Learning Profit Maximizing Diamond Cutting

Nikhil Agarwal
nikhilagarwa@umass.edu

Nishtha Nayar
nnayar@umass.edu

Shantanu Agarwal
shantanuagar@umass.edu

Abstract

Diamond cutting is the practice of shaping a diamond from a rough stone into a faceted gem. Modern methods involve scanning the rough diamond to obtain a 3D object and using manual supervision to plan the cutting process. The latest advancements in 3D deep learning seem compatible to this task but there is no prior work for the same. Thus, we propose a novel challenge of using 3D deep learning methods to plan diamond cutting from 3D scans of raw diamonds. As a first step towards achieving this task, we collected a simple novel synthetic data of 200 raw diamonds that mimic the real world problem. We propose deep learning models adapted from MeshNet[5], PointNet[7] and U-Net [2]. The proposed methods use supervised learning to identify the placement of diamond that maximizes the size of the final diamond and are evaluated on the closeness to the ground truth using chamfer distance and mIoU.

1. Introduction

3D volumetric analysis is one of the prominent problems of 3D Deep Learning. Deep learning methods have been used for primitive tasks like segmentation, object detection and correspondence, and complex tasks such as dense representation of 3D objects, surface reconstruction and generation. The architecture of the model depends greatly on the representation of the 3D object. Some most commonly used input representations are multi-view images [13, 4], voxels[2], sparse data structures like oct-trees[14, 10], point clouds [15, 7, 8], and meshes [5]. U-Net [2] uses voxel based 3D convolutional neural network for the task of segmentation for medical images. MeshNet [5] proposes using mesh based 3D inputs for the task of object classification. PointNet [7] works on point cloud input to generate a permutation invariant global representation of the point cloud.

In this paper, we propose deep learning methods to predict the placement of a brilliant diamond from within a raw diamond extracted from a mine. Diamonds extracted from mines, called rough diamonds, need to be processed and cut before they can exhibit their true brilliance. This is required because of the following reasons: i) Rough diamonds can have impurities like crystals and cracks, called *inclusions*. ii) Diamond's brilliance is a result of the amount of total internal reflection which varies with the way it is cut. Presence of colored segments inside the diamonds further complicates the process since the value of a diamond greatly depends on it's color. Diamond cutting is the practice of shaping a diamond from a rough stone into a faceted gem. Early methods involved manual investigation of the rough diamond and cutting by hand. With the advances in technology, modern tools became increasingly adopted in this practice to guarantee incredible precision. One of the latest methods involve using a specialized machine to scan the rough diamond, identify inclusions, plan out possible ways of cutting, selecting optimal cut and using lasers to obtain the final polished diamond. We propose methods to automate the manual step of planning for brilliant diamond cutting using 3D deep learning.

Since this work is first of it's kind, there are no openly available 3D datasets involving raw diamonds. To solve this problem and encourage future work, we created a synthetic dataset of 3D mesh based objects of raw diamonds that mimics the structure of the real-world diamond ore. We use this synthetic data to solve the aforementioned task using deep learning methods. We experiment with multiple input representations and use appropriate architectures. Multi-view images and sparse data structures are not suitable for the task since multi-view images will fail to capture the inclusions inside a solid diamond and sparse data structures will be equivalent to voxels since the solid 3D rough diamond object is not sparse. Thus, we use point clouds, meshes and voxels as our input representations and use appropriate architectures for diamond cutting planning.

To summarize, our contribution through this paper is as follows:

- We propose a novel task of predicting the position of the largest brilliant diamond with minimum inclusions that can be cut from a given raw diamond
- We synthesize a simple novel 3D dataset that mimics the structure of raw diamonds (described in Section 2)
- We use 3D deep learning networks inspired from MeshNet, U-Net and PointNet for the proposed task

2. Data Synthesis

Modern deep learning methods require increasingly large datasets to train and an proportional amount of data to adequately test the model performance for real-world applications. Unfortunately, 3D datasets are notoriously hard to collect and small in size compared to datasets available for tasks in Natural Language Processing and Computer Vision. Moreover, there is no openly available dataset of 3D rough diamond scans to the best of our knowledge. Thus, we designed a geometry based method to collect a simplified version of the real-world dataset that can be synthetically created. To accomplish this task, we used Blender [3] which is a free and open-source 3D computer graphics software tool-set that can be leveraged for creating 3D models. We simplified the data creation process using the following assumptions and constraints:

- The raw diamond surface is represented by a icosphere
- The inclusions are represented by deformed cubes that reside within the icosphere
- The inclusions do not intersect with each other
- Only one diamond can be extracted
- There are multiple ways to cut a diamond but we assume only brilliant cut diamond will be extracted
- all objects are represented by a 3D triangular mesh

We synthesize the diamond objects dataset using the following algorithm:

- Create a icosphere and randomly scale it along the x,y and z axis. Perturb it's vertices with small random distances.
- Place a brilliant diamond mesh within the icosphere. It is randomly scaled and translated within the sphere, ensuring no intersection with the outer mesh.
- Fill the space between the diamond and the outer icosphere with enough inclusions such that it is not possible to place a bigger diamond without intersection with any of the inclusions
- The final 3D object is a triangular mesh based object (exported as .obj).

Two versions of this datasets are maintained. One in which the diamond is axis aligned with the Blender environment. The other is the augmented dataset in which for each data point, we randomly rotate the entire mesh around the center of the Blender environment. We report the final performance of proposed networks on these two datasets. In order to create the input for our 3D deep learning network, we remove the diamond from the 3D object and store its center location, Euler rotation values and size as target labels. To facilitate the training of a robust 3D deep learning network, we use a 70/15/15 split of these inputs and labels for train, validation, and test sets.

3. Related Work

To the best of our knowledge there is no existing prior work for the problem of predicting brilliant diamonds. We briefly discuss some prior work which we have modified for our proposed models. 3D deep learning methods can be classified based on their input representation:

Mesh Based Learning: Meshes are good at depicting 3D models for rendering purposes, yet, they are complex structures with evolved concepts of vertices, edges, faces, and associations among these structures. For a long time, the focus of the community was on volumetric based models (voxelized data) and point cloud. The major disadvantages of using these models is the loss of information. In volumetric/voxelized data, the information loss increases as we decrease the resolution. Further, a lot of empty space is wasted in voxelizing in the naive implementation. Point clouds by definition are sparse and do not directly indicate any associate among different points. This makes mesh models a very viable option in comparison which is what MeshNet [5] discusses.

MeshNet [5] is pioneering work in 3D Deep Learning as it proposed a novel way to use Mesh based 3D models for 3D shape representation. The network uses the faces, normals and corners present in the data and passes it along to its two modules: Structural Descriptor and Spatial descriptor. The structural descriptor module creates feature vectors that represent the 3D model semantics. The spatial descriptor module creates feature vectors which represent the position of different 3D model components in the space. These two types of feature vectors are then used together for the task of object classification in the original implementation of the paper.

Point-Cloud Based Learning: PointNet [7] takes a point cloud as input and returns a global representation of the collection of points. This representation is invariant to the ordering of the points as it is fed to a shared MLP and the output is the symmetric max pooling on the outputs of the MLP. There are several computational advantages of using PointNet [7] over other complex networks. Further, the applications of this network include object classification, part segmentation and semantic segmentation among others.

Voxel Based Learning: Voxel based approaches in 3D deep learning are analogous to computer vision using Convolutional Neural Networks on images. The method remains the same as where sliding blocks of filters are used to perform convolutions and are applied locally on the input. The network learns more semantic representations of the original input as we go deeper into the network. The U-Net [2] is one such 3D CNN network used in the medical model segmentation task which uses 3D Convolutions, max pooling, skip connections, and up-convolutions to create a segmented output with the same size as the original input. The down-sizing is used to get more context about the input

and the up-sizing and skip connections are used to utilize this context along with precise local information about the input. The usage of the network is not limited to medical segmentation and can be extended to many other 3D deep learning tasks.

4. Proposed Methods

We are proposing three methods inspired from MeshNet [5], PointNet [7], and U-Net [2, 11]. All of them were implemented using PyTorch [6] and the code has been provided with the paper.

4.1. RM-Net (Regression MeshNet)

A modified version of MeshNet [5] is used to take as input the corners, faces, neighbouring face indices, face normals, the type of object (impurity or not), and the actual data labels which the network needs to learn. Figure 1 illustrates the network.

This modified MeshNet takes as input a list of neighbouring faces, centers, normals, impurity labels (which indicates if the object is an impurity or not), and corners. This input is then processed in the same way as is done by MeshNet [5]. The final layer of the network is changed from a classification layer to a regression layer by changing the softmax function which is used to gather the probabilities of different classes to tanh and sigmoid functions which output the translation, Euler rotations, and scale of the predicted diamond.

input_neighbours = $n_f \times 3$
input_corners = $n_f \times 9$
input_centers = $n_{ce} \times 3$
input_normals = $n_{no} \times 3$

n_f is the number of faces
For each face we have 3 corners (9 numbers)
n_{ce} is the number of centers. $n_{ce} = n_f$
n_{no} is the number of normals. $n_{no} = n_f$

The output of this network is a seven dimensional vector. The first three dimensions depict the position of the center of the diamond. The next three dimensions depict the euler rotations of the diamond and the last dimension depict the size of the diamond.

$O_{0:2}$ = (x, y, z)
$O_{3:5} = (R_x, R_y, R_z)$
$O_6 = S$

The loss function(s) used for training and evaluation have been covered in section 5.

4.2. Artificial Neural Network

The neural network based approach gets its inspiration from the PointNet [7] architecture which is a common method for encoding a global vector from a point cloud. The idea is that we can predict the center location of the diamond, its Euler rotation angles and its scale as a global vector of the point cloud that we get from the impurities. The intuition behind this is that the position information of impurities gives human-enough information to predict the largest diamond which will not intersect with these impurities. We would like our neural network to learn feature vectors that can encode this spatial and structural information. Figure 4 gives us the details of this network.

input_vertices = $n_{inclusions} \times 24$

n_inclusions is the number of inclusions. For each inclusion we have 8 corners and thus 24 numbers.
Output and loss function(s) are same as that of the RM-Net.

4.3. U-Net

3D dense convolutions on voxel grid is the 3D successor of 2D convolutions which are known to perform very well on vision tasks. We use the 3D U-Net [2] architecture to estimate the position of a 3D brilliant diamond inside a raw diamond. Specifically, we voxelize the input mesh (using resolution of 32 and 64 for our implementation) and identify the voxel that contains the center of ground truth diamond as the target. Let r be the resolution of voxelization, then the input is represented by $\mathbf{D} \in \{0, 1, 2\}^{r \times r \times r \times 1}$, where 0 denotes empty space, 1 denotes raw diamond and 2 denotes inclusion. The target center, rotation and scale is denoted by $(\mathbf{c}, \mathbf{r}; s)$ where $\mathbf{c}$ is the coordinate of the diamond center, $\mathbf{r}$ is the euler rotation along x, y and z axis. Let $\mathbf{v_c}$ denote the index of the voxel containing the target diamond center. The network architecture is described in Figure 3. Let n_c denote the number of output channels. We use the setting with n_c where the channels represent the following:

- Center Confidence: The first channel of the output for each voxel v_j (j is an index in $\{0, ..., r \times r \times r\}$ is $p(v_j)$ which represents the probability of that voxel containing the brilliant diamond center. Note that p_c is a softmax over all the output voxels since we have to choose only one diamond and want to maximize probability for only one voxel. Thus, this converts the task of identifying the center into a classification problem.
- Rotation: The next three channels predict the euler rotation for each of the 3 unit axis. It is denoted by $\hat{\mathbf{r}}(v_j); \hat{s}(v_j)]$.
- Scale: The last channel is the predicted scale of the brilliant diamond denoted by $\hat{s}(v_j)$.

During inference, we consider the diamond center to be

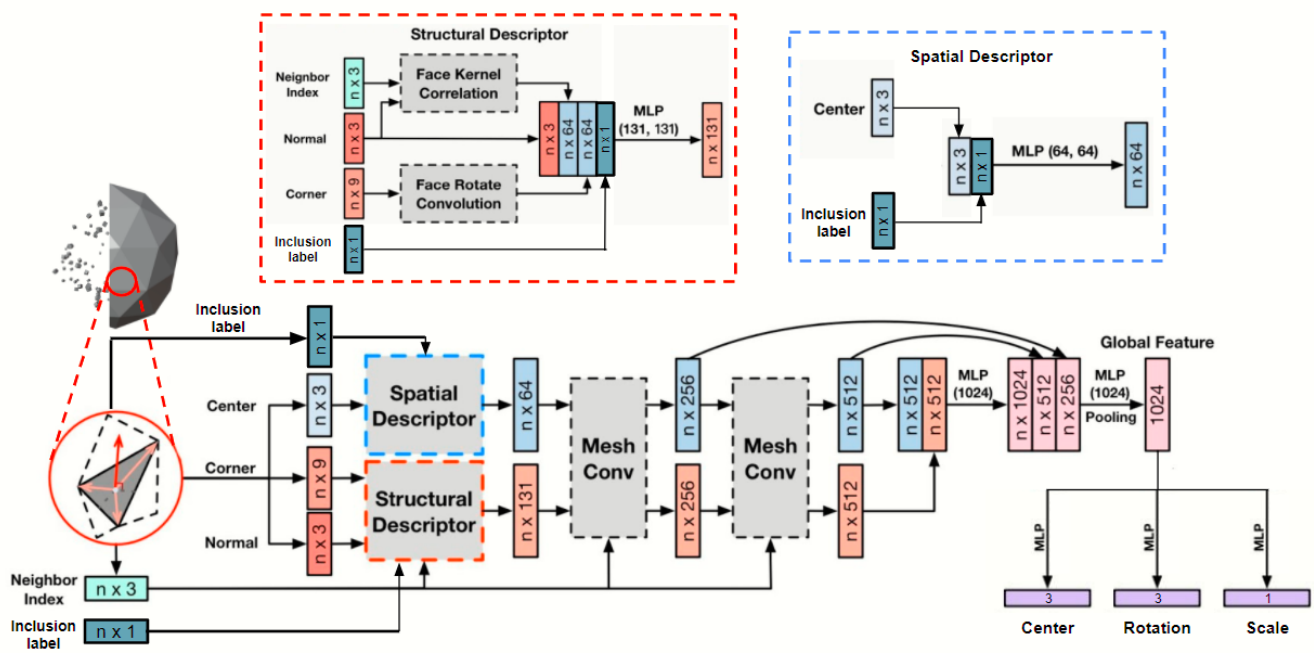

Figure 1. The architecture of MeshNet. The input is a list of neighbouring faces, corners, inclusion labels, centers and normals which are fed into the spatial and structural descriptors to generate initial spatial and structural features. The features are then aggregated with neighboring information in the mesh convolution blocks labeled as "Mesh Conv", and fed into a pooling function to output the global feature used for further tasks. Multi-layer-perceptron is labeled as "MLP".

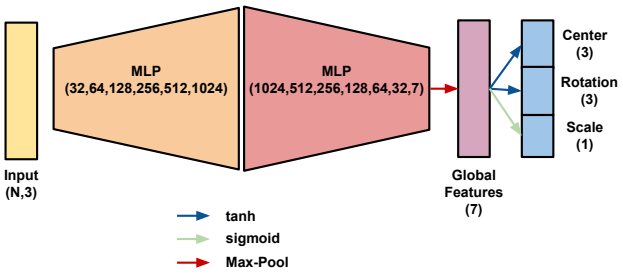

Figure 2. This neural network takes as input the vertices of the impurities. The size and number of hidden layer units have been chosen experimentally by checking the trend in training and validation losses. The final output is a vector of length seven. We use tanh and sigmoids in the regression layer to constraint the ranges as per that of the targets. Notice the use of max pooling layer for the prediction which is what makes this PointNet-inspired neural net permutation invariant.

the geometric center of the voxel with the maximum output probability $p(v_j)$. During training, we maximize $p(v_c)$ using a binary cross entropy loss , where v_c is the ground truth voxel. Because of the softmax probability, maximizing at v_c also reduces probability for other voxels that don't contain the center. To train the model for rotation and scale, we use regression using L1 distance of $[\hat{\mathbf{r}}(v_j); \hat{s}(v_c)]$ from the corresponding ground truth values.

The overall loss for a single training example is defined as weighted sum of the negative log likelihood loss and the L1 distace as follows:

$$L(\mathbf{c}, \mathbf{r}, c|\mathbf{D}) = -\alpha \log p(v_c) + (1-\alpha)(|\mathbf{t_r} - \hat{\mathbf{r}}| + |t_s - \hat{s}|)$$

Worth mentioning is that we also tried bounding box prediction method described in YOLO [9] but it turned out to be very computationally expensive for a dense 3D voxel grid. Lastly, we evaluate both models using the metrics described in Section 5.

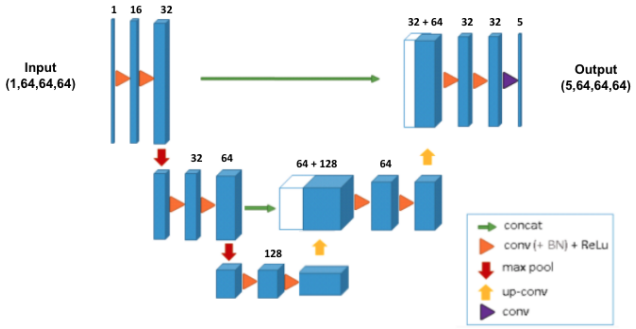

Figure 3. 3D U-Net architecture. The model takes in a single channel input whose values denote empty space, raw diamond of inclusion. The output has n_c =5 channels for our use case.

5. Loss function

To train a network effectively, we must choose a loss function such that minimizing it will lead to predictions that match our target labels. We have many options for this purpose including mean L1 loss, MSE loss, and mean intersection over union. Both are very good measures of how close a prediction is to the target labels. However, IoU drops to zero once there is no intersection between the prediction and the target. This makes it non-differentiable at these points as it can not differentiate the loss between two predictions which are both non intersecting with the diamond, which is not what we want. We want the diamond among these two which is closer to the target to have a lower loss. Although differentiable implementations of IoU are available, they are too complex for unaligned bounding boxes compared to MSE loss which is differentiable for all cases. MSE loss can be defined as:

$$MSE = \frac{\sum_{i}^{1:n} \|t_i - t_i^{'}\|^2}{n}$$

where t_i is the location vector of the i_{th} point of the target diamond and $t_i^{'}$ is the location vector of the i_{th} point of the predicted diamond. The number of fixed points chosen is n.

During the evaluation phase of the model, we use the model which gave the lowest validation loss and we calculate all the three losses i.e mean L1 loss, MSE loss, and mIoU. We choose 153 fixed points on the target diamond and calculate the mean L1 and MSE loss for the corresponding points on the predicted diamond. For the purpose of calculating the mIoU, we consider the tight bounding boxes for both the predictions and targets. L1 loss is as follows:

$$L1 = \frac{\sum_{i}^{1:n} \|t_i - t_i^{'}\|}{n}$$

6. Experimental Results

In this section we compare the different metrics i.e. mean L1 loss, MSE loss and the mIoU for the different networks that we have experimented with. We present these results as a table here.

The 'Augmented' column of the table indicates whether the model was trained and tested on augmented data or unaugmented data. For augmentation, we create four new data points from each datapoint by introducing a random rotation to the entire data point i.e. to all the objects in that data point. Since the number of parameters in the ANN are around a quarter of those in RM-Net, we see an inferior performance on all the metrics. The performance degrades on all the parameters as we choose more data to train and test on. One possible explanation for this is that the model is over-fitting when we are not providing it with augmented data to train on. When provided with rotation augmented data, the model starts showing a higher bias since it is underfitting as it is no longer complex enough to predict the underlying distribution of the target data. Note that, mIoU is only an approximate metric for augmented data because it does not account for bounding box rotation. One more thing to note here is that the networks might be having a hard time in understanding the underlying distribution since the target datapoints are generated using random distributions.

Augmented	Model	L1 loss	MSE loss	mIoU	#params
No	ANN	0.24	0.08	0.24	1.2M
	RM-Net	**0.18**	**0.05**	**0.40**	5.6M
	3D U-Net	0.28	0.14	0.32	1M
Yes	ANN	0.42	0.28	0.23	1.2M
	RM-Net	**0.35**	**0.19**	0.26	5.6M
	3D U-Net	0.37	0.22	**0.35**	1M

Table 1. Experiment results for the three models described in the paper. The 3D U-Net performance is reported for Model B that uses input voxelized with a resolution of 64. This table shows that MeshNet performs best for all metrics.

7. Conclusions and Future Work

In this paper, we have presented a new direction for using 3D Deep Learning to place a diamond inside a rough diamond with impurities. For this purpose, we present a novel synthetic dataset to encourage future work and propose three models to solve the problem. We observed that the 3D graphics and animation tool-sets like Blender provide a convenient way to generate a dataset that can be used to train 3D deep learning models. However, we need to mimic the read-world data distribution as closely as possible so that model performance can transfer equally well to real world.

The results obtained are promising and highlight the efficacy of modern 3D deep learning tools for the task. But they are still not good enough to replace manual human selection of the diamond especially because higher accuracy is a must in the process of cutting a diamond. As a future work, we should get 3D scans of real-world rough diamond and use them for training our 3D deep learning networks. We can also explore modern 3D deep learning methods used for medical images [1, 12] since they operate on a similar dense 3D input. Moreover, we plan to extend this work by using unsupervised learning to predict the diamond position. This would require including overlap with inclusions in the loss function and can be expected to improve model's generalization. Lastly, the task proposed in this paper can be generalised to placing any generic object inside a larger object with impurities.

References

[1] Christoph Angermann et al. “Projection-Based 2.5D U-net Architecture for Fast Volumetric Segmentation”. In: *CoRR* abs/1902.00347 (2019). arXiv: 1902.00347. URL: http://arxiv.org/abs/1902.00347.

[2] Özgün Çiçek et al. *3D U-Net: Learning Dense Volumetric Segmentation from Sparse Annotation*. 2016. arXiv: 1606.06650 [cs.CV].

[3] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation. Stichting Blender Foundation, Amsterdam, 2018. URL: http://www.blender.org.

[4] Angela Dai and Matthias Nießner. *3DMV: Joint 3D-Multi-View Prediction for 3D Semantic Scene Segmentation*. 2018. arXiv: 1803.10409 [cs.CV].

[5] Yutong Feng et al. *MeshNet: Mesh Neural Network for 3D Shape Representation*. 2018. arXiv: 1811.11424 [cs.CV].

[6] Adam Paszke et al. “PyTorch: An Imperative Style, High-Performance Deep Learning Library”. In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach et al. Curran Associates, Inc., 2019, pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[7] Charles R. Qi et al. *PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation*. 2017. arXiv: 1612.00593 [cs.CV].

[8] Charles R. Qi et al. *PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space*. 2017. arXiv: 1706.02413 [cs.CV].

[9] Joseph Redmon et al. “You Only Look Once: Unified, Real-Time Object Detection”. In: *CoRR* abs/1506.02640 (2015). arXiv: 1506.02640. URL: http://arxiv.org/abs/1506.02640.

[10] Gernot Riegler, Ali Osman Ulusoy, and Andreas Geiger. *OctNet: Learning Deep 3D Representations at High Resolutions*. 2017. arXiv: 1611.05009 [cs.CV].

[11] O. Ronneberger, P.Fischer, and T. Brox. “U-Net: Convolutional Networks for Biomedical Image Segmentation”. In: *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*. Vol. 9351. LNCS. (available on arXiv:1505.04597 [cs.CV]). Springer, 2015, pp. 234–241. URL: http://lmb.informatik.uni-freiburg.de/Publications/2015/RFB15a.

[12] Satya P. Singh et al. “3D Deep Learning on Medical Images: A Review”. In: *Sensors* 20.18 (2020). ISSN: 1424-8220. DOI: 10.3390/s20185097. URL: https://www.mdpi.com/1424-8220/20/18/5097.

[13] Hang Su et al. “Multi-view convolutional neural networks for 3d shape recognition”. In: *Proc. ICCV*. 2015.

[14] Peng-Shuai Wang et al. “O-CNN: Octree-based Convolutional Neural Networks for 3D Shape Analysis”. In: *ACM Transactions on Graphics (SIGGRAPH)* 36.4 (2017).

[15] Yue Wang et al. *Dynamic Graph CNN for Learning on Point Clouds*. 2019. arXiv: 1801.07829 [cs.CV].

A. Appendix

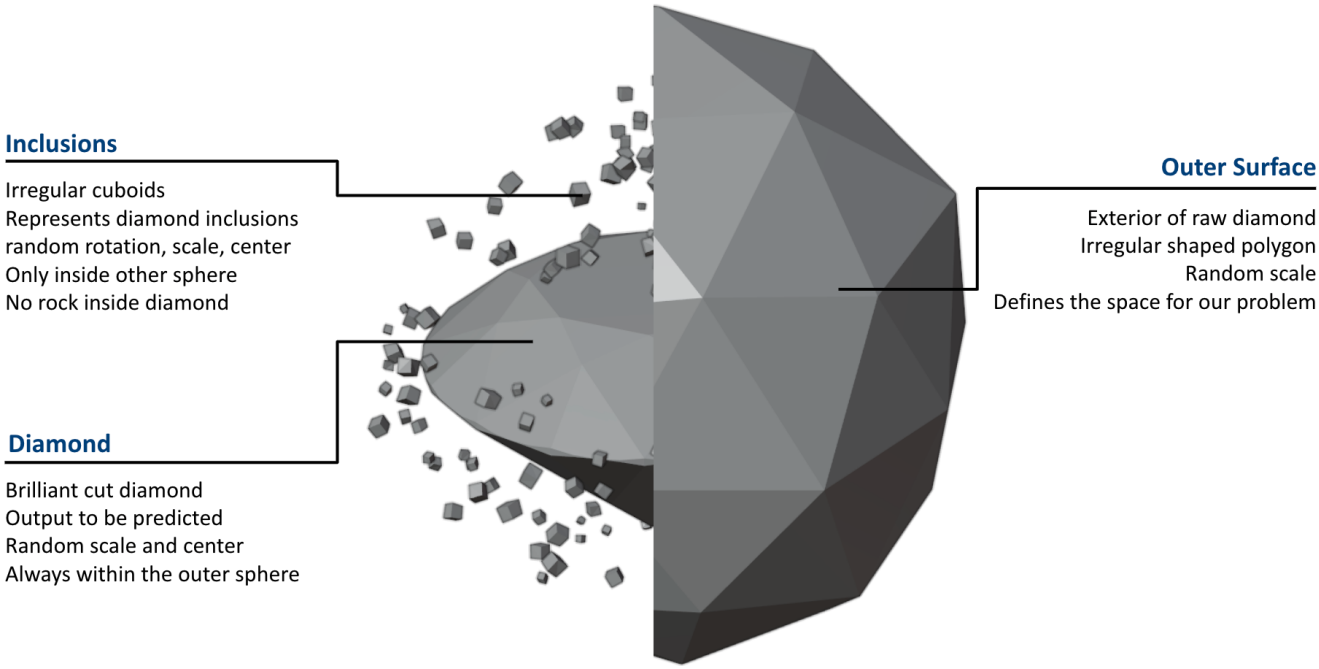

Figure 4. This figure explains an example datapoint synthesised using Blender.