Spatial Reasoning for Intelligent Control of Numerical Simulators

Ke-Thia Yao
Computer Science Department
Rutgers University
New Brunswick, NJ 08903, USA
kyao@cs.rutgers.edu
(908) 932-5263

Andrew Gelsey
Computer Science Department
Rutgers University
New Brunswick, NJ 08903, USA
gelsey@cs.rutgers.edu
(908) 932-4869

Abstract

Computational simulation is an important tool for predicting the behavior of physical systems. Many powerful simulation programs exist today. However, using these programs to reliably analyze a physical situation requires considerable human effort and expertise to set up a simulation by transforming a description of a physical system into a representation the simulation program can successfully process. Automating this process is not only of considerable practical importance but also raises significant spatial reasoning issues. The particular aspect of spatial reasoning we consider is the discretization of the spatial domain of a physical system. The discretization should be suitable as input to a partial differential equation solver. The method we use is able use geometrical and physical properties of the system, as well as numerical properties of the simulator to generate an appropriate discretization. The application domain described in this paper is the design of racing yachts.

Introduction

Computational simulation is an important tool for predicting the behavior of physical systems. Many powerful simulation programs exist today. However, using these programs to reliably analyze a physical situation requires considerable human effort and expertise. As a result, these simulation programs can't be reliably invoked by other programs. For example, human designers of complex objects like ships and airplanes typically run sophisticated simulation programs to analyze the object's physical behavior, but an automated system for designing complex objects could not easily include such a computational simulation as part the process it uses to evaluate new designs.

Artificial intelligence techniques seem essential in order to automate the simulation setup process. However, simple application of known AI technology appears inadequate for the task of automating this process. Basic spatial reasoning research is needed, particularly in the interaction of geometry and physics.

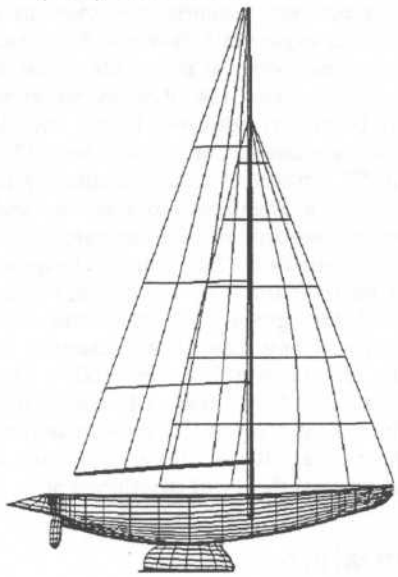

Figure 1: *Stars & Stripes*, winner of the 1987 America's Cup competition

Yachts

The Design Associate (DA) [Ellman *et al.*, 1992] is an automated design system for racing yachts like the one in Figure 1. In the process of designing a yacht, the DA must repeatedly evaluate candidate yacht designs. A large number of these evaluations are required, so the capability to automatically evaluate the performance of a candidate yacht design without human intervention is crucial for the success of the DA.

Part of the process of evaluating a yacht's performance involves computing the efficiency of the yacht's keel. Reliable computation of keel efficiency requires solving Laplace's differential equation for potential flows. For this purpose we use a computational fluid dynamics program called PMARC, a product of NASA Ames Research Center. However, the PMARC target

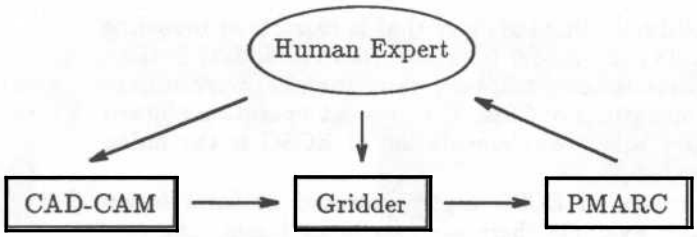

Figure 2: PMARC target environment

environment (Figure 2) is not compatible with automated use.

The input PMARC requires is a panelization — a discretization of a yacht's surface as a grid of *surface patches*, where each surface patch is an array of planar panels. There exist several types of algorithmic gridding approaches, such as algebraic formulae, conformal transformations, and partial differential equation based methods, each with its own strengths and weaknesses. Applying these methods to a particular CAD/CAM model requires considerable human expertise. Moreover, for complex geometries and complex physical situations these methods alone may not produce adequate grids for PMARC to give good results. Typically in these situations human expert using an interactive gridding program is needed to create the panelization. This interactive process usually requires several iterations of a loop in which the human expert looks at PMARC output, decides it is unacceptable, and uses the interactive gridding program to modify the grid to improve the PMARC output. In order for the Design Associate to evaluate candidate yacht designs without human intervention, PMARC must be invoked by an automated intelligent controller which can use spatial reasoning to form appropriate grids, etc. for input so that PMARC can run reliably without the supervision of a human expert.

The success of the human expert in gridding difficult geometries is in part due to the ability of expert to take advantage of the physical flow solutions to guide the gridding, which the typical gridding program is not able to do. If an automated intelligent controller is to replace the human expert, it must also be able to reason about the flow solutions.

Evaluation criteria

Through our discussion with hydrodynamicists we have formulated a list of grid evaluation criteria and constraints. On the basis of the geometric properties of the grid, these evaluation criteria attempt to predict the soundness of PMARC's output. We divide this list into four levels, ranging from constraints that absolutely must be satisfied to heuristic advice based on experience of our experts.

1. Simple connectedness constraint: surface patches must be simply connected, i.e., no holes.
2. Coverage constraint: patches must not overlap or leave gaps.
3. Planarity criterion: panels must be approximately planar.
4. Heuristic criteria:
   - following streamlines: grid lines should follow the streamlines of the fluid flowing over the body represented by the grid.
   - orthogonality: grid lines should intersect at right angles.
   - expansion ratio: the area of the adjacent panels should not increase by more than a fixed ratio.

The level one, simple connectedness constraint is actually a statement about the geometric representation used by PMARC. PMARC represents surface patches as a matrix of adjacent panels. This type of representation does not allow for holes between panels. Panelization of surfaces with holes must be done with multiple patches.

The level two, coverage constraint can be consider as a problem independent statement regarding the correctness of a grid. If a grid is to be correct for any physical situation at all, the adjacent surface patches of the grid must meet along a curve. They cannot overlap nor leave gaps.

A problem dependent statement of the correctness of grids states that grids must be a faithful discretization of the actual surface. One implication of problem dependent correctness is the planarity criterion. Each panel is represented by the points at its four corners, which may or may not be coplanar. If the corner points are not coplanar they cannot completely represent a body's actual surface.

The reasoning behind the level four criteria can be traced back to numerical properties of PMARC. For example, the need for orthogonal grids can be derived as follows. PMARC associates a constant number, velocity potential, with each panel in the grid. In order to compute the velocity vector PMARC needs to take the gradient of these velocity potentials. For each panel PMARC calculates its gradient by applying numerical differentiation using its four adjacent panels. Numerical error tends to be minimized if the adjacent panels are orthogonal with respect to each other.

These evaluation criteria in their present, qualitative form cannot easily be used to form actual grids. We have quantified these criteria and incorporated them into our program. Given a grid the program is able to judge how well the grid satisfies the evaluation criteria.

Streamline-based approach

Finding a grid that performs well against all the criteria is difficult, because some of these criteria are contradictory. For example, for an elliptic-shaped hull the streamlines start from a single point on the bow of the hull, travel along the hull, and leave from a single on

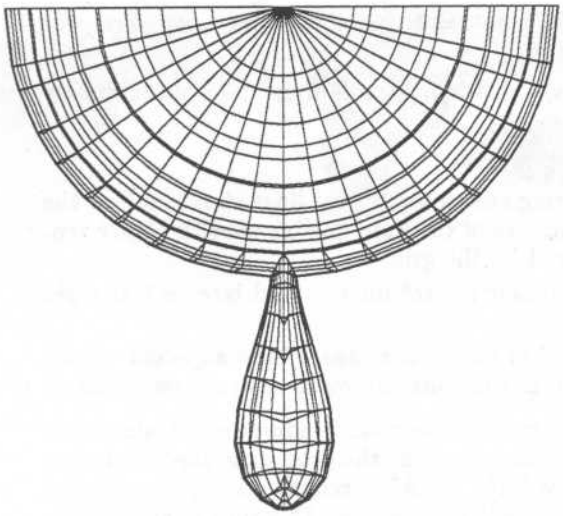

Figure 3: Frontal view of the automatically generated panelization of an ellipsoid hull and the *Stars & Strips* keel.

the stern of the hull. A gridder that gives precedence to the follow streamlines criterion will create triangular panels at the bow and the stern of the hull, see Figure 3. This of course violates the orthogonality criterion. Such a gridder that constructs grid lines from streamlines may be called a *streamline-based* gridder.

Despite its shortcomings the streamline-based approach has been successfully used by other researchers to form grids once the patch boundaries have been given. However, their streamline-based approach, as well as all existing automated gridding approaches, are good at forming grids on surface patches, but they are not capable of finding an appropriate set of patches to grid. In the *streamline-based reasoning* section of this paper we show how to use streamlines in a novel way to recognize natural patch boundaries of 3D geometries.

The success of the streamline-based approach is probably due to two reasons. One reason is that following stream criterion greatly limits the space of possible grids. The other reason is that while all other criteria are evaluated solely based on the geometry of the grid, the follow streamlines criterion is the only one that relates the geometry of the grid with the geometry of the physical solution — streamlines.

Example

Figure 3 depicts graphically a grid of a yacht, consisting of an elliptic hull and the keel of the *Star & Strips*. This grid was generated automatically by our gridding program. The grid is taken from a series of runs designed to measure the efficiency of the *Star & Strips* keel. Since hulls have negligible effect on keel efficiency, the *Star & Strips* hull is replaced with a simpler elliptic one. This gridder uses a subset of ECSG (extended constructive solid geometry, [Gelsey, 1992]). ECSG is a solid modeling language that is capable of modeling objects with free-form surfaces (as in Boundary Surface Representation), and yet retains the clear semantics of set operations of CSG. The only set operation allowed in this subset implementation of ECSG is the union operator.

Surfaces in ECSG are represented as *black boxes*. In this example there are two black boxes, one representing the hull surface and the other representing the keel surface. A black box can be thought of as a mapping from a unit square onto a surface patch. That is it takes a parametric coordinate, $(u, v) = ([0, \ldots, 1], [0, \ldots, 1])$, as input and outputs the corresponding 3-D Cartesian coordinate, (x, y, z).

The grid produced by the gridder also is expressed in terms of these black boxes. While difficult to see from Figure 3, the gridder actually divides the yacht surface into four patches: left-hull, right-hull, left-keel, right-keel. Each one of these patches is represented as a black box. Streamlines of the flow are simply lines to which a black box assigns constant u coordinates — they run lengthwise along the yacht. The v coordinates trace out lines roughly orthogonal to the streamlines.

Using this black box representation provides two advantages. One advantage is that it provides greater flexibility by hiding the implementation details of the surfaces. In this example, the hull black box is a set of simple algebraic equations, while the keel black box is a B-spline surface. The other advantage is that it gives the gridder more control in determining how densely to lay down grid lines. Using this representation the expansion ratio criterion is trivial to satisfy.

Algorithm

Given an ECSG object defined as the union of several components, the gridder performs the following four steps:

1. For each component
   - **Determine what areas of its surface need to be gridded.** Because of the union operation, some surface areas may be hidden. These hidden areas cannot be gridded according to the coverage constraint. In the yacht example, parts of the keel surface that protrude into the hull should be pruned, and a hole needs to be cut away from the hull to accommodate the keel.
   - **Use streamline-based classification to determine if the component is a *source/sink* object or a *line-source/line-sink* object.** The definitions of these two types of objects are given in a later section. Here we shall only say that the hull is a source/sink object while the keel is a line-source/line-sink object.
2. **Determine the surface patch boundaries by using the following heuristics:**

   (a) Grid a *source/sink* object using one patch.

(b) **Grid a *line-source/line-sink* object using two patches.**

(c) **If a patch contains a hole, then partition the patch along a streamline into two patches.**

The reasoning for the first two heuristics is again given later. The third heuristic is derived from the simply connectedness constraint. In the case of the yacht example, since the keel is a source-line/sink-line object, it is partitioned into two patches. As for the hull even through it is a source/sink object, it is partitioned into two patches because of the third heuristic.

3. **Grid each patch using streamline-based gridding.** This step reparametrizes the surface patches, such that constant u values approximately trace out streamlines.
4. **Check the integrity of the grid generated.** This gridder invokes many numerical subroutines, which may produce unexpected results. This step makes sure that at least the coverage constraint is satisfied.

The following subsections explain in detail about various aspects of the algorithm.

ECSG

The surfaces given as input to the gridder usually describe false surface areas that should not be gridded. The interactive gridding programs that exist today, such as I3G, are unable to distinguish between real and false surface areas, since they are not even aware of the semantics of a union operator. The human expert using the gridding programs is expected to make the appropriate semantic interpretation. Our gridder is able to correctly deal with these semantic interpretation problems by using ECSG.

Beside providing a semantically clear representation, ECSG also provides surface manipulation routines, such as intersection of objects, surface partitioning, and surface reparametrization. In implementing these routines, we have found it helpful to be able to reason with approximate geometric objects.

The *bounding box* of a object is defined to be the smallest box containing that object. The bounding box approximation enables the gridder to quickly determine if a particular point is not inside the object that the box bounds. This is useful in an efficient implementation of the intersection operation. By intersecting two bounding boxes the gridder can easily check if the two objects do indeed intersect. Then, by intersecting the surface of one object with the bounding box of the other object, the gridder is able to derive a good initial guess to ECSG's Newton's-method-based surface intersection algorithm. Although the bounding box seems to be a very coarse approximation of an object, we have found it to be adequate in dealing with sailing yachts in which a single keel is the only appendage attached to the hull. As we encounter more complex geometries, we can replace the boxes with convex hulls or unions of convex hulls.

Streamline-based reasoning

As it turns out the solution to Laplace's equation depends neither on the current state of the flow nor on time, the geometry of the object dominates the solution. And, since streamlines are key characteristics of the solution, analyzing how streamlines interact with geometry provides key insights to qualitative behaviors of Laplace's equations. These insights enable us to determine the topology of streamlines. In turn this topology provides natural boundaries for patches in grids.

The most immediate reasoning problem we encounter in streamline-based reasoning is how to get the initial set of streamlines, since we have not yet run PMARC to generate the solution from which streamlines are extracted. We have experimented with various methods of predicting the streamlines a priori. However, we have found the simple projection of the *free stream* vector onto the body surface to be a good approximation of the true streamlines. The free stream vector is the direction the fluid would have traveled if no object were in the water.

Surface point classification We first reason about how streamlines interact with individual surface points. Each surface point can be classified into the following qualitative categories.

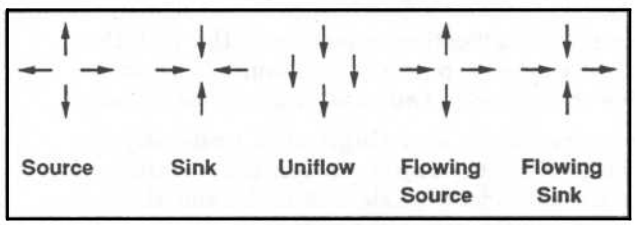

Figure 5: Node Classification

- *Source node.* A stagnation point where all the neighboring streamlines move away from it. This node type is one of two types of nodes which signals the divergence of streamlines around a body. For example, the point where the free stream first hits an elliptic hull is a source node.
- *Sink node.* A stagnation point where all the neighboring streamlines move toward it. This node type is one of two types of nodes which signals where the streamlines converge around a body. The point where streamlines leave the elliptic hull is a sink node.
- *Uniflow node.* A point where all the neighboring streamlines move in approximately the same direction. Most surface points are of this type. Also, notice streamlines can be derived by connecting uniflow nodes.

```
600 panels
min aspect ratio 1.01751 at panel 410, max aspect ratio 101.385 at panel 509
min orthogonality 16.6544 degrees at panel 500, max orthogonality 89.9987 degrees at panel 290
min noncoplanarity 0.237388 degrees between panel 483 and neighbor 484(2)
max noncoplanarity 166.694 degrees between panel 410 and neighbor 600(2)
max expansion ratio 9.05028 between panel 593 and neighbor 10(3)
condition number = 16.2
min pressure coefficient -0.660658 at panel 593, max pressure coefficient 0.80108 at panel 40
0 hull1
min chordwise doublet jump 2.22832e-05 between panel 120 and neighbor 320(2)
max chordwise doublet jump 0.141155 between panel 9 and neighbor 10(2)
min spanwise doublet jump 4.2132e-07 between panel 2 and neighbor 382(1)
max spanwise doublet jump 0.0199002 between panel 9 and neighbor 29(3)
min streamline angle difference 0.0 at panel 2
max streamline angle difference 3.44 at panel 28
:
```

Figure 4: Output

- *Flowing-source node.* A point that looks like a source node in one direction and an uniflow node in the other. This is the second type of divergent nodes, which usually can be found on the leading edges of keels and wings.
- *Flowing-sink node.* A point that looks like a sink node in one direction and an uniflow node in the other. This is type of convergent node usually can be found on the trailing edges of keels and wings.

Using this classification scheme and the fact that streamlines can only split/merge at source/sink nodes, we define the following two class of geometric objects.

Single source node and single sink node objects The surface of all the objects in this class consists of a single source node, a single sink node, and the rest uniflow nodes. Spheres, ellipsoids and other simple bodies of revolution are objects of this class.

If a surface only contain one source node and one sink node, then we can conclude that all the streamlines emanate from the source node and converge at the sink node. Here it is natural to define just one patch to cover this surface.

Single source line and a single sink line objects Flowing-source nodes tend form a line on the geometric object. This line corresponds to the location where streamlines split into two halves to go around a body. For instance, the leading edge of a keel is one such line. We call these lines source lines, because collapsing source lines into points produce source nodes. Sink lines are defined similarly. The surface of all the objects in this class consists of a single source line, a single sink line, and the rest uniflow nodes. These source and sink lines provide natural places to divide the surface into two patches.

Using only these two object class, one can already construct complex, geometric objects, for example a yacht consisting of a hull, a keel, and winglets on the keel. We can define other classes as the need arises.

Surface Integrity Checking

During the panelization process, the original surface definitions are transformed many times by the surface partitioning operators as well as by the surface reparametrizing operators. Accumulation of numerical errors, non-convergence of solvers, inappropriate parameters, or simply programming bugs may cause these transformations to fail. If this gridder is to be used in an automated environment, we need to ensure robustness. Due to the nature of numerical analysis methods it seems difficult to write codes that always are guaranteed to work. However, it is much easier to verify the result against expectation, and then modify the result if necessary. For example, the surface reparametrization operators works as follows:

1. State the expected numerical result. In this case the boundaries of original surface must match the boundaries of the reparametrized surface.
2. Perform the numerical operation. Trace streamlines over the original surface. Let these streamlines be grid lines in the constant u direction. Define grid lines in the constant v direction to be perpendicular to streamlines.
3. Verify the result of the operation against the proposed result. Streamline tracing on an irregularly-shaped surface may generate streamlines that do not cover that surface.
4. Upon failure, try to fix result or redo operation with alternative parameters. In this case stretch the new surface to cover original surface.

Implementation

The current implementation of the gridder is limited to geometries constructible from unions of two bodies.

This gridder has been tested on several examples, including Figure 3. Figure 4 is a partial output of the program showing the grid performance with respect to the evaluation criteria.

Future Work

The most immediate plan is to extend the gridder to handle more complex geometries, including the union and intersection of several bodies. Another direction is to make the gridder adaptive. That is to enable the gridder to feedback on the type of local information in Figure 4 to make local improvements to the grid. This is easy to accomplish with streamline-based gridding. For example, if feedback discovers a very skewed panel, it may be possible to locally reposition the streamlines to make it more orthogonal. Finally, we plan to add the wake geometry reasoning capabilities. One way to improve the predictive power of potential flow is to attaches vortex sheets to bodies to simulate wakes. Expert reasoning is needed to determine where to attach these vortex sheets and to determine the geometry of these sheets.

Related Work

[Jambunathan *et al.*, 1991] and [Andrews, 1988] discuss the use of expert systems technology to augment more traditional computational fluid dynamics programs. [Chao and Liu, 1991] successfully apply streamline-based gridding to 2D flow problems consisting of a single patch. Also, in 2D [John F. Dannenhoffer, 1992] has implemented a rule-based gridder capable of recognizing surface patch boundaries. Finally, [Santhanam *et al.*, 1992] identifies several key parameters in the grid feedback process.

Acknowledgments

The research on automated use of PMARC was done in consultation with Rutgers Computer Science Dept. faculty member Gerard Richter. We worked with hydrodynamicists Martin Fritts and Nils Salvesen of Science Applications International Corp., and John Letcher of Aero-Hydro Inc. Our research is part of the CAP (AI and Design) project, and benefited significantly from interaction with other members of the project. The CAP project is supported by the Defense Advanced Research Projects Agency and the National Aeronautics and Space Administration under NASA grant NAG2-645. The National Science Foundation provided additional support for this research though grant CCR-9209793.

References

Alison E. Andrews. Progress and challenges in the application of artificial intelligence to computational fluid dynamics. *AIAA Journal*, 26(1):40–46, January 1988.

Y. C. Chao and S. S. Liu. Streamline adaptive grid method for complex flow computation. *Numerical Heat Transfer, Part B*, 20:145–168, 1991.

T. Ellman, J. Keane, and M. Schwabacher. The Rutgers CAP Project Design Associate. Technical Report CAP-TR-7, Department of Computer Science, Rutgers University, August 1992.

Andrew Gelsey. Modeling and simulation for automated yacht design. In *AAAI Fall Symposium on Design from Physical Principles*, pages 44–49, 1992.

K. Jambunathan, E. Lai, S. L. Hartle, and B. L. Button. Development of an intelligent front-end for a computational fluid dynamics package. *Artificial Intelligence in Engineering*, 6(1):27–35, 1991.

III John F. Dannenhoffer. Automatic block-structured grid generation — progress and challenge. In Elaine Kant, Richard Keller, and Stanly Steinberg, editors, *AAAI Fall Symposium Series: Intelligent Scientific Computation*, pages 28–32, 1992.

Tharini Santhanam, J.C. Browne, J. Kallinderis, and D. Miranker. A knowledge based approach to mesh optimization in CFD domain: 1D Euler code example. In Elaine Kant, Richard Keller, and Stanly Steinberg, editors, *AAAI Fall Symposium Series: Intelligent Scientific Computation*, pages 115–118, 1992.