Published in final edited form as:
IEEE Trans Vis Comput Graph. 2013 December ; 19(12): 2783–2791. doi:10.1109/TVCG.2013.147.

Design by Dragging: An Interface for Creative Forward and Inverse Design with Simulation Ensembles

Dane Coffey[1] **[Student Member, IEEE]**, **Chi-Lun Lin**[2], **Arthur G. Erdman**[2], and **Daniel F. Keefe**[1] **[Senior Member, IEEE]**

Dane Coffey: coffey@cs.umn.edu; Chi-Lun Lin: linxx691@me.umn.edu; Arthur G. Erdman: agerdman@me.umn.edu; Daniel F. Keefe: keefe@cs.umn.edu

[1]Department of Computer Science and Engineering, University of Minnesota

[2]Department of Mechanical Engineering, University of Minnesota

Abstract

We present an interface for exploring large design spaces as encountered in simulation-based engineering, design of visual effects, and other tasks that require tuning parameters of computationally-intensive simulations and visually evaluating results. The goal is to enable a style of design with simulations that feels as-direct-as-possible so users can concentrate on creative design tasks. The approach integrates forward design via direct manipulation of simulation inputs (e.g., geometric properties, applied forces) in the same visual space with inverse design via "tugging" and reshaping simulation outputs (e.g., scalar fields from finite element analysis (FEA) or computational fluid dynamics (CFD)). The interface includes algorithms for interpreting the intent of users' drag operations relative to parameterized models, morphing arbitrary scalar fields output from FEA and CFD simulations, and in-place interactive ensemble visualization. The inverse design strategy can be extended to use multi-touch input in combination with an as-rigid-as-possible shape manipulation to support rich visual queries. The potential of this new design approach is confirmed via two applications: medical device engineering of a vacuum-assisted biopsy device and visual effects design using a physically based flame simulation.

Index Terms

design; simulation; direct manipulation; multi-touch

1 Introduction

Simulation (e.g., computational fluid dynamics (CFD), finite element analysis (FEA)) is poised to play an increasingly important role in design processes within engineering, science, entertainment, and art. Unfortunately, despite the clear importance to many disciplines, current frameworks for working with simulations do not complement (and in many cases inhibit) the human process of creative design. For example, in the medical devices industry there is now consensus that simulation-based engineering will be one of the most important keys to future innovation [5]; however, practicing engineers often need to

adopt a completely different design process in order to work with simulation tools. The major problems with today's approaches to simulation-based design are twofold:

1. Simulation tools lack an ability for designers to understand, compare, and make decisions based upon alternatives in a large design space (i.e., a set of many related simulations).
2. Simulation tools require an often distracting focus on boundary conditions, parameters, and other low-level details that in many cases are tangential to the designer's goals and may only be understood by a trained expert in computational methods.

Today, these limitations make simulation more useful as a method for detailed validation of a single design rather than for creative exploration of alternatives throughout the design process.

Outside of engineering, designers in other contexts (e.g., entertainment) face many of the same challenges in trying to appropriately fit physically based simulation into creative processes that rely upon human input. On one end of the spectrum of design tools, traditional paper and pencil as well as quick, sketch-inspired 3D modeling systems provide the immediacy, expressiveness, and human engagement needed to enhance creative design. On the other end of the spectrum, detailed simulations, often run in batch mode, provide one or more sets of physically accurate data. There is little in between in terms of computational tools that can facilitate creative design *together with* new insight from data-driven simulation.

In this paper, we argue that these challenges can in large part be solved through new computer graphics interfaces. Our work shares a common motivation with the early work in Design Galleries for setting simulation parameters for computer graphics and animation [12] and a series of recent results that focus on visualizing large sets (ensembles) of simulation results (e.g., [3, 13, 23]).

We introduce a direct manipulation interface we call Design by Dragging. Our goal is to enable the designer to focus entirely on the design problem, not the underlying details of simulations, and to react naturally to visualizations of simulation results. For the medical device engineer exploring the space of possible biopsy device designs (Figure 1), this means that he will sometimes want to drive the exploration in a specific direction by adjusting a geometric parameter of the device (e.g., changing the length of the tissue cutting window as in Figure 1 left). We call this "forward design". In addition, he will also want to react to the visualization of simulation results; he might ask, "What causes this region of high stress, and what would it take to move this region to another location or remove it entirely?" (Figure 1 right) We call this "inverse design" since the path it implies through the design space is specified in terms of simulation outputs rather than inputs.

One of the defining features of the interface is that it supports direct manipulation for both forward and inverse design strategies within the same interactive computer graphics display. For forward design, we accomplish this through interfaces in the style of successful direct manipulation 3D modeling tools (e.g., [25]). For inverse design, we navigate and interpolate

within a set of pre-computed simulations, displaying the best match based on the user's input. To make the user interaction feel as-direct-as-possible, we introduce the idea of directly manipulating data visualizations, such as scalar stress fields output from simulations, in order to create a continuous descriptive visual query. The style of the direct manipulation is reminiscent of recent shape manipulation interfaces based on dragging or "tugging" metaphors [6, 9]. The interface is useful in a mouse-based mode, but it is also particularly interesting when using a multi-point input device, such as a multi-touch display, which enables the user to specify rich visual queries that include global as well as local changes to the data.

The contributions of this paper include:

- A new accessible approach to simulation-based design for tuning parameters of computationally-intensive simulations and visually evaluating large sets of results.
- A strategy and data structure to establish correspondences between arbitrary scalar fields output from FEA and CFD simulations and morph between these fields.
- Algorithms for interpreting the intent of users' drag operations relative to parameterized models and simulation outputs.
- An in-place interactive data visualization for large sets of related simulation results.
- An extension of the core Design by Dragging interface that uses multi-touch input in combination with an as-rigid-as-possible shape manipulation to support rich visual queries.
- A long-term focused evaluation of the approach via an application to engineering design of a mechanical biopsy device in collaboration with expert mechanical engineers.
- A second, more exploratory, application to visual effects design using a physically based flame simulation to demonstrate how the approach can work with more irregular simulation outputs.

2 Related Work

Our work builds upon recent results in direct manipulation interfaces, simulation-based design tools, and ensemble visualization.

2.1 Direct Manipulation Interfaces

We employ a natural, direct manipulation user interface for modifying geometric models included in the simulations. Our approach is inspired by Sketch [25] and other more recent direct manipulation modeling interfaces, such as iWires [6]. Several interfaces (iWires and others [8, 9]) include a real-time simulation and/or constraint engine to add physical realism to direct inputs. These fit seamlessly into our framework. For inverse design, we utilize an As-Rigid-As-Possible Shape Manipulation algorithm together with a multi-touch interface to enable the user to specify rich visual queries more quickly and directly than can be accomplished with a single-point input device.

As a major focus of our work is inverse design, we are particularly interested in interfaces that control (or feel as though they control) an inverse process. A key challenge in creating such interfaces is balancing the goal of providing a direct, expressive interface to the user with the goal of steering the direct manipulation toward a valid configuration. Although they do not include simulation, some previous systems provide powerful examples of this idea. In Accessible Animation [15], the user navigates through a space bounded by valid keyframes either by using a control space widget (forward direction) or, in a more exciting way, by "tugging" directly on the animation (inverse direction). During a tug (similar to our inverse dragging), the system interprets the users intent and morphs from one valid configuration to the next most logical choice. The new configuration may imply a change to more than one aspect of the animation. For example, if the user tugs on the head of a character to turn the character around on the page, this could also imply a change in the characters feet, which may also need to turn to maintain a valid pose. A similar approach has shown promise for navigating through a space of registered car models simply by "tugging" on line drawings of the cars [18]. An inverse-feeling direct manipulation interface has also been used for browsing videos [4].

When these interfaces work correctly, they can feel almost magical to the user – there is no more direct way to query a video dataset for the time when a car drives out of a parking lot than to simply grab the picture of the car in the video and drag it along. We believe one of our most important contributions is achieving this sensation for users working with simulation datasets, including FEA or CFD data used by engineers in real design problems.

2.2 Design with Simulations

Our work builds upon the motivation set forth in early work on Design Galleries, which introduced the general approach of intelligently sampling a design space via pre-computed simulations and then presenting the results to the user in the form of a gallery of keyframes [12]. The simulations that best fit this approach are defined by two characteristics, which are shared by our applications: "high computational cost" and "unquantifiable output qualities". Together, these imply that the simulation-based design problem cannot be solved via a computational optimization; rather, it requires human input to the design, and likely, human visual analysis of intermediate results.

Many-Worlds [21], a system for trajectory-based character animation, also shares a similar motivation, but different approach and application. We target dense, continuous datasets; thus, our preview bubbles and visualizations-via-morphing must provide much more information than simple trajectories. Also, our goal configurations are less well defined, often requiring a trained engineer to interpret. Other approaches to design with simulations include refining and accelerating the simulation to the point that it can accept real-time inputs [2], using real-time simulation to provide suggestions of valid alternatives (e.g., in furniture design) [22], and developing simulation engines that can work toward a desired output based upon a detailed artistic specification (e.g., 3D curves created in Maya to show the direction a simulated flame should take) [1]. Our work differs from these examples in that it can be combined with existing simulation packages (e.g., Abaqus, Maya) and

provides a new level of directness for user interactions. Our intended contribution is not in real-time simulation, but the approach could be applied to simulations that run in real time.

2.3 Ensemble Visualization

The problem of visualizing sets of simulation results is an active area of research with recent applications to engineering design [13, 16], visual effects [3], and disaster management [17, 23]. These systems build upon a design gallery approach and provide a number of significant enhancements to the visualization including clustering, linked views, interactive brushing, timeline widgets, and queries by example. Many of these features could be combined with and complement our approach; however, our goal is to explore an alternative form of ensemble visualization with direct manipulation at its core. We argue that the directness of the interface makes it possible to do effective ensemble visualization in-place, that is, using essentially a single-view interaction and visualization space rather than visual layouts based on small multiples and multiple linked views.

3 Overview, Example Problem, and Definitions

We begin by providing a more detailed description of the example design problem introduced in Figure 1, which is used throughout the paper to explain the Design by Dragging interface. There are two preconditions for applying Design by Dragging to a new simulation-based design problem. We require:

1. A parameterized model that serves as input to the simulations.
2. A distance metric and morphing function that can be defined for each pair of simulation outputs within the design space.

Example parameterized model

The medical device pictured in Figures 1 and 2 is a mechanical system for performing biopsy surgeries. The distal end of the device has a sharp point and a tissue collection window near the tip. There is also an inner cutting cannula that has a sharpened distal edge that is intended to slice though tissue once vacuum pressure has pulled it through the collection window. Two motors located inside the hand piece drive the inner cannula so that it both rotates and translates as it moves across the collection window. The most important geometric parameters of the device are labeled in Figure 2. In Design by Dragging, all of the variation within the design space must be captured by the parameters specified for the model.

Configuration

We can visualize the device geometric parameters along with additional simulation input parameters, such as the location and magnitude of a load applied to the cannula, using a wheel plot (Figure 2 right). Each spoke of the wheel corresponds to a single parameter, with the minimum value at the center and the maximum at the extent. The red polyline that passes through each spoke indicates a unique configuration – a value has been set for each of the design parameters in the model. In our examples, these values are always scalar, and we normalize them based on the min and max for each parameter so that the values always

range from 0 to 1. With this strategy, a configuration can be stored simply as an *n*-dimensional vector of scalar values, where *n* is the number of parameters in the model.

Distance between configurations

Since each configuration can be stored as a vector, a convenient metric for the distance between two configurations is simply an Euclidean distance. By default, each parameter is weighted equally in the distance calculation, but it is possible to change the weighting using the interface described in Section 6. Thus, given any two configurations, *A* and *B*, each stored as an *n*-dimensional vector, the distance between the configurations is computed as

$$d(A,B)=\sum_{i=1}^{n}\sqrt{w_i*(A_i-B_i)^2}, \quad (1)$$

where the weight of the *i*-th parameter in the model, w_i, defaults to 1.

Design space

The design space is the space of all possible configurations. Consider running a FEA or multi-physics simulation to evaluate the utility of a particular configuration. This would provide data for just one of an infinite number of possible configurations. What designers would like to do is intelligently fill in this large space, discovering through this process cross-interactions between the parameters as well as how the parameters impact outputs.

System overview

Figure 3 provides an overview of the system. After inputting a parameterized model, the next step in Design by Dragging (diagrammed in boxes (b) and (c)) is to pre-compute a large number of simulations to sample the design space for a variety of different configurations. The information required to perform a visual morph between each pair of simulation outputs is also pre-computed at this stage. Boxes (d) and (e) represent the real-time, interactive loop that enables both forward and inverse direct manipulation with the design. This real-time interaction is our main contribution and is described next in Sections 4–6.

4 Direct Manipulation of Simulation Inputs (Forward Direction)

The first form of direct manipulation supported by Design by Dragging is forward design. This enables users to drive the design in a new direction by clicking and dragging in an appropriate location on the parameterized model to specify alternative values for simulation input parameters. To enable this interaction, the system must be able to interpret the intent of users' drag operations, and this requires a mapping from features of the parameterized model to simulation input parameters. Features can be defined differently depending upon the application. We find it is useful to think about a feature as a face, edge, cutout, or other geometric property of a model. For input parameters that do not have a natural direct mapping to the model geometry, it is often possible to introduce a proxy geometry, for example, a red arrow is used to indicate the location and direction of a load applied to the geometry during simulation.

Figure 4 shows the mapping from features to simulation input parameters for the biopsy device example. Each feature is associated with one or more simulation input parameters that can be manipulated interactively. To discover the user's intent, the system first identifies the feature upon which the user has clicked using raycasting. Then, once the user's cursor movement exceeds a threshold (e.g., 10 pixels) away from the initial click, the parameter to manipulate is set by finding the closest match between the user's motion vector and the ideal motion vectors associated with each parameter as illustrated in Figure 4. For example, after clicking on the outer cannula of the biopsy device, the user might drag along the direction of the axis of the cylinder to lengthen or shorten it, implying a change in the "outer cannula length", or he might drag perpendicular to this axis to imply a change in the "outer cannula outer radius". Once the input parameter is identified the display begins to update continuously in response to the dragging motion, as illustrated in Figure 1a–b and the accompanying video. The current implementation supports a variety of geometric manipulations (e.g., changing a length, radius, scale, or offset) for features in the model. The features are currently identified manually for each new design problem.

Internally, the system always maintains a notion of a "starting configuration", which is the configuration that is displayed at the beginning of a user interaction. After the parameter that the user is dragging has been identified, the system then calculates a "target configuration", which is the closest configuration (based on the Euclidian distance metric mentioned above) for which the value of the parameter being dragged matches the direction of the user's movement. For example, if the value of the outer cannula length parameter is 80mm in the starting configuration and the user drags in the direction that implies increasing this length, the system will identify the closest configuration with a outer parameter length that is greater than 80mm and mark this as the target. Let's assume it finds a configuration with length 82mm. The display will update as the user drags to morph between the starting configuration and the target configuration, so when the user's cursor position indicates a length of 81mm, the visualization will display a morph that is 50% of the way between the starting configuration and the target configuration. It is important to note that this interpolated visualization involves not just an interpolation of the parameter that is being actively modified but also all other parameters (both simulation inputs and outputs). When the user's cursor finally reaches the position of the target configuration, then this becomes the new "starting" configuration and drag operations can continue from this point, even without releasing the mouse button.

In some extreme cases, it is possible that the user's input might imply a large change to the geometry of the model. To account for this, we employ a proxy model for these direct manipulations. The proxy model is integrated into the callout window that displays a zoomed-in view of the visualization (Figure 1). This proxy model includes a notion of "settling" – when the user completes an interaction, the proxy model gradually morphs into the current working model to rectify any changes.

In the mouse-based implementation, all of these forward manipulations are controlled using mouse click and drag events within the zoomed-in view of the results, and inverse manipulations (described next) are controlled via right mouse clicks. Unicam-style camera

manipulations [25] are controlled using clicks and drags with either button within the main visualization window.

5 Direct Manipulation of Simulation Results (Inverse Direction)

The second method of direct manipulation design in Design by Dragging is to tug directly on the visualization of simulation results to navigate within the design space. As introduced in Related Work, this aspect of the interface is inspired by systems that exhibit an almost magical sense of direct manipulation. We introduce a similar style of as-direct-as-possible interaction, but using direct user manipulations on top of data visualizations.

5.1 Transitioning Between Configurations

Our goal is to smoothly transition from one valid configuration to the next most logical choice as the user interactively drags his cursor on top of the data visualization to "tug" it in a new direction. The strategy we introduce is to compare the motion of the user's cursor on top of the data visualization to the visual motion the visualization would undergo if it were to morph from the current configuration to another configuration. As the user drags the cursor, the algorithm repeatedly calculates the closest match between the user's motion and motion implied by the visual morph needed to reach possible target configurations. The best target is identified, and the visualization updates in real time as the user continues to drag.

The key to enabling this interface is knowing the transition from each simulation output to the others. In both of our example applications, the simulation outputs are 2D scalar fields (in the case of the biopsy device the 2D field is wrapped around the cannula of the device). If we think of the 2D scalar fields as grayscale images, then what is needed is an image warp, as commonly used in morphs for movies, games, and other computer graphics applications.

Given two images *A* and *B*, an image warp can be thought of as a function $f_{A \to B}$ that maps each 2D pixel location *p* in an original image to a new location p' in a second image, such that the transition between image *A* and *B* looks as smooth and natural as possible. We can write this as

$$p' = f_{A \to B}(p). \quad (2)$$

Note that image warping algorithms compute *f* at *every pixel location* in the image. Thus, *f* tells us where every pixel in image *A* will move during a smooth visual transition from *A* to *B*.

Figure 5 diagrams the algorithm for using image warps to transition between configurations. Moving from left to right within the figure, in (a), the user has just clicked on a point of interest within the 3D visualization. From (a) to (b), the location of the cursor has been projected from the 3D geometry into to same 2D image space where the scalar field warps have been computed. The image space pixel that is closest to cursor's position is then identified and labeled *p*. This is the cursor's pixel position at the *start* of the drag operation. At (c), the database of pre-computed image warps is used to evaluate the warping function at

pixel p for every possible target configuration. In other words, given the current starting configuration, A, the function $f_{A \to B}(p)$ is evaluated for all possible target configurations B_1, B_2, B_3, … within the design space. At (d), notice that this results in a set of new pixel locations p_1', p_2', p_3', … that describe how the pixel under the user's cursor would move according to each possible image warp. Continuing with the user interaction, at (e), the user has begun to drag the cursor, and the cursor's movement vector (shown in green) is projected into the image space. The best possible target configuration (highlighted in yellow) is identified using a formula that compares the green cursor movement vector to the red vectors shown in (d). Finally, at (f), notice that as the user continues to drag the cursor toward the target p', the visualization updates so that when the cursor has reached the target p', the data display has morphed to display the data for the target configuration.

There are some important aspects of the formula used at stage (f) in Figure 5. In addition to picking a target configuration for which p' is in line with the cursor's motion vector, it is also important to give preference to target configurations that are close to (in the local neighborhood of) the starting configuration. To make sure this is the case, the target configuration is selected by finding the configuration for which the dot product of the two vectors pictured in Figure 5 is greatest *and* the "distance" (calculated using Equation 1) between the starting configuration and the possible target configuration is smallest. Each of the two considerations is weighted equally.

The algorithm is also tuned experimentally to include some hysteresis. During a drag operation, once a target configuration has been identified using the strategy above, the user must make a definitive move toward another configuration in order to snap the interface away from the current target and orient toward a new target.

The result of this strategy is an interface with immediate feedback. The user can drag toward one target, see via the morph what this would do to the simulation outputs, then, if he wishes, change his mind, back up, and drag in another direction. As in forward dragging, once the cursor comes within a reasonable radius of the target, the target configuration becomes the new starting configuration and the algorithm proceeds fluidly. When using a mouse the button does not need to be released and re-clicked to continue dragging from this new starting point, so from the user's standpoint, the transition is seamless.

5.2 Single Cursor Manipulations

When interacting using a mouse or single touch, Design by Dragging makes use of a technique we call Preview Bubbles to provide a view of many possible design alternatives before beginning to drag. The benefit of these previews is that they provide an immediate visual suggestion of possible dragging operations, something that can encourage exploration of the design space and, perhaps, suggest new alternatives to the designer. The bubbles are only displayed in the callout window so that an uncluttered visualization of the output field is always visible in the main display area.

The preview bubbles fade into view immediately after the initial cursor down event to provide an in-place visualization of how the output scalar field would change if the user dragged in the direction of the bubble and released the cursor at the its center. The algorithm

first identifies possible target configurations as described above, then ranks them based upon the Euclidian distance metric, with the closest target configurations first. Preview bubbles are then placed on the screen in the order of the ranked list. For each target configuration, a bubble is placed at the screen space projection of its p' point as long as the bubble would not overlap with any bubbles that have already been placed. The specific number and size of the bubbles is adjustable via a parameter that we have set empirically for our applications.

Figure 6 shows an example of the interface, including the visual feedback during the drag operation. Notice that the target bubble enlarges and then gradually fades from the view as the user's cursor moves toward it and the underlying visualization morphs into a complete display of the target configuration. When the cursor reaches a position within a small threshold of the center of a bubble, the display snaps to the new target configuration, and the interface is restarted with a new set of preview bubbles. One interesting aspect of the interface is that since the radius of the bubbles is constant in screen space coordinates, the user can adjust the number of bubbles to display by zooming in or out, as shown in the accompanying video.

5.3 Multi-Cursor Manipulations

Multi-cursor (i.e., multi-touch) manipulations are most useful for global adjustments to the simulation outputs. Multiple cursors provide a richer input stream that enables more complex manipulations of the output space. Since these manipulations operate over a larger area, it is important to have specific feedback on how the control mesh that covers this area is moves, and this is the key visual feedback shown to the user within the interface. Preview bubbles are not used in this case; they would be confusing here because a separate set of preview bubbles would be needed for each cursor.

The manipulation begins with an initial touch and release on the scalar field visualization displayed in the callout window. The scalar value at this location is used to extract an iso-contour from the scalar field. The inside of the largest connected component of the contour is then triangulated to create a control mesh, as shown in Figure 7. This mesh is manipulated via natural multi-touch interactions using the as-rigid-as-possible shape manipulation algorithm [7, 9], which uses a real-time optimization to intuitively preserve the shape of the mesh given the deformation constraints implied by each of the control points at the user's fingers. As usual, during the manipulation, the system continuously updates and displays an interpolated result.

During multi-touch manipulations the target configuration is selected using the same strategy of comparing the cursor's movement during the drag operation to the image warps. The difference is that rather than a single cursor with a single motion vector, each vertex of the control mesh is treated as a virtual cursor. So, if there are 100 vertices in the control mesh, $f_{A \to B}$ will be evaluated 100 times for each candidate target configuration B and these values will be compared to the 100 motion vectors that describe how each vertex of the control mesh moved when the the as-rigid-as-possible algorithm was applied.

The result of this strategy is that with a few fingers the user can specify a much more sophisticated change in the shape of the scalar field than can be done with a single mouse

cursor. This change is propagated intuitively beyond the reach of the fingers because the as-rigid-as-possible algorithm makes the control mesh deform in an intuitive manner. Since the target configuration is selected based on the motion of the entire control mesh, this process can take into account even global changes to the scalar field. This gives the user a powerful way to quickly navigate the design space.

6 Additional Widgets for Guiding the Interface

Figure 8 shows an additional widget that can be used to guide the interface. Following Bruckner and Möller [3], we visualize the simulation parameter space as a wheel, with one spoke per parameter. Beyond this, we add an interactive visualization of the local neighborhood of the design space – when the user first begins a dragging operation, a set of black polylines are displayed to indicate the closest neighboring configurations. Transparency is applied to make the closest configurations stand out and those that are farther away fade to the background. Likewise, the history of the design space exploration is also visualized using transparent polylines; this time in red. Small dots along the axes are placed at every value for which a sample exists in the design space, and tick marks are used for parameters that take only discrete rather than continuous values, for example, wire that comes in only certain gauges.

The weights applied to each design parameter can be changed using a tab widget at the end of each spoke. Each tab has three quick states that set the values for w_i used in Equation 1. The states are: pinned (the weight is infinite, so the parameter will not vary during any interaction), normal weighting (the parameter is weighted 1, equal to all others), and free (the parameter has zero weight; it might vary dramatically even in results that are considered close to each other). Alternatively, a small slider at the end of each tab can be activated via a double-click, and a weight between 0.0 and 1.0 can be assigned using the slider. In practice, non-uniform weights are used most often in the quickset mode to limit the design space for constrained exploration; for example, by pinning the parameters for the load applied to biopsy device, the designer can explore the subset of design possibilities that have been simulated for a specific loading condition.

As shown in the accompanying video, two widgets in this style are displayed in the interface, one for input parameters and one for output parameters. Just as for inputs, the designer can pin any scalar valued output parameter (e.g., total power consumed) to limit exploration to a subset of the design space.

7 Morphing Between Simulation Outputs

The final important capability required to create the Design by Dragging interface is a mechanism for morphing between simulation outputs. As mentioned earlier, a 2D image morph is the right choice for our examples. Recall from Figure 3 that these morphs are calculated during a pre-computation step. A variety of algorithms could be used for this purpose, and application-specific algorithms are possible, for example, a warping algorithm has been developed to work specifically with imagery of fire [27]. Our implementation uses a single algorithm for both of the applications. Since the simulation outputs for both applications can be represented as 2D scalar fields, we encode the scalar fields as grayscale

images and use the elastix framework to perform a non-rigid warping on these images [10, 11].

The user's dragging movement could trigger a transition from a starting configuration to any of the other sampled configurations in the design space. So, by default the algorithm computes warps between every pair of configurations. If we view each configuration as a node in a graph and the edges as warps, then this would imply a fully connected graph. To reduce the amount of computation required, some of our examples use a random regular graph rather than a fully-connected graph. The random regular graph is ideal for this purpose because it ensures that each node is connected via a bounded max path length, and this property scales well even with low degree graphs. By computing warps only along the edges of this graph it is possible to reach every node, with the caveat that multiple warps may need to be composed together to do so.

8 Application to Medical Device Engineering

The research described here has been performed collaboratively with a team of mechanical engineers who focus on medical device development. The senior mechanical engineer on our team has more than 40 years experience in mechanical engineering, and has consulted for more than 50 companies on mechanical and product design. The specific biopsy device application described here is a real problem from his past experience; thus, we are confident that it is accurately representative of the types of design problems that engineers struggle to address using today's current design tools. The interface and application have been iteratively evaluated by these engineers in regular weekly design meetings for more than 2 years. In this section, we report on results and evaluative feedback from this work. Our goal has not necessarily been to create a radically improved design for this specific device but rather to demonstrate using a familiar problem the potential to design in a way that is fundamentally different than current approaches.

Balancing the tradeoffs inherent to designing the mechanical biopsy device is challenging. The problem cannot be solved via a strict optimization routine, and the design space is too large to sample exhaustively. In our specific examples, simulation is used to explore scenarios where a load is applied perpendicular to the main axis of the device. This is an important case to consider because during a breast biopsy operation a doctor may apply such a load by pushing the device against the ribcage in order to insert the device into difficult-to-reach regions. If the load is too great, the cutting action or another aspect of the design may fail. On the other hand, if the cannulas and/or motors are strengthened, the device may become too obtrusive, leading to a more invasive procedure, a device that is difficult for the surgeon to balance in his hand, or other problems. Our goal is to enable medical device designers to gain a better understanding of these tradeoffs and make better, more informed design decisions.

8.1 Application-Specific Extensions

Several extensions to the core Design by Dragging interface can be seen in this application. In addition to geometric input parameters, users can also set input parameters that are defined as a function. The motor load profile and tissue properties are two examples. Each

of these is associated with a proxy geometry in the model (the motor casing or a sphere shape drawn within the cutting window for tissue). When the cursor hovers over these an interactive X–Y plot is displayed, and the user can change the load profile or tissue property curves simply by clicking on and dragging each curve (see accompanying video).

Another extension evident in this application is wrapping the FEA output field around the 3D geometry of the device. During the warp calculations, the scalar field for these examples is mirrored and tiled on both sides of the dimension that is wrapped around the cylinder to create a larger image so that the warps are computed smoothly across the edges of the original scalar field. The image is then cropped down to the original size before texture mapping it onto the cylinder for visualization. During drag operations, drags that cross the texture boundary are identified and handled as a special case.

8.2 Data and Results

Two datasets were generated for this application; both relate to analyzing the stresses on the outer cannula during the loading condition mentioned above. Dataset 1 is an approximate set of solutions for normal stress calculated simply using a beam equation that does not account for the cutting window. Dataset 2 is a set of FEA solutions based on the actual cannula geometry computed using a tetrahedral meshing and the von Mises stress field output from the Dassault Systèmes Abaqus solver. These data are of sufficient resolution/precision to perform detailed design analysis. Both datasets contain results for a random sampling of valid configurations based on the following input parameters for the outer cannula: inner radius, outer radius, length, cutting window position, and cutting window length. Some additional statistics describing the datasets are presented in Table 1. In creating these datasets, there is an obvious tradeoff between the pre-computation time required and the size of the dataset, including completeness of the graph data structure used. In both cases, we decided we were willing to wait about 1 day to compute the data, so we used this as a general guideline for setting the number of configurations to sample and the degree of the regular random graph. Each set of warps took roughly 24 hours to compute on a Linux machine with 24 cores.

One reason for working with the first dataset based on simple beam equations is that the scalar fields that it produces have simple patterns that are easily understood; thus, we have been able to use this data to verify that the inverse dragging interface works as intended. This dataset is pictured in Figure 6. Notice that in part (a) of this figure, the user has clicked right on a contour edge between the yellow and green regions of the scalar field. As he begins to drag and the preview bubbles appear, he can see a number of possibilities for where he could “drag this edge” in order to arrive at a new configuration with a different scalar field.

Another reason for working with these two datasets that was suggested by our engineering colleagues is that this situation closely matches the design process that engineers follow. Designers often work by beginning with a broad exploration of a design space and then narrow in to explore the most promising areas in more detail. Since approximate solutions, such as beam equations, can often be computed in real time, Design by Dragging could be easily extended to begin with a first phase of real time exploration of a densely sampled

design space using approximate solutions. Then, once some areas of interest have been identified, these could be examined in more detail using more refined and more computationally expensive simulations.

Overall, the accompanying video provides the best documentation of the success of the application. The interface enables designers to fluidly navigate through the design space, exploring a variety of “what if” scenarios as a natural consequence of viewing data visualizations.

8.3 Domain Expert Evaluation

Since medical device engineers have been integral in the development and evaluation of the tool, we are confident that this approach not only matches the mindset they bring to this design problem but also fills a gap in current tools for working with simulations. The immediacy and directness is fundamentally different than approaches available in commercial and research tools, which do not include direct manipulation, inverse design, and morphs between simulation results.

Specifically, the engineering team members report: “The current practice for this type of problem shies away from using computational tools as they are not efficiently connected [for example, CAD design, simulation, and visualization often require separate software packages] and there is not a way to optimize one or more parameters in the FEA model while at the same time optimizing geometry or material selection parameters... Although there is some inverse functionality beginning to creep into software tools for mechanical design, this approach leapfrogs that. To be able to do inverse [design] for materials, geometry, and FEA is just a huge breakthrough. On top of that you can visualize it.”

A specific point of comparison is with the tools available in current commercial packages for organizing parameter studies. The engineering team members report, “An example of such a tool is iSight (used in the medical device industry because it drives Abaqus). iSight provides quantitative analysis tools and a FEA execution engine. Although the quantitative analysis tools do help gain information for making design decisions, the number of parameters optimized is usually limited to about three. Further, the user still has to go back to the post-processing module of the original FEA package to view the simulation results one by one.”

To our knowledge, no existing tools support Design-by-Dragging’s style of in-place ensemble visualization, where the user can view many results from within the design space simply by dragging in different directions. Further, none support user interactions on top of visualizations of simulation output fields to do either forward or inverse design. Each of these has been evaluated by the collaborating engineers as not just a useful feature but as providing a major breakthrough in the way that they can approach medical device design.

Several aspects of the interface (e.g., integrating 3D modeling with data visualization, depicting design history, presenting many simulation results in the same display) are a direct result of feedback from the domain experts. In terms of criticism of the tool, the current limitation is evaluated as being that the designer cannot make a drastic change in the

geometry of a device unless this can be captured by the design parameters. Thus, using mechanical engineering terminology, the system is most useful for the “detail” design phase rather than the “conceptual” design phase. We believe that it may be possible in the future to couple Design by Dragging with a quick, sketch-based 3D modeling interface and that this could facilitate use for conceptual design as well.

9 Application to Visual Effects

The second application that we explore is design of visual effects. This application does not include the same level of long-term collaborative development and evaluation. Thus, we view it as a secondary, more exploratory application that provides an initial data point for how Design by Dragging might be applied to additional simulation-based design problems outside of engineering. We picked a motivating example of a visual effects artist designing a physically-realistic flame animation to include in a movie or game for two reasons. First, there is prior work in the visualization community on design with pre-computed flame simulations [3]. Second, the scalar fields that result from our flame simulations tend to be much more irregular (less “structured”) than those calculated in the biopsy device FEA simulations; thus, they provide a challenging case to verify the reliability of the dragging operations.

9.1 Data and Results

Two fire simulations were generated using Autodesk Maya’s fluid animator with input parameters including: velocity noise, viscosity, temperature emission, fluid emission, and density emission. Dataset 1 is a set of solutions generated using a point-based flame emitter, as pictured in Figure 7. Dataset 2 contains solutions for a ring of fire (Figure 9). Again, the warps for each dataset were computed in roughly 24 hours on a 24-core Linux machine.

Figure 9 and the accompanying video demonstrate how inverse design is used to explore the flame datasets. Artists can search for a flame of a specific height or with wisps of a particular shape fanning out of the ring.

9.2 Observations and Testing

Because this application is targeted toward artists, it nicely illustrates what we believe is one of most powerful aspects of the approach. Although an artist may not have an intuition for the impact of a simulation input parameter with a name such as “density emission”, she is likely to have a strong intuition for how the shape of a flame should change in order to produce a desired effect. We have observed that the preview bubbles are particularly effective in this regard. In Figure 9, for example, notice that the previews sometimes identify potential target configurations that are significantly different (e.g., have a much higher density of fire) than the starting configuration. As shown in the video, users can rapidly explore these different alternatives by dragging toward and away from the various previews.

Another aspect of the interface that is particularly well suited to this application is the multi-touch inverse design strategy. We believe the reason for this is that there is much more global change captured in the flame simulation datasets, and the multi-touch interface

enables the user to more easily indicate major high-level changes to the output in addition to tweaking the output within a local region.

One useful insight from our iterative design and testing relates to the best way to implement the multi-touch technique using the as-rigid-as-possible shape manipulation algorithm. In early implementations, we applied as-rigid-as-possible shape manipulation to the entire scalar field by using a regular grid over the entire image as a control mesh. To the user, this feels like manipulating a rubber sheet. The current strategy creates a much more compelling and expressive interface by first extracting an iso-curve in the region of interest indicated by the user and then using the shape defined by this curve as the control mesh.

10 Future Work

From our work with practicing medical device engineers, we understand the value of integrating new interfaces like this one into the current workflows utilized by engineers. To this end, we have already developed a real-time, one-way connection between the Design by Dragging software and the Solidworks CAD software that can be used to keep a CAD model in sync with the current state of the Design by Dragging interface. (This is shown briefly in the accompanying video.) In the future, we believe it will also be possible to create a connection in the other direction – given an arbitrary CAD model as a starting point, existing algorithms (e.g., [6, 14]) may provide an excellent starting point for automatically identifying the key geometric parameters that can be adjusted within the model, then this parameterization could be input to Design by Dragging, replacing the manual process used now for box (a) in Figure 3.

We are also excited to extend this work to problems that involve 3D and 4D simulations. The core concepts in the interface should extend seamlessly to volume datasets; the 2D image morphing routine would simply need to be replaced with a 3D version. The existing interface could be directly applied to volume data visualized using slicing planes. Beyond this simple extension, the more interesting research would likely come from refining the user interface to take advantage of 3D input. During inverse design in 3D, we envision that designers might simply reach out with two hands to grab and reorient a streamline from a 3D flow dataset. In fact, we have already created a preliminary 3D implementation of this, which replaces the multi-touch input device with input from two handheld 3D trackers in virtual reality. One reason why we have specifically chosen to work with the as-rigid-as-possible algorithm is that it can extend to reshaping 3D curves and meshes [19, 26]. So, as suggested here, a natural extension of the work would be to reshape 3D flow lines or 3D iso-surfaces extracted from simulation results.

Finally, there are two modules with the framework outlined in Figure 3 that could be pulled be out of our current implementation and replaced with alternatives. The first is the sampling strategy used when pre-computing simulation results. To begin, we have tended to utilize a simple random sampling strategy to populate the design space, but it is clear that a variety of more intelligent strategies could be used, likely with great benefit to the designer. Since several existing algorithms may be directly applicable for this [12, 20, 24], we have not yet made it a focus of our research. Extending beyond current sampling algorithms, there is also

a valuable future research opportunity in making use of the design history recorded by the interactive tool to intelligently spawn new simulations in real time as indicated by box (f) in Figure 3.

11 Conclusion

Design by Dragging introduces a new approach for as-direct-as-possible design with simulations. The system demonstrates how new interface algorithms for both single and multi-cursor environments can be integrated with data visualizations and how user input can be interpreted relative to data to support intelligent inverse queries. In addition to providing new capabilities for design, the tool also provides a new form of in-place interactive ensemble visualization. The examples provided demonstrate the applicability to both engineering design and less constrained artistic design, both supported through a method of calculating correspondences and morphs in a result space of hundreds of 2D scalar fields output from simulations. Perhaps the most important qualitative contribution of the work is the almost “magical” feeling the interface can provide of directly manipulating data, even reshaping it with one’s hands, during inverse design. Because this makes design with simulations so much more accessible to users of varied backgrounds, we believe Design by Dragging has the potential to be applied broadly to many design problems and perhaps open up new application areas for creative design with data-intensive simulations.

Supplementary Material

Refer to Web version on PubMed Central for supplementary material.

Acknowledgments

This work was supported in part by the University of Minnesota Supercomputing Institute and by the Autodesk Research Donation Program.

References

1. Bangalore, A.; House, DH. A technique for art direction of physically based fire simulation. Proceedings of the Eighth Annual Symposium on Computational Aesthetics in Graphics, Visualization, and Imaging; CA. 2012. p. e’12p. 45-54.
2. Barbič J, Sin F, Grinspun E. Interactive editing of deformable simulations. ACM Trans Graph. 2012; 31(4):70:1–70:8.
3. Bruckner S, Moller T. Result-driven exploration of simulation parameter spaces for visual effects design. IEEE Transactions on Visualization and Computer Graphics. 2010; 16(6):1468–1476. [PubMed: 20975188]
4. Dragicevic, P.; Ramos, G.; Bibliowitcz, J.; Nowrouzezahrai, D.; Balakrishnan, R.; Singh, K. Video browsing by direct manipulation. Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI ’08; 2008. p. 237-246.
5. FDA. FDA regulatory science report. 2011. http://www.fda.gov/AboutFDA/CentersOffices/OfficeofMedicalProductsandTobacco/CDRH/CDRHReports/ucm274152.htm
6. Gal R, Sorkine O, Mitra NJ, Cohen-Or D. iWIRES: an analyze-and-edit approach to shape manipulation. ACM Trans Graph. 2009; 28(3):33:1–33:10.
7. Igarashi T, Igarashi Y. Implementing as-rigid-as-possible shape manipulation and surface flattening. J Graphics, GPU, and Game Tools. 2009:17–30.

8. Igarashi T, Mitani J. Apparent layer operations for the manipulation of deformable objects. ACM Trans Graph. 2010; 29:110:1–110:7.
9. Igarashi, T.; Moscovich, T.; Hughes, JF. As-rigid-as-possible shape manipulation. ACM SIGGRAPH 2005 Papers, SIGGRAPH '05; 2005. p. 1134-1141.
10. Klein S, Pluim JP, Staring M, Viergever MA. Adaptive stochastic gradient descent optimisation for image registration. Int J Comput Vision. 2009; 81(3):227–239.
11. Klein S, Staring M, Murphy K, Viergever MA, Pluim JPW. elastix: A toolbox for intensity-based medical image registration. IEEE Transactions on Medical Imaging. 2010:196–205. [PubMed: 19923044]
12. Marks, J.; Andalman, B.; Beardsley, PA.; Freeman, W.; Gibson, S.; Hodgins, J.; Kang, T.; Mirtich, B.; Pfister, H.; Ruml, W.; Ryall, K.; Seims, J.; Shieber, S. Design galleries: a general approach to setting parameters for computer graphics and animation. Proceedings of the 24th annual conference on Computer graphics and interactive techniques, SIGGRAPH'97; 1997. p. 389-400.
13. Matkovic K, Gracanin D, Jelovic M, Ammer A, Lez A, Hauser H. Interactive visual analysis of multiple simulation runs using the simulation model view: Understanding and tuning of an electronic unit injector. IEEE Transactions on Visualization and Computer Graphics. 2010; 16(6): 1449–1457. [PubMed: 20975186]
14. Mitra NJ, Yang Y-L, Yan D-M, Li W, Agrawala M. Illustrating how mechanical assemblies work. ACM Trans Graph. 2010; 29:58:1–58:12.
15. Ngo, T.; Cutrell, D.; Dana, J.; Donald, B.; Loeb, L.; Zhu, S. Accessible animation and customizable graphics via simplicial configuration modeling. Proceedings of the 27th annual conference on Computer graphics and interactive techniques, SIGGRAPH '00; 2000. p. 403-410.
16. Piringer H, Pajer S, Berger W, Teichmann H. Comparative visual analysis of 2d function ensembles. Comp Graph Forum. 2012; 31(3pt3):1195–1204.
17. Ribicic H, Waser J, Gurbat R, Sadransky B, Groller M. Sketching uncertainty into simulations. IEEE Transactions on Visualization and Computer Graphics. 2012; 18(12):2255–2264.
18. Smith R, Pawlicki R, Kókai I, Finger J, Vetter T. Navigating in a shape space of registered models. IEEE Transactions on Visualization and Computer Graphics. 2007; 13(6):1552–1559. [PubMed: 17968109]
19. Sorkine, O.; Alexa, M. As-rigid-as-possible surface modeling. Proceedings of the fifth Eurographics symposium on Geometry processing, SGP '07; Aire-la-Ville, Switzerland, Switzerland: Eurographics Association; 2007. p. 109-116.
20. Torsney-Weir T, Saad A, Moller T, Hege HC, Weber B, Verbavatz JM. Tuner: Principled parameter finding for image segmentation algorithms using visual response surface exploration. IEEE Transactions on Visualization and Computer Graphics. 2011; 17(12):1892–1901. [PubMed: 22034306]
21. Twigg, CD.; James, DL. Many-worlds browsing for control of multibody dynamics. ACM SIGGRAPH 2007 papers, SIGGRAPH '07; New York, NY, USA: ACM; 2007.
22. Umetani N, Igarashi T, Mitra NJ. Guided exploration of physically valid shapes for furniture design. ACM Trans Graph. 2012; 31(4):86:1–86:11.
23. Waser J, Fuchs R, Ribicic H, Schindler B, Bloschl G, Groller E. World lines. IEEE Transactions on Visualization and Computer Graphics. 2010; 16(6):1458–1467. [PubMed: 20975187]
24. Xu K, Zhang H, Cohen-Or D, Chen B. Fit and diverse: set evolution for inspiring 3d shape galleries. ACM Trans Graph. 2012; 31(4):57:1–57:10.
25. Zeleznik, RC.; Herndon, KP.; Hughes, JF. Sketch: an interface for sketching 3d scenes. Proceedings of the 23rd annual conference on Computer graphics and interactive techniques, SIGGRAPH '96; 1996. p. 163-170.
26. Zhang L, Liu L, Gotsman C, Gortler SJ. An as-rigid-as-possible approach to sensor network localization. ACM Trans Sen Netw. Jul; 2010 6(4):35:1–35:21.
27. Zhu L, Yang Y, Haker S, Tannenbaum A. An image morphing technique based on optimal mass preserving mapping. IEEE Transactions on Image Processing. 2007; 16(6):1481–1495. [PubMed: 17547128]

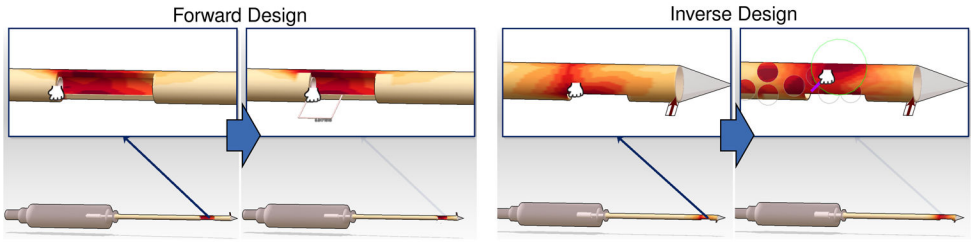

Fig. 1.

In Design by Dragging, designers navigate through a space of hundreds of simulation outputs using direct manipulation interfaces on top of data visualizations. In this example, medical device engineers refine the design of a mechanical biopsy device with two concentric cylinders (cannulas) that slide against each other to cut tissue. In forward design (left), drag operations change the value of simulation inputs, such as the length of the cutting window. In inverse design (right), drag operations directly change the simulation outputs. Users bend and reshape stress fields and the system responds continuously, displaying morphs between the pre-computed simulation results.

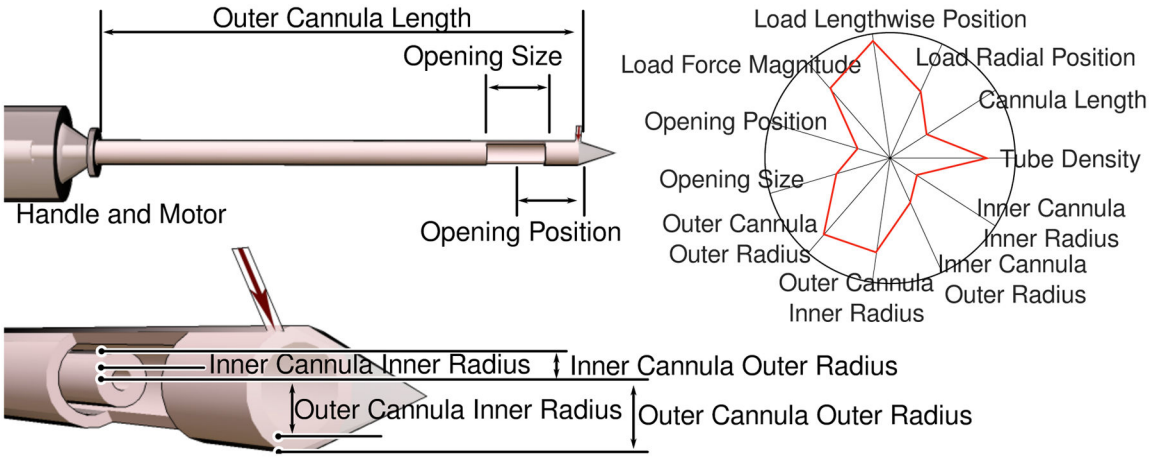

Fig. 2.

Left: A parameterized model that is used as input to Design by Dragging. Right: The design space implied by the parameters with a single device configuration highlighted in red.

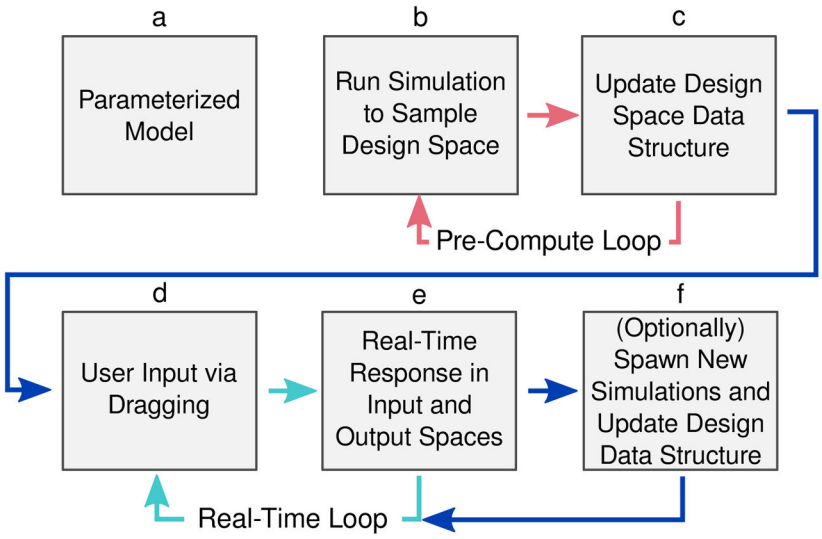

Fig. 3.
Design by Dragging pipeline.

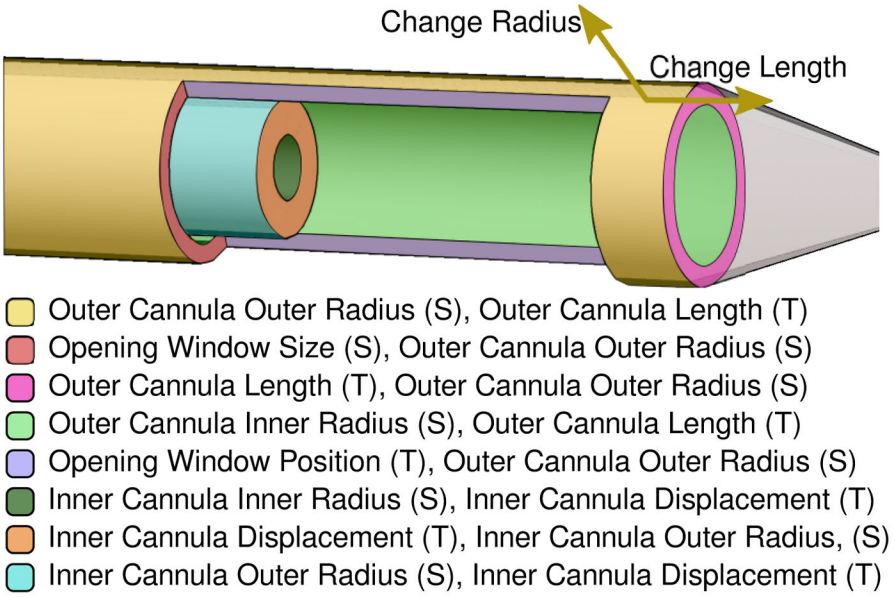

Fig. 4.

Each model has a set of logical features that users can select to manipulate input parameters. Since several parameters might be associated with a single feature, the direction of the dragging motion is used to select the specific parameter to modify (indicated above for a click on the outer cannula).

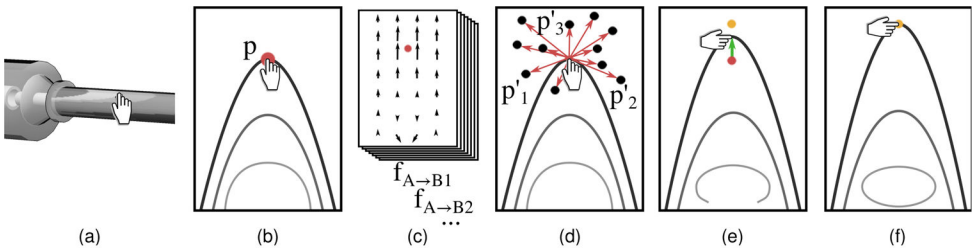

Fig. 5.

The interactive algorithm for transitioning from one configuration to the next as the user drags a cursor consists of a series of steps. The 3D cursor position is first projected into the 2D image space of the simulation scalar output fields. Then, the user's cursor movement is compared to the visual change that would occur in the data visualization during a morph from the starting configuration to each possible target configuration. When the best match is identified, the visualization itself is drawn as a morph controlled interactively by the user's cursor.

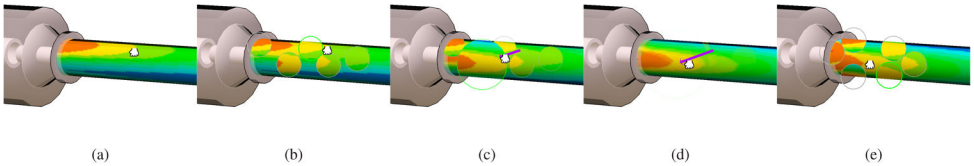

Fig. 6.

Inverse design using a single point manipulation: (a) the user begins by clicking on a point of interest, (b) preview bubbles are displayed, (c) the user drags toward an interesting preview, (d) the new simulation result morphs into place, and (e) a new set of preview bubbles is generated.

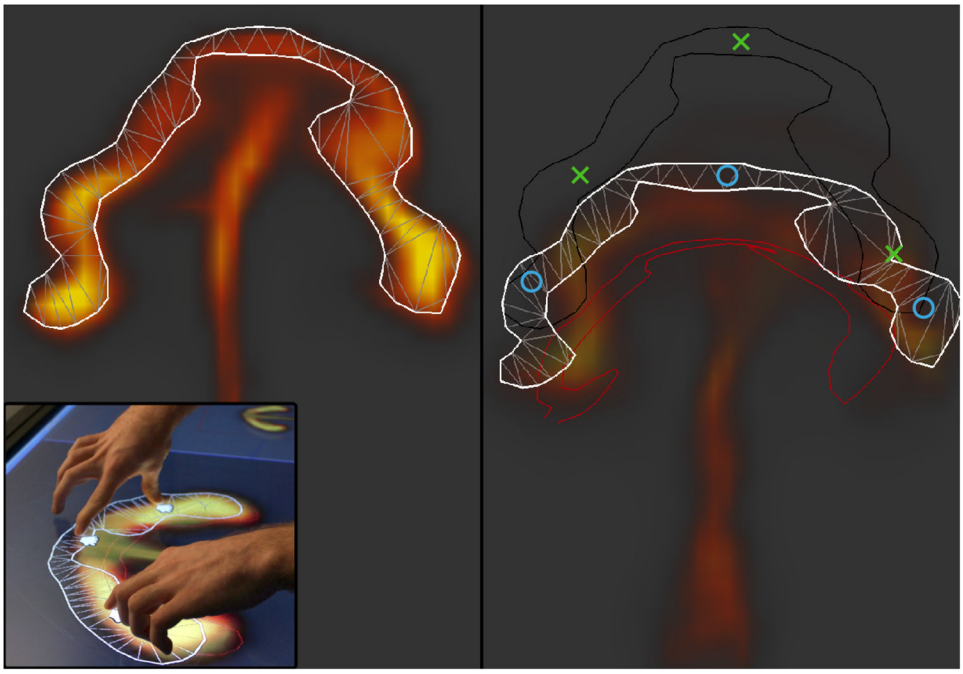

Fig. 7.

Left: In multi-touch inverse design, a control mesh is generated interactively when the user selects an iso-contour of the simulation data. Right: The mesh is reshaped based on simultaneous input from multiple fingers to create a detailed inverse query into the result space. This example comes from the flame simulation described in Section 9.

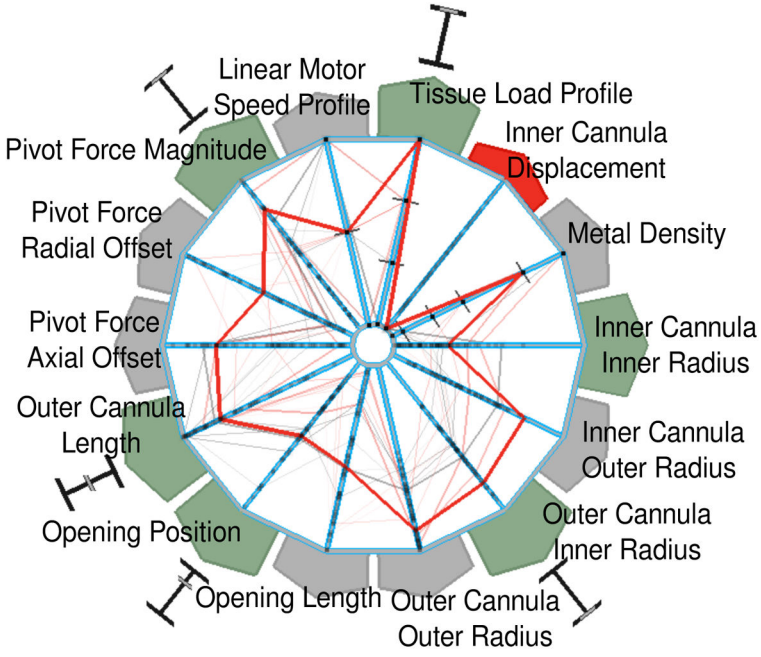

Fig. 8.

A parameter space wheel widget enables designers to guide the interface by pinning or weighting parameters.

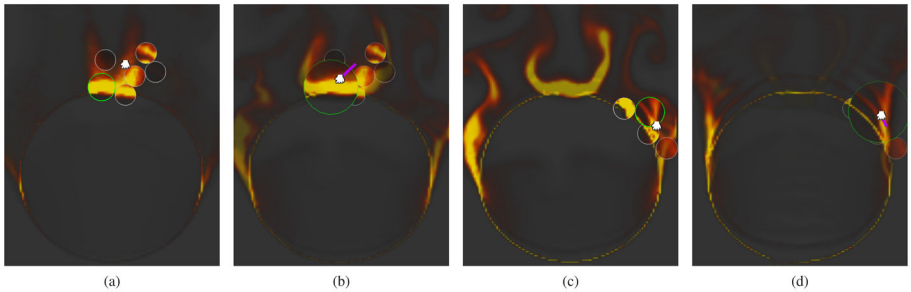

Fig. 9.

Inverse design of a flame effect. (a) Preview bubbles around an in initial click show that the flame shape can be dragged to transform into a more vibrant flame. (b) As the user drags, the new flame expands and morphs into place. (c–d) Clicking and dragging at a different location on top of this new flame allows the user to further refine the shape of the flame.

Table 1

Summary statistics for datasets used in both applications.

Dataset	Simulation Input Parameters	Resolution	Configurations Sampled	Graph Degree	Warps Computed
Biopsy Beam Eq., Normal Stress	14	100×200	1,500	22	33,000
Biopsy FEA, von Mises Stress	14	100×200	200	100	20,000
Flame Point Emitter, Temperature	10	40×60	600 (60×10 timesteps)	100	60,000
Flame Ring, Temperature	12	100×130	240 (30×8 timesteps)	240 (fully connected)	57,600