실시간 트랙 정보 모델링 및 고수준 센서 정보를 이용한 3차원 자동차 경주 제어기 설계

윤경오○, 김경중1)
연세대학교 공학대학원 컴퓨터공학과, 세종대학교 컴퓨터공학과
yoonkojs@naver.com, kimkj@sejong.ac.kr

Design of 3D Car Racing Controller Using Real-Time Track Modeling and High-Level Sensors

Kyongoh Yoon○ KyungJoong Kim1)
Department of Computer Engineering, Graduate School of Engineering, Yonsei University
Department of Computer Engineering, Sejong University

요 약

이 논문은 2011 TORSC 자동차 경주에 대한 인공지능적 접근방법을 나타내었다. 사람이 자동차 경주를 준비 할 때에는 여러 종류의 경기장, 트랙, 조건에서 연습하고 여기서 익힌 경험과 지식을 통해 실제 새로운 경기장에서 경주를 하게 된다. 본 연구에서는 이러한 학습과 적용의 단계를 두 단계의 학습으로 수행하였다. 특히 경주 조건인 트랙에 대한 경기 연습 즉, 기계 학습을 위해 트랙을 간단한 수치 자료로 구조화하고, 실시간 트랙 정보 구축으로 트랙의 형태를 파악하여 주행하는 방법을 제시하였다. 또한, 각 센서를 각 상황에 맞도록 구조화하여 고수준 센서화하는 방법으로 트랙 정보를 기록하였으며, 직관적인 효과 조정과 파악을 위해 휴리스틱을 적용하였다. 이러한 연구는 경쟁력 있는 스마트 자동차에 필요한 소프트웨어 모듈에 의미있는 한 부분이 될 수 있다.

1. 서 론

본 연구는 TORSC(The Open Racing Car Simulator) 환경 하에서, 계산지능과 게임에 대한 IEEE 컨퍼런스의 자동차 경주에 대한 인공 지능적 접근법에 관한 연구이다(http://www.ieee-cig.org/). 이 TORCS는 경주용 차의 많은 부분인 충돌, 마찰력, 기체역학, 연료소비 등의 정교한 물리 엔진과 정교한 3차원 그래픽 엔진, 그리고 여러 종류의 트랙, 차 모델, 컨트롤러 등 같은 특성 등 많은 게임 컨텐츠를 지닌 가장 좋은 무료 레이싱 게임이다.

기존의 TORSC 카 레이싱 트랙에 관한 연구를 몇 가지 살펴보면 트랙에서의 최적의 레이싱 라인을 찾기 위해 최소곡선경로(MCP, Minimum Curvature Path)와 최단경로(SP, Shortest Path)의 최적 상충점으로 모델링하는데 있어서 MCP와 SP의 선형조합을 적용 및 트랙을 의미 있는 구획으로 분해하는 간단한 휴리스틱을 적용하여 모든 구획에서 한 번에 최고의 상충점을 찾기 위해 유전자 알고리즘(Genetic Algorithm)을 탐색하는 방법을 적용하였다[1][3].

본 연구에서는 실질적으로 경주하는 동안에 트랙 상태를 파악하여 트랙 상태 정보를 저장하여 레이싱 단계에서 바로 활용할 수 있도록 하였다. 또한 이 트랙 정보는, 연습 경기 단계에서 각 트랙 상태별 최적의 효과를 찾는데 이용되었으며, 실제 카 레이싱 경기에서 트랙에 도달하기 전에 앞으로 나타날 트랙에 대한 상태 결정으로 이전 단계에서 찾아진 효과를 적용하는데 이용되었다.
또한, 각 효과자들을 제어하는 모듈에 대한 연구를 보면 직관적이고 효과적으로 각 부분의 제어를 위해 기어, 희망속도, 속도유지, 핸들, 경쟁자에 관한 기능을 모듈단위로 구현하였다[4].

1) 교신저자

자동차의 이동 성질은 복잡한 물리 공식과 TORCS 내의 구현을 통해 정확히 구현은 가능하다. 하지만, 이는 한정된 분야에만 적용 가능한 방법이며 TORCS의 공식을 그대로 가져다쓰는 것은 앞으로의 프로그램의 변화나 새로운 분야에 응용할 경우 매번 분석과 적용이 필요한 등 한계가 따르므로 본 논문에서는 차량의 특성을 주행 시에 저장한 로그 데이터를 이용하여 휴리스틱 기법을 통해 파악하고 이를 주행 코스에 적용 활용해 보는 방법을 시도하였다.

TORCS 자동차는 22개의 센서가 장착되어 있으며[2], 이를 계산하여 운전하도록 프로그래밍 한다. 이때, 차량이 위치가 트랙에서 계속 변화하므로, 이들 센서에 대한 평가 시간을 최소화 하도록 해야 한다. 또한 자동차 컨트롤러에 인공 지능적인 요소가 있고 알 수 없는 제한들이 있어서 셋업이 힘들다. 본 연구에서는 모든 센서의 정보를 다 활용하는 것이 아니라, 목적에 맞는 최적의 센서 값만을 활용하는 고수준 센서를 구현하였다.

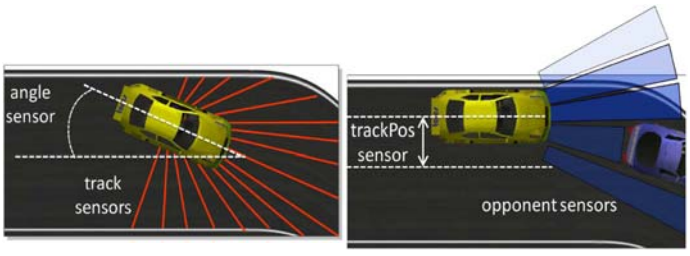

그림 1. TORCS에서 제공하는 기본 센서 정보들

2. 실시간 트랙정보 활용 클라이언트 컨트롤러 모듈

본 TORSC 자동차 경주를 위한 소프트웨어 아키텍처를 살펴보면, 아래의 그림과 같이 경주 상황 및 게임 엔진을 탑재하여 적용하고 이러한 환경에서 주행하는 서버

봇과 경주나 환경에서 들어오는 여러 가지 센서 값을 받아들이고 이에 대한 적절한 효과자를 제어하는 클라이언터 컨트롤러로 구성되어 있으며, 경주에 참가하는 자들은 클라이언트 컨트롤러를 프로그래밍 하게 된다[2].

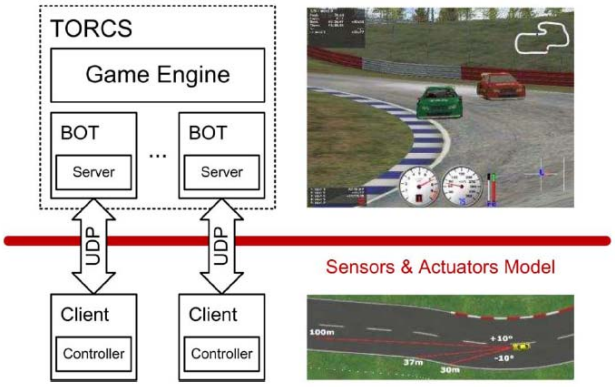

그림 2. 경주를 위해 개발된 API의 아키텍처

2.1. 트랙정보 구축

앞서 말한 바와 같이 차량 운행에 인공지능 적인 요소를 적용하려면 트랙정보의 구축은 필수적인 요소이다. 이러한 요소를 위해 트랙정보를 구축하려면 트랙의 상태를 인식해야 하는 문제가 있다. 트랙의 정보를 인식하기 위해서는 어떤 기준을 정한 후, 주어지는 여러 가지 센서 값들을 토대로 트랙 상태를 계산하여야만 한다. 즉 단순한 차량의 센서 값으로는 트랙의 상태를 파악하기 어렵다. 그러므로 고안한 고수준 센서는 차량과 트랙의 각도 센서와 차량의 전방 거리 센서를 조합하여 현재 차량이 있는 위치에서의 전방과, 양쪽 측면 센서들에서 많은 센서정보를 취할 수 있는 센서 중 전방을 기준으로 90도, 30도 센서를 선정하였다. 이 트랙정보 고수준 센서에 대한 내용은 2.2. 고수준 센서 부분에서 자세히 다룰 것이다.

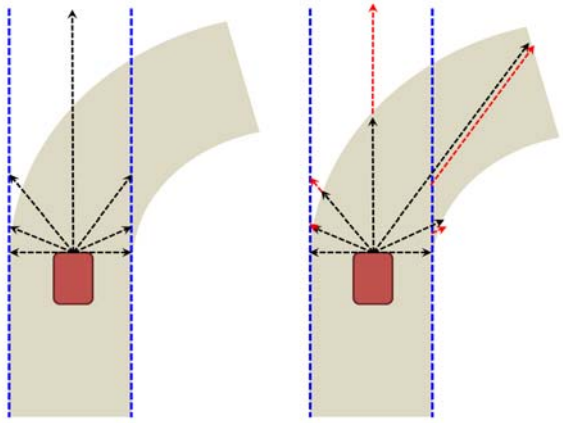

그림 3. 트랙 상태 정보 추출 개념

트랙 상태 정보 추출을 위해 간단한 개념을 살펴보면, 위의 그림에서 좌측의 그림의 검은색 점선 화살표는 직선 코스의 트랙(파란색 점선) 일 때의 예측 센서 값을 나타내며, 오른쪽 그림은 실질적인 센서 값(검은색 화살표)과 예측된 센서 값과의 차이(빨간색 화살표)를 나타낸다. 이러한 차이 값은 실시간 들어오는 센서의 값과 삼각함수를 활용하여 차이 값을 계산할 수 있다. 그러나 여러 가지 노이즈 및 센싱 오차가 있을 수 있으므로, 단순한 차이 값만을 가지고 트랙의 상태를 저장하고 판단하는 데에는 여러 가지 문제가 생긴다. 특히 차량이 펜스에 부딪혀서 멎어져 있는 상태나, 좁은 트랙에서 넓은 트랙으로 바뀔 때 등, 예기치 못한 변수에 의해 단순한 거리 계산만으로 만족할 만한 트랙의 상태 지표로 삼기에는 무리가 있다. 그래서, 본 연구에서는 감지된 센서 값을 활용하여 트랙의 특색 및 형상을 구축할 수 있는 기본 요소로서 트랙의 기울기 값을 계산하였다.

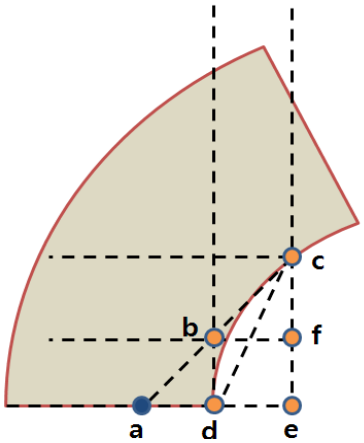

그림 4. 우로 굽은 트랙 설정

먼저, 우로 굽은 트랙의 경우 위의 그림4와 같이 표현하면, a는 차량의 위치이고 두 점 a, d의 거리는 전방 직선방향의 90도 센서 값이고, 두 점 a, c의 거리는 전방 직선방향의 30도 센서의 값이다. 그리고 이 지점에서의 트랙의 기울기를 ω 라고하고, $\angle dab = \theta$라 하면, $\angle cdb = \angle dce = \omega$(엇각)가 되고, 기울기 계산 값은 아래와 같다.

$$\overline{ad} = s_1,\ \overline{ac} = s_2,\ \overline{ab} = \frac{s_1}{\cos\theta},$$
$$\overline{bc} = s_2 - \frac{s_1}{\cos\theta},\ \angle cbf = \theta,$$
$$then,\ \overline{bf} = \cos\theta\left(s_2 - \frac{s_1}{\cos\theta}\right).$$
$$\overline{de} = \overline{bf},\ \overline{ce} = s_2\sin\theta,$$
$$\omega = \arctan\left(\frac{\overline{de}}{\overline{ce}}\right)$$
$$= \arctan\left(\frac{\cos\theta\left(s_2 - \frac{s_1}{\cos\theta}\right)}{s_2\sin\theta}\right)$$

식 1. 우로 굽은 트랙의 기울기 값 계산식

여기서, s_1은 전방에서 90도 방향 센서 값을 말하며 s_2는 30도의 센서 값을 말한다. s_1인 90도 센서 값을 선택한 이유는 기울기를 계산할 때에 삼각함수를 사용하는데, 이때의 밑변의 길이에 해당하는 기본 값으로 이용하며, s_2인 30도 센서의 값은 s_1과 60도의 차이가 나므로 트랙의 상황을 가장 잘 표현하는 값 중 하나로 선택되었다.

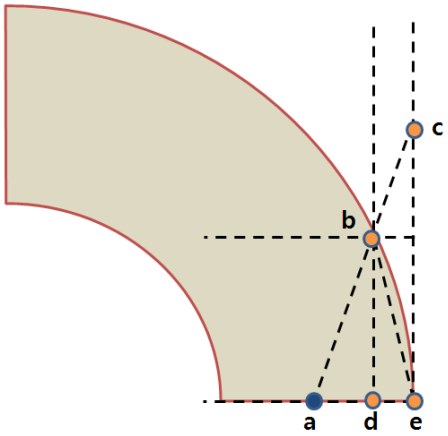

그림 5. 좌로 굽은 도로 설정

좌로 굽은 트랙의 경우 위의 그림5와 같이 표현하면, a는 차량의 위치이고 두 점 a, e의 거리는 전방 직선방향의 90도 센서 값이고, 두 점 a, b의 거리는 전방 직선방향의 30도 센서의 값이다. 그리고 이 지점에서의 트랙의 기울기를 ω 라고하고, $\angle dab = \theta$라 하면, $\angle ceb = \angle ebd = \omega$(엇각)가 되고, 기울기 계산 값은 아래와 같다.

$$\overline{ae} = s_1, \overline{ab} = s_2$$
$$then$$
$$\overline{db} = s_2 \sin\theta,$$
$$\overline{de} = s_1 - s_2 \cos\theta$$
$$\omega = \arctan\left(\frac{\overline{de}}{\overline{db}}\right)$$
$$= \arctan\left(\frac{s_1 - s_2\cos\theta}{s_2\sin\theta}\right)$$

식 2. 우로 굽은 트랙의 기울기 값 계산식

여기서, s_1은 전방에서 90도 방향 센서 값을 s_2는 30도 센서 값을 말한다.

이렇게 계산된 기울기 값은 좌로, 또는 우로 굽은 트랙에서 값의 차이를 두기위해 좌로 굽은 트랙은 기울기 값에 -1을 곱하므로써 양수와 음수로 좌, 우로 굽은 트랙인지 구분 가능하게 했으며, 기울기 값으로 트랙의 굽은 정도를 파악하게 하였다.

이러한 기울기 값은 센서 값이 들어 올 때 마다 기울기 값을 측정하면 데이터 양이 방대해지는 것은 물론, 차량이 펜스에 박혀있는 상황 또는 예기치 못한 상황에서는 잘못된 정보를 계속 저장할 수 있으므로 트랙 시작점에서의 거리 1m 단위로 측정하여 배열에 저장하였으며, 첫 번째 트랙을 완주 하면 모든 트랙의 정보를 축적할 수 있으므로 첫 번째 트랙에서만 트랙정보를 구축하도록 하였으며, 두 번째 트랙 주행 부터는 구축된 트랙정보를 활용하도록 하였다.

2.2. 고수준 센서

본 경기용 서버로부터 많은 센서정보를 전달 받을 수 있다. 그러나 이 많은 센서를 전부 감지하고 일일이 검사하는 것은 많은 비용이 든다. 그래서 우리는 카 레이싱에 필요한 감지 함수만을 정의하여 여러 센서들을 융합한 하나의 고수준 센서로 구축하였다. 아래는 고수준 센서를 표현한 것이다.

- the longest direction (from $-\pi$ to π)
- the shortest direction (from $-\pi$ to π)
- calculated curvature
- typedef struct {
  double s[19];
  double orientation;
  double distFromStart
  double d;
  } aggregatedSensorData

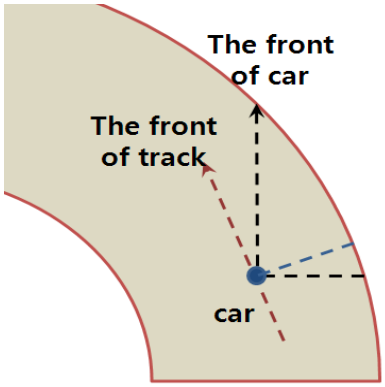

그림 6. 트랙정보 센서 설정

트랙정보를 저장함에 있어서도 고수준 센서를 사용하였다. 올바르게 트랙 정보를 구축하려면, 현재 차량의 위치 점에서 차량의 전방 위치와 트랙의 진행 방향이 정확히 일치하여야 한다. 그러나 위의 그림4에서와 같이 실질적으로 주행하다 보면 일치하지 않는 경우가 많이 발생된다. 이럴 때에는 차량의 전방 방향과 트랙의 방향과의 차이 센서 값인 degree값을 활용하여 선택하여야 할 센서를 수정하여, 우측의 검은색 센서 값이 아닌 파란색 센서 값을 선택하여 계산하면 올바른 트랙의 기울 값을 선택할 수 있다. 이러한 방법으로, 위의 2.1. 트랙정보 구축 부분에서 설명하였듯이, 현재 차량의 위치에서 트랙

의 전방 직진 방향으로 부터의 기울기 값을 계산하는데 전방으로부터 90도, 30도 센서 값을 사용하였으므로, θ 값은 60도가 된다. 이 기울기 계산 센서 모듈을 통하여 트랙 시작점으로부터 1m 마다 트랙의 굽은 방향 및 기울기 값이 표현된 결과 값을 추출할 수 있다.

2.3. 클라이언트 컨트롤러 모듈

앞서 구축된 트랙정보를 활용하여 두 번째 주행부터는 기울기 값을 활용하여 목표속도의 차이를 두었다. 그러나 어느 위치의 기울기 값을 활용하여 목표속도를 결정하는가에 따라 결과의 차이를 보였다. 그래서 휴리스틱을 적용하여 유동적인 기울기 값 선택 알고리즘을 선택하게 되었다. 즉, 빠른 속도로 주행할 때에는 먼거리의 기울기 값을, 느린 속도 주행 할 때에는 가까운 거리의 기울기 값을 선택하여 목표 속도를 선택함에 따라 경주용 차량의 원활한 주행이 가능하게 하였다.

3. 실험 및 결과

실험은 Test Case를 수행하면서 각 변수의 최적값을 구하고, 이 값을 적용하여, 초기 1바퀴를 돌면서 트랙 정보를 수집한다. 이후 2바퀴째부터 구간별 목표 속도를 유지하면서, 레이스를 수행하도록 한다. 초기 1바퀴를 수행할 때는 모든 구간에서 커브를 벗어나지 않은 정도의 속도(100km/h 정도)로 트랙을 주행하도록 하였다. 또한, 특별한 알고리즘을 적용한 방식이 아니고 휴리스틱의 개념을 도입한 테스트 이므로, 커브 각도의 분류와 목표 속도의 최적 값은 여러 번의 테스트 레이싱을 통해 이루어 졌다. 이 테스트에서 문제는, 코스마다 최적 값의 차이가 매우 컸으며, 같은 코스라 하더라도 시도할 때 마다 다른 결과를 가져왔다.

주로 Alpine 2 Course, GC Speedway 1, Wheel 1 에서 테트스 레이싱을 하였으며, Desired Speed Module에서 목표 속도를 수정하는 방향으로 최적화 작업을 진행하였다. 과제내의 Test 결과, 목표속도 1단계 구간은 평균 7도 미만에 목표속도 200, 2단계 구간은 평균 15도 미만에 목표속도 150, 3단계 구간은 평균 23도 미만 100, 4단계 구간은 평균 30도 미만 80, 5단계 구간은 그 이상 50 정도가 적당하였다.

트랙정보(기울기)	목표 속도(시속)
0 ~ 6	200
7 ~ 14	150
15 ~ 22	100
23 ~ 29	80
30 이상	50

표 1. 트랙정보와 목표속도

최종 3개의 트랙에서 트랙의 1회 주행 시에 소요되는 최저 시간을 기존방법, 즉 본 연구 방법을 적용하기 전 방법(TORSC측의 기본적인 배포 소스)과 본 연구를 통해 구현된 모듈을 적용한 방법을 측정한 결과는 아래의 표와 같다.

트랙명	최저시간		차이
	기존방법	트랙정보	
Alpine 2	02:46:79	02:02:13	44:66
GC Speedway 1	01:07:90	01:04:12	03:78
Wheel 1	02:50:76	02:26:50	24:26
총합 (차이 평균)	06:45:45	5:32:75	24:23

표 2. 최저 소요 시간 측정 결과

4. 결론 및 맺음말

본 연구에서는 트랙을 주행함에 있어서 고려해야할 몇 가지 요소를 인공지능적 고찰을 통해 구현해 놓았다. 그리하여, 앞의 결과처럼, 기존의 방법보다 상당부분 성능 향상을 보였다는 점에서 차량 운행의 인공지능 분야에 기여하였다. 그러나, 본 연구에서 제안한 모듈들의 효과성 및 주행 속도와의 원인 분석을 위해서는 트랙의 정보 구축 단계에서 단순히 거리별(m 단위별) 기울기에 그치지 않고, 기울기 정보를 활용하여 트랙을 특정 패턴별로 구획화하는 단계까지 수행하여야 할 것이며, 또한 이렇게 구획화된 정보를 토대로 목표속도를 추출하였으면 좀 더 좋은 성과를 거둘 수 있을 것이다.

또한, TORCS 게임 특성상, 변수와 노이즈에 많이 노출되어 있기 때문에, 한 가지 방법을 갖고 최적화를 하는 것이 한계를 보였다. 좀 더 본 연구 분야에 적합하고 문제 해결에 도움이 되는 다양한 데이터 마이닝 방법들의 시도를 통해 훨씬 더 최적화되고 발전적인 연구 결과를 도출할 수 있을 것이다.

이 논문은 2010년도 정부(교육과학기술부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업 (2010-0012876) 및 뇌과학 원천기술개발사업임 (2010-0018948).

5. 참고문헌

[1] Jan Quadflieg, Mike Preuss, Oliver Kramer, G ¨unter Rudolph, “Learning the Track and Planning Ahead in a Car Racing Controller” in 2010 IEEE Conference on Computational Intelligence and Games

[2] Daniele Loiacono, et al. “The 2009 Simulated Car Racing Championship“, IEEE TRANSACTIONS ON COMPUTATIONAL INTELLIGENCE AND AI IN GAMES, VOL. 2, NO. 2, pp. 131-147, JUNE 2010.

[3] Luigi Cardamone, Daniele Loiacono, Pier Luca Lanzi, and Alessandro Pietro Bardelli, “Searching for the Optimal Racing Line Using Genetic Algorithms“, 2010 IEEE Conference on Computational Intelligence and Games (CIG’10), pp. 388-394, 2010.

[4] E. Onieva, D. Pelta, J. Alonso, V. Milan´es, and J. P´erez, “ A modular parametric architecture for the TORCS racing engine,“ in Proceedings of the 5th international conference on Computational Intelligence and Games. IEEE Press, pp. 256-262, 2009.