한국군사과학기술학회지 제14권 제3호, pp. 327~334, 2011년 6월

학술논문 지상무기 부문

Modified A* - 방향별 속도지도를 활용한 무인차량의 지역경로계획

Modified A* - Local Path Planning Method using Directional Velocity Grid Map for Unmanned Ground Vehicle

이 영 일* 이 호 주* 박 용 운*
Young-il Lee Ho-Joo Lee YongWoon Park

Abstract

It is necessary that UGV(Unmanned Ground Vehicle) should generate a real-time travesability index map by analyzing raw terrain information to travel autonomously tough terrain which has various slope and roughness values. In this paper, we propose a local path planning method, MA*(Modified A*) algorithm, using DVGM (Directional Velocity Grid Map) for unmanned ground vehicle. We also present a path optimization algorithm and a path smoothing algorithm which regenerate a pre-planned local path by MA* algorithm into the reasonable local path considering the mobility of UGV. Field test is conducted with UGV in order to verify the performance of local path planning method using DVGM. The local path planned by MA* is compared with the result of A* to verify the safety and optimality of proposed algorithm.

Keywords : LPP(Local Path Plan), MA*(Modified A*), DVGM(Directional Velocity Grid Map), Autonomous Navigation, UGV(Unmanned Ground Vehicle)

1. 서 론

무인차량의 성공적인 임무수행을 위해서는 주어진 목표점까지 안전하고 효율적으로 이동하는 자율주행 기술이 선행되어야 한다. 무인차량이 본연의 효용적 가치를 가지기 위해서는 인간의 접근과 개입이 제한되는 험지 및 야지와 같은 거친 외부환경에서 임무를 완수해야하며, 이는 알려지지 않은 험지 및 야지에서의 자율주행 능력이 필요함을 의미한다. 이를 위한 가장 근본적이며 필수적인 기술 중 하나는 지형감지용 센서로부터 획득한 실시간 데이터를 이용하여 주행성 분석용 정보를 생성하고, 이를 활용하는 경로계획 관련 기술이다.

외부환경에서 운행하는 무인차량의 주행성 분석용 정보생성과 관련된 기존 방법으로는 점유격자(Occupancy Grid) 지도[1]와 주행성격자(Traversability Grid) 지도[2~5]가 있다. 점유격자 지도는 장애물에 대한 공간적 정보를 격자지도에 확률적으로 표현하는 방법이며, 주행성

† 2011년 2월 28일 접수~2011년 5월 13일 게재승인
* 국방과학연구소(ADD)
책임저자 : 이호주(hojoolee@yahoo.com)

격자 지도는 다양한 센서들을 활용하여 주행환경에 대한 고도자료가 포함된 3차원 지형지도를 생성하고 이 지형지도를 기반으로 주행성 분석을 통해 주행성 지도를 생성하는 방법이다. 무인차량의 주행성분석을 위한 두 가지의 방법은 명백한 장단점이 존재하며, 두 방법의 단점을 보완할 수 있는 방향별 속도지도(DVGM : Directional Velocity Grid Map) 생성에 관한 방법을 이전의 연구에서 제안하였다[6]. DVGM이란 무인차량에 장착된 지형감지용 센서로부터 탐지한 영역을 특정 크기의 격자로 나누고 각 격자에서 무인차량이 주행 가능한 속도를 지형 경사도 및 거칠기 그리고 장애물 정보를 반영하여 퍼지추론 방법을 통해 방향별로 산출하여 생성한 속도지도를 의미한다.

본 논문에서는 무인차량의 자율주행시스템을 위한 방향별 속도지도(DVGM) 기반의 지역경로계획 기법에 대해 언급한다. 특정치 이상의 경사도와 조도가 반복적으로 존재하는 야지에서 자율적으로 주행하기 위한 첫 단계는 주행환경에 대한 실시간 분석을 수행하는 것으로, 이를 위해 본 논문에서는 격자기반의 속도지도인 DVGM을 생성한다. 생성된 DVGM을 기반으로 무인차량이 실질적으로 주행하기 위해서는 주어진 목표점까지의 경로를 생성하는 지역경로계획기법이 요구되며, 격자기반의 지도인 DVGM에서 활용될 수 있는 대표적인 지역경로계획 기법으로는 A* 알고리즘[7]이 있다. 가장 널리 알려진 휴리스틱 탐색기법 중 하나인 A* 알고리즘은 이진법적인 장애물의 유무를 표현하는 장애물 격자지도상에서 이동경비관점의 거리우선 탐색을 수행한다. 하지만, DVGM은 장애물의 유무 정보만을 표현하는 격자지도가 아니며 또한, 각 격자에는 무인차량의 방향별 주행속도 정보를 포함하고 있어 거리우선 탐색이 아닌 속도우선 탐색을 수행할 수 있는 수정된 지역경로계획 기법이 요구된다.

본 논문에서는 이러한 요구사항이 반영된 DVGM 기반의 지역경로계획 기법인 Modified A*(MA*) 알고리즘을 제안하며, MA*를 활용하여 격자지도 기반으로 계획된 경로를 무인차량의 운동성을 고려한 주행가능 경로로 재생성하는 경로최적화(Path Optimization) 및 경로곡선화(Path Smoothing) 기법을 제안한다. 제안된 지역경로계획 기법의 성능은 무인차량을 활용한 시나리오 기반의 필드테스트를 통해 그 결과를 분석 및 검증한다.

2. 방향별 속도지도(DVGM) 생성

본 장에서는 무인차량이 험지 및 야지를 포함하는 거친 환경을 자율적으로 주행하기 위한 첫 단계인 주행성 분석에 대한 기존의 연구결과인 DVGM 생성 방법[6]을 간략하게 소개한다. Fig. 1에 보이듯 DVGM 생성의 출력결과인 특정 격자 $G_{(x,y)}$의 특정방향에 대한 추정 주행속도 $\bar{v}$ 산출에 영향을 미치는 입력 인자는 경사도 및 거칠기와 같은 지형적 특성, 그리고 장애물 정보로 구성된다. DVGM 생성시스템은 해당 격자 및 진행방향에 놓인 경사도 및 거칠기 정보를 반영하여 각각의 인자에 대한 퍼지집합의 단일값 및 단일값에 대한 소속함수값을 산출하며, 최종적으로 무게중심법을 이용한 비퍼지화를 통해 무인차량의 운용모드에 무인차량의 적정 주행속도를 계산한다.

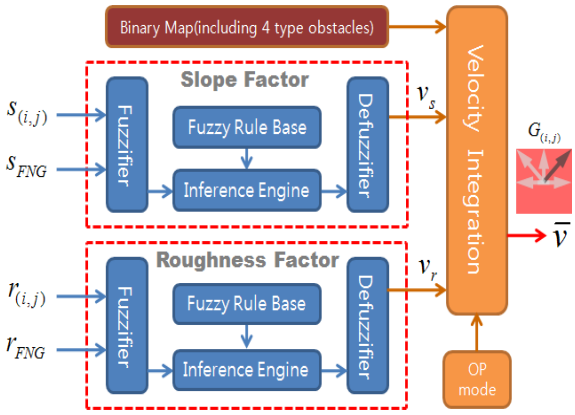

Fig. 1. DVGM 생성을 위한 퍼지추론

가. 경사도 인자에 의한 속도 추정

격자 $G_{(x,y)}$의 특정 방향에 대한 주행속도를 산출하기 위한 인자로 지형에 존재하는 경사도[8,9]를 반영한다. 해당 격자의 특정방향 경사도 $s_{(i,j)}$와 특정방향의 예측된 전방 패치 정보인 FNG(Front Neighborhood Grids) 경사도 s_f를 입력으로 준다. $s_{(i,j)}$와 s_f는 정규화된 영역 [0, 1]에서 각각 3개의 퍼지집합 {FLAT, SLOPED, STEEP}으로 분할되며, FLAT는 평평한 지형을, SLOPED는 약간 기울어진 지형을 그리고 STEEP는 가파른 지형을 의미한다. 출력 변수인 v_s는 4개의 퍼지집합 {VERY-SLOW, SLOW, FAST, VERY-FAST}로 분할되며, 안전우선 모드 및 속도우선 모드에 따라 차별화된 퍼지규칙을 적용한다.

나. 거칠기 인자에 의한 속도 추정

노면의 지질학적 불규칙성을 의미하는 지형의 거칠기 또한 특정 격자의 주행속도를 산출하기 위해 반영되는 중요한 지형정보이다[10~12]. 해당 격자의 특정방향 거칠기 $r_{(i,j)}$와 특정방향의 FNG 거칠기 r_f를 입력으로 준다. $r_{(i,j)}$와 r_f는 정규화된 영역 [0, 1]에서 각각 3개의 퍼지집합 {SMOOTH, ROUGH, ROCKY}으로 분할되며, SMOOTH는 조도가 없는 지형을, ROUGH는 약간의 조도를 포함한 지형, 그리고 ROCKY는 아주심한 조도를 포함하는 지형을 의미한다. 출력 변수인 v_r은 경사도에 의한 산출된 주행속도 v_s와 동일한 소속함수를 가지며, 퍼지규칙 역시 경사도 입력 인자에 의한 추론과 같이 안전우선 모드 및 속도우선 모드에 따라 차별화된 퍼지규칙을 적용한다.

다. 운용모드에 따른 최종 주행속도 통합

DVGM 생성을 위한 퍼지시스템의 입력은 경사도 정보와 거칠기 정보로 나뉘며 각 입력인자에 해당하는 정보를 반영한 적정 주행속도를 무게중심법을 이용한 비퍼지화를 통해 산출하고 이를 무인차량의 다양한 임무모드에 특성화된 방법으로 통합한 후 무인차량의 최종 주행속도를 추론해 낸다. 속도우선 모드는 가장 기본적인 모드로 무인차량이 가장 빠르게 이동할 수 있는 지역경로설정을 위한 DVGM 생성 모드로 식 (1)과 같이 최종속도를 추정한다. 안전우선 모드는 야지나 험지와 같은 지형을 주행해야할 경우 가장 안전하게 이동할 수 있는 지역경로설정을 위한 DVGM 생성 모드로서 식 (2)와 같이 최종속도를 산출한다.

$$\bar{v} = \frac{\sum V_p^s A_p^s + \sum V_p^r A_p^r}{\sum A_p^s + \sum A_p^r} \quad (1)$$

$$\bar{v} = \min\left(\frac{\sum V_p^s A_p^s}{\sum A_p^s}, \frac{\sum V_p^r A_p^r}{\sum A_p^r}\right) \quad (2)$$

3. DVGM 기반의 지역경로계획 기법

이진법적인 장애물의 유무정보만을 표현하는 장애물 격자지도를 활용하는 A* 알고리즘은 가장 널리 알려진 휴리스틱 탐색기법으로 '언제나 최적의 경로를 산출한다'는 허용성(Admissibility)이 검증된 알고리즘이다[7]. 그러나 A* 알고리즘은 장애물 격자지도상에서 이동경비관점의 거리우선 탐색을 수행하기 때문에, 각 격자에 무인차량의 방향별 주행속도 정보를 저장하고 있는 DVGM에 직접적으로 적용하기에는 적합하지 못하다. 따라서 본 장에서는 방향별 주행속도 정보를 포함하고 있는 DVGM을 활용하여 무인차량의 지역경로계획이 가능하도록 수정된 MA* 알고리즘을 제안한다. 또한 MA*를 활용하여 격자지도 기반으로 계획된 경로를 무인차량의 운동성을 고려한 주행가능 경로로 재생성하는 경로최적화 및 경로곡선화 알고리즘을 소개한다.

가. 표기법(Notation) 및 정의(Definition)

무인차량의 지역경로계획 기법에서 사용되는 알고리즘에 대한 정확하고 일관성 있는 논의를 위해 알고리즘 기술에 사용되는 표기법을 먼저 정의할 필요가 있는데 Table 1은 이를 보여준다.

Table 1. 알고리즘 기술용 표기법 및 정의

Notation	Definition
Neighbor(n)	격자 n에 인접한 격자들의 집합
Dist(n_1, n_2)	격자 n_1에서 격자 n_2까지의 격자기반 거리
$Dist_{min}$(n_1, n_2)	격자 n_1에서 격자 n_2까지의 최소거리
RC(n_1, n_2)	격자 n_1에서 격자 n_2로 회전 시 소비경비
V_{n-1}	DVGM의 격자 n-1에서 격자 n 방향으로 이동 시 격자 n-1에 저장된 추정속도
V_{avr}	DVGM의 전체 평균속도
G(x, y)	DVGM의 (x, y)에 놓인 격자
(x_0, y_0)	DVGM에서 무인차량의 현재 위치로 G(0, 0)
ΔG	DVGM의 한 격자 크기
f(n)	MA* 알고리즘의 평가함수로 f(n)=g(n)+h(n)
P_{num}	생성된 Path를 구성하는 포인트(격자)의 개수
P_i(x, y)	Path를 구성하는 i번째 포인트 G(x, y)
P'$_i$(x, y)	현재까지 smoothing된 path를 구성하는 포인트들 중 i번째 포인트 G(x, y)
x_{PS}, y_{PS}	P_S(x, y)의 x & y 좌표값
x_{PE}, y_{PE}	P_E(x, y)의 x & y 좌표값
Line[P_S, P_E]	P_S와 P_E로 이루어진 격자기반 LPP 경로
NBL[P_S, P_E]	P_S와 P_E로 구성된 장애물 없는 직선경로
SL_n	SegmentedLine을 구성하는 n번째 포인트

나. Modified A* 알고리즘

MA* 알고리즘은 기존의 A* 알고리즘을 수정하여 DVGM상에서 최소이동시간 관점의 지역경로계획이 가능하도록 변경한 알고리즘이다. Fig. 2에 보이듯, MA* 알고리즘의 기본적인 구조는 A* 알고리즘과 동일하나, 이동경비관점의 거리우선탐색이 아닌 주행속도우선탐색이 이루어져야 하는 차이점이 있다. 이를 위해 식 (4)에 보이는 것처럼, 무인차량의 시작점인 G(x_0, y_0)에서 특정 격자 G_n까지의 경비계산에 이동거리가 아닌 DVGM의 격자에 저장된 속도정보를 시간 도메인으로 변환시킨 주행시간을 활용한다. 또한 특정 격자 G_n으로부터 무인차량의 목표점인 G_{goal}까지의 최소경비를 추정하는 h(n) 계산에도 식 (5)에서처럼, 거리 기반이 아닌 DVGM의 평균속도를 이용하여 추정한 최소주행시간을 사용한다.

```
ModifiedAstar()
01. WHILE OPEN != empty
02.     pick n_best from OPEN such that f(n_best)≤f(n),
          ∀n ∈ OPEN;
03.     remove n_best from OPEN and add to
          CLOSED;
04.     IF(n_best == Goal) exit;
05.     expand n_best for all x ∈ Neighbor(n_best)
          that are not in CLOSED;
06.     IF(x ∉ OPEN)
07.         add x to OPEN;
08.     ELSE IF( g(n_best)+Dist(n_best, x)<g(x) )
09.         update x's backpointer to point to n_best;
f(n)
10. g(n) = g(n-1) + Dist(n-1, n) * RC(n-1, n) / V_n-1;
11. h(n) = Dist_min(n, Goal) / V_avr;
12. EF = g(n) + h(n); return EF;
```

Fig. 2. Modified A* 알고리즘 의사코드

$$f(n) = g(n) + h(n) \tag{3}$$

$$g(n) = g(n-1) + \frac{d(G_{n-1}, G_n)}{V_{G_{n-1}}} \tag{4}$$

$$h(n) = \frac{d_{\min}(G_n, G_{goal})}{\overline{V}} \tag{5}$$

where, $d(G_{n-1}, G_n)$: the length of grid from G_{n-1} to G_n, $d_{\min}(G_n, G_{goal})$: the minimum length from G_n to goal, $V_{G_{n-1}}$: the estimated speed at G_{n-1} of DVGM heading to G_n, $\overline{V}$: the average speed of DVGM

그러나 시작점에서 특정 격자 G_n까지의 경비를 산출하는 g(n) 계산식에 식 (4)와 같이 DVGM의 속도정보에 기반을 둔 주행시간만을 활용할 경우, Fig. 3과 같은 문제점이 발생한다. Fig. 3의 Goal 주변에서 이루어진 경로계획 결과(붉은 실선)를 분석해보면, 이동시간 관점에서는 최적화된 경로계획 결과이지만 무인차량이 이동시 소비되는 전체적인 경비관점에서는 빈번하고 심한 조향에 의한 이동경비 증가로 최적의 경로계획 결과라 할 수 없다. 따라서 g(n)값을 산출하는 식 (4)는 무인차량의 조향에 대한 경비값(RC : Rotational Cost)[13]을 포함하는 식 (6)으로 변경되어야 한다. 조향에 대한 경비값 RC는 격자기반의 MA* 알고리즘 특성상 45도 단위로 산출되며 선형적인 값이 될 수 있다.

$$g(n) = g(n-1) + \frac{d(G_{n-1}, G_n) \times RC(G_{n-1}, G_n)}{V_{G_{n-1}}} \tag{6}$$

where, $RC(G_{n-1}, G_n)$: rotational cost for moving from G_{n-1} to G_n

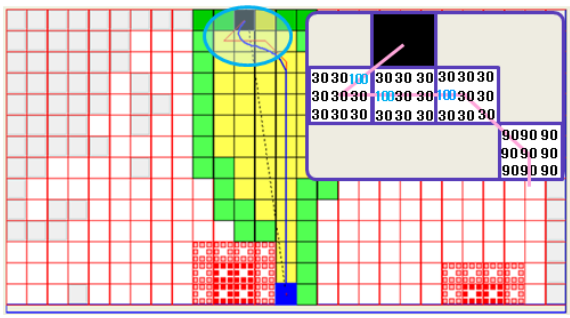

Fig. 3. MA*의 평가함수에서 RC의 필요성

다. Path Optimization 알고리즘

무인차량의 지역경로계획을 수행하는 MA* 알고리즘은 격자지도를 활용한다는 특수성으로 인해 무인차량이 선택할 수 있는 조향 값이 45도 단위로 한정될 수밖에 없는 문제점을 지닌다. 이러한 이유로 주어진 *Goal*까지 생성된 지역경로는 실질적인 이동시간관점의 최적경로가 되지 못하며, 또한 잦은 조향으로 인

해 추종성의 저하를 초래하게 된다. 제안하는 Path Optimization 알고리즘은 이러한 문제를 해결하기 위한 알고리즘으로 크게 두 단계로 구성된다. 먼저 계획된 지역경로를 구성하는 포인트들의 집합에서 각각의 포인트 $P_i(x, y)$에 대해 다른 포인트들과 생성 가능한 모든 경우의 직선을 추출한 후 장애물과 접촉하지 않으면서 길이가 가장 긴 직선을 선정한다. 다음 단계로, 첫 단계에서 추출된 두 포인트 사이의 직선경로 기준의 이동시간과 MA* 알고리즘에 의해 기 계획된 경로 기준의 이동시간을 비교하여 첫 단계에서 선정된 직선의 이동시간이 적거나 같을 경우, 두 포인트 사이의 경로를 격자단위를 무시한 직선화(Pruning)를 통해 경로 최적화를 수행한다. Fig. 4는 해당 알고리즘의 의사코드이다. 식 (7)은 지역경로를 구성하는 특정 두 포인트 사이의 경로에 대한 직전화 여부를 판별하는 수식이다.

```
BlockingCheck(PS(x, y), PE(x, y))
01. slope = (yPE − yPS) / (xPE − xPS);
02. FOR(i = xPS to xPE)
03.       y_lower = round(yPS + (i − xPS − 0.5)*slope);
04.       y_upper = round(yPS + (i − xPS + 0.5)*slope);
05.       IF(i == xPS) y_lower = yPS;
06.       IF(i == xPE) y_upper = yPE;
07.       FOR(j = y_lower to y_upper)
08.            IF(G(i, j) == obstacle)
09.                 return BLOCKED; break;
10. return NOT_BLOCKED;
Pruning(PS(x, y), PE(x, y))
11. Tplanned = Dist(Pn,Pn+1) / VPi, ∀n ∈ Line[PS, PE];
12. Tprune = Dist(SLn,SLn+1) / VSLn, ∀n ∈ NBL[PS, PE];
13. IF (Tprune ≤ Tplanned) return YES;
PathOptimization()
14. FOR(i = 0 to i < Pnum)
15.       FOR(j = Pnum to j > i)
16.            IF (BlockingCheck(Pi(x, y), Pj(x ,y)) ==
                NOT_BLOCKED) &&
                (Pruning(Pi(x, y), Pj(x ,y)) == YES))
17.                 remove points between Pi and Pj;
18.                 i = j; break;
```

Fig. 4. Path Pruning 알고리즘 의사코드

$$Pruning(P_n, P_{n+k}) = \left\{ \begin{array}{l} 1,\ if \sum_{i=n}^{NBL} \left(\frac{d_{NBL_i}}{v_{NBL_i}} \right) \le \sum_{i=n}^{n+k-1} \left(\frac{d(P_i, P_{i+1})}{v_{P_i}} \right) \\ 0,\ others \end{array} \right\} \quad (7)$$

where, SL : the number of points composing NonBlockedLine, SL_n : n_{th} point of NonBlockedLine

라. Path Smoothing 알고리즘

Path Optimization 알고리즘을 통해 지역경로를 최적화하여도 장애물의 배치에 따라서는 최대 90도의 조향이 요구되는 경로의 포인트가 존재할 수 있다. 다시 말해 최적화된 경로를 무인차량의 운동성을 고려한 추종 가능한 형태의 경로로 재생성하는 알고리즘이 요구된다는 의미이다. 제안하는 Path Smoothing 알고리즘은 Bezier Spline 알고리즘을 활용하는데 이는 3차 Spline 알고리즘 중 가장 보편적이고 곡선 제어가 용이한 알고리즘으로, 네 개의 제어점을 사용하여 첫 번째 포인트에서 네 번째 포인트를 잇는 곡선을 산출한다.

Path Optimization 알고리즘을 적용하여 직선화된 경로를 구성하는 모든 포인트 $P_i(x, y)$들은 무인차량의 조향이 요구되는 *EDGE*가 되는데 이는 Bezier Spline 알고리즘을 적용하는 기준점이 된다. *EDGE*를 기준으로 현재까지 Smoothing된 경로에서 마지막 두 포인트를 추출하고 세분화(Segmentation)시킨 최적화 경로에서 다음 두 포인트를 추출하여 Spline 알고리즘에 입력한 후 첫 번째 포인트와 네 번째 포인트를 반드시 지나는 추종성이 확보된 경로를 재 생성한다. Fig. 5는 해당 알고리즘의 의사코드이다.

```
Initialize()
01. Spline(P0(x, y), G(0, 1), P2(x, y), P3(x, y));
02. tag P3(x, y) with EDGE;
Spline(P0, P1, P2, P3);
03. Snum = sqrt((xP3 − xP0)^2 + (yP3 − yP0)^2) / Ssize;
04. FOR(n = 0 to Snum)
05.       xSPLINE = xP0*(1 − n)^3 + 3*xP1*n(1 − n)^2 +
                    3*xP2*n^2*(1 − n) + xP3*n^3;
06.       ySPLINE = yP0*(1 − n)^3 + 3*yP1*n(1 − n)^2 +
                    3*yP2*n^2*(1 − n) + yP3*n^3;
07.       DrawPoint(xSPLINE, ySPLINE);
PathSmoothing()
08. tag all points ∈ optimized path with EDGE;
09. segment the optimized path into ΔG;
10. Initialize();
11. FOR(n = 3 to Pnum)
12.       IF Pn(x, y) == EDGE
13.            Spline(P'n−2(x, y), P'n−1(x, y), Pn+1(x, y),
                    Pn+2(x, y));
14.            n = n + 2;
```

Fig. 5. Path Smoothing 알고리즘 의사코드

4. 시뮬레이션 결과 및 분석

본 논문에서 제안한 DVGM 기반의 지역경로계획 기법인 Modified A* 알고리즘의 성능검증을 위해 무인자율차량을 이용한 시험을 수행한다. 시험환경은 비교적 노면의 상태가 양호한 야지인 Soft Terrain이며, Fig. 6과 같이 Soft Terrain에 큰 경사도를 가지는 지형장애물을 인공적으로 설치하였다. 지형장애물의 양쪽에 전역경로점을 부과하여 무인차량이 반드시 지형장애물과 조우하는 시나리오를 작성한 후, DVGM 기반의 MA* 알고리즘과 Binary Map 기반의 A* 알고리즘을 이용하여 무인차량의 자율주행을 수행하여 그 결과를 비교 · 분석한다.

Fig. 6. 시험환경 및 지형장애물 배치

Fig. 7과 Fig. 8은 Binary Map 기반의 지역경로계획 기법인 A* 알고리즘을 활용하여 무인차량의 자율주행을 수행한 결과를 보여준다. 무인차량은 쌍안 카메라 및 레이저거리측정기와 같은 지형감지센서로부터 획득한 고도정보를 활용하여 장애물의 유무정보를 표현하는 Binary Map을 생성하며, 이를 기반으로 A* 알고리즘을 수행한다. 인공적으로 설치한 지형장애물은 점차적으로 증가하는 경사도를 가지는데, 이웃 격자간의 높이차를 기준으로 장애물을 판단하는 Binary Map에서는 장애물로 인식되지 못한다. 따라서 무인차량은 Fig. 8과 같이 지형장애물을 넘어가는 지역경로를 생성하게 되며, Fig. 7에 보이는 것처럼 무인차량이 지형장애물을 타고 넘는 자율주행을 수행함으로서 무인차량의 안전성을 저하시키는 요인으로 작용한다.

Fig. 9와 Fig. 10은 DVGM 기반의 지역경로계획 기법인 MA* 알고리즘을 활용하여 자율주행을 수행한

Fig. 7. A* 알고리즘 기반 자율주행 결과

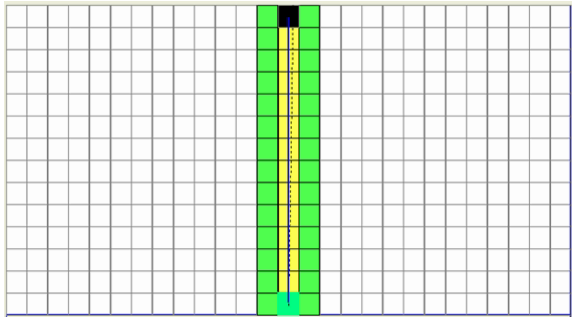

Fig. 8. Fig. 7에서의 A* 알고리즘 LPP 결과

Fig. 9. Modified A* 알고리즘 기반 자율주행 결과

결과를 보여준다. 무인차량은 지형감지센서로부터 획득한 고도정보를 활용하여 지형의 경사도 및 조도를 산출한 후, 이를 기반으로 무인차량의 주행 가능속도를 추정하여 방향별 속도지도(DVGM)를 생성한다. 지형장애물은 점차적으로 증가하는 경사도를 가지므로, DVGM에서 해당격자는 장애물로 인식되지는 않지만

주변지형보다는 낮은 주행속도를 가지며 이는 Fig. 10에 보이는 것처럼 무인차량의 정면에 짙은 노란색 격자로 표현되어 있다. 따라서 DVGM을 활용하여 지역경로계획을 수행하는 MA* 알고리즘은 Fig. 10과 같이 지형장애물을 우회하는 지역경로를 생성하게 되며, Fig. 9에 보이는 것처럼 무인차량이 지형장애물을 회피하여 주행하게 되어 A* 알고리즘 보다 더욱 안전한 자율주행을 수행함을 확인할 수 있다.

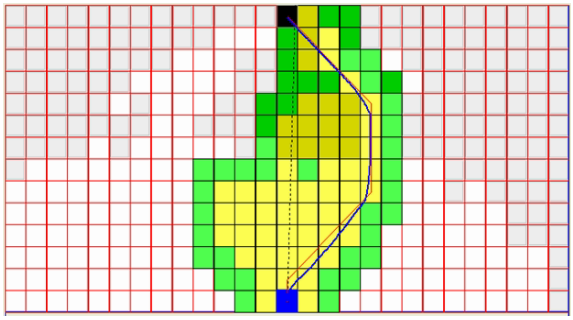

Fig. 10. Fig. 9에서의 MA* 알고리즘 LPP 결과

5. 결 론

본 논문에서는 무인차량의 자율주행시스템을 위한 방향별 속도지도 기반의 지역경로계획 기법을 제안하였다. 특정치 이상의 경사도와 조도가 반복적으로 존재하는 야지에서의 자율주행을 위해 주행환경에 대한 실시간 분석을 통해 주행성 분석지도인 DVGM을 생성하고, 이를 기반으로 지역경로계획을 수행할 수 있는 MA* 알고리즘을 제안하였다. 또한, 격자지도 기반으로 계획된 경로를 무인차량의 운동성을 고려한 주행가능 경로로 재생성하는 경로최적화(Path Optimization) 및 경로곡선화(Path Smoothing) 기법을 제안하였다. MA* 알고리즘 측면에서는, 이동경비관점의 거리우선탐색을 수행하는 A* 알고리즘과 달리 주행속도 우선탐색을 수행할 수 있도록 조향에 대한 경비가 적용된 주행시간 기반의 평가함수를 소개하였다. 경로최적화 알고리즘 관점에서는 추출된 두 포인트 사이의 직선경로 기준의 이동시간과 MA* 알고리즘에 의해 기 계획된 경로기준의 이동시간을 비교하여 경로를 직선화(Pruning)하는 알고리즘을 소개하였으며, 경로곡선화 알고리즘 관점에서는 최적화 알고리즘을 통해 생성된 경로에 무인차량의 운동성을 고려하여 보다 추종성이 향상된 지역경로를 재생성할 수 있도록 하였다.

제안한 방향별 속도지도 기반의 지역경로계획 기법의 성능은 야지에 인공적으로 설치된 경사도 지형장애물을 활용한 자율주행 시험을 통해 검증하였다. DVGM 기반의 MA* 알고리즘과 Binary Map 기반의 A* 알고리즘을 이용하여 해당 시나리오 상에서 자율주행을 수행시킨 결과, 제안한 알고리즘은 지형장애물을 인식하여 회피하는 지역경로를 생성하여 보다 안정적인 자율주행을 수행함을 확인할 수 있었다.

References

[1] Elfes, A., “Using Occupancy Grids for Mobile Robot Perception and Navigation”, IEEE Computer Magazine, 1989.

[2] H. Seraji and B. Bon, “Multi-range Traversability Indices for Terrain-Based Navigation”, Proc. of the 2002 IEEE Int. Conf. on Robotics and Automation, pp. 2674～2681, 2002.

[3] H. Seraji and A. Howard, E. Tunstel, “Safe Navigation on Hazardous Terrain”, Proc. of the 2001 IEEE Int. Conf. on Robotics and Automation, pp. 3084～3091, 2001.

[4] A. Howard, H. Seraji and E. Tunstel, “A Rule-based Fuzzy Traversability Index for Mobile Robot Navigation”, Proc. of the 2001 IEEE Int. Conf. on Robotics and Automation, 2001.

[5] Ye, C., & Borenstein, J., “T-transformation: Traversability Analysis for Navigation on Rugged Terrain”, the Defense and Security Symposium, 2004.

[6] 이영일, 이호주, 지태영, “무인차량의 주행성분석을 위한 방향별 속도지도 생성”, 한국군사과학기술학회지, Vol. 12, No. 5, pp. 549～556, Oct., 2009.

[7] Robin R. Murphy, Introduction to AI Robotics, A Bradford Book, The MIT Press, 2000.

[8] N. Yokoya and K. Yamamoto, “Fractal-Based Analysis and Interpolation of 3D Natural Surfaces and Their Application to Terrain Modeling”, Computer Vision, Graphic, and Image Processing, Vol. 46, pp. 284～302, 1989.

[9] D. Langer, J. K. Rosenblatt and M. Hebert, “A Behavior-Based System for Off-Road Navigation”, IEEE Trans. Robotics and Automation, Vol. 10, No. 6, pp. 776～783, 1994.

[10] H. Seraji, “Traversability Index : A New Concept for Planetary Robers”, Proc. of the 1999 IEEE Int. Conf. on Robotics and Automation, pp. 2006～2013, 1999.

[11] L. Huajun, Y. Jingyu and Z. Chunxia, “A Generic Approach to Rugged Terrain Analysis Based on Fuzzy Inference”, Proc. of the 8th Int. Conf. on Control, Automation, Robotics and Vision, pp. 1108～1113, 2004.

[12] M Castelnovi, R. C. Arkin and T. R. Collins, “Reactive Speed Control System Based on Terrain Roughness Detection”, http://smartech.gatech.edu/handle/1853/20784

[13] 이영일, 정 희, 김용기, “무인수중로봇을 위한 지능형 자율운항시스템”, 정보과학회논문지, Vol. 34, No. 3, March, 2007.