of rules is applied on these bounding boxes to differentiate between textual and non-textual regions. The rules are based on the bounding box aspect ratio, critical width and bounding box area. These conditions filter out a majority of non-textual regions from the list. After this each of the candidate region remaining is treated as a Region of Interest (ROI)-that is f_{ROI} as shown in Fig. 3.

We process the separated textual region and run Optical Character Recognition (OCR) using a Convolution Neural Network (CNN) [12]. The CNN is also able to recognize GD&T symbols and store the data in the data-structure for later use.

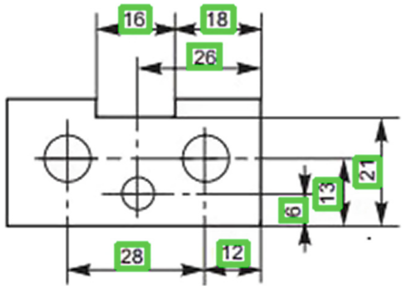


Fig. 3. Textual region detection

Major part of CAD drawing contains lines in various forms, feature lines, support lines, center lines, etc. Thus it is inevitable to detect lines precisely. Here we have used hybrid Hough based line detection where the Hough transform [2] is only used for initial guess. Firstly, corners are detected using Harris corner function [13]. This is used in combination with Hough line function to get the definition of candidate lines. Using this guess, we start scanning the pixels based on the definition of line and counting the number of nonzero pixels on the lines and based on some threshold determine precise unique lines from the candidate lines.

To determine the corner candidate using Harris function which requires determinant and trace of matrix M.

$$M = \sum_{\mathrm{x,y}} w(x,y) \begin{vmatrix} I_{\mathrm{u}}I_{\mathrm{u}} & I_{\mathrm{u}}I_{\mathrm{v}} \\ I_{\mathrm{u}}I_{\mathrm{v}} & I_{\mathrm{v}}I_{\mathrm{v}} \end{vmatrix} \tag{9}$$

Where I_{u} and I_{v} are the derivative of an image in x and y direction.

$$Corner = \lambda_1\lambda_2 - k(\lambda_1 + \lambda_2)^2 \tag{10}$$

where k is some small constant.

A line in image space transforms to point in Hough space (ρ, θ). we could determine such points to get the candidate lines in image space (x, y) as shown in Fig. 4.

$$r = x\cos\theta + y\sin\theta \tag{11}$$

$$y = -(\cos\theta/\sin\theta)x + r/\sin\theta \tag{12}$$

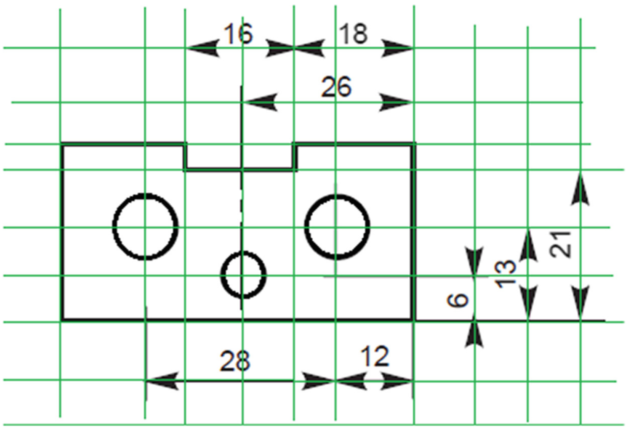


Fig. 4. Line detection.

Nonlinear entities like circles, splines, ellipse, etc. are used to represent surface, holes, circular dimensions etc. in CAD drawing image. Here we use the parametric equations of some of 2D conic shapes like circle, ellipse, parabola and hyperbola to generate the data for training a 4 layer CNN, which is used to predict these shapes. For simple circle detection the approach of using optimized curve fitting using Random sample consensus (RANSAC) gives satisfactory results, but for more robust results CNN can be preferred. The output of this stage is as shown in Fig. 5.

$$x = h + r\cos(t), y = k + r\sin(t) forcircle \tag{13}$$

Where (h, k) is the center of the circle, t is parameter, r is radius of circle.

$$x = h + a\cos(t), y = k + b\sin(t) forellipse \tag{14}$$

Where (h, k) is center of ellipse t is parameter, a is radius in $x - axis$ and b is radius in $y - axis$.

2.2 Entity Correlation

In this stage we correlate the individual extracted information to make sense of the data.

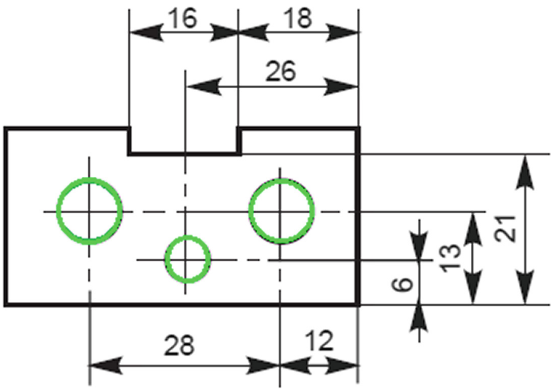


Fig. 5. Circle detection.

We first find the direction of the extracted arrowhead in which it is facing. Here the corners of the arrowhead are extracted and Euclidean distance of all the corners to the center are calculated. The maximum distance is the extreme most points of the arrow head. The direction vector from center to the extreme point gives the direction of arrow head as shown in Fig. 6.

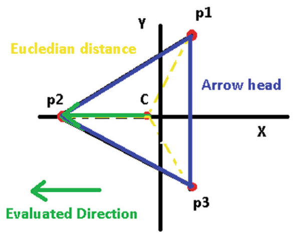


Fig. 6. Arrow head direction vector

For dimensional lines, we use the bounding box of the arrow head and its center to locate lines from the list of extracted lines on which it lies. Thus associating the arrow heads with its dimensional segments.

We correlate the individual text extracted with its dimensional line using the Axis Aligned Bounding Box (AABB) collision approach as shown in Fig. 7. This correlated information is stored in the single associated data structure as dimensions.

The final step would be to associate the dimensions to the individual extracted entity. This is done with the help of arrow head direction and associated support lines which traces back to the entity.

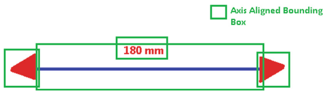
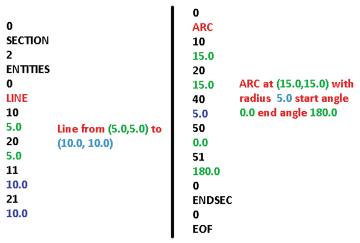


Fig. 7. Axis aligned bounding box association

2.3 Data Exporter

The extracted entity along with the associated dimensions are used to export the data into DXF file format [1]. A DXF file is divided into sections. The data comes in blocks, which is made from pairs of lines. The first line is a number code which represents what to expect next and the second line is the data itself. DXF files starts and ends with special blocks. The sample of line and arc entity is shown in Fig. 8. It could also be exported into DWG or equivalent file format if the structure of the format is known. This exported DXF file is user editable, which can be modified/edited in any CAD packages available.



```
0
SECTION
2
ENTITIES
0
LINE
10
5.0
20
5.0
11
10.0
21
10.0
```

Line from (5.0,5.0) to (10.0, 10.0)

```
0
ARC
10
15.0
20
15.0
40
5.0
50
0.0
51
180.0
0
ENDSEC
0
EOF
```

ARC at (15.0,15.0) with radius 5.0 start angle 0.0 end angle 180.0

Fig. 8. DXF sample block for line and arc

3 Results

The individual algorithms for entity detection and correlation were implemented in Python3 and the set of 15 images was processed by the application generating the output DXF file of the raster images. The generated DXF file could be visualized/modified/edited in any commercial CAD packages. The Fig. 9 shows 2 input raster images and its corresponding exported DXF file as output by the mentioned approach and visualized in open source CAD QCAD package.

Table 1 gives the summary of success rate with F_1 scores of individual entity detection algorithms.

ROC curve gives a good representation of the performance of individual algorithms is given in Fig. 10. Almost 90% of the data was extracted successfully from raster CAD image and was translated to user editable CAD file for further application.

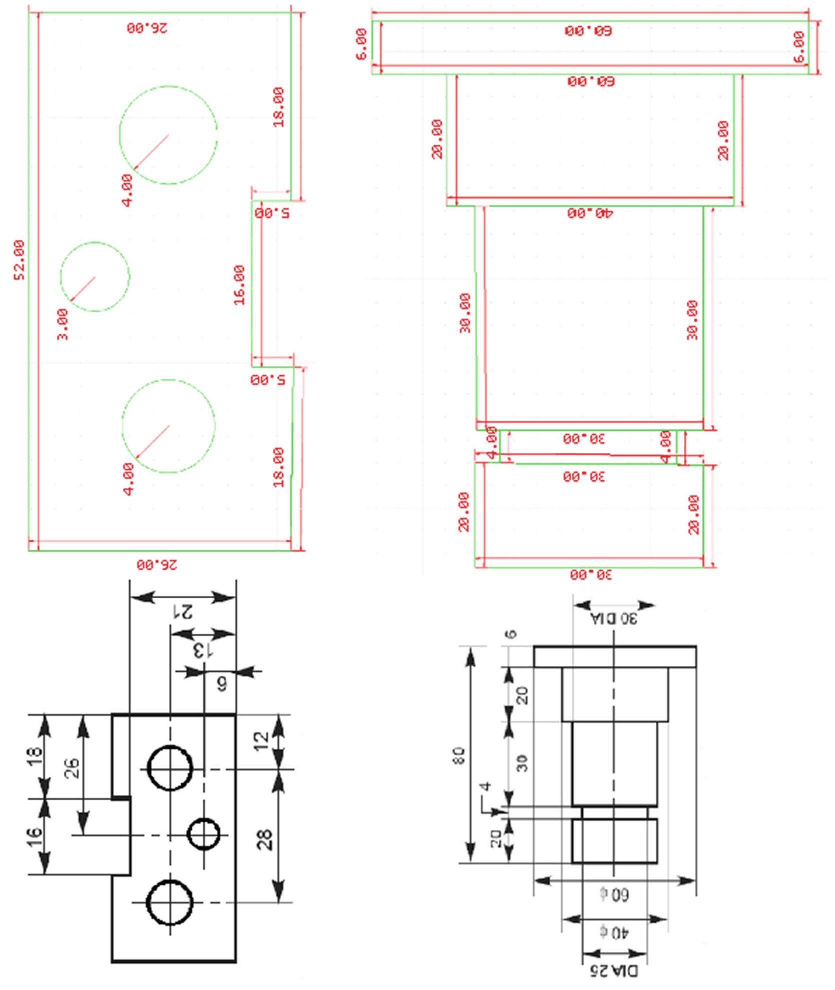


Fig. 9. Outputs of raster images exported to DXF file

Table 1. Precision - recall summary

Algorithm	Precision P = A/(A + B)	Recall R = A/N	F1 Score F1 = 2(P*R)/(P+R)	Area Under Curve (AUC)
Arrow head Detection	0.95952	0.96834	0.95961	0.80
Text Detection	0.926174	0.8625	0.893203	0.83
Line segment Detection	0.87175	0.91542	0.89153	0.79
Circle Detection	0.7944	1	0.8748	0.84

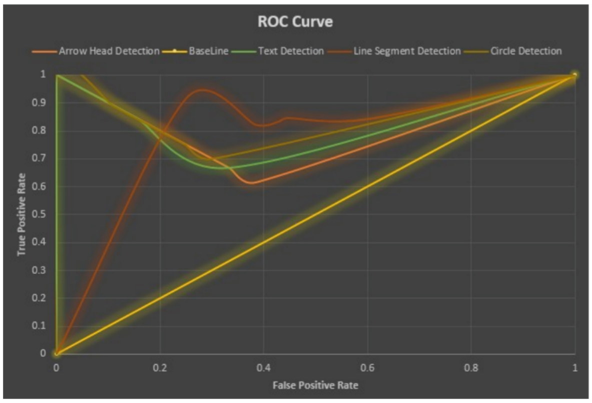


Fig. 10. ROC graphs of entity detection

4 Applications

The extracted information from these raster CAD drawing images can be used for multiple applications

- Digitization of legacy CAD drawing data which is still locked as hard copy drawings (as presented in this paper).
- Generation of inspection plans for Coordinate Measuring Machines (CMM) from raster image of CAD models for inspecting a part.
  - Here using the extracted dimensions and GD&T information we could generate measurement strategies for calculation of actual linear measurement and tolerance information, Execute the plan on actual CMM and check if it is within provided tolerance values.
- Generation of CAM tool paths for Computer Numerical Control (CNC) from raster image of CAD models for manufacturing a part.
  - Here using the extracted shape information we could generate CAM tool path, Execute the tool path on actual CNC machine.
- Generation of 3D model by correlating multiple views of the part from raster images of multiple views of the same part [8].

5 Conclusion

The CAD drawing, image data used in this study is a fairly good representation of the practically used data. For such a test data, using the proposed approach we were successfully able to translate 90% of the image to DXF data. Comparison of this approach with some of the similar existing vectorized approaches [3,4,6] is what I target for the future. Currently this approach suffers if the quality of the image is below average or the image captured is skewed. In these cases the accuracy of entity detection reduces due to skew-ness factor and image quality. The solution to this is trying out entire CNN based approach also might improve the translation rate of the application.

References

1. AutoCAD (2012) DXF Reference autocad 2012. https://images.autodesk.com/adsk/files/autocad_2012_pdf_dxf-reference_enu.pdf
2. Ballard, D.H.: Generalizing the hough transform to detect arbitrary shapes. Pattern Recognit. **13**(2), 111–122 (1981)
3. Bessmeltsev, M., Solomon, J.: Vectorization of line drawings via polyvector fields. ACM Trans. Graphics (TOG) **38**(1), 9 (2019)
4. Chen, J., Du, M., Qin, X., Miao, Y.: An improved topology extraction approach for vectorization of sketchy line drawings. Vis. Comput. **34**(12), 1633–1644 (2018). https://doi.org/10.1007/s00371-018-1549-z
5. Chhabra, A.K., Phillips, I.T.: Performance evaluation of line drawing recognition systems. In: 2000 15th International Conference on Pattern Recognition, Proceedings, vol. 4, pp. 864–869. IEEE (2000)
6. Donati, L., Cesano, S., Prati, A.: A complete hand-drawn sketch vectorization framework. Multimedia Tools Appl. **78**(14), 19083–19113 (2019)
7. Goutsias, J., Heijmans, H.J., Sivakumar, K.: Morphological operators for image sequences. Comput. Vis. Image Underst. **62**(3), 326–346 (1995)
8. Intwala, A.M., Magikar, A.: A review on process of 3D model reconstruction. In: International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT), pp 2851–2855. IEEE (2016)
9. Intwala, A.M., Kharade, K., Chaugule, R., Magikar, A.: Dimensional arrow detection from CAD drawings. Indian J. Sci. Technol. **9**(21), 1–7 (2016)
10. Narayan, K., Kannaiah, P., Venkata Reddy, K.: Machine Drawing. New Age International Publishers, New Delhi (2006)
11. Shen, Z., Shah, J.J., Davidson, J.K.: Analysis neutral data structure for GD&T. J. Intell. Manuf. **19**(4), 455–472 (2008)
12. Suzuki, K.: Pixel-based artificial neural networks in computer-aided diagnosis. In: Artificial Neural Networks-Methodological Advances and Biomedical Applications, pp 71–92. InTech (2011)
13. Trajković, M., Hedley, M.: Fast corner detection. Image Vis. Comput. **16**(2), 75–87 (1998)