Supplemental Material for
Inferring CAD Modeling Sequences Using Zone Graphs

1. Simplification by face loop

Please see Figure 1 for a demonstration of finding *face loops* in the target geometry for zone graph simplification.

2. Search

Please see Algorithm 1 and Algorithm 2 and for extrusion formulation and search details.

Algorithm 1 Extrusion Formulation

Input
  Input zone graph zg
Output
  Proposed extrusion list $exts$
procedure getExtrusions(zg)
  $exts \leftarrow []$
  for (sp, ep) in $plane\_pairs$ **do** ▷ iterating parallel plane pairs
    $v \leftarrow sp.nor * dis_{sp \rightarrow ep}$ ▷ compute extrusion vector
    $cgs \leftarrow getGroups(sp.faces, sp.cur\_graph)$
    $tgs \leftarrow getGroups(sp.faces, sp.tgt\_graph)$
    $igs \leftarrow getGroups(sp.faces, sp.idle\_graph)$
    $cyds \leftarrow genCylinders(cgs, tgs, igs, v)$ ▷ grouping faces into sketches and generating extrusions
    for cyd in $cyds$ **do**
      $e = Extrusion()$
      $e.zones \leftarrow findZones(cyd, zg)$ ▷ finding and labeling inside zones
      $e.bool \leftarrow boolType(e, zg)$
      $exts.add(e)$
    end for
  end for
  return $exts$
end procedure

3. Network Architecture

Please see Table 1, 2, 3 and 4 for network architecture details.

Algorithm 2 Search

Input
  Input zone graph zg
Output
  Reconstruction sequence seq
procedure search($zg, seq = []$)
  if $is\_target(zg)$ **then**
    return $True$
  end if
  if $terminate()$ **then**
    return $False$
  end if
  $exts = getExtrusions(zg)$
  $exts\_ranked = rankExtrusions(exts)$
  for e in $exts\_ranked$ **do**
    $zg = zg.update(e)$
    $ret \leftarrow search(zg, seq)$ ▷ recursive search
    if $ret = True$ **then**
      $seq.add(e)$
      return $True$
    end if
  end for
end procedure

Please see Figures 6, 7 for some ablation experiments with different model architectures.

4. Fusion 360 Gallery Success/Fail Case Summary

Tables 5, 6, 7 detail the percentage of Fusion 360 Gallery shapes that our method can/cannot reconstruct, breaking down successes and failures into subcategories. Overall, our method can reconstruct 80% of the shapes in the dataset.

We also show the effect of different strategies for constructing extrusion proposals. As described in Section 5.1 of the main paper, we consider only individual connected components or the union of all connected components in a face group as candidate extrusions. Here, we introduce a generalization of this scheme based on the idea of proposal

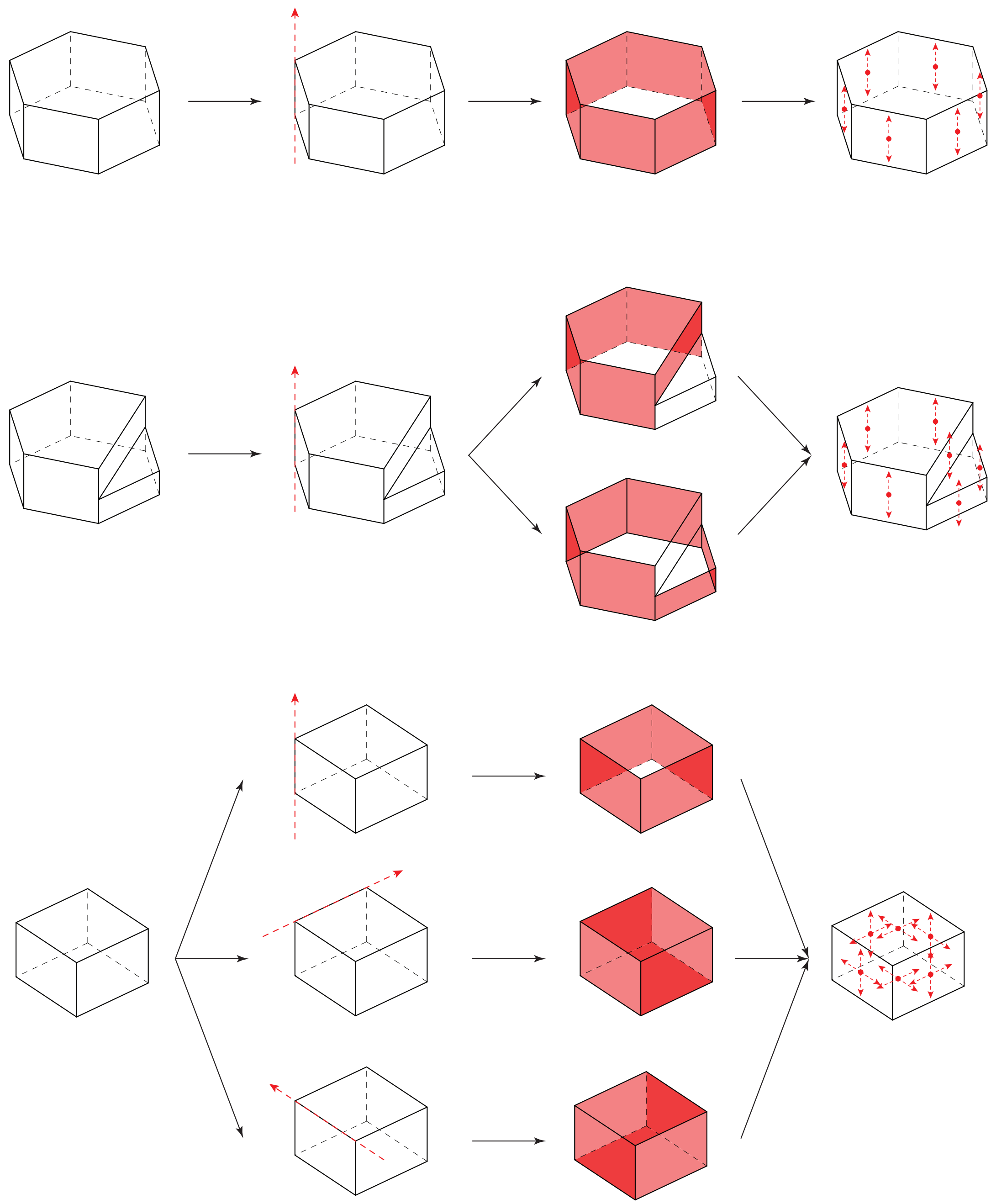


Figure 1: Zone graph simplification by finding *face loops*. Column 1: Target shapes; Column 2: Extrusion directions marked with red arrows; Column 3: Found *face loops*, highlighted in red; Column 4: Face extension directions determined by the extrusion directions associated with their *face loops*. Note that when multiple extrusion directions are found for a face, it will be extended in all found directions.

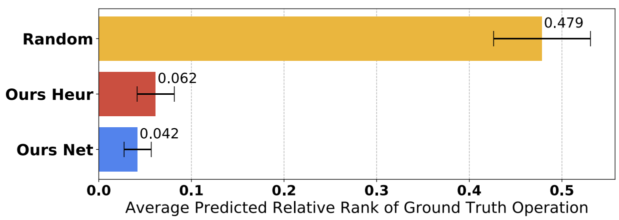


Figure 2: Comparing how different methods rank the ground truth extrusions used in modeling sequences from the Fusion 360 Gallery dataset without zone graph simplification.

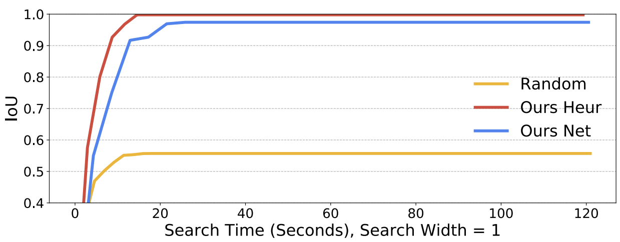


Figure 3: Reconstruction accuracy of the outputs of inferred programs vs. the time used to infer them. Search bandwidth = 1.

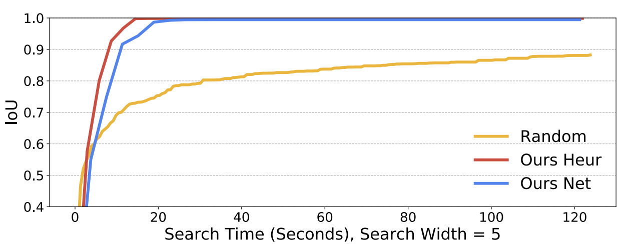


Figure 4: Reconstruction accuracy of the outputs of inferred programs vs. the time used to infer them. Search bandwidth = 5.

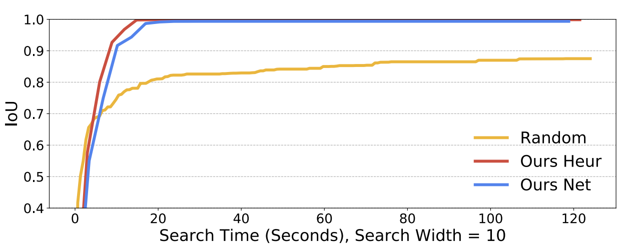


Figure 5: Reconstruction accuracy of the outputs of inferred programs vs. the time used to infer them. Search bandwidth = 10.

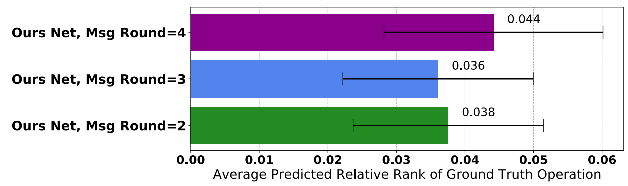


Figure 6: Comparing how different message passing round in GCN affects the average relative extrusion ranking.

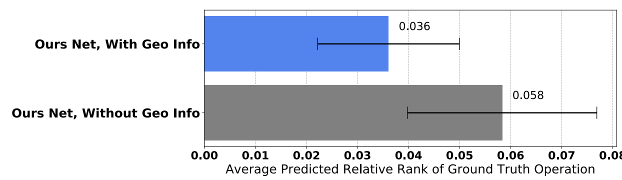


Figure 7: Comparing how geometry information (with and without point cloud) in the GCN affects the average relative extrusion ranking.

PointNet
Conv1d $(10, 64, 1)$
Batchnorm1d
LeakyRelu
Conv1d $(64, 128, 1)$
Batchnorm1d
LeakyRelu
Conv1d $(128, 128, 1)$
MaxPool
FC(128×128)
Batchnorm1d
LeakyRelu
FC(128×128)

Table 1: Detailed architecture of the PointNet we used in the project

Msg
FC(128×128)
Batchnorm1d
LeakyRelu
FC(128×128)
Batchnorm1d
LeakyRelu
FC(128×128)

Table 2: Detailed architecture of the Message Passing Layers we used in the project

levels. For a proposal level of k, within each face group, all subsets with size 1, 2 ... k and subsets with size N-k, N-k+1 ... N will be used as candidate extrusions (the scheme presented in Section 5.1 of the main paper uses $k = 1$). Larger k leads to more potential extrusions, which increases the percentage of the ground-truth modeling sequences from

GCN
Msg
Msg
Msg
MaxPool

Table 3: Detailed architecture of the Graph Convolutional Network we used in the project

MLP
FC(128×128)
Batchnorm1d
LeakyRelu
FC(128×128)
Batchnorm1d
LeakyRelu
FC(128×2)
Softmax

Table 4: Detailed architecture of the Multi-Layer Perceptron Network we used in the project

% of data	Description
80%	**Can reconstruct target**
60%	with GT sequence
8%	with different sequence (GT not captured by our proposals)
12%	with different sequence (GT overshadowed & unrecoverable)
20%	**Cannot reconstruct target**
16%	Insufficiently complete zone graph / GFA error
2%	Unsupported operations (e.g. tapered extrude, revolve)
2%	Crash/Hang/Timeout

Table 5: Percentage of Fusion360 shapes our method(With Zone Graph simplification, Proposal level = 1) can/cannot reconstruct (and why)

the dataset which can be covered, at the cost of a larger search space (and therefore more computation time). As show in Tables 5 and 6, by increasing extrusion proposal level k from 1 to 3, more ground truth extrusion sequences (from 60% to 63%) are captured by our proposals.

5. Qualitative Results

5.1. Ours vs. random and heuristics

Please see Figure 9 and 10 for additional examples of the qualitative comparison of the output of our model's inferred programs (Network) vs. those of Random and Heuristics.

5.2. Ours vs. InverseCSG

Please see Figure 11, 12 and 13 for examples of the qualitative comparison of the output of our model's inferred programs vs. those of InverseCSG. See Figure 14 for examples of the reconstruction results of our model's reconstruction vs. those of InverseCSG.

Figure 8, top compares the reconstruction error of our inferred programs to those of InverseCSG and Figure 8, bottom compares the search time of each method. Reconstruction error is calculated using 1-IoU. The search time for InverseCSG is taken from their paper and the InverseCSG error rate is calculated using the reconstructed models released by the InverseCSG authors.

% of data	Description
80%	**Can reconstruct target**
63%	with GT sequence
5%	with different sequence (GT not captured by our proposals)
12%	with different sequence (GT overshadowed & unrecoverable)
20%	**Cannot reconstruct target**
16%	Insufficiently complete zone graph / GFA error
2%	Unsupported operations (e.g. tapered extrude, revolve)
2%	Crash/Hang/Timeout

Table 6: Percentage of Fusion 360 Gallery shapes our method (with zone graph simplification, proposal level = 3) can/cannot reconstruct (and why)

% of data	Description
82%	**Can reconstruct target**
62%	with GT sequence
8%	with different sequence (GT not captured by our proposals)
12%	with different sequence (GT overshadowed & unrecoverable)
18%	**Cannot reconstruct target**
14%	Insufficiently complete zone graph / GFA error
2%	Unsupported operations (e.g. tapered extrude, revolve)
2%	Crash/Hang/Timeout

Table 7: Percentage of Fusion 360 Gallery shapes our method (without zone graph simplification, proposal level = 1) can/cannot reconstruct (and why)

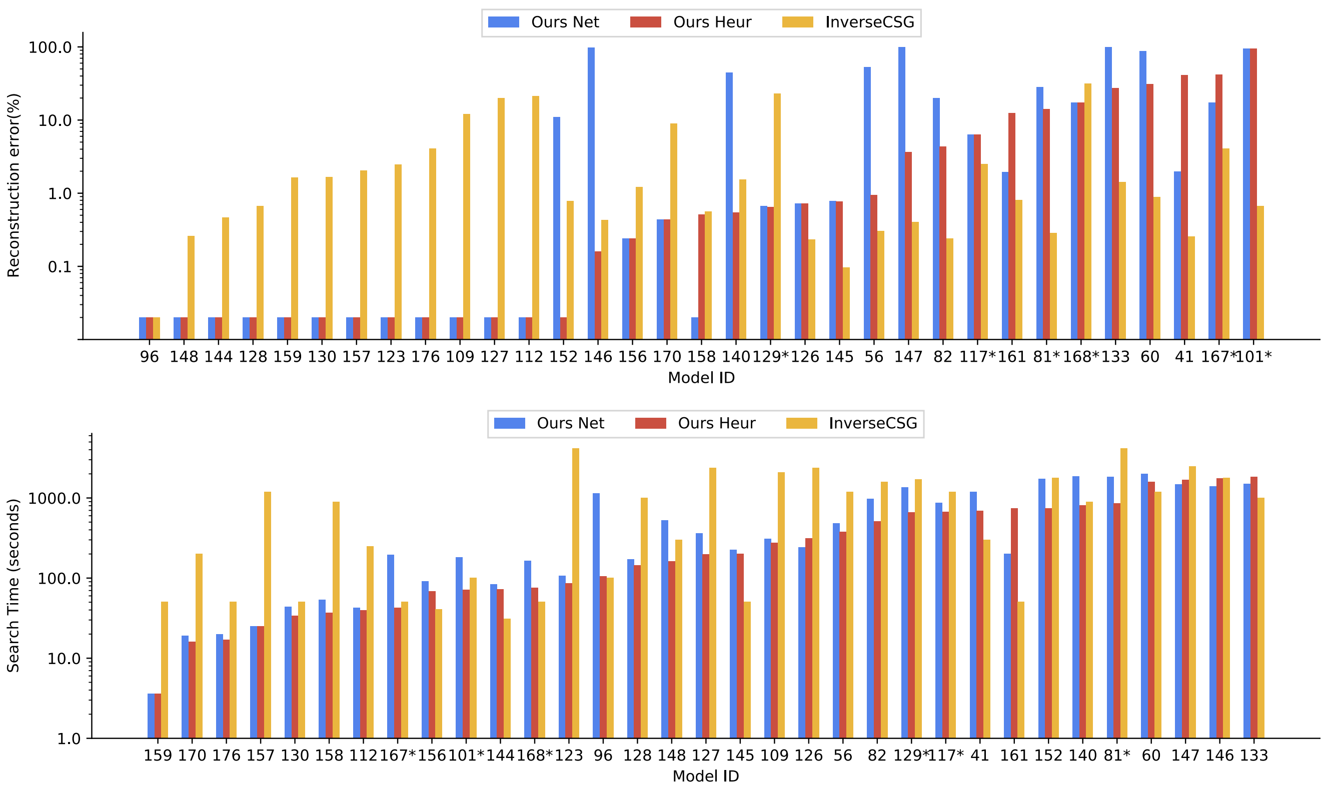


Figure 8: Per model comparison between InverseCSG and our method using heuristic (Ours Heur) or network guided (Ours Net) search. Results are show for 33 of the 50 models in the InverseCSG test set. Models IDs with an * contain operations other than sketch + extrude. Top: Reconstruction error computed using IoU. We use the reconstructed models released by the InverseCSG authors to evaluate. Bottom: Search time in seconds. To compare we use the official times from the InverseCSG paper.

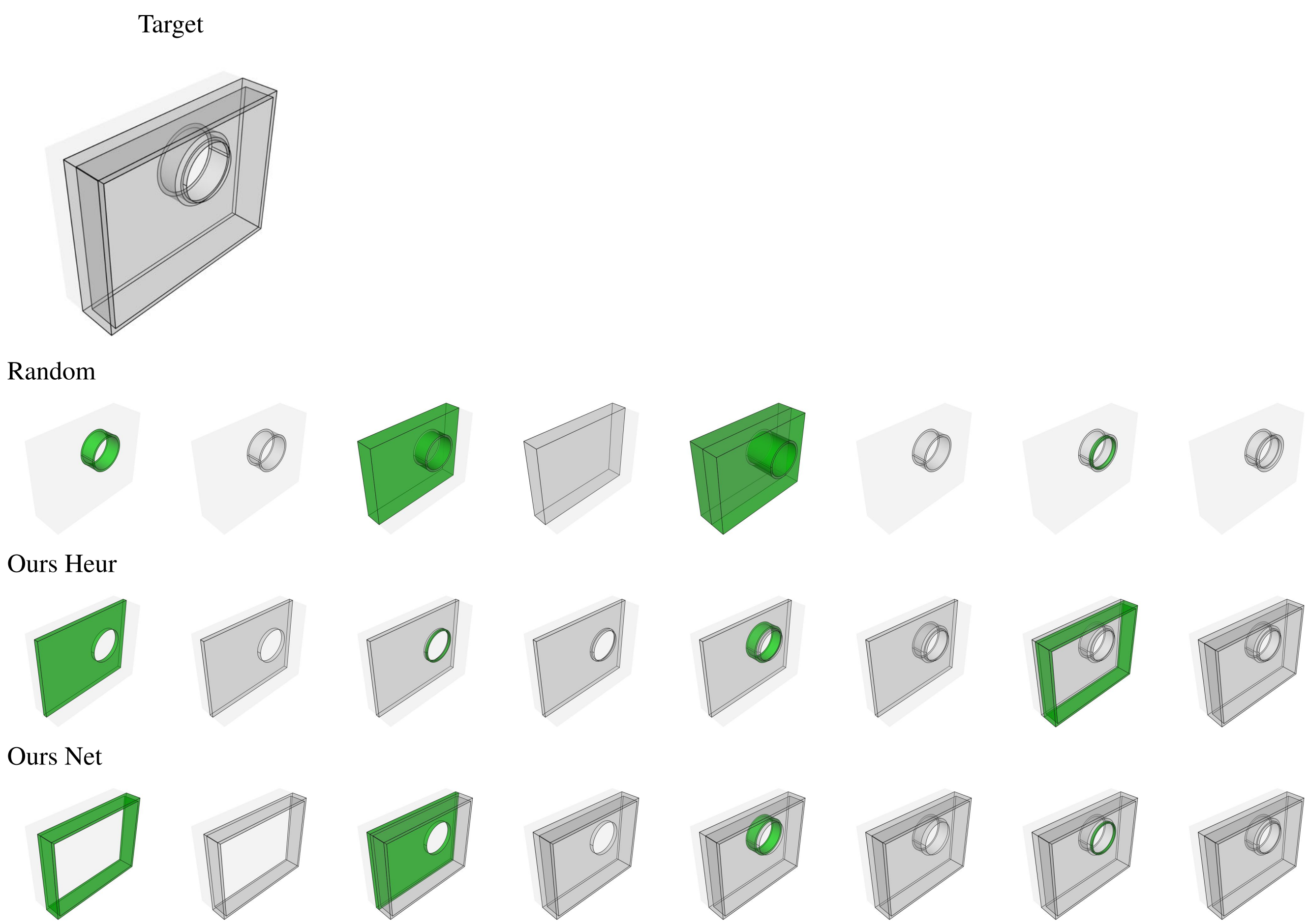


Figure 9: Qualitative comparison of the output of our model's inferred programs (Ours Net) vs. those of Random and Ours Heur. Green: addition, Red: subtraction, Grey: current. (Case 1)

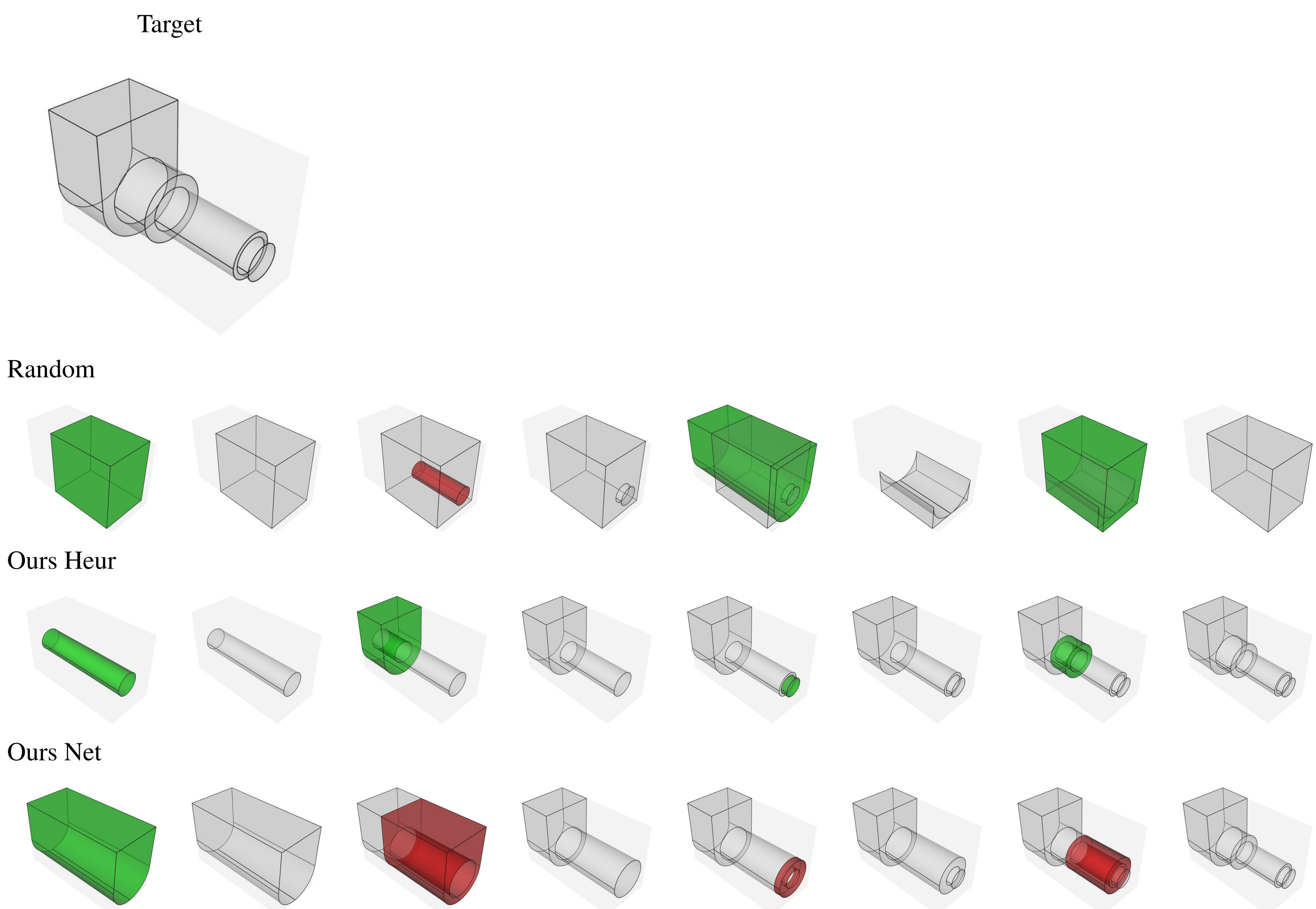


Figure 10: Qualitative comparison of the output of our model's inferred programs (Ours Net) vs. those of Random and Ours Heur. Green: addition, Red: subtraction, Grey: current. (Case 2)

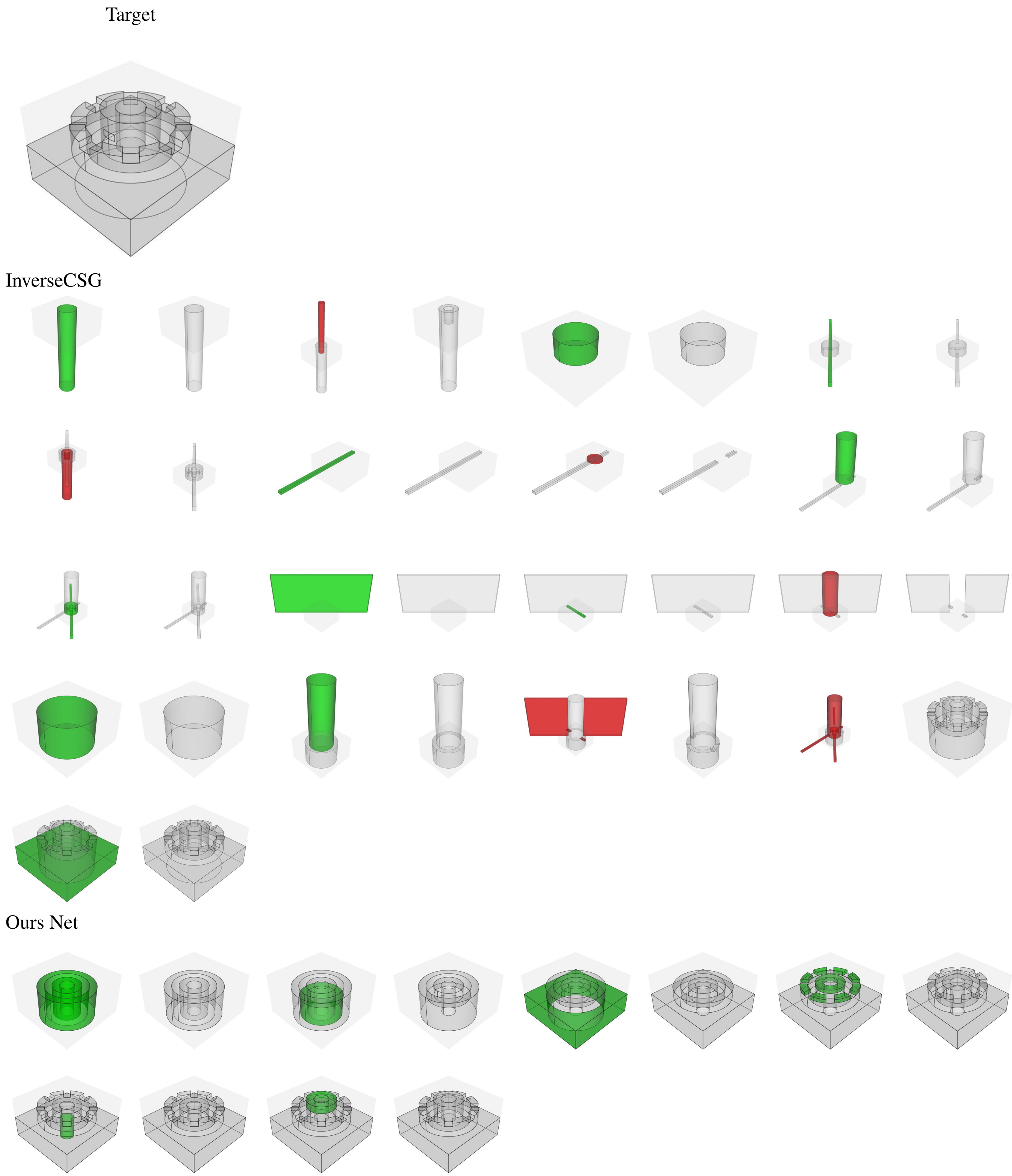
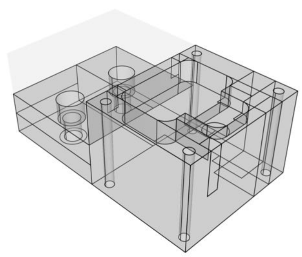


Figure 11: Qualitative comparison of the output of our model's inferred programs (Ours Net) vs. InverseCSG. Green: addition, Red: subtraction, Blue: intersection, Grey: current. (Case 1)

Target



InverseCSG

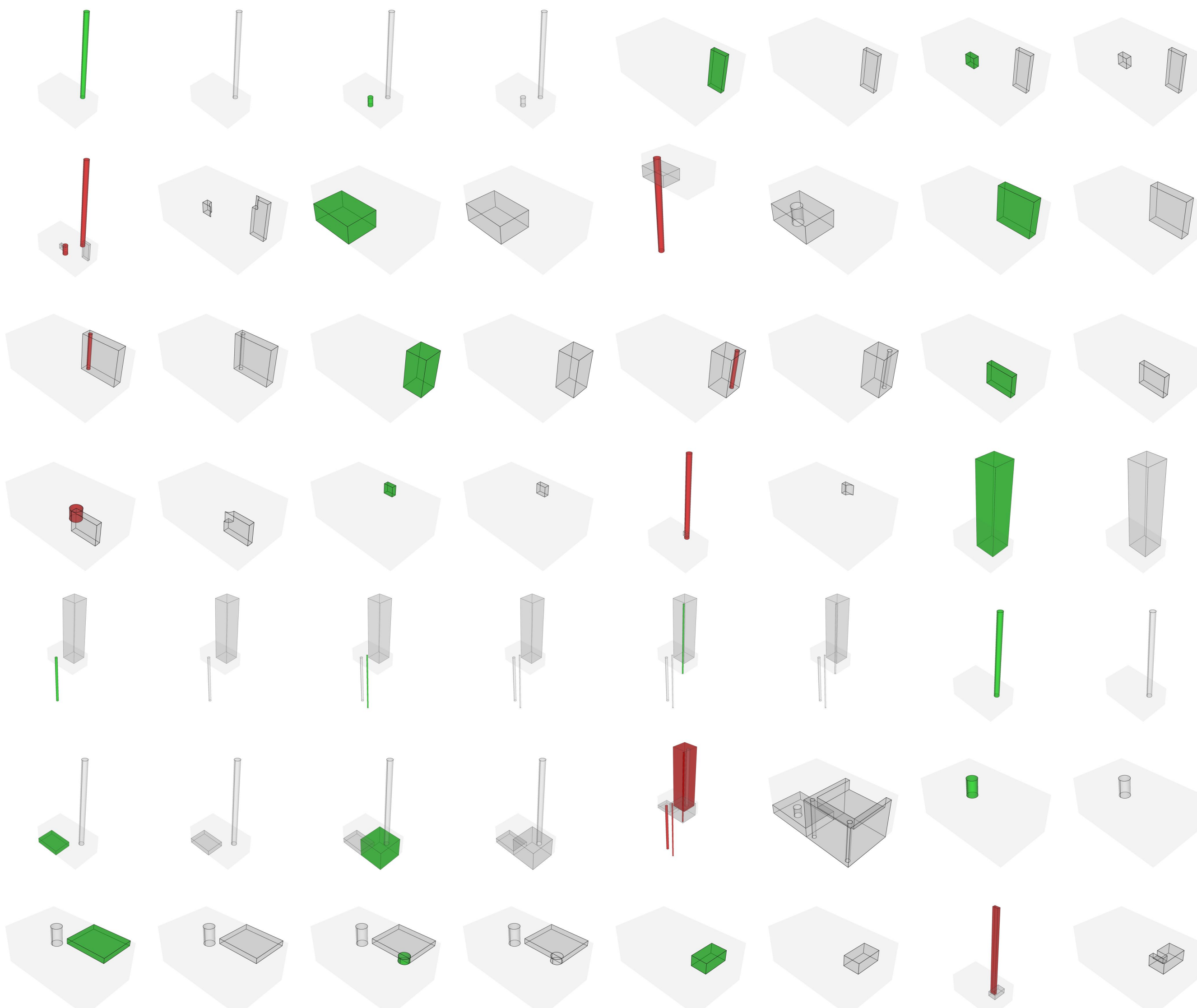
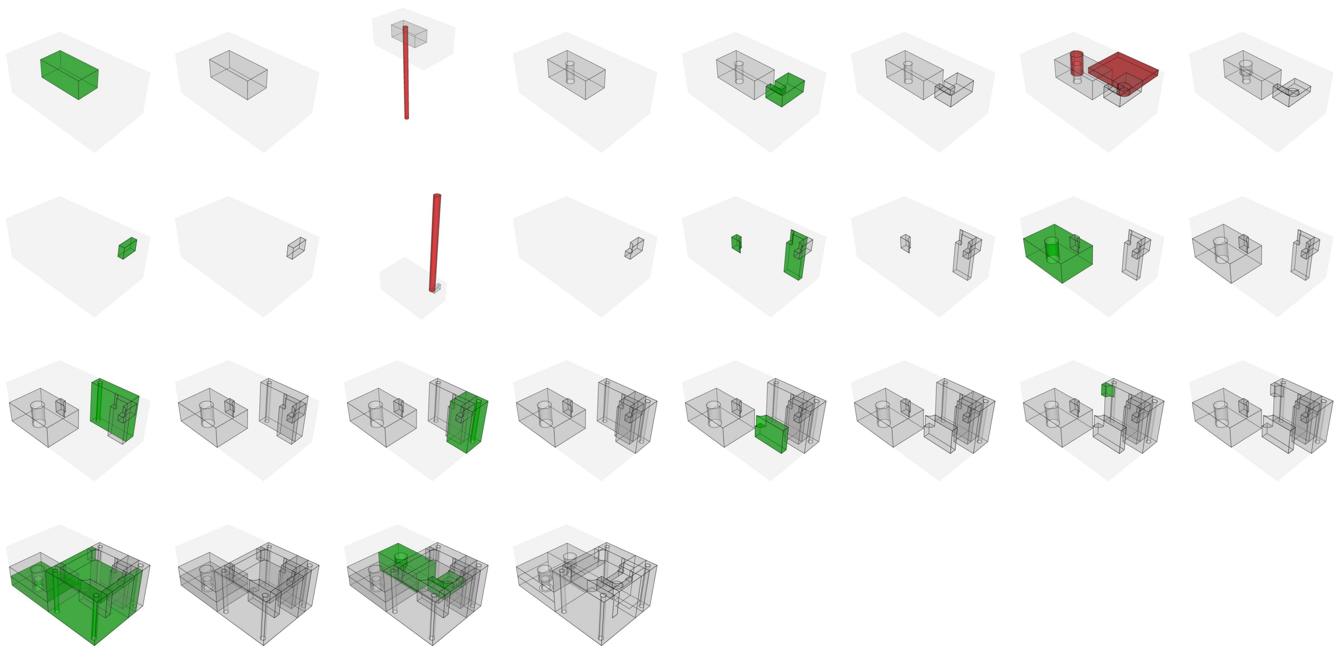


Figure 12: Qualitative comparison of the output of our model's inferred programs (Ours Net) vs. InverseCSG. Green: addition, Red: subtraction, Blue: intersection, Grey: current. (Case 2, Part 1)



Ours Net

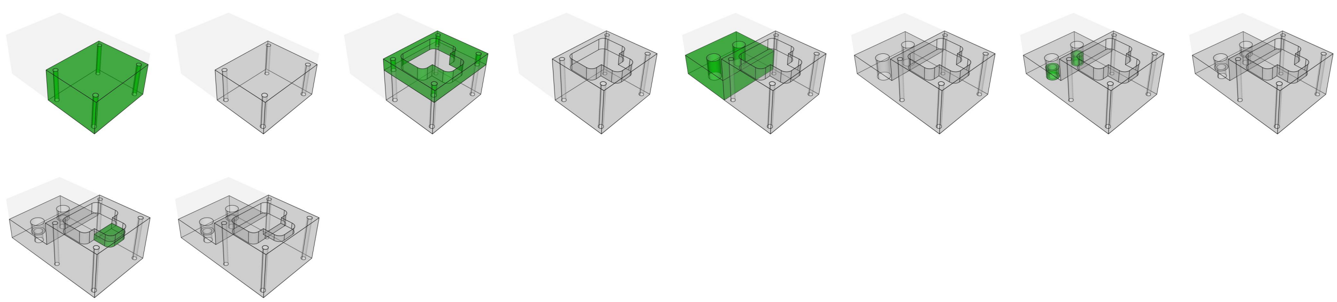


Figure 12: Qualitative comparison of the output of our model's inferred programs (Ours Net) vs. InverseCSG. Green: addition, Red: subtraction, Blue: intersection, Grey: current. (Case 2, Part 2)

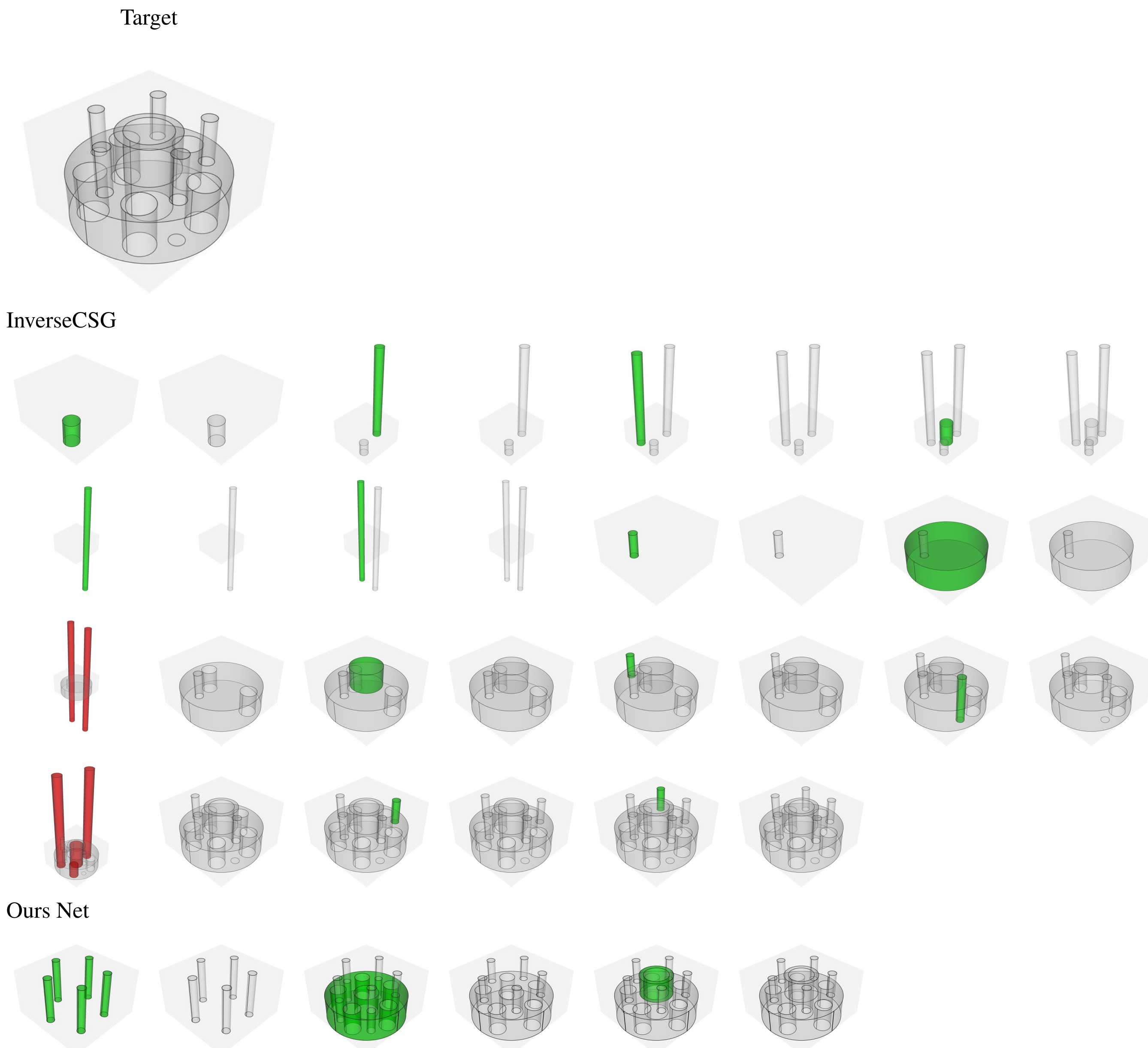


Figure 13: Qualitative comparison of the output of our model's inferred programs (Ours Net) vs. InverseCSG. Green: addition, Red: subtraction, Blue: intersection, Grey: current. (Case 3)

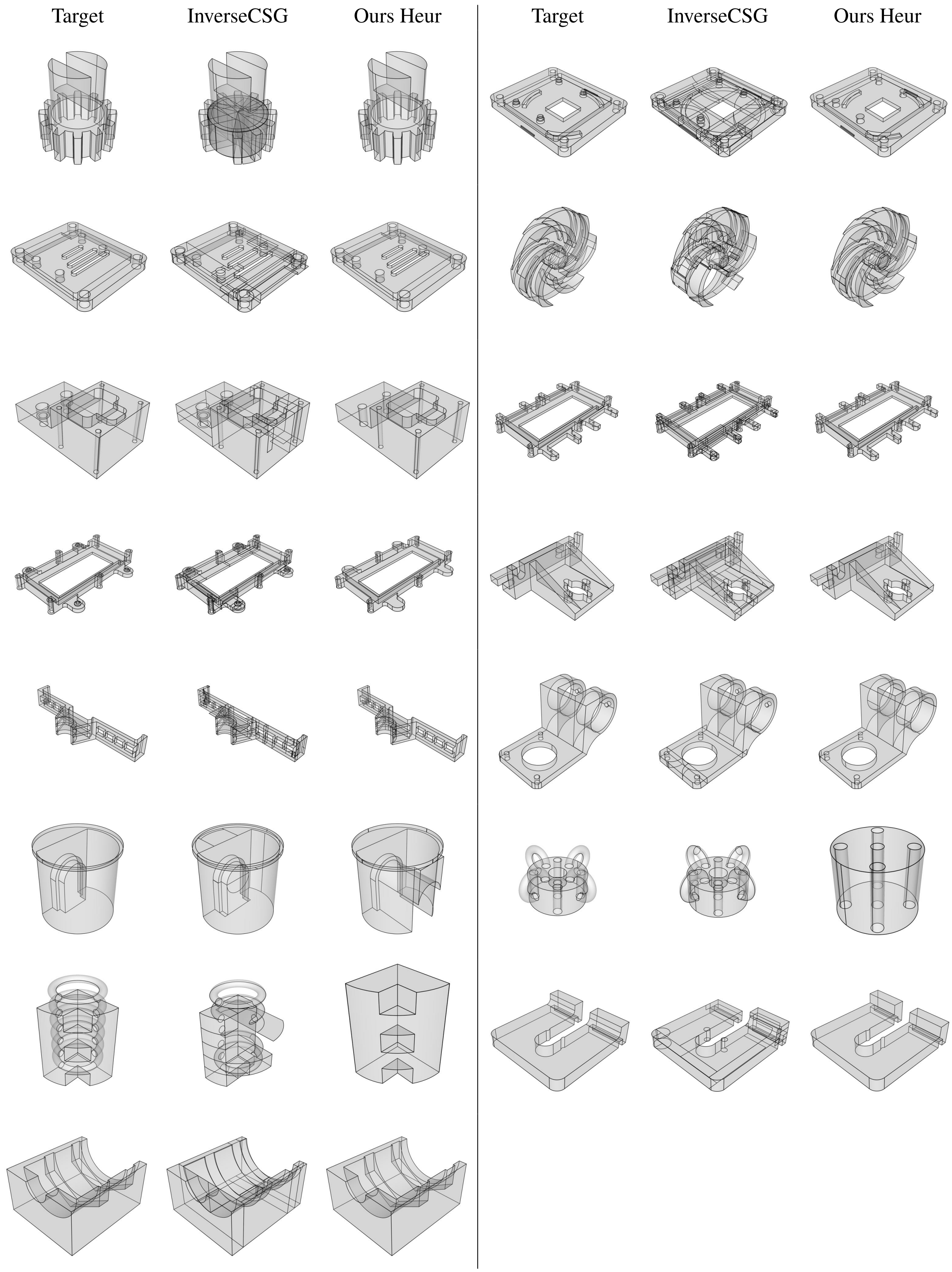


Figure 14: Reconstruction result comparison of InverseCSG vs Ours.