2003년도 대한전기학회 하계학술대회 논문집 2003.7.21-23

인간형 로봇의 동작 테스트를 위한 그래픽 시뮬레이터

황 병훈 김 지홍
전북大學校 제어계측 工學科

Graphic Simulator for processing test of Humanoid Robot

Hwang, Byunghun Kim, Jeehong
Control & Instrument Eng. of Chonbuk Nat. Univ.

Abstract - As make a simulator including user interface functions like start & stop, load parameters, record and save, view 3D display has a real-like length and numerical value of sizes, represent real-shape of inner and outer part ofrobot, make the possible fast and slow selective observation as a adjust a step, receiving the images through the image device which attached in robot, so make a motion tester simulator of humanoid robot which coded by windows based GUI(Graphic User Interface) program with a MMI(Man Machine Interface) function that user can watch the environment which included robot and use a images. For implement this, we use a design data that converted data which made by use a CAD for Laser RP(Rapid Prototyping)progress into C coding for simulator programing. Using OpenGL, an API of graphic, it has a efficiency and detail of graphic operation. To make and test animation data, it has the option of save and resume in animation.

1. 서 론

시뮬레이션이란 복잡한 문제를 해석하기 위하여 모델에 의한 실험, 또는 사회현상 등을 해결하는 데서 실제와 비슷한 상태를 수식 등으로 만들어 모의적(模擬的)으로 연산(演算)을 되풀이하여 그 특성을 파악하는 일로 정의되어 있다. 즉 실제 또는 가상의 동적 시스템 모형을 컴퓨터를 사용하여 연구하는 것을 말하며 모의실험 또는 모사(模寫)라고도 한다. 수학적 해답을 얻기 위해서는 수리 모델의 기초가 되는 가정이 현실에 비하여 너무 추상적이어서 실제 상태를 충분히 모델에 반영할 수 없게 되는 경우도 있다. 본 연구에서 2족 보행로봇 시뮬레이터를 제작함에 있어 실제 치수와 똑같은 데이터와 모양을 갖춘 그래픽 모델을 이용하여 Virtual Prototype을 제작하고 기구학적 해석을 통한 각 링크에 움직임이 실제와 유사하도록 만든다. 또한 로봇 개발자가 필요시 동작을 확인하고 저장할 수 있게 하며 임의의 동작을 반복하고 빠르게 혹은 느리게 움직이도록 할 수 있게 함으로서 관찰에 유용성을 높이고 동작중 그래픽 화면에 회전과 확대 축소를 통해 개발자가 보기를 원하는 관찰지점을 제공하고 세밀한 관찰을 가능하게 한다. 그래픽 API인 OpenGL은 이런 점에서 효율적이다. 이렇게 만들어진 시뮬레이터에 의한 실험은 실제 제작에 앞서 동작의 유용성을 먼저 알 수 있고 오동작에 의한 파손을 방지할 수 있다는 점에서 아주 중요한 역할을 하고 있다. 더불어 새로운 동작을 생성할 수 있도록 함으로서 보행 중에 상체의 동작을 생성 가능하도록 한다.

2. 본 론

2.1 시뮬레이터의 제작

우리가 로봇을 구동하는 데 있어서 그 보행 혹은 움직임 동작 확인 과정으로서, 사전에 계획된 수리적 모델에 의해 계산된 각 링크의 모션 명령에 따른 로봇의 동작을 그래픽 애니메이션을 통해 사전 검증하거나 각 부분의 부품들에 연결된 움직임을 관찰함에 있어, 실제 로봇에 사용될 부품에 치수와 일치하는 로봇 링크 기구 그래픽 모델링 작업이 필요하며 본 연구에서는 CAD 작업을 통해 RP방식으로 제작될 실제 로봇을 위해 생성된 로봇 설계 CATIA 데이터 파일을 이용하여 동작 시뮬레이션에 필요한 로봇 기구의 그래픽 모델을 얻는 방식을 택함으로서 실제 로봇과 일치한 치수의 그래픽 모델을 통해 실제 로봇을 개발함에 있어 여러 모션에 관한 움직임 데이터를 개발하고 테스트할 수 있도록 하였다. 또한 시각적 효과를 통한 효율적인 동작 관찰과 사용자의 시각적 편의를 제공하기 위해 3차원 공간상에서의 그래픽요소를 강화하고자 보편화된 OS인 Windows의 편리한 GUI 기능을 이용한 API와 OpenGL을 이용함으로서 효율적인 MMI의 기능을 구비하도록 하였다. 특히 OpenGL의 데이터 관리능력은 OpenGL Rendering Pipeline을 통해 다음과 같이 동작한다.

① Display Lists
모든 기하학적 데이터는 'display list'에 저장된다. 또한 'immediate mode'에서는 데이터를 즉시 처리하는 방법을 가지고 있기도 하다. 하나의 display list가 작동하기 시작하면, 'immediate mode'에서 저장된 데이터를 display list로부터 전송시킨다.

② Evaluators
모든 기하학적 primitive는 결국은 vertex 들로 구성된다. 파라미터로 표현된 곡선이나 표면은 초기에 제어점(control points)과 basis함수라 불리는 polynomial 함수로 표현되기도 한다. 'evaluator'는 제어점으로부터 표면을 나타내는 vertex의 유도 방법을 제공한다. 이 방법은 일종의 polynomial mapping으로서 표면법선이나 텍스처 좌표, 색, 제어점으로부터의 공간좌표값을 만들어 낸다.

③ Per-Vertex Operation s & Primitive Assembly
'per-vertex operations'단계에서는 vertex를 primitive로 바꾸는 역할을 한다.

④ Pixel Operations
픽셀 데이터가 프레임버퍼로부터 읽혀지면, 픽셀 전송동작(스케일, 바이어스, 맵핑, 클램핑)이 수행된다. 이런 결과물들은 적절한 포맷으로 한데 묶이고 시스템 메모리의 배열로 반환된다. 프레임버퍼에 있는 데이터를 프레임버퍼의 다른 부분이나 텍스처 메모리에 복사하기 위해선 특별한 픽셀 복사 작동을 한다. 데이터를 텍스처 메모리에 쓰거나 프레임버퍼로 돌리기전에 픽셀 전송동작을 통한 단일 전송하게 된다.

⑤ Texture Assembly
OpenGL 응용에서 좀 더 사실적으로 보이게 하기위해 기하학물체에 텍스처 이미지를 입힐 수 있다. 텍스처 이미지를 여러개 사용한다면, 이미지들을 텍스처 물체에놓아 쉽게 선택할 수 있도록 하는 것이 현명하다.

⑥ Rasterization

'Rasterization'이란, 기하학 또는 픽셀 데이터를 fragment로 바꾸는 것을 말한다. 각 fragment square는 프레임버퍼내의 한 픽셀에 해당한다. 선 또는 면 stipple, 선두께, 점크기, 음영모델, 또는 안티알리싱을 위한 범위 계산들은 마치 점들이 선으로 연결되어 있거나, 내부 픽셀들이 채워진 면에 대해 계산된 것처럼 고려된다.

⑦ Fragment Operations

값들이 실제로 프레임버퍼에 저장되기 전에 일련의 동작들이 fragment를 변경하거나 버리도록 수행된다. 이런 모든 동작들은 작동가능 혹은 작동 불가능하게 할 수도 있다.

뿐만 아니라 OpenGL의 코드 효율성은 코드 작성의 합리성으로 보여줄 수 있다. 다음의 코드 예는 이동변환 행렬과 회전 변환 행렬을 보여주는 것으로 OpenGL에서의 코드 작성이 얼마나 간단하고 편리한 것인지를 보여준다.

```
void Translate_Matrix
(GLfloat X, GLfloat Y, GLfloat Z, GLfloat T[][4])
{
        ///translate matrix
        T[0][0]=1; T[0][1]=0; T[0][2]=0; T[0][3]=X;
        T[1][0]=0; T[1][1]=1; T[1][2]=0; T[1][3]=Y;
        T[2][0]=0; T[2][1]=0; T[2][2]=1; T[2][3]=Z;
        T[3][0]=0; T[3][1]=0; T[3][2]=0; T[3][3]=1;
}
void X_Rotation_Matrix(GLfloat Rot, GLfloat Rx[][4])
{
        ///Rotation matirx
        Rx[0][0]=1; Rx[0][1]=0;
        Rx[0][2]=0;
        Rx[0][3]=0;
        Rx[1][0]=0;
      Rx[1][1]= fcos(Rot*GL_PI/180.0f);
      Rx[1][2]= -fsin(Rot*GL_PI/180.0f);
      Rx[1][3]=0;
      Rx[2][0]=0;
      Rx[2][1]= fsin(Rot*GL_PI/180.0f);
      Rx[2][2]= fcos(Rot*GL_PI/180.0f);
      Rx[2][3]=0;
        Rx[3][0]=0;
      Rx[3][1]=0;
Rx[3][2]=0;
Rx[3][3]=1;
}
```

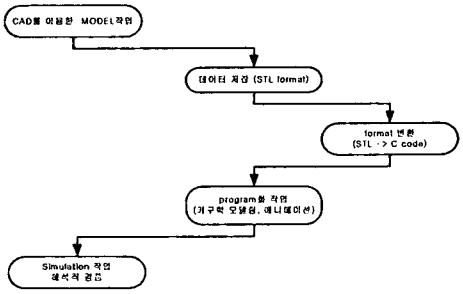
```
glTranslated(xMOV, yMOV+90, 0.0f);
glRotatef(xRot, 1.0f, 0.0f, 0.0f);
```

[표 1]코드작성의 효율(위 일반 C code, 아래 OpenGL code)

2.2 CATIA DATA 변환에 의한 Virtual Prototype과 각 부품의 조립

본 논문에서는 로봇 설계 CAD 데이터를 이용하였는데 CAD 데이터는 C 언어에서 핸들하기에 매우 많은 양의 데이터를 가지고 있기 때문에 여러 가지 제약을 수반하게 된다. 이런 문제를 해결하기 위해 다음의 두 가지 방법을 고려할 수 있다.첫 번째 방법은 원 데이터에 포함된 데이터들을 재구성하여 양을 줄이는 방법이다. CAD 데이터에 있는 많은 정보들 중에 유사성과 근접성이 높은 데이터를 생략하므로써 같은 그래픽 효과를 얻으면서도 데이터양을 줄일 수 있다. 그러나 이러한 방법은 높은 수준의 지능 프로그램을 요구하므로 그 수행과정이 복잡하다. 두 번째 방법으로는 그래픽 API인 OpenGL을 이용하여 유사하게 그리는 방법이다. 그래픽 API인 OpenGL에는 이미 몇 개의 기본 도형을 제공한다. 이것을 이용하여 부품을 유사하게 구성하고 조립하여 그래픽 효과를 높인다. 그러나 전자의 방법에 비하여 실제 모델을 묘사하기가 어렵고 단순화된 그림만을 얻을 수 있다. 이런 이유로 현실감 있는 그래픽 시뮬레이션을 위해서 CAD 모델 작업을 통해 얻어진 CAD 데이터를 필요한 부분만으로 재가공하고 이를 이용하여 프로그램 작업을 하였다.

다음 그림은 기구 설계 작업에 사용된 데이터로부터 동작 시뮬레이터를 작성하는 흐름도를 나타낸다.



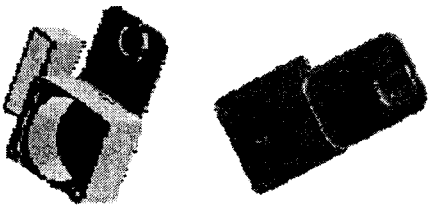
[그림 1] 시뮬레이터 작성 흐름도

다음 그림은 실제 CAD데이터를 Visual C코드로 변환하는 작업과정 및 결과그림을 나타낸다.

```
solid CATIA STL
 facet normal 0.995429e+00 0.000000e+00 -0.955031e-01
  outer loop
  vertex -1.090912e+01  1.579855e+01  1.860049e+02
  vertex -1.089999e+01  1.700000e+01  1.861000e+02
  vertex -1.089999e+01  1.576969e+01  1.861000e+02
  endloop
 endfacet
```

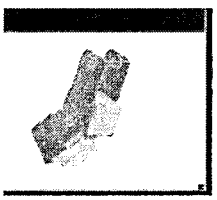
```
#include <windows.h>          // Window defines
#include <gl\gl.h>            // OpenGL
#include <gl\glu.h>           // GLU library

GLdoublevertices0[17853] =

-10.9091, 15.7986, 186.005,
-10.9, 17, 186.1,
-10.9, 15.7697, 186.1,
-10.9096, 17, 186.002,
-10.9364, 15.8246, 185.912,
```

[표 2] 모델 데이터 변환 코드



[그림 3] 부품 변환 결과 그림 (좌 : 모터실, 우 : 덮게)

다음 그림은 위의 과정을 통해서 변환된 각 블록들의 조합 그림이다.



[그림 4] 조립된 부품들에 의해 연결된 관절들의 모습

2.3 역기구학

말단장치의 위치와 방향을 관절변수로 나타내는 방법을 순기구학이라 한다면 역으로 말단장치의 위치와 방향에서 관절변수를 찾아내는 방법을 역기구학이라 한다.

역기구학 문제는 기구학적 형상에 대하여 기하학적인 풀이방법과 해석적으로 푸는 방법(closed form solution)

이 있는데 해석적 풀이 방법의 경우 비전 시스템으로 용접선을 따라가는 것과 같은 응용처럼 순기구학 방정식을 매우 빠른 시간 내에 풀어야 하는 경우 반복 계산에 의한 답보다 실질적으로 쓸모가 있다. 하지만 순기구학 방정식이 일반적으로 관절변수의 복잡한 비선형 함수꼴을 가지므로, 답이 있어도 찾기 어려운 문제이다. 본 연구에서는 해석적 답을 찾기 위한 새로운 방법을 제안한다.
가정: 발바닥면은 지면에 평행하다.

2.4 시뮬레이터의 기능

개발한 로봇이 사람의 명령을 받아 작업을 수행하기 위해서는 사람과 로봇 사이에 정보를 주고받을 수 있는 사용자 인터페이스가 필요하게 되는데, 본 연구에서는 로봇의 머리에 부착된 카메라를 통해 획득한 주변 환경 정보를 시뮬레이터의 그래픽 화면에 표시함으로써 사용자가 로봇의 상황에 따라 적절한 명령을 내릴 수 있게 하기 위한 의사 결정 보조 시스템의 역할까지 수행하는 사용자 인터페이스 기능을 추가하였다. 또한 보행 동작 중 상체(팔과 머리)의 동작을 지시할 수 있도록 하며 필요시 동작을 저장할 수 있도록 함으로서 수리적으로 규칙적이며 정적 보행을 위해 미리 계산된 보행과 달리 팔과 머리의 동작은 임으로 생성하여 이용할 수 있도록 하였다.

2.5.1 Fulldown 메뉴와 기능 창

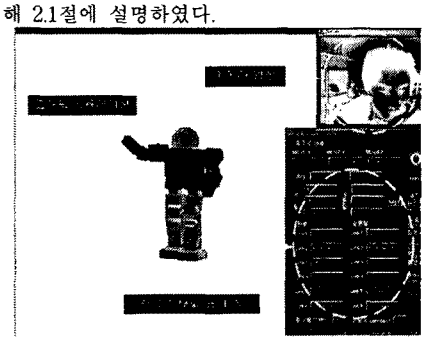
사용자의 편의성을 위하여 다중작업이 가능하도록 하였으며 전후좌우 및 위와 아래에서등 다양한 시점에서 로봇의 움직임의 관찰을 용이하도록 하였다. 또한 전체적인 모습과 세부적인 동작상태를 볼 수 있도록 창의 크기의 조절이 가능하도록 구현하였다.

2.6.2 동작 속도 조절과 데이터 저장

실제로 구현된 로봇의 움직임을 다양한 속도로 관찰. 실험하기 위하여 로봇 움직임에 속도의 변화를 주어 로봇의 링크 구조들이 그에 따라 어떠한 형태를 보이는지 볼 수 있도록 하였다. 그러므로써 로봇의 속도를 평가할 수 있도록 하였다. 또한 이처럼 다양하게 움직인 로봇의 움직임에 따른 데이터를 저장하고 다시 리로드할 수 있는 기능을 추가하여 실제 로봇의 궤적 생성에 이용이 가능함을 보였다.

2.6.3그래픽 모델의 단순화

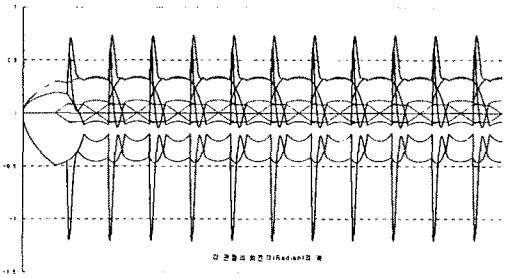
모델을 단순화시킴으로서 로봇의 외관과 내관의 다양한 형태의 모습들의 관찰이 가능하도록 구현하였다. 음영을 포함한 외부의 모습을 관찰할 수도 있으며 철사 구조물과 같은 내부의 모습이나 전면에서는 보이지 않는 후면의 모습까지 볼 수 있다. 2.6에서 설명한 순서대로 작성된 그래픽 시뮬레이션은 로봇의 움직임을 사전에 그래픽을 통해 검증하고 또한 사용자 인터페이스를 위하여 무선 통신을 통해 수신된 카메라 영상 데이터를 처리하고 무선 모뎀을 통해 로봇에 명령을 전달하는 기능과 각 관절의 상태값 및 모션 편집 기능을 추가하였다. 이를 다음 그림에서 보인다. 또한 본 논문에서 사용한 OpenGL에 대해 2.1절에 설명하였다.



[그림 5] MMI 툴 실행창

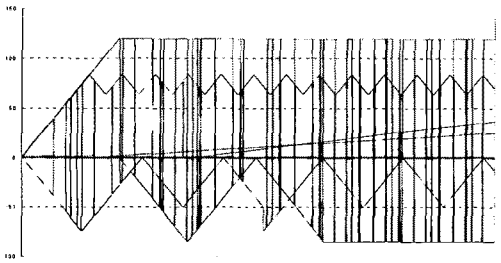
3. 결 론

본 시뮬레이터에서 이용된 로봇에 정적보행을 위해 미리 계산된 데이터([그림 6.])를 이용하여 시뮬레이터를 통해 구동함으로서 데이터의 오류에 의한 기구의 파손을 방지하고 보행을 위한 계산식을 검증할 수 있었으며 빠르고 느린 선택적 관찰과 동작 중 여러 방향에서의 관찰을 통해 예상할 수 있는 보행과 각 관절 기구의 움직임을 관찰함으로서 로봇에 외부 동작뿐 아니라 내부의 연결부 동작을 관찰할 수 있었다. 또한 기구학적 설계에 유용성을 확인함으로서 실제 로봇 제작에 선행하여 기구학적 모델을 확인할 수 있었다.



[그림 6] 다리 관절의 회전각(Radian) 궤적

또한 동작 중 정지기능과 저장기능을 이용하여 상체의 동작을 만들어 냄으로서 하체의 동작을 기반으로하는 손동작과 머리동작을 제작하여 실제 로봇에 적용하기 위한 테스트로서의 역할을 수행함과 동시에 새로운 움직임(Motion)을 만들어 지시할 수도 있음을 보였다.



[그림 7] 팔 관절의 회전각(Degree) 궤적

[참 고 문 헌]

[1] Raibert, M. Kuroki, Y., Playter, R. Ishida, T., Doi, T., "Dynamic Simulation of Humans and Humanoids, Humanoids" Proc. IEEE-RAS Int. Conf. Humanoid Robots 2001, pp. 265-270, 2001

[2] 진재호, "소형 휴머노이드 로봇 시스템에 관한 연구", 전북대학교 석사학위 논문, p10~p21, 2003

[3] Richard S. Wright Jr Michael Sweet "OpenGL superbible", The Waite Group Inc. pp30-46, 1996

[4] 김상형 "Windows A.P.I. 정복" 가남사, pp19-31, pp868-910, 2002