碩士學位論文

모발-매니퓰레이터의 실시간 다중 장애물 회피에 관한 연구

濟州大學校 大學院

메 카 트 로 닉 스 공 학 과

梁 亨 贊

2009 年 7月

碩士學位論文

모발-매니퓰레이터의 실시간 다중 장애물 회피에 관한 연구

濟州大學校 大學院

메카트로닉스공학과

梁　亨　贊

2009年　7月

모발-매니퓰레이터의 실시간 다중 장애물 회피에 관한 연구

지도교수 최 경 현

양 형 찬

이 論文을 工學 碩士學位 論文으로 提出함

2009年 7月

梁亨贊의 工學 碩士學位 論文을 認准함

審査委員長 ㊞

委 員 ㊞

委 員 ㊞

濟州大學校 大學院

A Study on the Real-time Multiplex obstacle avoidance of Mobile-Manipulator

Hyoung-Chan Yang

(Supervised by professor Kyung-Hyun Choi)

A thesis submitted in partial fulfillment of the requirement

for the degree of Master of Engineering

Department of Mechatronics Engineering

GRADUATE SCHOOL

CHEJU NATIONAL UNIVERSITY

2009. 7.

목 차

List of Tables

List of Figures

Summary

This paper proposes the Elastic Force application on the obstacle avoidance of Silvermate Robots. The method deals with the problem associated with a Silvermate robot driving to a goal configuration as avoiding obstacles. The initial trajectory of a robot is determined by a motion planner, and the trajectory modification is accomplished by adjusting the control points. The control points are obtained based on the elastic force approach. Consequently the trajectory of a robot is incrementally modified to maintain a smooth and adaptive trajectory in an environment with obstacles. The suggested algorithm drives the robot to obstacle avoid in real-time. Finally, The simulation studies are carried out to illustrate the effectiveness of the proposed approach.

Ⅰ. 서 론

현대 사회에 인간은 시간과 공간의 제약에서 자유로워지고 삶의 질이 향상을 위하여 산업용 로봇과 가정용 로봇 등 많은 분야에서 로봇을 사용하고 있다. 점점 로봇기술의 중요성이 날로 증대되고 있으며, 로봇은 인간이 단순 작업과 정밀작업 등을 대체하고, 인간이 할 수 없는 많은 분야에서 인간을 노동력을 대신하여 산업분야와 연구 분야, 일상생활에서 효율적인 발전을 이루고 있다.

인간의 노동력을 대처하기 시작한 산업용 로봇은 인간이 적응하기 어려운 불안정한 환경에서 오차율이 적은 정밀 작업을 할 수 있게 하고, 새로운 환경에 대한 탐사와 같이 불확실한 환경에도 로봇이 투입되어 보다 안전하고 복잡한 작업의 수행을 하고 있다. 또한, 세계적의 인구가 고령화 추세로 접어들면서 이를 보조할 가정용 및 실버용 로봇에 대한 관심도와 중요성이 나날이 증가하고 있다. 일상생활에서 사용되는 세탁기와 청소기 같은 인간의 노동력이 필요한 수동적인 보조도구와 다르게 가정용 로봇과 실버용 로봇은 이러한 인간이 해야 할 부분까지 능동적으로 일을 처리하기 때문에 삶의 질적 향상을 위하여 필요한 부분이라 할 수 있다.

이렇게 우리 생활에서 점점 빠질 수 없게 되는 로봇이기 때문에 로봇에게 주어진 다양한 미션에 대한 성공적인 수행을 위하여 로봇은 현재의 상황을 실시간으로 파악하고, 이를 능동적으로 대체할 수 있도록 해야 한다. 따라서 이를 자동으로 처리할 수 있는 로봇의 AI 알고리즘과 작업 스케줄러의 향상을 위한 기술개발은 필수일 수밖에 없다.

또한, 로봇이 능동적으로 미션을 수행하기 위해서 이동할 때, 이동 경로 위에 장애물이 있을 경우 안전하게 장애물을 회피할 수 있는 최적 경로 재탐색과, 경로의 수정하여 재조정할 수 있는 실시간 이동 기술은 로봇 분야의 가장 기본이면서도 중요한 축이라 할 수 있다.

본 연구에서는 위와 같은 실시간 이동 기술 중 장애물 발견되면 신속하고 안전하게 최적화된 경로를 재탐색할 수 있는 충돌 회피 알고리즘에 대하여 논

한다.

Ⅱ. 모바일 매니퓰레이터의 기구학 해석

1. 모바일-매니퓰레이터 로봇 구조

1.1 하드웨어 구성

본 논문에서 사용한 자율 이동로봇은 Fig. 1과 Fig. 2에서 보는 바와 같이 21C 프론티어 연구 개발사업의 일환인 인간기능 생활지원 지능로봇 사업단에서 노인 복지용 로봇 개발을 위한 목적으로 설계, 제작되었다. 이동로봇의 하드웨어는 (주)다사테크에서 Fig. 1과 같이 이륜구동 방식으로 제작한 주행 베이스와 Fig. 2와 같이 7축의 여유자유도를 갖는 매니퓰레이터가 모바일 양 쪽에 장착되어 있다.

이 매니퓰레이터는 3개의 링크가 연결되어 있으며 팔목에서 손목사이에 옵셋(off set)이 있어 일반적인 매니퓰레이터와 기구적 특성이 다르다. 각 조인트 사이에는 위치 센서, 관절 토크 센서 및 F/T 센서 등이 내장되어 있으며 파지와 조작 작업이 가능하다. 로봇의 시스템 구성 및 각 부의 인터페이스는 Fig. 3과 같이 표현되고 로봇에 탑재된 센서의 사양은 Table 1과 같이 정리하였다.[1][4]

Fig. 1 Mobile Robot

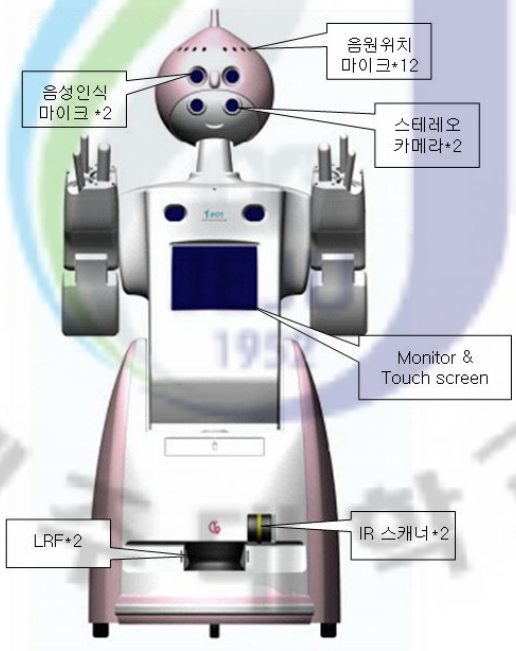


Fig. 2 Robot Configuration

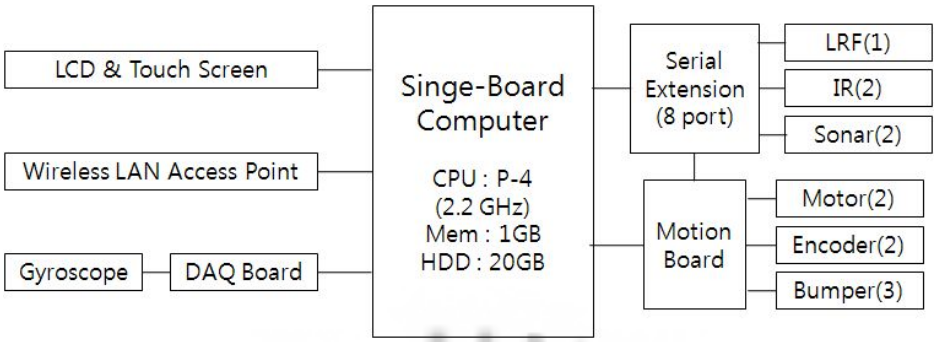


Fig. 3 System configuration of the mobile robot

Table. 1 Sensors specification

Item	Specification	
Laser Range Finder (LMS200-30106) [SICK]	Power Source : DC 24V	Front
	Operation Type : pulsed IR	
	Scan Range : 10m/180°	
	Angular Resolution : 1/0.5/0.25	
	Response Time : 13/26/53 ms	
	Interface Method : RS232/RS422	
IR Scanner (PBS-03JN) [HOKUTO]	Power Source : DC 24V	Front/rear
	Operation Type : pulsed IR	
	Scan Range : 0.2 ~ 3 m (distance)	
	Angular Resolution : 1/0.5/0.25	
	Response Time : 180ms	
	Interface Method : RS232	
Gyro (TA7319N3) [TAMAKAWA]	Power Source : DC 5V	Bottom
	Operation Type : Optical fiber	
	Scan Range : 00s	
	Error : 0.1	
	Interface Method : Analog (DAQ board)	
Ultra Sonar Sensor (PS40S) [Nicera]	Frequency : 40 KHz	12/360 ea/°
	Distance : 0.2 ~ 1 m	
	Resolution : mm	
	Interface Method : RS232C(molues)	
Bumper	Magnetic Limit Switch	3

2. 모바일 로봇의 기구학 해석

2.1 순 기구학 해석

모바일 로봇의 입력변수로부터 직각 좌표계의 위치와 그 시간 미분 값을 구하기 위하여 순 기구학을 구한다. Fig. 4에서는 본 논문에서 사용된 이륜구동 모바일 로봇의 기구학을 나타내었다.

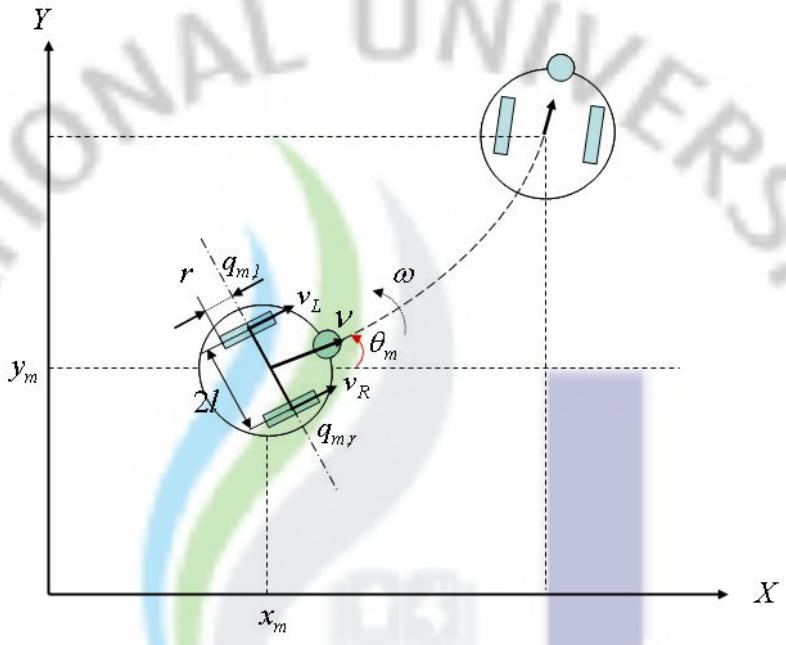


FIg. 4 Mobile robot modeling and coordinate system

여기서,

X, Y : Cartesian 좌표계에서의 world frame

x_m, y_m: Cartesian 좌표계에서의 모바일로봇 중심점의 좌표

q_{ml}, q_{mr}: 모바일로봇의 왼쪽 바퀴와 오른쪽 바퀴의 각속도

l : 모바일로봇 중심에서 바퀴 축까지의 거리

r : 모바일로봇 중심에서의 각속도

v_R, v_L : 모바일로봇의 왼쪽 및 오른쪽 바퀴의 선형속도

u : 모바일로봇 중심에서의 선형속도

ω : 모바일로봇 중심에서의 각속도

모바일 로봇의 기구학을 해석하기 위하여 Fig. 4와 같이 절대 좌표계를 설정하고, 2차원 평면으로 구성된 전역좌표계에서 속도 기구학을 통해 모바일 로봇의 상태를 위치와 방향을 갖는 벡터 $P=[x_m\ y_m\ \theta_m]^T$로 나타낼 수 있다. 일반적으로 모바일 로봇은 Non-holonomic 시스템으로 "순수 구름 조건(Pure rolling condition)"과 "미끌림 없음 조건(Non slipping condition)"이 필요하다. 순수 구름 조건은 바퀴와 접촉면 사이의 순간적 이동 방향으로의 상대속도가 0이라는 조건이므로 각 바퀴에 대하여 식 (1), (2)와 같이 나타낸다.

$$-\cos(\theta_m)\dot{x}_m-\sin(\theta_m)\dot{y}_m-l\dot{\theta}_m+r\dot{q}_{m,r}=0 \tag{1}$$

$$-\cos(\theta_m)\dot{x}_m-\sin(\theta_m)\dot{y}_m-l\dot{\theta}_m+r\dot{q}_{m,l}=0 \tag{2}$$

또한, 미끌림 없음 조건에 대하여 식 (3)과 같이 나타낸다.

$$-\sin(\theta_m)\dot{x}_m+\cos(\theta_m)\dot{y}_m=0 \tag{3}$$

식 (1)~식(3)에 의해 Cartesian space의 모바일 로봇의 속도와 각속도는 식(4)와 같이 결정된다.

$$\begin{aligned}\dot{x}&=\frac{r}{2}(\dot{q}_{m,r}+\dot{q}_{m,l})\cos\theta_m\\ \dot{y}&=\frac{r}{2}(\dot{q}_{m,r}+\dot{q}_{m,l})\sin\theta_m\\ \dot{\theta}_m&=\frac{r}{2l}(\dot{q}_{m,r}+\dot{q}_{m,l})\end{aligned} \tag{4}$$

그리고 모바일 로봇을 제어하는데 있어서의 두 제어입력인 선형속도 u와 각속도 ω로 달리 표현하면, 식(5), (6)과 같다.

$$u=\frac{1}{2}(\nu_R+\nu_L)=r\frac{q_{ml}+q_{mr}}{2} \tag{5}$$

$$\omega=\frac{1}{2l}(\nu_R-\nu_L)=r\frac{q_{ml}-q_{mr}}{l} \tag{6}$$

$[x_m\ y_m\ \theta_m]^T$와 $[u\ \omega]^T$은 자코비안 행렬 $J(p)$에 의해 식(7)과 같이 나타내어진다.

$$\dot{P}=J(p)\dot{q} \tag{7}$$

$$\dot{P}_m = \begin{bmatrix} \dot{x}_m \\ \dot{y}_m \\ \dot{\theta}_m \end{bmatrix} = J(p) \begin{bmatrix} u \\ \omega \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ \omega \end{bmatrix} \tag{8}$$

초기위치를 x_0, y_0, θ_0로 두면 모바일로봇의 현재위치는 식(9)과 같이 식(8)의 적분형태가 된다.

$$P_m = \begin{bmatrix} x_0 \\ y_0 \\ \theta_0 \end{bmatrix} + \begin{bmatrix} \int_t^0 u\cos\theta_m d\tau \\ \int_t^0 u\sin\theta_m d\tau \\ \int_t^0 \omega d\tau \end{bmatrix} \tag{9}$$

이러한 기구학적 관계로부터 로봇은 매 제어주기마다 위치벡터를 계산하며,u 및 ω는 각 바퀴의 엔코더 신호로부터 계산된다.[2]

3. 머니플레이트의 기구학 해석

3.1 순기구학 해석

본 논문에서 사용된 매니퓰레이터는 Fig. 5와 같이 7DOF로 구성되어져 있다. 모든 축은 DC서보 모터에 의해 구동이 된다. 본 논문에서는 1, 3, 5축의 회전중심을 Z방향으로 설정하여 해석 하였다. 매니퓰레이터의 기본 형태인 모든 조인트 값이 0일 때의 모양은 Fig. 6과 같이, 좌표계설정은 Table 2와 같이 나타낼 수 있다.

Fig. 5 Manipulator 3D Modeling

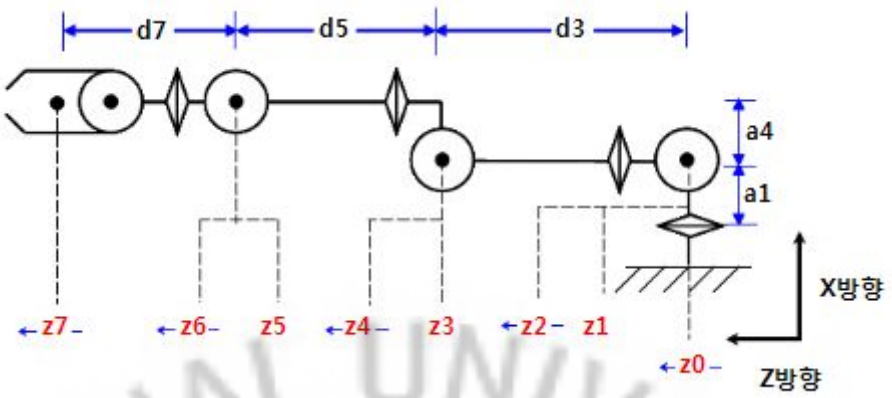


Fig. 6 Base design of Manipulator

Table 2 Llink Parameter

Joint	Joint Angle (θ)	Distance (d)	Offset Angle (α)	Common Length (a)
1	θ_1	0	$\pi/2$	a_1
2	θ_2	0	$-\pi/2$	0
3	θ_3	d_3	$\pi/2$	0
4	θ_4	0	$-\pi/2$	a_4
5	θ_5	d_5	$\pi/2$	0
6	θ_6	0	$-\pi/2$	0
7	θ_7	d_7	$\pi/2$	0

각 링크의 파라미터로부터 좌표계간의 동차행렬(homogeneous matrix)을 식(10)과 같이 구할 수 있으며, 이로부터 기저 좌표계와 마지막 좌표계 사이의 변환관계를 나타내는 동차행렬 식(11)를 얻게 된다.

$$A_1 = \begin{pmatrix} Cos\theta_1 & 0 & -Sin\theta_1 & a_1 Cos\theta_1 \\ Sin\theta_1 & 0 & Cos\theta_1 & a_1 Sin\theta_1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$A_2 = \begin{pmatrix} Cos\theta_2 & 0 & -Sin\theta_2 & 0 \\ Sin\theta_2 & 0 & Cos\theta_2 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$A_3 = \begin{pmatrix} Cos\theta_3 & 0 & -Sin\theta_3 & 0 \\ Sin\theta_3 & 0 & Cos\theta_3 & 0 \\ 0 & 1 & 0 & d_3 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$A_4 = \begin{pmatrix} Cos\theta_4 & 0 & -Sin\theta_4 & -a_4 Cos\theta_4 \\ Sin\theta_4 & 0 & Cos\theta_4 & -a_4 Sin\theta_4 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$A_5 = \begin{pmatrix} Cos\theta_5 & 0 & -Sin\theta_5 & 0 \\ Sin\theta_5 & 0 & Cos\theta_5 & 0 \\ 0 & 1 & 0 & d_5 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$A_6 = \begin{pmatrix} Cos\theta_6 & 0 & Sin\theta_6 & 0 \\ Sin\theta_6 & 0 & -Cos\theta_6 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$A_7 = \begin{pmatrix} Cos\theta_7 & 0 & Sin\theta_7 & 0 \\ Sin\theta_7 & 0 & -Cos\theta_7 & 0 \\ 0 & 0 & 1 & d_7 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{10}$$

$${}^0_7T = A_1A_2A_3A_4A_5A_6A_7 = \begin{pmatrix} x_x & y_x & z_x & p_x \\ x_y & y_y & z_y & p_y \\ x_z & y_z & z_z & p_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{11}$$

$$
\begin{aligned}
x_x &= C_7(C_6(C_5(C_4(C_1C_2C_3 - S_1S_3) - C_1S_2S_4) + (-C_3S_1 - C_1C_2S_3)S_5) \\
&\quad + (-C_1C_4S_2 - (C_1C_2C_3 - S_1S_3)S_4)S_6) + (C_5(-C_3S_1 - C_1C_2S_3) - (C_4(C_1C_2C_3 - S_1S_3) - C_1S_2S_4)S_5)S_7 \\
x_y &= C_7(C_6(C_5(C_4(C_2C_3S_1 + C_1S_3) - S_1S_2S_4) + (C_1C_3 - C_2S_1S_3)S_5) \\
&\quad + (-C_4S_1S_2 - (C_2C_3S_1 + C_1S_3)S_4)S_6) + (C_5(C_1C_3 - C_2S_1S_3) - (C_4(C_1C_2C_3 - S_1S_3) - S_1S_2S_4)S_5)S_7 \\
x_z &= C_6(-C_1C_3S_2 - (C_1C_2C_3 - S_1S_3)S_4) - (C_5(C_4(C_1C_2C_3 - S_1S_3) - C_1S_2S_4) + (-C_3S_1 - C_1C_2S_3)S_5)S_6 \\
y_x &= -C_1d_3S_2 - a_4C_4(C_1C_2C_3 - S_1S_3) + a_4C_1S_2S_4 + d_5(-C_1C_4S_2 - (C_1C_2C_3 - S_1S_3)S_4) \\
&\quad + d_7(C_6(-C_1C_4S_2 - (C_1C_2C_3 - S_1S_3)S_4) - (C_5(C_4(C_1C_2C_3 - S_1S_3) - C_1S_2S_4) + (-C_3S_1 - C_1C_2S_3)S_5)S_6) \\
y_y &= C_7(C_6(C_5(C_4(C_2C_3S_1 + C_1S_3) - S_1S_2S_4) + (C_1C_3 - C_2S_1S_3)S_5) + (-C_4S_1S_2 - (C_2C_3S_1 + C_1S_3)S_4)S_6) \\
&\quad + (C_5(C_1C_3 - C_2S_1S_3) - (C_4(C_2C_3S_1 + C_1S_3) - S_1S_2S_4)S_5)S_7 \\
y_z &= C_7(C_5(C_1C_3 - C_2S_1S_3) - (C_4(C_2C_3S_1 + C_1S_3) - S_1S_2S_4)S_5) - (C_6(C_5(C_4(C_2C_3S_1 + C_1S_3) - S_1S_2S_4) \\
&\quad + (C_1C_3 - C_2S_1S_3)S_5) + (-C_4S_1S_2 - (C_2C_3S_1 + C_1S_3)S_4)S_6)S_7 \\
z_x &= C_6(-C_4S_1S_2 - (C_2C_3S_1 + C_1S_3)S_4) - (C_5(C_4(C_2C_3S_1 + C_1S_3) - S_1S_2S_4) + (C_1C_3 - C_2S_1S_3)S_5)S_6 \\
z_y &= -d_3S_1S_2 - a_4C_4(C_2C_3S_1 + C_1S_3) + a_4S_1S_2S_4 + d_5(-C_4S_1S_2 - (C_2C_3S_1 + C_1S_3)S_4) \\
&\quad + d_7(C_6(-C_4S_1S_2 - (C_2C_3S_1 + C_4S_3)S_4) - (C_5(C_4(C_2C_3S_1 + C_1S_3) - S_1S_2S_4) + (C_1C_3 - C_2S_1S_3)S_5)S_6) \\
z_z &= C_7(C_6(C_5(C_3C_4S_2 + C_2S_4) - S_2S_3S_5) + (C_2C_4 - C_3S_2S_4)S_6) + (-C_5S_2S_3 - (C_3C_4S_2 + C_2S_4)S_5)S_7 \\
p_x &= C_7(-C_5S_2S_3 - (C_3C_4S_2 + C_2S_4)S_5) - (C_6(C_5(C_3C_4S_2 + C_2S_4) - S_2S_3S_5) - S_2S_3S_5) + (C_2C_4 - C_3S_2S_4)S_6)S_7 \\
p_y &= C_6(C_2C_4 - C_3S_2S_4) - (C_5(C_3C_4S_2 + C_2S_4) - S_2S_3S_5)S_6 \\
p_z &= d_1 + C_2d_3 - a_4C_3C_4S_2 - a_4C_2S_4 + d_5(C_2C_4 - C_3S_2S_4) + d_7(C_6(C_2C_4 - C_3S_2S_4) - (C_5(C_3C_4S_2 + C_2S_4) - S_2S_3S_5)S_6)
\end{aligned}
$$

[단, $C_n = \cos(\theta_n),\ S_n = \sin(\theta_n)$]

식(11)은 7축 로봇의 기구학을 대표하고, 여기서 4열의 p_x, p_y, p_z는 손끝(end-effector)의 위치를 나머지 요소들은 말단의 방향을 나타낸다.

3.2 역기구학 해석

역기구학이란 end-effector의 위치(position)와 방위(orientation)를 각 축의 각으로 변환하는 과정이다. 위치와 방위를 합쳐 자세(pose)라는 용어를 사용한다. end-effector의 자세를 식(12)와 같이 나타낸다.

$$
P = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} n & o & a & p \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{12}
$$

n, o, a는 방위를 나타내는 벡터로 각각 기본자세에서 end-effector의 x, y, z축 방향이 변환된 것이다. p는 위치를 나타내는 벡터이다. 위치 벡터를 식(13)처럼 표현할 수 있다.

$$p = transl(P) = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} \tag{13}$$

3.2.1 어깨부터 손목까지

$\theta = 0$일 때, 즉 $\theta_3 = 0$으로 고정하여 6자유도 매니퓰레이터 1, 2, 3축에 관한 식을 우하고, 그것을 이용하여 7자유도 매니퓰레이터 역기구학 식을 구한다. 이 절에서는 6자유도 매니퓰레이터와 관련된 식은 6자유도의 링크 파라미터 기호를 따른다.

세 축 이후의 축에 관한 식들은 매니퓰레이터의 방위를 만족시키도록 구하게 되는데, 그 전에 먼저 손목까지의 매니퓰레이터 위치가 Fig. 6에서 벡터 $\bar{p}$를 만족하도록 $\theta_1, \theta_2, \theta_3$를 구한다.

Fig. 6에서 $\bar{p}$는 식(14)로 표현할 수 있다.

$$\bar{p} = p - d_6 a = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} - d_6 \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} \tag{14}$$

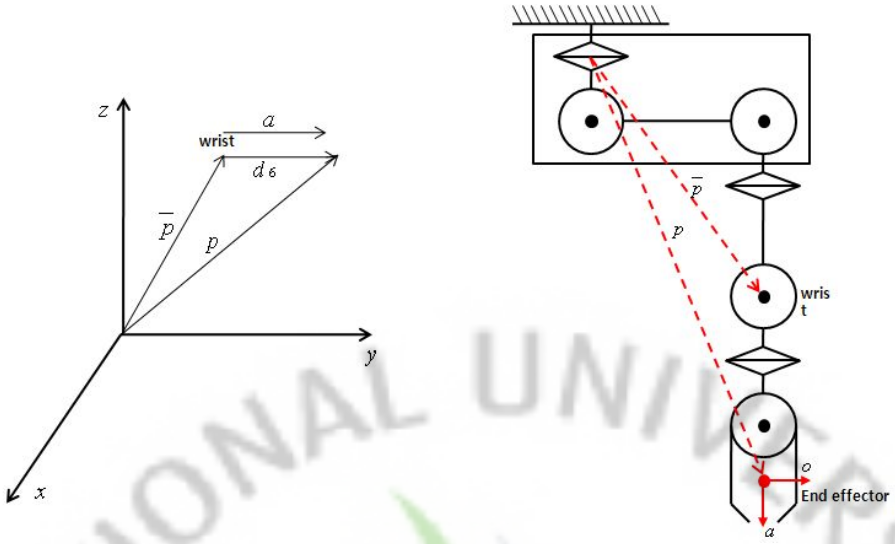


Fig. 7 6DOF Inverse kinematics

손목까지의 자세는 각축의 DH(Denavits Hartenberg) 행렬의 곱을 이용하여 식(15)와 같이 표현할 수 있다.

$$
\begin{aligned}
{}^{0}A_{w} &= {}^{0}A_{1}\,{}^{1}A_{2}\,{}^{2}A_{3}\,{}^{3}A\theta_{w} \\
&= \begin{bmatrix}
\cos\theta_1\cos(\theta_2+\theta_3) & -\sin\theta_1 & \cos\theta_1\sin(\theta_2+\theta_3) & a_2\cos\theta_1\cos\theta_2 \\
\sin\theta_1\cos(\theta_2+\theta_3) & \cos\theta_1 & \sin\theta_1\sin(\theta_2+\theta_3) & a_2\sin\theta_1\cos\theta_2 \\
-\sin(\theta_2+\theta_3) & 0 & \cos(\theta_2+\theta_3) & d_1-a_2\sin\theta_2 \\
0 & 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix} *** & 0 \\ *** & 0 \\ *** & d_4 \\ *** & 1 \end{bmatrix} \\
&= \begin{bmatrix} *** & \overline{p_x} \\ *** & \overline{p_y} \\ *** & \overline{p_z} \\ *** & 1 \end{bmatrix}
\end{aligned}
\tag{15}
$$

식(15)에서 $\overline{p_x}, \overline{p_y}, \overline{p_z}$는 식(16)과 같다.

$$
\begin{aligned}
\overline{p_x} &= \cos\theta_1(\sin(\theta_2+\theta_3)d_4+a_2\cos\theta_2) \\
\overline{p_y} &= \sin\theta_1(\sin(\theta_2+\theta_3)d_4+a_2\cos\theta_2) \\
\overline{p_z} &= \cos(\theta_2+\theta_3)d_4+d_1-a_2\sin\theta_2
\end{aligned}
\tag{16}
$$

식(16)을 연립하여 풀면 $\theta_1, \theta_2, \theta_3$은 식(17)과 같이 구할 수 있다.

$$\theta_1 = \tan^{-1}\frac{\pm\overline{p_y}}{\pm\overline{p_x}}$$

$$\theta_2 = \tan^{-1}\frac{(\overline{p_z}-d_1)(a_2-d_4\sin\theta_3)-d_4\cos\theta_3(\pm\sqrt{\overline{p_x^2}+\overline{p_y^2}})}{(\overline{p_z}-d_1)d_4\cos\theta_3+(a_2-d_4\sin\theta_3)(\pm\sqrt{\overline{p_x^2}+\overline{p_y^2}})} \tag{17}$$

$$\theta_3 = \tan^{-1}\frac{\overline{p_x^2}\overline{p_y^2}+(\overline{p_z}-d_1)^2-(d_4)^2-(a_2)^2}{\pm\sqrt{4(d_4a_2)^2-(\overline{p_x^2}+\overline{p_y^2}+(\overline{p_z}-d_1)^2-(d_4)^2-(a_2)^2)^2}}$$

로봇의 ±기호에 의해 매니퓰레이터가 좌/우 그리고 상/하로 접근하는 네 가지 방법의 결과가 나오는데, 본 연구에 사용되는 로봇의 모양을 고려하여 좌, 상 형태를 유지하도록 하여 식(18)으로 풀게 된다.

$$\theta_1 = \tan^{-1}\frac{-\overline{p_y}}{-\overline{p_x}}$$

$$\theta_2 = \tan^{-1}\frac{(\overline{p_z}-d_1)(a_2-d_4\sin\theta_3)+d_4\cos\theta_3\sqrt{\overline{p_x^2}+\overline{p_y^2}}}{(\overline{p_z}-d_1)d_4\cos\theta_3-(a_2-d_4\sin\theta_3)\sqrt{\overline{p_x^2}+\overline{p_y^2}}} \tag{18}$$

$$\theta_3 = \tan^{-1}\frac{\overline{p_x^2}\overline{p_y^2}+(\overline{p_z}-d_1)^2-(d_4)^2-(a_2)^2}{-\sqrt{4(d_4a_2)^2-(\overline{p_x^2}+\overline{p_y^2}+(\overline{p_z}-d_1)^2-(d_4)^2-(a_2)^2)^2}}$$

세 각에 대한 정보를 이용하여, 7자유도 매니퓰레이터의 역기구학을 풀 수 있다.

Fig. 8은 7자유도 매니퓰레이터의 역기구학을 푸는 방법에 관한 것이다. Fig. 8에서 ′은 $\theta=0$일 때의 값을 의미한다.

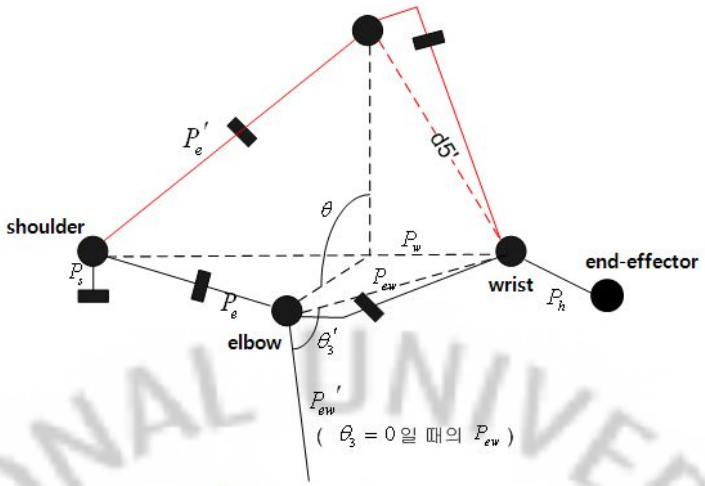


Fig. 8 7DOF Inverse kinematics

먼저 P_w를 식(19)과 같이 구한다.

$$P_w = p - p_h - P_s \tag{19}$$

이 때, p_h와p_s는 식(20)와 같다.

$$p_h = d(6) \times \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} \tag{20}$$

$$P_s = transl({}^0A_1)$$

그리고 P_e'를 식(21)과 같이 구할 수 있다.

$$P_e' = transl({}^0A_1 {}^0A_2) - P_s \tag{21}$$

지금까지 구한 수식을 이용하여 본격적으로 7자유도 매니퓰레이터의 역기구학을 구하게 된다. 이하의 모든 기호는 7자유도 링크 파라미터를 따른다. $n = P_w/|P_w|$일 때, P_e값은 식(22)과 같다.

$$P_e' = n(n \cdot P_e') + \{P_e' - n(n \cdot P_e')\}\cos\theta + (P_e' \times n)\sin\theta \tag{22}$$

P_e를 이용하여 실제 7DOF상의 를 식(23), (24)을 구할 수 있다.

$$\theta_1 = \tan^{-1}\left(\frac{-P_{ey}}{-P_{ex}}\right) \tag{23}$$

$$\theta_2 = \tan^{-1}\left(\frac{\sqrt{P_{ex}^2 + P_{ey}^2}}{P_{ez}}\right) \tag{24}$$

θ_4는 식(18)에서 구한 6자유도 매니퓰레이터에서의 세 번째 축(θ_{3_6dof})과 동일하나, 6자유도 매니퓰레이터와 7자유도 매니퓰레이터 기본 형태, Fig. 6에서 나타나는 각도 차인 $\frac{\pi}{2}$를 식(25)와 같이 해결한다.

$$\theta_4 = \theta_{3\_dof} - \frac{\pi}{2} \tag{25}$$

식 (23), (24)에서 세워진 각을 바탕으로 Fig. 8로 나타낼 수 있다. θ_3을 구하기 위해 θ_3'을 먼저 구한 후 식(26)으로부터 식(27)를 구할 수가 있다.

$$\theta_3' = \tan^{-1}\left(\frac{|P_{ew} \times P_{ew}'|}{P_{ew} \cdot P_{ew}'}\right) \tag{26}$$

$$\theta_3 = 2\sin^{-1}\left(\frac{2\sin\left(\frac{\theta_3'}{2}\right)}{\sin\theta_4}\right) \tag{27}$$

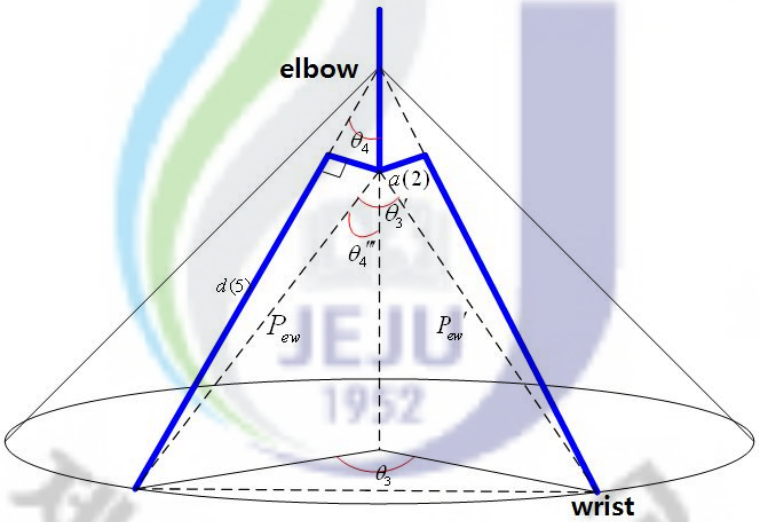
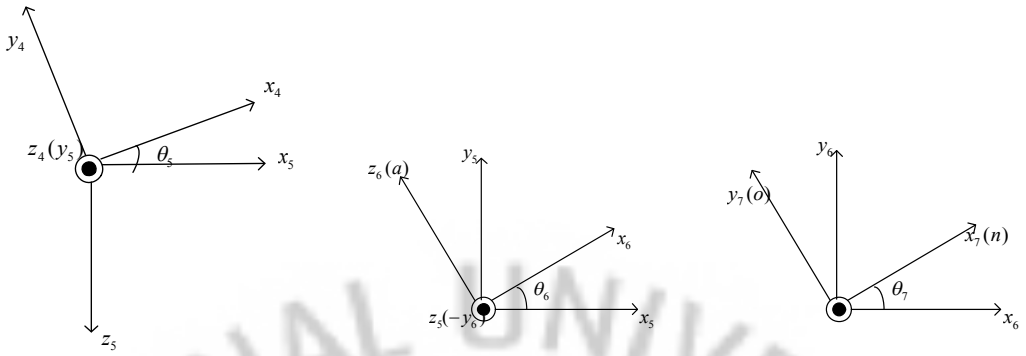


Fig. 9 θ_3

3.2.2 손목부터 end-effector까지

5축부터의 식을 단순화하기 위해 4축까지의 자세를 베이스로 하여 end-effector의 위치를 재 정의한다.

$$P' = \begin{bmatrix} n' & o' & a' & p' \\ 0 & 0 & 0 & 1 \end{bmatrix} = ({}^0A_4)^{-1}P \tag{28}$$



(a) θ_5 (b) θ_6 (c) θ_7

Fig. 10 $\theta_5, \theta_6, \theta_7$

5, 6, 7축은 Fig. 10의 각 벡터 방향을 이용하여, 식(29), (30), (31)과 같이 구할 수 있다.

$$\theta_5 = \tan^{-1}\frac{-z_5.x_4}{-z_5.y_4} = \tan^{-1}\frac{-a'_y}{-a'_x} \tag{29}$$

$$\theta_6 = \tan^{-1}\frac{-a.x_5}{-a.y_3} = \tan^{-1}\frac{-a'_x\cos(\theta_5) - a'_y\sin(\theta_5)}{a'_x} \tag{30}$$

$$\theta_7 = \tan^{-1}\frac{-n'_x\sin(\theta_5) + n'_y\cos(\theta_5)}{-o'_x\sin(\theta_5) + o'_y\cos(\theta_5)} \tag{31}$$

3.3 자코비안 행렬

운동중인 매니퓰레이터를 해석하는데 있어서 시간적으로 변화하는 위치와 방위를 표시하기위해서는 각 링크의 상대적 선속도와 회전속도를 고려하여야 한다.

end-effector의 좌표와 매니퓰레이터의 관절각도 좌표의 관계는 복잡한 비선형 대수방정식으로 표현되기 때문에 일반적으로는 이것을 해석적으로 푸는 것은 곤란하다. 이는 자코비안 행렬을 이용하여 이 복잡성을 피할 수가 있다. 관절형 매니퓰레이터의 경우 실제로 움직이는 것은 관절각도 이기 때문에 매니퓰레이터의 손끝의 위치 및 자세와 관절각도와의 관계가 필요하다.

자코비안 행렬을 구하기 위해서는 식 (11)의 4열의 1~3행까지의 P_x, P_y, P_z end-effector의 위치가 필요하다.

여기서 $P_x = f_1(q), P_y = f_2(q), P_z = f_3(q)$으로 정의하도록 하면,

$$
\begin{aligned}
f_1(q) &= C_7(-C_5S_2S_3 - (C_3C_4S_2 + C_2S_4)S_5) - (C_6(C_5(C_3C_4S_2 + C_2S_4) - S_2S_3S_5) - S_2S_3S_5) \\
&\quad + (C_2C_4 - C_3S_2S_4)S_6)S_7 \\
f_2(q) &= C_6(C_2C_4 - C_3S_2S_4) - (C_5(C_3C_4S_2 + C_2S_4) - S_2S_3S_5)S_6 \\
f_3(q) &= d_1 + C_2d_3 - a_4C_3C_4S_2 - a_4C_2S_4 + d_5(C_2C_4 - C_3S_2S_4) + d_7(C_6(C_2C_4 - C_3S_2S_4) \\
&\quad - (C_5(C_3C_4S_2 + C_2S_4) - S_2S_3S_5)S_6)
\end{aligned}
\tag{32}
$$

식(32)에서 변수와 티끝의 관계가 비선형이므로 이를 갖고 다루는 것은 매우 어렵기 때문에 이를 다음과 같이 시간에 대해 미분하여 선형적인 속도의 관계식으로 표현하여 사용하여야 한다.

조인트 변수가 다음과 같다면 $q_i = [\theta_1\ \theta_2\ \theta_3\ \theta_4\ \theta_5\ \theta_6\ \theta_7]$은 자코비안 행렬을 구하기 위해서 각각 $f_1(q), f_2(q), f_3(q)$ 조인트 변수에 대해 미분되어져야 한다.

여기서 $J_{ij} = \dfrac{\partial f_i(q)}{\partial q_j}$ $i = 1,2,3$ and $j = 1,2,\cdots,7$

$$\dot{P}=\begin{bmatrix}v\\ \omega\end{bmatrix}=\begin{bmatrix}J_v\\ J_\omega\end{bmatrix}\begin{bmatrix}\theta_1\\ \theta_2\\ \theta_3\\ \theta_4\\ \theta_5\\ \theta_6\\ \theta_7\end{bmatrix}=J\dot{q} \qquad J=\begin{bmatrix}J_{11} & J_{12} & J_{13} & J_{14} & J_{15} & J_{16} & 0\\ J_{21} & \ddots & & & & & 0\\ J_{31} & & \ddots & & & & 0\\ J_{41} & & & \ddots & & & 0\\ J_{51} & & & & \ddots & & 0\\ J_{61} & J_{62} & J_{63} & J_{64} & J_{65} & 0 & 1\end{bmatrix} \tag{33}$$

식(33)에서 J는 어떠한 속도의 제한조건도 없는 경우의 모바일 매니퓰레이터에 대한 자코비안 행렬(Jacobian Matrix)을 나타낸다.[3][4]

3.4 모바일-매니퓰레이터로봇의 기구학 해석

독립적인 목적으로 설계된 두 개의 로봇을 결합하여 하나의 작업수행을 위하여 각각의 로봇이 동시에 제어되어야 하므로 전체 시스템의 기구학을 해석하여 지능로봇이 제어되어야 한다. 모바일 로봇은 non-holonomic 시스템이고, 매니퓰레이터로봇은 holonomic 시스템이므로 이동머니퓰레이터의 기구학은 속도기구학을 통해 모바일로봇과 매니퓰레이터로봇을 결합한다.

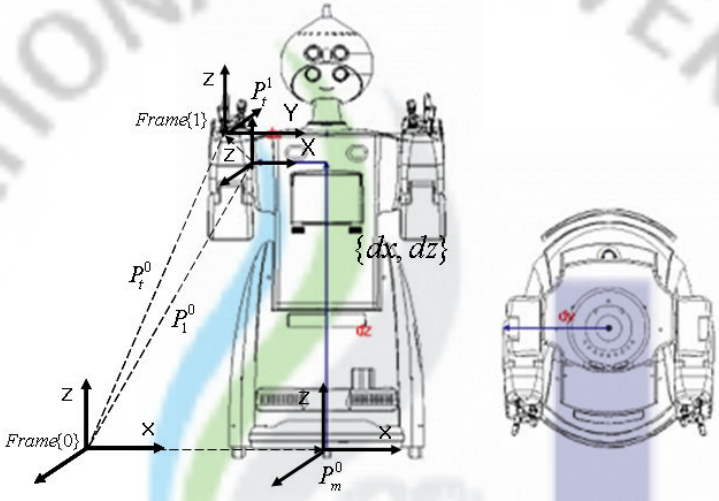


Fig. 11 Coordinate of Mobile Manipulator

Fig.11 에서는 모바일-매니퓰레이터의 모델링을 나타내었다. 매니퓰레이터의 조인트 변수를 $q_i = [\theta_1\ \theta_2\ \theta_3\ \theta_4\ \theta_5\ \theta_6\ \theta_7]^T$,모바일 로봇의 조인트 변수를 $q_m = [x_m\ y_m\ \theta_m]^T$라 정의하면 모바일-매니퓰레이터의 변수 식은 식(34)와 같다.

$$q = \begin{bmatrix} q_m \\ q_i \end{bmatrix} = [x_m\ y_m\ z_m\ \theta_1\ \theta_2\ \theta_3\ \theta_4\ \theta_5\ \theta_6\ \theta_7]^T \tag{34}$$

모바일로봇의 각 축에 대한 조인트 변수에 의해서 형성되는 모바일의 로봇과 매니퓰레이터와의 고정점의 위치 $P_1^0 = [p_x\ p_y\ p_z\ \theta_m]^T$를 프레임 {1}로 설정하였다. P_1^0은 고정 좌표계 프레임{0}을 기준으로 프레임 {1}까지의 위치벡터이다. 모바일-매니퓰레이터의 기구학은 우선 모바일로봇은 3축이므로 조인트 변수 $\dot{\theta_m}$은 바닥

면의 고정 프레임에 대한 직교좌표 공간상에서의 선, 각속도 $\dot{P}_m^0 = [V_m^0 \ \omega_m^0]^T$와 같으며 매니퓰레이터 로봇의 조인트 변수들의 벡터 $\dot{\theta}_t$의 선, 각속도는 $\dot{P}_t^1 = [V_t^1 \ \omega_t^1]^T$과 같다. 각각의 로봇에 대한 자코비안이 J_m^0, J_t^1으로 주어질 때, 모바일-매니퓰레이터 로봇의 자코비안을 J_1^0라고 하면, 바닥면에 대한 말단부의 선, 각속도 $\dot{P}_t^0 = [V_t^0 \ \omega_t^0]^T$는 다음 식 (35)과 같다.

$$\dot{P}_t^0 = \begin{bmatrix} V_t^0 \\ \omega_t^0 \end{bmatrix} = \begin{bmatrix} V_m^0 \\ \omega_m^0 \end{bmatrix} + \begin{bmatrix} \omega_m^0 + R_1^0 V_t^1 \\ R_1^0 \omega_t^1 \end{bmatrix}$$
$$= \begin{bmatrix} J_m^0 & J_t^0 \end{bmatrix} \begin{bmatrix} \dot{\theta}_m \\ \dot{\theta}_t \end{bmatrix} \quad (35)$$

R_1^0은 {0}에서 매니퓰레이터의 {1}까지의 회전변환행렬이다. 즉, 식 (35)에 의해 손끝의 운동은 모바일과 매니퓰레이터가 함께 관여함을 알 수 있다.[2][4]

Ⅲ. 충돌회피 알고리즘

모바일-매니퓰레이터 로봇의 장애물 회피를 어렵게 만드는 요소들은 다음과 같다. 먼저 모바일-매니퓰레이터 로봇은 자유도가 크다. 3차원 공간상의 장애물을 형상 공간 또는 관절 공간 장애물로 변환시키는 과정이 복잡하다. 또한 dimension을 갖는 형상 공간에서의 충돌을 피하기 위한 경로 및 궤적을 탐색하는 것도 많은 계산량을 요구하므로 실시간 적용이 어렵다. 이러한 문제는 고정된 매니퓰레이터 경우에도 형상 공간에서의 탐색을 어렵게 만들고 있으며, 모바일 부분까지 가진 모바일-매니퓰레이터의 경우는 한층 더 복잡한 계산을 요구한다.

모바일 로봇의 충돌 회피에 관해서는 적용성이 좋은 방법들이 많이 제안되어져 있다. 특히 holonomic특성을 가진 로봇의 경우 계획된 경로나 궤적을 충실히 따라갈 수 있어서 좋은 충돌 회피 특성을 보인다. non-holonomic 특성을 가진 모바일의 경우 운동 기구학적 동작 제한 때문에 충돌 회피 궤적이 존재하여도 실제 로봇의 동작이 이를 따라가기 어려워 적용에 문제가 발생하기도 한다.

모바일 로봇의 충돌 회피를 위한 대표적인 방법의 하나인 인공 전위계 방법은 오랫동안 연구되어져 왔으며, 이의 변형된 형태들도 많이 제안되어져 있다. 이 방법은 적용이 간단하고 비교적 좋은 특성을 가지고 있으나, 지역 최소점을 피하기 위한 추가적인 대책을 요구한다. 모바일 로봇의 궤적을 연속적인 원호로 취급하는 방법들은 non-holonomic 특성을 가진 모바일에도 적용할 수 있고, 부드러운 궤적을 가지므로 광범위하게 사용되어지고 있다.

모바일-매니퓰레이터 로봇의 장애물 회피를 위해 제안되어진 Elastic Strip 방법은 모바일 로봇의 충돌 회피 방법인 Elastic Band 방법을 발전시킨 것으로서 동작계획과 동작 수행 과정을 동시에 수행하여 holonomic 특성을 가진 모바일 로봇의 충돌을 피하면서 목적 형상까지 동작 시킨다. 그러나 non-holonomic 특성의 모바일부분을 가진 로봇에 적용하기 어렵고, 로봇의 sweep 공간, 탄성 터널(elastic tunnel), 충돌 검사등 공간상의 기하학적 계산에 많은 계산을 필요로 한다. 특히 로봇이 장애물과 가까이 위치해 있는 경우, 더 많은 경유 형상을 필요

로 하여 계산량이 많아진다.[8]

본 논문에서는 탄성 힘(elastic force)을 사용하여, 모바일-매니퓰레이터 로봇의 충돌 회피 동작 제어 방법을 제안한다. 이 방법은 동작 계획과 계획 수정을 반복하여 실시간으로 장애물을 피하는 궤적을 생성하여 로봇이 이를 따라감으로서 충돌을 피하고 목표 형상에 도달하게 한다.

제안된 방법은 가상의 탄성 힘을 이용한 장애물 회피 로봇 구동 방법으로서, 먼저 로봇의 초기 궤적을 구하고, 이 궤적상에 일정한 수의 중간 경유 형상을 선정한다. 이 경유 형상들 사이에 탄성 힘이 작용하여 로봇이 일정한 형상을 유지하는 동시에 장애물을 피하게 된다.

1. Elastic Strip

Elastic Strip 방법은 1993년 Khatib가 제안한 Elastic Band 알고리즘을 모바일 매니퓰레이터에 적용시킬 수 있도록 확장시킨 방법으로 상황에 따라 충돌회피를 위해 각각의 이동경로를 재설정 하게 하는 방식이다.

이동 경로를 설정하기 위해 초기에 로봇의 궤적을 선정한 후, 일정한 수의 경유 형상을 선정한다. 초기 궤적과 경유 형상은 이후의 추돌회피 궤적의 전반적인 형태를 결정하게 된다. 경유 형상의 숫자가 많을수록 로봇의 동작이 부드러워지고 충돌 보장성이 커지는 반면, 계산량이 많아져서 실시간 적용이 어려워질 수 있다. Fig. 12는 초기 궤적 및 경유 형상을 나타내었다

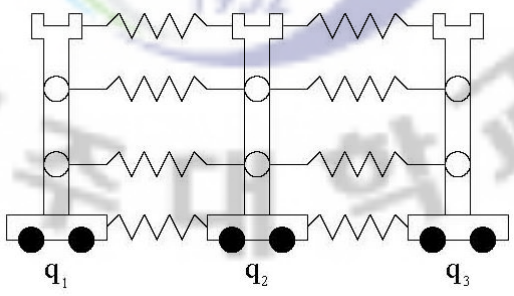


Fig. 12 Initial trajectory and way pdses

이 방식은 스프링의 탄성을 응용한 탄성력(내부 힘)과 장애물에서 발생하는 척력(외부 힘)에 의해 각각의 경로들이 갱신된다.

1.1 내부 힘(Internal Force)

내부 힘은 로봇이 정확한 목적 지점까지 이동할 수 있도록 도와주는 힘으로 Fig. 12과 같이 로봇의 궤적(trajectory)상의 형상(configuration)사이에 가상의 스프링이 연결되어 있다고 가정한다.

로봇의 주요 부분에 제어 점(control position)을 설정하며, 제어 점에는 로봇이 궤적의 이탈을 방지하기 위하여 내부 힘이 작용하게 된다.

내부 힘은 다음과 같이 계산되어 진다.

내부 힘 계산을 위해서는 궤적이 미리 설계되어 있어야 하며 설계된 궤적에 의하여 탄성력이 계산된다.

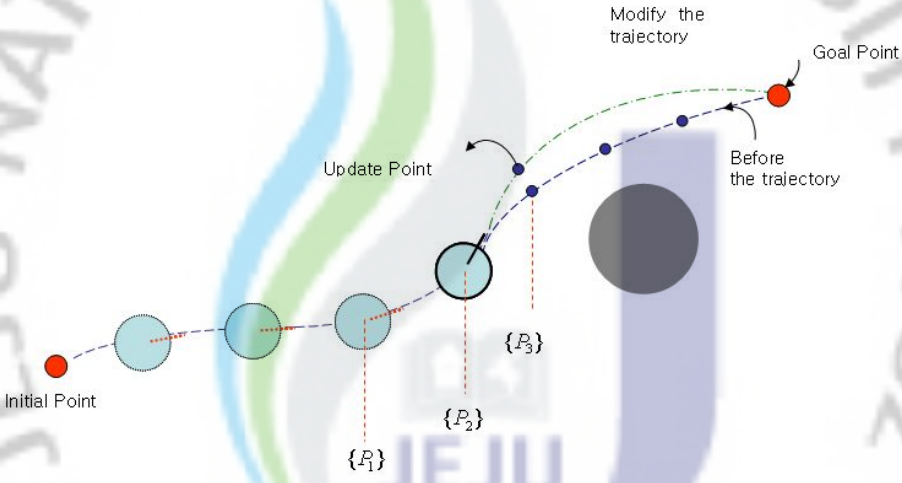


Fig. 13 Transformation procedure of the trajectory

이들 현재 {P2}가 다음 이동해야 할 {P3}과 거리가 길면 이 스프링 힘은 {P3}로 이동을 위해서 더 강한 스프링 힘을 주게 될 것이며 이에 의한 힘이 커지면 미리 계획된 궤적에 가까워지게 될 것이다. 내부 힘은 식(36)과 같이 나타낼 수 있다.

$$f_i = k_c\left(\frac{d_{i-1,i}}{d_{i-1,i+1}}\overline{P_{i-1}P_{i+1}} - \overline{P_{i-1}P_i}\right) \quad (36)$$

k_c = 수축이득

d = 각각의 경로들 사이의 거리

P = 각 경로들의 위치

$\frac{d_i^{j-1}}{d_i^{j-1}+d_i^j}$는 내부 힘에 대한 비례계수이며 이 값의 결정에 따라 계획된 궤적에 빨리 수렴하고의 정도를 결정하게 되며. r는 수정 궤적의 얼마나 부드럽게 수축하느냐에

내부 힘은 고무줄이나 용수철의 특성처럼 경로를 직선모양으로 만드는 성질이 있으며 직선으로 변화하는 동안에도 각 포인트 간의 거리 비율을 이전의 비율과 같이 만들어주며 경로를 변화시킨다. (Fig. 14) 탄성력의 이러한 특성은 어떤 점들의 이동경로를 더 짧고 부드럽게 만들어 준다.

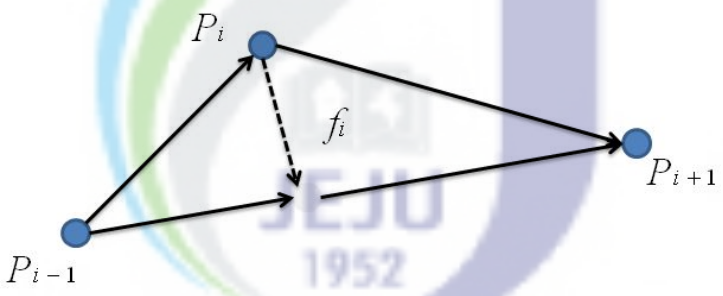


Fig. 14 Direction of Elastic Force

1.2 외부 힘(External Force)

외부 힘은 자율 이동 로봇의 경로계획이나 작업 공간 내에서 장애물과 접촉을 하였을 때 충돌 없이 목적 지점까지 이동하도록 도와주는 힘이다. 이 힘은 장애물로부터 발생하는 척력(인공전위계 힘)으로 장애물과 멀어지는 방향으로 발생하며 궤적에 변형을 일으켜 매끄럽게 궤적을 재 생성하게 한다.

Fig. 15은 모바일 및 매니퓰레이터의 각각의 제어 점 주위에 안전거리에 관해

표현하였다. 안전영역 안에 장애물 감지했을 경우 제어 점과의 가장 가까운 장애물과의 거리를 측정하게 되는데 이때 거리 측정 시 제어 점과 같은 z좌표상에서 센싱된 장애물의 x, y좌표를 고려한 최소 값을 구해야 한다. 안전거리는 작은 값을 넣게 되면 보다 넓은 장애물 회피 영역을 얻을 수 있으나 장애물과의 충돌 가능성 또한 늘어나기 때문에 적절한 값을 지정해야 한다.[4][7][9]

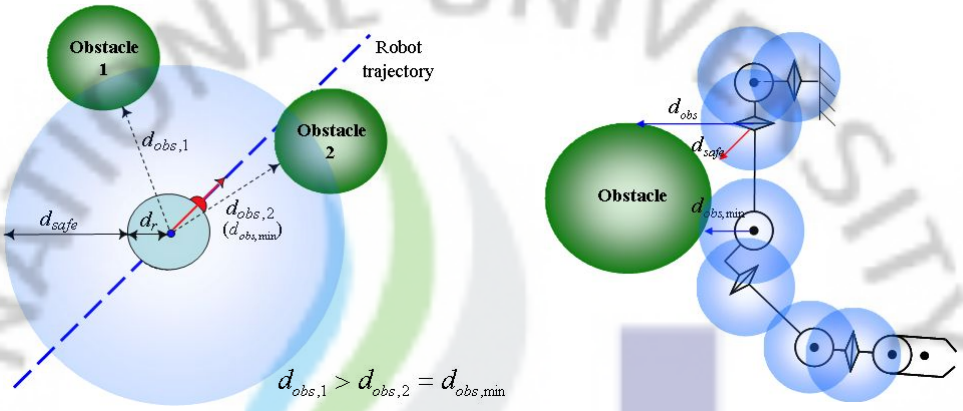


Fig. 15 A safety area of control point

d_{obs}: 로봇과 장애물간의 거리

d_r :모바일의 반지름

d_{safe}: 장애물 회피를 위한 경계 값

d_o: 로봇 중심점에서의 외부 힘 작용 영역

$d_o = d_{safe} + d_r$

$d_{obs} = \min(d_{obs,i}\} = d_{obs,\min}$

이들 안전영역에서 감지된 장애물을 피하기 위해서는 전역 좌표계상에서의 장애물의 위치 값을 계산하여야 한다. 장애물 좌표를 결정하기 위해서 다음과 같이 전역좌표계에 관한 로봇 방향계산으로 인하여 코사인 제2법칙 계산에 의하여 로봇중심으로의 장애물의 위치 및 방향을 얻을 수가 있다. 어느 방향 쪽에 위치하는 지에 관한 정보를 얻을 수가 있으며 장애물의 좌표를 $P_o = [x_o \; y_o \; \theta_o]^T$과 같이

지정한다.

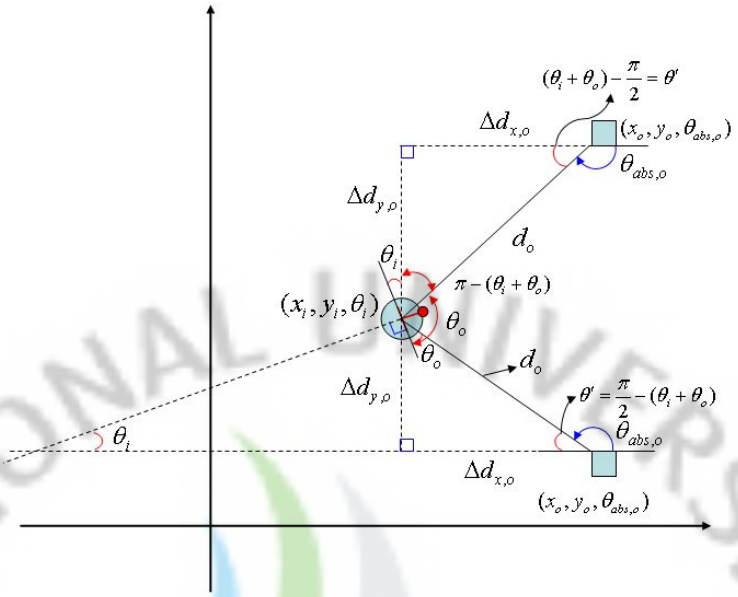


Fig. 16 Compute the position of obstacle

$$
\begin{aligned}
P_o &= [x_o \ y_o \ \theta_o]^T \\
x_o &= x_i + d_o Sin(\theta_i + \theta_o) \\
y_o &= y_i + d_o Cos(\theta_i + \theta_o) \\
\theta_{abs,o} &= -(\pi - \theta')
\end{aligned}
\tag{37}
$$

식 (37)은 전역좌표계에 대한 장애물의 위치 및 방향에 대한 식이다.

척력은 다음과 식(38), (39)과 같이 구해진다.

$$V_{ext}(p)=\begin{cases}\frac{1}{2}k_r(d_0-d(p))^2 & \text{if } d(p)<d_0\\ 0 & otherwise\end{cases} \quad (38)$$

$$F_p^{ext}=-\nabla V_{ext}=k_r(d_0-d(p))\frac{d}{\|d\|} \quad (39)$$

$$d=P_m-P_o,\quad \|d\|=\sqrt{(x_m-x_o)^2+(y_m-y_o)^2+(z_m-z_o)^2} \quad (40)$$

k_r = 척력이득

d_0 = 척력이 영향을 미치기 시작하는 위치까지의 거리

d_p = p점과 장애물 사이의 거리

d = 점p에서 장애물로의 벡터

k_r 값이 클수록 반발력이 더 커지게 된다.

2. 모바일-매니퓰레이터 로봇의 장애물 회피 알고리즘

매 샘플링 시간마다 이전의 샘플링되어 계획된 궤적을 수정한다. 궤적의수정은 자세(posture)들의 수정을 의미한다. 과정은 다음과 같다.

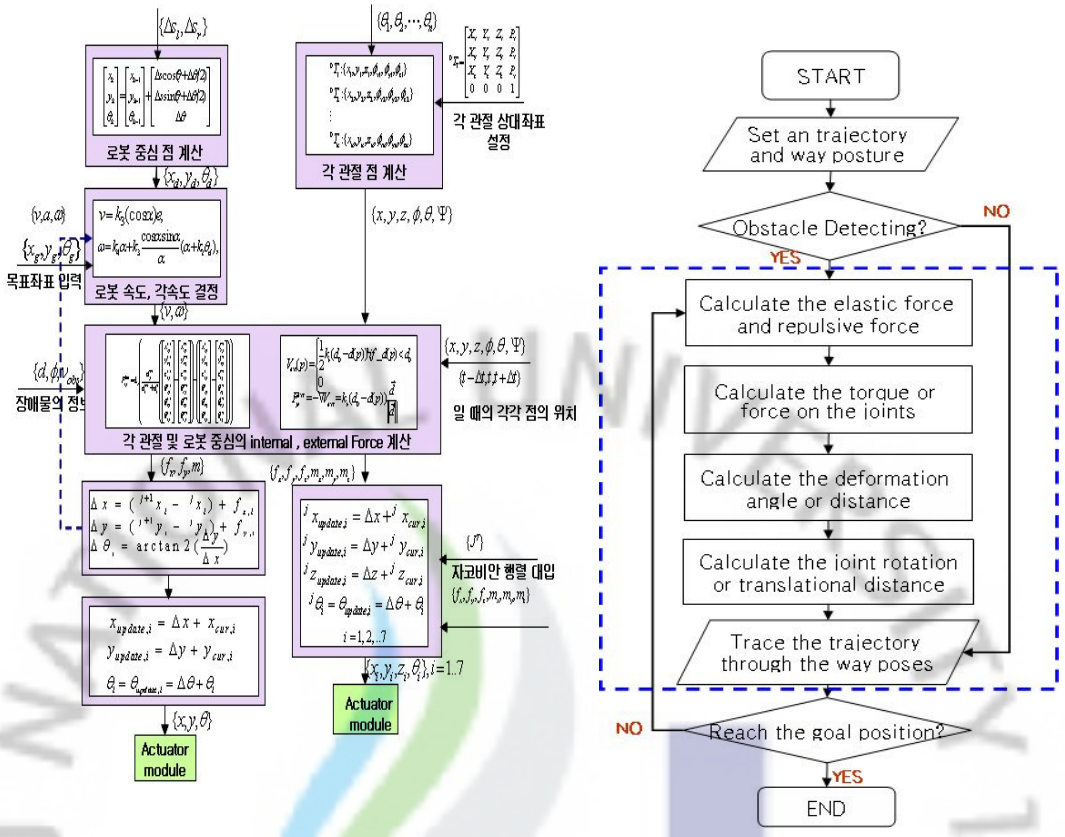


Fig. 17 Collision avoidance Algorithm

(1) 초기 궤적이 계획되고 순차적으로 주어진 자세를 따라 궤적이 주어진다.

(2) 이전의 궤적에 의해 궤적이 수정된다.

ⓐ 로봇의 제어 점(control point)에 가해지는 내부 힘과 외부 힘을 계산한다.

ⓑ 제어 점들에 가해지는 힘에 대해 로봇 조인트 및 방향의 토크 값을 계산한다.

ⓒ 조인트 및 로봇 방향의 토크 값에 대한 이동할 각도 및 거리를 계산한다.

ⓓ ⓒ에서 계산 된 조인트, 로봇 방향의 각도 변형에 의해 이동 및 회전한다.

(3) 다음 자세를 따라 궤적을 변형한다.

(4) 만약 로봇이 원하는 목표 자세에 도달하였는지 확인한 후 충분히 목표 자세에 도달하지 않았다면 순서를 멈춘 후 다시 (2)단계로 돌아간다.

2.1 외부 힘 변형(External Force Modification)

기존에 개발된 Elastic strips 기반의 obstacle avoidance 알고리즘에서는 Fig. 16과 같은 문제점이 발생하였다. 로봇의 최종점의 방향과 센서의 가장 근접한 점의 방향과 일치를 할 때 로봇은 Fig. 18(a)와 같이 물체와 충돌을 하였으며, 다중 물체를 회피할 때 Fig. 18(b)와 같이 로봇의 경로 생성에 어려움이 발생하였다.

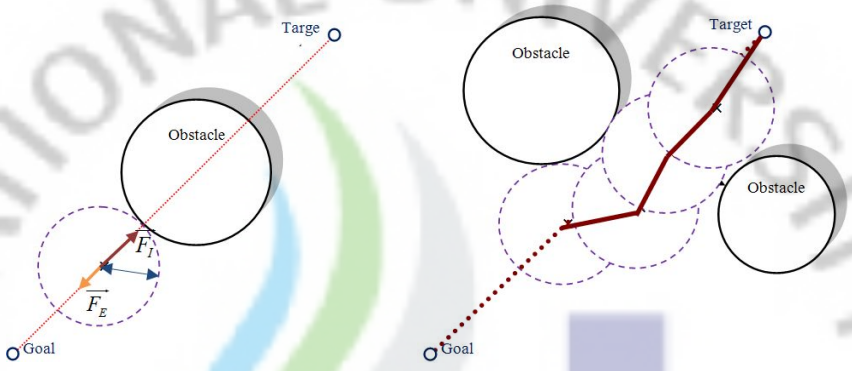


Fig. 18(a;b) Elastic strips based on the nearest obstacle

본 연구에서는 이러한 문제점을 해결하기 위하여 Modified Elastic strips 기반의 충돌 회피 알고리즘을 개발 하였다. Modified Elastic strips에서 고려하고 있는 내부 힘은 기존의 방법에서 제안하는 힘과 동일하며, 외부 힘은 두 개의 외부 힘을 고려하였다. 즉, 오른쪽 방향의 근접 점과 왼쪽 방향의 근접 점에 의해서 발생하는 외부 힘이다. 이들 외부 힘에 대하여 식(41)과 같다.

$$\overrightarrow{{}^{L(R)}d} = \overrightarrow{P_m} - \overrightarrow{{}^{L(R)}P_o}$$

$$\| {}^{L(R)}d \| = \sqrt{({x_m} - {}^{L(R)}x_o)^2 + ({y_m} - {}^{L(R)}y_o) + ({z_m} - {}^{L(R)}z_o)^2}$$

$$\overrightarrow{{}^{e}f_i(L/R)} = -\nabla V_{ext} = k_r(d_0 - {}^{L(R)}d_{obs,\min})\frac{\overrightarrow{{}^{L(R)}d}}{\| {}^{L(R)}d \|}$$

$$\overrightarrow{F_i} = \overrightarrow{{}^{e}f_i(L)} + \overrightarrow{{}^{e}f_i(R)} \qquad (41)$$

3. 궤적의 변형

각 제어점에 탄성력과 인공전위계 힘이 작용하면 이에 로봇의 관절 및 구동부에 토크가 발생한다. 본 논문에서는 Fig. 19과 같은 구조를 갖는 모바일-매니퓰레이터를 대상으로 Elastic strip 방법을 적용하였다.

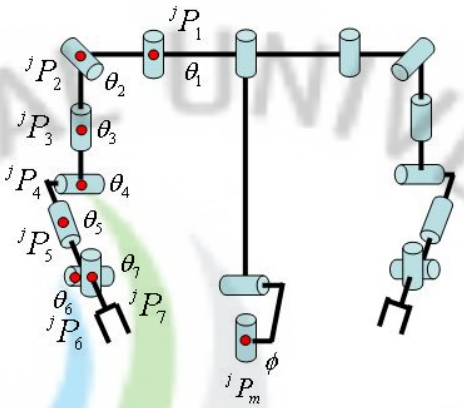


Fig. 19 ^jP_irepresents the control points on the body.

앞서 계산된 힘의 최종 힘은 내부 힘과 외부 힘의 벡터 합으로 계산된다. 식(42)에서 계산되어진 힘에 의하여 궤적의 변형을 일으킨다.

$$\overrightarrow{F_i} = \overrightarrow{{}^e f_i(L)} + \overrightarrow{{}^e f_i(R)} + \overrightarrow{{}^r f_i} \tag{42}$$

모바일은 식(43)과 같이 계획된 궤적$j, j+1$과 이동 값과 계산된 최종 로봇에 작용하는 힘의 합으로 이동 좌표값이 정해진다.

$$\begin{aligned} \triangle x &= ({}^{j+1}x_i - {}^jx_i) + F_{x,i} \\ \triangle y &= ({}^{j+1}y_i - {}^jy_i) + F_{y,i} \\ \triangle z &= ({}^{j+1}z_i - {}^jz_i) + F_{z,i} \end{aligned} \tag{43}$$

$x_{cur,i}$, $y_{cur,i}$, $z_{cur,i}$가 현재 로봇의 위치라면 최종적으로 모바일이 이동해야 할 좌표는 식(44)과 같다.

$$\begin{aligned} x_{update,i} &= \triangle x + x_{cur,i} \\ y_{update,i} &= \triangle y + y_{cur,i} \\ z_{update,i} &= \triangle z + z_{cur,i} \end{aligned} \tag{44}$$

모바일의 이동과 동시에 로봇의 방향을 결정해야 한다. 모바일의 이동 방향 결정은 식(45)에 의하여 로봇 방향이 결정된다.

$$\triangle\theta_m = \arctan2\left(\frac{\triangle y}{\triangle x}\right) \tag{45}$$
$$\theta_m = \theta_{update,m} = \triangle\theta_m + \theta_m$$

Fig. 18에서 내부 힘과 외부 힘은 점P_i에 가해지는 힘 f_i에 의해 발생하는 매니퓰레이터 토크들은 자코비안 행렬을 이용하여 구한다.

$$\Gamma = J^T f \tag{46}$$

머니퓰레이터의 조인트 회전식은 모바일과는 다르게 회전 및 이동에 관한 토크 값을 식(47)에서 자코비안 행렬과의 계산에 의하여 최종 조인트에서의 회전 운동 식을 얻어 낼 수 있다. 머니퓰레이터의 제어 점 $P_1, P_1, \cdots, P_7$에 작용하는 힘 $F_i = [f_x\ f_y\ f_z\ m_x\ m_y\ m_z]^T$을 식(33)에서 계산된 자코비안 행렬과 계산으로서 각 조인트에 주어지는 토크 값 $\Gamma_1 = [\tau_1\ \tau_2 \cdots \tau_7]^T$이 결정된다.

이때 주의해야 할 점은 각 조인트에 토크 값은 P_1에서 계산된 F_1에 의하여 머니퓰레이터의 각 조인트의 정적 평형을 위하여 계산된 토크 값이라는 것이다. 이 F_1와 같이 나머지 $F_2, F_3, \cdots, F_7$의 값을 계산하여 얻어진 토크 값은 $\Gamma_2, \Gamma_3, \cdots, \Gamma_7$이 되며 최종적으로 조인트에서 받는 토크 $\Gamma = [\tau_1\ \tau_2 \cdots \tau_i]^T$는 식 (48)과 같다.

$$\tau_i^j = J_i^T F_i$$

$$\begin{pmatrix}\tau_1\\ \tau_2\\ \tau_3\\ \tau_4\\ \tau_5\\ \tau_6\\ \tau_7\end{pmatrix} = \begin{pmatrix} J_{11} & J_{12} & J_{13} & J_{14} & J_{15} & J_{16}\\ J_{21} & \ddots & & & & J_{26}\\ J_{31} & & \cdots & \cdots & & J_{36}\\ J_{41} & & & & & J_{46}\\ J_{51} & & \cdots & \cdots & & J_{56}\\ J_{61} & & 0 & & \ddots & 0\\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} F_{i,x}\\ F_{i,y}\\ F_{i,z}\\ m_{i,x}\\ m_{i,y}\\ m_{i,z}\end{pmatrix} \tag{47}$$

$$\Gamma_{[7\times1]} = \sum J_7^T F_7 = \Gamma_1 + \Gamma_2 + \Gamma_3 + \cdots + \Gamma_7, \qquad i = 1, 2, 3, \cdots, 7 \tag{48}$$

관절에 가해지는 토크에 비례하여 각 관절을 회전시켜서 각 샘플링 시각마다

경유 형상을 구한다. i번째 관절의 회전각 $\triangle\theta_i$는 식(49)와 같이 구해진다.

$$\triangle\theta_i = \alpha_i \tau_i, \quad (i = 1, 2, \cdots, 7) \tag{49}$$

여기서 α_i는 i번째 조인트의 토크 Γ_i를 위한 회전의 비례상수이다. 관절의 이동량이 결정되어 각 경유 형상들이 결정되면 로봇은 이 경유 형상들을 차례대로 추적하여 장애물을 피하면서 목표 형상을 향하여 동작한다.

4. 실시간 충돌 회피 컴포넌트 및 Kernel 모듈

4.1. 실시간 충돌 회피 컴포넌트

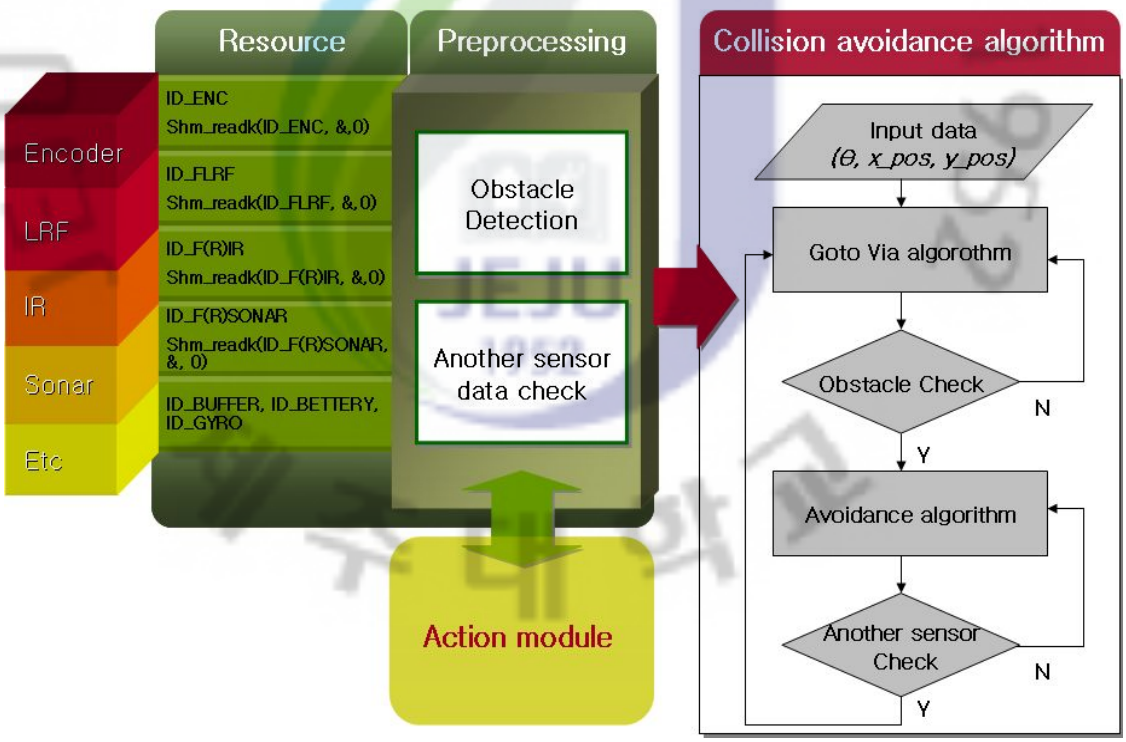


Fig. 20 Real time obstacle avoidance component

로봇의 실시간 충돌 회피 컴포넌트의 구조는 FIg. 20과 같다. 로봇에 있는 Encoder, URF, IR, Sonar 등의 센서는 내부적으로 각 센서별 ID_ENC, ID_FLRF, ID_FIR, ID_RIR 등의 아이디를 지니고 있고, RTAI 명령어 모듈에 있는 Shm_read() 명령어와의 조합으로 원하는 센서의 값을 얻을 수 있다. 로봇이 이동하면서 주기적으로 Shm_read(ID_FLRF) 명령을 사용하면 전방의 LRF 센서의 값을 받아들이게 되고, 로봇은 받아들인 전방 LRF 센서의 값을 이용하여 전방의 장애물 유무를 판단하게 된다. 이 때 센서의 값을 순차적으로 확인하면서 장애물이 발견되지 않을 경우, 기존의 알고리즘에 의해 주행하던 경로를 그대로 따라가면 되지만, 만약 주행 중 장애물을 발견하고 센서에서 그 값을 확인했을 경우에는 이벤트로 장애물을 발견했음을 알리고, 본 연구에서 구현하는 충돌 회피 알고리즘을 통해 새로운 경유 좌표를 도출해낸다.

충돌 회피 알고리즘은 센서에서 입력받은 장애물까지의 거리와 각도를 가지고 장애물 체크 여부에 따른 서로 다른 주행알고리즘을 사용하는 것을 나타내고 있다. 입력 데이터를 이용하여 기본 주행 명령어인 Goto Via 알고리즘을 이용하여 주행을 하게 되고, 장애물을 발견하지 않을 경우 계속 Goto Via al알고리즘을 수행하다가 장애물이 발견되면 새로운 Avoidance sensor 명령을 실행하고 있다. 이와 같은 루프가 목적지 도착 전까지 계속 반복된다.

4.2. elastic_strip_func 컴포넌트 함수

```
static void elastic_strip_func()
// main function of elastic strip.
{
    // Obstacle Check
    short angle = 0, distance = AVOIDANCE_RANGE;
    if(!obstacle_check(&angle, &distance))
    {
        return;
    }

    if(target_pose->cur >= target_pose->last)
    {
        return;
    }

    //--> Detect Obstacle : Working

    // Init
    struct mo_pose_data_struct posStart, posVia, posEnd, posTarget;
    struct mo_pose_data_struct inter, exter, elastic, posPass;

    // Get Position
    posStart.x = cur_pose->x;
    posStart.y = cur_pose->y;
    posStart.m_rad = cur_pose->m_rad;

    if(!GetPosition(&posTarget, target_pose->last - 1))
    {
        return;
    }

    // Calc Moving Position : Start -> Pass -> End
    get_posi(&posVia, posStart, posTarget, max_velocity, 1);
    get_posi(&posEnd, posVia, posTarget, max_velocity, 1);

    // Trans : Moving Position
    internal_force(&inter, posStart, posVia, posEnd);
    external_force(&exter, posVia, angle, distance);
    add_matrices(&elastic, inter, exter);

    // Create : Pass Position
    modify(&posPass, posVia, posEnd, elastic);

    // Sort Buffer : Insert Pass Position
    SortPosition(posPass, posTarget);

    //
    int i = 0;
    struct mo_pose_data_struct temp;
    for(i=target_pose->cur; i<target_pose->last; i++)
    {
        temp = target_pose->point[i];
    }
}
```

Fig. 21 elastic_strip_func function

로봇의 동작과 관련하여 입력되는 센서 값과 좌표 등은 공유메모리(Share Memory) 상에서 사용되어 진다. 하지만 사용자가 입력한 목적지의 값과 같은

고유의 데이터에 대해서는 공유메모리를 따로 사용하지 않고, 프로그램 내부에 임의의 버퍼를 생성하고 로봇이 진행해야하는 고유 좌표값을 저장하고 사용할 필요가 있다.

elastic_strip_func 함수는 현재 로봇의 좌표와 버퍼에 있는 목적지 좌표를 얻어와서 장애물을 회피하기 위한 최단경로의 경유점을 하나 생성하고, 생성된 좌표를 기존에 있는 버퍼에서 로봇이 다음좌표 이동을 위해 읽어야 하는 위치에 삽입시키게 된다. 따라서, elastic_strip_func 함수에 의해 로봇이 다음에 이동해야 하는 점이 갱신되게 되므로, 로봇은 기존처럼 버퍼에 있는 좌표를 순서대로 읽고, 그 좌표대로 진행한다.

4.3. obstacle_check 컴포넌트 함수

회피 알고리즘에 의해서 로봇이 목적지로 이동할 때, 로봇은 주기적으로 센서를 이용하여 장애물을 식별할 필요가 있다. shm_read 명령어를 사용하여 공유메모리(Share Memory)로 입력되는 센서 데이터의 값을 읽어 들이고, 그 값을 해석하여 장애물의 식별여부를 확인하고, 추가적으로 장애물이 인식된 거리와 각도에 대한 정보를 반환한다.

```
int obstacle_check(unsigned short *angle, unsigned short *distance )
// check obstacle and get the angle and distance with the nearest obstacle.
{
    int i, nFlags = 0;
    unsigned short temp;

    shm_readk(ID_FLRF, &flrf_data, 0);

    for ( i=31; i<331; i++ )
    {
        temp = flrf_data.data[i];
        if (temp < AVOIDANCE_RANGE )
        {
            *distance = temp;
            *angle = i / 2;
            nFlags = 1;
            break;
        }
    }

    return nFlags;
}
```

Fig. 22 obstacle_check function

4.4. GetPosition 컴포넌트 함수

로봇의 이동에 있어 목적지를 가지고 이러한 고유값을 저장하는 장소를 버퍼라고 했을 때, elastic_strip_func 함수에서는 새로 생성한 장애물 회피 경유점을 현재 로봇의 진행 위치에 따라서 바로 다음에 로봇이 읽게 될 좌표에 삽입할 필요가 있다. 이 때 새로운 장애물 회피 경유점을 생성하기 위해서는 현재 버퍼에 있는 목적지 좌표와 현재 로봇의 좌표 등의 정보를 얻을 필요성이 있고, 따라서 이러한 버퍼에서 좌표를 읽어올 때 편의성을 좋게 하기 위해 GetPosition 함수를 추가하게 되었다. GetPosition 함수는 데이터를 반환받을 변수와 읽어 들일 버퍼의 위치에 대한 매개변수를 가진다.

```
int GetPosition(struct mo_pose_data_struct *pData, int nIndex)
{
    if((nIndex < 0) || (nIndex >= NUM_MAX_PT))
    {
        return 0;
    }

    *pData = target_pose->point[nIndex];

    if(((*pData).x == 0) && ((*pData).y == 0))
    {
        return 0;
    }

    return 1;
}
```

Fig. 23 GetPosition function

5. 실시간 충돌회피 모듈 통합

5.1. 장애물 확인 및 좌표 버퍼 생성 알고리즘

5.1.1. 좌표 버퍼 생성 알고리즘

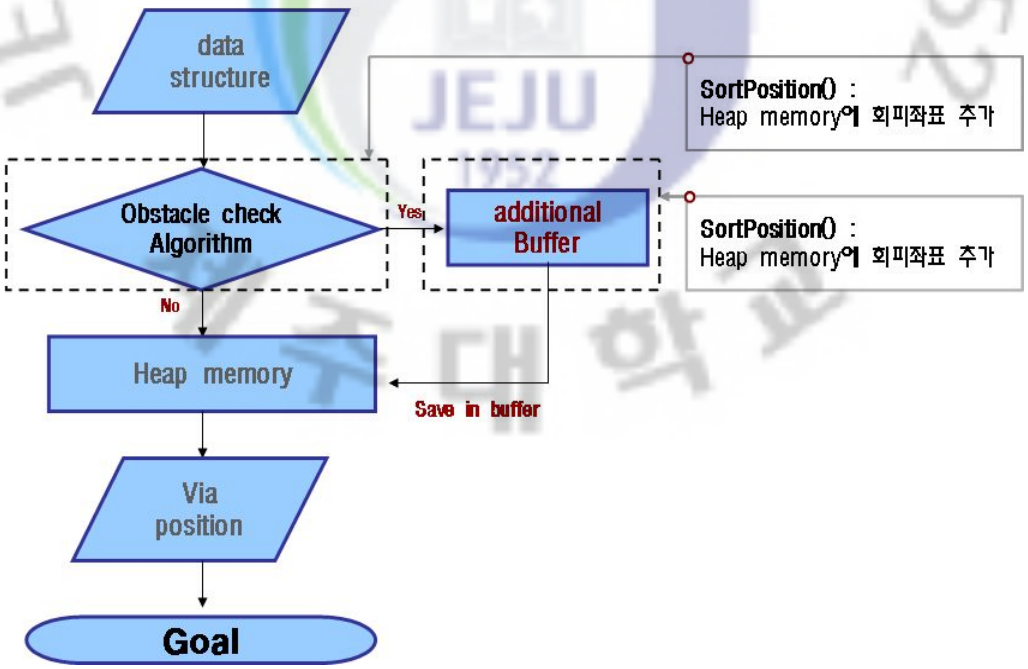


Fig. 24 장애물 확인 및 좌표 버퍼 생성 알고리즘

장애물 확인 및 장애물 회피 알고리즘 적용시 최적화된 방법으로 장애물을 확인하고 생성된 좌표를 모듈에 적용하기 위해서 힙-메모리 내에 버퍼를 생성, 회피 좌표를 추가하여 회피 알고리즘을 최적화 시킨다. 이것은 Fig. 24와 같이 Resource 내부의 Data structure에서 센서 모듈 정보를 읽고 장애물 확인 알고리즘을 통하여 장애물 여부를 판단하게 되는데, 장애물 판단 정보와 알고리즘을 통해 생성된 좌표정보를 좌표 버퍼를 생성하여 우선 저장해 놓고 힙-메모리에 적용하는 구조이다. 힙-메모리 내부에 추가적인 버퍼는 힙-메모리의 접근성과 힙-메모리 내부에서 좌표 추가 및 삭제가 용이하다.

5.1.2. 충돌 에러 보상

Modified Elastic strips 기반의 충돌 회피 알고리즘의 적용시 Via point는 로봇의 충돌반경으로 설정되어 있다. 이러한 충돌 반경은 로봇의 치수적인 직경만을 고려한 것이고, 모터와 엔코더, 센서의 반응시간 및 지연시간은 고려한 것이 아니므로 로봇의 회피 알고리즘 수행시 충돌하는 경우가 발생한다. Fig. 25와 같이 로봇의 충돌하는 현상을 방지하기 위하여 Via point 생성시 SortPosition() 함수를 사용하여 버퍼를 생성하고 이렇게 생성된 버퍼를 통해서 Via point를 특정 거리의 값으로 필터링 한다. 필터링 거리는 변수로 지정하여 실험 조건과 센서 조건에 맞게 설정 가능하고 SortPosition()함수 내부에서 버퍼 생성과 Via point 필터링 및 힙-메모리 삽입 기능을 추가하여 구현하였다.

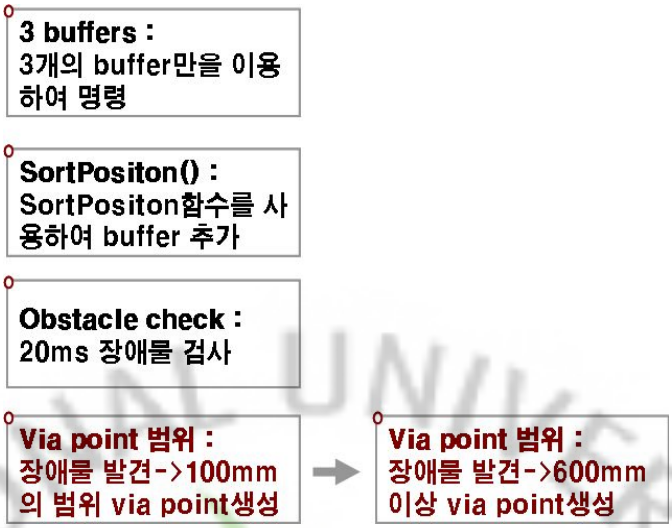


Fig. 25 Via point 생성 및 필터링

5.1.3. SortPosition 컴포넌트 함수

로봇은 충돌 회피 알고리즘에 의해서 장애물을 발견하면 새로운 경유점을 생성하게 된다. 이 새로운 경유점으로 로봇이 이동하기 위해서는 기존 로봇이 목적지로 이동하기 위해 사용하고 있는 경로 버퍼에 삽입할 필요가 있다.

Fig. 26의 SortPosition 함수는 IsSort에서 사용하도록 필터링 된 경유점 좌표를 기존 로봇이 사용하는 버퍼에 사용할 수 있도록 버퍼의 사용을 잠시 멈추고, 버퍼의 좌표를 재배열 한 후 다시 버퍼 사용을 활성화시켜 로봇이 정상 동작하도록 한다.

```
int SortPosition(struct mo_pose_data_struct posPass, struct mo_pose_data_struct posGoal)
{
    int i = 0, nIndex = target_pose->cur+1;

    //
    struct mo_pose_data_struct temp;
    temp = target_pose->point[nIndex-1];

    if(!IsSort(posPass, temp))  return;

    // Start Working
    rt_sem_wait( &target_pose->mutex );

    // Clear Buffer
    for(i=nIndex; i<target_pose->last; i++)
    {
        target_pose->point[i].x = 0;
        target_pose->point[i].y = 0;
        target_pose->point[i].m_rad = 0;
        target_pose->point[i].flag = 0;
    }

    // Via Setting
    target_pose->point[nIndex] = posPass;
    target_pose->point[nIndex].flag = 0;
    nIndex++;

    // Goal Setting
    target_pose->point[nIndex] = posGoal;
    target_pose->point[nIndex].flag = 1;
    target_pose->last = nIndex + 1;

    // End Working
    rt_sem_signal( &target_pose->mutex );

    return 1;
}
```

Fig. 26 SortPosition function

5.1.4. IsSort 컴포넌트 함수

```
int IsSort(struct mo_pose_data_struct data1, struct mo_pose_data_struct data2)
{
    int tempX, tempY, nLenght = 500;

    tempX = data2.x - data1.x;
    if(tempX < 0)   tempX *= (-1);

    tempY = data2.y - data1.y;
    if(tempY < 0)   tempY *= (-1);

    if((tempX < nLenght) || (tempY < nLenght))
        return 0;

    return 1;
}
```

Fig. 27 IsSort function

로봇은 장애물을 회피함에 있어서 빠른 주기로 장애물을 읽고 새로운 경유점을 생성하고 사용하는 과정을 반복하기 때문에 불필요한 움직임이 생기게 된다. 따라서 로봇이 장애물을 회피함에 있어 자연스러운 회전이 가능하도록 불필요한 경유점은 필터링할 필요가 있다. Fig. 27의 IsSort 함수는 기존에 로봇이 이동해야할 좌표와 새로 생성된 경유점 좌표를 비교하여 새로 생성된 경유점이 기존 좌표를 대체할지 여부를 결정하는 필터링 함수이다.

5.2. 메모리 좌표 재설정 및 재배열 알고리즘

로봇이 장애물을 회피할 때, 빠른 주기로 장애물을 체크하고, 새로운 경유점을 생성 후 버퍼에 삽입하기 때문에, 다량의 데이터 사용으로 인한 로봇의 부자연스러운 움직임에 대한 문제를 해결하기 위해서 새로운 경유점을 기존 좌표와 비교하여 버퍼의 삽입여부를 결정하는 필터링 함수가 필요하다. Fig. 28은 이러한 메모리 좌표 재설정 및 재배열을 위한 알고리즘과 구현된 함수를 나타낸다. avoid_func 통합 함수와 compare_position 함수는 장애물을 체크 할 경우 회피이동 좌표를 계산하고 추가된 버퍼 안에서 좌표를 재설정 및 재배열을 하는 기능을 한다. 또한 이렇게 재배열된 좌표는 다시 필터링 과정을 거치고 출발 및 목표

좌표를 갱신한다. 그림 18은 이러한 compare_position 함수를 나타내고 compare_position 함수는 IsSort 함수에서 거리 필터링 과정을 추가하여 개발하였다.

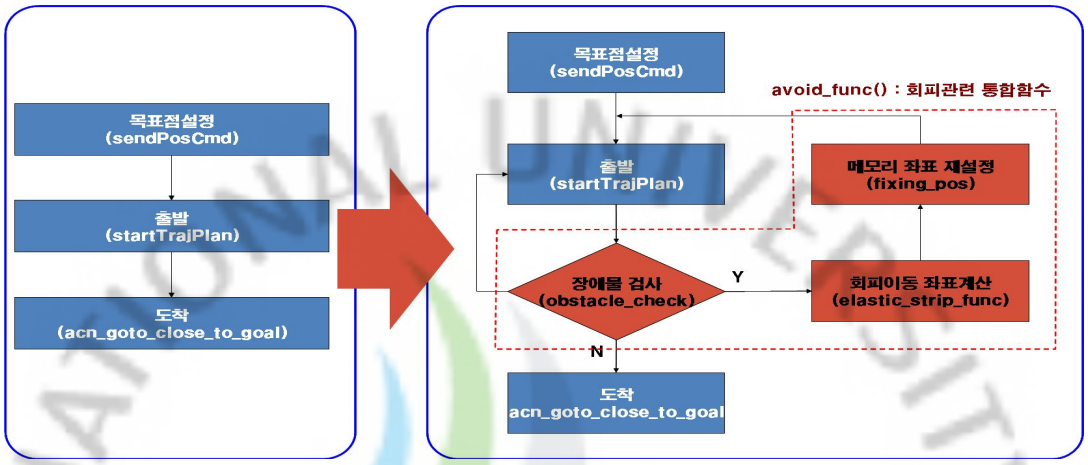


Fig. 28 메모리 좌표 재설정 및 재배열 알고리즘

Ⅳ. 시뮬레이션

1. 알고리즘 시뮬레이션

장애물의 위치에 따른 로봇의 충돌 회피 동작을 Fig. 29와 같이 시뮬레이션 하여 수행하였다. 먼저 장애물의 사이의 거리에 따른 로봇의 충돌 회피 궤적의 변화를 관찰하였다. 로봇의 궤적을 확인한 이에 적정 이득을 선정하여 불규칙적으로 흩어진 경우에 관하여 로봇의 충돌 회피에 관한 궤적에 미치는 영향을 관찰하였다.[4][6]

이 시뮬레이션을 바탕으로 Elastic Strip에서 사용되는 게인 값의 적절한 값을 k_r=0.5, k_c=0.3으로 정하였다.

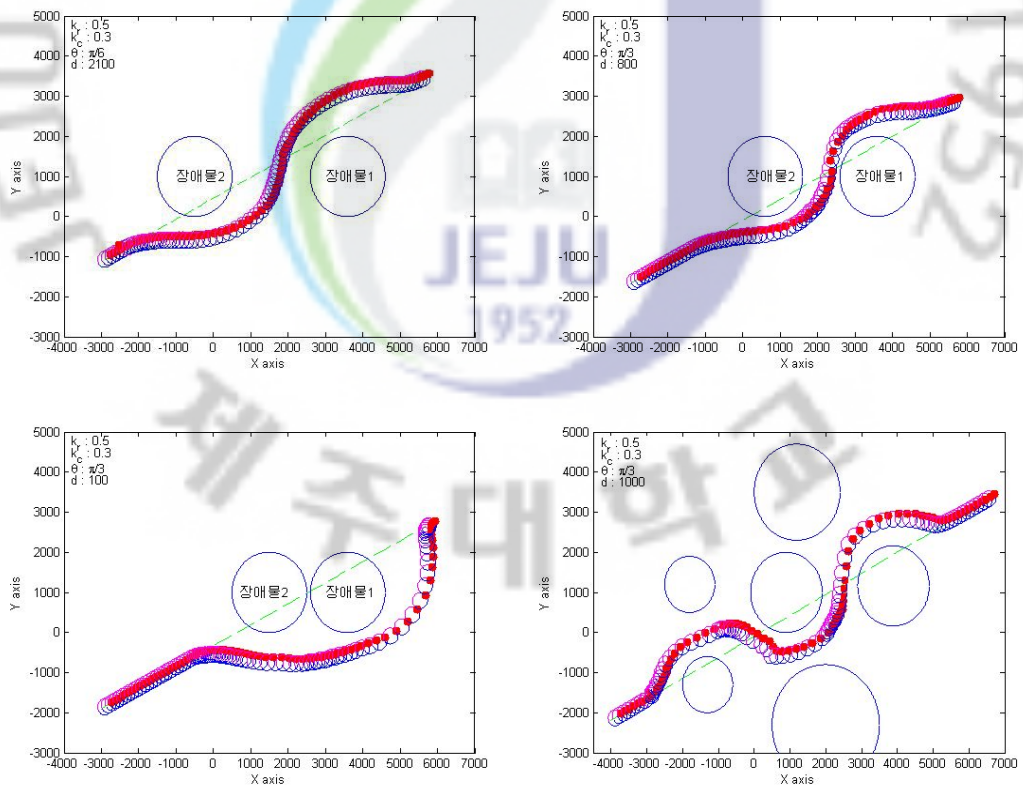


Fig. 29 로봇의 불규칙한 장애물 회피 시뮬레이션 결과

위 시뮬레이션 결과를 바탕으로 3D 모델링한 로봇(Fig. 30)을 갖고 Fig. 31과 같이 단일 장애물과 다중 장애물시의 실험을 행하였다. 또한 모발일 로봇의 시뮬레이션 결과 데이터를 이용하여 이를 매니퓰레이터에 적용하여 Fig. 32와 같이 시뮬레이션을 행하였다.

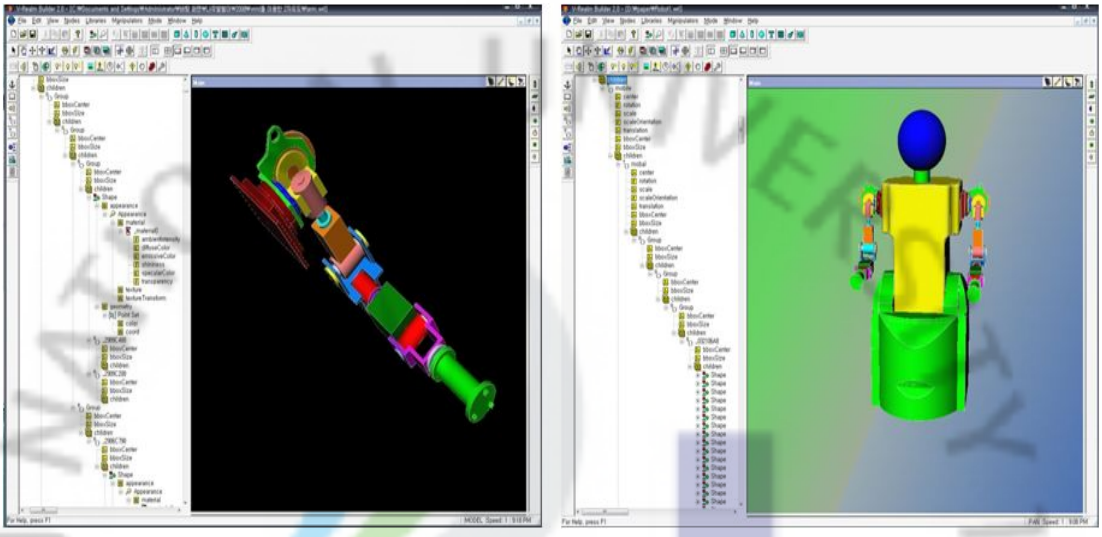


Fig. 30 3D modeling

Fig. 31과 Fig32에서 3D 시뮬레이션을 통하여 확인한 바와 같이 본 논문에서 제안한 충돌회피 알고리즘과 회피 모듈이 실제 적용이 가능하다는 것을 확인하였다.

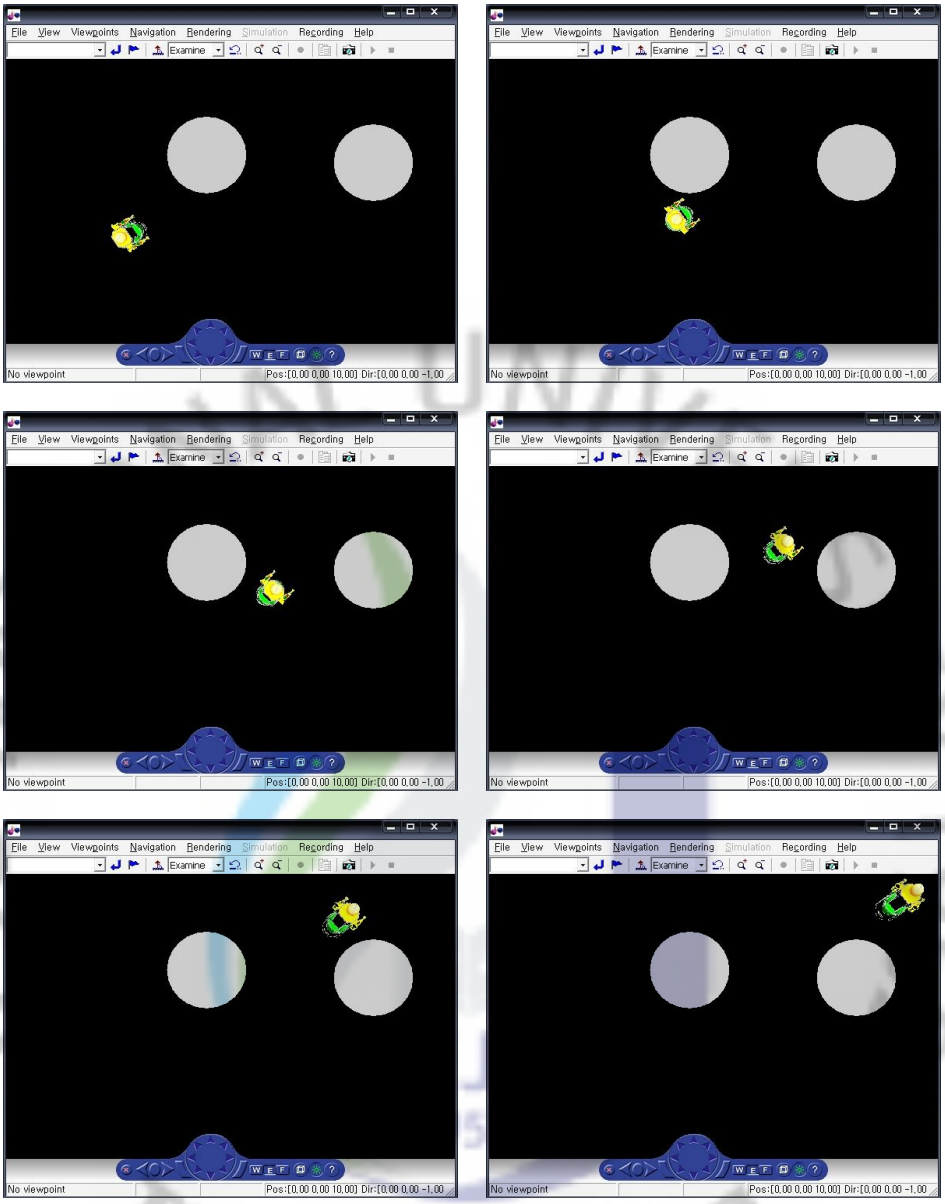


Fig. 31 3D 시뮬레이션 결과

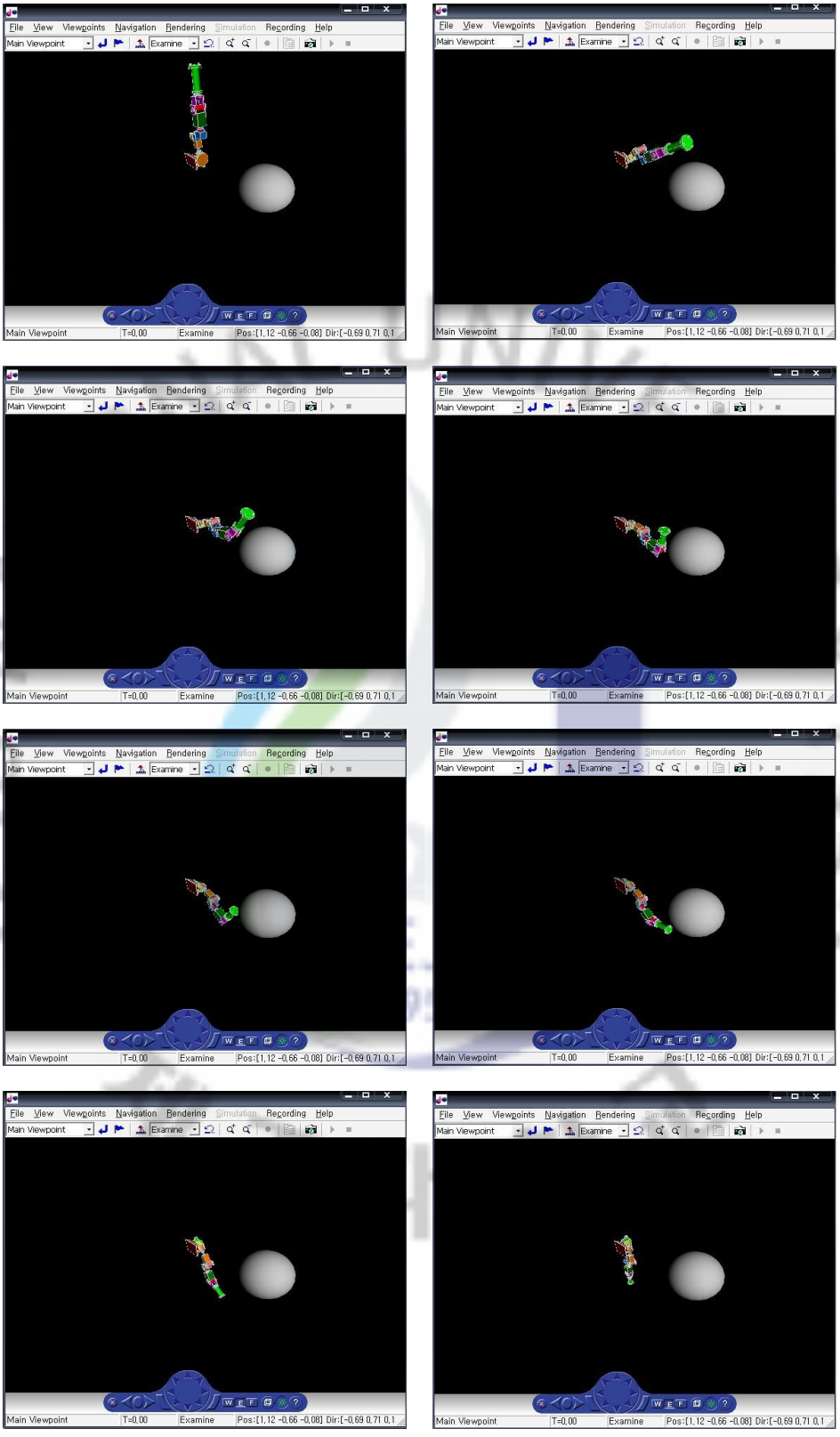


Fig. 32 매니퓰레이터 시뮬레이션 결과

2. 모바일 로봇의 충돌회피 실험

본 실험은 로봇의 주변에 장애물을 설치하고, 사용자가 입력한 목표까지 로봇이 장애물을 회피하며 자율 주행하는 실험이다. Fig. 33은 로봇의 이동 경로상에 장애물이 있을 경우 로봇이 잘 주행 하고 있음을 보여 준다. 로봇이 주행을 하다가 센서값에 의해 장애물이 감지되면, avoidance algorithm에 의해 경유점을 생성 후, 로봇은 새롭게 추가된 좌표를 읽고, 장애물을 회피하기 위해 좌측으로 방향을 전환하고 있다. 장애물을 회피하는 동안에도 내부적으로는 센서값에 의해 새로운 경유점을 생성하고 있지만 중요성이 낮은 경유점 정보에 대해서는 필터링 되어 버퍼에는 추가되지 않고, 중요성이 높은 경유점 정보에 대해서만 이동경로 버퍼에 삽입된다. 장애물을 완전히 회피했을 경우는 경유점 위치에 도달한 것이고, 센서도 더 이상 장애물을 인식하지 못하므로 초기의 로봇의 목적지로 목표를 정하고 방향을 전환한다. 장애물이 완전히 사라지면 목표위치를 향하여 주행하다 목표지점에 도착하는 것을 확인하였다.

Fig. 34은 로봇의 주행 중 로봇의 이동 경로상에 사람과 같은 이동 물체가 있을 경우 로봇이 이동 중인 장애물을 인식하고 회피하여 주행 하고 있음을 보여 준다.

Fig. 33 로봇의 장애물 회피 실험

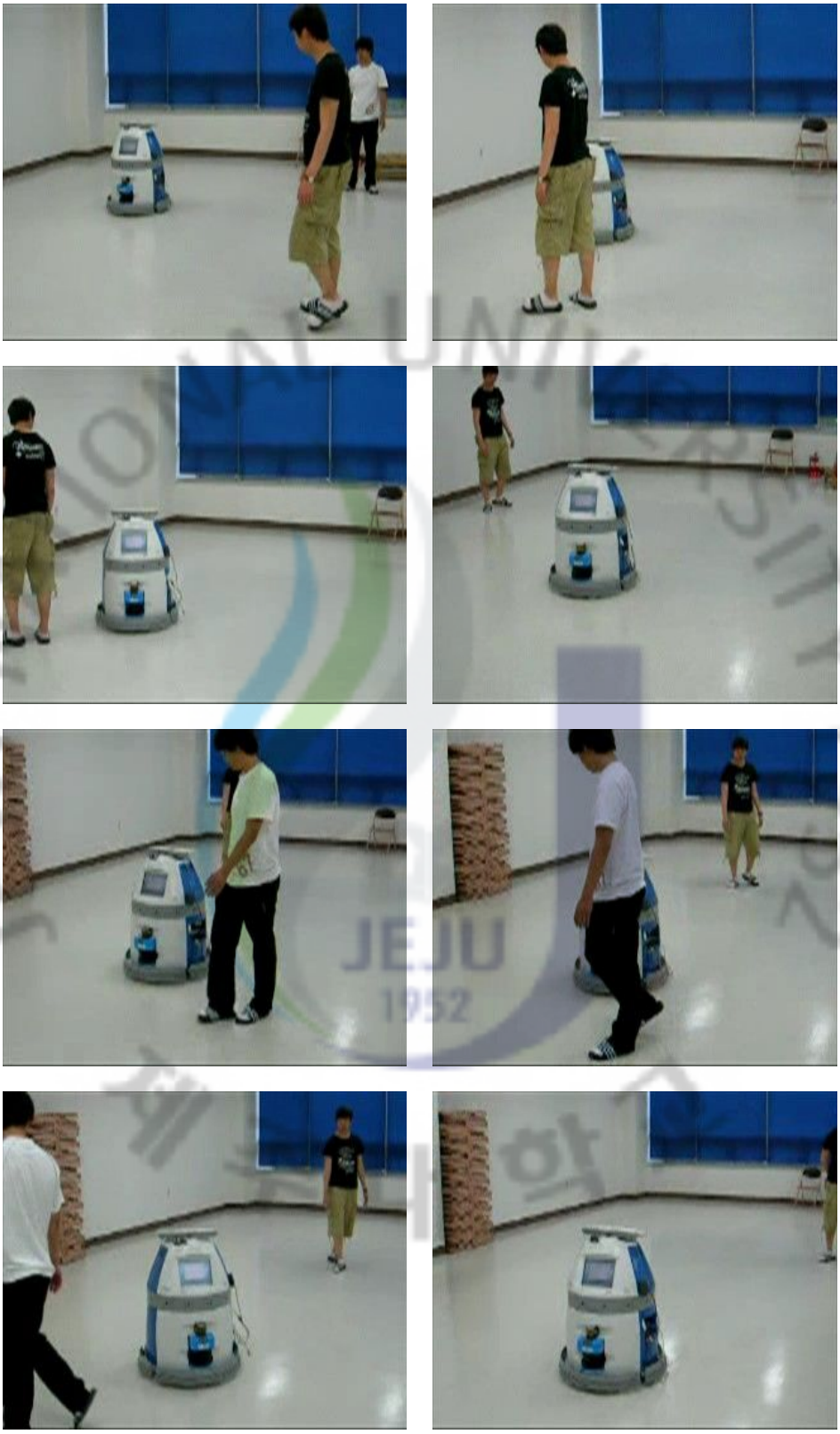


Fig. 34 로봇의 이동 중인 장애물 회피 실험

Ⅴ. 결론

본 논문은 지능형 로봇에 충돌 회피 알고리즘을 추가하여 로봇이 장애물을 실시간으로 인식하고 그 장애물을 회피할 수 있도록, 모발-매니퓰레이터에 대한 실시간 충돌 회피 모듈을 개발하는 것을 목표로 연구개발을 진행하였다. 모발-매니퓰레이터에 대한 실시간 충돌 회피 알고리즘을 개발하였다. 그리고, 개발된 알고리즘을 이용해서 실시간으로 시뮬레이션을 구현한 후, 완성된 결과를 이용하여 실제 로봇에 적용하였다. 마지막으로 기존 로봇 kernel 프로그램에 모듈을 통합 적용한 후, 구축된 시스템으로 실제 모듈 검증 테스트를 시도했다.

테스트 결과 로봇은 장애물을 실시간으로 인지하고 내장된 알고리즘에 따라 좌표를 변경하며 장애물을 회피하는 결과를 보여주었다. 다만, 엔코더와 센서 등의 하드웨어적인 스펙과 컴퓨터 시뮬레이션으로 확인했던 데이터 적용능력과의 차이로 인해서 실시간으로 생성되는 수많은 경유점 중에서 적용 중요도에 따른 몇 가지씩만 필터링하여 실제 적용하게 되었고, 따라서 실시간이라고는 하기 힘든 결과라 보여지는 부분은 아쉬운 부분으로 남아 앞으로 개선해야 할 방향이라고 할 것이다.

Ⅵ. 참고문헌

[1] 전성용, 2006. 2, “자율이동로봇의 실시간 주행을 위한 반사층의 제어구조”, 부산대학교 석사학위논문

[2] 정식훈, 2003, “최적경로를 이용한 이동매니퓰레이터의 물체이동에 관한 연구”, 부산대학교 석사학위논문

[3] 천홍석, 2007, “움직이는 원통형 물체를 잡는 매니퓰레이터를 위한 레이저 거리계 기반의 서보시스템”, 한국과학기술원 석사학위논문

[4] 조수정, 2007, “지능로봇을 위한 실시간 장애물 회피에 관한 연구”, 제주대학교 석사학위논문

[5] 조수정, 최경현, 김병국, “Elastic Force기반의 Silvermate 로봇의 실시간 장애물 회피”, 대한기계학회 추계학술대회 논문 초록집, pp. 41-46

[6] 최경현, 조수정, 양형찬. “탄성력을 이용한 실시간 장애물 회피에 관한 연구”, 한국공작기계학회 논문집, 제16권, 제5호, pp.33-40

[7] Oliver Brock, 1999, "Generating robot motion : the integation of planning and execution" PhD thesis, Stanford University

[8] 고낙용, 서동진, 2005. “다중 장애물 환경에서의 이동 매니퓰레이터의 충돌회피”, 대한기계학회 춘추학술대회 논문집, pp. 2147-2152

[9] 서동진, 2006. “서버-클라이언트 구조의 다중로봇 시뮬레이션 프로그램의 개발”, 조선대학교 박사학위논문

[10] 이정민, 2007, “매트랩을 이용한 로봇팔의 협업 제어”, 서울산업대학교 석사학위논문