CONSTRUCTION OF A MANIFOLD SHIP HULL MODEL, DERIVED FROM A WIREFRAME MESH

Herbert J. Koelman
Scheepsbouwkundig Advies en RekenCentrum (SARC) BV
The Netherlands
sarc@sarc.nl

Willem B. Soede
Scheepsbouwkundig Advies en RekenCentrum (SARC) BV
The Netherlands
sarc@sarc.nl

ABSTRACT

In naval architecture it is common practice to create a new design by means of a transformation or a modification of an existing initial or final design. Those existing designs may be available in digital format, or on a drawing which can be digitised. The lowest common denominator of the encountered digital formats is an unconnected collection of spatial curves, which can easily be converted into a wireframe representation. In order to enable these designs to be utilised in a ship hull CAD system which is based on a Boundary Representation solid model, a method must be available which converts that wireframe into a solid model. This paper gives a short survey of existing conversion methods, and identifies a shortest path approach as most appropriate for our goal. The original shortest path method is modified for 2-connected objects, and objects with many-sided faces, which are quite common in naval architecture. Finally an application example is presented, and the approach is evaluated.

KEYWORDS

Wireframe-solid conversion, Solid model, BREP construction, Ship hull modelling, Ship hull re-engineering.

1. INTRODUCTION

Although ship hull design practices vary from place to place, and from person to person, essentially, in general three hull design methods can be distinguished :

- *Ab initio* design (arbitrary free form design);
- Hull form generation, using mathematical formulae, or other automated processes;
- Transformation of a parent hull.

In recent years, we have been designing and implementing a CAD system for the design and engineering of ship hull forms, with the ambition to support all three hull design methods. This system, which is called Fairway and is described into detail in Koelman, H.J. (1997) and (1999), has in its current implementation all required functionality for *ab initio* design and hull form transformation. However, in many practical ship design situations, the transformation must be applied on some parent hull which is not readily available in Fairway format, but only on paper or in some digital format. So an import facility, including necessary conversion functions, was considered desirable.

In particular, the conversion functions must be capable to handle the following formats:

- Fairway's fundamental data structure is a solid model, represented with a Boundary Representation (BREP). The BREP is extended with support for curved curves and surfaces. In this respect Fairway differs from many traditional ship hull design systems, which use a parametric surface as a basic modelling entity;
- Ship designs may be available in committee-designed standards, such as IGES (IGES, 1993) or STEP (ISO, 1997), but in the majority of cases the carrier is DXF, which is the industry standard (at least in the Netherlands, where the marine industry is structured around quite a large number of small to medium-sized shipyards). Also propriety formats are

sometimes used. The used geometry representations are polylines, NURBS curves, NURBS surfaces and occasionally a specific equation to describe an elementary shape (such as a circular arc, or a parabola);
- Line drawings can be digitised, and the resulting curves can be assigned their spatial orientation and position. Those curves can be represented by polylines, splines or other representations.

For a broad field of application we have to use the lowest common denominator of all involved formats, and search for a method to import that format into Fairway. It appears that the *curve* entity can be used for that purpose. After all, the curve is one of the constituents of Fairway, while on the other hand curves are available in the imported formats, or can easily be derived from it (as in the case of NURBS surfaces).

In order to give a proper representation of the hull, the curves must be spatial, while the combination of curves must form a consistent and complete network. Based on geometrical neighbourhood testing such a network can readily be converted into a wireframe model.

What remains is the task to construct a solid model from that wireframe, which is the subject of this paper.

2. DEFINITIONS

This section contains a list of definitions, which is extracted from Agarwal, S.C. and Waggenspack Jr., W.N. (1992), Koelman, H.J. (1999) and Loseries, F. (1997) :

A *shell* is the boundary of a solid.

A *face* is a finite and non-self-intersecting part of a shell.

An *edge* is a non-self-intersecting boundary of a face, bounded by two vertices.

A *vertex* is the topological entity which corresponds to a metric point.

A *curve* is a point-bounded collection of points in 3-D Euclidian space, represented by a G^N continuous (which means with geometrically continuous derivatives up to degree N) function.

A *polyline* is an ordered sequence of straight line segments.

A polyhedral Boundary REPresentation, or *polyhedral BREP*, represents an object by the relationships between adjacent faces, edges and vertices.

An *HREP* (Hybrid model for ship hull REPresentation) is a BREP, extended with :
- Continuous curves and polycurves;
- Curved surface geometry.

A *path* is an alternating sequence of vertices and edges (vertex - edge - vertex - edge) that initiates at one vertex, and ends at another, and in which no vertex or edge occurs more than once.

A *cycle* is a path in which the initial vertex is also the final vertex.

An *internal cycle* is a cycle which does not lie on the shell of a solid.

A *graph* is a finite collection of edges and vertices.

A graph is *connected* if between every pair of vertices a path exists.

A graph is *N-connected* if at least N vertices (and their incident edges) must be removed to make the remaining graph disconnected.

A *fundamental cycle set* is a set of cycles in which at least one edge in each cycle is not contained in any other cycle.

3. WIREFRAME TO SOLID CONVERSION APPROACHES

From literature three approaches to convert a wireframe into a solid are known :
- Simplicial decomposition;
- Graph embedding method;
- Cycle space method.

3.1. Simplicial decomposition

With this approach a solid is considered as a composition of basic volumetric building blocks, that is tetrahedra. The algorithm of Agarwal, S.C. and Waggenspack Jr., W.N. (1992) converts a wireframe into a solid in three steps :

1. Decomposition of the wireframe into a set of tetrahedra;

2. Generation of a list of triangular faces from the tetrahedra, and removal of all duplicate triangular faces;

3. Merging the triangular faces which share a common edge to obtain the final faces.

A first remark which can be made about this method is that the first step, detection of the tetrahedron, requires geometrical data. Secondly, with a large number of faces the processing times might become be prohibitive,. because this method has exponential running time, as stated in Bagali, S. and Waggenspack Jr., W.N. (1995),

3.2. Graph embedding method

This method treats the wireframe model as an edge-vertex graph, and uses planar embedding testing to locate the faces of the wireframe. The method applies only to 3-connected planar graphs, and has linear running time (Bagali, S. and Waggenspack Jr., W.N. , 1995).

3.3. Cycle space method

This method relies on the use of fundamental cycles, and the use of heuristics to find fundamental cycles of minimum length.

In Bagali, S. and Waggenspack Jr., W.N. (1995) a variation on this method is described, which is called the 'Shortest path approach'. This method is based on the generation of short cycles, and the assumption that if these cycles do not make the wireframe unconnected, they may be considered as faces of the solid. Also this (quadratic running time) method applies only to 3- or more-connected planar graphs.

3.4. Selection of an appropriate method

Our selection of a method was based on three considerations :

- Reasonable running time;
- Suitable for all objects which might be encountered in ship design practice;
- Transparency.

The Shortest path method of Bagali & Waggenspack scores best on these criteria, except for the fact that it can only be applied on 3-connected objects. However, later on we will present a variant which can handle 2-connected objects.

4. THE ORIGINAL SHORTEST PATH APPROACH

In this section we present an overview of the original Shortest path method, and an example of the application of this method on a ship hull.

4.1. Basics of the algorithm

For all edges *e* of the wireframe two steps are performed :

1. A cycle generation step. This step finds all cycles which contain *e*, and which have no other edges in common. A cycle can be of the following type :

- True face;
- Internal cycle;
- Pseudo-cycle, which is the sum of two or more shorter cycles.

2. A face generation step, where internal cycles and pseudo-cycles are rejected, and only true faces are used as a valid face. Prior to this evaluation of cycles, they are sorted in increasing order of their (topological) length, in order to consider short cycles for inclusion as face before long cycles, so pseudo-cycles will be avoided as much as possible. As a deviation from this general rule, critical cycles (which are cycles which contain edges that have only been used once until so far) are evaluated before non-critical cycles.

The face generation step is performed with the function GENERATE_FACE :

function GENERATE_FACE(cycle *c*);
start of function
  if The addition of *c* to the list of faces does not make any edge being used more than twice
  and *c* is non-separable (which means that the removal of the cycle from the wireframe does not leave the wireframe unconnected. This condition was included to prevent the occurrence of internal cycles) **then**
    Add *c* to the list of faces;
  end if
  Remove *c* from the list of cycles;
end of function

While the complete algorithm is:

function ORIGINAL_CONSTRUCT_SOLID;
start of function

for edge *e* = 1 to total number of edges **do**
begin
Identify bounding vertices V1 and V2 of edge *e*, and generate as many paths, without any common edge, between V1 and V2 with Dijkstra's single source shortest path algorithm (Morris, J., 1998);
Convert each path into a cycle, by addition of *e*;
Sort the cycles in increasing order of their length.
for cycle *c* = 1 to number of cycles **do**
if *c* is critical **then**
GENERATE_FACE(*c*);
end if and for
for cycle *c* = 1 to number of cycles **do**
GENERATE_FACE(*c*);
end for
Mark *e*, so it is excluded from further usage in other cycles.
end for
end of function

As an exception, the last face is not generated in the regular way, instead it is the face bounded by edges which have been used exactly once.

4.2. The algorithm, applied on a ship hull

A (heuristic) assumption of this algorithm is that actual faces are always topologically short, which means they are bounded by a few edges only. However, in ship design practice a vessel is frequently represented by its port side or starboard semi-model only, because the other side of centre plane is symmetrical. In Figure 1 an example is shown, which demonstrates that the centre plane face of this object is bounded by more than hundred edges, and thus far from topologically short. Because this conflicts with the assumption, the proposed algorithm may find no solution.This fact can be demonstrated further with the hull of Figure 2:

Suppose in an early stage of the solid construction process, edge *e* under consideration is E4. Then the shortest path algorithm may find cycle V1 - V2 - V3 - V4 - V1. The edges of this cycle are marked 'unavailable' for the next

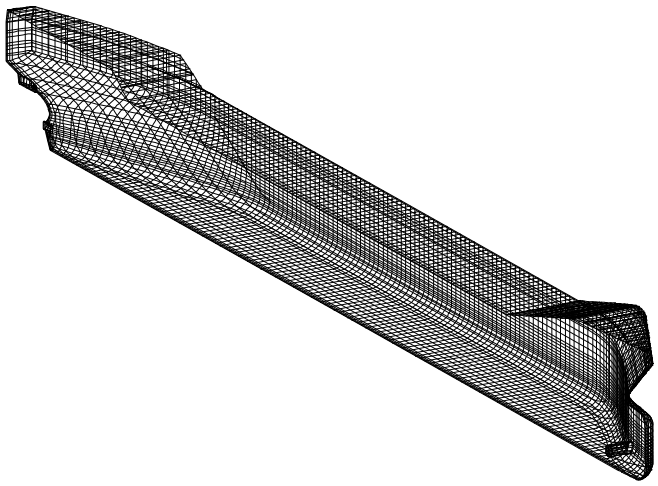

Figure 1. Starboard semi-hull, with closed deck (Courtesy of E. de Lint, Nijmegen)

cycles (except for E4 itself), so the next generated cycle cannot include the edges V1-V4, V3-V4 and V2-V3, but must include E3 and E5. This second cycle can, for example, be V1 - V2 - V5 - V6 - V7 - V8 - V9 - V10 - V11 - V12 - V13 - V1,

which is rejected as a valid face because it leaves the solid disconnected at removal. It could also be V1 - V2 - V5 - V6 - V3 - V9 - V10 - V4 - V12 - V13 - V1, which is (incorrectly) included as a valid face because it is a non-separable cycle.

Anyhow, after E4 is processed, it is excluded from further inclusion in cycles, so the valid closing face which consists of the edges E1 - E2 ... E9 - E10 etc. will not be found by the regular algorithm.

According to the last step of paragraph 4.1, the closing face is finally found, but unfound faces still remain if more than one topologically large face is present.

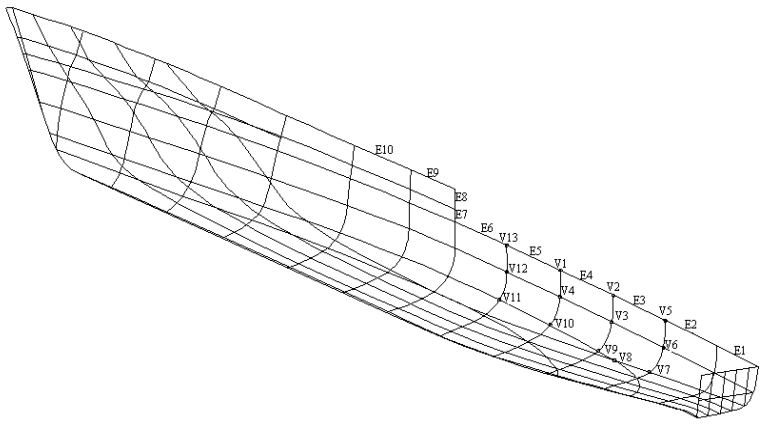

Figure 2. Shortest path applied to a ship hull

5. MODIFIED SOLID CONSTRUCTION ALGORITHM

The example of the previous section illustrates the necessity to modify the algorithm, in order to be suitable for many-sided faces. This modified algorithm is valid for a genus 0 manifold object, and iterates until Euler's equation F = 2 + E - V (where F is the number of faces, E the number of edges and V the number of vertices) is satisfied :

```
Function CONSTRUCT_SOLID_3_CONNEC-
TED;
start of function
   repeat
     for e = 1 to total number of edges do
      if e does not occur more than once in
       already existing face boundaries  then
         Identify bounding vertices V1 and V2 of
         edge e, and generate as many paths,
         without any common edge, between V1
         and V2;
         Convert each path into a cycle, by addition
         of e;
     end if and for
     Sort all cycles in order of ascending
     topological length;
     for cycle c = 1 to number of cycles do
      if c is critical then
          GENERATE_FACE(c);
      end if and for
     for cycle c = 1 to number of cycles do
       GENERATE_FACE(c);
     end for
   until F = 2 + E - V
end of function
```

This algorithm constructs solids from a given wire-frame, even if they contain many-sided faces. It also finds the closing faces in a regular way, so there is no reason for a final, exceptional step in order to find the closing face, as in paragraph 4.1.

However, just as in the original algorithm, application is limited to 3- or more-connected objects. The reason can be demonstrated with the 2-connected object of Figure 3, which can often be found in literature: every cycle containing the edges E1 and E2 are judged to be invalid as a face, because they leave the small part unconnected from the larger part. This inhibits the front and top faces from being recognised as valid faces.

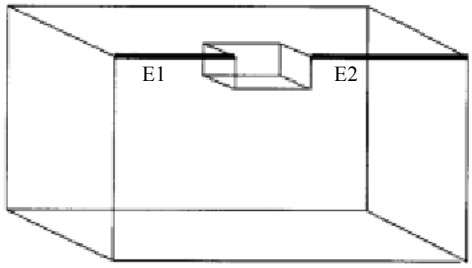

Figure 3. A 2-connected object

6. 2-CONNECTED OBJECTS

At first sight the object of Figure 3 could be considered as a freak object, which is of little significance to shipbuilding practice. However, consider the very simple schematic initial ship hull design of Figure 4. This one is highly 2-connected, because 24 combinations of vertices exist, which leaves the object separated at deletion (vertex-pairs 2-3, 4-5, 6-7, 2-5, 3-4, 4-7, 5-6 etc.). Although extra longitudinal sections, but even the inclusion of *one* additional edge, between the vertices 1 and 20, would make this object not 2-connected anymore, it is clear that for practical ship design application we cannot ignore 2-connected objects. Please consider in this respect also 'normal' ship hull wireframe models, which may contain 2-connected parts, for example, in the stern or bulb regions.

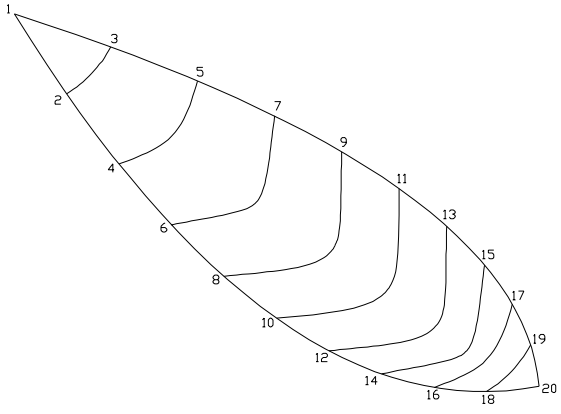

Figure 4. Simple bodyplan

For our algorithm being able to handle 2-connected objects, it was refined under the following consideration :

1. The fact that 2-connected objects are not suitable for our algorithm, until so far, stems from the non-separability requirement;

2. The non-separability requirement was included to reject internal cycles to be used as a face;

3. But if a cycle separates a 2-connected object, it cannot be an internal cycle. This new assumption is not valid for the object of Figure 3, but in many cases this is a practical way to look at 2-connected objects.

So, in case of a 2-connected object, the final algorithm omits the separability test for cycles which contain the separable edges :

```
function CONSTRUCT_SOLID_FINAL;
start of function
  CONSTRUCT_SOLID_3_CONNECTED;
  if F ≠ 2 + E - V then
    Create a list L of all edges that make the object
    disconnected;
    repeat
      Number of cycles = 0;
      for e = 1 to total number of edges do
       if e does not occur more than once in
         already existing face boundaries then
         Identify bounding vertices V1 and V2 of
         edge e, and generate as many paths,
         without any common edge, between V1
         and V2;
         Convert each path into a cycle, by addition
         of e;
       end if and for
      Sort all cycles in order of ascending
      topological length;
      for cycle c = 1 to number of cycles do
        if c is critical then
          GENERATE_FACE_2(c);
      end if and for
      for cycle c = 1 to number of cycles do
        GENERATE_FACE_2(c);
      end for
    until F = 2 + E - V or no additional faces are
    created anymore
  end if
end of function
```

This algorithm uses a slightly modified GENERATE_FACE :

```
function GENERATE_FACE_2(cycle c);
start of function
  if the addition of c to the list of faces does not
  make any edge being used more than twice
  and (c is non-separable or any edge of c is
  member of list L) then
    Add c to the list of faces;
  end if
  Remove c from the list of cycles;
end of function
```

7. IMPLEMENTATION, AND APPLICATION EXAMPLE

The proposed algorithm has been implemented in the Fairway system. In the ship design process the construction of the solid model is a one-off event, and there is no need for interactive response times. This consideration allowed us to choose simple auxiliary algorithms, such as :

- Sorting the cycles with a 'Heapsort' algorithm (Press, W.H. et al., 1986);
- The non-separability test with a *Depth First Search* algorithm (Eppstein, D., 1996);
- Dijkstra's algorithm in its basic form (Morris, J., 1998).

For these three tasks more calculation-efficient algorithms do exist, but, because the present performance is considered adequate, these have not been used.

The conversion module itself performs 3 steps :

1. Read DXF or IGES files containing spatial curves. Those curves may be represented by polylines or by NURBS;

2. Construct a wireframe model, based upon these curves. This is performed with a simple (geometric) neighbourhood test. The wireframe model is represented by lists of vertices and edges;

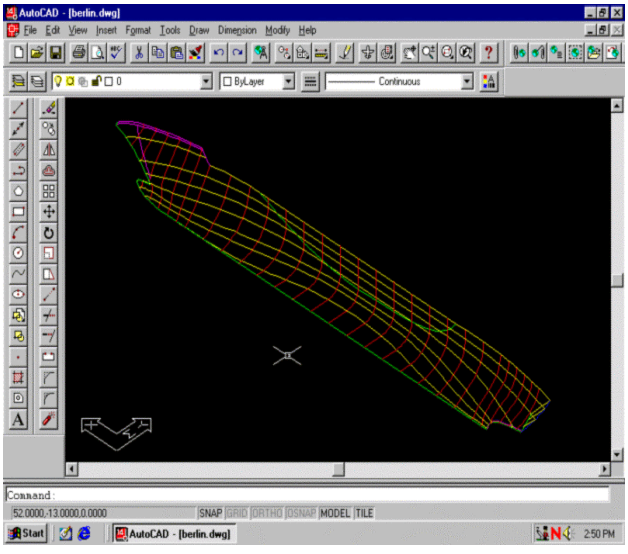

Figure 5. View of polylines of DXF file of my. 'Berlin'

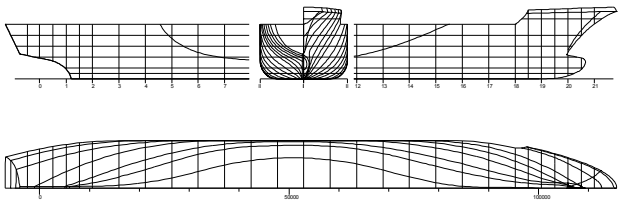

Figure 6. Wireframe model, constructed from the collection of spatial curves

3. Construct a solid model, with the proposed algorithm. The result of this construction process is an HREP, containing lists of curves, polycurves, vertices, edges and faces. Subsequently Fairway can import this HREP, and generate, display and process the geometry of the curved faces.

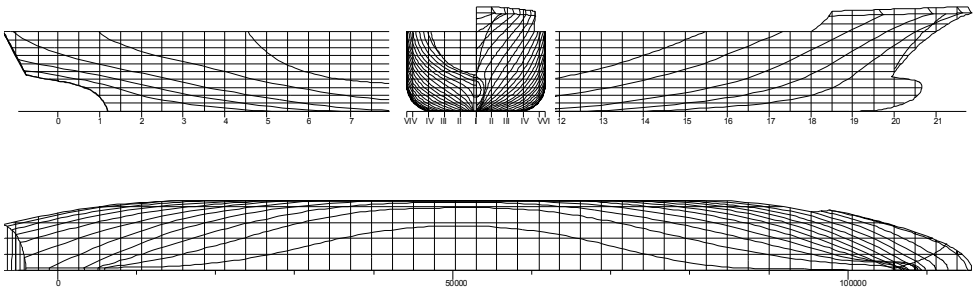

Figure 7. Final bodyplan, completed with interpolated sections

This module was implemented in Pascal, under Windows 95/98/NT/2000, and includes about 7000 lines of source code. Some (current) maxima are : number of vertices 12000, edges 24000, faces 12000 and cycles 48000. In order to demonstrate the approach we present an example, based upon the hull form presented in Bartlick et al. (1980). If this hull form of the 105.90 x 15.00 x 11.50 m 'Berlin' cruise vessel is available in DXF format, represented by polylines with an average (straight) line length of 0.25 m, as shown in Figure 5, then the construction process comprises the following steps (with times measured on a 350 Mhz Pentium II PC)

1. Importing the 403 KB of DXF file, creation of NURBS curves through the polyline points, and collecting single-connected curves in poly-curves. This step results in 24 sec. in 50 spatial polycurves, consisting of a total of 117 curves;.

2. Generation of wireframe, with a (user-specified) tolerance of 0.025 m to allow for inaccuracies in the supplied data, results in 135 sec. in a wireframe of 273 vertices and 500 edges;

3. The solid construction process results in 7 sec. in a BREP of 273 vertices, 500 edges and 229 faces.

The imported curves are shown in Figure 6, while Figure 7 shows a completed lines plan, including additional sections which have been interpolated on the curved hull surface. Figure 8 shows the shell of the solid model, as well as the design waterline.

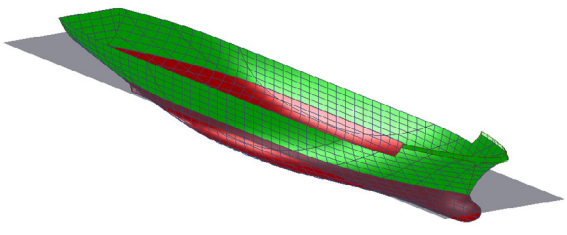

Figure 8. Rendered view of solid model of ms. 'Berlin'

8. THE CONVERSION PROCESS IN PRACTICE

The methods described in this paper have been in commercial use for 6 months now, and in that period quite some experience has been gained. The first fact is that, given a valid and 2-connected or more-connected wireframe model, the conversion method as described in this paper has been performed well for quite a number of hull form types. A possible pitfall in this respect is that a wireframe may be valid, but only 1-connected, while due to the rather dense wireframe model, that area of 1-connectivity is not immediately recognised by the program user. An example of a vessel where this phenomena occurs is displayed in Figure 9, with the area of 1-connectivity magnified in Figure 10. If an area of 1-connectivity exists, the program user is left no other option than to make it 2-connected by adding extra curves.

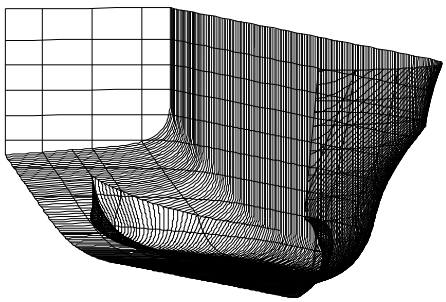

Figure 9. Wireframe model of vessel with bulbous bow and stern bulbs

Another area of concern in practice is the variety of data types and co-ordinate systems which can be used in an IGES or DXF file. For practical purposes our software can only recognise a subset of all possible DXF or IGES entities, and for an average user it is not always easy to determine whether a particular file is suitable.

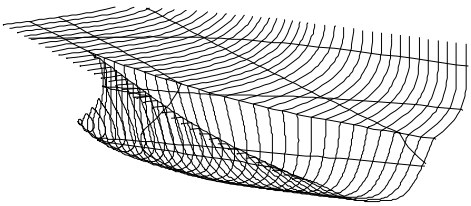

Figure 10. Magnification of stern bulb, showing the 1-connectivity of the bulb's extremity

Despite these points of interest, our overall impression is that with some explanation, and a detailed user-manual, the proposed method is suitable for general use in shipbuilding practice.

9. CONCLUSION

For ship design practices the need of a tool which constructs a solid model from a wireframe representation has been argued. The class of cycle space methods, especially the method of Bagali, S. and Waggenspack Jr., W.N. (1995), provides such methods, which are conceptually simple and efficient for practical purposes.

However, it was demonstrated that faces bounded by many edges, such as appear in ship designs, are not processed properly, and that 2-connected objects cannot be neglected. This paper presented a modified algorithm to overcome these deficiencies.

The proposed method is already in practical use, and with an aware user it can be a helpful tool in the ship design process.

REFERENCES

Agarwal S.C. and Waggenspack Jr. W.N. (1992) 'Decomposition method for extracting face topologies from wireframe models'. Computer-Aided Design volume 24, No. 3, pp. 123-140.

Bagali, S. and Waggenspack Jr., W.N. (1995) 'A shortest path approach to wireframe to solid model conversion'. ACM Symposium on Solid Modeling and Applications, Proceedings of the third symposium on Solid modeling and applications, May 17 - 19, Salt Lake City, UT USA, pp. 339-350.

Bartlick, H., Ganz, K., Heise, R., Klehe, J., Wilcken, A. and Wollmerath, H. (1980) 'Kreuzfahrtschiff "Berlin"'. HANSA, 117 Jahrgang, No. 24, pp. 1879-1887.

Courter, S.M. & Brewer, J.A. (1986). 'Automated Conversion of Curvelinear Wire-Frame Models to Surface Boundary Models; A Topological Approach' ACM SIGGRAPH '86, Vol. 20, No. 4, pp. 171-178.

Eppstein, D. (1996) 'ICS161: Design and Analysis of Algorithms. Lecture notes for february 15, 1996'. Web: www.ics.uci.edu/~eppstein/161/960215.html.

IGES (1993) 'Initial Graphics Exchange Specification'. U.S. Product Data Association.

ISO (1997) 'ISO TC184/SC4/WG3 N 571 (T12): Application Protocol: Ship Moulded Forms'. Web: www.nist.gov/sc4/, 14 February.

Koelman, H.J. (1997) 'Hull form design and fairing: tradition restored', The proceedings of The Sixth International Marine Design Conference. Sen, P. and Birmingham, R., editors. June 23-35, Newcastle upon Tyne, UK.

Koelman, H.J. (1999) 'Computer Support for Design, Engineering and Prototyping of the Shape of Ship hulls'. Ph. D. thesis, Delft University of Technology, Delft, The Netherlands.

Loseries, F. (±1997) 'Hybrid objects: A construction algorithm for topological cycles based on algebraic information'. Web: www.zgdv.de/departments-/z4/projectsandpublications/hybrid-objects.htm.

Morris, J. (1998) 'Dijkstra's Algorithm'. Web: odin.ee.uwa.edu.au/~morris/Year2/PLDS210/dijkstra.html.

Press, W.H., Flannery, B.P., Teukolsky, S.A. and Vetterling, W.T. (1986) 'Numerical recipes'. Cambridge University press. pp. 229-232.