VizCraft: A Problem-Solving Environment for Aircraft Configuration Design

The VizCraft problem-solving environment aids aircraft designers during conceptual design. It integrates simulation codes that evaluate a design with visualizations for analyzing a design individually or in contrast to other designs.

Visualization has let scientists gain an understanding of their data that was not previously possible. However, lack of integration among the various software modules often separates the visualization process from the computation that generates the data.

VizCraft is a problem-solving environment that aids designers during the configuration design of a high-speed civil transport (HSCT). VizCraft provides a graphical user interface to a widely used suite of simulation and analysis codes for HSCT design,[1] and it provides tools for visualizing the outputs of these codes. So, VizCraft provides an environment that combines visualization and computation, encouraging the designer to think in terms of the overall problem-solving task, not simply using the visualization to view the computation's results.[2]

1521-9615/01/$10.00 © 2001 IEEE

Amit Goel, Chuck A. Baker, Clifford A. Shaffer, Bernard Grossman, William H. Mason, and Layne T. Watson
Virginia Polytechnic Institute and State University
Raphael T. Haftka
University of Florida

The HSCT design problem

We want to minimize the *takeoff gross weight* (TOGW) for a 250-passenger HSCT with a range of 5,500 nautical miles and a Mach 2.4 cruise speed. The simplified mission profile includes takeoff, supersonic cruise, and landing. Typically, aircraft design comprises three distinct phases: conceptual, preliminary, and detailed design. The conceptual-design stage determines and sets major design parameters for the final configuration. It models an aircraft with a set of values for significant parameters relating to the aircraft geometry, internal structure, systems, and mission.

Individual designs can be (and are) viewed as points in a multidimensional design space. The designer must determine that a proposed design point

- is feasible (it satisfies a series of constraints) and
- has a figure of merit determined by an objective function.

The goal is then to find the feasible point with the smallest objective-function value. The multidisciplinary HSCT design problem uses TOGW as the objective function. TOGW is a nonlinear, implicit function of the 29 design variables that define the HSCT configuration and mission.

The HSCT design uses 68 nonlinear inequality constraints, categorized as geometric, aerodynamic, and performance. Geometric constraints ensure feasible aircraft geometries; examples include fuel volume limits and prevention of wing tip strike at landing with 5° roll. Some aerodynamic constraints impose realistic performance and control capabilities; examples include landing angle-of-attack limits, range requirements, and limits on the lift coefficient of the wing sections. Other aerodynamic constraints establish control of the aircraft during adverse flight conditions. These are complicated, nonlinear constraints that require aerodynamic forces and moments, stability and control derivatives, and center of gravity and inertia estimates.

In some respects, this is a classic optimization problem. The goal is to find the point that minimizes an objective function while meeting a series of constraints. However, solving this particular problem is difficult for several reasons.

First, evaluating an individual point to determine its value under the objective function and whether it satisfies the constraints is computationally expensive. A single aerodynamic analysis using a CFD code can take from one-half to several hours, depending on the grid used and flight condition considered.

Second, the presence of numerical noise in the function values inhibits the use of many gradient-based optimization methods. This numerical noise might result in inaccurate calculation of gradients, which in turn slows or prevents convergence during optimization. Or, it might promote convergence to spurious local optima.

Third, the problem's high dimensionality makes it impractical for many approaches that are often applied to difficult optimization problems. For example, genetic algorithms work poorly for this problem, because they require far too many function evaluations just to build a rich enough gene pool from which to begin evolution.

Fourth, the high dimensionality makes it difficult to even think about the problem spatially; most people's intuitions about 2D and 3D space transfer poorly when considering behaviors in ten or more dimensions, or even in four dimensions.

The region enclosed by the bounds on the variables is the design space. Its vertices determine a 29-dimensional hypercube. The problem's high dimensionality makes visualization of the design space difficult because most standard visualization techniques do not apply. In practice, we can hope to evaluate only a small fraction of the points in this design space. Evaluating a single point is expensive, and the number of points is impossibly large. Consider evaluating only the points that represent combinations for the extreme ends of the range in each parameter. In three dimensions, this would be the equivalent of evaluating the eight corners of a cube. In 29 dimensions, $2^{29} \approx 1/2$ billion point evaluations would be required.

The visualization challenge

Motivated by the difficulty and practical significance of the configuration design problem, aircraft designers are searching for both new ways to find better design points and new insights into the nature of the problem itself. Visualization holds some promise for providing insights into the problem by providing new interpretations of available data. Visualization, in conjunction with some form of organization for earlier point evaluations, might also help the designer search meaningfully through the design space.

In some respects, this is a classic optimization problem.

So, the challenge to the visualization community is to devise techniques that help aircraft designers apply their design expertise to the problem and guide the computation during conceptual design. Unfortunately, most traditional techniques for visualizing multidimensional spaces[4] do not apply to this problem.

Because the problem is so big, any attempt to visualize directly the entire space through such methods as time series techniques, animation, or the use of color or sound cannot succeed. (Of course, visualizing relatively low-dimensional sections of the design space might be helpful, as we show later.)

Parallel coordinates[5] and a matrix of 2D scatterplots are two well-known multidimensional visualization techniques. Both allow comparisons of an arbitrary pair of variables but do not help users recognize spatial relationships between points in the *N*-dimensional space. A disadvantage of scatterplots is that obtaining a comprehensive view of the relationships between variables in a high-dimensional space requires too many plots. Researchers have proposed clustering methods that attempt to map similarities in data records from a high-dimensional space into a 2D or 3D space.[6] Unfortunately, what it means for design points to be "similar" is not clear, aside

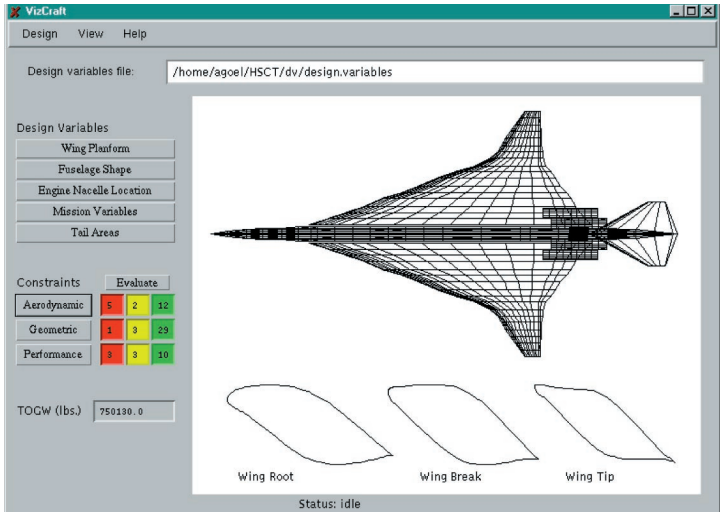


Figure 1. A VizCraft design view window showing aircraft geometry and cross sections of the airfoil at the root, leading edge break, and wing tip.

from such obvious measures as similar objective-function values. Nor is it clear how this approach would provide insight to designers.

It is interesting to compare the aircraft design problem to other, more common problems in multidimensional data analysis. To illustrate this class of problems, consider locating a place to retire. This might involve considering 10 or 20 variables, such as climate, population density, and crime rate. Data analysis for this problem depends on building an objective function that attempts to assign values to each parameter on some linear scale and relative weights to the various parameters.

These two problems have important differences that affect their visualizations. In the retirement problem, for each variable, more (or less) of most parameters is absolutely better (for example, a lower crime rate is absolutely better, regardless of other variables). The number of destinations is effectively fixed, and a point A that differs from point B in only one variable might not exist. In the aircraft design problem, all points in the parameter space are possible for consideration. However, you cannot simply choose the point that independently optimizes each parameter. One reason for this is that the constraints supply an independent limitation on the values of various parameter combinations, so that improving one parameter independently of the others might violate some constraint. More important, there is a nonlinear relationship between the parameters as they affect the objective function in the aircraft design problem. In particular, the objective function is nonmonotonic with respect to many of the design parameters.

First efforts

Researchers have already applied visualization techniques to the aircraft design problem in two ways. In the first method, a point in multidimensional space corresponds to a rough aircraft design. It is useful to the designer to see an iconic representation of the airplane shape that corresponds to a given point, such as Figure 1 illustrates. VizCraft transfers the parametric representation to physical coordinates and stores it in a particular geometry format, which serves as input to several of the analysis methods. It then formats the coordinate set as input to a plotting package.

The second method illustrates the power of even simple visualizations to provide insight into a difficult problem. Figure 2 shows a section of a 2D slice through the multidimensional design space using the points designated as Optimum 1, Optimum 2, and a suboptimal feasible point. This method creates the remaining points by linearly varying the design variables between all three points.

In Figure 2, the circles represent design points. Open circles represent feasible points, and filled circles represent points that have violated some constraints. The shading indicates the objective function's value. In this region of the design space, the objective function is relatively insensitive, resulting in a smooth "surface." This is because all three points selected have similar objective-function values—in the HSCT problem, numerical noise affects mostly the constraints. The curved lines represent the boundaries of four constraints. These lines are generated by interpolating the data achieved from the point evaluations. No simple independent equations exist that we can use to discriminate large sets of points as satisfying or violating an individual constraint, except as gross approximations.

Designers need insight into the shape of the design space in which they work. Knowledge of the constraint boundaries and variations in the objective-function value allows a more informed selection of optimal designs. This 2D slice visualization offers new insight into the design space's properties. The original motivation for its development derives from the results of an automated optimizer applied to the HSCT simulation. The optimizer was sensitive to initial conditions, in that providing one starting point yielded a local optimum, while providing another starting point yielded another local optimum that was 2,000 pounds lighter. Before creating the visualization, designers had not recognized that the constraints break the design

space into disjoint (at least in some hyperplanes) regions of feasible points.

Knowing whether to accept (or reject) what the optimizer tells us is important. Optimizers can have trouble with high-dimensional, highly constrained problems. Visualization in conjunction with optimization can provide understanding of the optimization process and the tradeoffs involved. However, it also sometimes requires guidance by an experienced engineer when the optimizer runs into trouble (such as when the objective function's gradient is nearly perpendicular to a constraint boundary).

Working with VizCraft

In the absence of a better automated technique for solving the problem, aircraft designers would benefit from better visualization tools for helping select better designs. One approach involves a visualization system such as VizCraft, which better manages the available information. VizCraft consists of a pair of tools for visualizing HSCT designs. The first tool lets the user quickly evaluate a given design's quality with respect to its objective function, constraint violations, and graphical view. The second implements the parallel-coordinates visualization; its goal is to let the user investigate effectively a database of designs.

The design point visualization tool

VizCraft provides a menu-driven GUI to the HSCT design code, a collection of C and Fortran routines that calculate the aircraft geometry in 3D, the design constraint values, and the TOGW value, among other things. We chose Java as the programming language for development because VizCraft users needed the ability to execute it from various platforms without concern for user interface library installation issues.

Figure 1 shows VizCraft's main window with a display of the HSCT planform (a top view) for a sample design. Below the planform are cross sections of the airfoil at the root, leading edge break, and wing tip. To make observation easier, the vertical dimension of the cross sections has been magnified. Before we developed VizCraft, designers used Tecplot to display the HSCT planform. We wrote a conversion routine to integrate the planform into VizCraft. This improvement meant that users would not have to run a different application along with VizCraft to view the aircraft geometry. Integration lets engineers shift easily between a design's visual

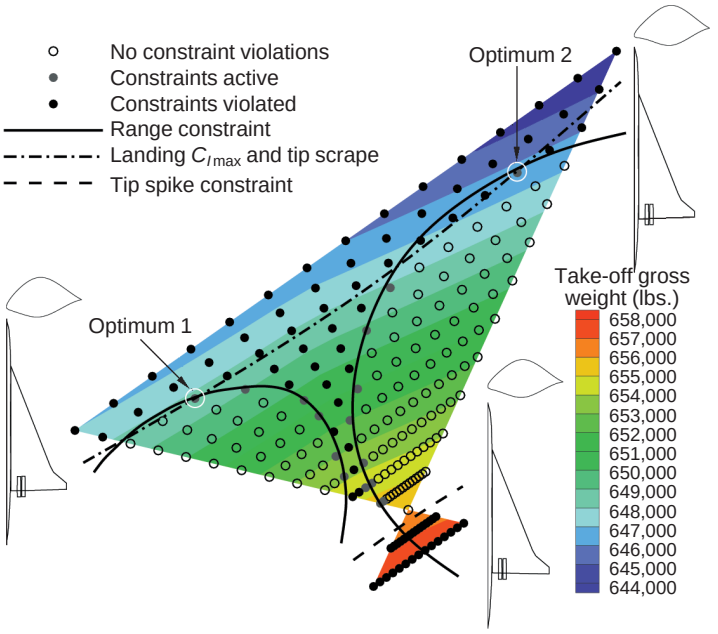


Figure 2. A 2D slice through a multidimensional parameter space using the points Optimum 1 and Optimum 2 and another suboptimal feasible point.

representation and points in the design space. We also added a VRML model of the HSCT planform, accessible from the menu bar. This model gives the user greater flexibility in manipulating the planform in 3D.

The panel to the platform's left in Figure 1 provides access to more information about the current design point. Design variables fall into five categories: wing planform, fuselage shape, engine nacelle location, mission variables, and tail areas. Constraints fall into three categories: geometric, aerodynamic, and performance. Clicking on the Wing Planform button in the main window pops up the window in Figure 3. This window displays the wing parameters and their values. The sliders on the right modify the values. Each modification of a value immediately updates the HSCT planform, and the value of TOGW on the vertical panel, to reflect the new geometry. VizCraft does not automatically evaluate constraints for the current design point after each change to an input parameter, however. Because constraint evaluation is time consuming, even for the low-fidelity model we are using (taking approximately 10 seconds on a dual-processor DEC Alpha 4100 5/400 under typical user loads), VizCraft evaluates constraints only when the user explicitly requests it.

Once VizCraft has evaluated the constraints, it gives the user feedback. In Figure 1, the red boxes indicate the number of violated constraints in that

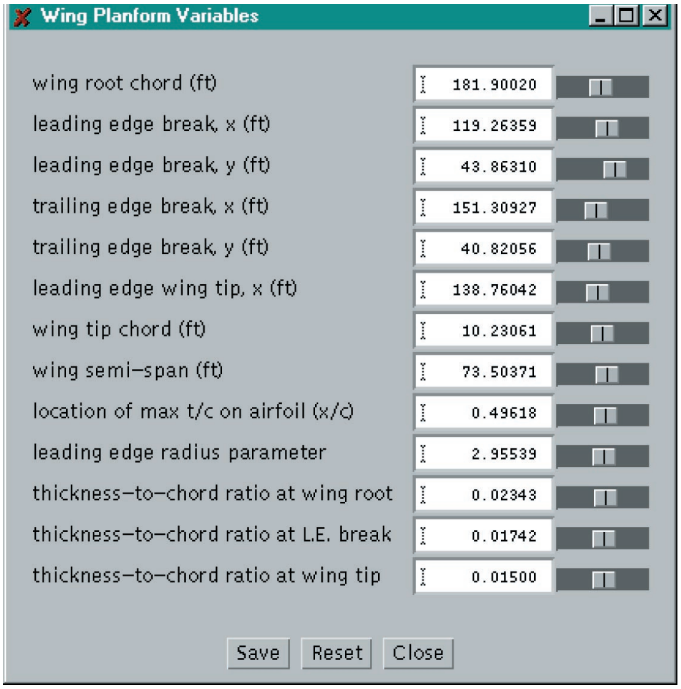


Figure 3. Input window for entering values of wing planform variables.

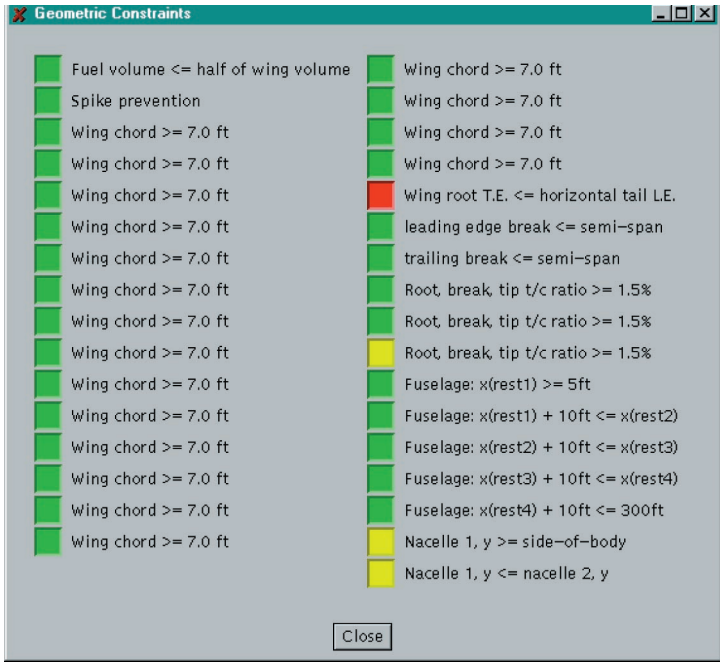


Figure 4. Geometric constraints for one design point. A red box indicates a violated constraint, a yellow box indicates an active constraint, and a green box indicates a satisfied constraint.

category, the yellow boxes indicate the number of "active" (that is, close to a constraint boundary) constraints, and the green boxes indicate the number of satisfied and inactive constraints.

Users can see a detailed list of each constraint category. For example, clicking on the Geometric button under Constraints calls up the window in Figure 4. This window lists the geometric constraints for the current design point; the colored box next to each one indicates whether it is violated (red), active (yellow), or inactive (green).

The parallel-coordinates tool

The tool described in the previous section provides a visualization of the aircraft derived from a given design vector and a convenient view of constraint violations for that vector. However, it does not help designers with the more difficult task of understanding how a proposed design compares with other designs. As discussed earlier, this task is complicated by the problem's high dimensionality and the resulting difficulty in visualizing or comprehending the multidimensional design space.

A parallel-coordinates visualization assigns one vertical axis to each visualization variable and evenly spaces these axes horizontally (see Figure 5a). This method contrasts with the traditional Cartesian coordinate system, in which all axes are mutually perpendicular. By drawing the axes parallel to one another, the user can represent points in many more than three dimensions.

In our application, potential visualization variables (equivalently, dimensions) include the design variables, the objective-function value (TOGW) and other derived values such as range, and the constraint values. VizCraft plots each visualization variable on its own axis and connects the values of the variables on adjacent axes by straight lines, as Figure 5a shows. Thus, a point in an n-dimensional space becomes a polygonal line laid out across the n parallel axes with $n-1$ line segments connecting the n visualization variables. Many such data points (in Euclidean space) will map to many of these polygonal lines in a parallel-coordinate representation. Viewed as a whole, these many lines might well exhibit coherent patterns, which could indicate an inherent correlation of the data points involved. This transforms the search for relations among the design variables into a 2D pattern recognition problem, and the design points become amenable to visualization.

This visualization scheme provides opportunities for human pattern recognition. By using color to distinguish lines and supporting various forms of interaction with the parallel-coordinates system, the scheme makes recognizable patterns

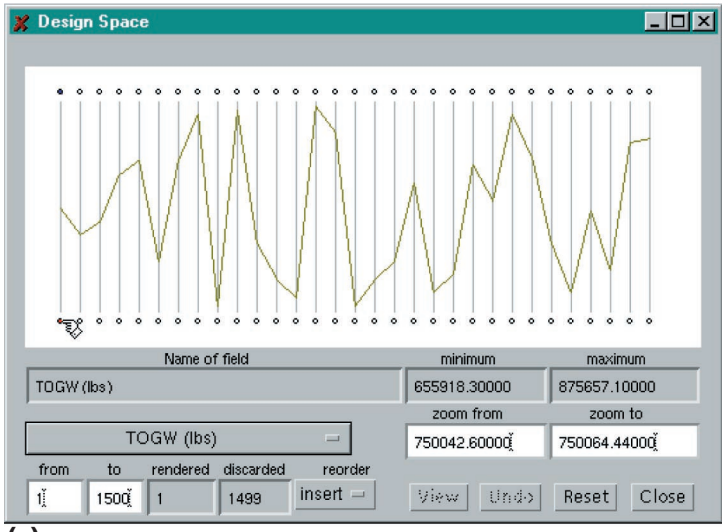
in the given database of design points. Given the upper and lower limits on each variable, the location of a polygonal line laid out across the n vertical axes gives some idea as to where that design point lies in the design space. The number of dimensions that we can visualize using this scheme is fairly large, limited only by the screen's horizontal resolution (although, as the number of dimensions increases, the axes come closer to each other, making it more difficult to perceive patterns).

The parallel-coordinates approach is also flexible, in that each coordinate can be individually scaled. Some might be linear with different bounds, while others might be logarithmic (although currently VizCraft does not support logarithmic scaling). This approach might help identify direct, inverse, and one-to-one relationships between the parameters. Scaling an individual parameter has another advantage, in that it helps us zoom into or out of a subset of the region of design space represented. This lets us effectively "brush out" or eliminate undesirable portions (we'll discuss brushing in more detail later).

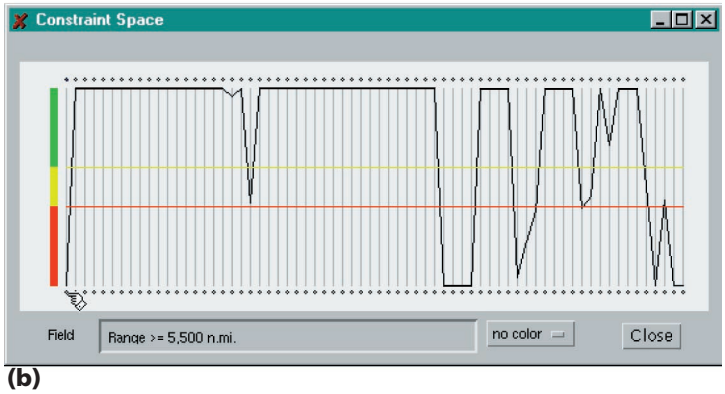
Figure 5a shows 31 values mapped onto 31 vertical axes. Placing the mouse cursor on one of the circles below the vertical lines will cause the Name of Field text field to display a description of the corresponding visualization variable. Text fields display the range and absolute range for the selected variable. VizCraft obtains the absolute ranges for all the design variables automatically by locating their minimum and maximum values from the given database of points.

Figure 5b shows the parallel-coordinates system for 68 constraints corresponding to the design point in Figure 5a. VizCraft normalizes all constraint values, making the range for violated, active, and inactive values consistent across the constraints. Values above the yellow horizontal line indicate inactive constraints, all those between the yellow and red lines indicate active constraints, and all those below the red line indicate violated constraints. By breaking up the range of constraint values into three regions, VizCraft lets the designer easily assess the merit of a given design point. A quick glance at the screen conveys most of the information the designer needs to form a judgment of the current point.

Figure 5b shows how easy it is to graphically identify the inactive and violated constraints and to determine to what degree each constraint has been violated, without having to deal directly with numbers. Representing just one design point in the parallel-coordinates system might help the designer quickly observe the level of constraint violations, but this view is little better than that provided by the single-point VizCraft tool. As I mentioned before, the real purpose of parallel coordinates in VizCraft is to let the designer browse a database of design points.



(a)



(b)

Figure 5. Parallel-coordinates representations: (a) One design point. Each vertical line represents a design parameter. The first line from the left represents the TOGW (take-off gross weight—the objective function), the second line represents the HSCT range, and the remaining 29 lines represent the design variables. (b) 68 constraints for the design point in Figure 5a. Horizontal lines split the vertical lines into three regions: satisfied (green), active (yellow), and violated (red).

We illustrate this process with a database of 1,500 design points selected uniformly from the entire design space. Figure 6a shows the rendered database. From this mass of data, the designer can use VizCraft's visualization controls to extract patterns.

VizCraft assigns each polygonal line (representing one design) a color based on the value of

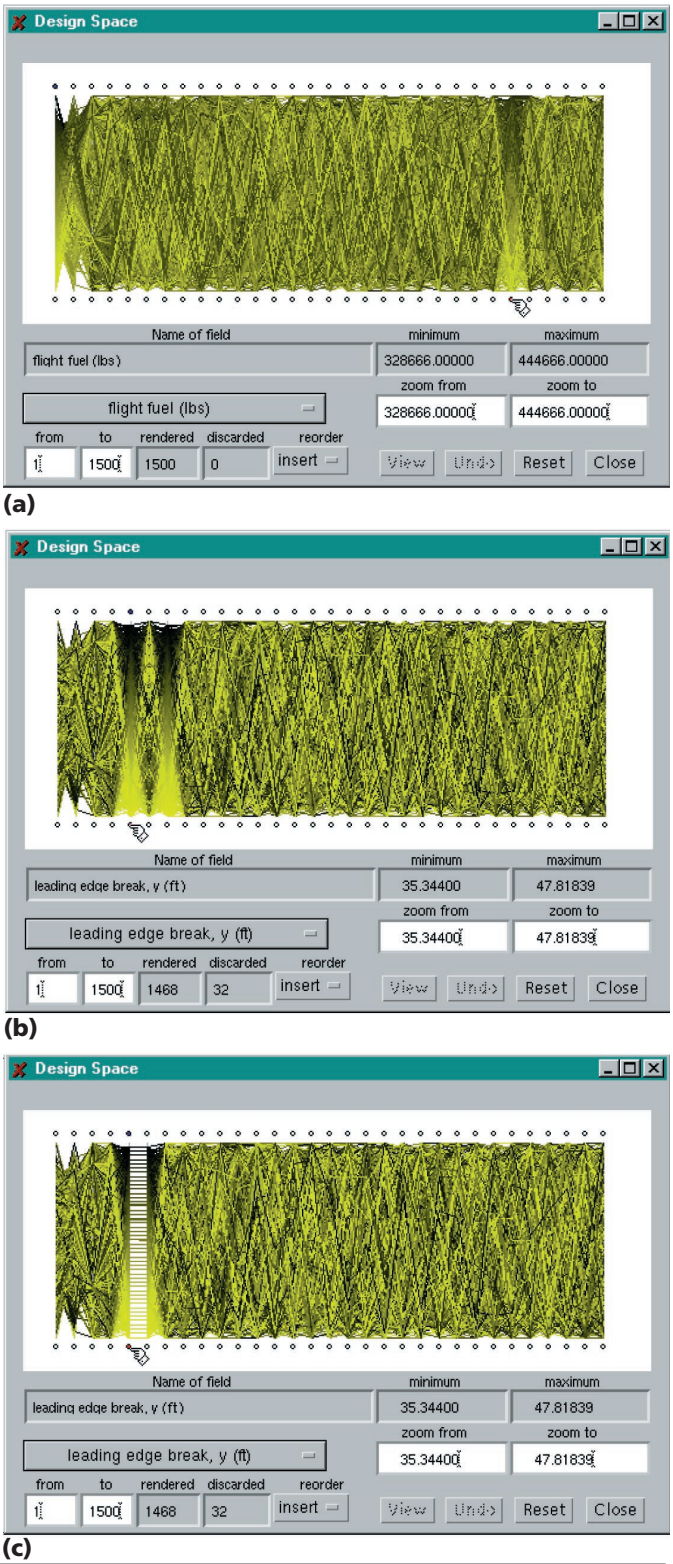


Figure 6. Browsing a design point database: (a) a parallel-coordinates representation of 1,500 design points selected uniformly from the entire design space; (b) selecting a color driver variable to highlight a relationship between two visualization variables; (c) rearranging the design variables to show a one-to-one relationship between two variables in the data set.

a particular visualization parameter. In Figure 6a, the first visualization (TOGW) determines the color. So, as lines span across the vertical axes, the user can identify those design points for which the TOGW is high or low. The design point with the lowest value of TOGW is yellow, and the one with the highest value is black. All other design points receive a linear interpolation between yellow and black. Because the design objective is to minimize the TOGW, the designer might initially be interested in the yellow lines. However, the parallel-coordinates tool's primary purpose is to let the user investigate correlations between various visualization parameters, independent of the specific application context. For example, discovering that certain design variable ranges are associated with bad designs might help the designer as much as learning that other ranges are associated with good designs.

The color gradation in Figure 6a shows that the sixth axis from the right is directly related to the first axis. It so happens that the sixth axis from the right represents the flight fuel's weight in pounds, which affects the TOGW directly. Figure 6a also shows that the second axis from the left is mildly correlated to the TOGW and flight fuel. This axis represents the range of the aircraft in nautical miles, which must be directly proportional to the amount of fuel added. Even though these particular relationships are obvious (once the viewer understands the parameters involved), they offer a good basis for understanding how to extract patterns from the data.

Visual data mining

A display of the full database, such as Figure 6a shows, is typically too overwhelming to offer any real understanding of the data. The real strength of the parallel-coordinates tool in VizCraft is the capability it provides for exploring the database. With VizCraft, the user can interact with the system's visual cues,[7] which help visualize the data set in *n*-dimensional space. In Figure 6a, a circle is above each vertical axis, and only the first circle on the left is shaded. The shaded circle indicates the visualization variable that is driving the gradation of color across the parallel coordinates. For example, TOGW is driving the color gradation in Figure 6a.

Driving the coloring

The user can select any visualization variable to drive the coloring by clicking inside the cir-

cle over the corresponding variable's axis. Clicking on the fifth circle (see Figure 6b), we see that that variable happens to share a direct relationship with the seventh visualization variable. This example shows that a clever selection of color drivers can help us extract patterns from the data set—patterns that are otherwise hidden underneath the volume of data. Such patterns must exist in the data set because 1,500 points in 29 dimensions is a very sparse experimental design.

This ability to select any visualization variable to be a color driver is quite important. Without it, the parallel-visualization technique suffers; correlations between variables become visually apparent only when the relevant variables are on adjacent axes. With the color driver capability, a user can quickly recognize high correlations between variables by simply stepping through the axes, selecting each in turn to be the color driver.

Rearranging the axes

VizCraft also lets the user rearrange the axes. For example, inserting the seventh axis before the sixth axis in Figure 6b results in Figure 6c. This figure clearly shows the one-to-one relationship shared by the fifth and seventh design variables.

Brushing

Although showing a large number of design points can help generate patterns that might interest the researcher at a holistic level, individual design points cannot be distinguished when too many are displayed at once. To allow clear views of individual design points, the user might wish to select from this design space a subregion of interest or a subregion that meets certain criteria. For example, the user might wish to eliminate all design points for which TOGW is greater than 700,000 pounds, or eliminate those points for which the range of the aircraft is less than 4,000 miles. The goal is to let the user gain some understanding of spatial relationships in *n*-space by selecting all data points that fall within a user-defined set. This technique of graphically selecting or highlighting subsets of the data set is called *brushing*.[8]

VizCraft makes it easy to extract regions of interest from the design space. For example, to select a region for which TOGW lies within a certain range, the user can select the circle below the TOGW axis and then enter the range in the Zoom From and Zoom To text boxes. This step eliminates all design points for which TOGW's value does not lie within this range. VizCraft recalibrates TOGW's axis to this new scale, while all other axes retain their current calibration.

Alternatively, the user can click on any axis, drag the mouse pointer up or down, and release it to zoom into a region of interest. Figure 7a shows the result of zooming into a region of low TOGW. The bottom text fields indicate that only four design points lie in the region of interest and that the remaining 1,496 points have been discarded. Because we are interested in designs that yield low TOGW values, we can now observe other design variables in this design subspace. Perhaps this view will help the designer gain insight regarding which values of these variables, or which combinations of these values, produced low TOGW values.

Figure 7b shows the set of constraints corresponding to Figure 7a. VizCraft provides three application-specific visualization options related to constraint violations. The No Color option renders all the polygonal lines in the default color. The All option colors each line according to this rule: design points that violate one or more constraints are red, and design points that satisfy all the constraints are green. With the Selective option, points that violate the selected constraint are red, points for which that constraint is active are yellow, and points that satisfy that constraint are green.

VizCraft makes it easy to extract regions of interest from the design space.

Putting it all together

As an example of how VizCraft can assist the designer, consider the database of 500 design points in Figure 8a. Exploration of the design points using the parallel-coordinates system proves useful here. By selecting the All option, the designer clearly observes that, out of the entire database, only three points (in yellow) are feasible. Although we could easily have a command-line program automatically filter out the feasible points and present them to the user, VizCraft provides visual feedback about the nature of the database and helps the user evaluate the few feasible points in visual contrast to the many infeasible points—and possibly discern a pattern.

In addition, the user can click and highlight certain design points and obtain the values of their design variables. For example, by highlighting the feasible design points of Figure 8a,

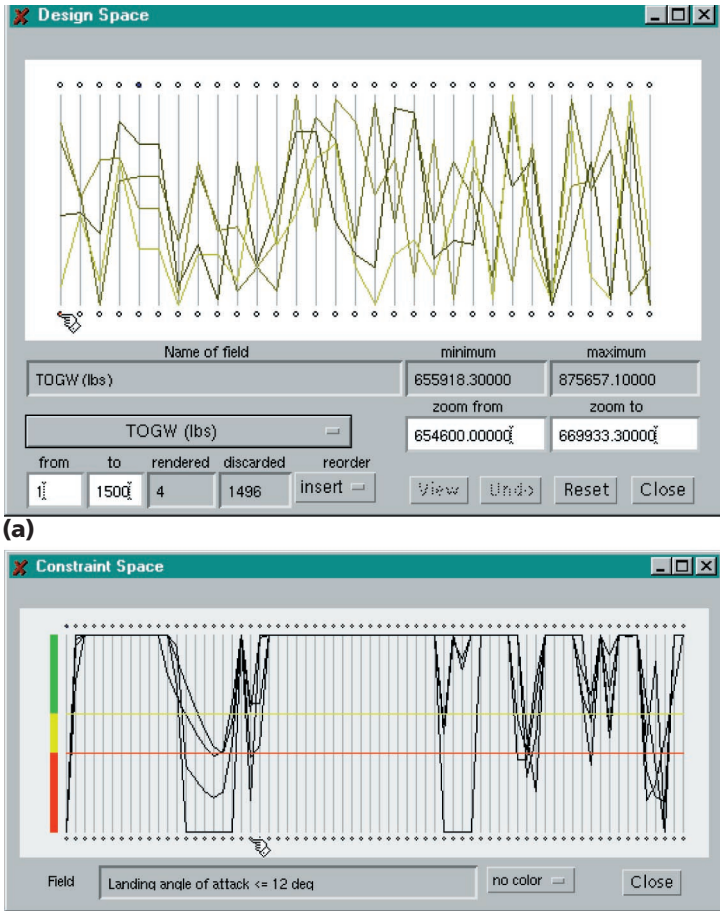


Figure 7. Graphically selecting data sets: (a) the result of brushing out design points lying outside a certain range of TOGW; (b) constraints for the four design points in Figure 7a.

the user obtains the design variables in Figure 8b and can now investigate the nature of the design variables that led to feasible designs. For instance, the values of the design variables for the three successful designs indicate that the variables are somewhat in the center of their ranges or higher, not at an extreme end. Further investigation of other databases can determine whether this is a conclusive requirement for better designs. Highlighting a design point in Figure 8b and clicking the View button computes the iconic representation of the aircraft in Figure 1. In this way, VizCraft provides an integrated environment for exploration and visualization of large data sets of HSCT designs.

VizCraft does not take the approach of locating and providing to the user an optimal design for some given problem. Clearly such a result would be ideal, but the state of the art in aircraft design has not reached that stage. Instead, VizCraft places the designer at the center of the decision-making process and seeks to integrate simulation and analysis tools with the best possible feedback through advanced visualization techniques. Thus, VizCraft in general, and the parallel-coordinates visualization technique in particular, does not solve design problems. VizCraft is a tool that aids designers in solving design problems.

By integrating both computation and visualization facilities and making them accessible from a high-level user interface, VizCraft has helped HSCT designers be more productive in a number of ways. The interface has streamlined the practice of exploring the effect of design variable combinations on aircraft performance for regions of the design space that have not previously been investigated. Where the designer originally had to manually change design variables in a file, run the analysis code, and then observe the results in a separate plotting package, VizCraft is able to perform these operations with a few clicks of a button. The data-mining capabilities of VizCraft have proven beneficial when large databases of HSCT performance data are available. Through the use of colored driving variables and brushing techniques, designers are able to visually correlate different design variable combinations and/or patterns that result in either very good or very bad aircraft performance.

Acknowledgments

This work was supported by NASA Grant NAG-2-1180, NSF Grant DMS-9625968, and AFOSR Grant F496320-99-1-0128.

References

1. D.L. Knill et al., "Response Surface Models Combining Linear and Euler Aerodynamics for Supersonic Transport Design," *J. Aircraft*, Vol. 36, No. 1, 1999, pp.75–86.
2. K. Brodlie et al., "GRASP ARC: A Problem Solving Environment Integrating Computation and Visualization," *Proc. IEEE Visualization '93*, IEEE Computer Soc. Press, Los Alamitos, Calif., 1993, pp. 102–109.
3. J. Dudley et al., "Multidisciplinary Optimization of the High Speed Civil Transport," Tech. Report AIAA-95-1024, Am. Inst. Aeronautics and Astronautics, Reston, Va., 1995.
4. E. Cluff, R.P. Burton, and W.A. Barrett, "A Survey and Characterization of Multidimensional Presentation Techniques," *J. Imaging Technology*, Vol. 17, No. 4, 1991, pp. 142–153.
5. A. Inselberg and B. Dimsdale, "Parallel Coordinates: A Tool for Visualizing Multidimensional Geometry," *Proc. IEEE Visualization '90*, IEEE Computer Soc. Press, Los Alamitos, Calif., 1990, pp. 360–375.

6. J. Assa, D. Cohen-Or, and T. Milo, "Displaying Data in Multidimensional Relevance Space with 2D Visualization Maps," *Proc. IEEE Visualization '97,* IEEE Computer Soc. Press, Los Alamitos, Calif., 1997, pp. 127–134.
7. A. Inselberg, "Multidimensional Detective," *Proc. 1997 IEEE Symp. Information Visualization* (InfoVis '97), IEEE Computer Soc. Press, Los Alamitos, Calif., 1997, pp. 100–107.
8. M.O. Ward and A.R. Martin, "High Dimensional Brushing for Interactive Exploration of Multivariate Data," *Proc. IEEE Visualization '95*, IEEE Computer Soc. Press, Los Alamitos, Calif., 1997, pp. 271–278.
9. Vanderplaats Research & Development, Inc., *DOT User's Manual*, Version 4.20, Colorado Springs, Colo., 1995.

Amit Goel received his MA in Computer Science from Virginia Polytechnic Institute and State University, where he developed VizCraft as part of his thesis. He received his BA in electrical engineering from the Indian Institute of Technology, Delhi. As part of his bachelor's thesis, he developed Web3D, a Web browser for rendering three-dimensional scenes using a custom interpreted modeling language. Find more information about him and his work at www.amitgoel.com.

Chuck A. Baker currently works as an engineer at Engineous Software. He received his BS in mechanical engineering from the University of Mississippi and his MS in aerospace engineering from Virginia Tech.

Clifford A. Shaffer is an associate professor of computer science at Virginia Tech. His current research interests include problem-solving environments, computer-supported cooperative work, component architectures, computer-aided education, visualization, and hierarchical data structures. He received his BS, MS, and PhD from the University of Maryland. He is a member of the ACM, the IEEE, Sigma Xi, and AAAS.

Bernard Grossman is a professor and head of the Aerospace and Ocean Engineering Department at Virginia Polytechnic Institute and State University. He also serves as director of the Multidisciplinary Analysis and Design Center for Advanced Vehicles. He obtained his BS in aerospace engineering and his MS and PhD in astronautics from the Polytechnic Institute of Brooklyn. From 1968 to 1980 he was in the Research Department at Grumman Aerospace Corporation. His research interests include computational fluid dynamics and multidisciplinary design optimization, and he has published more than 180 technical papers. He is an Associate Fellow of the American Institute of Aeronautics and Astronautics, where he has served as an Associate Editor of the *AIAA Journal* and a member of the Fluid Dynamics and MDO Technical Committees and was the Technical Chair of the 12th AIAA Computational Fluid Dynamics Conference.

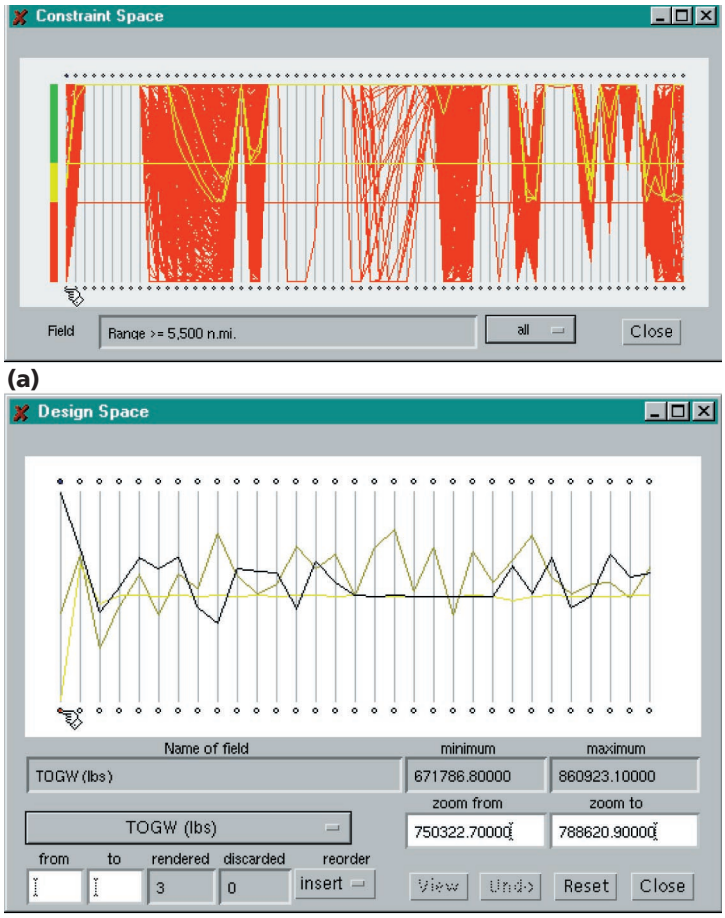


Figure 8. How VizCraft assists the designer: (a) using coloring to distinguish between good and bad design points; (b) design variables for the three feasible design points shown in Figure 8a. Figure 1 shows the iconic representation of one of the design points in Figure 8b.

William H. Mason is a technical specialist in advanced tactical aircraft configuration aerodynamics and design. At Virginia Tech, he teaches aircraft design and is involved with the Virginia Tech Multidisciplinary Analysis and Design Center for Advanced Vehicles, developing methods for multidisciplinary design optimization. At Grumman he spent 10 years applying CFD and experimental test results to the aerodynamic design of tactical aircraft, and five years in advanced systems developing advanced configurations concepts.

Layne T. Watson is currently a professor of computer science and mathematics at Virginia Tech. His current research interests include fluid dynamics, structural mechanics, homotopy algorithms, parallel computation, mathematical software, and image processing. He has worked for USNAD Crane, Sandia National Laboratories, and General Motors Research Laboratories and has served on the faculties of the University of Michigan

and Michigan State University, East Lansing, before coming to Virginia Tech. He received his BA (magna cum laude) in psychology and mathematics from the University of Evansville, Ind., and his PhD in mathematics from the University of Michigan, Ann Arbor. He is a member of Phi Kappa Phi, Blue Key, Psi Chi, Kappa Mu Epsilon, Phi Beta Chi, ACM, and SIAM.

Raphael (Rafi) T. Haftka received a BS and MS in aeronautical engineering from the Israel Institute of Technology, and a PhD in aerospace engineering from the University of California at San Diego. He worked for the Israeli aircraft industry, at Structures Research Associates, and at NASA Langley Research Center. He taught at the Israel Institute of Technology, the Illinois Institute of Technology, and Virginia Tech before joining the University of Florida, where he is Distinguished Professor in the Aerospace Engineering, Mechanics, and Engineering Science departments. His research interests are structural and multidisciplinary optimization.