Proceedings of the ASME 2019 14th International Manufacturing Science and Engineering Conference
MSEC2019
June 10-14, 2019, Erie, PA, USA

MSEC2019-3050

Geometric Accuracy Prediction for Additive Manufacturing Through Machine Learning of Triangular Mesh Data

Nathan Decker, Qiang Huang
Daniel J. Epstein Department of Industrial and Systems Engineering, University of Southern California,
Los Angeles, CA 90007 USA

ABSTRACT

While additive manufacturing has seen tremendous growth in recent years, a number of challenges remain, including the presence of substantial geometric differences between a three dimensional (3D) printed part, and the shape that was intended. There are a number of approaches for addressing this issue, including statistical models that seek to account for errors caused by the geometry of the object being printed. Currently, these models are largely unable to account for errors generated in freeform 3D shapes. This paper proposes a new approach using machine learning with a set of predictors based on the geometric properties of the triangular mesh file used for printing. A direct advantage of this method is the simplicity with which it can describe important properties of a 3D shape and allow for predictive modeling of dimensional inaccuracies for complex parts. To evaluate the efficacy of this approach, a sample dataset of 3D printed objects and their corresponding deviations was generated. This dataset was used to train a random forest machine learning model and generate predictions of deviation for a new object. These predicted deviations were found to compare favorably to the actual deviations, demonstrating the potential of this approach for applications in error prediction and compensation.

Keywords: Freeform Shape, Shape Representation, Shape Deviation Modeling, Random Forest Modeling

1. INTRODUCTION

Over the past several decades, there have been many great advances in the field of additive manufacturing (AM). New materials and processes are entering the market at a rapidly increasing pace, allowing for a growing number of exciting applications in fields such as aerospace and medicine. Despite these advances, many challenges remain, including the presence of substantial geometric differences between a 3D printed part, and the shape that was intended. These manufacturing defects can make it impossible to meet certain tolerances, and often lead to waste of both time and material. Approaches in the literature for addressing these errors have generally fallen into a number of broad categories including physics-based modeling [1–4], process parameter calibration [5,6], online monitoring and prescriptive product design adjustment [7,8].

Prescriptive product design adjustment (PPDA) seeks to predict and compensate for dimensional errors caused in 3D printing processes. PPDA is particularly useful since it can be used in conjunction with other approaches such as process parameter calibration in order to account for multiple sources of error. This approach has been used to predict both errors caused by a 3D printer and errors caused due to the shape of an object being printed. Tong, et al. [7], for example, utilized a process-driven error prediction model in order to predict errors for 3D printed parts. Once errors were predicted, the CAD file describing each shape to be printed was altered in order to compensate for the expected errors, yielding parts with less error.

One challenge of this approach is addressing print errors that are specifically due to the shape that is being printed. Recent work has demonstrated the efficacy of a statistical modeling approach for prescriptively compensating for errors in cylinders produced using stereolithography [8]. This approach was later extended to predicting errors generated by freeform shapes [9]. Currently, models used in prescriptive product design adjustment for shape-driven error are largely limited to predicting inaccuracies for planar objects that are extended into 3D, i.e. their shape does not vary with respect to height. This is done by predicting error for a single cross section, and then applying the results to the entire object. Because almost all functional parts are complex shapes that vary with respect to the z-axis, it is necessary to extend these approaches to three-dimensional space. Due to issues including shape complexity and interlayer interactions, this work has proven challenging. Thankfully, recent advances in machine learning provide a path forward.

A key step of shape accuracy control is to describe shape deviation for freeform 3D shapes. A related area of importance for the work that will be discussed here is the use of triangular mesh files to represent 3D shapes, a common practice in AM. This method of describing 3D shapes uses a series of triangles to form a surface that represents a shape. The most common file format for triangular mesh representations, the .stl format, stores the vertices for each triangle, as well as each triangle's corresponding normal vector. This can been seen in Figure 1. Because the triangles are flat, they cannot perfectly describe curved surfaces, but instead must approximate them using a large number of small triangles. For curved surfaces, a higher density of triangles is needed to reproduce a shape with fidelity, while for flat surfaces, a far lower density is necessary. This leads to inconsistencies in the density of a triangular mesh from shape to shape.

The relationship between triangular mesh data and the accuracy of 3D prints has been previously studied. One area of research has focused on reducing inaccuracies due to the imperfections in how triangular mesh files represent 3D surfaces. Navangul, et al. developed an algorithm to reduce these errors through modification of the triangular mesh file [10]. Triangular mesh data has also been used to predict shape-dependent print errors. Chowdhury and Anand used an artificial neural network based approach in conjunction with finite element analysis and triangular mesh data to predict and compensate for thermal expansion induced errors in 3D printed parts [11]. Chowdhury, et al. then extended this approach to include optimization of part build orientation using a model with orientation-based variables relevant to accuracy prediction [12]. Moroni, et al. demonstrated a method of identifying cylindrical voids in a part's shape using a triangular mesh file, and then predicting the dimensional error of the voids based on their geometric properties [13].

By using the triangular mesh file to derive information about local areas on the surface of the shape, it is feasible to predict shape-driven errors from multiple sources on any complex shape. This paper therefore puts forth a new method for predicting geometric print errors for any shape using triangular mesh data. By generating a set of predictor variables for vertices across the surface of the object based on the geometric properties of the triangular mesh file at each vertex, the problem of describing a complex shape can be simplified as any geometry can be roughly described using a finite number of points. This approach not only simplifies the task of describing the properties a complex shape that are most relevant to error generation, but it allows for predictive modeling of dimensional inaccuracies for complex parts. This is because the predictor variables describing the shape can be used to train a machine learning model to predict deviation at each corresponding vertex on the surface of the shape.

Due to the unique requirements of this approach, a novel method for measuring geometric deviations across the surface of a 3D printed object will also be proposed. This method utilizes subject matter knowledge about the Fused Deposition Modeling (FDM) 3D printing process utilized in this investigation, and can produce accurate and repeatable measurements of print error. This procedure can also be employed for other methods of additive manufacturing.

Finally an experiment to validate this approach using a number of 3D printed benchmarking objects will be presented. This experiment used a set of three 3D printed objects to train a machine learning model. The model was then used to make predictions for a fourth shape that was treated as a validation dataset. The predicted deviations from the model compared favorably to the actual deviations, demonstrating the potential of this approach for applications in error prediction and compensation.

2. MODELING METHODOLOGY

In order to capture the desired geometric properties of a surface using the triangular mesh file, a series of eight predictor variables $\mathbf{x}$ corresponding to each property under consideration was constructed. These predictor variables are then calculated for each vertex on the triangular mesh, allowing for a direct correspondence between the predictor variables and deviation y at each vertex.

This paper will focus on generating predictors for three broad areas of phenomena that have been shown to affect print accuracy. The first area of significance is the physical position of a vertex in a printbed. While this is generally a critical element of process-driven models, it will also be included here due to its potential interaction with shape-driven error. The next area of significance is the orientation and curvature of a surface. This will be used due to the association of properties such as surface slope with errors like the stairstep effect [14]. Further, surface curvature in the x-y plane can influence how material is deposited in the FDM process. The final area of significance will be linear thermal expansion effects. A number of predictor variables were specifically devised to quantify the effects of each of these broad areas of interest. These will now be discussed in further detail.

In order for this approach to produce an unbiased model, it is necessary that the triangular mesh be uniformly dense across the surface of the shape and have triangles of consistent size. A number of software packages can be used to remesh surfaces to meet these requirements. One such example is Meshmixer from AUTODESK. The specifics of each predictor variable in the training set are explained below.

2.1 Printbed Positioning Predictors

For the nth vertex v_n, the first three predictor variables ($x_{n,1}$, $x_{n,2}$, $x_{n,3}$) used in this model correspond to the x, y, and z coordinates of each vertex. These predictors seek to capture errors related to the actual position of the printed object within the printbed. Because care was taken in the setup of the validation experiment, these values from the .stl file directly correlated to each vertex's position within the 3D printer's printbed. This source of error is due to the positioning of the extruder as it traverses the build envelope, and has been identified as being significant in previous work [15]. One implication of this is that the same object printed in different orientations will have different predictor sets.

2.2 Surface Orientation and Curvature Predictors

The next four predictor variables are derived from the set of normal vectors corresponding to V_n triangular faces adjacent to a given vertex n. Each normal vector $\boldsymbol{S_i} = (1 \quad \phi_i \quad \vartheta_i)$, $i = 1, 2, \ldots V_n$, is expressed in spherical notation with radius 1, an elevation angle, and an azimuth angle. The predictor variables are calculated as follows, and illustrated in Figure 1.

$$x_{n,4} = median(\{\vartheta_i, i = 1, \dots V_n\}) \quad (1)$$

$$x_{n,5} = max(\{\vartheta_i, i = 1, \dots V_n\}) - min(\{\vartheta_i, i = 1, \dots V_n\}) \quad (2)$$

$$x_{n,6} = median(\{\varphi_i, i = 1, \dots V_n\}) \quad (3)$$

$$x_{n,7} = max(\{\varphi_i, i = 1, \dots V_n\}) - min(\{\varphi_i, i = 1, \dots V_n\}) \quad (4)$$

The first of these predictor variables is the median value of the azimuth angles in the set. This can be interpreted as the direction that the geometric features are facing. This is a useful term for predicting shape-related print errors. The second of these variables is the range in azimuth angles (i.e. max azimuth – min azimuth). This can be interpreted as an indicator of the curvature of the surface. Changes in curvature can affect how material is deposited from the extruder, and thus error. The third of these variables is the median value of the elevation angles in the set. This can be interpreted as the slope of the geometric features. This is of particular interest due to the correlation between slope and the prevalence of the stairstep effect, among other errors [14]. This variable is also useful for detecting overhangs, which can be difficult to print accurately. Finally, the fourth of these variables is the range in elevation angles (i.e. max elevation – min elevation). This can be interpreted as the degree to which the slope changes over the surface described by the triangular faces. This has relevance to both shape-dependent errors and the stairstep effect.

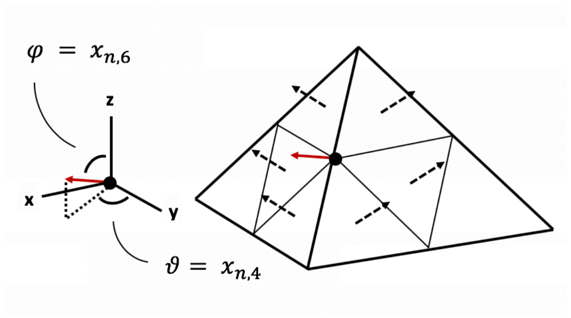

Figure 1. Normal Vector for Each Face Surrounding a Vertex with Median Vector

2.3 Linear Expansion Predictor

The final predictor variable proposed here is the distance from each vertex to the z-axis placed at the center of each shape (located at 0,0):

$$x_{n,8} = \sqrt{{x_{n,1}}^2 + {x_{n,2}}^2} \quad (5)$$

This is shown in Figure 2. This is of significance due to linear thermal expansion effects of the printed materials. Objects of larger size expand by a greater absolute distance due to scaling, necessitating a proxy for point distance from the rough center of expansion to be accounted for. Points on the surface that are a greater distance from what can be considered the center of the object will experience a greater degree of displacement.

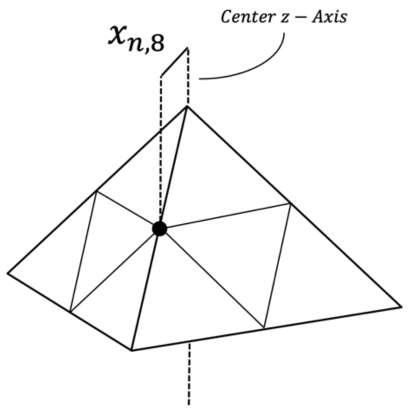

Figure 2. Distance Between Vertex and Central z-Axis

Given a .stl file, the set of each of these predictor variables can be quickly calculated for each vertex. While these predictors don't perfectly describe shape, they can give a good idea of the relevant geometric factors that can influence the accuracy of a 3D print. The relative efficacy of each of these predictor variables will be briefly evaluated in Section 4.4.

3. DEVIATION MEASUREMENT

A rigorous procedure for measurement, registration and analysis is necessary in order to produce deviation values that correspond to each set of predictor variables generated according to the method described above.

3.1 Scan Point Cloud Generation

This procedure begins by producing a dense point cloud of measurements of the surface of a 3D printed object. In the validation experiment described below, each object was scanned using a ROMER Absolute Arm with an attached laser scanner manufactured by Hexagon Manufacturing Intelligence. According to the manufacturer, this scanner has an accuracy of 80 µm. The objects were each scanned with dozens of passes from different angles so as to create scans with a far greater density than the density of vertices in the triangular mesh files. The result of this process was 3D scan point clouds describing the top surfaces of each of the objects. These scans also included points describing the table surface that the objects were resting on. These can be referred to as 'table points'. For the work that will follow in the sections below, the set of vertices from the triangular mesh file is defined as P, the set of points on the surface of the shape from the laser scan is defined as Q, and the set of 'table points' from the laser scan is defined as Q_{table}. This is illustrated in Figure 3.

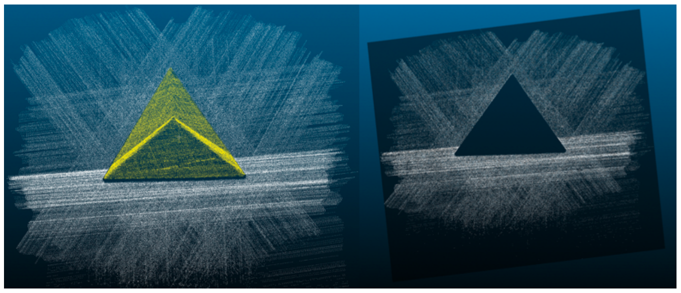

Figure 3. 3D Scan of Object (Gold) and Table Points (White) with Plane Fitted (Black)

3.2 Point Cloud Registration

In order to make comparisons between the measured scan point clouds and the surface described by their corresponding design triangular mesh, it is necessary to perform registration to align them. For simplicity, the 3D scan point clouds are moved into alignment with the original design triangular mesh file, which remains static. Because the two shapes being registered against each other are not exactly the same, this is a nonrigid registration problem, meaning that obtaining a quality alignment is substantially more difficult than the rigid case [16].

In order to address this difficulty and ensure that the results of the registration process are repeatable and independent of the shape being printed, a procedure was developed utilizing a subject matter knowledge. This pre-assumption is that the bottom face of an object is printed with perfect accuracy, while the remaining errors can be attributed to the other faces. This corresponds well with the physical realities of FDM printing, since plastic is applied to an essentially flat printbed surface. This assumption can be expressed more formally as follows. For a plane $f(x,y,z) = \beta_1 x + \beta_2 y + \beta_3 z + \beta_0 = 0$ fit to the 3D scanned table points in Q_{table}, the vector normal to this plane given by $\mathbf{N}_{Scan} = \nabla f = [\beta_1 \quad \beta_2 \quad \beta_3]^T$ should be equal to the vector normal to the surface of the actual shape as described by the CAD file or triangular mesh data.

In the case of this investigation, the bottom surface of each benchmarking object was placed on the x-y plane, meaning that for the plane fit to the bottom of the CAD file, $\beta_1 = 0$, $\beta_2 = 0$, $\beta_3 = -1$, and $\beta_0 = 0$, the normal vector for the bottom surface of the CAD file should be: $\mathbf{N}_{CAD} = \nabla f = [0 \quad 0 \quad -1]^T$.

Given that the 3D scan point cloud is transformed so that $\mathbf{N}_{Scan}$ and $\mathbf{N}_{CAD}$ are equal, the bottom planes of the scan point cloud and the triangular mesh vertices should be parallel. After an initial registration based on visual inspection, and the normal vectors are made equal, one further consideration is needed. The prior assumption further requires that there should be no distance between these two parallel planes. In the case of this investigation, because the bottom surface of each benchmarking object was placed on the x-y plane, β_0 of the plane f should be set to zero. This would require a translation along the z-axis.

One method for applying the implications of this assumption to the scan point cloud data is through the use of transformation matrices. Möller and Hughes provide a computationally inexpensive method for aligning two vectors that can be utilized for this purpose [17].

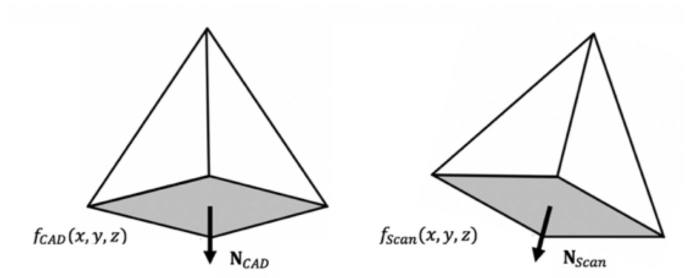

Figure 4. Illustration of Planes and Normal Vectors

Once alignment according to the prior assumption was conducted, further registration was performed using the iterative closest point algorithm described by Besl and McKay, among others [18]. The iterative closest point method is a popular algorithm for this application. This algorithm traditionally makes adjustments to the registration of the point clouds along all six possible degrees of freedom. In order to preserve the previous adjustments, only translation along the x and y axes, and rotation around the z-axis were allowed.

3.3 Shape Deviation Calculation

Once registration was completed, the distances between each vertex in the triangular mesh and the 3D scan point cloud were calculated. The algorithm used to calculate cloud to cloud distances is the nearest neighbor distance. This gives the distance between a vertex on the triangular mesh and its nearest neighbor on the compared measurement point cloud. This distance is returned in the form of a vector $\mathbf{d}_n = (dist\_x_n, dist\_y_n, dist\_z_n)$ where dist_x is the distance in the x-direction, dist_y is distance in the y-direction, and dist_z is distance in the z-direction. In order to derive a final deviation value that can be treated as a response variable in the model, it is necessary to take the length of each of these vectors. Signs are assigned according to whether the deviation represents a dimension that is too large or too small. This results in a set of response variables y_1 through y_N.

For a training dataset containing multiple parts, data $\{(\mathbf{x}_n, y_n), n=1,2,..N\}$ is the ensemble of the total N vertices. Note that each vertex may have a different number of adjacent triangle faces. For the validation experiment conducted in Section 4, there are three triangular mesh files that correspond to three different shapes that are all included in the training dataset.

4. MODEL GENERATION

In order to learn and predict the deviation of triangular meshes for freeform shapes, it is necessary to develop a predictive model based on the training data. Because triangular mesh files often contain tens of thousands of vertices, the size of the datasets generated by this method can be cumbersome, posing a computation challenge for machine learning methods. One computationally efficient modeling approach that can be utilized in this situation is the random forest method. One way to quantify the computational efficiency of a machine learning algorithm is time complexity, which reflects the number of computations that must be performed in order to generate a model, and thus time. The random forest algorithm has a worst-case-scenario time complexity on the order of: $O(MK\tilde{N}^2 \log \tilde{N})$ where M is the number of trees in the random forest, K is the number of variables drawn at each node, and Ñ is the number of data points N multiplied by 0.632, since bootstrap samples draw 63.2% of data points on average [19]. As a point of comparison, an algorithm such as gaussian process regression has a worst-case-scenario time complexity on the order of: $O(N^3)$ [20]. For the training set utilized in the proof-of-concept experiment that will follow, this is roughly three orders of magnitude more complex.

4.1 Random Forest Method

Researchers have successfully applied machine learning to make accurate predictions in a wide range of applications related to manufacturing. One particularly popular algorithm for applications is the random forest approach, which has been applied to predicting surface roughness of parts produced with AM, fault diagnosis of bearings, and tool wear, to name just a few use cases [21–23]. The random forest algorithm is a means of supervised ensemble learning originally conceived by Leo Breiman [24]. It utilizes regression or classification trees, which are a method of machine learning that recursively segments a given dataset into increasingly small groups based on predictor variables, allowing it to produce a response value given a new set of predictors [25]. The resulting structure of this segmentation process resembles the roots of a tree, and is shown in Figure 5. The random forest algorithm constructs an ensemble, or forest, of these trees, each trained on a subset of the overall dataset [23]. This overall process is explained in further detail below, and illustrated in Figure 6.

The goal of a regression tree is to generate a set of rules that efficiently segment the given training set using predictor variables in a way that generates accurate predictions of a response variable. This process begins with a single node, and randomly chooses a set of predictor variables to be used in dividing the dataset. Given P total predictor variables, it is generally recommended that the number of predictor variables sampled for each node be set to P/3 in the case of regression and √P in the case of classification [26]. Using this subset of predictor variables, the algorithm seeks to split the data at the node in a manner that minimizes the sum of the squares of error for each response label y_i:

$$SSE = \frac{1}{N_{node}} \sum_{i=1}^{N_{node}} (y_i - \bar{y})^2 \tag{6}$$

$$where\ \bar{y} = \frac{1}{N_{node}} \sum_{i=1}^{N_{node}} y_i \tag{7}$$

This process is then repeated for each resulting node until a predetermined condition is met. Two common conditions include a predetermined minimum number of data observations at a node, or a maximum tree depth [21]. Once the stopping condition is met, each of the terminal nodes is labeled with the average value of the responses for the observations contained by that node. New predictions are generated using a set of predictor variable values to navigate down the tree until arriving at a terminal node that corresponds to the predicted response value.

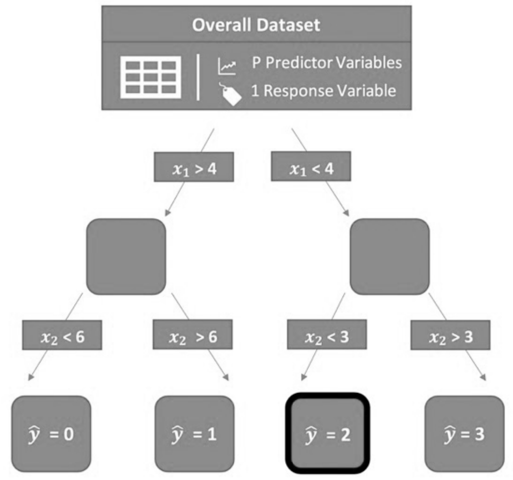

Figure 5. Single Regression Tree

The random forest algorithm begins by generating subsets, or 'bootstrap samples' from the overall dataset. These bootstrap samples are drawn randomly from the overall dataset with replacement, allowing for some data to be shared between samples [21]. A regression tree is then trained for each bootstrap sample.

In order to make predictions using a generated forest, the predictor variables are used to generate individual predictions from each tree. The average of this set of predictions is then given as the overall output of the ensemble.

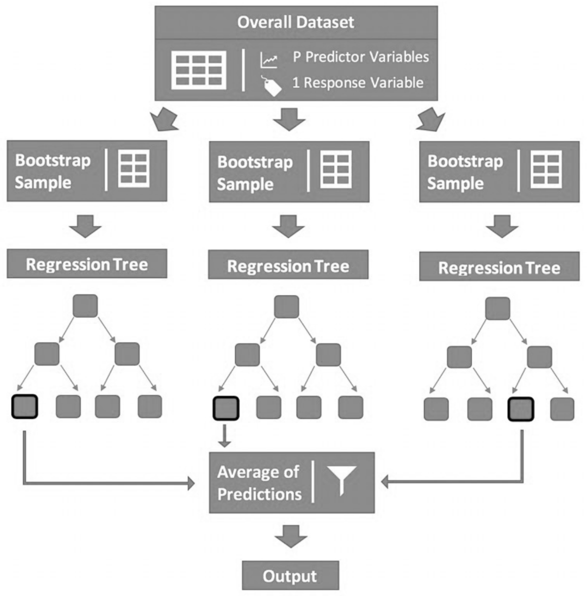

Figure 6. Ensemble of Trees Using Random Forest

4.2 Predictor Importance Assessment

In order to gain an understanding of the relative importance of the predictor variables used in this model, the out-of-bag permuted predictor change in error for each predictor variable was calculated during the validation experiment. This is a method for quantifying how significant a predictor variable is to the accuracy of the overall predictions produced by the random forest [27]. For each tree in the random forest, the training set data that wasn't used to train the tree, also referred to as the out-of-bag observations $\{(\mathbf{x}_n, y_n), n=1,2,..N_{out-of-bag}\}$ is used to generate a set of predictions of the response variable $\{\hat{y}_n, n = 1,2,..N_{out-of-bag}\}$. After this, the mean squared error of this set of predictions

$$MSE_{out-of-bag} = \frac{\sum_{n=1}^{N_{out-of-bag}}(\hat{y}_n - y_n)^2}{N_{out-of-bag}} \tag{8}$$

is then calculated. This is also referred to as out-of-bag error.

Then, for the first predictor variable $x_{:,1}$, each of its values in the dataset are permuted so as to randomize the values of that predictor variable's input. A new set of predictions is generated using this data, and the mean squared error (MSE) of these predictions is calculated. The change in prediction error is defined as the difference between the original and changed MSE values:

$$\Delta\text{Error}_{x_1,tree_1} = MSE_{x_1,tree_1} - MSE_{out-of-bag,tree_1} \tag{9}$$

$$\Delta\text{Error}_{x_1} = \frac{1}{T}\sum_{i=1}^{T}(\Delta\text{Error}_{x_1,tree_i}) \tag{10}$$

A large value of ΔError_{x_1} indicates that this is a significant predictor variable, since randomizing its input causes the predictions of the regression tree to become much worse.

This process is repeated for each of the predictor variables in the dataset, and for each of the regression trees in the random forest. The change in MSE is calculated for each predictor variable and averaged over all of the trees in the random forest. Each of these results is divided by the standard deviation of the ΔError values for the entire ensemble, and is outputted as the final significance value $S(x_n)$ for each predictor variable:

$$S(x_n) = \frac{\Delta\text{Error}_{x_n}}{std(\Delta\text{Error}_{ensemble})} \tag{11}$$

5. VALIDATION EXPERIMENT

In order to test the efficacy of the proposed method, it was necessary to develop a number of 3D printed shapes. Then, a dataset of predictor variables and deviation values was generated according to the procedure outlined in Sections 1 and 2. In order to limit the complexity of the dataset for this validation experiment, it was decided that each of the printed shapes would be nonconvex. The dataset was constructed according to the following procedure.

5.1 Test Object Design

Four benchmarking objects were designed in commercial CAD (computer-aided design) software and exported to a 3D printer as .stl files. These included a half-ovoid, a half-teardrop, a triangular pyramid, and a half-deltoidal icositetrahedron. These objects are chosen to represent different geometries, including varying curved and flat faces, as well as edges of various angles. In order to account for variability in mesh density from shape to shape, each shape was remeshed using Meshmixer, a piece of software from Autodesk, in order to produce a consistently dense mesh. This was done so as to accurately preserve the original shapes while also ensuring a uniform distribution of vertices across each surface. The edge length of each triangle in the mesh of each object is approximately half a millimeter.

5.2 Test Object Printing

Following this remeshing process, the benchmarking objects were printed on a MakerBot Replicator FDM 3D printer using MakerBot brand Polylactic Acid filament. In order to avoid any effects caused by varying patterns of infill for different shapes, each object was printed entirely solid (i.e. 100% infill). Care was taken to ensure that the point defined as the origin in the triangular mesh file for each object was printed at the exact center of the printbed. This ensured that the positions of each vertex in the triangular mesh directly corresponded to the positions of the printed objects within the printer's build envelope. This is a relationship that is useful when producing predictor variables, and will be discussed later. The printed benchmarking objects are shown in Figure 7.

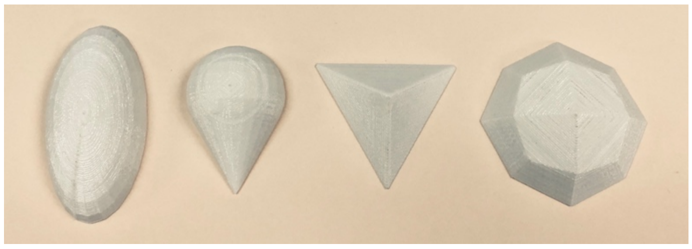

Figure 7. 3D Printed Benchmarking Objects

5.3 Deviation Computation Results

The deviation values for each of the 3D printed shapes were calculated according to the procedure in Section 2. These deviation values are shown in Figure 8, which is a heatmap of deviation values across the surface of each shape. The color at each point indicates the extent of the deviations across the surface. Red points correspond to parts of the shape that are too large, while blue points correspond to parts of the shape that are too small. It is valuable to point out that the half-deltoidal icositetrahedron exhibits a pattern that is somewhat unlike the other three shapes. This is likely a result of its larger size and dissimilar shape.

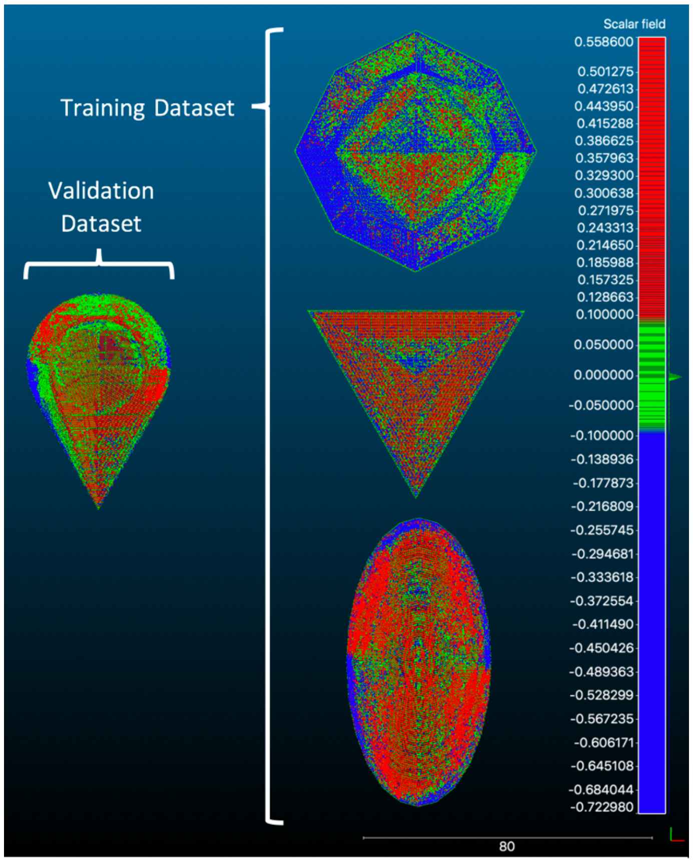

Figure 8. Deviation Values Across the Surface of Each Shape

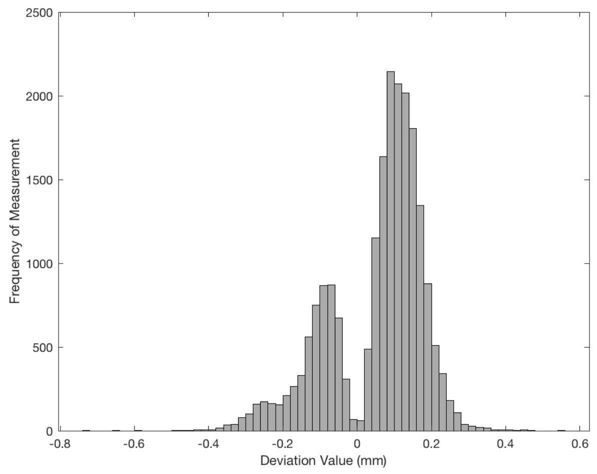

Figure 9. Histogram of Deviation Values for Teardrop Shape

In order to better understand the distribution of deviation magnitudes, a histogram showing the frequencies with which various magnitudes of deviation values occur is provided in Figure 9. This histogram is specifically for the teardrop shape, which is withheld as the validation dataset. None of the values from the bottom surface of the shape are included, as the deviations are assumed to be zero. It can be seen that most deviations are within 0.3 mm of the desired dimension. Further, it is interesting to note that this distribution is bimodal, with peaks in both the negative and positive range.

5.4 Model Training Results

Using the training dataset, an ensemble of regression trees was trained using the random forest method and MATLAB's Statistics and Machine Learning toolbox. The model was trained only on a training set consisting of the half-ovoid, half-deltoidal icositetrahedron and triangular pyramid, while data on the half-teardrop was withheld for validation of the created model. The minimum number of observations at each node was set to 8, while the number of trees in the ensemble was set to 30. This ensemble size was chosen due to the fact that experimental results indicated that further increases in ensemble size for this dataset yield increasingly small gains in out of bag error, as shown in Figure 10.

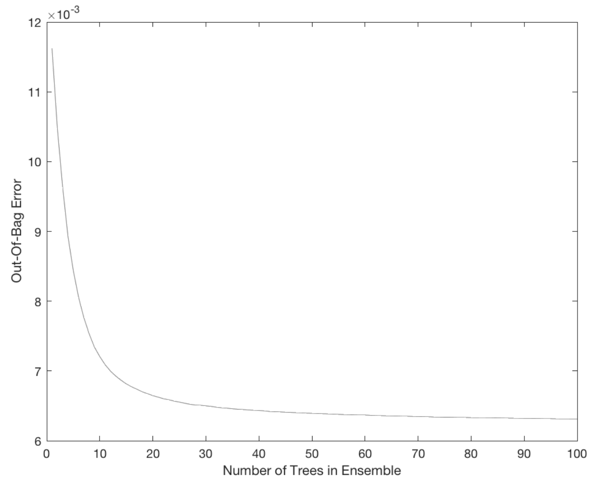

Figure 10. Out-Of-Bag Error vs. Number of Trees in Ensemble for Training Set

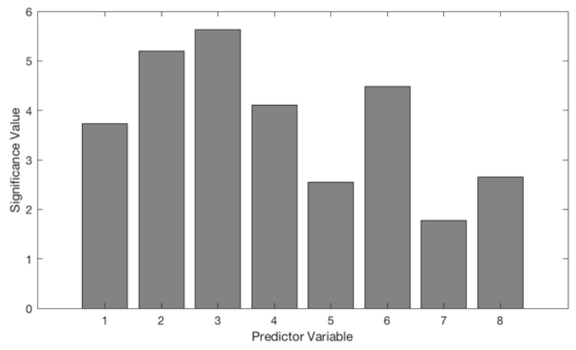

Figure 11. Significance Values of Each Predictor Variable

Thanks to the simplicity of the random forest algorithm, the total training time was approximately one minute, while predictions can be generated at a speed of roughly 110,000 predictions per second. The relative significance of each predictor variable was calculated according the procedure described in Section 3.2. These values are shown in Figure 11. These results suggest that each of the predictor variables contributes to the overall accuracy of the model. The least significant predictor variable in the model according to this metric is x_7, which corresponds to the range in elevation vectors. This makes intuitive sense, as there are relatively few jagged edges on the shapes included in the given dataset, meaning that these values will remain somewhat similar across most vertices.

5.5 Model Prediction Results

Using the validation data (teardrop shape predictor variables) that was withheld from the training set, a new set of predictions was generated using the random forest model. These predictions for deviation across the surface of the shape were then graphed alongside the actual deviation values for the teardrop shape, allowing for comparison. These two sets of data are illustrated in Figure 12. Coloring is applied using the same method found in Figure 7. Further, the percentage of deviation predictions within specific ranges from the actual deviation values are shown in Table 1. These percentages do not include prediction values for the bottom face, which were set to zero.

Table 1. Percentage of Deviation Predictions for Validation Dataset Within Given Intervals of the Actual Measured Value

Range (mm)	Percentage
±0.05	38.3%
±0.10	66.7%
±0.15	83.7%

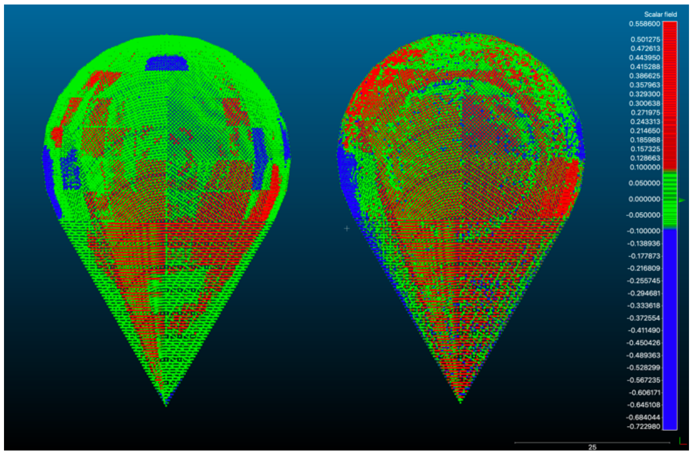

Figure 12. Predicted Deviation Values (left) vs. Actual Deviation Values (right) for Withheld Validation Dataset

It can be seen from these results that this method is capable of producing reasonably accurate predictions for a previously unseen shape from a small training set of just three similar shapes. Further, even though the half-deltoidal icositetrahedron has a remarkably different pattern of deviations than the other three shapes, the random forest model is able to discern the fact that this pattern is not applicable to the teardrop shape. This is a mark in favor of the efficacy of the model.

The predictions in Figure 12 might be useful for an operator of a 3D printer seeking to determine whether a specific 3D printed shape will be within a prespecified tolerance before beginning the print. This procedure might also be of use when determining the best orientation with which to print an object in order to maximize accuracy. Figure 12 also demonstrates that there is room for improving the accuracy of the model. This would likely include expansion of the initial training set, and refinement of the initial predictor variables. Further, the noise in the measurement of deviation values that can be seen in Figures 8 and 12 contributes to the error since the prediction values tend to remain relatively constant over a given area. Other methods of spatial measurement might be explored.

6. CONCLUSION AND FUTURE WORK

In conclusion, a method to model shape deviation in additive manufacturing using a set of predictors based on the geometric properties of a triangular mesh file was presented. Further, a method of deviation calculated was specifically developed for compatibility with the above modeling approach. Finally, a validation experiment was conducted in which a sample dataset of 3D printed objects and their corresponding deviations was generated. This dataset was used to train a random forest machine learning model, which was then used to generate predictions of deviation for a new object. These predicted deviations were found to compare favorably to the actual deviations, demonstrating the potential of this approach.

Future work might expand the complexity of the dataset used for training this machine learning model. Each of these shapes is symmetric about at least one plane, and all shapes are nonconvex. This offers some amount of similarity from shape to shape. Future datasets might be modeled after functional parts and include features such as voids. Future work might also focus on the use of this method for producing a prescriptive compensation policy for reducing errors by modifying the inputted triangular mesh file.

ACKNOWLEDGMENTS

This research was supported by the National Science Foundation under grant CMMI-1544917 and by a graduate research fellowship from the Rose Hills Foundation.

REFERENCES

[1] Hussein, A., Hao, L., Yan, C., and Everson, R., 2013, "Finite Element Simulation of the Temperature and Stress Fields in Single Layers Built Without-Support in Selective Laser Melting," Mater. Des., 52, pp. 638–647.
[2] Pal, D., Patil, N., Zeng, K., and Stucker, B., 2014, "An Integrated Approach to Additive Manufacturing Simulations Using Physics Based, Coupled Multiscale

Process Modeling," J. Manuf. Sci. Eng., 136(6), pp. 061022-1-061022-16.

[3] Steuben, J. C., Iliopoulos, A. P., and Michopoulos, J. G., 2016, "On Multiphysics Discrete Element Modeling of Powder-Based Additive Manufacturing Processes," *Proceedings of the ASME 2016 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, DETC2016–59634: pp. V01AT02A032. Charlotte, North Carolina, USA , August 21-24, 2016.

[4] Cattenone, A., Morganti, S., Alaimo, G., and Auricchio, F., 2018, "Finite Element Analysis of Additive Manufacturing Based on Fused Deposition Modeling (FDM): Distortion Prediction and Comparison with Experimental Data.," J. Manuf. Sci. Eng., 141(1), pp. 011010-1-011010-17.

[5] Lanzotti, A., Martorelli, M., and Staiano, G., 2015, "Understanding Process Parameter Effects of RepRap Open-Source Three-Dimensional Printers Through a Design of Experiments Approach," J. Manuf. Sci. Eng., 137(1), pp. 011017-1-011017-7.

[6] Mohamed, O. A., Masood, S. H., and Bhowmik, J. L., 2015, "Optimization of Fused Deposition Modeling Process Parameters: A Review of Current Research and Future Prospects," Adv. Manuf., 3(1), pp. 42–53.

[7] Tong, K., Joshi, S., and Lehtihet, E. A., 2008, "Error Compensation for Fused Deposition Modeling (FDM) Machine by Correcting Slice Files," Rapid Prototyp. J., 14(1), pp. 4–14.

[8] Huang, Q., Zhang, J., Sabbaghi, A., and Dasgupta, T., 2015, "Optimal Offline Compensation of Shape Shrinkage for Three-Dimensional Printing Processes," IIE Trans. (Institute Ind. Eng.), 47(5), pp. 431–441.

[9] Luan, H., and Huang, Q., 2015, "Predictive Modeling of In-Plane Geometric Deviation for 3D Printed Freeform Products," IEEE Int. Conf. Autom. Sci. Eng., pp. 912–917. Gothenburg, Sweden, August 24-28, 2015.

[10] Navangul, G., Paul, R., and Anand, S., 2013, "Error Minimization in Layered Manufacturing Parts by Stereolithography File Modification Using a Vertex Translation Algorithm," J. Manuf. Sci. Eng., 135(3), pp. 031006-1-031006-13.

[11] Chowdhury, S., and Anand, S., 2016, "Artificial Neural Network Based Geometric Compensation for Thermal Deformation in Additive Manufacturing Processes," *Proceedings of the ASME MSEC,* MSEC2016–8784: pp. V003T08A006. Blacksburg, Virginia, USA, June 27 - July 1, 2016.

[12] Chowdhury, S., Mhapsekar, K., and Anand, S., 2018, "Part Build Orientation Optimization and Neural Network-Based Geometry Compensation for Additive Manufacturing Process," J. Manuf. Sci. Eng., 140(3), pp. 031009-1-031009-15.

[13] Moroni, G., Syam, W. P., and Petró, S., 2014, "Towards Early Estimation of Part Accuracy in Additive Manufacturing," Procedia CIRP, 21, pp. 300–305.

[14] Strano, G., Hao, L., Everson, R. M., and Evans, K. E., 2013, "Surface Roughness Analysis, Modelling and Prediction in Selective Laser Melting," J. Mater. Process. Technol., 213(4), pp. 589–597.

[15] Wang, A., Song, S., Huang, Q., and Tsung, F., 2017, "In-Plane Shape-Deviation Modeling and Compensation for Fused Deposition Modeling Processes," IEEE Trans. Autom. Sci. Eng., 14(2), pp. 968–976.

[16] Tam, G. K. L., Cheng, Z. Q., Lai, Y. K., Langbein, F. C., Liu, Y., Marshall, D., Martin, R. R., Sun, X. F., and Rosin, P. L., 2013, "Registration of 3d Point Clouds and Meshes: A Survey from Rigid to Nonrigid," IEEE Trans. Vis. Comput. Graph., 19(7), pp. 1199–1217.

[17] Möller, T., and Hughes, J. F., 1999, "Efficiently Building a Matrix to Rotate One Vector to Another," J. Graph. Tools, 4(4), pp. 1–4.

[18] Besl, P. J., and McKay, N. D., 1992, "A Method for Registration of 3-D Shapes," IEEE Trans. Pattern Anal. Mach. Intell., 14(2), pp. 239–256.

[19] Gilles Louppe, "Understanding Random Forests: From Theory to Practice," PhD Thesis. University of Liège, Liège, Belgium, 2014.

[20] Williams, C. K., and Rasmussen, C. E., 1996, "Gaussian Processes for Regression," Adv. Neural Inf. Process. Syst., pp. 514–520.

[21] Wu, D., Wei, Y., and Terpenny, J., 2018, "Surface Roughness Prediction in Additive Manufacturing Using Machine Learning," *Proceedings of the ASME MSEC*, MSEC2018–6501: pp. V003T02A018. College Station, TX, USA, June 18-22, 2018.

[22] Tian, J., Ai, Y., Zhao, M., Fei, C., and Zhang, F., 2018, "Fault Diagnosis Method for Inter-Shaft Bearings Based on Information Exergy and Random Forest," *Proceedings of the ASME Turbo Expo*, GT2018–76101: pp. V006T05A017. Oslo, Norway, June 11-15, 2018.

[23] Wu, D., Jennings, C., Terpenny, J., Gao, R., and Kumara, S., 2017, "Data-Driven Prognostics Using Random Forests: Prediction of Tool Wear," *Proceedings of the ASME MSEC*, MSEC2017–2679: pp. V003T04A048. Los Angeles, California, USA, June 4–8, 2017.

[24] Breiman, L., 2001, "Random Forests," Mach. Learn., 45(1), pp. 5–32.

[25] Geurts, P., Irrthum, A., and Wehenkel, L., 2009, "Supervised Learning with Decision Tree-Based Methods in Computational and Systems Biology," Mol. Biosyst., 5(12), pp. 1593–1605.

[26] Hastie, T., Tibshirani, R., and Friedman, J., 2009, *The Elements of Statistical Learning*, Springer New York, New York, NY.

[27] Mathworks Inc., 2018, "Statistics and Machine Learning Toolbox: User's Guide (R2018b)" [Online]. Available: https://www.mathworks.com/help/stats/treebagger-class.html. [Accessed: 17-Nov-2018].