RAGDNet

A Region-Adjacency Graph for Semantic Segmentation of Mechanical Drawings Using Graph Neural Networks

Alexandre Monnier Weil[1,2] Nicolas Hili[2] Yves Ledru[2]

[1]Kaizen Solutions

[2]Univ. Grenoble Alpes

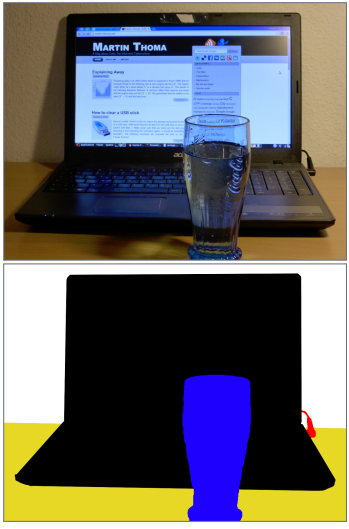
ICPR 2026 — Lyon

alexandre.monnier@univ-grenoble-alpes.fr

What is semantic segmentation?

- Assign a class label to every pixel of an image.
- Standard recipe: an encoder–decoder over the pixel grid (*U-Net*) or a transformer over patches (*SegFormer*).

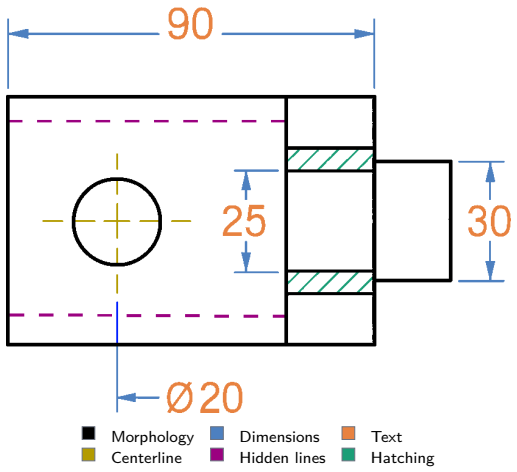
The question is always the same: *what* is it at this location?



M. Thoma, Wikimedia Commons, CC0

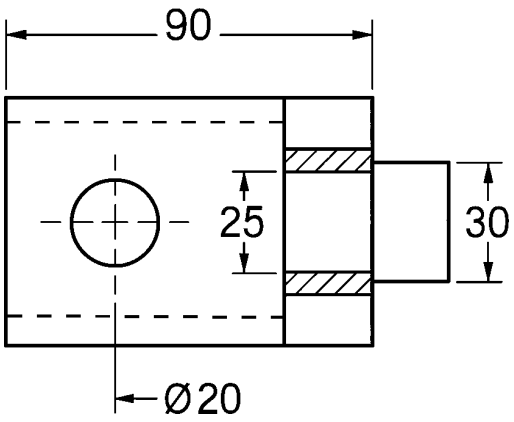
Why segment mechanical drawings?

- Standard medium of technical communication across many industrial domains.
- Semantics are lost in exchange formats: raster is a matrix of pixels (PNG, JPG), vector, a set of geometric primitives with no meaning attached (DXF, DWG).
- Downstream automation:
  - retrieving similar drawings in large databases;
  - automated cost estimation;
  - comparison of information between two versions.



Limitations of current approaches

- In our setting, drawings are binary and extremely sparse: in the drawing on the right, ink covers 7.3% of the pixels.
- Yet in a CNN as in a ViT, every pixel becomes a variable in the computation.



829 × 665 px, 40 380 inked, 7.3%

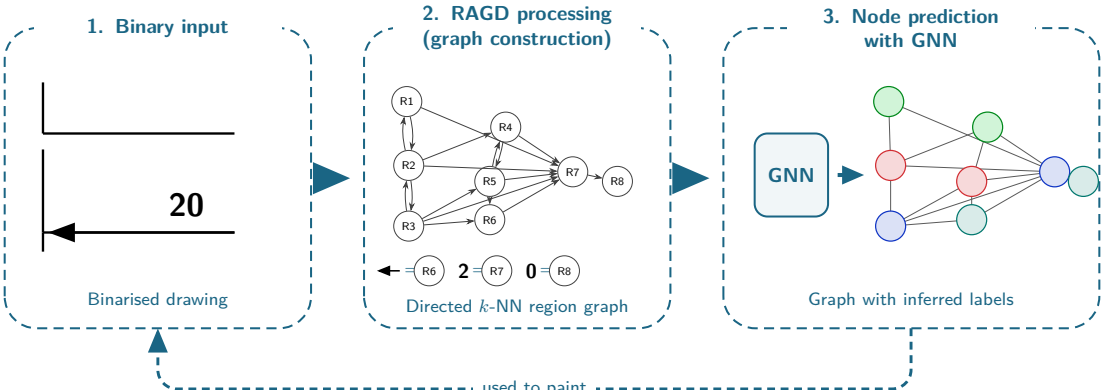
Contributions

- RAGDNet splits into two parts:
  - RAGD, the algorithm that turns a binarised drawing into a region-adjacency graph: regions become nodes, adjacency becomes edges;
  - the network that predicts one class per node of that graph, using a graph neural network.
- A labeled dataset of 2D CAD mechanical drawings, with four foreground classes.

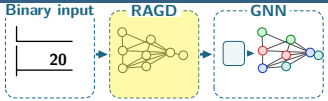
Outline

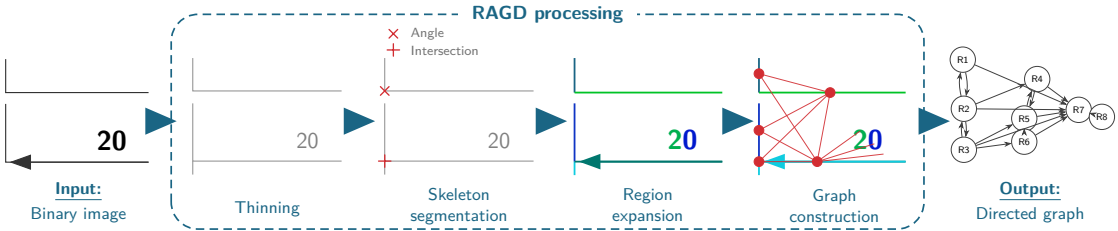
1. Method (RAGDNet)
2. Dataset
3. Experiments

Pipeline overview (key idea)



1 — From raster to region-adjacency graph

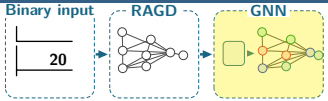




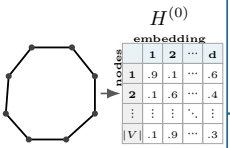
A directed k-NN region graph $G = (V, E)$, one node per stroke region.

- Parameters: corner angle range $[\tau_{\min}, \tau_{\max}]$ and k, the neighbours kept per node.
- Node features: the region polygon P_i, its simplified outer contour.
- Edge features: normalised distance $\hat{d}_{ij}$ and unit direction $\hat{u}_{ij}$ to each of the k nearest regions.

2 — Node prediction with GNN

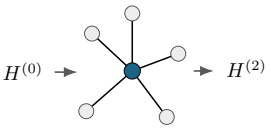


1. Polygon embedding



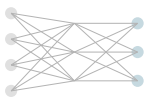
shared MLP φ
+ mean/max pooling

2. Message passing



2× GATv2
heads $= 2$
dims $= [256, 512]$
edge features $= 3$

3. Node prediction



MLP
+
Softmax

applied independently to each node

4. Output

classes

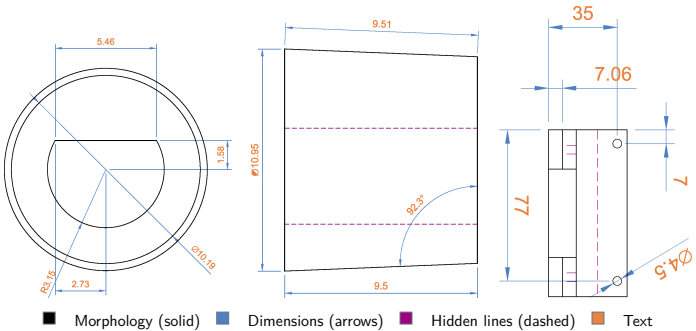
nodes	1	2	3	4
1	**0.59**	0.13	0.20	0.08
2	0.12	**0.60**	0.18	0.10
3	0.15	0.13	**0.53**	0.19
⋮	⋮	⋮	⋮	⋮
$\|V\|$	0.08	0.16	**0.61**	0.15

node-wise class scores

C classes

A CAD-derived annotated dataset

- Our dataset consists of 422 drawings.
- They come from a 3D dataset of mechanical parts: Autodesk Fusion flattens each part to 2D and generates the annotations.



Cross-validation results

Method	Morphology	Dimensions	Hidden lines	Text	mIoU
RAGDNet-GAT (ours)	0.90±0.02	0.87±0.01	**0.74**±0.04	**0.96**±0.01	**0.869**±0.019
RAGDNet-MLP	0.87±0.01	0.78±0.03	0.51±0.04	0.88±0.01	0.761±0.007
U-Net	0.94±0.02	0.88±0.03	0.56±0.04	0.93±0.04	0.826±0.030
SegFormer-B0	0.92±0.02	0.83±0.03	0.32±0.15	0.92±0.04	0.747±0.060
SegFormer-B2	**0.95**±0.02	**0.88**±0.07	0.56±0.19	**0.96**±0.03	0.838±0.075

- 5-fold experiments: 80% training, 20% validation.
- Best mIoU: 0.869 vs 0.838.
- Where each wins: a thin stroke gives a pixel model almost nothing to see, but it is still one node on the graph; on the thick classes the pixel models keep the edge, 0.95 vs 0.90.

Model size & inference speed

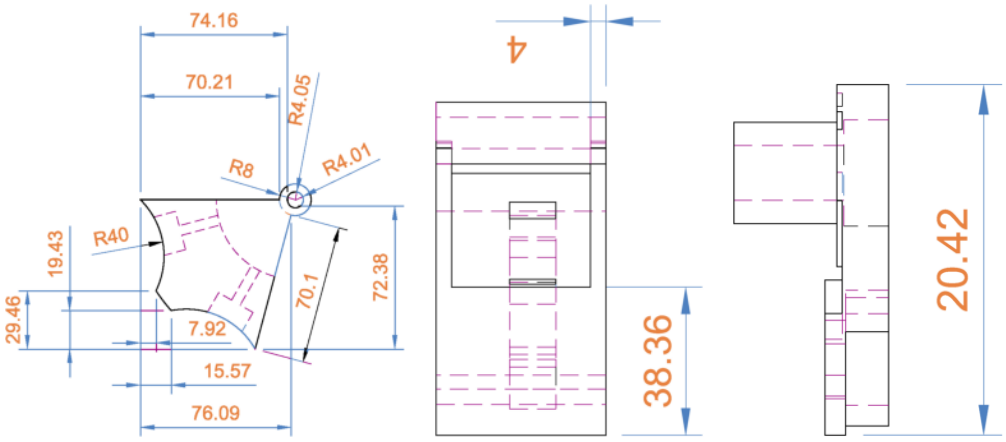
Table: Model size and CPU timing on an Intel Core i7-10610U CPU over 422 drawings.

Method	Params	Pre* (s)	Inf (s)	Total (s)
RAGDNet-GAT	**0.881M**	11.86±10.90	**0.03**±0.03	11.89±10.90
U-Net	31.0M	–	12.26±5.24	12.26±5.24
SegFormer-B0	3.75M	–	7.29±6.17	**7.29**±6.17
SegFormer-B2	27.3M	–	18.08±13.26	18.08±13.26

* A recent optimisation of the graph construction cuts this preprocessing time by a factor of ≈49, down to 0.26 s, a new total of ≈0.29 s.

- Inference with RAGDNet is much faster
- The model is also lightweight

Qualitative results



Conclusion

- We exploit the sparse, structured nature of mechanical drawings: RAGDNet shifts the prediction from the pixel grid to a Region Adjacency Graph. A dataset comes with it.
- This topological representation achieves pixel-based accuracy with a far more compact architecture and faster inference.
- Three limitations:
  - the dataset covers a limited set of classes and drawing conventions, short of industrial diversity;
  - the multi-stage image-to-graph construction lets early errors propagate into the final region graph;
  - instance segmentation is not supported yet.

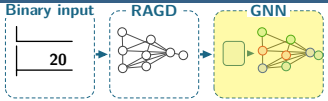
Future work

- Broader coverage: an industrial partner dataset, re-annotated to our label definitions, extending the taxonomy beyond the four foreground classes.
- Learnable graph extraction instead of fixed preprocessing heuristics: a differentiable construction, trainable end-to-end from raw pixels.
- Panoptic segmentation, to distinguish individual objects within each class.
- Longer term: legacy scanned drawings, with image restoration as a pre-processing step, ideally without extensive retraining.

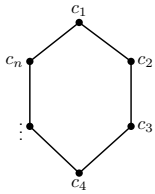
Thank you

Dataset & code: github.com/Alewep/RAGDNet · alexandre.monnier@univ-grenoble-alpes.fr

Polygon encoder

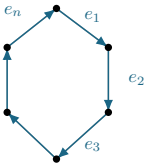


1 Polygon



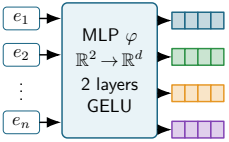
$p_i = (c_1, \ldots, c_n)$

2 Contour vectors



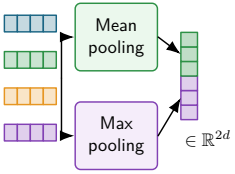
$e_j = c_{j+1} - c_j$

3 Shared MLP



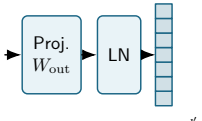
n descriptors in $\mathbb{R}^d$

4 Aggregation



no longer depends on n

5 Output

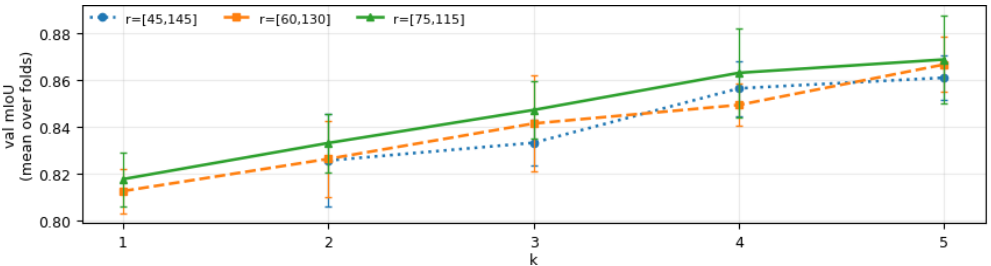


fixed size d'

- Contour vectors $e_j = c_{j+1} - c_j$ pass through a shared MLP.
- Pooling yields one feature vector per region.

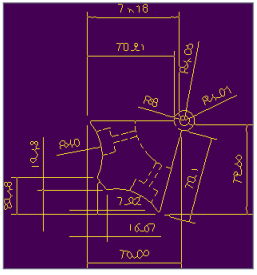
RAGD parameters

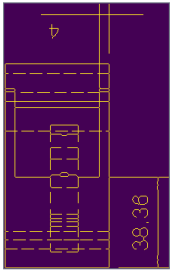
Parameter	Tested	Retained
$[\tau_{\min}, \tau_{\max}]$, corner angle range	$(75, 115)$, $(60, 130)$, $(45, 145)$	$[75°, 115°]$
k, directed k-NN size, $\mathrm{indegree}(v) = k$	$1, \ldots, 5$	5

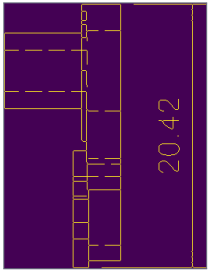


- Mean validation mIoU across folds, one curve per angle range.
- Increasing k helps, with diminishing returns.
- $[75°, 115°]$ is best for small k; the ranges converge as k grows.

Thinning







- We use Lee's thinning algorithm (Lee, Kashyap & Chu, 1994).
- It erodes the foreground iteratively, deleting a point only when it is *simple*: its removal changes neither connectivity nor topology.
- The result is a one-pixel-wide, 8-connected skeleton centred in the stroke. The same three drawings as in the qualitative results.

Junction detection

Convolve the skeleton with the 8-neighbour kernel: $D = \mathrm{Skel} * K$ counts the foreground neighbours of each pixel.

$$K = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

$$\begin{matrix} 0 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \end{matrix}$$
Endpoint, $D = 1$

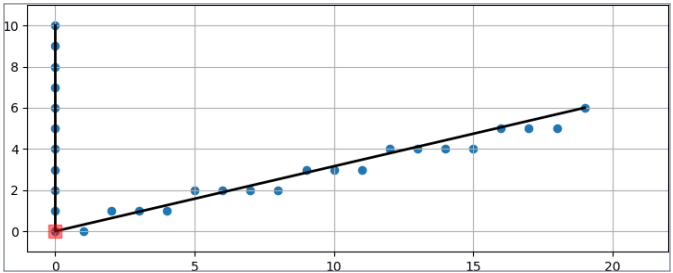
$$\begin{matrix} 0 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 0 \end{matrix}$$
Internal, $D = 2$

$$\begin{matrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \end{matrix}$$
T-junction, $D = 3$

$$\begin{matrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{matrix}$$
Cross, $D = 4$

- On a skeleton pixel, D is the degree of the branch point.
- $D = 1$ endpoint, $D = 2$ internal pixel, $D \geq 3$ junction.
- Cutting at every $D \geq 3$ splits the skeleton into independent branches.

Branch simplification & corners



- A branch is an ordered run of 8-connected skeleton pixels between two junctions, or a junction and an endpoint.
- Pixel quantisation makes raw branches zigzag, so each one is simplified by Douglas–Peucker (1973) with tolerance $\varepsilon \approx 2.12$ px.
- An interior point is a corner when its turning angle θ_i falls in $[\tau_{\min}, \tau_{\max}]$. The branch is split there.

Where the tolerance ε comes from

ε is not tuned: it is the largest displacement the pixel grid alone can produce, so Douglas–Peucker absorbs quantisation noise and nothing else.

- One axis: mapping a continuous point to the nearest pixel centre costs at most half a pixel, in x as in y.
- One point: the worst case is a diagonal displacement,

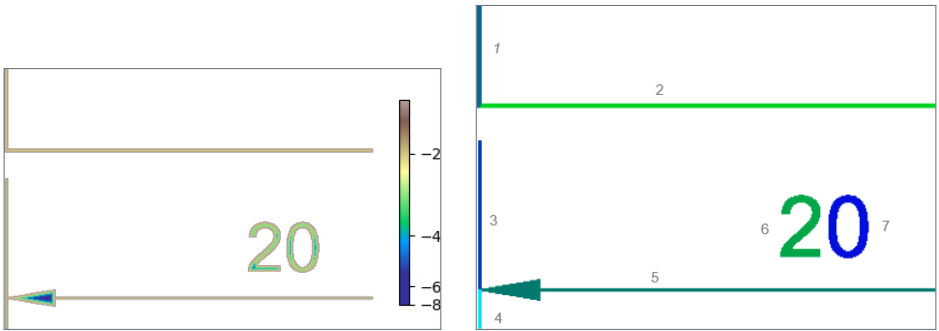
$$\eta = \sqrt{\left(\tfrac{1}{2}\right)^2 + \left(\tfrac{1}{2}\right)^2} = \frac{\sqrt{2}}{2} \approx 0.71 \text{ px.}$$

- Three points: the criterion measures the distance from a branch point to the segment carried by two others. All three are quantised, each off by up to η, so in the worst case the errors add up:

$$\varepsilon = 3\,\eta = \frac{3\sqrt{2}}{2} \approx 2.12 \text{ px.}$$

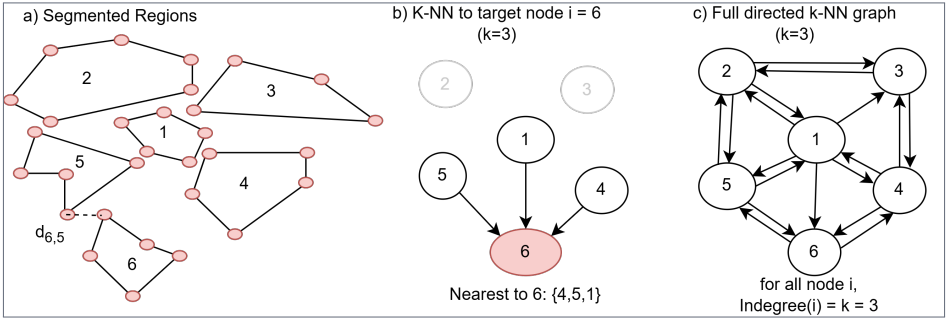
A conservative bound: it kills grid oscillation without touching a real direction change.

Watershed



- Relief: the negated distance transform, deepest at the centre of a stroke (left, colour bar $0 \to -8$).
- The skeleton branches are the seeds; basins flood outward until they meet.
- One region per branch (right), and those regions become the nodes of the graph.

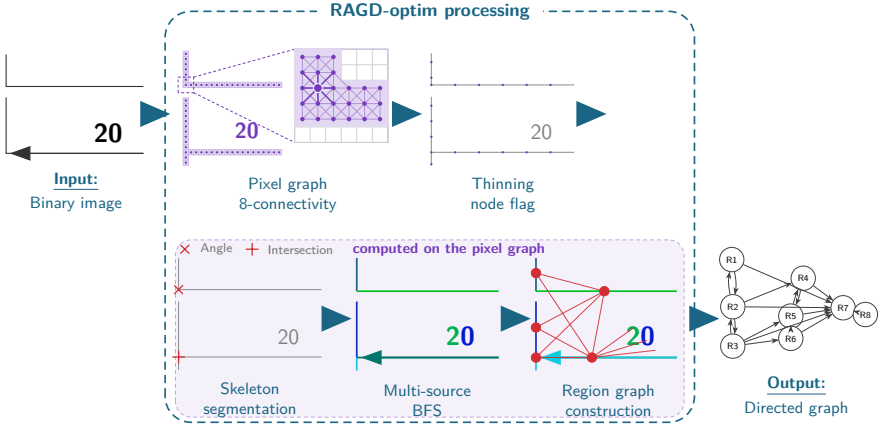
Graph construction



- One node per region; its feature is the simplified outer polygon (a).
- For each node i, the k nearest regions each send an edge *into* i (b), so $\mathrm{indegree}(i) = k$ for every node (c).
- Edge features: normalised distance $\hat{d}_{ij}$ and unit direction $\hat{u}_{ij}$.

RAGD-optim pipeline

Key idea: compute the graph earlier, to avoid processing the blank pixels that carry no information.



Graph construction cost

Table: Comparison of the mean execution time and standard deviation between RAGD and RAGD Optim.

Step	**RAGD (s)**	**RAGD Optim (s)**
Del Peak	3.534 ± 3.742	0.057 ± 0.025
Watershed	6.254 ± 6.033	0.013 ± 0.007
Region Graph	2.945 ± 2.850	0.086 ± 0.072
Skeletonize	0.068 ± 0.050	0.065 ± 0.049
Del Junction	0.027 ± 0.012	0.003 ± 0.001
Pixel Graph	n/a	0.039 ± 0.015
Total	$\mathbf{12.827 \pm 11.606}$	$\mathbf{0.264 \pm 0.117}$