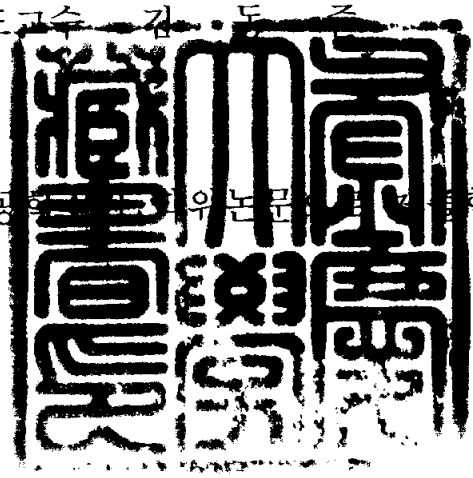
공학석사 학위논문

선형의 곡면 모델링과 곡면 순정에 관한 연구

지도교수 김 동 준



이 논문을 공학석사 학위논문으로 제출함

2004년 2월

부경대학교 대학원

조선해양시스템공학과

이 무 열

이무열의 공학석사 학위논문을 인준함

2003년 12월

주 심 공학박사 김 인 철 (印)

위 원 공학박사 김 용 직 

위 원 공학박사 김 동 준 

A Study on the Surface Modeling of Hull Form and the Surface Faring

Dept. of Naval Architecture & Marine System Engineering **Moo-Yeol Lee**

Directed by Professor **Dong-Joon Kim**

Abstract

In ship hull design, if the geometrical shape of ship hull can be defined effectively as three dimensional surface, the property of surface such as Gaussian curvature can be evaluated easily and the physical quantity such as volume, surface area, elasticity moment etc. can be calculated easily. This is the reason, many of hull form's surface modeling studies are proceeded.

The developed program in this study can create usable surface easily and quickly from three-dimensional wire-frame data for engineering production. It is a updated version of the early developed program, so it can be used for "surface creation" and "surface fairing".

The program based on MFC and OpenGL creates and shows the curves and surfaces by optimum algorism. The NURBS surface set that can be used for fairing by using NURBS curve net are made. In this process automating of surface creation process is available. Finally, by moving curve's data point that makes surface and confirming faring result through quick surface's changes, hull surface faring method can be implemented.

Programs are developed by using Visual C++, MFC and powerful graphic library OpenGL.

목 차

I. 서론

1. 개요

일반적으로 복잡한 자유곡면으로 이루어진 선형으로부터 주어지는 데이터는 3D(3-Dimension) 공간상의 점이거나 그 점들을 지나는 곡선들로 정의할 수 있다. 이때 주어지는 최소한의 데이터로부터 도면을 생성하는 것이 전통적인 선형설계의 방법이다.
전산장비와 CAD(Computer Aided Design) 기술의 발달과 더불어 선형을 정의하는 방법도 변화해 왔다. 범용 CAD의 경우 현재 솔리드 모델링을 넘어 그 이상의 영역을 모색하고 있다. 조선업에서도 그에 발맞추어 구조 및 의장설계의 상당 부분에서 CAD 기술은 상당한 발전을 보여 왔다. 각 조선소에서는 경쟁적으로 3D (Solid) Modeling CAD 시스템을 도입하려는 시도를 하고 있으며 이미 적용된 사례도 적지 않다. 그러나 선형설계 부분에서는 2D Lines 혹은 TRIBON과 같은 Wire Frame 모델링 시스템을 사용하고 있다. 이러한 선형모델과 구조 및 의장 모델간의 데이터형의 불일치 문제는 현재 각 조선소가 지향하는 PDM(Product Data Management)의 도입을 추진하는데 있어서 설계 생산성에 좋지 않은 요소로 작용하고 있다. 이러한 문제를 해결하는 방법으로 선형을 곡면으로 모델링하는 방법이 제시되었으며, 곡면 모델에 약간의 정보를 추가하면 Solid 모델로 나타낼 수 있게 된다.

공학설계에 있어서 곡면의 기하학적인 형상을 효과적으로 표현하고, 또 제작하는 문제는 오랜 기간 큰 관심의 대상이었다. 이는 공학적 설계대상을 3차원 곡면으로 직접 정의하게 되면 곡률 등과 같은 곡면의 성질을 쉽게 계산할 수 있으며, 부피, 곡면적, 탄성 모멘트 등과 같은 곡면에 관련된 물리량을 빠르고 비교적 쉽게 해석할 수 있기 때문이다. 곡면을 생성하는 방법은 주어진 데이터의 형태나 설계하려는 대상 혹은 적용분야에 따라 그 방법을 다르게 선택한다(박지선 등, 1994)

선형의 곡면화 방법에는 이전에도 많은 연구가 있어 왔다. 국내에서는 윤병호 등(1985)이 B-Spline 곡면을 이용하여 선체표면을 수치적으로 표현하는 연구를 하였다. 이후 김수영 등(1992)은 B-Spline 곡면식을 이용하여 3차원 자유곡면 생성을 시도하였고, Coons 곡면과 B-Spline 곡면과의 결합하는 방법을 사용하였다. 그러나 곡면을 생성하는 과정에서 파생된 곡면간의 결합문제와 각각의 곡면간의 접속문제는 해결과제로 남겨두었다. 박지선 등(1994)은 곡선망 선형을 이용하여 곡면간 기하학적 연속이 만족되는 곡면으로 선형을 정의하였다. 곡면을 생성하기 위해 Hermite 혼합(Bending), Coons Patch, Gregory Patch 보간방법 등을 적용하였다. 또한 임중현 등(1997)은 de Casteljau(드 카스텔죠 알고리듬)와 Bezier 조정점을 이용하여 자유곡면을 표현하였다. 그러나 이 두 방법은 생성된 곡면을 NURBS(Non-Uniform Rational B-Spline) 곡면으로 변환하기 어렵다는 점을 해결과제로 남겼다. 일반적인 CAD System에의 적용을 위해서는 곡면을 NURBS 곡면으로 정의하는 것이 필요하다. 다시 말하면 선형설계 시 생성된 선형 곡면정보가 후행 설계공정에 정확히 전달되기 위해서는 NURBS 곡면으로 정의하는 것이 필수적이다. 최근의 연구 중, 신현경 등(2002)은 선체형상의 자료점을 조정점으로 가정하는 방법을 사용하여 NURBS 곡면으로 정의하는 방법을 제시하였다.

선형을 곡면화하기 위해서는 주로 주어진 Lines 혹은 Wire frame 모델로부터 경계곡선을 얻어서 이를 이용하여 선형을 곡면화 방법 등이 이용된다. 선형을 수치화된 곡면으로 표현하기 위해 몇 가지 고려되어야 할 사항들이 있다. 첫 번째로 일반적인 상선의 경우 곡면이 다트립 형상을 가짐으로써 하나의 곡면 Patch로는 정의가 되지 않는다는 점과 두 번째로 각 곡면 Patch 간 자료점의 개수가 일치하지 않는다는 점이다. 따라서 곡면을 생성하기에 앞서 자료점의 개수를 일치시켜야 한다. 마지막으로 Section으로 구성된 곡선망으로 곡면을 생성할 경우, 사각 Patch 이외에 삼각형 혹은 오각형 등의 다각형 Patch가 나타나게 되므로 형상의 정의가 어렵다. 이러한 문제점은 이준호(2003)의 연구에서 문제가 제기되고 일부에 대해 해결방안을 모색하였다.

본 연구에서는 더 나아가 선형의 Wire Frame 모델의 3차원 곡선망을 이용하여 NURBS 곡면을 생성할 수 있는 프로그램을 작성하였다. 본 연구에서 개발한 프로그램은 MFC(Microsoft Foundation Class)와 OpenGL(Open Graphic Library)을 사용하여 작업과정을 시각적으로 표시하였으며 사용자와 컴퓨터 간에 Interactive 한 접근이 가능하도록 하였다. 또한 사용자가 곡선이나 곡면이론에 대하여 잘 알지 못하는 경우에도 간단하게 사용할 수 있도록 대부분의 작업을 자동화하여 최소한의 조작으로 곡면을 생성 및 순정이 가능하도록 제작되었다.

2. 연구배경 및 내용

현재 조선소의 선형설계부서에서는 대부분 2D Lines 혹은 Wire frame 모델을 이용하여 설계를 수행하고 있다. 그러나 선형을 곡면으로 정의하게 되면 선형설계에서 획득한 정보를 설계의 많은 분야에서 효율적으로 이용이 가능하게 된다. 이는 선형설계 이후의 설계에 있어서 부가되는 작업을 줄일 수 있어 설계 시수 절감에도 많은 도움이 된다.
이에 선박설계 전용 프로그램을 제작하는 업체들에서는 선형을 곡면화하는 기술을 개발하여 현재 가시적인 결과를 내고 있다.
대표적인 예가 선박설계 전용 CAD 시스템 개발사인 스웨덴의 TRIBON(이전 KCS)과 핀란드의 NAPA System, 그리고 국산 이지그라프사의 EzHull 등이 있다. TRIBON의 경우, 2003년에 기존의 Wire frame 기반의 TRIBON을 대체하는 PIM Soluton TRIBON M2를 발표하고 서비스에 들어갔다. 이번 TRIBON M2에서는 선형의 곡면모델링을 전면적으로 지원하고 있으며 현재 다른 선형 곡면 Modeler보다 우수한 성능을 지닌 것으로 추정된다(TRIBON Homepage). 국내의 EzHull도 곡면 모델링을 지원하지만 선형 곡면화를 위해서 새로 Mesh를 구성해야한다는 점이 단점으로 작용한다(EzGraph Homepage). 곡면 품질에 대해서는 비교 자료가 없는 관계로 판단을 유보한다.

초기 설계단계에서는 Simulation이나 기타 전시목적으로 선형의 형상 Model이 필요한 경우가 많으며 그 수요는 계속해서 증가하고 있다. 이런 경우 선형정보를 가지고 생성된 곡면의 품질보다는 최소한의 선형정보를 가지고 쉽고 빠르게 곡면을 생성하는 것이 더욱 중요하다.
이전의 연구에서는 선형의 순정과 곡면화을 위한 프로그램이 박성우(2000)에 의해 연구된 바 있다. 그러나 박성우(2000)의 연구에서 제작된 프로그램의 경우, 한 프로그램에 너무 많은 기능이 집약되어 있고 곡면을 생성하기까지 많은 단계와 조작이 필요하며 기본적인 곡면 이론을 숙지하고 있어야 한다는 단점이 있다. 또한 이렇게 생성된 곡면을 가시화하기 위해서는 단면곡선을 생성하여 보여줌으로써 가능하였다. 이는 직관적으로 곡면을 보고 검증할 수 없다는 단점을 가진다. 이후 이준호(2003)가 기존의 프로그램을 개선하였으나 여전히 곡면을 생성하는데에는 번거로운 과정을 거쳐야 했으며 곡면의 가시화를 위해 상용 프로그램을 사용하여야만 한다.

따라서 본 연구에서는 기존의 프로그램이 가진 Surface Modeling 기능의 단점을 개선하고자 곡면 생성기능과 곡면 순정기능을 중심으로 프로그램을 제작하고자 한다.
첫 번째로 MFC와 OpenGL을 기반으로 프로그램을 제작하여 생성된 곡선과 곡면을 최적화된 알고리즘으로 가시화한다. 두 번째로 NURBS 곡선망과 NURBS 곡면을 이용하여 순정 가능한 NURBS 곡면군을 생성한다. 세 번째로 번거로운 곡면 생성과정을 자동화하여 최소한의 조작으로 선형 전체를 곡면으로 구현하는 방법을 제시한다. 마지막으로 곡면을 구성하는 곡선의 데이터 Point를 조절함으로써 선형 곡면을 순정하고 즉각적인 곡면의 변화를 통해 순정 결과를 확인하는 방법을 제시하고자 한다.

Ⅱ. 본론

1. 선형 곡면화 프로그램 개요

1.1 개요

본 연구에서는 기존의 프로그램의 여러 기능 중에서 곡면화 기능에 해당하는 부분을 독립적인 프로그램으로 제작하였다. 기본적으로 Visual C++언어를 이용하여 작성되었으며, View에서의 각 객체 생성에 관련된 부분은 OpenGL과 Visual C++를 병행 처리하여 편리하고 빠르게 곡면생성 및 곡면 순정작업을 할 수 있도록 제작하였다.

작업자의 인터페이스와 좌표계는 기존 프로그램과 유사하게 구성하여 두 프로그램을 동시에 작업하더라도 혼란이 생기지 않도록 하였다.

1.2 프로그램의 개발 환경

본 프로그램은 MS사의 Windows XP Professional 환경에서 C언어와 MFC, 그리고 OpenGL을 이용하여 제작되었다. Compile Tool로는 Visual C++ 6.0을 사용하였다.

1.2.1 Windows

Windows 운영체제는 여러 가지 다양한 기능들과 함께 표준 보안, 용이성, 신뢰성과 Plug and Play, 사용하기 쉬운 GUI(Graphic User Interface), 하드웨어를 Control 가능하게 하는 수많은 Driver의 집합체라 할 수 있다(Microsoft Windows Homepage).

Windows 운영체제는 GUI를 이용하여 사용자와의 Interactive한 정보교환을 한다. 즉 사용자는 키보드 혹은 마우스를 이용하여 그래픽으로 표시되는 객체를 조작함으로써 필요한 작업 및 정보를 얻을 수 있으며 그 결과 또한 그래픽으로 표시된다. Windows 환경 하에서 작업을 하게 되면 많은 장점을 가진다. 첫 번째

로 대부분의 컴퓨터에서 운영체제로 Windows를 사용하고 있고, 응용 프로그램이 다양하며 기본적인 외양이나 사용방법에서 비슷하므로 사용자의 혼란을 줄일 수 있다. 두 번째로 이전의 DOS(Disk Operating System) 운영체제에서 프로그래머가 프로그램을 작성하는 경우처럼 화면과 프린터 같은 그래픽 디스플레이 장치 등의 하드웨어를 직접 액세스하지 않는다. 따라서 프로그래머는 어떤 종류의 장치가 시스템에 연결되어 있는지 알 필요가 없으며, 하드웨어에 독립적이며 호환과 확장이 가능한 프로그램을 제작할 수 있다. 세 번째로 Multi-Tasking과 Multi-Thread를 지원한다. 즉 사용자의 요구에 따라 여러 개의 Application 프로그램을 동시에 실행하고 다른 프로그램으로의 전환이 자유롭다. 또한 하나의 프로그램에서 또 다른 프로그램으로 데이터를 자유롭게 전환할 수 있다. 마지막으로 효율적인 Memory 관리를 들 수 있다. Windows 운영체제에서 자체적으로 Physical Memory와 가상 Memory 관리를 해줌으로써 사용자는 프로그램이 종료를 위해 Memory관리를 따로 해 줄 필요가 없다.

1.2.2 OpenGL과 MFC

OpenGL(Open Graphic Library) 그래픽스 시스템이란, 미국 SGI(Silicon Graphics)사가 개발한 Workstation급의 고품위 그래픽을 지원하기 위한 그래픽 하드웨어에 대한 소프트웨어 인터페이스를 말한다. OpenGL은 동적인 3차원 오브젝트들을 구성하거나 연산을 수행하기 위한 250여 개의 커맨드로 구성되어 있다. GLU(OpenGL Utility Library)는 이차 곡면이나 NURBS 곡선 및 곡면 등과 같은 모델을 그릴 수 있는 기능을 제공하고 있다(메이슨 우 등, 2003). 그러나 OpenGL은 윈도우 관련작업을 수행하거나 사용자의 입력을 받아들이는데 필요한 기능은 제공하지 않는다.

Microsoft사에서 개발한 MFC(Microsoft Foundation Class)는 Windows의 모든 기능과 특성을 발휘하도록 설계된 고기능 클래스들의 집합이며 C++ 언어로 만들어져 있다. MFC는 Visual C++에 내장되어, 각종 Windows 프로그램을 작성하는데 개발효율을 증대시켜 준다(김상혁, 1998). 하지만 MFC는 그래픽스 전문 클래스 라이브러리가 아니므로 곡면생성에 관한 연산을 수행하고, 또 화면에 표시하기

위해서는 프로그래머가 모든 알고리즘을 구현하고 MFC의 View 클래스와도 연결하여 주어야하는 문제가 있다. 설령 그렇게 하였다고 해도 신뢰도와 연산속도 등에서 최적화하기란 쉽지 않은 문제이다.

따라서 본 연구에서는 Visual C++의 MFC를 이용하여 전체 프로그램의 Frame과 사용자의 입력을 받아들이는 부분, 선형데이터를 읽고 쓰는 부분, NUB-Spline의 생성, 데이터 변환 등을 처리하였고, OpenGL은 NURBS 곡선 및 곡면의 생성과 각종 View에 관한 것을 처리하도록 하였다.

2. 프로그램의 구조

본 연구에서는 프로그램을 제작하기에 앞서 효율적인 프로그램 제작을 위하여 IDEF0를 이용하여 프로그램 분석을 실시하였다. IDEF0로 프로그램에서 구현되어야 할 각 기능을 Activity로 세분화하고 각각의 Activity에서 요구되는 입력 데이터와 출력 데이터 그리고 Activity를 구성하기 위한 Resource 등을 미리 규정함으로써, 후에 프로그램을 작성할 때 여러 가지 기능들이 중복되거나 누락되는 것을 방지할 수 있다.

다음은 본 연구에서 제작한 프로그램을 IDEF0로 분석한 것 중, 식별을 용이하게 하기 위해 세부적인 사항은 제거하고, 기본기능에 대해서만 Diagram으로 작성한 것이다.

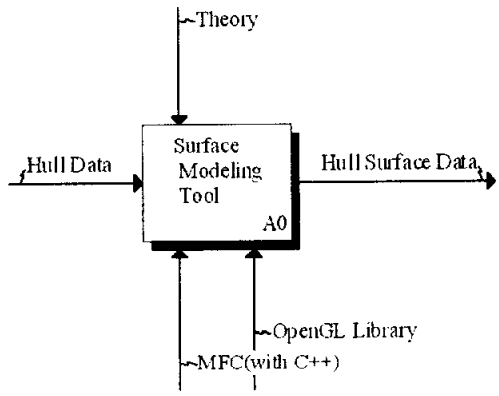


Fig. 2.1. A0, 프로그램 Overview

Fig. 2.1.은 본 연구에서 제작하게 될 프로그램의 Overview로써 입력 데이터로 기존 프로그램에서 제작된 선형 데이터를 이용하여 선형 곡면 데이터를 결과물로 출력 하는 프로그램임을 보여준다. 이때 입력되는 선형 데이터는 Wire Frame 형태의 선형 데이터이며, Theory는 곡선 및 곡면이론이다. 출력되는 선형 곡면 데이터는 곡면 모델이며, 프로그램을 제작 및 구성하기 위한 Library로는 MFC와 OpenGL을 이용하였다.

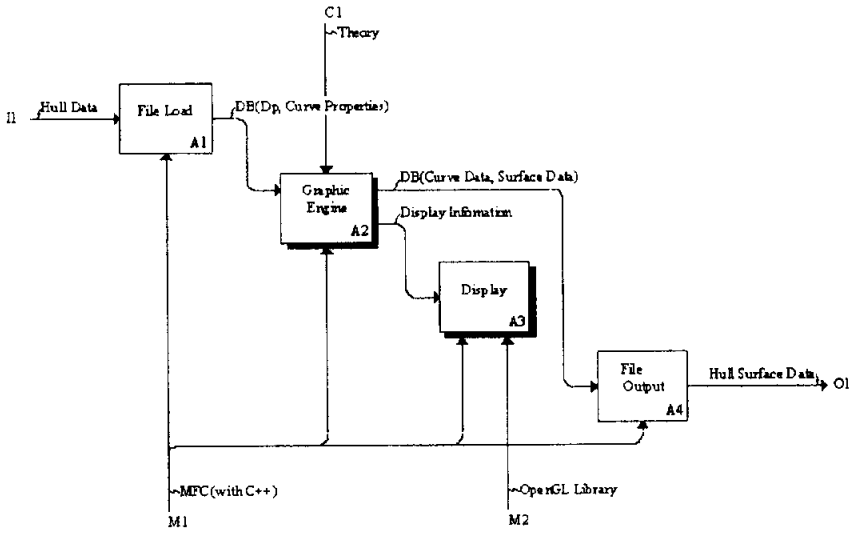


Fig. 2.2. A1, Main Structure of 곡면 모델ing Tool

본 연구에서는 Fig. 2.2와 같이 프로그램 구성을 크게 네 개의 기능별 Activity로 나누었다. 각각의 Activity는 입력 파일을 읽는 A1 Activity(File Load), A1에서 읽어 들인 데이터를 이용하여 곡선과 곡면정보를 생성하는 A2 Activity(Graphic Engine), A2에서 생성된 곡선 및 곡면정보를 화면에 표시하는 A3 Activity (Display), 마지막으로 A3에서 생성 및 가공된 곡선 및 곡면정보를 파일 형태로 출력하는 A4 Activity(File Output)를 의미한다.

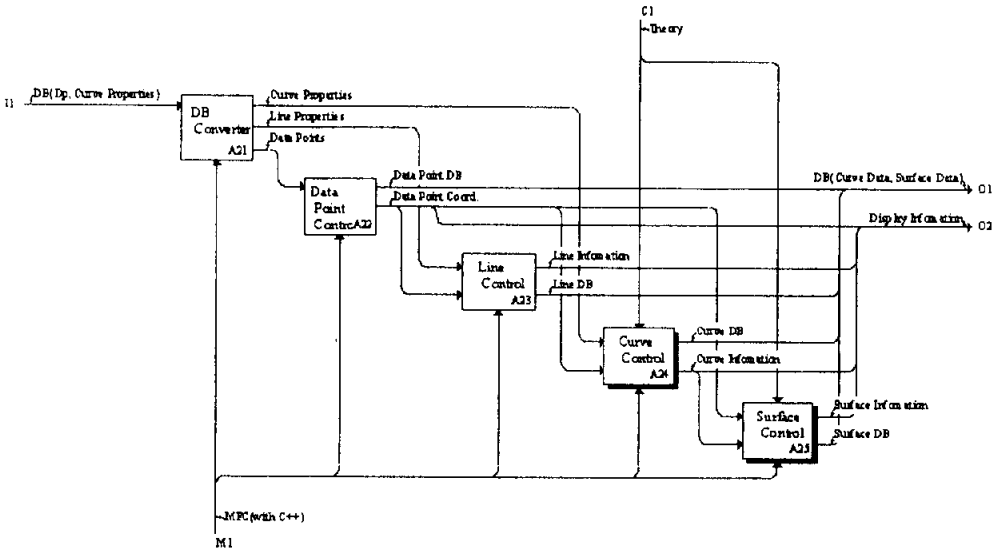


Fig. 2.3. A21, Structure of Graphic Engine

Fig. 2.3에서는 Fig. 2.2의 A2 Activity(Graphic Engine) 하위 Level Diagram이다. 먼저 A1 Activity(File Load)에서 생성된 Point 및 곡선 정보를 A21 Activity(DB Converter)에서 가공에 편리하도록 재분류를 실시한다. 다음 생성된 Point 정보는 A22 Activity(Data Point Control)에서, Line 정보는 A23 Activity(Linet Control), 곡선 정보는 A24 Activity(Curve Control)에서 DB 데이터로 가공한다. 이때 라인이나 곡선들은 데이터 Point Array를 속성으로 가지는 대신 데이터 Point에 부여된 Node ID Array를 속성으로 가진다. 이것의 장점은 후에 2개 이상의 곡선이 교차되는 Point의 좌표를 변경하였을 경우에 여러 곡선을 하나씩 조사하여 변경해주어야 하는 번거로움을 없앨 수 있으며, 정보의 중복을 최소화 할 수 있다. A25 Activity(Surface Control)에서는 A22와 A24에서 생성된 정보를 이용하여 곡면을 생성하고 DB화 한다.

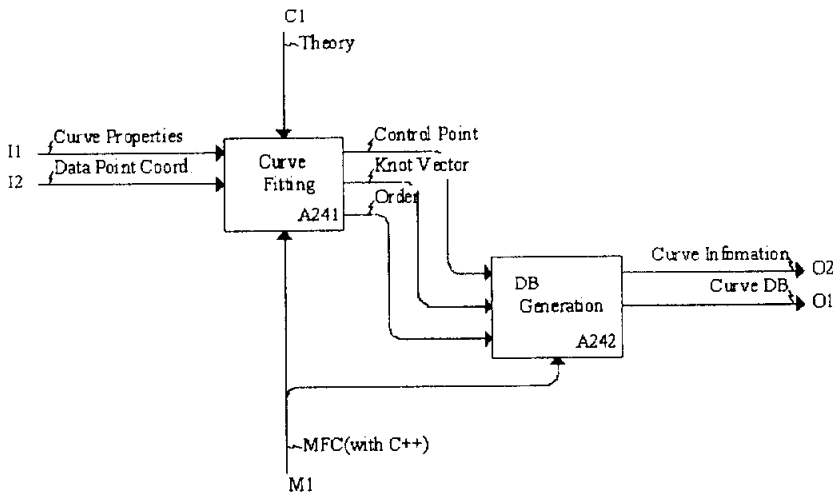


Fig. 2.4. A241, Structure of 곡선 Control

Fig. 2.3에서는 Fig. 2.3의 A24 Activity(Curve Control) 하위 Level Diagram이다. 입력 File에서 가져온 곡선 데이터는 입력 File의 Format에 따라 다르지만 기본적으로 데이터 Point Array로 되어 있다. 이를 NUB 곡선 혹은 NURBS 곡선으로 Fitting하는 과정을 거쳐야 OpenGL Library를 이용하여 화면에 표현할 수 있다. 곡선 Fitting의 이론적인 부분은 다음에 오는 3장에서 설명한 것을 기초로 여러 상황에서 응용 가능한 Function으로 C언어를 이용하여 제작하였다. 곡선 Fitting과 관련된 Function들은 A241 Activity(곡선 Fitting)에 모듈화 하였으며, Fitting된 곡선 정보는 후에 Display 모듈에서 바로 사용가능 하도록 A242 Activity(DB Generator)에서 저장된다.

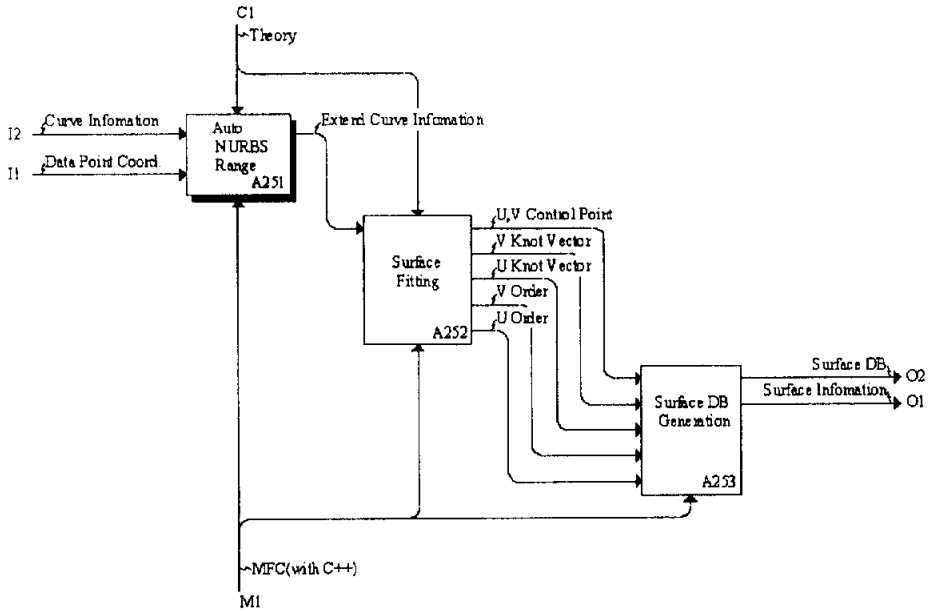


Fig. 2.5. A251, Structure of 곡면 Control

Fig.2.4의 A25 Activity(Surface Control)는 Fig. 2.4의 A242 Activity(DB Generation)에서 가공된 정보를 이용하여 곡면을 생성하는 단계로써, Fig. 2.5에서는 그 단계를 3단계로 세분화하였다. 하나의 NURBS 곡면으로 선형을 모델링하는 것은 매우 어려운 문제이고 아직 뚜렷한 해결방법이 없는 상태이다. 따라서 본 연구에서는 이준호(2002)가 제안한 방법에 따라 선체를 총 7~8개의 영역으로 나누어, 각각의 영역에 맞는 곡면 이론을 적용하는 방법으로 곡면을 생성하였다. 이때 A25 Activity(Surface Control)에서 프로그램 사용자의 편의를 위해 영역을 나누고 곡면을 생성하는 것은 프로그램이 자동으로 선택 및 생성하도록 하였다.

먼저 A251 Activity(Auto NURBS Range)에서는 선체를 7개의 영역으로 분할하여 그 영역에 포함된 곡선을 Cutting하는 작업을 수행한다. 다음 A252 Activity(Surface Fitting)는 A251 Activity에서 영역별로 Cutting된 곡선을 이용하여 곡면 Fitting을 수행하고, 생성된 곡면 정보를 A253 Activity(Surface DB Generator)로 보내, OpenGL Library에서 사용가능한 DB로 가공한다. 이때 사

용자의 선택에 따라 선체 혹은 선체의 일부분 중, 어떤 것을 표시할지 정보를 같이 가져온다.

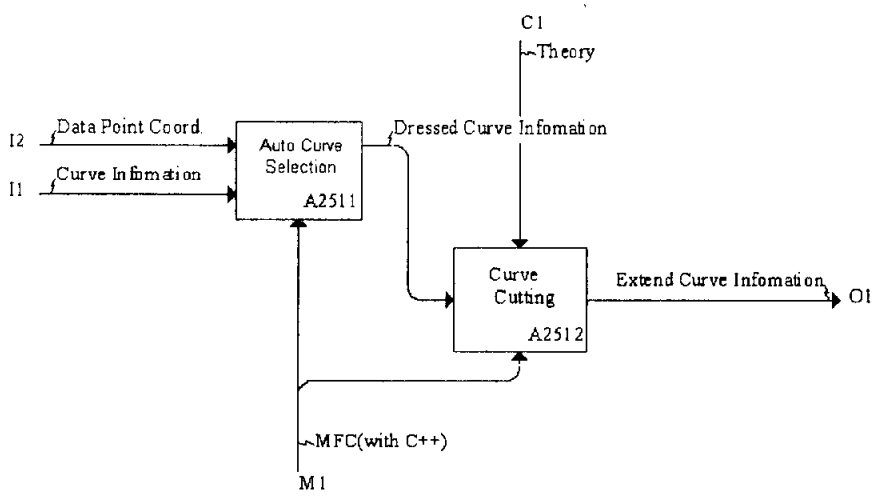


Fig. 2.6. A2511, Structure of Auto NURBS Range

Fig. 2.6은 앞에서 설명한 Fig. 2.5의 A251 Activity(Auto NURBS Range)를 크게 두 가지 기능별로 분류한 것이다. A2511 Activity(Auto Curve Selection)에 대해서는 차후 다시 설명을 할 것이다.

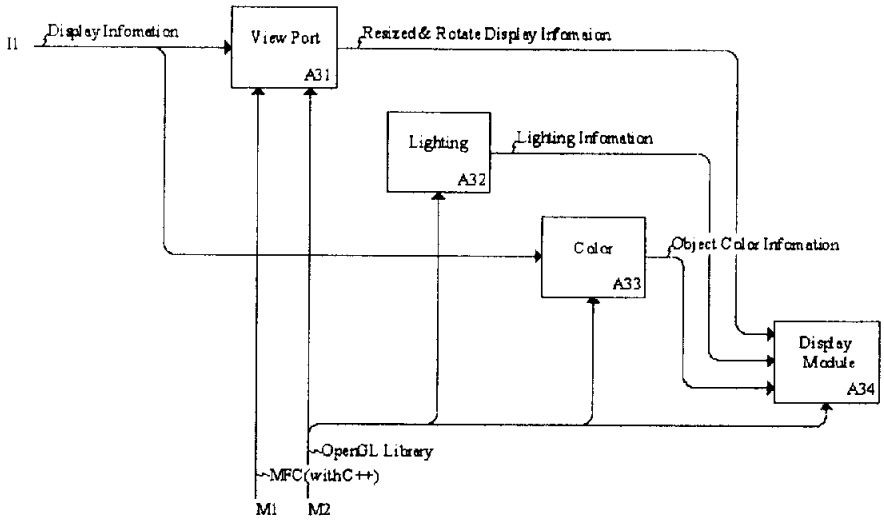


Fig. 2.7. A31, Structure of Display

마지막으로 Fig. 2.7은 앞에서 생성된 곡선 및 곡면 정보를 화면에 표시하기 위한 부분이며, OpenGL Library는 이 모듈에서 사용되었다. 특히 A34 Activity (Display Module)는 DB 정보를 화면에 표시하기 위하여 OpenGL Library를 이용한 Function들을 가지고 있다.

3. 프로그램의 구현

3.1 곡선의 정의와 가시화

3.1.1프로그램에서 사용된 곡선이론

일반적인 CAD 프로그램에서는 NURBS 곡선식을 사용하지만 선형을 이루는 곡선의 경우 가중치를 1로 고정하여 곡선을 표현하여도 충분한 결과를 얻을 수 있으며, 곡선의 조작이 쉬워진다는 장점이 있다. 따라서 본 연구에서는 곡선 표현식으로 NUB(Non-Uniform B-spline) 곡선식을 이용하여 곡선을 표현하였다.

3.1.1.1 NUB(Non-Uniform B-spline) 곡선 및 곡면의 정의

매트릭스형태로 Non-Uniform Cubic B-spline곡선을 표현하면 다음과 같다(윤태경, 1998)(Choi, 1991).

$$r^i(u) = UN_c^i V^i \; ; \quad 0 \le u \le 1 \; ; \quad i = 0, 1, \ldots, n-1$$

여기서,

$$U = [1 \quad u \quad u^2 \quad u^3],$$

$$V^i = [V_i \quad V_{i+1} \quad V_{i+2} \quad V_{i+3}]^T,$$

$$N_c^i = \begin{bmatrix} \frac{(\nabla_i)^2}{\nabla_{i-1}^2 \nabla_{i-2}^3} & (1-n_{11}-n_{13}) & \frac{(\nabla_{i-1})^2}{\nabla_{i-1}^3 \nabla_{i-1}^2} & 0 \\ -3n_{11} & (3n_{11}-n_{23}) & \frac{3\nabla_i \nabla_{i-1}}{\nabla_{i-1}^3 \nabla_{i-1}^2} & 0 \\ 3n_{11} & -(3n_{11}+n_{33}) & \frac{3(\nabla_i)^2}{\nabla_{i-1}^3 \nabla_{i-1}^2} & 0 \\ -n_{11} & (n_{11}-n_{43}-n_{44}) & n_{43} & \frac{(\nabla_i)^2}{\nabla_i^3 \nabla_i^2} \end{bmatrix}$$

$$n_{43} = -\{\frac{1}{3} n_{33} + n_{44} + (\nabla_i)^2 / (\nabla_i^2 \nabla_{i-1}^3)\},$$

$$\nabla_i^k = \nabla_i + \nabla_{i+1} + \ldots + \nabla_{i+k-1}.$$

여기서 V_i는 조절점이고 ∇_i는 노트간격(knot span)이다.

3.1.1.2 NUB 곡선 생성

1.3에서 정의된 NUB 곡선을 생성하기 위한 첫 번째 단계는 노트간격 $\{\nabla_i\}$를 결정하는 것이다. 노트간격 $\{\nabla_i\}$는 정규노트간격 $\{\nabla_0, \cdots, \nabla_{n-1}\}$과 확장(extended) 노트간격으로 나누어진다. 여기서 정규노트간격은 현길이근사(chord-length approximation)를 이용하여 구할 수 있고, 확장노트간격은 다중노트를 정의하여 사용한다. 현길이근사는 주어진 자료점 P_i, P_{i+1}을 이용하여 다음과 같이 나타낸다(윤태경, 1998)(Choi, 1991).

$$\nabla_i = |P_{i+1} - P_i| \text{ for } i = 0, 1, \ldots, n-1$$

확장 노트간격은 다중 노트표현식으로 다음과 같이 나타난다.

$$\nabla_{-2} = \nabla_{-1} = \nabla_0 ,$$
$$\nabla_{n+1} = \nabla_n = \nabla_{n-1}$$

다음 단계는 미지 조절점에 대한 선형연립방정식을 구성하는 것이다.
자료점 P_i , P_{i+1} 사이의 곡선 선분 $r^i(u)$ 는 다음의 관계가 성립한다.

$$r^i(0) = P_i , \; r^i(1) = P_{i+1} , \; (i = 0, \ldots, n-1)$$

위의 관계식을 1.3절의 NUB 곡선식에 대입하면 다음 식이 얻어진다.

$$f_i V_i + h_i V_{i+1} + g_i V_{i+2} = P_i \quad (i = 0, 1, \ldots, n)$$

여기서, $h_i = (1 - f_i - g_i)$
$$, f_i = (\nabla_i)^2 / (\nabla_{i-1}^2 \nabla_{i-2}^3)$$
$$, g_i = (\nabla_{i-1})^2 / (\nabla_{i-1}^2 \nabla_{i-1}^3)$$

곡선의 양 끝점에서 미분으로 접선벡터를 구하면 다음과 같은 선형방정식을 얻을 수 있다.

$$\hat{t}_0 \equiv \dot{r}^0(0)$$
$$= a_0 V_2 + (b_0 - a_0) V_1 - b_0 V_0$$

여기서, $a_0 = 3(\nabla_0 \nabla_{-1}) / (\nabla^2_{-1} \nabla^3_{-1})$,

$, b_0 = 3(\nabla_0)^2 / (\nabla^2_{-1} \nabla^3_{-2})$.

$$\hat{t}_n = \dot{r}^{n-1}(1)$$
$$= a_1 V_{n+2} + (b_1 - a_1) V_{n+2} - b_1 V_n$$

여기서, $a_1 = 3(\nabla_{n-1})^2 / (\nabla^2_{n-1} \nabla^3_{n-1})$

$, b_1 = 3(\nabla_n \nabla_{n-1}) / (\nabla^2_{n-1} \nabla^3_{n-2})$.

확장노트를 적용하여 정리하면 다음 식을 얻을 수 있다.

$$f_0 = 1 \; ; \; g_0 = 0,$$
$$f_n = 0 \; ; \; g_n = 1,$$
$$a_0 = 0 \; ; \; b_0 = 3,$$
$$a_1 = 3 \; ; \; b_1 = 0.$$

따라서 최종적으로 NUB 곡선 생성을 위한 선형연립방정식은 다음과 같이 구성이 되고, 방정식의 해를 구하여 미지 조절점을 쉽게 결정할 수 있다.

$$\begin{bmatrix} -3 & 3 & 0 & 0 & & & & \\ 1 & 0 & 0 & 0 & & & & \\ 0 & f_1 & h_1 & g_1 & & & & \\ & & & & \ddots & & & \\ & & & & f_{n-1} & h_{n-1} & g_{n-1} & 0 \\ & & & & & 0 & 0 & 1 \\ & & & & & 0 & -3 & 3 \end{bmatrix} \begin{bmatrix} V_0 \\ V_1 \\ V_2 \\ \cdot \\ \cdot \\ \cdot \\ V_{n+1} \\ V_{n+2} \end{bmatrix} = \begin{bmatrix} \hat{t}_0 \\ P_0 \\ P_1 \\ \cdot \\ \cdot \\ \cdot \\ P_n \\ \hat{t}_n \end{bmatrix}$$

여기서, $h_i = (1 - f_i - g_i)$,

$f_i = (\nabla_i)^2 / (\nabla_{i-1}^2 \nabla_{i-2}^3)$,

$g_i = (\nabla_{i-1})^2 / (\nabla_{i-1}^2 \nabla_{i-1}^3)$,

$\nabla_i^k = \nabla_i + \ldots + \nabla_{i+k-1}$.

3.1.2 프로그램에서의 곡선의 구현

본 연구에서 입력 데이터는 비교적 순정상태가 좋은 Wire Frame Model 데이터를 사용하였다. 입력 데이터로부터 읽어들인 데이터는 Class화 된 DB에 Point정보, 곡선정보, 그리고 Main Dimension 정보를 Write하여 자체 알고리즘으로 분류작업을 거친다. 즉 모든 Point와 곡선에 ID를 부여하고 중복된 Point나 곡선을 제거하고 Sorting을 실시한다. 곡선의 경우, Point 데이터 Array와 Start Vector, End Vector를 이용하여 NUBS 곡선으로 Fitting을 하고 Order, Node 수, Control Point Array, Knot Vector Array, Type, Color 등의 정보를 DB화 한다.
본 프로그램에서는 일반적인 Lines의 곡선 분류방법에 따라 종류를 총 8가지로 나누어 각각의 곡선에 고유의 Color를 부여하는 방법으로 곡선 속성을 정의하였으며 부가적인 작업을 위한 Temporary 곡선을 추가하였다. 다음은 본 프로그램에 적용된 곡선 속성이다.

① Station Line
② Water Line
③ Buttock Line
④ Center Line
⑤ Tangential Line
⑥ Knuckle Line
⑦ Deck Line
⑧ Space Curve
⑨ Temporary Curve

곡선을 Control하기 위해서는 많은 Function들이 필요로 하게 된다. 대표적으로 곡선 정보를 생성, 수정하거나 제거하는 기능과 곡선 정보를 가져오는 기능, 곡선 간 Intersection을 찾거나 Cutting하는 기능, DB와 정보를 교환하는 기능 등이 필요하며 이들은 각각의 상황에 유연하게 대처할 수 있게 C Function으로 제작하여 Module화 하였다. 위와 같이 조작이 된 곡선들은 OpenGL의

gluNurbsCurve() Function에 의해 화면에 표시된다.

다음의 Fig. 3.1.에서 Fig. 3.4까지의 그림은 본 연구에서 제작된 프로그램으로 곡선을 가시화 한 것들을 보여준다. 여기서 선체 중앙의 생략된 부분은 중앙평행부로써 프로그램 자체적으로 인식이 가능하도록 되어 있다.

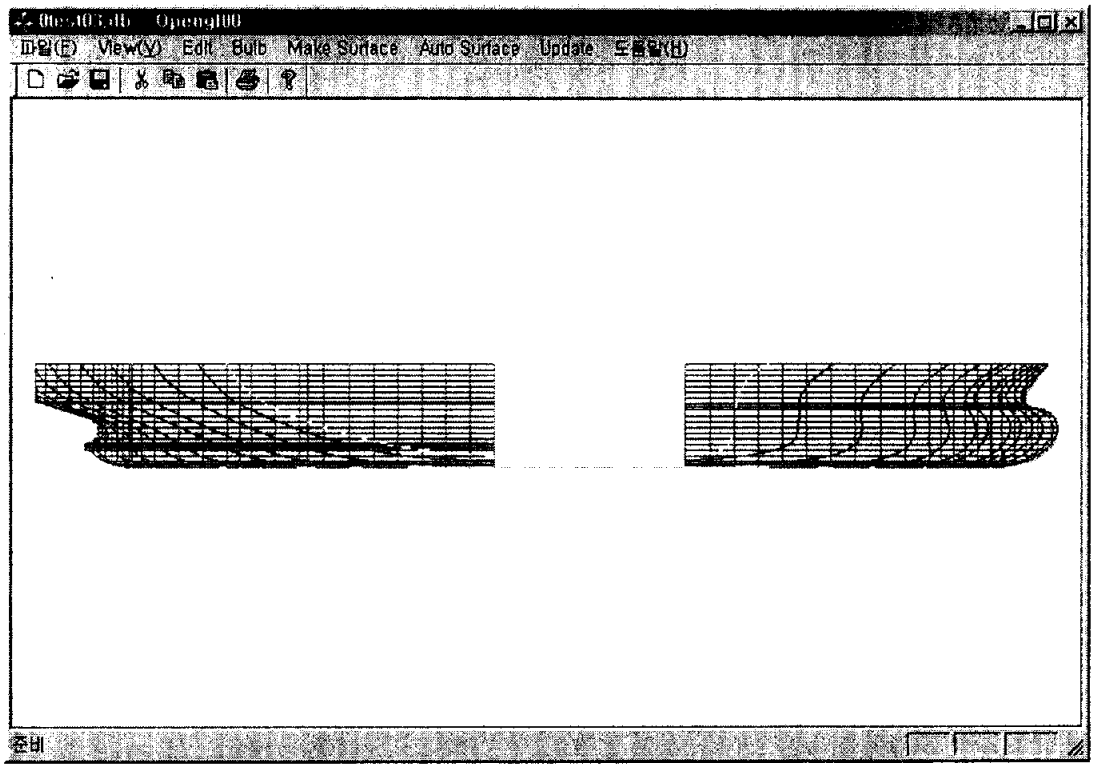


Fig. 3.1 Program Main Frame View

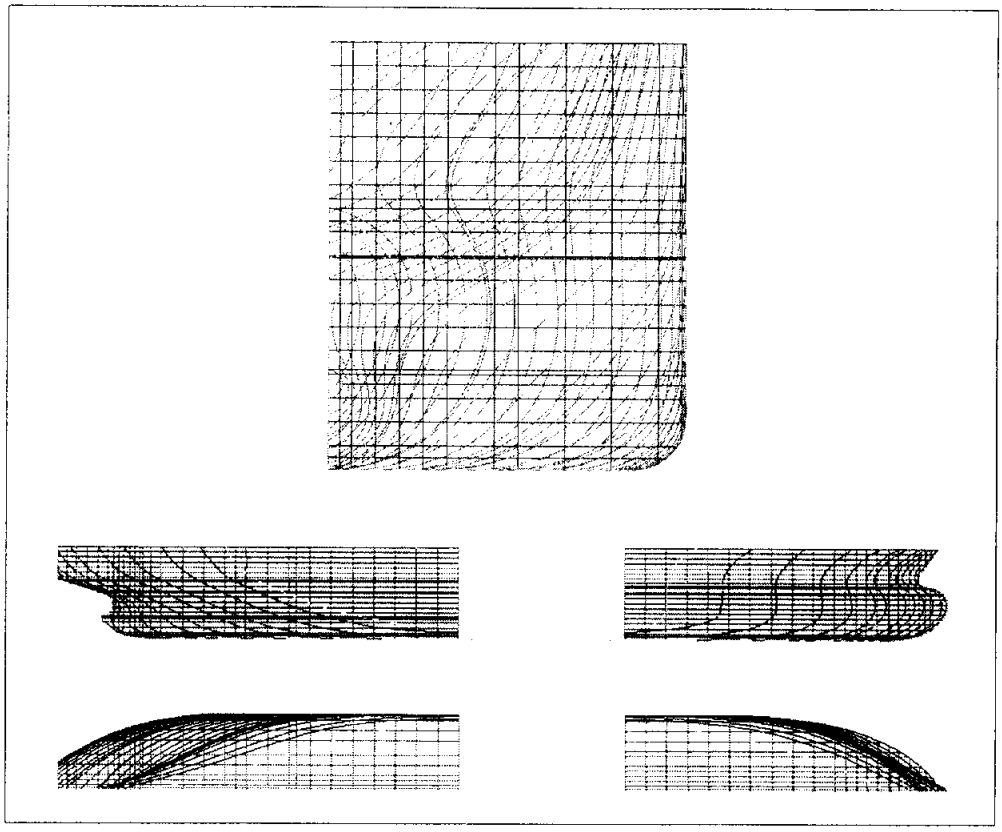


Fig. 3.2 Lines View

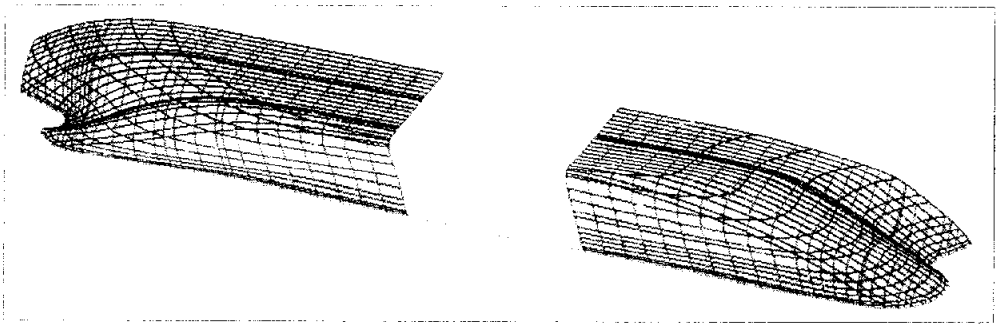


Fig. 3.3 Hull form Iso-metric View

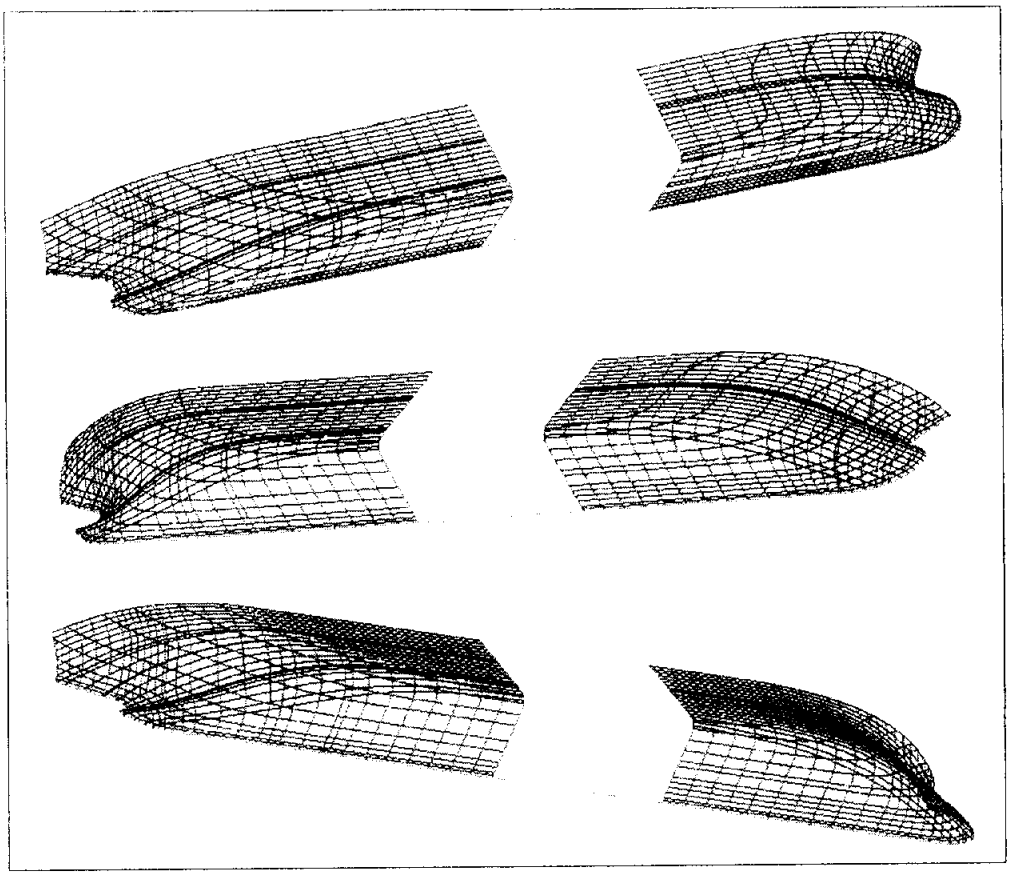


Fig. 3.4 Various Hull form Iso-Metric View

3.2 곡면의 정의와 가시화

3.2.1 프로그램에서 사용된 곡면이론

3.2.1.1 NUB 곡면 생성

본 연구에서는 경계곡선을 이용한 곡면화의 한 가지 방법으로 NUB 곡면을 생성하였다. 이는 사각형 면조각에 대해서만 곡면화가 이루어진다는 단점이 있으나

타 CAD 프로그램과의 정보교환을 위해서는 반드시 NURBS로 곡선과 곡면으로 변환하여한다는 점에서 Non-Uniform Cubic B-spline은 장점을 가진다.
Non-Uniform Cubic B-spline 곡면은 조작 없이 바로 NURBS 곡면으로 변환이 가능하다.

곡면은 $(m+1)\times(n+1)$개의 공유점(mesh points) $\{P_{ij}\}$와 u, v방향의 교차경계접선벡터 $s_{0,j}, \hat{s}_{m,j}$ ($j=0,\ldots,n$), $\hat{t}_{i,0}, \hat{t}_{i,n}$ ($i=0,\ldots,m$), 그리고 모서리공유점(corner mesh points)에서의 비틀림벡터 $\hat{x}_{00}, \hat{x}_{m0}, \hat{x}_{0n}, \hat{x}_{mn}$이 주어지면, 다음의 네 단계를 거쳐 생성된다(윤태경, 1998)(Choi, 1991).

1) 노트간격의 결정(Determine the knot spans)
현길이근사를 이용하여 u, v방향의 정규노트간격과 확장노트간격을 다음과 같이 구한다.

$$\triangle_i = \sum_{j=0}^{n} | P_{i+1,j} - P_{i,j} | \quad \text{for} \quad i=0,1,\ldots,m-1,$$

$$\nabla_j = \sum_{i=0}^{m} | P_{i,j+1} - P_{i,j} | \quad \text{for} \quad j=0,1,\ldots,n-1.$$

$$\triangle_{-2} = \triangle_{-1} = 0 \ , \ \triangle_{m+1} = \triangle_m = \triangle_{m-1}$$

$$\triangle_{-2} = \triangle_{-1} = 0 \ , \ \triangle_{n+1} = \triangle_n = \triangle_{n-1}.$$

2) 중간조절점의 결정(Find intermediate control point)
중간조절점 $\{C_{ij}\}$는 자료점의 각 열(column) j에 대해 NUB곡선을 생성하는 것을 말한다. 즉 자료점 $P_{i(j)} : i=0,1,\ldots,m$ 와 경계접선벡터 $\hat{s}_{0(j)}, \hat{s}_{m(j)}$로 구성되는 선형연립방정식에 의해 구해진다.

$$\begin{bmatrix} -3 & 3 & 0 & 0 & & & \\ f_0 & (1-f_0-g_0) & g_0 & 0 & & & \\ & & \cdot & & & & \\ & & & \cdot & & & \\ & & & & f_m & (1-f_m-g_m) & g_m \\ & & & & 0 & -3 & 3 \end{bmatrix} \begin{bmatrix} C_{0,j} \\ C_{1,j} \\ \cdot \\ \cdot \\ C_{m+1,j} \\ C_{m+2,j} \end{bmatrix} = \begin{bmatrix} \hat{s}_{0,j} \\ P_{0,j} \\ \cdot \\ \cdot \\ P_{m,j} \\ \hat{s}_{m,j} \end{bmatrix}$$

여기서, $f_i = (\triangle_i)^2/(\triangle_{i-1}^2 \triangle_{i-2}^3)$

$, g_i = (\triangle_{i-1})^2/(\triangle_{i-1}^2 \triangle_{i-1}^3)$ for $i=0,\ldots,m.$

3) 경계벡터의 결정(Determine boundary vectors)

경계접선벡터 $\{\hat{t}_{i,0}, \hat{t}_{i,n} : i=0,\ldots,m\}$와 모서리 비틀림벡터를 이용하여 경계벡터 d_i, e_i $(i=0,\ldots,m+2)$를 구한다.

$$d_i = 3(V_{i,1} - V_{i,0}),$$
$$e_i = 3(V_{i,n+2} - V_{i,n+1}).$$

V 방향의 교차경계접선벡터는 V방향 도함수에 의하여 다음과 같이 얻어진다.

$$t_{i,0} = \frac{\partial}{\partial v} r^{i,0}(0,0) \ , \quad t_{i,n} = \frac{\partial}{\partial v} r^{i,n-1}(0,1).$$

따라서, 다음과 같은 선형연립방정식이 구성된다.

$$f_i\, d_i + (1-f_i-g_i)d_{i+1} + g_i\, d_{i+2} = \hat{t}_{i,0} \ ; \quad \text{for } i=0,\ldots,m$$
$$f_i\, e_i + (1-f_i-g_i)e_{i+1} + g_i\, e_{i+2} = \hat{t}_{i,n} \ ; \quad \text{for } i=0,\ldots,m.$$

모서리에서 교차 도함수를 계산하여 다음의 관계식을 얻는다.

$$3(d_1 - d_0) = \hat{x}_{00} \; ; \quad 3(d_{m+2} - d_{m+1}) = \hat{x}_{m0}$$
$$3(e_1 - e_0) = \hat{x}_{0n} \; ; \quad 3(e_{m+2} - e_{m+1}) = \hat{x}_{mn}.$$

위의 선형연립방정식을 통해 경계벡터 d_i, e_i $(i=0,\ldots,m+2)$를 결정한다.

4) 조절점의 결정(Determine B-spine control vertices)

중간조절점의 각 행(row) i에 대한 조절점 $\{V_{ij}\}$을 구한다. 조절점 $\{V_{ij}\}$은 중간조절점과 경계벡터를 이용하여 최종적으로 다음과 같은 선형연립방정식을 각 행에 계산하여 얻을 수 있다.

$$\begin{bmatrix} -3 & 3 & 0 & 0 & & & \\ f_0 & (1-f_0-g_0) & g_0 & 0 & & & \\ & & \cdot & & & & \\ & & & \cdot & & & \\ & & & & f_n & (1-f_n-g_n) & g_n \\ & & & & 0 & -3 & 3 \end{bmatrix} \begin{bmatrix} V_{i,0} \\ V_{i,1} \\ \cdot \\ \cdot \\ V_{i,n+1} \\ V_{i,n+2} \end{bmatrix} = \begin{bmatrix} d_i \\ C_{i,0} \\ \cdot \\ \cdot \\ C_{i,n} \\ e_i \end{bmatrix}$$

2.3 Hermite 혼합(blending) Coons 면조각의 정의

사각형 곡면 조각 $r(u,v)$는 네 개의 경계곡선에 의해 경계된다(Choi, 1991).

경계조건

$$r(i,v) = a_i(v) \quad : \quad i = 0, 1$$
$$r(u,j) = b_j(u) \quad : \quad j = 0, 1$$

면조각의 모서리점(Corner Point)은

$$P_{ij} = r(i,j) \quad : \quad i,j = 0,1$$

$r_1(u,v)$과 $r_2(u,v)$는 경계조건을 만족하는 Ruled 곡면이라 하면

$$r_1(u,v) = (1-u)\, a_0(v) + u\, a_1(v)$$
$$r_2(u,v) = (1-u)\, b_0(v) + v\, b_1(u)$$

곡면조각 $r(u,v)$ 는

$$r(u,v) = r_1(u,v) + r_2(u,v) - r_3(u,v)$$

여기에서 $r_3(u,v)$를 수정곡면(Correction Surface) 라 한다. 수정곡면 $r_3(u,v)$를 계산하기 위해서 u=0에서의 경계조건으로부터

$$\begin{aligned} a_0(v) &\equiv r(0,v) \\ &= r_1(0,v) + r_2(0,v) - r_3(0,v) \\ &= \{a_0(v)\} + \{(1-v)\, b_0(0) + v\, b_1(0)\} - r_3(0,v) \end{aligned}$$

u=1에서 경계조건은

$$\begin{aligned} a_0(v) &\equiv r(1,v) \\ &= r_1(1,v) + r_2(1,v) - r_3(1,v) \\ &= \{a_1(v)\} + \{(1-v)\, b_0(1) + v\, b_1(1)\} - r_3(1,v) \end{aligned}$$

u= 0, 1 에서의 수정곡면의 경계값은

$$r_3(0,v) = (1-v)\, b_0(0) + v\, b_1(0)$$
$$= (1-v)\, P_{00} + v\, P_{01}$$

$$r_3(1,v) = (1-v)\, b_0(1) + v\, b_1(1)$$
$$= (1-v)\, P_{10} + v\, P_{11}$$

수정곡면 $r_3(u,v)$ 는 두 경계곡선의 선형 혼합으로 정의되므로

$$r_3(u,v) = (1-u)\, r_3(0,v) + u\, r_3(1,u)$$
$$= (1-u)\,(1-v)\, P_{00} + (1-u)\, v\, P_{01} + u\,(1-v)\, P_{10} + u\, v\, P_{11}$$

그러므로 수정곡면 $r_3(u,v)$는 네 모서리점에서의 선형혼합으로써 정의된다.

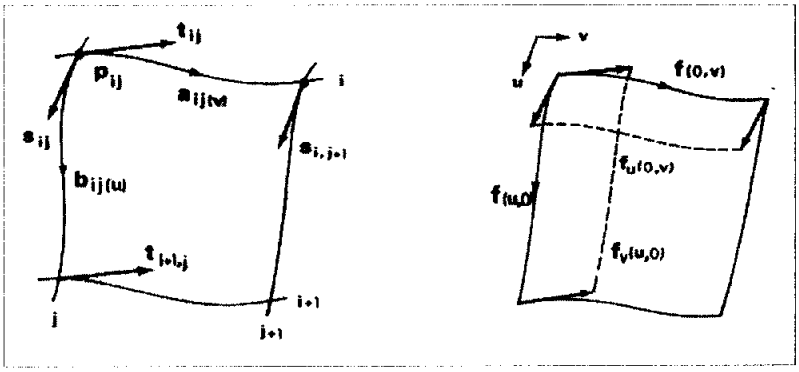


Fig 3.5 Estimation of Cross-Boundary Tangent function

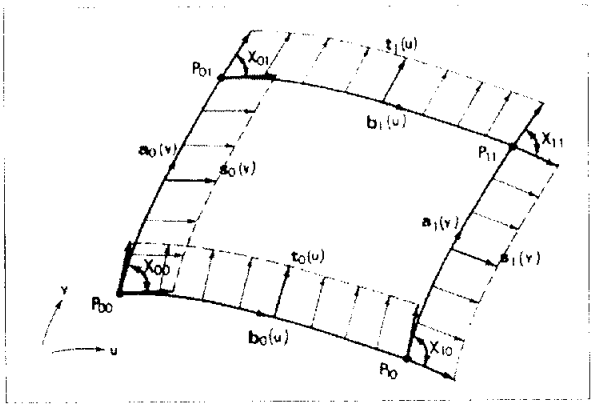


Fig 3.6 Hermite blended Coons Patch

Fig 3.5에서 $s_i(v)$와 $t_i(u)$가 각각 경계곡선 $a_i(v)$와 $b_i(u)$을 따라가는 교차경계접선을 나타낸다고 하자. 마주보는 경계곡선 $a_0(v)$, $a_1(v)$ 와 교차경계접선 $s_0(v)$, $s_1(v)$를 가지고 Fig. 3.6의 Hermite 혼합 Lofted 곡면이 표현된다.

$$r_1(u,v) = \alpha_0(u)\, a_0(v) + \alpha_1(u)\, a_1(v) + \beta_0(u)\, s_0(v) + {}_1(u)\, s_1(v)$$

Where,

$$\alpha_0(u) = H_0^3(u) = (1 - 3u^2 + 2u^3)$$

$$\beta_0(u) = H_1^3(u) = (u - 2u^2 + u^3)$$

$$\beta_1(u) = H_2^3(u) = (-u^2 + u^3)$$

$$\beta_0(u) = H_3^3(u) = (3u^2 + 2u^3)$$

유사한 방법으로 교차경계접선 $t_0(u)$, $t_1(u)$와 마주보는 경계곡선 $b_0(u)$, $b_1(u)$로부터 다른 하나의 Lofted 곡면이 얻어진다.

$$r_2(u,v) = \alpha_0(v)\, b_0(u) + \alpha_1(v)\, b_1(u) + \beta_0(v)\, t_0(u) + \beta_1(v)\, t_1(u)$$

$r_3(u,v)$는 수정곡면이다. x_{ij}는 다음과 같은 면조각 모서리(Patch Corner)에서의 비틀림벡터로 표시된다고 하자. (i, j= 0, 1에 대하여)

$$x_{ij} = \partial^2 r(u,v) / \partial u \partial v \mid_{u=i,\ v=j}$$

$$= \partial s_i(v)/\partial v \mid_{v=j}$$

$$x_{ij} = \partial^2 r(u,v) / \partial v \partial u \mid_{u=i,\ v=j}$$

$$= \partial t_i(u)/\partial u \mid_{v=j}$$

$$r_3(u,v) = [\alpha_0(u)\ \alpha_1(u)\ \beta_0(u)\ \beta_0(u)] \begin{bmatrix} P_{00} & P_{01} & t_0(0) & t_1(0) \\ P_{10} & P_{11} & t_0(1) & t_1(1) \\ s_0(0) & s_0(1) & x_{00} & x_{01} \\ s_1(0) & s_1(1) & x_{10} & x_{11} \end{bmatrix} \begin{bmatrix} \alpha_0(v) \\ \alpha_1(v) \\ \beta_0(v) \\ \beta_1(v) \end{bmatrix}$$

위을 식을 가지고 다음과 같이 선형 혼합방정식으로 표현함으로써 Cubic Hermite 혼합 Coons 면조각이 얻어진다.

$$r(u,v) = r_1(u,v) + r_2(u,v) - r_3(u,v)$$

$$= [\,\alpha_0(u)\ \ \alpha_1(u)\ \ \beta_0(u)\ \ \beta_0(u)] \begin{bmatrix} a_0(v) \\ a_1(v) \\ s_0(v) \\ s_1(v) \end{bmatrix}$$

$$+ [\,b_0(u)\ \ b_1(u)\ \ t_0(u)\ \ t_1(u)] \begin{bmatrix} \alpha_0(u) \\ \alpha_1(u) \\ \beta_0(u) \\ \beta_0(u) \end{bmatrix}$$

$$- [\,\alpha_0(u)\ \ \alpha_1(u)\ \ \beta_0(u)\ \ \beta_0(u)] \begin{bmatrix} P_{00} & P_{01} & t_0(0) & t_1(0) \\ P_{10} & P_{11} & t_0(1) & t_1(1) \\ s_0(0) & s_0(0) & x_{00} & x_{01} \\ s_1(0) & s_1(1) & x_{10} & x_{11} \end{bmatrix} \begin{bmatrix} a_0(v) \\ \alpha_1(v) \\ \beta_0(v) \\ \beta_1(v) \end{bmatrix}$$

Where,

$$\alpha_0(u) = (1 - 3u^2 + 2u^3)$$

$$\beta_0(u) = (u - 2u^2 + u^3)$$

$$\beta_1(u) = (-u^2 + u^3)$$

$$\beta_0(u) = (3u^2 + 2u^3)$$

$a_i(v)$, $b_j(u)$: *boundary curve*

P_{ij}, x_{ij} : *position* and *twist vectors at patch corners*

Fig 3.5에서 s_{ij}와 t_{ij}가 각각 공유점 P_{ij}에서 u 방향과 v 방향 접선을 나타낸다고 하자. 공유점을 추정하는 방법은 P_{ij}에서 공유곡선의 끝접선의 평균을 구하는 것이다. 즉

$$s_{ij} = \{ \ b'_{ij}(0) + b'_{i-1,j}(1) \ \} / 2$$

$$t_{ij} = \{ \ a'_{ij}(0) + a'_{i,j-1}(1) \ \} / 2$$

$$Where \ \ a'_{ij}(0) \equiv \partial a_{ij}(v) / \partial v \mid_{v=0}$$

$$b'_{ij}(0) \equiv \partial b_{ij}(v) / \partial u \mid_{u=0}$$

두 개의 공유점 접선의 Hermite 혼합함으로써 교차경계접선을 결정할 수 있다.

$$s_{ij}(v) = \alpha(v)\, s_{ij} + \beta(v)\, s_{i,j+1}$$

$$T_{ij}(u) = \alpha(u)\, t_{ij} + \beta(u)\, t_{i+1,j}$$

여기에서 $\alpha(t)$, $\beta(t)$는 Hermite 혼합함수 이다.

면조각의 네 점을 따라서 교차경계접선을 나타내면

$$f_u(0,v) \equiv s_{ij}(v)$$

$$f_u(1,v) \equiv s_{i+1,j}(v)$$

$$f_u(u,0) \equiv t_{ij}(u)$$

$$f_u(u,1) \equiv t_{i,j+1}(u)$$

Hermite 혼합함수 정의로부터 끝점에서의 두 번 미분값은 zero를 가지므로 공유 교차점에서의 비틀림벡터는 0이 됨을 알 수 있다.

$$f_{uv}(0,0) = \partial f_u(0,v)/\partial v \mid_{v=0}$$

$$= \alpha'(0)s_{ij} + \beta'(0)s_{i,j+1} = 0, \quad etc.$$

그러므로 Fig 3.5에서 사각형 영역에 대한 Hermite 혼합 Coons 면조각은 다음과 같이 정의된다.

$$r(u,v) = r_1(u,v) + r_2(u,v) - r_3(u,v)$$

$$r_1(u,v) = [\,\alpha(u)\ \beta(u)\ \gamma(u)\ \delta(u)\,]\begin{bmatrix} f(0,v) \\ f(1,v) \\ f_u(0,v) \\ f_u(1,v) \end{bmatrix}$$

$$r_2(u,v) = [\,f(u,0)\ \ f(u,1)\ \ f_v(u,0)\ \ f_v(u,1)\,]\begin{bmatrix} \alpha(v) \\ \beta(v) \\ \gamma(v) \\ \delta(v) \end{bmatrix}$$

$$r_3(u,v) = [\,\alpha(u)\ \beta(u)\ \gamma(u)\ \delta(u)\,]\begin{bmatrix} f(0,0) & f(0,1) & f_v(0,0) & f_v(0,1) \\ f(1,0) & f(1,1) & f_v(1,0) & f_v(1,1) \\ f_u(0,0) & f_u(0,1) & 0 & 0 \\ f_u(1,0) & f_u(1,1) & 0 & 0 \end{bmatrix}\begin{bmatrix} \alpha(v) \\ \beta(v) \\ \gamma(v) \\ \delta(v) \end{bmatrix}$$

Where, $\alpha(u) = (1 - 3u^2 + 2u^3)$

$\beta(u) = (u - 2u^2 + u^3)$

$\gamma(u) = (-u^2 + u^3)$

$\delta(u) = (3u^2 + 2u^3)$

$f(i,v),\ f(u,j)$ for $i, j = 0, 1$: *boundary curves*

$f_u(i,v),\ f_v(u,j)$ for $i, j = 0, 1$: *cross − boundary tangent*

3.2.2 프로그램에서의 곡면의 구현

3.2.2.1 NURBS 곡면의 생성

본 연구에서는 선형을 정의하는 데이터 Point와 NUB 곡선을 이용하여 NUB 곡면을 생성하였다. NUB 곡면을 생성하기 위해서는 NUB 곡면 Fitting 과정이 필요하다. 곡면 Fitting은 주어진 데이터 Point에서 Control Point를 구하는 작업이다.

$(m+1)\times(n+1)$개의 공유점$\{P_{ij}\}$와 u, v방향의 교차경계접선 Vector $S_{0,j}, \hat{S}_{m,j}\ (j=0,...,n),\ \hat{t}_{i,0}, \hat{t}_{i,n}\ (i=0,...,m)$, 그리고 모서리 공유점(Corner mesh point)에서의 비틀림 Vector $\hat{x}_{00}$, $\hat{x}_{m0}$, $\hat{x}_{0n}$, $\hat{x}_{mn}$이 주어지면 □□장에서 설명한 방법을 거쳐 NUB 곡면이 생성된다. 즉 곡면 상에서 시작 Vector(Start Vector)아 끝 Vector(End Vector)가 존재하기 위해서는 u방향과 v방향 각각의 경계곡선상의 데이터 Point의 개수가 같아야 한다(윤태경, 1998)(Choi, 1991)(이준호, 2002).

3.2.2.2 OpenGL Library를 이용한 곡면의 가시화

그래픽 하드웨어는 낮은 레벨의 관점에서 볼 때 점과 선, 그리고 삼각형이나 사각형과 같은 폴리곤만 그릴 수 있다. 부드러운 곡선이나 곡면은 수많은 선이나 폴리곤으로 잘게 나누어 표현하게 된다. OpenGL에서는 평가자(Evaluator)를 사용하여 모든 차수의 다항 Spline 곡선과 곡면을 표현할 수 있다. NURBS 렌더링에 필요한 함수는 OpenGL Library인 GLU에서 제공이 되고 있다. 그러나 실제로 NURBS를 이용한 렌더링은 이 평가자에 의해서 이루어지게 된다(메이슨 우 등, 2003).

본 연구에서 제작된 프로그램에서는 NURBS Object를 화면에 렌더링하기 위해

GLU의 함수를 사용하였다. 여기에서는 OpenGL의 곡면 생성과 관련된 부분만 소개한다.

먼저 NURBS Object를 생성하기 위해 gluNewNurbsRenderer()를 호출한다. 이 함수는 GLUnurbsObj 구조체에 대한 포인터로 된 새로운 NURBS Object를 반환하며 다른 NURBS함수가 사용되기 전 반드시 만들어야한다.

```
GLUnurbs* APIENTRY gluNewNurbsRenderer (void);
```

NURBS Object가 렌더링되는 방법을 정의나 변경을 하고자 할 때에는 NURBS Object의 속성(Property)을 조절한다. 이 속성의 인자에 따라 NURBS Object를 폴리곤으로 렌더링 할 것인지, 아니면 Wire Frame으로 그려질 것인지 결정하게 된다.

```
void APIENTRY gluNurbsProperty (
        GLUnurbs        *nobj,
        GLenum          property,
        GLfloat         value );
```

본 연구에서는 곡면을 폴리곤으로 처리하여 보여주기 위하여 속성을 GLU_DISPLAY_MODE로 지정하였다. GLU_DISPLAY_MODE의 기본값은 GLU_FILL로서 NURBS Object를 폴리곤으로 렌더링 하게 된다.

위와 같이 생성하게 될 NURBS Object 곡면에 대한 선행 준비가 끝나면 gluNurbsSurface() 함수를 실행하여 곡면을 렌더링 한다. gluNurbsSurface() 함수는 gluBeginSurface()와 gluEndSurface() 사이에 위치해야 하며 이 두 함수는 평가자의 상태를 저장하고 복원하기 위해 사용된다.

```
void APIENTRY gluNurbsSurface(
```

```
GLUnurbs        *nobj,
GLint           uknot_count,
float           *uknot,
GLint           vknot_count,
GLfloat         *vknot,
GLint           ustride,
GLint           v_stride,
GLfloat         *ctlarray,
GLint           uorder,
GLint           vorder,
GLenum          type);
```

이 함수는 NURBS 곡면 nobj의 정점을 정의한다. uknot_count와 vknot_count는 각각 u, v방향의 Knot Vector 개수, uknot와 vknot는 각각 u, v방향의 Knot Vector Sequence Array를 의미하며 uorder와 vorder는 각각 u, v방향의 다항식 차수를 설정하는데 사용된다. 여기에서는 Control Point의 개수가 설정되지 않는 대신 각각의 매개변수의 Knot Vector 개수에서 차수를 뺀 값이 Control Point의 개수로 결정된다. ctlarray는 Control Point를 가리키는 포인터이다. 마지막으로 type은 2차원 평가자 타입중 하나이다.

다음은 본 프로그램에서 실제로 사용한 예이다. gluNurbsSurface()의 입력 데이터는 위의 절에서 Fitting한 NUB 곡면 정보에 가중치를 1로 고정한 NURBS DB 데이터이다. 이때 uorder와 vorder는 각각 4로 주었다.

```
GLUnurbsObj *pNurb; // Nurbs Object
pNurb=gluNewNurbsRenderer();
gluNurbsProperty(pNurb, GLU_DISPLAY_MODE, GLU_FILL);
gluBeginSurface(pNurb);
        gluNurbsSurface(pNurb,nu+uorder,xu, nv+vorder,xv,
```

```
        3,3*(nu),ctl[0],uorder,vorder,GL_MAP2_VERTEX_3);
    gluEndSurface(pNurb);
```

3.2.2.3 선형 곡면의 가시화

Fig.3.6은 OpenGL을 이용하여 선형의 선수미부 데이터 Point를 가시화 한 것이다. 데이터 Point는 곡선과 곡면을 생성하는데 가장 기본이 되는 데이터이다.

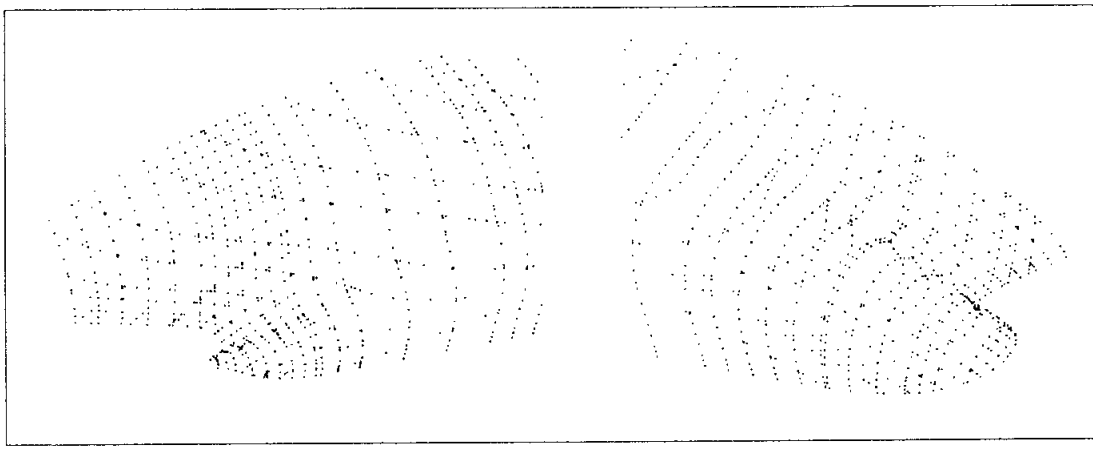


Fig. 3.6 Hull Aft and Fwd Part Data Point View

이 데이터 Point를 Sequence Array로 만들어 곡선 Fitting을 수행하였다. Fig. 3.7은 데이터 Point를 이용하여 선형의 선수미부를 곡선으로 처리하고 가시화 한 것이다.

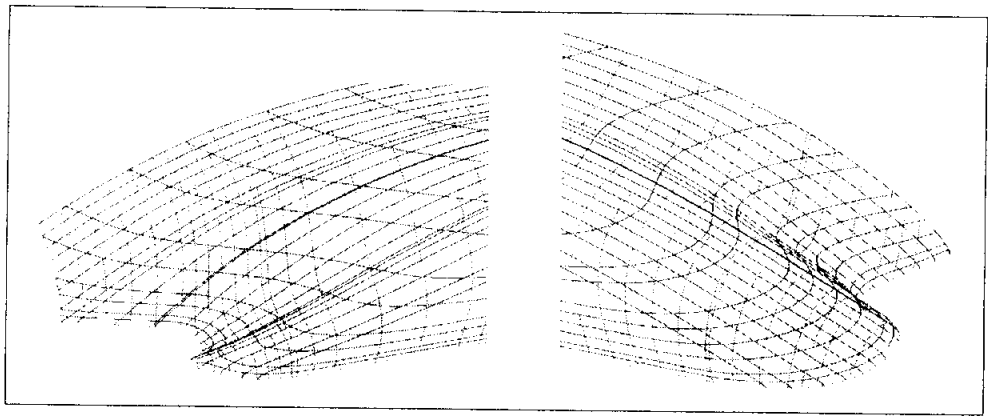


Fig. 3.7 Hull Aft and Fwd Part Curve View

Fig. 3.7에서 보는 것과 같이 선형은 매우 복잡한 형상으로 구성되어 있어 하나의 곡면 Patch로 표현하기에는 많은 어려움이 따른다. 따라서 본 연구에서는 이준호(2002)가 제안한 방법과 같이 일반적인 상선의 경우에 맞춰 7개의 구역으로 나누어 곡면화를 실시하였다.

Fig. 3.8 Hull Division for Surface Modeling

그러나 Fig. 3.8과 같이 여러개의 Patch로 구성된 곡면을 하나씩 생성하기 위해서는 조작이 어렵고 작업량도 증가하게 되어 합리적이지 못하다. 본 프로그램에서는 위의 작업을 프로그램 자체 알고리즘으로 분할하고 곡면생성을 자동화 하였다.

먼저 분할영역을 결정하기 위해서는 Fig. 3.9와 같이 곡선 정보를 이용하여 1번에서 6번까지의 Point를 찾아야 한다.

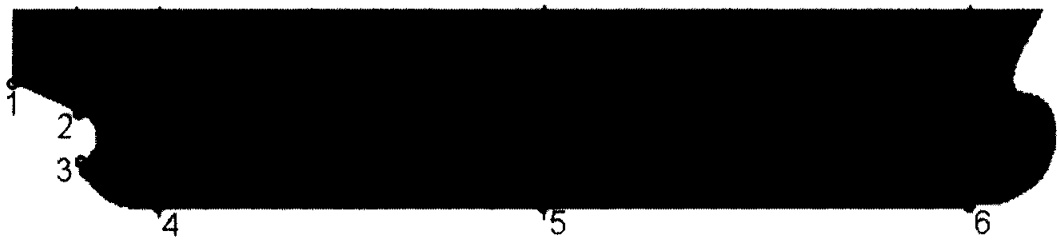


Fig. 3.9 Finding point for Hull Division

1번 Point의 경우 선미부 Center Line으로부터 Point Array정보와 Transom을 구성하는 Station Line과의 교차점을 이용하여 찾아낼 수 있다. 또한 2번 Point의 경우는 선미부 Center Line으로부터 가져온 데이터 Point 중 X값의 변화를 이용하여 찾을 수 있다. 이와 같이 곡선 데이터 혹은 곡선 간의 상호 연결 조건을 이용하여 6번까지의 Point를 프로그램을 이용하여 찾아 분할영역을 결정하였다. 이렇게 분할된 영역은 각각 다른 특성을 지니고 있는데 Fig. 3.9의 0, 1, 3, 4, 5번 영역의 경우 Station을 일정간격(1m)으로 분할 생성하여 내부 자료점(50개)을 생성, Mesh를 구성하여 NUB곡면을 생성할 수 있는 반면 2번 영역의 경우 Water Line(0.2m)을 이용하여 자료점(50개)을 생성하는 것이 곡면의 품질 향상에 도움이 됨을 알 수 있었다. 그러나 선수 Bulb의 경우 Station Line이나 Water Line을 이용하여 곡면을 생성하였을 경우 형상을 제대로 표현하지 못하는 문제가 발생하여 Hermite Coons Blending 방법을 사용하여 근사적으로 NUB 곡면을 생성하였다(이준호, 2002).

다음의 Fig. 3.10에서 Fig. 3.17까지의 그림은 본 연구에서 제작된 프로그램을 통해 0번 곡면부터 6번 곡면까지 단계별로 곡면을 생성하는 모습이다. 자동 곡면생성을 선택하였을 경우 이와 같은 과정이 사용자의 조작 없이 이루어지며 필요한 곡면만 생성 및 수정을 할 수도 있다.

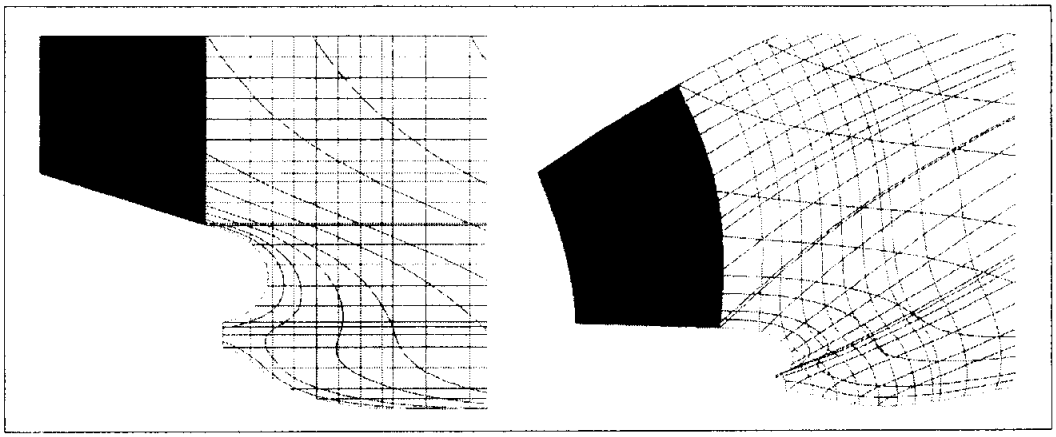


Fig. 3.10 Create No. 0 Surface Patch

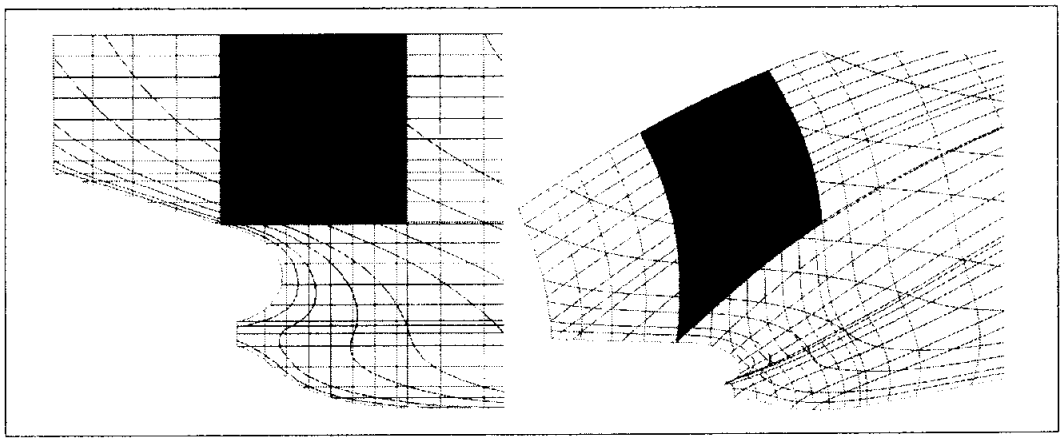


Fig. 3.11 Create No. 1 Surface Patch

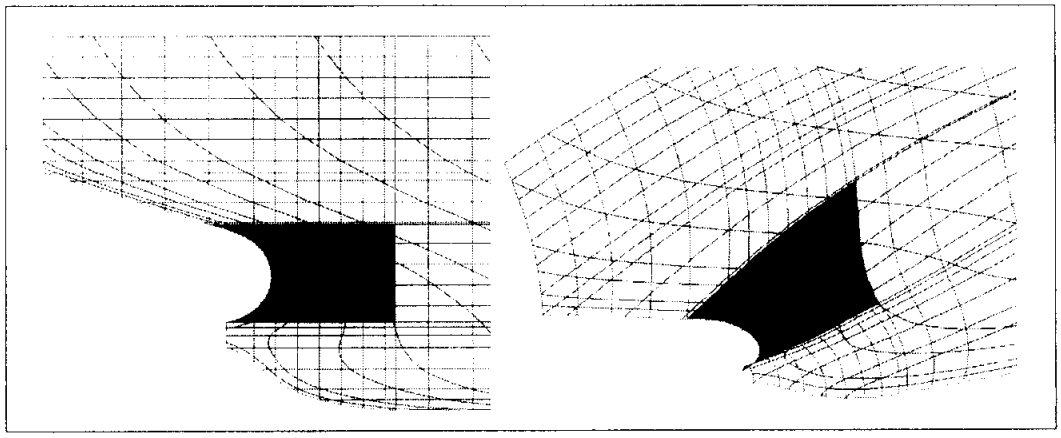


Fig. 3.12 Create No. 2 Surface Patch

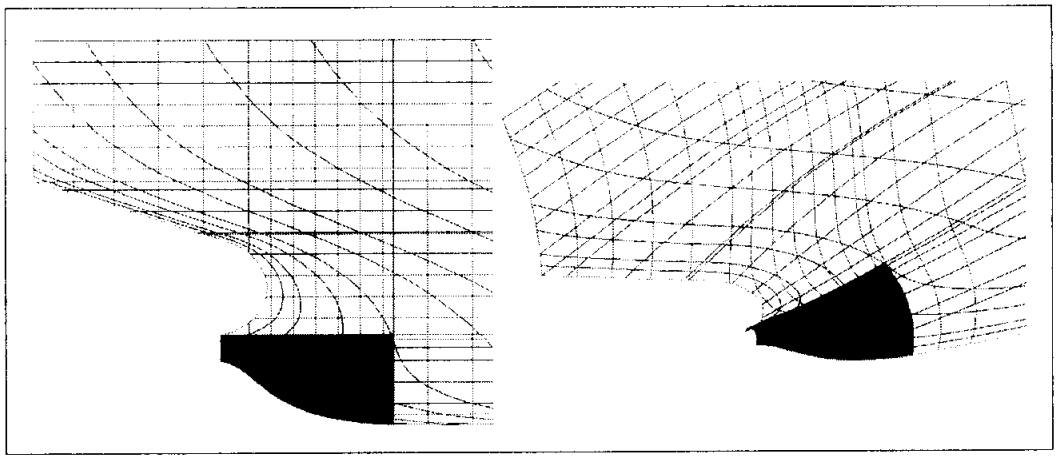


Fig. 3.13 Create No. 3 Surface Patch

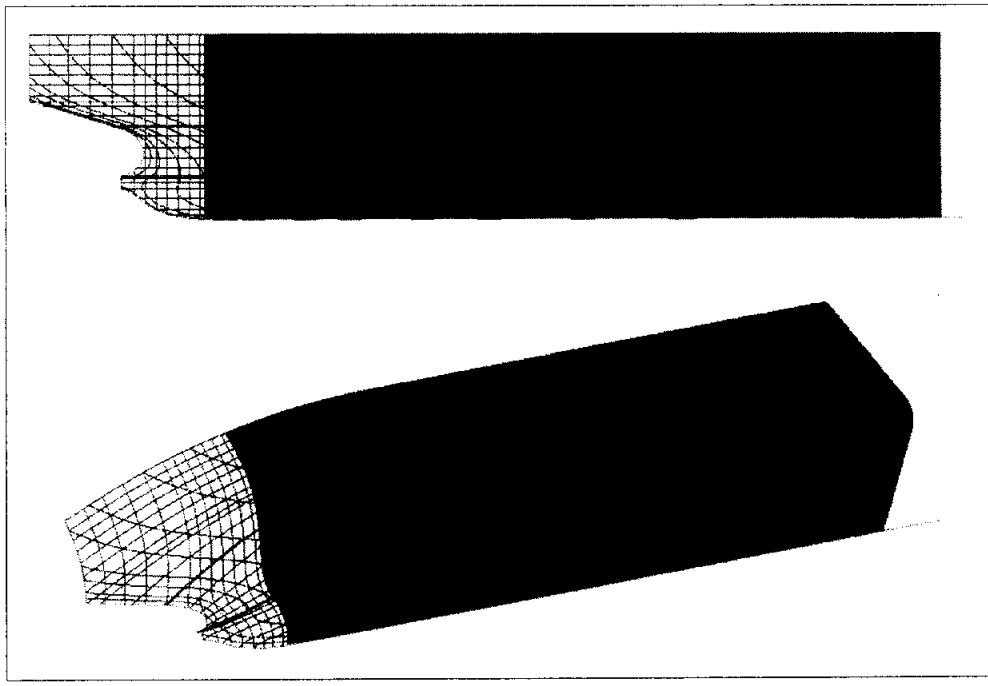


Fig. 3.14 Create No. 4 Surface Patch

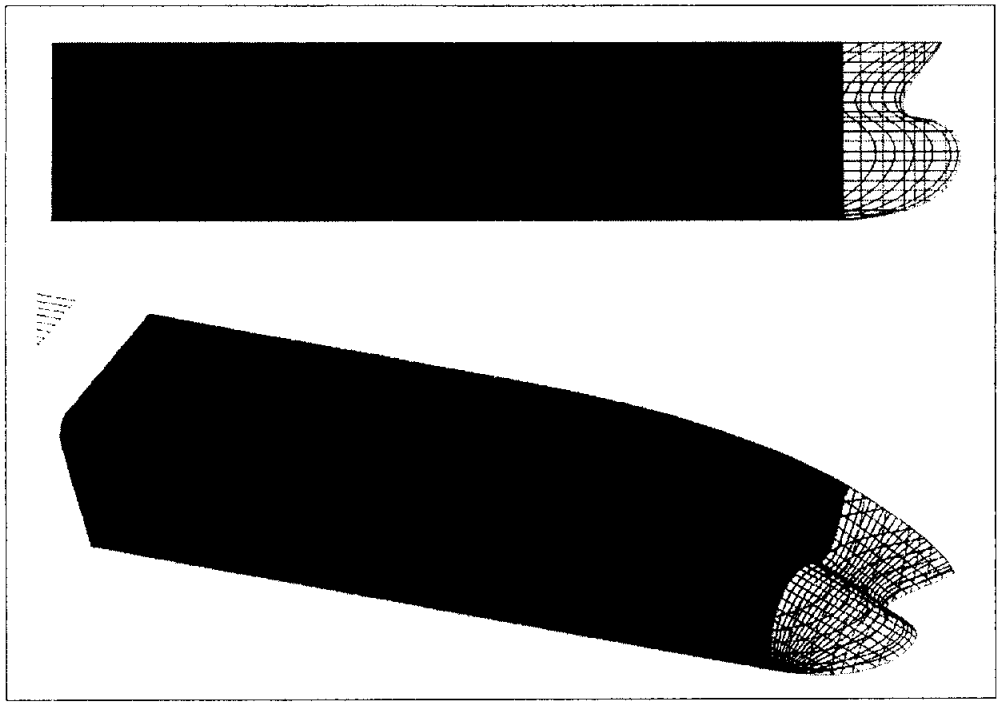


Fig. 3.15 Create No. 5 Surface Patch

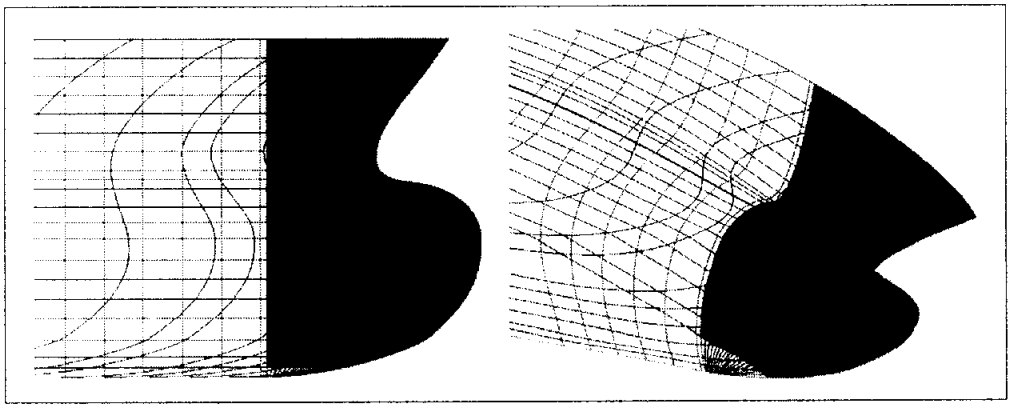


Fig. 3.16 Create Fwd Bulb Part Surface Patch

Fig. 3.16의 선수 벌브부분은 곡면화 하기에 앞서 Fig. 3.9의 6번 점과 6번 점이 이 지나는 Station Line을 기준으로 Center Line까지 Section을 5도씩 분할하여 교차 곡선을 생성한 후 6번 점이 지나는 Station과 생성된 곡선을 이용하

여 v방향의 곡선군으로 정의하고 선수부 Water Line을 u방향 곡선군으로 정의하여 곡면을 생성하였다. 이때의 곡면화 방법은 Hermite Coons Blending 방법을 사용하여 근사적으로 NUB 곡면을 생성하였다.

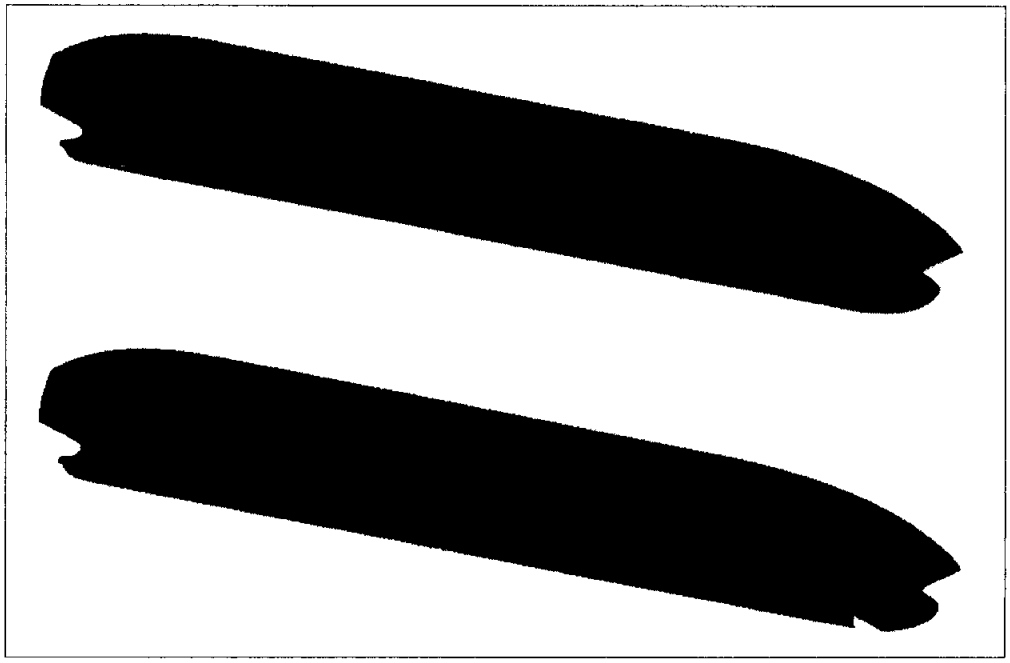


Fig. 3.17 Create Hull Form Surface

4. 곡면의 순정

선형을 곡면으로 표현하게 되면 이전의 Wire Frame 모델에서는 발견하기 힘든 순정도가 떨어지는 부분을 발견하기가 용이하다. 또한 지금까지의 곡면화 방법이 곡선망 정보를 근사적으로 Blending하여 표현하는 방법을 사용하고 있어 곡선망 정보가 충분하지 못한 경우 원하지 않는 결과가 나올 수도 있다. 그러나 곡면을 수정하여 순정하는 것은 상당히 까다롭고 어려운 작업이다. 일반적으로 곡면을 변형하거나 순정하는 방법으로 Control Point를 수정하는 방법을 사용하고 있다. 그러나 선형과 같이 곡면의 면적이 매우 크고 Control Point가 수 천, 수 만개가 되는 경우에는 수정하고자 하는 Control Point를 찾기도 어렵거니와 이를 일일이 수정하는 일이란 결코 쉽지 않은 일이다. 따라서 선형 곡면 순정의 경우에는 다른 방법을 사용하는 것이 합리적이라고 판단하고 본 연구에서는 곡면을 순정하는 여러 방법들 중에서도 곡선의 Point 데이터를 수정하는 방법을 사용하여 순정할 수 있는 방법을 프로그램에서 구현하여 모색하여 보았다.

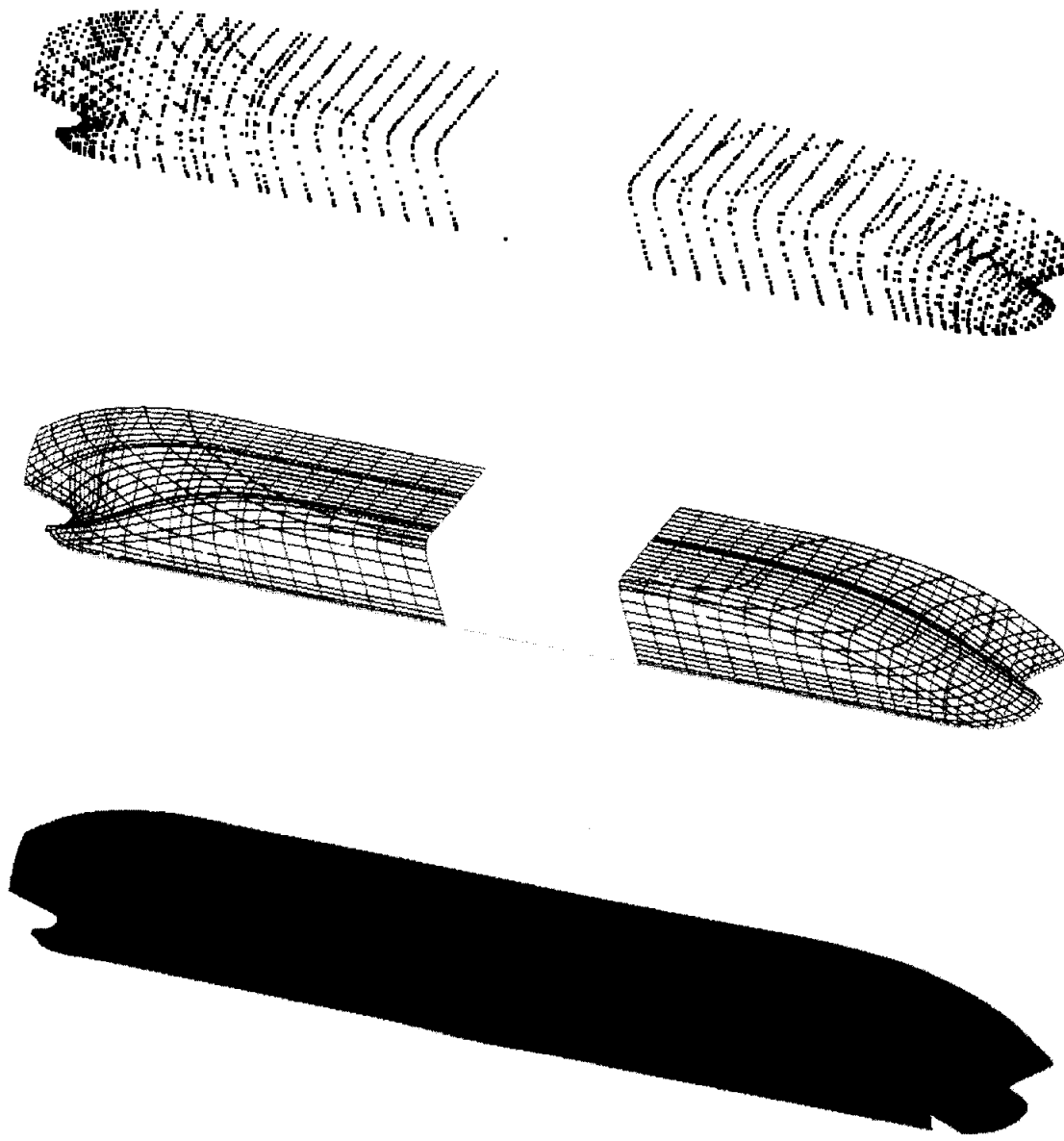


Fig. 4.1 Data Point, Curve and Surface View of Hull

Fig. 4.1은 각각 데이터 Point, 곡선 그리고 곡면을 이용하여 선형을 표현하였다.

OpenGL기반의 프로그램에서 화면상의 마우스 포인터를 이용하여 데이터 Point를 선택하는 기능을 구현하기 위해서는 먼저 서로 다른 View를 사용하는 OpenGL의 View와 Windows View를 일치시켜주는 것이 선행되어야 한다.

본 연구에서 제작된 프로그램에서는 사용자가 직접 데이터 Point 좌표를 입력하는 방법과 키보드를 사용하여 x, y, z, 방향으로 일정한 Step으로 데이터 Point 좌표를 수정하는 방법을 지원한다. 이렇게 수정된 데이터 Point 좌표는 바로 DB에 저장이 되며, 수정된 데이터 Point의 정보를 가지고 있는 곡선과 곡면을 다시 Fitting함으로써 화면을 갱신한다.

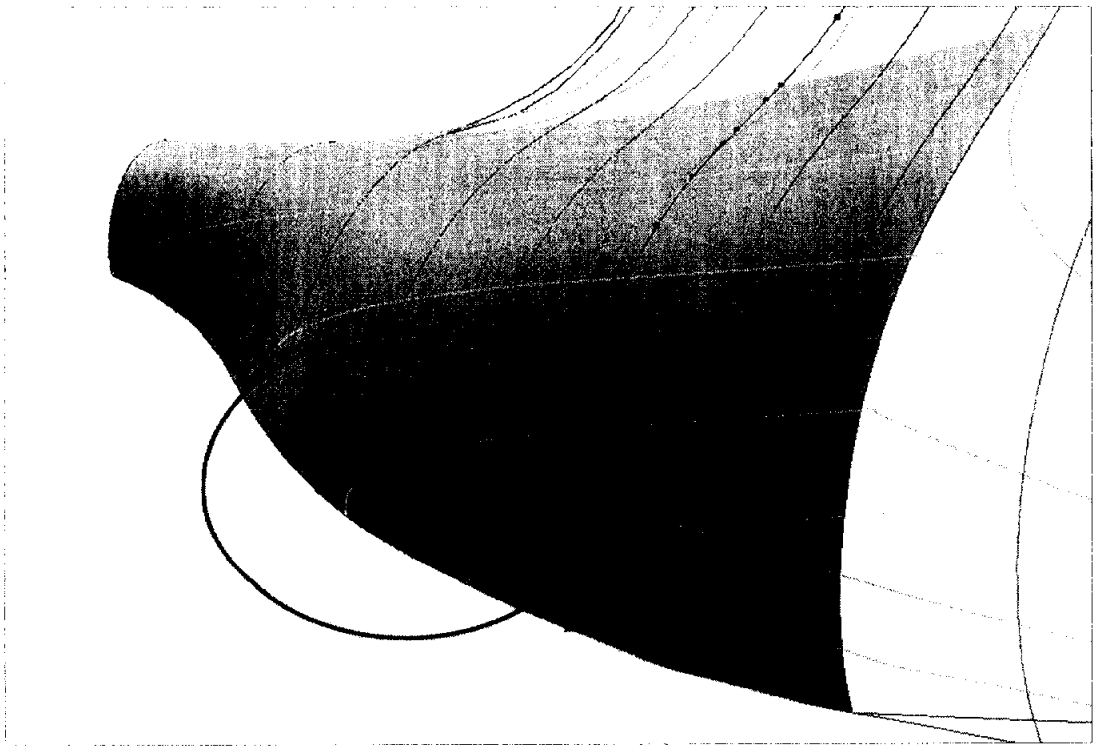


Fig. 4.2 Hull Aft Part No. 3 Surface

Fig. 4.2의 원으로 표시된 부분은 선미부에서 곡면의 순정도가 떨어지는 부분을 보여주고 있다. Buttock Line의 순정이 잘못된 것을 확연히 알 수 있다. 이러한

경우 기존의 Wire Frame 기반의 선형 순정 시스템에서는 Buttock Line뿐만 아니라 Station Line, Water Line의 교차 순정을 통해 순정을 실시한다. 그러나 Fig. 4.2와 같은 부분의 경우 순정 결과를 확인하기 어렵기 때문에 순정하는 작업이 쉬운 일이 아니다. 본 프로그램에서는 Wire Frame 기반의 선형 순정 시스템과 같이 교차 순정을 통해 순정을 하고 즉각적으로 곡면의 상태를 확인함으로써 순정 결과를 쉽게 파악할 수 있다.

다음의 Fig. 4.3은 선미부를 데이터 Point를 수정하는 방법으로 순정한 결과를 보여준다.

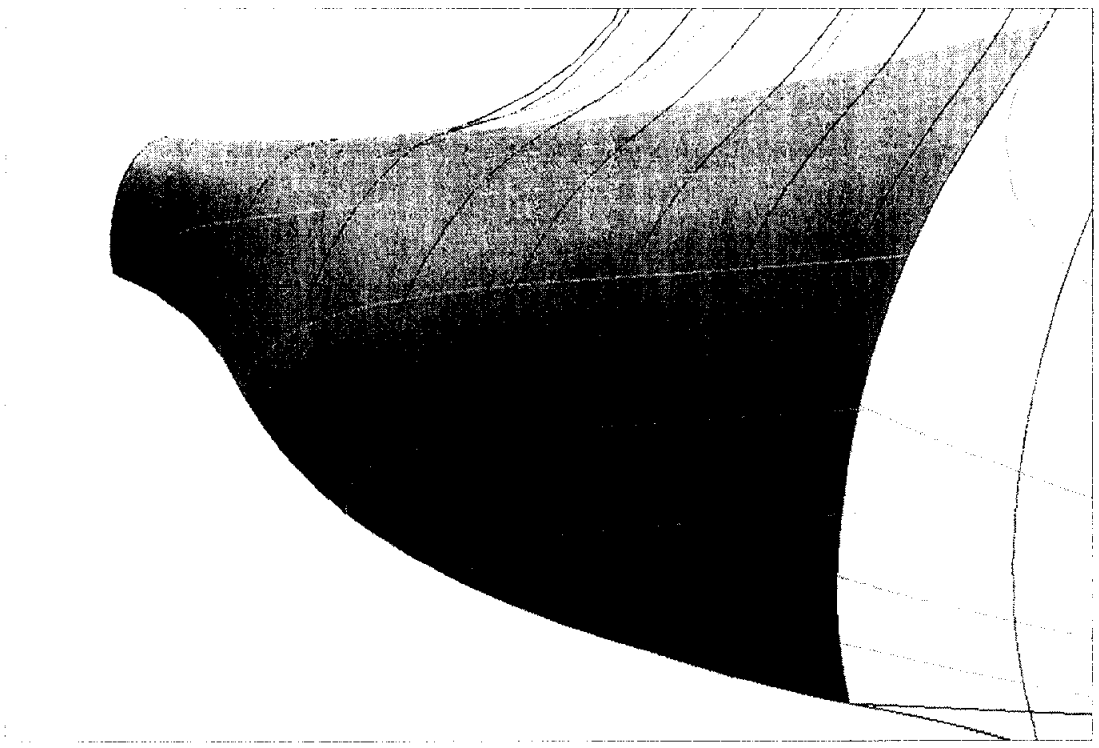


Fig. 4.3 No. 3 Surface' Surface Faring

5. 결과

5.1 프로그램의 주요 기능과 특징

본 연구에서 제작된 프로그램의 기능을 정리하면 다음과 같다.

1) GUI 관련 기능

① Viewing

키보드와 마우스의 간단한 조작으로 시점의 View의 Rotation, Zooming, Panning이 항상 가능하도록 하였다.

② Rubber Band 기능

곡선이나 곡면을 수정할 경우 변화된 형상을 즉각적으로 보여줌으로써 사용자의 판단을 도울 수 있도록 하였다.

③ Show/Noshow 기능

선형 데이터로부터 사용자가 원하는 형태의 정보, 즉 데이터 Point, 곡선, 곡면 정보를 선택적으로 가시화가 가능하며, 곡면의 경우 각각의 곡면 Patch별로 가시화가 가능하다.

2) 선형 곡면화와 곡면 순정 기능

① 선형의 가시화 기능

기존의 선형 곡선망 데이터로부터 Load한 데이터 Point, 곡선 그리고 프로그램에서 생성된 곡면 정보를 Widows View에 OpenGL Library를 이용하여 NURBS 곡선 및 곡면으로 가시화 하는 기능을 구현하였다.

② 선형 자동 곡면화 기능

선형 곡선망 데이터를 이용하여 선형을 총 7개~8개의 영역으로 나누고 각각의 영역에 NURBS 곡면 Patch를 생성하는 작업을 자동화하여 선형 곡면화에 필요한

절차를 최소로 간소화 하였다.

③ 곡면순정 기능

초기 선형 데이터로부터 곡선망 혹은 곡면을 생성한 후 곡면 상에 위치하는 곡선의 데이터 포인트를 수정하여 곡면의 품질을 향상시키기 위한 기능을 구현하였다. 이때 사용자는 데이터 포인트를 직접 입력하거나 Step을 정의하여 단계별로 수정이 가능하도록 하였으며, 이렇게 수정된 결과는 즉시 반영되어 가시화할 수 있도록 하였다.

5.2 실제 사용 예

곡면화와 곡면 순정에 사용된 모델선은 DWT 40,000급 Product Carrier이다.

각 기능에 대한 설명은 5.1장에서 이미 설명을 하였으므로 이번 장에서는 간단히 그림을 보여줌으로써 설명을 대신한다.

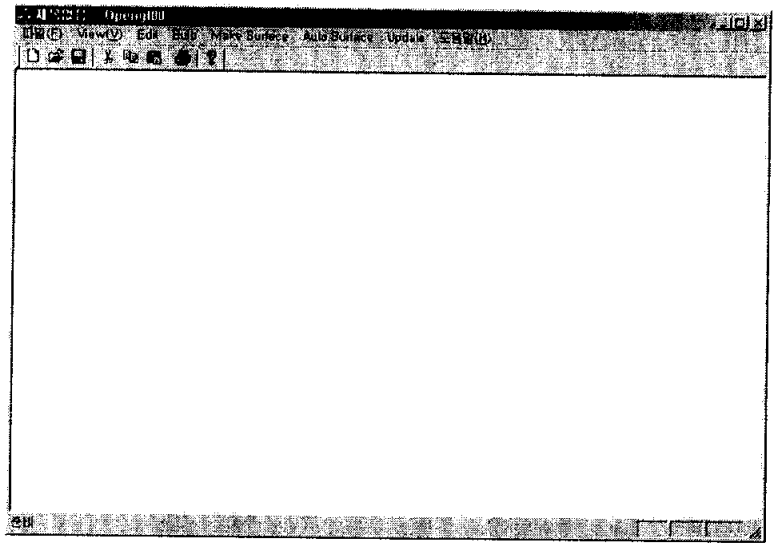


Fig. 5.1 Program Initial View

본 프로그램을 실행하면 Fig. 5.1과 같은 창이 뜬다.

5.2.1 기본 GUI

Fig. 5.2에서 Fig. 5.4까지의 그림은 프로그램의 기본기능을 보여준다.

1) File Load

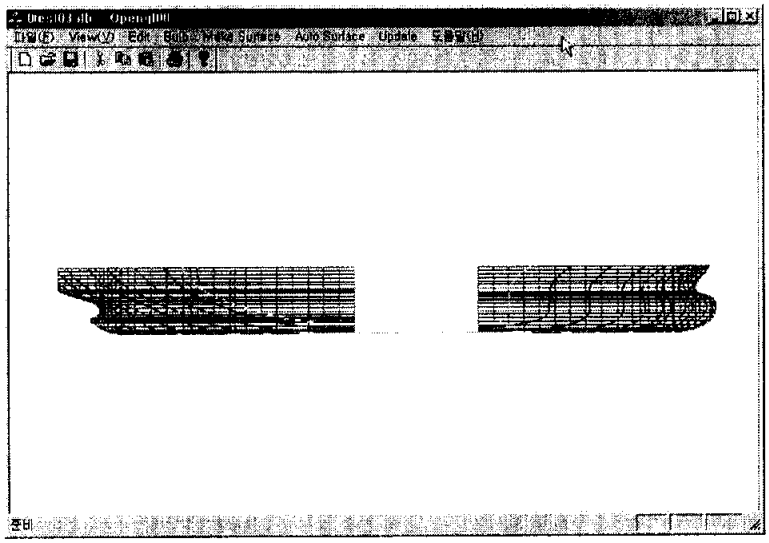


Fig. 5.2 Profile View(Load Hull Data)

2) Rotate View

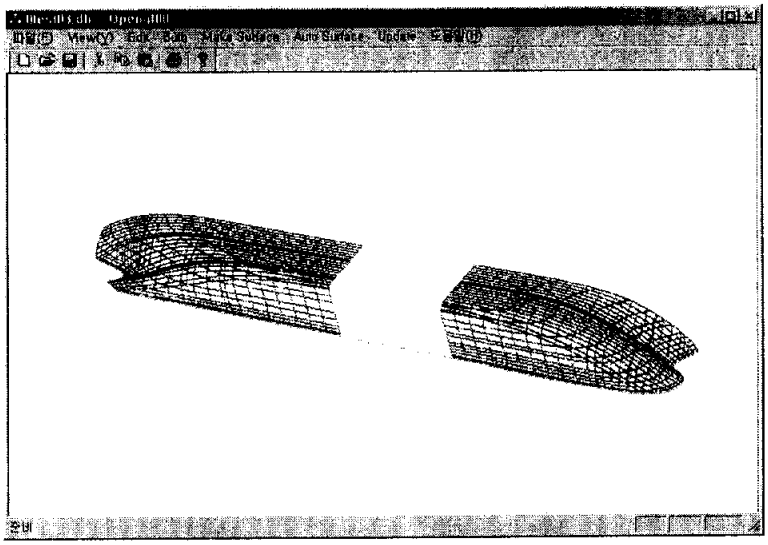


Fig. 5.3 Iso-metric View(Rotate)

3) Zooming

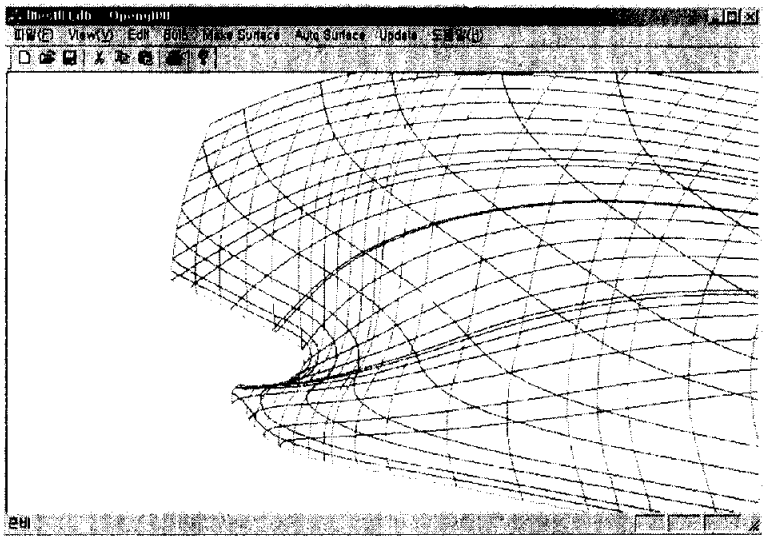


Fig. 5.4 Zooming

5.2.2 선형 곡면화 작업

1) 선형 전체의 곡면화

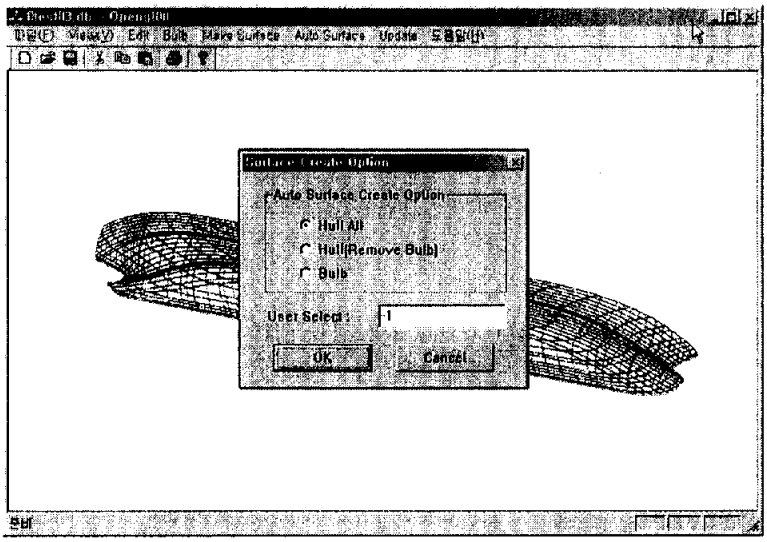


Fig. 5.5 Surface Create Option Dialog Box(Select Hull All)

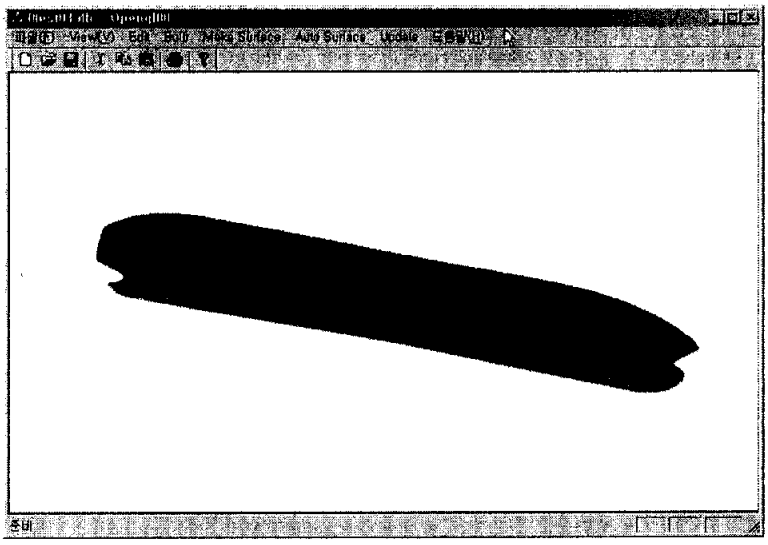


Fig. 5.7 Created Hull Surface

2) 부분 곡면화

부분 곡면화 Dialog Box의 옵션을 선택함으로써 선수 벌브부분을 제외한 나머지 부분만 곡면화 할 수도 있으며 선수 벌브만 곡면화 할 수도 있다. 그 외 원하는 곡면의 ID를 직접 입력하여 부분적으로 곡면화 할 수도 있다.

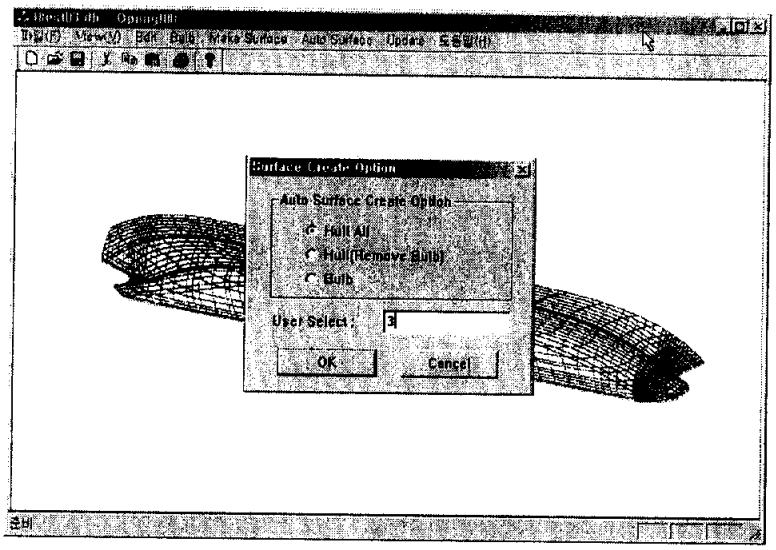


Fig. 5.8 Partial Create Surface Option(No. 3 Surface)

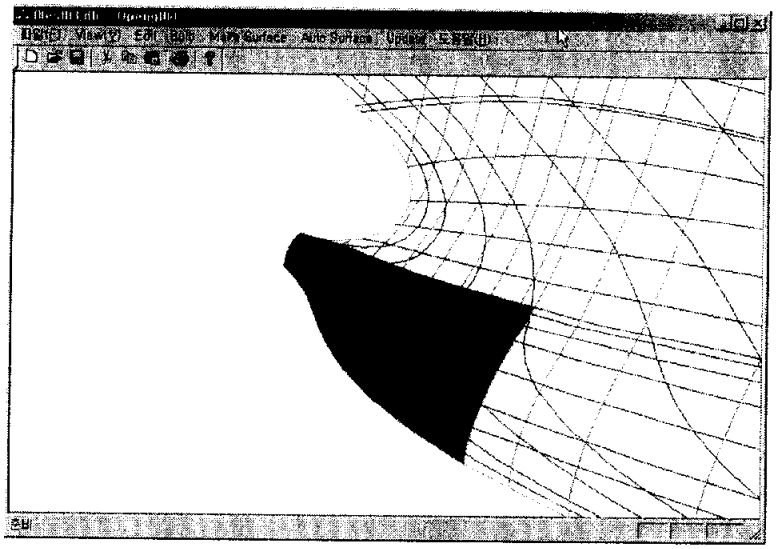


Fig. 5.9 Created Partial Surface(No. 3 Surface)

5.2.3 곡면의 순정작업

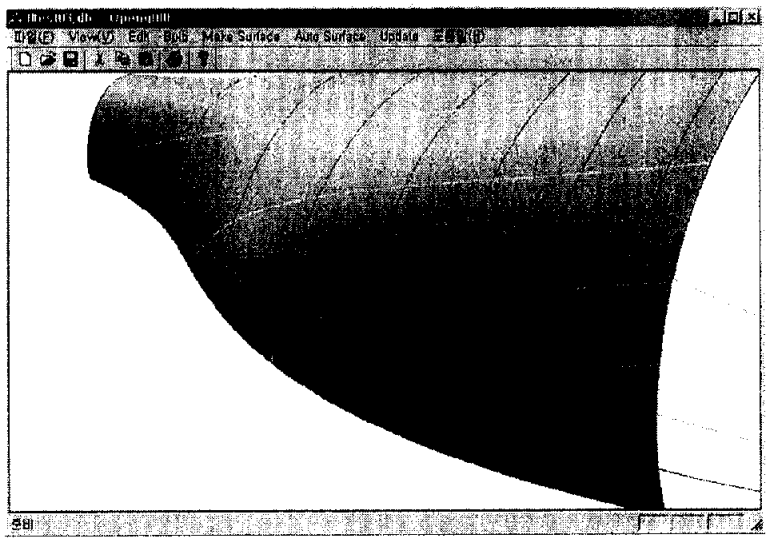


Fig. 5.10 Non Fair Surface(AFT Part)

1) 데이터 Point 좌표 직접입력하여 곡면을 순정하는 방법

(Fig. 5.11 ~ Fig. 5.12)

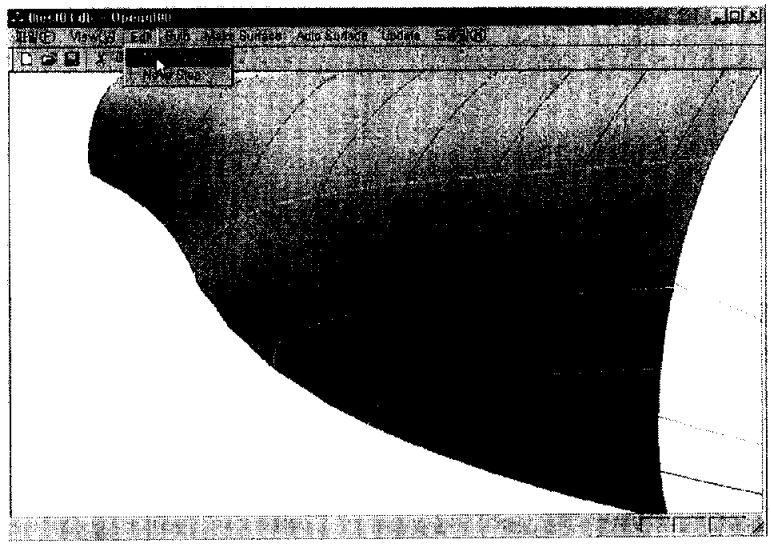


Fig. 5.11 Select Move Node(for Surface Fairing)

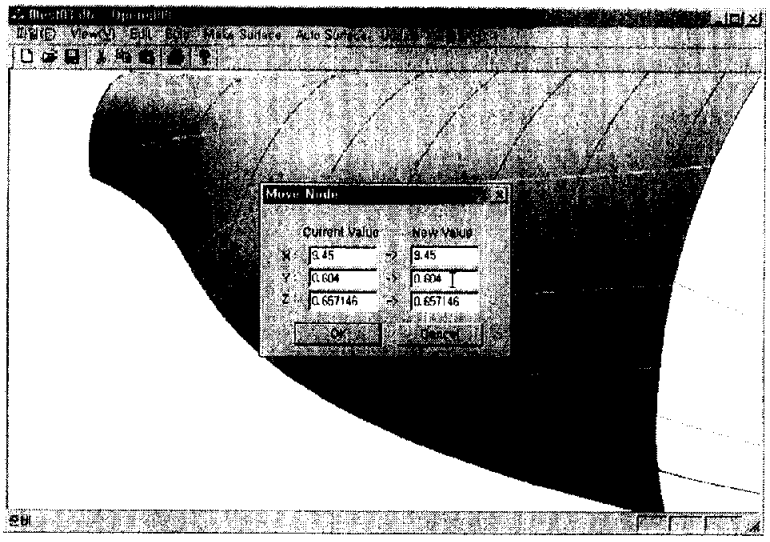


Fig. 5.12 Data Point Coordinate Change Dialog Box

2) 일정한 Step으로 데이터 Point 좌표를 변화시켜 곡면을 순정하는 방법 (Fig. 5.13~ Fig. 5.14)

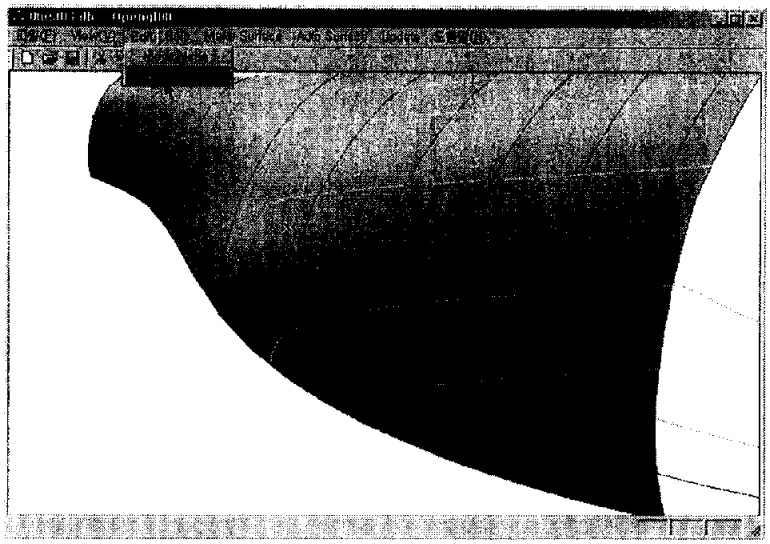


Fig. 5.13 Select Move Node by Step(for Surface Fairing)

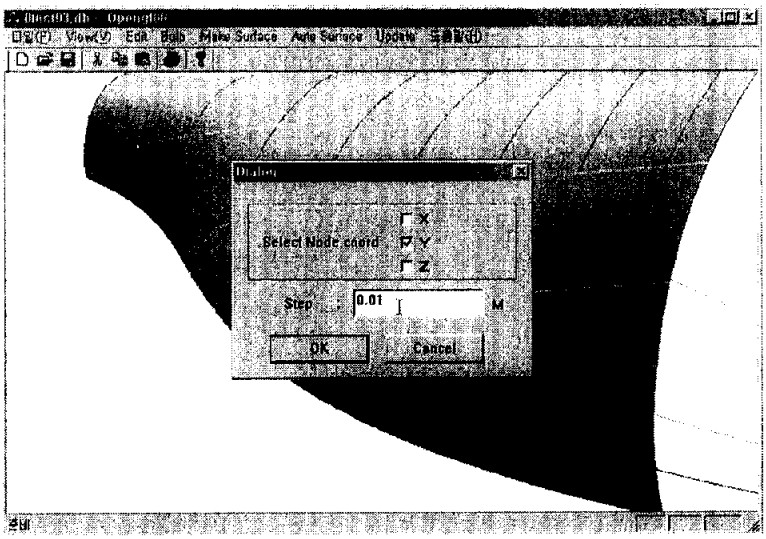


Fig. 5.14 Step Change Dialog Box

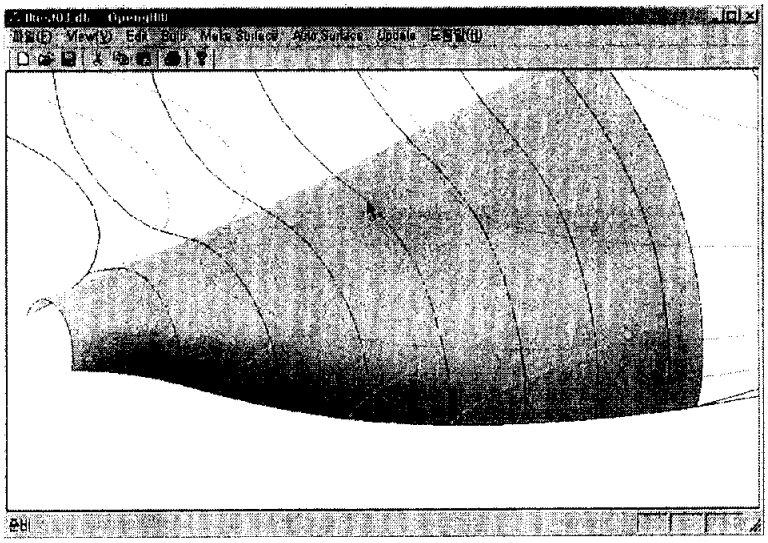


Fig. 5.15 Faired Surfac(AFT Part)

Fig 5.15는 5.2.3장의 두 가지 방법을 이용하여 곡면을 순정한 결과이다.

IV. 결론

지금까지 본 연구에서는 NURBS 곡선망과 NURBS 곡면을 이용하여 순정 가능한 NURBS 곡면군 생성 프로그램을 Visual C++의 MFC와 OpenGL을 이용하여 GUI로 제작하였으며 OpenGL Library를 본 프로그램에 도입함으로써 검증된 곡선과 곡면을 최적화된 알고리즘으로 가시화 하였다. 또한 번거로운 곡면 생성과정을 대폭 단축하여 최소한의 조작으로 선형 전체의 곡면을 구현하였으며, 기존의 Wire Frame 기반의 순정시스템과 같이 교차 순정법으로 선형곡면을 순정하고 즉각적인 곡면의 변화를 통해 순정 결과를 확인하는 방법을 제시하였다.

본 연구결과를 요약하면 다음과 같다.

1) Wire Frame 모델의 선형 데이터를 이용하여 선형 전체를 NURBS 곡면으로 생성, 표현하는 OpenGL기반의 프로그램을 제작하였다.

2) 자동으로 선형의 영역을 분할하고 곡면을 생성함으로써 선형의 곡면화를 위한 사용자의 작업량을 최소로 하였으며 곡선과 곡면 수정이 용이하도록 사용자 인터페이스를 강화하였다.

3) GUI를 이용하여 작업 결과를 시각적으로 확인하고 결과 처리를 위한 판단을 쉽게 할 수 있게 하여 작업 능률을 향상을 꾀하였다.

4) 곡면을 구성하는 곡선의 데이터 Point를 수정하는 방법으로 선형곡면을 순정하고 그 결과를 즉각적으로 확인할 수 있게 하였다.

이와 같은 연구 결과를 바탕으로 차후 연구에서는 다양한 선형에 적용 가능한 곡면생성 기법연구와 함께 곡면순정 기능의 향상, 곡면의 품질을 시각적으로 알기 쉽게 표시하는 기법에 대한 연구가 필요하다.

IV. 참고문헌

· 윤병호, 서승완, 김원돈, 김광욱, 1985, "B-Spline을 이용한 선체표면에 관한 연구", 대한조선학회지, 제 22권 제3호, p19~p26.

· D.F. Rogers, J.A. Adams, 1989 "Mathematical elements for computer graphics", 7th edition, McGraw-Hill

· Byoung K. Choi, 1991, "Surface Modeling for CAD/CAM", ELSEVIER.

· 김수영, 우일국, 1992, "B-Spline 곡면기법을 이용한 곡면형상 도출", 대한조선학회 논문집, 제 29권 제3호, p8~p17.

· 김동준, 윤태경, 1994, "선형의 순정기법에 관한 기초 연구" 대한조선학회 논문집, 제 31권 제 2호, p15~p21.

· 박지선, 김동준, 1994, "GC^1 곡면을 이용한 선형의 표현", 대한조선학회 논문집, 제 31권 제4호, p8~p17.

· 임중현, 이규열, 1997, "베지에 곡선모델(드 카스텔죠 알고리듬)을 이용한 곡면 통합 모델링 기법", 대한조선학회 논문집, 제 29권 제3호, p127~p138.

· 윤태경, 1998, "선체 외판 모델링을 위한 CAD 시스템 개발", 부경대학교 공학석사 학위논문, p6~p18.

· 신현경, 박규원, 박호균, 김일환, 2000, "NURBS 곡면을 이용한 선형표현에 관한 연구 - 자유곡면 가공기계 개발(I)", 대한조선학회 논문집, 제 37권 제 2호, p109~p117.

· 박성우, 2001, "영업설계를 위한 기본설계통합시스템" 부경대학교 공학석사 학위논문, p51～p55.

· 이준호, 2003, "범용 CAD Program에서의 응용을 위한 선형 곡면화 방법론에 관한 연구", 부경대학교 공학석사 학위논문.

· 메이슨 우, 젝키 네이더, 톰 데이비스, 데이브 슈리너, 2003, "OpenGL 프로그래밍 가이드 Third Edition", 인포북.

· 김상혁, 1998, "비주얼 C++ 정복 6.0", 가남사.

· Microsoft Windows 홈페이지, http://www.microsoft.com/korea/windows

· 이지그라프사 홈페이지, http://www.ezgraph.co.kr/

· TRIBON사 홈페이지, http://tribon.com

감사의 글

벌써 대학 합격 통지서를 받은 지 10년이 되었습니다. 이렇게 시간이 흐르는 동안 무엇을 하려했고 이루었는지 다시금 생각해 보며 지금까지 도움을 주셨던 많은 분들의 기억을 더듬어 봅니다.

먼저 학부와 대학원 과정 동안 저에게 학문의 길과 사람이 살아가야할 올바른 길을 일깨워 주시고 격려해 주신 인생의 스승님, 김동준 교수님께 진심으로 감사드립니다. 또한 저의 부족한 논문을 심사하시느라 수고하신 김인철 교수님과 김용직 교수님께 감사드립니다. 올해를 마지막으로 강단을 떠나시는 홍봉기 교수님 언제까지나 건강하시길 기원합니다. 모자란 저에게 가르치심을 베푸신 배동명 교수님과 구자삼 교수님께도 감사의 말씀을 올립니다.

연구실에서 학부시절부터 가장 오랜 시간을 같이 있었고 많은 것을 가르쳐주신 윤태경 수석님, 만날 때마다 제가 한없이 작아져야하는 고등학교 선배이신 심상목 선임님께 감사드립니다. 제가 나태하지 않도록 일깨워주신 연구실의 대부 민경철 선배님, 감사의 말씀과 함께 대모 형수님과 언제나 행복하시길 바랍니다. 작년 한 해 동안 동거동락하며 정말 많은 도움을 주셨던 박종헌 박사님, 주말마다 고생 참 많으십니다.
졸업한지 2년이 넘었지만 여러모로 저의 모범이 되어주신 성우 형, 제게 학교술과 상주모드의 묘미를 알게 해 주신 송한이 형, 밤마다 힐링포션을 외치며 전장을 함께 누빈 태윤이 형, 그리고 만년 막내 준호, 모두 제게 고맙고 그리운 분들입니다. 아무쪼록 자신의 길에서 성공하시길 빕니다.
만학의 즐거움에 빠지신 박근옹 선생님, 영국에 보낸 애인을 애타게 기다리는 석봉이, 요즘 작업모드에 바쁜 현수, 그리고 지금 영국에서 공부하고 있을 연희, 연구실에서 잡일을 도맡아 했던 동훈이, 미운 짓만 골라하는 미례, 나까야마씨, 고마운 마음과 함께 베풀지는 못하고 도움만 받다가 떠나버리는 것 같아 미안한 마음도 듭니다. 모두들 자기가 뜻하는 바를 이루었으면 합니다.

대학원 생활동안 많은 도움을 주신 배성룡 교수님, 하영록 선배님, 신기석 선배님, 이승철 선배님, 신창혁 선배님, 김철현 선배님께도 감사의 말씀을 올립니다. 그리고 언제나 바쁜 도군이와 저랑 같은 올빼미족 교덕이에게도 고맙다는 말씀을 전합니다.

언제 돈벌어 술 살꺼냐면서도 지금까지 잘 기다려준 제 오랜 친구들, 진용, 두혁, 영동이, 이런 친구들이 있어 사는 게 심심하지 않았습니다.
또 놀아주지 않는다고 항상 뾰족해져 있지만 제가 연구실 생활 한 것만큼이나 오래 기다려준 여자 친구 영금이에게도 이 자리를 빌어 감사의 말씀을 전합니다.

마지막으로 못난 저를 항상 믿어주시고 사랑으로 보살펴주신 아버지와 어머니에게 머리 숙여 감사드리며, 무중, 무락 두 동생에게도 고마운 맘을 전합니다.