EXTENDED ABSTRACT ICONIEA 2024

Track 1: Operations Research & Data Analytics: Artificial Intelligence

# Large Language Model for Computing Midcurve of a Thin Polygon

Yogesh Haribhau Kulkarni

AI Coach

yogeshkulkarni@yahoo.com

Track 1: Operations Research & Data Analytics: Artificial Intelligence

# Large Language Model for Computing Midcurve of a Thin Polygon

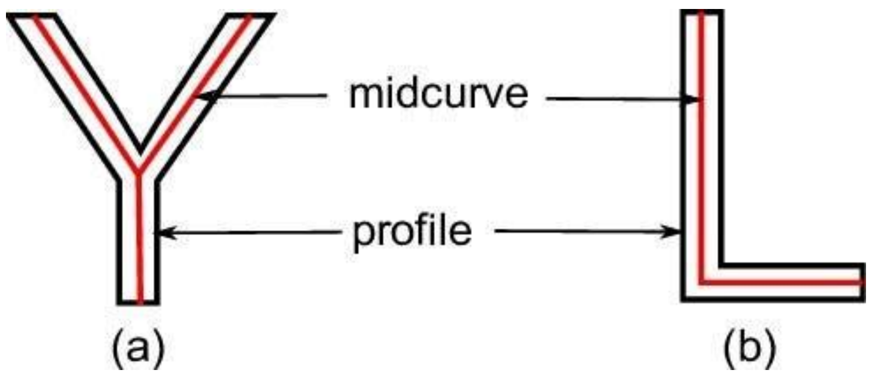
## ABSTRACT

*This paper explores the potential of LLMs (Large Language Models) in computing the midcurve of a 2D geometric profile. The midcurve, a curve equidistant from the bounding curve of the profile, simplifies the representation of the profile-shape while retaining essential geometric information. Despite various approaches such as Medial Axis Transform, Chordal Axis Transform, Thinning and Pairing, the problem remains unsolved due to complexity of shapes and variety of connections. The paper evaluates if the LLMs can be used to generate the midcurve of a 2D geometric profile.*

**Keywords:** Midcurve; Large Language Models; Dimension Reduction; Artificial Intelligence

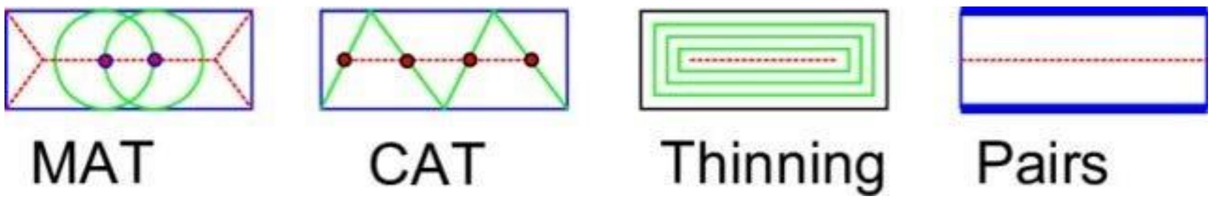
## 1. Introduction

A midcurve of a 2D geometric profile is a curve that lies equidistant from the bounding curves of the profile. It is a curve that represents the “middle” of the profile. The midcurve can be used to represent the shape of the profile in a simpler way than the original profile, while still retaining important geometric information about the profile.



Example of Midcurves of 2D Geometric profiles (ref)

Although many approaches like Medial Axis Transform, Chordal Axis Transform, Thinning, Pairing, etc., have been tried for decades, the problem remains unsolved due to complexity of shapes and variety of connections.



Midcurve Approaches (ref)

Research project MidcurveNN attempts to evaluate if Neural Networks can be used to generate the midcurve of a 2D geometric profile.

## 2. Problem Statement

- **Goal**: Given a 2D closed shape (closed polygon) find its midcurve (polyline, closed or open)
- **Input**: set of points or set of connected lines, non-intersecting, simple, convex, closed polygon
- **Output**: another set of points or set of connected lines, open/branched polygons possible

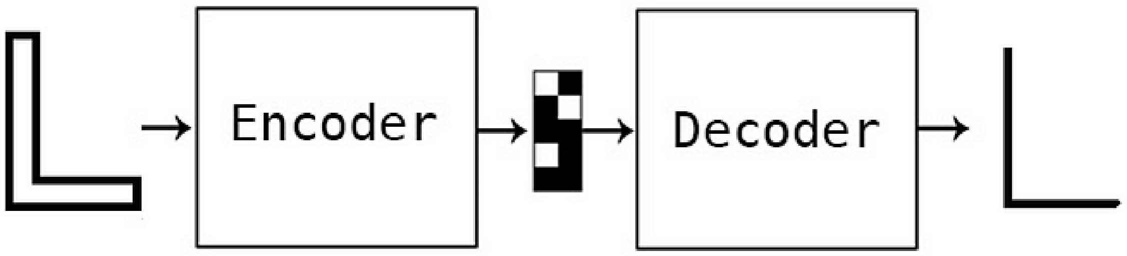
Essentially, if we consider vertices as nodes and lines as arcs then the polygon/polyline profile is nothing but a Graph and thus the midcurve generation is a Graph Summarization/Dimension-Reduction/Compression issue, i.e. reducing a large graph to a smaller graph preserving its underlying structure, similar to text summarization, which attempts to keep the essence.)

Geometry Dimension-Reduction or Compression can be viewed as encoder-decoder problem where larger input is fed to the encoder side whereas reduced output is generated by the decoder side.

To be able to use Neural Network based encoder decoders, both input and output needs to be in a fixed size vector form. So, can variable shape polygons/polylines be fed to the neural networks? Issues are:

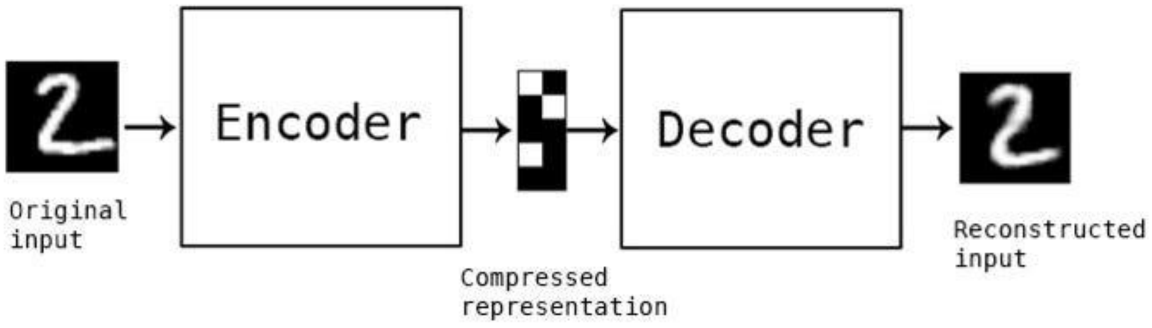
- Shapes cannot be modeled as sequences. Although polygon shape L may appear as sequence of points, it is not.
- All shapes cannot be drawn without lifting a pencil, which is possible for sequences. Say, Shapes like Y or concentric O, cannot be modeled as sequences. So, Midcurve transformation cannot be modeled as Sequence 2 Sequence network.
- How to represent a geometric figure to feed to any Machine/Deep Learning problem, as they need numeric data in vector form?
- How to convert geometric shapes (even after restricting the domain to 2D linear profile shapes) to vectors?
- The closest data structure is graph, but that's predominantly topological and not geometrical. Meaning, graphs represent only connectivity and not spatial positions. Here, nodes would have coordinates and arcs would have curved-linear shape. So, even Graph Neural Networks, which convolute neighbors around a node and pool the output to generate vectors, are not suitable.
- Need RnD to come up with a way to generate geometric graph embedding that will convolute at nodes (having coordinates) around arcs (having geometry, say, a set of point coordinates), then pool them to form a meaningful representation. The real crux would be how to formulate pooling to aggregate all incident curves info plus node coordinates info into a single number!!

So, though graphs are not perfect but with insertion of geometric information they can represent the 2D profiles well. Can Encoder-Decoder architectures (Transformers with Attention) be applied to generate the midcurve?



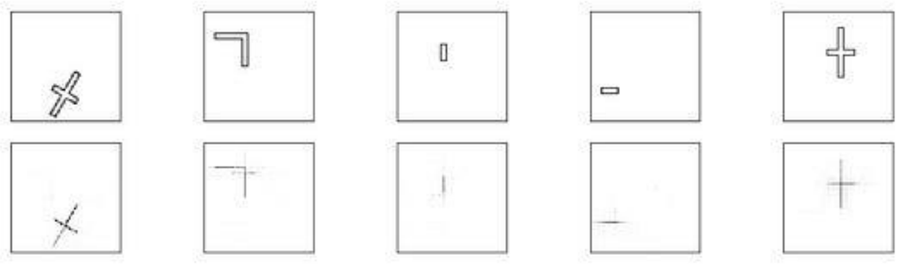
Midcurve by Encoder Decoder (ref)

Such Transformers taking variable length input and output graphs have not been established. So, would taking this problem to image domain help?



Auto Encoder on images (ref)

Various Image2Image networks have been tried by MidcurveNN project but with limited success. Here is a sample:



Simple Encoder Decoder network in Tensorflow/Keras (ref)

Now evaluating if LLMs (Large Language Models), like can be employed to generate the midcurve.

## 3. Proposed Idea

The idea is to give prompt telling what needs to be done along with some examples, and see if LLMs can generate shape for the test example. A few shots prompt developed:

*You are a geometric transformation program that transforms input 2D polygonal profile to output 1D polyline profile.*
*Input 2D polygonal profile is defined by set of connected lines with the format as:*
*input : [line_1, line_2, line_3,....] where lines are defined by two points, where each point is defined by x and y coordinates. So*
*line_1 is defined as ((x_1, y_1), (x_2,y_2)) and similarly the other lines.*
*Output is also defined similar to the input as a set of connected lines where lines are defined by two points, where each point is defined by x and y coordinates. So,*
*output : [line_1, line_2, line_3,....]*

*Below are some example transformations, specified as pairs of 'input' and the corresponding 'output'. After learning from these examples, predict the 'output' of the last 'input' specified. Do not write code or explain the logic but just give the list of lines with point coordinates as specified for the 'output' format.*

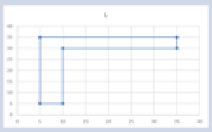
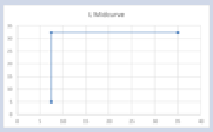
*input:[((5.0,5.0), (10.0,5.0)), ((10.0,5.0), (10.0,30.0)), ((10.0,30.0), (35.0,30.0)), ((35.0,30.0), (35.0, 35.0)), ((35.0, 35.0), (5.0,35.0)), ((5.0,35.0), (5.0,5.0))]*
*output: [((7.5,5.0), (7.5, 32.5)), ((7.5, 32.5), (35.0, 32.5)), ((35.0, 32.5) (7.5, 32.5))]*

*input: [((5,5), (10, 5)), ((10, 5), (10, 20)), ((10, 20), (5, 20)), ((5, 20),(5,5))]*
*output: [((7.5, 5), (7.5, 20))]*

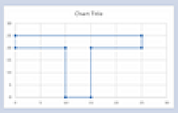
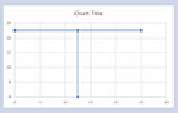
*input: [((0,25.0), (10.0,25.0)), ((10.0,25.0),(10.0, 45.0)), ((10.0, 45.0),(15.0,45.0)), ((15.0,45.0), (15.0,25.0)), ((15.0,25.0),(25.0,25.0)), ((25.0,25.0),(25.0,20.0)), ((25.0,20.0),(15.0,20.0)), ((15.0,20.0),(15.0,0)), ((15.0,0),(10.0,0)), ((10.0,0),(10.0,20.0)), ((10.0,20.0),(0,20.0)), ((0,20.0),(0,25.0))]* *output: [((12.5,0), (12.5, 22.5)), ((12.5, 22.5),(12.5,45.0)), ((12.5, 22.5), (0,22.5)), ((12.5, 22.5), (25.0,22.5))]*

*input:[((0, 25.0), (25.0,25.0)),((25.0,25.0),(25.0,20.0)), ((25.0,20.0),(15.0, 20.0)), ((15.0, 20.0),(15.0,0)), ((15.0,0),(10.0,0)), ((10.0,0),(10.0,20.0)), ((10.0,20.0),(0,20.0)), ((0,20.0),(0, 25.0))]*
*output:*

The first input example above represents 'L' shape (shown below) and the second is an 'I', whereas the 3rd is a 'Plus' sign shape.

| Profile Data | | Profile Picture | Midcurve Data | | Midcurve Picture |
|---|---|---|---|---|---|
| 5.0 | 5.0 |  | 7.5 | 5.0 |  |
| 10.0 | 5.0 | | 7.5 | 32.5 | |
| 10.0 | 30.0 | | 35.0 | 32.5 | |
| 35.0 | 30.0 | | 7.5 | 32.5 | |
| 35.0 | 35.0 | | | | |
| 5.0 | 35.0 | | | | |

The last shape for which LLM has been asked for the answer is actually a 'T' shape. The picture below shows the correct/actual answer as well.

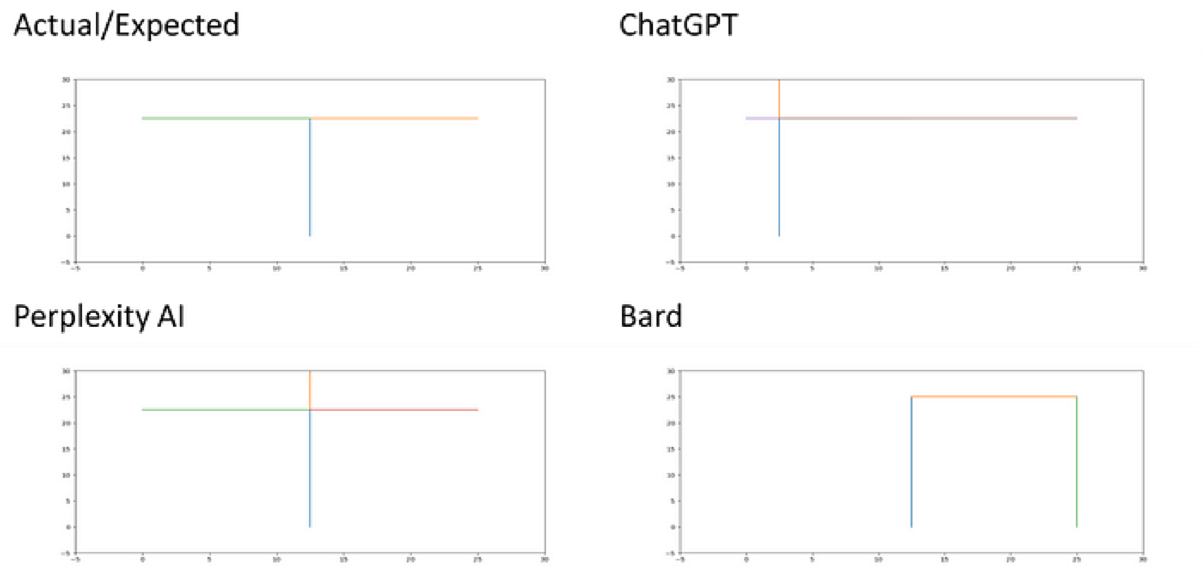
| Profile Data | | Profile Picture | Midcurve Data | | Midcurve Picture |
|---|---|---|---|---|---|
| 0 | 25.0 |  | 12.5 | 0 |  |
| 25.0 | 25.0 | | 12.5 | 22.5 | |
| 25.0 | 20.0 | | 25.0 | 22.5 | |
| 15.0 | 20.0 | | 0 | 22.5 | |
| 15.0 | 0 | | | | |
| 10.0 | 0 | | | | |
| 10.0 | 20.0 | | | | |
| 0 | 20.0 | | | | |

And the outputs computed by various LLMs (ChatGPT, Perplexity AI, Bard) , along with the real/actual answer:

*Actual: [((12.5,0), (12.5,22.5)), ((12.5,22.5),(25.0,22.5)), ((12.5,22.5),(0,22.5))]*
*ChatGPT: [((2.5, 0), (2.5, 22.5)), ((2.5, 22.5), (2.5, 45.0)), ((2.5, 22.5), (25.0, 22.5)), ((2.5, 22.5), (12.5, 22.5)), ((2.5, 22.5), (0, 22.5)), ((2.5, 22.5), (25.0, 22.5))]*

*Perplexity: [((12.5,0), (12.5, 22.5)), ((12.5, 22.5),(12.5,45.0)), ((12.5, 22.5), (0,22.5)), ((12.5, 22.5), (25.0,22.5))]*
*Bard: [((12.5, 0), (12.5, 25.0)), ((12.5, 25.0), (25.0, 25.0)), ((25.0, 25.0), (25.0, 0))]*

Visually here is how results from different LLMs look:



All of the above have failed. Even latest (Oct 2023), the results are:

- llama 7B g4_0 ggml: *(8, 17) & (64, 32)*: Wrong.
- Bard: *[((8.33,5),(8.33, 22.5)), ((8.33, 22.5), (25,22.5)), ((8.33, 22.5), (0,25))]*: Wrong.
- Hugging Chat*: [((12.5, 0), (12.5, 22.5)), ((12.5, 22.5), (25.0, 22.5)), ((25.0, 22.5), (25.0, 25.0))]*: a bit wrong on the last line
- GPT-4:

*[((12.5,0), (12.5,22.5)), ((12.5,22.5),(0,22.5)), ((12.5,22.5),(25.0,22.5))]* just change in sequence of lines, and that's inconsequential, so the answer is correct.

## 4. Conclusion

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. This paper presents a novel Large Language Model based approach which attempts to build such a generic model. One such model, GPT-4, seems to be very effective. Although other proprietary and open-source models need to catch-up with GPT-4, even GPT-4 needs to be developed further to understand not just sequential lines but graphs/networks with different shapes, essentially, the geometry.

## REFERENCES

- MidcurveNN: Encoder-Decoder Neural Network for Computing Midcurve of a Thin Polygon, viXra.org e-Print archive, viXra:1904.0429 http://vixra.org/abs/1904.0429
- CAD Conference 2021, Barcelona, pages 223-225 http://www.cad-conference.net/files/CAD21/CAD21_223-225.pdf
- CAD & Applications 2022 Journal paper 19(6) http://www.cad-journal.net/files/vol_19/CAD_19(6)_2022_1154-1161.pdf