




## MidcurveNN: Neural Network for Computing Midcurve of a Thin Polygon

Yogesh H. Kulkarni<sup>1</sup> 

<sup>1</sup>Pune, India, [yogeshkulkarni@yahoo.com](mailto:yogeshkulkarni@yahoo.com)

Corresponding author: Yogesh H. Kulkarni, [yogeshkulkarni@yahoo.com](mailto:yogeshkulkarni@yahoo.com)

**Abstract.** Multiple applications demand lower dimensional representation of geometries. Midcurve is one-dimensional(1D) representation of a two-dimensional(2D) planar shape. It is used in applications such as animation, shape matching, retrieval, finite element analysis, etc. Methods available to compute midcurves vary based on the type of the input shape (images, sketches, etc.) and processing (thinning, Medial Axis Transform (MAT), Chordal Axis Transform (CAT), Straight Skeletons, etc.). This paper talks about a novel method called MidcurveNN which uses Encoder-Decoder neural network for computing midcurve from images of 2D thin polygons in supervised learning manner. This dimension reduction transformation from input 2D thin polygon image to output 1D midcurve image is learnt by the neural network, which can then be used to compute midcurve of an unseen 2D thin polygonal shape.

**Keywords:** Midcurve, Encoder-Decoder, Neural Network

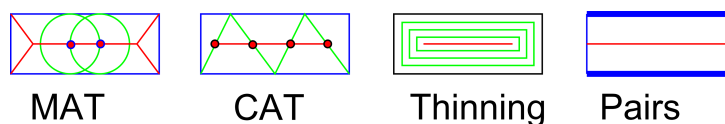
**DOI:** <https://doi.org/10.14733/cadaps.2022.aaa-bbb>

*This template file was made up based on an actual paper written by Orest Mykhaskiv, Jens-Dominik Müller, Pavanakumar Mohanamurthy, Mladen Banovic, Andrea Walther, Salvatore Auriemma and Herve Legrand. The permission to use the paper is appreciated.*

## 1 INTRODUCTION

A skeleton is a lower dimensional entity which represents shape of its parent object. It being simpler than the parent object, operations like pattern recognition, approximation, similarity estimation, collision detection, animation, matching and deformation can be performed efficiently on it than on the parent object.

Skeletons, also known as Medial Objects, can be computed via various mathematical formulations/approaches such as Medial Axis Transform (MAT), Chordal Axis Transform (CAT), Pairing, Thinning etc. Figure 1 shows some of these. More detailed analysis can be found in the survey paper [3].



**Figure 1:** Medial Object computation methods

In the current paper we focus on computing midcurve for 2D planar sketch profiles. Even in 2D profiles, shapes vary enormously. As the first level of simplification, we would deal with 2D polygons only (with an assumption that curved shapes can be converted to polygonal shape by faceting). Figure 2 shows some of the input shapes which can be considered. English alphabets are chosen for easy understanding and verification of the proposed method.

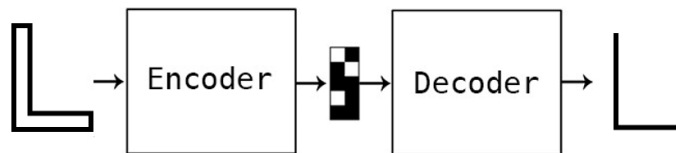


**Figure 2:** 2D Thin Polygonal shapes

## 2 PROPOSED METHOD





































































Computation of midcurve, in its original form, is transformation of a 2D thin closed, with/without-holes polygon to 1D open/closed/branched polyline. Paper [2] details one of the effective midcurve computation techniques, based on rule-based computational geometry approach. Such techniques have a shortcoming of not being scalable or generic enough to be able to handle variety of shapes. Deep Learning neural network architectures are showing potential of developing such generic models. This dimension reduction transformation should ideally be modeled as Sequence to Sequence (Seq2Seq) neural architecture, like Machine Translation. In the current problem, the input and the output sizes could be different not just in a single sample, but across all samples. Many current Seq2Seq neural networks need fixed size inputs and outputs, which if not present in data, are artificially managed by adding padding of certain improbable value. Such padding is not appropriate for the current midcurve computation problem, as the padding-value can inappropriately get treated as part of the valid input. In this initial phase, to circumvent the problem of variable size, image-based inputs and outputs are used, which are of fixed size. Both input and output polygonal shapes are rasterized into images of 100x100, thus making them fixed size for all samples, irrespective of the original shapes.

This paper proposes to use such network for midcurve computation in the form of image-to-image mode. Input images have thin polygonal shapes whereas output images have corresponding midcurve images. Figure 3 shows the Encoder-decoder architecture, called MidcurveNN.



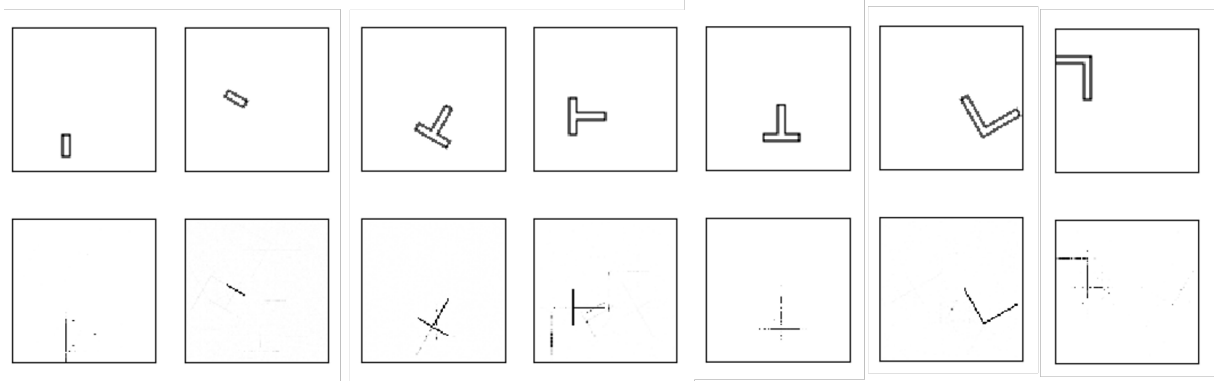
**Figure 3:** Encoder-Decoder Architecture

Epochs	Training loss	Validation loss
50	-17.6354	-8.3223
200	-16.8878	-7.7672

														
L_Profile_mirror_d_1.png	L_Profile_mirror_d_1_rotated_90.png	L_Profile_mirror_d_1_rotated_180.png	L_Profile_mirror_d_1_rotated_270.png	L_Profile_mirror_d_1_translated_10_10.png	L_Profile_mirror_d_1_translated_10_20.png	L_Profile_mirror_d_1_translated_20_20.png	L_Profile_mirror_d_1_translated_20_20.png	L_Profile_mirror_d_5.png	L_Profile_mirror_d_5_rotated_90.png	L_Profile_mirror_d_5_rotated_180.png	L_Profile_mirror_d_5_rotated_270.png	L_Profile_mirror_d_5_translated_10_10.png	L_Profile_mirror_d_5_translated_10_20.png	L_Profile_mirror_d_5_translated_20_10.png
														
L_Profile_mirror_d_5_translated_20_20.png	L_Profile_rotated_90.png	L_Profile_rotated_180.png	L_Profile_rotated_270.png	L_Profile_translated_10_10.png	L_Profile_translated_10_20.png	L_Profile_translated_10_20.png	L_Profile_translated_10_20.png	Plus_Midcurve.png	Plus_Midcurve_mirror_0.png	Plus_Midcurve_mirror_0_rotated_90.png	Plus_Midcurve_mirror_0_rotated_180.png	Plus_Midcurve_mirror_0_rotated_270.png	Plus_Midcurve_mirror_0_translated_10_10.png	Plus_Midcurve_mirror_0_translated_10_20.png
														
Plus_Midcurve_mirror_0_rotated_20_20.png	Plus_Midcurve_mirror_0_rotated_20_20.png	Plus_Midcurve_mirror_1.png	Plus_Midcurve_mirror_1_rotated_90.png	Plus_Midcurve_mirror_1_rotated_180.png	Plus_Midcurve_mirror_1_rotated_270.png	Plus_Midcurve_mirror_1_translated_10_10.png	Plus_Midcurve_mirror_1_translated_10_20.png	Plus_Midcurve_mirror_1_translated_10_20.png	Plus_Midcurve_mirror_1_translated_10_20.png	Plus_Midcurve_mirror_1_rotated_5.png	Plus_Midcurve_mirror_1_rotated_90.png	Plus_Midcurve_mirror_1_rotated_180.png	Plus_Midcurve_mirror_1_rotated_270.png	Plus_Midcurve_mirror_1_rotated_270.png
														
Plus_Midcurve_mirror_1_translated_10_20.png	Plus_Midcurve_mirror_1_translated_20_10.png	Plus_Midcurve_mirror_1_translated_20_20.png	Plus_Midcurve_mirror_1_translated_20_20.png	Plus_Midcurve_mirror_1_translated_20_20.png	Plus_Midcurve_mirror_1_translated_20_20.png	Plus_Midcurve_mirror_1_translated_20_20.png	Plus_Midcurve_mirror_1_translated_20_20.png	Plus_Midcurve_mirror_1_translated_20_20.png	Plus_Midcurve_mirror_1_translated_20_20.png	Plus_Midcurve_mirror_1_rotated_5.png	Plus_Midcurve_mirror_1_rotated_90.png	Plus_Midcurve_mirror_1_rotated_180.png	Plus_Midcurve_mirror_1_rotated_270.png	Plus_Midcurve_mirror_1_rotated_270.png
														

MidcurveNN encoder-decoder has been implemented in Python programming with Keras library [1]. Encoder takes input image of size  $100 \times 100 = 10000$ , then comes Dense layer with size 100 to form the encoded vector. Decoder takes encoded vector as input, then with a Dense layer expands back to  $100 \times 100 = 10000$  size of the output image. Relu activation is used for Encoder whereas Sigmoid for the decoder. AdaDelta optimizer with binary cross entropy as loss function is used to compute the losses. Table 1 shows loss across number of epochs.

Some of the results are shown in Figure 5. Inputs are at the top and output midcurve at the bottom.



**Figure 5:** Predicted Data: Inputs (thin polygons) and outputs (midcurves)

Shape on the top is the input thin polygon whereas the corresponding shape at the bottom is the predicted midcurve. It can be clearly seen that the network is able to localize the shape and learn the dimension reduction function reasonably well. It is still not perfect or usable, as some stray points are still being wrongly classified as the part of output midcurve. A better network model and/or post processing is needed to make output midcurve practically usable.

### 3 CONCLUSIONS AND FUTURE WORK

Traditional methods of computing midcurves are predominantly rules-based and thus, have limitation of not developing a generic model which will accept any input shape. MidcurveNN, a novel Encoder-Decoder network attempts to build such a generic model. This paper demonstrates that simple single layer encoder and decoder network can learn the dimension reduction function reasonably well. Although more development is necessary in evolving a better neural architecture, the current results show positive potential.

Working on truly variable size inputs (thin polygon) and outputs (polyline) using dynamic graph of Encoder-Decoder network can be attempted in the future. More and highly diversified data can help improve the quality of the output. Developing a formal representation of polygonal shapes with variations such as open/closed, with-without loops, branched as a coherent sequence of points is also on the agenda.

Yogesh H. Kulkarni <http://orcid.org/0000-0002-1686-7907>

### REFERENCES

- [1] Chollet, F.: Building autoencoders in keras. <https://blog.keras.io/building-autoencoders-in-keras.html>.
- [2] Kulkarni, A.K.M., Y. H.; Sahasrabudhe: Dimension-reduction technique for polygons. International Journal of Computer Aided Engineering and Technology, 9(1), 2017. <http://doi.org/10.1504/IJCAET.2017.080772>.
- [3] Kulkarni, S., Y. H.; Deshpande: Medial object extraction - a state of the art. In International Conference on Advances in Mechanical Engineering, SVNIT, Surat, 2010.