Proceedings of the 12th
KACC, October 1997

엄격히 상호 간섭하는 이동 로봇의 협동 제어

Cooperative Control of Tightly-Coupled Multiple Mobile Robots

* 이승환*, 이연정*

*경북대학교 전자·전기공학부(Tel : 053-950-6562; Fax : 053-950-5505; E-mail : yjlee@palgong.kyungpook.ac.kr)

Abstracts In this paper, we propose a cooperative multi-robot control algorithm. Specifically, the cooperative task is that two mobile robots should transfer a long rigid object along a predefined path. To resolve the problem, we introduce the master-slave concept for two mobile robots, which have the same structure. According to the velocity of the master robot and the positions of two robots on the path, the velocity of the slave robot is determined. In case that the robots can't move further, the role of the robot is interchanged. The effectiveness of this decentralized algorithm is proved by computer simulations.

Keywords multi-robot, mobile robot, path-tracking, cooperation, tightly-coupled

1. 서론

요즘은 공산품의 생산방식이 다품종 소량생산이 주가 되어가고 있다. 이러한 상황에 쉽게 대처하기 위해선 거대한 전용로봇을 사용하기보다는 작은 로봇 여러 대를 사용하는 것이 훨씬 유리하다. 이동로봇에 대해서도 마찬가지이다. 작은 여러 대의 협동로봇이 품종이 자주 바뀌는 상황을 전용로봇보다 더 쉽게 적응할 수가 있다. 그래서 현재 여러 곳에서 다중 협동 로봇에 대한 연구[1-5]가 활발히 진행되고 있다. 하지만 지금까지의 대부분의 협동작업 로봇에 관한 연구는 여러 대의 로봇이 서로 방해하지 않고 목표위치에 잘 도달하는 방법[5]이나, 로봇들이 각각 짐을 실어 나를 때의 교통 관제 방법[2,3] 등과 같이 각각의 로봇이 혼자 할 수 있는 일을 여러 대가 할 때 서로에게 방해받지 않고 효율적으로 작업을 수행하는 방법이 주로 연구되어 왔다. 그러나 실제적인 상황에선 한 대의 로봇으론 주어진 작업을 수행할 수 없고 여러 대의 로봇이 서로 긴밀하게 협동을 해야만 할 수 있는 일이 흔하다. 예를 들면 운반해야 할 물체가 이동로봇보다 훨씬 길거나, 물체의 두 곳 이상을 지지해서 운반해야 할 경우, 또는 작업공간이 협소해서 큰 이동로봇을 사용할 수가 없는 경우에는 부득이 작은 여러 대의 로봇이 협동하여 물체를 운반해야만 한다.

이러한 경우에는 한 대의 로봇일 때는 전혀 고려되지 않았던 각각의 로봇들간의 협력문제를 고려해야만 한다. 이러한 협력문제는 이전의 협동작업 로봇에서 볼 수 있는 느슨한(loosely coupled) 제한조건이 아니라 반드시 지켜야만 하는 강한(tightly coupled) 제한조건을 만족해야 한다. 즉 운반되는 물체의 형태나 길이가 일정하고 변형되어서는 안되며 로봇들이 이동해야 하는 경로가 부분적이나마 미리 결정되어 있기 때문에 각각의 로봇들은 이러한 제한 조건들을 만족시키기 위하여 행동의 제한을 받게 된다. 이를테면 곡선 경로를 통하여 긴 물체를 옮기는 로봇들은 서로의 위치와 속도에 따라 전진과 후진을 바꾸어가면서 이동을 해야만 한다.

이러한 경우 문제를 해결하기 위해서 두 가지 방식의 접근방법을 생각해 볼 수 있다.. 운반해야 하는 물체의 전체적인 속도와 각속도를 미리 결정하고 그에 따라 각각의 자체 지능이 없는 로봇들의 속도와 각속도를 결정하는 방식[3]이며 다른 한가지 방식은 본 논문에서 제안한 방법으로 어느 정도의 지능을 가진 각각의 로봇들이 자신의 경로를 미리 설정하고 운반해야 하는 물체에 의해 발생하는 제한조건을 통신을 통해서 각각의 경로에서 어떠한 속도로 움직일지를 결정함으로써 해결하는 방식이다.

첫 번째 방법은 중앙집중식으로 각각의 로봇들을 제어하는 방법으로 전체적인 목표를 이루기 위한 제어계획을 관장하는 중앙제어시스템이 있어야 한다. 두 번째 방법은 분산적으로 각각의 로봇을 제어하는 방법으로 전체적인 목표를 이루기 위한 제어계획과 협동계획이 각각의 로봇들에게 분산되어 있어야 한다.

여기서 제안하는 방법은 두 번째 방법에 해당하는 것으로 두 대의 로봇들은 센서를 이용한 주변환경의 인식을 통하여, 또는 미리 주어진 지도 정보를 통하여 부분적으로나마 경로를 미리 알고, 또 각각의 로봇들은 서로 양방향 통신을 하여 서로의 위치를 알고 있다는 가정하에 일정길이의 물체를 운반하기 위한 로봇의 속도 및 위치를 결정하는 알고리즘을 제안한다.

2. 문제설정

2.1 협동로봇의 특성

본 논문에서 가정한 로봇들은 각각의 로봇의 특성이 완전히 동일하다. 따라서 각각의 로봇은 다른 로봇의 역할을 대신할 수가 있다. 또한 각각의 로봇들은 완전히 자율적이다. 즉 전체적인 목표를 이루기 위한 계획은 각각의 로봇이 자신의 역할에 따라, 센서의 입력에 따라 독립적으로 수립한다. 즉 전체적인 목표를 이루기 위한 계획이 분산되어있다. 이러한 특성은 중앙집중식 로봇제어에 비해 각각의 로봇들은 센서의 입력에 따라 행동을 결정하기 때문에 환경변화에 적응하기가 쉽고 넓은 응용범위를 가진다.

2.2 협동로봇이 풀어야 할 문제

여기에서 해결해야 할 협동작업은 두 대의 통신 가능한 로봇이 직선과 원호로 구성된 전체경로를 빠져나가서 목표위치까지 도달하는 것이다. 이때에 각각의 로봇은 전체경로에 대한 정보는 없으며 단지 로봇이 위치한 부분경로에 대한 정보밖에 없다.

이와 비슷한 작업을 우리 주변에서 살펴보면 건축공사현장에서 두 사람의 인부가 버팀목이나 파이프처럼 긴 물체를 어떠한 방법으로 원하는 위치로 옮길 것인가? 하는 문제와 동일하다. 각각의 인부들은 시각에 의해서 자신만의 운반경로를 계획하며 상대방의 속도에 따라서 자기의 속도를 가변 시키면서, 서로서로 자기의 상태에 대해 얘기를 하면서 곡선의 길을 빠져나간다.

2.3 기본 알고리즘

협동을 위한 기본 알고리즘은 사람이 긴 물체를 옮길 때 사용

하는 방법을 그대로 모사 한다. 사람은 긴 물체를 옮길 때 앞에 가는 사람이 시각정보에 의해서 자기 임의대로 부분적인 경로와 속도를 설정하여 진행을 한다. 이때 뒤에 가는 사람은 앞의 사람과 상관없이 시각정보에 의해서 자기에게 맞는 적당한 부분경로를 설정하고 물체를 통해서 전해오는 힘을 감지하여 그 힘에 따라 미리 설정된 부분경로를 진행할 속도를 설정한다. 만일 이때에 뒷사람이 도저히 따라갈 수 없는 속도가 요구되어지면 앞에 가는 사람에게 천천히 가기를 요구하거나 역할을 바꾸어 뒷사람이 임의의 속도를 정하고 앞사람이 뒷사람의 속도에 맞추어 속도를 설정하도록 한다.

이러한 방법을 로봇에게 적용하면 다음과 같다. 일단 master 로봇과 slave 로봇을 설정한다. master 로봇은 앞에서 이동시켜야 할 물체를 끄는 사람과 같은 역할을 하며, slave 로봇은 뒤에서 물체를 통해서 전달받은 힘을 감지하여 속도를 설정하는 사람과 같은 역할을 한다. master 로봇이 센서 입력을 통하여 부분경로를 결정하고 적당한 속도를 결정한다. 그리고 이 논문의 로봇에서는 압력센서를 사용하지 않기 때문에 사람처럼 물체를 통해서 전달받은 힘에 의해 slave 로봇의 속도를 결정하는 방식을 사용할 수 없다. 그래서 slave 로봇이 master 로봇의 부분경로와 위치, 속도를 통신을 통해서 전달받고 그에 따라 계산에 의해서 slave 로봇의 속도를 결정하고 진행한다. 그리고 계산된 slave 의 속도가 slave 로봇이 도저히 따라갈 수 없는 속력일 때에는 master 로봇과 slave 로봇의 역할을 바꾸고 난 후 속도를 다시 결정한 후에 진행한다.

3. 부분경로에서 로봇 속도의 상관관계

3.1 부분경로

부분경로란 전체경로를 이루는 직선과 원호를 의미한다. 로봇의 부분경로는 slave 로봇의 속도를 결정한다는 점에서 중요하다. 로봇의 부분경로에 따른 속도 관계는 직선-원호, 직선-직선, 원호-원호 3 가지가 있다. 그리고 각각의 부분경로에 따른 속도 관계를 표현하기 위한 로봇의 변수들은 실제 구현했을 때 얻기 쉬운 변수를 선택했다. 즉 이동 로봇에서 가장 알아내기 쉬운 이동거리로 쉽게 변환이 가능한 변수로써 로봇의 위치를 표현했다.

3.2 직선과 원호

한대의 로봇의 원호에 있고 다른 한대는 직선에 있을 때의 각각 의 로봇의 속도에 대한 수학적 관계는 다음과 같다.
각각의 로봇의 위치는 다음과 같이 표현된다.

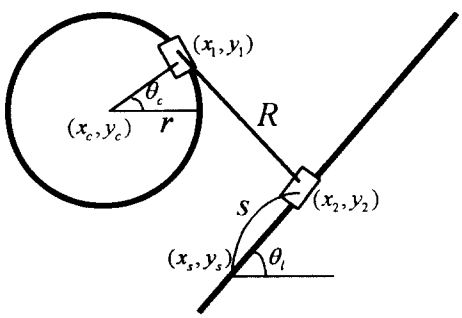


그림 1 직선과 원호 상에 위치하는 로봇들.
Fig.2 Robots on line and circle.

$$x_1 = r\cos\theta_c + x_c \qquad y_1 = r\sin\theta_c + y_c \tag{1}$$
$$x_2 = s\cos\theta_l + x_s \qquad y_2 = s\sin\theta_l + y_s \tag{2}$$

로봇들이 반드시 지켜야 할 조건은 다음과 같다.

$$(r\cos\theta_c + x_c - s\cos\theta_l - x_s)^2 + (r\sin\theta_c + y_c - s\sin\theta_l - y_s)^2 = R^2 \tag{3}$$

(3)번 식을 미분하면,

$$2\cdot(r\cos\theta_c + x_c - s\cos\theta_l - x_s)\cdot(-r\sin\theta_c d\theta_c - \cos\theta_c) + 2\cdot(r\sin\theta_c + y_c - s\sin\theta_l - y_s)\cdot(r\cos\theta_c d\theta_c - \sin\theta_l ds) = 0 \tag{4}$$

(4)번식을 정리하고 양변에 dt 로 나누면

$$\omega_c\cdot\{r\cos\theta_c\cdot(y_c - y_s) + rs\sin(\theta_c - \theta_l) - r\sin\theta_c\cdot(x_c - x_s)\} = v_s\cdot\{r\cos(\theta_l - \theta_c) - s + \cos\theta_l\cdot(x_c - x_s) + \sin\theta_l\cdot(y_c - y_s)\} \tag{5}$$

(5)번식을 선속도만으로 표현하면

$$v_c\cdot\{\cos\theta_c\cdot(y_c - y_s) + s\sin(\theta_c - \theta_l) - \sin\theta_c\cdot(x_c - x_s)\} = v_s\cdot\{r\cos(\theta_l - \theta_c) - s + \cos\theta_l\cdot(x_c - x_s) + \sin\theta_l\cdot(y_c - y_s)\} \tag{6}$$

3.2 직선과 직선

두 대의 로봇이 모두 직선에 있을 때의 각각의 로봇의 속도에 대한 수학적 관계는 다음과 같다.

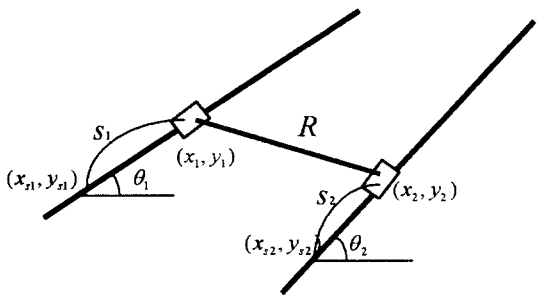


그림 2 직선과 직선 상에 위치하는 로봇들.
Fig.2 Robots on lines.

각각의 로봇의 위치는 다음과 같이 표현된다.

$$x_1 = s_1\cos\theta_1 + x_{s1} \qquad y_1 = s_1\sin\theta_1 + y_{s1} \tag{7}$$
$$x_2 = s_2\cos\theta_2 + x_{s2} \qquad y_2 = s_2\sin\theta_2 + y_{s2} \tag{8}$$

만족시켜야 할 조건은 다음과 같다.

$$(s_1\cos\theta_1 + x_{s1} - s_2\cos\theta_2 - x_{s2})^2 + (s_1\sin\theta_1 + y_{s1} - s_2\sin\theta_2 - y_{s2})^2 = R^2 \tag{9}$$

(9)번식을 미분하면,

$$2\cdot(s_1\cos\theta_1 + x_{s1} - s_2\cos\theta_2 - x_{s2})\cdot(\cos\theta_1 ds_1 - \cos\theta_2 ds_2) + 2\cdot(s_1\sin\theta_1 + y_{s1} - s_2\sin\theta_2 - y_{s2})\cdot(\sin\theta_1 ds_1 - \sin\theta_2 ds_2) = 0 \tag{10}$$

(10)번식을 정리하고 dt 로 나누면,

$$v_1\{s_1 - s_2\cos(\theta_1 - \theta_2) + \cos\theta_1\cdot(x_{s1} - x_{s2}) + \sin\theta_1\cdot(y_{s1} - y_{s2})\} = v_2\{-s_2 + s_1\cos(\theta_1 - \theta_2) + \cos\theta_2\cdot(x_{s1} - x_{s2}) + \sin\theta_2\cdot(y_{s1} - y_{s2})\} \tag{11}$$

3.4 원호와 원호

두 대의 로봇이 원호에 있을 때의 각각의 로봇의 속도에 대한 수학적 관계는 다음과 같다.

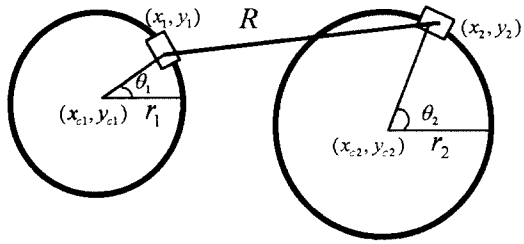


그림 3 원호와 원호 상에 위치하는 로봇들.
Fig.3 Robots on circles.

각각의 로봇의 위치는 다음과 같이 표현된다.

$$x_1 = r_1\cos\theta_1 + x_{c1} \qquad y_1 = r_1\sin\theta_1 + y_{c1} \tag{12}$$

$$x_2 = r_2\cos\theta_2 + x_{c2} \qquad y_2 = r_2\sin\theta_2 + y_{c2} \tag{13}$$

만족시켜야 할 조건은 다음과 같다.

$$(r_1\cos\theta_1 + x_{c1} - r_2\cos\theta_2 - x_{c2})^2 + (r_1\sin\theta_1 + y_{c1} - r_2\sin\theta_2 - y_{c2})^2 = R^2 \tag{14}$$

위와 같은 방식으로 양변을 미분하고 정리하면,

$$\omega_1\{r_1\cos\theta_1\cdot(y_{c1}-y_{c2}) - r_1\sin\theta_1\cdot(x_{c1}-x_{c2}) + r_1r_2\sin(\theta_1-\theta_2)\} = \omega_2\{r_2\cos\theta_2\cdot(y_{c1}-y_{c2}) - r_2\sin\theta_2\cdot(x_{c1}-x_{c2}) + r_1r_2\sin(\theta_1-\theta_2)\} \tag{15}$$

선속도로 표시하면.

$$v_1\{\cos\theta_1\cdot(y_{c1}-y_{c2}) - \sin\theta_1\cdot(x_{c1}-x_{c2}) + r_2\sin(\theta_1-\theta_2)\} = v_2\{\cos\theta_2\cdot(y_{c1}-y_{c2}) - \sin\theta_2\cdot(x_{c1}-x_{c2}) + r_1\sin(\theta_1-\theta_2)\} \tag{16}$$

4. 제어 알고리즘

4.1 전체적 흐름

Step1 : 임의로 한 로봇에게 master 다른 한 로봇에게는 slave 역할을 하도록 설정한다.

Step2 : 각각의 로봇은 센서데이터를 통하여 개별적인 부분경로를 설정한다.

Step3 : master 로봇의 속도를 임의로 설정한다.

Step4 : master 로봇의 부분경로와 속도, 위치에 따라 위에서 제시한 각각의 로봇간의 속도의 관계에 따라 slave 로봇의 속도를 계산해 낸다.

Step5 : 계산된 slave 속도의 크기가 물리적으로 로봇이 낼 수 없는 속도이거나 slave 의 속도의 방향이 경로의 방향과 다를 경우인지 체크하여 slave 속도의 타당성을 검사한다.

Step6 : slave 의 속도가 타당하지 못할 경우 master 와 slave 의 역할을 바꾸어 이번에는 이전에 slave 였던 로봇이 master 가 되어 속도를 임의로 설정하고 이전에 master 였던 로봇이 slave 가 되어 로봇간의 속도관계에 따라 속도를 계산하여 설정한다.

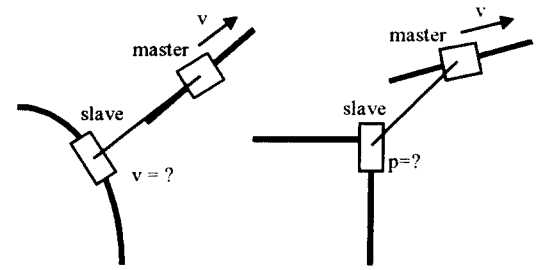
Step7 : slave 속도가 타당할 경우 미리 설정된 Δt 동안 master 와 slave 각각 자신의 속도로 진행한 후의 위치를 계산해 낸다.

Step8 : Δt 이후의 위치가 부분경로를 벗어났는지 검사하여 부분경로를 벗어났을 경우에는 다음 부분경로의 시작까지만 로봇을 진행하며, 부분경로를 벗어나지 않았을 경우에는 Δt 동안 진행한다.

Step9 : 로봇의 위치가 최종 목표에 도달했는지 검사하여 최종 목표에 도달했으면 종료하고 그렇지 않으면 Step1 부터 다시 시작한다.

4.2 Master-Slave 역할 바꾸기

제안한 알고리즘에서 master slave 의 역학 바꾸기는 아주 중요하다. 긴 물체를 곡선의 경로로 운반할 경우 master 의 임의의 속도에 대응하는 slave 의 속도는 언제나 타당한 것은 아니다. 다음 그림은 slave 가 master 속도에 대응하는 적합한 속도나 위치가 없는 경우를 나타낸다. (a)번 그림은 master 속도와 위치를 제안한 부분경로가 원호와 직선일 경우의 방정식에 넣어 slave 의 속도를 구했을 때 slave 의 속도는 무한대가 되는 지점이며 (b)번 그림은 slave 가 미분 불가능한 부분직선의 경계 점에 있는 경우로 좌측 부분경로로부터 타당한 slave 속도를 구할 수도 있고 우측 부분경로로부터 타당한 slave 의 속도를 구할 수도 있지만 구해진 속도에 의한 다음 위치는 경로 상에 존재하지 않는다. 이러한 두 가지 경우가 발생했을 때 각각의 로봇들의 master , slave 역할을 바꾸어 다시 속도를 찾으면 master, slave 양측에 모두 타당한 속도를 얻을 수 있다.



(a) slave velocity = ∞. (b) slave next position = ?.

그림 4 master slave 역할을 바꾸는 순간.

Fig. 4 Switching point.

4.3 로봇간의 거리오차

제안한 알고리즘이 완벽하게 수행되기 위해선 master 로봇의 속도에 따라 slave 로봇의 속도가 연속적으로 변화를 해야만 한다. 하지만 실제 로봇을 구동할 때든 소프트웨어적인 로봇을 구동할 때든 master 로봇의 상태를 무한히 빠른 속도로 읽어 들일 수가 없다. 따라서 slave 로봇은 미리 설정한 Δt 동안은 master 상태가 일정하다고 가정하고 slave 의 속도를 구하고 Δt 동안 진행하게 된다. 따라서 반드시 유지해야하는 로봇간의 거리에 오차가 생기게 된다. 이러한 오차를 줄이는 방법은 Δt 를 줄이거나 로봇의 속도를 줄여야만 한다. Δt 는 로봇의 통신시간이나 계산시간에 의해 결정되므로 줄이는데 한계가 있다. 따라서 거리오차를 허용한계 이하로 줄이기 위해선 경우에 따라 master 로봇의 속도와 slave 의 한계속도를 줄여야만 한다.

5. 모의실험

5.1 모의실험용 프로그램 (Cobot)

제안한 제어 알고리즘의 타당성을 실증하기 위하여 협동로봇과 작업환경을 소프트웨어적으로 구현한 프로그램으로 전체적인 모습은 다음과 같다.(visual C^{++} 사용)

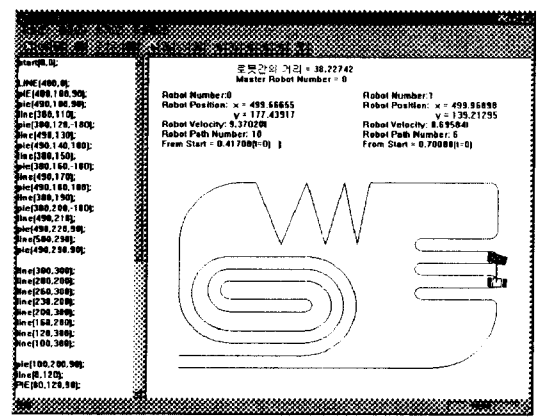


그림 5 Cobot Program.

Fig. 5 Cobot Program.

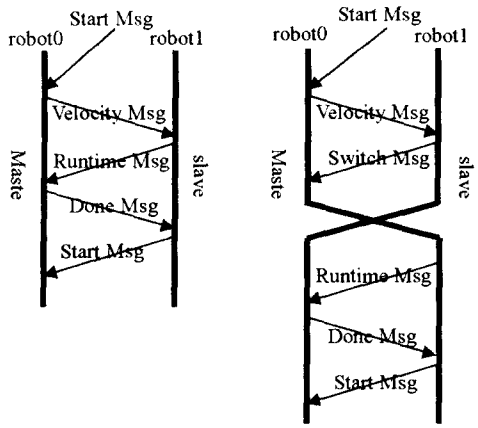
좌측 에디터 윈도우에 원하는 형태의 경로와 로봇의 초기위치를 에디터하면 우측 그래픽 화면에 그림으로 나타나게 되며 출발신호에 따라 제안한 알고리즘에 의해서 로봇들이 긴 물체를 경로에 따라 운반하는 모습을 보여주게 된다.

5.2 소프트웨어 로봇의 특징

cobot 프로그램에서는 멀티쓰레드 기법을 이용하여 각각의 소프트웨어 로봇이 독립적으로 동작하도록 만들어져 있다. 즉 분산식 로봇제어의 특징을 충분히 나타내기 위하여 각각의 소프트웨어 로봇을 구현한 모듈은 독립적인 쓰레드이며 메인 쓰레드는 각각의 소프트웨어 로봇을 제어하기보다는 단지 소프트웨어 로봇이 보내주는 정보를 화면에 보여주는 기능만 가지고 있을 뿐이다.

각각의 하드웨어적인 로봇이 완전히 동일한 기능을 가지고 있다는 기본 전제를 만족시키기 위해서 소프트웨어 로봇은 같은 클래스의 인스턴스로 구현되어있다. 따라서 모든 소프트웨어 로봇은 완전히 같은 기능을 가진다.

하드웨어적인 로봇들이 서로에게 양방향 통신을 하는 것과 동일하게 소프트웨어적인 로봇들은 서로서로 Msg(Message)를 주고받음으로써 양방향 통신을 한다.



(a) Switch 메시지가 발생하지 않았을 때.

(b) Switch 메시지가 발생하였을 때.

그림 6 메시지 전달

Fig.6 Transmit message

5.3 제안된 알고리즘의 구현을 위한 통신 방법

5.3.1 Switch 메시지가 발생하지 않았을 때

master 로봇에게 Start Msg 가 주어지면 알고리즘이 시작된다. Start Msg 를 받은 master 로봇은 임의의 속도를 설정한 후에 slave 에게 master 로봇의 속도와 위치, 부분경로를 Velocity Msg 에 실어서 보낸다. Velocity Msg 를 받은 Slave 로봇은 Velocity Msg 에 실려온 master 로봇의 정보를 가지고 자신의 속도를 결정한다. 이때 결정된 속도가 타당한가를 검사하여 타당하면 설정된 속도로 진행할 시간간격을 Runtime Msg 에 실어서 보낸다. Runtime Msg 를 받은 master 로봇은 Runtime Msg 에 실려온 진행시간만큼 진행을 한 후에 slave 로봇에게 Done Msg 를 보낸다. slave 로봇은 진행시간만큼 진행을 하고 Done Msg 를 master 에게 받으면 Start Msg 를 master 에게 보낸다.

5.2.1 Switch 메시지가 발생했을 때

이 경우는 위의 경우와 달리 slave 는 Velocity Msg 를 받고 자신의 속도를 결정했을 때 결정된 속도가 타당하지 못할 때에 slave 쪽에서 master 쪽으로 Switch Msg 에다가 자신이 원하는 속도와 현재 위치, 부분경로를 실어서 보낸다. Switch Msg 를 받은 master 로봇은 slave 로 역할이 바뀌면서 Switch Msg 에 실려온 정보를 가지고 자신의 속도를 계산해 설정한다. 그리고는 Runtime Msg 를 발생시킨다. 이 이후로는 단지 역할만 바뀌었을 뿐 방법은 완전히 동일하다.

5.4 모의실험 결과

다음 그림은 두 대의 로봇이 곡선의 경로를 빠져는 모습을 보인 것이다. 그림에서 짙은 색의 로봇이 master 로봇을 의미한다. 출발할 때에는 앞의 로봇이 master 이었으나 3 번 그림에서 한번의 switching 이 일어났다는 걸 알 수 있다. 그 이후에도 계속적인 switching 으로 길을 빠져나간다.

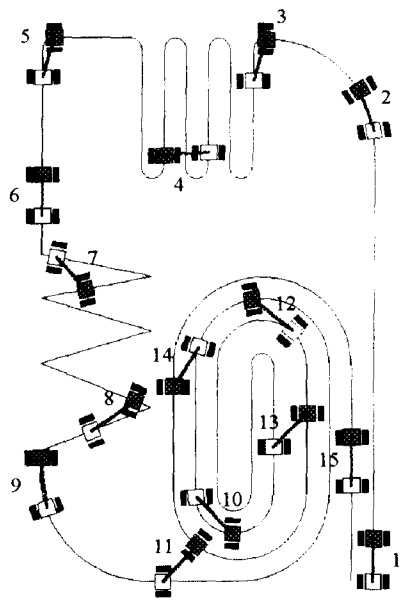


그림 7 모의실험 결과.

Fig. 7 simulation result.

6. 결론

본 연구에서 분산제어 방식으로 곡선 경로를 통하여 두 대의 로봇이 긴 물체를 옮기는 방법을 제시하였다. 이 방법은 기존의 중앙집중형 제어방법에 비해 알고리즘이 간단하고 작업공간의 변화에 잘 적응할 수 있으며 현실적인 방법임을 알 수 있었다. 향후에 하드웨어로 직접 제작하여 알고리즘의 타당성을 실증하고 나아가서 집단적인 이동로봇의 분산적인 제어방법을 연구할 계획이다.

참고문헌

[1]John S. Bay, "Design of the 'army-ant' cooperative lifting robot", IEEE Robot. Automat. Mag., vol 2., no. 1, Mar. 1995, http://armyant.ee.vt.edu/armyant-project.html.

[2]Rachid Alami, Frederic Robert, Felix Ingrand and Sho'ji Suzuki, "Multi-robot Cooperation through Incremental Plan-Merging", IEEE R&A, 1995.

[3]R. Alami, L. Aguilar, H. Bullata, S. Fleury, M. Herrb, F. Ingrand, M. Khatib and F. Robert, "A General framework for multi-robot cooperation and its implementation on a set of three Hilare robots", Preprints of the Fourth International Symposium on Experimental Robotics, ISER'95 Stanford, California, June 30-July 2, 1995.

[4]Yutaka J. Kanayama, "Rigid Body Motion Analysis towards Rotary Vehicle", IEEE Transactions on robotics and automation, vol. 13, no. 1, February 1997.

[5]김도윤, 정명진, "동적 환경에서 강화학습을 이용한 다중 이동로봇의 제어", Proceedings of the 11th KACC, October, 1996.