Automated Line Labelling: Dataset for Contour Detection and 3D Reconstruction

Hari Santhanam* Nehal Doiphode* Jianbo Shi
University of Pennsylvania
{harisan, lahen, jshi}@seas.upenn.edu

Abstract

Understanding the finer details of a 3D object, its contours, is the first step toward a physical understanding of an object. Many real-world application domains require adaptable 3D object shape recognition models, usually with little training data. For this purpose, we develop the first automatically generated contour labeled dataset, bypassing manual human labeling. Using this dataset, we study the performance of current state-of-the-art instance segmentation algorithms on detecting and labeling the contours. We produce promising visual results with accurate contour prediction and labeling. We demonstrate that our finely labeled contours can help downstream tasks in computer vision, such as 3D reconstruction from a 2D image.

1. Introduction

Computer Vision is filled with many different applications, like 3D shape understanding, segmentation, and robotic hand object interaction. For example, in the robot hand object interaction domain, robots have to adapt to learn how to pick up, sort, and grasp objects, which is a 3D segmentation and estimation problem. An understanding of the finer contextual details of these objects, even before training of the robot, could potentially be very impactful. Specifically, learning the contact points between a hand and an object is entirely dependent on an object's contours and junctions. Labelling the convexity of each contour and locating the junctions provides greater insight into the basic understanding of an object; it is like learning a vocabulary of simple words that make up larger sentences. This additional intuition can greatly aid in recognizing novel shapes, reducing the need for arduous human labelling of real objects with similar shapes. The primary problem is that no line labelled datasets currently exist. Thus, our main objective is to create the first 2D line labelled dataset, using information from 3D models.

*Equal contribution. Ordering determined at random.

The rules for convexity line labelling are mainly theoretical [30][12][20], making the actual implementation quite cumbersome and challenging. To bridge the gap in the literature and create a novel 2D line labelled dataset, we must formulate a new set of rules such that in practice implementation is feasible. Consequently, we generate an algorithm for automatic line labelling, that does not require human manual labelling power. The algorithm, which assumes as input a 3D STL mesh model, consists of three unique stages: contour extraction, contour classification, and contour grouping. We base the Huffman-Clowes [16][9] line labelling on the Thingi10K dataset [37], due to its rich variety of 3D printed models, compared to other benchmark datasets like ShapeNetCore [3]. With our automated algorithm, we generate 6,275 labelled 2D instances, originating from the genus 0 and genus 1 subsets of the Thingi10K dataset [37]. We convey a method of generating 2D line labels from 3D CAD models, that results in a novel dataset which can be used for downstream applications.

To illustrate our dataset's potential, we analyze the benchmark performance on state of the art segmentation architectures, like Mask2former [4] and SOLOv2 [31]. We also experiment with endpoint prediction in SOLOv2 [31] and Bézier curve early fusion with Mask2former [4]. Additionally, for models with no STL meshes where labelling is not possible, generalization of our 2D segmentation models is highly important. As a result, we show an ablation study on seen vs unseen objects in our dataset. We emphasize that with only 6,275 data instances, we are able to achieve promising levels of segmentation performance. To verify the use of our dataset, we demonstrate the utility for a 3D reconstruction task. Using the 3DR2N2 RNN based architecture [7], we show the increase in performance with the addition of our convexity labels as input to a separate encoder. Our performed experiments suggest that the utilization of our novel 2D line labelled dataset can be highly beneficial in downstream tasks in vision.

The main contributions of this paper are summarized as follows:

Contour Types	Train	Test
Obscuring	61226	34654
Concave	7509	4452
Convex	31143	16398
Total	99878	55504

Table 1. Dataset statistics

Error Types	Extraction	Labelling	Grouping
	0.68% = 8/1174	1.72% = 20/1166	4.55% = 53/1166

Table 2. Automated Labelling errors. Extraction error is computed over total number of contours extracted. Labelling and grouping errors are computed over correctly extracted contours.

(1) We propose a novel, dataset creating algorithm that takes as input a 3D STL mesh and outputs 2D extracted, classified, and grouped contours. This entire process is fully automated, with no human labelling power required. With a 3D STL mesh, and some additional hyperparameter tuning, an unlimited amount of 2D labelled data can be generated.

(2) We generate a novel line labelled dataset, consisting of 6,275 labelled instances from the genus 0 and genus 1 subsets of Thingi10K [37]. To our knowledge, this is the first 2D convexity line labelled dataset. The generated scenes have various contours, from straight lines to complex curves, with convexity line labels.

(3) We explore the performance of state of the art segmentation algorithms, such as Mask2former [4] and SOLOv2 [31], on our dataset. We study certain ablations of these architectures as well. Additionally, we show the results on seen and unseen objects from our dataset.

(4) We verify the utility of our labels by incorporating them into a 3D reconstruction model, and show the improved performance with the addition of an encoder to an RNN based architecture [7].

2. Related Work

There is very limited work on the creation of line labelled datasets. One main work, [15], uses a method to extract face, edge, visibility, and convexity-concavity information from a standard BREP CAD format of 3D models [21]. However, rendering of these models at different viewpoints requires a great deal of processing power and lacks support in Blender software. As a result, the convexity information in 2D cannot easily be determined. To aid in this venture and create an accessible dataset, we develop the faster runtime, novel methodology described in Section 3 that takes as input 3D STL meshes.

To understand the benchmark performance on our datset, we study CNN based contour detectors. For example, [19] uses a neural network to learn details at multiple scales for edge boundary prediction. Other CNN based work has focused on optimized training for contour prediction, and on classification of difficult edges, like shadows, illumination differences, and depth edges [25][28]. Fast-RCNN type of contour detectors also exist. Specifically, Mask-RCNN [14] has been used to accurately label the boundary of images, in order to improve segmentation performance [6]. Additionally, SOLOv2 [31] predicts masks by learning a kernel for every grid cell location. Segmentation architectures, like SOLOv2, are especially advantageous because they allow us to not only locate contours from our dataset, but also to classify their convexity type.

We also explore transformer based architectures for our benchmark study. Transformers are well known for their high levels of performance in natural language processing [32][17][27] and vision, due to their self-attention mechanism [18] [22]. For instance, ViT [10] embeds an image as a sequence of patches and removes convolutions altogether for image recognition. For object detection, DETR [1] uses a bipartite matching loss that ensures distinct predictions. A line detection transformer architecture, LETR [34], uses a multi-scale encoder and decoder, with a bipartite loss and distance loss, to accurately classify straight lines in the York Urban dataset. Moreover, the first edge detection transformer, EDTER [26], uses a global transformer to understand higher level details and a local transformer to understand finer details. Finally, for segmentation, Mask2former [4] uses masked attention and a pixel decoder for quality mask prediction. As mentioned for SOLOv2, we are especially motivated by Mask2former, as it can additionally predict our convexity labels.

We explore related works in 3D reconstruction that take advantage of invariant geometric properties across 2D viewpoints or perform estimation of priors, like depth and silhouette to improve reconstruction. For example, [35] enforces consistency between perspective transformations of a predicted 3D shape and its corresponding 2D input observation. Additionally, [33] is an end-to-end trainable model that estimates depths, normals, and silhouettes from 2D images to predict a 3D shape, followed by a reprojection consistency loss. In another work, Front2Back [36] induces structure and geometric constraints to accurately predict reflective symmetry and consistent silhouette from 2D images. El Banani et al. [11] propose a 3D geometry aware feature bottleneck for estimating novel viewpoints. Inspired by these works, we utilize contour properties from our proposed dataset for 3D object reconstruction from single and multiview images.

3. Dataset

In this section, we describe the process by which we create our 2D line labelled dataset, with labelled contours ranging from basic straight lines to complicated curves. Our

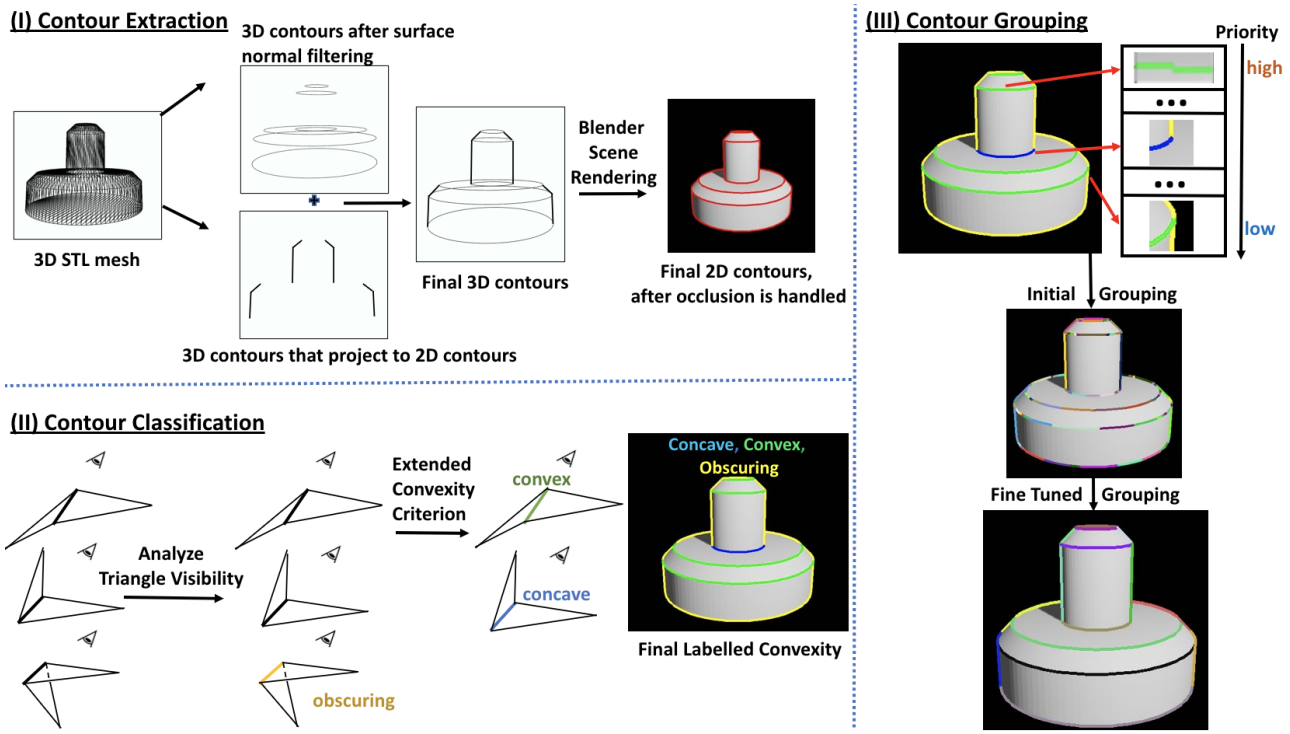

Figure 1. An overview of our dataset pipeline that consists of (I) Contour Extraction, (II) Contour Classification, and (III) Contour Grouping.

fully automated algorithm, as shown in Fig. 1, begins from 3D printed meshes and ends with 2D scenes that contain extracted, classified, and grouped contours. The final dataset consists of 6,275 shapes, with the distribution of convexity in rendered viewpoint images as shown in Table 1. Additionally, Table 2 shows error metrics for each stage of the automated dataset creation pipeline based on a random sample of 100 images.

3.1. Preliminaries

The Huffman and Clowes line labelling scheme encodes contour convexity information that is integral for understanding 3D scenes [16] [9] [29][24] [2]. Moreover, an entire scene can be fully depicted by the following contours: **(i) Concave**: Contours occur when two faces form a valley with respect to the camera. **(ii) Convex**: Contours occur when two faces form a ridge with respect to the camera. **(iii) Obscuring**: Contours are the combination of occluding and limb contours. Occluding contours occur when one visible face with respect to the camera intersects with another face that is out of view of the camera. Limb contours occur when a visible surface curves out of view of the camera [23].

3.2. Contour Extraction

Since the CAD models in Thingi10K are presented as STL models that are defined with 3-dimensional triangular meshes, we must first develop a method for extracting 3D contours. Initially, we posit that the intersection of adjacent triangles forms the set of prospective 3D contours. Note many of the meshes, especially ones that define cylindrical and spherical surfaces, contain triangles whose intersection does not signify the visible presence of a contour. Thus, we use a surface normal discontinuity to filter these out. Let A represent the initial set of prospective contours formed from the intersection of adjacent triangles, and let S represent the contours in A that adhere to $\theta_{disc} < 0.2$, where θ_{disc} represents the angle between the surface normals of two adjacent triangles. Additionally, since we aim to locate the 3D contours that contribute to accurate 2D line labelling, we must consider cases where a 2D contour results only due to projection. For instance, the projection of the lateral surface of a cylinder yields two obscuring (specifically limb) 2D contours that only occur due to projection. Consequently, we denote the set of 3D contours P, such that $P \in A$, $P \notin S$, with P adhering to the rules prescribed for an obscuring contour. The combination of S and P form the total set of 3D contours.

The software Blender allows us to render 3D models into 2D scenes, by varying the camera placement and light source for scene capturing. Naturally, the rendering can result in cases of occlusion; a point on a 3D contour is considered occluded, if it is not seen from the camera's vantage point. To aid in occlusion detection, we introduce a built-in Blender function, called ray_cast, that determines if a ray

can travel from the camera to a destination point on a 3D contour without obstruction. The simplest, yet most time-consuming method, would be to sample every point along the 3D contour and determine if it is occluded. Due to time constraints, we seek a faster run-time solution. An additional assumption we can make, due to empirical study of the renderings, is that only one occlusion occurs per contour. As a result, we identify the following main cases of occlusion for linear Contour C with endpoints c_1 and c_2: (i) both endpoints are visible, yet the contour is fully occluded, (ii) right endpoint occluded, with visible portion of contour containing left endpoint, (iii) left endpoint occluded, with visible portion of contour containing right endpoint, (iv) both endpoints are visible, yet occlusion occurs somewhere between c_1 and c_2.These occlusions are not exhaustive, but are the main ones we saw in genus 0 and genus 1 of Thingi10K.

The formal algorithm is listed as follows. Moving from left to right along the curve, use a search to locate the first visible and last visible point, with the ray_cast function. We perform the search by testing $N_{global} = 8$ points first, and then performing a confined search, $N_{local} = 30$, in the interval where the visibility of the points change. In the forward search, let $\hat{c_{1f}}$ be the first visible point and $\hat{c_{2f}}$ be the last visible point. Additionally, we perform the same search in the opposite direction, resulting in $\hat{c_{1b}}$ as the first visible point and $\hat{c_{2b}}$ as the last visible point. We summarize the cases as follows:

(1) $\hat{c_{1f}} = \hat{c_{2f}} = c_1$ and $\hat{c_{1b}} = \hat{c_{2b}} = c_2 \rightarrow$ both endpoints are visible, but the full segment is occluded.

(2) $\hat{c_{1f}} = c_1$ and $\hat{c_{2f}} \neq c_2$, and c_2 is not visible $\rightarrow$ right endpoint occluded, while left part of contour is visible.

(3) $\hat{c_{1b}} = c_2$ and $\hat{c_{2b}} \neq c_1$, and c_1 is not visible $\rightarrow$ left endpoint occluded, while right part of contour is visible.

(4) $\hat{c_{1f}} = c_1$, $\hat{c_{2f}} \neq c_2$, and $\hat{c_{1b}} = c_2$, $\hat{c_{2b}} \neq c_1 \rightarrow$ occlusion occurs somewhere between 2 endpoints, resulting in two contours.

3.3. Contour Classification

In order to classify the Huffman-Clowes contours, we first subdivide the problem to differentiate between obscuring contours and concave/convex contours. From the provided convexity definitions, the main discriminator between the two types are the triangular faces that intersect to form the contour. If both faces are visible (partially or fully) to the camera, then the contour is concave/convex. If either one of the faces is hidden, then the contour is obscuring. Therefore, determining the visibility of a triangle, amounts to learning important information about a contour's convexity.

A brute force solution for determining a triangle's visibility is by analyzing the ability for an incident ray, from the camera center, to strike every point within the triangle. As mentioned in Section 3.2, a ray cast method like this would be time consuming. As a result, we delineate points along each angle bisector, as a representative set of points to sample to determine triangle visibility. Formally, let $s_1 \dots s_n$ illustrate sample points along each angle bisector. Consider again the ray_cast function, that returns 1 if the incident ray can strike point s_i within the triangle. Consequently, triangle T is hidden if $\sum_{i=1}^{n} ray\_cast(s_i) = 0$, because not a single ray from the camera was able to strike any point within the triangle. Otherwise, the triangle is visible. This method allows us to differentiate completely between obscuring and concave/convex contours.

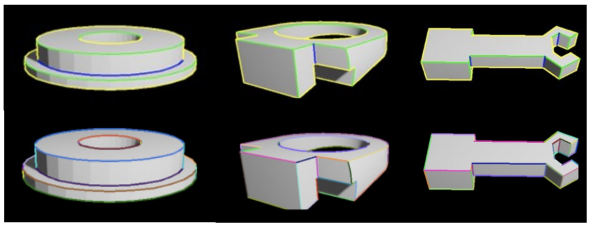

Figure 2. An example result of the resulting ground truth classifications (top row) and corresponding groupings (bottom row). In the top row, 'obscuring' contours are yellow, 'concave' contours are blue, and 'convex' contours are green.

Next, we follow the Extended Convexity Criterion in the work by Stein et al. [8] to differentiate between concave and convex contours. We establish vectors from the camera origin to each triangle centroid, as vectors $\vec{x_1}$ and $\vec{x_2}$. Moreover, the unit displacement vector, $\vec{d} = \frac{\vec{x_1} - \vec{x_2}}{||\vec{x_1} - \vec{x_2}||}$, and triangle unit normals, $\vec{n_1}$ and $\vec{n_2}$, fully establish the convexity of the contour, as a convex contour occurs when $\vec{n_2} \cdot \vec{d} < \vec{n_1} \cdot \vec{d}$ otherwise it is concave [8]. Final example results of classification are shown in Fig. 2.

3.4. Contour Grouping

The contours extracted are solely linear, as the curves in our dataset, to this point, are approximated with finite lines. In later downstream tasks, like contour detection, this is a challenge as state of the art segmentation algorithms cannot detect such small segments. In order to fully leverage the potential of our line labelled dataset, we group the smaller linear segments that form a particular curve. Formally, we construct a weighted graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$, where each contour is a node and each junction connecting two contours is an edge. The associated cost for each edge is the local curvature, which is measured by the angle between the two contours' unit normals. The objective is to link graph nodes, in such a way that minimizes the total cost, in order to formulate the minimum spanning tree (MST) [13]. The specifics of the grouping algorithm are defined as follows:

- We initialize a heap that designates each contour's

Metric	Seen - mAP			
	mAP	obscuring	convex	concave
SOLOv2	22.79	30.24	27.54	10.37
SOLOv2 + endpoint	23.54	31.68	29.82	12.65
Mask2former	27.3	37.43	**33.87**	12.19
Mask2former + earlyfusion	**29.37**	**42.31**	30.38	**15.04**

Table 3. Quantitative mAP results for seen models.

Metric	Unseen - mAP			
	mAP	obscuring	convex	concave
SOLOv2	17.46	27.73	20.47	5.15
SOLOv2 + endpoint	20.19	24.15	**29.82**	7.70
Mask2former	22.3	27.15	26.93	**12.34**
Mask2former + earlyfusion	**23.1**	**28.21**	27.80	11.23

Table 4. Quantitative mAP results for unseen models.

grouping priority, which is dependent on its number of neighbors and local curvature. The need for this ordering is essential, as we are more inclined to group "easier" contours that have little curvature differences, rather than "harder" contours closer to junctions. As each contour is popped off the heap, we deem if linking with its neighbors is necessary. We link contours A and B iff:

$$cost(A,B) = \begin{cases} 0 & l(A) = l(B), \theta_{AB} < C_{thresh} \\ 1 & otherwise \end{cases} \quad (1)$$

where $l(\cdot)$ returns the contour's label, θ_{AB} is the angle between contours A and B's unit normals, and C_{thresh} is a grouping threshold that determines if grouping is necessary. In order to minimize the total cost, the first condition must be satisfied for grouping to occur.

- The linking of two contours, A and B, results in super contour AB with unit normal $\hat{c_{AB}}$, which is the mean of $\hat{c_A}$ and $\hat{c_B}$, and label $l(AB) = l(A) = l(B)$. Future linking of super contours must adhere to the cost expression, with grouping for a super contour with its neighbor concluding when the cost for linking is 1. Overall grouping is concluded, when super contours can no longer be combined.

- We further link super contours at a local level, as the order of initial grouping influences group generation. To this end, we introduce the following criteria to perform additional grouping for super contours X and Y: **(i)** The labels must match, $l(X) = l(Y)$. **(ii)** If Y is X's right neighbor and X is Y's left neighbor, then Y must be the only right neighbor and X must be the only left neighbor. **(iii)** For given contours $x \in X$ and $y \in Y$, such that x and y are direct neighbors, the angle between their respective unit normals, θ_{xy}, must adhere to the condition $\theta_{xy} < C_{thresh2}$. Final example results of grouping are shown in Fig. 2.

4. Contour Labelling by Prediction

Consider 2D scenes, where 3D models do not exist for automated labelling. To analyze the convexity in these new scenes, we need a generalizable segmentation algorithm trained on our novel dataset. In this section, we aim to (1) understand the SOTA segmentation performance on our dataset and (2) analyze the performance on seen and unseen models, for understanding generalizability.

4.1. Approach

In order to formulate this as a segmentation problem, each contour C is dilated to form mask M, which has a thickness of 6 pixels wide. Since humans are able to classify the contour type by merely looking at the local context around a contour, the dilated masks preserve features like shading and reflectance. Moreover, each mask M is given a label of concave, convex, or obscuring. In order to avoid memorization of the boundaries and enhance generalization performance, we add ImageNet images as background clutter to the rendered 2D images.

We choose SOLOv2 [31] and Mask2former [5] as our state of the art architectures. These architectures are the top performing baselines and are representative of architectures that perform dynamic and attention based mask learning. Specifically, SOLOv2 [31] predicts a mask by learning a kernel for every location in a grid cell. Mask2former [4] uses masked attention to extract context features and a high quality pixel decoder that provides high resolution features for quality mask prediction. Furthermore, we perform ablation studies on these two architectures, as we seek to incorporate more contour specific information. For example, we add an endpoint regression step to SOLOv2 [31], which not only predicts the segmentation, like baseline SOLOv2 does, but it also predicts the endpoints of

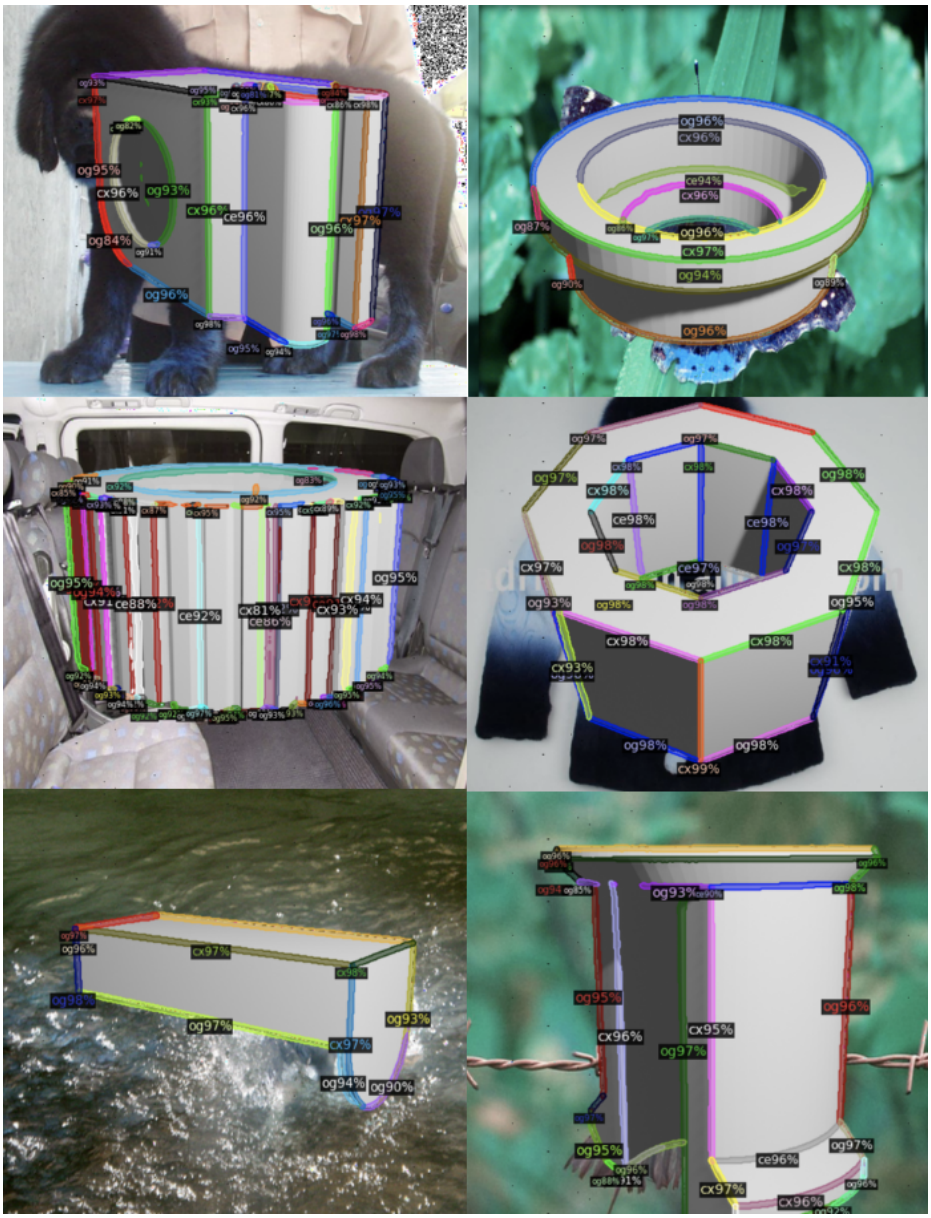

Figure 3. Visualization results for Mask2former [5] with early fusion. 'og' represents obscuring, 'cx' represents convex and 'ce' represents concave.

the contours. The idea is to provide a bottom up signal for potential downstream tasks, and also reduce confusion near junctions. For Mask2former [4], we experiment with early fusion with Bézier curve data. Contrary to baseline Mask2former, which is trained on just RGB 3 channel inputs, this Mask2former is trained with an additional channel representing the Bézier curve masks. The Bézier mask, in the new 4 channel input, provides a positional prior that can aid in finding the location of curved contours. Also, the Bézier masks are generated solely on the foreground objects and not with background clutter. Ultimately, Bézier curve analysis and Transformers are rooted in token-based symbolic reasoning, making them suitable for an integrated fusion in the computation path.

4.2. Experiments

In this subsection, we evaluate the regression ability of the instance segmentation frameworks, on both seen and unseen objects.

Dataset - We generate two types of data: 1)Unseen models - Data is randomly split such that each split has a disjoint set of models, (2) Seen models - Data is randomly split such

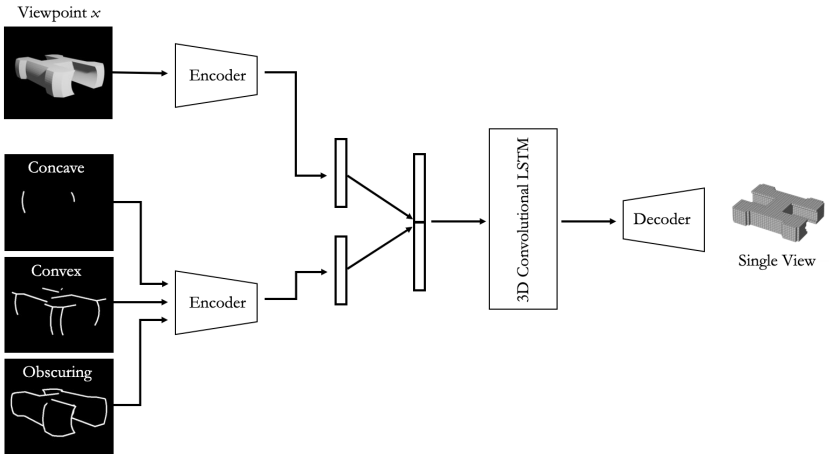

Figure 4. Modified Architecture of 3DR2N2 [7], for single view, with latent encoder fusion.

that the splits have overlapping models.

Metrics - Mean Average Precision(mAP) and Contour Accuracy(CA) are used as metrics for evaluation. mAP is the mean Average Precision metric commonly used in instance segmentation frameworks. We calculate a labelwise mAP metric too. CA is calculated as the IoU overlap between the detected contour mask and ground truth contour mask, after matching detections to the groundtruth.

Results - The obtained results are given in Tables 3 and 4. For SOLOv2 [31], it is observed that adding an endpoint regression loss improves performance, compared to baseline SOLOv2. In Mask2former [5], early fusion with Bézier data also helps improve performance, compared to baseline Mask2former. Also, concave and convex contour predictions generalize well, as the unseen and seen mAP numbers across architecture are very similar. Finally, comparing the mAP from both tables, we observe that Mask2former + earlyfusion has the best performance for contour prediction in both seen and unseen cases. For further analysis, we compute the CA metric on the Mask2former + early fusion model. For seen objects, the CA is 0.81, whereas for unseen objects the CA is 0.74. The numbers denote significant overlap with the ground truth mask, indicating a high quality of predictions. Visualizations of contour predictions on both categories are shown in Fig 3. Through this analysis, we determine that (1) Mask2former + earlyfusion is the best performing SOTA method and (2) its performance is better on seen data than on unseen data, revealing its generalization performance.

5. Application for 3D Reconstruction

In this section, we demonstrate the utility of our generated dataset to improve performance for a 3D reconstruction task on Thingi10K [37].

5.1. Approach

For our task, we adapt the framework in 3DR2N2 [7], a recurrent neural network based model that takes as input a sequence of image viewpoints, or a single viewpoint, and outputs a 3D voxelized reconstruction. Since other algorithms use either point-cloud or mesh representations, and require engineering for additional input modalities, 3DR2NR allows us to directly quantify the impact of our dataset. In order to incorporate our labels, we modify the architecture as shown in Fig. 4, with a single viewpoint as input for illustration. Instead of a single encoder, we propose two encoders to account for the difference in the modality type of the inputs. Thus, the input to the first encoder is V, or up to 10 randomly sampled unique viewpoints for a particular model. The input to the second encoder is C, or a 3 channel binary mask image, where each channel is a binary mask for the label type. After initial processing, the output latent vectors from the encoders are fused together to encode both the shape and local contour features. The resulting vector is then decoded using a 3D Convolutional LSTM and a 3D Convolutional Decoder to output voxel occupancy probabilities.

In [7]'s experiments involving ShapeNet [3], each category consists of over 1000 views available for training. In our case, we are limited to around 10 views per model, with rarely any overlap in model category. Since we have fewer viewpoints, we aim to improve generalization performance by applying various types of transformations to our input, such as random crop, resize, flip, and rotation.

5.2. Experiments

To validate our approach, we choose the 3D-LSTM-3 architecture in [7] as the baseline. Our experiments aim to address two questions - 1)How does the proposed dataset

Model	I:256, V:32		I:256, V:64		I:128, V:32		I:128, V:64	
	seen	unseen	seen	unseen	seen	unseen	seen	unseen
3DR2N2	0.45	0.23	0.47	0.27	0.43	0.21	0.33	0.18
3DR2N2 + contour labels	0.53	0.31	0.55	0.40	0.51	0.26	0.50	0.25

Table 5. Iou for varying input image size and varying output voxel sizes for baseline vs modified architecture.

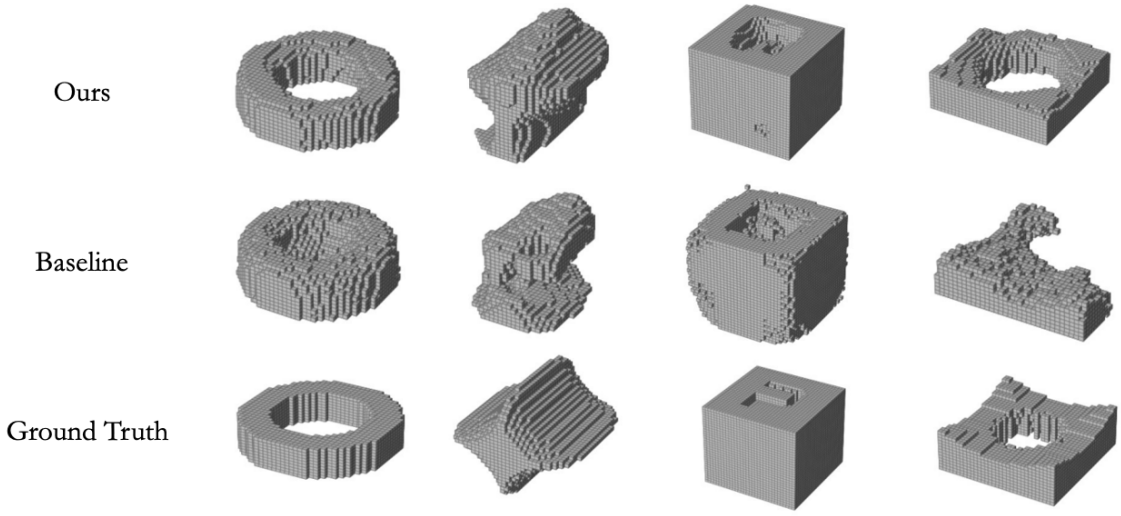

Figure 5. Visualization comparison of 3DR2NR with labels (Ours), 3DR2N2 (Baseline), and Ground Truth. The first 2 columns are for seen objects and the latter 2 columns are for unseen objects.

perform in the task of 3D reconstruction? and 2)How does the proposed dataset affect generalization to different viewpoints for both seen and unseen objects?

Dataset - We use the same dataset written in Section 4.1, but with splits (80/10/10 %).

Metric - We use the Intersection over Union (IoU) metric to evaluate our 3D reconstruction performance. IoU calculates the voxel similarity between the thresholded prediction and groundtruth voxel.

Results - As shown in Table 5, we see that our added labels improve IoU performance across every input image size and voxel size, for both seen and unseen models. This is consistent with prior works that use geometry information to improve reconstruction. Specifically, in our case we encode contour convexity information in 3DR2NR to remove some ambiguity in reconstruction [36] [33]. A visual comparison of the improvement with our added labels is shown in Fig.5. Clearly, quantitatively and qualitatively, our added convexity information is highly beneficial. This is especially impressive because Thingi10K is a challenging dataset for reconstruction, because it consists of very diverse shapes, varying in size, appearance, and geometry. Such an improvement in a challenging task illustrates the utility of our 2D contour labelled dataset, and highlights the importance of its creation.

6. Conclusion

With an entirely automated algorithm, we create the first ever 2D line labelled dataset that requires no human manual labelling power. Assuming a 3D STL mesh as input, we demonstrate the ability to generate rendered 2D scenes that contain extracted, classified, and grouped contours. Our 6,275 instance dataset, formed from a subset of genus 0 and genus 1 of Thingi10K, is especially advantageous for its potential use in other vision downstream tasks. For situations where 3D CAD models do not exist, a 2D contour detection framework is valuable. We are able to demonstrate good performance of state of the art segmentation algorithms for predicting contour labels on our line labelled dataset. Finally, we show the utility of our dataset for a 3D reconstruction task. We believe that our line labelled dataset serves as a foundational tool for greater object understanding.

Acknowledgments

We gratefully appreciate the support from Vy Corporation.

References

[1] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European conference on computer vision*, pages 213–229. Springer, 2020.

[2] Indranil Chakravarty. A generalized line and junction labeling scheme with application to scene analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (2):202–205, 1979.

[3] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015.

[4] Bowen Cheng, Ishan Misra, Alexander G Schwing, Alexander Kirillov, and Rohit Girdhar. Masked-attention mask transformer for universal image segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1290–1299, 2022.

[5] Bowen Cheng, Ishan Misra, Alexander G. Schwing, Alexander Kirillov, and Rohit Girdhar. Masked-attention mask transformer for universal image segmentation. 2022.

[6] Tianheng Cheng, Xinggang Wang, Lichao Huang, and Wenyu Liu. Boundary-preserving mask r-cnn. In *European conference on computer vision*, pages 660–676. Springer, 2020.

[7] Christopher B Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In *European conference on computer vision*, pages 628–644. Springer, 2016.

[8] Simon Christoph Stein, Markus Schoeler, Jeremie Papon, and Florentin Worgotter. Object partitioning using local convexity. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 304–311, 2014.

[9] M. B. Clowes. On seeing things. *Artif. Intell.*, 2:79–116, 1971.

[10] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[11] Mohamed El Banani, Jason J. Corso, and David Fouhey. Novel object viewpoint estimation through reconstruction alignment. In *Computer Vision and Pattern Recognition (CVPR)*, 2020.

[12] Eugene C Freuder. Information needed to label a scene. In *AAAI*, pages 18–20, 1980.

[13] Ronald L Graham and Pavol Hell. On the history of the minimum spanning tree problem. *Annals of the History of Computing*, 7(1):43–57, 1985.

[14] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[15] Binkui Hou, Zhigao Huang, Huamin Zhou, and Dequn Li. A hybrid approach for automatic parting curve generation in injection mold design. *The International Journal of Advanced Manufacturing Technology*, 95(9):3985–4001, 2018.

[16] David A. Huffman. Realizable configurations of lines in pictures of polyhedrat. 2013.

[17] Katikapalli Subramanyam Kalyan, Ajit Rajasekharan, and Sivanesan Sangeetha. Ammus: A survey of transformer-based pretrained models in natural language processing. *arXiv preprint arXiv:2108.05542*, 2021.

[18] Salman Khan, Muzammal Naseer, Munawar Hayat, Syed Waqas Zamir, Fahad Shahbaz Khan, and Mubarak Shah. Transformers in vision: A survey. *ACM Computing Surveys (CSUR)*, 2021.

[19] Jyri Kivinen, Chris Williams, and Nicolas Heess. Visual boundary prediction: A deep neural prediction network and quality dissection. In *Artificial Intelligence and Statistics*, pages 512–521. PMLR, 2014.

[20] Del Lamb and Amit Bandopadhay. Shape from line drawings: beyond huffman-clowes labeling. *Pattern recognition letters*, 14(3):213–219, 1993.

[21] Joseph G Lambourne, Karl DD Willis, Pradeep Kumar Jayaraman, Aditya Sanghi, Peter Meltzer, and Hooman Shayani. Brepnet: A topological message passing system for solid models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12773–12782, 2021.

[22] Tianyang Lin, Yuxin Wang, Xiangyang Liu, and Xipeng Qiu. A survey of transformers. *arXiv preprint arXiv:2106.04554*, 2021.

[23] Jitendra Malik. Interpreting line drawings of curved objects. *International journal of computer vision*, 1(1):73–103, 1987.

[24] Samihah Mardzuki and Habibollah Haron. Interpretation of drawing: A review on line labeling. In *The 4th Postgraduate Annual Research Seminar (PARS08)*, volume 30, pages 247–250, 2008.

[25] Mengyang Pu, Yaping Huang, Qingji Guan, and Haibin Ling. Rindnet: Edge detection for discontinuity in reflectance, illumination, normal and depth. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6879–6888, 2021.

[26] Mengyang Pu, Yaping Huang, Yuming Liu, Qingji Guan, and Haibin Ling. Edter: Edge detection with transformer. *arXiv preprint arXiv:2203.08566*, 2022.

[27] Xipeng Qiu, Tianxiang Sun, Yige Xu, Yunfan Shao, Ning Dai, and Xuanjing Huang. Pre-trained models for natural language processing: A survey. *Science China Technological Sciences*, 63(10):1872–1897, 2020.

[28] Wei Shen, Xinggang Wang, Yan Wang, Xiang Bai, and Zhijiang Zhang. Deepcontour: A deep convolutional feature learned by positive-sharing loss for contour detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3982–3991, 2015.

[29] EC-K Tsao and W-C Lin. Constraint propagation neural networks for huffman-clowes scene labeling. *IEEE transactions on systems, man, and cybernetics*, 21(6):1536–1548, 1991.

[30] David L Waltz. Generating semantic descriptions from drawings of scenes with shadows. 1972.

[31] Xinlong Wang, Rufeng Zhang, Tao Kong, Lei Li, and Chunhua Shen. Solov2: Dynamic and fast instance segmentation. *Advances in Neural information processing systems*, 33:17721–17732, 2020.

[32] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, pages 38–45, 2020.

[33] Jiajun Wu, Yifan Wang, Tianfan Xue, Xingyuan Sun, Bill Freeman, and Josh Tenenbaum. Marrnet: 3d shape reconstruction via 2.5 d sketches. *Advances in neural information processing systems*, 30, 2017.

[34] Yifan Xu, Weijian Xu, David Cheung, and Zhuowen Tu. Line segment detection using transformers without edges. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4257–4266, 2021.

[35] Xinchen Yan, Jimei Yang, Ersin Yumer, Yijie Guo, and Honglak Lee. Perspective transformer nets: Learning single-view 3d object reconstruction without 3d supervision. *Advances in neural information processing systems*, 29, 2016.

[36] Yuan Yao, Nico Schertler, Enrique Rosales, Helge Rhodin, Leonid Sigal, and Alla Sheffer. Front2back: Single view 3d shape reconstruction via front to back prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 531–540, 2020.

[37] Qingnan Zhou and Alec Jacobson. Thingi10k: A dataset of 10,000 3d-printing models. *arXiv preprint arXiv:1605.04797*, 2016.