IACC'21
Malta

MidcurveNN

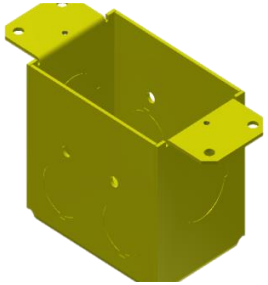
Application of Deep Learning to Geometric Algorithms

Yogesh Kulkarni

Introduction

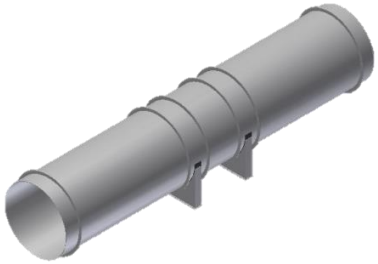
Strength Analysis by CAE

Aerospace

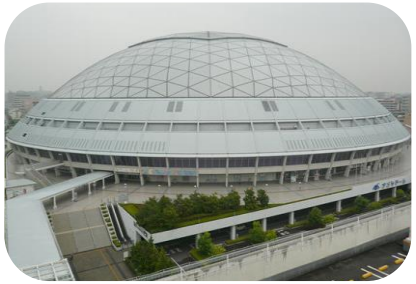


Machinery

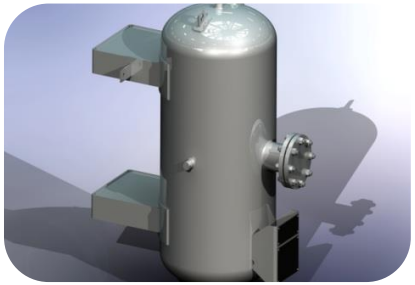
Consumer



Energy



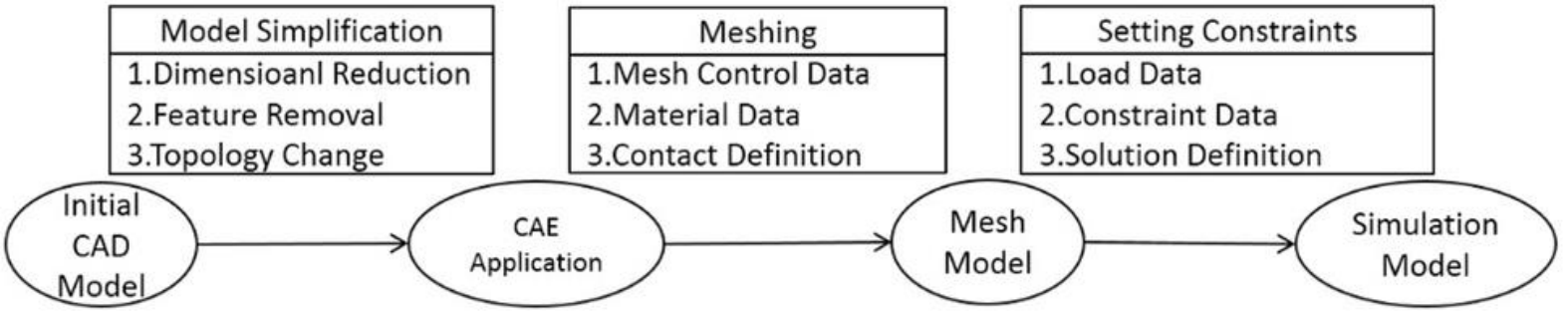
Construction



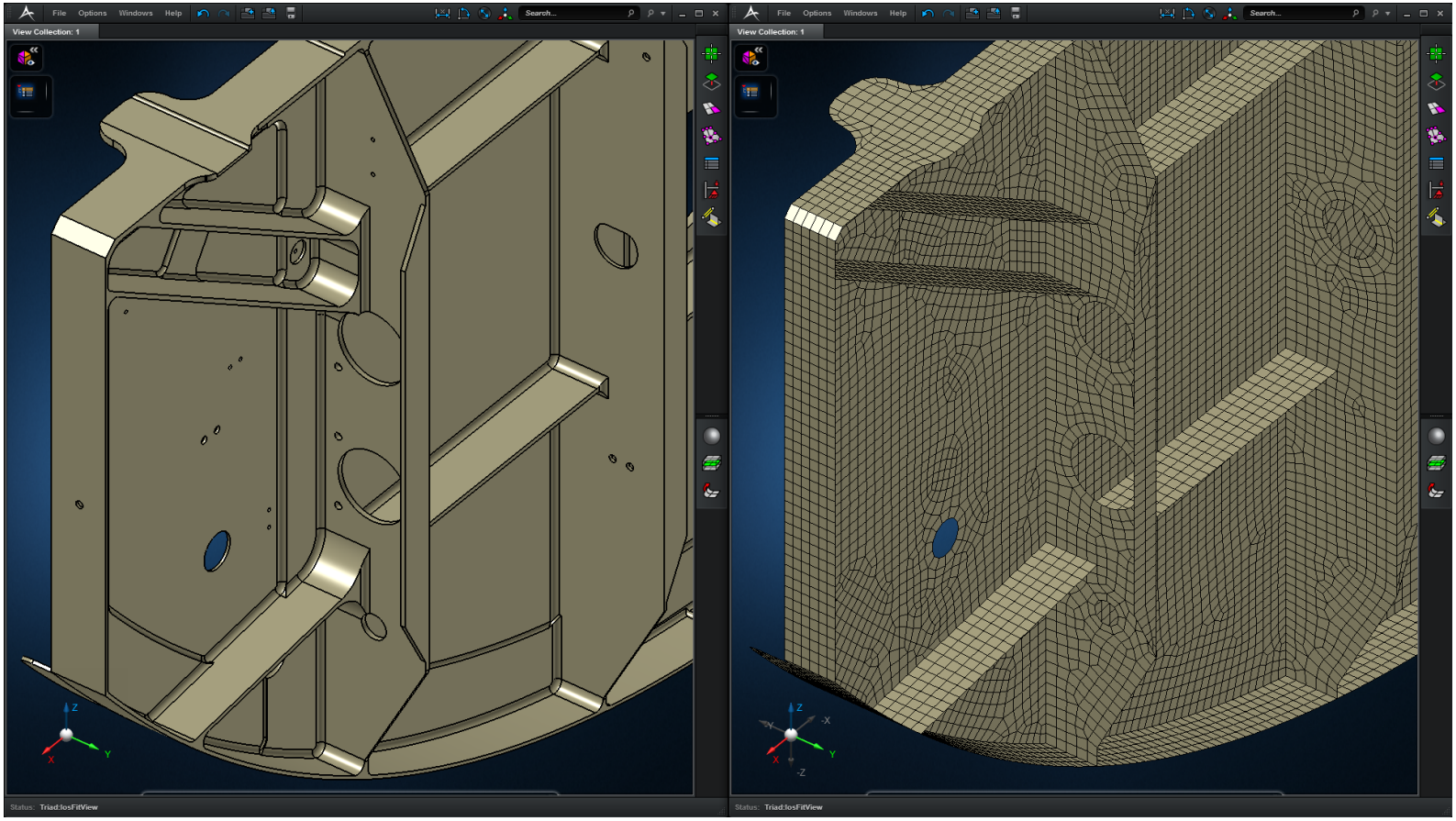
Industrial

Can we use shapes directly?

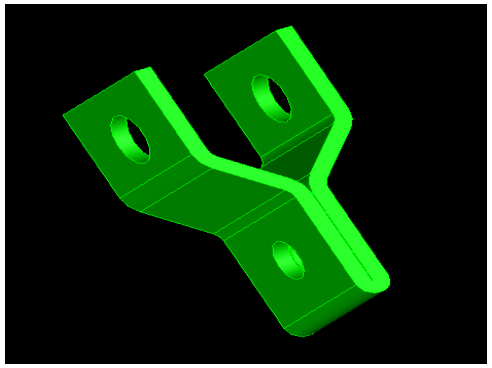
- CAD : Designing Shapes
- CAE : Engineering Analysis
- CAD->CAE: Simplification for quicker results.



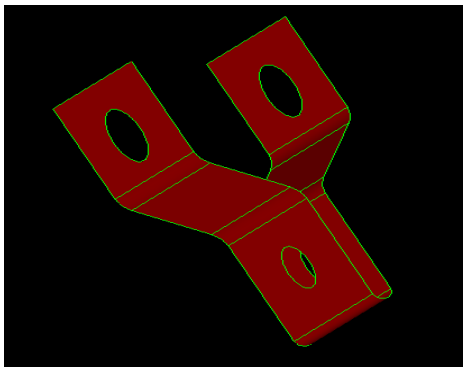
CAD-CAE


File
Options
Windows
Help
Search...
View Collection: 1
Status: Triad:IosFitView
File
Options
Windows
Help
Search...
View Collection: 1
Status: Triad:IosFitView

Midsurface is?



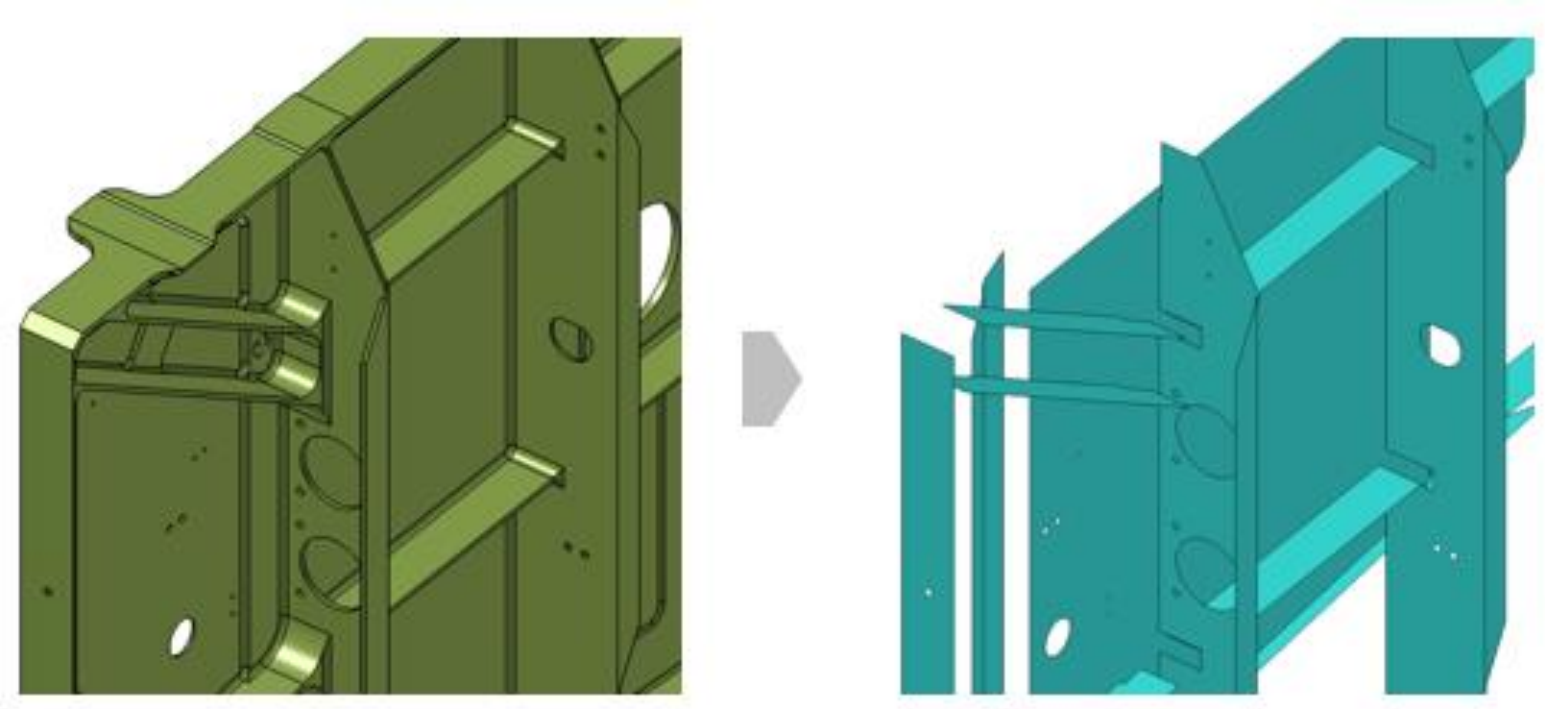
Input: Solid



Output: Midsurface

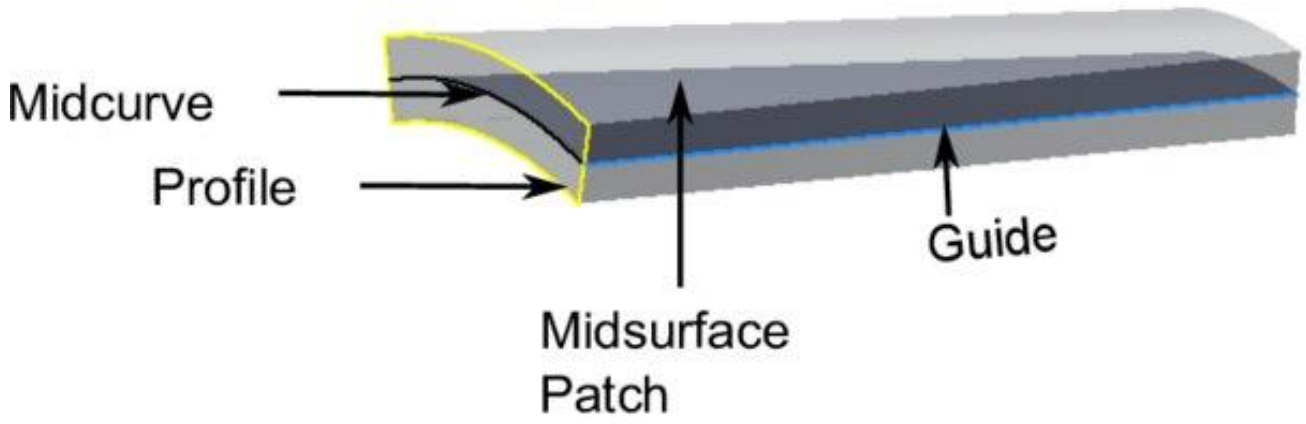
- Widely used for CAE of Thin-Walled parts
- Computation is challenging and still unsolved

Look at the output



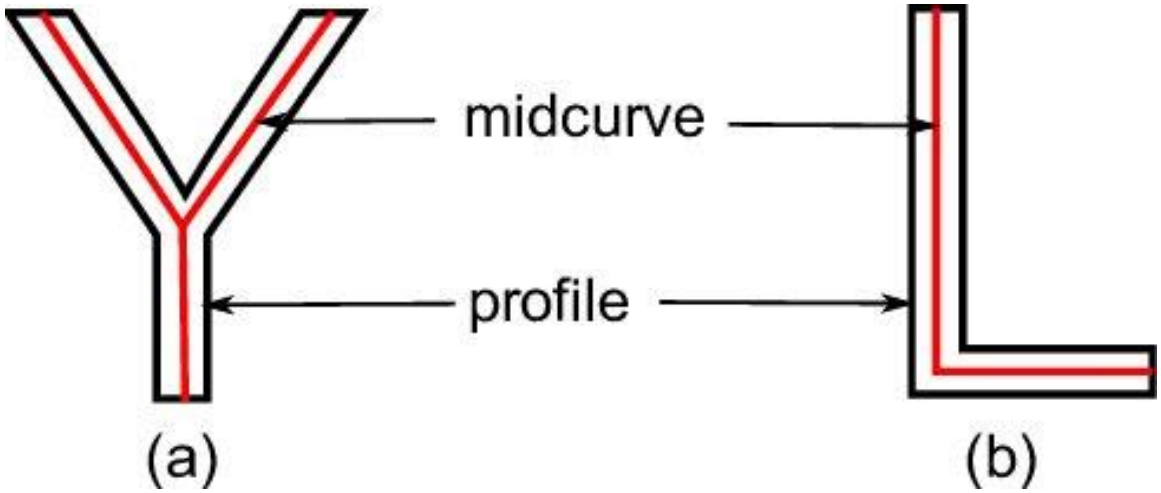
Midsurface Computation

- Midsurface of a Patch is Midcurve of its profile extruded.
- So, it boils down to computing 1D midcurve of a 2D profile



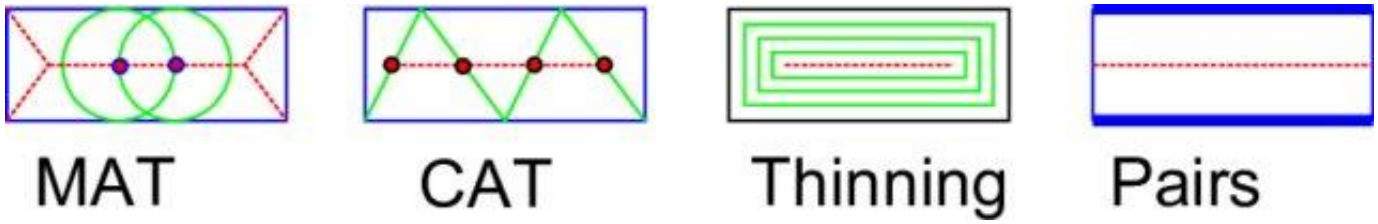
What is a Midcurve?

- **Midsurface : From 3D thin Solid to 2D Surface**
- **Midcurve : From 2D Profile to 1D Curve**

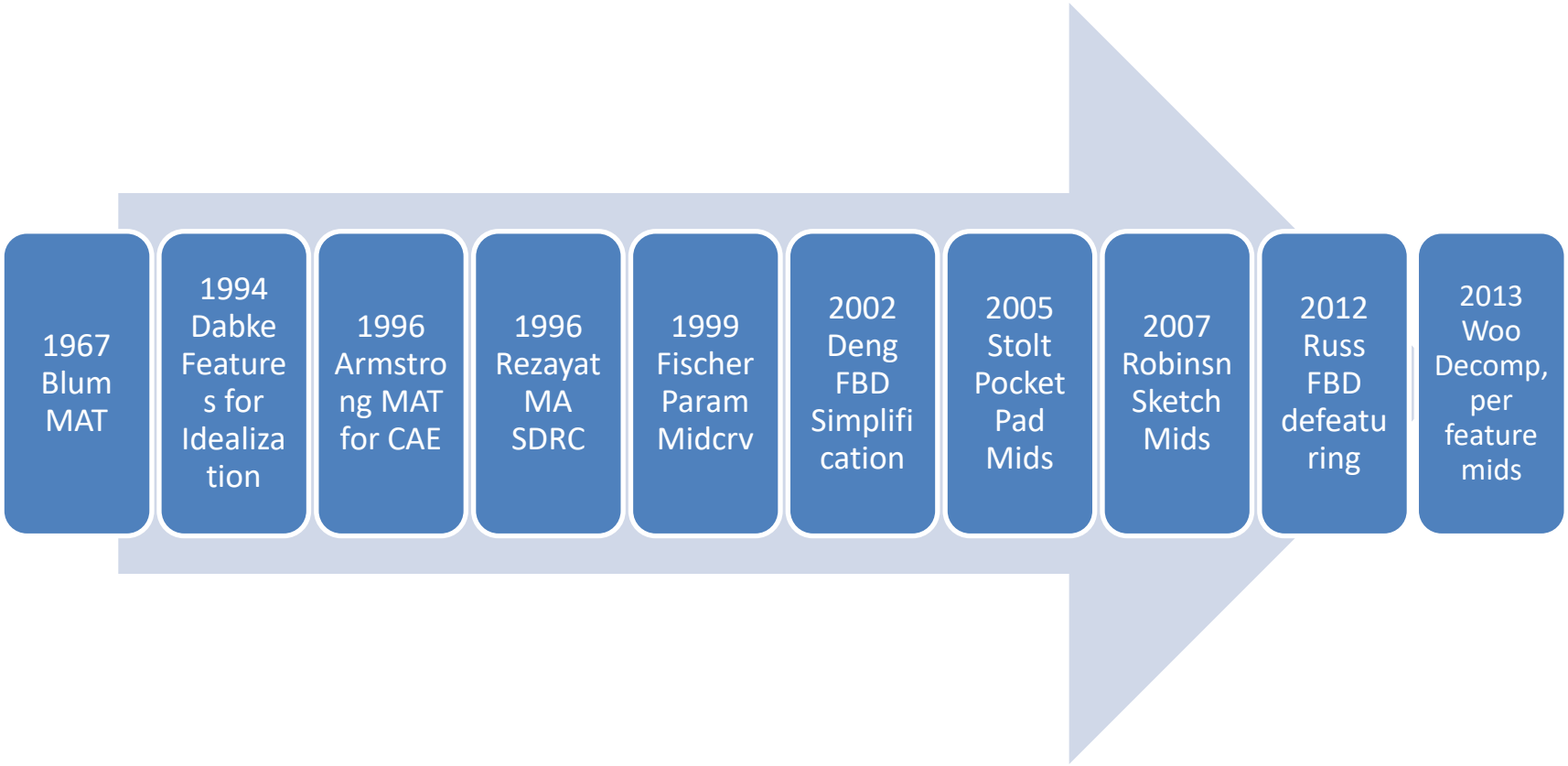


Many Approaches

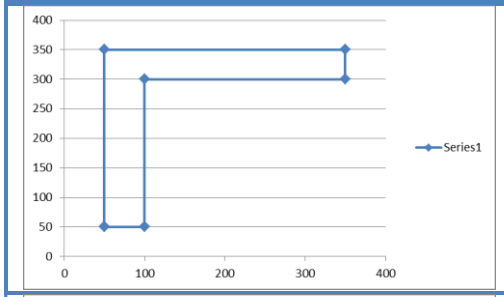
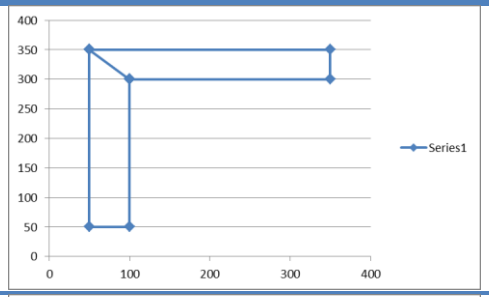
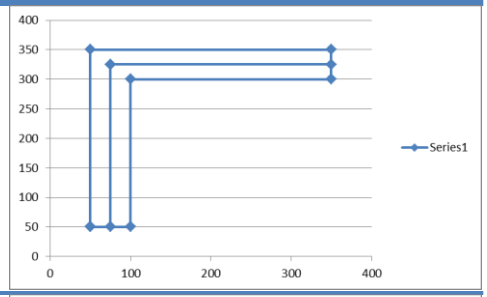
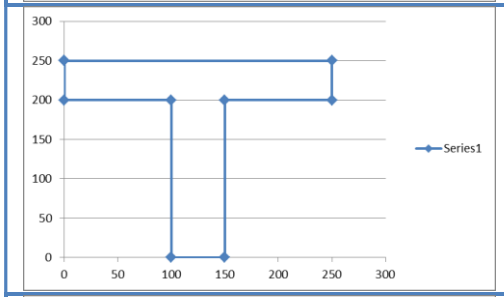
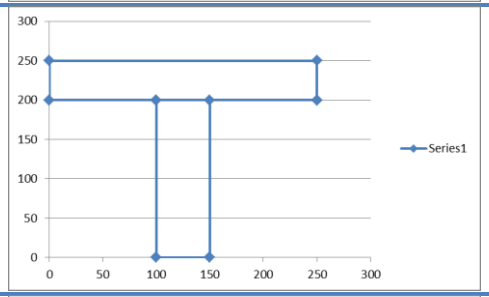
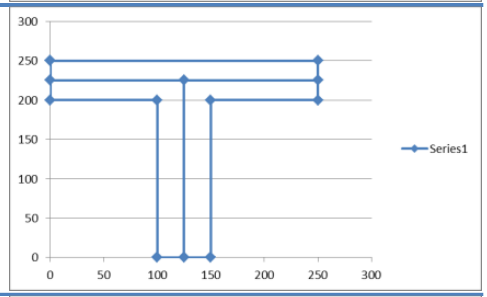
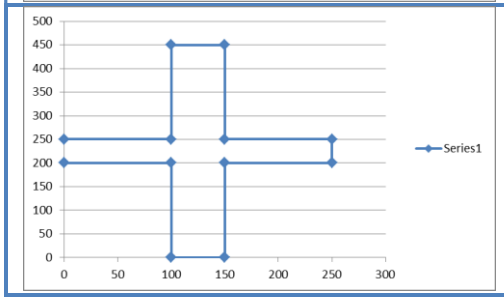
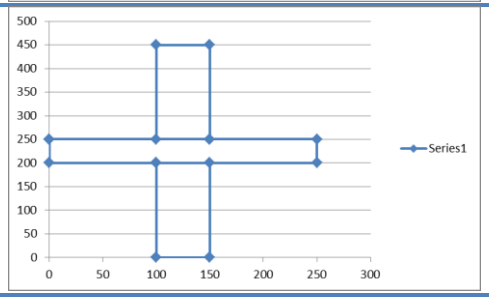
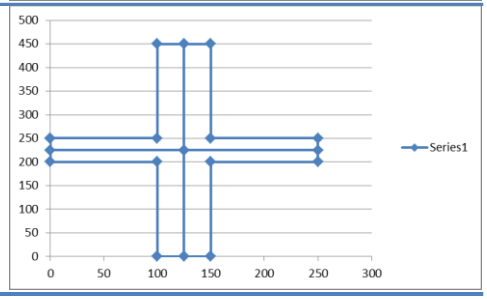
- More than 6 decades of research…
- Most CAD-CAE packages…
- Rule-based!! Heuristic!! Case-by-case basis!!



When-What


1967 Blum MAT
1994 Dabke Features for Idealization
1996 Armstrong MAT for CAE
1996 Rezayat MA SDRC
1999 Fischer Param Midcrv
2002 Deng FBD Simplification
2005 Stolt Pocket Pad Mids
2007 Robinsn Sketch Mids
2012 Russ FBD defeaturing
2013 Woo Decomp, per feature mids

2017: My PhD Work: Rule-based

Shape	My Impl Partitioning	My Impl Midcurves
		
		
		

Limitations

- Fully rule-based
- Need to adjust for new shapes
- So, not scalable

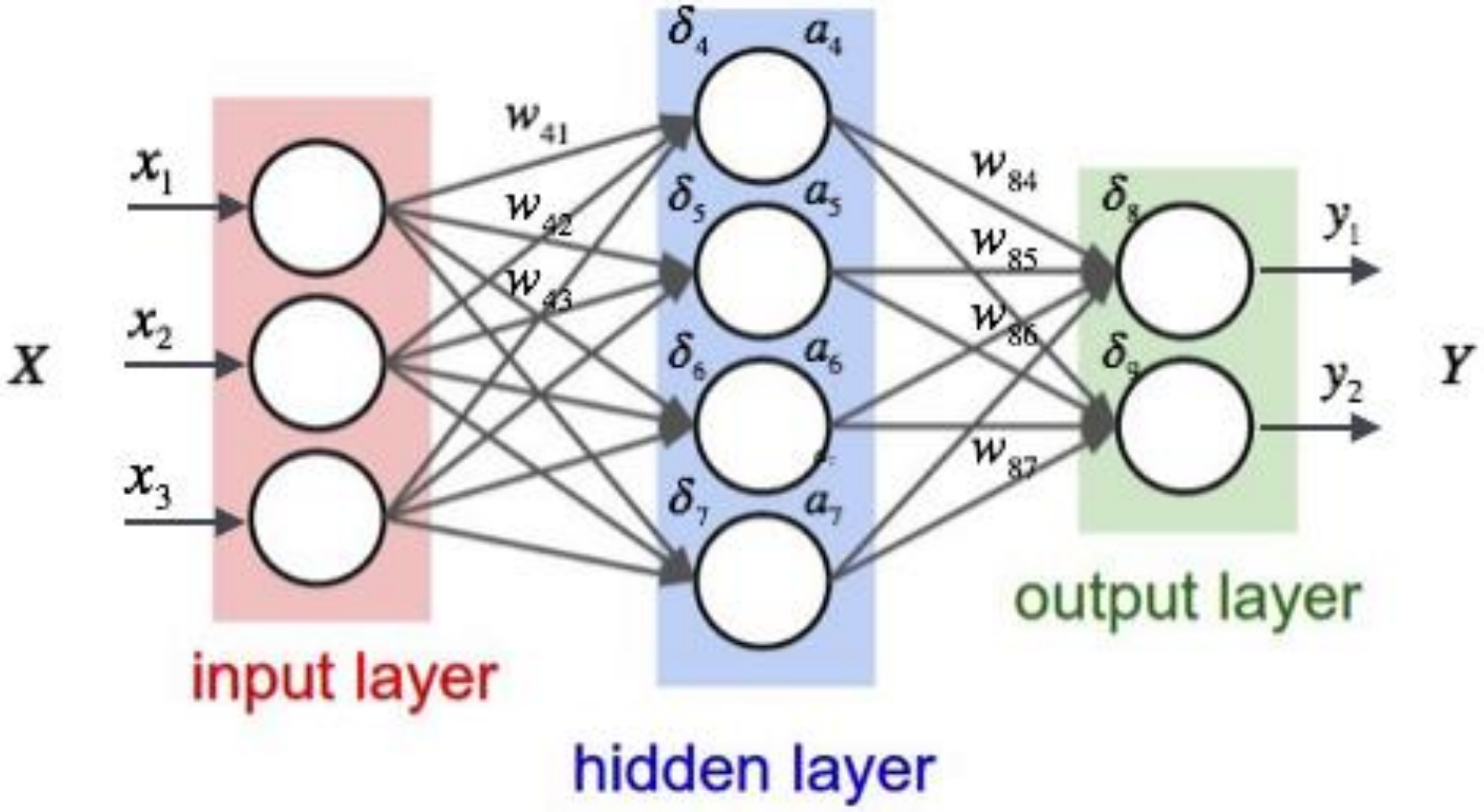
Idea

Can Neural Networks “learn” the dimension reduction transformation?

How?

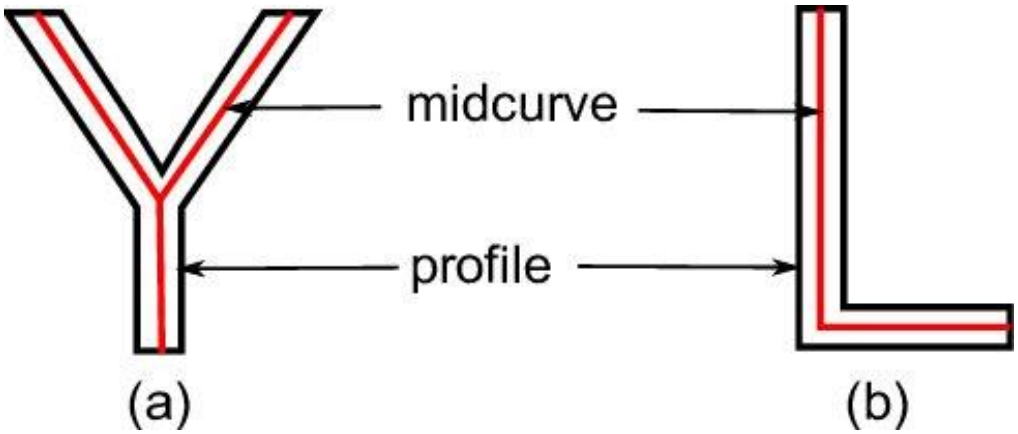
- Supply lots of training data of profiles and their corresponding midcurves and train.
- Then given an unseen profile, can Neural Network compute a midcurve, mimicking the original profile shape?

Midcurve by Neural network



Midcurve == Dimension Reduction

- Like PCA (Principal Component Analysis), wish to find Principal curve
- That ‘represents’ the original profile shape

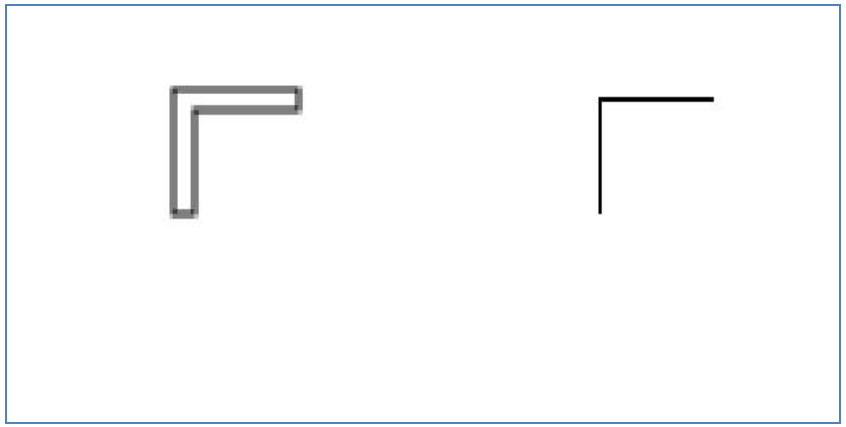


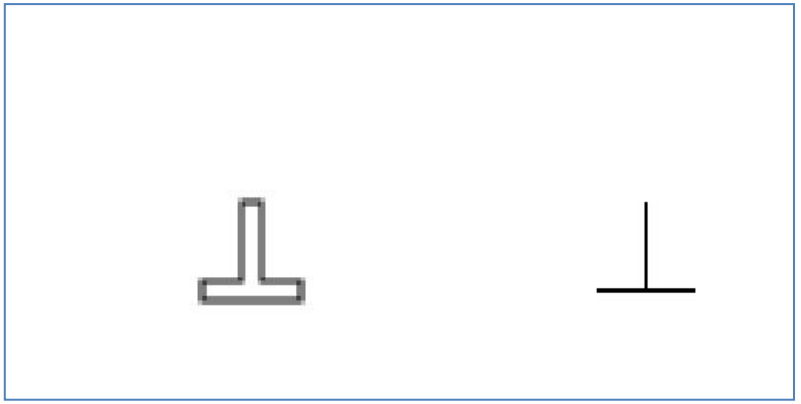
Variable Size Encoder Decoder

- OK for NLP, say Machine Translations, where padding values like “-1” can be added along with other words (vectors or indices)
- But in Geometry, its not OK.
- Because any value can represent a Valid Input, even though we don’t want it to be the input.

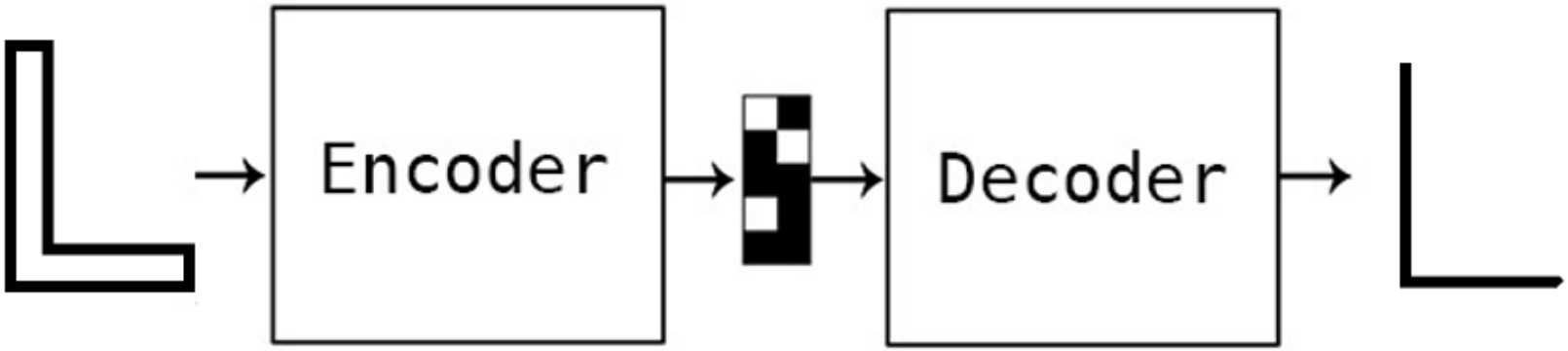
A Twist to the problem

- Input: Black & White Image of 2D profile
- Output: Black & White Image of 1D midcurve





For Dimension Reduction



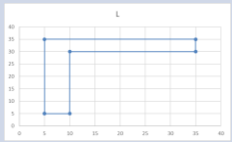
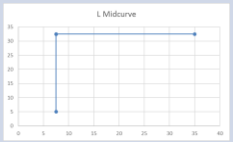
For Deep Learning

- Need lots of data
- Had just few input output image pairs
- How to augment/populate large variations…

DATA PREPARATION

Data

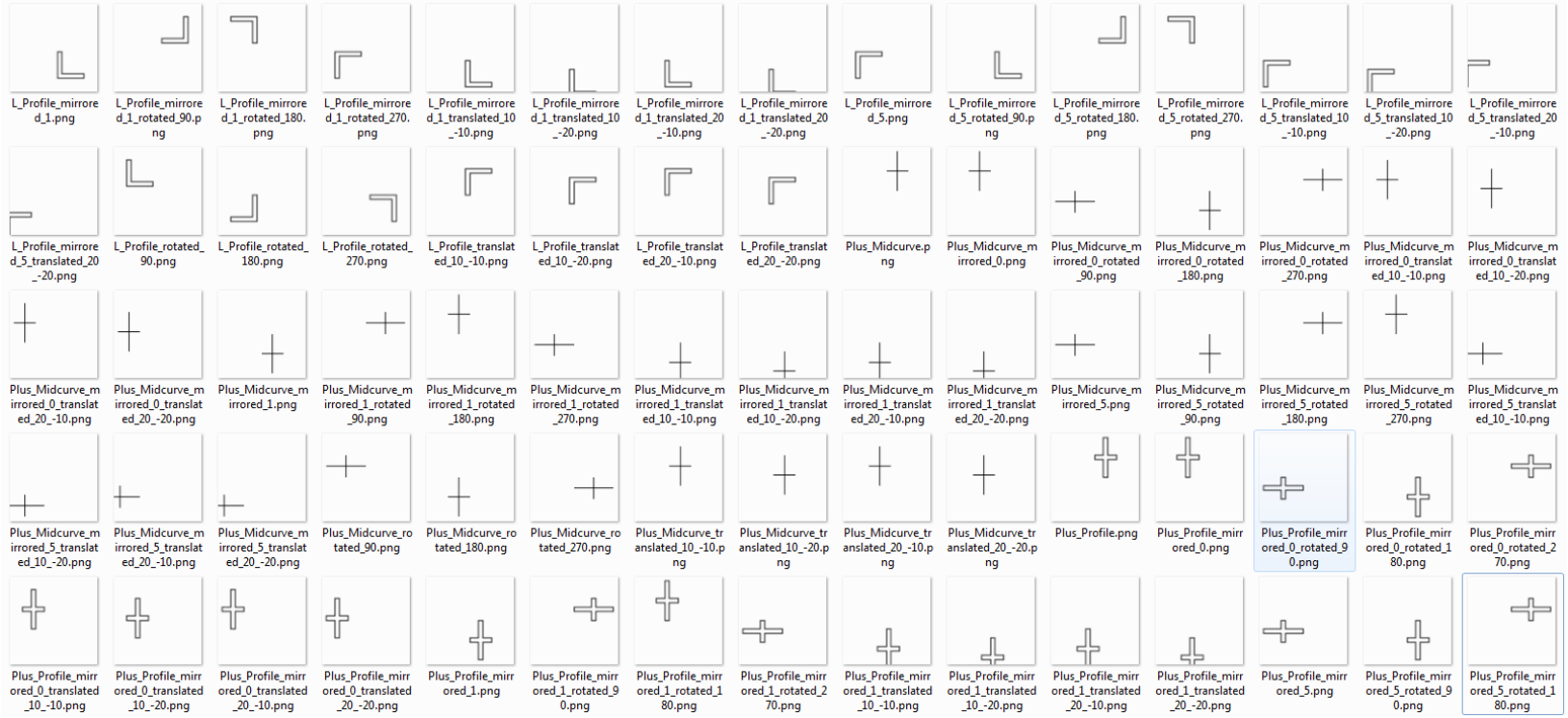
Original input and output are in the form of polylines, meaning a list of points, each having x,y coordinates

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				

Augmentation

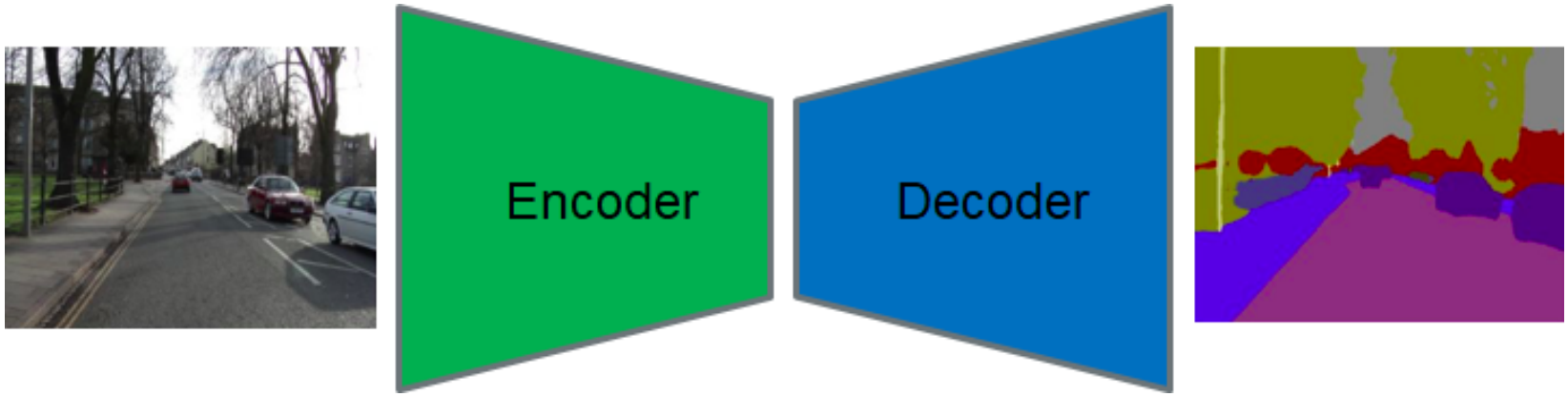
- Such few profile shapes, are just not enough for Neural Networks to train.
- Need more with as much diversity as possible.
- Will need to artificially augment data with transformations, like pan, rotate, mirror, etc.
- All needs to be automatically, programmatically

Training Data Samples



MIDCURVE BY NEURAL NETWORK

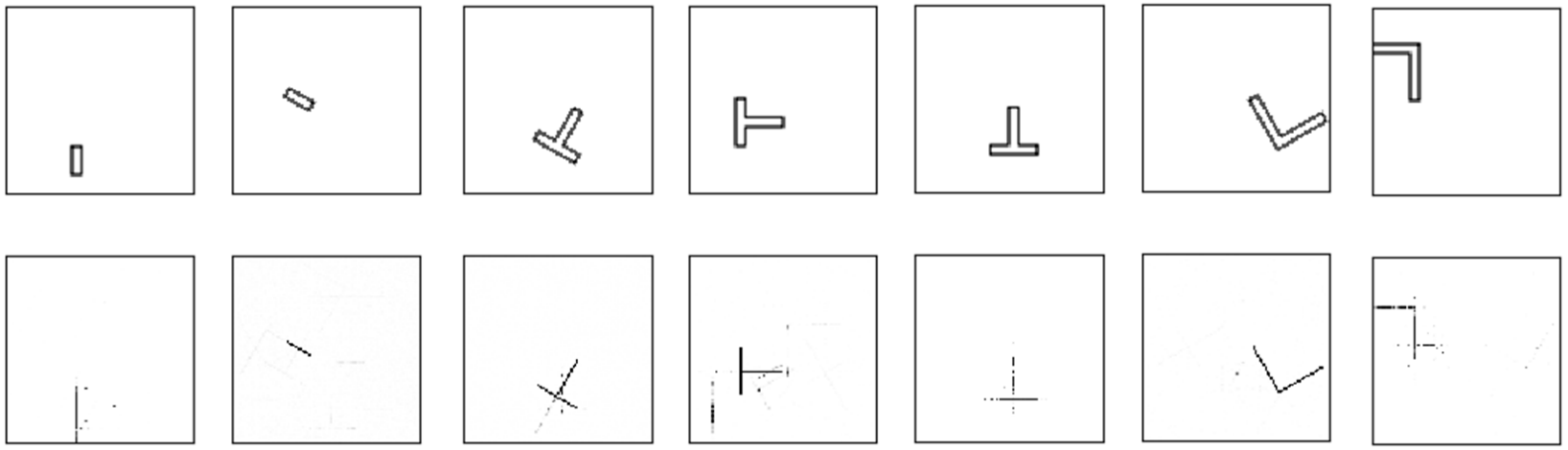
Simple Encoder Decoder



Keras Implementation

```
input_img = Input(shape=(input_dim,))

encoded = Dense(encoding_dim,
activation='relu',activity_regularizer=regularizers.l1(10e-5))(input_img)
decoded = Dense(input_dim, activation='sigmoid')(encoded)

autoencoder = Model(input_img, decoded)

encoder = Model(input_img, encoded)
encoded_input = Input(shape=(encoding_dim,))
decoder_layer = autoencoder.layers[-1]
decoder = Model(encoded_input, decoder_layer(encoded_input))

autoencoder.compile(optimizer='adadelta', loss='binary_crossentropy')
```

Results



Results

- Not very perfect but encouraging
- NN is correct with
  - The location (bounding box)
  - Dimension Reduction is seen
- But, still some stray points and misses

What can be done?

- For the noise, use bounding boxes
- Feedback into error term: differencing with the known output expected
- Classify single pixel image as the skeleton, and rest as noise.

What Next?

- Add denoiser network after the current one
- More Network Architectures
- Sequence-to-Sequence based approaches, taking closed thin polygon as input and polyline as output
- Extending to 3D, ie Midsurface

Thank you

yogeshkulkarni@yahoo.com