Proceedings of the 12th
KACC, October 1997

축구 로봇을 위한 제어기 설계
The Design of Controllers for Soccer Robots

김광춘, 김동한, 김용재, 김종환

한국과학기술원전기 및 전자공학과
(Tel:+82-42-869-5448; E-mail: kckim,dhkim,yjkim,johkim@vivaldi.kaist.ac.kr)

Abstracts In this paper, two kinds of controller are proposed for a soccer robot system.. One for Supervisor and defense mode, and the other for attack mode. Robot soccer game has very dynamic characteristics. Furthermore, there exist competitions between agents. The soccer-playing robot should take an appropriate action according to its surroundings. Initially, an attack mode controller using a vector field concept is designed, then a supervisor and a defense mode controller are designed with a Petri-net. The efficiency and applicability of the proposed controllers are demonstrated through a real robot soccer game(MiroSot 97).

Keywords multi-agent, robot, soccer, MiroSot.

1. 서 론

Multi-agent system을 형성한다는 것은 많은 수의 Agent들로 이루어진 Ingellgent System의 구현 원리를 정하고 그 System 내에서 Agent들이 독자적인 행동을 하고 동시에 서로 협력을 할 수 있도록 하는 동작 Mechanism을 제공해 줄 수 있는 것을 의미한다. Multi-agent system은 한 개체 System이 할 수 없는 일을 성취할 수 있을 것으로 기대하고 있다. 왜냐하면 One Agent System의 능력은 궁극적으로 제한되어 있기 때문이다. Multi-agent System이 협력 작업을 하게 될 때 좋아 질 수 있는 것은, 한 System만으로 해결될 수 없는 아주 복잡한 일들의 성취도가 Multi-agent 일 때는 높아질 수 있다는 점과, 여러 일을 할 수 있는 우수한 능력의 한 System을 만드는 것보다 능력은 다소 떨어지지만 여러 종류의 만들기 쉽고 싼 System을 제작하는 것이 유리할 수 있다는 점이다.

이러한 Multi-agent system의 연구에 적합한 환경으로 로봇 축구대회인 MiroSot을 들 수 있다. 대부분의 Multi-agent System의 연구 방향이 Cooperation에 집중되어 있으나, 축구 경기는 같은 Team끼리의 협력을 요구할 뿐만 아니라 상대팀과의 경쟁도 요구하고 있다는 것이 큰 차이점이다. 이런 능동적인 환경에서의 Multi-agent 제어 Algorithm은 Mobile Robot의 기구학적, 동력학적 특성을 고려한 low level에서부터, 3개의 능동적인 장애물인 상대 로봇을 고려하여 전술을 적용하는 high level 까지 포괄하여야 한다.

이러한 환경에서 간단하고 처리속도가 빠른 Algorithm을 선택해야 하고, 실제 로봇의 구조적인 특성에 맞추어서 개발되어야 한다. 이에 본 연구에서는 먼저 여러 방식의 로봇 구동 방식을 제안하고 구현했다. 이런 방식들은 Algorithm에서 low level function으로 선택, 이용되며, high level에서 Robot Soccer를 Discrete Events System 으로 Modeling하고 이를 Petri Net으로 나타내어 실제 MiroSot에 적용하는 방식을 취한다.

개발된 Algorithm의 타당성은 실제 MiroSot 경기의 결과로 충분히 보여질 수 있다. 실제 적용한 결과, Programming 시간을 단축하게 됐고, Debugging도 쉽게 할 수 있게 되었으며 제2회 MiroSot 대회에서 3위에 입상하는 성적을 거두게 되었다.

2. Mobile Robot의 제어 방식

2.1 Path Planning 과 Path Following 에서의 문제점

본 장에서는 자율 이동 로봇의 경로발생 및 자세제어 방법을 개발하고 실제로 구현한다. 기존의 로봇 경로 발생 및 제어는 복잡한 방식을 가지고 있으며 실시간 제어나 고속의 제어가 어렵다는 단점이 있다. 따라서 여러 가지 제어방식을 개발하고 구현하여 상황에 맞게 이용 가능하도록 한다.

로봇 축구대회에서 필요한 로봇 제어는 다음과 같은 조건을 만족해야 한다.

(a) 실시간 제어가 반드시 가능해야 한다.
(b) 로봇의 속도가 빠른 상태에서 제어가 가능해야 한다.
(c) 이동하는 장애물의 회피가 가능해야 한다.
(d) 오차 및 노이즈에 강인한 제어가 필요하다.

이러한 움직임을 보여주기 위해서는 어떻게 이러한 경로를 생성할 것인가라는 문제와, 경로가 생성되었다면 어떻게 그 경로를 따라갈 것인가라는 문제를 해결해야 한다. 로봇이 가지는 nonholonomic constraint 에 의해, 경로를 따라가는 문제는 간단한 수식으로 계산되지 않는다. 또한 기존의 로봇 경로 발생 및 제어는 복잡한 방식을 가지고 있으며 실시간 제어나 고속의 제어가 어렵다는 단점이 있다.

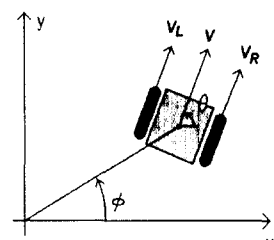

그림 1 기구학적 모델
fig.1 Modeling of Kinematics

2.2 Mobile Robot의 Modeling

로봇의 외형은 그림 1 과 같다. 로봇은 두 개의 바퀴를 가지고 있으며, 바퀴면에 수직한 미끄러짐이 없으며 따라서 바퀴의

병진 운동 성분은 바퀴면에 수직한 성분만이 나타난다고 가정한다(non-slipping & pure rolling). 로봇의 자세 벡터 $\overrightarrow{P}$와 속도 벡터 $\overrightarrow{Q}$는 다음과 같은 관계를 가진다. 여기서 속도 벡터는 로봇 중심의 병진 운동 속도와 로봇 중심을 기준으로 한 각속도로 이루어진다. 다음은 로봇의 kinematics이다.

$$\overrightarrow{P} = \begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} V \\ \omega \end{bmatrix} = J(\theta)\overrightarrow{Q} \qquad (1)$$

(1) 식을 보면 얻고자 하는 위치 및 각도의 제어시 다음과 같은 제한조건이 만족되어야만 한다. (nonholonomic constraint)

$$G \cdot \overrightarrow{P} = [\ \sin\theta \ \ -\cos\theta \ \ \ 0\ ] \begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{\theta} \end{bmatrix} = \dot{X}\sin\theta - \dot{Y}\cos\theta = 0 \qquad (2)$$

이것은 로봇의 순간 진행방향은 로봇이 향하고 있는 각도 θ와 같아야 한다는 의미이다.

그림 1과 같은 구조의 로봇은 다음과 같은 Lagrange's equation을 갖는다. 여기서 로봇의 질량은 M 이고, rotational Inertia는 I 이다. 바퀴의 질량과 rotational inertia 모두 0으로 근사화했다.

$$\text{Lagrange's equation} : \quad \frac{d}{dt}\frac{\partial L}{\partial \dot{q}} - \frac{\partial L}{\partial q} = \tau$$

$$\begin{bmatrix} F_L \\ F_R \end{bmatrix} = \begin{bmatrix} \frac{M}{4} + \frac{I}{L^2} & \frac{M}{4} - \frac{I}{L^2} \\ \frac{M}{4} - \frac{I}{L^2} & \frac{M}{4} + \frac{I}{L^2} \end{bmatrix} \begin{bmatrix} a_L \\ a_R \end{bmatrix} = Hq \qquad (3)$$

위의 식에 의해 얻고자 하는 가속도가 a_L, a_R 일 때, 모터에서 가해야 하는 힘을 구할 수 있다. 양 바퀴에서 내는 힘과, 나타나는 양 바퀴의 가속도는 coupling이 되어 있으며, $\frac{M}{4} = \frac{I}{L^2}$ 일 경우에는 양 바퀴가 독립적이 된다.

2.3 기존 방식의 Vector field에 의한 재해석

Potential Field Method는 가고자 하는 위치로부터 인력을, 피하고자 하는 장애물로부터 척력을 받아서, 그 합력에 비례하는 방향으로 움직이는 방식이다. 이 방식은 가장 일반적으로 이용되는 로봇제어 방식이다. 간단하고 실시간 처리가 가능하다는 데에 장점이 있다. 그러나 속도를 일정하게 유지하지 못하고, 장애물이 많은 경우 진동을 할 가능성이 있으며, 도착 위치에서의 방향성을 보장하지 못한다. 다음 그림은 장애물이 있을 때 각 위치에서 받는 힘의 방향을 나타낸 것이다.

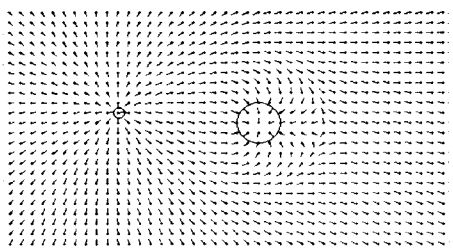

그림 2 중력장 방식
fig.2 Potential Field Method

이 그림에 나타난 벡터장은 그 위치에서 로봇이 가야 할 방향이다. 크기는 위치에 따라 큰 차이를 보이므로 나타내지 않았다. 여러가지 path planning 과 obstacle avoidance algorithm은 위와 같은 벡터장의 형태로 해석 가능하다. 여기서 나타낸 Potential Field Method의 단점들은, 그림과 같은 벡터장에서 찾아낼 수 있으며 이 벡터장을 인위적으로 변화시킨다면 더 좋은 제어가 가능할 것이다.

본 연구에서는 모든 부분에서의 벡터의 크기가 모두 같도록 하였다. 벡터장 내에서 로봇의 각도와 벡터의 방향오차를 줄여나가는 방식은 다음과 같다.

$$\omega = K_P\ (\angle \overrightarrow{V(x,y)} - \theta\ ) \qquad (4)$$

$$V = constant$$

$\overrightarrow{V(x,y)}$: 위치(x,y)에서의 벡터장. 크기는 1이다.

K_P : feedback gain

discrete time에서 위의 식은 다음과 같이 나타낼 수 있다.

$$\theta[n+1] = (1 - K_p T)\theta[n] + K_p F(\vec{P}) \qquad (5)$$

단, $F(\vec{P}) = \angle \overrightarrow{V(x[n], y[n])}$

T : Sampling time

2.4 수비수 및 Goal Keeper를 위한 제어방식 개발

수비수와 골키퍼는 Position field와 Angle field에 의해 이루어지는 moiton을 이용하여 제어된다. 다음 그림 3과 그림 4는 Position field와 Angle field의 모양을 나타낸 것이다.

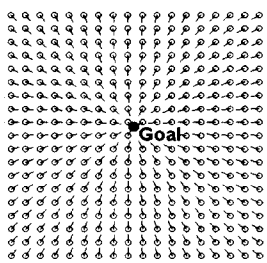

그림 3 위치 벡터장
fig.3 Position Field

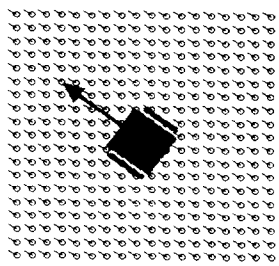

그림 4 각도 벡터장
fig.4 Angle Field

Position field는 기본적으로 Potential Field Method와 같은 의미를 가진다. 이 field 에서 로봇은 지정된 목표점으로 이동하게 되지만, 어떠한 경로로 움직이고 최종 방향은 어느 쪽인지를 보장하지 않는다. 이것은 obstacle이 존재하지 않을 때의 Potential Field Method를 vector field로 나타낸 것이며 움직임도 유사하다. 최종 위치에서 로봇이 이루는 각도가 중요하지 않은 경우에 이 방식을 이용한다. Angle field는 로봇이 정지해 있는 채로 각도만을 조정해야 할 때 이용된다. 이때 로봇의 병진운동 속도는 0이 되어야 하므로, 식 4에서 V는 0이어야 한다.

이 두가지 Vector Field를 이용하여 수비수와 골키퍼의 움직임을 형성하게 되며, 자세한 algorithm은 Section 3에서 다룰 것이다.

2.5 공격수를 위한 제어방식 개발

다음은 로봇이 최종각도를 보장하기 위한 벡터장을 나타낸 것이다. 로봇의 최종 위치는 가운데이고 최종 각도는 0 으로 오른쪽을 향하게 되어있다. 아래 그림에서 보이는 원과 원 내의 직선은 로봇의 처음 위치와 각도이고, 곡선들은 위에서 설명한 (4) 식에 의해 simulation한 경로이다.

축구경기에서 공격수는, 최종위치를 공의 위치로 하였고 최종 각도는 골대를 향하게 하는 방법으로 프로그래밍을 하였다. 그 결과 공을 차거나 모는 움직임을 하게 된다. 공의 움직임이

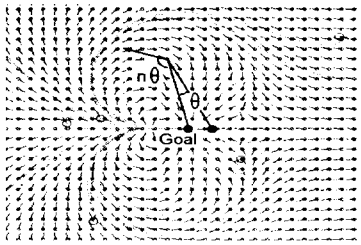

그림5 최종위치를 보상하기위한 Vector Field
fig.5 Vector Field for Final Position

변화함에 따라 벡터장의 형태도 변하며, 로봇은 어느 위치에 있었든 계속해서 공 뒤쪽으로 접근하여 공을 몰거나 찰 수 있다.

2.6 Obstacle에 의한 Vector field의 변화

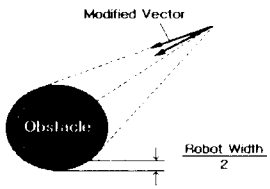

그림 6 벡터의 변화
fig.6 Modification of Vector

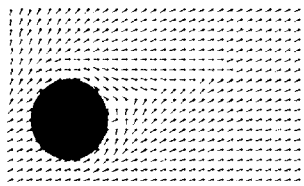

그림 7 장애물에 의한 벡터장
fig.7 Field Modified by Obstacle

장애물을 피하는 동작은 그림 6와 같이 벡터를 변화시켜서 얻어낸다. 어떤 위치에서의 벡터가 장애물을 향하고 있을 때 그 벡터를 장애물의 경계를 향하도록 바꾸는 것이다. 이때 로봇의 부피를 고려하여 장애물의 크기를 늘려서 고려해야 한다. 이러한 방법으로 개선된 벡터장은 그림 7과 같다.

다음은 위의 장애물 회피 방식과 앞에서 설명한 공격수 제어를 위한 vector field를 이용하여simulation한 것이다. Simulation에서는 로봇의 kinematics만 고려하였다. 로봇의 입력은 왼쪽바퀴의 속도와 오른쪽 바퀴의 속도이다. 로봇은 앞서 설명한 diserete time에서의 각도 보정식 (4)을 이용하였다.

3. System Modeling Using Petri Net and Construction of Strategy

3.1 Objects and Scope of the Research

본 연구의 목표는 효과적인 로봇 축구 전략의 수립이다. 본 연구에서 다루고자 하는 것은 전체 로봇 축구 시스템에 관한 것이 아니라 공격, 수비, 골키퍼 역할을 담당하는 로봇을 어떻게 control하는가 이므로 다음과 같은 가정을 한다.

(a) 매 sampling time마다 공격, 수비, 골키퍼 역할을 하는 3개의 로봇의 기준 좌표계(reference frame)에 대한 좌표를 vision system으로 부터 입력받는다.
(b) 매 sampling time 마다 기준 좌표계에 대한 공의 정확한 좌표를 vision system으로 부터 입력받는다.
(c) 매 sampling time마다 control을 관장하는 processor는 효과적으로 계산을 수행하여 각 로봇에 명령을 전달한다.

본 연구의 목표는 그림 8 의 supervisory controller를 Petri Net 을 이용하여 design하는 것이며 실제로 이를 바탕으로 controller를 구현, MiroSot system에 적용하여 이를 검증하는 것이다.

3.2 Structure of the Supervisory Controller

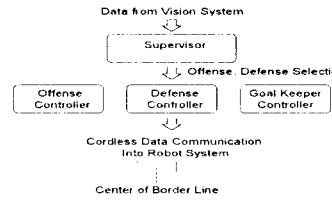

그림 8 제어기의 구조
fig. 8 Structure of Supervisory Controller

Vision system은 경기장을 가로 방향으로 바라보고 있으며, 본 연구에서는 항상 우측이 수비 진영, 좌측이 공격 진영이 된다. 기준 좌표계의 원점은 공격 진영, 즉 좌측 상단에 위치한다. 본 연구에서는 Border Line이라는 가상의 선을 도입한다. Border Line은 공의 따라 움직이며 매 sampling time마다 공의 위치에 따라 Border Line은 움직이게 된다. Border Line은 얼마간의 thickness를 지니게 된다. Robots중 Goal Area안쪽에서만 움직이는 Robot를 Goal Keeper Robot이라 정의하며, Border Line 좌측에 위치한 Robot은 공격을 담당하는 Offense, 우측에 위치한 Robot은 수비를 담당하는 Defense라 정의한다. Supervisory Controller는 그림8와 같은 구조를 지니고 있다고 설정한다.

3.3 Supervisor

그림 8 의 Supervisor는 Vision system으로 부터 공과 Robots의 위치 정보를 입력 받아서 Goal Keeper를 제외한 2대의 Robot중 어떤 Robot이 공격과 수비에 배치되어야 하는 것을 결정한다. 즉, 공격과 수비를 담당하는 Robot이 정해져 있는 것이 아니라 각 상황에 맞게 공격과 수비를 담당하는 Robot을 배정하는 역할을 한다. Supervisor의 여러 State를 Petri Net으로 Modeling 하기 위해 실제 상황을 다음과 같은 states로 나눈다.

(a) Robot 1 is offending, Robot 2 is defending
(b) Robot 1 is defending, Robot 2 is offending

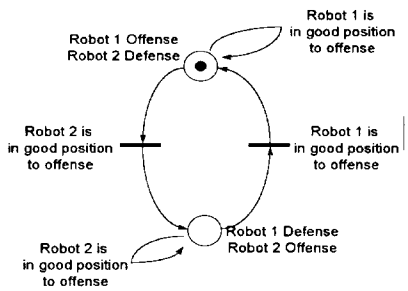

그림 9 Petri Net 모델
fig. 9 Petri Net Model of the Supervisor

이 경우 그림 9 과 같은 Petri Net 으로 간단히 Supervisor를 modeling 할 수 있다. 그림 9 에 나타난 상황을 실제로 programming 을 통해 구현하며, 그 subroutine을 NormalGame() 로 정의한다. Robot 1 과 Robot 2중 어느 것이 offense, defense에 유리한지는 공의 위치로 부터 각 Robot 과의 거리와 각도로 정의한다. 기본적으로 MiroSot에 사용되는 Robot들은 직진성은 매우 뛰어남에 반해 회전성은 매우 불리하므로, 공이 Robot앞에 놓이는 경우 Robot이 공을 향해 직선주행함으로서 Shoot동작을 일으킨다. 공이 Robot과 가까운 위치에 있는 경우 offense하는것이 더 유리하며, 더욱이 공, Robot, 그리고 상대방의 Goal Area가 직선을 이룰 경우 공격하는것이 더 유리하다. 따라서 본 연구에서는 아래와 같은 공수 전환 Switching Condition을 설정하고 실제 적용하도록 한다.

"현재 state에서 offense를 맡고 있는 Robot 과 defense를

맡고 있는 2대의 robot이 있다고 가정한다. 이 경우 offense robot과 ball의 거리를 r1, defense robot과 ball과의 거리를 r2라 두며, 각 robot의 앞부분과 공의 진행방향이 이루는 각도를 각각 Ang1, Ang2라 둔다. 이 때

*r2 < r1 * 2 & -45 Deg.<Ang2<45 Deg.*

의 조건이 만족되면 Robot 1과 Robot 2의 Role이 바뀐다."

3.4 Defense Controller

3.3절에서 Supervisor의 역할은 Goal Keeper를 제외한 2대의 Robot중 어떤 Robot이 공격과 수비를 맡아야 하는지를 정하는 것임을 알 수 있었다. 어떤 Robot이 일단 Defense를 하는데 유리하다고 판단되면 Supervisory Controller는 그 Robot에 Defense 역할을 적절히 수행하도록 명령을 내리게 된다. 이 역할을 담당하는 Supervisory Controller의 한 부분을 Defense Controller라 정의한다. 수비를 담당하는 Robot의 역할은 적의 공격으로 부터 공을 막아야 하는 것이므로 항상 공보다 오른쪽에 위치하는 것이 바람직하다. 실제 여러가지 상황이 발생할 수 있으나, 간단히 다음과 같은 State가 있다고 가정한다.

(a) Defense Robot is in the back of Ball
(b) Defense Robot kicks Ball
(c) Suicide Goal Position
(d) Defense Robot contacts with Ball in the back of ball

각 State에 대한 설명을 좀더 자세히 하기 위해 아래 그림 10를 참조한다. (a) state는 수비수가 공의 뒤에 위치한 경우이므로 적당한 수비 위치를 가지는 경우이며, (b) state는 수비수가 공의 위치로 이동하여 공을 차내는 경우이다. (c) state는 수비수가 공의 전방에 위치하는 경우이므로 수비수가 공을 차기 위해 ball쪽으로 이동하면 되려 자책골을 유발할 수 있는 위치이므로 주의하여야 한다. (d) state는 공이 수비수와 바로 붙어있는 경우로서 (c) state와는 차이가 있지만 자책골을 유발할 수 있는 위치는 아니다.

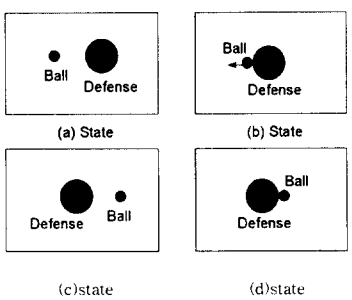

그림 10 수비 제어시의 state
fig. 10 State of the Defense Controller

실제 MiroSot 경기 도중에 이보다 더 많은 state가 존재할 수 있겠지만, 본 연구에서 설계한 defense controller는 크게 위의 4가지 경우가 있다고 가정한다. 이를 바탕으로 한 defense controller의 Petri Net model은 그림 11 과 같다.

그림 11 에서 'Angle'은 defense를 담당하는 Robot의 진행방향과 Ball을 잇는 직선이 기준 좌표계의 X축과 이루는 각을 의미한다. 또한 'Dis_x' 변수는 Ball과 defense를 담당하는 Robot과의 Pixel상의 거리로 정의한다. Defense를 담당하는 Robot이 (a) state에 있는 경우 Angle이 45도 이하이고 거리가 25 pixel 이하이면 defense controller는 Robot에게 Ball의 위치로 이동할 것을 명령하며 이는 곧 Robot이 Ball을 상대방 진영으로 차 내는 것을 의미한다. 즉 (b) state로의 이동이 발생하는 것이다. (b) state에서 Angle이 45도 이상이거나 Dis_x가 25 이상이면 Defense Robot은 다시 (a) state로 이동한다. (a) state에서 Ball이 Suicide Condition, 즉 Ball이 Defense Robot 우측으로 이동하면 state는 (c) state로 이동하게 된다.

(b) state에서 조준이 정확하게 되지 않아 Defense Robot이 Ball을 차 내는것을 실패하게 되면 Robot은 계속 진행, 결국 state는 (c) state로 변하게 된다. (c) state에서는 Robot이 Ball을 피하면서 Ball의 뒤쪽으로 이동하라는 명령을 Defense Controller가 내리므로 'Escape from Suicide' action이 일어나게 되어 state는 (a) state로 복귀하게 된다.

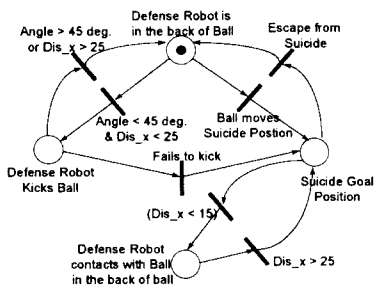

그림 11 수비 제어기의 Petri Net 모델
fig.11 Petri Net Model of the Defense Controller

(c) state에서 Dis_x가 15 이하면 state는 (d) state로 이동하게 되고 (d) state에서 Defense Robot이 Ball과 멀어지는 방향으로 이동해 Dis_x가 25 이상이 되면 state는 다시 (c) state 로 이동하게 된다

본 연구에서는 그림 11 에 나타난 상황을 실제로 programming을 통해 구현하며, 그 subroutine을 kick3() 함수로 정의한다.

3.5 Goal Keeper Controller

Goal Keeper Controller는 Goal Keeper 역할을 담당하는 Robot에게 명령을 내린다. Goal Keeper Controller는 Supervisor의 Switching Function인 NormalGame()의 영향을 받지 않는다. Goal Keeper Robot은 수비측(우측) Goal Area내에서만 운동 가능하게 설정하였다. 이는 Goal Keeper Robot으로 하여금 Goal을 막는 것에만 전념토록 하기 위한 것이다.

Goal Keeper의 가장 중요한 역할은 무엇보다도 상대방 공격수의 Shoot으로 부터 Goal을 방어하는데 있다. 본 연구에서는 이 역할을 효과적으로 수행하기 위하여 Ball이 Goal Area와 얼마만큼 떨어져 있는가에 따라 Goal Keeper Robot에게 내리는 Command를 달리하는 방식을 취한다. 이를 위하여 Ball과 Goal Area와의 거리를 다음 3가지로 분류한다.

(a) Far Distance
(b) Medium Distance
(c) In Goal Area

여기서 (a)의 'Far Distance'는 수비측 Goal Area와 Ball과의 거리가 60 pixel이상인 경우를 지칭하며 (b)의 'Medium Distance'는 거리가 60 pixel이하이고 20 pixel 이상인 경우를 의미한다. (c)의 'In Goal Area'는 Ball과 Goal Area와의 거리가 20 pixel이하인 경우이다.

Goal Keeper Controller의 Strategy를 세우기에 앞서 predictor() 라는 subroutine을 설정하기로 한다. predictor()는 Ball이 'Far Distance'의 경우에 있을 때 Ball의 진행 방향을 예측하는 함수이다. MiroSot 에서 Ball은 그 진행 방향이 거의 직선이 됨을 경험을 통해 알 수 있다. Vision system에서는 공의 위치를 매우 정확히 알 수 있으므로 5 Sampling time동안 Ball의 위치를 Memory에 저장하도록 한다. 이 때 기준 좌표계 상에서 5 Sampling time동안 공의 위치를 점으로 표시할 수 있고 Curve Fitting Algorithm을 이용, 직선의 방정식을 구할 수 있

다. 구해진 직선의 방정식으로 부터 앞으로의 공의 진행 방향을 예측하는 Subroutine이 바로 predictor()이다. 그림 12 는 predictor()의 역할을 보여준다.

Goal Keeper는 아래와 같은 state를 가지고 있다고 가정한다.

(a) Goal Keeper is in the center of Goal Area
(b) Goal Keeper is in TARGET position
(c) Goal Keeper is in ASSUM position
(d) Goal Keeper is in the Ball position

(a) state는 경기가 처음에 시작되거나 중단후 재개되어 kick off되는 경우, Goal Keeper가 Goal Area의 중앙에 위치하는 경우를 의미한다. 이러한 상황하에서 Ball이 Goal Keeper쪽으로 오는 경우는 발생하지 않기 때문이다. (b) state에서 'TARGET' position은 그림 13에 나타나 있다. 즉, predictor()가 예측한 Ball의 진행 방향의 최종 위치가 'TARGET' position이다. (c) state에서 'ASSUM' position이라 함은 현재의 Ball Position과 Goal 대의 Center를 잇는 직선상의 한 점을 의미한다. (d) state에서는 Goal Keeper가 아주 긴박한 상황이 되어 Ball position에 가 있는 경우를 나타낸다. 이를 바탕으로 한 Goal Keeper Controller의 Petri Net model은 그림 13와 같다.

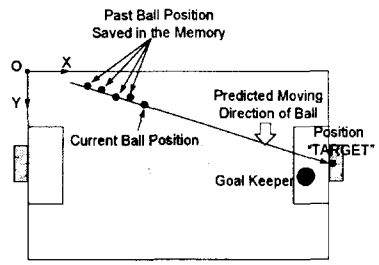

그림 12 'predictor()' 함수
fig. 12 Subroutine 'predictor()'

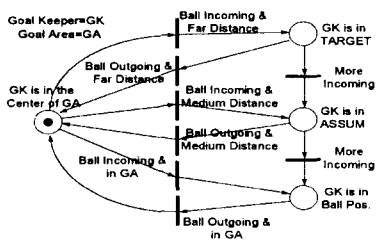

그림 13 골키퍼의 Petri Net 모델
fig.13 Petri Net Model of Goal Keeper Controller

그림 13 에서 Initial Kick Off State에서는 Goal Keeper Robot이 Goal Area의 중앙에 위치해 있다. 만약 Ball이 점점 Goal Area를 향해 다가오지만 거리가 'Far Distance'인 경우 predictor() subroutine을 Goal Keeper Controller는 Call하게 되고 예상되는 Goal 의 Position인 'TARGET'으로 Goal Keeper는 이동하게 된다. 이 때 다시 Goal 이 'Far Distance'인 state에서 멀어지게 되면 Goal Keeper는 다시 Center로 복귀한다.

만약, 상대방의 기습 공격으로 Ball 이 다가오면서 'Medium Distance'의 거리를 가지게 되면 Goal Keeper Controller는 Robot에게 'ASSUM' Position으로 이동할 것을 명령하게 되고 이런 위기 상황을 벗어나게 되면 Goal Keeper는 다시 원래 위치로 복귀한다. 이와 마찬가지로 Ball이 다가오면서 'Goal Area'안쪽으로 들어와 버리면 Goal Keeper는 Ball의 Position으로 이동하게 된다. Goal Keeper가 'TARGET' Position에 있는데 Ball이 점점 다가오게 되면 Goal Keeper는 'ASSUM' Position으로, 더욱더 다가오게 되면 Ball Position으로 이동한다.

본 연구에서는 그림 13.에 나타난 Situation을 Goalie2()라는 Subroutine으로 구현한다.

3.6 Offense Controller

본 연구에서는 Offense Controller는 신속한 공격과 전략의 수립을 위해 2.5 에서 설명한 vector field를 이용한다. 이 vector field에서 goal을 ball로 위치하게 하고, 방향을 상대편 골대로 정하면 공을 몰거나 차게 된다. 이런 방식은 매우 빠른 제어를 가능하게 해준다. 공격수는 조건이 만족되면 빠른 시간 내에 공격이 이루어져야 하며, 따라서 복잡한 state변화같은 것들은 algorithm을 비효율적으로 만든다.

2.5에서 설명한 field를 이용하면 단하나의 state로 공격을 수행할 수 있다.

4.결론

본 논문에서는 동적인 로봇축구 경기에서, Petri net을 이용한 Supervisor controller와 defense mode controller를 제안했고, 하위 level controller로서, vector field를 이용한 controller를 제안했다. 위의 controller를 실제 경기에 이용한 결과 Miro팀은 MiroSot 97대회에서 3위의 성적을 거두어 그 우수성을 확인했다. vector field를 이용한 controller의 경우, 모든 상황(벽 근처나 구석등)에서 적절한 움직임을 보여서 controller에 대한 믿음을 주었다.

제안한 controller에는 상대팀에 대한 정보를 이용하지 않았고 우리팀 로봇간의 통신을 하지 않았다. 하지만, 로봇 축구에는 각 개체간의 협동, 학습이 필요하므로 추후에 이에 대한 연구를 더 해야겠다.

5. 참고 문헌

[1] Real-time Obstacle Avoidance for Fast Mobile Robots (J.Borenstein & Y.Koren)

[2] Motion Control Analysis of Mobile Robot.(J.Borenstein & Y.Koren)

[3] The Vector Field Histogram - Fast Obstacle Avoidance for Mobile Robot (J.Borenstein & Y.Koren)

[4] Structural Properties and Classification of Kinematic and Dynamic Models of Wheeled Mobile robots (Guy Campion, Georges Bastin, and D'Andrea-Novel)

[5] Path Planning Using Linear Parametric Curve Based on Geometry Mapping of Obstacles (Ihn Namgung Jung-Gyu Kim)

[6] Holonomic and Omnidirectional Vehicle with Conventional Tires (Masayoshi Wada & Shunji Mori)