Aerodynamic Design Exploration through Surrogate-Assisted Illumination

Adam Gaier, Alexander Asteroth, Jean-Baptiste Mouret

Aerodynamic Design Exploration through Surrogate-Assisted Illumination

Adam Gaier[*]
CNRS / Université de Lorraine, Villers-lès-Nancy, 54600, France
Bonn-Rhein-Sieg University of Applied Sciences, Sankt Augustin, 53757, Germany

Alexander Asteroth[†]
Bonn-Rhein-Sieg University of Applied Sciences, Sankt Augustin, 53757, Germany

Jean-Baptiste Mouret[‡]
Inria Nancy – Grand Est
CNRS / Université de Lorraine, Villers-lès-Nancy, 54600, France

A new method for design space exploration and optimization, Surrogate-Assisted Illumination (SAIL), is presented. Inspired by robotics techniques designed to produce diverse repertoires of behaviors for use in damage recovery, SAIL produces diverse designs that vary according to features specified by the designer. By producing high-performing designs with varied combinations of user-defined features a map of the design space is created. This map *illuminates* the relationship between the chosen features and performance, and can aid designers in identifying promising design concepts. SAIL is designed for use with computationally expensive design problems, such as fluid or structural dynamics, and integrates approximative models and intelligent sampling of the objective function to minimize the number of function evaluations required. On a 2D airfoil optimization problem SAIL is shown to produce hundreds of diverse designs which perform competitively with those found by state-of-the-art black box optimization. Its capabilities are further illustrated in a more expensive 3D aerodynamic optimization task.

Nomenclature

$m()$	Mean function
$k()$	Covariance function
$GP()$	Gaussian Process model
K	Kernel matrix
$\mathbf{k}$	Covariance vector
D	Observation set (input/output pairs)
μ	Mean
σ^2	Variance
C_D	Coefficient of Drag
C_L	Coefficient of Lift
X_{up}	Position of highest point on along upper surface of airfoil
Z_{up}	Height of highest point on upper surface of airfoil

Subscripts

t	Total number of observations

[*]PhD Candidate, Université de Lorraine / Bonn-Rhein-Sieg University of Applied Sciences and AIAA Student Member.
[†]Professor, Bonn-Rhein-Sieg University of Applied Sciences, Sankt Augustin, 53757, Germany
[‡]Researcher. Inria Nancy - Grand Est, 615 rue du Jardin Botanique, 54600 Villers-les-Nancy, France

I. Introduction

Design optimization techniques are often thought of by their creators as the final step in the design process. Algorithms are developed to squeeze every last bit of performance out of a solution, and emphasis is placed on the refinement of designs to their most optimal form.[1] If, however, the goal is to support designers then this focus on optimality may very well be misplaced. In an interview study conducted by Autodesk[2] to better understand how professional designers, engineers, and architects use optimization tools they found that optimization was most commonly used not at the end of the design process, but at the beginning. Rather than using optimization algorithms to solve design problems, they were instead used to explore the range of possible solutions. By first generating a range of candidate solutions, designers familiarize themselves with the design space to better understand design concepts and trade-offs.

When faced with a high dimensional design space with many interacting parameters it can be difficult to form an intuition about which combinations of features will lead to high performance. In these cases optimization tools can be employed to produce a batch of design alternatives which best satisfy the objectives and constraints of the problem. These design alternatives act as concrete way points in high dimensional parameter space, helping designers understand various approaches to satisfy their objectives. These designs illustrate the trade offs in the design space, as well as the assumptions and consequences inherent to the problem definition and constraints. As constraints and objectives are reconsidered and adjusted, their effect is made clear by the designs which result from the optimization process. It is within the design regions found by this collaborative human-machine exploration that the designer works, refining designs to match precise requirements along with intangibles such as aesthetics.

The tool most commonly cited in these interviews to produce the variety of designs needed for design exploration was multi-objective optimization.[2] If the objectives being considered are in opposition to each other the result of the optimization process is a Pareto front of solutions, each representing a different trade-off.[3] It may, however, be the case that during this explorative process the designer's interest is not only in the maximization of performance, but in the effect of different design features on performance, along with the interaction of the features themselves. A better understanding of the problem space can then be gained through experimenting with the effect of different features, and this process can lead to more inspired designs than maximization of objectives alone.

To assist designers in this generative exploration process a new class of algorithms known as illumination algorithms, or quality-diversity (QD) algorithms,[4] could be applied. These algorithms use evolutionary approaches to produce a variety of high-performing solutions, rather than converging on a single optimum. One such algorithm, Multi-dimensional Archive of Phenotypic Elites (MAP-Elites)[5,6] explicitly explores the relationship between user-defined features and performance. Designers select a few features deemed interesting or important, such as weight or structural strength, and MAP-Elites produces high-performing solutions which span the possible variations of those features. Known as *illumination*, this process reveals the performance potential of the different regions of the feature space.

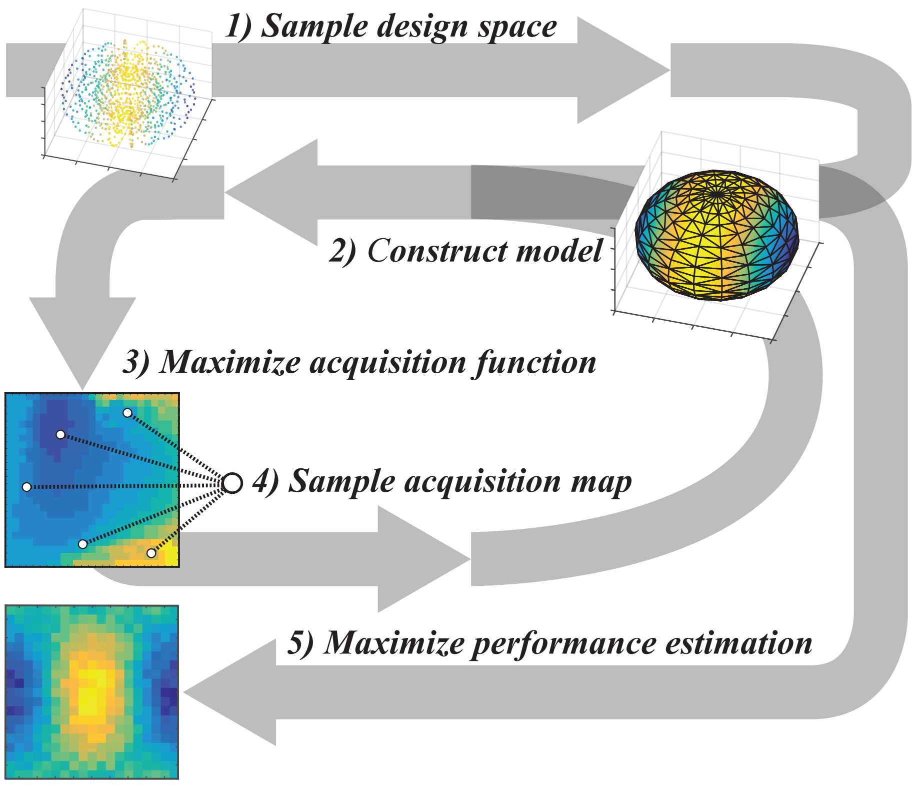
While MAP-Elites is effective at finding a variety of high-performing solutions, this feat is only accomplished at great computational cost. In its best known application MAP-Elites was used to build a repertoire of thousands of different hexapod robot controllers, which then allowed the walking robot to quickly recover from damage to its limbs and motors[5] . While this damage recovery could be achieved online by the robot after only trying a few alternate behaviors, the repertoire itself was produced offline and required twenty million walking simulations. In fluid dynamics applications, where a single simulation can take hours, application of such a computationally intensive algorithm is simply unrealistic.

It is common when faced with computationally expensive optimization problems to make use of approximate models of the objective function, or surrogate models. These models predict the performance of solutions that have not been evaluated, based on the known performance of the solutions which have.[7–9] Though it is unrealistic to model the entire design space, it is possible to model the high performing regions. Modeling is done by the careful sampling of solutions, coordinated by the optimization algorithm itself. By optimizing an *acquisition function* which balances exploring new areas of the design space and refining the model in regions already known to perform well, accuracy in high performing regions can be improved and optima found. These computationally efficient models can be used in place of the objective function during optimization, greatly accelerating the process. By incorporating these surrogate-assistance techniques into the evaluation-heavy illumination process, it is potentially possible to use MAP-Elites even in computationally expensive design problems.

In this paper we introduce the Surrogate-Assisted Illumination (SAIL) algorithm for design space exploration. We integrate surrogate-assistance techniques with the MAP-Elites algorithm, allowing it to be used in computationally expensive design problems while maintaining its original capabilities. Before demonstrating its use in a 3D aerodynamics problem we analyze the performance of the algorithm in a simpler 2D airfoil design case, showing that SAIL is:

- *Divergent* - Produces a diversity of solutions which vary across a user-defined continuum;
- *Accurate* - Predicts behavior of the objective function in high-performing regions;
- *High-Performing* - Produces near optimal solutions;
- *Efficient* - Performs under computational constraints.

Figure 1 describes the operation of the SAIL algorithm: 1) an initial set of observations is produced by sampling the parameter space and evaluating the resulting designs, 2) a surrogate model is produced from these parameter/performance pairs, 3) the MAP-Elites algorithm is used to maximize the acquisition function in every region of the feature space, producing an *acquisition map* of solutions that are most likely to improve the model in high performing regions, 4) a sample of solutions is drawn from the acquisition map, evaluated, and added to the set of observations. Steps 2-4 are repeated until the the computational budget is reached. The performance predictions of the model can then be used to produce a *prediction map* of the estimated optimal designs in each region of the feature space, illuminating the performance potential of each region of the feature space and the relationship between features.



1) Sample design space
2) Construct model
3) Maximize acquisition function
4) Sample acquisition map
5) Maximize performance estimation

Figure 1. ***Surrogate-Assisted Illumination (SAIL)***
1) **Sample design space to produce initial solutions.**
2) **Construct model of the objective function based on samples.**
3) **Maximize the acquisition function, which balances exploitation and exploration, in every region of the feature space, producing an *acquisition map*.**
4) **Draw the next samples to test on the objective function from the *acquisition map*. Repeat steps 2-4.**
5) **Maximize fitness as predicted by the resulting model, producing a *prediction map* of high-performing designs.**

II. Related Work

Quality Diversity and MAP-Elites

Quality diversity (QD) algorithms[4] use evolutionary methods to produce a set of diverse, high quality solutions within a single run. Rather than seeking a single global optimum, QD algorithms discover as many types of solutions to a problem as possible, and produce a best possible example of each type. For this reason they are also referred to as *illumination* algorithms, as they illuminate the performance potential of different regions of the solution space.

One such illumination algorithm, the MAP-Elites algorithm[5,6] is designed to produce high-performing solutions across a continuum of n user-defined feature dimensions. MAP-Elites first divides the feature space into a grid, or map, of n-dimensional bins. This map holds the set of solutions, with each bin holding a single solution. When the map is visualized, with each bin colored according to the performance of the solution it contains, it provides an intuitive overview of the performance potential of each region of the feature space.

To initialize MAP-Elites a set of random solutions are first evaluated and assigned to bins. The bin to which a solution is assigned is based on its features. If, for example, the feature space is 2D with one dimension for weight and another for cost, a low cost and low weight solution would be placed in the low cost, low weight bin location of the map. If the bin is empty, the solution is placed inside. If the bin is already occupied the two solutions are compared, and the one which performs better is placed in the bin while the other is discarded. Through this competitive process each bin contains the best solution found so far for each combination of features. These solutions are known as *elites.*

To produce new solutions, elites are chosen randomly, their parameters perturbed by adding Gaussian noise, and then evaluated and assigned a bin based on their features. These new solutions have two ways of being added to the map: discovering an unoccupied bin, or out-competing an existing solution for its bin. Repetition of this process produces an increasingly explored feature space and an increasingly optimal collection of solutions, *illuminating* the performance potential of the entire feature space. The MAP-Elites algorithm is summarized below in Algorithm 1.

Algorithm 1 MAP-Elites

```
1: function MAP-ELITES($objective\_function()$, $\mathcal{X}_{initial}$)
2:     $\mathcal{X} \leftarrow \emptyset$                                  ▷ empty map containing parameter vectors
3:     $\mathcal{P} \leftarrow \emptyset$                                  ▷ empty map containing performance values
4:     $\mathcal{X} \leftarrow \mathcal{X}_{initial}$, $\mathcal{P} \leftarrow objective\_function(\mathcal{X}_{initial})$
5:     for iter $= 1 \rightarrow I$ do
6:         $\mathbf{x} \leftarrow random\_selection(\mathcal{X})$
7:         $\mathbf{x}' \leftarrow random\_variation(\mathbf{x})$
8:         $\mathbf{b}' \leftarrow feature\_descriptor(\mathbf{x}')$
9:         $\mathbf{p}' \leftarrow objective\_function(\mathbf{x}')$
10:        if $\mathcal{P}(\mathbf{b}') = \emptyset$ or $\mathcal{P}(\mathbf{b}') < \mathbf{p}'$ then
11:            $\mathcal{P}(\mathbf{b}') \leftarrow \mathbf{p}'$, $\mathcal{X}(\mathbf{b}') \leftarrow \mathbf{x}'$
12:        end if
13:    end for
14:    return ($\mathcal{X}$, $\mathcal{P}$)                              ▷ Return illuminated map
15: end function
```

MAP-Elites has been shown to be effective in exploration and optimization in a variety of domains including the design of walking soft robot morphologies,[6] the generation of images that fool deep neural networks,[10] and the evolution of robot controllers capable of adapting to damage.[5]

Surrogate-Assisted Optimization

Evolutionary optimization approaches typically require a large number of evaluations before acceptable solutions are found. In many applications these performance calculations are far from trivial, and the computational cost of repeated evaluations becomes prohibitively expensive. In these cases approximate models of the objective function, or surrogate models, are used in their place. Surrogate-assisted optimization has been a particularly useful approach in the computationally demanding context of fluid dynamics.[11–14]

Due to the sheer number of evaluations required by MAP-Elites, even when evaluations are inexpensive surrogate-assistance has the potential to accelerate the illumination process dramatically.

Surrogate-assisted optimization often takes place within the wider framework of Bayesian optimization (BO).[5,9,15] BO approaches the problem of optimization not just as one of finding the most optimal solution, but of modeling the underlying objective function in high-performing regions. To estimate the objective function probabilistic models are used, giving each sample a predicted objective value (mean) and a confidence in that prediction (variance). New samples are chosen where the model predicts a high objective value and where prediction uncertainty is high. How the emphasis of predicted performance and prediction uncertainty is balanced is determined by an *acquisition function.* In the active learning process of BO, the sample which maximizes the acquisition function is chosen as the next observation, evaluated, the result included in the set of observations, the model updated, and the process repeated.

A variety of data-driven machine learning techniques such as polynomial regression, support vector machines, and artificial neural networks can be used to construct surrogate models.[7,8] However, for model-based methods that require probabilistic results, such as Kriging[16] and BO,[15] Gaussian process (GP) models[17] are typically used.

Gaussian Process Models

In the active learning context of surrogate-assisted optimization a measure of model uncertainty is essential, as this allows for the balancing of exploring unknown regions of the search space and exploiting regions known to contain high-performing solutions. GP[17] models are often used as for surrogate-assisted optimization as they are effective even when only a small number of samples are available and, crucially, their predictions include a measure of certainty.

Gaussian process models are a generalization of the Gaussian distribution: where a Gaussian distribution describes random variables, defined by mean and variance, a Gaussian process describes a random distribution of functions, defined by a mean function m, and covariance function k.

$$f(x) \sim GP(m(x), k(x, x')) \tag{1}$$

In much the same way as a neural network model can be thought of as a function that returns a scalar value given an arbitrary input vector x, a GP model can be thought of as a function that, for a given x, returns the mean and variance of a normal distribution, with the variance indicating the certainty of the prediction.

Gaussian process models calculate predictions of unknown samples based on their distance to known samples in input space, a relationship defined by a covariance function. A common choice is the squared exponential function: the closer the points are in input space the more closely correlated they are in the output space:

$$k(\mathbf{x_i}, \mathbf{x_j}) = \exp\Big(-\frac{1}{2}\|\mathbf{x}_i - \mathbf{x}_j\|^2\Big) \tag{2}$$

Given an observation set $D = (x_{1:t}, f_{1:t})$ where $f_{1:t} = f(x_{1:t})$, a matrix of covariances is built. In the simple noise-free case this results in the kernel matrix:

$$K = \begin{bmatrix} k(x_1, x_1) & \cdots & k(x_1, x_t) \\ \vdots & \ddots & \vdots \\ k(x_t, x_1) & \cdots & k(x_t, x_t) \end{bmatrix} \tag{3}$$

When considering a new point (x_{t+1}) the value $(f_{t+1} = f(x_{t+1}))$ can be derived using the normal distribution built from this kernel matrix[a]:

$$\begin{bmatrix} \mathbf{f}_{1:t} \\ f_{t+1} \end{bmatrix} \sim \mathcal{N}\Bigg(\mathbf{0}, \begin{bmatrix} K & \mathbf{k} \\ \mathbf{k}^T & k(\mathbf{x}_{t+1}, \mathbf{x}_{t+1}) \end{bmatrix}\Bigg) \tag{4}$$

where $\mathbf{k} = [k(\mathbf{x}_{t+1},\ \mathbf{x}_1), k(\mathbf{x}_{t+1},\ \mathbf{x}_2),\ \ldots\ , k(\mathbf{x}_{t+1},\ \mathbf{x}_t)]^T$ allowing us to compute the GP as:

$$P(f_{t+1}|D_{1:t}, x_{t+1}) = \mathcal{N}\Big(\mu_t(x_{t+1}), \sigma_t^2(x_{t+1})\Big) \tag{5}$$

[a]for simplicity we assume the zero mean function: $m(x) = 0$

where:

$$\mu_t(x_{t+1}) = \mathbf{k}^T \mathbf{K}^{-1} \mathbf{f}_{1:t} \tag{6}$$

$$\sigma_t^2(x_{t+1}) = k(\mathbf{x}_{t+1}, \mathbf{x}_{t+1}) - \mathbf{k}^T \mathbf{K}^{-1} \mathbf{k} \tag{7}$$

gives us the predicted mean and variance for a normal distribution at the new point x_{t+1}. If we were then to evaluate the objective function at this point, we would add it to our set of observations D, reducing the variance at x_{t+1} and at other points near to x_{t+1}.

In this pure generalized form, our GP model weighs variations in every dimension equally, applying the same squared exponential relationship regardless of input dimension. For higher dimensional problems each dimension's effect on the output can also be weighted via a technique known as automatic relevance detection (ARD). The hyperparameters which weigh each dimension are set by maximizing the likelihood of the model given the data.[17] This increases model accuracy, and the weighting provides an human-readable estimation of the relative importance of each input dimension.

Visualization of Surrogate Models

Effective surrogate models are precise enough to be used by optimization algorithms, but as black-box models of high dimensional problem spaces, the knowledge they contain is inherently difficult to grasp. Despite the wealth of information encoded within these models, few attempts in the literature have been made to make this knowledge accessible.

Obayashi[13] and Jeong[14, 16] use self-organizing maps (SOMs) to project the results of single surrogate-assisted optimization runs from a high dimensional parameter space to a 2D space based on similarity of the samples in the observation set in parameter space. This allowed different design regions to be identified, as well as the design trade-offs they represent. In work by Jouhaud[18] an airfoil design optimization problem was solved using a surrogate model, and the resulting model visualized. As only two parameters were optimized and used by the model the search space could be easily sampled and presented as a 2D plane with the fitness, drag, and lift of different parameter regions shown by isolines.

Neither of these approaches can be used to intuitively visualize a high-dimensional surrogate model. In the approach of Obayashi and Jeong only precisely evaluated solutions are visualized, making the SOM not a description of the design space as interpreted by the current surrogate model, it is instead a history of the surrogate model's construction. While Jouhaud's approach provides the encoded knowledge of the surrogate model in an easily understood form, it is only a viable technique because of the low dimensionality of the problem, and cannot be scaled to higher dimensional design problems. By making use of a surrogate model to perform illumination of the feature space, SAIL allows intuitive visualization of the surrogate model's knowledge through the lens of chosen low-dimensional features.

III. Surrogate-Assisted Illumination

SAIL operates by modeling the underlying objective function in different high-performing regions of the search space. Knowledge of the objective function can then be applied to the production of high performing designs in different regions of the feature space. These high performing designs can in turn be examined as a whole in order to improve our understanding of the relationships between features, performance, and parameters. Deliberate sampling of the objective function to better understand its behavior and locate its optima is also the goal of Bayesian optimization (BO),[9, 15] and we borrow from its methods and vocabulary.

BO requires a probabilistic model of the objective function, such as a GP, and an *acquisition function*, which describes the utility of sampling a given point. The model of the objective function is produced from previously observed samples, the point with the greatest utility is found, evaluated, and added to the set of observations. The model is then reconstructed with these additional samples, and the process repeated.

It is infeasible to model the entirety of the search space when evaluations are computationally expensive, so sampling is directed towards the interesting regions of the search space. While in BO this interesting region is the region around the highest performing *point*, in SAIL we would like to find optima in all regions of the feature space, and so must model the region around a high performing *slice*.

How we define utility in the acquisition function determines how the search space is sampled. Balance should be maintained between exploration, sampling points with high uncertainty, and exploitation, sampling points which are likely to have high performance. In SAIL, the *upper confidence bound* (UCB)[19] is used as an acquisition function. UCB judges potential observations optimistically, favoring uncertainty under the assumption that higher uncertainty hides a potentially higher reward. A high mean ($\mu(x)$) and large uncertainty ($\sigma(x)$) are both favored, with relative emphasis tuned by the parameter κ.

$$UCB(\mathbf{x}) = \mu(\mathbf{x}) + \kappa\sigma(\mathbf{x}) \tag{8}$$

UCB performs competitively with more complex acquisition functions such as Expected Improvement (EI) and Probability of Improvement (PI).[15, 20] These acquisition functions rely on comparisons to the current optimum, while UCB is based only on the underlying model. This is more appropriate for SAIL, as solutions in lower performing regions of the feature space would all have vanishingly small probabilities of improving on the optima found in the highest performing regions. A more local approach, such as calculating EI only within bins, is also fraught with difficulties. In many cases there will be more bins than observations, and many bins will not contain previous observations for comparison, leaving local performance measures incalculable.

To explore a diversity of designs SAIL models the objective function around high-performing solutions that cover the entire feature space. In order to predict performance in this entire slice of the search space, we first produce designs with every combination of features. This is done by dividing the feature space into bins, and using MAP-Elites (*see Section II: Quality Diversity and MAP-Elites*) to produce designs which maximize the acquisition function in each bin. The resulting map, which contains designs with maximal predicted performance and uncertainty, we call the *acquisition map.*

It is from this acquisition map that we select new samples to be evaluated. In order to draw new observations from the feature space in a uniform fashion, we employ a Sobol sequence.[21] This space-filling sequence allows us to select bins in an increasingly dense, but evenly spaced manner. Once evaluated the performance of the samples can be added to our set of observations and the GP model reconstructed. We can then produce a new acquisition map with the updated GP model, and repeat the process until our evaluation budget is exhausted.

Algorithm 2 Surrogate-Assisted Illumination

```
1:                                                ▷ Initialize with G solutions drawn from Sobol sequence
2: $\mathcal{X} \leftarrow Sobol_{1:G}$, $\mathcal{P} \leftarrow PE(\mathcal{X})$                    ▷ PE = precise evaluation
3:
4: 1) Produce Acquisition Map
5: for iter = 1 $\rightarrow$ precise_evaluation_budget do
6:     $\mathcal{D} \leftarrow (\mathcal{X}, \mathcal{P})$                  ▷ Observation Set: Parameters, Performance
7:     $\mathcal{GP} \leftarrow Gaussian\_process\_model(\mathcal{D})$
8:     $acquisition() \leftarrow (UCB(), \mathcal{GP}())$
9:     $(\mathcal{X}_{acq}, \mathcal{P}_{acq}) = \text{MAP-ELITES}(acquisition(), \mathcal{X})$
                                                  ▷ Select solutions from acquisition map for PE
10:    $\mathbf{x} \leftarrow \mathcal{X}_{acq}(Sobol_{iter})$
11:    $\mathcal{X} \leftarrow \mathcal{X} \cup \mathbf{x}$, $\mathcal{P} \leftarrow \mathcal{P} \cup PE(\mathbf{x})$
12: end for
13:
14: 2) Produce Prediction Map
15: $\mathcal{D} \leftarrow (\mathcal{X}, \mathcal{P})$                      ▷ Observation Set: Parameters, Performance
16: $\mathcal{GP} \leftarrow Gaussian\_process\_model(\mathcal{D})$
17: $prediction() \leftarrow mean(\mathcal{GP}(x))$
18: $(\mathcal{X}_{pred}, \mathcal{P}_{pred}) = \text{MAP-ELITES}(prediction(), \mathcal{X})$
```

The SAIL algorithm is more precisely defined in Algorithm 2. An initial set of designs is first produced. For maximum coverage of the parameter space we again use a Sobol sequence,[21] this time to ensure even coverage of the *parameter* space. These designs and their performance form a set of observations D, which is used to construct a GP model. An empty acquisition map is then created and filled with the designs from D, along with their utility as judged by the acquisition function. These designs are taken as the starting population for MAP-Elites (Algorithm 1) which then illuminates the map as described in Section II: an elite is selected and its parameters perturbed to produce a new solution, it is assigned a bin based on its features, and it finally competes for the bin if it is not occupied. This illumination process repeats for a number of iterations, and results in an acquisition map of elite solutions who each maximize the acquisition function in their bin.

From the acquisition map we select the next samples for evaluation. To ensure even coverage of the *feature* space, we again employ a Sobol sequence to direct the sampling, this time producing coordinates in feature space rather than parameter values. These coordinates indicate the bin to be sampled, and the individual stored is precisely evaluated on the true objective function. Once evaluated these new designs and fitness pairs are added to our observation set D and the process can be repeated.

The mean prediction of the resulting GP model can then used as the objective function of MAP-Elites, and a *prediction map* produced. This map is an estimate of the relationship between features and performance, including a predicted optimal design for each bin. As only the surrogate model of the objective is used for evaluation, this prediction map can be produced with minimal computation.

Not only does this prediction map contain a variety of high-performing designs it also presents the effects of features on performance in an intuitive visual format, allowing optimal or interesting regions to be easily identified. Parameter values of the found optimal designs can also be viewed through this feature space lens, allowing different design regions to be distinguished and providing insight into the solution representation. If, for example, the optimal parameter values are all at the extreme of the allowed range, its bounds are likely too restrictive. If optimal parameter values across the feature space appear noisy and randomly distributed, it can be deduced that the parameter value has little effect on performance, either because it is genuinely unimportant or the parameter range is so restrictive that variation within that range has little effect.

IV. Experimental Setup

Objectives and Constraints

To test the ability of SAIL to create a variety of high performing designs we evaluate its performance on a 2D airfoil optimization problem. Our objective is the reduction of the C_D of an airfoil, while matching the area and lift of the high-performing RAE2822 airfoil. Foils are evaluated at an angle of attack of 2.7°, at Mach 0.5 and Reynolds number of 10^6. To ensure that area and lift constraints are closely followed we include quadratically increasing fitness penalties for violations. As the C_D values are quite small we take the negative logarithm of C_D as our drag, with higher values being more desirable. The evaluation criteria are defined as:

$$\mathit{fitness}(x) = \mathit{drag}(x) \times \mathit{penalty}_{\mathit{lift}}(x) \times \mathit{penalty}_{\mathit{area}}(x) \tag{9}$$

where $\mathit{drag}(x) = -log(C_D(x))$

$$\mathit{penalty}_{\mathit{lift}}(x) = \begin{cases} \left(\frac{C_L(x)}{\mathit{lift}_{RAE2822}}\right)^2, & \text{if } C_L(x) < lift_{RAE2822} \\ 1, & \text{otherwise} \end{cases} \tag{10}$$

$$\mathit{penalty}_{\mathit{area}}(x) = \left(1 - \frac{|area - area_{RAE2822}|}{area_{RAE2822}}\right)^7 \tag{11}$$

During optimization both C_D and C_L values are approximated with Gaussian Process models. The UCB of the drag prediction is taken as the drag component:

$$\mathit{drag}(x)' = \mu_{\text{drag}}(x) + \kappa\sigma_{\text{drag}}(x) \tag{12}$$

We treat the lift prediction as a classification problem. Based on the mean and variance of the predicted lift, we penalize foils based on the *probability* that their lift is less than that of the base RAE2822 foil. The GP prediction based lift penalty is defined as:

$$\mathit{penalty}_{\mathit{lift}}(x)' = 1 - P(C_L(x) < lift_{RAE2822}) \tag{13}$$

Representation

Airfoils are encoded using the airfoil-specific PARSEC parameterization.[22] Polynomial expressions are used to encode design features, such as the height of the airfoil and the angle of the trailing edge. The PARSEC representation allows a wide variety of designs to be expressed with a small number of design parameters. The sharpness and end point of the trailing edge is fixed to that of the base RAE2822 foil, and the remaining 10 parameters are shown below in Figure 2.

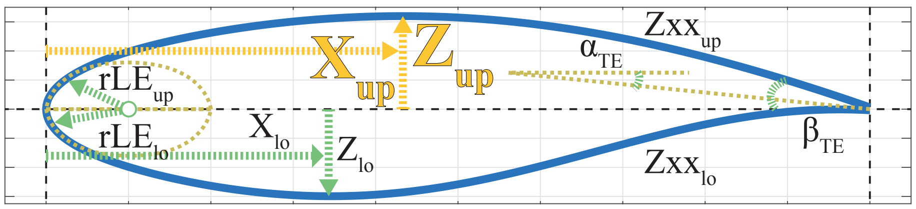


Figure 2. The 10 parameters used to define an airfoil. Dimensions of variation (X_{up} and Z_{up}) in gold.

Illumination algorithms allow us to explore the design space along a continuum in feature space. In this example we choose two of the PARSEC parameters: the height of the highest point on the top side of the foil (Z_{up}), and the location along the length of the wing of this highest point (X_{up}). These two parameters were chosen because of the large effect on the drag of the foil they were found to have in early tests. We divide the possible range of Z_{up} and X_{up} into 25 partitions, resulting in a 25×25 grid, for a total of 625 bins.

In practice we would typically define feature dimensions that do not correspond to parameter values. This allows the exploration of higher level features that are the result of parameter combinations, enabling the projection of a high dimensional representation onto a lower feature dimensional space of the designer's choosing (*see discussion in Section VI: Representation*). These higher level features should be chosen based on the designer's experience, or to explore a particular relationship. Here we select parameter values as features so that performance to enable comparison to other, more traditional, optimization algorithms.

Baseline and Hyperparameters

As one of the primary objectives of the SAIL algorithm is to achieve results in computationally expensive domains, the unit of comparison used is the number of precise evaluations (PE) required to achieve a certain result. Model construction and other algorithmic costs are ignored and assumed to be dominated in truly expensive optimization problems. The optimality of the designs in the prediction maps produced by SAIL are compared to those produced by 1) the original MAP-Elites algorithm, 2) the black-box optimization algorithm *Covariance Matrix Adaptation Evolution Strategy* (CMA-ES),[23] and 3) a surrogate-assisted variant of CMA-ES (SA-CMA-ES).

SAIL is given a total computational budget of 1000PE. 50PE is used to evaluate a set of designs produced through parameter sampling which form the basis of the initial GP model. At each iteration (Algorithm 2 lines 4-12) 10 additional individuals are evaluated and incorporated into the GP model until the computation budget is exhausted. This result was compared with the MAP-Elites algorithm given a computational budget of 10^5PE.

To better understand the true performance of our algorithm in finding optimal solutions, we compare it to the black-box optimization algorithm CMA-ES,[23] which is designed to produce a single solution. As the dimensions of variation correspond to parameter values, by restricting the allowed parameter ranges of the representation we can confine the search within a single cell. Each bin is treated as an independent search problem which CMA-ES is given a budget of 1000PE to solve. In addition we employ a surrogate-assisted variant of CMA-ES, SA-CMA-ES, to solve each of these subproblems. Within a single bin 25 initial solutions are sampled from the parameter space and used to produce an initial GP model. CMA-ES is then used to maximize the UCB based acquisition function described above. The single found optimum is evaluated and incorporated into the GP model. This process is repeated until a budget of 100PE is exhausted.

Each approach was replicated 20 times with different random seeds.[b] Unless otherwise mentioned, all values are medians across these replicates. As valid initial designs whose highest point was found at the leading edge of the wing (high Z_{up} and low X_{up}) could not be found due to geometric constraints inherent to the PARSEC representation,[24] only the remaining 577 bins were considered in comparisons. Where possible, standard implementations of algorithms were used, including CMA-ES,[25] Gaussian Processes,[26] and airfoil simulation.[27]

[b]One replicate, including data gathering, with 8 cores of a Intel Xeon 2.6GHz processor required approximately: SA-CMA-ES:32h, CMA-ES:80h, SAIL:12h, MAP-Elites:14h

V. Results

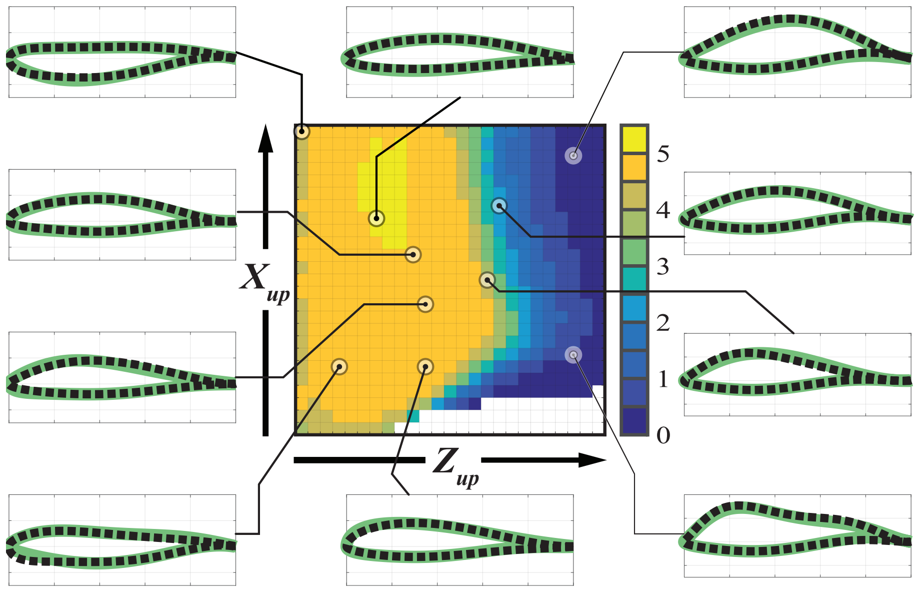


Figure 3. ***Design Space Overview with SAIL*** **– Prediction map produced by SAIL after 1000PE. Border: Median performing designs found by SAIL in green, best designs found after *all* CMA-ES runs in black. Fitness is dominated by the drag component: $-log(C_D)$, e.g. a fitness of 5 is equivalent to a C_D of 1e−5**

In Figure 3 we see SAIL's prediction of the best performing designs in each region of the feature space. The values and designs shown are the median predicted performance over 20 runs of SAIL. The designs produced in these median runs are similar to those optimal designs found by CMA-ES, a selection of which are shown in black around the border. These were the best designs produced over all 20 runs of CMA-ES in each bin, using nearly 11.5 million PE to produce the whole map. It is these designs that we designate as the 'optimum' design in each bin for comparison.

Airfoil height (Z_{up}) is the most influential feature, with thinner airfoils typically corresponding to lower drag values (and higher performance). The location of the airfoils highest point (X_{up}) has a more nuanced effect, dependent on the height of the airfoil. The best performing airfoils were not those at the extremes of the feature space, as a certain lift must be maintained, thinner airfoils are not always better. Though this map is only a prediction, in the following sections we present evidence that it presents a faithful representation of the performance potential of the design space.

Accuracy

At the conclusion of the SAIL algorithm a prediction map is produced by illuminating the feature space without rewarding uncertainty. This gives us our model's 'best guess' of the optimal designs in the feature space. In this two-dimensional airfoil case it was possible to precisely evaluate every design in the prediction map and so judge its accuracy in different regions of feature space. The median prediction, true values, and their absolute percentage error are shown in Figure 4 below.

Over the majority of the feature space, especially for C_D, the predictions made by the model are reliably accurate. 90% of drag ($log(C_D)$) predictions and more than 80% of lift (C_L) predictions are within 5% of their true value. Drag errors are clustered in one region of the feature space, where the flow simulator was less likely to converge and produce valid results. In these experiments when the simulator did not converge the result was simply discarded and the next sample taken, leaving this region with fewer samples on which to base a prediction.

To allow for effective optimization our models are designed to accurately predict performance in optimal regions of the design space across a slice of the feature space. This is done by directing the sampling process toward high performing solutions in different regions of the feature space. To measure the accuracy of our models in this high fitness slice we test their ability to accurately predict the performance of the optimal solutions found by CMA-ES. We compare the models built through selection guided by SAIL to models built by evenly sampling the parameter space with a Sobol sequence.[21]

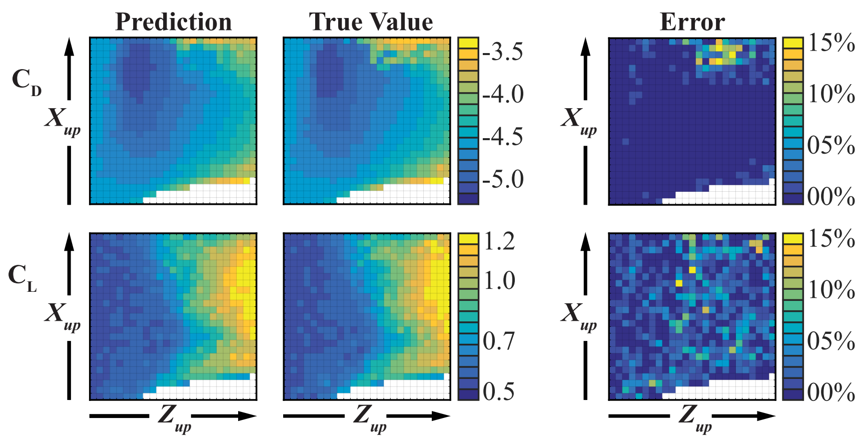


Figure 4. ***Drag and Lift Predictions and True Value in Each Bin***
Predicted and true values of drag and lift for designs in each bin after 1000PE. Median values over 20 replicates.

To further tease apart the factors which contribute to SAIL's modeling of the feature space we also examine the effect of the choice of acquisition function. As discussed in Section III, we guide selection through the upper confidence bound (UCB), a combination of the mean and variance of the prediction. This should guide the search process to promising unexplored regions. The MAP-Elites illumination process is also designed to find diverse high quality solutions. To differentiate the contribution of each to directing the search towards the high fitness slice we test SAIL using an acquisition function of *only* the mean, and *only* the variance.

The accuracy of each resulting model's drag prediction on the optimal design set is measured at various stages of the sampling process. Figure 5 shows that using the SAIL algorithm to select samples from across the feature space improves accuracy over evenly sampling the parameter space, regardless of whether uncertain or high-performing solutions are favored. When both the uncertainty and performance are considered, as it is with the UCB, SAIL produces models which are an order of magnitude more accurate on the high fitness slice than uniform sampling.

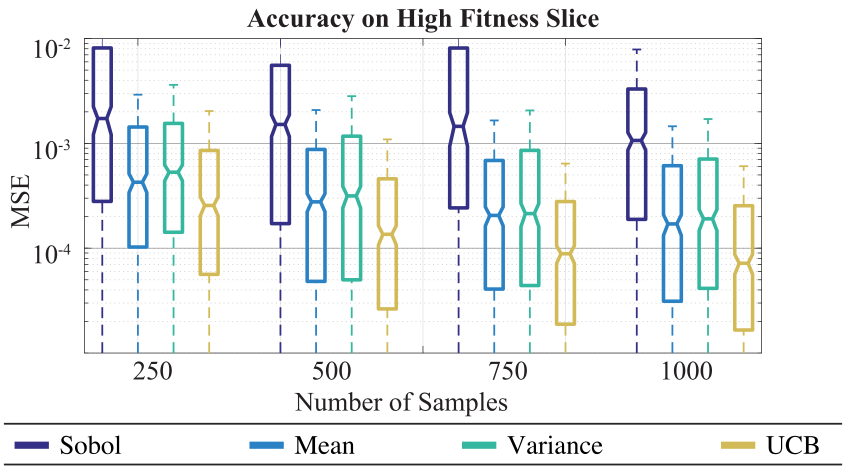


Figure 5. ***Accuracy of Sampling Strategies***
Mean squared error (log scale) of drag prediction on optimal designs. Models constructed using designs sampled from parameter space using a Sobol sequence or selected from acquisition maps produced with the mean, variance, or the UCB of the prediction.

Optimality and Efficiency

It is not our goal to compete with algorithms designed to find one very high-performing solution. If we are to accurately portray the performance of feature space, however, it is required that the solutions which we produce represent the best designs in the region. We again rely on the designs produced through repeated applications of CMA-ES as our benchmark of optimality. The median designs found by SAIL in each bin after 1000PE were compared to the best found by CMA-ES in each bin after 20 runs for 1000PE in each bin ($\approx$ 11.5 million PE in total). In Figure 6 we see the median predicted fitness in each bin, the true fitness of the designs found after evaluating them, and the 'optimal' fitness in each bin found by CMA-ES.

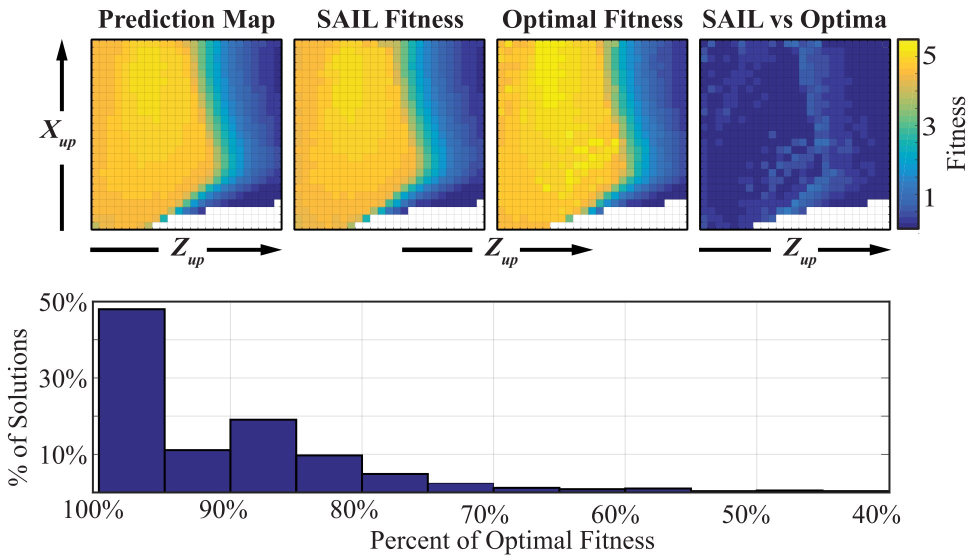


Figure 6. ***Performance of Designs Found By SAIL***
Top**: Median predicted and true performance of designs found by SAIL with a budget of 1000PE compared with performance of optimal designs found by 20 runs of CMA-ES per bin ($\approx$11.5 million PE)**
Bottom**: Optimality of SAIL designs per bin.**

Though the solutions found by CMA-ES have higher performance, as we would expect from a state-of-the-art black box optimization algorithm using four orders of magnitude more evaluations, SAIL provides a near-optimal approximation of the feature space. The fitness potential of each bin is well illuminated: found designs in nearly half the bins are within 5% of the optimum, and the general relationship between features is accurately portrayed.

We are unaware of such feature space exploration algorithms beyond MAP-Elites, and so judge the data efficiency of our algorithm in comparison to CMA-ES. CMA-ES was not designed for use across a multitude of subproblems, and so the number of PEs required to arrive at an optimized feature map is highly dependent on the number of bins that are defined in the map. For a more informative comparison we therefore compare SAIL to the performance of CMA-ES within a single bin as well as its performance in optimizing the map as a whole. We include a surrogate-assisted variant of CMA-ES (SA-CMA-ES) (*see Baseline and Hyperparameters, previous section*) to control for the ease of modeling the problem, and also compare its performance in a single bin and across the entire map.

Figure 7 shows these comparisons of performance per evaluation. As optimization progress may vary according to the bin, we take the single bin performance as the median performance over all bins. Performance itself is measured as the percent of the optimum found by CMA-ES in each bin. As this optimum is the highest performance over *all* runs, the median CMA-ES run does not typically reach this value. Performance of the individuals produced to construct the initial surrogate models is set to 0%, with the first valid performance indicators at 25PE/bin for SA-CMAES and 50PE for SAIL.

Given the budget required by CMA-ES to find a near optimal solution in a single bin, SAIL finds designs of comparable performance in *every* bin. Comparisons between CMA-ES and MAP-Elites and their surrogate-assisted variants SA-CMA-ES and SAIL reveal that the gains from surrogate-assisted optimization are even greater for MAP-Elites than the traditional optimizer. Even when MAP-Elites is given an evaluation budget two orders of magnitude greater than SAIL it cannot produce solutions of similar performance.

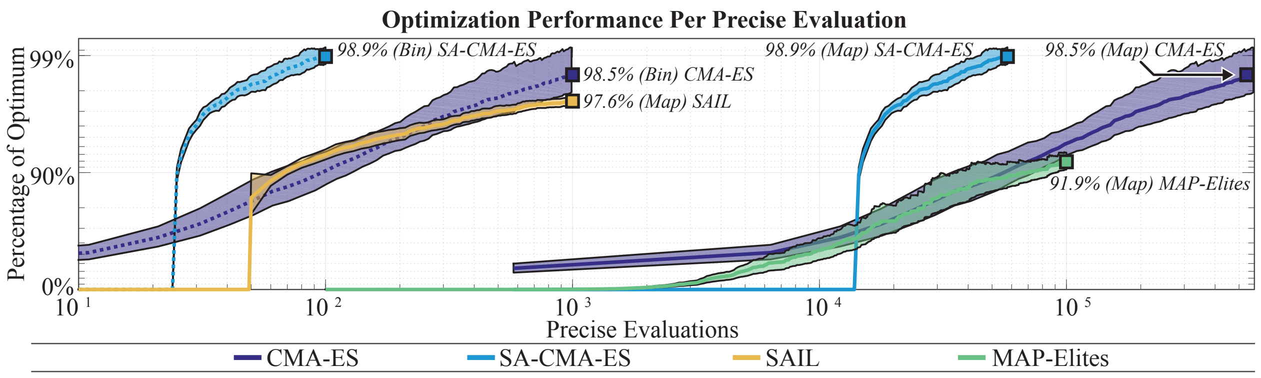


Figure 7. ***Optimization Efficiency in a Single Bin and Over the Entire Design Space***
Computational efficiency of CMA-ES, SA-CMA-ES, MAP-Elites, and SAIL measured in in precise evaluations. ***Bin*: median progress towards optimum in every bin. *Map*: performance of CMA-ES and SA-CMA-ES is median bin performance multiplied by number of valid bins. Performance of individuals produced to construct initial models is set to 0%. Bounds indicate one standard deviation over 20 replicates. PEs and performance in log scale.**

VI. Case Study: Velomobile Design

Having established the capabilities of the SAIL algorithm in a 2D case, we now present design exploration results in the more computationally demanding context of 3D aerodynamics. We apply the SAIL algorithm to the problem of creating aerodynamic shells for fully faired recumbent bicycles. Such streamlined vehicles hold human powered top speed (144.17 km/h [c]) and distance (680km in 12 hours, 1219km in 24 hours [d]) records due to their highly tuned aerodynamics.

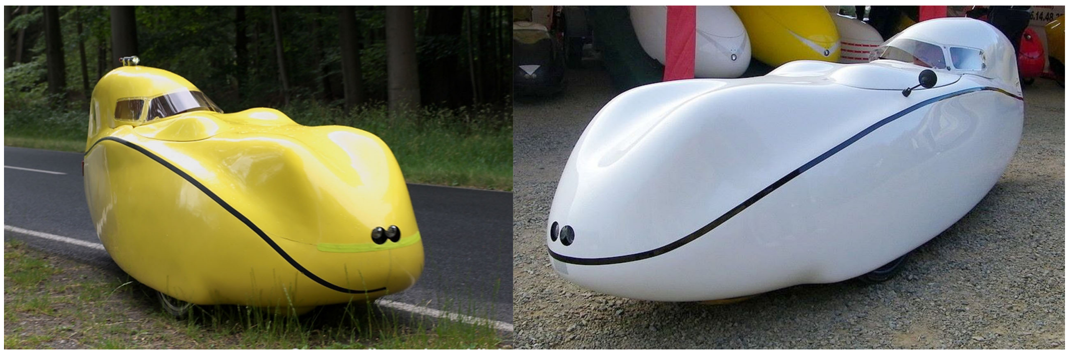


Figure 8. Record-setting velomobiles have non-intuitive shapes

Due to constraints such as rider movement and comfort, three-wheeled designs built for distance races, known as velomobiles, often have non-intuitive shapes (see Figure 8). These high-performing, but odd, designs suggest that the domain may be rich in interesting design concepts. By producing a set of varied designs, we illustrate the advantages of SAIL over approaches focused purely on optimization or exploration, and within the limited computational budget enforced by the use of 3D CFD simulation.

Representation

In this work we focus on a simplified base body of a velomobile only, without more complex structures such as wheel housings or a cockpit for the rider's head. We describe this three dimensional shape with a series of airfoil-like curves defined with the PARSEC[22] representation (*see Section IV: Representation*). Viewed from the top, the vehicle is defined as a symmetrical airfoil (*Top* curve). The side profile is defined by three curves: one fixed curve for the bottom and two parameterized curves for the top, one in the center of the vehicle (*Mid* curve) and one which forms a 'ridge' for the knees (*Ridge* curve). A final curve connects this ridge to a flat bottom to form the view from the front (*Rib* curve). This *Rib* is defined along a unit vector,

[c]`http://www.recumbents.com/wisil/whpsc2016/results.htm`
[d]`http://www.whpva.org/land.html`

and is scaled to maintain the curves defined by the *Top* and *Ridge* curves in 32 sections. From the top view ridges follow the same curve as the *Top* curve of the body. All curves are connected to each other by splines. An example of these curves and the 3D shape they produce is illustrated below in Figure 9.

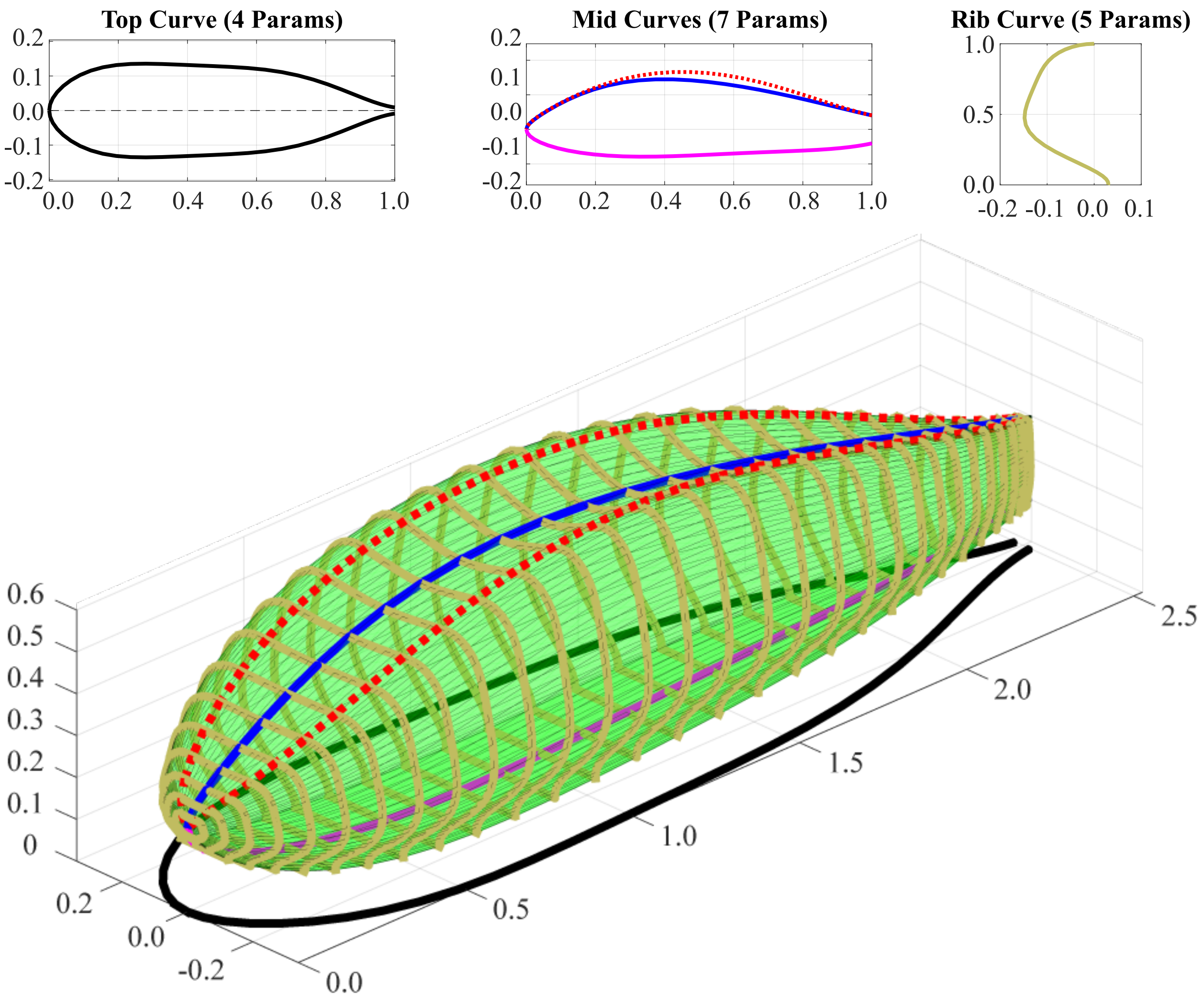


Figure 9. A three-dimensional velomobile shape produced from a set of curves

Table 1. The 16 parameters used to determine the shape of a velomobile shell. The tallest point of the ridge is based on the height of the middle point.

Curve	PARSEC Parameter			
Top	Leading Edge Radius	Position of Tallest Point	Tallest Point	Curvature
Mid	Leading Edge Radius	Position of Tallest Point	Tallest Point	Curvature
Ridge	Leading Edge Radius	Position of Tallest Point	Relative Height	
Rib	Leading Edge Radius Angle of Trailing Edge	Position of Tallest Point	Tallest Point	Curvature

Each PARSEC airfoil can be described by 10 or more parameters, but as we are only producing one curve rather than an entire foil, only the parameters which describe one side are necessary. After initial tests with SAIL we examined the automatic relevance detection coefficients in our Gaussian Process models (*see Section II: Gaussian Process Models*). From the values of these coefficients we could identify parameters which had negligible influence on performance across the entire features space. By fixing these less important parameters we were able to reduce the representation to the 16 values in Table 1. Reducing the number of parameters allows both optimization and modeling of the design space to be performed more effectively.

Objective and Dimensions of Variation

The produced velomobile shells are judged on the drag force they produce when traveling at 20 m/s (72 km/h). Flow simulation is performed with the OpenFOAM[e] CFD toolbox. We explore two dimensions of variation: volume and curvature. Volume is computed as the convex hull of the design, curvature is the average curvature of all ribs. Volume is necessary for both the rider and the equipment inside of the shell. It is obvious that larger volume shapes will tend to have a larger amount of drag, but it is useful for a designer to be able to select a high-performing design, and if it is too small, select a slightly larger one. Curvature of the design can allow for a smaller frontal area while still maintaining volume, for example to accommodate the rider's knees. It also has the effect of adding structural integrity to the light-weight shell. In areas where the shell is flat it is also weak, enough to buckle and alter shape at high speeds or in high winds, as well as being easily damaged in accidents.

VII. Results

SAIL was run with a budget of 1000 PE, with 200 PE used for the initial sampling, and 10 new samples drawn after illumination. The volume and curvature feature space was divided into 20 sections, producing a 20 × 20 feature grid. To analyze the true performance of the algorithm we evaluated all 400 designs in the prediction map at the end of the run. As SAIL relies on stochastic processes for optimization this was repeated 15 times: unless otherwise mentioned all values refer to the median performance of these replicates.

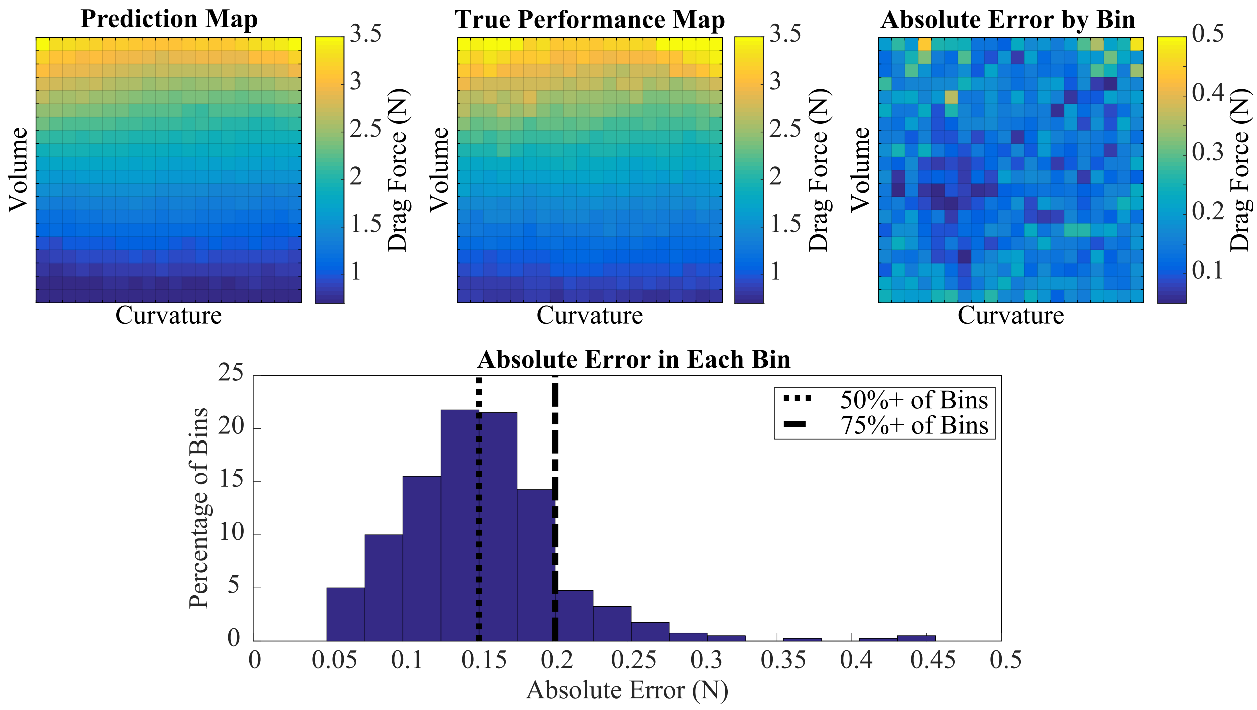


Figure 10. Predicted performance of designs found by SAIL, the true performance of the designs found through simulation, and the absolute error in each bin. Median results after 15 replicates each with a budget of 1000 PE.

The prediction map produced along with the true performance of the designs in the prediction map are shown in Figure 10. While the model is optimistic in its predictions, the error of the prediction is quite low, with the majority of designs performing only .15N less than predicted, and clearly captures the unsurprising relationship between increased volume and greater drag force. In Figure 11 a selection of designs produced in a single run from different regions of the map are shown.

[e]http://www.openfoam.com/

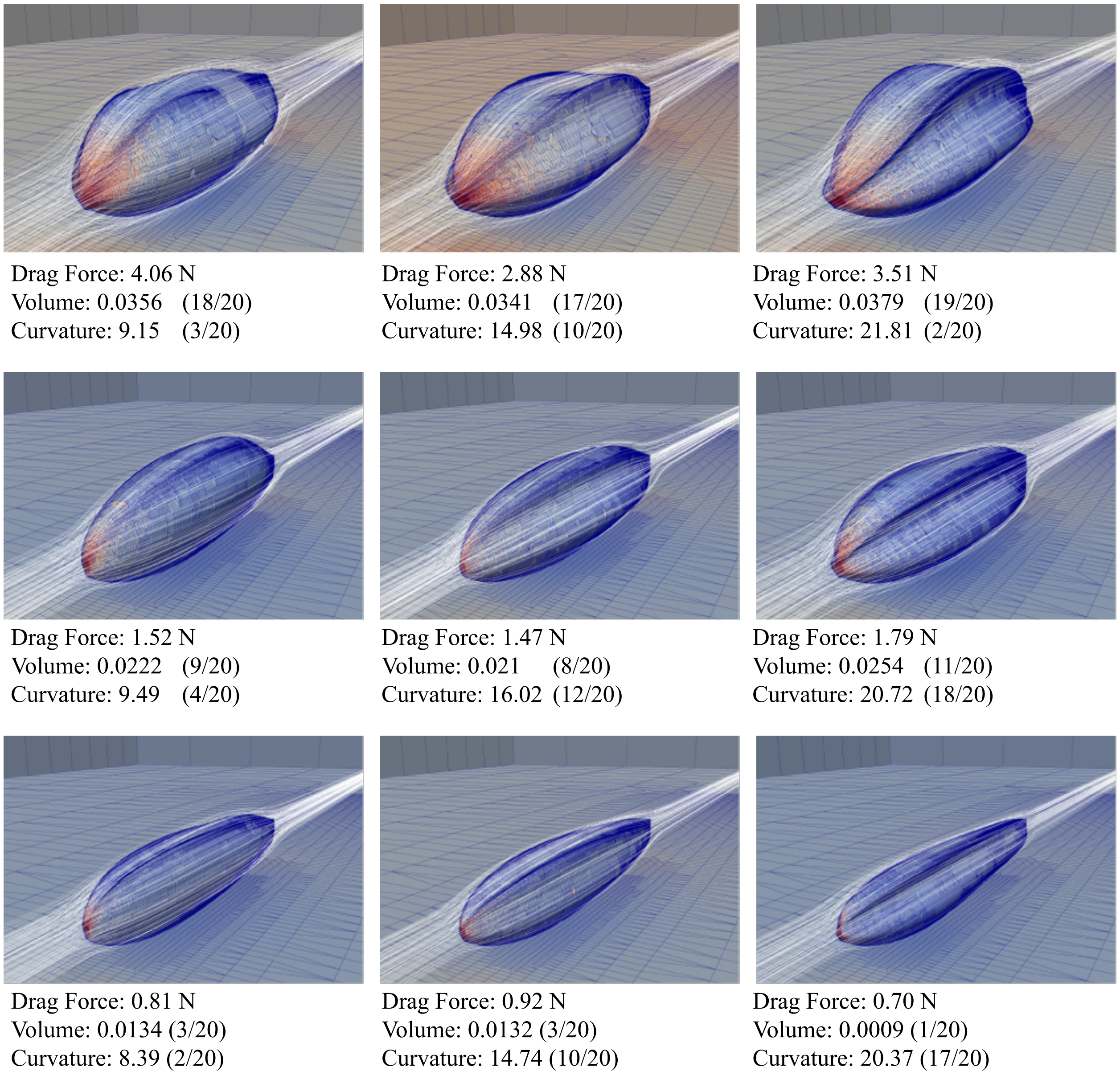


Figure 11. A selection of designs produced in a single run of SAIL. Left: Lower Curvature, Up: Increased Volume Parenthesis after category indicate the bin where each design is located

We can see the variety of designs in the two specified axes of variation. A clear continuum can be observed, with each design similar to its neighbors, and those at opposite ends of the feature space having little in common. This variety allows a designer to quickly scan through a collection of high-performing designs according to characteristics and requirements of their own choosing.

For example, the design at the bottom right, while performing very well has such a low volume that a rider may not fit inside. The designs in the middle row, while large enough for the rider, may require some adjustment if the proper gearing is to fit inside. In the case that adjustments to the shell or creative configuration of the internal mechanisms is not sufficient, a larger high-performing design such as the shell in the middle of the top row can be taken as a starting point.

More Explorative than Pure Optimization

When producing a single design a traditional optimizer will perform better than a design space exploration algorithm, but is seriously limited in its ability to explore according to the interest of the designer. This is despite the fact that an optimizer will not always converge on identical solutions. CMA-ES was given the same 1000PE budget to find optimal solutions for our problem and, depending on the initial starting solutions, converged on different designs. Two such designs are shown in Figure 12.

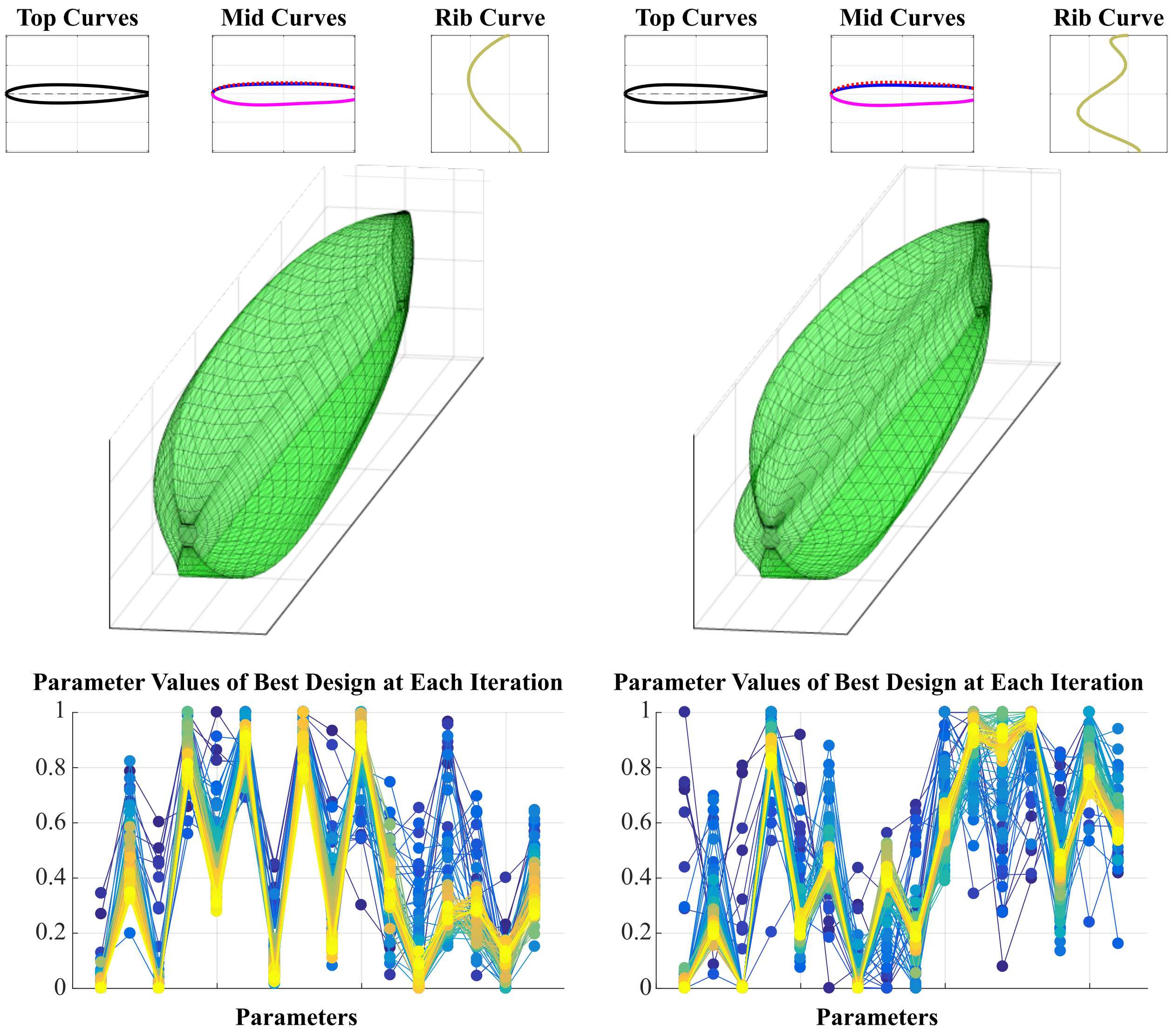


Figure 12. CMA-ES Shapes found in two separate runs using 1000PE with different starting positions. *Top:* Designs and their component curves *Bottom:* Parameter values of best performing design found at each CMA-ES iteration in the separate runs, warmer colors indicate later iterations.

These designs are in many ways dissimilar, one with a single gentle curve on the side, and the other favoring high curvature. The progress of the optimization was tracked and the parameter values of the best found individuals is shown at the bottom of Figure 12. Each run shows convergence on different optima, each in different areas of the parameter space.

While this seems to make the case for exploration using traditional algorithms, if optimization progress is instead tracked in feature space rather than parameter space (Figure 13) an altogether different picture emerges. As volume is closely linked with drag force the search for optimality quickly converges on the smallest designs allowable by the representation. The search is confined to a very small area of the feature space, leaving us with only a narrow range of designs. This may be desirable behavior in an optimization context, but provides little assistance to a designer trying to understand the problem space more broadly.

To a designer the optimal volume may depend on other constraints not easily inserted into the algorithm, such as how other mechanisms and equipment could be arranged within the shell. By producing a selection of designs that vary according to their needs the designer can find promising designs by exploring a menu of options.

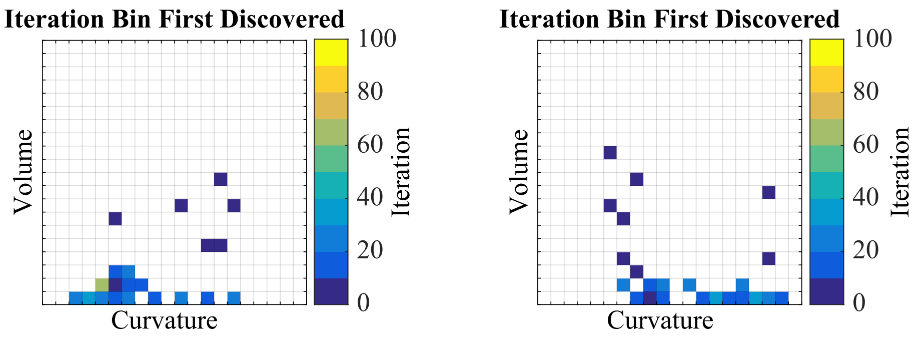


Figure 13. Feature space exploration in two sample runs of CMA-ES. First iteration in which a design is found in each bin. White bins indicate that no solution was ever ever found with this combination of features.

More Optimal than Pure Exploration

To produce representative designs across the feature space, sampling of the parameter space alone will typically not be sufficient. In the earlier airfoil experiments the feature space was a subset of the parameter space. In that case evenly sampling the parameter space produced an even sampling of the feature space. In this case, however, the volume and curvature features are the result of a combination of parameters. This causes evenly spaced sampling of the parameter space to result in an uneven sampling of the feature space.

To illustrate this, 1000 samples were produced by sampling the parameter space with a Sobol sequence, recording the resulting designs position in feature space. This process was repeated 10 times and the number of times each bin was discovered recorded. The frequency at which each bin was sampled over each run is shown in the left of Figure 14, revealing a clear bias in parameter space toward low curvature shapes, in addition to the extremes of the feature space being left undiscovered in every attempt.

Even when the feature space is well sampled the designs are not necessarily 'good' examples of designs in this feature region. As volume is tightly coupled with performance their general relationship is clearly visible from parameter sampling, but as examples of high-performing designs they are insufficient. There is no guarantee that the designs found by sampling are any better than average for their bin, as no optimization of any kind has taken place. The median fitnesses in Figure 14 display this volume/drag force relationship so smoothly only because they are median results of repeated sampling. In the case of more limited sampling the results can easily be much noisier. Furthermore as the designs which fill each bin were produced pseudo-randomly without any method of improvement, searching through these solutions for common design characteristics makes little sense.

SAIL simultaneously explores the feature space and optimizes within each bin. This allows SAIL to not only more thoroughly explore the feature space, but produce high-performing examples of designs for each feature combination. The improvement of designs produced by SAIL over those found by parameter sampling is shown at the bottom of Figure 14.

VIII. Conclusion and Discussion

A design space exploration algorithm, Surrogate-Assisted Illumination (SAIL), was introduced. By combining surrogate-assisted optimization techniques with quality-diversity methods SAIL produces a large variety of high-performing solutions with limited computational effort. Produced designs vary according to features described by the user, giving the designer the flexibility to explore design characteristics that interest them, allowing their own domain knowledge and intuition to lead them to interesting design regions. By producing accurate models of the design space and designs which those models predict will perform well,

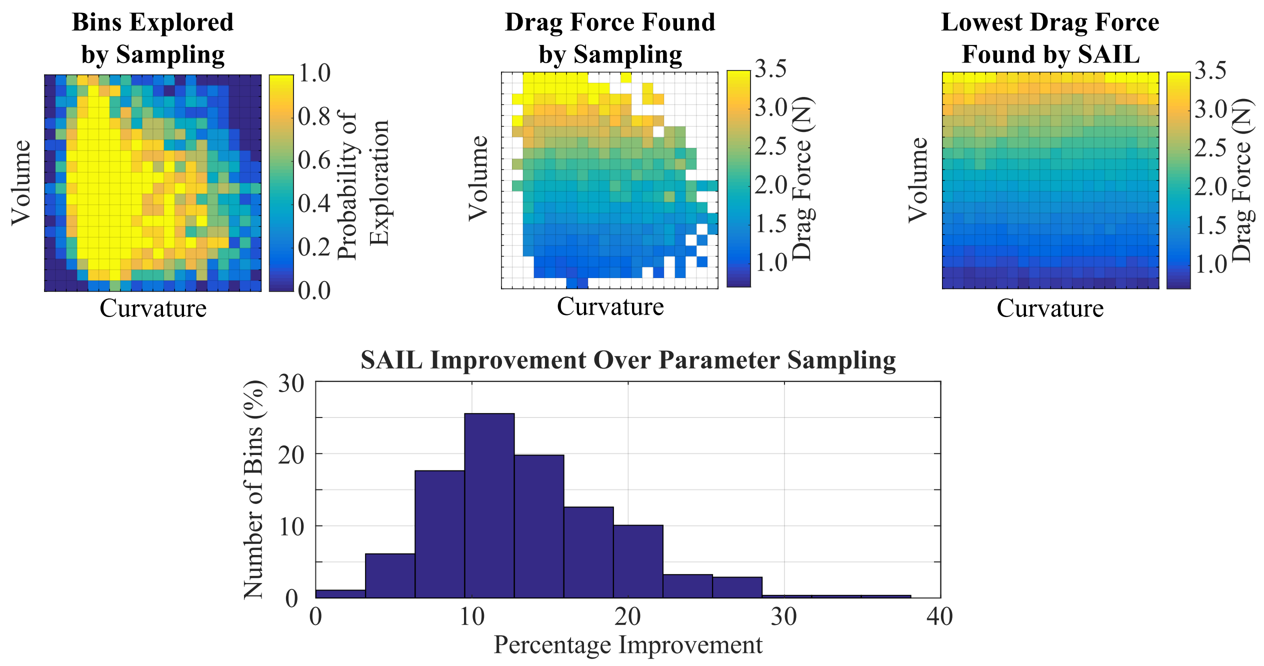


Figure 14. ***Left:*** **Probability of sampling a bin with 1000 samples drawn from a Sobol sequence.** ***Middle:*** **Median drag force values found by sampling the parameter space.** ***Right:*** **Median drag force values found in a median run with SAIL.** ***Bottom:*** **Improvement per bin of SAIL over parameter sampling with a Sobol sequence. Only bins which were sampled by parameter sampling 3 or more times in the 10 trials are shown.**

an explanatory map of the relationship between features and performance is produced.

By accelerating the illumination process, exploration of the design space can be done rapidly and in an any-time fashion. Once new samples are available the surrogate model can be reconstructed and the estimated performance potential of the entire feature space updated. This ’real-time’ view of the search space differentiates the SAIL algorithm from techniques which visualize the precisely evaluated solutions used to produce a model. By visualizing the current best predictions rather than the set of evaluated solutions we see the search space as the model currently understands it; the evaluated solutions are only a history of the optimization process.

The same model can be used to quickly produce prediction maps of different resolution and different features combinations. This allows more detailed exploration of interesting areas, and visualization of the design space through various feature lenses. Data mining techniques designed to extract common design principles from a set of high-performing designs can still be used in conjunction with SAIL. That designs are produced along a continuum defined by the designer provides the added ability to choose elements of variation in high-performing designs to examine more closely.

The capability to rapidly understand the performance potential of the design space through concrete high-performing examples is a potential boon to designers. SAIL not only accelerates the generative design cycle, but allows the effect of user-defined features to be examined, adding new flexibility to cooperative human-machine design exploration. Generative design tools such as SAIL, which consider more than objectives, can help designers understand what is possible, beyond what is optimal.

Acknowledgments

This work received funding from the European Research Council (ERC) under the European Union’s Horizon 2020 research and innovation programme (grant agreement number 637972, project "ResiBots") and the German Federal Ministry of Education and Research (BMBF) under the Forschung an Fachhochschulen mit Unternehmen programme (grant agreement number 03FH012PX5 project "Aeromat").

References

[1]Hornby, G., Globus, A., Linden, D., and Lohn, J., "Automated Antenna Design with Evolutionary Algorithms," *Space 2006*, 2006.

[2]Bradner, E., Iorio, F., and Davis, M., "Parameters tell the design story: Ideation and abstraction in design optimization," *Simulation Series*, 2014.

[3]Deb, K., "Unveiling innovative design principles by means of multiple conflicting objectives," *Engineering Optimization*, 2003.

[4]Pugh, J., Soros, L., and Stanley, K., "Quality Diversity: A New Frontier for Evolutionary Computation," *Frontiers in Robotics and AI*, 2016.

[5]Cully, A., Clune, J., Tarapore, D., and Mouret, J.-B., "Robots that can adapt like animals," *Nature*, 2015.

[6]Mouret, J.-B. and Clune, J., "Illuminating search spaces by mapping elites," *arXiv preprint:1504.04909*, 2015.

[7]Jin, Y., "A comprehensive survey of fitness approximation in evolutionary computation," *Soft computing*, 2005.

[8]Forrester, A. I. J. and Keane, A., "Recent advances in surrogate-based optimization," *Progress in Aerospace Sciences*, 2009.

[9]Shahriari, B., Swersky, K., Wang, Z., Adams, R., and de Freitas, N., "Taking the human out of the loop: A review of bayesian optimization," *Proceedings of the IEEE*, 2016.

[10]Nguyen, A., Yosinski, J., and Clune, J., "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images," *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, 2015.

[11]Giannakoglou, K., Papadimitriou, D., and Kampolis, I., "Aerodynamic shape design using evolutionary algorithms and new gradient-assisted metamodels," *Computer Methods in Applied Mechanics and Engineering*, Sep 2006.

[12]Dumas, L., "CFD-based optimization for automotive aerodynamics," *Optimization and Computational Fluid Dynamics*, 2008.

[13]Obayashi, Sand Sasaki, D., "Visualization and data mining of Pareto solutions using self-organizing map," 2003.

[14]Jeong, S., Chiba, K., and Obayashi, S., "Data Mining for Aerodynamic Design Space," *Aerospace Computing, Information and Communication*, 2005.

[15]Brochu, E., Cora, V., and Freitas, N. D., "A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning," *arXiv preprint arXiv:1012.2599*, 2010.

[16]Jeong, S., Murayama, M., and Yamamoto, K., "Efficient optimization design method using kriging model," *Journal of aircraft*, Vol. 42, No. 2, 2005, pp. 413–420.

[17]Rasmussen, C. and Williams, C., "Gaussian Process for Machine Learning," *Gaussian Process for Machine Learning*, 2006.

[18]"A surrogate-model based multidisciplinary shape optimization method with application to a 2D subsonic airfoil," *Computers and Fluids*, 2007.

[19]Srinivas, N., Krause, A., Kakade, S., and Seeger, M., "Gaussian Process Optimization in the Bandit Setting: No Regret and Experimental Design," *Proceedings of the 27th International Conference on Machine Learning (ICML 2010)*, 2010.

[20]Calandra, R., Seyfarth, A., Peters, J., and Deisenroth, M. P., "An experimental comparison of Bayesian optimization for bipedal locomotion," *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, IEEE, 2014.

[21]Niederreiter, H., "Low-discrepancy and low-dispersion sequences," *Journal of Number Theory*, 1988.

[22]Sobieczky, H., "Parametric airfoils and wings," *Recent Development of Aerodynamic Design*, 1999.

[23]Hansen, N. and Ostermeier, A., "Completely derandomized self-adaptation in evolution strategies," *Evolutionary computation*, 2001.

[24]Padulo, M., Maginot, J., Guenov, M., and Holden, C., "Airfoil Design Under Uncertainty with Robust Geometric Parameterization," *17th AIAA/ASME/AHS Adaptive Structures Conference*, 2009.

[25]Hansen, N., "CMAES Version 3.61.beta," `https://www.lri.fr/~hansen/cmaes_inmatlab.html`, 2008.

[26]Rasmussen, C. and Nickisch, H., "Gaussian Process Regression and Classification Toolbox," http://www.gaussianprocess.org/gpml/code/, 2016.

[27]Drela, M., "XFOIL airfoil simulator," raphael.mit.edu/xfoil/, 2013.