Optimal Path Planning for Car-like Off-road Vehicles

Raphael Linker, Tamir Blass
Faculty of Civil and Environmental Engineering
Technion – Israel Institute of Technology
Israel
linkerr@tx.technion.ac.il

***Abstract*— A simple algorithm for planning the optimal path for a car-like vehicle operating in outdoor environments is presented. The algorithm is based on the well-known A* approach, which is applied in the vehicle's "configuration space" and is adapted to take into account constraints that are specific to the type of vehicle and environment considered: limited steering angle, limited range of pitch and roll angles permissible, forward motion preferable, frequent turning not desirable. The performance of the algorithm is illustrated on several examples based on randomly generated maps as well as on more realistic orchard maps.**

***Keywords*—configuration space, A* algorithm, outdoor environment**

I. Introduction

Path planning has been extensively studied, both for indoor and outdoor applications. Procedures that compute directly the final path taking into account the kinematics and dynamics constraints of the vehicle have been developed (e.g. [1]). However, such methods are very demanding computer-wise. Therefore, the most commonly used approach consists of combining a global planner that ignores the kinematics and dynamics constraints of the vehicle, with a local planner that refines the initial trajectory taking into account the vehicle constraints (e.g. [2]). For global planning, the most popular algorithms are various adaptations and improvements of the A* and D* algorithms, some of them including efficient replanning of the path whenever new obstacles are detected along the way [3-4]. Such algorithms use a square grid representation of the terrain in which the vehicle operates, in which each cell has eight edges (corresponding to the eight adjacent cells) that are either traversable or untraversable. In order to accommodate outdoor applications, a traversal cost that reflects the difficulty to cross the respective area can be defined for each cell [5]. The vehicle is represented by a point that at each time step can move to the center of any of the "reachable" neighbors, and the algorithm determines the cell sequence that minimizes the total cost to the specified goal. As recognized in [5], the main limitation of this approach is that it restricts headings and heading changes to 45^o increments. As a result, the planned path is not only sub-optimal but may also include unnecessary turns that can not always be removed by post-processing. In [5], an interpolation-based approach that allows for virtually any steering and yields much more realistic paths for car-like vehicles, was developed. The present paper presents a different approach for relaxing the 45^o steering constraint, having in mind that the algorithm is intended for car-like vehicles for which the maximum steering angle is well below 45^o. The approach detailed below is based on the so-called configuration space [6] that is commonly used to determine collision-free paths for rigid or articulated objects. The "configuration space" approach reformulates the original path-planning problem that consists of determining a collision-free path for an object of finite dimensions, to that of determining a path for a dimensionless point instead. This is achieved by defining the appropriate configuration space of the object and mapping the obstacles in this space. In his book [6], Latombe presents an extension of this approach to car-like vehicles, assuming that steering is limited either to straight-ahead or maximum left or right steering, which is a rather crude approximation of the reality. The present work presents an extension of Latombe's approach, in which steering can be changed in a step-wise manner over the full steering range of the vehicle. Latombe's approach is furthermore modified so that additional user-defined constraints can be added, ensuring for instance that the roll and pitch angles of the vehicle remains within acceptable limits.

II. Algorithm Description

It is assumed that a map that includes the exact location of all the obstacles is available. It is also assumed that the vehicle characteristics (size and turning radius), initial attitude (location and orientation) and desired (goal) attitude are available. The algorithm, which is based on the A* algorithm [7] is presented in Figure 1, using the terminology commonly used for the A* algorithm. S and G denote the initial and final point, respectively; O denotes the OPEN list of states (States that have been reached during the search but from which the search has not yet been continued. Initially O={S}); C denotes the CLOSED list of states (States that have been reached and from which all the subsequent single steps have been evaluated. Initially C={}). At any time step, the states that have been reached with minimal cost are moved from the OPEN list to the CLOSED list and all the feasible moves that originate from these states are investigated. The OPEN list is updated and the search continues until the goal has been reached or the OPEN list is empty. A main advantage of the A* algorithm is that it is *complete* in the sense that it always finds the optimal path from S to G if such a path exists.

```
At step i:
1. If OPEN list O is not empty
  2. Move all states with lowest t(x) from OPEN list temporary list T
  3. For k=1.. number of states in T
    3.1. Move x(k) to CLOSED list C
    3.2. If x(k)=G, goal has been reached. Terminate search and go to step 4
    3.3. Determine the list of states (L) that can be reached from x(k) and are not in C
    3.4. For n=1..number of states in L
      3.4.1. Denote $\rho$=L(n)
      3.4.2. Calculate the cost of move from x(k) to $\rho$
      3.4.3. Calculate the cost of the path from S to G passing through $\rho$: t($\rho$)
      3.4.4. If the state $\rho$ is already in O and t($\rho$)<previous estimate saved in O
               Update OPEN list: replace previous path to $\rho$ by current one, update t($\rho$) and pointer P
             Else
               If the state $\rho$ is not already in O
                 Add $\rho$ and associated t($\rho$) and pointer P to O
               End
             End
    End (for n)
  End (for k)
Else
  There is no path from S to G. Issue error message and abort execution.
End (if)
4. Optimal path from S to G has been found. Display path by backtracking from G using pointer P
```

Figure 1. Algorithm flowchart

The particularity of the present algorithm is in the way the different costs are evaluated and the restrictions imposed on the feasible moves. Step 3.3 is realized in the configuration space, and determines which states can be reached while avoiding obstacles. The "configuration space" approach is depicted graphically in Figure 2. After defining a reference point on the object (vehicle), an "augmented" obstacle is obtained, so that if the angle of the object does not change (translation movement only), the object does not enter the obstacle if the reference point does not enter the augmented obstacle (greyed area). Since the actual shape of the augmented obstacle varies depending on the object angle, the initial 2D representation is transformed into a 3D representation in which each layer contains the equivalent obstacles for a given object angle.

For a car-like vehicle, purely rotational moves (which correspond to strictly upward and downward moves in the configuration space) are not permitted and the point is forced to move either forward or backward. The number of layers that can be reached in the configuration space (N) is dictated by the radial resolution of the configuration space (Δ) and the maximum steering angle ($\Psi_{\max}$) which is assumed to be a multiple of Δ:

$$N = 2 \cdot \left(\frac{\Psi_{\max}}{\Delta} \right) + 1 \tag{1}$$

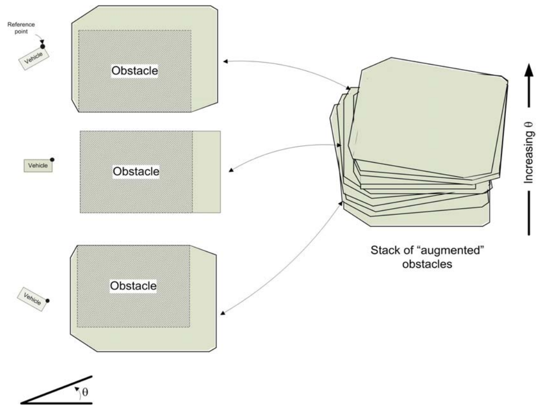

Figure 2. Schematic representation of the configuration space approach.

Note that Δ is also the minimal steering angle ($\Psi_{\min}$), and the number of states that can be reached is 2N. A move that requires moving to a non-adjacent layer in the configuration space (steering angle larger than $\Psi_{\min}$) is considered as feasible only if the cells that correspond to the starting and ending states in all the traversed layers are free of obstacles.

The cost of each feasible move is calculated in Step 3.4.2 according to:

$$\begin{aligned} u\left(x(k),\rho\right) &= f_1\left\{\text{steering angle, travel direction}\right\} + \\ &+ f_2\left\{\text{position}\right\} + f_3\left\{\text{roll and pitch angles}\right\} \end{aligned} \tag{2}$$

where $x(k)$ and ρ denote the current and next states, respectively, and u is the cost of the move. In (2), the first term corresponds to the travel cost associated with the steering angle and travel direction, and can be used to penalize turning and/or backing up; the second term is the traversal cost associated with the cell, which can be used to prevent or favor travel in certain regions; and the third term is the cost associated with roll and pitch angles. If a topography map is not supplied, all the calculations can be performed in the configuration space. Otherwise, it is necessary to estimate the location of the four wheels of the vehicle in order to calculate the roll and pitch angles. Although these calculations are straightforward, it must be noted that this requires distance calculations that slow down the computations significantly.

An estimate of the cost of the full path from the starting state S to the goal state G passing through ρ is calculated in Step 3.4.3:

$$t(\rho) = u\left(S, x(k)\right) + u\left(x(k), \rho\right) + h(\rho, G) \tag{3}$$

In (3), the first term is the cumulative cost of the path until $x(k)$, the second term is the cost of the current step (2) and the third term is a heuristic estimate of the cost from $x(k)$ to G (which has not been planned yet). This third term is included in order to prioritize steps that move the vehicle toward G, and removing it results in a blind (not goal-oriented) search. The L_1 distance (so-called Manhattan distance, which is the sum of the absolute differences of the coordinates of the two points) is commonly used for this task due to its low computation requirements.

III. ILLUSTRATIVE RESULTS

The algorithm presented in the previous Section was tested for a 2.5m by 1.0m vehicle with a maximum steering angle equal to 15^{o}. The resolution of the grid map was 1m by 1m, and the resolution in the configuration space was 5^{o}, so that the vehicle steering was discretized into three levels. All the non-obstacle cells were assumed to be similarly traversable (i.e. f_2=0). The traveling costs of the permitted moves are

TABLE I. TRAVEL COSTS

Travel direction	Steering change	Cost (Arbitrary Units, A. U.)
Forward	0	5
	$\pm 5^{o}$	7
	$\pm 10^{o}$	9
	$\pm 15^{o}$	11
Reverse	0	20
	$\pm 5^{o}$	22

summarized in Table 1. All the other moves were disabled by setting the associated cost to extremely large values.

Roll and pitch angles were penalized using the following function:

$$f_3 = K_a\left[\frac{1}{K_P}\max\left(K_p\alpha,\alpha^2\right)+\frac{1}{K_R}\max\left(K_R\beta,\beta^2\right)\right] \quad (4)$$

where α and β are the roll and pitch angles, respectively. K_R and K_P correspond to the angles at which the respective cost function changes from linear to quadratic. Unless otherwise stated, the simulations were performed with the weighting parameter K_a equal to 10 Arbitrary Units (A.U.), and K_P=K_R= 20 A .U. This can be interpreted as penalizing moderately roll and pitch angles below 20^{o} while penalizing much more heavily higher angles.

The heuristic cost estimate from the current cell ρ to the target G was calculated as

$$h(\rho,G)=K_{dist}\left(\left|\rho_x\text{-}G_x\right|+\left|\rho_y\text{-}G_y\right|\right)+K_{angle}\left|\Gamma_\rho-\Gamma_G\right|. \quad (5)$$

In (5), the first term is the Manhattan distance from ρ to G and the second term is the difference between the current and desired heading. The weighting parameters K_{dist} and K_{angle} were set to 1 A.U.. These parameters determine the relative weight of the heuristic term that “pulls” the vehicle toward the goal. It is important to note that this term does not influence the optimal path itself nor the ability of the algorithm to find the optimal path, but merely influences the number of “dead-end” and “misleading” paths investigated and hence the computation time.

Figure 3 shows the results for a simple maneuver that requires the vehicle to move to a parallel position 5m to the left of its current position. In the presence of an obstacle in front of the final position, the algorithm predicts a classic maneuver that consists of initial backing-up followed by forward&turning move (Fig. 3A). If a steep “bump” (or hole) (45^{o} slope angle) is placed behind the vehicle, the optimal path requires several “backing-up, forward&turning” sequences in order to avoid unsafe pitch and roll angles. (Fig. 3B). Finally, if reverse travel is not allowed (by setting to appropriate costs to very high values), the algorithm plans a much longer and

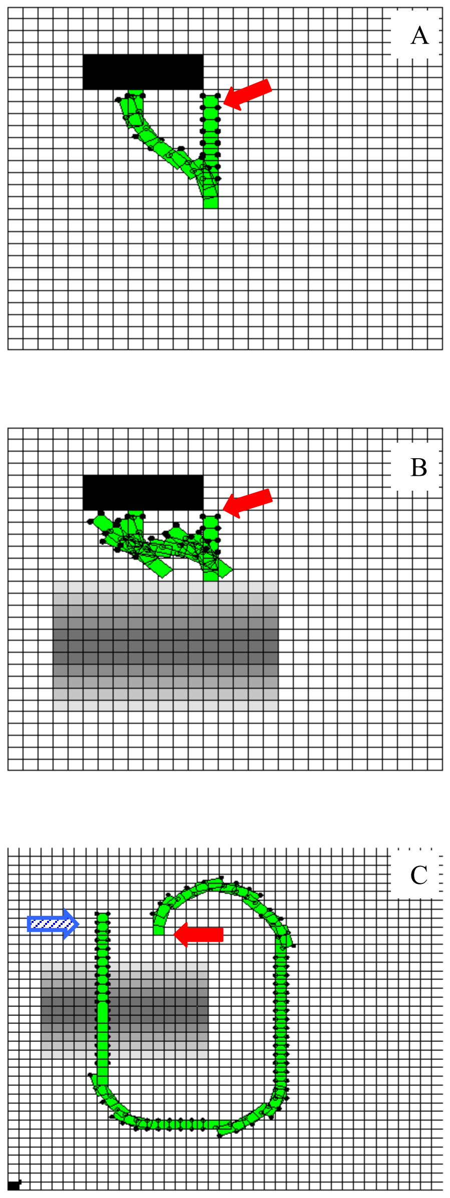

Figure 3. Simulation results for a simple “parallel move”. The front of the vehicle is indicated by two black dots (“headlights”) and the initial and final locations are indicated by a solid and striped arrow, respectively. The shaded area in frames B and C correspond to a bump with 45^{o} slopes. In frame C, travel in reverse direction is not permitted.

wider path (Fig. 3C). In this case the planned path goes over the bump since there is no alternative. However, it must be noted that the vehicle hits the bump “head-on” (pitch angle only), avoiding any roll angle that would further increase the path cost. Clearly, setting K_R and K_P to different values yields different paths, and in this specific example there is no feasible path when K_R=K_P=0.

Figures 4 and 5 show the ability of the algorithm to find the optimal path in environments cluttered with numerous

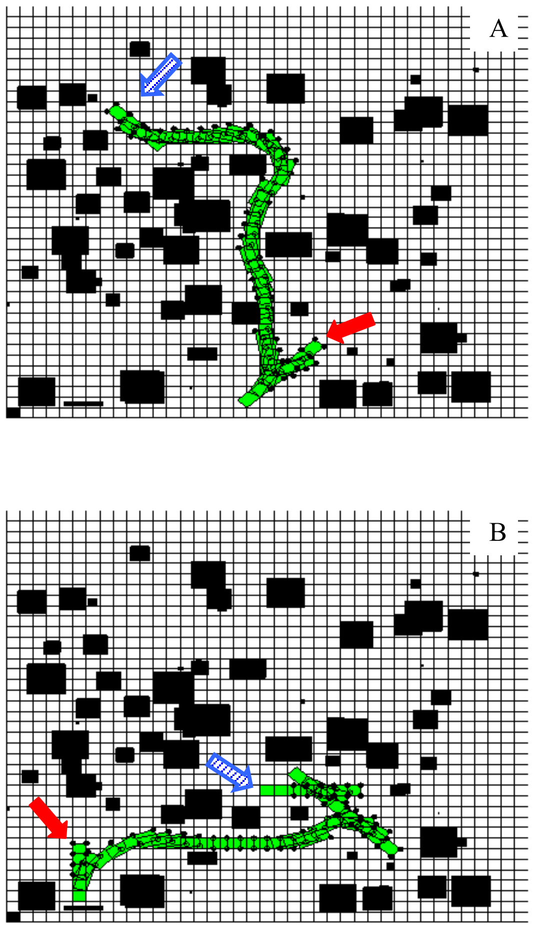

Figure 4. Path planning in clutered flat environments.

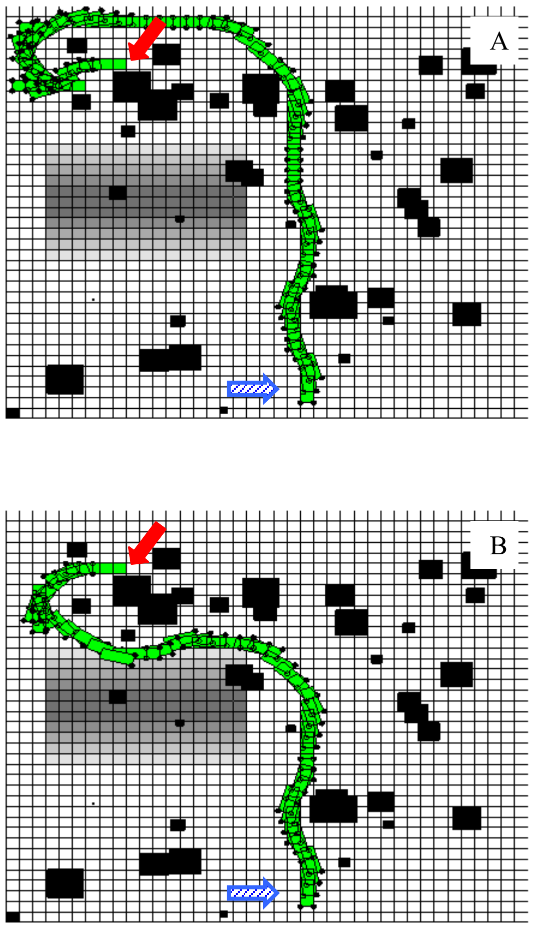

Figure 5. Path planning in clutered environments. The gray area correponds to a bump with 45^o slopes. The pitch and roll costs differ in frames A and B (see text)

obstacles. The simulations shown Fig. 4A and Fig. 4B were performed using a flat map, while in Fig. 5 a bump with 45^o slopes has been added (gray area). Since roll and pitch angles are heavily penalized, especially beyond 20^o, the planned path avoids the bump altogether. If higher roll and pitch angles are allowed, for instance by setting K_P and K_R to 40^o, the optimal path briefly climbs over the bump (Fig. 5B).

One of the applications envisioned for the present algorithm is path-planning for agricultural vehicles (e.g. [2], [8], [9]), either as an assistance system for the human driver or as a high-level component of a fully autonomous vehicle. For such vehicles, one of the most common maneuvers is the "end-of-row turn", in which the vehicle exiting a row has to enters the adjacent one. For such situations, the algorithm plans different types of paths, depending on the maneuver space available: simple turn if the rows are sufficiently wide (Fig. 6A), so-called "light-bulb" turn if the rows are narrow but sufficient space is available beyond the end of the rows (Fig. 6B) and "turning, backing-up, turning" if there is not enough space available for the "light-bulb" turn (Fig. 6C). All three paths are intuitively correct and correspond to standard agricultural practices.

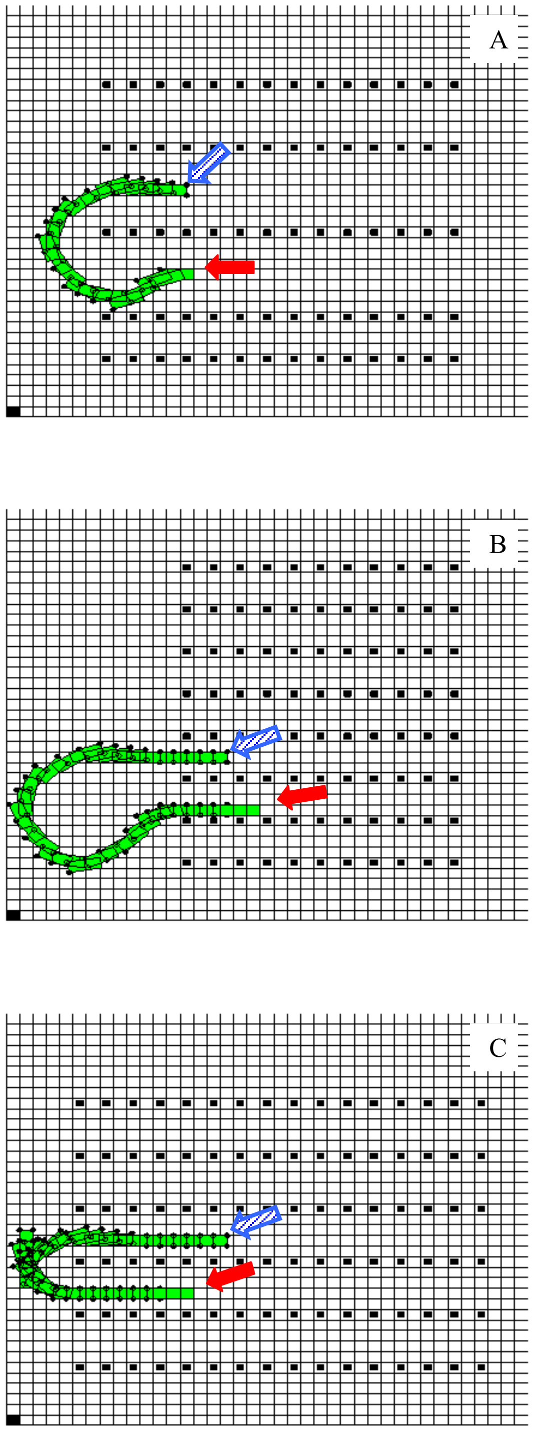

Figure 6. Simulations of "end-of-row turn" maneuvers that are common in agriculture.

IV. CONCLUSION

A simple algorithm for planning optimal paths for car-like vehicles operating in outdoor unstructured environments has been presented. The algorithm is based on the classical A* algorithm, which is used to determine the optimal path using the vehicle configuration space. However, contrary to existing methods that use the configuration space, the permissible moves are restricted in order to reflect the vehicle properties. The path cost (and optimality) is determined using various penalty functions that depend on the steering angle, travel direction, roll and pith angles, and region traversability. Although the planned path depends on a large number of parameters that have to be defined by the user, the role and influence of these parameters is intuitively clear so that it is relatively easy for the user to set the parameters to values such that the planned path meets the user's expectations (i.e., no dangerous roll/pitch angles, no travel in reverse direction, no sharp curve, etc.).

ACKNOWLEDGMENT

T. Blass acknowledge the financial support of the Israeli Ministry of Agriculture (Shalem Fund).

REFERENCES

[1] Z. Shiller, and Y. R. Gwo, "Dynamic motion planning of autonomous vehicles" IEEE Transactions on Robotics and Automation, vol 7, pp. 241-249, 1991.

[2] S. Vougioukas, S. Blakmore, J. Nielsen, and S. Fountas, "A two-stage optimal motion planner for autonomous agricultural vehicles" Precision Agriculture, vol 7, pp. 361-377, 2006.

[3] A. Stentz, "Optimal and efficient path planning for partially-known environments," in Procedings of the IEEE International Conference on Robotics and Automation, pp. 3310–3317, 1994.

[4] S. Koenig, and M. Likhachev, "Fast Replanning for Navigation in Unknown Terrain" IEEE Transactions on Robotics, vol. 21, pp. 354-363, 2005.

[5] D. Ferguson, and A. Stentz, "Using interpolation to improve path planning: The field D* algorithm" Journal of Field Robotics, vol. 23, pp. 79-101, 2006.

[6] J. C. Latombe, Robot motion planning. Kluwer Academic Publ.,Boston, MA, 1991.

[7] N. J. Nilsson, Principles of Artificial Intelligence, Morgan Kaufmann, Los Altos, CA, 1980.

[8] T. Pilarski, M. Happold, H. Pangels, M. Ollis, K. Fitzpatrick, and A. Stentz "The Demeter system for automated harvesting" Autonomous Robots, vol 13, pp. 9-20, 2002.

[9] A. Stentz, C. Dima, C. Wellington, H. Herman, and S. Stager "A system for semi-autonomous tractor operations" Autonomous Robots, vol 13, pp. 87-104, 2002.