한국 CAD/CAM 학회 논문집
제 6권 제 2호 2001년 6월 pp. 78-88

기술논문

그래프의 분석과 병합을 이용한 기하학적제약조건 해결에 관한 연구

권오환*, 이규열**, 이재열***

A Study on the Geometric Constraint Solving with Graph Analysis and Reduction

Kwon, O. H.*, Lee, K. Y.** and Lee, J. Y.***

ABSTRACT

In order to adopt feature-based parametric modeling, CAD/CAM applications must have a geometric constraint solver that can handle a large set of geometric configurations efficiently and robustly. In this paper, we describe a graph constructive approach to solving geometric constraint problems. Usually, a graph constructive approach is efficient, however it has its limitation in scope; it cannot handle ruler-and-compass non-constructible configurations and under-constrained problems. To overcome these limitations, we propose an algorithm that isolates ruler-and-compass non-constructible configurations from ruler-and-compass constructible configurations and applies numerical calculation methods to solve them separately. This separation can maximize the efficiency and robustness of a geometric constraint solver. Moreover, the solver can handle under-constrained problems by classifying under-constrained subgraphs to simplified cases by applying classification rules. Then, it decides the calculating sequence of geometric entities in each classified case and calculates geometric entities by adding appropriate assumptions or constraints. By extending the clustering types and defining several rules, the proposed approach can overcome limitations of previous graph constructive approaches which makes it possible to develop an efficient and robust geometric constraint solver.

Key words : Geometric constraint solver, Feature based design, Parametric design, Constraint graph

1. 서 언

현재 산업계에서는 설계에 소요되는 시간을 줄임으로써 경비를 절감하고 시장의 변화에 민첩하게 대응할 수 있는 방법을 찾아내기 위해서 많은 노력을 기울이고 있다. 따라서 CAD시스템 개발자들은 제품의 모델링에 소요되는 시간을 줄이기 위해서 새로운 모델링 방법들을 개발해 왔다. 그러한 기술들 중에서 현재의 CAD시스템에 많이 사용되고 있는 기술이 바로 특징형상을 기반으로 한 모델링(feature based modeling) 방법과 매개변수를 이용한 모델링(parametric modeling) 방법이다. 이러한 방법을 이용하면 제품의 치수 변경 등에 따른 재 작업시간을 줄일 수 있을 뿐만 아니라 기 모델링 된 제품이나 부품의 결과를 재활용할 수 있으므로 Fig. 1에 나타난 것처럼 설계변경을 빠르고 쉽게 수행할 수 있다.

이러한 장점 때문에 현재 기계분야에서는 특징형상을 기반으로 한 CAD시스템을 많이 이용하고 있으나 조선분야에는 아직 이용되지 않고 있다. 그러나 선체의 구조를 이루는 요소들 중에는 Fig. 2와 같이 특징형상으로 표현할 수 있는 것들이 많이 있으며, 이와 같은 설계방법을 적용하면 설계과정에 소요되는 시간을 줄일 수 있다.

이와 같이 특징형상과 매개변수를 이용한 설계방법

*학생회원, 서울대학교 조선해양공학과
**종신회원, 서울대학교 조선해양공학과
***정회원, 한국전자통신연구원 동시공학연구팀
- 논문투고일: 2000. 11. 9
- 심사완료일: 2001. 2. 19

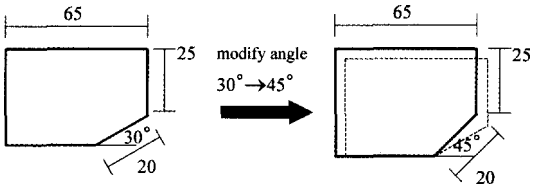


Fig. 1. An example of parametric design.

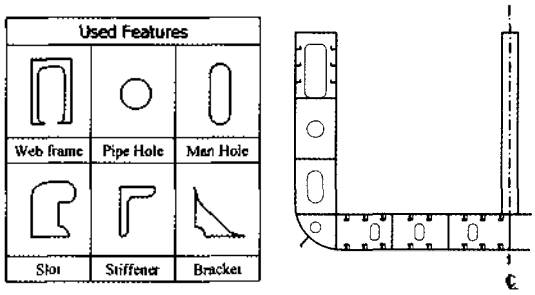


Fig. 2. An example of ship structural model composed with parametric features.

을 적용하기 위해서는 기하요소들 사이의 제약조건을 해석해서 풀어낼 수 있는 기하학적 제약조건 해결자(geometric constraint solver)가 반드시 필요하다.

따라서, 본 연구에서는 이러한 모델링 방법을 조선 CAD 시스템에 도입하기 위해서, 기하학적인 제약조건 해결자를 만드는 것을 목적으로 한다.

2. 관련 연구 및 현황

기하학적 제약 조건 문제는 Fig. 3과 같이 기하요소와 기하요소 사이의 제약조건으로 이루어져 있으며, 기하학적 제약조건 해결자는 제약조건을 만족하도록 기하요소의 위치와 형태를 결정해주는 역할을 한다. 따

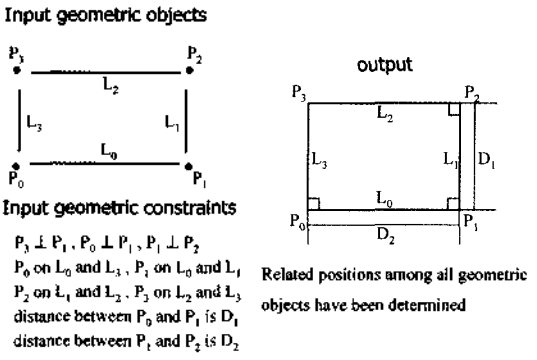


Fig. 3. Geometirc constrained problem.

라서 기하학적 제약조건 해결자는 매개변수를 이용한 모델링방법에 있어서 가장 중요한 요소이다.

지금까지 개발된 기하학적 제약조건 해결자는 크게 수치적 방법(numerical approach)과 constructive approach로 나눌 수 있다.

수치적 방법은 기하학적인 제약조건을 연립방정식으로 변환하고 이러한 연립방정식을 풀어냄으로써 제약조건을 해결하는 방법이다. 수치적 방법의 경우는 기하학적 제약조건문제 전체를 하나의 계산과정으로 풀어낼 수 있으며, under-constrained인 경우도 다룰 수 있다는 장점이 있지만, 원하는 해를 얻기 위한 초기조건을 구하기 어렵고 안정성이 부족하기 때문에 이러한 오류가 누적되어 심각한 결과를 초래할 수도 있으며, 반복 계산을 수행하므로 계산 속도가 느리다는 단점을 가지고 있다[1,2]. 이러한 단점들을 해결하기 위해서 지금까지 수치적인 계산을 위해서 사용해온 newton-raphson 방법 대신 homotopy[3]나 최적화[4]를 이용하는 방법들이 제안되었으나 여전히 계산속도가 느리다는 단점은 해결되지 못하고 있다.

Constructive approach는 사람이 자와 컴퍼스를 이용하여 제도하는 것과 비슷하게 이미 정의된 기하요소들을 기준으로 새로운 기하요소들의 위치와 형태를 점차적으로 구성해 나가는 방법이다[5-12]. 따라서 constructive approach에서는 기하요소의 계산순서를 결정하는 것이 중요한 문제가 된다. 대부분의 constructive approach는 기하학적인 제약조건 문제를 그래프(Graph)로 표현하고 그래프를 분석함으로서 기하요소의 계산순서와 방법을 결정하고 있다. 기하학적 제약조건 문제에서 대부분의 기하요소는 자와 컴퍼스를 이용하여 제도하듯이 순차적으로 값을 계산하는 것이 가능한데, 이러한 경우를 ruler-and-compass constructible이라고 하며 간단한 대수학적 계산에 의해서 해결된다. 그러나 이와 같은 방법만으로는 계산이 불가능한 경우가 존재하며 이러한 경우를 ruler-and-compass non-constructible이라 한다. 현재 대부분의 constructive approach는 기하학적 제약조건 문제가 well-constrained이고 ruler-and-compass constructible인 경우만 언급하고 있는데, 실제 설계과정에서 나타나는 기하학적 제약조건 문제들을 살펴보면 대부분의 문제들이 ruler-and-compass non-constructible인 형태를 포함하고 있으며, 설계 과정 중에서 under-또는 over-constrained가 된다. 따라서 constructive approach를 이용하기 위해서는 ruler-and-compass non-constructible인 경우와 under- 또는 over- constrained인 경우를 해결하기 위한 방법을 추가적으로 구성해야 한다. constructive approach의 대표적인 예로는 Owen[5]

과 Lee[7,8]를 들 수 있다.

Owen[5]은 D-cubed Ltd.의 설립자로서 DCM을 개발했으며, DCM은 현재 많은 캐드시스템에서 매개변수를 이용한 설계방법을 지원하기 위한 모듈로서 사용되고 있다. Owen은 기하학적 제약조건 문제를 그래프를 이용하여 표현하고, 삼각형 형태의 서브그래프로 분할한 후 이를 다시 병합함으로써 계산순서를 결정하는 방법을 제안하였으나 논문에는 well constrained이며 ruler-and-compass constructible인 경우에 대해서만 언급하고 있다.

Lee[7,8]는 기하학적 제약조건을 표현하는 그래프에서 계산이 가능한 형태를 나타내는 clustering type들을 제안하였으며 이것을 병합하여 기하요소의 계산순서를 결정하는 방법을 개발하였다. 그리고 clustering type들을 ruler-and-compass constructible과 ruler-and- compass non-constructible인 경우로 나누어 각각 대수학적인 계산방법과 수치적인 반복계산을 적용하여 계산함으로서 constructive approach의 단점을 보완하였다.

본 연구에서 Lee가 제안한 clustering type들을 basic clustering subgraph로 하고, well-constrained이더라도 basic clustering subgraphs만으로 해결할 수 없는 경우와 under-constrained인 경우도 해결할 수 있도록 새로운 clustering subgraph들을 제안한다. 계산과정은 크게 construction plan생성과정과 construction plan실행과정으로 구성되어 있는데, construction plan생성과정에서는 기하요소의 계산순서와 방법을 결정하며, construction plan실행과정에서는 앞에서 결정한 계산순서 따라서 ruler-and-compass constructible인 경우는 대수학적인 방법을, ruler-and-compass non-constructible인 경우는 수치적인 반복계산방법을 적용하여 기하요소의 값과 위치를 계산한다. Fig. 4는 사각형 모양의 통안에 직사각형 모양의 막대기를 집하게 놓는 문제로서 ruler-and compass non-constructible인 경우를 포함하

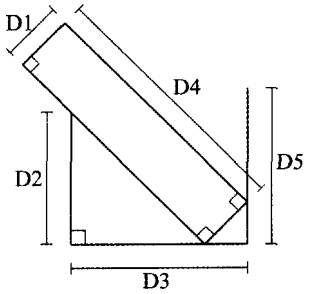


Fig. 4. "A bar lies in the box" problem (ruler-and-compass non-constructible problem)[13].

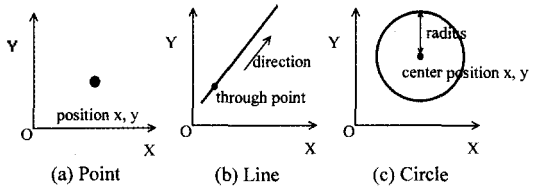


Fig. 5. Representation of geometric entities.

고 있는 예이다. 따라서 Fig. 4를 풀기 위해서는 수치적인 반복계산을 수행해야 한다.

3. 그래프를 이용한 기하학적 제약 조건 해결자

3.1 기하요소

본 연구에서 개발된 제약조건 해결자는 Fig. 5에 나타난 것처럼 2차원 평면상의 점, 직선, 원으로 이루어진 2차원 강체를 대상으로 한다. 여기서 직선은 무한직선을 나타낸다.

3.2 기하요소의 자유도

2차원 평면 위의 강체는 하나의 축에 대한 회전과 두 방향으로 병진운동이 가능한 자유도를 가지게 된다. 그러나, 2차원 평면 위의 점, 직선, 원은 Fig. 6에 나타난 것처럼 서로 다른 자유도를 가지게 된다.

3.3 기하요소와 제약조건의 관계

제약조건은 앞에서 언급한 기하요소들을 원하는 형태로 위치시키기 위한 기하학적인 연관관계나 값이다.

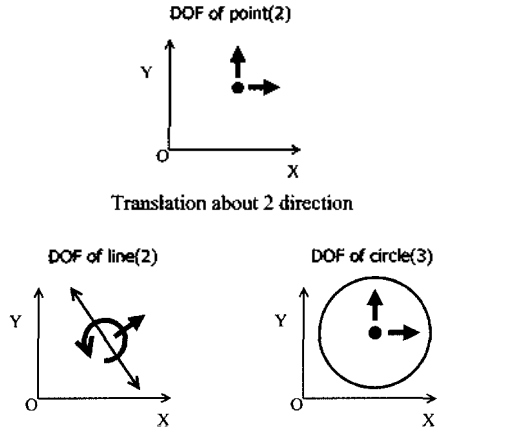


Fig. 6. DOF(Degree of Freedom) of geometric entities.

Table 1. Geometric constraints and their valency

Constraint type	Associated geometric entities	Valency
Distance	Point, Point	1
	Point, Line	1
	Point, Circle	1
	Line, Line	2
Incidence	Point, Line	1
	Point, Circle	1
Coincidence	Point, Point	2
	Line, Line	2
Tangency	Line, Circle	1
	Circle, Circle	1
Angle	Line, Line	1
Parallelism	Line, Line	1
Concentricity	Point, Circle	2

그러나 어떠한 제약조건이 모든 기하요소에 대해서 적용될 수 있는 것은 아니다. 따라서 기하요소와 제약조건을 Table 1과 같이 분류하면 기하요소와 제약조건 사이의 가능한 조합을 쉽게 알아볼 수 있다[14].

Table 1에 표시된 valency는 기하요소에 각각의 제약조건이 주어졌을 때 사라지는 기하요소의 자유도이다. 그리고 Table 1에서 두 직선 사이의 Distance와 Coincidence, Concentricity의 경우는 valency가 2이다.

3.4 그래프를 이용한 기하학적 제약조건 문제의 표현과 구조적인 분석

본 연구에서 제안한 기하학적 제약조건 해결자는 제약조건 문제의 기하요소는 그래프의 정점(vertex)으로, 기하요소 사이의 제약조건은 그래프의 간선(edge)을 이용하여 표현하였다. 이와 같이 기하학적 제약조건 문제를 나타내는 그래프를 제약조건 그래프(constraint graph)라고 한다. Fig. 7은 "사각통 내에 접하는 막대" 문제를 제약조건 그래프로 표현한 것이다.

이와 같이 그래프를 이용하여 제약조건 문제를 표현하면 아래와 같은 특성들을 정의할 수 있으며, 제약조건 문제가 well-constrained인지 또는 over-, under-constrained인지 알아낼 수 있다[12]. 이때 모든 정점의 DOF는 2이며, 간선의 valency는 1이다.

Definition 1: 제약조건 그래프 $G=(V,E)$에 대해서 $|V|=n$이고 $|E|=m$일 때 $m=2\times n-3$이고, G의 임의의 subgraph인 $G'=(V',E')$이 $|V'|=n'$이고 $|E'|=m'$일 때 $m'\leq 2\times n'-3$을 만족하면, G는 well-constrained 이다.

Definition 2: 제약조건 그래프 $G=(V,E)$에 대해서, G의 임의의 subgraph인 $G'=(V',E')$이 $m'>2\times n'-3$인 경우가 존재하면, G는 over-constrained인 부분을 포함하고 있다.

Definition 3: 제약조건 그래프 $G=(V,E)$에 대해서, over-constrained인 부분이 없고, $m<2\times n-3$이면, G는 under-constrained이다.

3.5 그래프의 정점 병합을 이용한 기하학적 제약조건 해결자

Lee는 기하학적 제약조건 문제가 잘 정의된 경우, 이것을 그래프를 이용하여 표현하고 Fig. 8의 clustering type들을 이용하여 기하요소의 계산순서와 방법을 결정하는 방법을 제안하였다. 본 연구에서는 이러한 clustering type들을 basic clustering subgraph로 정의하였다.

Fig. 8의 basic clustering subgraph들을 살펴보면 (a), (b), (c), (d)의 모든 경우가 제약조건이 가지는 valency 만큼의 자유도를 빼고 나면 3개의 자유도를 가지게되므로 강체를 형성할 수 있다. 따라서 제약조건 해결자는 제약조건 그래프에서 위의 basic clustering subgraph들을 찾아내어 하나의 정점으로 병합하고 정점의 기하요소는 강체에 포함시킴으로서 기하요소의 계산 순서를 결정할 수 있다. 이때 병합된 결과로 나타난 정점을 클러스터(cluster)라 하며 클러스터의 자유도는 3이다. 제약조건 해결자는 모든 기하요소가 하나의 강체로 변환될 때까지 이러한 과정을 반복하게 되는데 기하학적 제약조건 문제가 잘 정의된 대부분의 경우(well-const-

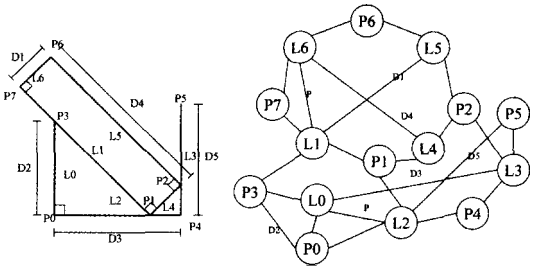
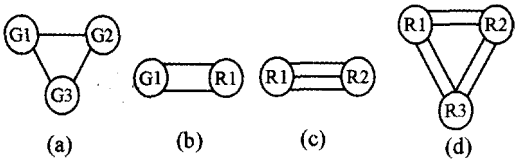


Fig. 7. An example of constraint graph.



G_i : geometric entity(or cluster) with 2 DOF
R_i : clusters (or geometric entities) with 3 DOF

(a),(b) : Ruler-and-compass constructible
(c),(d) : Ruler-and-compass non-constructible

Fig. 8. Basic clustering subgraphs.

rained problem) 제약조건 그래프에 하나의 정점만이 남게 된다. 그리고 기하요소의 값은 일반적으로 대수학적인 계산방법을 이용하여 계산한다. 그러나 Fig. 8의 (c)와 (d)같은 경우는 강체들 사이에 제약조건이 주어진 경우이므로 비선형방정식으로 표현된다. 따라서 (c)와 (d)는 수치적인 방법으로 계산한다.

본 연구에서는 기하학적 제약조건 해결자를 개발하기 위해서 앞에서 설명한 basic clustering subgraph와 그래프의 정점을 병합하는 방법을 이용하여 문제 해결 과정을 아래와 같이 두 단계로 구성하였다.

단계 1: 제약조건 그래프를 이용하여 construction plan생성과정을 거쳐 기하요소의 값을 계산하는 순서와 방법을 결정한다.

단계 2: 단계 1에서 결정한 순서와 방법에 따라 construction plan실행과정을 수행하여 기하요소의 값을 결정한다.

3.5.1 제약조건 해결자의 construction plan생성과정

construction plan생성과정은 제약조건 그래프를 이용하여 기하요소의 값을 계산하기 위한 순서와 방법을 결정하는 과정이다.

본 연구에서 제안된 construction plan생성과정은 다음과 같다.

과정 0: 전처리과정을 통해서 의사(pseudo) 기하요소를 생성한다.

과정 1: 제약조건 그래프가 정점하나로 구성되어 있으면 construction plan생성과정을 끝낸다. 그렇지 않으면 과정 2를 수행한다.

과정 2: 초기 조건을 가정해서 초기 클러스터를 만든다. 초기 클러스터는 기하요소 두 개의 자유도 3개를 가정해서 형성할 수 있다. 예를 들면 점과 그 점을 지나는 직선의 경우 점의 좌표값과(자유도 2) 직선의 방향(자유도 1)을 가정해서 초기 클러스터를 생성할 수 있다.

과정 3: 클러스터를 기준으로 rigid chain을 찾아내서 정점을 병합함으로써 새로운 강체와 클러스터를 생성한다. rigid chain을 더 이상 찾을 수 없는 경우에는 과정 1부터 다시 수행한다.

여기서 과정 0은 valency가 2인 간선으로 연결된 정점을 미리 병합하여 자유도가 2인 의사기하요소를 (pseudo geometric entity) 생성하는 과정이다. 이때 의사기하요소는 일반적인 기하요소와 동일하게 생각하면 된다.

Fig. 9는 "사각통 내에 접하는 막대" 문제의 construction plan생성과정을 나타낸 것이다.

Fig. 9의 step (0)을 보면 과정 0의 전처리 과정을

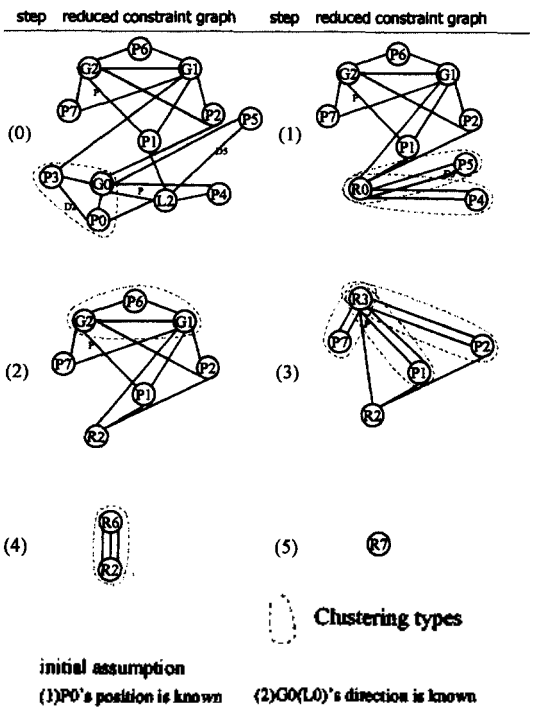


Fig. 9. Construction plan generation of "A bar lies in a box" problem (Fig. 4).

거치면서 의사 기하요소인 G0, G1, G2가 생성되었으며 P0, G0, P3로 초기강체 R0를 형성하고 있다. Fig. 9의 step (1)에서는 Fig. 8의 clustering type들을 이용하여 정점을 병합하고 있다. 이와 같이 초기강체를 형성하는 과정과 clustering type들을 찾아내서 하나의 정점으로 병합하는 과정을 반복하면 step (4)의 그래프를 얻을 수 있다. step (4)을 살펴보면 두 개의 강체가 세 개의 제약조건으로 연결되어 있는 ruler-and-compass non-constructible인 clustering type이 존재하므로 수치적인 계산을 수행해야 함을 알 수 있으며, step (5)과 같이 계산 순서가 모두 결정되면 제약조건 그래프가 하나의 정점으로 병합됨을 알 수 있다.

3.5.2 제약조건 해결자의 construction plan실행과정

Construction plan실행과정은 construction plan생성과정에서 결정된 순서와 방법에 따라서 기하요소의 값을 계산하는 과정이다. 기하요소의 값을 계산하는 순서는 클러스터에 정점이 병합되는 순서와 동일하며 계산방법은 기하요소와 제약조건 그리고 clustering type에 따라서 결정된다.

Fig. 10에는 Fig. 7의 construction plan생성 과정이 간략하게 요약되어 있다. 먼저 Fig. 9의 step (0)-(3)에

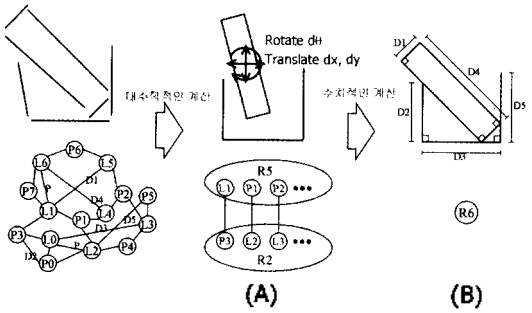


Fig. 10. Calculation process of "A bar lies in a box" problem.

서 사용된 clustering type들이 ruler-and-compass constructible이었으므로, 대수학적인 계산방법을 이용하면 Fig. 10의 (A)와 같이 사각통 모양의 강체와 막대기 모양의 강체를 얻을 수 있다. 이때 계산식은 기하요소와 제약조건의 종류에 따라서 결정된다. Fig. 9의 step (4)를 보면 강체들 사이에 제약조건이 주어져 있는 ruler-and-compass non-constructible인 clustering type이 병합되고 있다. 이러한 경우는 하나의 강체를 고정시키고 다른 강체를 병진이동과 회전이동시킴으로서 제약조건을 만족시킬 수 있다. 따라서 강체의 병진이동 값과 회전이동 값을 구하기 위한 비선형 방정식을 구성하고 이 방정식을 풀기 위한 반복계산이 수행되면 Fig. 10의 (B)와 같은 계산 결과를 얻을 수 있다.

위의 문제를 풀기 위한 비선형방정식은 참고문헌[7,8]에 자세하게 나타나 있다. 이때 비선형방정식의 해를 구하는 방법으로는 Newton-Raphson의 방법이 가장 많이 이용되고 있으며, 본 연구에서는 수치적인 계산의 안정성을 위해서 newton-raphson의 방법과 최적화를 이용한 방법을 병행하였다.

3.6 복잡한 형태를 가진 문제를 해결하기 위한 clustering type

대부분의 well-constrained인 경우 Fig. 8의 basic clustering subgraph들만을 이용하여 construction plan생성 과정을 수행할 수 있다. 그러나 복잡한 형태를 포함하고 있는 문제의 경우는 이러한 basic clustering subgraph들만으로 해결할 수 없는 경우가 존재한다. Fig. 11을 보면 기하학적 제약조건 문제가 well-constrained임에도 불구하고, 초기강체를 생성한 후에 더 이상 clustering type을 찾을 수 없다. 따라서 이러한 문제들을 해결하기 위해서 clustering type을 확장할 필요가 있다. 이러한 문제들을 일반적으로 해결하려면 clustering type을 특정한 형태로 제한해서는 해결할 수 없다. 실제로 Fig. 8의 clustering type들을 살펴보면 정점의 개수가 3개 이하인 것을 알 수 있다. 그러나 기하학적 제약조건 문제를 제약조건 그래프로 표현하면 정점의 개수가 4개 이상인 clustering type들을 필요로 하게 된다. 따라서 본 연구에서는 제약조건 그래프에서 4개 이상의 정점을 가지면서 자유도가 3이 되는 subgraph를 새로운 clustering type인 extended clustering subgraph으로 정의하여 clustering과정을 수행하였다. extended clustering subgraph는 clustering과정에서 basic clustering subgraph를 더 이상 찾을 수 없는 경우에 사용되며, 제약조건 그래프에서 두 개이상의 cycle을 찾아내서 병합함으로써 찾아낸다. Fig. 11을 보면 R1, R3, L6으로 구성된 C1과 R2, R3, L6으로 구성된 C2를 찾아내고 이 두 cycle이 가지는 정점들과 이 정점들을 연결하는 간선으로 subgraph를 구성하여 자유도가 3인 subgraph를 찾아냈음을 보여준다. 이 subgraph를 하나의 클러스터로 병합하면 clustering과정을 마칠 수 있다.

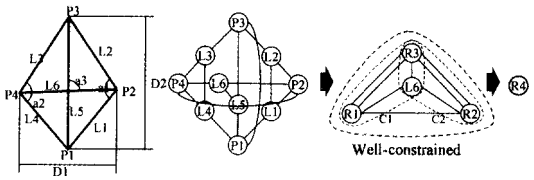


Fig. 11. An example problem that has extended clustering subgraphs and its constraint graph.

Extended clustering subgraph는 여러 개의 기하요소의 값을 동시에 계산하는 과정을 필요로 하기 때문에 수치적인 방법으로 계산된다. 수치적인 계산과정은 앞에서 설명한 ruler-and-compass non-constructible의 경우와 같다.

4. Under-constrained인 경우를 해결하기 위한 해결자의 확장

3장에서는 기하학적 제약조건 문제가 잘 정의되어 있을 때(well-constrained), 문제를 푸는 방법에 대해서 설명하였다. 그러나 매개변수를 이용한 모델링을 수행하는 과정에는 항상 under-constrained인 경우가 나타나게 된다. Under constrained문제는 기하요소들이 가지는 자유도의 합이 제약조건이 제한하는 자유도의 합보다 클 때 생기게 되는데 제약조건을 모두 입력하기 전까지는 항상 기하요소가 가진 자유도가 크므로 under-constrained문제가 된다. Fig. 12에는 이러한 예가 나타나 있다. Fig. 12를 살펴보면 모두 18개의 point들이 있으므로 36개의 자유도가 존재하고 거리 제약조건이

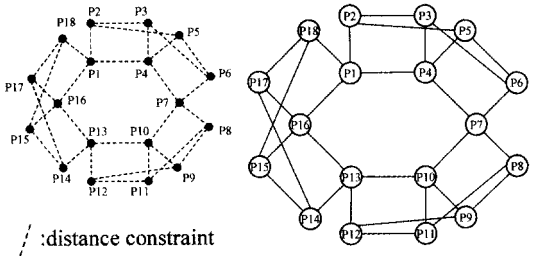


Fig. 12. An example of an under-constrained problem.

30개가 존재하므로 모두 6개의 자유도가 남게 된다. 따라서 well-constrained인 문제가 되기 위해서는 3개의 제약조건이 추가되어야 한다.

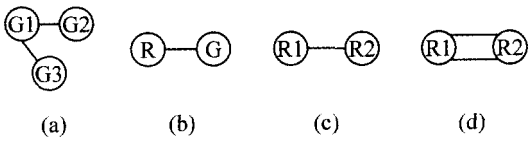
Under-constrained인 경우는 일반적으로 해가 무한하다. 따라서 사용자가 의도하는 형태를 가지도록 적절한 제약조건이나 가정들을 추가해서 해를 구해내야 한다. 이때 가장 중요한 점은 어떤 기하요소들 사이에 어떤 제약조건을 추가하느냐 하는 것이다.

본 연구에서는 위와 같은 문제를 해결하기 위해서 새로운 clustering type으로 under-constrained clustering subgraph를 정의하였으며 이것들을 이용해서 제약조건이 추가될 적절한 위치를 찾아내는 방법을 개발하였으며 이미 계산된 다른 제약조건과 충돌을 피하기 위해서 제약조건을 추가하는 대신 가정을 이용하는 방법을 개발하였다.

Under-constrained인 경우에도 well-constrained와 마찬가지로 construction plan생성과정과 construction plan 실행과정으로 구성되어 있다.

4.1 Under-constrained인 경우의 construction plan 생성과정

Under-constrained인 경우에는 자유도가 3인 basic clustering subgraph와 extended clustering subgraph만으로는 construction plan생성과정을 수행할 수 없다. 따라서 Fig. 13과 같이 basic clustering subgraph에서 하나 또는 두 개의 간선을 제거한 under-constrained



G_i : geometric entity(or pseudo geometric entity) with 2 DOFs
R_i : clusters with 3 DOFs

Fig. 13. Under-constrained clustering subgraphs.

clustering subgraph를 새로운 clustering type으로 제안하였다.

Under-constrained clustering subgraph는 다른 clustering subgraph들과 마찬가지로 다음단계에서 계산될 기하요소를 찾아내는데 쓰이게 되며 클러스터로 병합된다. 이때 under-constrained subgraph는 하나 또는 두 개의 제약조건이 부족한 상태이므로 클러스터로 병합되면서 제약조건이 추가되는 것이다. Fig. 13의 (a), (b), (d)의 경우는 하나의 제약조건이, (c)의 경우는 두 개의 제약조건이 추가된다. 그러나 실제로 under-constrained인 제약조건 그래프를 살펴보면 under-rigid clustering subgraph가 상당히 많이 존재함을 알 수 있다. 따라서 이중에서 어떤 subgraph를 선택할 것인지를 결정해야한다. 이러한 선택이 잘못되게 되면 부족한 제약조건이 모두 추가되어도 하나의 정점으로 병합할 수 없는 경우가 있다. Fig. 14에 이러한 예가 나타나 있다. Fig. 14는 Fig. 12를 under-constrained clustering subgraph를 이용하여 해결하는 과정이며 모두 6개의 자유도가 남아있기 때문에 under-constrained인 문제이다. 따라서 well-constrained인 문제가 되기 위해서는 3개의 자유도를 제한할 수 있는 제약조건들이 추가되어야 한다. 먼저 Fig. 14의 (1)을 보면 초기 강체를 생성하기 위해서 under-constrained clustering subgraph(a)를 선택하고 이를 병합한다. 이때 제약조건이 1개 추가되어 전체적으로 5개의 자유도가 남게 된다. (2)와 (3)에서는 under-constrained clustering subgraph(b)를 찾아 병합하여 다시 제약조건을 각각 1개씩 추가하여 전체적으로는 well-constrained가 되었다. 그러나 (4)에서 알 수 있듯이 제약조건 그래프에서 더 이상의 basic

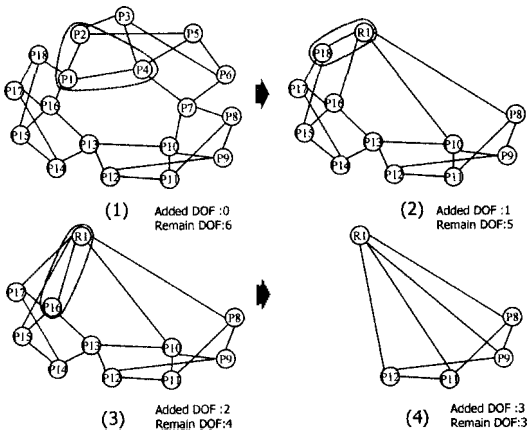


Fig. 14. An example of wrong selection of under-rigid sub graph.

clustering subgraph를 찾을 수 없으며 extended clustering subgraph를 이용해야 하므로 효율성이 떨어지게 된다.

본 연구에서는 이러한 문제점을 피하기 위해서 아래와 같이 under-constrained clustering subgraph를 선택하는 규칙을 정했다. 이를 위해서 제약조건 그래프를 cycle을 포함한 경우와 포함하지 않은 경우로 나누었으며 각각의 경우에 따라서 서로 다른 규칙을 적용하도록 했다.

A. 제약조건 그래프가 cycle을 포함하지 않은 경우

1. 제약조건 그래프에 자유도가 3인 클러스터가 없으면 under-constrained clustering subgraph(a)를 이용하여 초기강체를 형성한다.

2. 제약조건그래프에서 under-constrained clustering subgraph(b), (c), (d)를 찾아내서 하나의 클러스터로 병합한다.

B. 제약조건 그래프가 cycle을 포함한 경우

1. 제약조건 그래프의 cycle중에서 가장 적은 자유도를 가졌으며 정점의 개수가 가장 많은 cycle C를 찾아낸다.

2. cycle C에 자유도가 3인 클러스터가 없으면 under-constrained clustering subgraph(a)를 이용하여 초기강체를 형성한다.

3. cycle c에서 under-constrained clustering subgraph (b), (c), (d)를 찾아내서 하나의 클러스터로 병합한다.

이와 같이 cycle의 유무에 따라서 경우를 나눈 이유는 제약조건 그래프에 cycle이 존재하는 경우 under-constrained clustering subgraph에 의해서 병합된 기하요소가 다시 원래의 강체에 영향을 미치기 때문에 추가된 제약조건이나 가정이 다른 제약조건과 충돌할 수 있지만, 제약조건 그래프에 cycle이 존재하지 않는 경우에는 제약조건 그래프가 트리(tree) 형태가 되기 때문에 하나의 강체를 기준으로 다른 기하요소들의 값을 계산해 나가더라도 다른 제약조건들과 충돌이 일어나지 않기 때문이다. 그리고 최소의 자유도와 최대의 정점을 가지는 cycle을 찾는 이유는 앞에서 언급했듯이 부족한 자유도를 모두 추가하고도 하나의 정점으로 병합되지 않는 경우를 피하기 위해서이다.

이제 지금까지 언급한 모든 clustering type들을 모두 포함하고 위의 규칙을 적용시킨 제약조건 해결자의 전체적인 흐름은 Fig. 15와 같다.

Fig. 16은 Fig. 12의 under-constrained인 문제의 construction plan생성과정을 보여주고 있다.

Fig. 16의 step 1을 보면 최소의 자유도와 최대의 정점을 가지는 C1을 찾아내고, step에서는 C1에 강체가 없기 때문에 초기강체를 형성하여 clustering을 진행하는 과정을 보여준다. 이러한 과정을 반복적으로 수행하면 결과적으로 step 7에서 볼 수 있듯이 하나의 정점으로 병합된 제약조건 그래프를 얻을 수 있다. step 2, step 4, step 6에서 각각 하나씩의 제약조건이 추가되었으므로 모두 3개의 제약조건이 추가되었다고 생각하면 이 문제는 well-constrained가 된다.

4.2 Under-constrained인 경우의 construction plan 실행과정

Under-constrained인 경우도 well-constrained와 마찬

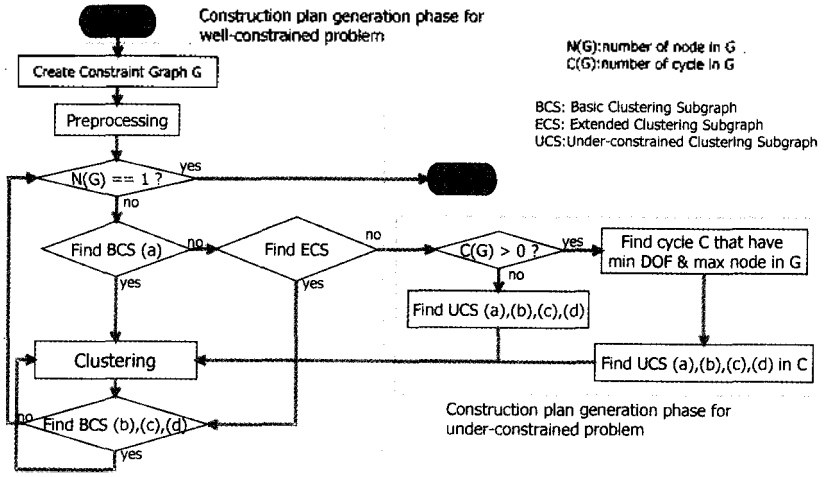


Fig. 15. Construction plan generation flow of the presented algorithm that contains the clustering procedure of the under-constrained problem.

가지로 construction plan실행과정에서 construction plan생성과정에서 결정된 계산순서에 따라서 기하요소의 값을 계산한다. 그러나 under-constrained clustering subgraph가 병합된 경우에는 기하요소의 값을 계산하기 위한 제약조건이 부족하므로 제약조건이 추가되어

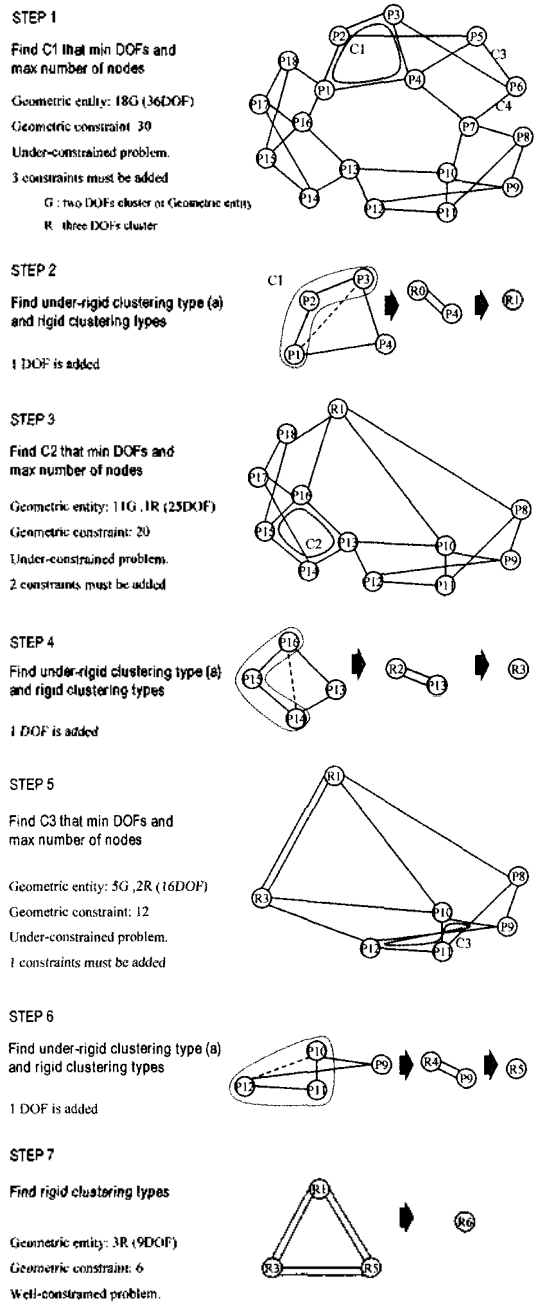


Fig. 16. An example of the construction plan generation phase of an under-constrained problem.

계산되어야 하는데, 이때 실제로 제약조건을 추가하는 경우에는 다른 제약조건들과 충돌이 발생할 위험이 있기 때문에 일반적인 경우를 처리할 때 어려움이 있다.

따라서 본 연구에서는 기하요소의 계산과정에 적절한 가정을 추가하여 계산 과정을 수행하게 된다. 이때 사용되는 가정에 따라서 기하학적 제약조건 해결자의 특성이 결정될 수 있으며, 이러한 가정들은 기존의 제약조건과 충돌하지 않는다면 어떠한 것이라도 될 수 있다. 본 연구에서는 계산을 위해 사용하는 가정들을 이미 주어진 기하요소의 형태를 최소로 변형시키면서 제약조건을 만족하도록 설정하였다. 예를 들어 점의 위치를 계산하는 경우, 정확한 점의 위치를 계산하기 위해서는 두 개의 제약조건이 필요하지만 under-constrained clustering subgraph내에서는 제약조건이 하나만 존재하게 된다. 따라서 이때에는 점의 이동거리가 최소이면서 하나의 제약조건을 만족하도록 점의 위치를 계산하게 된다. Fig. 17에는 이러한 가정으로 계산하는 예가 나타나 있다. Fig. 17은 Fig. 16의 step 2에서 병합된 under-constrained subgraph가 계산되는 과정이다. P1, P2, P3로 초기강체가 형성되는 과정에서 하나의 제약조건이 부족한 상태이다. 먼저 P1, P2를 이용하여 강체를 형성하고 나면 P3는 P2와 거리가 D1이라는 제약조건 하나만 남기 때문에, P3는 P2를 중심으로 하고 반지름이 D1인 원위의 어느 점에라도 놓일 수 있다. 여기에 P3가 현재 위치에서 최소거리를 이동한다는 가정을 추가하면 P3는 그림과 같이 하나의 위치를 결정할 수 있다.

이와 같이 실제로 제약조건을 추가하기보다 가정을 추가하여 계산을 하게되면, 추가되는 제약조건의 종류와 값을 결정하는 과정을 생략하게 되므로 효율적이고, 초기의 스케치에서 변형이 가장 적은 상태로 제약조건을 만족하게 되므로 사용자에게 편리함을 제공할 수 있다.

이와 마찬가지로 직선의 경우에는 "직선의 방향벡터가 변하지 않는다"와 같은 가정이 추가될 수 있다.

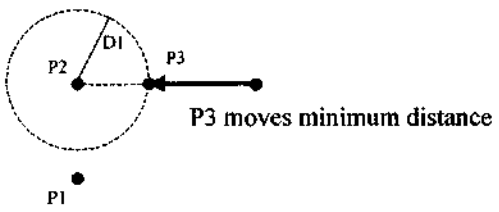


Fig. 17. An example of a construction plan evaluation phase.

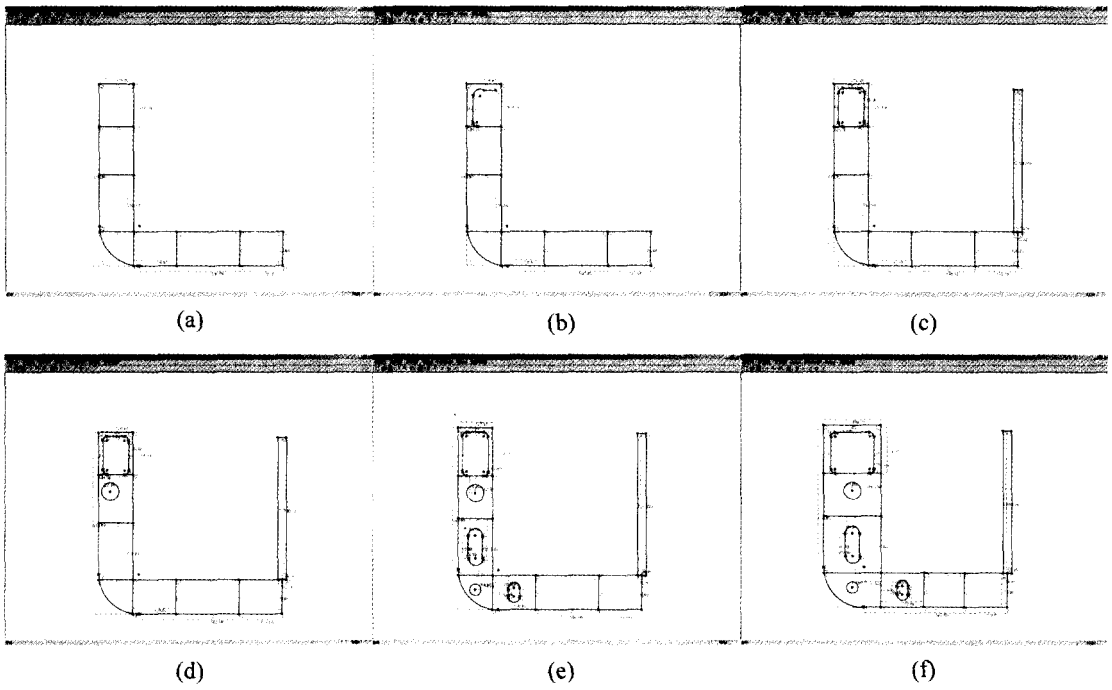


Fig. 18. A sketch procedure of a complex problem (simplified midship section of a container ship).

5. 개발된 제약조건해결자를 이용한 계산 예

Fig. 18에는 개발된 제약조건 해결자를 이용하여 기하학적 제약조건 문제를 해결한 예가 나타나 있다. 사용된 예는 간략화된 컨테이너 선의 중앙단면과 구조부재들로서 모델링을 수행하는 과정이 나타나 있다. Fig. 18의 (a)는 선체의 형상과 탱크의 격벽들을 배치하는 과정이며, (b)-(e)는 선체 중앙단면에 부재들을 배치하는 과정이다. (f)에서는 완성된 설계 결과에 제약조건의 값을 바꿈으로서 설계변경에 쉽게 대응할 수 있음을 보여주고 있다. 이 모델은 모두 106개의 기하요소와 173개의 제약조건으로 이루어져 있으며, ruler-and-compass nonconstructible인 경우와 extended clustering subgraph가 나타나기 때문에 본 연구에서 개발한 새로운 방법들로 계산하고 있다. 또 그림에서도 알 수 있듯이 모델링을 수행하는 과정에서는 항상 under-constrained인 경우가 나타나게 되는데 (b), (c), (d)가 이러한 경우이다. 이러한 경우는 under-constrained subgraph를 이용하여 주어진 제약조건을 만족하도록 계산을 수행하고 있음을 알 수 있다.

6. 결 언

본 연구에서는 그래프 정점의 병합을 이용하여 매개변수을 이용한 모델링방법의 핵심요소중의 하나인 기하학적 제약조건 해결자를 개발하였다. 개발된 제약조건 해결자는 constructive approach를 이용하여 기하요소가 계산되는 순서를 결정하며, 대수학적인 방법으로 계산하는 경우와 수치적인 계산을 이용하는 경우로 나누어 계산을 수행한다. 그리고 기존의 clustering type으로는 해결하지 못하는 문제를 해결하기 위해서 새로운 clustering type을 제안하였으며, under-constrained인 경우는 기하요소의 값을 계산하기 위한 충분한 정보가 주어져 있지 않은 상태이므로 이미 주어진 기하학적 제약조건을 위배하지 않도록 적절한 가정을 추가하여 기하요소의 값을 계산하였다. 그리고 개발된 기하학적 제약조건 해결자를 이용하여 컨테이너선의 간략화된 중앙선체와 구조부재들을 모델링하는데 적용하여 보았다. 개발된 제약조건 해결자는 모델링 중간에 나타나는 under-constrained문제들을 주어진 제약조건만으로 풀어내어 효율적인 모델링과정을 지원함을 알 수 있었다. 향후 under-constrained인 경우의 계산방법과 3차원상의 제약조건문제를 해결하는 방법을 보강하여 특징형상과 매개변수를 이용한 조선 CAD시스템에 적용해볼 계획이다.

참고문헌

1. Kramer, G.A., *Solving Geometric Constraint Systems: A Case study in Kinematics*, The MIT Press, 1992.
2. Hsu, C. and Bruederlin, B.D., A hybrid constraint solver using exact and iterative geometric constraints, *CAD Systems Developments: Tools and Methods. Roller and Brunet (eds.)*, Springer, pp. 265-279, 1997.
3. Lamure, H. and Michelucci, D., "Solving geometric constraints by homotopy," *IEEE Transactions on Visualization and Computer Graphics*, Vol. 2, No. 1, pp. 28-34, 1992.
4. Ge, J.X., Chou, S.C. and Gao, X.S., "Geometric constraint satisfaction using optimization methods," *Computer-Aided Design*, Vol. 31, No. 14, pp. 867-879, 1999.
5. Owen, J.C., "Algebraic solution for geometry from dimensional constraints," *Proc. 1st Symp. Solid modelling foundations & CAD/CAM Applications*, ACM Press, pp. 379-407, 1991.
6. Bouma, W., Fudos, I., Hoffmann, C.M., Cai, J. and Paige, R., "Geometric constraint solver," *Computer-Aided Design*, Vol. 27, No. 6, pp. 487-501, 1995.
7. 이재열, 김광수, "A Geometric Constraint Solver for Parametric Modeling," 한국 CAD/CAM학회논문집, 제3권, 제4호, pp. 211-222, 1998.
8. Lee, J.Y. and Kim, K., "A 2-D geometric constraint solver using DOF-based graph reduction," *Computer-Aided Design*, Vol. 30, No. 11, pp. 883-896, 1998.
9. Fudos, I. and Hoffmann, C.M., "A graph-constructive approach to solving systems of geometric constraints," *ACM Transactions on Graphics*, Vol. 16, No. 2, pp. 179-216, 1997.
10. Solano, L. and Brunet, P., "Constructive constraint-based model for parametric CAD systems," *Computer-Aided Design*, Vol. 26, No. 8, pp. 614-622, 1994.
11. Chou, C.-S., Mechanical Theorem Proving, Kluwer, Netherlands, 1987.
12. Ait-Aoudia, S., Hamid, B., Moussaoui, A. and Saadi, T., "Solving geometric constraints by a graph-constructive approach," Information Visualization, 1999. Proc. 1999 IEEE International Conference on, pp. 250-255, 1999.
13. "Examples of ruler-and-compass non-constructible configurations," http://www.d-cubed.co.uk, Home page of D-Cubed Ltd.
14. Aldefeld, B., "Variation of geometries based on a geometric resoning method," *Computer-Aided Design*, Vol. 20, No. 3, pp. 117-126, 1988.

권 오 환

1998년 서울대학교 조선해양공학과 학사
2000년 서울대학교 조선해양공학과 석사
2000년~현재 서울대학교 조선해양공학과 박사과정
관심분야: Feature based design, parametric design, geometric solving

이 규 열

1971년 서울대학교 공과대학 조선공학과 학사
1975년 독일 하노버 공과대학 조선공학 석사
1982년 독일 하노버 공과대학 조선공학 박사
1975년~1983년 독일 하노버 공과대학 선박설계 및 이론연구소 주정부 연구원
1983년~1994년 한국기계연구원 선박해양공학연구센터, 선박설계, 생산자동화 연구사업(CSDP)단장
1994년~2000년 서울대학교 공과대학 조선해양공학과 부교수
2000년~현재 서울대학교 공과대학 조선해양공학과 교수
관심분야: 최적설계, 형상모델링, CALS

이 재 열

1992년 포항대학교 산업공학과 학사
1994년 포항대학교 산업공학과 석사
1998년 포항대학교 산업공학과 박사
1998년~현재 한국전자통신연구원(ETRI) 컴퓨터·소프트웨어 기술연구소 동시공학연구팀 선임연구원
관심분야: network-centric CAD, feature-based modeling, parametric design, virtual prototyping, assembly modeling