韓國 컴퓨터情報學會 論文誌
第13卷 第2號, 2008. 3.

2008-13-2-1-2

OpenMP 병렬프로그램을 이용한 그물의 수중형상 시뮬레이션 구현

박 명 철*, 박 석 규**

Implementation of Underwater Simulation of a Net using OpenMP

Myeong-Chul Park*, Seok-Gyu Park**

요 약

수중에서 그물의 형상변화는 다양한 벡터에 의해 영향을 받게 된다. 그러나 그물의 각 입자마다 모든 벡터의 영향을 계산하는 것은 정확성과 사실성은 증대하지만, 방대한 계산량으로 처리 시간이 많이 소요된다. 기존의 시뮬레이션 방법들은 물리적 사실성을 회생하고 시각적인 사실성을 유지하는 범위에서 수중 가상현실을 시뮬레이션으로 구현하였다. 본 논문에서는 입자들의 병렬처리를 통하여 물리적, 시각적 사실성을 모두 만족하는 시뮬레이션을 제안한다. 병렬처리를 위해서는 OpenMP를 이용하였고, 사실적 그래픽 표현은 OpenGL을 사용하여 구현하였다. 본 논문에서 구현한 시뮬레이션은 게임 및 해양수산 분야에서 모델분석이나 전문가 시스템구축을 위한 기초자료로 활용될 수 있을 것이다.

Abstract

The net shape effects by the various vectors in underwater. Each particle of the net calculating the effect of all vectors augments an accuracy and reality. But, the time complexity becomes larger because of huge calculation. The previous techniques reduced a physics reality. And embodied the underwater virtual reality which augments visual reality with simulation. In this paper, parallel processing the particles, it embodied the simulation which is satisfied a physical reality and time reality. The parallel processing used the OpenMP, and the reality graphic expression used the OpenGL. The simulation which this paper proposes will be the possibility becoming the fundamental data for a model analysis or a specialist system from game and marine field.

▸ Keyword : OpenMP, Simulation, Parallel Processing, OpenGL

• 제1저자 : 박명철
• 접수일 : 2008. 1. 16, 심사일 : 2008. 2. 13, 심사완료일 : 2008. 3. 8.
* 송호대학 컴퓨터정보과 전임강사 **강원도립대학 컴퓨터응용과 부교수

Ⅰ. 서 론

수중에서 그물의 형상변화는 어획성능을 결정하는 중요한 정보로 이용될 수 있다. 관련연구 결과[1]에 따르면 일반적인 어획 성공률은 50%정도로 보고되고 있다. 이러한 성공률을 높이기 위하여 시뮬레이션 기법을 통한 다양한 연구가 이루어지고 있다 [2]. 그러나, 입자시스템을 이용하는 시뮬레이션분야에서 각 입자의 연산은 그 양이 매우 방대하므로 시간적 복잡도가 매우 크다는 문제점을 가진다. 물리적 사실성과 정확성을 희생하고 시각적 사실성을 부각시킬 수 있지만, 이는 시뮬레이션 본연의 요구사항에서 멀어지는 문제점을 가진다. 특히, 수중현상을 위한 시뮬레이션에서는 변화무쌍한 환경요인이 현장 계측을 어렵게 하고 비선형적인 결과를 초래하므로 실제 환경과 불일치성을 가지는 특징이 있다. 이러한 문제를 극복하기 위하여 최근에는 비선형적으로 복잡하고 혼돈스러운 수중현상을 연구하는 비선형과학 분야에서 컴퓨터 모델링과 시뮬레이션 기법을 이용하여 해석하고 재현하는 시도들이 다양하게 이루어지고 있다 [2,3,4]. 수중의 어구에는 다양한 벡터들(유체저항, 침강력, 어선의 예인력 또는 죔장력, 바람 및 파도 등)이 각각의 입자에 영향을 주게 되는데, 이러한 합성벡터를 실시간으로 계산하여 시뮬레이션 하는 것은 시간적 복잡도를 한층 증가시키므로 실시간 표현에 많은 어려움을 가진다.

본 논문에서 그물의 각 입자들에 주어지는 합성벡터들을 모두 적용하면서 시각적 사실성도 동시에 만족할 수 있는 시뮬레이션을 구현한다. 각 입자시스템은 공유메모리 기반의 병렬프로그램(OpenMP)[5,6]을 통하여 병행적으로 처리함으로써 시간적 부담감을 크게 줄일 수 있다. 각각의 입자들이 상호 유기적으로 영향을 미치므로 병행성 있는 계산 구조를 만들기 위해서는 먼저 입자들의 병행연산모델을 구축하였다. 구축된 모델을 통하여 그물의 각 입자는 최적의 연산시간을 가지며, 그 결과를 통하여 실시간으로 화면상에 표현된다. 이때, 사용자 인터페이스를 통하여 어선의 위치변화, 수중의 해류 방향과 속도를 등의 벡터를 조정할 수 있게 하여 사실성을 한층 높였다. 구현된 시뮬레이션은 병렬처리없이 수행되는 기존 시뮬레이션에 비해 3배 이상의 연산시간을 줄임으로서 사용자에게 정확성과 사실성을 동시에 만족시킬 수 있었다. 이는 그물의 투망 경과시간에 따른 형상 변화를 예측할 수 있으므로 어군의 도피방지에 적용할 수 있어 어획량 증대에 기여할 수 있다.

본 논문의 2장에서는 시뮬레이션을 위한 기본 모델을 기술하고, 3장에서는 실제 구현 과정에 대해 기술한다. 4장에서는 시뮬레이션 결과를 평가하고, 마지막 5장에서 결론과 향후 연구 과제를 기술한다.

Ⅱ. 모델구축

2.1 병렬처리를 위한 OpenMP 모델

현재 사용되고 있는 병렬 라이브러리는 공유 메모리형 병렬 지원 라이브러리인 OpenMP와 분산 메모리형 병렬 지원 라이브러리인 MPI[7,8]가 광범위하게 사용되고 있다. 이중에서 1997년에 산업표준으로 채택된 OpenMP는 공유메모리와 분산된 공유메모리를 기반으로 하는 다중프로세서를 위한 병렬 프로그래밍 모델이다. 이 프로그램은 실리콘 그래픽사에 의해 개발되기 시작하여 다른 여러 컴퓨터 벤더(Compaq, Intel, IBM, SUN)들이 참여하여 개발하였다. 표준 C/C++를 확장하는 병렬 디렉티브와 라이브러리들의 집합 및 SPMD(Single Program Multiple Data) 수행 환경을 명시하고 있다. 디렉티브는 OpenMP를 지원하는 컴파일러에서 인식하는 API이다. OpenMP 프로그램은 원시 프로그램의 수정없이 OpenMP에 명시되어 있는 디렉티브를 삽입함으로써 쉽게 병렬 프로그램화를 할 수 있는 장점이 있다. <그림 2-1>은 그물의 각 입자들을 초기화하는 병렬처리 구조이다. 여기서, *#pragma omp parallel for shared* 디렉티브가 해당 변수를 병렬적으로 접근한다는 의미로 해석되어 번역된다.

```
void SetupRC()
{
int i;
glEnable(GL_DEPTH_TEST);
#pragma omp parallel for shared(g_part)
for (i=0 ; i<MAX_PARTICLES ; i++) {
    g_part[i].top.x = 0.0f;
    g_part[i].top.y = 0.0f;
    g_part[i].top.z = 0.0f;
    g_part[i].bottom.x = 0.0f;
    g_part[i].bottom.y = 0.0f;
    g_part[i].bottom.z = 0.0f;
    g_part[i].active = false;
    g_part[i].start_y = -100.0f;
    g_part[i].angle = 0.0f;
}
```

그림 2-1. OpenMP 프로그램의 예
Fig. 2-1 Example of OpenMP Program

2.2 수중형상 시뮬레이션 모델

〈그림 2-2〉에서 보이는 것처럼, 그물의 3차원 좌표 값은 그물의 부력, 침강력(Sinking), 죔장력(Pursing), 어선의 뜸줄장력(Pulling), 해수의 흐름에 따른 저항력 등으로 평가되어 모델링 된다.

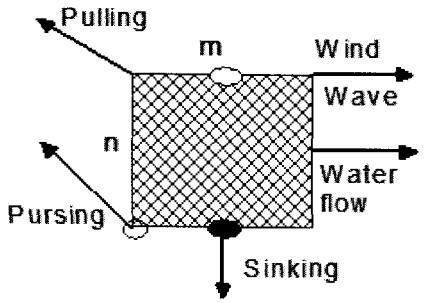


그림 2-2. 그물에 미치는 주요 영향 벡터
Fig. 2-2 The schematic diagram of the main factors acting on a net strip of purse seine

그물은 크기는 수평 길이 2000m, 수직 깊이 160m 이내에서 변경 가능하게 설계하였다. 기본적으로 수평은 500개의 입자로 나누고 수직은 40개의 입자로 나누어 그물의 단위 판넬은 4*4 m로 구성되어 진다. 물론, 물리적 사실성을 증대시키기 위하여 다수의 입자로 변화시킬 수도 있다. 그물이 침강하는 깊이를 Y_N이라 하고, 전체 침강력을 F_S라 할 때, 침강시간 T_Y는 아래 관계식으로 나타낼 수 있다. 이 때, F_S는 1m당 작용하는 힘으로 침강을 위한 끌줄, 그물의 하단의 루프와 링 무게를 의미한다.

$$T_Y = 0.9\ Y_N(Y_N/F_S)^{1/2} \quad \cdots\cdots (1)$$

$$Y_Y = (F_S\ T_Y^2/0.81)^{1/3} \quad \cdots\cdots (2)$$

여기서, F_S=15kg/m 정도이며, Y_N은 최대 160m이다.

그물줄 길이를 l, 그물 직경을 d, 그물판넬 넓이 S, 해수의 속도 V_W, 해수의 방향 a, 저항계수 C_R, 해수의 밀도 ρ에 의한 해수의 저항력 F_W 은 다음과 같이 나타낼 수 있다.

$$F_W = C_R\ \rho(d/l)S\ V_W^2 sin(a)/2 \quad \cdots\cdots (3)$$

여기서, C_R=1.0 이고 ρ=1.05 이다.

그물을 오므리거나 끌 때, 수평 m번째 노드의 그물저항 R_M과 끄는 힘과 오므리는 장력 F_P는 그물의 움직이는 속도 V_N에 대해 다음과 같은 관계식으로 나타낼 수 있다.

$$R_M = m\ C_R\ \rho\ S\ V_N^2/2 \quad \cdots\cdots (4)$$

$$F_P = k\ R_M\ e^{-\mu\theta}/(2\pi) \quad \cdots\cdots (5)$$

여기서, θ는 그물을 이등분했을 때의 수평각도이고, μ는 그물 하단을 당길 때의 고리와 로프의 마찰계수이다. 그리고 오므리는 속도와 오므리는 장력은 계수 값을 통하여 제안할 수 있어야 한다. 또한, F_P와 어선의 해수저항에 의해 어선은 그물의 안쪽으로 이동하게 된다. 바람과 파도에 따른 정도값 F_F는 수직 첫 번째 노드의 경우(n=1), 식(3)과 유사하게 영향을 끼치는 것으로 하였다. 위에 언급한 네 가지 장력으로 저항력 벡터 F_R을 아래와 같이 나타낼 수 있다.

$$F_R = F_S + F_W + F_P + F_F \quad \cdots\cdots (6)$$

결과로서 생기는 힘 벡터에의 그물노드 $ds(m,n)$의 힘은 그 크기에 비례하고 다음과 같이 지수함수로 바꾸었다.

$$ds(m,n) = f(dx,dy,dz) = \lambda F_R\ e^{-\beta(m,n)} \quad \cdots\cdots (7)$$

여기서, λ=0.01m/kg 이고 β=0.2 로 가정한다.

ds의 벡터에 의해 그물의 이음매들의 3차원 좌표들은 수평과 수직의 주변 이음매들과 함께 계산될 수 있다.

유한요소법에 따라 어구의 길이방향과 깊이방향으로 유한요소로 등분하여 각 노드별 분력에 따른 좌표의 이동이 시간에 따라 힘의 균형을 이루도록 하여 어구형상을 추정하였다. 각 부분별로 침강저항, 유수중 유체저항, 그물양끝을 조울 때 뜸줄장력, 죔줄장력, 바람에 의한 표면 유동, 어선의 이동 등에 따른 부분망지의 유체저항의 합성벡터에 지수 함수적으로 비례하여 각 부분망지의 좌표변위를 1초단위로 추산하였다. 해류 등의 유향유속은 전 방향에서 최고 3k't까지, 투망속도는 12k't까지로 투망방향은 자유로이 조정할 수 있도록 하였다. 〈그림 2-3〉은 수치모델 중 그물의 이동을 위한 병렬처리의 한 부분이다.

```
1:  void Move_Net(void)
2:  {
3:      float m_x,m_bx;
4:      double x1,y1,z1,x2,z2;
5:      int i,j;
6:      #pragma omp parallel for shared(g_part)
7:      for(i=0; i<MAX_PARTICLES;i++){
8:          if(g_part[i].active){
9:          if(g_part[i].power_Vx==0 && g_part[i].power_Vz==0){
10:         m_x = pow(Vw1,1.8f)*(1-sin(g_part[i].angle));
11:         m_bx =pow(Vw2,1.8f)*(1-cos(g_part[i].angle));
12:         g_part[i].power_Vx =
                m_x*pow((1+sin(g_part[i].angle)),2.0f);
13:         g_part[i].power_Vxb =
                m_bx*pow((1+cos(g_part[i].angle)),2.0f);
14:         }
15:         g_part[i].top.x -=
                g_part[i].power_Vx*1./(1.+exp(2.*PI-8.*PI*i/H_n));
16:         g_part[i].bottom.z -=
                g_part[i].power_Vxb*1./(1.+exp(2.*PI-8.*PI*i/H_n));
17:         if(i==MAX_PARTICLES-1) x_1 = g_part[i].top.x;
18:         #pragma omp parallel for shared(g_part)
19:         for(j=0;j<=Mesh_D/2;j++)
20:             if(g_part[i].mid[j].active == true) {
21:               x2=g_part[i].mid[j].x;
22:               x2 -=
                    g_part[i].power_Vx*1./(1.+exp(2.*PI-8.*PI*j/H_n));
23:               z2=g_part[i].mid[j].z;
24:               z2 -=
                    g_part[i].power_Vxb*1./(1.+exp(2.*PI-8.*PI*j/H_n));
25:             }
26:             else break;
27:         }
28:       else break;
29:     }
30: }
```

그림 2-3. 그물이동을 위한 병렬처리
Fig. 2-3 Parallel processing for moving net

III. 시뮬레이션의 구현

본 논문에서 제안하는 모델의 성능을 검증하고 그 결과를 보이기 위해 윈도우즈 환경에서 C++/OpenGL[9]를 사용하여 시뮬레이터를 구현하였다. 본 어구형상모델을 가지고 여러 가지 경우를 시뮬레이션 한 결과 투망에서 부터 쵬작업 완료까지를 실시간으로 수중 형상을 3차원으로 애니메이션 할 수 있었으며, 수중흐름에 의한 처지는 형상, 쵬작업에 의한 Gate 현상, 바람과 파도 등에 의한 Together현상 등을 구현할 수 있었다. 또한 그물의 노드수도 뜸줄과 망심에 따라 화면 해상도에 비례하여 가변적으로 동작하게 된다. 먼저, 어구를 형성하는 각 노드의 3차원 위치정보를 기반하여 어구 형상을 드로잉할 수 있는 도구인 OpenGL로 구현하고, 이동, 위치변경, 크기변경, 뷰 관점 변경, 회전과 같은 고유의 기능은 C 루틴의 접목으로 구현하였다. 아울러, 사용자가 직접 어구를 조정할 수 있는 부가적인 메뉴 인터페이스를 두어, 어구 형상에 영향을 줄 수 있는 투망방향과 속도, 쵬작업 장력과 쵬 시간, 수심별 유향유속 등 요소를 변화시킬 수 있는 기능을 두었다.

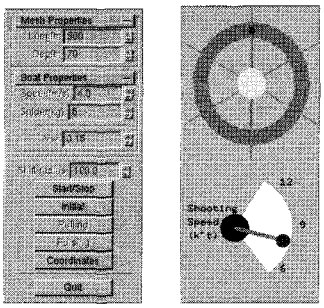
미립자 시스템은 각 입자의 속성들을 초기화하고 시간적 흐름에 따라 수치적 모델에 준하여 속성들을 갱신한다. 수중형상 변화를 위한 그물의 각 입자들은 〈그림 3-1〉과 같이 구성하였다.

```
typedef struct {
        POSITION top;
        POSITION bottom;
        bool active;
        double angle;
        POSITION mid[100];
         double power_Vx,power_Vz,power_Vxb;
} particles;
```

그림 3-1. 그물을 위한 입자시스템 속성
Fig. 3-1 Attributes of particle systems for mesh

여기서, *top*과 *bottom*은 상단의 뜸줄과 하단의 발줄의 좌표 값을 위한 구조체이고, *active*는 현재 뷰 상의 활성화 유무를 판단하기 위한 속성이다. 그리고 *angle*은 어선을 기준으로 각도값 정보를 위한 속성이고, *mid* 배열구조체는 뜸줄과 발줄 사이의 각 입자를 위한 좌표 값 속성이다. 마지막으로 *power_Vx,Vz,Vxb*는 해당 입자에 가해지는 환경벡터값을 위한 속성이다.

〈그림 3-2〉는 사용자를 위한 인터페이스 도구를 보이고 있다.



(a) (b)

그림 3-2. 어구 조절을 위한 사용자 인터페이스
Fig. 3-2 User interface for simulation

〈그림 3-2〉의 (a)는 그물과 어선에 대한 정보를 조절할 수 있는 인터페이스 화면이고, (b)는 어선의 방향과 속도를

사용자가 조정할 수 있는 일종의 조정 장치이다. 이는 현실감있는 시뮬레이션을 구현하기 위한 도구로서 고정된 환경의 정형화된 벡터만으로는 사용자의 요구사항을 만족 시킬 수 없기 때문에 다양한 입력 벡터를 조정을 통하여 사용자의 만족감을 극대화 시키는 역할을 한다.

3.1 수중의 환경벡터를 적용하지 않은 예: 모형1

먼저, 수중의 해류방향과 해류속도등의 환경벡터를 적용하지 않은 그물의 형상변화를 〈그림 3-3〉에서 보이고 있다.

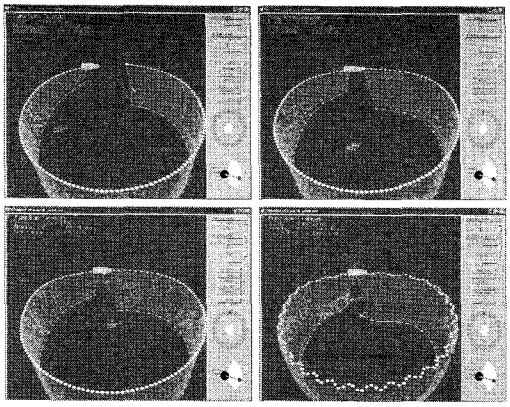


그림 3-3. 저항벡터를 적용 하지 않은 결과: 모형1
Fig. 3-3 The result of simulation without resistance vector

〈그림 3-3〉에서 보는 것과 같이 수중의 환경벡터를 적용하지 않았을 때는 그물의 투망형상이 정적인 형태를 가지고 있어 시각적 사실성이 매우 떨어지는 것을 확인할 수 있다. 이는 현장계측을 위한 실용적 모델로 사용되기에 어렵다. 실제 어획과정에서도 그물을 정방형의 일정한 모습으로 투망하지 않고 어군의 위치 변화에 따라 동적으로 투망한다.

3.1 수중의 환경벡터를 적용한 예: 모형2

본 논문에서 제안하는 환경벡터를 적용한 수중의 그물형상 변화는 〈그림 3-4〉와 〈그림 3-5〉에서 보이고 있다. 〈그림 3-4〉는 상단의 해류속도가 0.1m/s(V_W1)이고 하단의 해류속도가 0.3m/s((V_W2)일 때의 시뮬레이션 결과이다. 〈그림 3-3〉과 비교해 볼 때 측면의 형상변화가 해류의 영향에 따라 변화한 것을 확인할 수 있다. 어선에서 당기는 장력을 크게 하거나 침강력을 크게 하면 더 늘어지는 현상을 볼 수 있었다. 〈그림 3-5〉는 해류의 속도를 더 크게 했을 때의 형상변화를 보이고 있다. 최종 죔이 완료되었을 때 각 시뮬레이션의 결과를 다름을 확인할 수 있다.

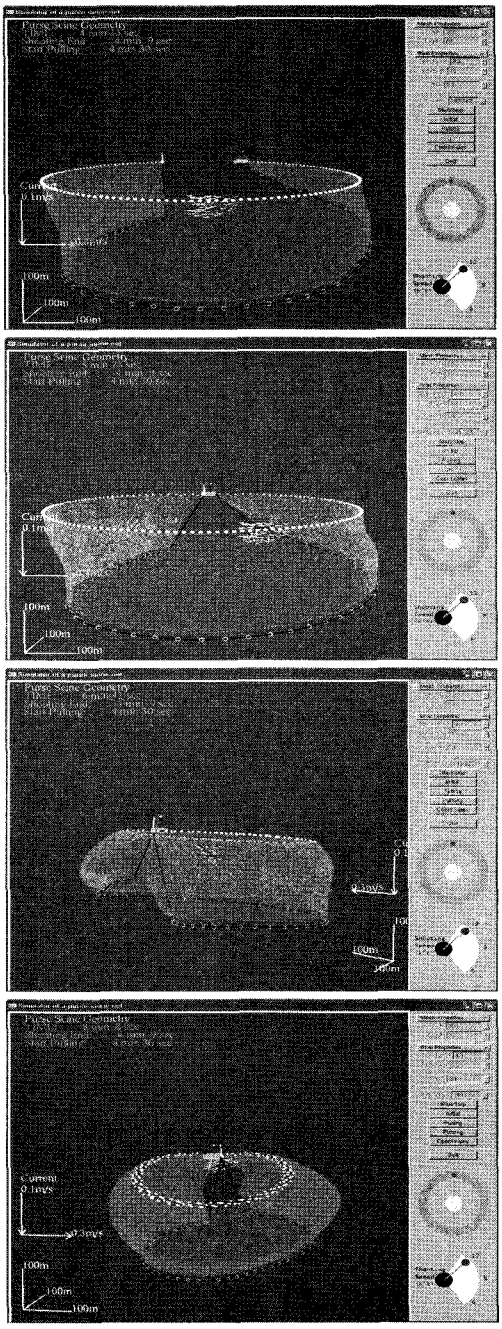


그림 3-4. 저항벡터를 적용한 결과: 모형2
Fig. 3-4 The result which applies the resistance vector (VW1=0.1, VW2=0.3)

또한, 그물의 무게에 의해 어선이 그물의 안쪽으로 견인되는 현상도 실제 계측사진과 유사함을 확인할 수 있었다. 사용자의 이해도를 높이기 위하여 다양한 각도에서 결과를 볼 수 있도록 방향키를 통하여 자유자재로 뷰 시점을 변화할 수 있게 구현하였다.

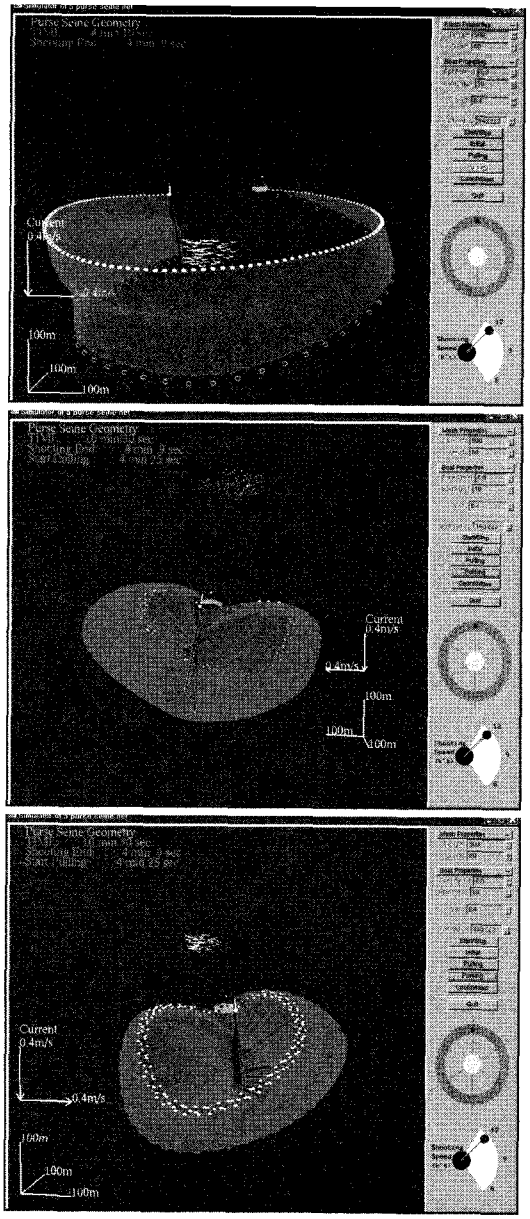


그림 3-5. 저항벡터 적용한 결과: 모형3
Fig. 3-5 The result which applies the resistance vector (VW1=0.4, VW2=0.4)

Ⅳ. 결과분석 및 평가

병렬처리의 유용성을 분석하기 위하여 펜티엄4 2.53GHz의 CPU와 1GB의 메모리에서 시뮬레이션에 따른 완료시간을 측정한 결과를 〈표 2〉에 나타내었다. 측정에 사용된 모형은 〈표 1〉과 같이 세 가지 모델을 이용하였다.

표 1. 분석을 위한 모델
Table 1. Analysis for model

Type / 요소	A	B	C
뜸줄길이(수평)	2000m	2000m	2000m
망심(수직)	70m	80m	80m
뜸줄입자	300	600	900
망심입자	30	40	40
VW1	0.1	0.4	0.4
VW2	0.3	0.3	0.4
어선의 속도	12k't	12k't	12k't
어선의 장력	5000kg	5000kg	5000kg
침강력	10kg	10kg	15kg

시간적 복잡도는 계산량을 증가시키는 주요소인 뜸줄과 망심의 입자수에 비례하기 때문에 길이가 긴 뜸줄의 입자수를 300, 600, 900으로 급격한 차이를 두었다. 이는 병렬처리의 효율성을 입증할 수 있는 주요소가 된다. 어선의 속도나 그물을 끄는 장력은 각 입자의 계산수를 줄일 수 있는 요인이 되기 때문에 비교분석을 위해 동일한 값으로 측정모델을 구성하였다. 어선이 그물을 당길 때 각 입자의 좌표값이 어선의 좌표에 도달하면 입자는 연산 루틴에서 소멸되는 것으로 간주한다. 여기서, 어선의 속도와 장력은 각 입자의 이동거리를 결정하기 때문에 서로 다른 값으로 측정했을 때 실제 입자들의 계산량에 영향을 미칠 수 있기 때문에 동일 값으로 측정하였다. 측정 모델의 뜸줄과 망심의 길에 대한 실제 어획과정에서 그물을 투망하는 시간은 일반적으로 10분에서 15분 내외라고 알려져 있다. 시뮬레이션에서 측정시간은 다른 요소에 비해 뜸줄의 입자수에 의해 가장 크게 영향을 받는다. 뜸줄의 입자수가 600일 때 실제 투망과 유사한 경과시간을 보였고 시각적 효과도 매우 우수함을 보였다. 측정에서는 병렬성의 효율성을 입증하기 위하여 900까지

입자수를 늘려 측정하였다. 실제 투망과 유사한 모델의 요소값은 Tpye B가 가장 알맞은 모델이다.

표 2. 성능비교분석(Time)
Table 2. Analysis of performance comparison

타입 \ 수행방법	순차처리	병렬처리	비율
Type A	6분 12초	4분 5초	65.8%
Type B	26분 31초	10분 12초	38.4%
Type C	85분 2초	32분 12초	37.9%

〈표 2〉의 측정결과에서 보듯이 각 타입의 병렬처리시 동일조건의 순차처리에 비해 매우 효과적인 시간 복잡도를 갖는 것으로 확인되었다. Type B의 경우, 실제 현장에서의 투망시간과 유사한 수행시간을 가짐으로서 물리적 사실성도 만족함을 알 수 있다. 그물의 수중형상도 사실성 있게 표현됨으로 연구의 목적에서 언급한 물리적 사실성과 시각적 사실성을 모두 만족하는 시뮬레이터라는 것을 증명한다.

V. 결론 및 향후연구

본 논문은 수중환경에서 어구의 형상 변화를 사실성 있게 보여주기 위하여 다양한 환경벡터를 병렬처리하여 사실성을 부각시키는 시뮬레이터를 제안하였다. 논문의 두 가지 목적은 물리적 사실성과 시각적 사실성을 모두 만족하는 시뮬레이터를 구현하는 것 이었다. 물리적 사실성을 높이기 위하여 어구의 각 입자에 다양한 환경벡터를 적용하여 병렬처리함으로서 시각적 사실성에 영향을 주지 않고 시뮬레이터를 구현할 수 있었다. 본 연구 결과는 해양수산 분야의 분석 자료로서 활용될 수 있을 것이며, 교육용 자료로서도 이용될 수 있다. 지금은 선망이라는 제안된 어구에만 적용하였지만, 향후 다양한 어구에도 적용할 수 있는 보다 확장성 있는 시뮬레이터 개발에 대한 연구를 지속할 것이다.

참고문헌

[1] Kim, Y-H., Huh, T. Sinking movements and catch of the tuna purse seine off Southeast Guam. Bull. Tong-yeong Fish. Jr. Coll., 22, 1-6, 1987.

[2] 박명철, 김용해, 하석운, 혼돈이론을 응용한 예망어구에 대한 어류반응 행동모델의 수중현상 시각화, 한국해양정보통신학회논문지, 1226-6981, 제8권3호, pp.645-653, 2004.

[3] F.C.Hoppensteadt, "Analysis and simulation of chaotic systems," Spriger-Verlag, p.297, 1993.

[4] K.Matuda, and N.Sannomiya, "Computer simulation of fish behaviour in relation to fishing gear," Bull. Japanes Soc. Sci. Fish., 46(6) : pp.689-697, 1980.

[5] Kuhn, B., P. Petersen, and E. O'Toole, "OpenMP versus Threading in C/C++," EWOMP '99, Lund, Sweden, Sept. 1999.

[6] Netzer R. H. B., and B. P. Miller, "What are Race Condition? Some Issues and Formalization," Letters on Programming Lang. and System, 1(1):74-88, ACM, 1992.

[7] Message Passing Interface Forum. MPI: A Message-Passing Interface standard, version 1.1. http://www.mpi-forum.org/docs/, 1995.

[8] Message Passing Interface Forum. MPI-2: Extensions to the Message-Passing Interface. http://www.mpi-forum.org/docs/, 1997.

[9] Wright, R.S. & Sweet, M. OpenGL SuperBible, Eaite Group Press, 2000.

저 자 소 개

박 명 철

2007년 : 경상대학교 컴퓨터과학과 공학박사
2007년 ~ 현재 : 송호대학 전임강사
관심분야 : 시뮬레이션, 임베디드 소프트웨어, 병렬프로그래밍 및 디버깅

박 석 규

2005년 : 경상대학교 컴퓨터과학과 공학박사
2001년 ~ 현재 : 강원도립대학 부교수
관심분야 : 소프트웨어 신뢰성, 시스템분석, 멀티미디어