Journal of
Ship&Ocean
Technology
©DECEMBER 2001

SOTECH Vol. 5, No. 4, pp. 1~18, 2001

SDL: A New CAGD Library for Conputer-Aided Ship Design

George D. Koras[1], Panagiotis D. Kaklis[1] and Apostolos D. Papanikolaou[1]

[1]Department of Naval Architecture and Marine Engineering, National Technical University of Athens, Greece; E-mail: papa@deslab.ntua.gr

Abstract

Software written in academic environments is rarely integrated and/or reused. Moreover, typical academic software products do not communicate with commercial CAD packages, consequently functionality that is readily available to industry is not available to academic researchers. The Surface Design Library (**SDL**) is a library serving as a framework under which, CAGD software, intended for research or educational use, can be written in such a manner that it becomes a reusable part of a useful CAGD system. Moreover SDL is equipped with an IGES translator so that it communicates with commercial CAD packages. This paper gives a brief overview to SDL and presents examples of using it for high quality fairing performed on a naval ship hull, before employing commercial CAD packages for further design and analysis.

Keywords: computer aided ship design, computer aided geometric design, fairing

1 Introduction

Surface Design Library (**SDL**) is a C library for Computer Aided Geometric Design (**CAGD**) developed by members of the Ship Design Laboratory of the National Technical University of Athens (**SDL-NTUA**). The principal aim of SDL is to support the geometric kernel of the research and educational activities, materialized in the context of SDL-NTUA, and integrate, under one structured framework, software written by members of SDL-NTUA as well as students of NTUA working for their Diploma theses.
Undoubtedly, the SDL-NTUA computing environment is heterogeneous from both the hardware and software point of view. Currently SDL-NTUA comprises:

- 6 workstations (3 SGI Indys, 1 DECstation 5000, 1 DEC 3000, 1 Compaq AXP 3000) and
- 20 personal computers.

This hardware operates under several operating systems, namely:

- SGI IRIX and Digital UNIX for the workstations,

- Windows 98, Windows NT and Linux for the PCs,

and hosts several, in-house or commercial, software packages for:

- Ship Design (Autoship^{TM}, TRIBON^{TM})
- Computational Fluid Dynamics (Shipflow^{TM}, NEWDRIFT: an in-house 3D panel-based seakeeping program)
- Geometric Modeling (AutoCAD^{TM}, Mechanical Desktop^{TM}, Superscape^{TM}).

A primary pursuit of the SDL design and implementation team was and still is to provide for a wide range of high quality CAGD tools, with priority given to functionality that pertains to design. Next, given that geometry representation and processing resides in the kernel of most **C**omputer-**A**ided **S**hip **D**esign (**CAshipD**) applications and influences decisively their performance, an equally important pursuit is to establish robust and efficient communication between SDL and CAshipD application packages.
The present paper is structured as follows. Section 2 gives a brief overview of the library. Section 3 focuses on the communication with CAD or CAshipD packages. Section 4 presents and illustrates with several figures two test cases that are likely to occur in the context of CAshipD. More specifically, SDL is used for high quality automatic fairing and the results are then exported to Autoship for further analysis and elaboration. The paper concludes with Section 5, which summarizes the current status of SDL and proposes areas of further development in the near future.

2 An overview of SDL

Technically speaking, SDL is a collection of structures, macros and functions, implemented in the C programming language, that intends to support the development of CAGD and CAshipD applications. It can be controlled through an **A**pplications **P**rogrammer's **I**nterface (API) and can be viewed from three different viewpoints:

(a) The API developer's viewpoint,

(b) the **A**pplication **P**rogrammer's (**AP**) viewpoint, and

(c) the **A**pplication **U**ser's (**AU**) viewpoint.

For the initial development phase of SDL, the AP's viewpoint was considered to be more important, that the AU's point of view, as, at this stage, SDL is mostly used for the development of CAGD/CAshipD applications supporting the research activities of SDL-NTUA, i.e., applications written by people with a certain level of programming skills. At a second stage, SDL will be used for the development of applications designed to assist the educational activities of the lab, and as such, should be more friendly and appealing; therefore the AU's viewpoint will be equally important.
To make the AP's work easier, SDL was required to have the following three basic features:

- **Open architecture**: It is not required that the AP is knowledgeable of the SDL internal structure. Instead, convenience functions are available, permitting the AP to perform several tasks at the cost of one function call. Nevertheless, access to lower-level SDL functions is also free, should the AP need to possess more direct control over SDL.

- **Dynamic memory management**: Memory requirements of CAGD/CAshipD applications typically fluctuate during their runtime and can reach high levels, especially when heavy optimization problems are to be solved or high resolution graphics are to be generated. Therefore it is important that every geometric or graphic SDL process uses exactly as much memory as needed, freeing it at the end and not leaving garbage behind it.

- **Portability**: SDL should be useful for developing both large scale research applications, imposing significant hardware requirements, as well as small scale educational software, running on inexpensive platforms. It is, thus, required that SDL is portable between different hardware platforms and different operating systems. As already mentioned, the adopted programming language is standard ANSI C, but when graphics code is concerned, portability is not easily achieved. Towards this objective, OpenGL(Neider et al 1994) was employed, as it seems to be a standard that has been accepted by the vast majority of hardware and software vendors. Initially developed under SGI IRIX, SDL has been ported to Linux and Windows NT systems.

The tools currently provided by SDL fall in one of four categories:

- **Data management tools**: Tools that store in memory and manage therein the definition data for the geometric entities encountered within SDL, i.e., mainly curves and surfaces. These tools essentially set rules that APs should follow so that different applications are compatible with each other.

- **Graphics and User Interface (UI) tools**: Tools for the realistic visualization of geometric entities and analysis results, as well as tools for interacting with the AU.

- **Geometric modeling tools**: Tools that constitute the number-crunching core of SDL. They are based on CAGD theoretic results in order to handle a variety of CAGD problems, e.g., curve / surface evaluation, interpolation, calculation of Differential Geometry invariants, curve / surface fairing, etc.

- **Curve and Surface Evaluation tools**: Tools such as porcupine plots, curvature plots, reflection lines, isophotes, etc., that allow the user to evaluate the quality of curves and surfaces and comprehend their shape, revealing features not easily visible in a simple rendered image of the designed object.

2.1 Data management

This subset of SDL contains tools that enable the AP to handle the SDL geometric entities. Thus, for each entity, the following functionality is typically provided:

- the relevant C structure definition,

- at least one function that creates the entity, allocating memory for its storage and initializing its definition data,
- macros that provide read / write access to the structure's members and, finally,
- at least one function that destroys the entity, freeing the memory used for its storage.

SDL structures can be classified in two categories: (i) **P**rimary structures (Ps) and (ii) **S**econdary structures (Ss). In Ss, auxiliary geometric entities are stored, e.g., vectors, points, arrays and grids of points. Ps then include Ss, either directly or, more often, via a pointer. Overall, SDL maintains a list of visible entities, which contains pointers to PS that should be redrawn every time the application's window is redrawn. Filters are provided that scan this list and pick entities that are of a specific type (e.g., B-spline curves), so that the AP can collectively apply some operator to them, e.g., raise the degree of every B-spline curve on screen.

2.2 Graphics and user interface

The significance of graphics output in CAGD is readily justifiable. Moreover, as CAGD/CAshipD and Computer Graphics are communicating but distinct scientific areas, it is useful to relieve the CAGD/CAshipD researcher from the burden of developing software for the visualization of the geometric entities they work with. The following are the most important aspects of the SDL graphics functionality offered both to the AP and the AU:

- Rotating and translating the visible objects.
- Zooming in and out.
- Visualizing points, lines, planes, curves, surfaces, etc.
- Creating shaded images of the objects (see Figure 1), while letting the AU control the lighting parameters.
- Displaying 2D and 3D graphs as well as color plots of scalar values (e.g., curvature plots).
- Creating image files in the **G**raphics **I**nterchange **F**ormat (GIF) and **E**ncapsulated **P**ostscript (EPS) files containing the SDL graphics output.

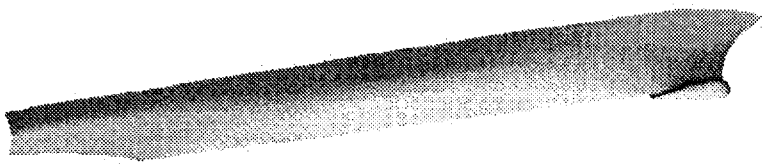


Figure 1: A shaded view of a naval ship model imported in SDL

Elementary **U**ser **I**nterface (**UI**) functionality is also offered by SDL, so that the AP does not have to write code that deals with menus, user input, etc. This is important not only because the

typical AP will want to focus on the application rather than programming UIs, but also because the AP is provided with a standard interface to the windows environment and, thus, portability of applications between different windows systems is facilitated. The most important aspects of UI functionality in SDL are the following:

- **Picking**: When the AU clicks in the SDL window, the SDL application is notified and provided with information about the picked pixel and the geometric entity that lies on it, as well as entity-specific information. For example, if the entity is a curve, the SDL application will be provided with the parametric position on which the user clicked and a flag indicating whether this location coincides with a *nodal point* or not.
- **Menus**: All SDL applications share some common menus, that allow the AU to control, e.g., the display parameters. Moreover, any application can define its own menus and connect one of its functions to each menu entry, hence instructing SDL to call the function when the corresponding menu entry is chosen by the AU.
- **User Input**: Functions are defined that request some input from the AU, i.e., a number, a string etc. Note that in different windows systems this user input will have different appearance, assuming the common look and feel of the native system applications, without the AP having to change the application code.

2.3 Geometric modeling

Currently the dominant curve / surface representation in SDL is the B-spline representation (Figure 2), see, e.g., (Farin 1997, Hoschek and Lasser 1993). Moreover, simple and composite Bezier curves and surfaces are also defineable. Whenever possible, however, SDL developers should (and do) strive to write code that is as generic (i.e., representation independent) as possible. For this purpose, along with the data defining each SDL curve or surface (e.g., knot vectors, control points, etc.) a pointer to an *evaluator function*(i.e., a function that calculates a point on the curve / surface, given a parametric position) is stored. Therefore, SDL applications can call evaluators via this pointer, so that the SDL application code is the same regardless of the underlying representation. Pointers to functions calculating derivatives are also provided to the SDL application for similar use.

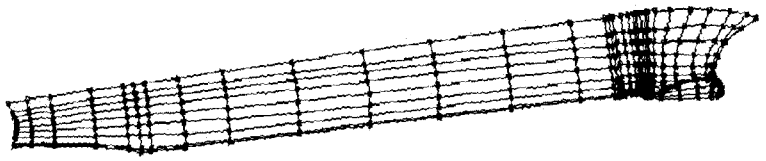


Figure 2: The B-spline surface control net of the model of Figure 1

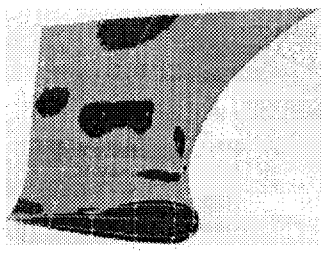
However, a lot of B-spline specific functionality is also available: knot insertion, subdivision, degree elevation, B-spline basis functions multiplication, as well as B-spline specific curve and surface fairing algorithms.

Apart from curves and surfaces, other "lower-level" geometric entities are also implemented, e.g., point arrays and point grids. An entity, that is also useful in a wide range of applications,

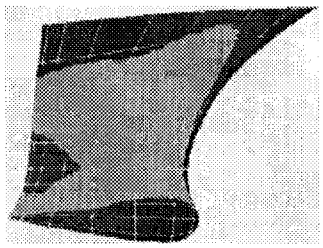
is the *triangulation* which is a collection of triangles, allowing the storage of scalar values and/or vectors at each triangle vertex. As these scalars / vectors can then be visualized, this entity could be useful for visualization of scalar or vector fields obtained from various types of analyses on a ship hull (e.g., hydrodynamics, stress analysis etc.).

2.4 Curve and surface evaluation tools

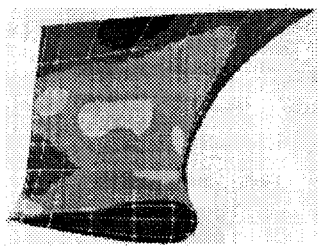
Visual inspection of the ship hull geometry is the most elementary method to evaluate its quality. Realistic shaded images, as the one in Figure 1, are typically employed for this purpose, but they are not always sufficient. Evaluation tools that take into account derivatives of the surface are more useful. Consider, for example, the Gaussian curvature color-plot in Figure 3(a). We can see that the bow is overall non-convex apart from: (a) the bulb area (which of course should be convex) and (b) other relatively small areas, that ought to be non-convex. This plot cannot inform the designer whether the latter areas are *humps* or *pits*; the designer can only understand this in conjuction with a mean curvature color-plot (see Figure 3(b)). Negative mean curvature in the vicinity of the design waterline indicates that the convex area there is a pit, while the bulb and the convex area near the deck line are humps, since the mean curvature there is positive. All this information can be derived from just one color plot: the Koenderink's shape index plot(Koenderink 1990) (see Figure 3(c)).



(a) The Gaussian curvature. Light/dark gray indicates non-convex/convex areas



(b) The mean curvature. Light/dark gray indicates negative/positive mean curvature



(c) Koenderink's shape index varying from -1 (light gray) to 1 (dark gray)

Figure 3: Using SDL tools to understand the shape in the bow area

Other curve / surface evaluation tools are porcupine plots (depicting the normal vector at points on the curve / surface), curvature graphs, reflection lines, isophotes, etc.

3 Communication with CAD packages via IGES

From the above discussion it should be clear that SDL was not intended to be an all-purpose marine design software tool; it rather focuses on the geometric side of the marine design process. Moreover, SDL, at the current stage of its development, assumes a user with some mathematical background while CAD packages should accommodate designers with low mathematical education. For these reasons, ever since the beginning of the SDL development, interchangeability of geometric models with commercial CAD packages was considered necessary. The Initial Graphics

Exchange Specification, (NIST 1997) (**IGES**), on the other hand, is a product data exchange standard that has been accepted and used by the CAD/CAM/CAE communities and is now supported by nearly all relevant software packages. Consequently, not only the existence of an IGES translator within SDL was deemed necessary, but also IGES conventions were taken into account when figuring out the details of the SDL data structure. For example, some data that should be written in the Directory Section Entry for every IGES entity are included in the *SDLgeneral* structure, which is present in every SDL entity.
Let us recall that an IGES file is organized in five sections, namely:

- the Start section (S), which is but a set of comments, ignored by IGES-processing software,
- the Global Section (G), which includes parameters affecting all entities in the file,
- the Directory Section (D), organized in *Directory Entries*, which serves to provide a list of the entities in the IGES file and includes information useful for all types of entities,
- the Parameter Section (P), organized in *Parameter Records*, which contains the data defining each entity, and
- the Terminate Section (T), which contains checksums that help the diagnosis of incomplete file transmission.

Strict technical requirements exist for the format of an IGES file, e.g., the length of any line should be 80 characters, the last eight being used for a *sequence number* that counts lines within each section. Such technical requirements are a burden to the geometry-oriented programmer and had to be encapsulated in functions of generic use. These functions allow, for example, the AP to successively write data, contained in C-language variables, to an IGES *Parameter Record* and, finally, write the record to the file, taking care of the format details. Such functions form the *technical layer* of the SDL-IGES translator.

The *geometry layer* of the SDL-IGES translator contains functions, that use the technical layer to translate SDL geometric entities to IGES *Directory Entries* and *Parameter Records*. To do so, they maintain C data structures designed to contain all the data that will be written to the IGES file, essentially storing in RAM the whole file and modifying it therein. Currently, the geometric entities supported by the SDL-IGES translator are:

- Copious Data Entity (Entity Type Number 106),
- Transformation Matrix Entity (Entity Type Number 124),
- Rational Parametric B-Spline Curve (Entity Type Number 126), and
- Rational Parametric B-Spline Surface (Entity Type Number 128).

Finally, the single function comprising the *file layer* of the SDL-IGES translator, calls and coordinates all of the above functions to write the file on disk, taking care of interactions between sections.

For the SDL-IGES translator to be tested, models were exported to two software packages, among those that are currently in use by SDL-NTUA: Autoship and Mechanical Desktop for AutoCAD. Figure 4 is a screen-dump from Autoship, depicting sections of a naval ship model along with the boundary curves of the five B-spline surfaces comprising the model.

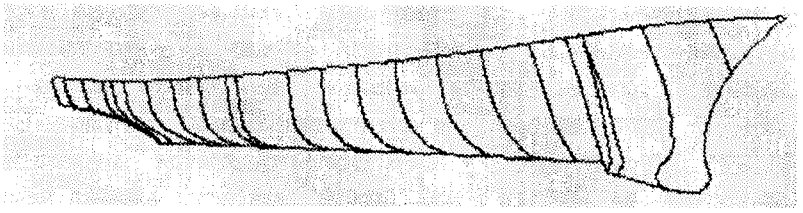


Figure 4: Stations of the model created in Autoship

4 Curve and surface fairing in SDL

Curve and surface fairing has been one of the research focal points of the CAGD group within SDL-NTUA for the last few years (see (Hoschek and Kaklis P.D. 1996), (Nowacki and Kaklis 1998), (Applegarth et al 2000), (Pigounakis 1997), (Pigounakis and Kaklis 1996), (Pigounakis et al 1996), (Pigounakis and Kaklis 1997), (Pigounakis and Kaklis 1999), (Gerostathis et al 1999), (Kaklis and Koras 1998), (Koras and Kaklis 1999), (Kapniaris 1995), (Voutsinas 1997), (Gerostathis 1997), (Saraidaris 1998), (Kostas 1998), (Kaklis and Koras 1998), (Koras and Kaklis 1999)). A major part of the software developed for these works has been integrated into SDL. Moreover, an attempt to adopt a common fairing problem definition is under way so that different fairing methods share a common API and a common UI. Specifically, a fairing problem is defined as a constrained minimization problem in which the objective function can be:

- a *fairness measuring* functional (see (Greiner 1999) and references therein), or
- a *fidelity functional*, measuring the distance between the initial and the faired curve / surface (see (Kaklis and Koras 1998)), or
- a weighted combination of both (see (Pigounakis and Kaklis 1996, Pigounakis et al 1996)).

The objective function depends on a finite set of points (typically Bezier / B-spline control points)[1], hereafter called *free points*.
The associated constraints can be:

- tolerance constraints imposed on the point coordinates, and/or
- shape constraints imposed on:
  - points on the curve / surface (see (Kaklis and Koras 1998, Koras and Kaklis 1999)), hereafter called *constrained points*, and/or,
  - entire segments / patches of the curve / surface (see (Pigounakis and Kaklis 1997, Koras and Kaklis 1999)), hereafter called *constrained segments / patches*.

In conformity with the above approach, C-language structures are implemented in SDL which hold the data necessary for defining a fairing problem. SDL-UI functions were written, allowing the user to define the fairing problem, namely:

[1]The rational spline case being catered for by the use of 4D points.

- choose an objective function,
- specify free points and tolerances,
- pick points / segments / patches on the curve / surface to impose shape constraints on them.

Using the data contained in the afore-mentioned structures, the SDL fairing functions set up the minimization problem, building the matrices required for the communication with the optimization software and, then, call the routine which is appropriate for the type of the problem, e.g., least squares fitting, quadratic programming, nonlinear programming, etc. We currently employ the NAG(NAG LTD. 1990) library to solve constrained minimization problems.

The ensuing subsections present two test cases in which SDL was used for curve (4.1) and surface (4.2) fairing and the results were imported to Autoship for further analysis.

4.1 Curve fairing

The results presented in this section were obtained using methods and software developed in (Pigounakis 1997, Kapniaris 1995). The curves to be faired are six B-spline curves interpolating station points digitized from the FORMDATA(Guldhammer 1962) systematic series. The adopted objective function is:

$$J(Q) = \frac{1}{3}J_0(Q) + \frac{1}{3}J_{r1}(Q) + \frac{1}{3}J_{r2}(Q) \tag{1}$$

where Q(u) is the faired curve, and

$$J_0(Q) = \sum \|d_i - d_i^0\|^2 \tag{2}$$

is the fidelity functional, measuring the distance between the initial $\{d_i^0\}$ and the faired $\{d_i\}$ control points. Furthermore:

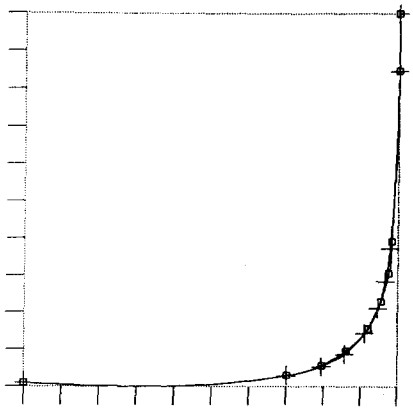
$$J_{r1}(Q) = \int \|Q^{(2)}(u)\|^2 du \qquad \text{and} \qquad J_{r2}(Q) = \int \|Q^{(3)}(u)\|^2 du \tag{3}$$

are fairness measuring functionals based on the second- and third- order parametric derivatives $Q^{(2)}$(u) and $Q^{(3)}$(u) of the curve, respectively.

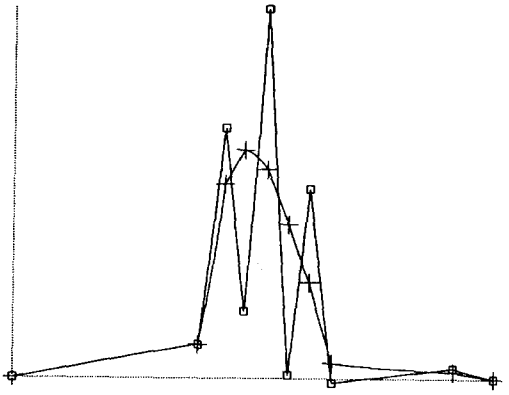
The above objective function is accompanied with the following constraints: The first two and the last two control points of each curve are fixed to ensure that the endpoints and the end tangents of the curves do not change, while the remaining control points are set free. Constrained points were added when and where fairing resulted in unwanted inflection points. In Figure 5(a), the initial and the faired curves along with their curvature plots are depicted; both curves are depicted by solid line with □/+ denoting nodal points on the initial / faired curve, respectively.

Although the difference between the initial and the faired curves is hardly visible, the curvature plots indicate a substantial improvement of the curve quality, demonstrated by the elimination of many, obviously extraneous, inflection points.

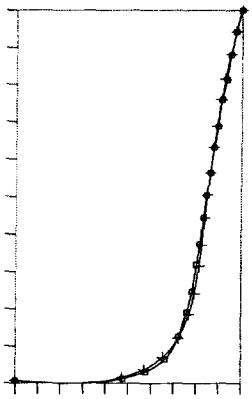
Finally, all stations were imported in Autoship for further analysis. In Figure 6 porcupine plots of Station 2 before and after fairing are given.



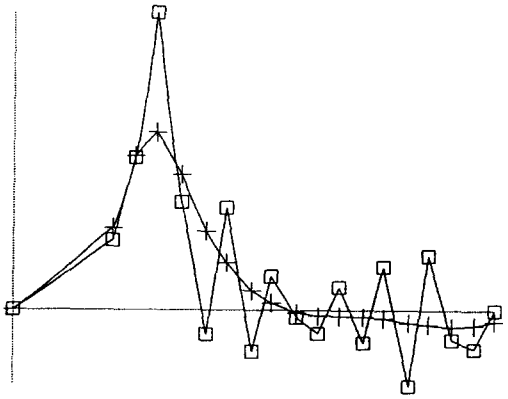
(a) Station 1



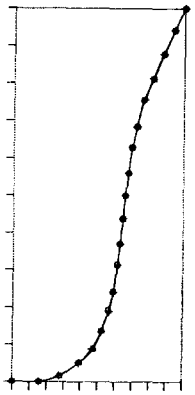
(b) Station 1: Curvature Plot



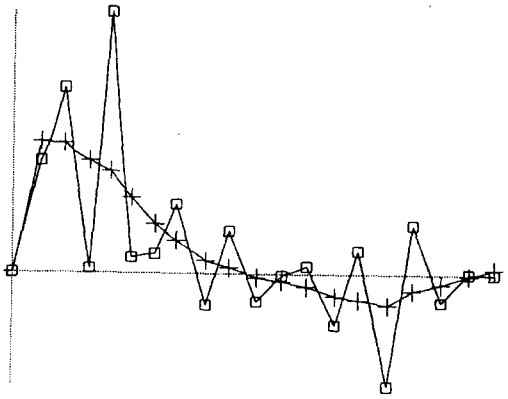
(c) Station 2



(d) Station 2: Curvature Plot

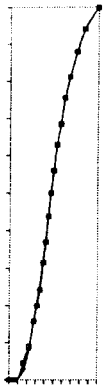


(e) Station 3

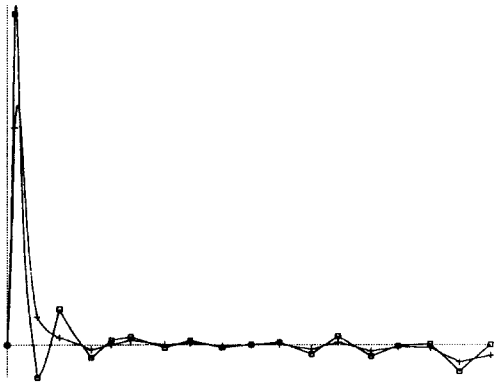


(f) Station 3: Curvature Plot

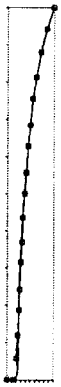
Figure 5: Initial(□) and faired(+) FORMDATA stations along with their curvature plots(cont'd)



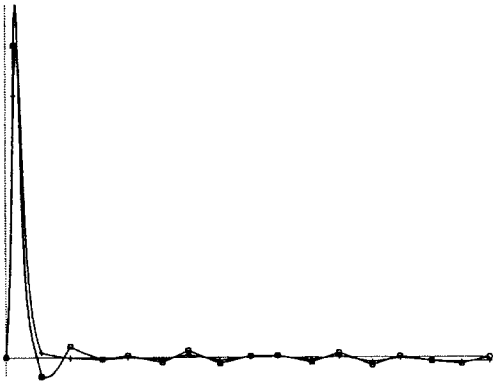
(g) Station 4



(h) Station 4: Curvature Plot

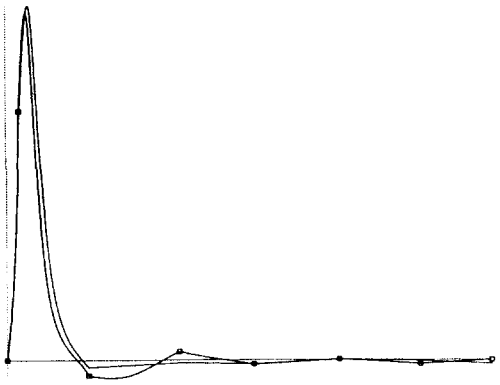


(i) Station 5



(j) Station 5: Curvature Plot

(k) Station 6



(l) Station 6: Curvature Plot

Figure 5: (cont'd)

Figure 6: Porcupine plots of the initial(left) and the faired(right) Station 2, created in Autoship

4.2 Surface fairing

In this subsection, we present results obtained using methods focusing on fairing surfaces via eliminating *shape failure areas*, i.e., surface regions where the Gaussian and/or mean curvature exhibit wrong sign. These methods are described, in detail, in (Kaklis and Koras 1998, Koras and Kaklis 1999, Kaklis and Koras 1998). The test case, we present here, deals with the mid-body part of the naval ship model of Figure 1. A Gaussian curvature plot(Figure 7) reveals a large non-convex area (indicated by light gray) near the deck line, while it is legitimate to assume that the designer would want the hull to be convex in this area. On the other hand, not every non-convex area should be rejected; as the lower part of the stern should be non-convex(see Figure 1), the non-convex area appearing at the lower-after part of the mid-body should be acceptable. Similarly, because of the bow being predominantly non-convex(see Figure 3), the two non-convex regions at the fore mid-body are also acceptable, especially as they enlarge, tending to merge, as we move towards the bow. In fact, where the mid-body and bow meet, there is an annoying, very thin convex stripe, probably a result of poorly chosen boundary conditions.

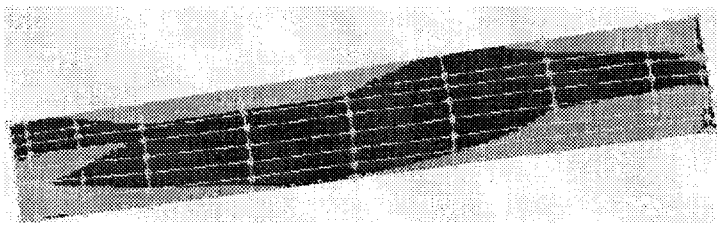
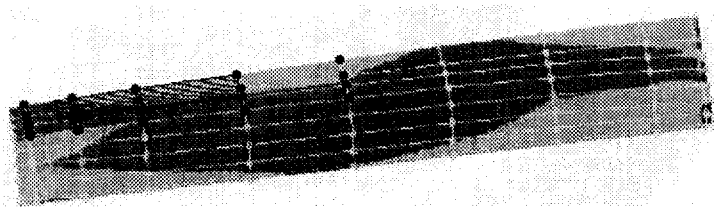


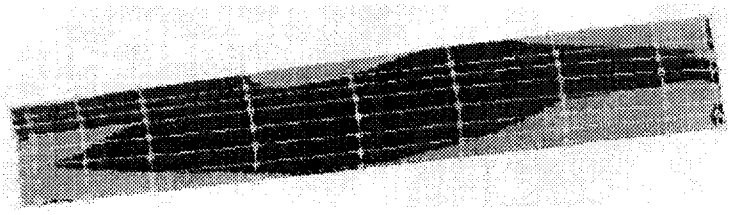
Figure 7: Control point grid and Gaussian curvature plot of the initial surface. Light gray and dark gray indicate non-convex and convex areas respectively

To improve the situation just described, we firstly concentrate on the non-convex area near the deck line. We select two patches, depicted with a black hatching in Figure 8(a), to be constrained

patches, i.e., we impose on them the convexity constraints developed in (Koras and Kaklis 1999). Our *free points* will be the control points depicted as black spheres in Figure 8(a), while the objective function will be a fidelity functional similar to that introduced in equation (2), so that we get as small a perturbation of the initial surface as possible. Since the convexity constraints are highly non-linear, the objective function being quadratic will keep the optimization problem from becoming prohibitively complex.



(a) Constrained patches(black hatching) and free control points(black spheres) for the first step of convexity enforcement



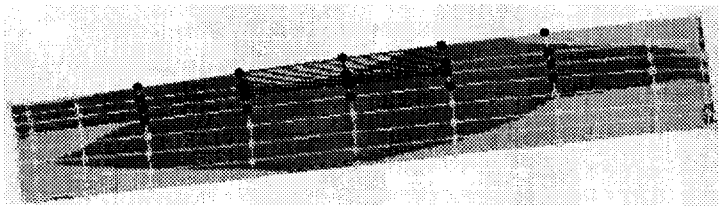
(b) The result of the first step

Figure 8: The first step

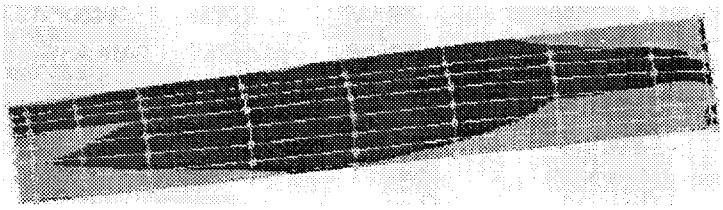
The result is shown in Figure 8(b). We can see that the non-convex area has shrunk to satisfy the convexity constraints. We take one more step in the same direction, the constrained patches and free points shown in Figure 9(a), with the obtained results given in Figure 9(b).
In Figure 10, we can see porcupine plots (generated with Autoship) of three u-constant lines[2] of the surface, chosen to span the shape failure area dealt with above. Since each column of the initial control point grid lies entirely on a plane perpendicular to the longitudinal axis of the ship, u-constant lines are stations, a property almost true for the final surface as well, because the optimization induced very small changes on the surface. As we can see in Figures 10(a), 10(c) and 10(e), the normal vector on the u-constant lines of the initial surface is pointing inwards, near the deck line, and then outwards. This is an obviously unwanted feature of the surface which was healed, as can be seen from Figures 10(b), 10(d) and 10(f).

As mentioned earlier, there is a small, unwanted convex region at the junction between the mid-body and the bow; Figure 11(a) depicts a close-up view of the area. To eliminate this region, we picked 12 constrained points(black pyramids in Figure 11(b)) on which we imposed negative Gaussian constraints as described in (Kaklis and Koras 1998). An almost entire column of the

[2]Curves resulting when fixing the u-parameter of a surface $S(u, v)$



(a) Constrained patches and free control points for the second step of convexity enforcement



(b) The result of the second step

Figure 9: The second step

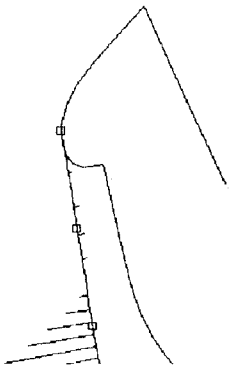
control point net was selected for free points(black spheres in Figure 11(b)); note however that boundary control points have been kept fixed in order to preserve the continuity of the ship model. The objective function is the same as before and the result of the optimization process is shown in Figure 11(c). We can see that the convex region has considerably shrunk but it is obvious that we cannot eliminate it completely without modifying the boundary control points[3].

5 Conclusions

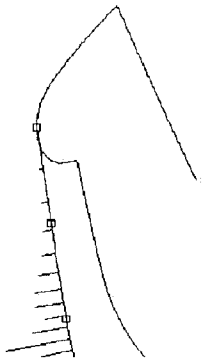
The integration of academic software and the reusability of software modules can enhance greatly the productivity of a research team. Moreover, easy communication with commercial CAD packages can aid to connect research to engineering practice. Higher level education can also profit when software written for research purposes is used in teaching to demonstrate concepts and methods. These are the primary advantages in developing and using a library like SDL.
Currently, SDL offers tools for:

- Description of geometry via the B-spline or Bezier representation,
- CAGD operations, with a focus on fairing of curves and surfaces,
- Visualization of geometry and quick development of a UI, and
- Evaluation of curve and surface quality.

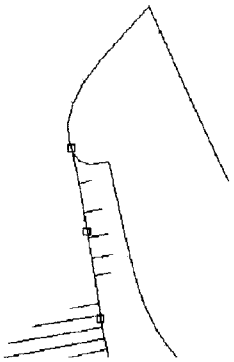
[3]Recall from standard B-spline theory (see, e.g., [3]) that the boundary curve of a B-spline surface is affected only by the control points lying on the corresponding boundary of the control net.



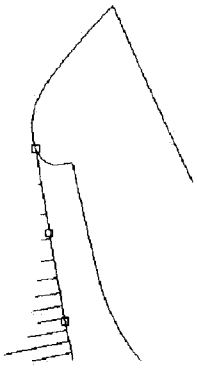
(a) Initial surface, u-constant line 5



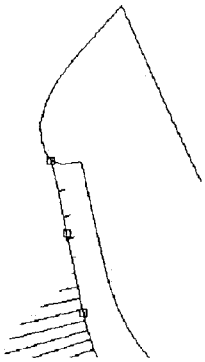
(b) Faired surface, u-constant line 5



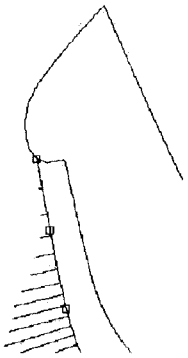
(c) Initial surface, u-constant line 6



(d) Faired surface, u-constant line 6

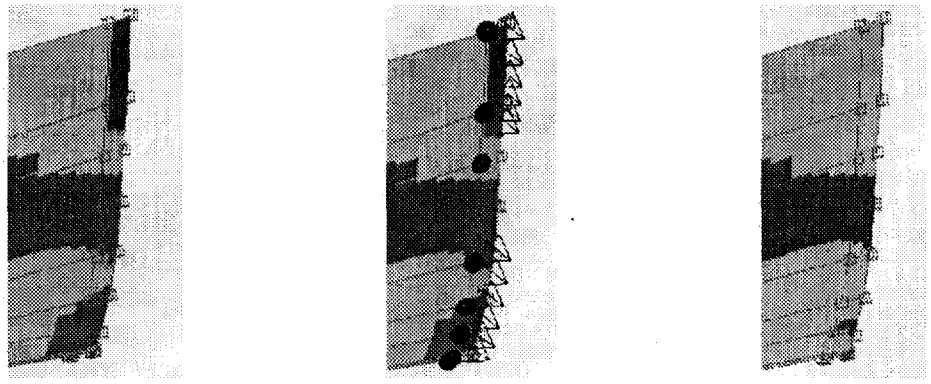


(e) Initial surface, u-constant line 7



(f) Faired surface, u-constant line 7

Figure 10: Porcupine plots of u-constant lines(stations) created in Autoship



(a) The initial surface

(b) Free control points and constrained points

(c) The result

Figure 11: Convexity enforcement with SDL using point constraints

Our primary goal for the near future is to enrich the SDL toolkit with functions that pertain to Ship Design. Such a task could be the enrichment of the, currently fairing-oriented, hull optimization functionality, with constraints involving integral characteristics of the ship; e.g., prescribing the ship volume within some tolerance, etc. Another major development direction could be the implementation of known algorithms to important CAGD problems such as *surface-surface intersection.*

Finally, we intend to develop SDL applications that would assist teaching Analytical and Differential Geometry, Computer Graphics and, of course, CAGD, at a Naval Architecture and Marine Engineering Department. Such applications could be used in classroom demostration as well as laboratory practice providing students with hands-on experience in the use of geometric and graphics algorithms.

Acknowledgements

The first of the authors would like to thank Dr. N.S. Sapidis for his contribution in the initial design phase of SDL and for many valuable discussions concerning software engineering throughout the last years. Thanks are also due to the people who, along with the first of the authors, participated in the development of SDL: Dr. K.G. Pigounakis coded the curve fairing algorithms that appear in his Ph.D. thesis. T.P. Gerostathis implemented the simple and composite Bezier curves and surfaces as well as the surface fairing algorithms of his diploma thesis. K. Kostas coded the optimal parametrizations he developed for his diploma thesis. V. Voutsinas implemented the isophotes, highlights and reflection lines. S. Kapniaris implemented the curve fairing methods of his diploma thesis and I. Sarantidis coded the porcupine plots.

References

APPLEGARTH, I., KAKLIS, P.D. AND WAHL, S. 2000 Benchmark Tests on the Generation of

Fair Shapes subject to Constraints. B.G. Teubner, Stuttgart-Leipzig, approximately pp. 70

FARIN, G. 1997 Curves and Surfaces for Computer-Aided Geometric Design, A Practical Guide. Fourth Edition, Academic Press, New York

GEROSTATHIS, T.P. 1997 Non-linear Variational Techniques for Fairing Tensor Product Bezier and B-spline Surfaces. Diploma Thesis, Dept. Naval Arch. & Mar. Engrg, National Technical University of Athens

GEROSTATHIS, T.P., KORAS, G.D. AND KAKLIS, P.D. 1999 Numerical Experimentation with the Roulier-Rando Fairness Metrics. Mathematical Engineering in Industry, **7, 2**, pp. 195-210

GREINER, G. 1999 Modeling of Curves and Surfaces Based on Optimization Techniques, in Creating Fair and Shape-Preserving Curves and Surfaces. H. Nowacki and P.D. Kaklis (eds.), B.G. Teubner, Stuttgart-Leipzig, pp. 11-27

GULDHAMMER, H.E. 1962 FORMDATA: Some Systematically Varied Ship Forms and their Hydrostatic Data. Danish Technical Press, Copenhagen

HOSCHEK, J. AND KAKLIS, P.D. 1996 Advanced Course on FAIRSHAPE. B.G. Teubner Stuttgart-Leipzig, pp. 288

HOSCHEK, J. AND LASSER, D. 1993 Fundamentals of Computer Aided Geometric Design. A.K. Peters, Wellesley, Massachusetts

KAKLIS, P.D. AND KORAS, G.D. 1998 A Quadratic-Programming Method for Removing Shape-Failures from Tensor-Product B-spline Surfaces. Computing Supplement, **13**, pp. 177-188

KAKLIS, P.D. AND KORAS, G.D. 1998 Convexity-Preserving Fairing of Parametric Tensor-Product B-spline Surfaces. presented in "Freiformkurven und Freiformflaechen", Mathematical Institute of Oberwolfach

KAPNIARIS, S. 1995 Part A: Fairing of 5th degree Parametric Spline Curves, Part B: Fairing of B-spline Curves under Constraints-Production of Faired Ship Lines under Shape and Integral Constraints-Application in Ship Design. Diploma Thesis, Dept. Naval Arch. & Mar. Engrg, National Technical University of Athens

KOENDERINK, J.J. 1990 Solid Shape. The MIT Press, Cambridge, Massachusetts

KORAS, G.D. AND KAKLIS, P.D. 1999 Control Point Loci for Locally Convex Parametric Surfaces. presented in "Sixth SIAM Conference on Geometric Design", Albuquerque, USA

KORAS, G.D. AND KAKLIS, P.D. 1999 Convexity Conditions for Parametric Tensor-Product B-spline Surfaces. Advances in Computational Mathematics, **10**, pp. 291-309

KOSTAS, K. 1998 Optimum Parameterization for Tensor Product B-spline Surfaces. Diploma Thesis, Dept. Naval Arch. & Mar. Engrg, National Technical University of Athens

NAG LTD. 1990 The NAg Fortran Library Manual-Mark 14, (1st ed.). UK

NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY 1997 Initial Graphics Exchange Specification (IGES). , JK468.A8 A3 no.177-1 1996, Gaithersburg, MD

NEIDER, J., DAVIS, T. AND WOO, M. 1994 OpenGL Programming Guide.) Release 1, Addison-Wesley, Massachusetts

NOWACKI, H. AND KAKLIS, P.D. 1998 Creating Fair and Shape-Preserving Curves and Surfaces. B.G. Teubner Stuttgart-Leipzig, pp. 288

PIGOUNAKIS, K.G. 1997 Fairing Methods of Planar and Space Curves under Design Constraints-Applications in Computer-Aided Ship Design. Ph.D. Thesis, NTUA, Athens

PIGOUNAKIS, K.G. AND KAKLIS, P.D. 1996 Convexity-Preserving Fairing. CAD, **28, 12**, pp. 981-994

PIGOUNAKIS, K.G. AND KAKLIS, P.D. 1999 Fairing of 2D B-splines under Design Constraints. Mathematical Engineering in Industry, **7, 2**, pp. 165-178

PIGOUNAKIS, K.G. AND KAKLIS, P.D. 1997 "Smoothing Spatial Cubic B-splines under Shape Constraints" in Curves and Surfaces with Applications in CAGD. A. Le Mehaute C. Rabut and L.L. Schumaker (eds), Vanderbilt University Press, pp. 345-354

PIGOUNAKIS, K.G., SAPIDIS, N.S. AND KAKLIS, P.D. 1996 Fairing Spatial B-spline Curves. Journal of Ship Research, **40**, **4**, pp. 351-367

SARAIDARIS, J. 1998 FORMDATA Standard Series Fairing. Diploma Thesis, Dept. Naval Arch. & Mar. Engrg, National Technical University of Athens

VOUTSINAS, V. 1997 Computer Aided Fairness Estimate of Surfaces. Diploma Thesis, Dept. Naval Arch. & Mar. Engrg, National Technical University of Athens