EV3의 S/W와 H/W의 단점과 난점을 극복하기 위한 벡터 해석적인 접근

한보람
경희고등학교 2학년

EV3의 S/W와 H/W의 단점과 난점을 극복하기 위한 벡터 해석적인 접근

한보람(3학년8반)

e-mail : ldsboram@naver.com

Vector interpretive approach to overcome the shortcomings and difficulties of S/W and H/W of EV3

Han Boram

요 약

오늘날 과학, 경제, 생명공학 등 여러 분야에서 수학의 쓰임새가 자연과학의 발전에 따라 쓰임새가 많아지고 있는 터이다. 본 논문은 하드웨어나 소프트웨어의 결함을 극복하기 위하여 벡터 해석적인 방법을 두루 활용하여, 컬러 센서의 인식된 색을 결론내릴 수 있는 소프트웨어를 벡터의 개념을 일정 부분 활용하여 사용한 결과, 변화에 적응 가능한 산출물을 얻을 수 있었으며, 옴니 휠을 제어하는 프로그램 또한 벡터 해석적인 방법으로 하드웨어의 난점을 극복할 수 있게 되었음을 보이고, 또한 모터의 하드웨어적 결함을 극복하기 위한 호장 접선 재매개화를 시도할 때도 벡터의 개념을 알고 있어야 접근이 가능했던 것을 토대로 벡터 해석적인 접근이 로봇의 여러 단점과 난점을 유의미하게 극복할 수 있음을 보였다.

1. 서 론

오늘날 실생활 여러 곳에서는 수학의 쓰임새가 점점 높아지고 있다. 특히 과학, 경제, 인공지능 등 여러 분야에서 수학의 쓰임새가 자연과학의 발전에 따라 쓰임새가 많아지고 있는 터이다. 특히 수학에서 중요한 역할을 지탱하고 있는 벡터와 행렬은 그야말로 여러 자연과학의 중추의 역할을 가지고 있다고 말할 수 있을 것이다.

처음에 교육용 로봇을 목적으로 제작된 LEGO사(社)의 MINDSTORM EV3(이하 EV3)는 그 내부 프로그램을 공개한 이후로 세계 여러 개인 및 기업들의 여러 업데이트와 부가적 소프트웨어의 개발 및 새로운 외부 센서 개발과 센서 소프트웨어 개발이 이루어졌다. 그리하여 개인이 직접 로봇을 만들고 그것을 구동시킬 수 있고, 다양한 프로그램을 개발 가능한 활용도 높은 로봇의 대표주자로 자리 잡게 된 지 꽤 되었다. 하지만 그럼에도 불구하고 EV3의 대표적인 소프트웨어와 하드웨어 결점이 여전히 남아 있다. 예를 들면 소프트웨어에서의 특정 모터에 지정 가능한 파워의 한계값은 −100 ~ +100인 반면에 실제로 작동 가능한 파워의 한계값은 −80 ~ +80인 것으로 인해 고속으로 움직이는 로봇에서 소프트웨어에서 입력된 모터 파워값이 실제 구동 모터 파워값과 불일치하여 로봇이 의도하지 않은 방향으로 움직이는 단점과, 색깔 인식 센서의 구동 원리가 센서 자체 내부에서 인식된 색상을 0부터 7(센서에 따라서는 0부터 17) 사이의 자연수에 단순히 대응시켜 받아들여 외부 조명의 밝기나 센서의 높이를 반영할 수 없게 되어 색깔 인식 센서의 오류가 잦은 단점 등이 존재한다.

본 논문은 EV3 로봇의 입력과 출력의 문제, 즉 EV3 색깔 인식 센서의 저성능에 관한 문제와 3축 옴니 휠(Omni wheel) 차체(車體) 구동과 그 단점에 대한 보완에 관한 문제를 다룸에 있어서 기하에서 핵심적인 역할을 하는 벡터와 행렬의 본질을 간략히 파악하고 벡터 해석적인 방법이 어떻게 로봇의 문제점을 인식하고 해결하는 과정으로 사용되었는지를 통해 벡터적인 접근이 로봇의 제어에 효과적임을 말하고자 한다.

특히 벡터의 유용함을 고등학생 수준의 벡터와 행렬로도 쉽게 알 수 있고 추상적인 수학이 실시간으로 움직이는 형태로 시각화하여 벡터와 행렬을 도구로 하는 수학적 사고에 의한 문제 해결 과정이 로봇, 특히 EV3에 어떻게 적용되는지 실시간으로 작동되는 형태로 시각화하여 직관적으로 보여줄 수 있다는 점에서 의의가 있다고 본다.

2. 컬러 센서 인식 성능 향상

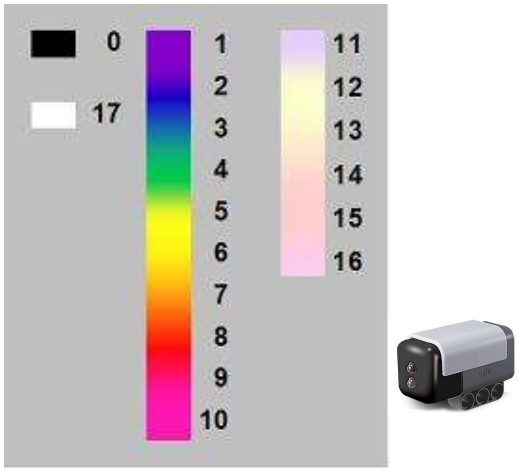
2.1 EV3의 컬러 센서

EV3의 대표적인 컬러 센서에는 크게 2개의 센서가 존재하는데, EV3 Color Sensor(이하 컬러 센서)와 Hitechnic Color Sensor(이하 하이테크 센서)가 있다. 두 센서의 공통점은 자신이 인식한 색깔의 색상을 본체 컨트롤러로 자연수의 형태로 전송한다는 공통점을 보인다. [그림 1]처럼 컬러 센서는 센서 내 자체 회로의 계산을 통해 색깔 미인식 시 0, 흑색은 1, 청색은 2, 녹색은 3, 황색은 4, 적색은 5, 백색은 6, 갈색은 7의 값을 전송한다.

(그림 1) 컬러 센서의 색깔 인식 방식

반면에, 하이테크 센서는 인식한 색깔의 삼원색 값인 RGB값을 토대로 컬러 스펙트럼에서 가장 가까운 색에 대

응되는 0부터 17 사이의 값을 본체 컨트롤러로 전송한다. [그림 2]에서 보듯, 예를 들면 흑색을 인식하면 0을 나타내고, 청색을 인식하면 2, 또는 3을 산출하는 방식이다.



(그림 2) 하이테크 센서의 색깔 인식 방식

2.2 EV3의 컬러 센서의 단점

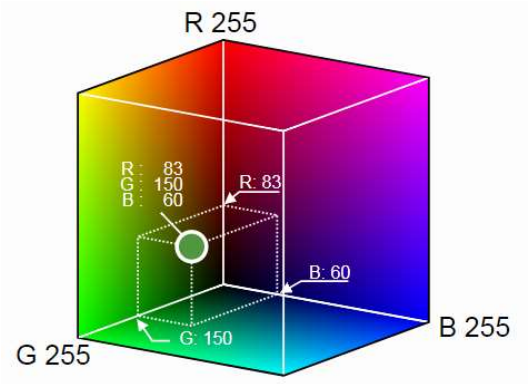
하지만 앞서 말한 것처럼 컬러 센서와 하이테크 센서는 몇 가지 단점을 안고 있는데, 우선 첫 번째에 언급한 컬러 센서는 황색과 갈색을 밝은 환경에서 구분하지 못한다. 이는 컬러 센서의 이용 목적인 일상생활에서의 색깔 인식에 크게 방해되는 것으로서, 치명적인 단점이다. 다음으로, 두 센서 모두 주변의 밝기를 고려하지 않고 색깔 값을 전송하기 때문에 프로그램 제작 환경에서 설정한 색깔 값이 실제 구동 현장에서의 인식되는 색깔 값과 다르게 되어 오류를 일으킬 수 있다. 마지막으로, 두 센서 모두 색깔 인식 기준이 모호한 까닭에 주요 색깔이 불명확하게 구분되어 주변 밝기에 영향을 받는 센서의 단점과 결합하여 프로그램 제작 및 하드웨어 제작에 지장을 초래한다. 물론 컬러 센서의 주변광 센서모드를 사용하여 주변의 밝기를 인식할 수 있지만, 컬러 센서의 인식에 변화를 주는 밝기의 변화가 컬러 센서로 이용되는 주변광 센서에 유의미하지 않은 값으로 감지되기 때문에 이는 좋은 해결책이 아니다.

2.3 색입체와 벡터의 개념을 활용

이 문제를 해결하기 위하여 하이테크 센서에 내장된 고유 기능을 이용하기로 했다. 컬러 센서와는 다르게 하이테크 센서는, [그림 3]과 같이 RGB값을 색깔 값으로 바꾸어 나타내기 전에 RGB 값을 가져올 수 있다. 한 가지 예시를 들어 보면, 하이테크 센서에서는 백색에 해당하는 $(R,G,B)=(255,255,255)$의 입력이 들어오게 된다면, $(255,255,255)$를 [그림 2]처럼 17의 값으로 바꾸어 출력하게 되지만 [그림 3]의 방법으로 이 $(255,255,255)$의 값을 그대로 가져올 수도 있다는 것이다. 이때, (R,G,B)의 값을 R축, G축, B축으로 이루어진 삼차원 직교좌표계의 점 하나에 대응시킬 수 있는데, 이를 [그림 4]와 같이 색입체라고 한다.

[그림 4]에 주어진 색깔은 $(R,G,B)=(83,150,60)$의 입력을 표현한 것으로서, 우리 눈에 약간 어두운 초록색으로 인식되는 색깔이다. 원점$(0,0,0)$에 가까울수록 어두워지며, 원점에서 멀수록 밝아지는 색깔 구성을 띈다.

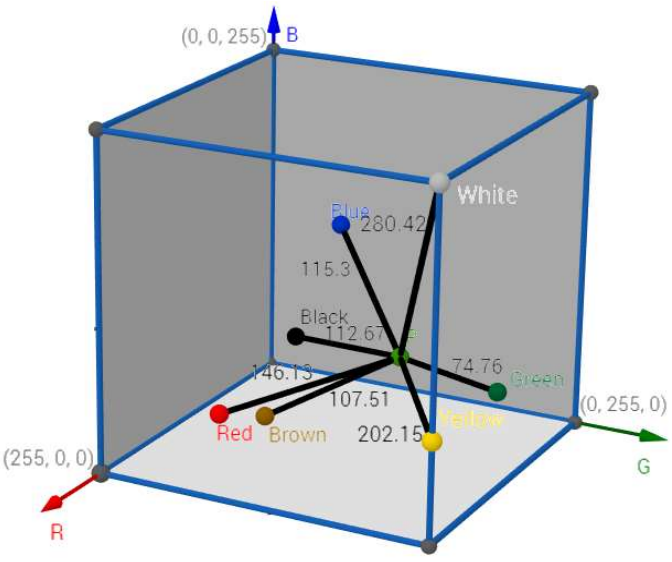
(그림 3) 삼원색 값의 인식



(그림 4) 색입체의 얼개

이때, 하이테크 센서 위에 파란색을 올려놓은 뒤 $(R,G,B)=(38,78,138)$로 인식했을 때, 점$(38,78,138)$을 점 B로 생각하기로 하자, 이때, B의 위치벡터를 $\overrightarrow{OB}$라 하자. 그런데 어떤 색깔 $(R,G,B)=(x,y,z)$를 인식했을 때, 마찬가지로 점(x,y,z)를 점 P로 생각하기로 하자, 이때, P의 위치벡터를 $\overrightarrow{OP}$라 하자. 이때 청색점 B의 위치벡터와 입력색점 P의 위치벡터의 차의 길이인 $|\overrightarrow{OB}-\overrightarrow{OP}|=|\overrightarrow{PB}|$를 청색에 대한 오차도라고 하자.

이때, 흑색점, 녹색점, 황색점, 적색점, 백색점, 갈색점에 대해 똑같은 방법으로 각각의 오차도를 구한다. 그리고 그 오차도 중에서 가장 작은 오차도를 갖는 두 점은 서로 위치가 비슷한 것으로 간주하고 그 색깔을 결과값으로 내놓는다.



(그림 5) 색입체를 활용한 인식 과정의 얼개

[그림 5]는 상기 설명한 과정을 그림으로 쉽게 풀어낸 것이다. 각각의 색깔점은 실제 로봇을 구동하는 현장에서

즉석으로 입력한 것으로 현장의 조명의 밝기와 센서의 상태를 반영하는 색깔점이다. 이때, 주어진 색깔은 $(R,G,B)=(83,150,60)$이 입력되었는데, 이 입력색점으로부터 각각의 색깔점까지의 거리를 잰 결과, 미리 입력한 녹색 색깔점까지의 거리가 74.76으로 가장 적은 거리를 나타내었다. 따라서 입력된 색깔은 초록색으로 간주되어 출력된다.

이러한 방식을 채택하는 소프트웨어를 직접 제작하여 사용한 결과, 얻을 수 있는 가장 좋은 효과는 앞서 언급한 컬러 센서와 하이테크 센서가 안고 있는 몇 가지의 단점을 많이 줄일 수 있다는 것이다. 우선 조명 상태가 달라지거나 센서의 위치가 달라져도 실제 현장에서의 색깔값을 채취할 수만 있다면, 금방 새 환경에 적응하여 안정적으로 동작하는 데에 기여하게 된다. 또한 황색과 갈색이 구분이 힘든 점도 극복하게 된다. 그뿐 아니라, 인식하고자 하는 새로운 색이 필요하게 되면 언제든지 소프트웨어에 색깔점을 추가하는 번거롭지 않은 과정을 통하여 해결할 수 있게 된다.

3. 3축 옴니 휠 차체 제작과 프로그래밍

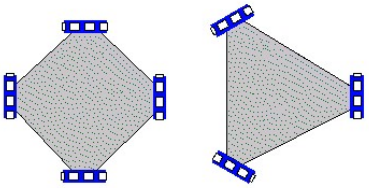
3.1 옴니 휠

EV3의 차체(車體)를 굴러가게 하는 바퀴에 자동차 타이어과 같은 원리로 제작된 고무 타이어가 주를 이루지만, 옴니 휠이라고 불리는 특수한 타이어가 존재한다. 옴니 휠은 [그림 6]에서 보는 것처럼 바퀴 하나에 바퀴 여럿이 더 붙어있는 형태의 바퀴로, 바퀴를 한 차체에 수평이 아닌 방향으로 달았을 때, 일반 고무 타이어는 서로의 속도에 의해 끌리게 되어 마찰이 발생되지만, 옴니 휠은 오히려 서로의 속도의 성분이 수평이 아닐 때에라야 서로 영향을 거의 주고받지 않는다, 즉, 서로 마찰을 발생시키지 않는다.

(그림 6) 옴니 휠과 이를 장착한 3축 옴니 휠 차체

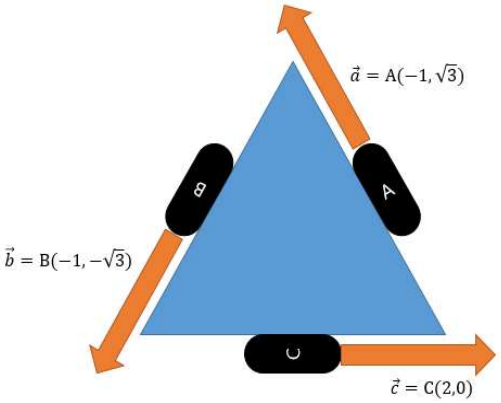
따라서 옴니 휠은 서로의 바퀴에 의해 영향을 주고받지 않는 이상적인 바퀴에 고무 타이어보다 더 가까우므로, 로봇에 달린 옴니 휠의 어느 시점의 순간 속도가 각각의 속도벡터로 나타낼 수 있을 때, 로봇의 속도는 각 바퀴의 속도벡터의 합으로 나타낼 수 있음을 뜻하게 된다.

하지만 실제로 옴니 휠을 달아 로봇을 구동시킬 때 제일 먼저 직면하는 문제는 로봇이 의도된 곳으로 움직이기는커녕 로봇이 제 스스로 회전을 하게 되는 문제가 발생하게 된다. 실제로 두 성분만 생각하여 각각 한 바퀴쌍에 적용하면 잘 움직이는 4축 옴니 휠 차체와는 달리 3축 옴니 휠 차체는 이러한 이유로 제어하기 어렵기로 알려져 있다.



(그림 7) 4축 옴니 휠 차체와 3축 옴니 휠 차체의 얼개

3.2 옴니 휠 차체 제어



(그림 8) 3축 옴니 휠 차체 제어의 얼개

우선 [그림 8]과 같이 차체의 각 옴니 휠에 작용하는 바퀴의 속도벡터의 단위벡터를 계산의 편의성을 위해 단위벡터의 길이를 2로 정하기로 하였다. 이때 [그림 8]에서 각 기본 벡터의 길이(모터 파워)를 A,B,C로 정하고, 옴니 휠의 합으로 인한 차체의 속도벡터를 (x',y')로 정하는 등 조건을 모두 정하고 나면, 다음과 같은 관계가 발생한다.

$$
\begin{aligned}
&(x',y')=\vec{a}+\vec{b}+\vec{c}\\
&=A(-1,\sqrt{3})+B(-1,-\sqrt{3})+C(2,0)\\
&=(-A-B+2C,\sqrt{3}A-\sqrt{3}B)
\end{aligned}
$$

$$
\Rightarrow x'=-A-B+2C,\ y'=\sqrt{3}A-\sqrt{3}B
$$

또한 3축 옴니 휠을 제어하기 어려운 이유로 알려져 있는 로봇이 제 스스로 회전하는 현상은 각각의 속도가 무게중심점을 기준으로 원운동하는 현상에 기인한 것이다. 이로 인해 힘은 아니지만 속도벡터가 돌림힘으로 작용하는 현상 때문에 각각의 회전 속도를 더했을 때, 원하는 회전 속도 S가 나오는 것으로 계산하면, 속도벡터 $\vec{a},\vec{b},\vec{c}$의 방향이 모두 같은 방향임을 고려할 때, 다음과 같은 관계가 추가로 발생한다.(무회전을 의도할 경우 $S=0$이다.)

$$A+B+C=S$$

처음에 발생한 관계에서 얻어낸 두 식과 마지막 식을 연립하면 다음과 같은 연립방정식을 얻어낼 수 있는데, 이는 x',y',S에 의해 A,B,C가 결정되는 일종의 일차변환(행렬식)으로 볼 수 있으므로 연립방정식 형태와 행렬식 형태로 주어진 조건을 정리하면 다음과 같다.

$$\begin{cases} -A-B+2C=x' \\ \sqrt{3}A-\sqrt{3}B=y' \\ A+B+C=S \end{cases}$$

$$\Rightarrow \begin{pmatrix} -1 & -1 & 2 \\ \sqrt{3} & -\sqrt{3} & 0 \\ 1 & 1 & 1 \end{pmatrix}\begin{pmatrix} A \\ B \\ C \end{pmatrix}=\begin{pmatrix} x' \\ y' \\ S \end{pmatrix}$$

$$\Rightarrow \begin{pmatrix} A \\ B \\ C \end{pmatrix}=\begin{pmatrix} -1 & -1 & 2 \\ \sqrt{3} & -\sqrt{3} & 0 \\ 1 & 1 & 1 \end{pmatrix}^{-1}\begin{pmatrix} x' \\ y' \\ S \end{pmatrix}$$

$$\Rightarrow \begin{pmatrix} A \\ B \\ C \end{pmatrix}=\begin{pmatrix} -\frac{1}{6} & \frac{\sqrt{3}}{6} & \frac{1}{3} \\ -\frac{1}{6} & -\frac{\sqrt{3}}{6} & \frac{1}{3} \\ \frac{1}{3} & 0 & \frac{1}{3} \end{pmatrix}\begin{pmatrix} x' \\ y' \\ S \end{pmatrix}$$

행렬의 역할이 일차변환 함수 그 자체임을 생각한다면, 맨 마지막 식 우변에 붙은 3×3행렬은 입력되는 값인 x', y', S에 의해 결정되는 옴니 휠 차체 제어함수 그 자체라고 생각할 수 있다. 이 식에 의하여 A, B, C의 값(모터 파워)은 다음과 같이 결정된다.

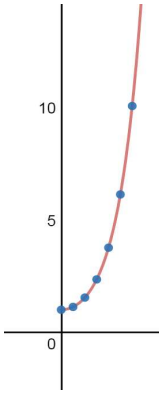
$$A=-\frac{1}{6}x'+\frac{\sqrt{3}}{6}y'+\frac{1}{3}S$$

$$B=-\frac{1}{6}x'-\frac{\sqrt{3}}{6}y'+\frac{1}{3}S$$

$$C=\frac{1}{3}x'+\frac{1}{3}S$$

3.3 호장 접선 재매개화

서론에서 언급한 EV3 소프트웨어에서의 특정 모터에 지정 가능한 파워의 한계값은 $-100 \sim +100$인 반면에 실제로 작동 가능한 파워의 한계값은 $-80 \sim +80$인 것으로 인해 고속으로 움직이는 로봇에서 소프트웨어에서 입력된 모터 파워값이 실제 구동 모터 파워값과 불일치하여 로봇이 의도하지 않은 방향으로 움직이는 단점 때문에, 고속으로 움직이는 구간에서 실제로는 95 내외의 파워를 내야 할 구간에서 실제로는 80으로 움직이는 현상이 발생한다. 수학적으로 아무리 로봇 제어를 잘 하더라도, 실제 현실에서의 물리적 한계로 다른 보완책이 강구되는 상황이다.



(그림 9) 점$((t, \cosh(t)))$의 0.5초 간격 이론상 움직임 (현수선)

[그림 9]와 같이 로봇의 움직임을 $(t, \cosh(t))$로 하기 위하여 $(\frac{dx}{dt}, \frac{dy}{dt})$에 해당하는 $(1, \sinh(t))$인 속도벡터를 적용하고 회전하는 정도를 0으로 두면 몇 초가 되지 않아 속력의 크기(이 경우에는 $\sqrt{1^2+(\sinh(t))^2}=\cosh(t)$)가 상상할 수 없이 치솟게 된다. 이러한 상황에서 모터 파워를 일정 속력 내외로 유지하면서 차체가 움직이는 궤적을 일정하게 만들기 위해 '호장 접선 재매개화'라는 수학적 원리를 덧대기로 하였다.

속도벡터의 크기가 1인 곡선이되 수학적으로 모양이 변형되지 않게 하기 위해서는, 임의의 시점 k까지 이동한 경로의 길이가 t여야 한다. 속도벡터의 크기를 k까지 적분하면 점의 이동 길이가 나오게 되므로 길이함수를 $L(k)$로 두면 다음이 성립한다.

$$L(k)=\int_0^k \sqrt{(\frac{dx}{dt})^2+(\frac{dy}{dt})^2}\,dt=t$$

이때, 시간 t동안 움직인 거리가 t이게 하여 주는 시점 k를 구하려면 길이함수의 역함수를 이용하여 구하여야 한다. 이때 길이함수 L의 역함수를 L^{-1}이라 두면 다음이 성립한다.

$t=L(k)$

$\Rightarrow k=L^{-1}(t)$

따라서 시간 t가 주어지면 움직인 거리가 t이게 만들어 주는 시점 k가 길이함수의 역함수에 의해 정해지고, 정해진 이 시점을 따라 곡선을 따라 움직이면 곡선 전 구간을 속력이 1인 등속도로 움직이게 된다. 이때 재매개화라는 수학적 기술이 요구되는데, 어떤 곡선 $(x(k), y(k))$가 k에 의해 매개되고 있었다면, $k=f(t)$라는 재매개화를 통해 $(x(f(t)), y(f(t)))$로 만들어 곡선의 전체적인 형태는 변하지 않고 매개변수를 바꾸는 과정을 의미하는 것이다. 이때 함수 $k=f(t)$에 제한범위가 존재하면, 재매개화되었을 때 곡선의 형태가 끊어질 수도 있으나, $L(k)$는 길이함수이기 때문에 그 함수도 역함수도 증가함수이고, 따라서 정의역 상과 치역에서 $t \geq 0$인 시점에서 제한범위는 없다고 보아야 한다. 이때, 앞서 점의 속도를 등속도로 만들어 주는 k의 정체가 $k=L^{-1}(t)$임을 알았으므로, 이 함수를 이용하여 재매개화 하면, $(x(L^{-1}(t)), y(L^{-1}(t)))$와 같은 결과를 얻게 된다. 실제로 등속도인지 확인하여 보면 다음과 같다.

$$\sqrt{(\frac{d}{dt}x(L^{-1}(t)))^2+(\frac{d}{dt}y(L^{-1}(t)))^2}$$

$$=\sqrt{(x'(L^{-1}(t))(L^{-1})'(t))^2+(x'(L^{-1}(t))(L^{-1})'(t))^2}$$

$$=\sqrt{(x'(L^{-1}(t))\times\frac{1}{L'(L^{-1}(t))})^2+(y'(L^{-1}(t))\times\frac{1}{L'(L^{-1}(t))})^2}$$

$$=\sqrt{(\frac{x'(L^{-1}(t))}{|v(L^{-1}(t))|})^2+(\frac{y'(L^{-1}(t))}{|v(L^{-1}(t)|})^2}$$

$$=\frac{\sqrt{(x'(L^{-1}(t)))^2+(y'(L^{-1}(t)))^2}}{|v(L^{-1}(t))|}=\frac{|v(L^{-1}(t))|}{|v(L^{-1}(t))|}=1$$

따라서 실제로 속력이 1인 등속도임을 확인할 수 있다. (단, $|v(t)|$는 속도벡터의 길이인 속력함수이다.)

3.4 호장 접선 재매개화의 적용

그러나 대부분의 경우에서 L^{-1}, 즉 길이함수의 역함수를 구하기는 매우 어려우며, 심지어 길이함수를 초등함수(다항함수, 지수함수, 로그함수에 사칙연산 및 합성을 하여 만들 수 있는 함수)로 나타낼 수 없는 함수가 실제로는 훨씬 더 많다. 그래서 길이함수를 초등함수로 구할 수 있는

함수와 그렇지 않은 함수로 나타나는 함수를 이용하여 서로 둘을 비교해 보자.

(1)L^{-1}을 초등함수로 구할 수 있는 경우

쌍곡코사인(현수선)의 움직임인 $(t,\cosh(t))$와 같은 점의 움직임 같은 경우, $(\frac{dx}{dt},\frac{dy}{dt})$에 해당하는 $(1,\sinh(t))$인 길이함수를 구하게 되면 $\int_0^t \sqrt{1^2+(\sinh(t))}\,dt=\sinh(t)$이 나오고, $\sinh(t)$의 역함수를 구하면 다음과 같이 구할 수 있다.

$$s=\sinh(t)=\frac{e^t-e^{-t}}{2}$$
$$\Rightarrow 2s=e^t-e^{-t}\Rightarrow e^t-2s-e^{-t}=0$$
$$\Rightarrow e^{2t}-2se^t-1=0$$

이때 e^t를 이차방정식 근의 공식을 이용하여 풀면,

$$\Rightarrow e^t=s+\sqrt{s^2+1}$$
$$\Rightarrow t=\ln(s+\sqrt{s^2+1})=L^{-1}(s)$$

따라서 주어진 함수를 길이함수의 역함수로 재매개화하면 다음과 같이 주어진다.

$$(x,y)=(\ln(s+\sqrt{s^2+1}),\cosh(\ln(s+\sqrt{s^2+1})))$$
$$\Rightarrow(\frac{dx}{ds},\frac{dy}{ds})=(\frac{1}{\sqrt{s^2+1}},\frac{\sinh(\ln(s+\sqrt{s^2+1}))}{\sqrt{s^2+1}})$$

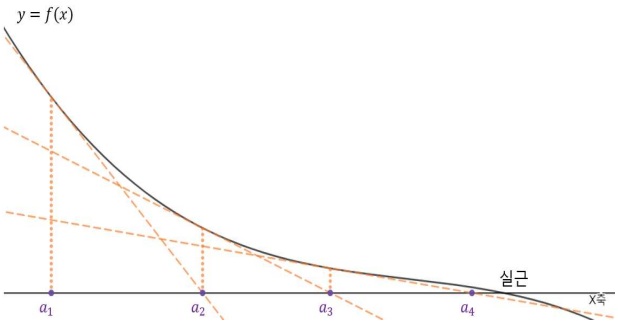
앞서 보인 것처럼 이 속도벡터의 길이를 직접 구해보면 1이 나온다. 실제로 이를 3축 옴니 휠 차체에 움직이도록 구현한 뒤 EV3 본체 컨트롤러 모니터에 언제나 자신의 속도 $(\sqrt{(\frac{dx}{ds})^2+(\frac{dy}{ds})^2})$를 표시하도록 하였더니 언제나 1을 표시하는 것을 확인할 수 있었다.

(2)L^{-1}을 초등함수로 구할 수 없는 경우

L^{-1}을 초등함수로 구할 수 없는 경우, 대부분의 경우에는 길이함수 $L(k)=\int_0^k \sqrt{(\frac{dx}{dt})^2+(\frac{dy}{dt})^2}\,dt=t$ 또한 정적분 기호를 활용하지 않고는 표현할 수 없는 경우가 많다. 따라서 적분의 근삿값인 리만합(곡선 아래의 넓이를 구하기 위해 직사각형 수백 개로 쪼개어 합쳐 구한 넓이)을 구하는 함수를 구하고, 그 함수의 역함수 값의 근삿값을 구하기 위하여 뉴턴의 방법을 활용하는 방법을 생각할 수 있다. 우선 리만합을 구하기 위하여 아래끝을 0, 위끝을 첫째 변수(a), 직사각형의 수를 둘째 변수(n)로 받아내어 적분을 수행하는 다음과 같은 함수 프로그램을 만든다.

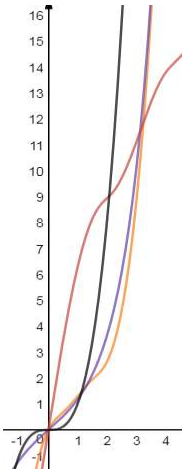
$$\int_0^a f(x)dx\fallingdotseq\sum_{k=1}^{n} f(\frac{ak}{n})\frac{a}{n},(n\geq 100a)$$

또한 함수가 주어졌을 때 그것의 역함수의 근삿값을 구할 수 있는 뉴턴의 방법을 활용하기로 했다. 뉴턴의 방법은 어떤 함수의 근의 근사값을 구할 수 있는 방법 중의 하나이다. 뉴턴의 방법을 이용하여 $L(x)-\alpha=0$의 근의 근삿값을 구할 수 있으므로, $L^{-1}(\alpha)$의 근삿값을 구하는 것과 같은 것이다.



(그림 10) 뉴턴의 방법의 얼개

함수 $f(x)$와 임의의 a_n에 대하여 a_n에서 그은 접선의 x절편 a_{n+1}은 $a_{n+1}=a_n-\frac{f(a_n)}{f'(a_n)}$인 관계가 발생하는데, 이를 반복하면 실제 실근인 $L^{-1}(\alpha)$에 빠른 속도로 효과적으로 수렴하게 된다. 그런데 뉴턴의 방법에서 주의할 점은 초기 가정치인 a_1이 $L^{-1}(\alpha)$에 꽤 가까워야 하며, $f'(a_1)\fallingdotseq 0$인 a_1을 초기 가정치로 두어서는 안 된다는 것이다. 하지만 주어진 함수인 L은 증가함수이고, 위에 설명한 많은 방법들을 쓰는 이유는 상기 설명했듯 로봇의 속도가 고속으로 치닫기 때문에 쓰는 것이기에 거의 언제나 $f'(a_1)\gg 0$이다. 또한 이러한 대부분의 길이함수 L은 치닫는 개형이 x^3과 비슷하기 때문에 초기 가정치는 $a_1=\sqrt[3]{\alpha}$로 두면 초기 가정치를 $L^{-1}(\alpha)$에 꽤 가깝게 둘 수 있다.



(그림 11) 다양한 함수의 길이함수와 흑색 x^3그래프

적색 그래프는 $(t,2t+5\sin t)$의 길이 함수, 자색 그래프는 $(t,\cosh(t))$의 길이함수, 주황색 그래프는 $(t,t(t-1)(t-2))$의 길이함수이다. 모두 흑색 그래프 x^3와 개형의 많이 유사하다는 것을 알 수 있다.

하지만 위의 계산을 모두 처리하는 데 1.0 ~ 1.5초가 걸린다. EV3 시간의 최소단위가 0.001초임을 고려한다면, 이는 약 1000 ~ 1500배의 계산량을 요구하는 특성 때문에 실시간으로 EV3를 사용하여 계산을 해내는 것은 거의 불가능에 가깝다. 따라서 역함수 값을 0.1 ~ 0.3초 단위로 미리 모두 계산한 뒤, 이를 속도식 $(\frac{x'(L^{-1}(t))}{|v(L^{-1}(t))|},\frac{y'(L^{-1}(t))}{|v(L^{-1}(t)|})$에 대입 후 3축 옴니 휠 제어 함수에 대입하여 구동하는

방향으로 나아가기로 하였다. 그런 뒤 EV3 본체 컨트롤러 모니터에 언제나 자신의 속력을 표시하도록 하였더니 다양한 경로에 대하여 0.7～1.5를 표시하는 것을 확인할 수 있었다. 이는 계산의 기반이 거의 모두 근사값에 있기 때문에 이상적인 속도인 1에서 오차가 발생한 것으로 보인다. 이를 보정하기 위해 속도벡터의 길이에 자신의 속도를 나누어 단위벡터로 만들어 운행하였더니, 기존의 경로와 크게 다르지 않으면서 등속도로 운동하는 차체를 관찰할 수 있었다. 그럼에도 불구하고 5초 정도의 가동을 위하여 60초 정도의 역함수 근삿값 계산을 위한 사전 계산 시간을 필요로 하게 되는 단점이 생겼다.

4. 결론 및 제언

하드웨어나 소프트웨어의 결함을 극복하기 위하여 벡터 해석적인 방법을 두루 활용하여, 컬러 센서의 인식된 색을 결론내릴 수 있는 소프트웨어를 벡터의 개념을 일정 부분 활용하여 사용한 결과 변화에 적응 가능한 산출물을 얻을 수 있었으며, 옴니 휠을 제어하는 프로그램 또한 벡터 해석적인 방법으로 하드웨어의 난점을 극복할 수 있게 되었다. 또한 모터의 하드웨어적 결함을 극복하기 위한 호장 접선 재매개화를 시도할 때도 벡터의 개념을 알고 있어야 접근이 가능했다. 따라서 벡터 해석적인 접근이 로봇의 여러 단점과 난점을 어느 정도 극복할 수 있음을 보일 수 있었다.

그럼에도 불구하고 계산 기능의 성능 저하는 벡터 접근적인 방법으로도 극복이 힘들었고, 또한 벡터를 이용하여 문제를 해결하는 것은 다시 말하자면 계산에 더욱 의존하는 것이 되므로, 이에 따라 EV3의 근본적인 계산 기능, 즉 처리 속도가 향상되지 않으면 벡터 해석적인 접근으로도 한계가 있게 된다.

본 논문은 EV3의 단점을 극복하는 데 벡터 해석적인 접근이 유용하게 쓰일 수 있음을 보이기도 하였고, 또한 근본적인 계산 성능이 향상되지 개선된다면 벡터 해석적인 접근과 합쳐져 EV3의 단점을 획기적으로 개선함에 이를 것으로 보이므로, 근본적으로 계산 성능을 극복할 수 있는 다른 접근의 후속 연구가 필요함을 알게 되었다.

ABSTRACT

In many places of life today, the use of mathematics is increasing. In particular, the use of mathematics in various fields such as science, economics, and biotechnology is becoming more and more popular as the natural science advances. This paper briefly grasps the essence of vectors and matrices that play a key role in mathematics, and explains the vector approach is effective for controlling the robot through how vector-based methods are used in recognizing and solving the problems of input and output of EV3 robots.

As a result of using vector analytical methods to overcome defects in hardware and software, many defects could be overcome in many areas. The result of using the concept of the vector which is able to conclude the recognized color of the color sensor with certainty was obtained satisfactory result, the program that controls the omni wheel can also overcome the hardware difficulties in a vector interpretive manner. Also, when attempting to use re-parameterization to create an unit speed curve to overcome the hardware defect of the motor, it was necessary to know the concept of the vector. Therefore, it can be said that the vector was used successfully to overcome the shortcomings and difficulties of EV3.

Nonetheless, the decline of the computation function could not be avoided by the vector approach, because it was a problem that was closer to engineering and physical part rather than mathematical part. Therefore, although it was able to achieve some degree of success in overcoming the disadvantages and difficulties of EV3, it came to the conclusion that it was impossible to overcome it completely. In addition, solving problems using vectors means being more dependent on mathematics, so it relies more heavily on the computation function of EV3. Thus, if the underlying computational function of the EV3, that is, the processing speed is not improved, the vector interpretation approach will show its limitations again.

참 고 문 헌

[1] 김상대, 조영완, 김승우. "옴니휠을 이용한 이동 로봇 개발에 관한 연구." 제어로봇시스템학회 국내학술대회 논문집, (2009.9): 676-680.

[2] 김상대, 김승우, 조영완. "자이로센서 기반의 전방향 로봇 자세제어에 관한 연구." 제어로봇시스템학회 합동학술대회 논문집, (2009.12): 177-180.

[3] 심호석, 김동우, 박석순, 이수철. "옴니휠을 이용한 1인 탑승 로봇 개발." 한국재활복지공학회 학술대회 논문집, (2011.11): 171-174.

[4] 김진욱, 정유철, 고윤호. "복도 환경에서 자율 주행이 가능한 옴니-휠 기반의 순찰 로봇." 정보 및 제어 논문집, (2012.10): 421-422.

[5] 연지흠, 하윤철, 공은혜, 정민영, 정명진. "전방향 이동 가능한 균형 볼 로봇." 대한기계학회 춘추학술대회, (2017.11): 2667-2668.

[6] 신혜원, 이순걸, 김병수, 문상찬, 김형. "옴니휠 타입 중수로 로봇의 동역학적 모델링 및 해석." 한국정밀공학회 학술발표대회 논문집, (2013.5): 43-44.

[7] 박재한, 김순철, 이수영. "전방향 이동성을 갖는 안정한 볼봇 개발." 제어로봇시스템학회 논문지, 19.1 (2013.1): 40-44.

[8] 진태석. "전방향 구동이 가능한 이동로봇 설계 및 운동제어." 한국지능시스템학회 학술발표 논문집, 26.1 (2016.4): 242-243.

[9] 박동배, 이희진, 차동혁. "전방향 로봇 시스템의 개발." 제어로봇시스템학회 국내학술대회 논문집, (2008.10): 798-802.

[10] 이정형, 이형직, 정슬. "힘 반향 기법을 이용한 전방향 이동 로봇의 원격 제어." 정보 및 제어 논문집, (2007.4): 243-245.