ESTIMATING FUNCTIONAL PERFORMANCE FOR USE IN THE AESTHETIC DESIGN PROCESS

by

MATTHEW PAUL KENT

A thesis submitted to the
University of Birmingham
for the degree of
DOCTOR OF PHILOSOPHY

School of Computer Science
Faculty of Engineering and Physical Sciences
University of Birmingham
October 2014

Abstract

In engineering fields such as automobile design, optimisation of functional performance properties often conflicts with aesthetic optimisation. Functional performance feedback into the aesthetic design software may therefore improve the convergence of the design process. Unfortunately, many functional performance scores such as aerodynamic drag require intensive computational effort. We consider the use of machine learning approaches to instead provide *estimates* of these functional performance scores. We study the problems encountered when developing such an estimation function. The use of a historically accumulated data set of STereoLithography-format designs and their performance scores is suggested. We first look at preparing such a data set as training data for a machine learning task. Our first major novel contribution combats this problem in a manner similar to voxelisation. We next look at generating the regression function, seeking to achieve good generalisation across a large space of possible designs and for a problem where dimensionality reduction is challenging. Our second major novel contribution deals with this problem using an ensemble regression framework incorporating multiple data representations. Finally, we look at strategies of combining these two novel systems into a complete system. Upon evaluation, we conclude that our original aims have been met by this complete system.

Dedicated to family lost during this project: Grandma; Nana t. L.; Nana t. C. and Russ.

Acknowledgements

I would like to express my thanks to my supervisor at the University of Birmingham, Xin Yao, for giving me the opportunity to undertake this work, and my supervisor at the Honda Research Institute Germany, Markus Olhofer, for his support and guidance. I also offer sincere thanks to the Honda Research Institute for funding this project and giving me the opportunity to visit the institute regularly, learn from their scientists, make friends and perfect my German.

My visits to the Honda Research Institute would not have been the same without the colleagues and friends I made over there, including, but not limited to: Giles; Edgar; Sarah; Olga; Niko; Henry; and Christopher. Similarly a big thank you to the support staff, in particular Burkhard Zittel for his regular help regarding the computer systems, cluster use, and large scale data transfer.

I would like to thank support staff at the department of computer science, with special thanks to Laura Jackson for regularly arranging my trips to Germany, and Patricja Adams for her help with the administrative side. I am also am grateful to my fellow research students with whom I regularly held friendly conversation including in particular: Haobo; Loretta; Jakramate; Rodrigo and Shereen.

I send thanks to my friends in Liverpool and the surrounding areas, in particular John, Wombat, and Liam, for their support, help and regular distraction. So many thanks go to Anna for her support and encouragement during the closing stages of this adventure.

And finally, a big thank you to my family for their help and support over these few years.

Contents

1 Introduction **1**

1.1 Guiding questions . . . 3

1.2 Contributions . . . 4

1.3 Thesis layout . . . 5

2 Integrating Functional Performance into Aesthetic Design **7**

2.1 A motivating example . . . 7

2.2 Solutions for automated design processes . . . 10

2.2.1 Surrogate assisted design optimisation . . . 14

2.3 Related problems in the literature . . . 17

2.3.1 Building a regression function to aid engineering surface design . . . 17

2.3.2 Guiding human design decisions . . . 18

2.3.3 Feature analysis . . . 18

2.3.4 Knowledge management in engineering . . . 19

2.3.5 Concurrent engineering design . . . 20

2.3.6 Interactive evolutionary optimisation . . . 20

2.4 Machine learning for estimating functional performance . . . 21

2.4.1 Differences between our approximation and a surrogate . . . 22

2.4.2 Machine learning applied to our problem . . . 23

2.4.3 Finding a suitable source of training data . . . 24

2.5 Mathematical formulation of the problem . . . 28

2.6 Assessing approximation function performance . . . 29

2.6.1 Accuracy metrics . . . 29

2.6.2 The ranking metric . . . 30
2.6.3 Evaluating approximation function performance scores . . . 31
2.7 Conclusion . . . 32
2.7.1 Summary . . . 32
2.7.2 Novel contributions . . . 33

3 Preprocessing Surface Design Descriptions for Machine Learning 35

3.1 Problem introduction . . . 35
3.1.1 Engineering surface design representations . . . 36
3.1.2 Dimensionality reduction for a high dimensional non-linear problem . . . 38
3.1.3 Synergising representation & dimensionality reduction for the bump problem 39
3.2 Generating a single representation from varied surface design representations . . . 41
3.3 Dimensionality reduction for machine learning of engineering problems . . . 43
3.3.1 Feature selection methods . . . 44
3.3.2 Feature extraction methods . . . 45
3.4 Generating an experimental data set . . . 49
3.4.1 Adding realism . . . 53
3.4.2 Taking FFD handles as input . . . 56
3.4.3 Creating an ideal data set via feature selection . . . 56
3.5 Generating a data set exhibiting the bump problem . . . 57
3.6 A novel conversion framework: KGrid . . . 58
3.6.1 Aims of the approach . . . 58
3.6.2 The method . . . 59
3.7 Empirical analysis of KGrid . . . 62
3.7.1 Applying ISOmap to clustered data . . . 63
3.7.2 Comparing the KGrid representation with the ideal representation D^C . . . 67
3.7.3 Comparing the KGrid representation with the voxelised representation . . . 71
3.7.4 Comparing ISOmap and PCA for dealing with the bump problem . . . 79
3.7.5 Evaluation of KGrid by the original aims . . . 87
3.8 Analysis of the ISOmap representation of the automobile data set . . . 98
3.9 Conclusion . . . 99

3.9.1 Summary . . . 99
3.9.2 Novel contributions . . . 103
3.9.3 Future work . . . 103

4 Functional Estimation across a Large Unfocused Input Space 105

4.1 Problem introduction . . . 106
4.1.1 Large design representations and available training data . . . 106
4.1.2 Learning a large domain without a local design space focus . . . 106
4.1.3 Utility to the human designer . . . 107
4.2 Regression functions in machine learning . . . 108
4.2.1 Overview of regression approaches . . . 108
4.2.2 Ensemble regression approaches . . . 111
4.2.3 Improving regression performance for D^C . . . 116
4.3 A novel ensemble framework . . . 117
4.3.1 Aims of the framework . . . 117
4.3.2 Clustering techniques . . . 117
4.3.3 The method . . . 119
4.4 Empirical analysis of the ensemble framework . . . 121
4.4.1 Choice of three component technologies . . . 121
4.4.2 Experimental studies using the ideal data set D^C . . . 124
4.4.3 Analysis of the framework's performance on the ideal data set D^C . . . 127
4.4.4 Evaluation of the ensemble framework by the original aims . . . 137
4.5 Conclusion . . . 138
4.5.1 Summary . . . 138
4.5.2 Novel contributions . . . 138
4.5.3 Future work . . . 139

5 A Complete System for Performance Estimation of Surface Designs 141

5.1 Problem introduction . . . 141
5.2 Combining KGrid with the ensemble framework . . . 142
5.2.1 Aims of a complete system . . . 142
5.2.2 Methods of forming a complete system . . . 143

5.3 Empirical analysis of the suggested systems . . . 152
5.3.1 Option one experiment (no dimensionality reduction) . . . 153
5.3.2 Option two experiment (with dimensionality reduction) . . . 154
5.3.3 Option three experiment (with dimensionality reduction for clustering only) 155
5.3.4 Option four experiment (two separate dimensionality reductions) . . . 156
5.3.5 Summary of findings . . . 157
5.3.6 Evaluation of the suggested systems by the original aims . . . 158
5.4 Integrating the system into the design process . . . 159
5.4.1 On request assessment . . . 160
5.4.2 Auto refreshing assessment . . . 160
5.4.3 Embedding information into the visual surface representation . . . 160
5.5 Conclusion . . . 162
5.5.1 Summary . . . 162
5.5.2 Novel contributions . . . 162
5.5.3 Future work . . . 163

6 Conclusion **165**

6.1 Conclusion of the thesis . . . 165
6.2 A Summary of novel contributions presented and their significance . . . 166
6.3 Answers to the guiding questions . . . 169
6.4 Research limitations . . . 171
6.5 Further research directions . . . 172

References **175**

List of Figures

2.1 Photo of a wind-tunnel experiment involving a scale model of an aeroplane. . . . 11

2.2 Renderings of simulated airflow around the Honda Civic (European) design. . . . 12

2.3 Illustration of CFD mesh and CFD solver in action. 13

3.1 Wire-frame view of the Honda Civic (European) from an STL representation. . . 37

3.2 Illustration of the bump problem. 40

3.3 Two-dimensional ISOmap embedding of a handwritten number '2'. 50

3.4 Three-dimensional ISOmap embedding of pixel-images of a rendered face. 50

3.5 Regions of the vehicle design most strongly associated with each control group. . 53

3.6 Example classes for each of the 8 design groups in the automobile data. 54

3.7 Example classes for each of the 3 design groups in the spheres data. 58

3.8 2D visualisation of a uniform KGrid. 63

3.9 2D visualisation of KGrid representation generation for a surface. 64

3.10 2D visualisation showing how a surface is represented under KGrid. 65

3.11 Plots comparing KGrid with the ideal representation D^C in terms of MSE. . . . 74

3.12 Plots comparing KGrid with the ideal representation D^C in terms of r^2. 75

3.13 Plots comparing KGrid with the ideal representation D^C in terms of PPC. . . . 76

3.14 Scatter plots using PCA with KGrid and D^C representations. 77

3.15 Plots comparing KGrid with the voxelised representation in terms of MSE. . . . 81

3.16 Plots comparing KGrid with the voxelised representation in terms of r^2. 82

3.17 Plots comparing KGrid with the voxelised representation in terms of PPC. . . . 83

3.18 Scatter plots using ISOmap with KGrid and voxelisation representations. 84

3.19 Scatter plots using PCA with KGrid and voxelisation representations. 85

3.20 Plots comparing KGrid with the voxelised representation in terms of MSE (spheres). 89

3.21 Plots comparing KGrid with the voxelised representation in terms of r^2 (spheres). 90
3.22 Plots comparing KGrid with the voxelised representation in terms of PPC (spheres). 91
3.23 Plots comparing ISOmap with PCA in terms of MSE (spheres). 92
3.24 Plots comparing ISOmap with PCA in terms of r^2 (spheres). 93
3.25 Plots comparing ISOmap with PCA in terms of PPC (spheres). 94
3.26 Scatter plots using ISOmap with KGrid and voxelisation representations (spheres). 95
3.27 A plot of the 2D ISOmapped experimental data. 100
3.28 A plot of the 2D PCA'd experimental data. 101

4.1 Diagram of the framework architecture. 122
4.2 Box charts for MSE and r^2 scores to accompany the data in table 4.1. 128
4.3 Box charts for PCE and PPC scores to accompany the data in table 4.1. 129
4.4 Box charts for MSE and r^2 scores to accompany the data in table 4.2. 130
4.5 Box charts for PCE and PPC scores to accompany the data in table 4.2. 131
4.6 Box charts for MSE and r^2 scores to accompany the data in table 4.3. 132
4.7 Box charts for PCE and PPC scores to accompany the data in table 4.3. 133
4.8 Illustration of the interesting attribute in clusters D and G. 136

5.1 Diagrammatic representation of each system combination option. 147

List of Tables

3.1 Frozen control groups per design group. 54
3.2 Cluster bridging variation 1 performance scores. 67
3.3 Cluster bridging variation 2 performance scores. 68
3.4 Cluster bridging variation 3 performance scores. 68
3.5 Performance of artificial neural networks with varying numbers of hidden neurons. 71
3.6 KGrid representation versus the ideal representation using PCA. 72
3.7 KGrid representation versus the ideal representation using ISOmap. 73
3.8 KGrid representation versus the voxelised representation using ISOmap. 79
3.9 KGrid representation versus the voxelised representation using PCA. 80
3.10 KGrid representation versus the voxelised representation using ISOmap (spheres). 87
3.11 KGrid representation versus the voxelised representation using PCA (spheres). . . 88

4.1 Ensemble framework performance on split one of the data (all clusters). 127
4.2 Ensemble framework performance on split two of the data (clusters AFGH). . . . 127
4.3 Ensemble framework performance on split two of the data (clusters BCDE). . . . 127
4.4 Performance of the ensemble framework where $\gamma = 0$. 134
4.5 Dimensionality after PCA by cluster. 135

5.1 Results for the system without dimensionality reduction (option 1). 153
5.2 Results for the system with dimensionality reduction (option 2). 155
5.3 Results for the system with dimensionality reduction (option 2; spheres). 155
5.4 Results for the system with clustering dimensionality reduction (option 3). 156
5.5 Results for the system with two dimensionality reductions (option 4). 157
5.6 Results for the system with two dimensionality reductions (option 4; spheres). . . 157

Chapter 1

Introduction

In many engineering disciplines a number of performance goals may exist at any one time. These can be very generally grouped into either functional or aesthetic goals.

Functional goals involve the design meeting some quantitative performance target, such as a physical property like aerodynamic drag and lift; stability; security; and energy-efficiency. Often these goals require some physical experiment to be carried out, although with advancements in computer technology we have seen computers being able to generate high-fidelity estimations of these functional performance properties using complex models of physical processes (such as Computational Fluid Dynamics (CFD) simulations in the case of aerodynamic performance, or Finite Element Modelling (FEM) in the case of stability).

Aesthetic goals, relating to the human-perceived appearance of the engineering design, are qualitative and subjective. It is important to note that because of this human-subjectivity, it is unfeasible to expect a computer to evaluate aesthetic design quality to a useful level of fidelity, especially in a domain that is seeking strictly novel designs.

This distinction in goals leads to a distinction between the types of human designer assigned to each task. In the case of optimising functional performance, a talented engineer with an understanding of the underlying functional process is usually required. He/she should be able to make a reliable assessment of the design changes required to make improvement to the particular functional performance goal. On the other hand, aesthetic optimisation is carried out by a talented artist, able to make artistic changes which will improve the design's quality the most according to their personal and subjective view.

Under the popular Concurrent Engineering design methodology, the preference is to build

integrated design teams that are as cross-functional as possible [83, 73]. In practice, some divisions may always remain, such as the separation between artists and engineers (as artists will prefer not to have engineers watching over and directing their work) which then causes separate teams to emerge, and the concurrent engineering design process becomes iterative [83]. We then understand that:

- These teams are in practice mutually exclusive.
- Each team does not possess the knowledge needed in order to fully understand the work of the other team.
- Each team may perform design modifications which conflict with the other team's design goals and modifications.

According to researchers at the Honda Research Institute, these frictions lead to a slower engineering design optimisation process, with the changing design being passed back and forth between the aesthetic and engineer teams many times. With management pushing for shorter design-to-market time-frames, this inevitably leads to some compromise on the part of each team's objectives.

A question then arises from this problem: can we aid the engineering design process, speeding it up, reducing the design-to-market time-frame, and also reducing the need for design compromise from the design teams?

An obvious solution is the integration of some useful functional performance feedback into the design software of the aesthetic designer, indicating the effect that particular design decisions may have on the design's functional performance (and thus, the functional performance team's goals).

As an example, with this system in place, the engineering team may pass a design to the artist team for aesthetic optimisation, with the guiding instruction that a given functional performance score is to be only preserved or improved, and under no condition worsened.

The functional performance feedback system could then indicate to the artist the functional performance of the design in its current form, and hence they can judge whether this constraint has been maintained. Further this data could be embedded into the design representation within the aesthetic design software, indicating which regions of an engineering surface design could be modified for the optimal functional performance change.

The main problem we will encounter with this, is that for many functional performance properties there exists no evaluation that can be performed so fast that it wouldn't result in a large slow-down in the aesthetic designer's progress. For example, a typical CFD simulation can take hours or days depending on the fidelity sought, during which the aesthetic designer would have to wait in order to assess properly their current modifications.

We will see in the following chapter that in the fields of automated optimisation it is common to use the output of a machine learned function, such as a neural network, as an estimate of the functional performance value. This machine learned function bases its estimate on a training data set of inputs and outputs from the same problem domain.

In this thesis we will look at the major challenges found in generating an estimation function for this purpose and attempt to resolve these problems using novel approaches.

1.1 Guiding questions

Here we will list five core questions that will be answered during the course of this thesis.

1. **Question:** *How can we generate estimations of functional performance for use in the aesthetic design process of an engineering firm, with minimal computational cost?* **Guiding principle:** In the next chapter, we will break this over-arching question into the following four sub-questions.

2. **Question:** *What problems are encountered when attempting to apply an off-the-shelf machine learning algorithm to the problem of estimating functional performance of engineering surface designs for use in guiding the aesthetic design process?* **Guiding principle:** When trying to implement a system like this using machine learning techniques, what kind of problems can we expect to encounter, if any. The answer to this question will lead to the next two questions.

3. **Question:** *How do we acquire a strong data set of training samples for use in this machine learning task, containing relevant knowledge for guiding the aesthetic design process?* **Guiding principle:** Machine learning algorithms typically assume some bank of training data is available from which to learn the input to output relationship. Sampling plans can often lead to data sets that are informative, but not necessarily fit for purpose. In

engineering surface design it is not unreasonable to assume that such a wealth of training data does indeed exist and also contains knowledge regarding previously interesting designs from an aesthetic perspective. Unfortunately, this training data usually takes on a variety of representations which makes it impossible to learn from without some careful preprocessing.

4. **Question:** *How do we generate a regression function with good generalisation for a highly-non-linear and non-uniformly sampled problem with multiple sub-spaces of interest?* **Guiding principle:** Human designers are difficult to predict and may radically alter a design, transforming it into a completely different subspace of the design space. This means that we must be ready to estimate functional performance across a large space of possible designs. Fortunately, only a subset of design features affect functional performance [34, 48, 42]. For a given artist or design-style we can then expect that a further reduced subset of these features may be modified by a human designer [56, 125]. Unfortunately, the exact subset changes throughout the space of possible designs, making dimensionality reduction and regression difficult.

5. **Question:** *How can we provide a system which human designers can place trust in and extract the most utility?* **Guiding principle:** Many machine learning methods from computer science result in an function which is somewhat mysterious and "black-box" like (i.e. from the user's perspective it is merely presented with an input and an output is produced: the inner workings are not seen). Such a system can be difficult for a human who is expert in a particular field (such as aerodynamics) to place trust in. Further, they cannot easily acquire any meaningful information from the learned system. We are interested in methods which maximise the potential for a human engineer to acquire some information, and in methods which they can trust due to some level of transparency in the system.

1.2 Contributions

1. **The problem.** A novel problem is described and a method of generating an experimental data set exhibiting the problem's properties is provided. The solution to the problem is

divided into two sub-problems handled in separate chapters.

2. **A novel representation conversion approach suitable for manifold-learning approaches of dimensionality reduction.** We present a novel method of preprocessing the varied data set in order to generate a representation suitable for machine learning. We give an example of a problem under-researched in the literature and demonstrate the power of manifold learning techniques in solving this particular problem. This involves a novel study of the application of a particular manifold learning technique to an engineering problem.

3. **Ensemble of expert regression functions with expert-specific representations.** A novel framework is described for dealing with the core problem described. It is shown through experimental analysis to be the best option considered from a selection of ensemble regression approaches. We justify why this may be the case and suggest compelling reasons why this method may be the best choice for somebody considering setting up the system described in this thesis.

4. **A complete framework for generating a functional performance estimation function based on a non-uniformly sampled data set of diverse designs in varying representations.** After confirming that the novel solutions to the two sub-problems work well in isolation, we demonstrate how they may be combined effectively in order to take a varied data set and generate from it a functional performance estimation function which can be reliably used in engineering tasks. We take the opportunity to make some suggestions regarding the actual integration of the system into the aesthetic design process.

1.3 Thesis layout

The structure of the remainder of this thesis is as follows:

Chapter 2 describes the problem central to this thesis in more rigour. We put forward a motivating example of a real-world engineering surface design task demonstrating the problems central to this thesis and review the literature related to this top. In particular we take inspiration from the field of surrogate assisted evolutionary optimisation in dealing with this task. Following this inspiration we look at machine learning and formulate the problem in terms of a machine

learning task. We describe the key properties of the problem described thus far, answering in particular the second guiding question mentioned above. We round off the chapter with a mathematical formulation of the problem and a description of the performance evaluation scores used throughout this thesis's experimental studies.

Chapter 3 outlines known approaches for the problem of preprocessing the data set (such that it is in a uniform representation) and subsequently performing dimensionality reduction on it. We discuss the synergy between these two tasks and the benefit of using a particular family of dimensionality reduction techniques (manifold techniques). We then describe the generation of two test data sets used in particular experiments within this thesis: a realistic automobile data set and an artificial data set featuring a property we are interested in. We present a novel method of generating a uniform data representation with low computational effort and compare two families of dimensionality reduction on this data representation, answering the third guiding question and paying attention to the fifth guiding question. Empirical results using the experimental data sets are provided and key comparisons are made in order to draw conclusions regarding this novel method.

Chapter 4 outlines known approaches towards solving the problem of machine learning a regression function for the problem central to this thesis. We pay particular interest to ensemble methods. We then present a novel ensemble framework combining dimensionality reduction and clustering as an answer to the fourth guiding question but also paying attention to the fifth guiding question. We justify the use of our framework and perform empirical experiments using our novel framework and the experimental automobile data set.

Chapter 5 demonstrates how the novel solutions to sub-problems in chapters 3 and 4, proven to work in isolation, can be combined with dimensionality reduction methods in a framework. We perform empirical experiments using multiple methods of combination and the experimental data sets in order to draw conclusions. We also suggest methods of integrating the system into the aesthetic design process. This chapter seeks to answer first guiding question, while, like its predecessors, paying attention to the fifth.

Chapter 6 concludes this thesis and summarises the novel contributions and significance of the work. We then describe the limitations of this work and suggest some further research directions which could be taken to further the research in this field.

Chapter 2

Integrating Functional Performance into Aesthetic Design

In this chapter we will consider how we can include functional performance information in the aesthetic design process, drawing inspiration from surrogate assisted design optimisation. A real-world motivating example is provided and the particular challenges that make our task difficult are described, setting us up for the following chapters where solutions to these challenges are developed.

2.1 A motivating example

For an example of an engineering surface design task such as that described in the chapter 1, we look at the domain of automobile design. All of the claims in this section regarding automobile design have been accumulated through interactions with the researchers tackling related problems at the Honda Research Institute Europe, Offenbach, Germany.

Automobile design optimisation can be very broadly grouped into functional and aesthetic objectives. The functional performance of an automobile design can have major impacts on the experience of the user (for example its speed; acceleration; handling; safety; and energy consumption). A prominent and specifically interesting functional property is the aerodynamic drag of a design, which acts as a frictional force, affecting (amongst others things) speed and energy consumption. Typically, designers will seek to minimise this force. The aesthetic quality of an automobile is self explanatory and totally subjective.

Optimally, aesthetics and functional performance should go hand in hand. Unfortunately the two design teams optimising for these objectives almost never share designers, assuming an automobile company has sought the most talented individuals to be in these teams. Seemingly, they each optimise very different properties of the design, but being teams of very different designers it should be expected that they show very little regard for the work of the other (a rivalry could even develop) [92].

In such a situation, a number of options exist:

1. Implement an iterative design cycle, where the design is passed between the teams for optimisation. Assume that some converged design which is satisfactory to both design teams will be met.

2. Have teams rank and inform each other of features which they believe are important to their particular goal.

3. Have the teams supervise each other during optimisation.

4. Integrate performance information into the design software in order to either/both:

   (a) Inform designers of the effect of their changes on the other team's optimisation goals.

   (b) Suggest to designers features of the design which have little effect on the other team's optimisation goals (similar to the second option, feature ranking).

The first option is what is performed in practice. Each team works effectively in isolation from the other, waiting on the other to perform their next iteration of design optimisation (aesthetic or functional). Some information from a team such as "please preserve feature x" may be supplied to the other team but such an approach can severely stifle the progress and novel creativity of a team, which has a direct effect on the length of the design-to-market phase of the product development and the quality of the final product.

The second option tries to relieve this stifling effect by being less restrictive and giving the other team some indication of which features they should try adjusting before other features. Unfortunately human designers are prone to focusing on only a small subset of features which they know are important, and thus may be over-defensive of a small set of the most important features, while less caring of other features. Further we cannot feasibly expect a human designer to rank every single feature of a design.

For human designers the third option can be an irritating prospect. Especially as it is likely that, for example, a designer from one team may be very defensive about design decisions they made previously and try to preserve these through the supervision of the other team's optimisation.

The fourth option appears best as:

1. It succeeds in introducing the optimisation goals of the other team into the design process of one team.

2. It avoids human bias towards particular features, giving a balanced appraisal of the effect each feature has.

3. It is far less invasive than having another human designer supervise you and its advice can be more easily disregarded allowing creative freedom.

4. Automobile designers working with the Honda Research Institute have requested such a system.

A computational design assessment system is thus a good solution for us to investigate.

The benefits of introducing the performance information relating to other team's goals into the design process of a particular team impact on both industry and consumer:

- From an industry standpoint, improving on the iterative design cycle method would improve the profitability of a design by bringing it to market sooner.

- From a consumer standpoint, if both teams can work with greater respect to the others goals, then automobile designs of superior quality can be found.

Assessing the aesthetic quality of an automobile is not a trivial task and it is unfeasible to expect that a computational system capable of doing this will be available in the near future, although work has been conducted in this area [26, 78, 88]. This is simply due to the subjective nature of aesthetic quality. Different artists often favour vastly different designs, and even if that were not the case, aesthetic quality is a challenging function to understand and a very difficult property to quantify.

Fortunately, for assessing functional performance of an automobile (e.g. the aerodynamic properties) systems often already exist. Computational Fluid Dynamics (CFD) is the name

given to the family of particle simulation software which can be used for, amongst other things, estimating to a high fidelity the aerodynamic performance of a design (which otherwise must be measured using a physical experiment such as a 'wind tunnel' experiment as shown in figure 2.1). In the second image of figure 2.2, we see the rendering of the airflows around a vehicle design, with the velocity at each point in each flow being indicated by the heat-map colour. A popular open-source CFD suite is OpenFOAM [36], although many high quality commercial options exist.

Despite CFD being a simulation-based approximation of a physical process, the results are very good, although they depend on a number of parameters. Of particular importance is the construction of the cell-mesh used for the simulation of the particle movements and pressures etc. In the top half of figure 2.3 we can see the 2D mesh around a vehicle design. Another important issue affecting CFD fidelity is that is is an iterative process which requires the simulation to converge, which can often push the time required for a single modest-fidelity simulation into hours, and for a high-fidelity simulation days may be required, perhaps weeks.

The artistic designer (and the organisation they are working for) cannot afford to wait an hour after each change to see what the predicted aerodynamic effect is. Nor can they afford to wait n hours while the effects of n modifications on aerodynamic performance are assessed.

Shortly we will learn of surrogate assisted design optimisation. From this, we will conclude that a good option is to create an approximation of the functional performance (e.g. aerodynamic drag) which is computationally quick to perform. We will expect to trade accuracy for speed and so the approximation we aim for will be considered an even lower level approximation of the physical process than CFD itself is.

Many automobile manufacturers have a long history of design and have in recent years been accumulating designs and their functional performance scores during iterative design processes. We will consider using these to discover the knowledge required by our approximation.

2.2 Solutions for automated design processes

We have identified that the nature of aesthetic optimisation implies that that the process is human-driven. Despite this, it is of interest to look towards automated design optimisation systems and see whether a similar problem is encountered in that field, and if so, how it is dealt

Figure 2.1: Photo of a wind-tunnel experiment involving a scale model of an aeroplane (source: "Windkanal" by JeLuF - http://en.wikipedia.org/wiki/Image:Windkanal.jpg. Licensed under Creative Commons Attribution-Share Alike 3.0 via Wikimedia Commons).

with effectively.

In the field of automated design optimisation, evolutionary optimisation techniques [8] (including genetic algorithms, evolution strategies, genetic programming) are some of the most studied and used for complex functional objectives with multiple modalities.

Assuming some 'fitness' function to be minimised, evolutionary algorithms generally involve searching through a solution space using some stochastic perturbation of a population of solutions and evaluating each solution by this fitness function.

While powerful methods, evolutionary optimisation techniques tend to require a large number of iterations (termed generations) before they converge. The speed of the overall process therefore depends heavily on the computational speed of the fitness function [60]. A slow fitness function results in a slow convergence, as the optimiser effectively pauses at each fitness evaluation, waiting for fitness functions to compute.

Non-evolutionary optimisation methods such as stochastic gradient descent [130] and simulated annealing [69], which effectively work on a single solution, have been successfully applied to aerodynamic optimisation problems (including others with computationally intensive fitness functions). Evolutionary optimisation methods have also been successfully applied to such problems [98] but these are typically seen as unsuitable for more complex problems because of the extra computational overhead required for optimising populations of multiple solutions.

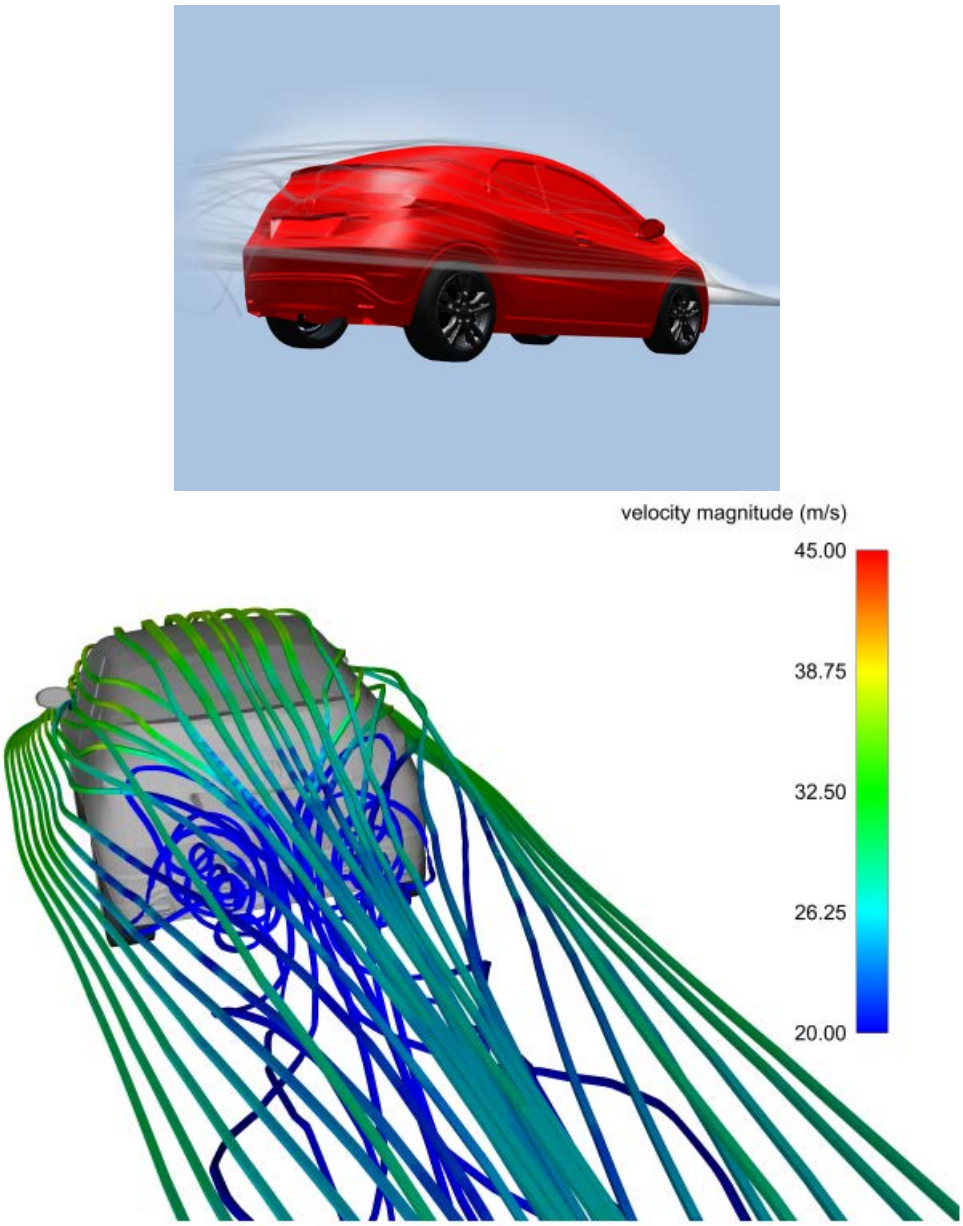


Figure 2.2: Graphical renderings of simulated airflow around the Honda Civic (European) design. In the first image, the airflows have been visualised such that the viewer can better understand what is meant by the term. In the second image, facing the rear of the design, heat-map colours indicate the velocity of particles at each point of the airflow following the results of a CFD simulation (source: kindly provided by the Honda Research Institute Europe, Offenbach, Germany).

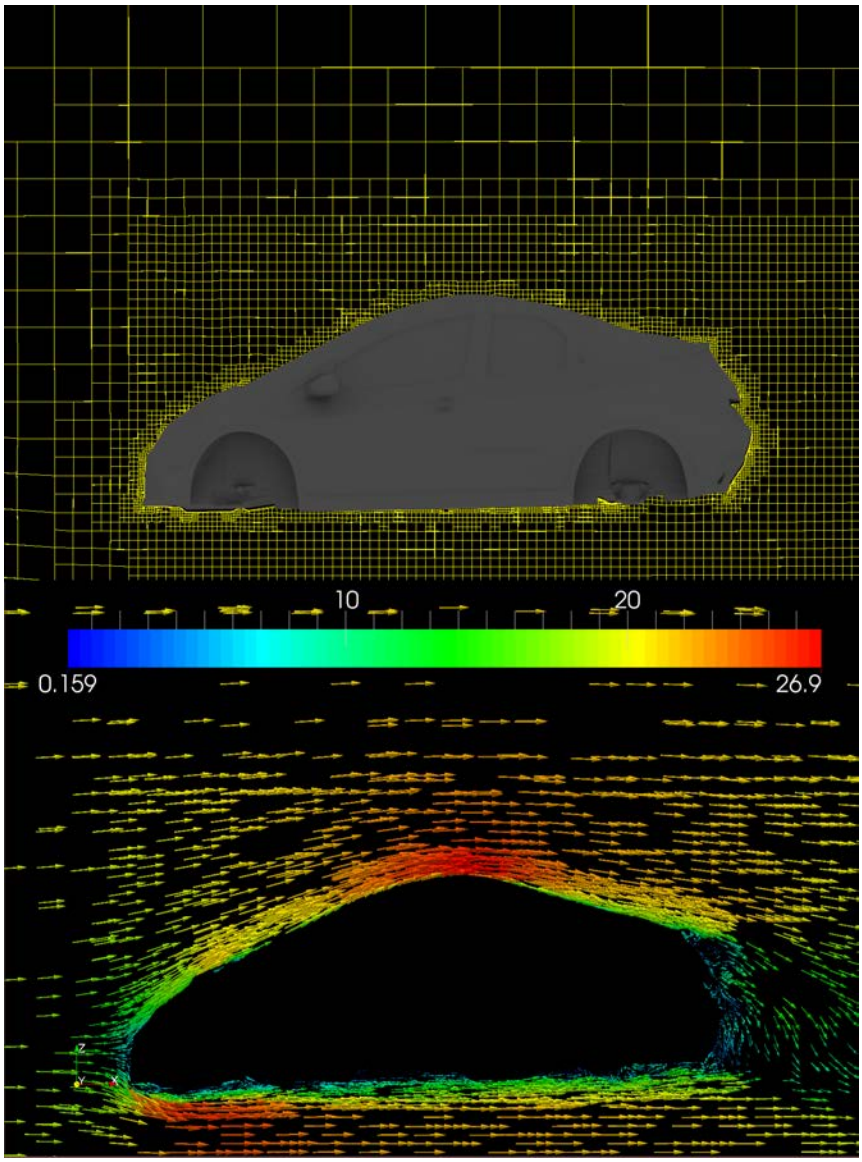


Figure 2.3: Illustration of CFD mesh and CFD solver in action. The top part of the image shows in yellow the 2D CFD mesh generated for a vehicle design (the vehicle design, a modification of the Honda Civic (European) used later, is rendered purely for illustrative effect). A CFD solver operates only on this yellow mesh and in the bottom half of the figure we see the simulated velocity of particles over the automobile's surface represented by coloured arrows.

2.2.1 Surrogate assisted design optimisation

Driven by this problem, research has been directed in recent years towards measures that can be taken in order to reduce the time required for the whole evolutionary optimisation process. The most popular approach is the use of a computationally cheap 'surrogate' fitness function in place of the true fitness function [60, 61].

Usually, this involves finding a function (often through Machine Learning methods) which approximates as closely as possible, the true fitness function. So long as this surrogate function can effectively rank the fitness of designs, it will be useful [119]. Then this surrogate can be used instead of the true fitness function in some proportion of the fitness calculations during the optimisation. This approach of using a surrogate function has been successfully applied to engineering design optimisation problems [126, 138, 34, 68]. The main effect of deploying the surrogate is a speed-up of convergence in terms of computational time.

Incorporating surrogates in the evolutionary optimisation process

The proportion and regularity of fitness calculations to be estimated via the surrogate has been termed model-management [45, 62, 32]. Broadly speaking, there are two schemes to this: individual and generational. Under generational model management, whole generations are evaluated by the surrogate for some proportion of the generations (i.e. it is determined on a generational basis), whereas under individual model management, it is determined on an individual basis, independent of the other individuals. In [45, 62, 32] the authors adapt the proportion of individuals evaluated by the surrogate according to the estimated accuracy of the surrogate. The results are positive, seeing an improvement in convergence speed.

A popular approach is to build local surrogates as they are needed depending on the spatial location of the individual to be evaluated [146, 101, 102, 32]. This can be realised by non-parametric methods with fast training phases such as nearest-neighbour regression (as in [32]) or radial basis functions (as in [99]). In [146] the authors combine a global Gaussian Process surrogate with local radial basis function surrogates. At each stage of the evolutionary algorithm, the global surrogate filters the population to find the most promising individuals which then each undergo gradient-descent optimisation using the local surrogates. This approach has been more recently integrated into a memetic algorithm framework where individuals of the population are

optimised individually using local surrogate functions but later collaborate with other individuals to form solutions which are then evaluated by the original fitness function [55].

The authors of [63] generate surrogates in an interesting manner on each iteration. They apply k-means clustering to the individuals at each generation, clustering the data. Then they evaluate the k individuals closest to each cluster centre by the original fitness function. Using the k individuals and with their now-known fitness scores, they construct an artificial neural network ensemble and evaluate the remaining individuals using this as a surrogate. They associate areas of high error in the ensemble with samples for which there exists high variance between artificial neural network predictions.

Using multiple surrogates

In [41, 80] the authors demonstrate the power of combining multiple surrogates into an ensemble-surrogate. In particular, in [41] an ensemble is constructed of a polynomial response surface, a Kriging model and a Radial Basis Neural Network. By looking at the variance between estimated outputs of each of the three surrogates they are then able to assess the relative accuracy of the surrogates and generate weights for each (for example, if a surrogate produces output very different from the other two, it can be judged to have poor accuracy). The weighted sum of the surrogates is then used as the ensemble output.

In [80] the authors use a simple average of surrogate outputs for their ensemble output achieving good results. They also develop what they call a 'multi-surrogate' framework, where each individual is optimised locally in a generation according to each surrogate independently (so with M surrogates, a generation involves M local optimisations of each individual). The best solution is then carried forward through the optimisation process.

Optimising surrogates for the optimisation process

Optimisation of surrogates is investigated in [57] where neural networks are employed as surrogates in evolutionary design optimisation tasks. The optimisation of the surrogates' architectures is performed such that the neural networks are capable of being able to learn different problems of a common domain fast and with high accuracy. Application of the framework to a turbine blade optimisation demonstrates significant improvement over neural networks which have not had their structures optimised.

In [123] the authors investigate the impact of co-evolving the surrogates along with the evolutionary optimisation. Their framework consists of three components which each evolve independently: 1) the solutions to the overall optimisation problem; 2) the surrogates themselves; and 3) the trainers of the surrogates. Importantly, the surrogate trainers are evolved to cause the most variance between the surrogates functions. The results on randomly generated test functions are encouraging.

CFD specific examples of surrogate deployment

It is worth noting that there exist some CFD specific approaches to surrogacy [100, 35, 77]. In [77] the authors use as surrogate a low-fidelity physics based function to approximate the aerodynamic function of the design. In [35] they take a different and very interesting approach. As CFD is an iterative process, they are able to interrupt the process after a low-number of iterations. By training surrogate functions on these partially converged results, and then giving as input the partially converged results, they are able to cut the computational cost of optimisation algorithms substantially.

Alternative uses of surrogates

Surrogate functions aren't always used solely for the task of estimating fitness during the evolutionary search. In [66] the authors perform mathematical analysis on artificial neural network surrogate trained during the optimisation process in order to determine the sub-space with the greatest impact on the fitness function. They restrict the optimisation to this subspace reducing the computational effort required. The authors of [112] look at surrogates of CFD in optimisation, but interestingly one of the main uses of the surrogate is to perform sensitivity analysis on the design attributes.

The authors of [109] suggest using a surrogate as part of a probing technique, allowing them to generate a huge number of offspring in each generation and carry only those who are most promising according to the surrogate over to the next generation. In [114, 113] the authors suggest 'informed operators' as augmented versions of the standard evolutionary operators mutation and crossover/combination. These suggested operators very generally involve performing a large number of the original operations, and then selecting the best individual constructed according to a surrogate.

Side-effects of using surrogates

Negative side effects do occur, and depend largely on the level of dependence upon and accuracy of the surrogate. For example, over-reliance on a surrogate function which ranks the fitness of design incorrectly could lead to poor solutions and result in a slow down in convergence and an increase in computational time. The authors of [80, 79] term this the *curse of uncertainty*.

Interestingly, a surrogate function with incorrect ranking isn't always a bad idea. Sometimes the errors can cause a search to enter interesting regions of the design space which it may have otherwise not encountered, resulting in the discovery of potentially novel solutions. The authors of [79] term this the *bless of uncertainty*. They utilise a separate surrogate (alongside an ensemble-surrogate) constructed of a low-order polynomial regression model termed the 'smoothing' surrogate. This smoothing surrogate is designed to take advantage of the *bless* as its simplicity makes it inaccurate and this causes optimisations to enter new areas from which the main ensemble-surrogate is to learn from and improve. These new areas may contain interesting designs and help the optimisation to find global minima/maxima faster.

2.3 Related problems in the literature

We now describe other pieces of research relevant to the problem central to this thesis.

2.3.1 Building a regression function to aid engineering surface design

There currently exists no single method of generating a surrogate regression function suitable for guiding the design choices of a human aesthetic designer. Indeed, the designers at the automobile firm Honda do not currently use such a system, largely because no system has proven to be accurate enough to be of use in this manner. Despite this, and its possible benefits, research into methods of data mining from engineering design data for the purpose of aiding human design is very scarce.

The authors of [42] look at a related problem where they use a data set of engineering surface designs in order to generate a human readable decision tree of rules that can be used to make design decisions. Their approach, which we shall return to in the next chapter, involves a reference model and representing samples as offsets from this reference model. We will later see that this approach to converting a data set of surface designs is not suitable to our problem.

The authors also show how the knowledge extracted can be embedded into the surface design representation in order to convey useful information to the human designer. In [44] they have extended this work to allow for the discovery and assessment of 'design concepts', demonstrating their approach on an automobile design problem in order to learn about the relationship between design concepts and aerodynamic properties.

2.3.2 Guiding human design decisions

Of particularly high relevance to our work is [92] in which the authors build on the field of surrogate assisted design optimisation in order to develop a 'designer assistance system'. The aim of the paper is to show how the designer assistance system may help convergence speed in the human driven design process. They opt for a simple experimental setup: a shape matching task where the user is given a rectangle made up of 16 control points and is given the task of increasing a performance score. The performance score involves a hidden and unknown shape, and is calculated as the overall similarity of the users shape to this target shape. When the user selects any control point, the effect on performance of moving that particular point in a discretised set of possible directions is generated via an artificial neural network and displayed to the user visually. The user can then make local modifications based on this. It is shown that human designers produce better designs using this information feedback than they would if simply being shown the design performance after each design change.

The findings support the significance of our work as they show potential for our approximation to speed-up the engineering surface design process through guidance of aesthetic design decisions. Although clearly this task is very simplified and avoids the challenges of real world engineering surface design problems that we will have to deal with.

2.3.3 Feature analysis

On a more general front, some more works can be found which look at the problem in terms of finding the most relevant features in the design data.

The authors of [97] use self organising maps in order to find groups of similar designs and gain insight into the relationship between attributes and performance, and the trade-offs in the domain. A second self organising map is built based on the first which clusters the design variables such that their purpose with regards to the functional performance may be understood.

They demonstrate their approach on supersonic wing and supersonic fuselage designs. In [22] the authors extended this research by also applying the analysis of variance technique in order to identify the most important design parameters (with each approach making up for the others disadvantages). Again, the research was applied to a supersonic wing design.

In [33] the authors investigate the discovery of free-form features on surface designs for easier feature-based manipulation (here the authors define a feature as a group of geometric attributes which combined have some kind of semantic design meaning). Their approach involves then classifying these features by functional meaning. They later extend their approach to classifying features by aesthetic meaning.

These themes of representation generation and feature extraction return later in this thesis. We will look at the challenging properties of our data set and suggest novel solutions to these tasks which build upon the failings of approaches already in the literature.

2.3.4 Knowledge management in engineering

Since the early nineties, Knowledge Management (KM) has become a research field with strong implications for engineering design. KM has become a broad term covering processes of exploiting an organisations knowledge resources to achieve its objectives [4, 90]. Amongst the organisational objectives of KM are improved performance and innovation [46], which can lead to shorter design cycles and superior designs. Interviews, collaboration and the sharing of knowledge are key to KM approaches. Computational KM approaches include tools that facilitate knowledge accumulation and distribution, such as intranets/wikis, discussion forums, expert systems, knowledge-bases and decision support systems [25].

The application of some KM approaches to engineering is known as Knowledge Based Engineering (KBE) [120, 67]. A typical KBE systems may be a database of previous product parts that can be retrieved and used in future design cycles. Thus KBE approaches differ in how they encode engineering data in a way that it can be usefully retrieved later. The authors of [120] suggest that incorporation of KM into these KBE work-flows is a topic that requires careful consideration, suggesting a framework for this purpose. A number of established KBE solutions now exist on the market and many are currently listed at [140].

The authors of [10] focus on the reuse of engineering design knowledge, in particular process-knowledge, product-knowledge and task-knowledge, rather than geometrical data. They suggest

a framework for reusing knowledge, incorporating best practice, design rationale, and knowledge-based support. Key to their approach is a 'common design data model', which is used both for storing knowledge and also reusing knowledge. More recently this approach has been extended to Product-Service systems design [11].

An important topic in KM is that of knowledge representation. The authors of [21] review the development of this field with regards to product design and suggest the trends that may continue into the future.

2.3.5 Concurrent engineering design

Briefly mentioned in the introduction of chapter 1, concurrent engineering is a product development methodology based on concurrency of development tasks [73]. It represents an alternative method to the traditional sequential development method (the 'waterfall' method),whereby each development task is performed in isolation from the previous and next tasks, and no task is ever revisited without winding back all the tasks that occurred since its completion.

Concurrent engineering design employs cross-functional teams; concurrent design of parallel subsystems; sharing of new design information; and project effective project management [73]. The core motivations behind concurrent engineering design are that all development processes should be taken into account, and design issues should be identified as early as possible so that their impacts are mitigated.

Through the use of concurrent engineering, firms can expect to see such advantages as reduced time-to-market[111]; reduced development costs [111]; and improved product quality. [73].

Concurrent engineering has now become popular in engineering [83], with the European Space Agency being amongst the methodology's proponents [2]. Presently, much of the active research goes into tools to aid concurrent engineering such as the modelling tool described in [27] for use in space mission design tasks.

2.3.6 Interactive evolutionary optimisation

It is worth noting a family of evolutionary optimisation approaches described as interactive. Interactive approaches represent evolutionary optimisation algorithms designed with fitness functions in mind that could be not be modelled or evaluated by computer (such as aesthetic quality) [133]. The idea is simple: the standard evolutionary operators are applied to a population of solutions

in order to arrive in new generations, but rather than evaluating solutions by a fitness function, they are presented to a human user who must then evaluate them and choose the parents of the next generation. The problem with these approaches is that they quickly, and understandably, lead to human fatigue [133] which in turn degrades their effectiveness.

Also important are the group of *interactive multi-objective* optimisation approaches [94]. Under this approach, the automated optimisation is usually optimised against a set of objective fitness functions. The human's user can then be guide the optimisation through modification of 'preferences', at certain intervals in the optimisation process [86, 127]. These preferences determine how strongly each of the objectives should be weighted during automated optimisation and this method is shown in [9] to result in superior multi-objective optimisation when compared to a totally automated process based on fixed preferences.

2.4 Machine learning for estimating functional performance

Inspired by surrogate assisted design optimisation and its deployment of a computationally cheap surrogate function, we consider using a computationally cheap 'approximation' function of the true functional performance evaluation function, using its output to inform the aesthetic designer.

A major difference between our intended deployment of an approximation function and the deployment of a surrogate in optimisation, is that we will seek to use our approximation function at all times: constantly estimating the current design's performance and analysing the estimated effects of individual attribute changes. Contrast this to surrogate assisted design optimisation, where the use of the surrogates can be sparse and never replaces all fitness evaluations.

The generation of such an approximation function will require the selection of some suitable function generating process, such as those from Machine Learning, including: response-surface-methodology; Support Vector Machine; Gaussian Process; Neural Networks etc.

Very generally, machine learning [13, 95] is concerned with a group of algorithms which, given a data set of some input patterns and their respective output patterns sampled from a function, are expected to learn rules which apply well to input and output patterns **not** contained in the input set. We call this latter set the 'unseen' samples.

How well the machine learning algorithm does this can be termed its level of 'generalisation'. Generalisation of an algorithm is problem dependent and so we might say a particular machine

learning algorithm has good generalisation for a particular problem. This means that it can predict the output patterns for unseen input patterns well for this particular problem. Good generalisation is a desirable property of any machine learning algorithm.

This raises the issue of how one assesses the performance of a machine learning algorithm. In section 2.6 we will look at some common approaches of empirically assessing the performance of the generated approximation function, but for now we look at formulating the machine learning problem.

We denote the approximation function as $\hat{f}$: a function which maps input pattern x to output pattern $\hat{y}$; $\hat{f} : x \to \hat{y}$. Under this notation, $\hat{f}$ is assumed to be an approximation of some observable function f, which itself maps x to an output pattern y; $f : x \to y$. Thus $\hat{f}$ outputs an approximation of y, $\hat{y} = y + \varepsilon$ where ε is some error value (which may itself be a function of x).

The machine learning algorithm is the process through which $\hat{f}$ is generated. A machine learning algorithm which arrives at some approximation $\hat{f}$ of the observable function f, invariably requires a set of N sample patterns, D, from which to learn.

Each sample pattern consists of a tuple $\{x_i, y_i\}_{i=1}^{N}$. Within D we denote all N input patterns by $X = \{x_i\}_{i=1}^{N}$, and all N output patterns by $Y = \{y_i\}_{i=1}^{N}$.

There are many machine learning algorithms and some are deterministic while others are not. In both cases, but especially the non-deterministic case, it is good practice to run the machine learning algorithm a number of times and evaluate its performance using a set of data put aside earlier prior to training. This 'unseen' set of samples can then be used to evaluate the resulting approximation $\hat{f}$. For this reason it is common practice to divide D into a *training set* D_{TR} of N_{TR} pattern tuples and a *testing set* D_{TE} of $N - N_{TR}$ pattern tuples.

It is important that the machine learning algorithm never encounters any of the testing set tuples in D_{TR}. Providing this holds and D_{TE} spans the space of possible patterns well, the level of generalisation of the approximation $\hat{f}$ can be estimated (which is dependent on the generalisation quality of the machine learning algorithm).

2.4.1 Differences between our approximation and a surrogate

Our desired approximation is particularly distinguishable from surrogate assisted design optimisation when we consider the training set D. For surrogate assisted design optimisation, it is

possible to generate D using a space-filling sample plan scheme such as Latin Hyper Squares (LHS) [34]. Because optimisations will occur locally to the initial design, it is logical to specify this sampling in a spatially local area, knowing that a stochastic optimisation will generate samples along any direction in the space of possible designs for evaluation.

Contrast that with our task, where a non-stochastic human designer will have certain aims with regards to aesthetic design and it will be clear that it is hard to predict where estimations will be required in a large space of possible designs. One technique that can be used in surrogate assisted design optimisation is to regenerate a sample plan and retrain the estimation function when a design leaves a high-accuracy region of the space of possible designs. This is unfeasible for us, as this would cause the design assistance system to be unavailable for hours, maybe days, when the designer causes the design to leave such a local region and triggers this retraining process.

Another difference to consider, is that the stochastic optimisation is likely to approach unfeasible or unrealistic designs. This means that a uniformly sampled D (e.g. using LHS) may prove quite useful for an evolutionary optimisation problem. A human designer is far less likely to approach these forms of design, and so such a D would contain a high proportion of unimportant samples and thus wasted functional performance evaluations.

2.4.2 Machine learning applied to our problem

When applying machine learning to the task of generating an approximation to act as our functional performance estimation system, we encounter a number of obstacles:

1. **Large design representations.** Engineering surface designs can be very complex geometric structures. In domains such as automobile design they can vary in many, many ways and each of these ways is associated with one or multiple features. These truths mean that each design's input pattern must be large enough to correctly represent the small variations in surface design. At a minimum, for any two different designs there must exist some attribute that distinguishes them sufficiently and in a way that a machine learning algorithm can learn from. This property makes dimensionality reduction of the surface design's representation challenging.

2. **Available training data.** Machine learning algorithms generally suffer from the *curse of*

dimensionality [12, 99]. The curse broadly states that as the dimensionality of an input domain increases, the number of training data samples required by a machine learning algorithm increases exponentially. As previously noted for our motivating example, it is a slow process to generate training data by virtue of the computationally costly CFD function. It is unfeasible to imagine we will ever have enough training data for a problem with the original design representation.

3. **Appropriateness of training data.** As noted in section 2.4.1, a uniform sampling plan is inappropriate and even wasteful. Somehow we need to generate a sampling plan that samples only in regions of the design space which will be useful to our machine learning algorithm and approximation use. This implies that we somehow find design space regions which are interesting to the aesthetic designer.

4. **Learning a large domain.** Our approximation is expected to be trained and ready to aid the aesthetic designer regardless of whatever changes they make. It is possible that the designer will make substantial changes to a design which almost completely transform it, taking it into a distant region of the space of possible designs. This implies that our computational design assessment system needs to be able to estimate for designs across a very large domain. Coupled with the fact that functional performance is often highly non-linear, this task becomes very challenging for machine learning algorithms.

2.4.3 Finding a suitable source of training data

At the end of our motivating example in section 2.1, we noted that many automobile manufacturers have a long history of design and may have in recent years been accumulating designs and their functional performance scores. To elaborate, the automobile manufacturer may have digital representations of many previously considered designs with their associated aerodynamic performance scores. These designs may have resulted from a mixture of human driven optimisation and computer-driven optimisation and could be deemed as historically interesting.

In this thesis we consider using such an accumulated data set of previously interesting designs. By deploying such a data set, we greatly reduce the impact of the second obstacle in section 2.4.2 because we have a large source of data to hand. We also greatly reduce the impact of the third obstacle because each sample in this data set was previously interesting to the design process

and therefore likely to be, by inference, potentially interesting to the aesthetic designer in future design processes.

Also, obstacle 4 is somewhat alleviated as this data set, while still spanning a large space of designs, will be more concentrated into the areas of interest, meaning that the machine learning algorithm need only learn a smaller subspace of the design space.

Unfortunately, such a data set, while helping obstacles 2 and 3, also has problematic properties.

Non-uniform surface representation

As we will discuss in chapter 3, engineering surface designs are usually represented, in their rawest form, as a collection of geometric primitives. Common formats will reduce the geometry to 3D faces, often triangles. One very common format like this is the STereo Lithography (STL) format, which almost all engineering surface design software is able to output. It is possible to extract the 3D points from all the faces and remove redundant points to form a 'point cloud'.

Formats such as STL are very flexible, and provide very few rigid format constraints. This means that the number of faces (and thus 3D points in the point cloud) can vary between surface designs. Further, the order of faces (and again the 3D points) in the surface design can also vary.

In practice, these formats vary massively. For example, some automobile designs have features such as rear-spoilers which other automobile designs lack entirely (and so the geometric primitives required for this feature is also lacking). Further, it is common for the number and order of primitives to change during the design process: an artist may decide that more detail is required in a specific surface region, and increase the resolution of geometric primitives in that region.

We cannot use such data easily because machine learning algorithms generally assume that every input sample has the same number of attributes and they also expect that the ith attribute in any two design representations describes the same information (in our case the same location on the automobile's surface).

We need to deal with these problems through some layer of preprocessing in order to prepare our data set so that it is usable with machine learning algorithms.

Large design representations

This property of the data set is an aggravation of the first obstacle in section 2.4.2. Engineering surface designs stored as geometric primitives (like in STL) tend to consist of a huge number of geometric primitives. One of the driving forces behind this is that these formats can never *truly* represent a curve, and must discretise the curve into a number of tiny segments.

To illustrate the size of the design representation, let us consider a concrete example: in the next chapter we will generate a data set based on an automobile surface design in STL format. This formatted surface design consists of $1,129,239$ individual triangular faces. If we strip this surface design down to its 3D points only, we are given $555,397$ 3D points ($1,666,191$ individual attributes).

The size of this representation is indicative of the size that many designs in the automobile domain will have (and this also applies to other engineering surface domains). Of course, any machine learning algorithm we use is therefore susceptible to the curse of dimensionality.

The curse of dimensionality is a common problem in machine learning, and as a result there has been a lot of research activity in the direction of reducing dimensionality. Dimensionality reduction will be given greater focus in the next chapter. But for now we must make some note that for our problem we can expect the state of the art approaches to dimensionality reduction to struggle with our data set. The reasoning behind this is thus:

- The aesthetic designer will seek a high degree of design flexibility. Any representation with reduced dimensionality must be able to represent the relevant (i.e. those with impact on functional performance) differences between designs in this representation.

- Given the space of possible designs S: for some subspace $A \subseteq S$ we can expect that a particular subset of attributes will be commonly modified by the aesthetic designer. In another subspace of $B \subseteq S$ it may be another entirely exclusive subset of attributes which are commonly modified. Any representation with reduced dimensionality would need to contain the attributes relevant in each of these subspaces, resulting in a representation larger than that needed for each subspace A or B.

- A logical approach to dimensionality reduction could restrict our representation with reduced dimensionality to only those attributes with some effect on the functional performance. But for many functional performance scores, including aerodynamic scores, we

would run into a very similar problem: for some subspace $A \subseteq S$ one set of attributes may affect functional performance while in another subspace $B \subseteq S$ a completely different set of attributes affect the functional performance (this is due to the presence of attribute interactions [59, 43], attributes with different effects depending on other attributes). Yet again, this means that the resulting representation would be larger than that needed for each subspace A or B.

Lack of a single design space focus

This property of the data set builds on the fourth obstacle in section 2.4.2. As described earlier in this chapter, surrogates have been used in computational optimisation frameworks successfully. Key to their successful deployment is the understanding that the surrogate fitness function need only be called in the subspace of S which is being searched within [34]. This space is usually quite limited when compared to the size of the space of possible designs and is usually contiguous. Should the optimisation enter into completely unsampled subspaces of S, a new sample plan in the unknown subspace could be generated and used to update the surrogate [60, 62].

Generating a regression function to act as a surrogate in this way requires a relatively small and concentrated data set of training samples. This process is made easier because the more concentrated sample set can be taken from a smaller region of the space of possible designs and thus is able to represent a smaller 'window' of the function being learned. In this 'smaller window' of the function it is easier for a machine learning algorithm to learn the relationships between inputs and outputs and thus more accurately estimate the function's outputs [146, 34].

In generating our approximation, we do not have this advantage, as we seek to build a regression function which performs well across the whole space of designs which may possibly be encountered, $E \subseteq S$, which is unknown. Our choice of data set gives some indication towards E in the previously interesting designs it contains, but it also contains many relatively small and highly sampled regions of the design spaces (resulting from different optimisation and design processes) and is non-contiguous.

Across the space of possible automobile designs the relationship between aerodynamic drag and each attribute can vary quite wildly, with some attributes having a positive effect in one region and a negative effect in another. This is quite a common problem for functional performance functions used in engineering [34].

In effect, the 'window' of the function being learned is broken into many disjoint windows which each have very different input to output relationships. This will make the task of learning a regression function to act as our approximation challenging.

2.5 Mathematical formulation of the problem

We now take the opportunity to build on the notation provided so far. We assume that we have a data set D consisting of:

- N STL-format surface designs, $X = \{\mathrm{x}_i\}_{i=1}^{N}$, where the ith sample $\mathrm{x}_i = \{x_{i,1}, x_{i,2}, ..., x_{i,d}\}$, d_i is the number of 3D triangles forming the ith design's surface (which can differ for different i), and the jth triangle$x_{i,j} = \{\vec{x}_{i,j,1}, \vec{x}_{i,j,2}, \vec{x}_{i,j,3}\}$ is represented as 3 3D points, such that $1 \leq k \leq 3$ and $\vec{x}_{i,j,k} \in \mathbb{R}^3$.
  - The order of the 3 points forming each triangle indicates the front of that triangular face such that the cross-product $\vec{a} \times \vec{b}$ where $\vec{a} = \vec{x}_{i,j,2} - \vec{x}_{i,j,1}$ and $\vec{b} = \vec{x}_{i,j,3} - \vec{x}_{i,j,1}$ produces a vector $\vec{c}$ which when normalised equals the normal vector of the jth triangular face of the ith design, $\vec{n}_{i,j}$.
  - The order of triangles in each surface design can differ: attributes $x_{i,j}$ and $x_{k,j}$, where $k \neq j$, do not necessarily refer to the same piece of surface design information.
- N functional performance values, $Y = \{y_i\}_{i=1}^{N}$, where $y_i \in \mathbb{R}$, and $y_i = f(\mathrm{x}_i)$ where $f(\mathrm{x}_i)$ is the functional performance function of the ith design (we focus only on one functional performance score in this work).

X consists of designs:

- taken from the space of possible designs S.
- which have **not** been continuously and uniformly sampled from S.
- which have been taken from an interesting subset of S, the encounterable subspace $E \subset S$.

Y consists of values

- dependent on the input designs X by way of $f : \mathrm{x_i} \rightarrow y_i$.

- non-linearly dependent on the attributes of input designs.
- which depend on a subset of the attributes making up each design.

We seek to construct a regression function $\hat{f}(\mathrm{x}) = y + \varepsilon$ where ε is some error term, to approximate $f(\mathrm{x})$ based only on our data set D.

Our primary objective is to arrive at a regression function $\hat{f}(\mathrm{x})$ which can be used as an approximation function to guide the decisions of a human designer. This should ultimately reduce the time-to-market time required for a engineered product and/or improve the quality of the marketed product. Our second objective is to minimise ε across the space of possible designs S, and in particular across the encounterable subspace of possible designs $E \subseteq S$.

The regression function $\hat{f}(\mathrm{x})$ should compute in a small amount of time (seconds) so as to not severely interrupt the human designer's design process.

2.6 Assessing approximation function performance

In this section we look at the performance metrics which we will be using in this thesis to evaluate and compare regression functions for use as our approximation.

2.6.1 Accuracy metrics

Here we describe three very common performance metrics which focus on the accuracy of the regression function in terms of the difference between estimated and true, target output values.

Mean squared error

This is the average of the squared error, where error is the difference between the estimated output for the test set and its target output. This is one of the most well established metrics when optimising as its interpretation is simple: the lower it is, the better.

Unfortunately it isn't easy to assess what "good mean squared error" really is as there is no upper-bound on the metric. The mean squared error score can thus only be used to compare regression functions where the same training and testing data is being used with each regression function (as is the case here). It becomes easier to interpret in a normalised data set, but is still not as easy to interpret as the other measures we will look at.

Mean squared error: $MSE(\hat{f}) = \frac{1}{N_{TE}} \sum_{i=1}^{N_{TE}} (\hat{f}(\mathrm{x}_i) - y_i)^2$

where N_{TE} is the number of samples in the testing data set D_{TE}, and x_i and y_i are the ith input and output in D_{TE} respectively.

r^2 correlation coefficient

This is the correlation between the estimated outputs for the test set and its the target outputs. This score is particularly useful when one considers that error isn't as important for our problem as ensuring that if design x_i is worse than design x_j, then the estimation results in the same conclusion (the Ranking score in section 2.6.2 will focus more on this property).

The higher this score is, the better and it is easy to assess "good correlation coefficient" as it is upper-bounded by 1 and lower-bounded by 0.

Correlation coefficient: $r^2(\hat{f}) = \left(\frac{\sum_{i=1}^{N_{TE}} (\hat{f}(\mathrm{x}_i) - \bar{f}(\mathrm{x}))(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{N_{TE}} (\hat{f}(\mathrm{x}_i) - \bar{f}(\mathrm{x}))^2} \sqrt{\sum_{i=1}^{N} (y_i - \bar{y})^2}} \right)^2$

where $\bar{f}(\mathrm{x})$ is the average output of $\hat{f}$ over all the samples in D_{TE} and $\bar{y}$ is the average functional performance in D_{TE}.

Percentage error

This is the squared error as a percentage of the range of target output values in the data. Clearly, it is easier to assess "good percentage error" in comparison to MSE as it is a percentage.

Percentage error: $PCE(\hat{f}) = \frac{100}{N_{TE} * (y_{max} - y_{min})} \sum_{i=1}^{N_{TE}} (\hat{f}(\mathrm{x}_i) - y_i)^2$

where y_{max} and y_{min} are the largest and smallest functional performance value in D_{TE}, respectively.

2.6.2 The ranking metric

For a specific problem like this it is important to evaluate algorithmic approaches not only in terms of accuracy but also in terms of how correctly they rank. We already hinted on the logic behind this when discussing the r^2 correlation coefficient score: an aesthetic designer is more generally interested in whether a design modification worsens, betters, or does not affect performance than what the exact performance value is.

For this purpose we propose a simple ranking metric: Prediction of Performance Change. This score computes the percentage of such predictions (worsens, betters or no-effect) correctly

assessed:

Prediction of performance change: $PPC(\hat{f}) = \frac{1}{N_{TE}^2} \sum_{i=1}^{N_{TE}} \sum_{j=1}^{N_{TE}} \pi_{i,j}$

$$\text{where } \pi_{i,j} = \begin{cases} 1 & \hat{f}(\mathrm{x}_i) > \hat{f}(\mathrm{x}_j); y_i > y_j \\ 1 & \hat{f}(\mathrm{x}_i) < \hat{f}(\mathrm{x}_j); y_i < y_j \\ 1 & \hat{f}(\mathrm{x}_i) = \hat{f}(\mathrm{x}_j); y_i > y_j \\ 1 & \hat{f}(\mathrm{x}_i) > \hat{f}(\mathrm{x}_j); y_i = y_j \\ 0 & \textit{otherwise} \end{cases}$$

The third and fourth conditions are specially chosen such that if the user is given an incorrect 'does not affect performance' prediction for a particular design change, but the change results in a performance improvement (here **minimising** the functional performance score as is the case with aerodynamic drag), then this is not penalised in the score.

Clearly it is easy to assess "good PPC" as it is bound between 0 and 1 with one being the best score.

2.6.3 Evaluating approximation function performance scores

At many points in this thesis we will be comparing results from competing experimental approaches. Usually these results will be in the form of the above numerical scores, but averaged over a large number of runs.

All claims that approach X out-performs approach Y are statistically proven using appropriate techniques. Sometimes though, this 'out-performance' may appear quite small, and so the questions:

- *"What small differences are worth taking notice of?"*
- *"Is the best result found 'good enough'?"*

need to be addressed.

Firstly, this author takes the opinion that any statistically significant improvement is worth noticing. If it means the difference between 88% accuracy and 89% accuracy, then that is still a beneficial improvement to the aesthetic design process. In industry, time often translates into profit and therefore even a small improvement can translate into an increase in profit. When that industry commands as much money as the automobile industry does, then a small

improvement can easily equate to large amounts of money. Further, from a research perspective, such improvements are always worth noticing because they can often point in the direction of further research that could potentially increase these small improvements dramatically.

Secondly, the notion of "good enough" is a difficult concept. Optimally, we would want to achieve a PPC ranking score of 1, but this appears highly unlikely to be achieved for many problems, especially those addressed in this thesis. Therefore, taken with the statistical significance of the results, I recommend to the reader that they view all of these results of being significant for the reasons given above, but that they understand that there is likely to always be room for improvement with regards to this problem.

2.7 Conclusion

In this section we conclude the chapter with a summary of its most important contents and novel contributions.

2.7.1 Summary

In this chapter we looked at the problem central to this thesis in greater detail. As a motivating example we took automobile design: a real-world domain where aesthetic quality and functional performance are both given high value. As this is an under-researched problem, we looked at computational design processes for inspiration; specifically evolutionary optimisation. In this field we discovered surrogate assisted design optimisation were being employed for similar purposes.

We also gave an overview of other research areas related to our problem. One piece of related literature, found in [92], is very significant as provides encouraging evidence that providing the human designer with information during the visual design process can speed up convergence and result in improved solutions.

Inspired by surrogate assisted design optimisation, we looked at transferring this technique to our problem: in particular, we considered using machine learning techniques to create an approximation function for use in the aesthetic design process.

Following this we described our problem formally and discussed the problems that arise when applying "off-the-shelf" machine learning techniques. It was suggested that one of the

major problems, acquiring a training data set, could be solved by using the data bank of designs and performance scores accumulated at an engineering surface design firm. Simultaneously this gives us the opportunity to learn about previously interesting regions of the space of possible designs. But while this decision solves a few of the problems, it aggravates and accentuates some of the other problems. These problems will be tackled in the remaining chapters of this thesis.

Towards the end of the chapter we formulated the problem mathematically and described the four performance metrics using these terms which will be used throughout this thesis.

2.7.2 Novel contributions

The major contributions of this chapter include an analysis of an under-researched problem with real world significance, coupled with a motivating example to demonstrate its significance. By suggesting an accumulated training data set to solve one of the problems, we develop the original problem such that another deeper analysis of the problem must be given. This involves a novel analysis of a data set with particular properties that is previously unseen in the literature.

Chapter 3

Preprocessing Surface Design Descriptions for Machine Learning

In this chapter we look at the challenge of taking our data set, consisting of varied surface designs in a flexible format, and preparing it to function as input into the machine learning process. We will look at the challenges of the data set relevant to this chapter in more detail, leading onto a discussion of surface design representation conversion and dimensionality reduction. During this discussion we will encounter the 'bump problem' and suggest how it may be dealt with. We will then discuss the generation of a test data set of automobile surface designs exhibiting the properties previously described in section 2.4.3 which will be used throughout future experiments in this thesis. Following this we will discuss the generation of toy test data set that exhibits the previously mentioned 'bump problem'. Finally, we will discuss a novel and computationally fast method of converting surface design representations into a single format, and apply this to our data sets. We will also pay particular attention to a promising dimensionality reduction approach and how well it performs in synergy with this conversion process with regards to the 'bump problem'.

3.1 Problem introduction

In the previous chapter we described some challenges of the data set that we would need to overcome in order to solve the problem of generating a regression function exhibiting good performance from this data set. Our task is to take our accumulated data set of engineering surface

designs, D, and arrive at a version, D^U, suitable for machine learning a regression function to perform as our approximation. We refer to this as the preprocessing of D. In the following three subsections we look in detail at three challenges of the data set relevant to this chapter.

3.1.1 Engineering surface design representations

When engineers design the surface of an object, such as that of an automobile, they typically use some computer aided design (CAD) software such as Maya.

As previously described, flexible surface design representations typically involve discretisation of the surface into primitives: either curved surfaces, faces or a mixture of both. Curved surfaces (such as NURBS surfaces) consist of a number of control points, while faces consist of $a \geq 3$ edges (with a vertices; 2 per edge and each vertex being shared by two edges).

With there being a variety of commercial and non-commercial CAD software suites, and also a variety of functional performance evaluation suites, formats have been developed for easy geometry exchange between independent software suites. These formats can include a range of information, such as colour intensities and texture coordinates which we do not need for functional estimation. The most prevalent format designed with a focus on the geometry is STereoLithography (STL) due to its simple and easily parsed format [116, 20].

STereoLithography (STL) representation

An STL file for a single design consists of a number of triangular 'facet' descriptions, in **no particular order**. Each facet is itself described by a list of 3 vertices, describing the points of the triangular facet, which are themselves ordered in a specific manner. The vertices are ordered such that they indicate the side of the triangular facet which we would designate as the 'front', i.e. the side which the triangular facet's normal vector faces away from: given the 3D vertices of a triangular facet in the order v_1, v_2, v_3; the cross product vector formed of vectors $\overrightarrow{v_1 v_2}$ and $\overrightarrow{v_1 v_3}$ would be in the same direction as the normal vector $\overrightarrow{n}$ of the surface. As the cross product is anti-commutative, reversing the operands such that we calculate the cross product of vectors $\overrightarrow{v_1 v_3}$ and $\overrightarrow{v_1 v_2}$, would result in a vector pointing in exactly the opposite direction. In computer rendered graphics processing, this ordering rule can be used to indicate when a triangular facet is facing away from the viewer and therefore when the facet need not be drawn.

Figure 3.1 shows an automobile design in the STL representation, with the edges of triangular

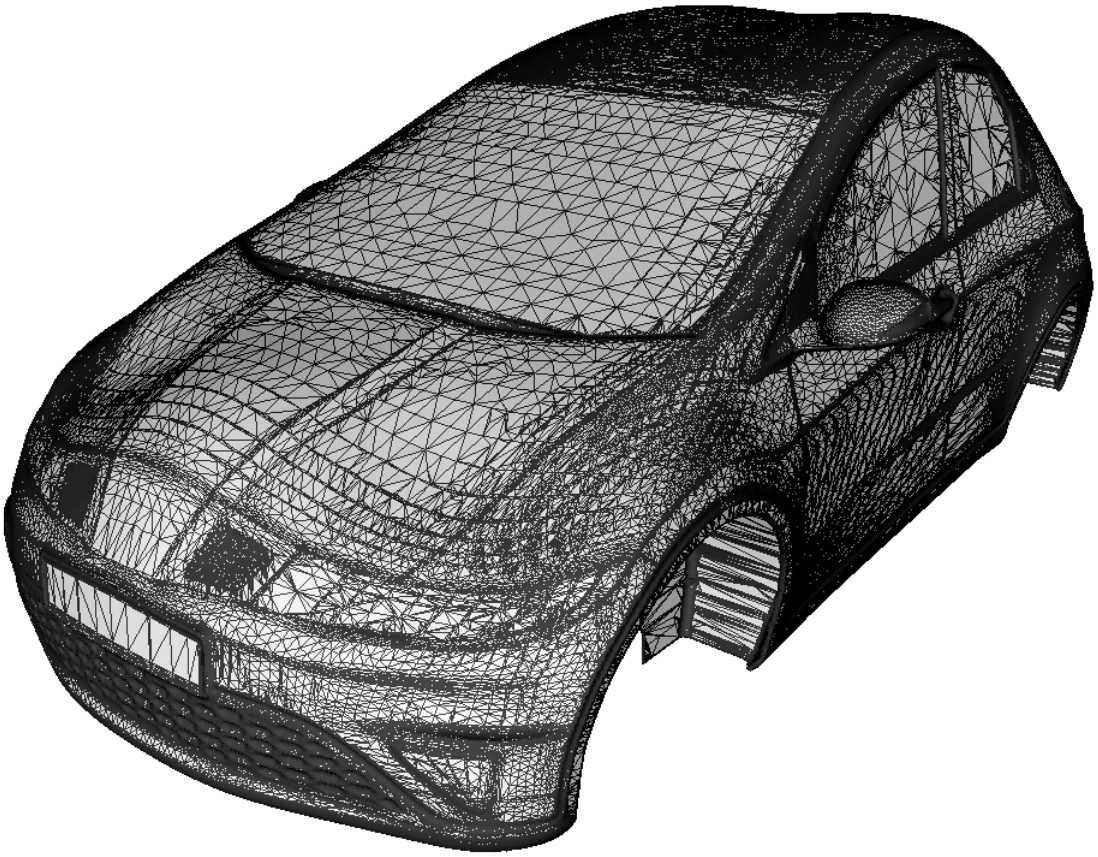


Figure 3.1: Wire-frame view of an automobile design showing how the surface of the Honda Civic (European) is constructed from triangular facets in the STL representation. The design consists of 555,397 unique 3D points which use used to describe 1,129,308 triangular faces.

facets visible. Note the increase in triangular face resolution in regions of the surface which are designed to appear curved.

STL is a great example of a 'complete' representation, i.e. it can feasibly be used to represent any geometry as long as curves are discretised into enough facets. STL is also well established format supported by virtually all CAD, 3D design software and functional performance suites (such as CFD), so it won't be challenging to generate STL representations from a historic and accumulated data set consisting of a mixture of different formats.

Problems with STL

STL formats can be very large and take time to parse if they are stored as ASCII text files. Fortunately, a binary format exists which reduces the size substantially and removes the need to

parse a text file.

For machine learning purposes, the STL format poses a number of problems. Firstly, as previously mentioned, the triangular faces are unordered. There are no rules on the ordering and since there is no linear connectivity between facets (i.e. each facet shares edges with three other facets, and each facet can share vertices with more than 3 facets) there is no method of generating a linear file format. It is therefore impossible to associate the first facet across the data set with any particular surface detail.

Secondly, the resolution of the STL format is variable, meaning that the number of facets is variable. Clearly, the STL format does not represent curves directly. Any curves must be discretised (although most CAD software applies smoothing to its geometry and so this discretisation need not be high resolution) and the surface designer is free to increase the resolution of a region of the design in order add more detail there. Fixing the design resolution is not feasible as designers need flexibility to implement their goals. It is therefore even harder to associate facets with functional change since some facets (and the vertices that form them) will exist in some designs and not in others.

3.1.2 Dimensionality reduction for a high dimensional non-linear problem

Surface design data formats, such as STL, often consists of a massive number of attributes (millions). This huge number of attributes renders the generation of a machine learned regression function either unfeasible or computationally very expensive (which may still render it still unfeasible for many situations).

We therefore seek a dimensionality reduction function $g(\mathrm{x}) : \mathrm{x} \longrightarrow \mathrm{x}^{'}$ where $\mathrm{x} \in \mathbb{R}^{d}$, $\mathrm{x}^{'} \in \mathbb{R}^{d^{'}}$ and $d^{'} \ll d$. We will review dimensionality reduction techniques shortly, but here we will describe the problems of our data set which a suitable dimensionality reduction technique will need to overcome, which have already been mentioned in sections 2.4.2 and 2.4.3.

Firstly, we have a very high dimensional data set taken from a large design space. This means *considerable* dimensionality reduction will need to be applied and yet we want to maintain as much information as is relevant to the functional performance.

Secondly, the aesthetic designer will expect a high degree of artistic freedom across the entire space of possible surface designs, and functional performance measures are easily affected by surface changes. This indicates that a truly compact representation will be non-trivial to find

because our low-dimensional reduced representation will need to be able to represent these surface changes when they are relevant to the functional performance.

Thirdly, let us consider two non-exclusive sets of attributes:

- Attributes which are likely to be changed by the aesthetic designer, which we denote subset A.
- Attributes which affect the functional performance, which we denote subset B.

then we will want to filter attributes out leaving us with just the intersection C of sets A and B: $C = A \cap B$. This will represent the smallest possible set of attributes needed for our approximation. This is a good approach, but the problem identified previously is that some attributes may be modified by the designer in one subspace of the space of possible designs S, while a completely different and exclusive set may be modified in another subspace, making A quite large. Further, some attributes may have effect on functional performance in one subspace of S, while a completely different and exclusive set has effect in another subspace making B quite large. Therefore C will itself be large, affecting the method of dimensionality reduction that can be applied. Because of this variation across S we consider the problem to be highly non-linear.

3.1.3 Synergising representation & dimensionality reduction for the bump problem

The representation conversion process may be performed such that it applies a form of dimensionality reduction to the data. Despite this, we will still aim for a rather complete converted representation of the surface data and apply a true dimensionality reduction technique to this. This means that when considering the first problem of surface data representation conversion, we should consider a representation conversion process which synergises well with the dimensionality reduction technique subsequently applied. Clearly, a chosen dimensionality reduction technique needs to deal with the problems mentioned in the previous subsection and work with a surface data representation.

An important issue involving both the data representation and dimensionality reduction, is that surface design formats such as STL are not particularly effective for machine learning problems, as illustrated with the 'bump problem'. In figure 3.2 we see an illustration of the 'bump problem' concept. In the first two plots of the figure, we see a set of points defined (in

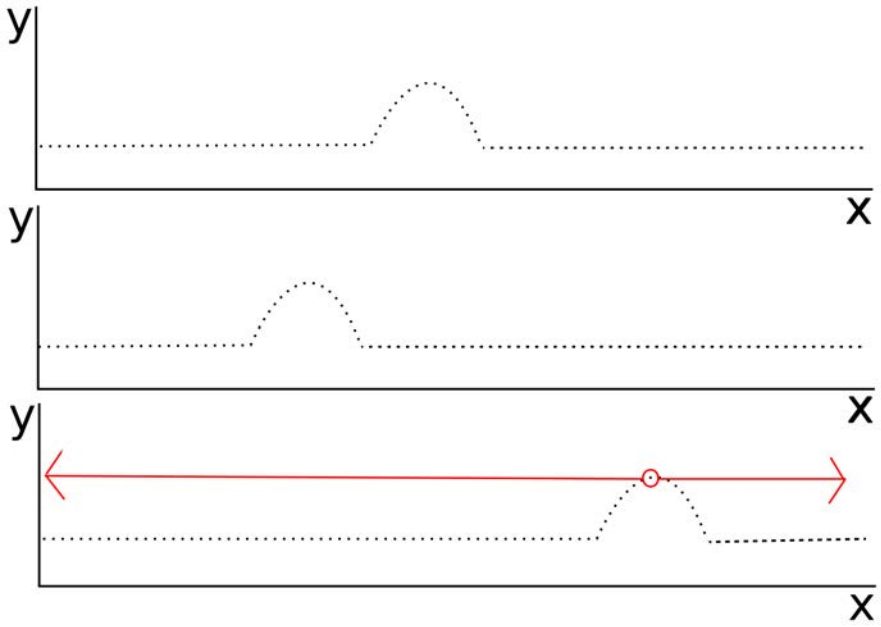


Figure 3.2: Illustration of the bump problem.

2D) on the x and y-axes. We imagine that these points constitute a connected surface (with a non-illustrated facet in between).

In each of the three plots we see there is a feature of the surface, a bump. The bump problem exists because for many formats, STL included, different attributes (points) are required to represent the bump depending on where the bump is along the x-axis. Assuming that this bump affects our functional performance, a regression function would require as input any point that may be involved in the bump at any time: for our illustrated example that could easily be every point, if the bump could be positioned at any location along the x-axis.

But, as illustrated in the final plot, we only need a single attribute to represent the x-axis position of the bump. Expecting a machine learning algorithm to infer the performance rules for each position of the bump (and thus every possible subset of points involved in the bump at each point) is not totally unreasonable, but certainly not as trivial as associating performance rules with a single x-axis value.

The challenge of dimensionality reduction for the bump problem is then finding this x-axis feature from the data set. Later on we will describe a dimensionality reduction approach that may deal well with this problem.

3.2 Generating a single representation from varied surface design representations

In this section we will review existing methods of generating a single representation of varied surface design representations.

This is a field within which there is not much research. This is possibly because accumulated data sets of varied historical designs are only now beginning to gain enough size to be of use to some machine learning task.

As previously mentioned, when faced with a similar problem (i.e. using diverse surface designs in some machine learning process) the authors of [42] approach the problem using the idea of a reference model and offsets. Their end goal is a system of decision-tree based rules which can be used to guide some design process (human or computer driven). Their method assumes the selection of some surface design which we refer to as the reference design, x^R. This need not be a real design and can be fabricated and even optimised for purpose. The designs in the data set are then calculated as offsets from x^R:

1. For each point in the surface x^R we seek a matching point in the surface being converted, x. This is done through a novel similarity score incorporating point distance and surface normals similarity.

2. Then an offset is be calculated between the matched points in order to arrive at a representation of x constructed of these offsets.

This is a good approach, and the similarity calculation based on both distance and surface normal helps ensure that point matching is done more effectively than if were solely based on distance. Also, a level of dimensionality reduction can be incorporated into the process by choice of a reference model with fewer points to be matched. The authors are able to demonstrate the effectiveness of their methods on a turbine blade design.

Clearly, a turbine blade design is a relatively simple shape compared to that of an automobile. Most notably, their method will only work for designs within some threshold of similarity to the reference model, x^R, as it is not hard to perceive cases where a shape substantially different from x^R will cause problems. We can conclude that this method will not perform well for data sets of highly diverse designs.

For example, the previously mentioned example in automobile design where some designs have a spoiler and others do not. A spoiler on an automobile can have dramatic effect on the functional performance of a design. Clearly, if x^R does not feature a spoiler, this cannot be modelled in the representation. On the other hand, if it does feature a spoiler, designs without spoilers could exhibit unexpected behaviour when matched in this way.

Another approach employed in [14] involves the voxelisation of the designs. Voxelisation is the process of building a 3D representation of a surface through the division of the encompassing space into equally sized voxels. It is effectively the 3D extension of pixelisation. In [52] the authors demonstrate a method of generating a voxelised representation of a surface from point-cloud data (i.e. a set of points found on the surface). They demonstrate their approach on aerial and terrestrial data sets.

Canonical voxelisation with a surface design involves projecting 3D 'rays' (a line segment with an origin and direction) for each voxel on each axis [107]. So for an $x \times y \times z$ voxelisation, we must generate $x \times y + z \times y + x \times z$ rays. Each of these rays must then be tested against each of the N facets in the surface. This can be quite costly as the triangle intersection tests are not cheap, requiring computation of barycentric coordinates for each one.

Finally, all $x \times y + z \times y + x \times z$ voxels must be processed, each using the results of three ray tracings in order to determine whether the voxel lies inside or outside of the surface. In order to ease computation, this is typically performed under canonical voxelisation in a binary fashion (inside our outside of surface) leaving us with a binary representation highly dependent on the original x , y, and z parameters. Extending it to a continuous representation (indicating the amount of intersection with a surface that a given voxel has) naturally requires extra consideration and computation.

Clearly, a $x \times y + z \times y + x \times z$ representation can be very large and some level of dimensionality reduction must be applied before the data is suitable for machine learning.

Another fairly simple approach could be to place sensor points in 3D space and to have these measure the closest distance to the surface, and whether they are inside or outside of the surface. This therefore builds a level of dimensionality reduction into the process through the choice of a low number of sensor points.

To save computational effort, it may be enough to record the distance to the nearest point, and avoid finding the nearest point on the face of the nearest triangle (an operation which would

be considerably more expensive to perform).

This approach depends heavily on the number and placement of such sensor points in the 3D space. To this end, both these parameters may need to be optimised using some computational optimisation system (e.g. gradient decent, evolutionary optimisation). Unfortunately this makes the approach non-deterministic and also adds considerable computational complexity to the process.

3.3 Dimensionality reduction for machine learning of engineering problems

As this and future chapters employ dimensionality reduction in their experiments, this section will briefly review existing methods of dimensionality reduction which can be used for building an approximation of functional performance. Particular focus will be given to two well-established approaches which we will use later on.

A dimensionality reduction algorithm is itself a form of machine learning and must ultimately result in a dimensionality reduction function $g : \mathrm{x} \rightarrow \mathrm{x}^{'}$ which can map an input $\mathrm{x} \in \mathbb{R}^{d}$ into a lower-dimensional input $\mathrm{x}^{'} \in \mathbb{R}^{d^{'}}$ where $d^{'} \ll d$. The dimensionality reduction function g can generally be described as either a feature selection or a feature extraction method [110, 70, 31]. In this sense, 'feature' is used in a sense synonymous with our use of 'attribute' so far in this thesis.

One of the great challenges in finding g is to balance the trade-off between *compactness* and *completeness* [64]. Compactness is self explanatory and relates to the how small the reduced dimension representation is (the smaller is typically the better for combating the curse of dimensionality). Completeness on the other hand refers to how capable the reduced representation is in capturing all of the information required for describing each sample. On one end of the completeness scale, the non-reduced representation will allow for all differences between samples to be represented (i.e. it is totally complete). At the other extreme, a reduced representation may allow for only one attribute to represent the differences between samples, effectively putting the samples on a linear scale. Clearly, completeness is problem dependent: for functional performance scores such as aerodynamic drag, the colour of the surface is irrelevant and can be ignored without any loss of completeness towards our task.

3.3.1 Feature selection methods

Feature selection methods of dimensionality reduction generally discard the $d - d'$ attributes in the data assessed to be the "least important" with regards to the function being learned. Feature selection, it should be noted, can happen at the human level using domain knowledge. For example, aerodynamic performance of engineering surfaces is not affected by the colour of that surface. This attribute can then be discarded by the human engineer.

Computational methods of feature selection can be broadly categorised into one of three categories: wrapper, filter and embedded methods [48].

Wrapper methods

Wrapper methods effectively search for the optimal set of attributes by evaluating different subsets of attributes for the desired task. In our case, that would mean evaluating a regression function using each subset in order to evaluate that subset. The available data is divided into training and testing data, and a regressor would be built on the training data. The testing data is then run through the model and its accuracy is recorded as that subsets performance on the testing data. Clearly, we seek to optimise this performance. In this way, we search for the dimensionality reduction function g that results in the most accurate model.

This approach is applied to surrogate-assisted optimisation of an aerofoil shape in [135] and the benefits of this dimensionality reduction on the surrogate used in an optimisation process are shown.

Clearly, this approach is computationally intensive, as many models must be regenerated with different data and then tested. Further, it is dependent on the testing data set. If this is too small or chosen poorly then the converged solution may not be suitable to the broader task.

Filter methods

Under filtering we perform our search for a reduced dimensionality representation through the scoring of attributes with an easy to compute metric. We simply remove the $d - d'$ attributes with the 'worst' scores (or where d' is unknown, we use some threshold score). Filtering can be a simple unsupervised approach (meaning it **does not** take the functional performance value into account) such as :

- Removing the attributes with the least variance in the data set.
- Removing the attributes with the least range throughout the data set.
- Consolidating attributes which correlate highly throughout the data set.

A supervised approach (meaning it **does** take the functional performance value into account) may remove input attributes which have the least correlation with the output variables, making the assumption that they have little effect on the output variables [49] (we will apply such an approach later in section 3.4.3). This has been applied to an engineering task in [42] with a correlation threshold of ± 0.3 and for a regression problem the approach is intuitive, taking into account the effect of attributes on the output value.

Aside from correlation, another common metric for determining whether attributes are useful to a regression task is to calculate their Mutual Information score. This approach has been used for feature selection in [74, 85].

In order for filtering methods to function well for a high-dimensional problem, there needs to be a large amount of data available. This is a luxury that many real world problems, including our own, do not have. Another problem that filtering methods exhibit is that they can easily ignore the presence of interacting features (features which on their own have no direct effect on the functional performance, but when combined with other features, do have an effect). Interacting features are common in engineering surface domains and the search and discovery of them is covered in [59, 145].

Embedded methods

This final group is quite limited but includes all methods that perform feature selection as part of the model training process. A good example of such an approach is LASSO [137], an approach to linear regression via least-squares which attempts to drive as many of the linear regression coefficients as possible to zero, thus acting as a form of dimensionality reduction.

3.3.2 Feature extraction methods

Extraction methods map the data into a lower-dimensional representation with a dimension of $d^{'} \ll d$ while trying to maintain as much information in the data as possible. The resulting representation rarely resembles the original representation and the resulting attributes cannot

be directly linked to any single attributes in the original representation. In this sense, feature extraction is very similar to compression.

There are a number of ways of performing feature extraction and we will look in detail at two approaches which we will use later on. We can generally group feature extraction methods into linear and non-linear approaches.

Linear

This family of approaches includes Principal Component Analysis, Independent Component Analysis, and Linear discriminant analysis. These methods involve linear algebra such as the highly popular unsupervised method Principal Component Analysis (PCA) [65, 128] which we will now take time to describe in detail.

Under PCA the covariance of all attributes is used to find the eigenvectors best describing the variance and covariance of the data set. By mapping the data onto the $d^{'} \ll d$ eigenvectors with the greatest eigenvalues, PCA effectively embeds the data in a representation which captures the greatest variance in the data.

Under PCA we assume a set of sample data, X, where C is the covariance matrix for the attributes of X (containing the variance and covariance of attributes in X). We compute the eigenvectors and eigenvalues for C as V and w respectively. Where $d^{'}$ is the target dimensionality of the PCA process, we order the vectors of V by their eigenvalues w and retain only the first $d^{'}$ eigenvectors (as columns) in matrix W.

Then a single sample x is embedded as a $d^{'}$ dimensional design representation $\mathrm{x}^{'}$ by performing:

$$g(\mathrm{x}) = \mathrm{x}^{'} = (W\mathrm{x}^{\top})^{\top}$$

PCA effectively compresses the data into a representation of specified dimensionality, while aiming to retain as much of the variance in the data set as possible. It is possible to reverse this PCA embedding function g so that $\underset{\leftarrow}{g} : \mathrm{x}^{'} \rightarrow \check{\mathrm{x}}$ where $\check{\mathrm{x}} \in \mathbb{R}^{d}$ but some information will be lost in the process. In this way PCA behaves similarly to *lossy* compression techniques.

One disadvantage of linear approaches like this is that they don't take the functional performance term into account. As mentioned, the embedding process aims to embed the data in a

representation capturing the greatest variance in the data. This is sensible if all the attributes generally have the same strength of effect on the functional performance, but for problems such as ours this can be quite ineffective. For example, it is entirely possible that the attributes with the most variance in the data set have only a small effect on the functional performance, while the other attributes with less variance, affect the functional performance more strongly.

Another disadvantage of linear approaches such as PCA is that they are linear approaches and therefore may not perform well on some classes of problems. The effects of surface changes on functional performance are typically non-linear and thus our problem falls into this troubling class of problems.

To reiterate, given a d-dimensional representation in our problem domain, it is possible that any attribute x_i, where $1 \leq i < d$, has a very strong effect on functional performance in some subspace of the design space, but no effect whatsoever in most of the design space. If most of the d attributes behave in this manner, linear dimensionality reduction will be sub-optimal. A non-linear approach that is able to fit these "changes in attribute sensitivity" across the design space would be better suited to such a task.

Non-linear

This family of approaches to dimensionality reduction contains Kernel-PCA (a non-linear implementation of PCA using kernels), the manifold techniques (e.g. Locally Linear Embedding (LLE) and ISOmap), and Semidefinite Embedding [76]. Of these technique groups, the most prominent are the 'manifold learning' techniques.

Manifold learning approaches assume that the most important variation of the data exists on some low-dimensional non-linear manifold in the design space. The dimensionality reduction problem involves finding this manifold and embedding all data on it in a lower-dimensional representation.

A real world example of a manifold can be considered to be the planet upon which we live. Our world is three-dimensional. The cities of the world lie in a variety of 3D positions (some are elevated high above sea level, others are closer to sea level). For an arbitrary set of orthogonal three-dimensional axes, each city can be given some position.

When a map of the worlds cities is drawn up, the tendency is to recognise that the cities lie on a 2D manifold (longitude and latitude) on the earths surface and that the true 3D nature

of the world can be ignored as most of the 3D data (the uninhabited inside of the planet and the atmosphere above ground level) is not interesting to us. This, in effect, is manifold-based dimensionality reduction in practice, taking us from a 3D positioning system, to a 2D system. The key is to recognise that a 2D manifold exists on which the data of interest (e.g. city locations) can be positioned.

For arbitrary data sets, manifold learning then involves uncovering such a low-dimensional structure in a high-dimensional data set through the analysis of the data set. The two most prominent approaches to manifold learning are Locally Linear Embedding (LLE) and ISOmap.

LLE involves finding the set of k nearest neighbours for each point x and finding the set of weights for the neighbours which, once the k nearest neighbours are combined in a weighted sum, best recreates the original point x [118]. A low dimensional embedding of the points is found via eigenvector-based optimisation with the aim that each point can be recreated with the set of weights calculated previously.

LLE has been shown to be faster to compute than its main rival in the manifold family, ISOmap, and also to perform better on many problems. But, unfortunately, LLE has been shown to perform more poorly with data sets which have not been uniformly sampled, as is the case with our data set.

ISOmap consists of the following steps [134], given an input data set $D = \{\mathrm{x}_1, ..., \mathrm{x}_N\} \in \mathbb{R}^d$ where $\mathrm{x}_i \in \mathbb{R}^d$, a nearest neighbour function (i.e. k-nearest neighbours or ϵ-distance threshold method) and a shortest paths function (e.g. Floyd's or Dijkstra's):

1. Form the a nearest-neighbour graph with edge weights $W_{i,j} = \|\mathrm{x}_i - \mathrm{x}_j\|$ for neighbouring samples x_i and x_j.

2. Compute the shortest path distances between all pairs of samples. Store these distances in $\mathbf{Z}$.

3. Return $\mathbf{Y} = MDS(\mathbf{Z}, b)$.

where $\mathbf{Y}$ is the embedded data, and the function MDS is a Multi-Dimensional Scaling algorithm which takes as input the geodesic distance matrix $\mathbf{Z}$ and the target dimensionality $b < d$.

The $\|\mathrm{x}_i - \mathrm{x}_j\|$ distance function can be chosen to be the euclidean distance between x_i and x_j:

$$\|\mathrm{x}_i - \mathrm{x}_j\| = \sqrt{(\mathrm{x}_{i,1} - \mathrm{x}_{j,1})^2 + ... + (\mathrm{x}_{i,d} - \mathrm{x}_{j,d})^2}$$

where $\mathrm{x}_{i,j}$ is the jth attribute of the ith sample in the data set.

As opposed to LLE, ISOmap can be used in conjunction with a non-uniformly sampled data set, and therefore will be more appropriate for our work.

In [134] and since, ISOmap has been shown to perform well for pixel images. Looking at figures 3.3 and 3.4 we can gain a bit more understanding of what ISOmap does. For both figures the original data sets were pixel images containing many pixels and thus many attributes at one per pixel (64x64 in the case of figure 3.4). ISOmap was able to find a low-dimensional manifold embedding where the low-dimensional (i.e. most-significant) attributes appear to have significant meaning to the human eye (we can say that these attributes have semantic meaning).

To date, there are no studies employing ISOmap for dimensionality reduction in engineering problems like our own. This is largely owing to the fact that ISOmap is heavily dependent on the distance measure used in calculating the nearest neighbourhood graph, and that for very high-dimensional problems based on 3D points in euclidean space (such as engineering problems) a good distance measure is difficult to find. This then leads to ISOmap having difficulty extracting the spatial information from a data set containing raw attributes (such as the coordinate values of the points on an STL surface).

But, importantly, there are good reasons to think that ISOmap may be a suitable way of dealing with the previously mentioned 'bump problem' described in section 3.1.3 if given a suitable representation from which to learn. Later on in this chapter we will elaborate on this and then try to prove this using experimental studies.

3.4 Generating an experimental data set

In this section we describe the generation of an experimental data set exhibiting the properties described in section 2.4.3. This data set will be used in this thesis to evaluate the novel solutions put forward.

Generating an experimental data set gives us control over the data, in particular:

- We can enforce properties we are interested in.

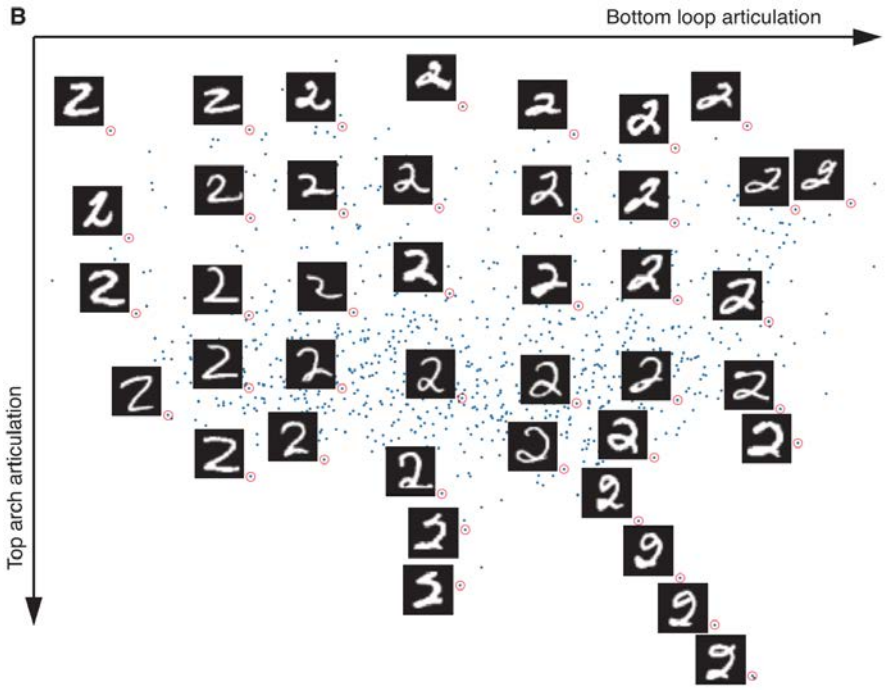


Figure 3.3: Two-dimensional ISOmap embedding of a handwritten number '2' taken from [134]. The axes are labelled with an interpretation of their meaning.

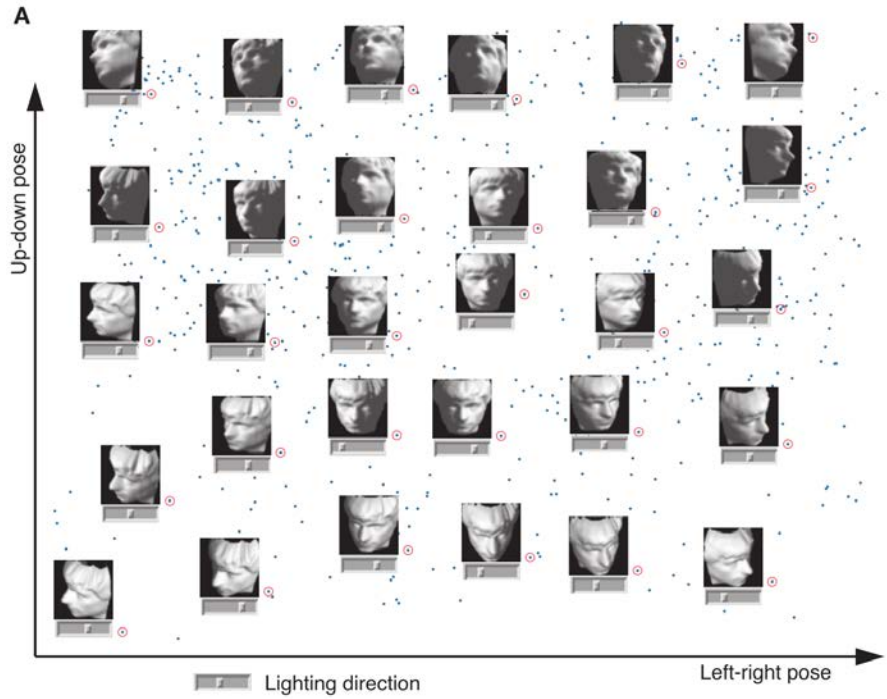


Figure 3.4: Three-dimensional ISOmap embedding of pixel-images of a rendered face taken from [134]. The axes are labelled with an interpretation of their meaning. A third dimension "Lighting direction" is illustrated with the slider bar under each magnified sample.

- We know that the points in each surface design representation are actually match-able and which points match between designs. This unrealistic property will allow us to make comparisons with a theoretically perfect solution.
- We can make the designs different enough that they are interesting, but similar enough that a machine learning algorithm can learn from them.

The properties we want to enforce in the data set (summarised from the discussion in section 2.4.3) are namely:

1. Samples are taken non-uniformly from a large space of designs.
2. Each sample is a high-dimensional surface description with an associated functional performance score.
3. The number of attributes (triangular facets and their points) in each surface varies, and the order of these attributes varies too.
4. The optimal compact representation in one location of the design space differs from that in another (due to the functional performance score).
5. The data set has a comparatively small sample size (compared to the average number of attributes in a sample).

At this point, the reader should be made aware that due to a limitation of the research, there is one important property missing from this list. We will elaborate on that property in the next section and generate a separate experimental data set exhibiting this property, albeit a highly artificial one.

When generating our experimental data set we use two technologies, Free-form Deformation and OpenFOAM.

OpenFOAM is an open source CFD software suite which can be used, amongst other functions, for estimating the aerodynamic performance of a 3D design. For a full reference see [36].

Like most CFD software suites, OpenFOAM does not work on the surface data itself but rather on a mesh representation of the space surrounding the object, broken down into cells. The mathematical model behind most CFD approaches involves the simulation of particle flow on a per-cell basis, calculating the particle flow interaction between the neighbouring cells. Typical

for a CFD simulation is a preprocessing step where a human engineer must either completely construct or refine this CFD mesh; itself a costly process requiring human intervention.

Free-form Deformation (FFD [124, 24]) is a control-point based method of making design deformations and lends itself well to sculpting. Unfortunately it is quite restrictive (i.e. restricted by the control point layout) and so tends to be useful for only very general design tasks.

One of the most attractive elements of FFD is that it can be used not only to modify a design, but also the previously mentioned CFD cell mesh, allowing us to effectively automate this mesh generation for any design modified by FFD [93] (providing a CFD mesh was provided for the original surface design being deformed). As it is a deformation approach, it maintains the structure of both the deformed surface and the deformed mesh.

Inspired by the motivating example given in section 2.1, we took the Honda Civic (European) automobile design in STL format as a base engineering surface design (previously seen in figure 3.1 and provided by the Honda Research Institute Europe, Offenbach, Germany). This design consists of $555,397$ 3D points arranged into $1,129,239$ triangle faces. Looking at the unique 3D points alone, we note that we have a total of $d = 1,666,191$ real-number attributes describing points on the surface of the design.

Considering that each triangle is made up of three 3D points, one can say that expanded, the design is represented by $1,129,239 \times 3 \times 3 = 10,163,151$ real-number attributes. However due to the redundancy of shared points this can be reduced to an array of the $555,397$ 3D points and a set of three integers for each triangle indexing an entry of the array of 3D points, totalling $1,129,239 \times 3 = 3,387,717$ integer attributes. By modifying the $555,397$ 3D points alone, we can modify the design while the triangle-face structure stays the same.

We chose to generate eight groups of 150 'similar' designs, with the members of each group being considerably different from the other seven groups. This gave us a total of 1200 designs.

The first step in applying FFD is to construct the control-point lattice in which the designs will be embedded. Our 3-dimensional lattice was constructed with the following control point dimensions: 9 on the x-axis, 7 on the y-axis and 9 on the z-axis, giving a total of 567 control-points.

Eleven exclusive subsets of adjacent FFD and symmetrical control points were assigned 'control group' names $G_1, .., G_{11}$. In figure 3.5 these groups have been labelled. The effect of this is that making modification to control group G_1 has a linear effect on geometry which is strong on

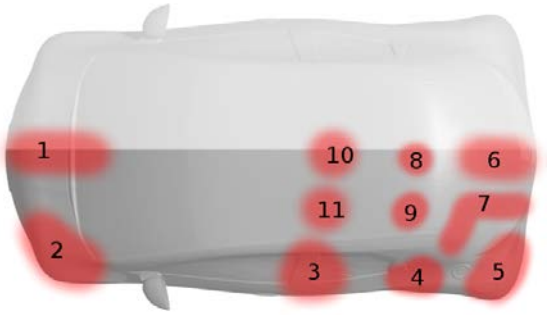


Figure 3.5: Regions of the vehicle design most strongly associated with each control group.

the front-centre of the vehicle, and has no effect elsewhere.

In order to achieve our eight groups of designs, we modified control groups according to Gaussian distributions $N_i^j = N(\vec{\mu_i}^j, \Sigma_i^j)$, where $i = 1...8$ is the index of the design group; $j = 1...11$ is the control group index. In order to ensure the designs of each design group $i = 1...8$ were substantially different, a different $\vec{\mu_i}^j$ and Σ_i^j was used for each i. By using Gaussian distributions we ensure there is an average design in each design group and all other designs in the design group are similar to this. In figure 3.6 we can see example designs from each generated group.

As a further element of realism we made sure that not all design groups vary in the same surface design regions. We did this by freezing modifications of certain control groups for each design group. Table 3.1 indicates which control groups were frozen and for which design group.

Using these Gaussian distributions we were then able to generate the modified designs in our data set, and also automatically generate their matching CFD meshes. The designs were evaluated by a the OpenFOAM CFD solver in order to estimate their aerodynamic drag values as functional performance (these are real numbers in the region of 230.0 - 280.0).

We then have the base version of our accumulated data set D. In the following subsections we discuss some slight modifications which we were able to make.

3.4.1 Adding realism

In this thesis we are looking at taking a data set of diverse engineering surface design representations and creating a single data set of unified representations for use in machine learning tasks.

Interestingly, because of the way in which this experimental data set was generated (i.e.

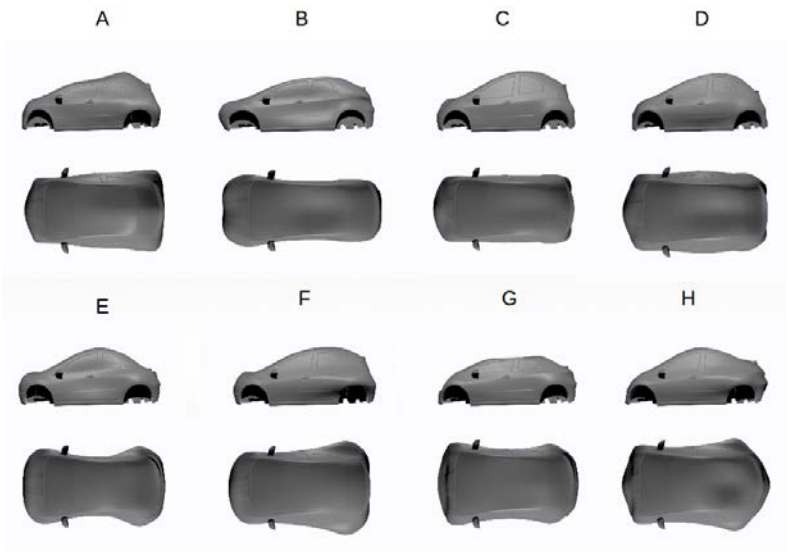


Figure 3.6: Example classes for each of the 8 design groups in the automobile data.

Design group	Frozen control groups
1	5,4,6,7
2	8,9,10,11
3	3,10,11
4	2,3,4,5
5	1,2,3,10,11
6	3,7
7	1,3,6,8,9,10,11
8	4,6

Table 3.1: Frozen control groups per design group.

through FFD) we have a data set where each surface design consists of the same number of points, and the ith point on each surface design conveys information about the same geometric part of the surface across all the designs. This is of course unrealistic (as described in the introduction to this chapter) and does not meet the third desired property mentioned above in section 3.4.

This unrealistic 'matched' representation will be beneficial when evaluating approaches towards regression in chapter 4, but effectively solves this chapters problem with regards to unifying varied engineering surface designs.

In order to prepare our experimental data set D for experiments involving representation unification, we need to modify the data set to make it more realistic and therefore to exhibit this more realistic property.

The process for doing this is quite simple: for each sample x_i in D:

1. Split a randomly chosen number of the triangles that make up x_i into three smaller triangles.

2. Reorder the triangles that make up x_i by randomly swapping pairs of triangles.

The first step is designed to emulate the act of the aesthetic designer increasing resolution in particular areas of the surface design representation. Often they will do this to realise a particular surface shape that may not have been possible in the original surface design representation. As the areas of the surface design being split differ between stylistic approaches (i.e. between styles and artists), it is hard to predict where these splits may be. Therefore, although this process is not truly random, we feel it can be sufficiently emulated by random selection in order to generate a data set with the same properties that make the motivating problem so challenging.

The second step is important as in many engineering surface design descriptions, and in particular STL, the order of the triangles is arbitrary, often determined by the exporting function of the design software. For example, it could simply be the order in which the triangles were created, the order in which the triangles are found on a particular axis, or ordered by size of triangular face.

Once this entire process is completed, we have a more realistic data set D, denoted by D^R, which exhibits the challenging properties we are interested in.

3.4.2 Taking FFD handles as input

Interested readers should note that after using FFD to generate D, we not only have the deformed designs that form D but also the positions of the FFD control points which lead to each in design in D.

So, based on our control point lattice's dimensions, we can form a $567 \times 3 = 1701$-dimensional data set from these, D^F. This data set is a by-product of our method and although not used in this work, could be useful in future and related work.

3.4.3 Creating an ideal data set via feature selection

In this chapter we will seek to compare our novel solution with a data set which has been pre-processed flawlessly. We think of this as an unfeasible and ideal data set. In chapter 4, we will also seek to use our novel solution with this in order to isolate the positive (or negative) effects of this chapters work. Clearly, this requirement is fulfilled by D as previously mentioned in section 3.4.1.

Unfortunately, the immense dimensionality of each sample in D, $d = 1,666,191$, makes D unfeasible as a training set for a machine learning algorithm. Machine learning algorithms simply will not successfully infer any rules from a data set of this dimension in a realistic time frame. Further, the rules inferred will be less reliable (due to the curse of dimensionality), and a single estimation may take many minutes to perform.

We can apply some dimensionality reduction to bring the number of points (and thus attributes) in D down to a more acceptable number of points via feature selection. Specifically, a simple correlation analysis was performed, replacing groups of 3D points which correlated highly with one-another in the data set with a single representative point (the point in the group which had the greatest range in 3D space of the geometry).

In [42] a correlation threshold of ± 0.3 is used for a similar purpose based on psychological research given in [23], where it is shown that a correlation coefficient of less than 0.3 doesn't define an observable correlation in practice. After performing the above process with a correlation threshold of ± 0.3 we were left with 508 3D points($508 \times 3 = 1524$ attributes). Trials with values greater than 0.3 were attempted but resulted in a rapid increase in the number of attributes. Knowing that our data set had been generated based on 567 FFD handles, and given the support

in the literature, we felt that the 508 3D points found using ± 0.3 was appropriate. We refer to this version of D as D^C, and call it the 'ideal' representation.

3.5 Generating a data set exhibiting the bump problem

In section 3.3 we placed particular emphasis on the manifold learning technique ISOmap. Towards the end of the ISOmap description we also suggested that manifold learning techniques such as ISOmap may be able to deal with the 'bump problem' (refer to section 3.1.3) in a way superior to other dimensionality reduction approaches.

The reasoning behind this stems from the way in which ISOmap can generate a single attribute based on an abstract feature which is represented by multiple attributes in the original representation. For example, looking at the figures 3.3 and 3.4, we see that ISOmap has found axis attributes which each depend on many of the pixels in the original images (and thus many of the original attributes).

Unfortunately, a major limitation of this thesis' research is that the bump problem was identified only after the experimental data set in section 3.4 had been generated, and proved too late in the project to regenerate the automobile data with the bump problem. It is however important that we are able to test our theories regarding ISOmap and the bump problem on a data set that features this property. To this end, we create another data set which, although perhaps featuring less real-world significance, does feature the bump problem quite strongly, and as previously stated, the bump problem is significant for our motivating problem.

We are interested in enforcing the same properties as those in section 3.4, and so our method is quite similar. The key differences are:

- instead of using a base automobile STL design and creating modifications of this design, we use an STL sphere as our base design.
- our modifications are restricted to procedurally moving a bump along the top the design, along the x-axis.
- we again create clusters of 150 samples each in our data set, but this time we create just three via modification of the 'base design' for each cluster.

Once again, we use a OpenFOAM CFD solver to compute the drag values for the designs for

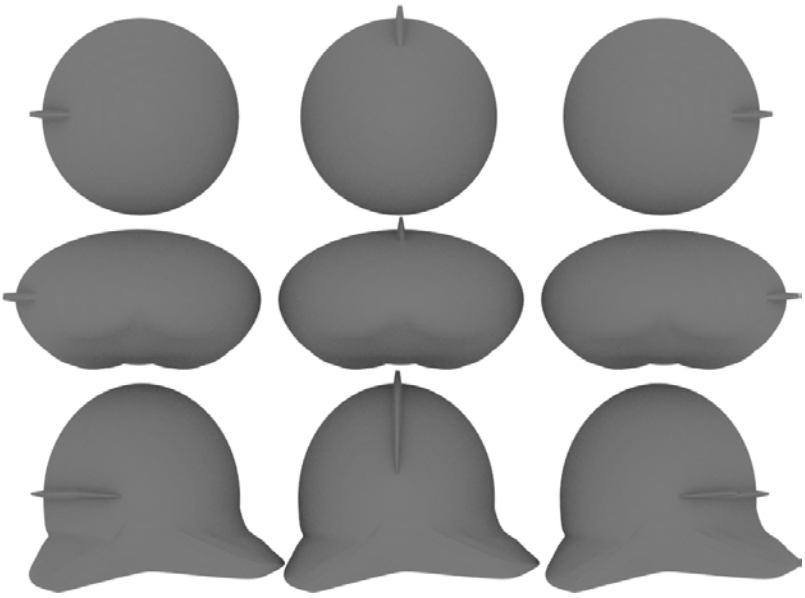


Figure 3.7: Example classes for each of the 3 design groups in the spheres data. The examples taken show the bump at its left-most, central and right-most positions.

the 450 designs generated (with the airflow direction being along the x-axis). In figure 3.7 we show three of these STL designs for each of the three clusters in our data set. The images are based on the samples where the bump is at its left-most position, centre position, and right-most positions respectively.

For the remainder of this thesis, we shall refer to this data set as the sphere data set, D^S. We will be using this data set in chapters 3 and 5, where we have the opportunity to prove the benefits of using ISOmap to combat the bump problem.

3.6 A novel conversion framework: KGrid

In this section we suggest a novel approach to converting our data set of diverse STL format engineering surface designs into a format suitable for machine learning. We begin by summarising the aims of such a conversion system before describing the approach.

3.6.1 Aims of the approach

Our approach should address the following over-arching problem:

1. Given a data set D of N surface descriptions formed of triangular faces (as in STL format).

2. Where each surface description in D is non-standardised, i.e.:

   (a) Each of the N surface descriptions can feature a different number of triangular faces.

   (b) No standard order of faces exists across D.

3. Generate a representation of D suitable for use as input into a machine learning process (either dimensionality reduction or generating a regressor).

In other words, take a data set of surface designs described by an arbitrary number of unordered triangular faces, and produce a single representation for use in machine learning. This single representation should:

1. Compute in as short a time as possible (seconds) for a given design.
2. Enable a machine learner which achieves comparable performance to other, more computationally intensive, approaches.
3. Improve the application of ISOmap to engineering problems in comparison to other approaches.
4. When coupled with ISOmap, be shown to deal with the 'bump problem' for engineering surface design data.
5. Perform the above across a broad space of possible surface designs.

3.6.2 The method

The methodology, which we term KGrid, involves generating a 3D grid of cells and approximating the intersection of each cell with the surface design. In this way it is similar to canonical voxelisation, but there are notable differences:

- The intersection value for a grid position is continuous, not binary, providing an estimate of how much each cell intersects with the surface design.
- In order to facilitate a fast and efficient computation, the surface intersection is approximated rather than explicitly computed.
- Canonical voxels are cuboid in 3D space. In KGrid we opt for spherical cells as the intersection test is simpler than with cuboid cells. Also, as the spheres overlap at their closest points the representation may be more suitable for ISOmap, as a surface feature will rarely intersect just one spherical cell.

Algorithm 3.1 Construction phase for KGrid.

1. Given the parameter τ, the number of sphere-cells on the x-axis; and data set D containing N STL designs, $x_i \in \mathbb{R}^{d^{x_i} \times 3 \times 3}$.
2. Find min_x, min_y, min_z as the minimum value on each axis in D.
   (a) Subtract some padding, ε, from each of these.
3. Findmax_x, max_y and max_z as the maximum value on each axis in D.
   (a) Add some padding, ε, to each of these.
4. Calculate the partition width $p = (max_x - min_x)/\tau$.
5. The number of sphere-cells on the x-axis, $K_x = \tau$.
6. The number of sphere-cells on the y and z-axis, K_y and K_z, can then be determined such that:
   (a) $p \times K_y \geq max_y - min_y$.
   (b) $p \times K_z \geq max_z - min_z$.
7. Calculate the radius $r = \sqrt{3p^2}$ such that each sphere touches its grid neighbour in each direction.
8. Initialise the KGrid as a grid of $K_x \times K_y \times K_z$ spheres with radius r:
   (a) $s_{i,j,k}$ is sphere at the ith position on the x-axis, jth on the y-axis and kth on the z-axis.
   (b) The centre point of sphere $s_{i,j,k}$,
   $c_{i,j,k} = \{min_x + (p/2) + ip,\ min_y + (p/2) + jp,\ min_z + (p/2) + kp\} \in \mathbb{R}^3$.
   (c) The radius of sphere $s_{i,j,k}$, $r_{i,j,k} = r$.

The KGrid method involves a grid construction phase shown in algorithm listing 3.1.

The padding, ε, is chosen such that it is likely that future previously unseen surface designs will fit within the extremes of the KGrid. The τ parameter effectively defines the resolution of the KGrid and will likely be problem specific.

In figure 3.8 we can see a 2D visualisation of KGrid's layout. The key notions to take from this figure are that the centres of the sphere-cells are computed such that there is no space between any sphere-cells. The effect of this is that each sphere-cell overlaps with its neighbours at the closest point (shaded in grey).

We then use the initialised KGrid to convert surface design x into $\hat{\text{x}}$ as shown in algorithm listing 3.2.

Algorithm 3.2 Converting design x into $\hat{\text{x}}$ via KGrid.

1. Initialise an array, $\hat{\text{x}}$, containing $K_x \times K_y \times K_z$ entries where $\hat{\text{x}} \in \mathbb{R}^{K_x \times K_y \times K_z}$ and $\hat{x}_{i,j,k}$ represents the entry in $\hat{\text{x}}$ corresponding to sphere $s_{i,j,k}$.
2. For each sphere $s_{i,j,k}$, where $1 \leq i \leq K_x$, $1 \leq j \leq K_y$, $1 \leq k \leq K_z$:
   (a) Find the closest point v to $c_{i,j,k}$ in the triangular surface description x.
   (b) Find the list of triangles L which involve v as l_m where $1 \leq m \leq M$ and M is the number of triangles in L.
   (c) Project $c_{i,j,k}$ onto the plane of each triangle in L as $c^m_{i,j,k}$, where $1 \leq m \leq M$, and use barycentric coordinates to determine if $c^m_{i,j,k}$ is in that triangle.
      i. If so, store the $c^m_{i,j,k}$ in list M.
   (d) For each triangle in L, $1 \leq m \leq M$:
      i. Project $c_{i,j,k}$ onto the plane of each triangle in L as $c^m_{i,j,k}$.
      ii. Determine via barycentric coordinates if $c^m_{i,j,k}$ is the mth triangle.
         A. If so, store $c^m_{i,j,k}$ in list M.
         B. Else, discard $c^m_{i,j,k}$.
   (e) If M is empty,
      i. Set $u = v$.
      ii. Set $\vec{n}$ as the average normal of the faces including v.
   (f) Else if M is non-empty
      i. Set u to be the point in M closest to $c_{i,j,k}$.
      ii. Set $\vec{n}$ as the normal of the triangle on which it was found.
   (g) Compute Q as the plane intersecting point u with normal $\vec{n}$.
   (h) If Q intersects with $s_{i,j,k}$, set $\hat{x}_{i,j,k}$ to be a value in [0...1] representing the intersection of Q with $s_{i,j,k}$ as a fraction of $2r_{i,j,k}$.
   (i) If Q does not intersect with $s_{i,j,k}$, then compute which side of Q that $s_{i,j,k}$ is in. If in front, $\hat{x}_{i,j,k} = 0$; else if behind, $\hat{x}_{i,j,k} = 1$.

The result of this process is a representation which *approximates* the volume of the 3D space within the grid being intersected by the surface design. To aid conceptualisation, in figure 3.9 we go through the generation of the attribute value for one of the sphere-cells in a design (refer to the figure caption).

We consider it an approximation of this because we spare ourselves costly triangle intersection tests across the whole design (as with canonical voxelisation), at the cost of some accuracy, instead performing a computationally simpler search for the nearest 3D point. This computational speed-up is further facilitated by the use of overlapping *spherical* cells (meaning that the grid need not be so granular as a voxelisation grid). In figure 3.10 is a 2D visualisation of KGrid operating on a surface design which shows how accuracy errors manifest (refer to figure for caption).

As with canonical ray-based voxelisation, KGrid can be made more efficient through use of space-partitioning techniques. For example, using a binary space partitioning (BSP) tree, one can avoid a number of unnecessary distance checks and observe a substantial computational speed up.

It is recommended that some dimensionality reduction technique then be applied to this representation before training of a regressor.

Immediate benefits

So what have we gained against other approaches? Making comparisons to the approaches in section 3.2, we see that KGrid:

1. Can be used on an arbitrary range of designs (compare to the reference model approach).
2. Is an approximation and so does not require exhaustive computation (compare to the canonical voxelisation approach).
3. Is non-stochastic and does not require a costly learning process.

3.7 Empirical analysis of KGrid

In this section we empirically analyse the performance of KGrid in a set of experiments. First we will discuss the application of ISOmap to clustered data, as this is slightly problematic and will require some careful thought. We then perform some useful experimental comparisons, using the

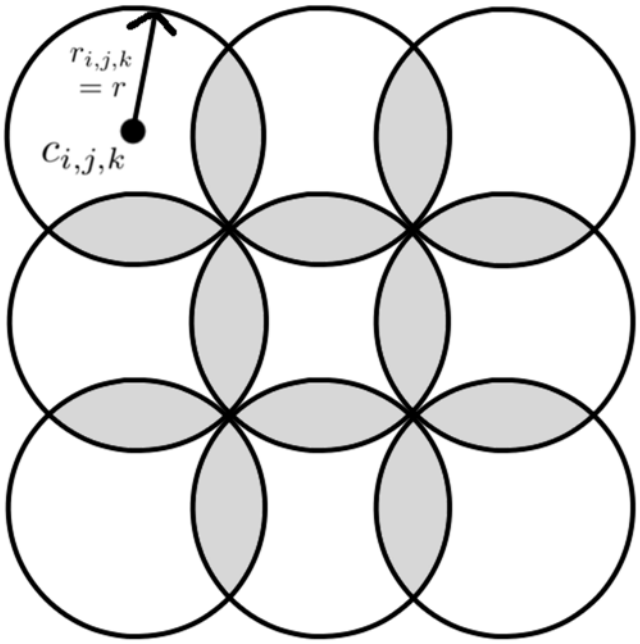


Figure 3.8: 2D visualisation of a uniform KGrid. Each sphere-cell has the same radius, the position of its centre is merely different. Note that each sphere overlaps with all of its neighbours (overlaps have been shaded grey). This also happens in the third dimension, such that there exists no space that is not within at least one sphere-cell.

results of these to draw conclusions regarding the effectiveness of our solution. Finally we will evaluate the method by the aims originally summarised in section 3.6.1.

3.7.1 Applying ISOmap to clustered data

Shortly we will perform experiments involving KGrid in order to evaluate its suitability to the task. Our experiments will involve PCA and ISOmap as dimensionality reduction techniques. We look at PCA because it is well established and understood, while we look at ISOmap for the reasons previously mentioned in section 3.5. Before we can do this though, we must first discuss how one applies ISOmap to data which is clustered, such as our data sets D^R and D^S.

Canonical ISOmap uses a nearest neighbourhood graph where each sample has k edge-connections, or conversely an ϵ value can be used to define the maximum distance at which two samples may be given an edge-connection in the graph. This nearest-neighbourhood graph is then used as basis for the generation of a shortest paths graph.

Reasonable values of either k or ϵ on a clustered data set such as ours will result in a shortest paths graph where the clusters are not inter-connected and so shortest paths aren't calculated

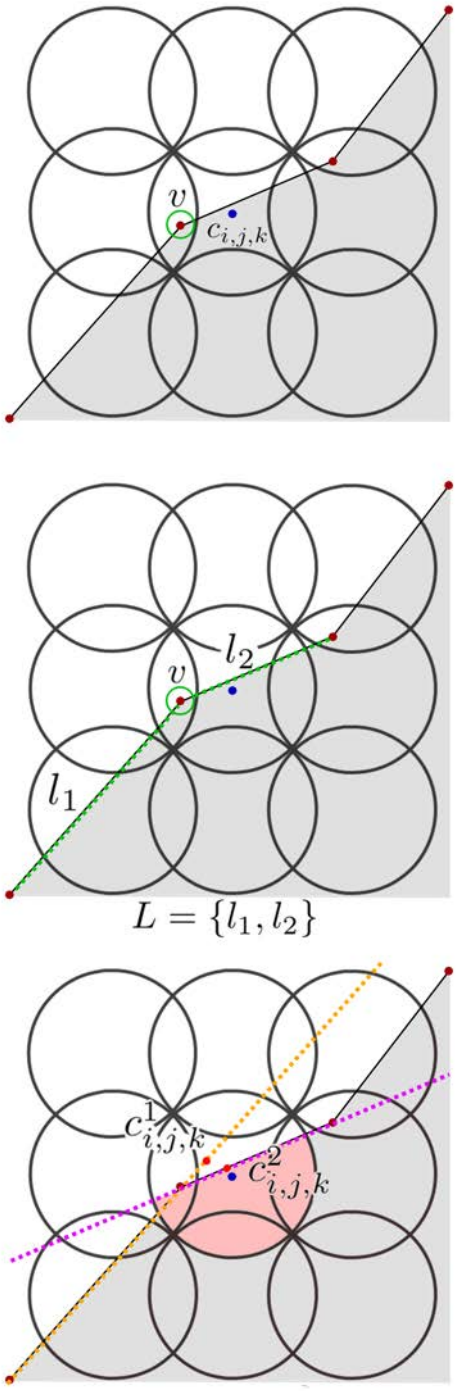


Figure 3.9: 2D visualisation of representation generation for a surface. Here, the intersection value for cell i, j, k is being calculated given a surface (represented as a black line with shading on the inside). In the first image, the nearest point to $c_{i,j,k}$ is identified, v. Following this, in the second image, we list all the primitives that share v in list $L = \{l_1, l_2\}$. Finally, given the planes (in 3D) formed of the each primitive in L, l_1, l_2, we project $c_{i,j,k}$ onto each (using the plane's normal), giving $c^1_{i,j,k}$ and $c^2_{i,j,k}$ respectively. A primitive intersection test with each of these shows that only $c^2_{i,j,k}$ is inside its respective primitive (l_2) and is thus goes into list M. As this is the only entry in list M, the plane Q is set to be the plane on which l_2 sits. The intersection of Q with the i, j, kth sphere-cell (shaded light-red) is then used to determine the value for cell i, j, k (here it might be ≈ 0.6).

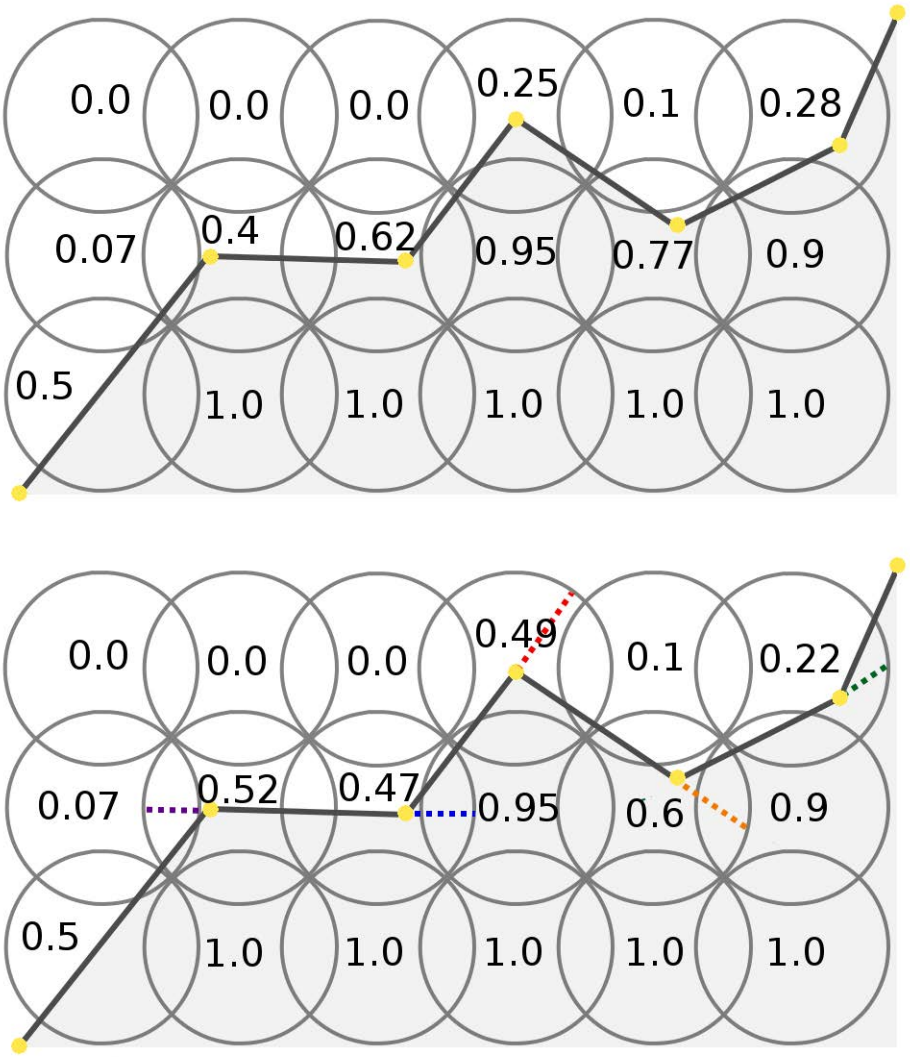


Figure 3.10: 2D visualisation showing how a surface is represented under KGrid. Each number (determined by eye) represents how much of the sphere (from one side to the other, rather than area) is intersected by the surface design (black outline, shaded grey inside). The values in the top image are what we might expect from an ideal approach. In saving computation, KGrid estimates these using just one face intersecting each primitive. Thus, the values in the bottom image are what KGrid would likely find (the coloured & dashed lines indicate the overlap of the plane used as Q in the calculation for that sphere, where applicable). Note the error in the fourth sphere along, top row.

between the members of two separate clusters. On the other hand, making k or ϵ large enough that all clusters connect under canonical ISOmap will damage the overall effectiveness of ISOmap. Canonical implementations of ISOmap allow the user to choose **one** of the shortest path graph clusters and to discard the other clusters along with their samples.

For this reason, canonical ISOmap is clearly not designed for use on clustered data sets such as our own. In fact, applying canonical ISOmap to D^C, a KGrid representation of D^R/D^S, or a voxelised representation of D^R results in eight shortest path graph clusters (one for each cluster generated in the data set). If we select a single cluster, clearly we discard the other 1050 samples: too many samples to lose.

In the literature there is no clear way of dealing with this in a way that would enable ISOmap to function effectively. In our tests we have looked at three ways of dealing with this "cluster bridging" problem. To reiterate, the problem is how to ensure that shortest-paths exist between all pairs of points, given a clustered data set for which the graph has been found using the k nearest neighbours approach:

- **Option 1:** After generating the initial shortest path graph, look for any pairs of points with no shortest path between them calculated (i.e. the default distance). Of these pairs of points, find the pair with shortest distance. Then make k connections from each of these two points to other points for which they have no shortest path (thus connecting their clusters).

- **Option 2:** After generating the initial shortest path graph, look for pairs of points with no shortest path between them calculated. Of this set of pairs without a shortest path between them, connect the k pairs of points which are closest to each other.

- **Option 3:** A combination of options 1 and 2.

Once one of these processes has been performed, the shortest path algorithm must be re-run based on the current state of the shortest-path graph. If at the end, we again find that there are pairs without a shortest path between them, we repeat the above (either variations 1, 2 or 3 depending on what was chosen). The process finishes when all pairs of points have a shortest path between them calculated.

We conducted experiments comparing each of the above cluster bridging approaches across a range of k values and ISOmapped dimensionalities. The experiments involved measuring

$d^{'}$	MSE	r^2	PCE	PPC
1	0.421	0.581	1.524	0.762
2	0.292	0.710	1.057	0.818
3	0.271	0.732	0.981	0.830
4	0.262	0.740	0.949	0.835
5	0.250	0.752	0.906	0.841
6	0.246	0.756	0.890	0.844
7	0.242	0.761	0.875	0.846
8	0.238	0.764	0.862	0.847
9	0.226	0.776	0.816	0.851
10	0.217	0.785	0.787	0.853
11	0.196	0.806	0.709	0.857
12	0.193	0.809	0.699	0.858
13	0.193	0.809	0.699	0.858
14	0.194	0.808	0.702	0.858
15	0.191	0.811	0.690	0.861
16	0.192	0.811	0.693	0.861
17	0.194	0.808	0.703	0.860
18	0.196	0.807	0.709	0.860
19	0.192	0.811	0.695	0.861
20	0.194	0.808	0.704	0.860

Table 3.2: Cluster bridging variation 1 performance scores. Results are averaged over 1000 runs.

evaluation scores of a regressor trained on the reduced dimensionality ISOmap representation of the KGrid representation of the realistic automobile data set D^R ($\tau = 30$). Through trial and error we found the best ISOmap k value for this data set to be $k = 5$. The results for each approach, averaged over 1000 runs for each value of $1 \leq d^{'} \leq 20$, are shown in tables 3.2, 3.3 and 3.4 respectively.

Variation 1 is seen to perform best in general, and this conclusion is supported by the statistical Wilcoxon Signed-Rank Test with p-values of <0.0001 given for all d. Therefore variation 1 is used in future experiments. Statistically, the difference between variations 2 and 3 were less clear sometimes and so conclusions cannot be made when comparing these.

We suggest that its formulation and application to the task of using ISOmap with clustered data is a novel contribution of this thesis.

3.7.2 Comparing the KGrid representation with the ideal representation D^C

For the first experiment we use the unfeasible 'ideal' automobile data set D^C as a benchmark, and assess the performance of the KGrid representation of the automobile data set D^R, denoted

d'	MSE	r^2	PCE	PPC
1	0.503	0.499	1.822	0.743
2	0.323	0.679	1.169	0.812
3	0.288	0.715	1.042	0.827
4	0.255	0.747	0.924	0.837
5	0.243	0.759	0.880	0.845
6	0.215	0.787	0.777	0.851
7	0.212	0.790	0.768	0.852
8	0.216	0.786	0.783	0.850
9	0.221	0.781	0.801	0.849
10	0.211	0.791	0.765	0.852
11	0.216	0.787	0.780	0.851
12	0.217	0.785	0.788	0.850
13	0.209	0.793	0.758	0.853
14	0.210	0.792	0.762	0.853
15	0.212	0.791	0.766	0.853
16	0.211	0.792	0.765	0.853
17	0.214	0.789	0.773	0.853
18	0.213	0.790	0.770	0.853
19	0.212	0.791	0.767	0.854
20	0.215	0.788	0.779	0.853

Table 3.3: Cluster bridging variation 2 performance scores. Results are averaged over 1000 runs.

d'	MSE	r^2	PCE	PPC
1	0.443	0.559	1.603	0.758
2	0.319	0.683	1.156	0.814
3	0.298	0.705	1.076	0.821
4	0.258	0.744	0.936	0.836
5	0.247	0.755	0.896	0.843
6	0.215	0.787	0.778	0.851
7	0.213	0.789	0.773	0.852
8	0.209	0.793	0.756	0.853
9	0.208	0.794	0.751	0.853
10	0.211	0.791	0.764	0.852
11	0.215	0.787	0.777	0.850
12	0.218	0.784	0.791	0.849
13	0.212	0.791	0.765	0.851
14	0.211	0.791	0.766	0.852
15	0.212	0.791	0.766	0.852
16	0.215	0.788	0.777	0.851
17	0.217	0.786	0.787	0.851
18	0.217	0.786	0.788	0.851
19	0.214	0.789	0.778	0.853
20	0.214	0.789	0.776	0.855

Table 3.4: Cluster bridging variation 3 performance scores. Results are averaged over 1000 runs.

D^K, in comparison. The experimental setup is as follows:

1. We perform some dimensionality reduction technique on the data set, taking the dimensionality of the original data from d down to $d^{'}$.

2. We split the 1200 samples with uniform selection into a training and testing set of 600 samples each.

3. Of the 600 training data we take 150 samples to one side as a validation set (leaving 450 samples for training).

4. We train an artificial neural network with a single hidden layer of 3 neurons for 400 iterations using back-propogation.

   (a) For each iteration we record the artificial neural network and its accuracy on the validation set of 150 samples.

5. We retain the artificial neural network which was shown to perform best on the 150 validation samples.

6. We record the performance of the artificial neural network in terms of the scores from chapter 2 with the testing data set.

7. We repeat steps 2 to 6 for 1000 runs and average the performance scores. These score averages are then indicative of each approach's performance.

We perform this experiment for various values of $d^{'}$ and as previously mentioned we use PCA and ISOmap in separate experiments as dimensionality reduction techniques.

For all experiments presented in this thesis we find that 3 hidden neurons suffice: in table 3.5 we see the experimental results when testing an artificial neural network using D^C. Although accuracy does increase slightly with each increase in hidden neurons, it is a very small change. Further, it was found that these small changes were consistent and proportional across all approaches presented throughout this thesis. Thus we regard it as acceptable to perform experiments with three hidden neurons, extrapolating our results for higher numbers of hidden neurons. Finally, keeping the number of hidden layer neurons low reduces computational

effort required to train the networks. Similarly, we choose to perform 400 iterations of back-propogation throughout this thesis because it was found experimentally that in all cases the best weights had been found (using the validation set) by the 300th iteration, and usually before the 200th iteration.

The τ parameter of KGrid is set to 30 when generating D^K from D^R, as this results in a representation that performs well (in comparison with τ values between 10 and 75) and consists of a reasonable number of attributes (i.e. a number which can be worked with in 8GB of computer memory). The data is split into equal sized training and testing sets (600 samples each) as is common in machine learning literature. The number of runs performed is maintained at 1000 for all of this thesis, as it results in a large-enough sample set of results from highly statistically significant observations and conclusions can be made.

In table 3.6 we see the results where PCA has been used (the MSE, r^2 and PPC columns are plotted in the second diagram of figures 3.11, 3.12 and 3.13 respectively). All results were tested for significance with the Wilcoxon Signed-Rank test, with boldface indicating $p < 0.0001$. As expected, the ideal representation D^C performs best, but encouragingly, the KGrid representation is seen to perform quite well despite lacking the advantages of D^C (consider that we might expect a much higher discrepancy from the ideal representation). It should be noted that $d^{'} = 2$ seems to have been a sweet-spot resulting in a similar performance in both approaches with $0.001 < p < 0.1$ across the scores.

It is interesting to create scatter plots of estimated-output against target-output for assessing machine learning tasks like this. Since it is seen at $d^{'} = 2$ that neither has a statistically significant score we choose to do this here. We generated scatter plots for the first three runs and looked for the best correlation in them. The scatter plots from the two best of these runs are shown in figure 3.14 (with KGrid at the top and D^C at the bottom). We see positive correlations with similar strength in both, but note that the correlation in the KGrid plot is stronger between -3 and 0, as it has managed to reach the lower values.

In table 3.7 we see a slightly different pattern when ISOmap is used instead of PCA (the MSE, r^2 and PPC columns are plotted in the first diagram of figures 3.11, 3.12 and 3.13 respectively). We found that for this data set, $k = 5$ resulted in the most accurate results. Interestingly, KGrid with ISOmap performs comparatively well for all samples, with the KGrid with ISOmap combination even outperforming D^C with ISOmap very slightly from $d^{'} = 11$ to

Hidden neurons	MSE	r^2	PCE	PPC
3	0.15	0.851	0.542	0.885
5	0.146	0.854	0.53	0.886
7	0.145	0.856	0.526	0.887
9	0.145	0.856	0.524	0.888

Table 3.5: Comparison of accuracy scores for artificial neural networks with varying numbers of hidden neurons in their single hidden layers. Data generated based on D^C.

$d' = 19$. At $d' = 20$ the performances were interestingly quite similar, with p-scores of 0.0074, 0.1676 and 0.0308 for MSE, r^2 and PCE respectively.

Although the comparison being made is between KGrid and the ideal representation (i.e. D^K and D^C) we should also compare the results under PCA and ISOmap. When we do this, we see that for this automobile data set, the approach involving PCA clearly outperforms all other approaches. The failings of ISOmap to perform well are likely due to the automobile data set being unsuitable for ISOmap, i.e. it contains no bump-like features to exploit and its samples cannot be easily placed on a low-dimensional manifold.

Despite this, we can conclude from both tables that KGrid is forming a representation that is indeed useful to the machine learning process, performing comparatively well to D^C when ISOmap was used, and noticeably worse when PCA was used (and yet, still achieving scores useful to the machine learning process). Based on table 3.7 it can also be argued that KGrid generates a representation which is more useful to ISOmap than even the ideal representation is able to.

3.7.3 Comparing the KGrid representation with the voxelised representation

The second experiment compares the KGrid representation of the automobile data set D^R, denoted D^K, with a voxelised representation. KGrid can be perceived as a continuous approximation of canonical voxelisation, and so this comparison is particularly interesting.

KGrid sets up its sphere-cells equidistant from one another based on a single parameter, τ (which we set to 30 as in the last experiment). This results in a KGrid resolution of $30 \times 18 \times 11$. The extremes of the voxelised space along each axis are also set to those used for the KGrid (such that all samples fit within the voxelised space).

Utilising the same experimental setup as in the previous experiment (section 3.7.2) we compare voxelisation and KGrid via the accuracy scores of a regressor. As in the previous experiment

	KGrid with PCA				D^C with PCA			
$d^{'}$	MSE	r^2	PCE	PPC	MSE	r^2	PCE	PPC
1	0.768	0.237	2.783	0.667	**0.513**	**0.491**	**1.858**	**0.752**
2	0.437	0.563	1.583	0.761	0.433	0.568	1.570	0.761
3	0.271	0.731	0.978	0.834	**0.236**	**0.764**	**0.856**	**0.850**
4	0.240	0.761	0.869	0.848	**0.174**	**0.826**	**0.632**	**0.875**
5	0.218	0.783	0.791	0.855	**0.153**	**0.848**	**0.554**	**0.884**
6	0.197	0.803	0.714	0.865	**0.145**	**0.856**	**0.526**	**0.889**
7	0.197	0.804	0.714	0.866	**0.147**	**0.854**	**0.534**	**0.886**
8	0.202	0.799	0.733	0.863	**0.144**	**0.857**	**0.520**	**0.886**
9	0.185	0.816	0.670	0.872	**0.129**	**0.872**	**0.468**	**0.891**
10	0.176	0.825	0.638	0.874	**0.123**	**0.878**	**0.444**	**0.893**
11	0.180	0.821	0.651	0.872	**0.121**	**0.879**	**0.440**	**0.894**
12	0.176	0.825	0.637	0.875	**0.123**	**0.878**	**0.444**	**0.893**
13	0.173	0.829	0.624	0.877	**0.125**	**0.876**	**0.452**	**0.892**
14	0.165	0.836	0.599	0.880	**0.119**	**0.882**	**0.429**	**0.895**
15	0.158	0.843	0.573	0.881	**0.113**	**0.888**	**0.410**	**0.898**
16	0.158	0.843	0.571	0.881	**0.112**	**0.889**	**0.405**	**0.899**
17	0.153	0.848	0.554	0.884	**0.113**	**0.888**	**0.409**	**0.898**
18	0.151	0.850	0.547	0.885	**0.114**	**0.887**	**0.413**	**0.898**
19	0.145	0.856	0.525	0.886	**0.115**	**0.886**	**0.417**	**0.897**
20	0.146	0.855	0.529	0.886	**0.116**	**0.885**	**0.419**	**0.897**

Table 3.6: Comparison of representations for use with PCA. $\tau = 30$. Averaged over 1000 runs.Boldface indicates "best score for that $d^{'}$ with a statistical significance of $p < 0.0001$". $d^{'}$ is the dimensionality of the PCA embedding.

	KGrid with ISOmap				D^C with ISOmap			
$d^{'}$	MSE	r^2	PCE	PPC	MSE	r^2	PCE	PPC
1	**0.421**	**0.581**	**1.524**	**0.762**	0.438	0.564	1.586	0.754
2	0.291	0.709	1.055	0.817	**0.284**	**0.718**	**1.027**	**0.826**
3	0.271	0.731	0.980	0.830	**0.258**	**0.742**	**0.937**	**0.837**
4	0.262	0.739	0.948	0.834	**0.245**	**0.756**	**0.889**	**0.840**
5	0.250	0.751	0.905	0.840	**0.241**	**0.760**	**0.873**	**0.844**
6	0.246	0.755	0.889	0.843	**0.232**	**0.769**	**0.841**	**0.848**
7	0.242	0.760	0.875	0.845	**0.231**	**0.770**	**0.837**	**0.849**
8	0.238	0.763	0.861	0.846	**0.215**	**0.787**	**0.778**	**0.852**
9	0.225	0.776	0.815	0.850	**0.213**	**0.789**	**0.770**	**0.853**
10	0.217	0.784	0.786	0.852	**0.213**	**0.789**	**0.771**	0.854
11	**0.196**	**0.805**	**0.708**	0.856	0.211	0.791	0.763	0.856
12	**0.193**	**0.808**	**0.698**	0.857	0.210	0.792	0.759	0.857
13	**0.193**	**0.808**	**0.698**	0.857	0.207	0.795	0.750	**0.858**
14	**0.194**	**0.807**	**0.701**	0.857	0.199	0.802	0.721	**0.863**
15	**0.191**	**0.811**	**0.689**	0.860	0.196	0.806	0.708	**0.864**
16	**0.191**	**0.810**	**0.693**	0.860	0.198	0.803	0.718	**0.862**
17	**0.194**	**0.808**	**0.702**	0.859	0.199	0.803	0.721	**0.861**
18	**0.195**	**0.806**	**0.708**	0.859	0.202	0.801	0.729	**0.861**
19	**0.191**	**0.810**	**0.694**	0.860	0.202	0.799	0.732	**0.861**
20	0.194	0.808	0.703	0.860	0.196	0.806	0.710	**0.866**

Table 3.7: Comparison of representations for use with ISOmap. $K = 5$ and $\tau = 30$. Averaged over 1000 runs. Boldface indicates "best score for that $d^{'}$ with a statistical significance of $p < 0.0001$". $d^{'}$ is the dimensionality of the ISOmap embedding.

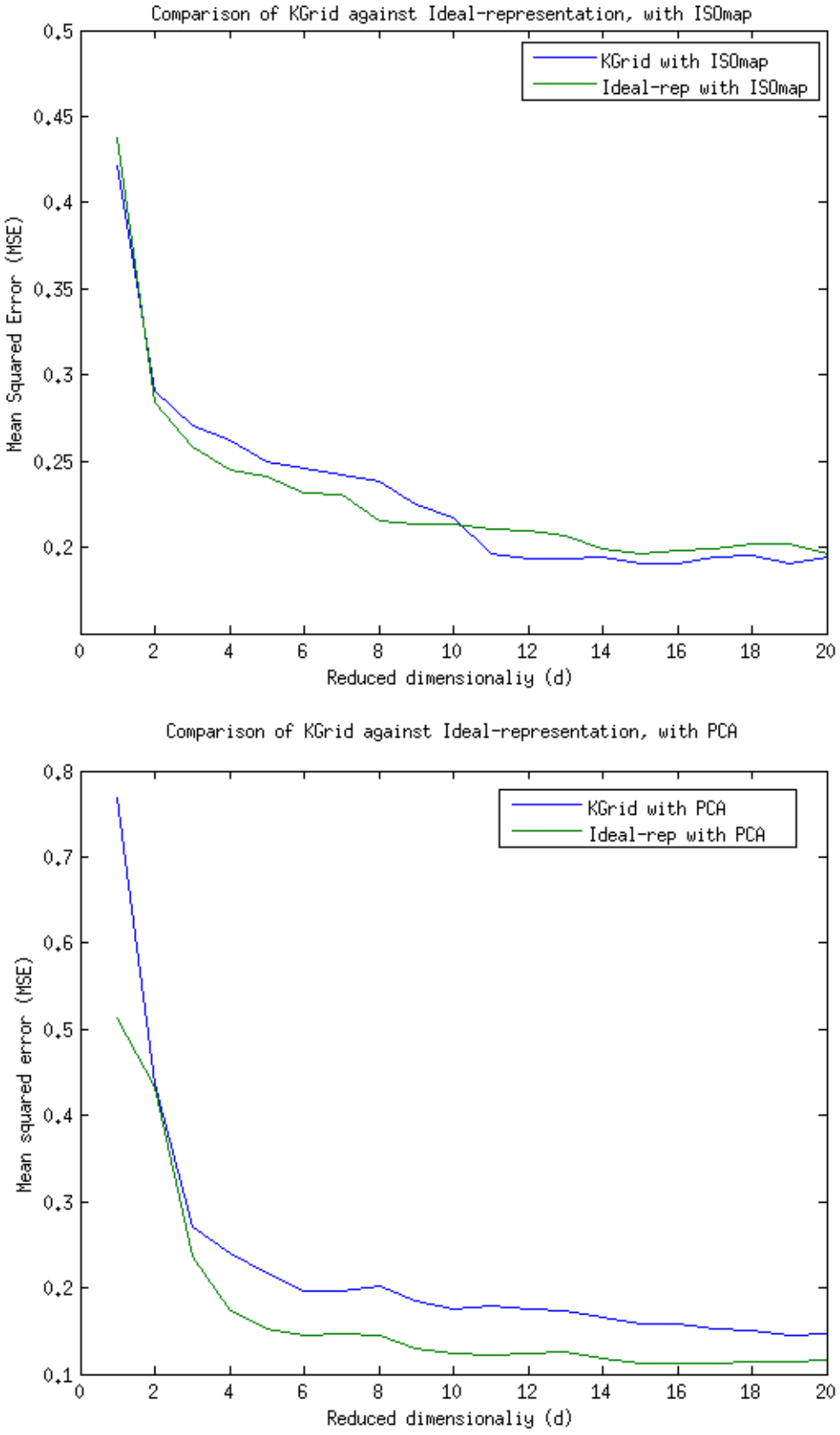


Figure 3.11: Plots comparing KGrid with the ideal representation D^C in terms of MSE.

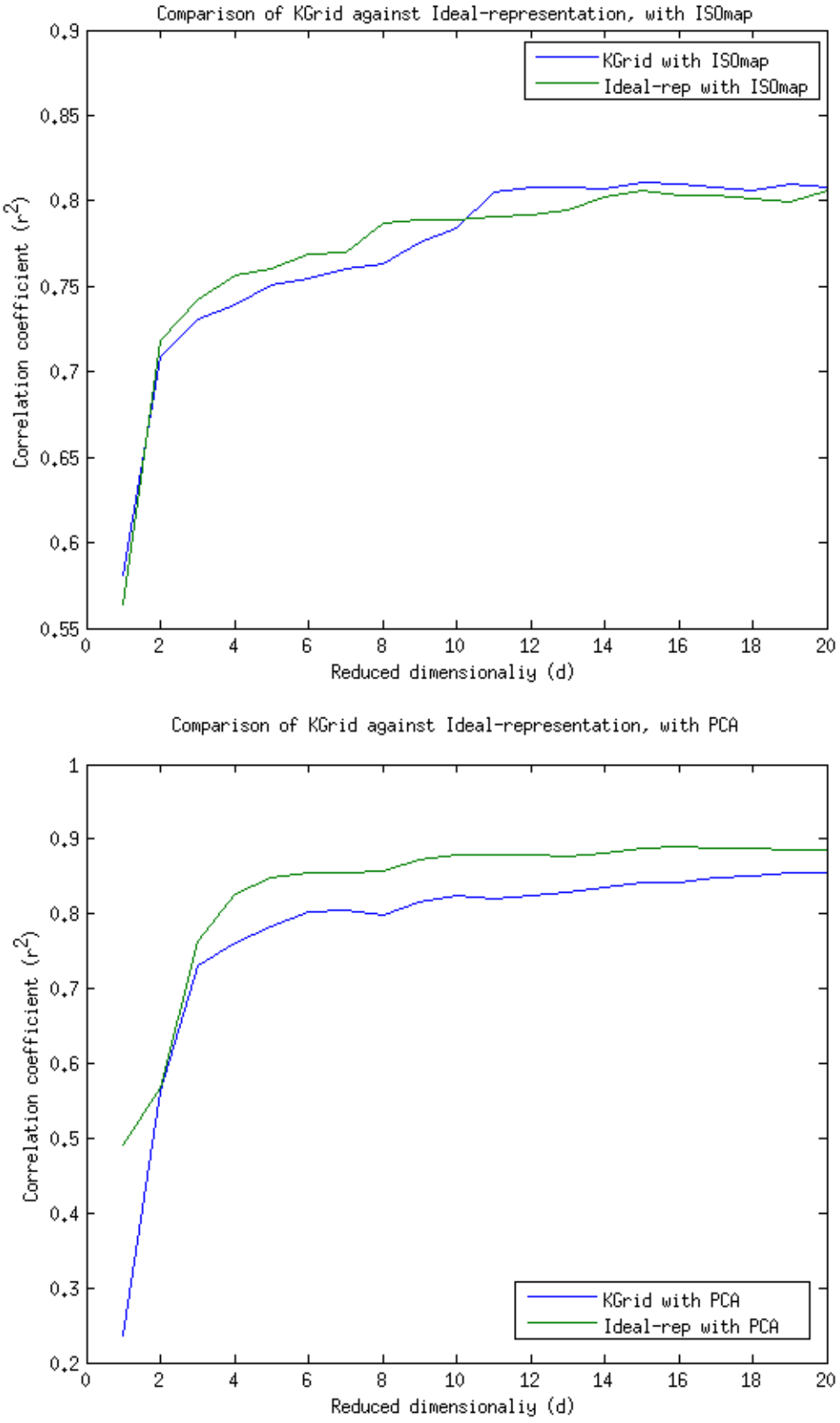


Figure 3.12: Plots comparing KGrid with the ideal representation D^C in terms of r^2.

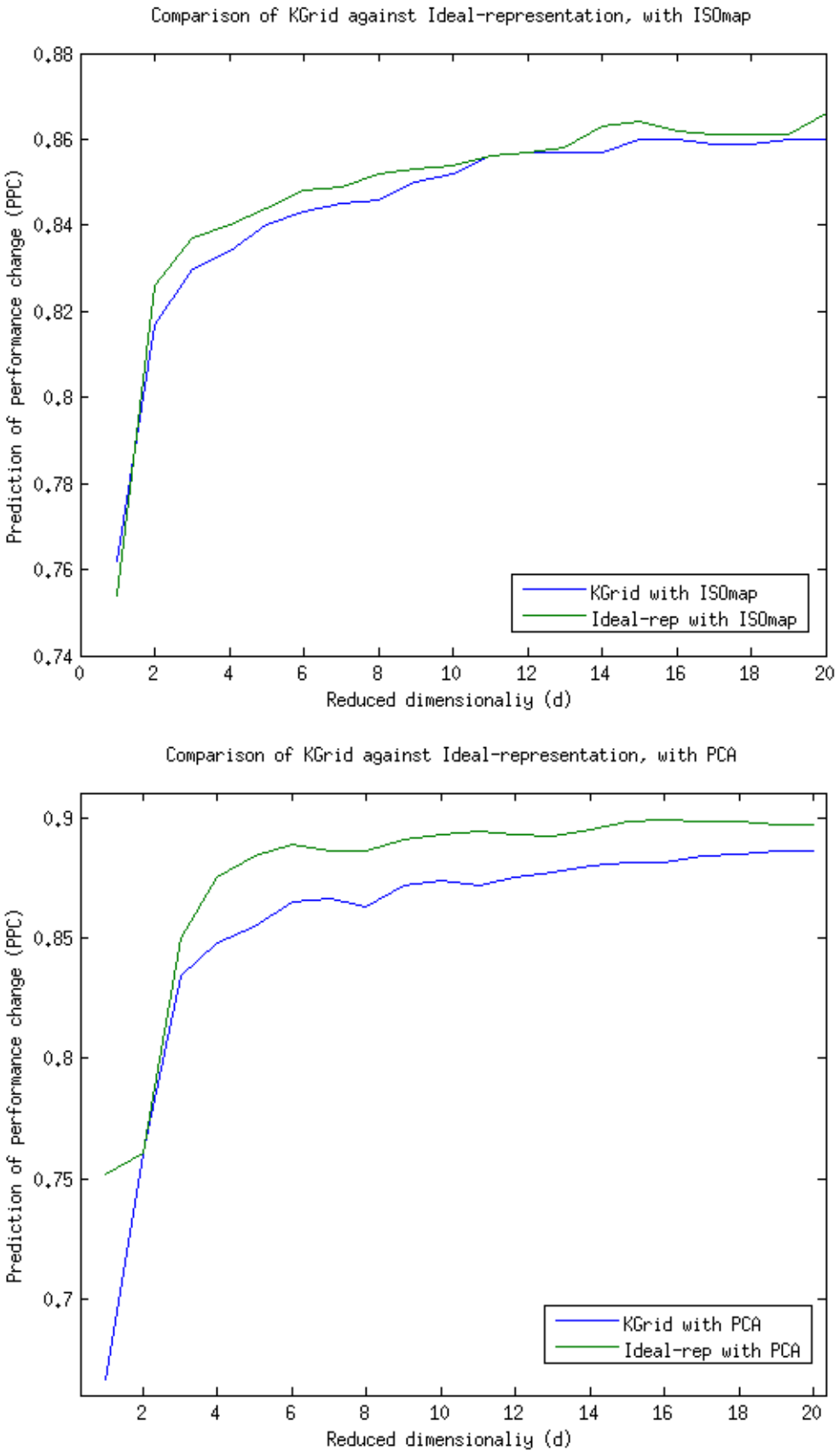


Figure 3.13: Plots comparing KGrid with the ideal representation D^C in terms of PPC.

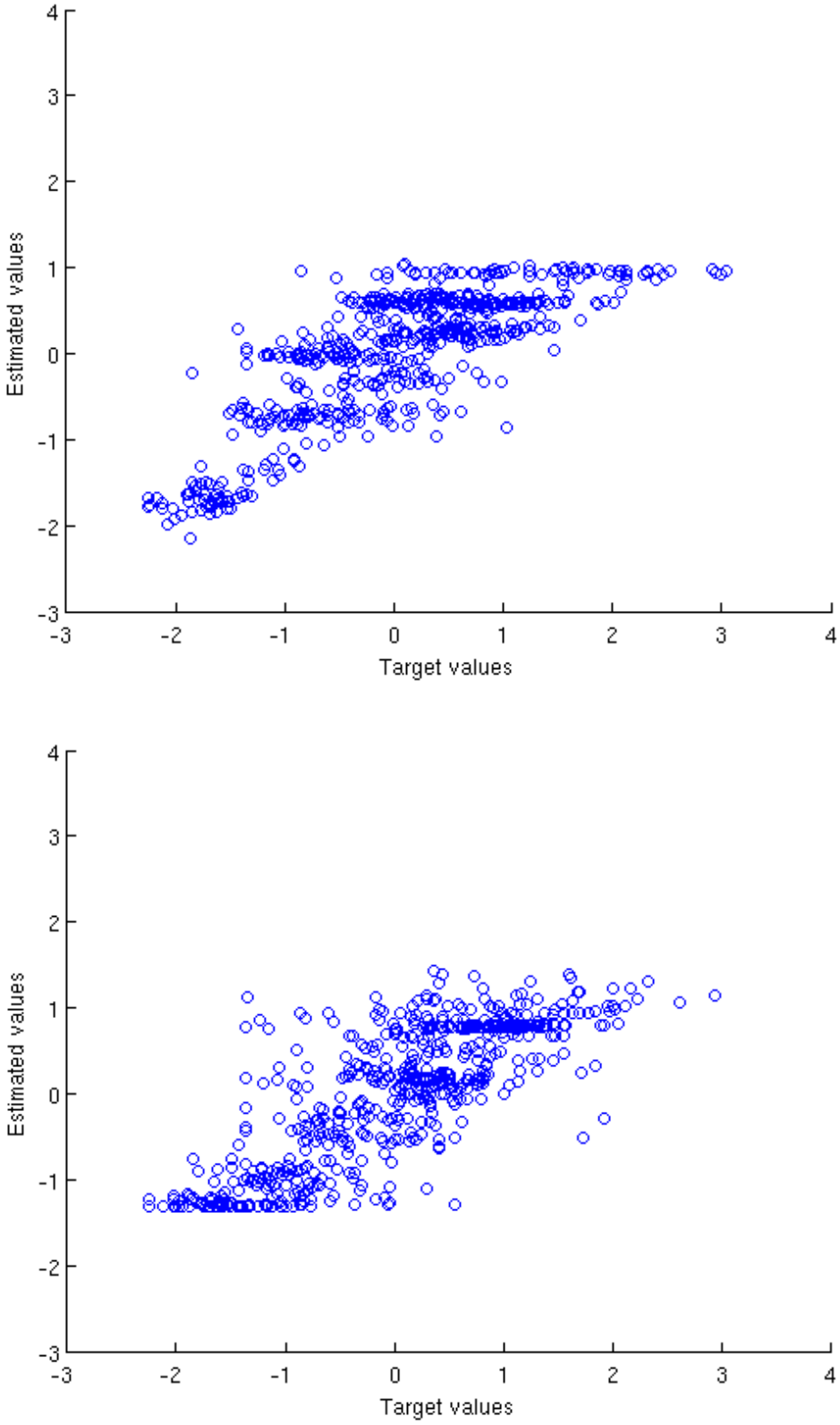


Figure 3.14: Scatter plots of estimated-output against the target-output values. Top image: KGrid with PCA at $d^{'} = 2$. Bottom image: D^{C} with PCA at $d^{'} = 2$.

we use ISOmap and PCA as dimensionality techniques in separate experiments.

The results of the comparison where ISOmap has been used are in table 3.8 (the MSE, r^2 and PPC columns are plotted in the first diagram of figures 3.15, 3.16 and 3.17 respectively). We see that generally they start off similarly, with each approach doing well for roughly half of the cases where $d' < 11$. The significance tests (performed via the Wilcoxon signed-rank test) show that, while in terms of accuracy scores (MSE, r^2, PCE) it is hard to distinguish the two approaches for these lower d' values, in terms of the ranking metric, PPC, the voxelisation generally performs significantly better for a larger range of d' values. Things change considerably when $d' \geq 11$. From this point onwards the KGrid representation results in a consistently and significantly more accurate (i.e. in terms of MSE, r^2, PCE) regression function. Although for $11 \leq d' \leq 14$ the PPC score of neither approach is significantly better, at $d' \geq 15$ the KGrid approach comes out on top for all scores, including PPC.

Again we look at some scatter plots. This time we ran the experiments for $d' = 3$ (since no statistical significance was seen for either r^2 score). Both plots are quite similar but the plot for the voxelisation data set appears to have a slightly stronger correlation.

The results of the comparison where PCA has been used are in table 3.9 (the MSE, r^2 and PPC columns are plotted in the second diagram of figures 3.15, 3.16 and 3.17 respectively). Curiously for $d' = 1$, KGrid achieves a significantly better PPC score, while voxelisation proves significantly better in terms of r^2. Then for $2 \leq d' \leq 3$, the voxelisation approach proves significantly better. From this point onwards though, the voxelisation approach never performs significantly better than the KGrid approach for any score. Where $6 \leq d' \leq 11$, KGrid significantly takes the upper hand in at least one of the scores for five values of d'. Then at $d' = 12$ the KGrid approach takes a consistent and significant lead through to $d' = 20$. Note that the effectiveness of PCA as dimensionality reduction for machine learning problems means that it gets considerably lower than the results in table 3.8. Further, the performance continues to improve as d' increases (whereas for the ISOmap comparison the scores seemed to flatten at $d' \approx 15$).

The scatter plots for these results show that both approaches perform similarly. In figure 3.19 we can see the scatter plots for $d' = 11$ and they appear very similar (although perhaps the correlation seems stronger in the -3 and 0 again).

This data seems to confirm that KGrid can compete well with voxelisation, and out-perform it at dimensions past a threshold (here $d' \approx 10$). We note that KGrid, Voxelisation, ISOmap

	KGrid with ISOmap				Voxelisation with ISOmap			
$d^{'}$	MSE	r^2	PCE	PPC	MSE	r^2	PCE	PPC
1	0.421	0.581	1.524	0.762	**0.388**	**0.615**	**1.404**	**0.772**
2	**0.291**	**0.709**	**1.055**	0.817	0.310	0.692	1.121	0.815
3	0.271	0.731	0.980	0.830	0.269	0.732	0.975	**0.834**
4	0.262	0.739	0.948	0.834	**0.255**	**0.746**	**0.924**	**0.840**
5	**0.250**	**0.751**	**0.905**	**0.840**	0.262	0.739	0.950	0.836
6	**0.246**	**0.755**	**0.889**	**0.843**	0.254	0.747	0.920	0.840
7	0.242	0.760	0.875	0.845	**0.236**	**0.765**	**0.856**	**0.848**
8	0.238	0.763	0.861	0.846	**0.230**	**0.771**	**0.833**	**0.850**
9	**0.225**	**0.776**	**0.815**	0.850	0.230	0.771	0.834	0.850
10	0.217	0.784	0.786	0.852	**0.214**	**0.788**	**0.773**	**0.854**
11	**0.196**	**0.805**	**0.708**	0.856	0.206	0.795	0.747	0.856
12	**0.193**	**0.808**	**0.698**	0.857	0.204	0.797	0.739	0.857
13	**0.193**	**0.808**	**0.698**	0.857	0.205	0.797	0.742	0.856
14	**0.194**	**0.807**	**0.701**	0.857	0.203	0.798	0.733	0.857
15	**0.191**	**0.811**	**0.689**	**0.860**	0.205	0.796	0.742	0.857
16	**0.191**	**0.810**	**0.693**	**0.860**	0.205	0.797	0.741	0.857
17	**0.194**	**0.808**	**0.702**	**0.859**	0.205	0.796	0.744	0.857
18	**0.195**	**0.806**	**0.708**	**0.859**	0.206	0.795	0.747	0.857
19	**0.191**	**0.810**	**0.694**	**0.860**	0.207	0.794	0.750	0.856
20	**0.194**	**0.808**	**0.703**	**0.860**	0.210	0.792	0.760	0.855

Table 3.8: Comparison of KGrid and voxelisation for ISOmap with $K = 5$ and $\tau = 30$ and a voxelised data set of $30 \times 18 \times 11$ binary voxels. Averaged over 1000 runs. $d^{'}$ is the dimensionality of the ISOmap embedding. Boldface indicates "best score for that $d^{'}$ with a statistical significance of $p < 0.0001$".

and PCA are each non-stochastic processes and the variation in the data comes from the the regression model and splitting of data into training and testing sets. It would seem then, that KGrid is indeed a good alternative to voxelisation for higher dimensionalities.

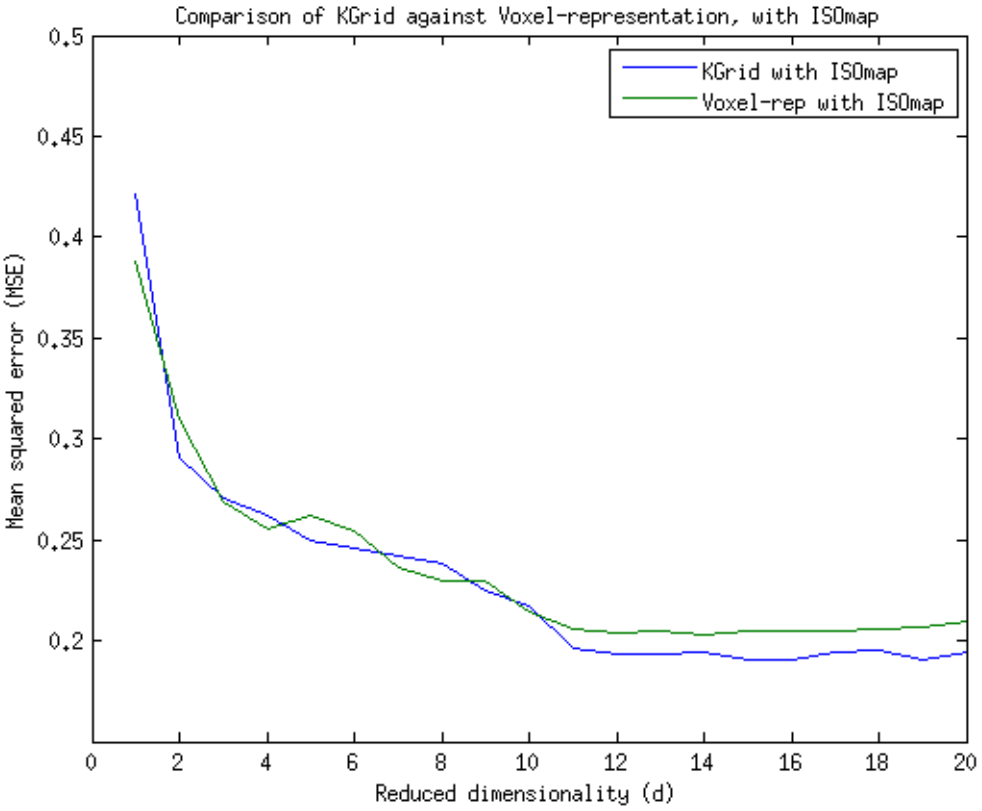
The benefit is further highlighted when one considers the computational simplicity of KGrid compared to voxelisation and that for our problem the end goal is a designer-assistance system, which must take as input the designer's current draft and return functional performance as quickly as possible.

3.7.4 Comparing ISOmap and PCA for dealing with the bump problem

The third experiment involves the comparison of the KGrid representation of the sphere data set D^S with a voxelised representation of this data set. Experimental parameters were changed accordingly to reflect the size of D^S (225 testing samples and 225 training, with 56 validation samples taken from this) and the number of clusters in D^S (three).

	KGrid with PCA				Voxelisation with PCA			
$d^{'}$	MSE	r^2	PCE	PPC	MSE	r^2	PCE	PPC
1	0.768	0.237	2.783	**0.667**	0.761	**0.245**	2.757	0.665
2	0.437	0.563	1.583	0.761	**0.315**	**0.686**	**1.142**	**0.813**
3	0.271	0.731	0.978	0.834	**0.256**	**0.745**	**0.927**	**0.841**
4	0.240	0.761	0.869	0.848	0.240	0.760	0.873	0.848
5	0.218	0.783	0.791	0.855	0.216	0.785	0.781	0.855
6	**0.197**	**0.803**	**0.714**	0.865	0.201	0.800	0.727	0.864
7	0.197	0.804	0.714	**0.866**	0.200	0.801	0.725	0.864
8	**0.202**	**0.799**	**0.733**	**0.863**	0.209	0.792	0.757	0.861
9	0.185	0.816	0.670	**0.872**	0.184	0.817	0.666	0.870
10	**0.176**	**0.825**	**0.638**	**0.874**	0.185	0.817	0.669	0.868
11	0.180	0.821	0.651	0.872	0.179	0.822	0.649	0.872
12	**0.176**	**0.825**	**0.637**	**0.875**	0.180	0.822	0.651	0.874
13	**0.173**	**0.829**	**0.624**	**0.877**	0.178	0.823	0.646	0.874
14	**0.165**	**0.836**	**0.599**	**0.880**	0.174	0.827	0.629	0.877
15	**0.158**	**0.843**	**0.573**	**0.881**	0.172	0.830	0.621	0.878
16	**0.158**	**0.843**	**0.571**	**0.881**	0.168	0.833	0.607	0.879
17	**0.153**	**0.848**	**0.554**	**0.884**	0.167	0.834	0.606	0.879
18	**0.151**	**0.850**	**0.547**	**0.885**	0.169	0.832	0.612	0.878
19	**0.145**	**0.856**	**0.525**	**0.886**	0.161	0.841	0.581	0.880
20	**0.146**	**0.855**	**0.529**	**0.886**	0.160	0.841	0.581	0.880

Table 3.9: Comparison of KGrid and voxelisation for PCA with $\tau = 30$ and a voxelised data set of $30 \times 18 \times 11$ binary voxels. Averaged over 1000 runs. $d^{'}$ is the dimensionality of the PCA embedding. Boldface indicates "best score for that $d^{'}$ with a statistical significance of $p < 0.0001$".



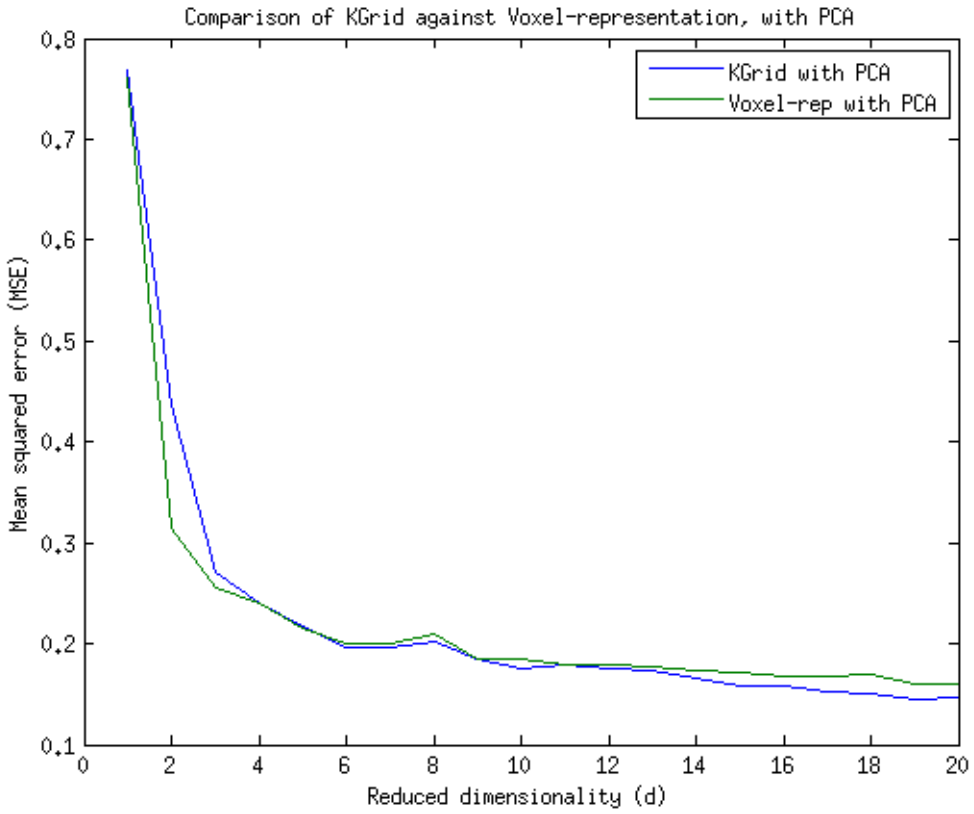
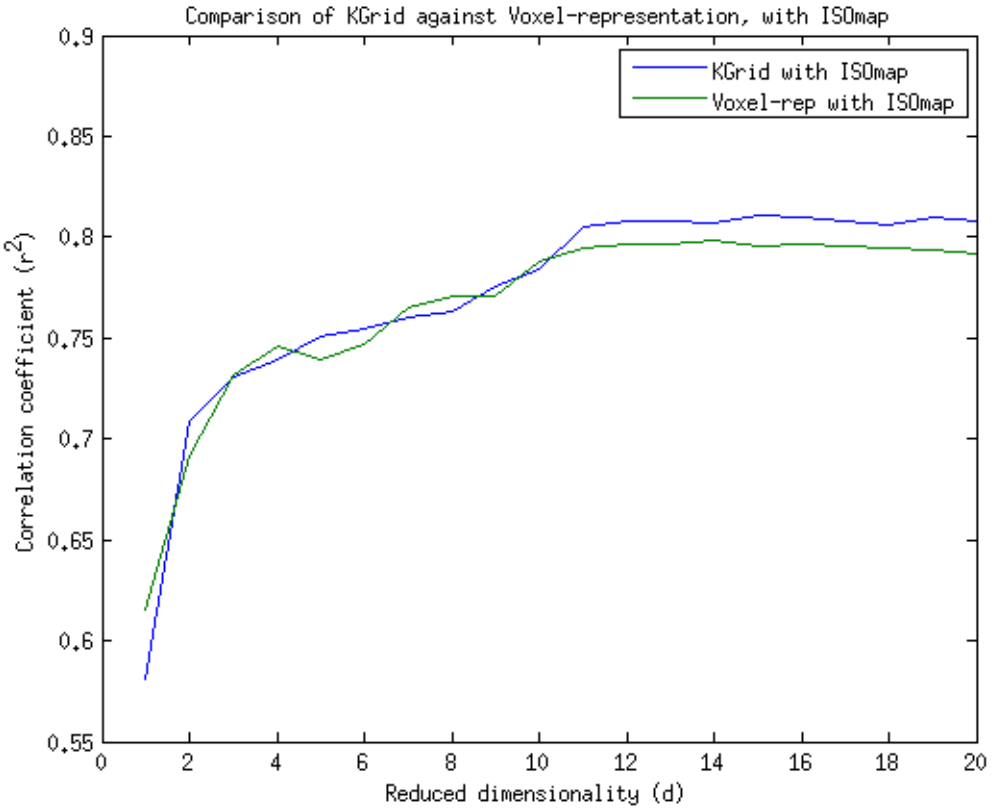


Figure 3.15: Plots comparing KGrid with the voxelised representation in terms of MSE.



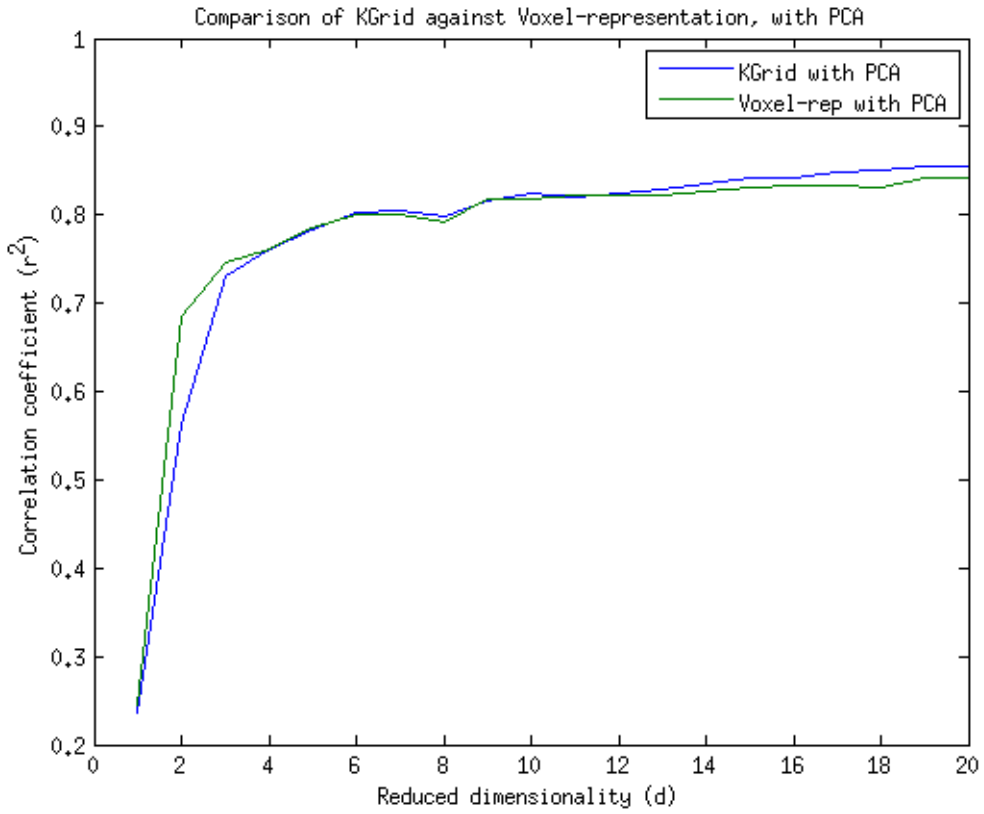
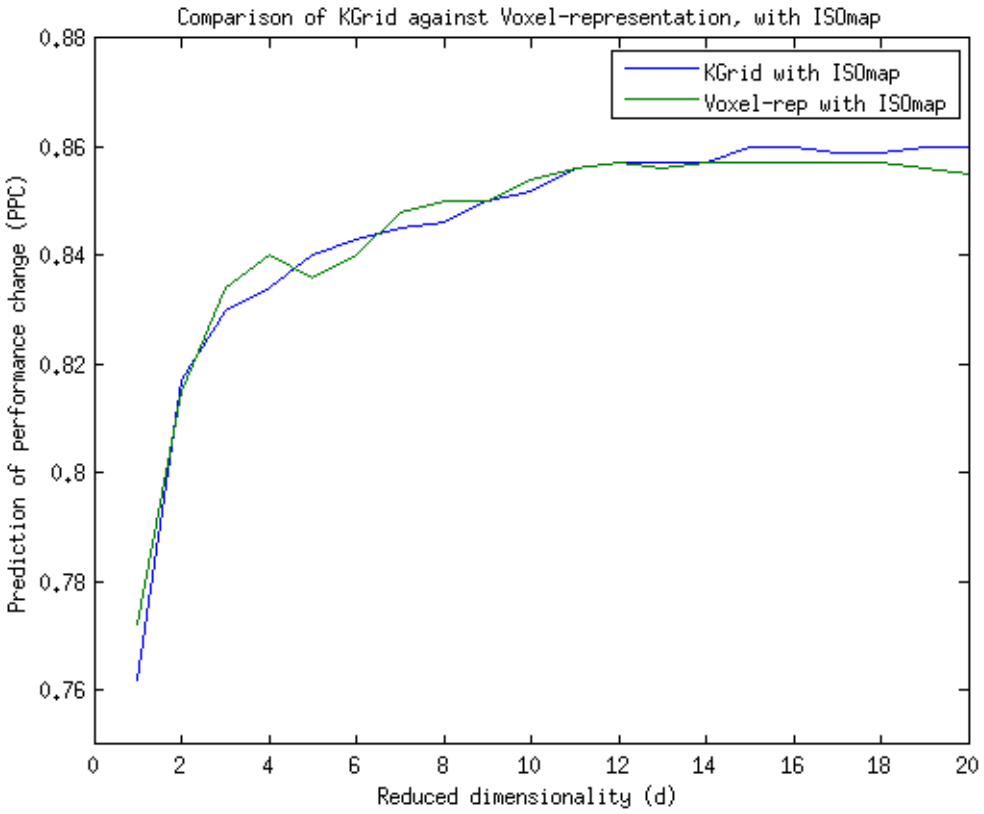


Figure 3.16: Plots comparing KGrid with the voxelised representation in terms of r^2.



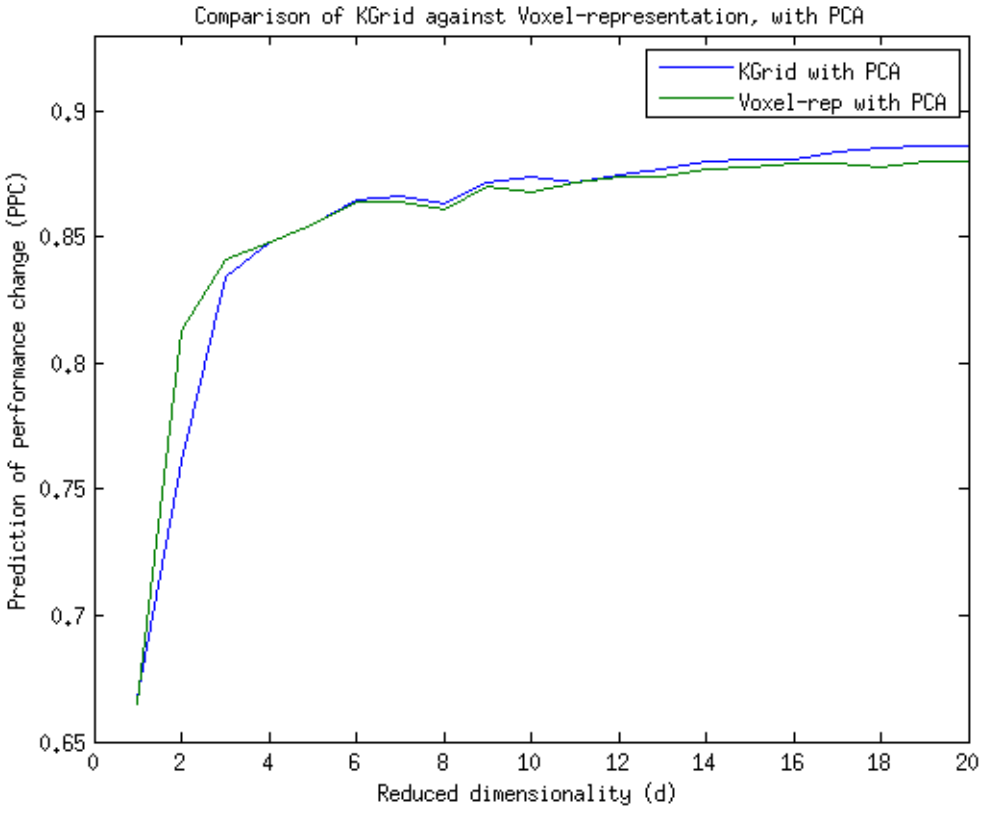


Figure 3.17: Plots comparing KGrid with the voxelised representation in terms of PPC.

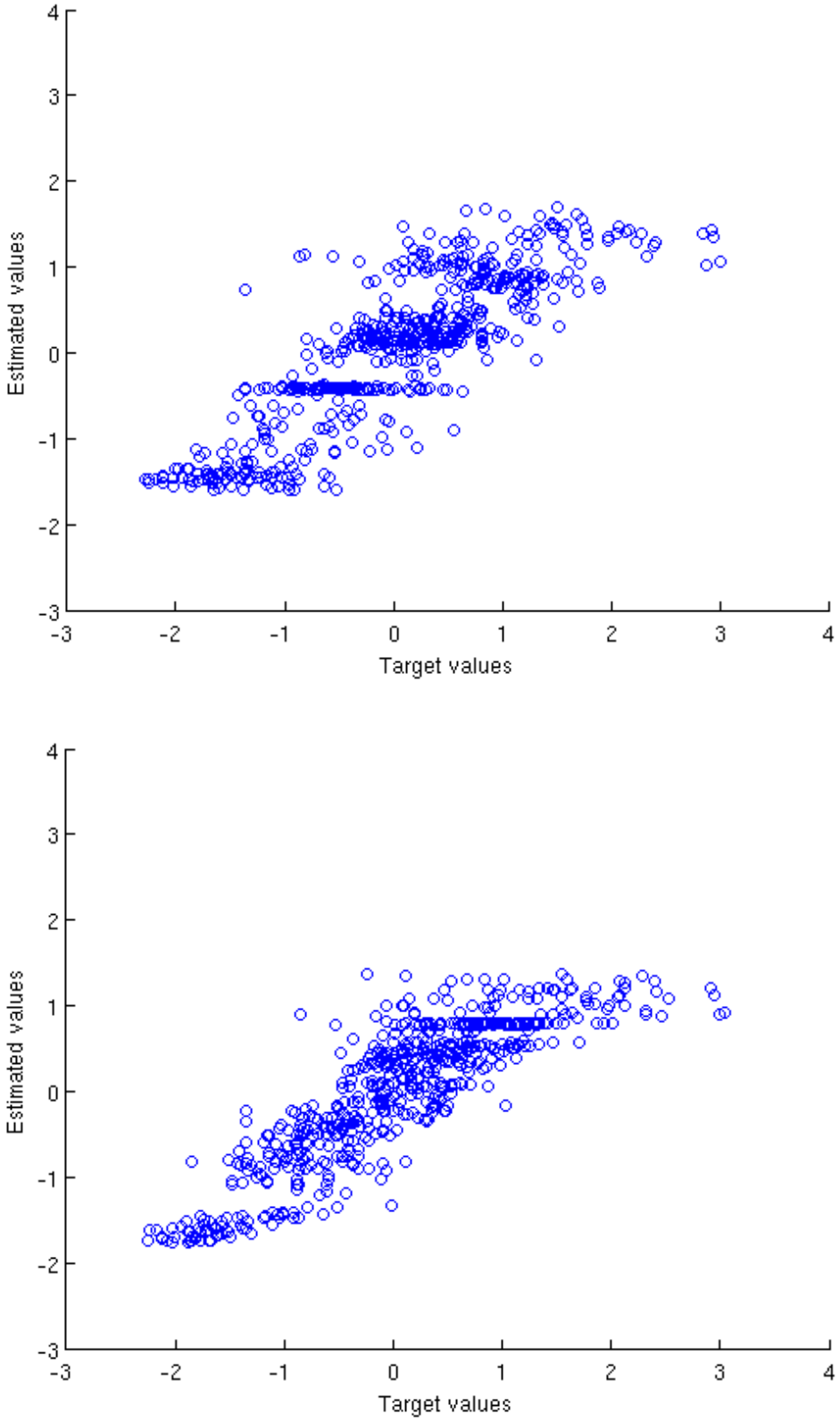


Figure 3.18: Scatter plots of estimated-output against the target-output values. Top image: KGrid with ISOmap at $d^{'} = 3$. Bottom image: Voxelisation with ISOmap at $d^{'} = 3$.

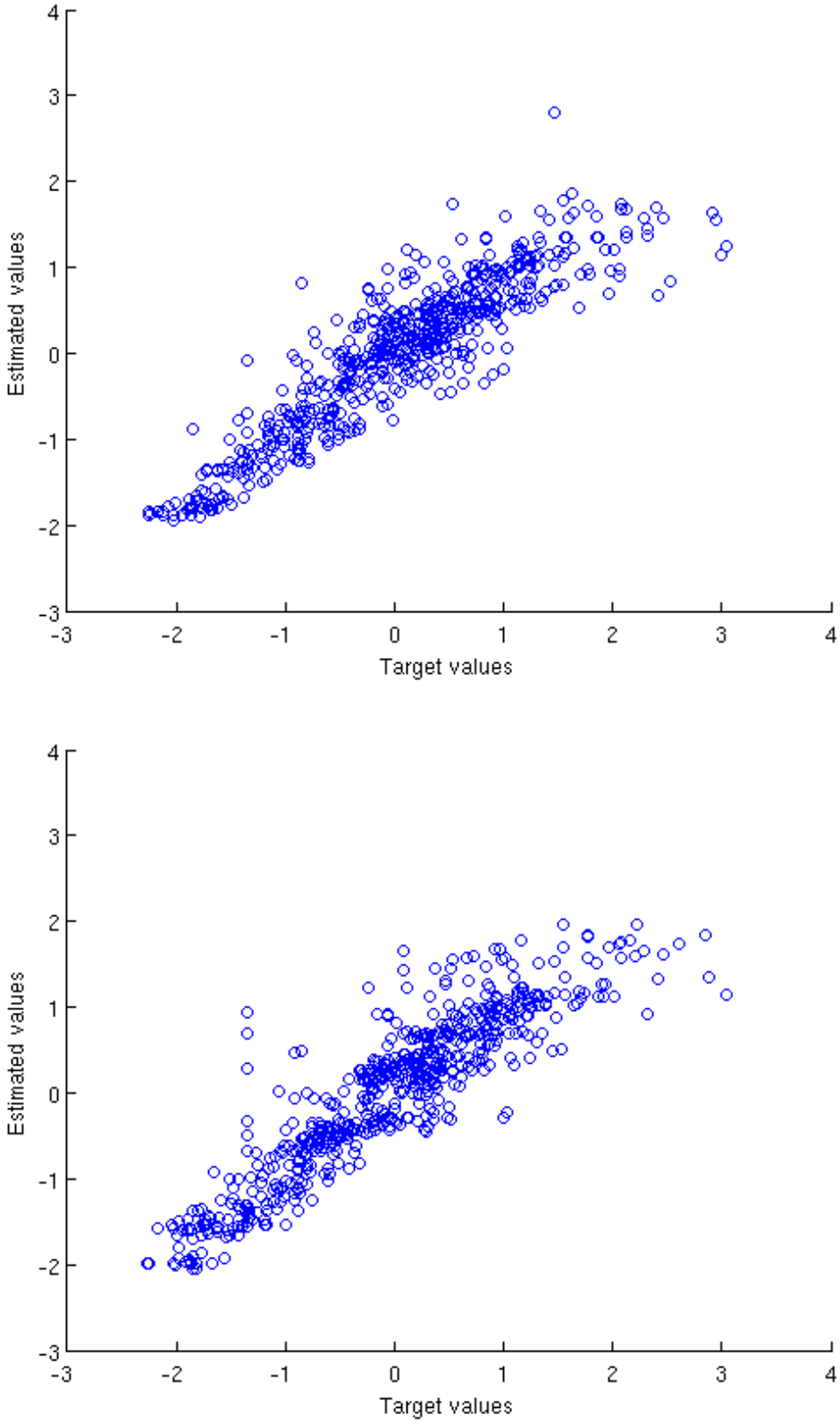


Figure 3.19: Scatter plots of estimated-output against the target-output values. Top image: KGrid with PCA at $d^{'} = 11$. Bottom image: Voxelisation with PCA at $d^{'} = 11$.

For this experiment we set up our spheres with $\tau = 130$. To arrive at this value, we tested the process with τ values ranging from 30 to 200, in increments of 10. Through this method, 130 was found to perform best, even better than tests at $\tau = 125$ and $\tau = 135$ performed. This resulted in a very large grid resolution $130 \times 105 \times 94$, but noting that (i) the designs are symmetrical, and (ii) design changes ('bumps') stretch over the z-axis with no variation, we were able to reduce the z-width of the KGrid-contained-space and thus achieve a KGrid resolution of $130 \times 105 \times 1$ without losing any information relevant to machine learning of the problem. This same KGrid resolution will be used in the experiments coming up in chapter 5.

Utilising the same experimental setup as in the previous experiments (sections 3.7.2 and 3.7.3) we compare voxelisation and KGrid via the accuracy scores of a regressor. As in the previous experiments we use ISOmap and PCA as dimensionality techniques in separate experiments. For this data set, the best k parameter for ISOmap was found to be 2.

Tables 3.10 and 3.11 show the performance scores where ISOmap and PCA have been used respectively. We see again that KGrid seems capable of competing well with voxelisation, supporting our earlier claim that it offers a good alternative to KGrid. We note in particular that in terms of accuracy, the regression functions it results in clearly outperform the voxelisation approach at $d = 1...2$. We note than when used with PCA, there ends up being little significant difference between either approach, while with ISOmap the voxelisation approach ends up performing a little better. These comparisons are illustrated in figures 3.20 through 3.22.

When we compare the two tables, 3.10 and 3.11, we see that ISOmap seems significantly more capable of generating a representation of the spheres data set for use in machine learning, leading to a dramatic increase in our performance metrics. This is what we had hoped to observe when we generated the spheres data set in section 3.5. These comparisons are illustrated strongly in figures 3.23 through 3.25.

In the top image of figure 3.21 we see the r^2 scores for the voxelised and KGrid representations both reach similar high-points (at $d = 6$ and $d = 8$ respectively). It is therefore interesting to perform a scatter plot of at these points for each approach. These are generated similarly to our previous scatter plots and we show the two 'best' plots (of three generated) in figure 3.26. We see generally see strong positive correlations, particularly from about 0 onwards. Certainly, in this figure the voxelisation approach seems to have achieved a slightly stronger plot.

It seems then, that the experiments with the spheres data set support the claims of the

	KGrid with ISOmap				Voxelisation with ISOmap			
d'	MSE	r^2	PCE	PPC	MSE	r^2	PCE	PPC
1	**0.335**	**0.667**	**2.268**	**0.763**	0.420	0.586	2.846	0.756
2	**0.254**	**0.749**	**1.721**	**0.801**	0.334	0.671	2.262	0.774
3	0.244	0.758	1.652	0.815	0.264	0.740	1.785	0.816
4	0.236	0.768	1.595	0.827	0.232	0.773	1.571	**0.838**
5	0.238	0.765	1.612	0.831	0.225	0.778	1.524	**0.846**
6	0.207	0.798	1.399	0.848	**0.152**	**0.881**	**1.029**	**0.890**
7	0.194	0.812	1.312	0.860	**0.145**	**0.880**	**0.978**	**0.889**
8	0.123	0.883	0.835	0.886	0.150	0.869	1.016	0.885
9	**0.141**	0.873	**0.953**	0.885	0.154	**0.879**	1.046	**0.898**
10	0.171	0.850	1.156	0.881	0.193	0.852	1.309	**0.891**
11	0.178	0.839	1.205	0.876	0.193	**0.849**	1.305	**0.889**
12	0.195	0.826	1.316	0.873	0.209	0.840	1.415	**0.887**
13	0.221	0.806	1.497	0.867	**0.210**	**0.832**	**1.424**	**0.884**
14	0.233	0.793	1.577	0.863	**0.226**	**0.816**	**1.532**	**0.881**
15	0.242	0.784	1.634	0.862	**0.219**	**0.815**	**1.484**	**0.879**
16	0.289	0.767	1.960	0.859	**0.233**	**0.804**	**1.575**	**0.876**
17	0.315	0.735	2.137	0.851	**0.272**	**0.783**	**1.841**	**0.872**
18	0.412	0.711	2.789	0.845	**0.277**	**0.774**	**1.878**	**0.869**
19	0.347	0.708	2.351	0.843	**0.296**	**0.760**	**2.005**	**0.864**
20	0.314	0.746	2.127	0.861	0.302	0.745	2.043	0.860

Table 3.10: Comparison of KGrid and voxelisation representations of the spheres data set for ISOmap with $K = 2$ and $\tau = 130$ and a voxelised data set of $130 \times 105 \times 1$ binary voxels. Averaged over 1000 runs. d' is the dimensionality of the ISOmap embedding. Boldface indicates "best score for that d' with a statistical significance of $p < 0.0001$".

previous subsection: KGrid is a suitable alternative to voxelisation. But more importantly, they show that non-linear dimensionality reduction approaches such as ISOmap are considerably superior to linear approaches such as PCA for handling problems exhibiting the bump problem. Curiously we also note that KGrid is superior to voxelisation for both dimensionality reduction approaches where $d < 3$.

3.7.5 Evaluation of KGrid by the original aims

In section 3.6.1 we listed some properties that our solution should aim to exhibit. In this section we will evaluate whether our solution, KGrid, has suitably demonstrated that it exhibits these properties.

	KGrid with PCA				Voxelisation with PCA			
d'	MSE	r^2	PCE	PPC	MSE	r^2	PCE	PPC
1	**0.647**	**0.360**	**4.384**	0.724	0.649	0.358	4.397	0.724
2	**0.648**	**0.364**	**4.384**	0.723	0.650	0.363	4.398	0.722
3	0.650	**0.363**	**4.399**	0.725	0.650	0.362	4.400	0.725
4	0.648	0.359	4.387	0.723	0.650	0.358	4.399	0.723
5	**0.649**	**0.360**	**4.392**	0.725	0.652	0.357	4.409	**0.727**
6	**0.650**	0.360	4.400	0.726	0.652	0.358	4.412	**0.727**
7	0.650	0.357	4.405	0.727	**0.649**	0.359	4.395	0.725
8	0.650	0.361	4.400	0.725	0.646	0.365	4.375	0.721
9	0.651	0.359	4.409	**0.725**	0.650	0.361	4.400	0.724
10	0.650	0.360	4.398	0.725	0.650	0.360	4.402	0.724
11	0.644	0.362	4.363	0.722	0.645	0.362	4.370	0.720
12	**0.652**	**0.359**	**4.412**	0.726	0.655	0.356	4.433	**0.729**
13	0.651	0.358	4.406	0.727	0.648	0.361	4.385	0.723
14	0.650	0.361	4.399	0.726	0.650	0.361	4.400	0.724
15	0.648	0.360	4.385	0.723	0.648	0.360	4.387	0.722
16	0.654	0.356	4.423	0.728	0.649	0.361	4.390	0.723
17	0.651	0.360	4.406	0.726	0.647	0.364	4.378	0.721
18	0.650	0.361	4.397	0.725	0.649	0.362	4.391	0.723
19	0.652	0.359	4.415	0.725	0.649	0.361	4.398	0.722
20	0.656	0.357	**4.439**	0.727	0.656	0.357	4.443	0.727

Table 3.11: Comparison of KGrid and voxelisation representations of the spheres data set for PCA with $\tau = 130$ and a voxelised data set of $130 \times 105 \times 1$ binary voxels. Averaged over 1000 runs. d' is the dimensionality of the PCA embedding. Boldface indicates "best score for that d' with a statistical significance of $p < 0.0001$".

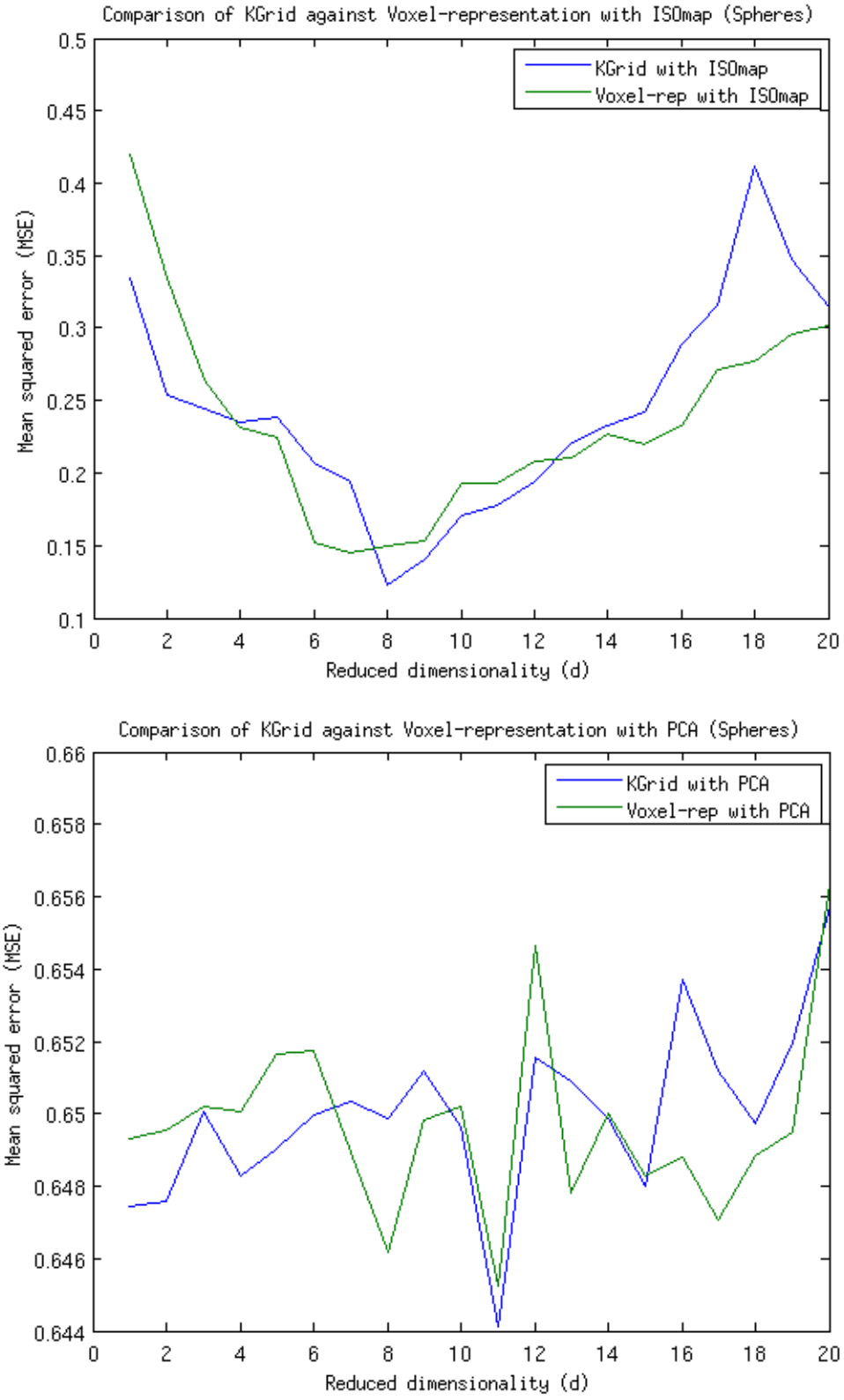


Figure 3.20: Plots comparing KGrid with the voxelised representation in terms of MSE for the spheres data set.

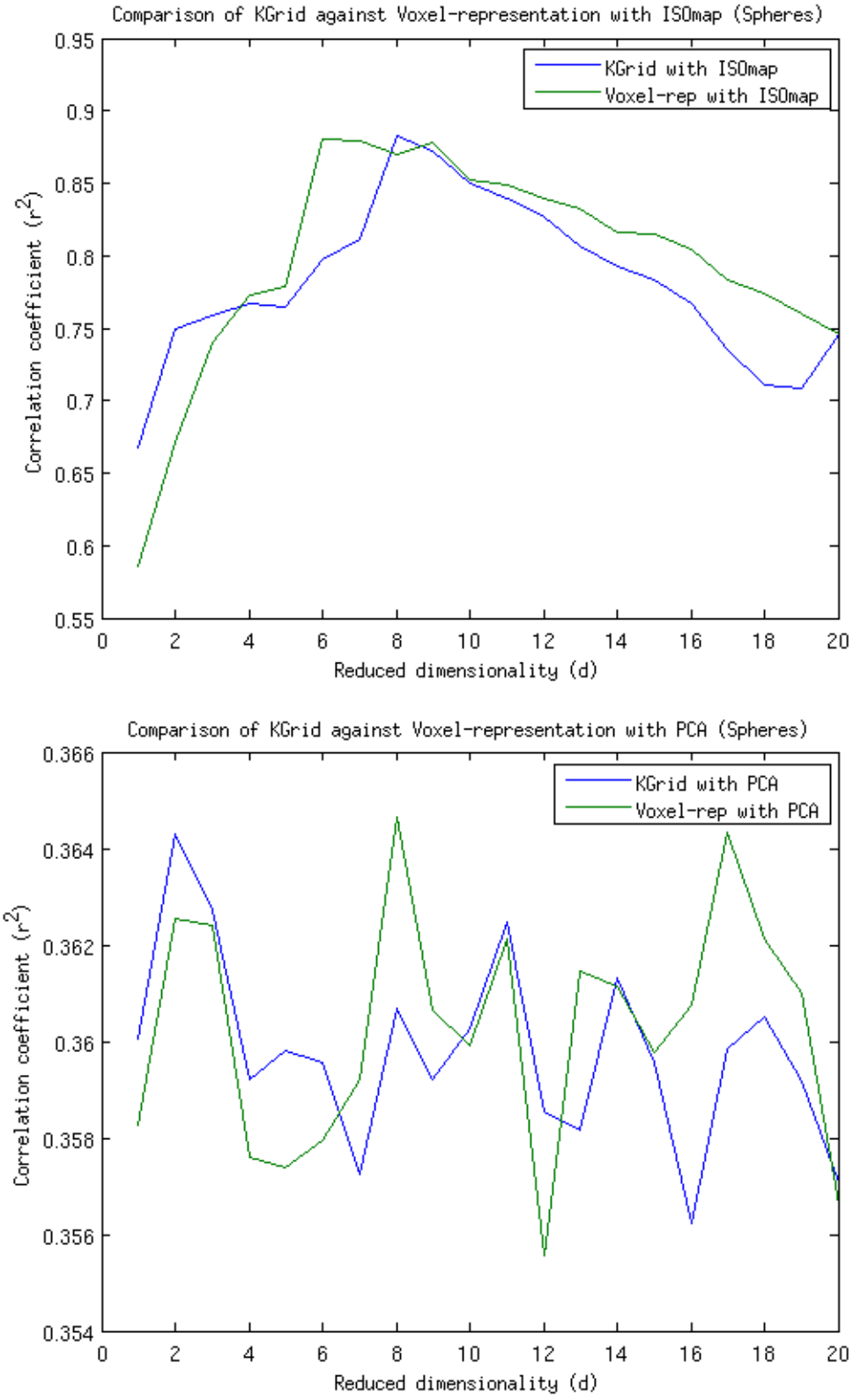


Figure 3.21: Plots comparing KGrid with the voxelised representation in terms of r^2 for the spheres data set.

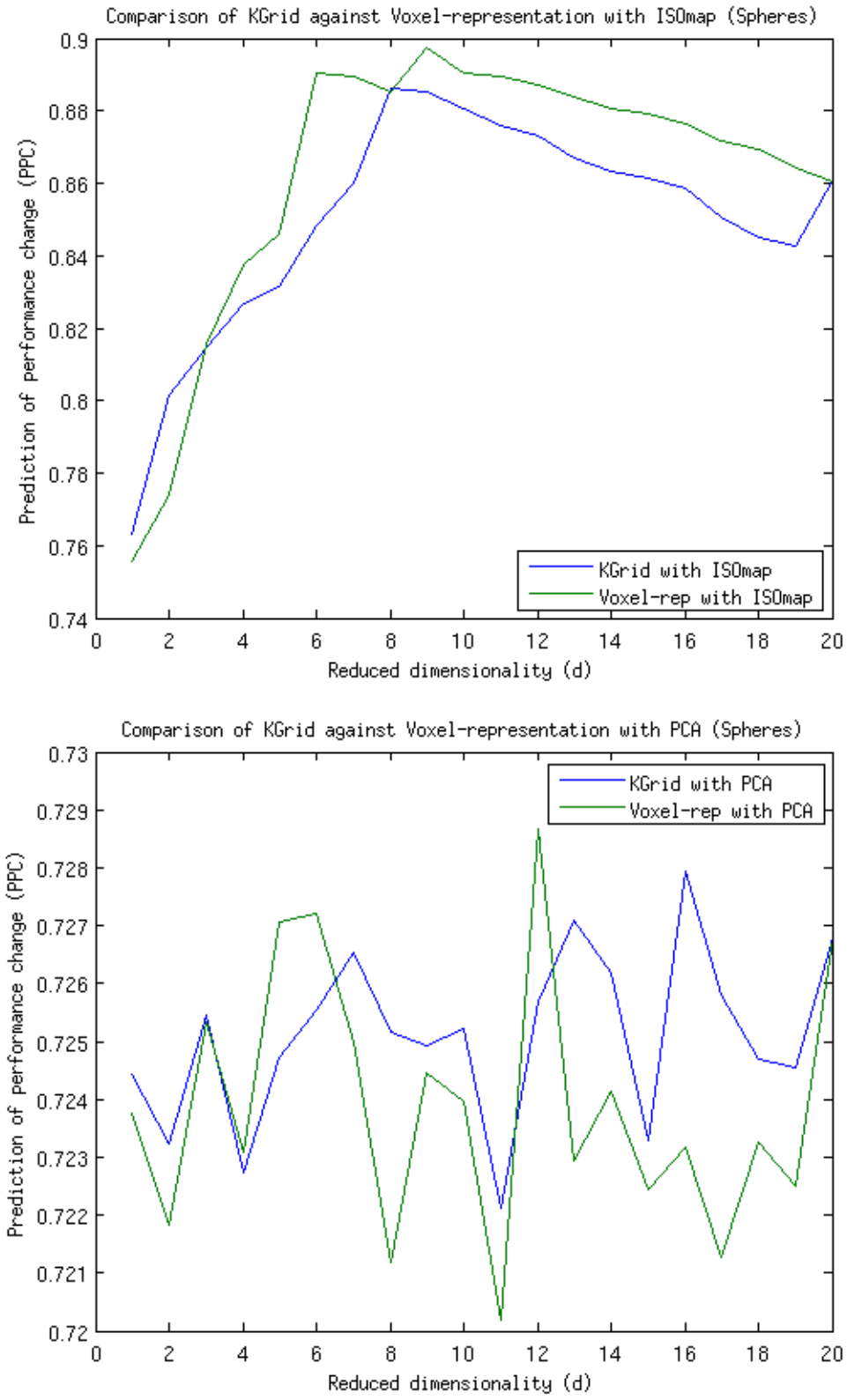


Figure 3.22: Plots comparing KGrid with the voxelised representation in terms of PPC for the spheres data set.

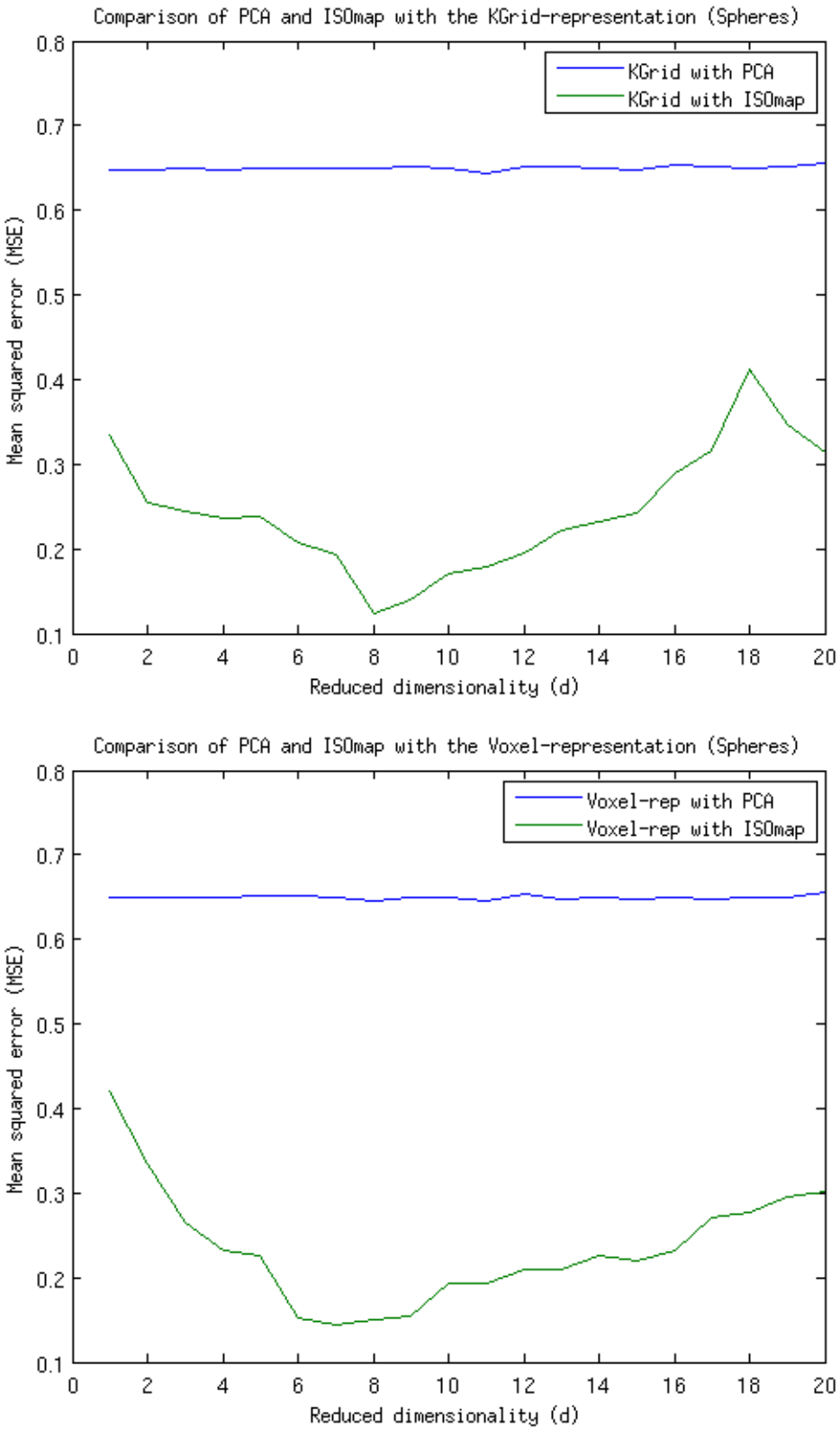


Figure 3.23: Plots comparing ISOmap with PCA in terms of MSE for the spheres data set.

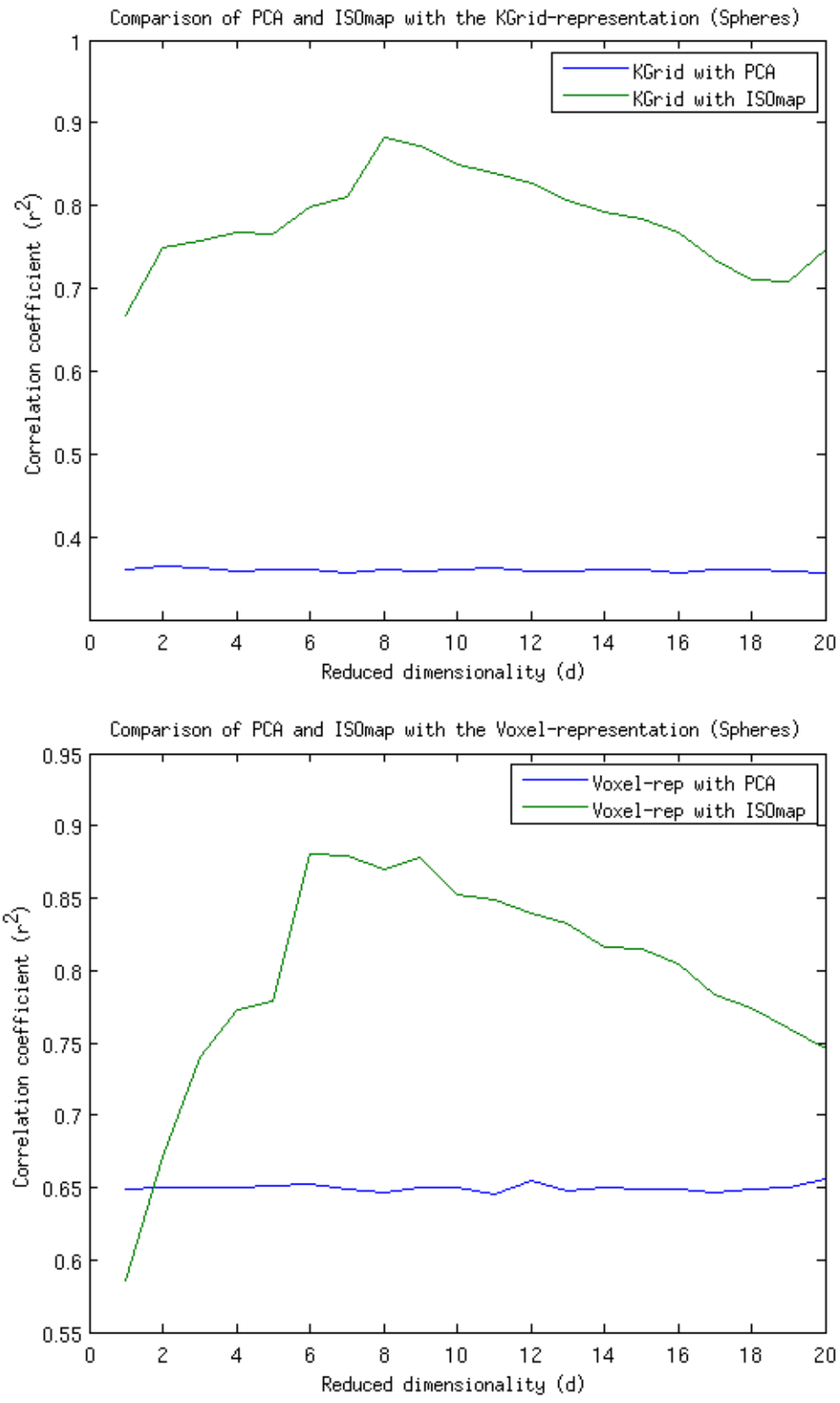


Figure 3.24: Plots comparing ISOmap with PCA in terms of r^2 for the spheres data set.

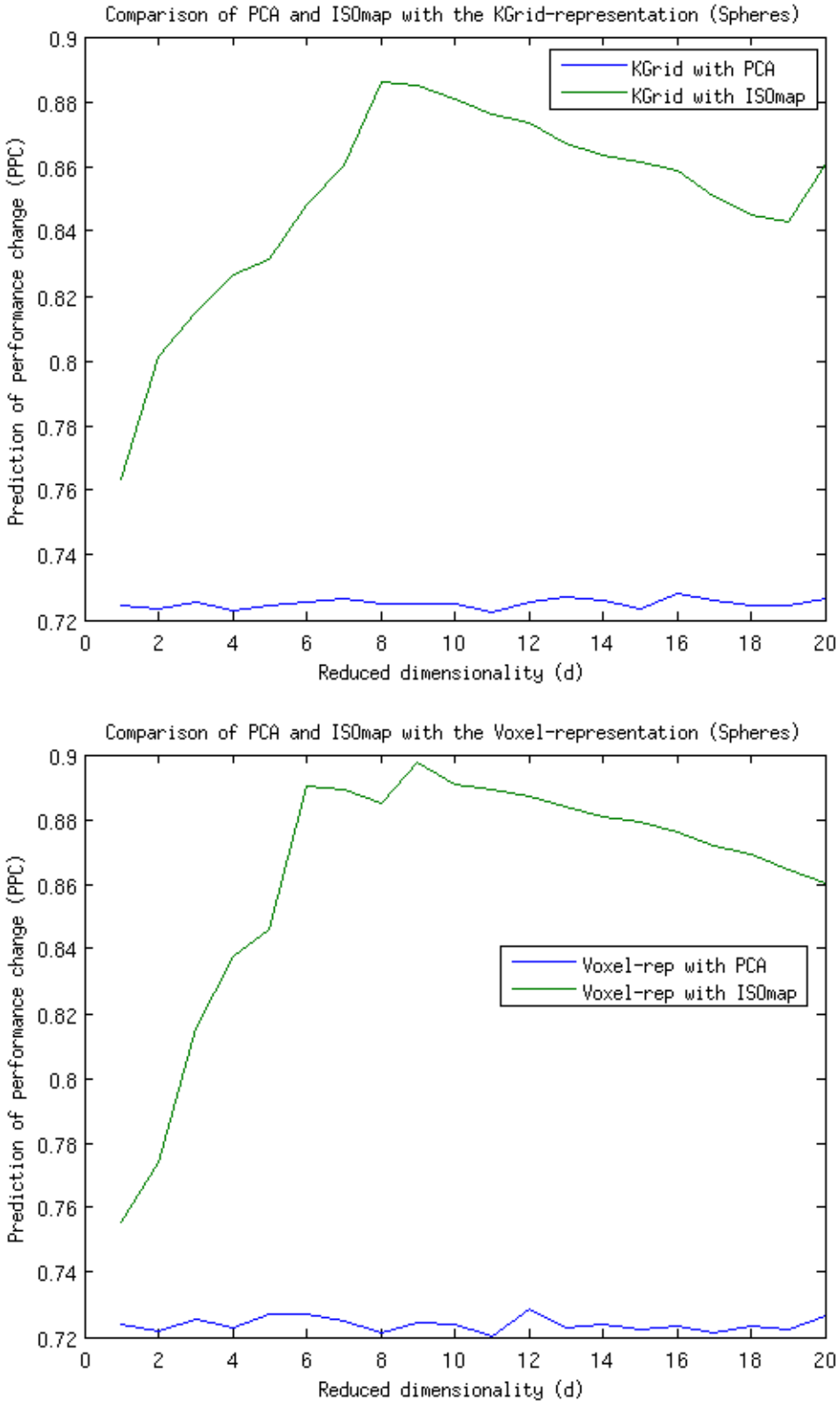


Figure 3.25: Plots comparing ISOmap with PCA in terms of PPC for the spheres data set.

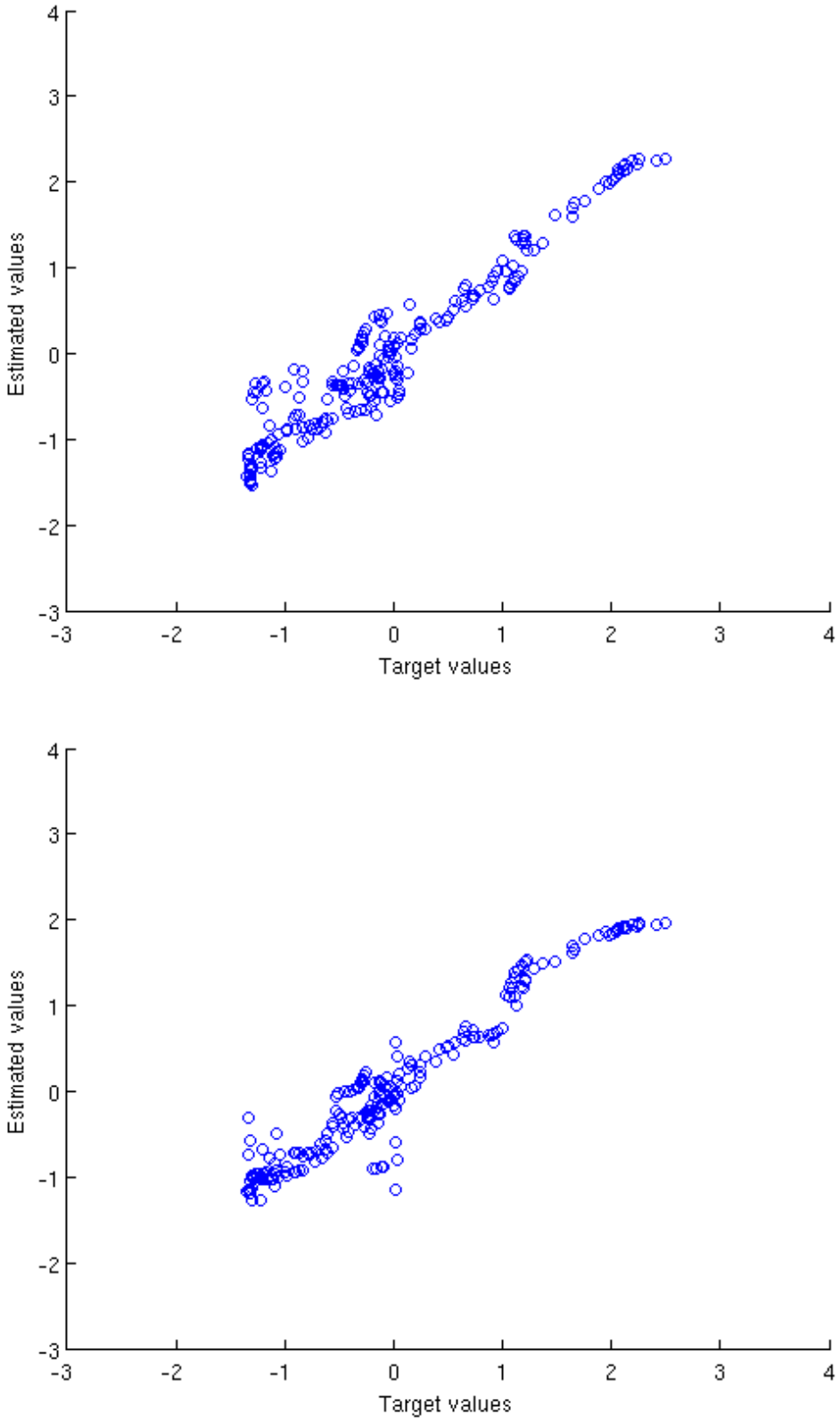


Figure 3.26: Scatter plots of estimated-output against the target-output values for the sphere data set. Top image: KGrid with ISOmap at $d^{'} = 8$. Bottom image: Voxelisation with ISOmap at $d^{'} = 6$.

A single representation from multiple triangle-based surface representations

The results of sections 3.7.2 through 3.7.4 show that KGrid was able to find a representation suitable for machine learning tasks.

Fast computation

Both canonical voxelisation and KGrid are dependent on a potentially costly search procedure, although the former's can be considered far more exhaustive.

Canonical voxelisation traces an axis-aligned ray through the 3D space for each cell. Each ray is then tested against all the triangles in the data set for an intersection. Every point along the ray where it intersects with a triangle (it may intersect multiple times) is marked and later used, through intersecting of the rays, to compute whether a given cell is inside or outside the surface.

This means, for an $m \times n \times o$ voxelisation, we have $m \times o + n \times o + m \times n$ rays, each of which must run an intersection test against N triangles. This can be quite costly as the triangle intersection tests are not very cheap, requiring computation of barycentric coordinates for each one. Finally, all $m \times n \times o$ voxels must be processed, each using the results of three ray tracings in order to determine whether the voxel lies inside or outside of the surface.

In comparison, KGrid, with $m \times n \times o$ cells, must search all $\tilde{N}$ points in the surface (typically $\tilde{N} \ll N$) in order to find the closest point v before doing a triangle intersection test against the $M \ll \tilde{N} \ll N$ triangles which share v. This is the extent of the searching.

It should be clear that speed up will be observed for both approaches if we employ spatial partitioning techniques such that only a subset of triangles or points need be evaluated.

A single machine learner comparable to more computationally intensive approaches

The results of section 3.7, comparing KGrid with the unfeasible ideal representation of the automobile data set, showed that it performed comparably when used with ISOmap and did not perform terribly when used with PCA. This is very encouraging.

The results of section 3.7.3 show that for the automobile data set, KGrid was more suitable than canonical voxelisation for higher dimensional machine learning tasks and did not perform much worse in lower dimensions.

Finally, the results of section 3.7.4 show that for the spheres data set, KGrid was able to compete very effectively with the voxelisation approach (particularly where PCA was used). Where ISOmap was used, it lost moderate accuracy in comparison.

The single representation improves application of ISOmap to engineering problems when compared to voxelisation.

We note that canonical voxelisation, being binary, requires a high resolution voxel grid in order to enable ISOmap to solve the problem. To illustrate this, consider the case where the bump is about the same size as a voxel, and that the vector {1,0,0,0} has the same euclidean distance from {0,1,0,0} as it is from {0,0,0,1}. ISOmap won't be able to correctly compute the distances of these two samples effectively. Clearly KGrid has a benefit in being continuous and this also means that it isn't needed in such a high resolution as the canonical voxelisation.

The results of section 3.7.3 where the automobile data set was used, showed that KGrid was a better choice than voxelisation for higher dimensional machine learning tasks. The results of section 3.7.4, where the spheres data set was used, showed this was true at low dimensions ($d < 3$), otherwise performing very similarly. The reasons behind this apparent contradiction are likely due to the very different natures of the two data sets, since the former experiment represents an engineering problem without the bump problem, and the latter represents a data set with the bump problem but minimal engineering similarities.

The single representation coupled with ISOmap solves the bump problem

The results of section 3.7.4 where the spheres data set is used, show a highly significant improvement where ISOmap is used in place of PCA. A similarly strong improvement is seen regardless of whether KGrid or voxelisation is used. Thus, the single representation coupled with ISOmap appears to solve the bump problem.

The single representation should function across a broad space of possible designs

Some approaches (such as the design-offset approaches) are based on a single surface design used for reference. The effect of this is that representations are more accurate in the design space immediately surrounding this reference design, while they are less accurate further away outside this design.

KGrid (and voxelisation also) does not suffer from this, and could, theoretically, take any design as input.

3.8 Analysis of the ISOmap representation of the automobile data set

One of the aims of KGrid is to create a representation better suited to ISOmap than other approaches. Sections 3.7.2 through 3.7.4 show that KGrid is a good choice when an ISOmap representation is sought. We have also seen that in section 3.7.4 ISOmap is highly capable of dealing with problems exhibiting the bump problem, such as data set D^S. Unfortunately our experimental automobile data set D^R is not a good data set for demonstrating ISOmap's suitability for the bump-problem (as it contains no bump features) and so we also see in sections 3.7.2 and 3.7.3, that for machine learning tasks, PCA is a better option than ISOmap for building an accurate regressor for the automobile data set.

But accuracy aside, there is another reason why we might be interested in using ISOmap: being a manifold technique, ISOmap may generate a representation whose attributes are easier for a human to associate some meaning with. We've seen an example of this working in the literature, specifically in the images shown in section 3.3. These "meaningful features" may then be useful to a human designer and thus the design process.

To test this idea, we can simply plot the automobile data set samples according to their 2D ISOmap and PCA representations respectively and see if we can identify some human interpretable meaning of the axes. For the reader, we embellish the plots with images of randomly chosen designs (two per cluster) and indicate where they are located on the plot. In figure 3.27 we see this plot for ISOmap, and in figure 3.28 for PCA.

Comparing the two, we first notice how ISOmap manages to spread out the data clusters in 2D space far more effectively than PCA does, with the eight ISOmap clusters being easily distinguishable, but with some cluster overlap in the clusters of the PCA plot.

The next observation is that in figure 3.27 the vertical axis of the ISOmap representation (corresponding to the second attribute) seems to correlate with the width of the central region of the design. Then, looking at the horizontal axis (the first attribute), it would seem that this correlates with the length of the design. It would seem that some human interpretable meaning

could indeed be identified.

Looking now to figure 3.28, we notice that there is no such clear feature-to-axis relationship as could be seen in figure 3.27. This is largely because of the way in which the different design clusters are clustered and overlapping in the 2D space when PCA has been used. Indeed, it seems much harder to apply some human interpretable meaning to the axes.

Feature extraction approaches to dimensionality reduction such as PCA involve compressing data in a method similar to lossy compression. In the case of PCA, the covariances and variances of the attributes are used to find a low-dimensional compression of the data. While PCA performs very well for machine learning tasks, it results in a representation which proves difficult for a human designer to associate with clear geometrical changes in the surface designs.

In contrast, ISOmap has been shown to take the data and reduce it to two attributes which can be associated with particular design features (and we have seen this in other ISOmap studies in section 3.3). Despite the artificial construction of our data set, these findings have of real world relevance as we can expect an historically accumulated data set engineering surface designs to also contain clusters of data (clusters may be formed of: design classes (hatchback, saloon etc.); style groups; and designs resulting from independent optimisation processes) throughout which certain global manifold features can be found, such as the two seen in figure 3.27.

We therefore conclude that the first two features generated via ISOmap certainly seem to be more meaningful to human interpretation than those generated via PCA. This could, for example, be useful in enabling the human designer to visualise the extremely high-dimensional design space more easily and to understand how very general design changes affect performance scores. Further attributes in the ISOmap representation not shown here (i.e. three or more) may also convey geometric meaning that a human designer can interpret and learn from.

3.9 Conclusion

3.9.1 Summary

This chapter looked at the task of converting an accumulated data set of diverse engineering surface design descriptions into data ready for machine learning processes. We discussed the particular challenges involved with using the STL format, as well as additional problems that stem from using a very high dimensional data set for learning a highly non-linear regression

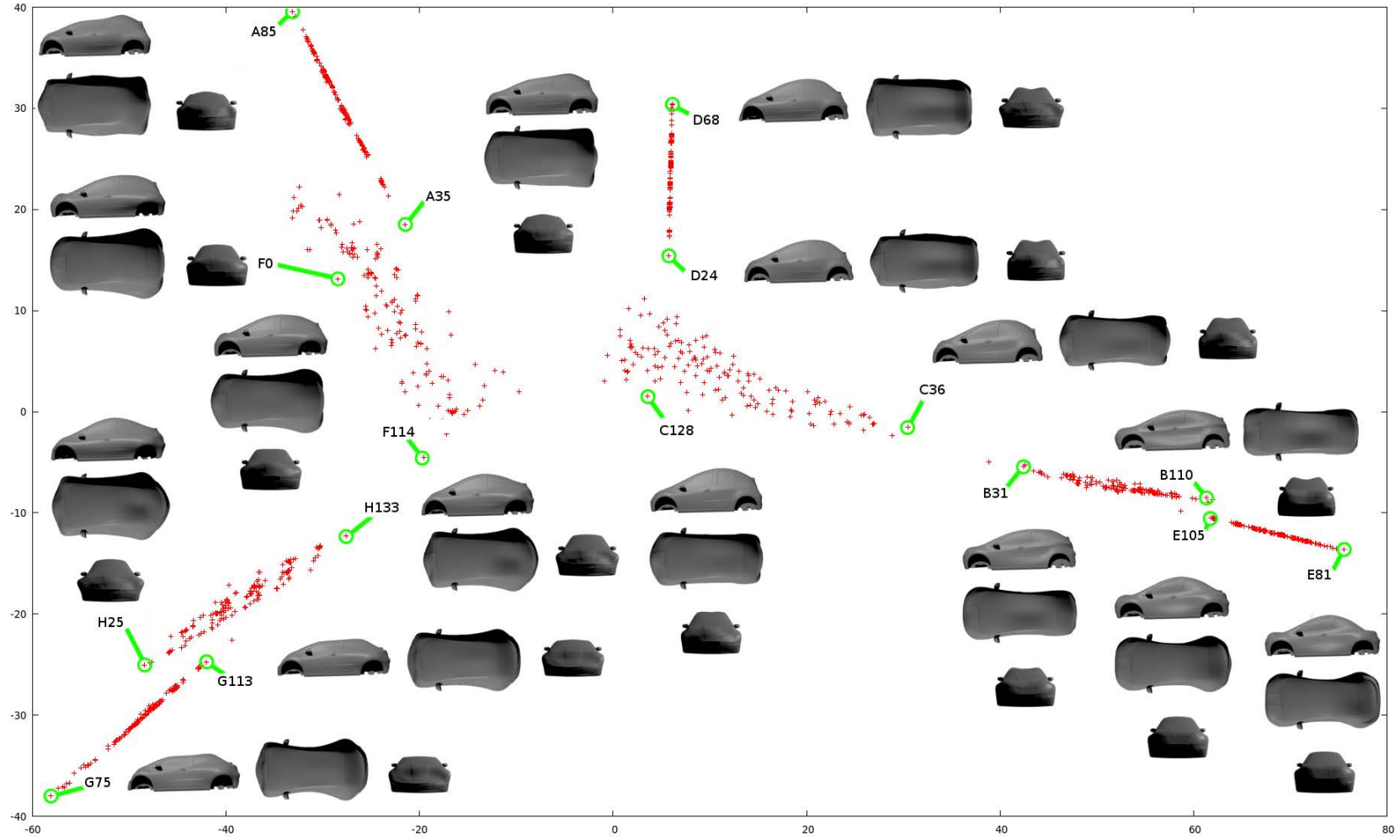


Figure 3.27: A plot of the 2D ISOmapped experimental data. At a glance, the vertical axis seems to correspond roughly with the overall width of the vehicle. The horizontal axis is a little less clear, but could be taken to be related to the overall height of the vehicle. Here $K = 5$ and $\tau = 30$.

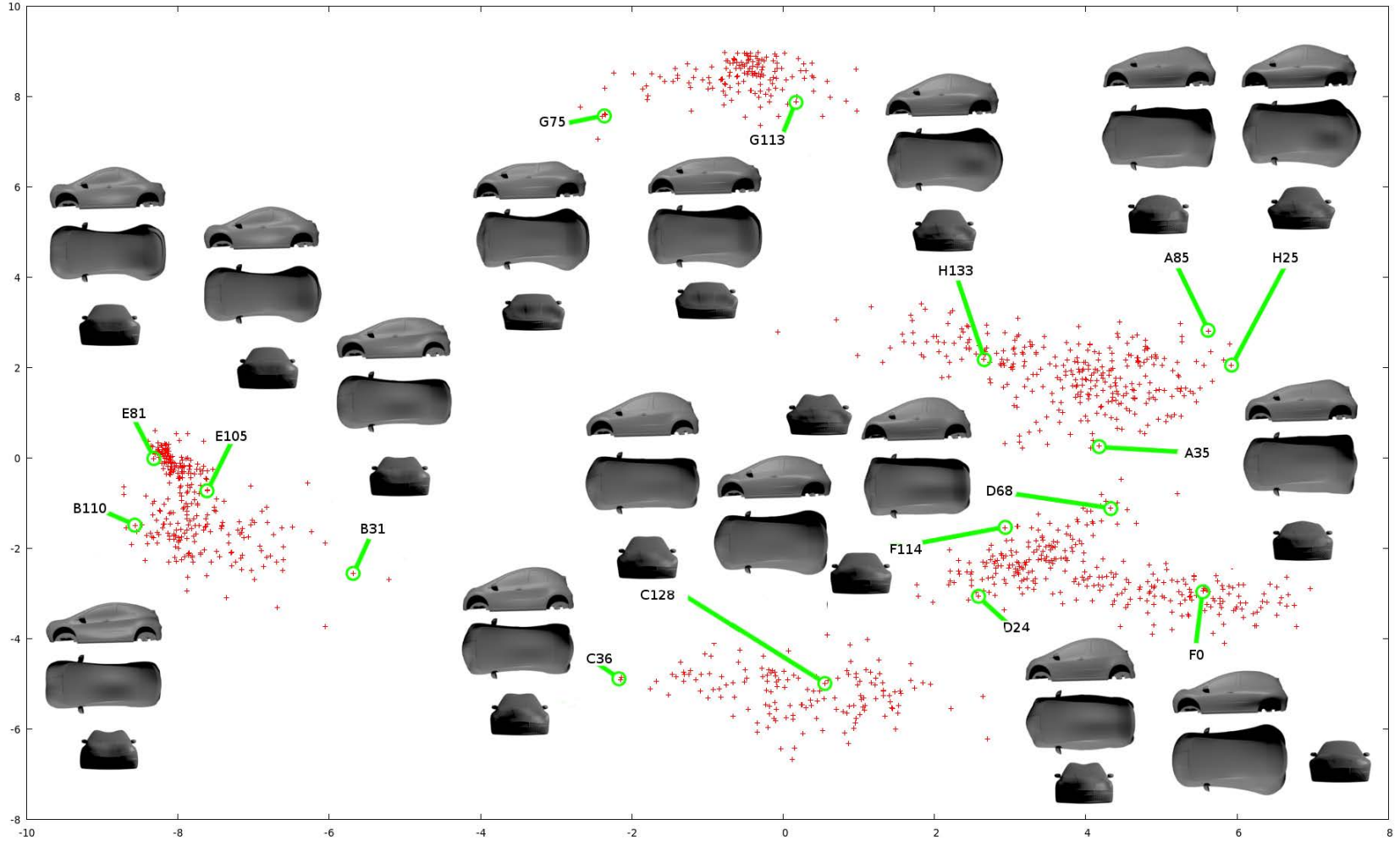


Figure 3.28: A plot of the 2D PCA'd data. The meaning of the vertical and horizontal axes in this data are less clear and some of the clusters overlap considerably, whereas they did not for figure 1. $K = 5$ and $\tau = 30$.

function. The machine learned function should be able to estimate the functional performance of an unseen design as accurately as possible and as quickly as possible.

We further identified that dimensionality reduction will play an important role and that could be chosen to work in synergy with the conversion of the diverse engineering surface designs into a single representation. We also described the 'bump problem' with illustration.

Following this, we discussed some established methods in the literature which have been used for similar purposes to representation conversion and also discussed their shortcomings given our accumulated data set of diverse engineering surface designs. Next we briefly discussed dimensionality reduction techniques and focused especially on a linear approach, PCA, and a non-linear approach, ISOmap.

For upcoming experimental purposes we then described the generation of a realistic experimental data set of automobile designs exhibiting properties which we are interested in, describing in the process several versions of the data set which might be useful to us. Identifying that our automobile data set did not feature the bump problem property, we then described the generation of an artificial data set involving modifications of spheres and exhibiting the bump problem.

Following this we listed some aims of a desired conversion process described in the introduction and suggested a novel approach to achieving these aims: KGrid. KGrid is designed to be a computationally faster approximation of a continuous voxelisation. We evaluated KGrid using our experimental data sets, making comparisons with voxelisation and an unfeasible 'ideal' representation of the automobile data set, and further making comparisons between PCA and ISOmap using the spheres data set. Evaluating the KGrid method and these results we concluded that KGrid was seen to meet our aims. Certainly, KGrid functions in its purpose, performing as well as and faster than voxelisation.

For the automobile data set we also found that basing a high-dimensional ISOmap representation on a KGrid representation results in an embedding that is better suited to machine learning tasks than both voxelisation and an ideal representation. For the spheres data set we saw clear evidence that ISOmap is superior to PCA when dealing with the bump problem.

Finally we analysed the 2D representation produced by each ISOmap and PCA and saw that the ISOmap features separated the automobile data better into its known clusters and also appeared to have a human interpretable meaning on each of the axes. This is encouraging as it indicates that an ISOmap representation could be of more utility to a human designer than a

PCA representation.

3.9.2 Novel contributions

The major contributions of this chapter include: a suggested method of generating an experimental data set with the properties described in chapter 2; a method of generating an artificial data set with those properties and also exhibiting the bump problem; a method of applying ISOmap to clustered data without discarding samples; a description of a problem we call "the bump problem"; a method of approximating the voxelisation of an STL surface design (KGrid) for use in machine learning with particular benefits to ISOmap and reduced computational burden; a study of ISOmap for an engineering problem including an illustration of its benefits over PCA for finding very low-dimensional human interpretable representations; and experimental results showing that a manifold technique such as ISOmap is a superior method to a linear-algebra based method such as PCA for solving the bump problem.

3.9.3 Future work

With regards to KGrid, an interesting work direction may involve looking at optimisation of the KGrid cells (placement and radius) as they need not be uniformly sized or distributed. It can easily be imagined, that some regions of the surface design (and thus the 3D space encompassing designs) benefit more from very fine KGrid cells than other regions.

Since the ISOmap experimental results using the spheres data set are almost the inverse of those using the automobile data set, it would be interesting why this exactly this. This could require an in-depth parameter study to uncover the relationship. Similarly, there will probably of KGrid/Voxelisation resolution where KGrid outperforms Voxelisation for the reasons given in the fourth sub-subsection of subsection 3.7.5.

Another area of research would be to look at efficiency impacts in terms of spatial partitioning. For example, by partitioning the 3D space such that all the surface points are sorted into partitions, we can substantially speed up the the nearest point search of KGrid. Although it can be argued that the same approaches can be applied to voxelisation through the sorting of the triangles into partitions, we note that it is a far less trivial task than for KGrid. Firstly, there are usually substantially more triangles than points, so for the performance boost we would need more partitions. Secondly, surface design triangles vary in size considerably and so triangles

would often end up in multiple partitions, unless partitions were made large enough to encompass any triangle, and then once again, we would end up with partitions containing many triangles for testing.

With regards to ISOmap for engineering purposes, we can consider supervision of the ISOmap process. ISOmap has been often used in literature to create low-dimensional embeddings where the features capture some human-perceivable change in the data. It is less commonly used for dimensionality reduction in engineering problems like this (i.e. where there is some performance value to be estimated by a regression function). It is possible that inclusion of the performance value in the distance calculation, which is used to generate the nearest-neighbourhood graph, may be of benefit to finding a low-dimensional embedding of the data which better captures design changes with impact on performance.

Chapter 4

Functional Estimation across a Large Unfocused Input Space

This chapter looks at the task of finding a suitable regression function to act as our approximation. We will first introduce the problem and highlight the challenges that make our problem unique. Following this we will take a high-level look at regression approaches in the literature before focusing on ensemble regression techniques in more detail. Having evaluated the state of the art, we suggest a novel ensemble framework suited to our problem before evaluating it against competing ensemble frameworks using our test set. We pay particular attention to generating a regression function that offers the human designer some utility.

The aim of chapter 3 was to pre-process an historically accumulated data set into a data set with a representation that best captures the data within, i.e. a representation which accurately and adequately encodes all the design variations required to build the most accurate approximation function possible. In this chapter we are concerned with the separate process of generating a regression function, which we would like to develop in a representation-agnostic manner. In this chapter we assume that the optimal representation has been found using an arbitrary approach and therefore only use and refer to the unfeasible 'ideal' automobile data set D^C. Thus the findings in this chapter can apply to another problem regardless of the data pre-processing performed.

4.1 Problem introduction

In section 2.4.2 we listed some of the general issues that make applying machine learning to this task challenging. Through use of the historically accumulated data set of previously interesting designs, we managed to alleviate some of the initial challenges, but some challenges remain that make applying machine learning to our task challenging. Two broad groups of challenges emerge and we will now look at these challenges.

4.1.1 Large design representations and available training data

As described in section 2.4.2, machine learning algorithms suffer highly from the curse of dimensionality. To repeat: the curse states that as the dimensionality of an input domain increases, the number of training data samples required by a machine learning algorithm increases exponentially.

In section 2.4.3, having discussed our source of training data (and its format) we also described how a truly compact representation will be highly challenging to find, as any representation chosen needs to be complete enough to represent any changes that the designer makes which affect the functional performance. As mentioned previously, finding a complete yet compact representation for our data set is highly challenging. This is specifically because the set of attributes likely to be modified by the human designer, and the set of attributes which have effect on the functional performance, will likely change throughout the space of possible designs.

The problem we then have at the point of machine learning a regression function, is that we have a high-dimensional data set, but with still relatively few training samples. For example, consider that D^C, the 'ideal' data set, has 1524 attributes, but we only have 1200 samples. Clearly, some dimensionality reduction technique such as PCA, discussed in the previous chapter, is needed, but depending on the strength of the dimensionality reduction applied (i.e. how great $d - d^{'}$ is) we run the risk of losing valuable information. This trade-off is difficult to manage and in this chapter, we will suggest a novel way of dealing with this.

4.1.2 Learning a large domain without a local design space focus

As described in sections 2.4.2 and particularly section 2.4.3, machine learning approaches are more robust when applied to smaller sub-domains of a larger non-linear problem domain. This

is simply because it is easier for a machine learning algorithm to learn the necessary rules in a smaller region of a non-linear function. Widening this 'window' can in effect lead to confusion of the machine learning algorithm as it struggles to learn the relationship between input design and functional performance.

Despite knowing this, we must be aware that a regression function, when functioning as our approximation, needs to be prepared to estimate the functional performance for any design that the human designer needs evaluated. For the motivating example in chapter 2, automobile design, this can be a very large domain from which the user might need to make an estimate anywhere.

Fortunately, for engineering surface design tasks such as automobile design, the human designer will not request functional performance estimates from the whole domain with uniform distribution. As identified in section 2.4.3, the use of an historically accumulated data set of previously interesting designs gives us some information about the nature of human design and thus the sub-domains from which we can expect the user to request estimates. The challenge of learning a regression function for a large domain therefore simplifies somewhat. Unfortunately, the nature of functional performance functions and the lack of contiguity between sub-domains that the user is sampling means that the learning of single regression function may be made more difficult, as important sub-domains may appear to have completely different input to output relationships.

4.1.3 Utility to the human designer

Finally, we must always consider what utility we can offer the designer when choosing a system. We saw in section 2.2 that one use for surrogate functions in engineering optimisation is analysis of the function being modelled. For example, an engineer can use such a function to study surface design attribute interactions and their effect on functional performance. We should ensure that a competent user of our system is able to get such information out of our regression function when required.

4.2 Regression functions in machine learning

In this section we will take a general look at approaches to generating regression functions, followed by a more detailed look at ensemble regression techniques, as these exhibit good potential as solutions to our challenging task.

4.2.1 Overview of regression approaches

In chapter 2 we saw that regression functions take the form $\hat{f}(\mathrm{x}) = y + \varepsilon$, where y is the output of the original function being modelled $f(\mathrm{x}) = y$, and ε is some error term. In this section we will talk more generally about how one goes about forming a regression function.

Generation of a regression function must typically be performed using some regression analysis method (such as those methods in machine learning). The key task of any regression analysis method is to find a relationship between x (the independent variable(s)) and y (the dependent variable). The relationship found can be used to find $E(y|\mathrm{x})$.

Regression analysis approaches and their resultant regression functions can be broadly categorised as being either parametric or non-parametric. Non-parametric approaches tend to use functions of the N samples in the data set, $D = \{X, Y\}$. Such methods are often kernel based such as kernel regression [96, 139]: $\hat{f}(\mathrm{x}) = \frac{\sum_{i=1}^{N} K_h(\mathrm{x}-X_i)Y_i}{\sum_{i=1}^{N} K_h(\mathrm{x}-X_i)}$, where K_h is a kernel function with bandwidth h taking as input $\mathrm{x} - X_i$ (for example, euclidean distance could function as a kernel function).

Non-parametric methods like this can be considered to be interpolators of the data. In this (and similar approaches) the parameters of the kernel function (in this case just h) are all that need estimating for the data set, the kernel functions themselves combine to generate $\hat{f}(\mathrm{x})$.

Gaussian Process regression, or Kriging, is a popular kernel-based method incorporating Gaussian Processes and using prior covariances to interpolate rather than a polynomial. This approach is heavily supported for use in engineering by the authors of [34] and has been shown to work well in a surrogate-assisted framework in [55], where it is used for the optimisation of an engine turbine disk.

Another prominent non-parametric method is the Classification and Regression Tree (CaRT) [18]. This method is based on a decision tree structure where each interior node in the tree represents a decision and a chosen branch is traversed depending on the outcome of that decision

(for example, if attribute $x_i > 0.5$ branch 1 is traversed, otherwise branch 2). The traversal finishes on output nodes which instead of representing a decision, give a final estimated output value.

Clearly, non-parametric methods, often interpolating, are not learning the relationship between input attributes and the output value being estimated. Rather they are just basing their estimate on the 'similarity' to previously observed samples in the most direct sense. Any parameters in non-parametric methods (such as h in kernel regression) can be tuned to the problem but are not strictly fit to the problem. Non-parametric approaches can thus be robust and effective, and their deterministic nature can make them better than some of the parametric approaches we will see shortly.

Parametric methods typically involve finding (or learning) a set of parameters which can be used with the input x in order to calculate the predicted output $\hat{f}(\mathrm{x}) = \hat{y}$. Linear regression is such a method [29]. For the general case, linear regression for a d-dimensional input involves a function like $\hat{y} = \beta_0 + x_1\beta_1 + x_2\beta_2 + ... + x_d\beta_d$, where $\overrightarrow{\beta} = \beta_0, \beta_1, \beta_2 ... \beta_d$ is the vector of parameters learnt for the particular problem.

Learning, under parametric regression approaches, requires a set of training data from which to learn the relationship between x and y. For example, in the case of general linear regression, if we take $X \in \mathbb{R}^{N \times d+1}$ to be our sample data set of inputs (with the first entry corresponding to x_0 set to 1) and $Y \in \mathbb{R}^{N \times 1}$ to be the respective outputs, then $\overrightarrow{\beta} = (X^\top X)^{-1} X^\top Y$ (clearly, the constraint $N \geq d$ must hold for this to be effective). Least squares analysis is best applied for fitting $\overrightarrow{\beta}$ to difficult data.

Polynomial regression [40] adds more flexibility to linear regression through the extension of linear regression to contain polynomials, such that $\hat{y} = \beta_0 + x_1\beta_1 + x_2^2\beta_2 + ... + x_d^d\beta_d$. Fitting $\overrightarrow{\beta}$ can be done again via least squares.

A popular parametric model used in regression is the Artificial Neural Network (ANN). There has been copious research surrounding ANNs in the latter half of the 20th century and since the turn of the 21st century research has begun to wane, but mostly as ANNs have been so thoroughly researched. The Multi-Layer-Perceptron (MLP) [47, 51, 75] is the most popular form of artificial neural network with an input layer of 'neurons', a hidden layer of neurons each activating according to a logistic function (e.g. sigmoid), and then an output layer with, in the case of regression, a single node outputting a value via a linear combination function. An

ANN such as MLP is typically trained using the backpropagation method, where the error of the ANN for a given input is propagated backwards through the layers, and their weights adjusted accordingly.

Extreme Learning Machine (ELM) [54] has a very similar architecture to an MLP, but instead of training the hidden-layer weights, chooses these at random. The weights for the output layer are all that is trained and this is performed using linear algebra, incorporating the outputs of the randomly-generated hidden layer and the target outputs of the data set.

Support Vector Machine (SVM), an approach used for classification, was modified in [129], producing the Epsilon-Support Vector Machine (E-SVM) algorithm which is designed for regression tasks.

As a broad interpretation of the difference between non-parametric and parametric approaches, it can be then understood that non-parametric approaches to regression perform well locally to previously seen samples. On the other hand, parametric approaches focus on learning the relationship between inputs and outputs such that the output for samples substantially different from previously seen training samples can be better estimated.

Clearly, choice of regression or machine learning function is not a trivial issue. Different approaches work best for different problems and some approaches can work very poorly for other problems. In this review we mentioned only the most common approaches which appear repeatedly in the literature, but note, for a given problem it is entirely reasonable to expect a specific and optimal solution to exist, often being found only through testing and evaluation.

A number of problems occur specifically with parametric regression approaches. The first problem to mention is over-fitting [136]. Over-fitting describes the case where the parameters of the regression function are found such that the regression function performs excellently on the training samples, but will perform poorly for future unseen samples, sometimes giving wildly incorrect values. This can be caused by an overly complex model (such as one with more parameters than samples are available or use of polynomial regression for a linear problem) or by the training algorithm (such as gradient decent methods minimising the error on the training set so strongly that the regression function ceases to generalise well).

Popular techniques for dealing with over-fitting are: regularisation [5], which typically involves applying a penalty to the number of parameters such that simpler regression functions are preferred; early stopping of the training algorithm when an iterative approach such as gradient

descent is used; and validation/cross-validation [71, 28]. Validation approaches generally involve reserving a subset of the training data (called the validation set) to test for the generalisation of the regression function. Then, if competing parameter sets are available, the one that performs best on the validation set should be chosen. For example, during an iterative training process, the performance of the regression function at each iteration could be checked and the best parameter set recorded. Cross-validation extends the validation technique to involve multiple training sets and validation sets in the search for the best parameters.

Another reason why a validation approach may be needed to compare and evaluate competing parameter sets would be due to reliance on stochastic values. For example, the initial hidden layer weights in an MLP are usually initialised to random values, which means that the parameters learnt through gradient descent are highly dependent on this initialisation.

4.2.2 Ensemble regression approaches

Overview of ensemble regression approaches

For an excellent introduction to ensembles please see [19]. For a recent survey including the most recent developments refer to [91].

As the authors of [91] note, much of the ensemble modelling in the literature focuses on the classification task, with less emphasis on the task of regression, despite its high significance. They broadly define ensemble learning as "a process that uses a set of models, each of them obtained by applying a learning process to a given problem. This set of models (ensemble) is integrated in some way to obtain the final prediction". Following this, they suggest that the ensemble construction process consists of (i) ensemble generation (ii) ensemble pruning and (iii) ensemble integration.

In the first step the set of regression functions is generated. The second step, pruning, then seeks to remove regression functions which offer little to the process, often by assessing diversity within the ensemble, and removing regression functions whose error correlates highly with others. The final step, integration, is the description of how one takes this set of regression functions and arrives at a single predicted value for the ensemble.

The most basic ensemble technique could therefore be seen as the *simple averaging of predictors*. Here, the ensemble generation consists of generating K regression functions independently

of one another. Ensemble pruning can then be optionally applied in order to remove regression functions whose error correlates highly with another regression function. This leaves us with a revised number of regression functions, $\dot{K} \leq K$. Finally the ensemble output is calculated via a general integration function:

$$\hat{f}(\mathrm{x}) = \sum_{i=1}^{\dot{K}} \alpha_i \hat{f}_i(\mathrm{x})$$

where $\hat{f}(\mathrm{x})$ is the predicted output of the ensemble for input x; $\hat{f}_i(\mathrm{x})$ is the predicted output of the ith regression model for input x; and α_i is the weight associated with the ith model. In this case of simple averaging $\alpha_i = \alpha = \frac{1}{\dot{K}}$. While integration step weights based on simple averaging are most common, some ensemble generation approaches lead to different weight generation (as with Boosting [30]). Generally $\sum_{i=1}^{\dot{K}} \alpha_i = 1$ and if the weights aren't chosen uniformly then they can be chosen to reflect the confidence in each regression function. This can be done just once during ensemble generation based on the training data as in [30, 108] (in [1] weight generation is tackled as an optimisation problem), or dynamically ("on the fly") for each unseen input by assessing each regression function's accuracy on similar samples in the training data [141]. The authors of [105] demonstrate that non-uniform and optimised weightings result in the surrogates with better accuracy and identify that different surrogates perform better in different design space regions.

Diversity is prized within an ensemble, such that the main differences between ensemble generation approaches tend to simply involve different methods of achieving diversity. The authors of [91] divide ensemble generation approaches into *data manipulation* and *regression algorithm manipulation*.

Data manipulation approaches

This group of approaches is subdivided into (i) sub-sampling from training set; (ii) manipulating input features; and (iii) manipulating output features.

Sub-sampling is the method which has attracted the most research attention. Its premise is that, assuming the regression algorithm is unstable (highly sensitive to training data as with decision trees and neural networks), then regression functions trained on different subsets of the training set will exhibit diversity.

One of the most popular of these methods is bootstrap aggregating, or "bagging" [15], which is a very similar method to the simple averaging ensemble described above. The difference is that for each model a training set of $\dot{N} \leq N$ samples is uniformly and randomly selected, with replacement, from the training set D. If N is "large" and $\dot{N} = N$ then we can expect $\approx 63.2\%$ of the $\dot{N}$ samples to be unique, the rest being duplicates [6].

Another approach to generating multiple training sets can be found in [106]. Here the authors suggest dividing the training set in a v-fold cross-validation inspired way, creating a neural network regression function for each of the v folds, such that $K = v$, using the rest of the samples for model-selection purposes according to the traditional validation method.

A very important approach developed for classification problems is AdaBoost [121], a member of a class of algorithms described as "boosting". Like bagging, boosting relies on a random selection of a subset of training samples for each model. The big difference is that samples are not chosen uniformly, but rather weighted according to how well previous models in the ensemble were able to predict for them. In AdaBoost this is realised by giving uniform probability to all samples at the initial classification function generation stage. Then the classification error for each sample is used to modify the probability of this sample being chosen in the next iteration, such that if it was correctly classified, the probability of it being chosen would drop slightly, whereas if it was incorrectly classified, the probability would increase.

This method has been then modified for regression tasks [38, 7]. The former study involves converting the regression data set into a classification data set, while the latter involves using an error threshold to determine when a sample has been "incorrectly classified" in the AdaBoost algorithm. A well performing application of AdaBoost to regression has been seen in [30], where errors are normalised to the [0...1] range and a more rigorous and mathematical approach is applied to the task of adapting weights.

The approach of manipulating input features broadly involves replacing the original data set with another where the input features have been changed in some way. In [53] the authors generate an ensemble of decision trees where each decision tree sees only a randomly selected subset of the inputs (a "random subspace"). The authors of [122] replace this filtering of features with generation of new features based on random combinations of features, noting a slight improvement.

An alternative, but computationally costly approach, is effectively to optimise (through gra-

dient search or evolutionary approaches) the chosen subsets in order to maximise model accuracy and diversity, as in [103, 143].

In [37] the authors suggest the addition of Gaussian noise to the inputs in order to manipulate input features. Their results are shown to be comparable to bagging.

Another interesting approach is Rotation Forest [115]. For the generation of each regression function, this approach first partitions the input features into κ disjoint subsets, then Principal Component Analysis is applied to each subset in order to generate a new set of features. This method has been studied with regards to regression in [144] where its performance was better than random forest ensembles, and comparable to bagging.

There is little work on manipulating output features to date, with [16] being the most notable example. Here the author simply adds Gaussian noise to the output values in a similar way to that seen with input features in [53].

Regression algorithm manipulation approaches

The second group of ensemble generation approaches involves manipulating the regression function learning algorithm itself. In [91] the authors identify three groups of methods, amongst which are: (i) manipulating parameter sets; and (ii) manipulating the learning algorithm.

In the simple-averaging ensemble we originally gave, we didn't immediately explain why this works. In truth, it will only work with regression function learning algorithms which involve a stochastic element. For example, if our simple-averaging ensemble consisted of neural networks we could expect each one to be generated with a set of randomly chosen, and thus different, initial weights, similar to the approach in [117]. This is an example of manipulating parameter sets and has been extended to the random selection of hidden layer nodes and numbers of hidden layers [108, 50].

Manipulation of the learning algorithm is by its nature learning algorithm specific and therefore there are no general approaches to this. Manipulation of the learning algorithm can be performed either sequentially (only previous regression functions have an impact on the current function's learning) or in parallel (each regression function takes into account the error of the overall ensemble and the error of each of the other regression functions).

These approaches are often performed with neural network ensembles, where the error value used for learning the neural network is modified to include some penalty for the correlation of

the neural network with others in the ensemble. This is performed in a sequential ensemble generation framework in [117] and indeed our simple averaging ensemble example would be seen as sequential.

This approach is extended in [58] where models are added to the ensemble sequentially, and then trained in parallel at each iteration. This work can therefore be perceived as a mixture of sequential and parallel approaches. In this work, further diversity was achieved by adding hidden nodes to each network.

Another interesting approach is ADDEMUP [104]. In ADDEMUP the error function involved in training each neural network takes into account both the accuracy of that neural network and the contribution this neural network makes to the diversity of the ensemble according to the well established bias/variance decomposition [72]. When adding to the ensemble, new initial neural networks are generated through the genetic operators of mutation and crossover, and then trained using a weighted sample scheme similar to AdaBoost.

A truly parallel approach is Negatively Correlated Learning [81]. It differs largely from previous works through the use of the negative correlation term rather than the bias/variance decomposition. An extension to this work, [82], uses genetic operator of mutation to optimise the neural network weights.

In [39] the authors also use this mutation operator to evolve an ensemble of neural networks alongside another operator which can modify neural network structure. In particular their approach invokes diversity through the use of independently-evolving sub-populations and a multi-objective fitness function combining both ensemble accuracy and individual neural network fitness. This approach has only been applied to classification and it is not clear how well it may suit regression problems.

Random Forests [17] is an ensemble of decision trees using a randomly selected feature subset at each node. This approach is combined with bagging [15] in the final algorithm and has been shown to generalise well for a range of problems.

Ensemble generation techniques can be problem dependent and this makes it impossible to draw general conclusions towards "the best technique", hence our resistance to claim shows of superiority in any of these studies. Also, many ensemble generation methods are dependent on certain parameters and this makes it difficult to compare experimental results from different pieces of research. The authors of [91] show this in particular by looking at two pieces of

conflicting research.

4.2.3 Improving regression performance for D^C

We have seen an overview of regression approaches and looked in detail at ensemble regression techniques. Ensemble regression techniques typically improve upon the generalisation of stand-alone regression methods, while still suffering from the same problems as stand-alone regression methods (although arguably mitigating the negative effects of these problems). Similar to stand-alone regression approaches, ensemble regression is best chosen in a problem specific manner. For ensembles, this implies that the choice of base regressors and ensemble architecture requires care.

Driven by our particular problem, we note that D^C is a non-uniform and clusterable data set taken from a large space of designs with different low-dimensional representation mapping functions in each cluster. Each cluster differs in terms of:

- The attributes most likely to be varied in that sub-space of the design space.
- The effects and interactions that attributes have on the output in that sub-space of the design space.

None of the ensemble techniques mentioned previously are specifically designed to deal with such a data set, and we suggest that observing and taking advantage of these specific properties in our problem could be key to improving accuracy of an ensemble regressor.

We also note that all of these ensemble approaches are very much "black-box" approaches: the algorithm performs some process and the resulting ensemble is output with no comprehensible meaning. Maintaining some semantic value in the ensemble's regression functions may improve utility for the engineer/artist using the system offering:

- Attribute/feature analysis for particular classes of design.
- The opportunity to study subsets of the ensemble's regressor's functionality.
- Greater confidence in the ensemble's inner workings and therefore its estimations.
- The ability to sense weakness in areas of the ensemble regressor and reinforce these areas using knowledge from the ensemble.

4.3 A novel ensemble framework

In this section we will propose a novel ensemble framework, listing its aims, additional technologies that are involved and finally its methodology.

4.3.1 Aims of the framework

In development of the novel ensemble framework we have looked at the properties of the data set mentioned thus far. These lead us to aim to create an ensemble which:

1. Has good generalisation across a large space of possible designs.
2. Incorporates multiple low-dimensional representations of the data (through multiple dimensionality reduction mapping functions).
3. Focuses constituent regression functions on individual subspaces of the design space based on the geometric similarity of designs in the data set.
4. Can interpolate smoothly between the subspaces.
5. Is internally modularised in a meaningful way to enable designer utility and promote trust.

4.3.2 Clustering techniques

Our methodology relies on a probabilistic clustering algorithm in order to determine subspaces of the design space in which each of the ensemble's constituent regression functions should focus. For this reason it is important to briefly give the reader some background of clustering algorithms.

Clustering is another prominent field of Machine Learning. Clustering is tasked with taking a data set and finding within it 'clusters' of data. This is typically done ignoring the output term, and hence termed "unsupervised".

For our framework we are seeking a Machine Learning technique for finding, unsupervised, the clusters in our training data set. The algorithm chosen must be able to assign an input with probabilities of belonging to each data set.

The most well-known and commonly employed clustering approach is K-means clustering [84]. When using common distance metrics such as euclidean distance, this can be described as a spherical clustering algorithm. This means that the cluster's shapes can only be spherical (i.e.

the cluster covers the same distance in every possibly direction): ellipsoids and arbitrary shapes are ruled out. The K-means algorithm also requires the user to set the number of clusters K before execution.

While this is the most common clustering algorithm, and can be useful in a number of applications, several other clustering themes do exist.

Fuzzy clustering [3] embellishes typical clustering algorithms by allowing fuzzy membership of samples to clusters. Hierarchical clustering [132] is an approach where the clusters fall into a treelike structure, where multiple clusters can become members of other clusters, until all the data can be said to be within one cluster.

Probabilistic clustering [89, 131] is similar to fuzzy clustering, in that samples can be assigned to multiple clusters to varying degrees. But instead of fuzzy memberships, probabilistic clustering is a statistical method whereby samples are assigned to clusters according to a probability, where all probabilities sum to 1.

Gaussian Mixture Model (GMM), is a popular probabilistic clustering process where the clusters are modelled as a set of multivariate Gaussian distributions. This allows the clusters to take on a non spherical shape, and the whole data set is seen conceptually as a mixture of these Gaussian distributions.

One can fit GMM to a set of n sample points, $D = \{\vec{\mathrm{x}}_i\}_{i=1}^{n}$, where each $\vec{\mathrm{x}}_i$ is a vector of d values. We consider each $\tilde{\mathrm{x}}_i$ as an observation of the random variable $\tilde{\mathrm{x}}$. The aim of the GMM is to estimate the probability distribution function of D and we assume that this distribution is composed of an mixture of $k = 1, ..., K$ Gaussian distributions, each with mean μ^k and variance-covariance matrix Σ^k. That is to say, the probability distribution of $\tilde{\mathrm{x}}$, $p(\overrightarrow{\mathrm{x}})$ is assumed to be of the following form:

$$p(\tilde{\mathrm{x}}) = \sum_{k=1}^{K} \alpha_k . p_k(\tilde{\mathrm{x}})$$

where

$$p_k(\tilde{\mathrm{x}}) = N(\mu^k, \Sigma^k)$$

and $N(\mu^k, \Sigma^k)$ is the Gaussian probability distribution function with the parameters μ^k and Σ^k. The α_k are mixing weights and these sum to 1, i.e. $\sum_{k=1}^{K} \alpha_k = 1$. Often, the kth mixing weight

Algorithm 4.1 Generation of the ensemble of expert regressors.

1. Given a training set D of N_D samples, each consisting of some inputs $\vec{\mathrm{x}}_i$ and output y_i, and a candidacy-threshold γ, create a probabilistic clustering C on the inputs $\vec{\mathrm{x}}_1, ..., \vec{\mathrm{x}}_i, ..., \vec{\mathrm{x}}_{N_D}$ to find K clusters and associate $\vec{\mathrm{x}}_i$ with a probability, ρ_i^k, of belonging to each cluster, $1, ..., k, ..., K$.
2. For each cluster, $j = 1...K$:
   (a) Apply a dimensionality reduction approach G_j on all those $\vec{\mathrm{x}}_i$ where $\rho_i^j \geq \gamma$.
   (b) Train a suitable regressor, R_j, on all inputs in D after G_j has been applied, weighting their contribution by their ρ_i^j.

is set to the *prior* probability for the kth Gaussian to have generated $\tilde{\mathrm{x}}$, i.e. $\alpha_k = p(k|D) = \sum_{i=1}^{n} p(k|\vec{\mathrm{x}}_i)$.

The training of GMM therefore requires that suitable μ^k, Σ^k and α_k of the K Gaussian distributions are found for the sample data.

The most popular approach to fitting these parameters is Expectation-Maximisation (EM). Under the EM process we seek to find the μ^k, Σ^k and α_k that maximise the likelihood of the GMM's parameters being correct. That is, where $\Theta = \{\mu^1, ..., \mu^k, \Sigma^1, ..., \Sigma^k, \alpha_1, ..., \alpha_k\}$, we wish to $\max_\Theta L(\Theta|D)$, which is equivalent to $\max_\Theta p(D|\Theta)$.

In our experimental studies of the framework to follow we employ GMM as our clustering mechanism, with justifications to be given.

4.3.3 The method

Here we describe our novel ensemble framework. The framework depends on three core component technologies:

1. A probabilistic clustering method.
2. A dimensionality reduction method.
3. A regression function which can be trained via a weighted training algorithm.

The conceptual framework of these is indicated in figure 4.1. The choice of technology for each component will be problem dependent although later in this chapter we offer some suggestions.

We generate our ensemble of experts as shown in algorithm listing 4.1.

Algorithm 4.2 Estimation for an unseen sample $\vec{\mathrm{x}}$.

1. Cluster $\vec{\mathrm{x}}$ using C acquiring ρ^k for $k = 1...K$.
2. Estimate the output for $\vec{\mathrm{x}}$ as $\hat{y} = \sum_{j=1}^{K} \rho^j R_j(G_j(\vec{\mathrm{x}}))$.

This process will generate our expert regressors independent of each other, with the aim of an ensemble of regression functions with de-correlated predictions and specific dimensionality-reduction functions. Step 1 simply performs the splitting of our data based on each samples design information. Step 2(a) will generate an embedding of reduced dimensionality for this cluster. Step 2(b) creates each expert regressor function using all the data but weighting each samples contribution by the probability that it belongs to this experts cluster.

The estimation for an unseen sample $\vec{\mathrm{x}}$ is performed as shown in algorithm listing 4.2.

Summarised, the ensemble contains one probabilistic clustering algorithm operating on the whole of $D \subseteq S$. This is done specifically with the aim of exploiting the known clusterable nature of the data set. Using the clustering algorithm, the data can be weighted as belonging to each of the clusters to some degree of probability.

For each cluster, a pairing of one dimensionality reduction function and one regression function, will be generated. These are generated such that they are specific to only the samples which belong to that cluster with high probability.

Firstly, a simple threshold (e.g. $\gamma = 0.9$) restricts the samples which are used for each cluster's dimensionality reduction function, G_j. This is a novel part of the framework because this is precisely where we attempt to exploit the knowledge that G_j may differ for different values of j, and that the resulting low-dimensional representation will be more useful than if a representation based on the whole data set were used.

Secondly, the continuous probability values themselves can be used with the ensemble to perform weighted sample training. This can vary between different algorithms but the basic notion is that through weighted sample training, the regression function R_j focuses more on samples that belong to it, and less on other samples. This results in smoothing between clusters as they all see the whole data set but focus particularly in the subspace encompassing their cluster.

The framework described bears striking similarity to the "mixtures of experts" approach [19, 142, 87], where individual neural networks specialise in a subset of the data. There are important differences between the mixtures of experts approach and the framework described here though. Firstly, mixtures of experts canonically use a gating function (usually a classification network) to select a single expert to estimate the output for a particular input. Our framework instead combines the output of all of its expert regressors using a weighted sum that is derived from a probabilistic clustering algorithm. Secondly, our framework features a separate dimensionality reduction function for each expert regressor. This is a particularly novel component of the framework described and it is not found in mixtures of experts.

It may seem obvious that the per-expert dimensionality reduction should be beneficial for our problem. Few pieces of research look into problems like ours where the data falls into clusters for which the optimal reduced-dimensionality representation can differ strongly for each of these clusters. Thus, although it makes sense for our problem domain, it doesn't seem beneficial for most other problem domains. Because of this, it remains untested, and there has been no framework described which explicitly incorporates expert regression functions with their own specialised dimensionality reduction functions.

4.4 Empirical analysis of the ensemble framework

In this section we will set up our novel ensemble framework with a selection of technologies and give justifications for these choices. We will then look at empirical results in terms of the four performance scores of interest given in section 2.6. We then attempt to convince the reader of the validity and significance of our findings using some analyses of the framework. Finally we evaluate the ensemble framework by the original aims.

4.4.1 Choice of three component technologies

We have aimed for an ensemble framework which makes few assumptions regarding the three component technologies: clustering; regression function; and dimensionality reduction. Despite this there are some modest restrictions on the clustering and regression function used.

In this section we will describe the technologies we used for each component with reasoning.

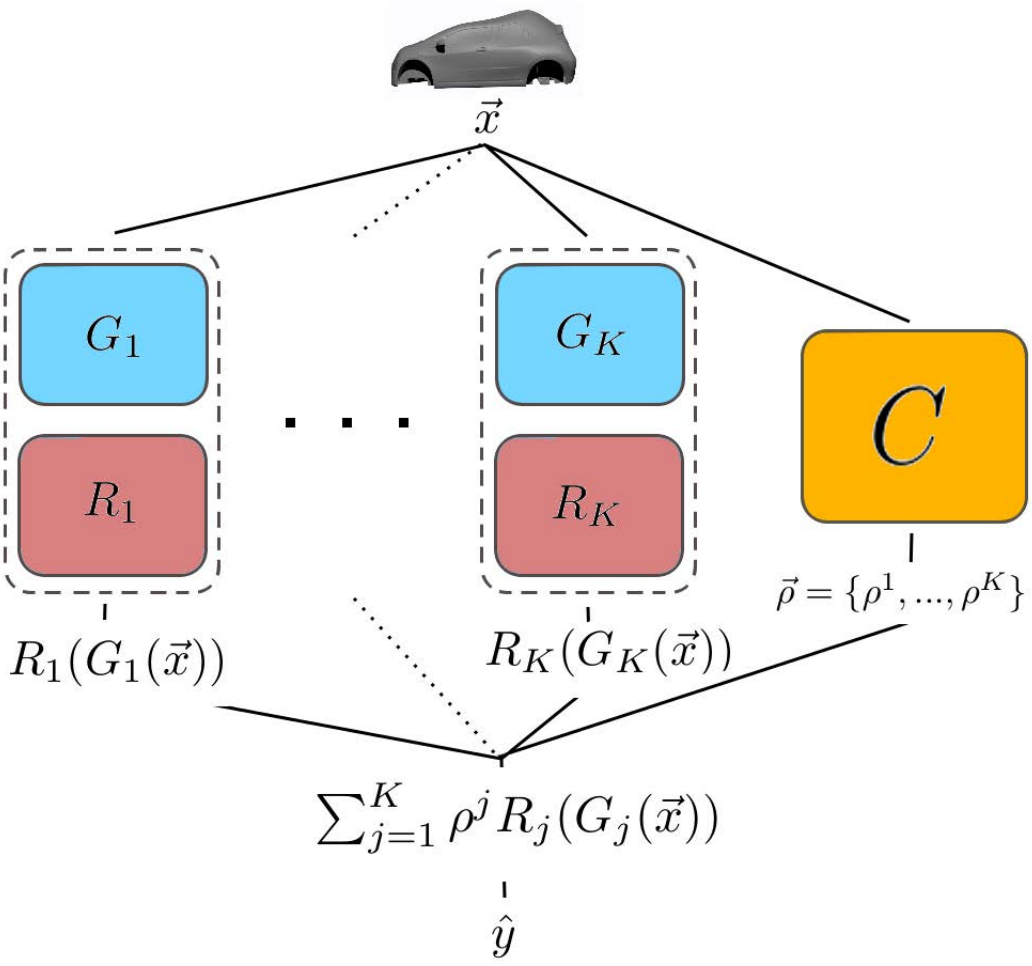


Figure 4.1: Diagram of the framework architecture. Notation follows that in section 3.

Clustering method

For our framework we are seeking a clustering technique for finding, unsupervised, the clusters in our training data set. The algorithm chosen must be able to assign an input with probabilities of belonging to each cluster.

As we seek a clustering mechanism capable of clustering the data set using probability assignments, GMM seems like a logical choice. Although we should note that any clustering approach can be made psuedo-probabilistic. K-means for example could assign probabilities based on distances to each centroid over the summed distances to all centroids.

Other considerations to make include the expected shape of the clusters in the data set. Since we generated the experimental data set to reflect a realistic data set containing design clusters (i.e. clusters of: design class (hatchback, saloon etc); style groups; or designs resulting from independent optimisation processes) we know that it contains clusters that follow combinations of trivariate Gaussian distributions. Gaussian cluster shapes like these are an entirely plausible property of a real data set and this rules out centroid approaches such as K-means for our experimental data set. On the other hand, hierarchical and hybrid approaches can deal with clusters of any shape (an L-shaped cluster for example). Such approaches are quite attractive but as we are aware of the cluster shapes in our data set, GMM will suffice.

For any real-world implementation it could be argued that by breaking an L-shaped cluster into two Gaussian clusters (as GMM would do) the design space would be further granularised, aiding the regression ensemble's performance.

So for these reasons we chose GMM as our clustering component. Our findings were that when the number of clusters $K = 8$, GMM was able to cluster the data set without error.

Regression function

When looking for a suitable machine learned regression function for our problem we considered a number of 'off-the-shelf' machine learning algorithms. Amongst them: MLP (the "Multi-Layer Perceptron" single-layer Feed-forward Network) [47, 51, 75]; Linear Regression [29]; ELM, the extreme learning machine [54]; E-SVM Regression, the Epsilon-Support Vector Machine algorithm [129]; CaR-Tree, the classification and regression-tree [18] ; and NN-Regression, the nearest-neighbour regression algorithm.

A major factor in the choice of regression function for use in the suggested framework is its ability to accept sample weights during the model parameter fitting process. We therefore choose MLP for future experiments because: it is easily modified to accept sample weights during training (through weighting of the error terms used in the backwards propagation); we consider it to be exemplar of a reliable machine learning algorithm that is well researched and understood; and in simple experimental comparisons with the other algorithms mentioned above it had resulted in a regression function with good accuracy on the testing set.

Dimensionality reduction

The final piece of the framework which is open to choice is the dimensionality reduction algorithm. Such an algorithm must ultimately result in a function $g : \tilde{\mathrm{x}} \rightarrow \tilde{\mathrm{x}}^{'}$ which can map an input $\tilde{\mathrm{x}} \in \mathbb{R}^{d}$ into a lower-dimensional input $\tilde{\mathrm{x}}^{'} \in \mathbb{R}^{d^{'}}$ where $d^{'} < d$. The dimensionality reduction function g is generally classed as either a filtering or extraction method [31].

Principal Component Analysis (PCA) [65] is a simple and effective extraction method. As described in section 3.3, under PCA the covariance of all attributes is used to find the eigenvectors describing the data set. By mapping the data onto the $d^{'}$ eigenvectors with the greatest eigenvalues, PCA effectively embeds the data in a representation which captures the greatest variance in the data.

In trials PCA was shown to outperform manifold techniques such as ISOmap [134] and Locally Linear Embedding [118] for D^{C}. The sensitivity method suggested in [42] was also trialled but resulted in representations too large to use.

4.4.2 Experimental studies using the ideal data set D^{C}

For our experiments we divided the 1200 samples of automobile data set D^{C} into 400 training data and 800 testing data, with uniformly distributed selection (without replacement). For just this chapter we use a split of 400:800 (instead of the 600:600 split seen elsewhere) in order to better demonstrate that our suggested ensemble framework performs well for problems with comparatively small data sets. We built each ensemble with eight MLP models (one MLP per cluster found) using the same parameters given in section 3.7.2 and for the same reasons given in that section. Convergence of each MLP was tested using a 120 sample validation subset taken from the training data without replacement (leaving 280 samples for training).

For each clusters PCA process we automatically selected the target dimensionality d' for each cluster by setting it to select the first d' eigenvectors whose eigenvalues summed up to $>99\%$ of the sum of all eigenvalues (this results in representation that is compact enough to work with with, but still large enough to include much of data variance).

The results for the framework using the aforementioned core components averaged over 1000 runs are given in table 4.1. These results have been tested for significance using the non-parametric Wilcoxon signed rank statistical test. A p value of 0.001 was used with the Holm–Bonferroni method of performing multiple comparisons in order to determine whether hypotheses should be rejected or accepted. Comparisons were drawn with a selection of ensemble techniques:

- Average-ensemble (simple averaging of eight independent regression function outputs).
- Bagging (using a bootstrap of 400 samples, selected with replacement) [15].
- Druckers AdaBoost [30].
- Random Forest [17].
- Negatively Correlated Learning (with two inner-iterations and $\lambda = 0.7$ as in [81]).

We see that for this problem the suggested framework performs best on average, achieving notably better scores than all competitors. Looking at PPC, the score of most interest to us, we see that it is almost at 0.9, while others are closer to 0.88.

Looking at the box-plot graphs of this data, in figures 4.2 and 4.3, we get a more solid impression of the performance of the suggested framework. In the box-plots: the red line through the middle indicates the position of the median sample; the box itself represents 50% coverage of the data, with its lower and upper edges at the 25% and 75% quantile of the data respectively; the maximum whisker/error-bar length is 1.5 times the height of the box (roughly equal to $\pm 2.7\sigma$ and 99.3% coverage); and all samples outside of these maximum whisker lengths are then plotted as outliers (red crosses).

We see that the suggested framework performs best in terms of all the data shown in the graphs. Notably, the outliers of Random Forest for the r^2 are very strong and of the run performed, these out perform the suggested framework. But the mere fact they are outliers casts doubt on the performance of Random Forest.

Focusing on the PPC box chart, we see that the lower quartile of the suggested framework's data is above the upper quartile of all other methods tested. This is an encouraging result.

Druckers AdaBoost seems to perform poorly but this can be attributed to the limitation on ensemble size (eight models) which doesn't allow a good Boosting ensemble to form. The other ensemble methods tested all perform very similarly to one another.

In order to try and demonstrate the validity of these results we also test on two further splits of the data (considering the first split to be all the samples). Split two, containing clusters A, F, G, and H is chosen as these designs are somewhat similar (see figure 3.6). The other clusters, B, C, D and E form a third split. For both splits the 600 training data were divided into 200 training and 400 testing data, and the eight MLP models were reduced to four to match the expected clusters in the data set. Results for these splits are in tables 4.2 and 4.3 respectively.

In these tables we see that the average performance of the suggested framework is better than that of the comparative ensemble methods. The split represented in table 4.3 seems to have contained data which was particularly difficult to learn from or estimate for correctly. We see that in this table Random Forest achieves a comparable average r^2 score to the suggested framework, while Druckers AdaBoost has particular trouble learning.

Studying the box plots in figures 4.4 through 4.5 we see that for the table 4.2 split of the data, the suggested framework again exhibits good performance with its PPC score again showing a lower quartile which is above the upper quartiles of all other approaches.

For the seemingly difficult table 4.3 split of the data, in figures 4.6 and 4.7 we clearly see that while the suggested framework generally performs best, it has produced some outliers which distort the graphs considerably. This highlights a possible issue with the suggested framework: by focusing regression functions on individual clusters we effectively make each regression function more susceptible to outliers in the data set.

Despite this, we reiterate that our approach is a more transparent modelling approach than that of the other 'black box approaches' such as Random Forest: a very complicated modelling structure which is challenging to understand from the outside. The suggested framework is an example of a far more transparent approach, which may instil confidence in designers and engineers alike, with each ensemble-member having a logical meaning. In the case where one of the regressors has been substantially affected by outliers (as is the case in table 4.3), it would be reasonable to remove that regressor or supply it with more training data local to its cluster.

Averages	MSE	r^2	PCE	PPC
Average-ensemble	0.144	0.857	0.517	0.887
Bagging	0.145	0.856	0.517	0.887
Druckers AdaBoost	0.169	0.834	0.603	0.876
Random Forest	0.152	0.859	0.541	0.885
NCL	0.142	0.859	0.509	0.889
Suggested framework	**0.119**	**0.882**	**0.426**	**0.898**

Table 4.1: Performance on split one of data: all 1200 samples. Bold indicates the best score for that particular test. All results have been statistically tested and found to be significant using a combination of Wilcoxon signed rank tests and the Holm–Bonferroni method with $p = 0.001$.

Averages	MSE	r^2	PCE	PPC
Average-ensemble	0.118	0.884	0.586	0.905
Bagging	0.120	0.882	0.595	0.904
AdaBoost	0.136	0.867	0.688	0.896
Random Forest	0.137	0.889	0.671	0.899
NCL	0.137	0.866	0.683	0.896
Suggested framework	**0.096**	**0.906**	**0.479**	**0.916**

Table 4.2: Performance on split two of the data, Clusters AFGH. Bold indicates the best performance for that score. All results have been statistically tested and found to be significant using a combination of Wilcoxon signed rank tests and the Holm–Bonferroni method with $p = 0.001$.

'Black box' methods do not allow this.

4.4.3 Analysis of the framework's performance on the ideal data set D^C

In this subsection we run some further experiments to look at the functioning of the ensemble framework for the experimental data set.

Averages	MSE	r^2	PCE	PPC
Average-ensemble	0.263	0.740	0.905	0.844
Bagging	0.270	0.734	0.930	0.842
Druckers AdaBoost	0.342	0.672	1.173	0.819
Random Forest	0.261	**0.767**	0.887	0.848
NCL	0.280	0.726	0.964	0.838
Suggested framework	**0.237**	**0.768**	**0.811**	**0.856**

Table 4.3: Performance on split three of the data, clusters BCDE. Bold indicates the best performance for that score. All results have been statistically tested and found to be significant using a combination of Wilcoxon signed rank tests and the Holm–Bonferroni method with $p = 0.001$.

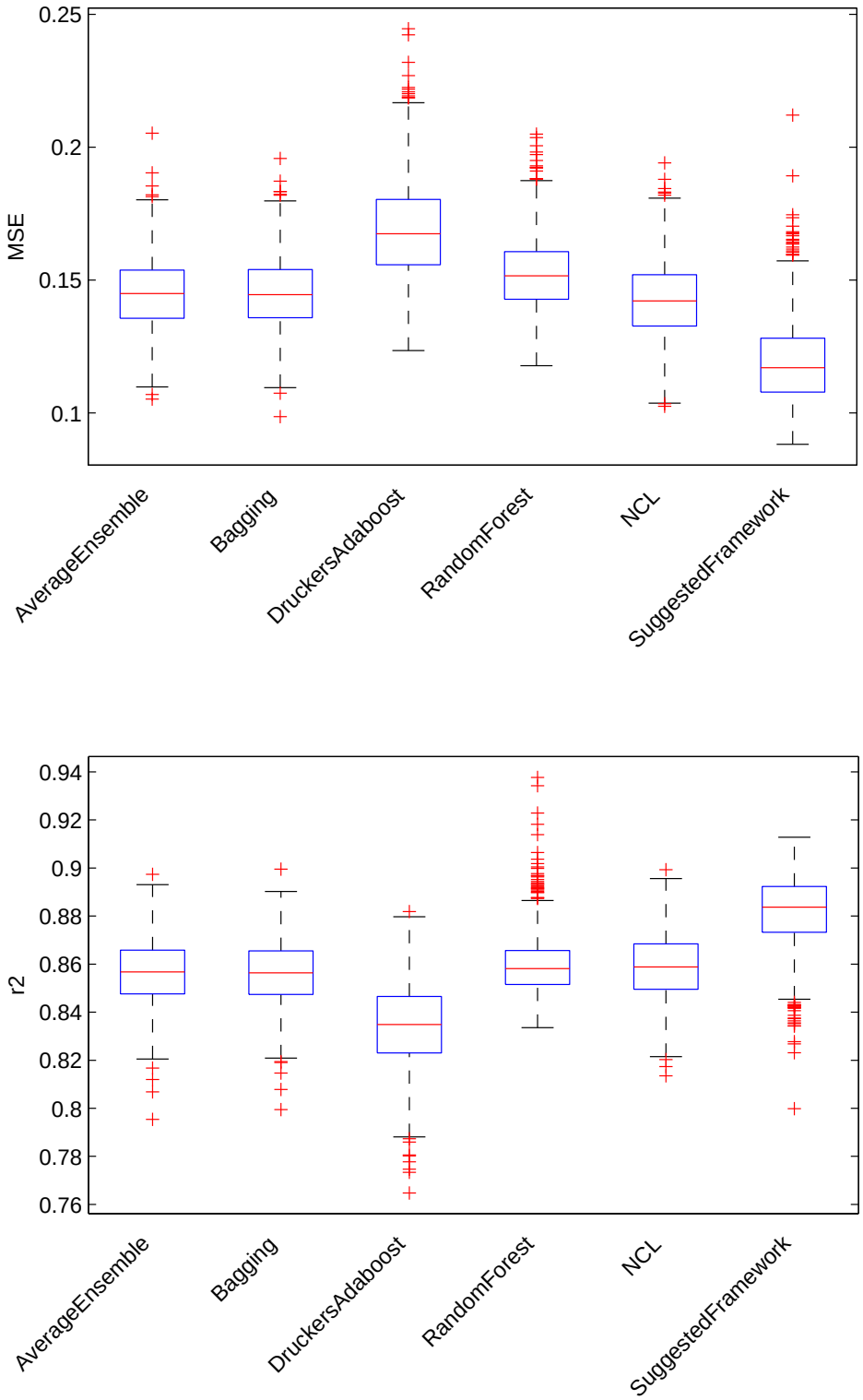


Figure 4.2: Box charts for MSE and r^2 scores to accompany the data in table 4.1.

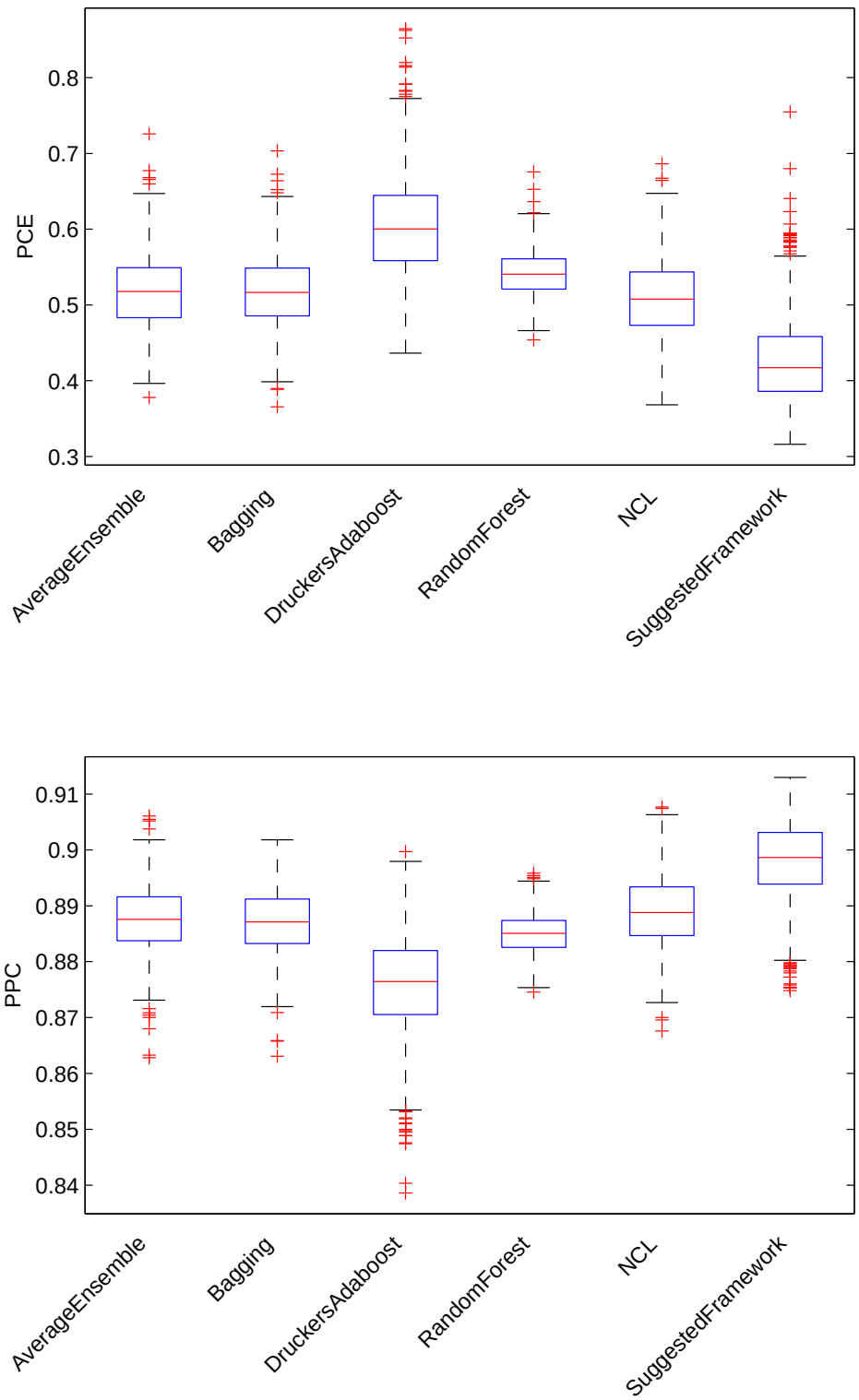


Figure 4.3: Box charts for PCE and PPC scores to accompany the data in table 4.1.

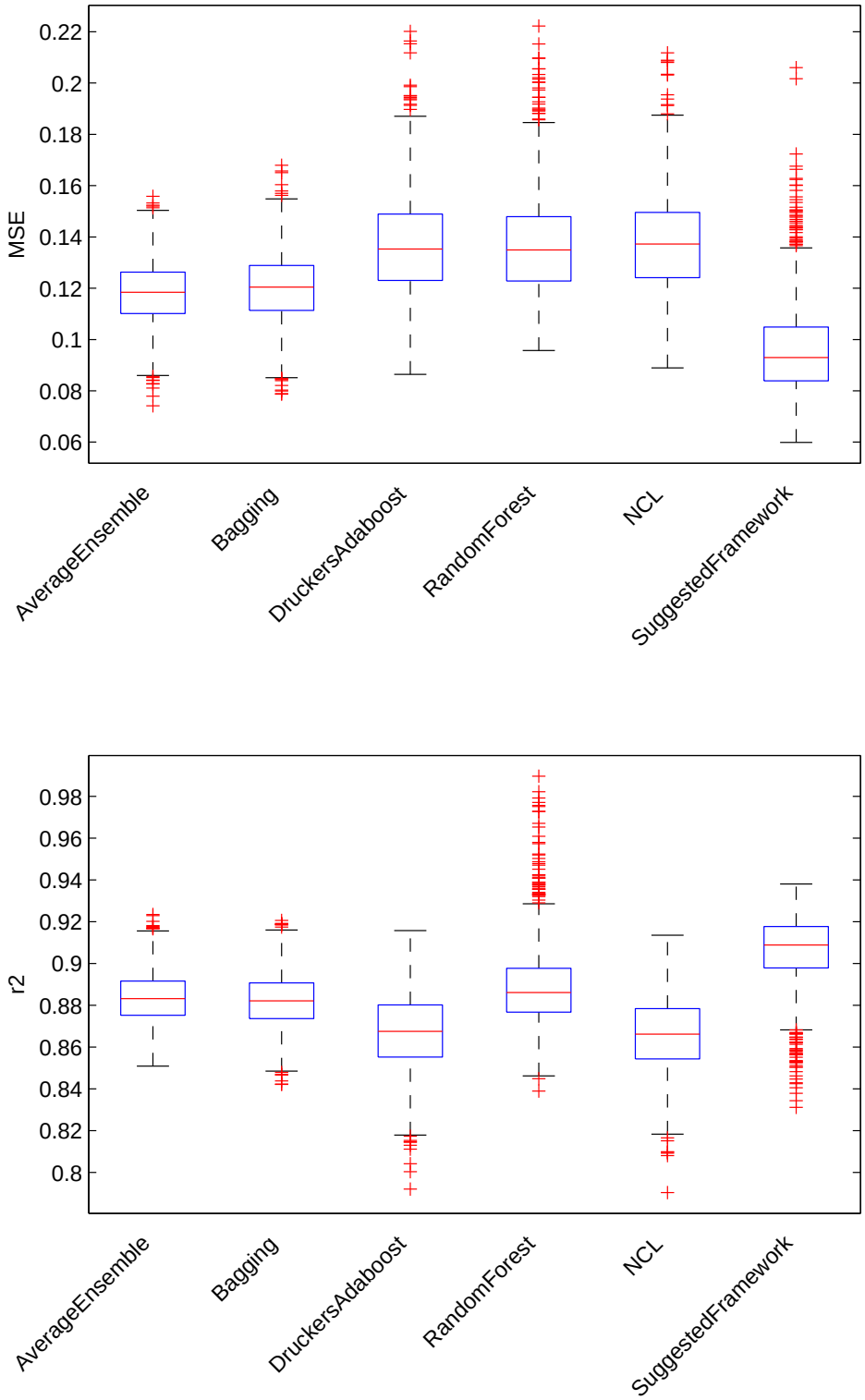


Figure 4.4: Box charts for MSE and r^2 scores to accompany the data in table 4.2.

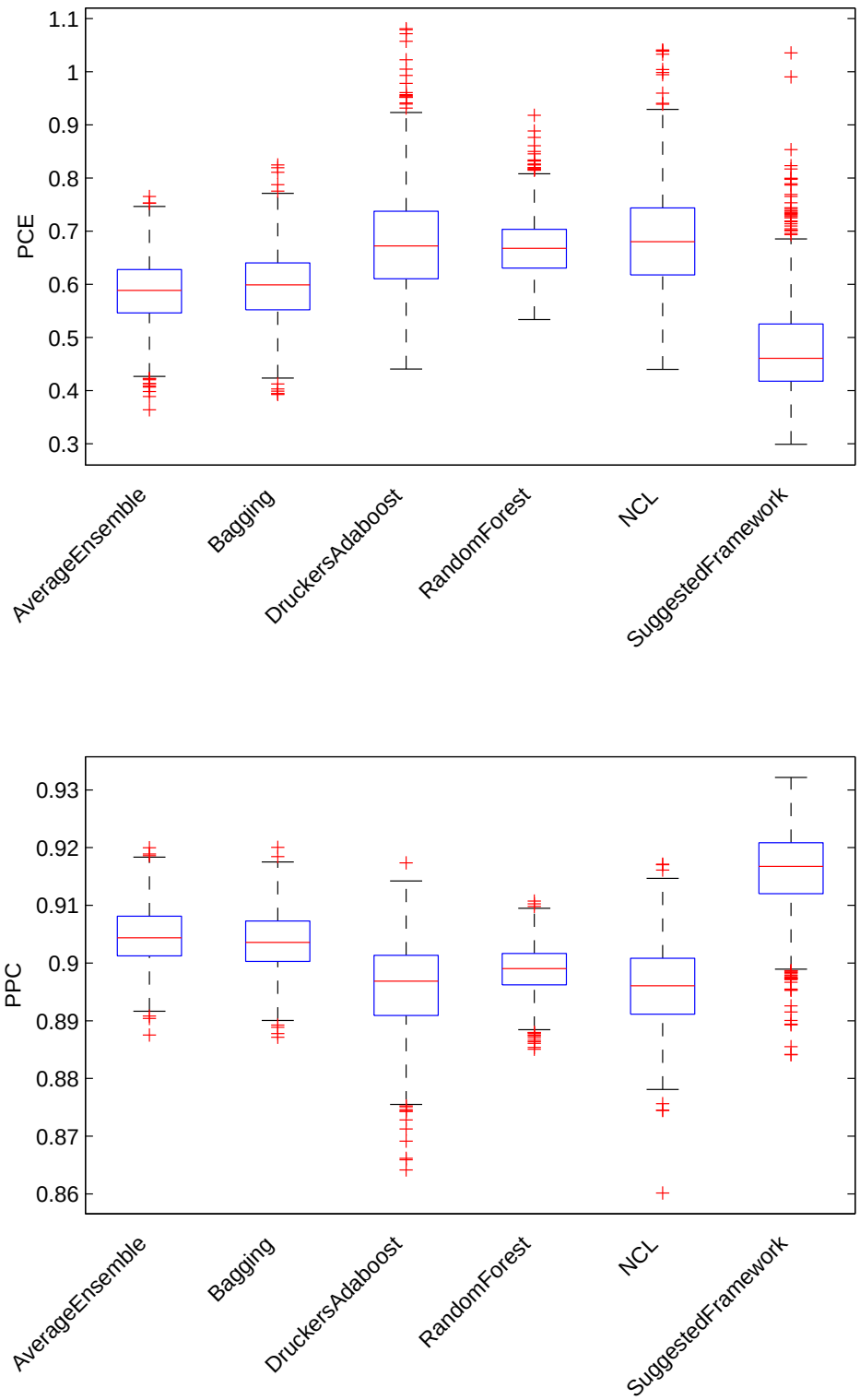


Figure 4.5: Box charts for PCE and PPC scores to accompany the data in table 4.2.

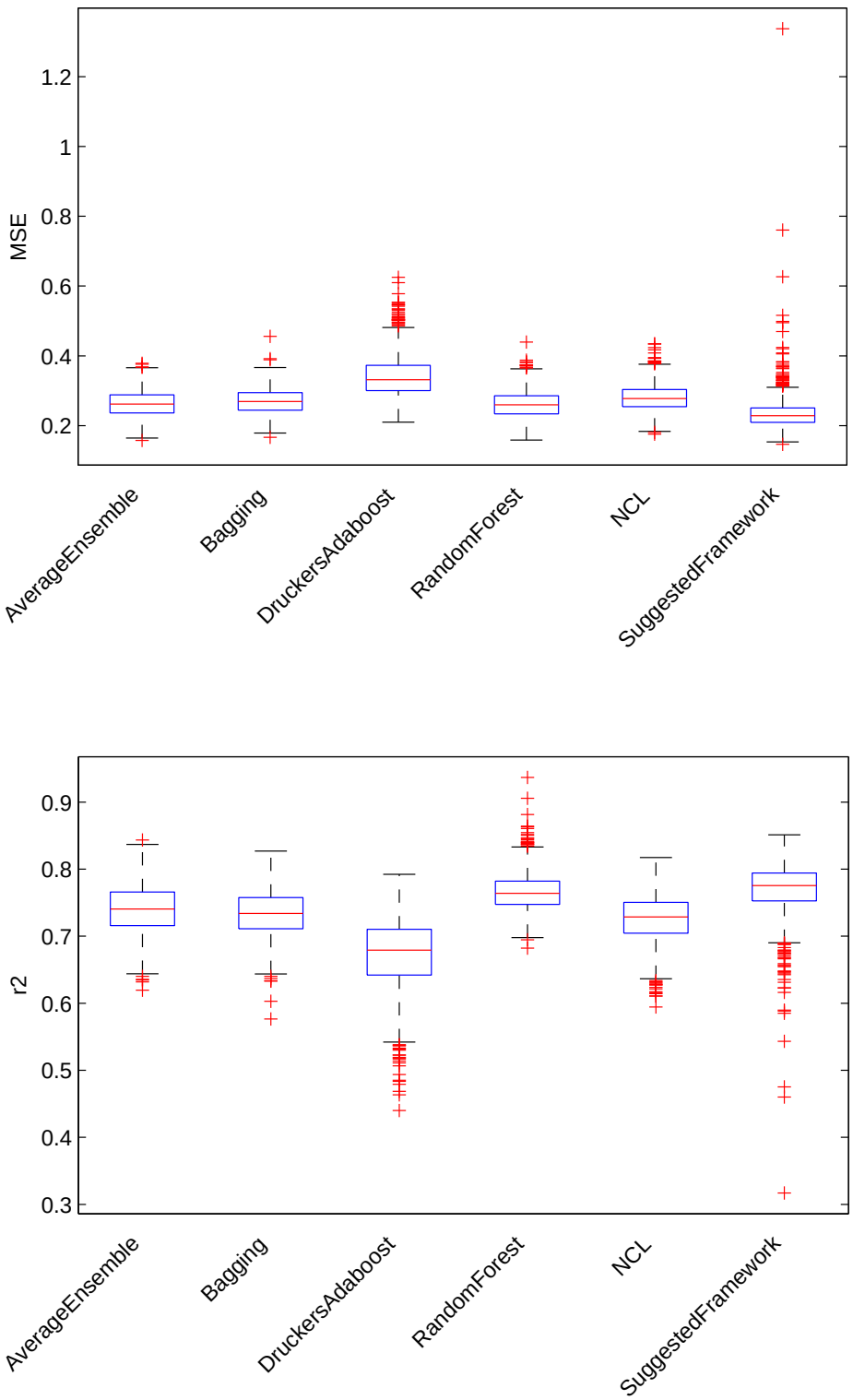


Figure 4.6: Box charts for MSE and r^2 scores to accompany the data in table 4.3.

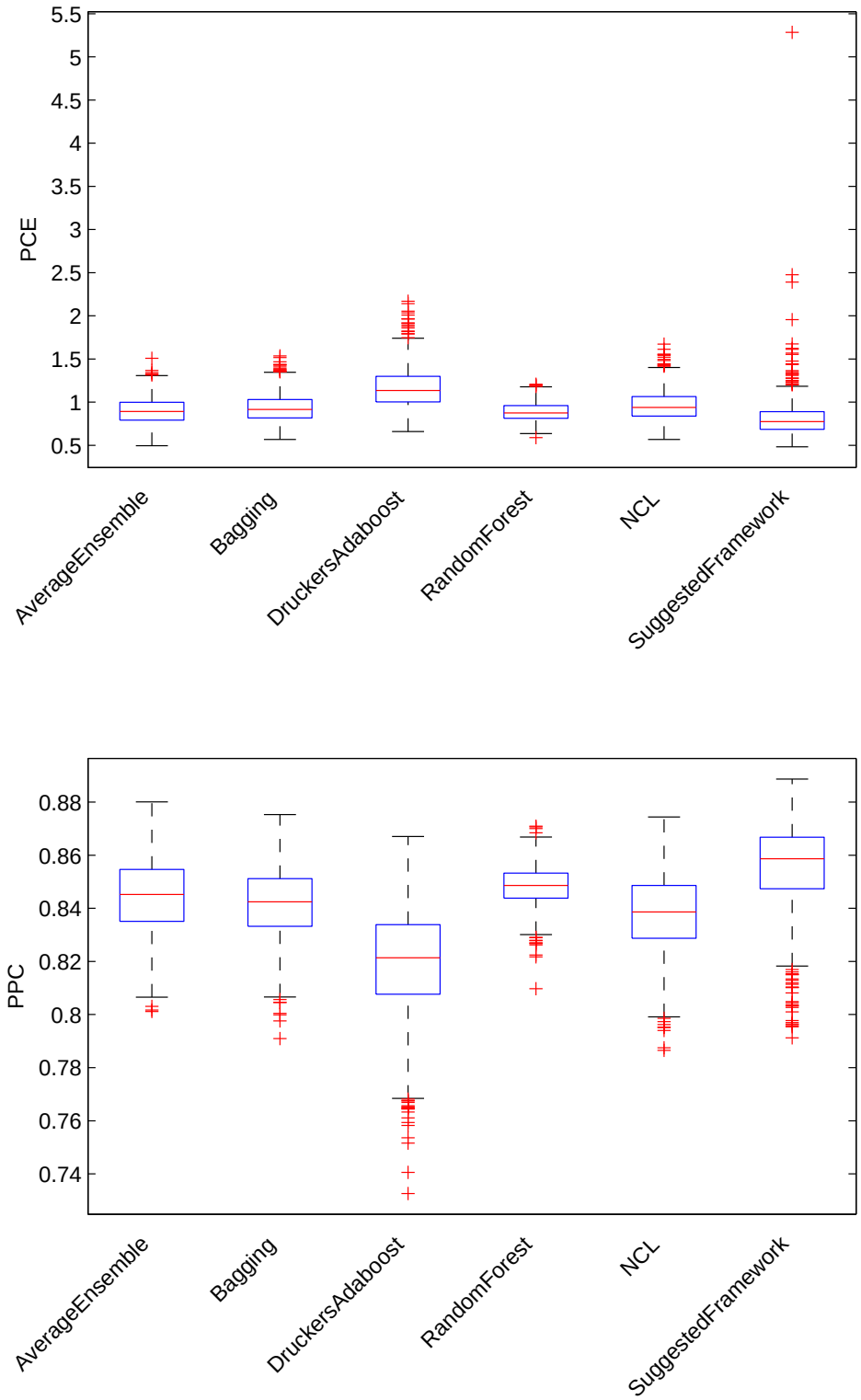


Figure 4.7: Box charts for PCE and PPC scores to accompany the data in table 4.3.

Averages	Models	MSE	r^2	PCE	PPC
Suggested framework	8	0.133	0.868	0.481	0.894

Table 4.4: Performance of the ensemble framework where $\gamma = 0$.

The effect of per cluster dimensionality reduction

We have seen that our novel ensemble framework performs better than other ensemble frameworks for this task. Throughout, we have suggested that this is due to a particularly novel component: dimensionality reduction applied to each cluster individually.

We then wonder, are the performance benefits really enhanced by this feature or is having a regressor for each cluster mostly responsible for this? In order to answer this we set up an experiment where the candidacy-threshold $\gamma = 0$, meaning that dimensionality reduction (here PCA) for each cluster was identical (as it would include all candidates, all samples in D, in its dimensionality reduction). The experiment was otherwise identical to the experiment producing results in table 4.1.

The result from this experiment are in table 4.4. By comparing table 4.4 to table 4.1 we can see that both of these components have a positive effect, but that most of performance improvement must be due to the cluster-specific dimensionality reduction.

Differences between each cluster's dimensionality reduction

If we look at table 3.1, we see a variation in the number of frozen FFD control-point groups per design cluster. Since we implemented an automated selection of target dimensionality in the PCA process of section 4.4.2 experiments, we expect variation between the selected target dimensionality for each cluster, which should correlate with the frozen FFD control-point groups. I.e. for a cluster with more frozen FFD control-point groups, we expect a lower target dimensionality to be chosen (since fewer parameters are free).

Running the same experiments as those in table 4.1, table 4.5 shows the average selected target dimensionality (with standard deviation) chosen for each cluster. A correlation can indeed be observed, reinforcing our belief in the underlying function of the ensemble framework.

Cluster	Avg dimensionality [std-dev]	Free CP-groups
A	7.101 [0.469]	7
B	5.81 [1.087]	6
C	6.81 [1.039]	8
D	5.481 [0.782]	7
E	4.19 [0.893]	6
F	6.747 [0.967]	9
G	3.899 [0.810]	4
H	7.848 [0.833]	9
Whole data	6.25 [0.73]	11

Table 4.5: Dimensionality after PCA for each cluster found (averaged over 100 runs with standard deviation ('std-dev') given in square brackets). The number of free control-point groups is given for each class for comparison.

Differences between clusters in terms of functional performance

One of the benefits of the suggested framework is that it clearly partitions the design space in a human-perceptible way. For a number of engineering problems, including the one described in this chapter, we find that this is very useful because the relationships between input attributes and the functional performance vary quite strongly in the space of possible designs.

In order to demonstrate this, we look for an attribute in two clusters of our experimental data set which behave in very different ways owing to the subspace of possible designs which the clusters inhabit. We were able to identify a single attribute in clusters D and G which behaved in an interesting manner.

Examples of clusters D and G are repeated in figure 4.8 and a red dot on the image illustrates the location of the point containing this attribute. This attribute represented the y-coordinate of a point on the side. For cluster D, it emerged that increasing this attribute (bringing the red dot closer to the camera in the side view) caused the automobile's drag to decrease (from -0.0745716 normalised drag value with a change of -0.2 meters to -1.06294 +0.2 meters). Conversely, for cluster G, the same change would cause the automobile's drag to increase (from 0.256261 to 0.804465 respectively).

This simple demonstration shows how the clustering of data set in order to partition the design space can allow for easier identification of rules which can be used both in improving a regression function, and for imparting knowledge on a human designer.

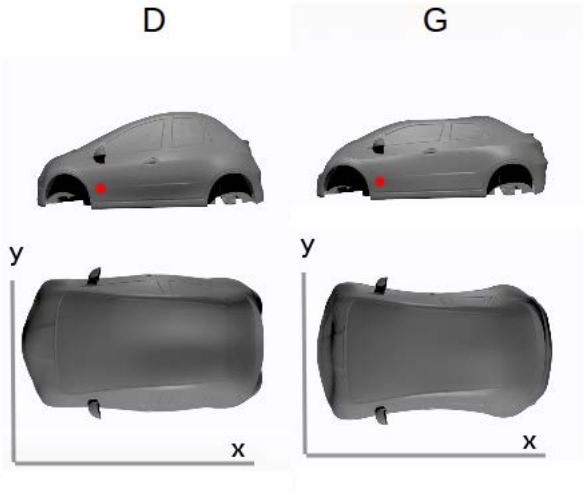


Figure 4.8: Clusters D and G. The red spot along the side just below the wing mirror illustrates the location on the surface of the interesting attribute.

Utility for the designer

When this system is available to the designer they have available a system which can quickly determine similar designs according to its clustering algorithm. Then the system can be used, rather than as an assessment tool, for:

- looking at the attributes with most variance in clusters similar to a design of interest in order to help the designer consider popular modifications for this design.
- analysing the relationship between design attributes and the output for a particular design cluster with little extra effort.

For example, if we create a new sample by $\vec{C} = 0.5(\vec{A}) + 0.5(\vec{B})$ where A and B are two samples in different clusters, then we have created a sample which doesn't totally fall into either of A's or B's cluster. Doing this with a sample from *cluster* B and another from *cluster* C resulted in a design which was clustered into cluster F with probability 0.809 and cluster H with probability 0.191. If this design had been the designer's initial impression then they could look at clusters F and H (with more emphasis on F) and consider the sort of design changes that are popular for these design clusters.

If then the designer was required to improve the functional performance of the design, they could use the models associated with F and H (again, with more emphasis on F) in order to rank attributes by the sensitivity of the functional performance towards these. They may find

surprising options emerge which improve the designs performance while having little effect on the subjective aesthetic quality of the design.

4.4.4 Evaluation of the ensemble framework by the original aims

In this section we refer to the original aims listed in section 4.3.1:

Has good generalisation across a large space of possible designs

This is partly dealt with through use of an ensemble to improve generalisation. In particular though, we also deal with this by forcing each expert regressor to become specialist in a particular sub-space of this large space of possible designs.

Incorporates multiple low-dimensional representations of the data (through multiple dimensionality reduction mapping functions)

This is key to the framework and explicitly performed.

Focuses constituent regression functions on individual subspaces of the design space based on the geometric similarity of designs in the data set

This is also key to the framework. The geometric similarity is calculated by the probabilistic clustering algorithm, and an expert regressor is built for each cluster.

Can interpolate smoothly between the subspaces

The use of a probabilistic clustering function, assigning probabilities to each cluster (and their expert) which must always sum to one, enables this property.

Is internally modularised in a meaningful way to enable designer utility and promote trust

Geometric similarity of engineering surface designs is a property that can be seen and understood by a human designer. That designs in a cluster behave more similarly than those outside is also intuitive for a human designer. In creating per-cluster expert regressors, with their own optimised representations, the ensemble framework attempts to tackle the problem in a way that the designer knows what is going on internally. As thus, this property is met.

4.5 Conclusion

4.5.1 Summary

This chapter looked at the task of learning a regression function which can be integrated into the engineering surface design software, using a historically accumulated data set of previously interesting surface designs.

This class of problem is therefore unusual as the training data set contains results from discrete design processes, taken from a large design space. The data set covers only a narrow subspace of this large design space, non-uniformly, and through this it contains knowledge regarding interesting subspaces of the design space. Further, the input attributes across this data set affect performance in a non-linear manner, such that their effect varies strongly across the design space, and the variance and covariance of the data is considerably different in different sub-spaces of the design space.

We presented a novel ensemble framework for generating a regression function suitable to this problem. Our framework uses the prior engineering knowledge about the problem and the training data to find interesting subspaces of the design space through the training set and to improve accuracy by allowing individual design space regions their own low-dimensional mapping. It also has the added benefit of being transparent and easier to conceptualise, potentially being of greater utility to designers than typical "black-box" approaches such as Random Forest.

We tested our novel ensemble framework using the 'ideal' experimental automobile data set D^C. In comparison with other popular ensemble frameworks, our novel framework performed tended to perform better for this regression problem, especially in terms of the PPC score which is especially relevant to our problem.

We also tried to demonstrate that our framework performs as suggested through analysis of its functioning on set tasks. We demonstrated the positive effect of the particularly novel calculation of cluster-specific dimensionality reduction functions, and further demonstrated that performing this was important through analysis of each cluster's dimensionality reduction function.

4.5.2 Novel contributions

The major contributions of this chapter include a novel ensemble framework for generating a regression function suitable to the problem central to this thesis; a demonstration on an experi-

mental data set which exhibits these properties, showing the benefits of the ensemble framework; analysis of the framework; and proof that this framework is less of a black box and thus could be of greater utility to a human designer.

4.5.3 Future work

Future research may look at the qualitative and/or quantitative benefit of using this regression framework in the real-world engineering design process. Another direction may focus on merging clusters according to some functional similarity (e.g. the dimensionality reduction function is very similar). This would simplify the ensemble while strengthening the sample support for the resulting merged cluster, potentially improving its accuracy.

Already mentioned is that the historical data set we would be working with contains knowledge about designs which were previously interesting to designers. This spares us having to generate data using an automated *design of experiments* methodology, which would likely sample from regions of the design space which contain uninteresting designs. The topic of active learning looks at strengthening a learned algorithm after it has been generated, using new data as it is generated. As mentioned, the ensemble framework utilises the knowledge in the training data set in order to determine the interesting subspaces of the large design space. We can use this extracted knowledge (i.e. the clusters) along with their individual low-dimensional mapping functions, in order to generate new data which may be of greater use to online learning for this ensemble framework.

Chapter 5

A Complete System for Performance Estimation of Surface Designs

So far in this thesis we have tackled two constituent challenges in creating our functional performance estimation system: the preparation of our accumulated data set; and building an ensemble regression function with improved performance for this data set. In this chapter we consider how to combine these independently developed technologies, suggesting multiple methods of using dimensionality reduction in between the two sub-systems.

This chapter aims to suggest and evaluate methods of combining the novel solutions of chapters 3 and 4 in creation of a complete system for solving the problem of estimating functional performance based on engineering surface design data. Thus, this chapter will assume that we are working with the realistic automobile data set D^R, and the spheres data set D^S.

The complete system will be analysed empirically using the performance scores developed in section 2.6 in order to assess its suitability for the task. We will also discuss ways this functional performance estimation system could be integrated into the aesthetic design process.

5.1 Problem introduction

In this thesis we have presented a challenging task found in real world engineering, dividing this single task into two challenging sub-problems. We have then put forward novel solutions for each

of these two sub-problems, which have been analysed, solved and experimented with in isolation.

We are then faced with the question of how best to combine these two sub-systems to create a complete functional performance estimation system for solving the core problem behind this thesis. This is a non-trivial task as there are reasons why we may want to perform a layer of dimensionality reduction in between the two independent systems:

1. We may wish to reduce the computational burden for the ensemble framework.

2. We may wish to reduce the effects of the curse of dimensionality on the ensemble framework.

3. We may have reasons to believe that totally different dimensionality reduction methods will be optimal for the clustering and regression tasks in the ensemble framework.

The question arises: what different methods are there, and which might perform best?

5.2 Combining KGrid with the ensemble framework

In this section we will begin by listing the aims of a complete system and then suggest several ways in which the two systems may be combined with (or without) a layer of dimensionality reduction in between.

5.2.1 Aims of a complete system

Our original motivation was to improve the convergence rate of engineering design processes where both quantitative and qualitative optimisations are sought. In particular, we noted that qualitative performance, being human subjective, must be optimised by a human hand (as is the case with aesthetic quality). At the same time, quantitative objectives (such as functional performance scores like aerodynamic drag) are also being maintained and improved. The qualitative objectives, being optimised by human hand, could often have negative impacts on the quantitative objectives. In order to mitigate this risk, we consider the integration of some functional performance estimation system in the aesthetic design process.

In this thesis we focused on the implementation of this functional performance estimation function, suggesting that it could learn from the data set of historically accumulated and previously interesting engineering surface designs and their known functional performance scores.

Thus, our complete system should be able to take such a data set of surface designs with diverse surface design representations (such as the STLs in D^R and D^S) coupled with their known functional performance scores, and generate a functional performance estimation function with the following properties:

- Must generalise well to future previously unseen designs in varied and previously unseen representations, allowing their functional performance estimates to be output.
- Must deal well with a very large representation for which it is hard to find a representation that balances compactness and completeness well.
- Must generalise well based on a relatively small training data set taken non-uniformly from a large design space, despite having a very large representation.
- Need not be particularly accurate in a single subspace of designs, but rather be fairly accurate throughout the space of possible designs, and highly accurate in multiple areas of interest from an aesthetic design perspective (determined from the training data set).
- Must be able to take any surface design and quickly (seconds) calculate an estimate of that surface designs functional performance score.
- Offers the human designer potential utility.

5.2.2 Methods of forming a complete system

In this section we will talk about methods of combining KGrid from chapter 3 with the ensemble framework of chapter 4, to form a complete system for functional performance estimation of engineering surface designs. Please note, that although dimensionality reduction happens inside each expert regressor of the ensemble framework, KGrid is a necessary first step in the system. This is because if we are to use the suggested ensemble framework with D^R or D^S, then clearly the samples must be pre-processed such that they can be used by the probabilistic clustering algorithm of the ensemble framework. With this constraint, we can suggest the following ways of combining both sub-systems into a single complete system:

1. KGrid converts the data into a unified representation D^U with dimensionality d^U. The ensemble framework (which involves performing per-cluster dimensionality reduction) can

then be used with D^U but the probabilistic clustering central to the ensemble framework may perform very slowly owing to the large d^U.

2. KGrid converts the data into a unified representation D^U with dimensionality d^U. We then apply a stronger dimensionality reduction g_V to D^U that reduces the dimensionality from d^U to $d^V \ll d^U$(but still large enough to be a very complete representation of D^U) as data set D^V. The ensemble framework (which involves performing additional per-cluster dimensionality reduction) can then be used with D^V. This adds the computational burden of fitting a dimensionality reduction function g_V but reduces computational burden in the ensemble framework.

3. KGrid converts the data into a unified representation D^U with dimensionality d^U. We then apply a stronger dimensionality reduction g_V to D^U that reduces the dimensionality from d^U to $d^V \ll d^U$(but still large enough to be a very complete representation of D^U) as data set D^V. The ensemble framework then uses D^V just for clustering the data, but then performs its per-cluster dimensionality reduction and regression on data set D^U. This also adds the computational burden of fitting a dimensionality reduction function g_V but reduces computational burden ***only*** *during clustering* in the ensemble framework.

4. KGrid converts the data into a unified representation D^U with dimensionality d^U. We then apply a stronger dimensionality reduction g_V to D^U that reduces the dimensionality from d^U to $d^V \ll d^U$ as data set D^V. We then apply a stronger dimensionality reduction g_W to D^U that reduces the dimensionality from d^U to $d^W \ll d^U$ as data set D^W. The ensemble framework then uses D^V for clustering the data, but then performs per-cluster dimensionality on D^W. This adds the computational burden of fitting a dimensionality reduction functions g_V and g_W but reduces computational burden optimally in the ensemble framework for both cluster and regression.

Each of these four approaches is illustrated diagrammatically in figure 5.1 to help aid conceptual distinction.

It is important to understand that before the completed system can be used, it must complete an initial setup process such that the ensemble and dimensionality reduction functions are trained and ready to receive data.

The suggested ensemble framework requires specification of:

1. Training data.
2. A probabilistic clustering algorithm and a number of clusters to find (assuming these will not be automatically derived).
3. A base regressor learning algorithm and its parameters.
4. A per-cluster dimensionality reduction learning function and its parameters.

Clearly each of these is arguably problem specific (refer to comments on choice of these components in chapter 4.4.1). Of particular importance to the creation of one of these combined systems is how one chooses the dimensionality reduction functions, as there exist multiple levels of dimensionality reduction in the four combinations methods suggested above. The reasons why multiple representations may be beneficial for each combinations is tied closely to the dimensionality reduction method to be chosen. For each suggested combination option above, respectively:

1. Here we only have each cluster's dimensionality reduction function g_2^i for mapping D^U to D^i (for $1...i...K$ clusters) and so this need only be good for per-cluster regression purposes.
2. Here we first want a balanced dimensionality reduction function g_1 for mapping D^U to D^V which is suitable to clustering and forming D^i from. Then we need a dimensionality reduction function g_2^i for forming each D^i (for $1...i...K$ clusters) from D^V.
3. Similar to option 2, but now g_1 can be specialised for forming a representation of D^U suitable for clustering the data while g_2^i must be able to form each D^i from D^U.
4. Similar to option 3, but with an additional choice of dimensionality reduction $g_{1'}$ for generating D^W. Clearly g_1 need only be suitable for forming a representation of D^U specifically for clustering the data, while $g_{1'}$ needs to be chosen optimally for forming a representation of D^U suitable for each cluster's g_2^i function (which leads onto regression).

KGrid is a non-stochastic process and so does not necessarily require an initialisation phase (but note, finding the initial grid dimensions or some form of grid optimisation can be considered part of the system initialisation phase).

We now take a closer look at each of the four suggestion combination methods, making a bit clearer the unique properties of each and their advantages and disadvantages.

Option one: no dimensionality reduction between KGrid and the ensemble framework

Under this approach we have $2 + K$ different representations of the data set: D^R/D^S, D^U and D^i for $i = 1...K$ (where K is the number of clusters in the ensemble framework).

The system setup process for option one is presented in algorithm listing 5.1. Clearly, once the system has been setup, an unseen input to be assessed by this system must pass through the same representation conversion process. This entails the process listed in algorithm listing 5.2. Advantages of this approach:

- Each cluster receives a very complete representation of the data and is better able to find its own optimal dimensionality reduction mapping.
- Assessment of an unseen input only requires K applications of dimensionality reduction (compare this with the other options coming up).

Disadvantages of this approach:

- Clustering is performed on a very high-dimensional representation and thus could take some time without prior dimensionality reduction.
- A large amount of computational effort is required when (i) fitting a dimensionality reduction function for each cluster and (ii) converting an unseen x^U into each cluster's low-dimensional representation.

Option two: a single dimensionality reduction between KGrid and the ensemble framework

Under this approach we have $3 + K$ different representations of the data set: D^R/D^S, D^U, D^V and D^i for $i = 1...K$ (where K is the number of clusters in the ensemble framework).

The system setup process is presented in algorithm listing 5.3. The process of assessing a previously unseen input is listed in algorithm listing 5.4. Advantages of this approach:

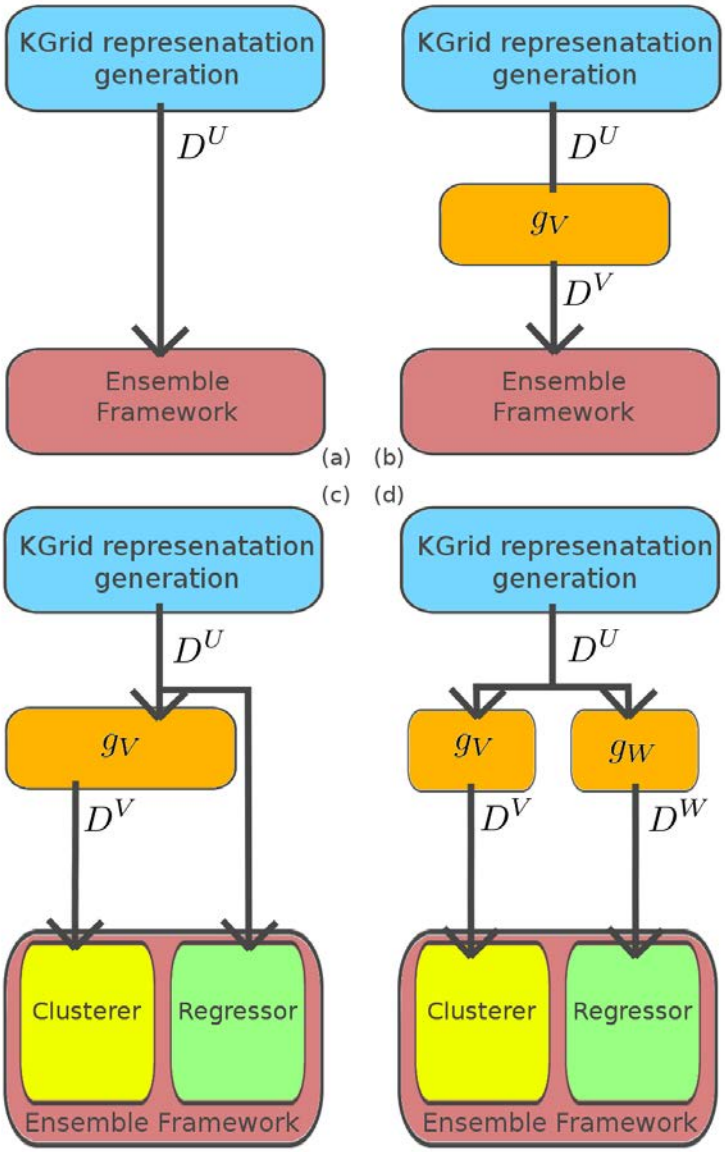


Figure 5.1: Diagrammatic representation of each of the four options: (a) option one; (b) option two; (c) option three; and (d) option four.

Algorithm 5.1 Option one system initialisation phase.

1. Given the training data D consisting of N samples, suggested framework parameters (K, and regressor parameters), and the KGrid parameter τ.
2. Setup KGrid with the given τ value, taking the min and max in each axis from analysis of the data set (plus the recommended ε padding).
3. Use the KGrid to generate data set D^U of standardised K-Grid representations.
4. Train a suggested framework $\hat{f}_{ens}$ ensemble using D^U, and the given suggested framework parameters.

Algorithm 5.2 Option one assessment of a previously unseen input.

1. Given unseen input x.
2. Generate the KGrid representation x^U.
3. Output the estimate $\hat{f}_{ens}(\mathrm{x}^U)$.

Algorithm 5.3 Option two system initialisation phase.

1. Given the training data D consisting of N samples, suggested framework parameters (K, and regressor parameters), and the KGrid parameter τ.
2. Setup KGrid with the given τ value, taking the min and max in each axis from analysis of the data set (plus the recommended ε padding).
3. Use the KGrid to generate data set D^U of standardised K-Grid representations.
4. Perform dimensionality reduction on D^U giving D^V.
5. Train a suggested framework $\hat{f}_{ens}$ ensemble using D^V, and the given suggested framework parameters.

- The ensemble framework must only operate on a lower-dimensionality version of the data, for both clustering and per-cluster dimensionality reduction, saving computational effort.

Disadvantages of this approach:

- Requires that an unseen input x be converted into representation x^V, which could be costly.
- Assessment of an unseen input requires $K + 1$ applications of dimensionality reduction.
- Adds more computational burden to setup when fitting the $g : D^U \rightarrow D^V$ dimensionality reduction function.

Algorithm 5.4 Option two assessment of a previously unseen input.

1. Given unseen input x.
2. Generate the KGrid representation x^U.
3. Generate a low-dimensional representation of x^U, x^V.
4. Output the estimate $\hat{f}_{ens}(\mathrm{x}^V)$.

Algorithm 5.5 Option three system initialisation phase.

1. Given the training data D consisting of N samples, suggested framework parameters (K, and regressor parameters), and the KGrid parameter τ.
2. Setup KGrid with the given τ value, taking the min and max in each axis from analysis of the data set (plus the recommended ε padding).
3. Use the KGrid to generate data set D^U of standardised K-Grid representations.
4. Perform dimensionality reduction on D^U giving D^V (a representation specifically for clustering).
5. Train a suggested framework $\hat{f}_{ens}$ ensemble using D^V **and** D^U, using D^V for clustering, D^U as a pre-regression representation, with the given ensemble framework parameters.

Option three: a single dimensionality reduction is applied to the KGrid output but only used for clustering in the ensemble framework, the output of KGrid is used for regression

Under this approach we have $3 + K$ different representations of the data set: D^R/D^S, D^U, D^V and D^i for $i = 1...K$ (where K is the number of clusters in the ensemble framework).

The system setup process is presented in algorithm listing 5.5. The process of assessing a previously unseen input is listed in algorithm listing 5.6. Advantages of this approach:

- The ensemble framework must only operate on a lower-dimensionality version of the data for clustering, saving computational effort.
- D^V can be generated such that it is effective for probabilistic clustering (i.e. its potential for regression is unimportant). For example, D^V could be generated via ISOmap, giving the human designer a readily available human-interpretable view of the design space.

Disadvantages of this approach:

- Requires that each unseen input be converted into representation x^V, which could be costly.
- Assessment of an unseen input requires $K + 1$ applications of dimensionality reduction.
- Requires the suggested framework to be modified to accommodate D^V *and* D^U.

Algorithm 5.6 Option three assessment of a previously unseen input.

1. Given unseen input x.
2. Generate the KGrid representation x^U.
3. Generate a low-dimensional representation of x^U, x^V.
4. Output the estimate $\hat{f}_{ens}(\mathrm{x}^U, \mathrm{x}^V)$.

Option four: two dimensionality reduction functions are applied to the KGrid output, one for the clustering in the framework, and the other for regression

Under this approach we have $4 + K$ different representations of the data set: D^R/D^S, D^U, D^V, D^W and D^i for $i = 1...K$ (where K is the number of clusters in the ensemble framework).

The system setup process is presented in algorithm listing 5.7. The process of assessing a previously unseen input is listed in algorithm listing 5.8. Advantages of this approach:

- The ensemble framework must only operate on a lower-dimensionality version of the data, for both clustering and per-cluster dimensionality reduction, saving computational effort.
- D^V can be generated such that it is effective for probabilistic clustering (i.e. its potential for regression is unimportant). For example, D^V could be generated via ISOmap, giving the human designer a readily available human-interpretable view of the design space and also being optimal for clustering, but *may not* be the best choice for learning regression functions from.
- D^W can be generated such that it is effective for regression (i.e. its potential for probabilistic clustering is unimportant). For example, D^W could be generated via PCA, which acts as a good form of compression but is not human-readable and *may not* be the best choice for clustering the data set).

Disadvantages of this approach:

- Requires that each unseen input be converted into representation x^V and x^W, which could be costly.
- Assessment of an unseen input requires $K + 2$ applications of dimensionality reduction.
- Requires the suggested framework to be modified to accommodate D^V *and* D^W.

Algorithm 5.7 Option 4 system initialisation phase.

1. Given the training data D consisting of N samples, suggested framework parameters (K, and regressor parameters), and the KGrid parameter τ.
2. Setup KGrid with the given τ value, taking the min and max in each axis from analysis of the data set (plus the recommended ε padding).
3. Use the KGrid to generate data set D^U of standardised K-Grid representations.
4. Perform dimensionality reduction on D^U giving D^V (a representation specifically for clustering).
5. Perform dimensionality reduction on D^U giving D^W(a representation specifically for pre-regression dimensionality reduction).
6. Train a suggested framework $\hat{f}_{ens}$ ensemble using D^V **and** D^W, using D^V for clustering, D^W as a pre-regression representation, with the given ensemble framework parameters.

Algorithm 5.8 Option 4 assessment of a previously unseen input.

1. Given unseen input x.
2. Generate the KGrid representation x^U.
3. Generate a low-dimensional representation of x^U, x^V.
4. Generate a low-dimensional representation of x^U, x^W.
5. Output the estimate $\hat{f}_{ens}(\mathrm{x}^W, \mathrm{x}^V)$.

5.3 Empirical analysis of the suggested systems

As there does not currently exist a single system for dealing with this problem we cannot make comparisons against such a system. Instead, we can empirically test the four suggested frameworks above and make comparisons between them. Although the automobile data set D^R is most relevant for these tests, we also test the options 2 and 4 using the spheres data set D^S to see if the ISOmap continues to outperform PCA for the bump problem (we omit option 1 and 3 as D^S is a higher-dimensional data set and so it is computationally prohibitive to test any approach without dimensionality reduction - see paragraph two of summary subsection 5.3.5).

For each experiment we used a very similar setup to previous experiments in this thesis:

- As in chapter 3, KGrid was setup with $\tau = 30$ and $\tau = 130$ for D^R and D^S experiments respectively.
- Probabilistic clustering was again chosen for the suggested ensemble framework.
  - The number of clusters K was set to 8 for D^R, and 3 for D^s.
- MLP was chosen as the base regressor for each cluster in the ensemble framework, with the same parameters as in previous chapters (and for the same reasons):
  - Three hidden neurons per MLP.
  - Back-propogation plus was again used for training the MLP weights.
- As in chapter 3, the D^R data set was divided into 600 testing and 600 training data and for the same reasons.
  - 150 training samples taken at random and without replacement to form a validation set used for testing convergence.
- Similarly the D^S data set was split into 225 testing and 225 training data.
  - 56 training samples taken at random and without replacement to form a validation set used for testing convergence.
- For the same reasons given in chapter 3, 1000 runs were always performed and results were tested for significance using the non-parametric Wilcoxon signed rank statistical test.

g_2^i	MSE	r^2	PCE	PPC
PCA	0.333	0.677	1.202	0.832
ISOmap	0.449	0.563	1.627	0.784

Table 5.1: Empirical results for the system without dimensionality reduction (option 1) using the automobile data set.

- The performance of a regression function was measured according to the scores MSE, r^2, PCE and PPC as described in section 2.6.

- For the per-cluster dimensionality reduction functions, PCA again chooses the number of components based on the sum of the eigenvalues (i.e. selecting the first $\bar{d}$ eigenvalues which sum up to >99% of the entire eigenvalue sum) for the same reasons as given in section 4.4.2.

- For all uses of ISOmap with D^R, $k = 5$ for the same reasons as those given in chapter 3. Similarly, when used with D^S, $k = 2$ was used. ISOmap's shortest-path graph was found incorporating the same cluster bridging technique as that was used in chapter 3's experiments.

5.3.1 Option one experiment (no dimensionality reduction)

The first approach is the simplest to set up and requires no modification of the ensemble framework. We test each method, ISOmap and PCA, for the role of g_2^i using only the automobile data set D^R (for reasons given in the introductory paragraph of this section).

The average scores over 1000 runs, shown in table 5.1, represent the worst scores that either of the approaches produces. The distribution of these results have been compared with the other approaches using the non-parametric Wilcoxon Signed-Rank test, producing p-values of $p < 0.0001$.

We note that PCA outperforms ISOmap as the dimensionality reduction method within each cluster of the ensemble framework. This holds for all the comparisons we make in this chapter, again with p-values of $p < 0.0001$ according to the non-parametric Wilcoxon Signed-Rank test. As mentioned in chapter 3, where this is also the case, this is most likely due to D (and thus D^R) not containing the sort of variations for which we imagine ISOmap would outperform PCA.

5.3.2 Option two experiment (with dimensionality reduction)

The second approach is also quite simple to set up and again requires no modification of the suggested ensemble framework. It does, though, require the selection of some dimensionality reduction, g_1 (which we test with each method, PCA and ISOmap). We set the reduced dimensionality of g_1's output to be 50, which should be large enough to form a complete representation, but small enough that the compactness is beneficial to machine learning processes. We will test both D^R and the spheres data set D^S.

In table 5.2 we can see the average scores over 1000 runs for this approach using D^R. Using the non-parametric Wilcoxon Signed-Rank test, we conclude (to $p < 0.0001$) that the extra layer of dimensionality reduction between KGrid and the ensemble framework improves accuracy substantially against the option one framework. Similarly (and also to $p < 0.0001$) we can conclude that PCA is also a better option for g_1 in all cases.

Results generated using the spheres data set D^S are presented in table 5.3. The differences between each pair of rows are significant to $p < 0001$. The first thing we notice is that all three accuracy scores are very poor, while the ranking metric, PPC, is fairly good. This consistently happens throughout this section, to varying degrees, and appears to be an artefact of the data set: each cluster contains samples that can be ranked linearly (depending on the location of the bump). Now, the training and validation sets are chosen with uniform distribution, but they may still sample poorly at the two extremes of the bumps position (in which case samples from the two extremes will be found in the testing set instead). The result is 'extreme' numbers being returned by the regression functions at these ('out of bounds') points. Despite this, these extreme numbers are usually in the correct 'direction' (i.e. the most extreme samples in the training set extrapolate in the correct direction) and so the PPC score doesn't suffer badly.

Given that this data set contains the bump problem, we see that the best results are found where ISOmap has been used to generate the low-dimensional representation of the data set. We further see that use of PCA for each expert regressor in the ensemble framework, g_2^i, results in better performance (regardless of what approach is chosen for g_1). These results are expected: where PCA is used for g_1 the representation is too poor for the ensemble framework to use effectively, while ISOmap results in a useful representation. Within the framework, despite performing well, ISOmap loses out to PCA for the job of g_2^i. This is because in the case where

g_1	g_2^i	MSE	r^2	PCE	PPC
PCA	PCA	0.124	0.877	0.451	0.9
	ISOmap	0.187	0.819	0.676	0.871
ISOmap	PCA	0.164	0.837	0.592	0.877
	ISOmap	0.195	0.81	0.704	0.865

Table 5.2: Empirical results for the system with dimensionality reduction (option 2) using the automobile data set.

g_1	g_2^i	MSE	r^2	PCE	PPC
PCA	PCA	5.208	0.249	35.28	0.788
	ISOmap	6.144	0.255	41.43	0.724
ISOmap	PCA	3.312	0.731	22.34	0.844
	ISOmap	1.271	0.501	8.587	0.801

Table 5.3: Empirical results for the system with dimensionality reduction (option 2) using the spheres data set.

PCA was used for g_1 the data is too poor for ISOmap to work with, and where ISOmap has been used for g_1, PCA managed to leverage its own advantages upon those of ISOmap: a second layer of ISOmap did not add anything to the data in comparison.

5.3.3 Option three experiment (with dimensionality reduction for clustering only)

The third approach requires some slight modification of the ensemble framework. Under the third approach, g_1 is used to generate a representation of D^U, D^V designed specifically for clustering. Then each cluster's g_2^i is applied to D^U. We set the reduced dimensionality of g_1's output to be 20 in all cases under the understanding that less information is needed to successfully cluster this data. This time only D^R will be tested for the reasons given in this sections introductory paragraph.

In table 5.4 we see that where PCA has been used as g_1, the performance has dropped slightly against the results in table 5.2 (although is still substantially better than the results in table 5.1). The p-value for these comparisons were all $p < 0.0001$.

On the other hand, where ISOmap has been used as g_1, the average appears to have improved. Indeed, where PCA is then used as g_2^i , the improvement over option two is shown to be statistically significant to $p < 0.0001$, while where ISOmap was then used as g_2^i , the significance is shown to be $p < 0.01$. This could be attributed to the ISOmap representation being easier to cluster (recall figures 3.27 and 3.28 in chapter 3 which showed the clusters being more easily

g_1	g_2^i	MSE	r^2	PCE	PPC
PCA	PCA	0.155	0.848	0.561	0.883
	ISOmap	0.201	0.803	0.727	0.864
ISOmap	PCA	0.16	0.844	0.579	0.881
	ISOmap	0.201	0.802	0.727	0.864

Table 5.4: Empirical results for the system with dimensionality reduction for clustering only (option 3) using the automobile data set.

distinguished under ISOmap than PCA).

5.3.4 Option four experiment (two separate dimensionality reductions)

The fourth approach requires some slight modification of the ensemble framework. Under the fourth approach, g_1 is used to generate a representation of D^U, D^V designed specifically for clustering. Next, a representation of D^U, D^W, is generated via $g_{1'}$ specifically for regression. Each cluster's g_2^i is applied to D^U while the training and estimation of performance is applied to D^W. The dimensionality of the representation output by g_1 is set to 20 (again, this should be enough for clustering). The dimensionality of the representation output by $g_{1'}$ is set to 100 (which should be large enough to form a complete representation, but small enough that the compactness is beneficial to machine learning processes).

The results for this are in table 5.5. Looking at all cases where g_2^i is performed using PCA, the figures are very similar whichever approach was used for g_1 with a small improvement where PCA was used (statistical significance calculated as $p < 0.01$). On the other hand, the approach used for $g_{1'}$ played a stronger role in these cases (significance as $p < 0.0001$).

Then considering the results where g_2^i is performed using ISOmap, we see no statistically significant difference between g_1 implementations ($p = 0.2891$ and $p = 0.1074$ where $g_{1'}$ is PCA and ISOmap respectively). Seemingly PCA is definitely the better option for $g_{1'}$, with statistical significance of $p < 0.0001$ for all comparisons.

In table 5.6 we see the results for this experiments when using the spheres data set. As in table 5.3 we see that some poor accuracy scores, but in general, there is a marked improvement. Interestingly, we again see the best results where PCA and ISOmap have been used in conjunction, for g_2^i and $g_{1'}$ respectively. Using either PCA for the clustering dimensionality reduction, g_1, appears to result in a very small accuracy change, but with quite a large p-value of of 0.091. These results support the observations made in experiment 2, and further support the claim

g_1	$g_{1'}$	g_2^i	MSE	r^2	PCE	PPC
PCA	PCA	PCA	0.122	0.88	0.44	0.901
		ISOmap	0.184	0.82	0.667	0.871
	ISOmap	PCA	0.164	0.838	0.593	0.877
		ISOmap	0.233	0.808	0.842	0.865
ISOmap	PCA	PCA	0.122	0.879	0.444	0.9
		ISOmap	0.184	0.818	0.668	0.870
	ISOmap	PCA	0.165	0.837	0.599	0.877
		ISOmap	0.198	0.808	0.717	0.864

Table 5.5: Empirical results for the system with two dimensionality reductions (option 4) using the automobile data set.

g_1	$g_{1'}$	g_2^i	MSE	r^2	PCE	PPC
PCA	PCA	PCA	0.83	0.553	5.626	0.806
		ISOmap	2.838	0.423	19.51	0.764
	ISOmap	PCA	0.116	0.892	0.783	0.888
		ISOmap	3.707	0.774	25.43	0.863
ISOmap	PCA	PCA	0.321	0.7	2.178	0.804
		ISOmap	41.93	0.489	284.3	0.759
	ISOmap	PCA	0.126	0.886	0.851	0.888
		ISOmap	0.42	0.82	2.895	0.844

Table 5.6: Empirical results for the system with two dimensionality reductions (option 4) using the spheres data set.

that ISOmap is a good choice for tasks which exhibit the bump problem. It seems also, that in comparison to the results in table 5.3, the lower dimensionality of the clustering representation has had a positive effect for all cases.

5.3.5 Summary of findings

The best results are clearly in tables 5.5 and 5.6, involving the final system setup, and using differing strengths of PCA dimensionality reduction at each stage. Although at this point in time, it is hard to draw up any strong conclusion regarding the best dimensionality reduction techniques for each stage, as only two have been used. Despite this, there is now evidence to suggest that the best system combination method is the fourth method, with different dimensionality reduction techniques feeding into the clustering and the regression sub-systems of the ensemble framework.

An important observation to note, is that the 'training' of each PCA function took considerably longer (several hours) than it did for ISOmap (minutes), depending on the dimensionality of the data set. Although this is also dependent on the function used to calculate an inverse matrix, it is always likely that an ISOmap function can be fit faster than a PCA function. It was for this

reason that options 1 and 3 were not tested with D^S, as it had a very large dimensionality.

5.3.6 Evaluation of the suggested systems by the original aims

In section 5.2.1 we summarised the aims of a complete system, based on the aims in chapters 1 and 2. Here are those aims repeated and numbered:

1. Must generalise well to future previously unseen designs in varied and previously unseen representations, allowing their functional performance estimates to be output.

2. Must deal well with a very large representation for which it is hard to find a representation that balances compactness and completeness well.

3. Must generalise well based on a relatively small training data set taken non-uniformly from a large design space, despite having a very large representation.

4. Need not be particularly accurate in a single subspace of designs, but rather be fairly accurate throughout the space of possible designs, and highly accurate in multiple areas of interest from an aesthetic design perspective (determined from the training data set).

5. Must be able to take any surface design and quickly (seconds) calculate an estimate of that surface designs functional performance score.

6. Offers the human designer potential utility.

In this section we will evaluate the combined system by these aims.

The first aim is to do with the broad variety of representations that engineering surface design data can be in. This was a challenge dealt with by KGrid, and its incorporation into the combined system thus deals with this. This is further supported empirically by the demonstrated use of D^R and D^S in the experiments of this chapter.

The next aim is dealt with explicitly by the ensemble framework. Through the use of per-cluster dimensionality reduction, the ensemble framework attempts to find the best reduced representation for the subspace that each cluster exists in. This aim is also dealt with in the latter three system combination frameworks suggested in this chapter, where preliminary dimensionality reduction has been applied prior to the ensemble framework.

The third and fourth aims are also dealt with explicitly by the ensemble framework, which divides the non-uniformly sampled design space across expert regression functions. They are also not a problem for KGrid, which has been specifically designed to find unified representations from across a large space of possible designs. This is also supported empirically by the demonstrated use of D^R and D^S in the earlier experiments of this chapter.

The fifth aim is fulfilled by all the core components. Dimensionality reduction mapping functions, clustering and regression functions, once trained, compute very fast. The generation of a unified representation is the greatest chance for slow-down, and KGrid, being an approximated approach to voxelisation, has been designed to compute fast (with opportunities for speed up where e.g. spatial partitioning techniques are used).

The final aim, offering the human designer potential utility, is supported by the transparent nature of the ensemble framework. This allows a human designer to extract expert regressors in order to study a sub-space of the design space with ease. It also instils in the user a level of trust in the way the system works.

Further, some of the combined system frameworks suggested earlier in this chapter allow for multiple representations, and some of these might be useful to a human designer. For example, in a combined system framework where a regression-specific dimensionality reduction mapping function is needed, it may be reasonable to deploy ISOmap for this purpose. As shown in chapter 3, ISOmap generates low-dimensional attributes which may be more useful for the human user's analysis of feature effects.

We summarise therefore, that the combined system enabled by the novel solutions presented in this thesis, is indeed capable of solving the challenging task originally put forward in chapter 1 and 2.

5.4 Integrating the system into the design process

The system behind this thesis has been suggested for the overall task of guiding the design decisions made by a human designer. This is where the significance of the suggested system largely comes from and in this section we will consider different ways of integrating the proposed system into the design process. At this point in time, we do not know for sure if these approaches would be effective in the human-design process, although as described in section 2.3, the results

in [92] are encouraging.

5.4.1 On request assessment

In this first approach, the aesthetic designer working in CAD software (such as Maya) *requests* the estimated performance score of their design. The result can be displayed to them as part of a user-interface component.

This has the benefit of being very unintrusive but allowing the aesthetic designer to make assessments as-and-when they would like to. This is also an disadvantage, as it would be very easy for the aesthetic designer to ignore this functionality if they wanted to.

5.4.2 Auto refreshing assessment

Another alternative is to have the estimated functional performance permanently shown in the user interface. Each time the design is altered, the design can be processed and assessed by the functional performance estimation system in order to update this figure.

This has the benefit of being more obvious, and thus less likely to be a totally ignored facility to the aesthetic designer. It also requires no action on the part of the aesthetic designer before up-to-date information is provided.

It has the disadvantage that it depends heavily on the processing time required to generate an estimation. If it takes more than seconds it may be required to be performed by an interruptible thread so that the process can restart with the most up to date version of the surface design.

5.4.3 Embedding information into the visual surface representation

The final alternative we consider is the most intrusive, and potentially the most useful. This approach involves embedding attribute sensitivity information in the surface design representation of CAD software that the aesthetic designer is working in.

For example, with a design constructed of triangular faces, the individual vertices make up the modifiable attributes of the design. These are typically what the designer works with. Each vertex can be said to have some level of effect on the functional performance of the design. This can be positive, negative, or no-effect. Further, the strength of the effect (i.e. how strongly the vertex affects the performance within some distance of movement) can be placed on a linear scale (0...1).

The CAD software could use the functional performance estimation system proposed in this thesis to investigate the affect of individual vertices (or groups of vertices) on functional performance. It could then overlay this estimated effect information onto the vertices that constitute the design, for example colouring:

- Vertices with roughly no-effect green.
- Vertices with a strong positive effect blue.
- Vertices with a strong negative effect red.

Further, the strength of the effect could be represented in the intensity of the appropriate colour (although the above colour scheme is merely an aid to helping the reader understand the concept, more complex 'heat-map' ranges of colour may be more appropriate). The aesthetic designer is then able to make informed choices about which attributes they would like to modify, and by how much of a modification.

This can be considered 'optimal' in terms of the assistance it gives the designer. Encouraging certain design alterations and discouraging others. The effect of this could be much faster convergence to a design which suits engineering constraints and aesthetic constraints, as well as completely novel and superior designs.

The disadvantages though, are:

- The designer will find it hard to ignore this system and may find it too intrusive. An option to disable it, may lead to it being permanently disabled and thus a pointless exercise.
- The designer may feel too restricted and that their 'artistic freedoms' are being impinged upon and therefore feel unhappy in their work.
- The error in the functional performance estimation function may lead to designers optimistically arriving at a design they find extremely satisfying, but may turn out to be unsuitable when the true functional performance function is used. This may cause disappointment and frustration.

As mentioned in section 2.3, an implementation similar to this has been investigated in [92]. The authors found that the system was effective in guiding the human designer's design decisions for their simple shape-matching design task.

5.5 Conclusion

5.5.1 Summary

In this chapter we looked at how best to combine the novel solutions of previous chapters, KGrid and the ensemble framework, in order to solve the problem central to this thesis.

We looked at the aims that such a system should have and suggested four different system topologies for the completed system. The complete system takes as input a data set of surface designs in a form such as STL and produces a regression system suited to a problem which:

- Covers a large space of possible designs.
- Has available test data which is sampled non-uniformly and clustered.
- Features a performance function which is highly non-linear with regards to input attributes.

attaining a good level of generalisation and offering some utility to the human designer.

With no competing systems to test against, we tested each of the four systems empirically against one another. Comparing them, we drew preliminary conclusions that the most complex approach, with two total dimensionality reduction layers between KGrid and the ensemble framework, performs best as a combined system. We evaluated the resultant system by the original aims and concluded that our system was indeed able to meet the requirements set out originally.

We then suggested several methods of integrating the system into the human guided design process (i.e. design software) which could be used to inspire further work.

5.5.2 Novel contributions

The major contributions of this chapter include a demonstration that the systems developed in isolation in the previous chapters combine to form a regression function with an improved level of accuracy (better than or comparable to the 'ideal' set D^C being used with an 'off-the-shelf' machine learning algorithm), enabling the system to be used for its target purpose; suggestions of how dimensionality reduction functions may be used in between these systems to improve generalisation/accuracy; an demonstration of these four approaches allowing us to evaluate these options; and a set of suggestions as to how one may best feed back the estimations of functional design performance to the aesthetic surface designer to help inspire future work.

5.5.3 Future work

The empirical studies comparing the four combination approaches require considerably more investigation: firstly, in terms of the algorithms used (there may be some more obvious choices for a 'clustering specific dimensionality reduction approach', for example); and secondly in terms of parameters (i.e. the level of dimensionality reduction applied at each stage).

There is also a need to actually test a system like this in the aesthetic design process, which could be evaluated using a questionnaire approach. This could also be an opportunity to compare the suggested approaches of integration of the functional performance estimation in the design software.

Chapter 6

Conclusion

In this chapter we conclude the thesis and subsequently summarise the main novel contributions, highlighting the significance of each. Next, following the knowledge acquired in the course of this thesis, we summarise the answers to the guiding questions listed in section 1.1. We then discuss the limitations of the research presented. Finally, we suggest several further research directions to be taken with regards to estimating functional performance for use in, and integrating functional performance into, the aesthetic design process.

6.1 Conclusion of the thesis

This thesis began by asking what methods could be employed to achieve tighter coupling between human driven aesthetic optimisation and functional performance optimisation. Our suggestion was that some estimate of functional performance could be introduced into the aesthetic design process, giving the aesthetic designer feedback regarding the effects of their proposed design changes.

Inspired by surrogate assisted optimisation, we concluded that one option would be to use machine learning techniques to develop a computationally fast estimate of the functional performance in order for this to be integrated into the software of the aesthetic designer.

Realising that machine learning approaches must learn from a wealth of data, and that this data can be difficult to generate, we suggested the use of an historically accumulated data set of surface designs and their known performance scores. In engineering surface design, this meant that the data would likely consist of a set of diverse representations. We suggest that these could

be converted with ease into STL representations, but that these would also come with a level of diversity as STL is a flexible surface format.

In chapter 3 we looked at the task of generating a low dimensional representation of this historically accumulated data set consisting of diverse STL designs, suggesting the computationally efficient and voxelisation-inspired KGrid in the process and showing its potential to be used with ISOmap to create representations that deal effectively with the 'bump problem'.

In chapter 4 we then looked at regression techniques, in particular ensemble regression techniques, and we looked at the nuances of our problem domain in order to determine an ensemble framework which would be appropriate to our problem, while being of maximal utility to a human engineer. One of the hallmark nuances of our problem was that the highly non-linear nature of functional performance functions (such as aerodynamic drag) means that over a broad space of possible designs, the optimal design representation at each location can vary substantially. Similarly in different regions of the design space, aesthetic designers will choose to modify different sets of attributes with greater frequency. These two truths lead us to create an ensemble framework consisting of many design representations.

Finally, in chapter 5, with no existing system to compare against, we compared several ways of combining the novel solutions from chapters 3 and 4 to create a complete system. We found that the most complex method, incorporating multiple dimensionality reduction functions between KGrid and the ensemble framework performed best. We finished by suggesting ways that this complete system could then be actually integrated into the aesthetic design software.

6.2 A Summary of novel contributions presented and their significance

1. **The problem.** In chapter 2 a particular problem, not yet given particular attention in the literature, was described and given focus with its real world embodiment and significance given. In chapter 3 we also demonstrated a method of generating experimental data exhibiting the properties of the problem. The problem is significant especially because:

    - The decoupling of aesthetic and functional design optimisation is a known problem at the Honda Research Institute and has recently become an active research area.

- It has been shown on a toy problem in [92] that information can substantially improve human-driven design convergence.
- This is a system that has been requested by designers.

2. **A novel representation conversion approach suitable for manifold-learning approaches of dimensionality reduction.** In chapter 3 we discussed representation conversion of a data set of engineering surface designs (in a format such as STL). We also described the bump-problem and suggested that manifold learning techniques such as ISOmap are better at solving these problems than linear dimensionality reduction techniques. To this end we sought a representation extraction method suitable for a data set of diverse STL designs, computationally fast, and also suitable for performing manifold learning upon. We suggested KGrid for this purpose and showed it's application to the experimental data sets. We also saw that an ISOmap representation would be more useful to a human engineer than a representation generated via PCA. This chapter's work is particularly significant for several reasons:

   - STL designs are a common format and any 3D surface design can be typically converted into an STL representation. This means that it is feasible for any engineering firm to convert all their historical designs into STL format.
   - There exist several approaches of representation conversion which are suitable for particular cases. For the case where a broad variety of design surfaces are in the data set, only voxelisation seems appropriate and reliable. Voxelisation can be slow to compute though, which is why KGrid, an approximation, trades accuracy for speed in arriving at a representation similar to voxelisation, and potentially also be more suitable for ISOmap (where the resolution of the KGrid matches that of the voxelisation).
   - Manifold learning techniques such as ISOmap are an effective way of dealing with the 'bump problem'. Unfortunately, ISOmap tends to show difficultly in extracting the spatial information from a data set containing raw attributes (such as the coordinate values of the points on an STL surface). It's application towards pixel data is show to be strong, and the same can be seen in its application to voxel-like KGrid representations.

3. **Ensemble of expert regression functions with expert-specific representations.** In chapter 4 we looked at the task of generating a regressor with good generalisation across a large design space where the optimal design representation may be totally different between two different sub-spaces of the design space. We suggested an ensemble framework of expert regression functions for spatially separate clusters where each expert regression function generated its own dimensionality reduction function, seeking the optimal design representation in its particular subspace of the design space. This chapter's work is particularly significant for the following reasons:

    - Typically regression functions need only be specific in a small design space region (such is the case with surrogate assisted design optimisation). In contrast, our regression function aims to be particularly accurate in a number of design space regions, with these regions having been learned from the data set.
    - Large and highly non-linear physics-based functions of surface designs often exhibit input attributes which affect the output strongly in one region of the design space, and hardly at all in other regions. The suggested framework deals with this by using multiple spatial-specific low-dimensional design representations.
    - Aesthetic designers do not modify attributes uniformly across all regions of the design space. Yet again, the suggested framework deals with this by using multiple spatial-specific low-dimensional design representations.
    - The vast majority of ensemble regression functions operate as a mysterious black-box system, taking input and giving output. This suffices well for automated design and optimisation, but in our case, where a human is involved, it beneficial to add in some transparency, clearing up the mystery so that (i) an engineer can extract meaningful information and (ii) more trust can be placed in how the ensemble is working when a human is using the system for guidance.

4. **A complete framework for generating a functional performance estimation function based on a non-uniformly sampled data set of diverse designs in varying representations.** In chapter 5 we demonstrated how the two technologies from chapters 3 and 4 may be combined in order to take a varied data set (such as in STL format) and

generate from it a function which can take any surface design and output an estimation of its performance. Four approaches were given and each was demonstrated empirically. Also provided were several suggestions for including this system in the surface design process. The significance of this work is made greater by the fact that currently no system such as this is in use at Honda (and likely other automobile design companies) because no system can be made that couples the necessary level of accuracy (i.e. a level upon which the designer can put enough trust in order to guide their design process) with the required input variations (both representation and range of designs).

6.3 Answers to the guiding questions

1. **Question:** *How can we generate estimations of functional performance for use in the aesthetic design process at an engineering firm, with minimal computational cost?* **Answer summary:** In chapter 5 we showed a complete system for this. Assuming a data set of accumulated and historically interesting engineering surface designs, we can process these using KGrid and some dimensionality reduction to acquire our training data set. Then these can be used with the suggested ensemble framework in order to learn the relationship between input and output across a large space of possible designs, using multiple dimensionality-reduction mapping functions to create the most compact representation in each subspace of the design space.

2. **Question:** *What problems are encountered when attempting to apply an off-the-shelf machine learning algorithm to the problem of estimating functional performance of engineering surface designs for use in guiding the aesthetic design process?* **Answer summary:** In chapter 2 we discussed this topic. The largest problems were: acquiring a training data set (and the steps that might be required as part of that); ensuring that this data set featured relevant samples; machine learning from a small data set with large dimensionality; and learning for a large and highly-nonlinear problem with non-uniform sampling.

3. **Question:** *How do we acquire a strong data set of training samples for use in this machine learning task, containing relevant knowledge for guiding the aesthetic design process?* **Answer summary:** Following chapter 2's suggestion, chapter 3 began with the assumption

that an accumulated historical data set of previously interesting designs is available. We then needed to preprocess this data set containing varied engineering surface design representations, and apply dimensionality reduction to it, in such a way that it would be useful to the machine learning task. We suggest that the preprocessing could be performed using the novel KGrid approach, which can be summarised as a fast computing approximation of voxelisation. We then showed that PCA worked well for reducing the dimensionality of the KGrid-format automobile data set, while also observing that using ISOmap as dimensionality reduction approach is superior to PCA for problems exhibiting the bump-problem, or where human interpretation of features is important.

4. **Question:** *How do we generate a regression function with good generalisation for a highly-non-linear and non-uniformly sampled problem with multiple sub-spaces of interest?* **Answer summary:** Good generalisation can often be associated with the use of ensemble regressors. We note that the unique properties of this problem specifically indicate that regression functions should find the task of machine learning for sub-spaces of the design space easier than for the whole space. Also, dimensionality reduction approaches should prove to be more powerful when restricted to sub-spaces. From these observations we have suggested a novel ensemble framework in chapter 4 which creates expert regression functions for sub-spaces and also sub-space specific dimensionality reduction functions. The latter reduces the effects of the curse of dimensionality in each sub-space (creating the most compact representation for that sub-space), while the use of the expert regressors makes learning a highly non-linear functional performance function easier.

5. **Question:** *How can we provide a system in which human designers can place trust and extract the most utility?* **Answer summary:** In chapters 3 through 5 this was a recurrent theme. The most important suggestions that answer this question are (for each of those chapters respectively): (i) using KGrid with a dimensionality reduction technique such as ISOmap in order to create a representation that is human interpretable; (ii) using the novel ensemble framework, the insides of which are modular and easy for the human user to understand. The expert regressors can be extracted and analysed in isolation for further utility; and (iii) using multiple representations in the overall system, e.g. for clustering, for regression, and for human interpretation.

6.4 Research limitations

The most striking limitation is that we have not been able to test the system in its intended setting: as a guide for aesthetic design processes in engineering surface design tasks. This was due to various constraints such as a lack of access to a suitable data set and aesthetic designer subjects. Future research should address this.

A similar limitation was that a real historical data set of automobile designs taken from an engineering firm's records was not available. We thus had to use some procedures based on random number generation in order to generate an experimental data set of automobile designs which exhibited a number of properties that we expect in the real data set. This research would gain significance if it could have been tested on a real data set of this nature.

Another striking limitation was that our realistic data set of automobile designs did not feature the 'bump problem', which caused us to rely on an artificial data set that did feature this problem. It would have been better to use a single automobile data set that featured the bump problem.

Pilot parameter studies were performed in order to generate suitable parameters for the experiments in this thesis. Although sufficient for conclusions drawn in this thesis, there have been no extensive parameter studies presented and these should be investigated and given in future research.

A similar limitation is that the experimental comparisons and evaluations in this thesis merely go to show advantages over other frameworks, but do not show what approaches work optimally within these frameworks (this is most evident in chapter 5 where only two different dimensionality reduction approaches are tested). More exhaustive research using different technologies for dimensionality reduction, regression and clustering could have enhanced the significance of the novel contributions presented in this thesis.

This research focused on the design process at a single firm, and did not investigate structured design methods used at other firms. Investigation of this may uncover additional significance for the research presented and suggest new ways in which the research can be integrated into the design process.

An interesting limitation to consider, is that throughout this thesis, the only data source considered was historical data sets of design surfaces and their functional performance scores. Of

course, many firms in engineering design domains also have design guidelines and rules at hand. These could have been used to guide the generation of the experimental data set, or have even been worked into the functional performance estimation process.

Finally, the design space limits for the novel approaches presented in this thesis are intentionally very loose. There is thus scope to apply these approaches to problems outside of the automobile design domain. A demonstration of this would have enhanced the significance of this work.

6.5 Further research directions

Ideas for further research have been presented at the end of each of chapters 3, 4 and 5. From a more general viewpoint though, we consider the following research ideas:

We imagine that the system will be trained with data and then be permanently available for use by the designer(s). This means the system may be idle for a lot of time and so we should consider strategies for constantly improving the system's accuracy. This is closely related to the field of active learning (that is, learning after the initial training phase has been performed). The key to this will be sampling in such a way that we are effectively preempting the future design choices of the human designers while also taking into account regions where our estimation function can be said to be weak.

This leads naturally onto the idea of predicting human design decisions, which could be large research topic in itself. Two levels of prediction may exist: (1) future design trends in the automobile industry; (2) favourite design choices of a particular artist. In each case, this is a data mining task, analysing the choices that a designer makes. The first case can be perceived as an extrapolation of the choices made by the total set of designers, although it may also emerge that some designers carry a larger weight than others. This is currently a very much under-researched topic.

Another simple idea is embellishing the input surface design data with more than just position values. It might be a good idea to incorporate other surface data such as curvature and thickness in order to improve the estimation function's performance.

Finally, there exists a lot of work to be done in assessing the impact of such a system on the human driven aesthetic design process. It is entirely possible that the system is of limited use as

it restricts the creativity of the designer. But it may also be true that the system helps inspire creativity, such as has been seen in surrogate assisted design optimisation.

References

[1] Erdem Acar and Masoud Rais-Rohani. Ensemble of metamodels with optimized weight factors. *Structural and Multidisciplinary Optimization*, 37(3):279–294, 2009.

[2] European Space Agency. Concurrent design facility, 2014. [Online; accessed 15-October-2014].

[3] Mohamed N. Ahmed, Sameh M. Yamany, Nevin Mohamed, Aly A. Farag, and Thomas Moriarty. A modified fuzzy c-means algorithm for bias field estimation and segmentation of mri data. *IEEE Transactions on Medical Imaging*, 21(3):193–199, 2002.

[4] Maryam Alavi and Dorothy E Leidner. Review: Knowledge management and knowledge management systems: Conceptual foundations and research issues. *MIS quarterly*, pages 107–136, 2001.

[5] Ethem Alpaydin. *Introduction to machine learning*. MIT press, 2004.

[6] Javed A Aslam, Raluca A Popa, and Ronald L Rivest. On estimating the size and confidence of a statistical audit. In *Proc. 2007 USENIX/ACCURATE Electronic Voting Technology Workshop (EVT'07)*, 2007.

[7] Ran Avnimelech and Nathan Intrator. Boosting regression estimators. *Neural computation*, 11(2):499–520, 1999.

[8] Thomas Back. *Evolutionary algorithms in theory and practice*. Oxford Univ. Press, 1996.

[9] Oliver Bandte and Sergey Malinchik. A broad and narrow approach to interactive evolutionary design–an aircraft design example. In *Genetic and Evolutionary Computation–GECCO 2004*, pages 883–895. Springer, 2004.

[10] David Baxter, James Gao, Keith Case, Jenny Harding, Bob Young, Sean Cochrane, and Shilpa Dani. An engineering design knowledge reuse methodology using process modelling. *Research in engineering design*, 18(1):37–48, 2007.

[11] David Baxter, Keith Goffin, and Marek Szwejczewski. Factors supporting knowledge integration in global innovation projects: An exploratory study. *Creativity and Innovation Management*, 22(4):408–419, 2013.

[12] Richard Bellman, Richard Ernest Bellman, Richard Ernest Bellman, and Richard Ernest Bellman. *Adaptive control processes: a guided tour*, volume 4. Princeton University Press Princeton, 1961.

[13] Christopher M Bishop et al. *Pattern recognition and machine learning*, volume 1. springer New York, 2006.

[14] Zdravko Bozakov, Lars Graening, Stephan Hasler, Heiko Wersing, and Stefan Menzel. Unsupervised extraction of design components for a 3d parts-based representation. In *Neural Networks, 2008. IJCNN 2008.(IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on*, pages 2009–2016. IEEE, 2008.

[15] Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

[16] Leo Breiman. Randomizing outputs to increase prediction accuracy. *Machine Learning*, 40(3):229–242, 2000.

[17] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[18] Leo Breiman, Jerome H Friedman, Richard A Olshen, and Charles J Stone. Classification and regression trees. wadsworth & brooks. *Monterey, CA*, 1984.

[19] Gavin Brown. Ensemble learning. In *Encyclopedia of Machine Learning*, pages 312–320. Springer, 2010.

[20] Marshall Burns. *Automated fabrication: improving productivity in manufacturing*. Prentice-Hall, Inc., 1993.

[21] Senthil K Chandrasegaran, Karthik Ramani, Ram D Sriram, Imré Horváth, Alain Bernard, Ramy F Harik, and Wei Gao. The evolution, challenges, and future of knowledge representation in product design systems. *Computer-aided design*, 45(2):204–228, 2013.

[22] Kazuhisa Chiba and Shigeru Obayashi. Data mining for multidisciplinary design space of regional-jet wing. *Journal of Aerospace Computing, Information, and Communication*, 4(11):1019–1036, 2007.

[23] Jacob Cohen. *Statistical power analysis for the behavioral sciences*. Academic press, 2013.

[24] Sabine Coquillart. *Extended free-form deformation: a sculpturing tool for 3D geometric modeling*, volume 24. ACM, 1990.

[25] Kimiz Dalkir. *Knowledge management in theory and practice*. Routledge, 2013.

[26] Ritendra Datta, Dhiraj Joshi, Jia Li, and James Z Wang. Studying aesthetics in photographic images using a computational approach. In *Computer Vision–ECCV 2006*, pages 288–301. Springer, 2006.

[27] Meenakshi Deshmukh, Volker Schaus, Philipp M Fischer, Dominik Quantius, Volker Maiwald, and Andreas Gerndt. Decision support tool for concurrent engineering in space mission design. In *Concurrent Engineering Approaches for Sustainable Product Development in a Multi-Disciplinary Environment*, pages 497–508. Springer, 2013.

[28] Pierre A Devijver and Josef Kittler. *Pattern recognition: A statistical approach*. Prentice-Hall London, 1982.

[29] Norman R Draper. Applied regression analysis bibliography update 1998-99. *Communications in Statistics-Theory and Methods*, 29(9-10):2313, 2000.

[30] Harris Drucker. Improving regressors using boosting techniques. In *ICML*, volume 97, pages 107–115, 1997.

[31] Imola K Fodor. *A survey of dimension reduction techniques*. Technical Report UCRL-ID-148494, Lawrence Livermore National Laboratory, 2002.

[32] LG Fonseca, HJC Barbosa, and ACC Lemonge. On similarity-based surrogate models for expensive single-and multi-objective evolutionary optimization. In *Computational Intelligence in Expensive Optimization Problems*, pages 219–248. Springer, 2010.

[33] Marzia Fontana, Franca Giannini, and Maria Meriana. Free form features for aesthetic design. *International Journal of Shape Modeling*, 06(02):273–302, 2000.

[34] Alexander Forrester, Andras Sobester, and Andy Keane. *Engineering design via surrogate modelling: a practical guide*. John Wiley & Sons, 2008.

[35] Alexander IJ Forrester, Neil W Bressloff, and Andy J Keane. Optimization using surrogate models and partially converged computational fluid dynamics simulations. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Science*, 462(2071):2177–2204, 2006.

[36] The OpenFOAM Foundation. *The OpenFOAM Foundation homepage*. [http://www.openfoam.org/; accessed 13-October-2013].

[37] Eibe Frank and Bernhard Pfahringer. Improving on bagging with input smearing. In *Advances in Knowledge Discovery and Data Mining*, pages 97–106. Springer, 2006.

[38] Yoav Freund and Robert E Schapire. A desicion-theoretic generalization of on-line learning and an application to boosting. In *Computational learning theory*, pages 23–37. Springer, 1995.

[39] Nicolás García-Pedrajas, César Hervás-Martínez, and Domingo Ortiz-Boyer. Cooperative coevolution of artificial neural network ensembles for pattern classification. *IEEE Transactions on Evolutionary Computation*, 9(3):271–302, 2005.

[40] JD Gergonne. The application of the method of least squares to the interpolation of sequences. *Historia Mathematica*, 1(4):439–447, 1974.

[41] Tushar Goel, Raphael T Haftka, Wei Shyy, and Nestor V Queipo. Ensemble of surrogates. *Structural and Multidisciplinary Optimization*, 33(3):199–216, 2007.

[42] L. Graening, S. Menzel, M. Hasenjäger, T. Bihrer, M. Olhofer, and B. Sendhoff. Knowledge extraction from aerodynamic design data and its application to 3d turbine blade geometries. *Journal of Mathematical Modelling and Algorithms*, 7(4):329–350, 2008.

[43] Lars Graening, Markus Olhofer, and Bernhard Sendhoff. Interaction detection in aerodynamic design data. In *Intelligent Data Engineering and Automated Learning-IDEAL 2009*, pages 160–167. Springer, 2009.

[44] Lars Graening and Bernhard Sendhoff. Shape mining: A holistic data mining approach for engineering design. *Advanced Engineering Informatics*, 28(2):166–185, 2014.

[45] Lars Gräning, Yaochu Jin, and Bernhard Sendhoff. Individual-based management of meta-models for evolutionary optimization with application to three-dimensional blade optimization. In *Evolutionary Computation in Dynamic and Uncertain Environments*, pages 225–250. Springer, 2007.

[46] Jatinder ND Gupta and Sushil Kumar Sharma. *Creating knowledge based organizations*. Igi Global, 2004.

[47] K. Gurney. *An introduction to neural networks*. CRC, 1997.

[48] Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *The Journal of Machine Learning Research*, 3:1157–1182, 2003.

[49] Mark A Hall. *Correlation-based feature selection for machine learning*. PhD thesis, The University of Waikato, 1999.

[50] Sherif Hashem. Optimal linear combinations of neural networks. *Neural networks*, 10(4):599–614, 1997.

[51] Simon Haykin. *Neural Networks: a comprehensive foundation*. Prentice-Hall, 2004.

[52] Tommy Hinks, Hamish Carr, Linh Truong-Hong, and Debra F Laefer. Point cloud data conversion into solid models via point-based voxelization. *Journal of Surveying Engineering*, 2012.

[53] Tin Kam Ho. The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):832–844, 1998.

[54] Guang-Bin Huang, Qin-Yu Zhu, and Chee-Kheong Siew. Extreme learning machine: theory and applications. *Neurocomputing*, 70(1):489–501, 2006.

[55] Zhangjun Huang, Chengen Wang, Jian Chen, and Hong Tian. Optimal design of aeroengine turbine disc based on kriging surrogate models. *Computers & structures*, 89(1):27–37, 2011.

[56] James M Hughes, Daniel J Graham, and Daniel N Rockmore. Quantification of artistic style through sparse coding analysis in the drawings of Pieter Bruegel the Elder. *Proceedings of the National Academy of Sciences*, 107(4):1279–1283, 2010.

[57] M. Hüsken, Y. Jin, and B. Sendhoff. Structure optimization of neural networks for evolutionary design optimization. *Soft Computing*, 9(1):21–28, 2005.

[58] Md M Islam, Xin Yao, and Kazuyuki Murase. A constructive algorithm for training cooperative neural network ensembles. *IEEE Transactions on Neural Networks*, 14(4):820–834, 2003.

[59] Aleks Jakulin and Ivan Bratko. Testing the significance of attribute interactions. In *Proceedings of the twenty-first international conference on Machine learning*, page 52. ACM, 2004.

[60] Yaochu Jin. A comprehensive survey of fitness approximation in evolutionary computation. *Soft computing*, 9(1):3–12, 2005.

[61] Yaochu Jin. Surrogate-assisted evolutionary computation: Recent advances and future challenges. *Swarm and Evolutionary Computation*, 1(2):61–70, 2011.

[62] Yaochu Jin, Markus Olhofer, and Bernhard Sendhoff. A framework for evolutionary optimization with approximate fitness functions. *IEEE Transactions on Evolutionary Computation*, 6(5):481–494, 2002.

[63] Yaochu Jin and Bernhard Sendhoff. Reducing fitness evaluations using clustering techniques and neural network ensembles. In *Genetic and Evolutionary Computation–GECCO 2004*, pages 688–699. Springer, 2004.

[64] Yaochu Jin and Bernhard Sendhoff. A systems approach to evolutionary multiobjective structural optimization and beyond. *Computational Intelligence Magazine, IEEE*, 4(3):62–76, 2009.

[65] Ian T Jolliffe. *Principal component analysis*, volume 487. Springer-Verlag New York, 1986.

[66] Marios K. Karakasis and Jean-Antoine Desideri. Model reduction and adaption of optimum-shape design in aerodynamics by neural networks. Research Report RR-4503, 2002.

[67] Simon L Kendal and Malcolm Creen. *An introduction to knowledge engineering*. Springer, 2007.

[68] Manas S Khurana, Hadi Winarto, and Arvind K Sinha. Airfoil optimization by swarm algorithm with mutation and artificial neural networks. In *AIAA Aerospace Sciences Meeting Including The New Horizons Forum and Aerospace Exposition*, 2009.

[69] Scott Kirkpatrick. Optimization by simulated annealing: Quantitative studies. *Journal of statistical physics*, 34(5-6):975–986, 1984.

[70] Josef Kittler. Feature selection and extraction. *Handbook of pattern recognition and image processing*, pages 59–83, 1986.

[71] Ron Kohavi et al. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *IJCAI*, volume 14, pages 1137–1145, 1995.

[72] Anders Krogh, Jesper Vedelsby, et al. Neural network ensembles, cross validation, and active learning. *Advances in neural information processing systems*, pages 231–238, 1995.

[73] Andrew Kusiak. *Concurrent engineering: automation, tools, and techniques*. John Wiley & Sons, 1993.

[74] Nojun Kwak and Chong-Ho Choi. Improved mutual information feature selector for neural networks in supervised learning. In *International Joint Conference on Neural Networks, 1999. IJCNN'99.*, volume 2, pages 1313–1318. IEEE, 1999.

[75] J. Lawrence. *Introduction to neural networks*. California Scientific, Nevada City, CA 95959., 1993.

[76] John A Lee and Michel Verleysen. *Nonlinear dimensionality reduction*. Springer, 2007.

[77] Leifur Leifsson and Slawomir Koziel. Multi-fidelity design optimization of transonic airfoils using physics-based surrogate modeling and shape-preserving response prediction. *Journal of Computational Science*, 1(2):98–106, 2010.

[78] Congcong Li and Tsuhan Chen. Aesthetic visual quality assessment of paintings. *IEEE Journal of Selected Topics in Signal Processing*, 3(2):236–252, April 2009.

[79] D. Lim, Y. Jin, Y.S. Ong, and B. Sendhoff. Generalizing surrogate-assisted evolutionary computation. *IEEE Transactions on Evolutionary Computation*, 14(3):329–355, 2010.

[80] Dudy Lim, Yew-Soon Ong, Yaochu Jin, and Bernhard Sendhoff. A study on metamodeling techniques, ensembles, and multi-surrogates in evolutionary computation. In *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pages 1288–1295. ACM, 2007.

[81] Yong Liu and Xin Yao. Ensemble learning via negative correlation. *Neural Networks*, 12(10):1399–1404, 1999.

[82] Yong Liu, Xin Yao, and Tetsuya Higuchi. Evolutionary ensembles with negative correlation learning. *IEEE Transactions on Evolutionary Computation*, 4(4):380–387, 2000.

[83] Y-S Ma, Gang Chen, and Georg Thimm. Paradigm shift: unified and associative feature-based concurrent and collaborative engineering. *Journal of Intelligent Manufacturing*, 19(6):625–641, 2008.

[84] David JC MacKay. *Information theory, inference and learning algorithms*. Cambridge university press, 2003.

[85] Pradipta Maji. Mutual information-based supervised attribute clustering for microarray sample classification. *IEEE Transactions on Knowledge and Data Engineering*, 24(1):127–140, 2012.

[86] R Timothy Marler and Jasbir S Arora. Survey of multi-objective optimization methods for engineering. *Structural and multidisciplinary optimization*, 26(6):369–395, 2004.

[87] Saeed Masoudnia and Reza Ebrahimpour. Mixture of experts: a literature survey. *Artificial Intelligence Review*, 42(2):275–293, 2014.

[88] Jon McCormack. Facing the future: Evolutionary possibilities for human-machine creativity. In Juan Romero and Penousal Machado, editors, *The Art of Artificial Evolution*, Natural Computing Series, pages 417–451. Springer Berlin Heidelberg, 2008.

[89] Geoffrey McLachlan and David Peel. *Finite mixture models*. John Wiley & Sons, 2004.

[90] Bahram Meihami and Hussein Meihami. Knowledge management a way to gain a competitive advantage in firms (evidence of manufacturing companies). *International Letters of Social and Humanistic Sciences*, (03):80–91, 2014.

[91] João Mendes-Moreira, Carlos Soares, Alípio Mário Jorge, and Jorge Freire De Sousa. Ensemble approaches for regression: A survey. *ACM Computing Surveys (CSUR)*, 45(1):10, 2012.

[92] S Menzel, M Olhofer, B Sendhoff, et al. A metamodel-driven interactive framework for a designer assistance system. In *DS 60: Proceedings of DESIGN 2010, the 11th International Design Conference, Dubrovnik, Croatia*, 2010.

[93] Stefan Menzel and Bernhard Sendhoff. Representing the change-free form deformation for evolutionary design optimization. In *Evolutionary computation in practice*, pages 63–86. Springer, 2008.

[94] Kaisa Miettinen, Francisco Ruiz, and Andrzej P Wierzbicki. Introduction to multiobjective optimization: interactive approaches. In *Multiobjective Optimization*, pages 27–57. Springer, 2008.

[95] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of machine learning*. MIT press, 2012.

[96] Elizbar A Nadaraya. On estimating regression. *Theory of Probability & Its Applications*, 9(1):141–142, 1964.

[97] Shigeru Obayashi and Daisuke Sasaki. Visualization and data mining of pareto solutions using self-organizing map. In *Evolutionary multi-criterion optimization*, pages 796–809. Springer, 2003.

[98] Shigeru Obayashi and Takanori Tsukahara. Comparison of optimization algorithms for aerodynamic shape design. *AIAA journal*, 35(8):1413–1415, 1997.

[99] Yew S Ong, Prasanth B Nair, and Andrew J Keane. Evolutionary optimization of computationally expensive problems via surrogate modeling. *AIAA journal*, 41(4):687–696, 2003.

[100] Yew-Soon Ong, Kai Yew Lum, and Prasanth B Nair. Hybrid evolutionary algorithm with hermite radial basis function interpolants for computationally expensive adjoint solvers. *Computational Optimization and Applications*, 39(1):97–119, 2008.

[101] Yew Soon Ong, PB Nair, AJ Keane, and KW Wong. Surrogate-assisted evolutionary optimization frameworks for high-fidelity engineering design problems. In *Knowledge Incorporation in Evolutionary Computation*, pages 307–331. Springer, 2005.

[102] Yew-Soon Ong, Prasanth B Nair, and Kai Yew Lum. Max-min surrogate-assisted evolutionary algorithm for robust design. *IEEE Transactions on Evolutionary Computation*, 10(4):392–404, 2006.

[103] David W Opitz. Feature selection for ensembles. In *AAAI/IAAI*, pages 379–384, 1999.

[104] David W Opitz and Jude W Shavlik. Generating accurate and diverse members of a neural-network ensemble. *Advances in neural information processing systems*, pages 535–541, 1996.

[105] Feng Pan and Ping Zhu. Design optimisation of vehicle roof structures: benefits of using multiple surrogates. *International Journal of Crashworthiness*, 16(1):85–95, 2011.

[106] Bambang Parmanto, Paul W Munro, and Howard R Doyle. Reducing variance of committee prediction with resampling techniques. *Connection Science*, 8(3-4):405–426, 1996.

[107] Sandeep Patil and B Ravi. Voxel-based representation, display and thickness analysis of intricate shapes. In *Computer Aided Design and Computer Graphics, 2005. Ninth International Conference on*, pages 6–pp. IEEE, 2005.

[108] Michael P Perrone and Leon N Cooper. When networks disagree: Ensemble methods for hybrid neural networks. Technical report, DTIC Document, 1992.

[109] Anna Persson, Henrik Grimm, and Amos Ng. Metamodel-assisted global search using a probing technique. In *The IAENG International Conference on Artificial Intelligence and Applications (ICAIA'07)*, pages 83–88, 2007.

[110] Pavel Pudil and Jana Novovičová. Novel methods for feature subset selection with respect to problem knowledge. In *Feature Extraction, Construction and Selection*, pages 101–116. Springer, 1998.

[111] Wen Quan and He Jianmin. A study on collaborative mechanism for product design in distributed concurrent engineering. In *Computer-Aided Industrial Design and Conceptual Design, 2006. CAIDCD'06. 7th International Conference on*, pages 1–5. IEEE, 2006.

[112] Nestor V Queipo, Raphael T Haftka, Wei Shyy, Tushar Goel, Rajkumar Vaidyanathan, and P Kevin Tucker. Surrogate-based analysis and optimization. *Progress in aerospace sciences*, 41(1):1–28, 2005.

[113] Khaled Rasheed, Xiao Ni, and Swaroop Vattam. Comparison of methods for developing dynamic reduced models for design optimization. *Soft Computing*, 9(1):29–37, 2005.

[114] Khaled Rasheed, Swaroop Vattam, and Xiao Ni. Comparison of methods for using reduced models to speed up design optimization. In *GECCO*, pages 1180–1187, 2002.

[115] Juan José Rodriguez, Ludmila I Kuncheva, and Carlos J Alonso. Rotation forest: A new classifier ensemble method. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(10):1619–1630, 2006.

[116] LE Roscoe et al. Stereolithography interface specification. *America-3D Systems Inc*, 1988.

[117] Bruce E Rosen. Ensemble learning using decorrelated neural networks. *Connection Science*, 8(3-4):373–384, 1996.

[118] Sam T Roweis and Lawrence K Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000.

[119] Thomas Philip Runarsson. Constrained evolutionary optimization by approximate ranking and surrogate models. In *Parallel Problem Solving from Nature-PPSN VIII*, pages 401–410. Springer, 2004.

[120] P Sainter, K Oldham, A Larkin, A Murton, and R Brimble. Product knowledge management within knowledge-based engineering systems. In *Design Engineering Technical Conference, Baltimore, Setembro*, 2000.

[121] Robert E Schapire. The strength of weak learnability. *Machine learning*, 5(2):197–227, 1990.

[122] Alon Schclar and Lior Rokach. Random projection ensemble classifiers. In *Enterprise Information Systems*, pages 309–316. Springer, 2009.

[123] Michael D Schmidt and Hod Lipson. Coevolution of fitness predictors. *IEEE Transactions on Evolutionary Computation*, 12(6):736–749, 2008.

[124] Thomas W Sederberg and Scott R Parry. Free-form deformation of solid geometric models. In *ACM Siggraph Computer Graphics*, volume 20, pages 151–160. ACM, 1986.

[125] Lior Shamir and Jane A Tarakhovsky. Computer analysis of art. *Journal on Computing and Cultural Heritage (JOCCH)*, 5(2):7, 2012.

[126] Timothy W Simpson, JD Poplinski, Patrick N Koch, and Janet K Allen. Metamodels for computer-based engineering design: survey and recommendations. *Engineering with computers*, 17(2):129–150, 2001.

[127] Karthik Sindhya, Ana Belen Ruiz, and Kaisa Miettinen. A preference based interactive evolutionary algorithm for multi-objective optimization: Pie. In *Evolutionary Multi-Criterion Optimization*, pages 212–225. Springer, 2011.

[128] Lindsay I Smith. A tutorial on principal components analysis. *Cornell University, USA*, 51:52, 2002.

[129] Alex Smola and Vladimir Vapnik. Support vector regression machines. *Advances in neural information processing systems*, 9:155–161, 1997.

[130] James C Spall. *Introduction to stochastic search and optimization: estimation, simulation, and control*, volume 65. John Wiley & Sons, 2005.

[131] H.G. Sung. *Gaussian mixture regression and classification*. PhD thesis, RICE UNIVERSITY, 2004.

[132] Gabor J Szekely and Maria L Rizzo. Hierarchical clustering via joint between-within distances: Extending ward's minimum variance method. *Journal of classification*, 22(2):151–183, 2005.

[133] Hideyuki Takagi. Interactive evolutionary computation: Fusion of the capabilities of ec optimization and human evaluation. *Proceedings of the IEEE*, 89(9):1275–1296, 2001.

[134] Joshua B Tenenbaum, Vin De Silva, and John C Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

[135] Yoel Tenne, Kazuhiro Izui, and Shinji Nishiwaki. Dimensionality-reduction frameworks for computationally expensive problems. In *IEEE Congress on Evolutionary Computation (CEC) 2010*, pages 1–8. IEEE, 2010.

[136] Igor V Tetko, David J Livingstone, and Alexander I Luik. Neural network studies. 1. comparison of overfitting and overtraining. *Journal of chemical information and computer sciences*, 35(5):826–833, 1995.

[137] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

[138] G Gary Wang and S Shan. Review of metamodeling techniques in support of engineering design optimization. *Journal of Mechanical Design*, 129(4):370–380, 2007.

[139] Geoffrey S Watson. Smooth regression analysis. *Sankhyā: The Indian Journal of Statistics, Series A*, pages 359–372, 1964.

[140] Wikipedia. Knowledge-based engineering — wikipedia, the free encyclopedia, 2014. [Online; accessed 15-October-2014].

[141] Kevin Woods, W Philip Kegelmeyer Jr, and Kevin Bowyer. Combination of multiple classifiers using local accuracy estimates. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(4):405–410, 1997.

[142] Seniha Esen Yuksel, Joseph N Wilson, and Paul D Gader. Twenty years of mixture of experts. *IEEE Transactions on Neural Networks and Learning Systems*, 23(8):1177–1193, 2012.

[143] Gabriele Zenobi and Padraig Cunningham. Using diversity in preparing ensembles of classifiers based on different feature subsets to minimize generalization error. In *Machine Learning: ECML 2001*, pages 576–587. Springer, 2001.

[144] Chun-Xia Zhang, Jiang-She Zhang, and Guan-Wei Wang. An empirical study of using rotation forest to improve regressors. *Applied Mathematics and Computation*, 195(2):618–629, 2008.

[145] Zheng Zhao and Huan Liu. Searching for interacting features. In *IJCAI*, volume 7, pages 1156–1161, 2007.

[146] Zongzhao Zhou, Yew Soon Ong, Prasanth B Nair, Andy J Keane, and Kai Yew Lum. Combining global and local surrogate models to accelerate evolutionary optimization. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 37(1):66–76, 2007.