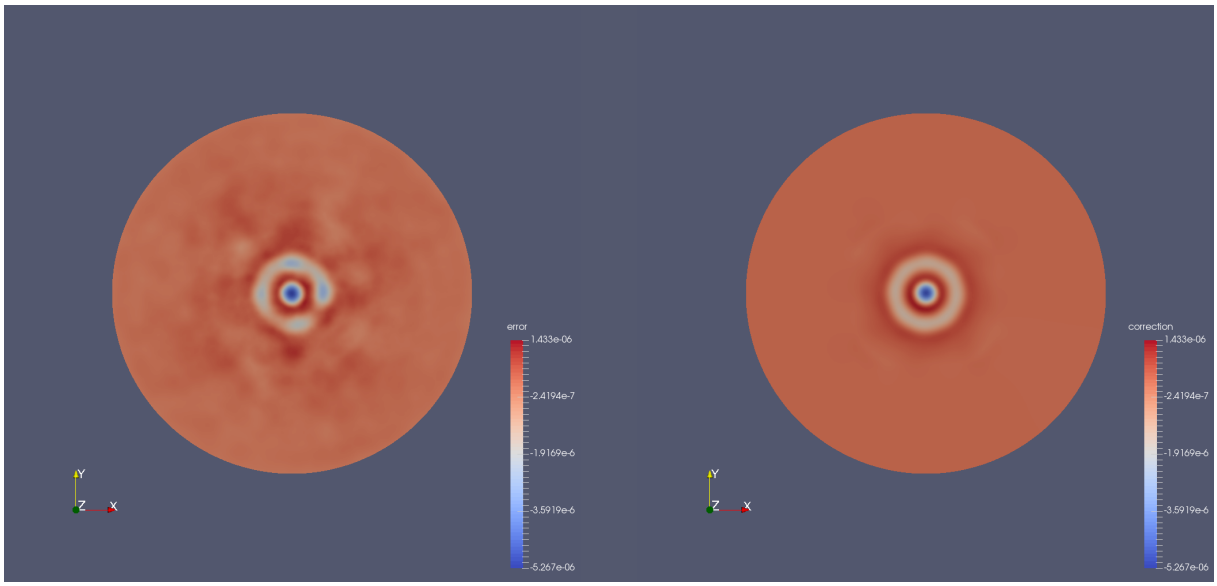
CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG




# Machine Learning Based Error Prediction for Spray Painting Applications

Master's Thesis in Complex Adaptive Systems

Paul Lange

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY AND UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2016

Master's Thesis

# Machine Learning Based Error Prediction for Spray Painting Applications

Paul Lange


CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
*Division of Computing Science*
Chalmers University of Technology
and University of Gothenburg
Gothenburg, Sweden 2016

Machine Learning Based Error Prediction
for Spray Painting Applications
Paul Lange



Examiner: Graham Kemp
Computer Science and Engineering, Chalmers University
Supervisor: Niklas Karlsson
Fraunhofer-Chalmers Centre for Industrial Mathematics
Supervisor: Peter Damaschke
Computer Science and Engineering, Chalmers University

Master's Thesis
Department of Computer Science and Engineering
Division of Computing Science
Chalmers University of Technology and University of Gothenburg

Cover: Comparison of the measured projection error and the predicted projection error for a flat disc painting scenario.

Gothenburg, Sweden 2016

Machine Learning Based Error Prediction
for Spray Painting Applications

Paul Lange
Master's Thesis in Complex Adaptive Systems
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

ABSTRACT

Physics-based simulation tools for spray painting exist but are not fast enough to be useful for automatic optimization of the spray painting process.

The results of spray painting depend primarily on the process parameters, the path of the paint applicator and the geometry of the target object. These factors affect the air flow and the electrostatic field and are hard to incorporate in approximate simulation tools.

This thesis proposes a novel approach based on combining fast, approximate simulations with machine learning based error correction. The proposed approach is to create a height profile of the target geometry from the local coordinate system of the paint applicator. This height field captures most parameters that affect the simulation error and can be used as an input for machine learning regression algorithms. These algorithms are then trained to estimate the painting error. The training is performed with a set of samples from common painting scenarios that are generated beforehand.

Creating a training set for dynamic simulations is time-consuming. Static simulations can sufficiently approximate dynamic simulations and are therefore used for training. This drastically improves the time to create training sets and reduces training time for the machine learning models.

Linear regression, tree-based regression models and support vector regression are compared on benchmarking problems and especially tree-based regression methods show promising prediction accuracy and are able to reduce the projection error more than 40% on real world benchmarks. Tree-based models are also the fastest algorithms among the compared regression models.

Finally, a way to integrate the proposed method into the simulation framework is presented. The results are investigated for different artificial and real world painting scenarios.

Keywords: spray painting, computer aided development, machine learning, regression, error prediction, error correction, simulation.

ACKNOWLEDGEMENTS

I would like to thank to my supervisor Niklas Karlsson for his expertise, help and support. During many fruitful discussions he gave valuable feedback and positively influenced this thesis.

I also thank my examiner professor Graham Kemp for his support and comments and my supervisor Professor Peter Damaschke for his valuable help in regard to machine learning and validation of machine learning models.

I also want to thank Stefan Jakobsson for his explanations of the spray painting simulation framework and valuable feedback.

Last but not least I thank the Fraunhofer-Chalmers Centre for the opportunity to work on this thesis and the support I received.

Paul Lange, Gothenburg, June 2016

# CONTENTS

# LIST OF FIGURES

# 1 INTRODUCTION

As development cycles shorten and products have to comply with ever higher standards, industry relies increasingly on virtual product development and therefore on effective simulation methods. Computer aided design helps to reduce the need for expensive prototyping, accelerates information gains and opens possibilities for automatic optimization methods which are not feasible within the traditional product development cycle.

The use of simulation tools is indispensable in certain disciplines like the mechanical design of components and structures, where mature software for finite element analysis exists and is commonly used during the development cycle [8]. In other areas like computational fluid dynamics adoption of simulation tools is in progress. In the area of surface treatment simulation methods are highly specialized on specific use-cases and not readily available. This stems from the fact that complex multi-physics processes have to be considered, which poses problems in simulation technology as well as in computational capacity.

Surface treatment is a term for industrial processes that alter the surface of a product in order to achieve certain properties. These properties may be functional (e.g. corrosion resistance, hardness or smoothness), aesthetic (e.g. color, glossiness or finish) or a combination of both. Common examples for surface treatment methods are spray painting in the automotive and furniture industries and thermal spraying in the aerospace industry.

Spray painting in the automotive industry has a big environmental impact. It is the process that consumes most water and chemicals, while producing most waste and pollution. Furthermore, roughly 40% of the energy in automotive manufacturing is consumed in paint shops [7].

To support sustainable production and an efficient product development the Fraunhofer-Chalmers Research Centre for Industrial Mathematics[1] (FCC) develops the state-of-the-art simulation software IPS Virtual Paint [12] for electrostatic rotary bell sprayer painting (see Chapter 2), a technique used by companies mainly within the automotive industry. The software is able to simulate around three seconds of painting time in one hour of real time currently. This is faster than other approaches but still requires significant time for large scale geometries.

1 http://www.fcc.chalmers.se/

In order to automatically generate and optimize paths for spray painting, significantly faster methods are necessary. One approach is to project pre-measured paint thickness distributions on the geometry (see Section 2.3). This method works in real-time but results in a considerable loss of accuracy.

To solve this problem a hybrid simulation model is presented in this thesis. It combines the fast simulation model with an error correction. With this approach it is possible to achieve high simulation speeds and high accuracy.

The error prediction model is based on machine learning algorithms (see Chapter 3) which stands in stark contrast to traditional approaches which employ analytical models of the error. Analytical modeling of the error is hard, as it usually depends on a multitude of factors whose interactions are hard to isolate. In contrast, machine learning is based on exploiting statistical patterns in a set of samples which is called the training set. The process of creating the training set and the necessary tools is a fundamental part of this thesis and presented in Chapter 5.

To use machine learning techniques in the context of spray painting, certain simplifications are necessary. Furthermore, a procedure to capture the relevant parameters of a painting scenario are required. The approach taken in this thesis is presented in Chapter 4.

The results of the hybrid simulation model are presented in Chapter 6. Here the error prediction performance of different machine learning algorithms is compared. Afterwards the predictions are used to correct the projection error in real geometries and the results are compared to the projection.

# 2 SIMULATION OF SPRAY PAINTING

In this chapter the spray painting process as well as different simulation approaches are presented. While not central for this thesis, the theoretical foundations are crucial to understand the complexity of the physics-based simulation. It also motivates the use of approximate simulation methods and clarifies the trade-offs of the different methods.

Spray painting in the automotive industry is commonly performed with the electrostatic rotary bell sprayer (ERBS) method. Here the liquid paint is injected in the center of a rotating bell. From there it travels to the bottom edge, where it is atomized and forms small paint droplets.

To drive the droplets towards the target geometry two techniques are used. First, air is exhausted around the bell to create an air flow towards the surface. Second, the paint particles are charged electrostatically by applying a potential difference between target and applicator. This potential difference usually lies in the order of 50 kV to 100 kV.

A picture of the ERBS method can be seen in Figure 2.1, where a flat surface is painted.

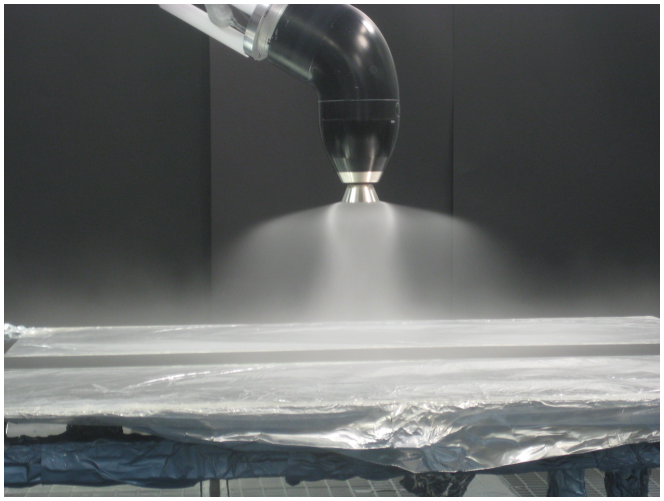


Figure 2.1: Rotary bell sprayer mounted on a robot arm while painting a flat surface.

The simulation tools developed at FCC focus on the electrostatic rotary bell sprayer (ERBS) method. By employing unique algorithms for two-way coupled simulations of air flows, electrostatic fields and charged paint droplets, simulation times are drastically reduced compared to other simulation frameworks.

The following sections give an overview over the physical modeling of the physics-based simulation and the projection simulation. Additionally, two methods for estimating paint thickness from droplet impacts are presented and an approximate paint simulation method introduced.

## 2.1 MULTI PHYSICS FRAMEWORK

The simulation of electrostatic rotary bell spray painting is characterized by "multiphase and free surface flows, multiphysics, multiscale phenomena and large, moving geometries" [12]. It therefore poses challenges in both mathematical formulation and computational complexity.

The two major factors on paint distribution are the air flow and the electrostatic field. The air flow is simulated by solving the incompressible Navier-Stokes-equations.

$$\nabla \cdot \bar{u} = 0 \tag{2.1}$$

$$\rho_f \frac{\partial \bar{u}}{\partial t} + \rho_f \bar{u} \cdot \nabla \bar{u} = -\nabla p + \mu \nabla^2 \bar{u} + \bar{s} \tag{2.2}$$

Here $\bar{u}$ is the fluid velocity, $\rho_f$ is the fluid density, $p$ is the pressure, $\mu$ is the dynamic viscosity and $\bar{s}$ is the droplet source term.

These equations are discretized on a Cartesian octree grid that is refined in areas of high interest, such as around geometries and the applicator, and coarsened in areas where less accuracy is needed (an illustration of the refinement around objects is shown in Figure 2.2). Objects are handled by using immersed boundary methods [10, 11].

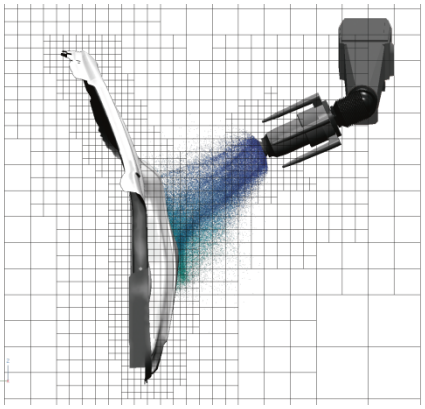


Figure 2.2: Illustration of paint applicator, target geometry, computational octree grid and the simulated droplets. The grid refinement can be seen around the geometry and the applicator.

The electrostatic solver is based on a similar octree-based grid, where the potential of target geometry and applicator are modeled

using immersed boundary conditions. The governing equation is Poisson's equation.

$$\nabla^2 \phi = -\frac{\rho}{\epsilon} \tag{2.3}$$

Here $\phi$ is the potential, $\rho$ is the droplet space-charge density, and $\epsilon$ is the permittivity of the fluid, which is air in this case.

The paint droplets are simulated as Lagrangian particles. The trajectory is governed by the Basset-Boussinesq-Oseen equation.

$$\rho_p \frac{d\bar{u}_p}{dt} = (\rho_p - \rho_f)\bar{g} - \bar{u}_r|\bar{u}_r|C_d \frac{\rho_f}{\rho_p}\frac{m_p}{2r_p} + \bar{E}q_p \tag{2.4}$$

Here $\bar{E}$ denotes the electric field, $\rho_p$ and $\rho_f$ denote the droplet and fluid density, $\bar{g}$ is the gravitational acceleration, $\bar{u}_p$ is the droplet velocity, $\bar{u}_r$ the relative droplet velocity, $m_p$ is the droplet mass, $r_p$ is the droplet radius, $q_p$ is the droplet charge and $C_d$ is the drag coefficient. For more information about the numerical algorithms used for solving these equations see [12].

## 2.2 PAINT THICKNESS ESTIMATION

The physics-based simulation calculates the trajectories of the paint droplets and the location of impact on the target geometry mesh.

Based on the droplet impacts the resulting paint thickness distribution on the geometry mesh has to be calculated. For that problem several solutions with varying complexity and accuracy exist. Two of them will be presented below.

### 2.2.1 *Binning*

The most common thickness estimation method is the binning approach. Here the painted surface is segmented into small bins. This segmentation is usually already available from computer aided design tools or can be generated efficiently.

Once the bins are available, the paint thickness is estimated as the sum of the volume of all impacts divided by the area of the bin.

$$T_b(b) = \frac{1}{A(b)} \sum_{i=1}^{N_b} V_i \tag{2.5}$$

Here $b$ denotes the index of the bin, $A(b)$ is the area of bin $b$, $V_i$ is the volume of a particle $i$, and $N_b$ is the number of particles that impacted in bin $b$. Subsequently the thickness estimation can be smoothed by interpolating values for the mesh nodes from the values calculated for the individual bins.

A problem with the binning method is its sensitivity on the size and shape of the bins. Too small bins result in high variance in the estimated thickness, while large bins result in details getting lost by

the bleeding effect. Therefore optimal bin sizes depend on the spray painting process parameters and thus are hard to achieve in general.

### 2.2.2 *Kernel Density Estimation*

A different method for paint thickness estimation is kernel density estimation [18]. The fundamental idea here is that the surface paint thickness can be interpreted as the density of the paint droplet impact volumes. Therefore, estimating the paint thickness equals to estimating the unknown Probability Density Function (PDF) from the given samples (i.e. the droplet impacts). This approach results in the following equation for the thickness estimate at position $\bar{x}$.

$$T_h(\bar{x}) = \frac{1}{h^2 S_K} \sum_{i=1}^{N} V_i K\left(\frac{\bar{x} - \bar{x_i}}{h}\right) \tag{2.6}$$

Here $\bar{x_i}$ denotes the position of impact $i$, $N$ is the total number of impacts, $K$ is a positive, normalized and symmetric kernel function and $S_K$ is a surface normalization constant for $K$.

$$S_K = 2\pi \int_0^\infty xK(x)dx \tag{2.7}$$

Common choices for kernel functions include the Epanchnikow and Biweight kernels. The parameter $h$ is called bandwidth and is crucial for good results. It can be chosen automatically using methods like cross-validation.

To reduce bleeding over sharp edges the surface is locally flattened around impacts, which leads to accuracy problems with complex geometries. In order to not underestimate electrostatically induced edge-effects, this approach can be extended to use multivariate bandwidths. For more information see [18].

## 2.3 PROJECTION METHOD

For testing or optimization of paint paths, faster simulation times are necessary. The projection method offers a solution by drastically simplifying the underlying model.

Technically, the projection method is a deposition model. Generally, there are two approaches towards deposition models – analytic and footprint based.

In the analytic approach paint thickness is approximated by analytical models that are then integrated in timely and spacial dimensions. These methods are not used in IPS Virtual Paint, as they are difficult to develop and often are highly specific to the application. See [6] for more information about analytic deposition models.

The footprint-based approach is based on a measurement of the directional mass flow of paint for the given process parameters in a

representative environment. This is usually done by spraying a flat surface for a fixed time from a fixed spray gun position. The resulting thickness is then measured and can be seen as the representation of the directional mass flow. The advantage of this method is that it can handle arbitrary flow conditions, even highly unsymmetrical ones.

In practice it is not efficient to measure the footprint for all possible process parameters in experiments. Therefore, in IPS Virtual Paint experiments are used for tweaking the parameters of the physics-based simulation, while the projection footprints are generated from the results of physics-based simulations.

Once a projection footprint has been generated, it can be applied to a target geometry by re-projecting it onto the target surface (see [9] for details). This projection is fast but also the reason for decreased accuracy.

As the footprint and the projection on the target geometry are static, the thickness from projection is calculated in every time step of the simulation. The result is then obtained via temporal integration.

# 3 MACHINE LEARNING

Machine learning is a scientific discipline that studies computer systems that automatically improve with experience without being explicitly programmed to do so.

Machine learning techniques are usually divided in two groups. In the supervised group the aim is to learn a mapping between given inputs x and outputs y. The set of pairs $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$ is called a training set with $N$ training examples.

The inputs $x_i$ are usually vectors of scalar values. These values represent properties of the problem at hand and are called features.

In unsupervised learning the aim is to find information or patterns in the data. Thus, only the inputs $x_i$ are given in the training set $\mathcal{D} = \{x_i\}_{i=1}^{N}$. As unsupervised learning will not be used in this thesis, no further information will be given here.

A different division can be made depending on the form of the output. If the output variables are of categorical form and thus part of a finite-size set, the problem is called a classification problem. Depending on the size of this set there are further distinctions. If there are only two possible classes one has a binary classification problem, whereas problems with more than two classes are called multi-class classification problems.

If the outputs $y_i$ are continuous the resulting problem is called a regression and represents the mapping from $x \in \mathbb{R}^N$ to $y \in \mathbb{R}$. Some specific regression methods can generate multiple outputs and thus map to $y \in \mathbb{R}^M$.

In the following sections the theory behind a subset of regression models will be presented.

## 3.1 LINEAR REGRESSION

Linear regression is a simple approach for predicting a continuous response based on one or more features. In case of a prediction based on one feature it is commonly called simple linear regression.

Simple linear regression assumes an approximately linear relationship between the feature $x$ and the response $y$, which can be formulated as follows.

$$y \approx \beta_0 + \beta_1 x \tag{3.1}$$

Here $\beta_0$ and $\beta_1$ are unknown parameters of the model, which have to be fitted to the training set. In the one-dimensional case $\beta_0$ and $\beta_1$ are also known as intercept and slope.

The fitted parameters $\hat{\beta}_0$ and $\hat{\beta}_1$ lead to a model that predicts $\hat{y}$ for a given $x$.

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x \tag{3.2}$$

The goal is now to minimize the distance between the prediction $\hat{y}$ and the known response $y$. A common approach is to minimize the least squared criterion. This is equal to minimizing the residual sum of squares (RSS).

$$\text{RSS} = \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 \tag{3.3}$$

This approach is visualized in Figure 3.1. The blue points represent the training set. The black line is the linear regression obtained by minimizing the RSS. The grey lines show the difference between predicted and actual value for each data point.

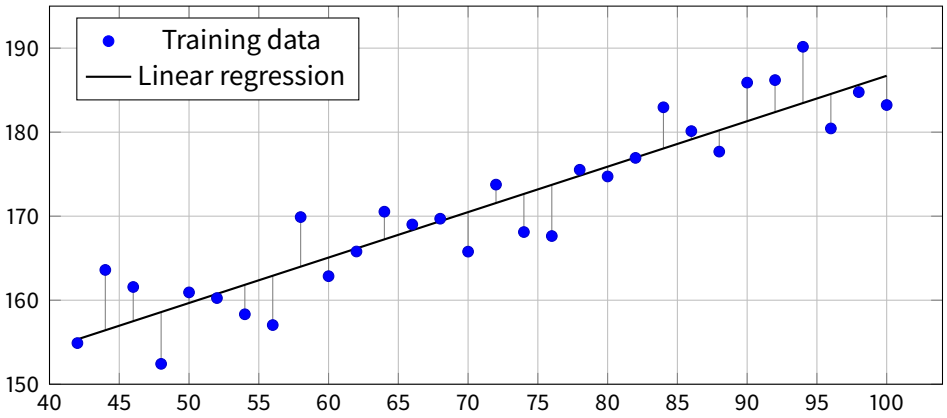



Figure 3.1: Training data with the fitted linear regression. The error for each sample is shown in grey.

In practice often more than one feature is available. Thus the basic model in (3.1) is extended to accommodate multiple features. Every feature gets a separate slope coefficient. This results in the following model.

$$y = \beta_0 x_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p \tag{3.4}$$

Here $\beta_0$ is the intercept term, so $x_0 = 1$. Equation (3.4) can also be written as $y = \langle \bar{\beta}, \bar{x} \rangle$, where $\langle \cdot, \cdot \rangle$ denotes the dot product.

It is important to note that the linear regression method is intended for a low-dimensional approach. Thus, the number of samples in the training set should be much greater than the number of features. In the high-dimensional setting, where the number of features is bigger than the number of samples, issues such as problems with overfitting arise.

## 3.2 DECISION TREES

A decision tree [5] is a machine learning method that stratifies the predictor space (the set of possible combinations of the features $x_i$) into several regions. These splitting rules can be visualized in form of a tree (see Figure 3.3), which is where the method takes its name from.

Tree-based methods are easy to understand and interpret, but usually their performance is not competitive with more advanced methods. However, ensemble methods based on tree-based methods exist and improve their performance. Two ensemble methods will be presented after a description of the general decision tree.

To use a regression tree it first has to be trained. This is done by dividing the predictor space into $J$ non-overlapping regions $R_1, \ldots, R_J$. Subsequently the return value can be calculated by computing the average of all training samples that fall into region $R_J$.

Prediction is now trivial. The algorithm only has to determine which region the given sample belongs to and can then return the previously calculated value.

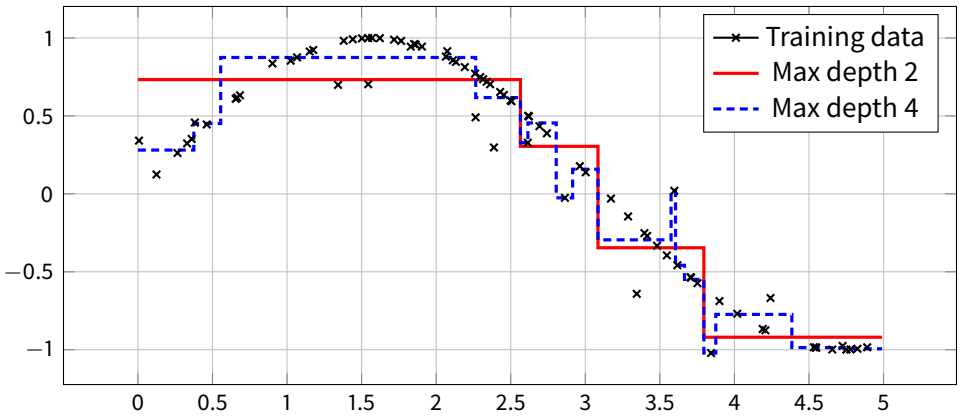



Figure 3.2: One-dimensional regression trees trained on noisy data with maximum depths of 2 and 4.

An example for simple regression trees with only one feature can be seen in Figure 3.2. Here two regression trees have been trained with noisy data from a sine wave. During training the regression trees were limited to different maximum tree depths. One can see that the tree with a maximum depth of two cannot represent the curve as it is to biased. In comparison, the tree with a maximum depth of four is already prone to over fitting as can be seen at some of the noisy outliers (e.g. $x \approx 3.6$ or $x \approx 3.8$). The internal structure of the tree with a maximum depth of two can be seen in Figure 3.3.

The problem for training the tree is how to stratify the predictor space. For simplicity and ease of interpretation usually high-

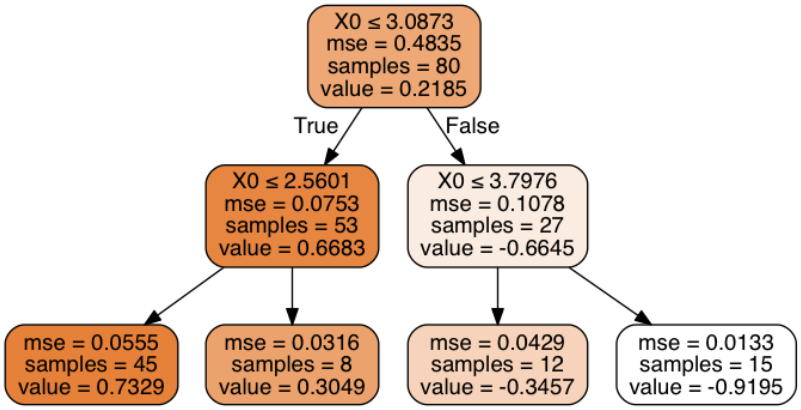



Figure 3.3: Internal structure of a regression tree with a depth of two.

dimensional rectangles, so called boxes, are chosen for division. In theory however, there is no limitation on the form of subdivision.

The goal is now to find a division of regions $R_1, R_2, \ldots, R_J$ that minimizes the residual sum of squares (RSS) which is given by the following term, where $\hat{y}_{R_j}$ is the average of all training samples in $R_j$.

$$\text{RSS} = \sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2 \tag{3.5}$$

It is computationally infeasible to consider all possible partitions of the predictor space into $J$ regions, especially as the optimal value of $J$ itself is unknown. Therefore a top-down, greedy approach is usually taken.

Top-down means that the algorithm starts with one region, that contains all samples. This regions is subsequently divided into smaller regions. Greedy means that the algorithms selects the best split available, without consideration for splits that might lead to better divisions in a later stage.

In every step the algorithm checks every feature $X_j$ and every possible split $s$. By each of these combinations a pair of half-planes

$$R_1(j,s) = \{X | X_j < s\} \quad \text{and} \quad R_2(j,s) = \{X | X_j \geq s\} \tag{3.6}$$

is created. The values of $j$ and $s$ are chosen to minimize the following equation.

$$\sum_{i:\, x_i \in R_1(j,s)} (y_i - \hat{y}_{R_1})^2 + \sum_{i:\, x_i \in R_2(j,s)} (y_i - \hat{y}_{R_2})^2 \tag{3.7}$$

After a split has been found, the same procedure is applied recursively to both of the resulting half-planes until some terminal condition for tree growth is hit. Examples for these conditions include maximum tree depth, minimal number of samples per leaf node or a maximum number of leaf nodes in the whole tree.

### 3.2.1 *Tree pruning*

An important question when training decision trees is the number of divisions $J$ and thereby the size of the tree. While a big tree with many divisions is usually at risk of overfitting and thus does not generalize well on new data, a small tree might not capture important structural information. However, in general it is impossible to know when to stop growing a tree due to the horizon effect, which makes it impossible to know if an additional node will have a big impact on prediction performance [15]. It is therefore a common strategy to grow trees until each node contains a small number of samples and then prune it in order to obtain a subtree.

The problem is that, depending on the size of the tree, a large number of subtrees may exist, which makes it infeasible to check all of them. Here cost complexity pruning offers a solution by just comparing the best subtrees as measured by the cost of the tree $C_\alpha$.

$$C_\alpha(T) = \sum_{m=1}^{|\tilde{T}|} \sum_{i:\ x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha|\tilde{T}| \tag{3.8}$$

It consists of an error term and a penalty term for the complexity of the tree $\alpha|\tilde{T}|$, where $|\tilde{T}|$ is the number of leaf nodes in tree $T$ and $\alpha$ is a free parameter that determines the penalty for complex trees. If $\alpha$ is zero, no penalty on tree complexity is applied. For growing $\alpha$, the algorithm favors smaller trees by allowing higher errors in exchange for smaller trees.

In the next step the best value of $\alpha$ is determined using either cross-validation or an additional validation set. Then the optimal subtree can be calculated. For a more detailed description see [5].

### 3.2.2 *Bagging*

Decision trees as described in section 3.2 are prone to overfitting and thus exhibit high variance. This stands in contrast to methods like linear regression, which show high bias. This means that if the data set is halved and decision trees and linear regressors trained on both halves, the resulting models will look similar for the linear regression, but might be quite different for the case of decision trees.

Bagging [3] is a general purpose ensemble algorithm which constructs several instances of an estimator from a given data set and combines their predictions into a final prediction. This provides a way to reduce variance, based on the foundation that averaging reduces variance. For example, the variance of the average $\bar{Z}$ of a set of $n$ observations $Z_1, Z_2, \ldots, Z_n$ with variance $\sigma^2$ is $\sigma^2/n$.

As bagging reduces variance, it works best with complex models like fully developed trees. For that reason no tree pruning is performed when using bagging in the context of decision trees.

As described above, bagging works by creating multiple instances of decision trees. Ideally, all trees would be trained on different data sets which are usually not available. Therefore one creates different data sets by drawing samples (with replacement) from the main data set. Subsequently a decision tree is trained for each of these data sets. The final prediction of the bagging classifier can then be calculated from the results $\hat{y}^i(x)$ given the following equation, where $B$ denotes the number of individuals.

$$\hat{y}_{\text{bagging}}(x) = \frac{1}{B} \sum_{b=1}^{B} \hat{y}^b(x) \tag{3.9}$$

### 3.2.3 *Random forests*

When building the ensemble of individual trees in bagging, at every split all features are considered. This usually leads to good features (in the sense of strong prediction abilities) being chosen early during tree growth. This in turn results in an ensemble of strongly correlated trees in which the variance does not get reduced as much as for uncorrelated trees.

Random forests [4] present a further improvement over bagged trees by decorrelating the individual trees in the ensemble. This is done by limiting every split to a random subset of $m$ of the total $p$ features. This results in $(p - m)/p$ of the splits not seeing the strong features and thus growing more diverse trees with less correlation, which results in higher accuracy. Furthermore, as fewer features have to be considered, random forests are generally faster than bagging methods.

In this regard bagging can be seen as a special case of Random Forests with $m = p$. For best results, typically $m \approx \sqrt{p}$ is chosen for random forests.

## 3.3 SUPPORT VECTOR REGRESSION

Support vector machines (SVMs) are a set of supervised learning methods that have been shown to perform well in a variety of settings. One particular advantage is that they work well in high-dimensional spaces.

SVMs are primary used for classification problems. To model regression problems with support vector machines a model called Support Vector Regression (SVR, [17]) is used. The idea here is to find a function $f(x)$ that includes the real targets $y_i$ in a deviation range of $\epsilon$ (see Figure 3.4 for an illustration) and is as flat as possible at the same time. Based on the parameter $\epsilon$ it as also known as $\epsilon$-SVR.

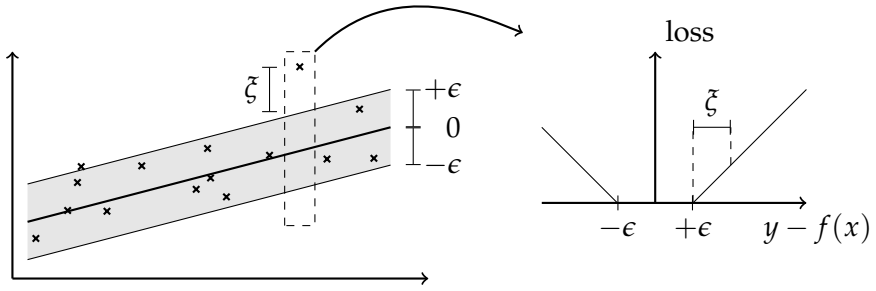



Figure 3.4: Illustration of a linear $\epsilon$-SVR and the $\epsilon$-insensitive loss function (illustrated with inspiration from [16]).

Assume a linear function $f$ of the following form, where $\langle \cdot, \cdot \rangle$ denotes the dot product.

$$f(x) = \langle w, x \rangle + b \tag{3.10}$$

The flatness requirement can be interpreted as searching for small $w$. One possible way for ensuring this is to minimize the norm $\|w\|$. This results in the following optimization problem, where the constraints ensure that all targets $y_i$ lie within the $\epsilon$-corridor.

$$\begin{aligned} &\text{minimize} && \frac{1}{2}\|w\|^2 \\ &\text{subject to} && \begin{cases} y_i - \langle w, x_i \rangle - b \leq \epsilon \\ \langle w, x_i \rangle + b - y_i \leq \epsilon \end{cases} \end{aligned} \tag{3.11}$$

The assumption of all $y_i$ lying within the $\epsilon$-corridor of $f(x)$ is not always true for reasonable values of $\epsilon$. Also one might want to allow certain small errors. In that case the model can be extended by introducing slack variables $\xi_i$ and $\xi_i^*$ to cope with the constraints of the optimization problem (3.11) which would otherwise be infeasible. This leads to the following extended formulation.

$$\begin{aligned} &\text{minimize} && \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}(\xi_i + \xi_i^*) \\ &\text{subject to} && \begin{cases} y_i - \langle w, x_i \rangle - b & \leq \epsilon + \xi_i \\ \langle w, x_i \rangle + b - y_i & \leq \epsilon + \xi_i^* \\ \xi_i,\, \xi_i^* & \geq 0 \end{cases} \end{aligned} \tag{3.12}$$

Here $C$ determines the trade-off between allowing deviations bigger than $\epsilon$ and the flatness of $f(x)$ and is usually chosen with a cross-validation approach. The extensions $\xi_i$ and $\xi_i^*$ of the constraints corresponds to an $\epsilon$-insensitive loss function $|\xi|_\epsilon$, which is illustrated in Figure 3.4.

$$|\xi|_\epsilon = \begin{cases} 0 & \text{if } |\xi| \leq \epsilon \\ |\xi| - \epsilon & \text{otherwise} \end{cases} \tag{3.13}$$

The next step is to construct a Lagrange function from the objective function. This step is well described in [17] and will not be repeated here. It finally leads to the dual optimization problem, where $\alpha_i$ and $\alpha_i^*$ are Lagrangian multipliers.

$$\begin{aligned} &\text{maximize} \quad \begin{cases} -\frac{1}{2}\sum_{i,j=1}^{N}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)\langle x_i, x_j\rangle \\ -\epsilon\sum_{i=1}^{N}(\alpha_i - \alpha_i^*) + \sum_{i=1}^{N} y_i(\alpha_i - \alpha_i^*) \end{cases} \\ &\text{subject to} \quad \begin{cases} \sum_{i=1}^{N}(\alpha_i - \alpha_i^*) = 0 \\ 0 \leq \alpha_i, \alpha_i^* \leq C \end{cases} \end{aligned} \tag{3.14}$$

From one of the saddle point conditions of the primal objective function follows $w = \sum_{i=1}^{N}(\alpha_i - \alpha_i^*)x_i$. With Equation (3.10) it results in the following term for $f$.

$$f(x) = \sum_{i=1}^{N}(\alpha_1 - \alpha_i^*)\langle x_i, x\rangle + b \tag{3.15}$$

The parameter $b$ can be computed by exploiting the Karush-Kuhn-Tucker conditions (see [17] for more information). Notable is also that $f(x)$ can be described solely by dot products of the data and $w$ does not have to be computed explicitly. This observation readies the way for nonlinear predictions by using kernels. $f$ can then be calculated as follows, where $K(\cdot,\cdot)$ is a kernel function.

$$f(x) = \sum_{i=1}^{N}(\alpha_i - \alpha_i^*)K(x_i, x) + b \tag{3.16}$$

The term kernel here refers to a different class of functions than the kernel functions mentioned in Section 2.2.2. In the context of SVMs a kernel is a similarity function for a pair of samples. In kernel density estimation a kernel is a weighting function.

Examples for general kernels that are in common use for support vector regression are listed below [16].

- Linear kernel: $K(x, y) = \langle x, y\rangle$
- Polynomial kernel: $K(x, y) = (\langle x, y\rangle + r)^n$
- Radial basis function kernel (RBF): $K(x, y) = \exp\left(-\gamma\|x - y\|^2\right)$
- Sigmoid kernel: $K(x, y) = \tanh(\kappa\langle x, y\rangle + \Sigma)$

The values of the hyperparameters used in the kernels are usually found by cross-validation.

The results of support vector regression with different kernels on the example from Section 3.2 can be seen in Figure 3.5. As expected, the linear kernel cannot fit a sinus-function. The polynomial kernel gives a better fit, while the RBF kernel predicts the underlying function really well.

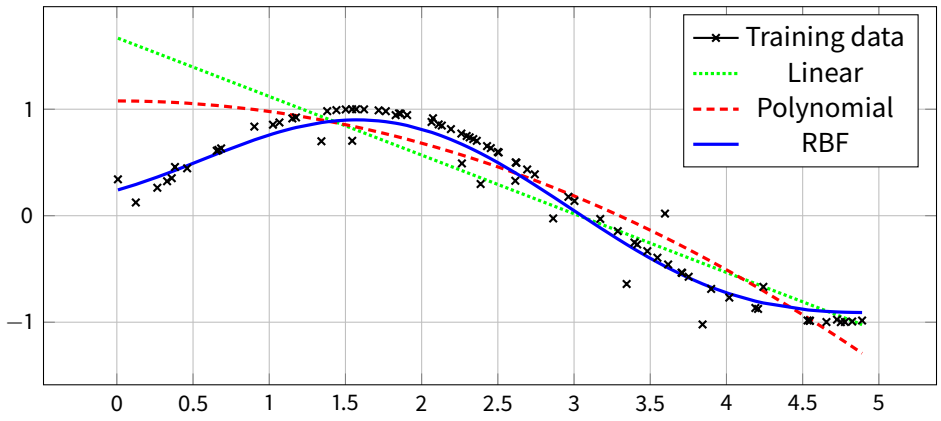



Figure 3.5: One-dimensional support vector regression trained on noisy data with with different kernels functions.

# 4

# MACHINE LEARNING APPROACH TO PAINT THICKNESS ESTIMATION

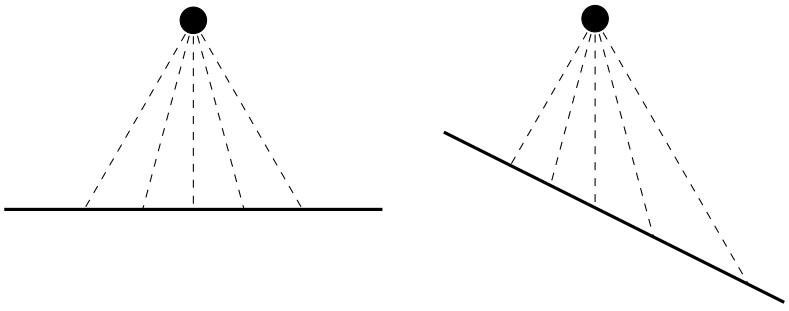
After presenting the theory of spray painting simulation and the introduction of some popular machine learning algorithms, this chapter presents the fundamental ideas of the hybrid simulation model.

First the selection of the response variable for regression is explained. Then the approach to incorporating all relevant parameters into features for the machine learning algorithm is presented.

## 4.1 RESPONSE VARIABLE

The choice to use machine learning algorithms instead of analytic error models for the error prediction task leads to the fundamental question which quantity should be predicted.

The obvious choice would be to predict the paint thickness directly. This approach has the advantage that no simulation tools other than the trained model are necessary. Therefore, the necessary measurements and calibrations for the projection method could be omitted. Additionally the source code of the projection method could be removed and thus ease maintenance of the simulation software.



(a) Even surface. (b) Tilted surface.

Figure 4.1: A paint applicator with opening angle of 60 degrees over surfaces with different tilt.

A different approach to machine learning is the prediction of the error of the projection method. This error is called projection error and represents the difference of the paint thickness of the physics-based simulation and the projection simulation.

Predicting the projection error has different advantages. First, the estimate of the error can simply be added to the thickness calculated with the projection simulation. Thus the method adds little complexity and can easily be exchanged once a better error prediction is available.

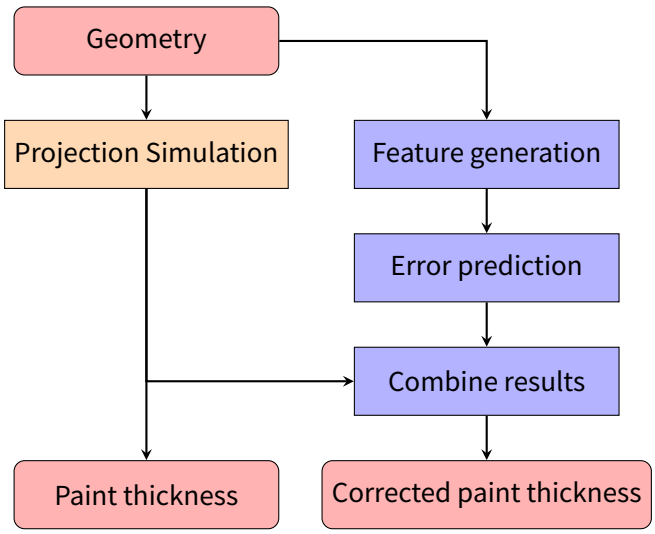



Figure 4.2: Schema of the hybrid simulation model.

Second, by predicting the projection error, one can utilize some intrinsic knowledge. In Figure 4.1 we see an applicator over two differently tilted surfaces. The rays, which are indicated by dashed lines, are representative for the projection simulation. In Figure 4.1a, where the surface is not tilted, they hit the surface evenly spaced. In comparison, in Figure 4.1b the surface is tilted relative to the applicator and thus the rays hit unevenly spaced. In the projection simulation rays that lie further apart correspond to a bigger area covered by the same paint volume, and thus less resulting paint thickness. As a result of this, the projection method encodes some knowledge about the geometry and the position of the applicator relative to the surface of the target. This intrinsic knowledge does not have to be learned by the machine learning algorithm, which results in more accurate simulation results.

Based on this reasoning this thesis focuses on estimating the projection error instead of predicting the paint thickness directly. The resulting hybrid simulation model can be seen in Figure 4.2. The left part represents the projection method, while the right part shows the necessary steps for the machine learning based error correction.

## 4.2 PROCESS PARAMETERS

In order to generate input features for the regression model, a generic way to get information about the painting scenario is necessary. Ideally one wants to include all relevant parameters into the features that are supplied to the regression algorithm.

There are different sources of parameters that affect the spray painting simulation. The three major sources are listed below.

- The process parameters of the paint applicator have a big impact on the simulation. Examples for such parameters are the volume of emitted paint per second, the direction and velocity of the shape air flow and the applied voltage.
- The applicator path, i.e. the motion of the robot arm, describes the applicators path and therefore its distance, orientation and speed relative to the target geometry.
- The target geometry with its geometrical features shapes the airflow and the electrostatic field around it and thus influences paint deposition.

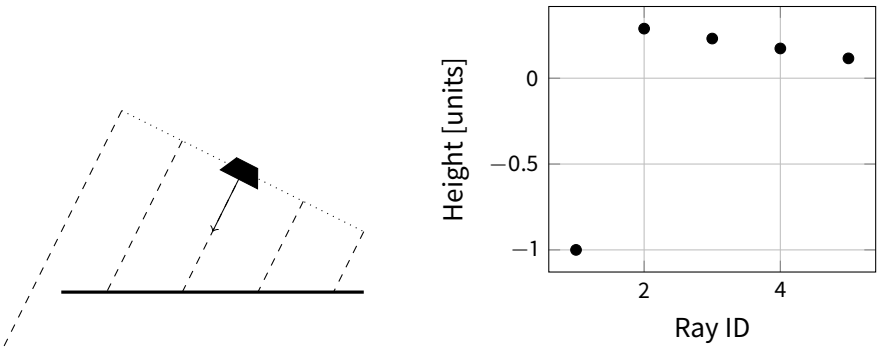
For the scope of this thesis the process parameters of the applicator were set by using the same applicator for the generation of the whole training set. The parameters used are listed in Section 5.1. This reduction of parameters simplifies the machine learning task significantly.

A different parameter that has an impact on paint thickness is the velocity of the applicator over the surface. A high velocity leads to an asymmetrical air flow, which in turn results in an asymmetrical paint distribution. However, this effect is negligible at small velocities and generally weaker than other factors like the distance of the applicator to the target. For this reason dynamic simulations can be sufficiently approximated by static simulations. These simulations are faster and need smaller simulated time frames to give meaningful results, which accelerates the generation of the training set as well.

## 4.3 FEATURE CREATION

After the mentioned simplifications, the remaining parameters are the applicator's position and orientation relative to the target geometry, and the surface structure of the target. One way to incorporate these parameters into the features for the machine learning algorithm is to generate a height map of the target geometry from the applicator's point-of-view. This can efficiently be done by utilizing ray tracing.

The principal idea can be seen in Figure 4.3a. From the position of the applicator and with its current direction rays are generated. They are then traced until they hit a triangle of the target or expire. In case of a hit, the distance to the triangle as well as the exact hit location in the triangle are returned.




(a) Schematic illustration of ray tracing from the applicator.

(b) Sample outputs for ray tracing.

Figure 4.3: Illustration of ray tracing with parallel rays and the respective outputs.

## 4.4 ROTATION AND VARIATION

As the multi-physics simulations are computationally expensive it is advisable to keep the number of painting scenarios as low as possible. It is therefore important to extract as much information as possible from each scenario.

There are several ways to do that. A simple way is to rotate the applicator around its symmetry axis. This idea is illustrated in Figure 4.4 for the one-dimensional case. Here the line, along which the ray tracing is performed, gets rotated and thus allows to retrieve multiple samples from the same scenario.

Another idea that is not yet implemented is to add small variations to the applicator's position and direction. This way more possible states of the applicator's position and direction get covered which results in a higher quality training set.

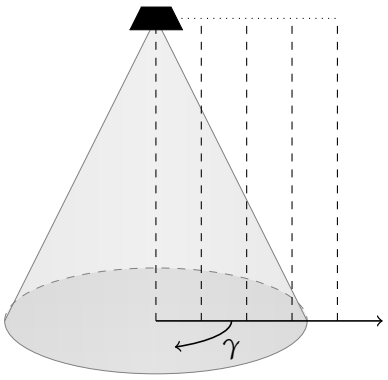



Figure 4.4: Rotation of rays to extract more samples from a painting scenario in the one-dimensional case.

# 5 METHODOLOGY

## 5.1 DATA SET CREATION

To train the machine learning algorithm to predict the projection error, a representative set of common painting scenarios in the form of a training set is necessary. This training set should cover the space of possible painting scenarios. It should also avoid redundancy, which is necessary to keep the training set reasonable sized and the training process itself fast.

The creation of the necessary tools to manage the painting scenarios and generate data sets from them was a essential part of this thesis. It will be presented in this chapter.

*Organization of painting scenarios*

As described in Section 4.2, the machine learning approach is based on some simplifications that reduce the parameter space. The remaining parameters, which have to be covered by the training set, are the applicator's position and orientation relative to the target geometry and the surface structure of the target. These parameters have the highest impact on the paint thickness and, besides all simplifications, still span a significant parameter space.

In this space, the parameters are systematically varied to create a comprehensive data set. In order to simplify management of the painting scenarios, the simulation files are organized in a tree. In it every node represents a common parameter (e.g. the same model or options file), while its children differ in one parameter. This method allows easy addition of further parameters or painting scenarios and deduplicates common files at the same time.

*Implementation of painting simulations*

After the different painting scenarios have been created, they have to be transformed into valid input files for the simulation tools. The simulation tools rely on several files which contain settings for different parts of the simulation.

- The geometry file stores the triangulation of the target object as well as its position in space.

- The option file includes settings for the simulation. These include the boundaries of the simulation, output frequency of intermediate results or the selected solvers.
- The applicator file contains the process parameters of the applicator. As described in Section 4.2, the same process parameters are used for the generation of the whole training set in this thesis. The settings of the adopted applicator are listed in Table 1.
- The path file stores the position of the applicator over time and controls the applicators paint flow (i.e. controls if paint particles are emitted or not).
- The script file is the interface to the simulation program and used to set up the simulation with help of the other mentioned files.

The static multi-physics simulations consist of three stages. First a duration of 0.4 seconds is spent to initialize the air flow and to stabilize it. During this time the emission of paint droplets is disabled. Subsequently, a period of one second follows in which paint is ejected. Then follows a third part with a duration of 0.5 seconds, where the paint flow is turned off again. The idea of the last stage is to give the paint particles that are still in the air time to reach the target geometry before winding the simulation down. This improves the accuracy of the simulation.

| Parameter | Value |
|---|---|
| Bell RPM | 27 000 rpm |
| Paint flow | 380 cc/min |
| Shape air flow | 600 l/min |
| Voltage | 80 000 V |

Table 1: Paint applicator process parameters.

*Running simulations efficiently*

After the static simulations have been created and added to the tree structure, the results for each simulation have to be calculated. This is a highly computationally expensive task. Depending on the actual parameters and the available hardware, one simulation can take several hours to complete. The set of painting scenarios itself comprises of several hundred painting scenarios.

In order to make it feasible to work with this high number of painting scenarios, a parallel system for running the simulations has been implemented. It is able to distribute the required work on multiple computers while offering a simple interface. As the

simulation tasks are independent of each other, the system scales linearly with the number of computers it runs on and therefore allows big reductions in total runtime.

*Creation of input features*

As supervised learning is used, both the inputs and the expected outputs of the error prediction have to be collected for the training set. The input consists of the height field generated by ray tracing.

For the ray tracing, a layout of the rays has to be chosen. This thesis uses an approach were the rays are parallel and directed in the direction of the applicator. This method is displayed in Figure 4.3a and gives a high-level overview over the geometry.

A different possibility is a layout that resembles a cone. Here the origin of all rays is the origin of the applicator but point towards the same endpoints as in the parallel method. This method gives a more local estimate but might run into problems for more complex geometries.

The ray tracing framework is based on the OptiX ray tracing package [13] by NVidia. It provides a low-level ray tracing framework for highly parallel architectures. In particular the OptiX Prime APIs are used, as they are "specialized to deliver high performance for intersecting a set of rays against a set of triangles" and exhibit better performance than the OptiX APIs.

*Calculating the expected output*

To train the regression algorithms the expected output (i.e. the projection error) is required. The projection error is the difference between the simulated paint thickness from physics-based simulation and projection simulation.

The features are supplied in a grid which is defined by the rays. The same grid is used for the error prediction results. However, the paint thickness values from multi-physics and projection simulation are only known at the nodes of the geometry mesh. These nodes usually do not coincide with the hit locations of the rays as illustrated in Figure 5.1. For that reason the paint thicknesses at the intersection points of the rays with the geometry have to be calculated. As the barycentric coordinates of the ray intersections are known, the required values can be interpolated from the known nodal values in the target geometries mesh with the following equation.

$$t = up_1 + vp_2 + wp_3 \tag{5.1}$$

Here $t$ is the interpolated thickness for one ray. The barycentric coordinates of the hit are denoted $u$, $v$ and $w$ and are multiplied with the paint thicknesses $p_i$ at the triangle's vertices.

Once the paint thicknesses at the intersection points of the rays are known for the physics-based and projection simulation, the projections error can be calculated as the difference of them.

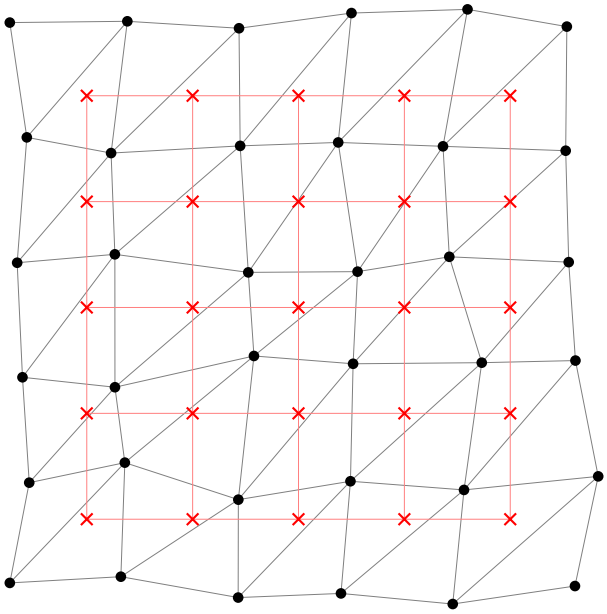


Figure 5.1: Example geometry mesh (black) with rectilinear grid created by ray tracing (red) in 2D. In three dimensions the ray tracing grid is not necessarily rectilinear.

*Scaling*

Before using the training set for training a regression model the input features and expected outputs should be scaled. This makes it easier for the algorithms to extract valuable information (especially support vector machines are sensitive to badly scaled features).

The data sets are scaled as follows.

- The features are scaled to have a mean of zero and unit variance.
- The outputs are scaled to be contained in the interval $(0, 1)$.

When scaling is applied to a data set the applied transformations have to be saved. This is important as the same transformations have to be performed on any input to the algorithm and its corresponding output.

## 5.2 AVAILABLE DATA SETS

To check the impact of the training set on prediction performance, the training is performed on different data sets. The evaluation of predictive performance is carried out on different data sets as well.

Most of these data sets are based on simplified target geometries that are created to train or to test special geometric features. As a basis usually a flat disc is chosen. Subsequently, geometric features that are present in real world geometries are added. Examples for these feature are curvature, tilt or bumps.

The disc geometry has a diameter of 1 m. This size ensures that in the middle of the disc no edge effects affect the simulation results. When moving closer to the edge, edge effects appear and can be trained as well.

There are three training set used for the training of the different regression algorithms.

- The EASY data set is the simplest and consists of 180 unique painting scenarios which are based on a flat disc geometry. The scenarios vary the radial position and the height of the applicator. From each scenario 8 samples are extracted, so that the data set consists of 1440 samples.

- The MEDIUM data set combines the EASY data set with variations of the disc geometry. Here discs with varying curvature and discs that are tilted with different angles relative to the applicator are used. The aim is to represent more realistic painting scenarios where the applicator is not directed orthogonally to the target geometry. This data set consists of 6120 samples from 765 unique painting scenarios.

- The HARD data set is the biggest and expands the MEDIUM data set with further cases. These are cases in which the disc geometry contains bumps in different versions. This data set contains 12360 samples from 1545 painting scenarios.

For the assessment of the predictive performance, the data set that was used for training is used for cross validation. Additionally performance is calculated on three other data sets.

- The INTERPOLATION data set contains painting scenarios which interpolate parameter values used in the EASY set. It therefore gives a hint about the generalization abilities of the regression model. It contains 24 samples from 3 unique painting scenarios.

- The COMPLEX data set is based on disc geometries that have complicated structure like sine-waves. These cases are not included in any of the training sets and thus show how well a model can extrapolate and adapt to unknown painting scenarios. It contains 120 samples from 15 painting situations.

- The SAAB HOOD data set contains painting situations that are based on a real world Saab car hood geometry and the respective robot movement. From this path static cases have been sampled

in different heights. Therefore, this data set shows how well a model can be expected to work in paint simulation of a real world geometry. It consists of 93 painting situations from which 744 samples are extracted.

## 5.3 MACHINE LEARNING

The implementation of the machine learning algorithms relies on the scikit-learn library [14]. This is a machine learning library written in Python that provides a consistent interface to a variety of implemented algorithms. This makes it easy to test and compare different algorithms with low implementational overhead.

In scikit-learn performance sensitive code is usually implemented in C which gives it reasonable performance compared to pure Python libraries. Furthermore it offers a high-level python API that is easy to use and integrate with other tools.

In this thesis, a subset of the available regression algorithms available in scikit-learn is compared. The selected algorithms are listed below. Their theoretical background is presented in Chapter 3.

- Linear regression is used as an indicator of whether linear models can be used in the context of spray painting error prediction. However, it is unlikely that good results will be achieved as linear regression usually works best in low-dimensional settings.
- Tree-based regression models are tested in the form of simple regression trees and regression random forests. They represent simple non-linear models that are easy to interpret and have good performance characteristics. The number of individual trees in the random forest is set to 10.
- The support vector regression model is included in the comparison as it is effective in high dimensional spaces and efficient. In the comparison an RBF kernel is chosen.
- Additionally, a dummy predictor is added to the comparison. It averages the training data and returns this value as the prediction. It can therefore be seen as a reference in performance and gives a better perspective on the performance of other models.

### 5.3.1 *Encoding position*

An important property of the error prediction task is that multiple outputs are required, i.e. the error prediction at different positions in space. Common regression approaches are only able to predict a scalar continuous value and are therefore not sufficient in this case. There are three different concepts to handle multiple outputs.

MULTIPLE MODELS The simplest way is to encode the position implicitly by creating and training independent models for every position. This approach has the advantage of being easy to implement. A disadvantage is that correlations between adjacent positions are neglected and thus predictive power is wasted.

Furthermore, as the number of positions grows (e.g. the two-dimensional case), performance of this method collapses as multiple models have to be stored and evaluated.

POSITION ENCODING A different approach is to explicitly model position relative to the applicator as an input to the regression model. Thereby only one model has to be trained and stored. However, it still has to be evaluated once for every position.

Another problem is that the encoding of position as a feature is not trivial. The simplest way to do this is to supply the position as a real numbered feature. But this encoding is generally hard to interpret by the machine learning algorithm and leads to bad prediction results.

Another approach is the so called one-hot-encoding, which transforms the continuous value into a discrete representation. The problem here is that a large number of features are generated, which slows down the training process.

MULTIPLE OUTPUTS The third way is to use special regression algorithms that have the multi-output characteristic. This means that they can predict vectors instead of scalars. This makes it possible to use just one model, which is used to output all required predictions and thus incorporates position implicitly.

The advantages are compelling. As only one model is used, performance is better than with the other approaches. Furthermore, the training is usually faster and can result in better predictions, as correlations between positions can be exploited.

The problem is that not all regression methods support multiple outputs. In scikit-learn the models for linear regression and tree-based models can handle multi-output regression. For support vector regression multiple outputs are currently not implemented. however, there is a theoretical foundation for multi-output SVRs [19, 20].

## 5.4 APPLICATION OF ERROR PREDICTIONS

After the regression model has been used to predict the projection error, one needs to use that information to improve the projection method's thickness result. The problem here is that the corrections are not calculated on the geometry mesh but on an independent grid which is created by the rays.

A exemplary, two-dimensional case is visualized in Figure 5.1. Here the black nodes belong to the irregular geometry grid. The red crosses

denote the intersections of the rays with the geometry. They form a rectilinear grid which is shown in light red.

In order to make the corrections usable for improving the projection method simulation, they have to be mapped back onto the geometry grid. This is accomplished by a least-squares method. The general idea is to find the nodal corrections $\bar{v}$ from the hit corrections $\bar{p}$ using the following relation.

$$\bar{p} = B\bar{v} \tag{5.2}$$

Here $B$ denotes the barycentric interpolation matrix, which can be calculated from the ray tracing results. Given this relation it is possible to create a minimization problem to find $\bar{v}$.

$$\min_{\bar{v}} \|\bar{p} - B\bar{v}\|^2 + \alpha\bar{v}^T M\bar{v} + \beta\bar{v}^T K\bar{v} \tag{5.3}$$

Here the additional regularization terms $\alpha\bar{v}^T M\bar{v}$ and $\beta\bar{v}^T K\bar{v}$ have been added. These terms ensure that the system is non-singular and therefore solvable. $M$ denotes the mass matrix and $K$ is the stiffness matrix of the geometry mesh. These matrices stem from the finite element method (FEM) and contain information about the mass and stiffness properties of the target objects triangulation. $\alpha$ and $\beta$ are parameters that control the trade-off between details and smoothness.

The term can now be expanded and a matrix $A$ factored out.

$$\min_{\bar{v}} \|\bar{p}\|^2 - 2\bar{p}^T B\bar{v} + \underbrace{\|B\bar{v}\|^2}_{=\bar{v}^T B^T B\bar{v}} + \alpha\bar{v}^T M\bar{v} + \beta\bar{v}^T K\bar{v} \tag{5.4}$$

$$A = B^T B + \alpha M + \beta K \tag{5.5}$$

From that we get the following:

$$\min_{\bar{v}} \|\bar{p}\|^2 - 2\bar{p}^T B\bar{v} + \bar{v}^T A\bar{v} \tag{5.6}$$

It is known that $A$ is positive definite, so we find the minimum after deriving for $\bar{v}$, which can be solved easily.

$$A\bar{v} = B^T\bar{p}^T \tag{5.7}$$

The values of $\alpha$ and $\beta$ have to be chosen empirically to get a good trade-off between preserving important features and smoothing the prediction.

# 6 RESULTS

In this chapter the results of this thesis are presented. First, the different regression algorithms are compared. Then the error prediction model is used to correct the error on different painting scenarios. Here positive and negative examples for the hybrid simulation model are shown.

## 6.1 COMPARISON OF REGRESSION ALGORITHMS

To compare the performance of the projection simulation with the hybrid simulation model, a measure of performance is necessary. There are three paint thickness estimates from which this measure can be constructed.

The physics-based simulation is assumed to be the right result and the reference for the faster, approximate methods. The projection method is the current base line performance and will be compared to the hybrid model, which includes the error correction.

A good measure for the performance of an approximate method is the following relative error, which assesses how similar the results are to the physics-based simulation.

$$E_{\text{hybrid}} = \frac{\|\bar{P}_{\text{physics}} - \bar{P}_{\text{hybrid}}\|}{\|\bar{P}_{\text{physics}}\|} \tag{6.1}$$

Here the norm of the paint thickness vectors $\bar{P}$ are taken. Results will be presented using the $l^2$ norm, which gives a good overview over the predictive performance. A value close to zero denotes good results, as then physics-based simulation and hybrid simulation are very similar.

The same measure can be calculated for the projection simulation. It then serves as a reference which should be improved by the hybrid model.

$$E_{\text{projection}} = \frac{\|\bar{P}_{\text{physics}} - \bar{P}_{\text{projection}}\|}{\|\bar{P}_{\text{physics}}\|} \tag{6.2}$$

First the results for the error prediction with 50 rays per dimension are presented. The rays are equidistantly spaced over a length of 40cm in each dimension. This results in a grid of 2500 rays and therefore 2500 features as the input for the regression models.

In Figure 6.1 we see $E_{\text{hybrid}}$ plotted for the selected regression algorithms (see Section 5.3) and the different data sets (see Section 5.2). Here, all models were trained on the easy data set.

We see that the dummy predictor has nearly no effect on the relative error. For the cross validation and the interpolation data sets the error decreases slightly but it increases for the other two data sets.

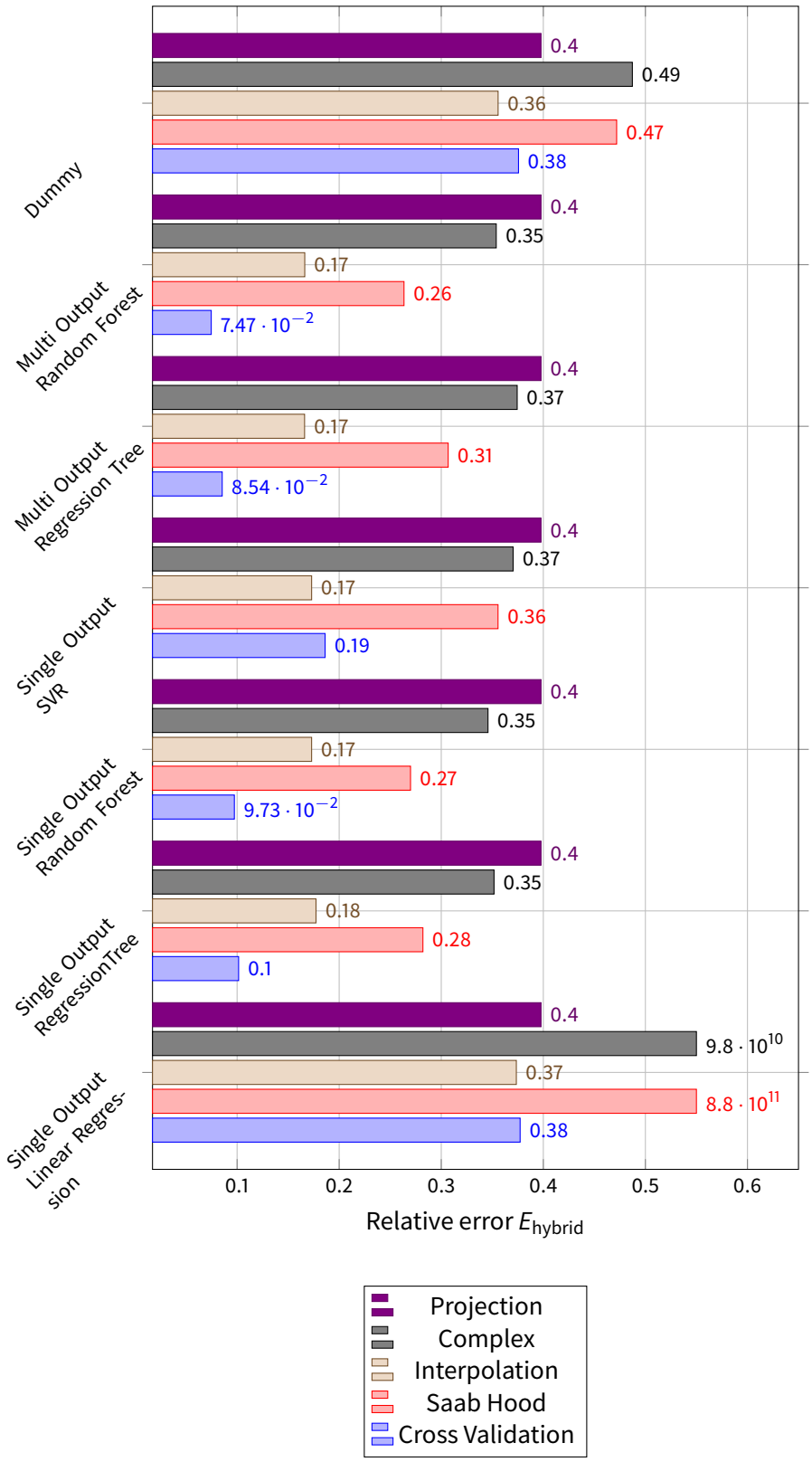



Figure 6.1: Relative error $E_{\text{hybrid}}$ for different regression methods and different test sets. All regression models were trained with the EASY data set. The relative error of the projection method is shown in purple for comparison.

Linear regression shows a similar picture for the cross validation and the interpolation data set. Here small gains are made. However, the error for the other two data sets increases dramatically and has to be clipped to fit the diagram. It seems that linear regression overfits to the given data and thus can handle cross validation and the interpolation cases, but is unable to generalize to the more complicated validation data sets.

Support vector regression shows better results. For cross validation and the interpolation data set the relative error is more than halved. The relative errors for the Saab hood data set and the complex data set exhibit small gains.

The different tree-based methods show the best results. The cross validation relative error decreases by more than 75 percent and the relative error of the interpolation data set is more than halved. The relative error of the Saab hood set decreases around 25 percent while the relative error for the hard data set improved only marginally.

The regular regression tree models seem to perform slightly worse than the random forest models. This is expected, as a random forest is an ensemble of regression trees. No clear advantages for the single-output or multi-output models can be seen in Figure 6.1.

In Figure 6.2 the results for training the regression models with the medium training set are shown. This training set includes curved and tilted surfaces as well and should improve predictive performance for the Saab hood data set. One can also notice that the relative error of the projection method grows slightly, as harder painting scenarios are now contained in the training set.

The changes in relative error compared to the training with the easy data set are shown in Figure 6.3. Here it is evident that the extension of the training set improves predictive performance. The highest gains are made for the complex and the Saab hood data sets. This shows that the regression algorithms extract information and are able to generalize from it.

The linear regression decreases its relative error the most but is still the worst model in the comparison. The reason for the improvement is probably that the number of samples is growing in comparison to the number of features, which in generally considered necessary in order to get good results from linear regression.

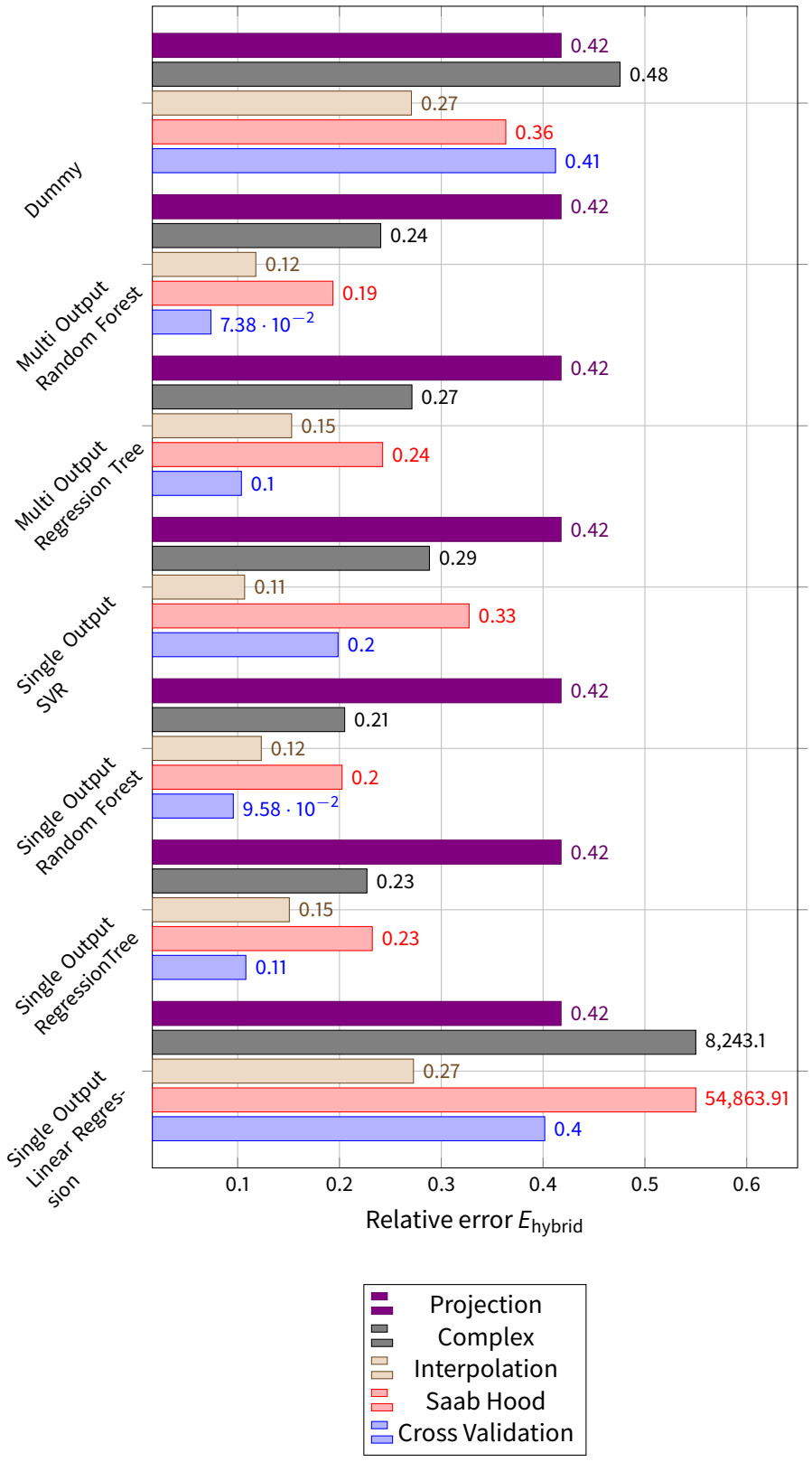



Figure 6.2: Relative error $E_{\text{hybrid}}$ for different regression methods and different test sets. All regression models were trained with the MEDIUM data set. The relative error of the projection method is shown in purple for comparison.

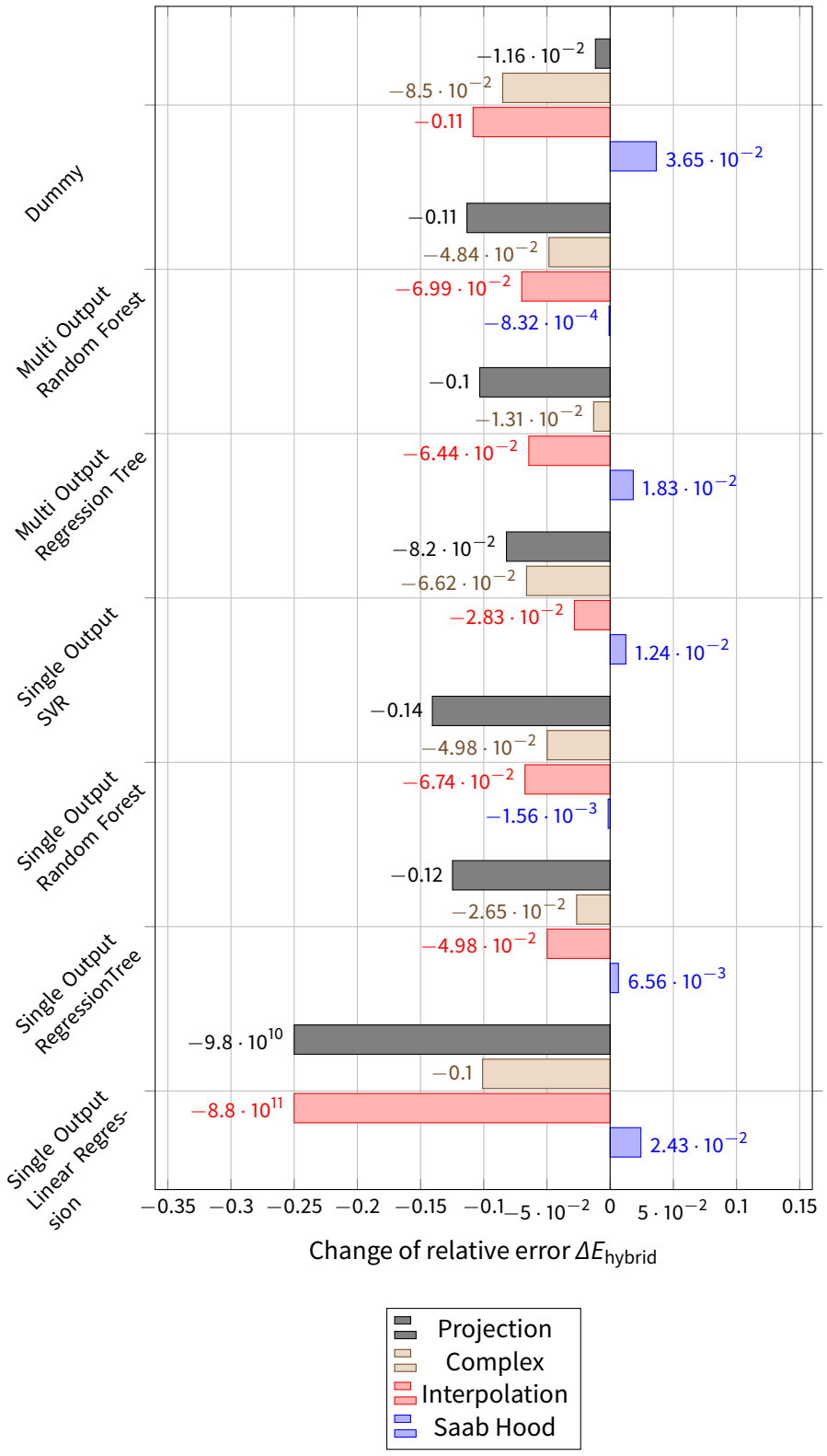



Figure 6.3: Change of the relative error $\Delta E_{\text{hybrid}}$ for training with training sets EASY and MEDIUM. Changes for different regression methods and different test sets are shown.

Figure 6.4 shows the relative error for regression algorithms trained with the hard data set. This is the biggest training set and includes additional painting scenarios with bumps in the target geometry. The relative error of the projection methods increases again. This in an indicator that this data set is even more difficult to handle for the projection simulation.

Here only multi-output models are shown. This is because training the single-output models requires a lot of memory and time, now that the training set contains 12360 samples. In comparison, the multi-output models can be trained more efficiently. Figures illustrating this behavior are presented further down.

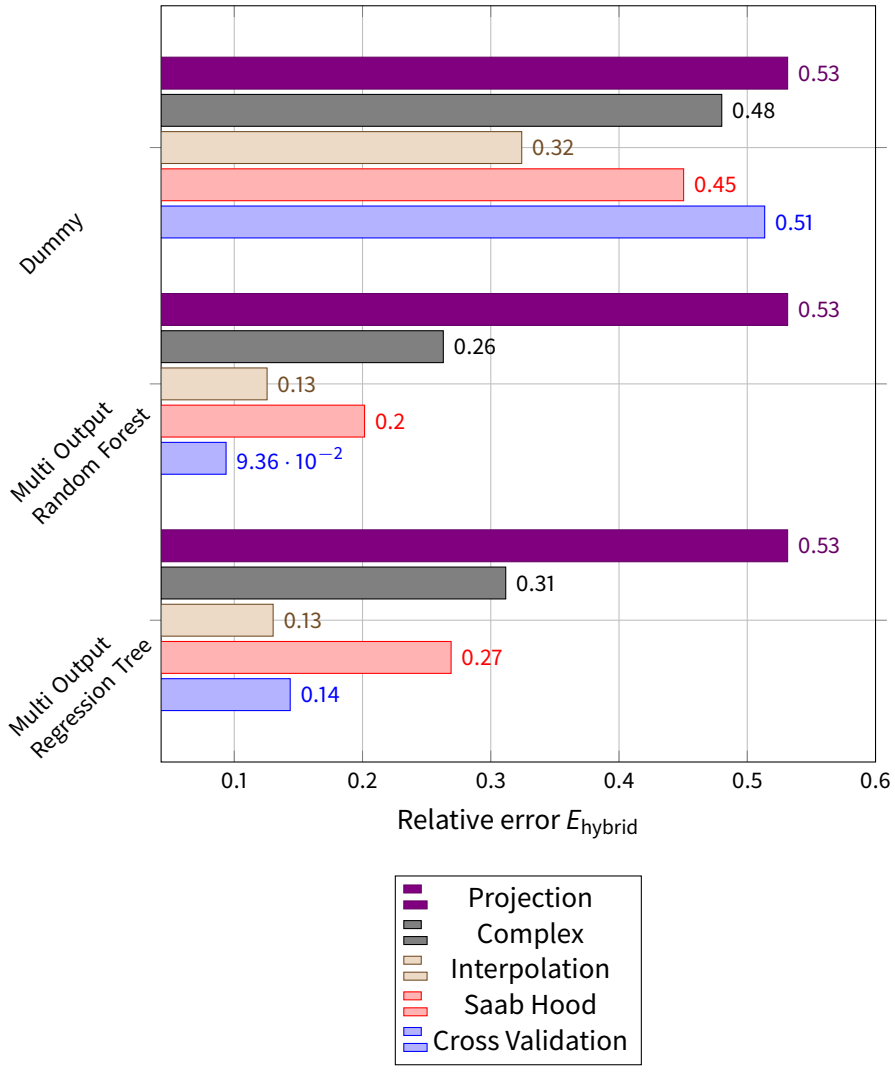



Figure 6.4: Relative error $E_{\text{hybrid}}$ for different regression methods and different test sets. All regression models were trained with the data set HARD. The relative error of the projection method is shown in purple for comparison.

The changes in relative error compared to the training with training set 2 are shown in Figure 6.5. One can see that the performance of the dummy regressor decreases as expected with growing difficulty of the training set. The relative error of the multi-output, tree-based methods degrades slightly. This looks like a decrease in performance but really is an increase. While the relative error of the projection method grows from 0.42 to 0.55, the tree-based models can nearly keep their relative error stable which means that their predictive abilities increase.

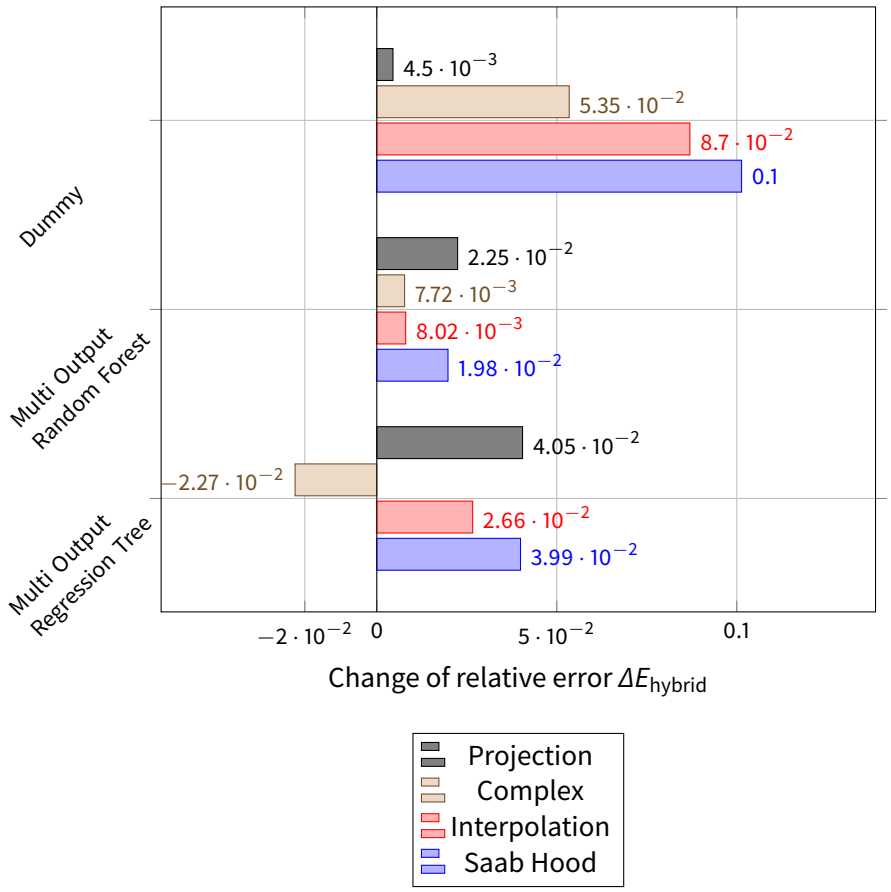



Figure 6.5: Change of the relative error $\Delta E_{\text{hybrid}}$ for training with training sets MEDIUM and HARD. Changes for different regression methods and different test sets are shown.

In Figure 6.6 the training times on the EASY training set are presented for the different regression algorithms. The measurements were taken on a computer with an Intel Core i7-4770 CPU and 32 Gigabyte RAM.

Regression trees are the fastest method and multi-output models have an advantage over single-output models here. SVR is slower than regression trees but faster than single- and multi-output random forests. The multi-output model of random forests is also faster to train compared to the single-output model which requires one minute longer. Linear regression is the slowest of the compared methods. This is surprising as efficient solvers are available for linear regression.

The result therefore hints at a suboptimal implementation of linear regression inside scikit-learn.

Figure 6.7 shows the training times for multi-output regression tree and random forests models for different sizes of training sets. The training of random forests is around one order of magnitude slower than the training of regression trees. The scaling of the training times is similar with regard to training set size.

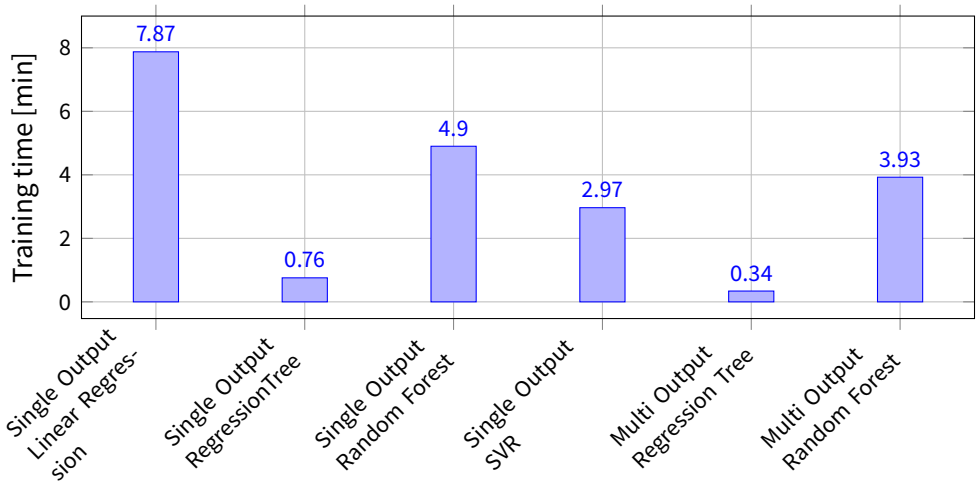



Figure 6.6: Training time for different regression methods.

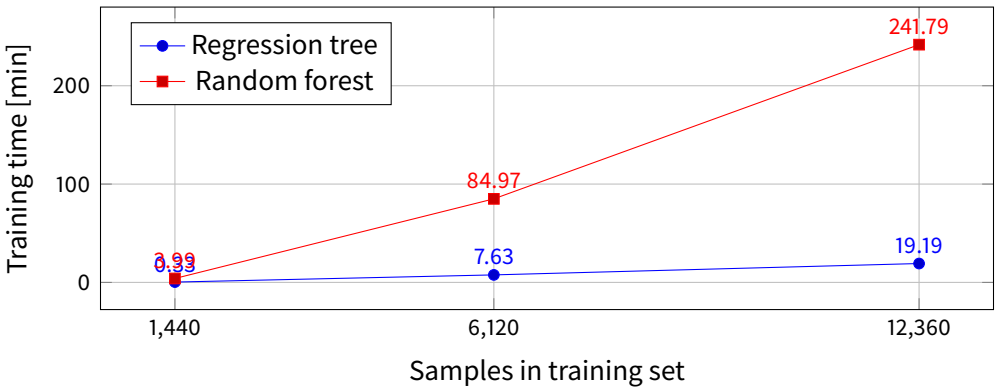



Figure 6.7: Development of training times for different training sets.

Figure 6.8 compares the prediction frequency of the different regression models. Note the logarithmic scale of the ordinate axis. One can see that multi-output models generally outperform single-output models in this regard.

The multi-output regression tree outperforms all other models by a huge margin. After that comes the multi-output random forest which is still a magnitude faster than the fastest single-output models.

Interesting to note is that the size of the training set seems to have no negative impact on the prediction frequency of most models. Only the SVR model halves its prediction speed. The other regression mod-

els even show small improvements in prediction speed for growing training set size.

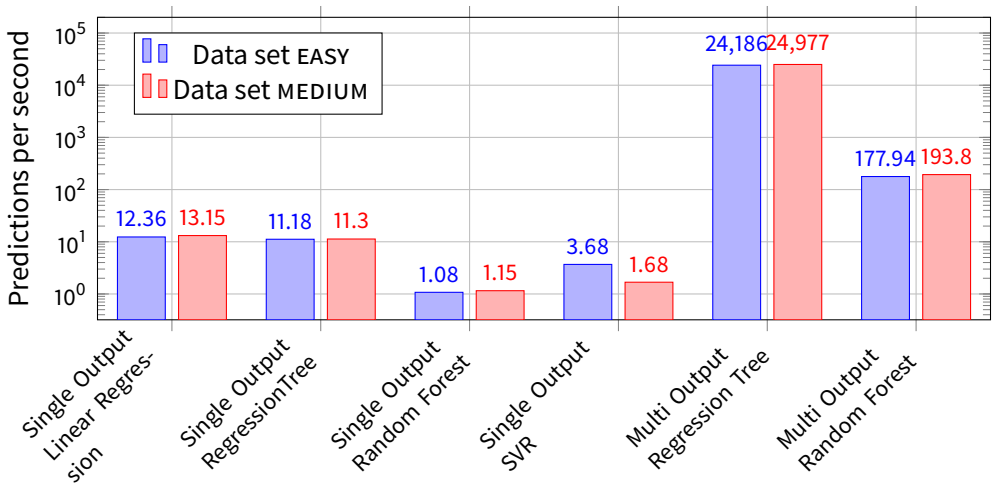



Figure 6.8: Performance of error prediction for different regression methods. Note the logarithmic scale of the ordinate axis.

In Figure 6.9 the size of the serialized regression models is plotted. As expected the linear regression model is the smallest. After that follow the regression tree and random forest models. The serialized SVR model is the biggest.

For the HARD data set only tree-based multi-output models are compared. Figure 6.10 shows the size of the trained models with different sizes of the training set. One can see that the size of the serialized models grows linearly with the number of samples contained in the data set.

Another interesting point is that the random forest model is always roughly 6.3 times bigger than an individual regression tree. As the random forest consists of ten individual trees, some optimizations must be performed inside scikit-learn.

*Conclusion*

In this section, several regression algorithms were compared. Their predictive performance for error correction has be evaluated on different training sets. Furthermore, other properties such as training time, prediction frequency and the size of the serialized models have been compared.

The result of this comparison is that tree-based methods work best for the error prediction task. Random forests have the best error prediction abilities in the comparison while having good predictions speed as well. Regression trees are faster to train and offer higher prediction speed, but their error prediction is worse in return. The

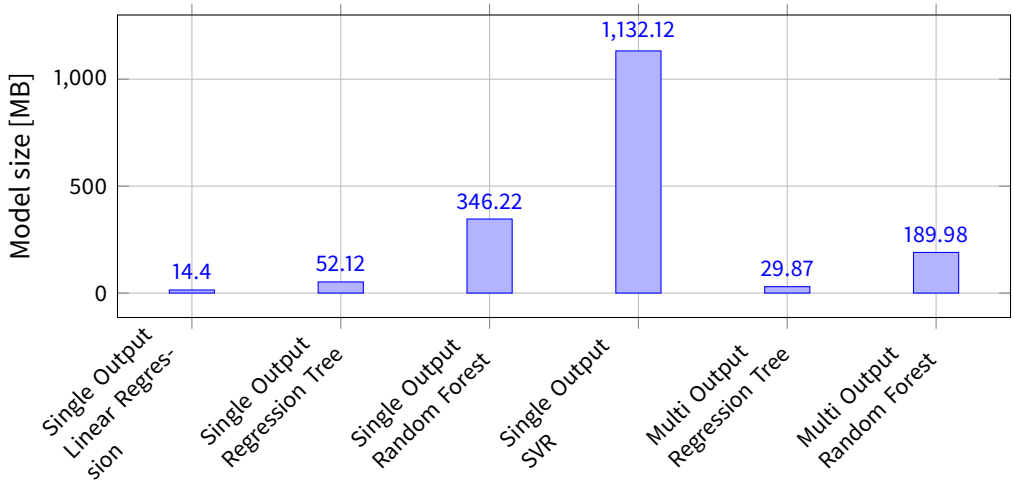



Figure 6.9: Model size for different regression methods.

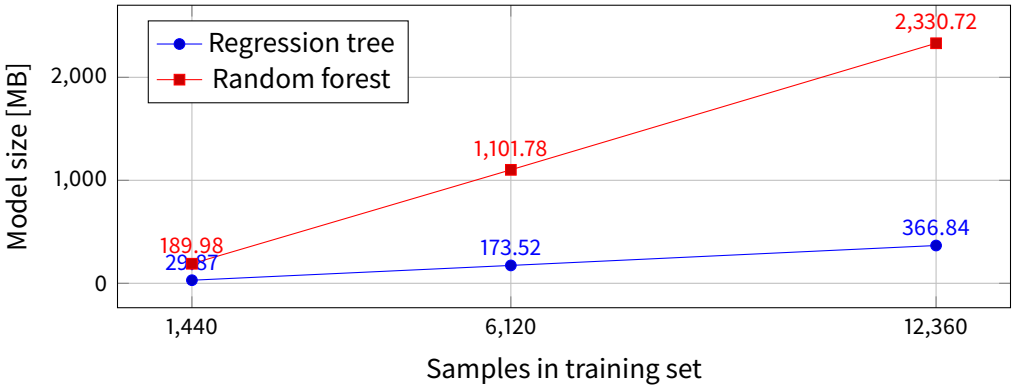



Figure 6.10: Development of model size for different training sets.

decision which algorithm to use is therefore also dependent on the size of the training set and the number of rays that are used.

The support vector regression model does not generalize as well as the tree-based algorithm. As a result, the error predictions on the Saab hood benchmark are mediocre. A reason for these result might be the feature scaling. In other tests the SVR model was really sensitive to the scaling of features and outputs, so different scaling might improve performance.

The SVR model is also the slowest model in the comparison. A problem here is also that SVR is not implemented as a multi-output model in scikit-learn, which explains some of the performance differences.

Linear regression as a model is too biased for the error prediction task. Therefore the error prediction actually resulted in worse results. A reason for that might be the choice of feature representation and the high-dimensional nature of the error prediction problem.

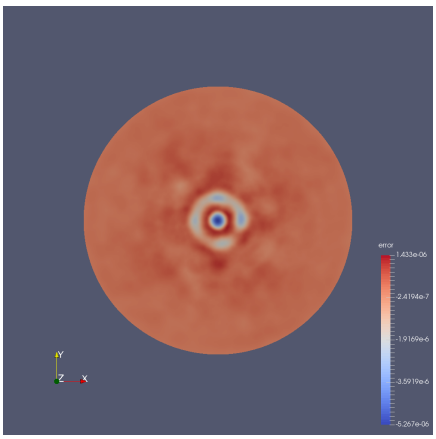
## 6.2 APPLIED CORRECTIONS

In the last section the performance of the error prediction is averaged over whole data sets and and it is thus not tangible how error prediction improves individual scenarios. For this reason, certain painting scenarios are selected and visualized in this section. For each scenario, the projection error, the error prediction and the error of the corrected paint thickness are presented. The error prediction itself is generated by a random forest which was trained on the hard data set.

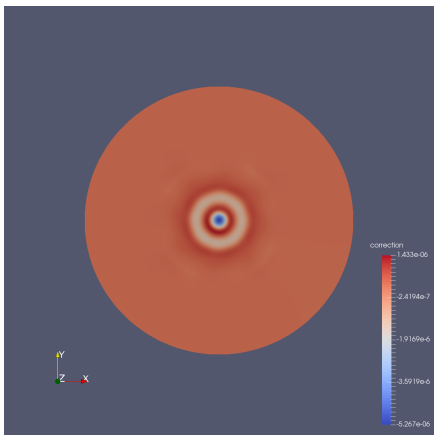
One of the simplest painting scenarios can be seen in Section 6.2. Here a flat disc geometry is painted with the applicator being positioned in 20 cm height over the center of the disc.

Figure 6.11a shows the real projection error. It is asymmetrical due to some grid effects in the multi-physics simulation. Figure 6.11b shows the predicted error. It captures the general features of the error and is symmetric. The error of the corrected thickness is shown in Figure 6.11c and shows a vastly improved picture with less and weaker spikes of error.

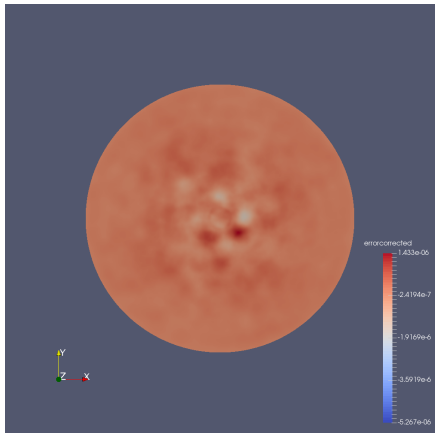
For this scenario the relative error decreases from $7.55 \cdot 10^{-5}$ to $4.39 \cdot 10^{-5}$, which is a 40% reduction. This reduction is remarkable as the projection method is adjusted on flat scenarios and thus gives good thickness estimates.




(a) Measured error.




(b) Predicted error.



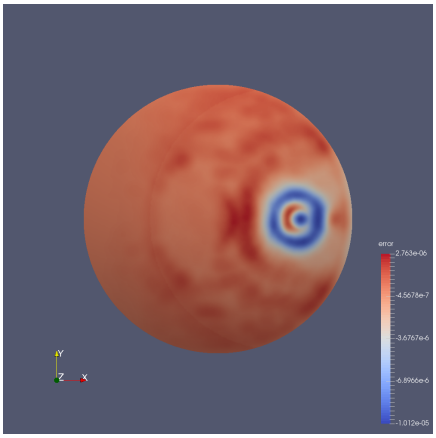

(c) Corrected error.

Figure 6.11: Flat disc scenario with centered applicator in 20 cm height.

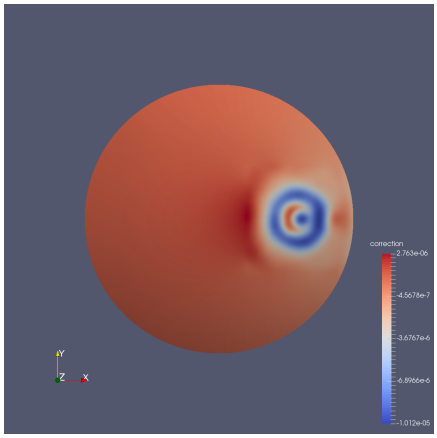
A more complicated painting scenario is shown in Section 6.2. Here a curved disc geometry is painted with the applicator being positioned in 20 cm height over the disc with an offset of 30cm from the center. This position of the applicator introduces edge effects which have to be respected by the error prediction model.

Figure 6.12a shows the projection error. It exhibits strong asymmetry based on the curvature of the geometry and the edge effects. Figure 6.12b shows the successfully predicted error. The error of the corrected thickness is shown in Figure 6.12c.

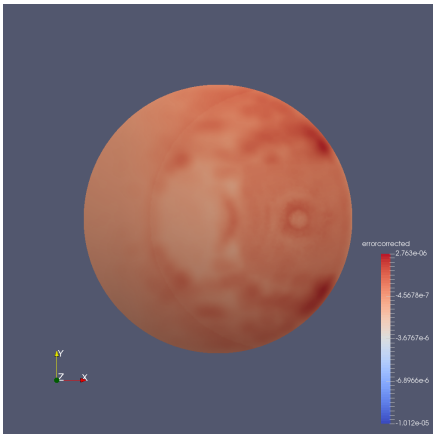
For this scenario the relative error decreases from $3.88 \cdot 10^{-4}$ to $1.30 \cdot 10^{-4}$, which is a 66% reduction.




(a) Measured error.




(b) Predicted error.




(c) Corrected error.

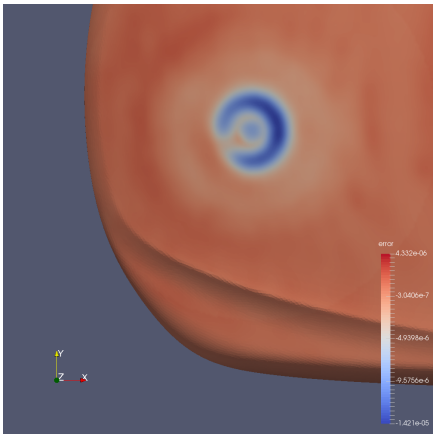
Figure 6.12: Curved disc scenario with the applicator in 20cm height and 30cm offset from the center.

The previous two examples are taken from the cross validation set and thus are expected to display good performance. Now a scenario from the Saab hood data set is displayed. Here the applicator is located 20cm over the hood geometry and not close to any edge.

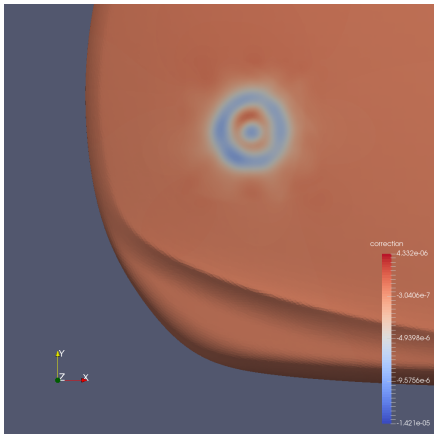
Figure 6.13a shows the projection error and Figure 6.13b shows the predicted error. Here the error prediction model correctly predicts the general shape of the error but underestimates its magnitude.

A possibility to improve the predictive performance for similar scenarios is to add scenarios that combine curvature and tilt to the training set.

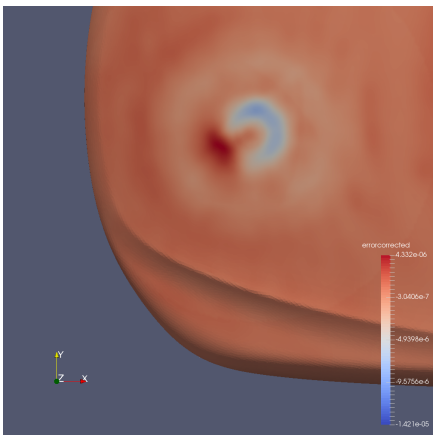
The error of the corrected thickness is shown in Figure 6.13c. The relative error is nearly halved from $2.70 \cdot 10^{-4}$ to $1.48 \cdot 10^{-4}$.




(a) Measured error.




(b) Predicted error.



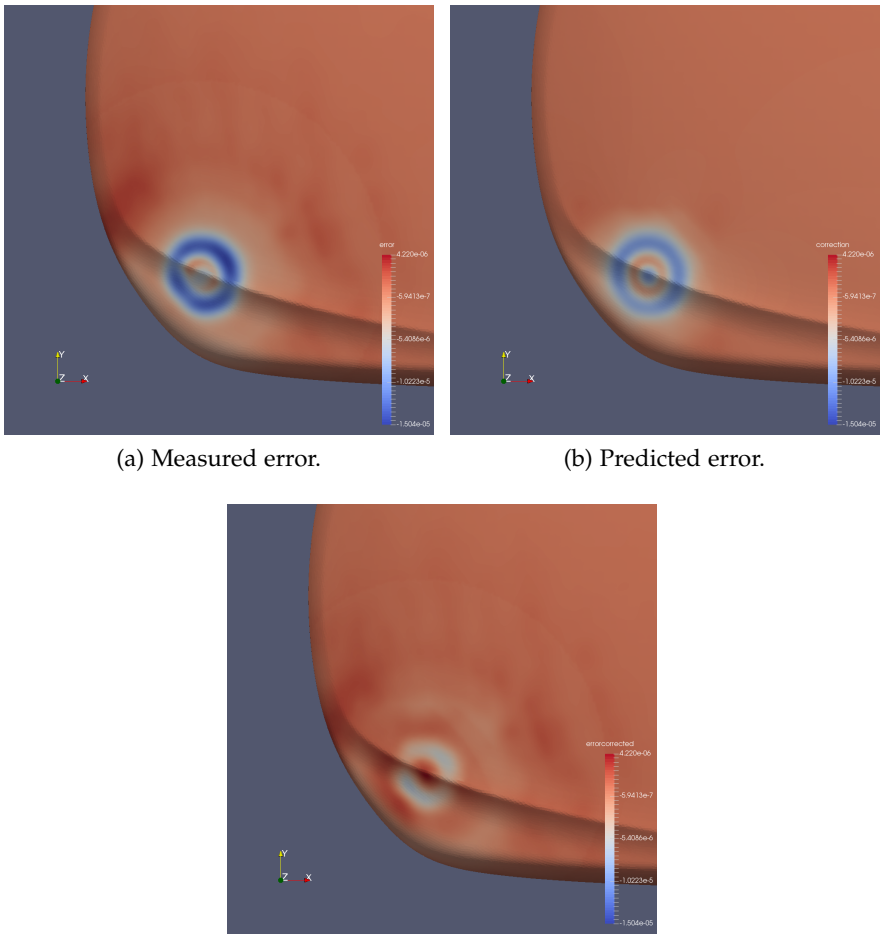

(c) Corrected error.

Figure 6.13: A scenario from the Saab hood data set. The applicator is positioned 20cm over the target geometry.

This scenario is also taken from the Saab hood data set. Here the applicator is located 25cm over an edge in the hood geometry.

The projection error and the predicted error can be seen in Figures 6.14a and 6.14b. Here the error prediction model correctly predicts the general shape of the error but again underestimates its magnitude.

The error of the corrected thickness is shown in Figure 6.14c. The relative error decreases from $3.11 \cdot 10^{-4}$ to $1.50 \cdot 10^{-4}$, a 52% improvement.




(a) Measured error.

(b) Predicted error.

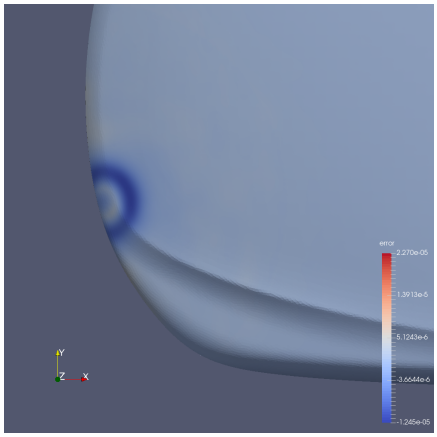
(c) Corrected error.

Figure 6.14: A scenario from the Saab hood data set. The applicator is positioned 25cm over the target geometry.

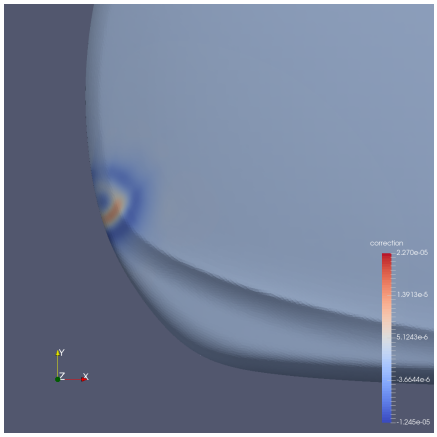
The last example is again taken from the Saab hood data set. This time the applicator is located 20cm over the edge of the hood geometry.

Figure 6.15a shows the projection error and Figure 6.15b shows the predicted error. Here the error prediction model is able to partially predict the blue ring, but miss-predicts the orange spot on one side.

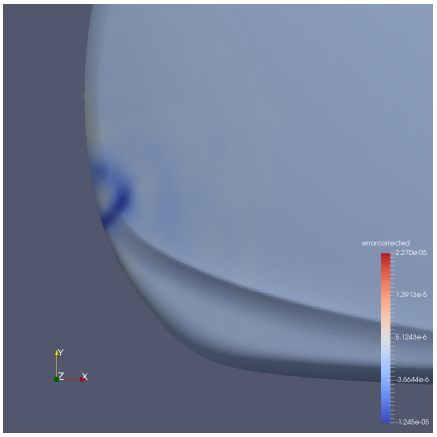
For that reason the error can not be reduced. The error of the corrected thickness is shown in Figure 6.13c. The relative error increases slightly from $3.51 \cdot 10^{-4}$ to $3.56 \cdot 10^{-4}$.



(a) Measured error.



(b) Predicted error.



(c) Corrected error.

Figure 6.15: A scenario from the Saab hood data set. The applicator is positioned 20cm over the edge of the target geometry.

# 7 CONCLUSIONS

Spray painting is a surface treatment process that is widely used in industry. It has a high environmental impact which can be reduced by automatic optimization tools. These tools rely on accurate simulations of the painting process, which are generally not fast enough to be useful in the context of automatic optimization. For that reason simplified, approximate simulation methods like the projection simulation are used. They achieve high simulation speed but also result in a considerable loss of accuracy.

In this thesis a novel approach for improving the approximate simulation methods is presented. By combining the projection simulation with an error correction a hybrid model with good performance and improved accuracy is created.

The error prediction is based on regression algorithms which rely on input features that capture the important parameters of the problem. These features are created in form of a local height field of the target geometry from the paint applicators coordinate system. That way captures all relevant parameters and can be efficiently computed using ray tracing.

The regression algorithms require a set of paint scenarios to learn from. For that reason, a database of common painting situations has been created. To accelerate the creation of the training set static simulations are used and efficient parallel tools for the management of the database have been implemented.

The error prediction capabilities of linear regression, support vector regression and tree-based regression methods are compared. The impact of different training sets on the error prediction performance is investigated as well. The results show that linear models cannot predict the projection error and thus non-linear models have to be used.

Tree-based methods provide the best error prediction abilities and are able to reduce the projection error more than 40% on real world benchmarks. Tree-based models are also the fasted algorithms among the compared regression models. From benchmarks follows that multi-output methods should be preferred of single-output methods in the context of error prediction, where the error has to be predicted in multiple positions.

# 8 FUTURE WORK

It has been shown that the general approach of using machine learning for improving simulation results is feasible. However, the current work flow is just a proof of concept and can be improved in numerous ways.

A first point is parameter selection. The presented concept contains many parameters, like the number of rays per dimension or the width of the ray tracing area. These parameters have been chosen according to best knowledge, but have not been validated systematically. Further parameter studies might be sensible here.

Additionally, some of the machine learning algorithms in scikit-learn are not implemented efficiently and thus waste computational resources. Performance gains are possible by switching to more efficient low-level libraries.

Besides these viable improvements to the current process, there are possible enhancements to the general approach.

An important factor for predictive performance is the compilation of the training set. Currently the training set consists of manually selected cases that are expected to improve predictive performance. As has been shown, increasing the data set can result in accuracy gains but does not necessarily do so. Therefore, it is advisable to find ways to construct efficient training sets (i.e. sets with small numbers of cases but good training results) for given target geometries. This would both speed up training times of the regression models as well as reducing the time spent to generate new training data.

A related aspect is the confidence of the prediction. Currently there is no way to assess whether the error correction is reliable or just an uncertain guess. A measure like this would yield opportunities to evaluate worst-case performance as well as providing further ways to improve the training set.

Another area where improvements can be made is the feature representation. The current approach with generating the height map from the applicator's view is efficient and captures all important parameters, but it also duplicates a lot of information. Different approaches for feature engineering exist, from principal component analysis to more recent deep learning techniques [2, 1]. Making use of these techniques can result in more meaningful features, which in turn reduces the number of features necessary for an accurate prediction.

A different, promising concept to reduce the number of features is to make use of symmetries. Currently the models do not exploit symmetry and thus the training set contains redundant data (i.e. different rotation angles of the applicator). By employing symmetry, the model could focus on smaller areas, which would improve prediction performance.

# BIBLIOGRAPHY


[1] Y. Bengio. Learning deep architectures for AI. *Foundations and Trends in Machine Learning*, 2(1):1–127, January 2009. ISSN 1935-8237. doi: 10.1561/2200000006.

[2] Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, Aug 2013. ISSN 0162-8828. doi: 10.1109/TPAMI.2013.50.

[3] L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996. ISSN 1573-0565. doi: 10.1023/A:1018054314350.

[4] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001. ISSN 1573-0565. doi: 10.1023/A:1010933404324.

[5] L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees*. Wadsworth and Brooks, Monterey, CA, 1984.

[6] S. Duncan, P. Jones, and P. Wellstead. A frequency-domain approach to determining the path separation for spray coating. *IEEE Transactions on Automation Science and Engineering*, 2(3):233–239, 2005.

[7] F. Edelvik, A. Mark, N. Karlsson, and J. S. Carlson. Math-based algorithms and software for virtual product realization implemented in automotive paint shops. submitted, 2016.

[8] J. K. Hastings, M. A. Juds, and J. R. Brauer. Accuracy and economy of finite element magnetic analysis. In *33rd Annual National Relay Conference*, 1985.

[9] D. Hegels, T. Wiederkehr, and H. Müller. Simulation based iterative post-optimization of paths of robot guided thermal spraying. *Robotics and Computer-Integrated Manufacturing*, 35:1 – 15, 2015. doi: 10.1016/j.rcim.2015.02.002.

[10] A. Mark and B. G. M. van Wachem. Derivation and validation of a novel implicit second-order accurate immersed boundary method. *Journal of Computational Physics*, 227(13):6660 – 6680, 2008. ISSN 0021-9991. doi: 10.1016/j.jcp.2008.03.031.

[11] A. Mark, R. Rundqvist, and F. Edelvik. Comparison between different immersed boundary conditions for simulation of complex fluid flows. *Fluid dynamics & materials processing*, 7(3):241–258, 2011.

[12] A. Mark, B. Andersson, S. Tafuri, K. Engstrom, H. Sorod, F. Edelvik, and J. S. Carlson. Simulation of electrostatic rotary bell spray painting in automotive paint shops. *Atomization and Sprays*, 23(1):25–45, 2013. ISSN 1044-5110.

[13] S. G. Parker, J. Bigler, A. Dietrich, H. Friedrich, J. Hoberock, D. Luebke, D. McAllister, M. McGuire, K. Morley, A. Robison, and M. Stich. Optix: A general purpose ray tracing engine. *ACM Transactions on Graphics*, August 2010.

[14] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[15] S. J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Pearson Education, 2 edition, 2003. ISBN 0137903952.

[16] B. Schölkopf and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA, USA, 2001. ISBN 0262194759.

[17] A. J. Smola and B. Schölkopf. A tutorial on support vector regression. *Statistics and Computing*, 14(3):199–222, 2004. ISSN 1573-1375. doi: 10.1023/B:STCO.0000035301.49549.88.

[18] S. Tafuri, F. Ekstedt, J. S. Carlson, A. Mark, and F. Edelvik. Improved spray paint thickness calculation from simulated droplets using density estimation. In *ASME 2012 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, pages 339–347. American Society of Mechanical Engineers, 2012.

[19] E. Vazquez and E. Walter. Multi-output support vector regression. In *13th IFAC Symposium on System Identification*, pages 1820–1825. Citeseer, 2003.

[20] S. Xu, X. An, X. Qiao, L. Zhu, and L. Li. Multi-output least-squares support vector regression machines. *Pattern Recognition Letters*, 34(9):1078 – 1084, 2013. ISSN 0167-8655. doi: 10.1016/j.patrec.2013.01.015.