JOURNAL OF AEROSPACE COMPUTING, INFORMATION, AND COMMUNICATION
Vol. 3, September 2006

Two-Point Flight Path Planning Using a Fast Graph-Search Algorithm

Wen-Ying Chang,* Fei-Bin Hsiao† and Donglong Sheu‡
National Cheng Kung University, Tainan 70101, Taiwan

This paper presents a method, using a fast graph-search algorithm, of finding a feasible flight path for an air vehicle that flies between two locations. This flight path must satisfy the many constraints required to make the flight safe and efficient. We start by constructing a virtual terrain as a search space above the real terrain, to take into account real flight conditions and the limitations of the vehicle's performance. Consideration of safe altitude, the horizontal safety distance of the flight path and the phase of take-off and landing are included. The idea of a virtual terrain could also eliminate a significant amount of search space, from 3-Dimensions to 2-Dimensions, which takes much less computation time, but which may have a shortcoming in rugged terrain where most path points are higher than the cruise altitude. Hence we propose a further process, which takes less than a second with no extra computational load, to overcome this problem. A dimensionless fuel consumption ratio between climbing and level-turn is proposed to deal with the case of level flying between valleys. If climbing requires greater fuel consumption than taking a level turn, the algorithm chooses the level altitude flight path, hence improving the vertical smoothness of the flight. Using all these methods, including multi-resolution terrain and a fast searching method using a heuristic, we have successfully reduced the computation time of the algorithm to an acceptable level, and the simulation results show that our algorithm is feasible.

Nomenclature

$C(i, j, m)$	=	cost tensor
DTEM	=	digital terrain elevation map
$D(i, j, m)$	=	Distance function
dh	=	altitude difference between two neighboring points
EG	=	number of extended frontiers for hierarchical scheme
$F(i, j)$	=	Forbidden matrix
FCC	=	fuel consumption of climbing
$FCLT$	=	fuel consumption of level turn
g_d	=	grid space(size)
g_{dx}, g_{dy}	=	grid space in x and y coordinate
H	=	cruise altitude
Hc	=	safe altitude for flying over obstacles
$h(i, j)$	=	heuristic function

Received 24 May 2006; revision received 02 August 2006; accepted for publication 26 August 2006.

* Ph.D. Student, Institute of Aeronautics and Astronautics. winstar@rpv.iaa.ncku.edu.tw
† Professor, Institute of Aeronautics and Astronautics, AIAA Fellow. fbhsiao@mail.ncku.edu.tw
‡ Associate Professor, Institute of Aeronautics and Astronautics. donglong@mail.ncku.edu.tw Mailing Address: Institute of Aeronautics and Astronautics, National Cheng Kung University, Tainan 70101, Taiwan

i, j, k	=	integer grid indices
(i_d, j_d)	=	integer grid index of destination point
(i_s, j_s)	=	integer grid index of start point
KD	=	heuristic factor
Kdh	=	dimensionless factor of fuel consumption ratio
$L(i, j, m)$	=	Mapping function
$(L_1(i, j, m), L_2(i, j, m))$	=	one of the eight adjacent points of point(i, j)in direction m
$M(i, j, m) = 1$	=	Admissible matrix
m	=	integer move indices
mt	=	take-off direction
ml	=	landing direction
n	=	number of running step of algorithm
nl	=	stages number of landing slope
nt	=	stages number of take-off slope
$P(i, j)$	=	point in grid space
RC	=	maximum climbing rate of the air vehicle
$R_{\min}$	=	minimum radius of level turn
RC_{pg}	=	climbing altitude per grid distance
r	=	integer move indices of the points in second frontier
$r^*(i, j, m)$	=	optimal move tensor
res	=	map down sampling factor
S	=	number of states of search space
S_h	=	horizontal safety distance for passing through obstacles
$S_m(i, j)$	=	admissible set of first frontier
$S_r(i, j, m)$	=	admissible set function of second frontier
t	=	time needed for flying through a grid space
UC	=	upper ceiling of air vehicle
V	=	flight speed of air vehicle
W	=	vehicle's weight
$x(j), y(i), z(i, j)$	=	Cartesian coordinates of virtual terrain
$z_r(i, j)$	=	altitude of real terrain
$\alpha(m, r)$	=	angle between two path segments with move indices m and r
ρ	=	passing parameter

I. Introduction

DURING the past few decades, in the investigation of path planning with fixed or moving obstacle avoidance, many methods have been developed, such as potential fields,[1,2] graph search algorithms[3–5] and evolutionary algorithms (EA)[6,7] etc. In path finding problems, graph search algorithms, such as Dijkstra's algorithm,[8] which can also be used for finding shortest paths from a single vertex to all other vertices in the graph, or the A* algorithm,[9] which is the same as Dijkstra's algorithm but uses a heuristic estimate to guide itself rapidly towards the goal vertex, and the D* algorithm,[10] a variant of A*, which is designed to find the shortest path in a dynamic environment, are the most popular methods. Evolutionary algorithms use artificial intelligence (AI) methods, such as genetic algorithms, that use mechanisms inspired by biological evolution, and neural networks, to derive heuristic functions to help solve the path finding problems.

Dynamic programming (DP) which relies on the principle of optimality,[11] is one well-known method of finding the shortest flight path. We note that Dijkstra's algorithm is closely related to dynamic programming. In computer science DP is a method for reducing the computation time of algorithms exhibiting the properties of overlapping subproblems and optimal substructure.[12] Applications of dynamic programming have been widely used in economics, finance, operations management, circuit design, robotic motion and path planning, route planning of airplanes and

ships, and the like. In addition, the DP method has been proved to provide a globally optimal solution if a feasible solution exists.[13] To provide a globally optimal solution, a whole field of extremals has to be computed so that the computation time is very high. Therefore, a new scheme of constructing a minimized search space for those extremals is proposed to reduce the calculation time.

Concerning the application areas of the dynamic programming method, enormous efforts have been made in robotic motion and path planning in contrast to flight path planning. Due to a pressing need for onboard real-time path planning for commercial jets in the next decade, much research has been focused on flight path planning in recent years.[14,15] Moreover, the increasing employment of unmanned air vehicles (UAVs) has motivated the researchers looking into this field of flight path planning as well.[4,6,7] Path planning for an air vehicle is an important but laborious process that not only needs to consider the 3-D environment but also has to take the flight altitude and safety measures into account. For example, moving path planning for mobile ground robots on rough terrain[16] does not need to consider aircraft conditions such as take-off, landing, cruise and the essential altitude for flying over the obstacles, etc. Since those considerations may cause critical situations in path planning for the air flight safety, it is necessary to take the 3-D environment of flight space into account in air vehicle path planning. However, more computation time will be incurred, in particular when applying the dynamic programming method. Fortunately, aircraft flight procedures are required to abide by the civil aviation regulations throughout the course of take-off, cruise and landing. Therefore, in employing the DP method for the shortest flight planning, this paper proposes a novel concept of constructing a 2-D search space instead of 3-D one to incorporate the flight states of the vehicle and its flight constraints to accelerate the computation efficiency, that is to make this problem easier in determining the flight path planning on a 2-D virtual terrain. Furthermore, the horizontal safety distance and the safe altitude for air vehicles should be considered in path planning because the aircraft navigation error does exist in real flight. By introducing the virtual terrain, both considerations can be taken into account as well. If necessary, the dangerous zones such as high mountains, terrible weather, or war theaters can be treated as restricted airspace by setting forbidden matrix in part **A** of section **III**. In addition, we use a post-process technique to smooth out the flight-path altitudes by eliminating some possible altitude variations when 2-D search space is applied over rugged terrain. Herewith, the possible and attainable solution space of the shortest flight path can be referred to as a search space which the air vehicle will fly through to the destination between two locations which may be far apart in distance.

Briefly, the method in this paper is based on references[5,8,9,11,16] with the following main changes and improvements: (1) introducing the virtual terrain to involve the phases of take-off, cruise and landing and the horizontal safety distance, and also significantly reduce the search space; (2) applying post-processing to smooth the resultant path in vertical plane by cutting off the wave trough rapidly; (3) including fuel consumption ratio to find a flight path between valleys for the map with a lot of mountains or the flight path with the least frequent change of altitude.

II. Problem Statement

This paper deals with shortest path planning for an air vehicle to fly between two cities (locations) on a real terrain using the dynamic programming method. The digital map of Taiwan, which we used to demonstrate our algorithm, shows that these two cities may be far apart and separated by many high mountains; global knowledge of this environment is required to address safety concerns. There are many sources to acquire digital terrain elevation maps (DTEM), such as the U.S. Geological Survey (USGS).[17] In this paper, the free DTEM of Taiwan was downloaded from USGS, and was transformed to XYZ grid data by the free software "MICRODEM".[18] To simplify the problem in this approach, it is assumed that the air vehicle flies with constant speed in the static environment with no influences from atmospheric circulation. In solving this problem, there are three parts that should be considered while applying the DP method.

First of all, we apply an approximate discretization scheme to our simplified model to allow the application of dynamic programming. That is to say, the algorithm deals with the discrete search space. Also the DTEM of the real terrain should be transformed into the Cartesian grids data, as shown in Fig. 1, for which the DTEM would completely occupy a region involving the start point and the destination point. In this step, it is necessary to choose a proper meshed size of grid for the dynamic programming calculations. It is noted that the size of grid influences very much the computation time, namely the bigger the mesh size of grids is, the less the computation time is. On the contrary, however, the accuracy of the terrain strikingly depends on the size of grids. Hence, how to choose a proper

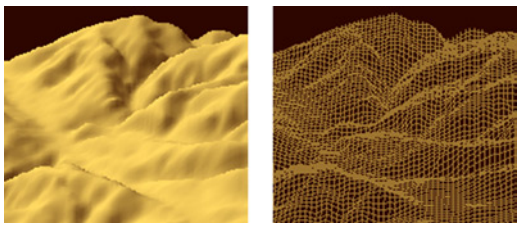

Fig. 1 DTEM transformed into XYZ grids data.

mesh size of grids is critical to the compromise between the computation time and the terrain accuracy needed for the flight path plan.

Secondly, in order to reduce the computation time a virtual terrain should be constructed first above the real DTEM terrain according to some physical conditions and constraints imposed by the aircraft performance such as the climbing rate, the upper ceiling and the cruise altitude etc. One key step here could be taking the states of take-off and landing into account to construct the virtual terrain. The other key point, however, is to take into account the safe altitude and horizontal safety distance between flight path and obstacles due to navigation error. Moreover, dangerous zones such as high mountains, terrible weather, or war theaters can be considered by setting forbidden matrix, if necessary.

Finally, we apply the dynamic programming method to the pre-processed virtual terrain to determine the shortest flight path which conforms to the additional constraints as assumed and described in the next section. Also, a post-process technique is proposed to smooth the altitude of the resulting path.

III. Basic Solution Methodology

We begin with a uniform discretization of the 3-D space of spatial positions. The discretized flight path is made up of straight line segments passing through the grid points. Without doubt, the actual flight path could be smoothed by such as B-spline curves. As we said in the previous section, we replaced the 3-D search space with 2-D space for reduced computation time. The eliminated third dimension, altitude z, has to be limited to the artificial surface $z(i, j)$ which is a virtual terrain we shall discuss later. Now we consider the 2-D discrete grid point (i, j), where the indices i and j are positive integers to determine the position in the respective coordinates.

That is to say, we can determine the spatial position in an abscissa x, an ordinate y and altitude z separately with $x = x(j)$, $y = y(i)$ and the corresponding altitude $z = z(i, j)$, as shown in Fig. 2. The transformation of the DTEM of the real terrain into the Cartesian grids data has the matrix form defined as $z_r = z_r(i, j)$ to represent the elevation z_r of the terrain for any grid point (i, j). The proper size of grids g_d could be determined by the minimum radius of level turn $R_{\min}$ and the flight speed V of an air vehicle as long as $V \times t = g_d \geq R_{\min}$, where t is the time needed for flying through a grid distance, and the relation between g_d and $R_{\min}$ can be shown in Fig. 3. Here the length of grid is assumed to be equal to the width of grid. If it is unequal, the grid size must be $g_d = \min(g_{dx}, g_{dy})$, where g_{dx} and g_{dy} denote the grid's width and length separately. Obviously it is impossible to make a $R_{\min}$ turn in a small grid size, but an easy turn slightly bigger than $R_{\min}$ could be made in a large grid size.

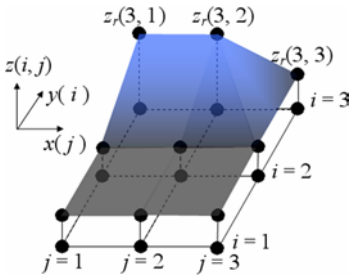

Fig. 2 Depiction of real terrain on *xyz* coordinate axes and indices i and j.

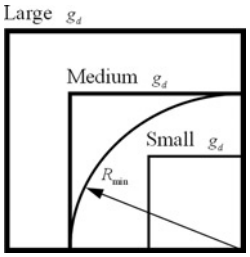

Fig. 3 Relationship of turn radius to grid size.

After the transformed real terrain is obtained, the next step in then to construct the virtual terrain $z = z(i, j)$. Most research in flight path planning have not taken into account the states of flight yet, including take-off, landing and cruise etc. Using such an approach is not feasible to satisfy the situation in practice. This paper then intends to include the take-off, landing, and cruise altitude H as the flight conditions in the path planning and incorporate with the aircraft maneuvering capability, such as minimum turn angle and climbing rate, to avoid any possible collision with the mountains or any other obstacles during the flight operations proposed in the present study.

Besides the advantage of taking less computation time, there is another great benefit of virtual terrain. This is particularly so in the case of civil aviation with its stringent safety precautions. That is, the fly-over altitude defined as Hc could be considered to make a complete safe and secure altitude for flying over obstacles. Also the consideration of horizontal safety distances S_h between air vehicle and obstacles could be involved into construction of virtual terrain as well. Both considerations in vertical and horizontal safety can be implemented by following scheme.

For every point (i, j) do
  If $z_r(i, j) > H$
    For $m = \{1, \ldots, 8\}$ do

$$dh = |z_r(L_1, L_2) - z_r(i, j)|\,; ztmp(m) = \begin{cases} z_r(i, j) + \dfrac{S_h \times dh}{g_d}, \text{if } z_r(L_1, L_2) > z_r(i, j) \\ z_r(i, j) + Hc, \text{else.} \end{cases}$$

    End-For
    $z(i, j) = \max(ztmp(m = 1 \sim 8), z_r(i, j) + Hc)$
  Else, $z(i, j) = \max(H, z_r(i, j) + Hc)$
End-For

where move m is the index of eight adjacent directions and $(L_1, L_2) = (L_1(i, j, m), L_2(i, j, m))$ is its corresponding adjacent point, which are both referred to Section **III. B**: Move and Mapping. The equation of $ztmp(m) = z_r(i, j) + (S_h \times dh)/g_d$ is derived from the relation of similar triangle as $(z(i, j) - z_r(i, j)/dh = S_h/g_d$, as shown in Fig. 4.

Due to the aircraft navigation error, it is necessary to take into account not only the maximum climbing rate but also the safe altitude for flying over obstacles and the horizontal safety distance for passing by obstacles. For

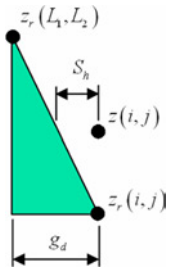

Fig. 4 Consideration of horizontal safety distance.

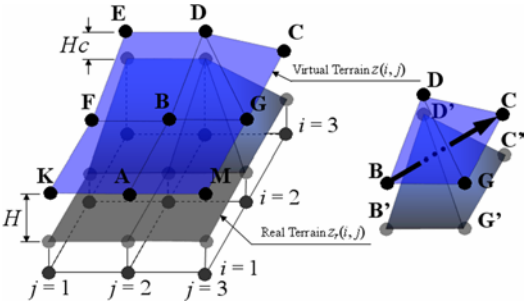

Fig. 5 The path BC is not admissible for intersection happened, namely path BC under the real terrain line D'G'.

example in Fig. 5 if there is the possibility of intersection between path and terrain, the path **BC** will turn out to be inadmissible.

Moreover, the take-off and landing phases should be included in constructing the virtual terrain as inverted pyramids whose slopes separately stand for the slopes of the take-off and landing. The scheme of constructing the inverted pyramid on the virtual terrain was shown below:

Let $z(i_p, j_p) = z_r(i_p, j_p)$;
For every point (i, j) in the n-th frontier of point (i_p, j_p) at step n, where $n = 1 \sim np$
If $z(i, j) = H$ and $n \times RC_{pg} < H$
If $z_r(i_p, j_p) + n \times RC_{pg} \geq z_r(i, j) + Hc$, then $z(i, j) = z_r(i_p, j_p) + n \times RC_{pg}$
Else, $z(i, j) = z_r(i, j) + Hc$

where point (i_p, j_p) will be the starting or the destination point, and np is the stage number of the corresponding slope. Examples of constructing the inverted pyramids can be seen in Fig. 6. The take-off and the landing slopes depicted by setting the stages number $np = nt$ and $np = nl$ are dependent on the climbing altitude per grid distance $RC_{pg} = RC \times t$, where RC is the maximum climbing rate of the vehicle. The stages number nt and nl are shown in Eqs. (1) and (2):

$$nt = ceil\left(\frac{H - z_r\,(i_s, j_s)}{RC_{pg}}\right) \tag{1}$$

$$nl = ceil\left(\frac{H - z_r\,(i_d, j_d)}{RC_{pg}}\right) \tag{2}$$

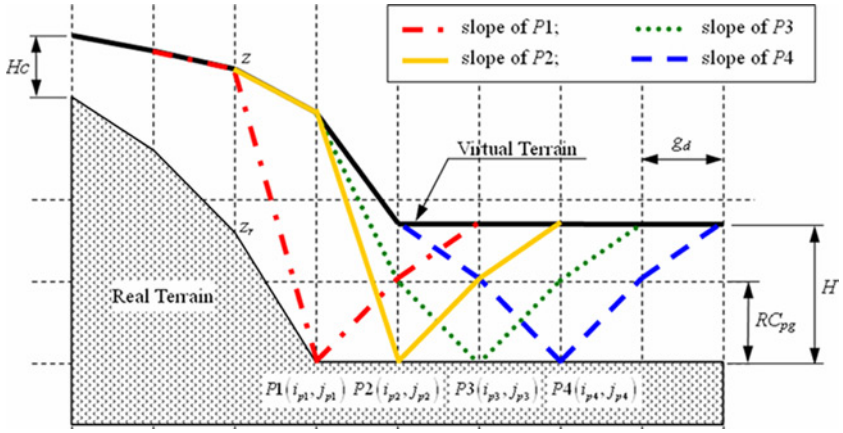

Fig. 6 Profile of the take-off or landing slopes.

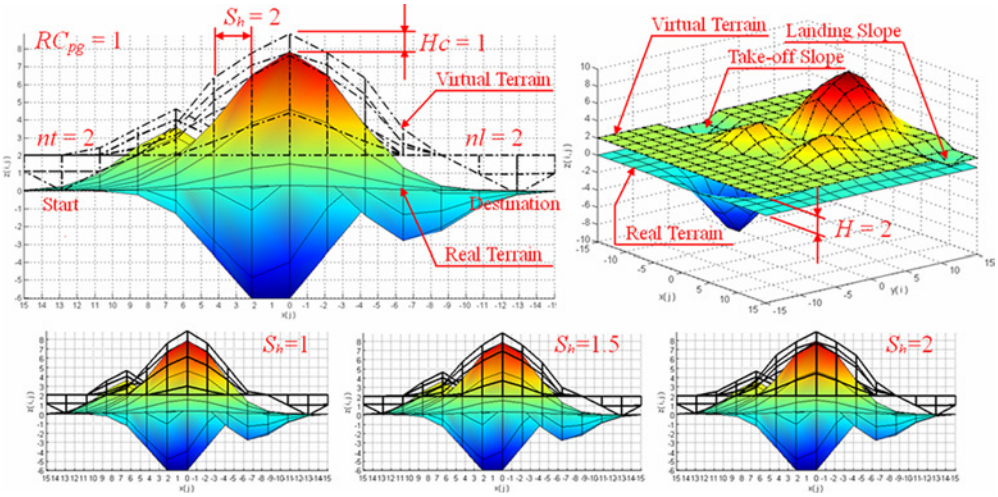

Fig. 7 Top left: a profile of virtual terrain (dash-dot line); Top right: 3-D view of virtual terrain; Bottom: the effect of varying S_h.

where the function $ceil(q)$ rounds the value of q to an nearest integer greater than or equal to q. The indices (i_s, j_s) and (i_d, j_d) respectively denote the starting point and the destination point. Also, the maximum climbing rate RC stands for the maneuverability of the air vehicle and is given by

$$RC = \frac{\text{excess power}}{W} \tag{3}$$

where the excess power is the difference between the power available and the power required, and W is vehicle's weight. For a detailed discussion, please refer to reference.[19]

To summarize the constructing measure of virtual terrain, the first step is to set up the following parameters: H, Hc, S_h, RC, (i_s, j_s) and (i_d, j_d), and then the stages number nt and nl can be calculated. Finally according to these parameters and the real terrain $z_r = z_r(i, j)$, the virtual terrain $z = z(i, j)$ shown in Fig. 7 could be established.

After the virtual terrain has been established, the following step is to apply the dynamic programming method to determine the minimal flight path which shall conform to the additional constraints imported. To conduct the calculation in the beginning, one should define the cost tensor, related functions and additional constraint functions.

A. Forbidden Matrix

If the grid point (i, j) is located at a dangerous zone, such as high mountains, terrible weather, and war theater, or outside the grid map, then the point (i, j) is forbidden. Thus the forbidden matrix can be defined as:

$$F(i, j) = 1, \quad \text{if } (i, j) \text{ is not forbidden} \tag{4}$$

$$F(i, j) = 0, \quad \text{if } (i, j) \text{ is forbidden} \tag{5}$$

When a point has been set as forbidden, it will be eliminated from the search space.

B. Move and Mapping

From a point (i, j) moving to its adjacent point is referred to as a mapping, and we define mapping function as $L(i, j, m)$ which will return an array with two components (L_1, L_2) to indicate the corresponding adjacent points, where the move index m denotes the eight adjacent directions. Namely, there are eight ways of move in total and can be depicted in Fig. 8. In the following section, we use $(L_1(i, j, m), L_2(i, j, m))$ to denote the mapping result of a point (i, j) by move m.

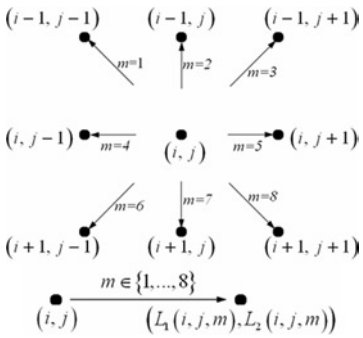

Fig. 8 Move and mapping.

C. Distance Function

The distance function calculating the distance between two adjacent points is defined as follows:

$$D(i,j,m)=\begin{cases}-1, & \text{if } F(i,j)=0 \text{ or } F(L_1(i,j,m),L_2(i,j,m))=0;\\ \begin{pmatrix}\text{distance between } (i,j)\\ \text{and } (L_1(i,j,m),L_2(i,j,m))\end{pmatrix}, & \text{otherwise}\end{cases} \tag{6}$$

If both point (i, j) and its adjacent point $(L_1(i, j, m), L_2(i, j, m))$ are not forbidden, distance function $D(i, j, m)$ will return their Euclidean distance.

D. Admissible Set

During each cost updating process there are two steps for checking in which the adjacent points are allowed to pass by for the point being processed. The first step is to check that the altitude difference between them does not exceed the maximum climbing altitude per grid distance. The second step is to check that the turning angles of the neighboring points from allowed adjacent points do not exceed the constraint of minimum radius of level turn. Due to the constraint of the minimum radius of level turn, the concept of frontier[20] is introduced into our algorithm to implement the turning angle restriction.

A function defined to handle the constraints of the maximum climbing altitude per grid distance and also the upper ceiling or maximum altitude allowed to fly is shown below:

$$M(i,j,m)=\begin{Bmatrix}0, & \text{if } dh > RC_{pg} \text{ or } z(L_1(i,j,m),L_2(i,j,m)) > UC\\ 1, & \text{otherwise}\end{Bmatrix} \tag{7}$$

where $dh = |z(L_1(i, j, m), L_2(i, j, m)) - z(i, j)|$ denotes the altitude difference between point (i, j) and its adjacent point $(L_1(i, j, m), L_2(i, j, m))$, and UC denotes the upper ceiling of air vehicle. If $M(i, j, m) = 1$, it means that the move m from point (i, j) is admissible.

Now we are going to define the frontier that is used to implement the turning angle constraint. For any point $P(i, j)$, its adjacent points can be defined as frontiers. The first frontier is the nearest point set which includes those eight points around point $P(i, j)$ and the second frontier is the farther point set which includes those sixteen points around the first frontier, and so on. As an example in Fig. 9, the first frontier of point **M** is the set of points in the dark gray area, and the second frontier is the set of points in the light gray area. Then we consider that the air vehicle flies from point **M** to point **F** in Fig. 9. The point **G** in the first frontier is one of the points on the way to the destination. In consideration of the constraint of minimum radius of level turn, we need to check whether or not the acute angle $\alpha(m, r)$ between these two path segments **MG** and **GF** can satisfy Eq. (8). If it is true, that is to say, these two path segments are admissible, and then the corresponding admissible move $m \in \{1, \ldots, 8\}$ of first frontier and move $r \in \{1, \ldots, 8\}$ of second frontier are put into the admissible set functions S_m and S_r separately. These two functions

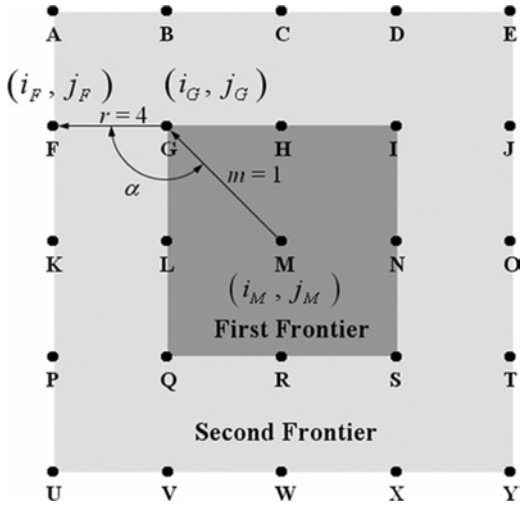

Fig. 9 The concept of expanding frontiers.

can be well defined later after the following equations of α been deduced.

$$\alpha \geq 2 \tan^{-1} \frac{R_{\min}}{\rho} \tag{8}$$

where $R_{\min}$ denotes the minimum radius of level turn and ρ denotes the parameter of passing through one point, i.e. point **G** in this example. By setting the passing parameter ρ shown in Fig. 10, one can claim that point **B** is passed if a sphere with radius ρ centered at point **B** has been passed by our air vehicle. Next, let the length of $\overline{PB}$ and $\overline{BQ}$ are both equal to radius ρ, and the length of $\overline{OP}$ and $\overline{OQ}$ are both equal to $R_{\min}$. Then arc $\widehat{\mathbf{PQ}}$ centered at point **O** connects $\overline{AP}$ and $\overline{QC}$ to make up the flight path from point **A** to point **C**. In this way, the acute angle α between $\overline{AB}$ and $\overline{BC}$ can be obtained by using the following equation:

$$\tan \frac{\alpha}{2} = \frac{R_{\min}}{\rho} \tag{9}$$

Therefore, the constraint of minimum radius of level turn $R_{\min}$ can be converted into the constraint of the acute angle α of two path segments as shown in Eq. (8). If $g_{dx} = g_{dy}$, the acute angle α which would be constrained to the multiples of $\pi/4$ can be defined as a look-up table for reducing the calculation time.

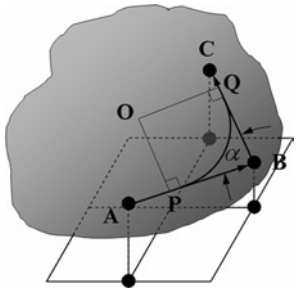

Fig. 10 The constraint of minimum radius of level turn.

After the constraints of radius of level turn and climbing rate have been discussed, we can define the admissible sets S_m of the first frontier and S_r of the second frontier as following:

$$S_m(i, j) = \{m: m \in \{1, \ldots, 8\}, D(i, j, m) \neq -1, M(i, j, m) = 1\} \tag{10}$$

$$S_r(i, j, m) = \left\{ \begin{array}{c} r: r \in \{1, \ldots, 8\}, \alpha(m, r) \geq 2 \tan^{-1} \dfrac{R_{\min}}{\rho}, \\ C(L_1(i, j, m), L_2(i, j, m), r) \neq -1 \end{array} \right\} \tag{11}$$

We use admissible set function $S_m(i, j)$ to find out the set of admissible move m in first frontier. Where in Eq. (10), $D(i, j, m) \neq -1$ implies that the point $(L_1(i, j, m), L_2(i, j, m))$, which is from point (i, j) by move m, is not in forbidden area. Also by using Eq. (11), we can find out the admissible move r that satisfies the turn angle constraint. Here $C(L_1(i, j, m), L_2(i, j, m), r) \neq -1$ means that the path from point $(L_1(i, j, m), L_2(i, j, m))$ to its adjacent point by move r has been reached and has cost $C(L_1(i, j, m), L_2(i, j, m), r)$.

For each point (i, j) there are up to eight costs $C(i, j, m = 1 \sim 8)$ in corresponding admissible move m and the minimum one of them is recorded in the extended cost tensor which is defined in the following subsection.

E. Extended Cost Tensor and Optimal Move Tensor

On the way to the destination point (i_d, j_d), a minimum cost for point (i, j) and its corresponding move m and r in its first frontier and second frontier would be recorded separately in extended cost tensor $C(i, j, m)$ and optimal move tensor $r^*(i, j, m)$ at each updating process. Initially, the extended cost tensor is set as following:

$$\text{For every point } (i, j), \text{ let } C(i, j, m = 1 \sim 8) = \left\{ \begin{array}{ll} 0, & \text{if } (i, j) = (i_d, j_d) \\ -1, & \text{otherwise} \end{array} \right\} \tag{12}$$

where $C(i, j, m) = -1$ indicates that the point (i, j) by move m can not reach the destination point (i_d, j_d) or has not been updated yet. Before running the dynamic programming method, the start and the destination points should be given as inputs and then the algorithm works backwards from the destination point to calculate the minimum cost for grid points in expanding the frontier on virtual terrain until all points have been processed and a minimum cost of the start point $C(i_s, j_s, m)$ is determined.

F. Dynamic Programming Method

After the relative functions are well defined, we can write down our basic algorithm as following:

Set $C(i, j, m) = -1$ for all points except $C(i_d, j_d, m) = 0$
For every point (i, j) in the n-th frontier of point (i_d, j_d) at step n
  For $\forall m \in S_m(i, j)$, find $S_r(i, j, m)$
    If $S_r(i, j, m)$ is not empty
      If $\min_{r \in S_r(i, j, m)} (D(i, j, m) + C(L_1(i, j, m), L_2(i, j, m), r)) \leq C(i, j, m)$
        $C(i, j, m) = \min_{r \in S_r(i, j, m)} (D(i, j, m) + C(L_1(i, j, m), L_2(i, j, m), r))$
        $r^*(i, j, m) = \arg \min_{r \in S_r(i, j, m)} (D(i, j, m) + C(L_1(i, j, m), L_2(i, j, m), r))$

We proceed in this manner for $n = 1, 2, 3, \ldots$ until the cost of start point has been updated, i.e. $C(i_s, j_s, m) \neq -1$, and then the optimal path with minimum distance cost can be found by the following procedure. Otherwise, no feasible

solution exists if $C(i_s, j_s, m) = -1$ for all move m.

$$(i^*, j^*) = (i_s, j_s)$$
$$m^* = \arg \min_{m \in \{1,\ldots,8\}} C(i_s, j_s, m)$$
While $(i^*, j^*) \neq (i_d, j_d)$
$$(i^*, j^*) \leftarrow (L_1(i^*, j^*, m^*), L_2(i^*, j^*, m^*))$$
$$m^* \leftarrow r^*(i^*, j^*, m^*)$$
End

where the grids indices and the corresponding move of the shortest path from the start point to the destination point are recorded in array (i^*, j^*) and m^*, respectively.

G. Direction Limitations of Take-off & Landing

The direction limitation of landing ml can be implemented by setting the cost of destination point $C(i_d, j_d, m) = 0$ only if $m = ml$, otherwise $C(i_d, j_d, m) = -1$. On the other hand, the direction limitation of take-off mt can be carried out by initially setting the move of shortest path $m^* = mt$, i.e. the first move of start point is limited to mt.

IV. Algorithm Improvement

In this section we introduce some concepts to improve both safety and the computation time of our algorithm. The first two concepts are used to reduce the computation time of the algorithm and the next three topics are focused on improving the air vehicle's flight path qualities.

A. Multi-Resolution (Hierarchical) Scheme

It is well known that the amount of computation time mainly depends on the size of the search space. An effort that has reduced a significant amount of search space by restricting it to a virtual terrain surface has been implemented in the previous section. Now we further apply the multi-resolution or so called hierarchical scheme to reduce the search space. The hierarchical scheme has been widely used in numerical analysis and optimization, and we describe here a simple coarse-fine scheme that is convenient for our application. The coarser level map is simply obtained by down sampling the original fine map where the down sampling factor is defined as *res*. For secure concern, the altitude of each sampled point was taken as the maximum altitude over the original fine map points it covers.

The scheme begins with finding a shortest path at coarser level, and then uses the resulting coarse path which is stored with point indices (i^*, j^*) to define a sequence of search space over the original fine level. The search space is first extended from the last point of (i^*, j^*), i.e. the destination point (i_d, j_d), with some number of frontiers. The number of extended frontiers is defined as EG, then use the same algorithm to update the minimum cost for those points in the search space. The next search space is extended from the next point of (i^*, j^*) in inverse order. Until the start point (i_s, j_s) has been processed, we can obtain the final path in the original fine level by using the same procedure of finding the shortest path in the previous section. For example in Fig. 11, the sampling factor is $res = 5$ and the number of extended frontiers is $EG = 5$. The yellow thin line is the coarse level path and the purple thick line is the fine level path.

B. The Introduction of Heuristic

As with the A* algorithm, we can apply the heuristic function $h(i, j)$ to reduce the search space. Concerning the A* algorithm, the heuristic function which is defined on the points of search space serves as an estimate of the cost of the shortest path from that point to the destination point. In our application, the heuristic function estimates the cost of the shortest path from that point to the start point. We will first search those points (i, j) which have a value of $h(i, j) + C(i, j, m)$ lower than KD times of the Euclidean distance between the start point and the destination point. The heuristic DP therefore restricts the search to grid points within an ellipse around the path as shown in Fig. 12. The simplest estimate of heuristic is the Euclidean distance from the current point to start point. If no solution exists, the algorithm will start again with larger value of KD. If all the search space has been updated and there is still no solution, then we can say that no feasible solution exists.

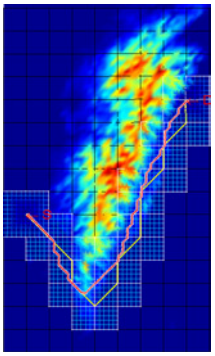

Fig. 11 Illustration of the multi-resolution scheme.

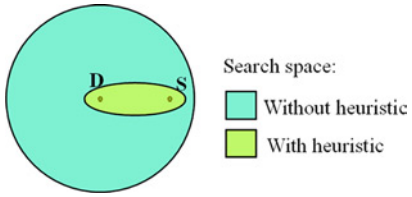

Fig. 12 Comparison of search space between algorithms with heuristic and without heuristic.

C. Post-Processing of Resulted Path

In the case of rugged terrain where there are a lot of points of search space having an altitude higher than the cruise altitude, the resultant path may have a frequently changing altitude. To deal with this problem we propose a post-processing method to cut off the wave trough of the resultant path.

D. Path Smoothing (Curve Fitting)

In practice, the resultant flight path should not be made up of straight line segments. There are many kinds of curve fitting methods, such as cubic spline curve, Bezier curve and B-spline curve, etc. which can be applied to smooth out the resultant path. The computation time of curve fitting is based on the order of the curve fitting function. The lower orders take less time, but the smoothness will go down. The acceptable lowest order in practice is the second order, which satisfies position and velocity continuity. In the other hand, using higher order of curve fitting function will increase the risk of hitting terrain because the smoothing deviation is proportional to the order (see Fig. 13.) To guarantee a secure flight path, we can choose an appropriate order which has the maximum smoothing deviation smaller than the grid size g_d, and choose a conservative value of the horizontal safety distance $S'_h = S_h + g_d$ as well. In this paper a third-order B-spline curve fitting function is used to smooth the resultant path.

E. Fuel Consumption Ratio

Besides considering the cost of the shortest distance flight path, the cost of fuel consumption can also be taken into account. In avoiding an obstacle, the fuel consumption by climbing should be more than making a level turn. A dimensionless factor of fuel consumption ratio taken between these two ways can be defined as $Kdh = FCC/FCLT$, where FCC is the fuel consumption of climbing and $FCLT$ is the fuel consumption of a level turn. Therefore, the cost function can be redefined as $C(i, j, m) = \min_{r \in S_r(i,j,m)}(Kdh \times dh + D(i, j, m) + C(L_1(i, j, m), L_2(i, j, m), r))$, where $dh = |z(i, j) - z(L_1(i, j, m), L_2(i, j, m))|$. In addition, by enlarging this factor the resultant path can prefer to find the way between valleys for the map with a lot of mountains. Also the frequent change of altitude can be improved because the extra changes of altitude will cause a large increase of cost.

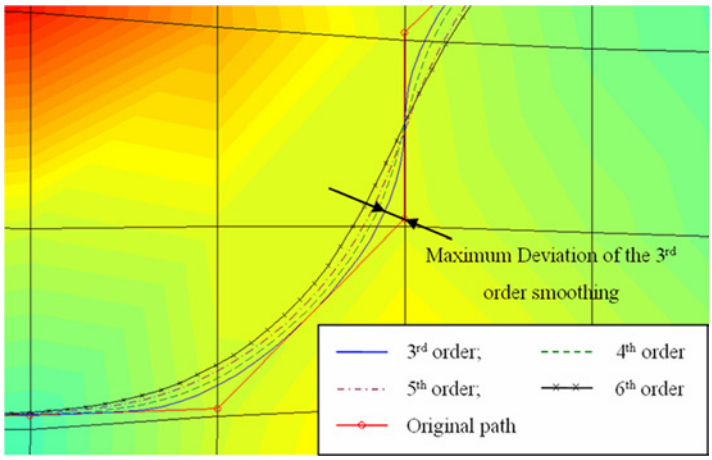

Fig. 13 Varying orders of smoothing function.

V. Experimental Results

At the beginning of our simulation experiments, we demonstrate the virtual terrain concept used in our algorithm by conducting a simulation followed by comparing the performance and effect of the proposed improvement. The issues we consider include: (1) performance of basic 2-D DP with post-processing compared to 3-D DP; (2) basic 2-D DP varying turn angle limits with restricted take-off and landing directions; (3) Multi-resolution scheme and heuristic improvement; (4) effect of fuel consumption and path smoothing.

The DTEM which was used to conduct our simulation is a $122 \times 90\,\text{km}^2$ map of northern Taiwan and will be sampled to a 101×151 matrix $z_r(i, j)$. Its corresponding longitude and latitude of bottom left is (N24°00′, E120°41') and of top right is (N24°48′, E121°56'). Hence the grid size can be obtained by $g_d = \min(g_{dx}, g_{dy}) = 809\,\text{m}$ where the grid width $g_{dx} \doteq 809\,\text{m}$ and the grid length $g_{dy} \doteq 886\,\text{m}$. By making an assumption of constant speed flight that $V = 80\,\text{km/hr} \doteq 22.2\,\text{m/s}$ the flying time for going through a grid can easily be obtained by $t = g_d/V \doteq 37\,\text{sec}$. The minimum radius of level turn is set to $R_{\min} = 300\,\text{m}$ and then the condition of grid size $g_d > R_{\min}$ has been satisfied. Furthermore, assume that the maximum climbing rate of the air vehicle that $RC = 6.5\,\text{m/s}$, and that the maximum climbing altitude per grid distance can be determined as well, that is, $RC_{pg} = RC \times t = 240.5\,\text{m}$. By setting the passing parameter $\rho = 300\,\text{m}$ one can transfer the constraint of minimum radius of level turn to the angle $\alpha = 2\tan^{-1}(R_{\min}/\rho) = \pi/2$. as in Eq. (9). That is to say, during all procedures the angle between any two path segments should not be smaller than the angle value of $\pi/2$.

To build up the virtual terrain, three parameters still need to be defined, as do the start point and the destination point. The three parameters are cruise altitude H, safe altitude Hc and horizontal safety distance Sh. Here we define $H = 1000\,\text{m}$, $Hc = 200\,\text{m}$ and $S_h = 200\,\text{m}$ to make up a secure virtual terrain as 2-D search space, and set the start point as $(i_s, j_s) = (97, 9)$ and the destination point as $(i_d, j_d) = (9, 145)$. Fig. 14 is the constructed virtual terrain. This terrain pre-process time in calculation takes only about 4 seconds in this case. Of course, the computation time depends on the number of points and number of possible moves for each point. The size of search space of 3-D DP is the total number of states S in the spatial grid, namely $S = 101 \times 151 \times k$, where k is the total number of points in the vertical axis. As we discussed previously, the less search space it has, the less computation time it takes. We therefore utilize the concept of virtual terrain to restrict the search space to a 2-D virtual terrain surface, and its size is just $S = 101 \times 151$. By applying the virtual terrain, the search space has been significantly reduced by a factor of k. The algorithm is going to explore all the points for updating the cost on this space frontier by frontier. The hardware for calculation employed here is a laptop personal computer (PC) with Intel® Pentium® M 1.73 GHz CPU and 1 GB RAM. Both the algorithms and the graphics engines utilize the MathWorks MATLAB®.

The first experiment compares the performance between the basic 2-D DP with post-processing and the basic 3-D DP. An upper ceiling of $UC = 2500\,\text{m}$ has been chosen for this simulation. The results are shown in Fig. 15. The total computation time (runtime) of the latter case is almost 32 times of that of the former and without a significant

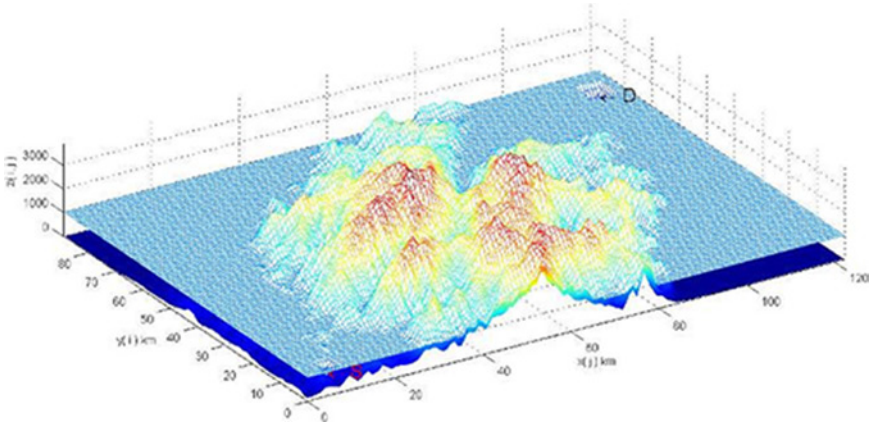

Fig. 14 Virtual terrain.

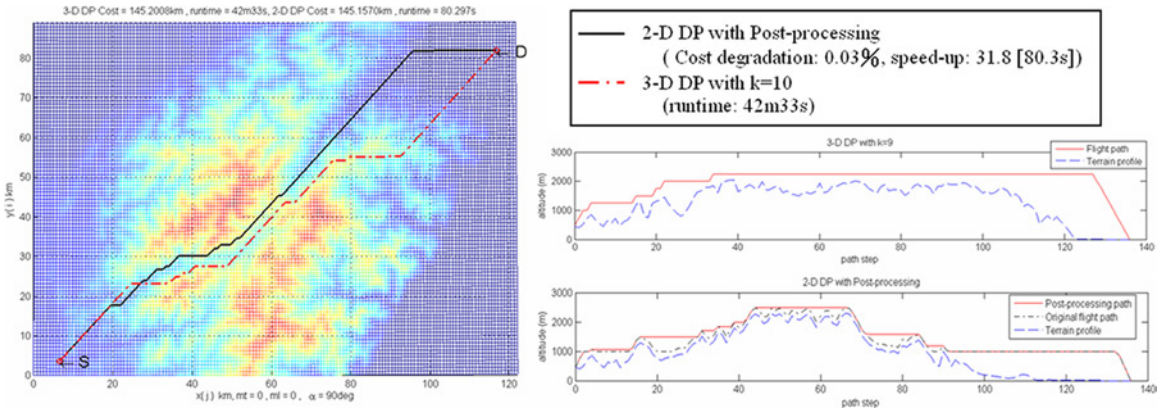

Fig. 15 2-D DP with Post-processing vs. 3-D DP.

degradation of cost. Note that the latter has 24 moves for each point, which in contrast with only 8 moves of the former. In addition, the total number of points in the vertical axis of the latter is $k = 10$, that is, 10 layers in z axis; by contrast it is $k = 1$ in the former. The whole altitude profiles of the resulted path are shown in Fig. 15. Note that the post-processing efficiently takes less than a second to cut off the wave troughs such that the shortcoming of frequent up and down behavior can be overcome.

We next examine briefly the effect of varying the minimum turn angles and also the limited directions of take-off and landing. Shown in Fig. 16, the red solid lines represent the resulted paths while the black lines beginning from

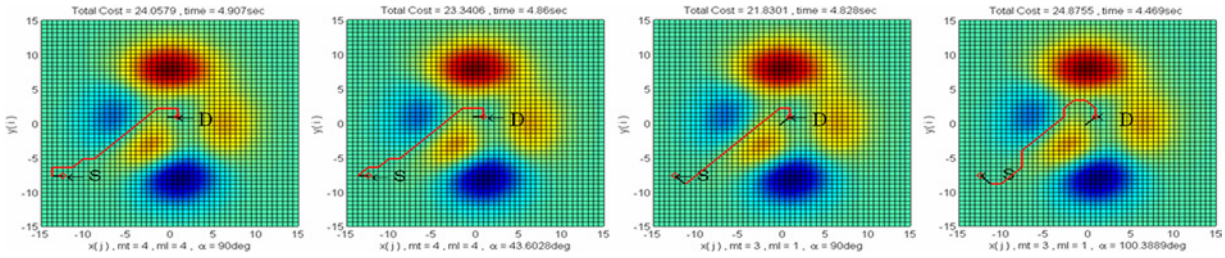

Fig. 16 The effect of varying the minimum turn angles and also the limited take-off and landing directions.

the start and destination points are used to indicate the take-off mt and landing ml direction, respectively. As the angle α is defined to denote the maximum turning angle, hence, all the angles between any two segments in the resulted paths should not be larger than the angle α. This value can be adjusted according to the passing parameter or minimum radius of level turn of the air vehicle.

Now, we consider the improvement of basic 2-D DP. In this paper we proposed the concept of virtual terrain to reduce the search space. With the post-processing we proved previously that it is practical. Moreover, now we apply the so called hierarchical scheme or multi-resolution scheme to speed up our basic algorithm. The result is shown in Fig. 17. The algorithm begins finding the path on a coarser map which is down sampled from the original finer map with a sampling factor of $res = 2$. In this way the matrix size of the coarser map is 51×76. When the result coarser path has been obtained, the algorithm begins to find the shortest path on the finer map by search the extended frontiers of the coarser path with $EG = 2$. The hierarchical algorithm takes 26.6 seconds to find the shortest path on coarser map and 9.75 seconds to find the final shortest path on the original map. That is to say, the hierarchical 2-D DP takes only 36.3 seconds to find a solution for a shortest path with only 0.33% cost degradation (compared to the basic 2-D DP reference which takes around 80 seconds.)

Furthermore, there is another improvement that is using the heuristic estimate. By setting the heuristic factor KD we can speed up the algorithm in variant levels as shown in Fig. 18. However, the corresponding cost degradation might be unacceptable especially when the fuel consumption ratio is considered.

The consideration of fuel consumption ratio can restrain the flying path from the frequent changing altitudes because the extra changes of altitude will cause a large increase of cost as discussed previously. The idea shows clearly in Fig. 19. That is, with a larger Kdh such as 20, the flying path will find out a path with minimum altitude changing. Note that in this case the start point was set as $(i_s, j_s) = (97, 9)$ and the destination point was set as $(i_d, j_d) = (97, 129)$. In Fig. 20 we emphasize the effect of Kdh on a DTEM map in southern Taiwan where is separated by mountains. By setting $Kdh = 10$, we can have a path flying through valleys with minimum altitude changing to reach the destination. As for a very large $Kdh = 50$, a result path flying at the cruise altitude without altitude changing can be obtained if needed.

Finally, the line-segmented path can be smoothed by post-processing of B-spline curve fitting with parameters of order 3. The result is depicted in Fig. 21, where the smoothed path should be more practical for a real flight.

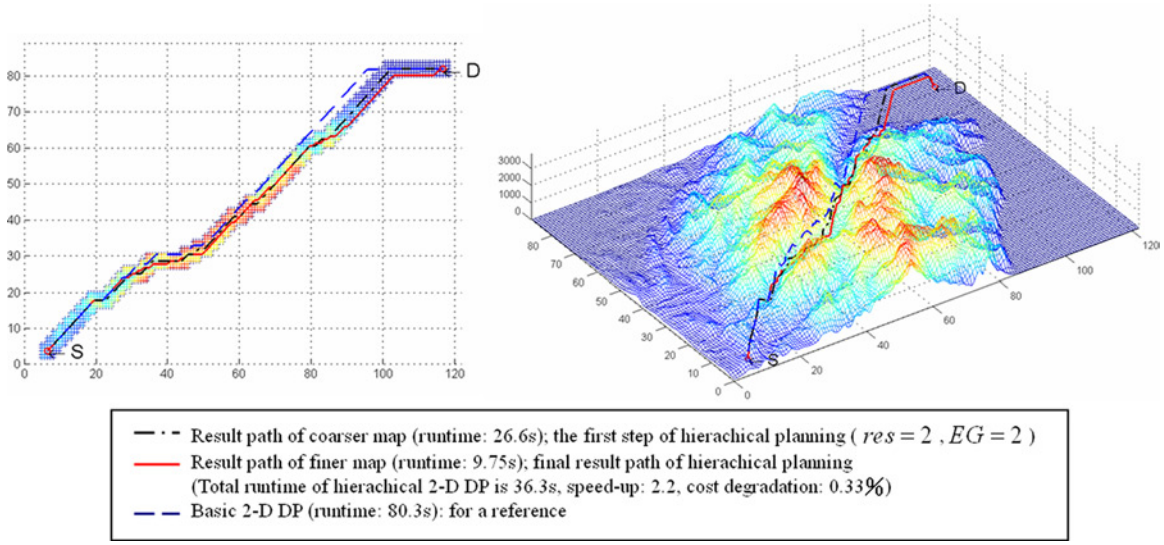

Fig. 17 Hierarchical 2-D DP (the search space is shown in left figure).

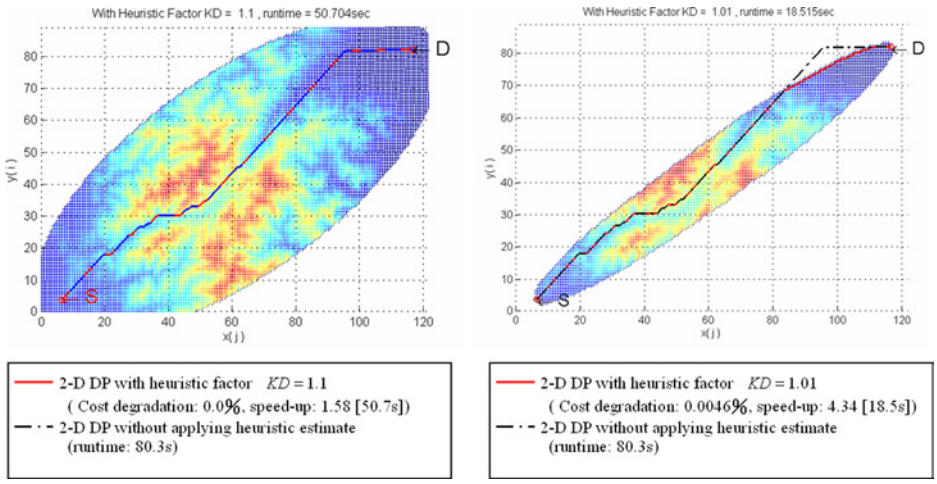

Fig. 18 The effect of heuristic factor *KD*.

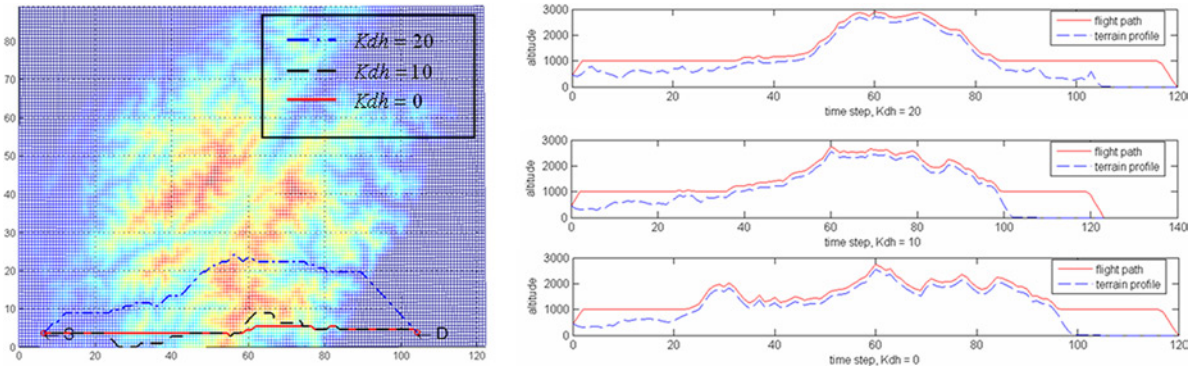

Fig. 19 The effect of fuel consumption ratio *Kdh*.

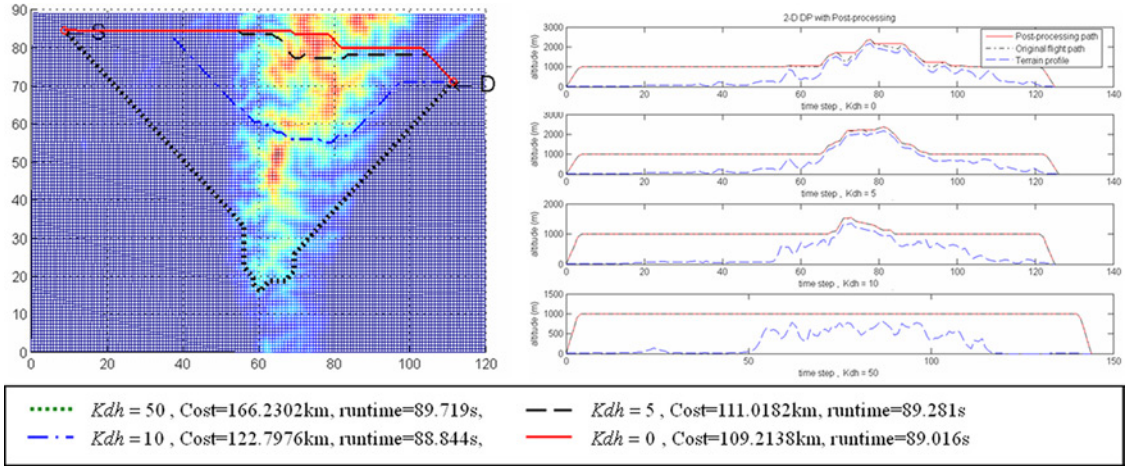

Fig. 20 With $Kdh = 10$ to have a path flying through valleys, and with a very large $Kdh = 50$ to have a path flying at cruise altitude without altitude changing.

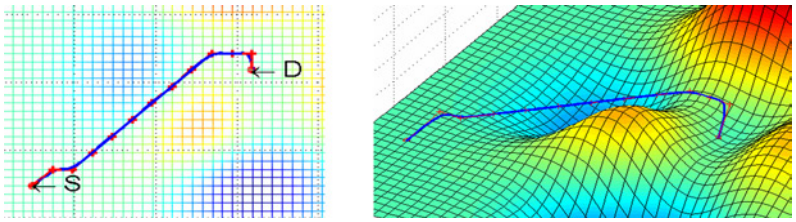

Fig. 21 The path smoothing by B-Spline curve fitting.

VI. Conclusions

A study of utilizing the dynamic programming (DP) method to determine a guaranteed shortest flight path on the DTEM has been investigated. The concept of reducing search space by construction a virtual terrain as a 2-D search space has been demonstrated. By using post-processing the possible shortcoming of rugged terrain when 2-D search space is applied could be overcome. Also the efforts including the multi-resolution (hierarchical) scheme and heuristic estimate are conducted to reduce the computation time (runtime) of the algorithm significantly. The simulation results show that it is feasible to use the DP method to make the flight path planning between two points efficient. Owing to the virtual terrain, which has been set up with consideration of both vertical and horizontal safety, it makes the flight path more credible. The consideration of fuel consumption ratio and the post-processing make it possible to find a more comfortable flight path or to save more fuel without frequent up-down movement. Furthermore, to satisfy any additional constraints which make the simulation more realistic, the cost function of the DP method is allowed to be multi-variable and easy to be implemented.

Acknowledgments

This work is supported by the National Science Council, Taiwan, under the Contract Number NSC 93-2212-E-006-034. The authors also acknowledge the assistance provided by Richard Hirst of JPBH Consulting Ltd, UK, in proofreading a draft of this paper.

References

[1]Hwang, Y. and Ahuja, N., "A Potential Field Approach to Path Planning," *IEEE Transactions on Robotics and Automation*, Vol. 8, No. 1, 1992, pp. 23–32.

[2]Barraquand, J., Langlois, B., and Latombe, J., "Numerical Potential Field Techniques for Robot Path Planning," *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 22, No. 2, 1992, pp. 224–241.

[3]Pai, D. K. and Reissell, L.-M., "Multiresolution Rough Terrain Motion Planning," *IEEE Transactions on Robotics and Automation*, Vol. 14, No. 1, Feb., 1998, pp. 19–33.

[4]Sinopoli, B., Micheli, M., Donato, G., and Koo, T. J., "Vision Based Navigation for an Unmanned Aerial Vehicle," *Proceedings of IEEE International Conference on Robotics and Automation*, Korea, May 2001, pp. 1757–1764.

[5]Rippel, E., Gill, A. B., and Shimkin, N., "Fast Graph-Search Algorithms for General-Aviation Flight Trajectory Generation," *Journal of Guidance, Control, and Dynamics*, Vol. 28, No. 4, July-August 2005, pp. 801–811.

[6]Rathbun, D., Kragelund, S., Pongpunwattana, A., and Capozzi, B., "An Evolution Based Path Planning Algorithm for Autonomous Motion of a UAV Through Uncertain Environments," *Digital Avionics Systems Conference*, Vol. 2, pp. 8D2–1∼8D2–12, 2002.

[7]Nikolos, I. K., Valavanis, K. P., Tsourveloudis, N. C., and Kostaras, A. N., "Evolutionary Algorithm Based Offline/Online Path Planner for UAV Navigation," *IEEE Transactions on Systems, Man and Cybernetics—Part B: Cybernetics*, Vol. 33, No. 6, Dec. 2003, pp. 898–912.

[8]Dijkstra, E. W., "A Note on Two Problems in Connexion with Graphs," *Numerische Mathematik*, Vol. 1, No. 1, Dec. 1959, pp. 269–271.

[9]Hart, P. E., Nilsson, N. J., Raphael, B., Correction to "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," *SIGART Newsletter*, 37, 1972, pp. 28–29.

[10]Stentz, A., "The Focussed D* Algorithm for Real-Time Replanning," *Proceedings of the International Joint Conference on Robotics and Automation*, Aug. 1995, pp. 3310–3317.

[11]Bellman, R. E., "*Dynamic Programming*," Dover Publications, Dover edition, 2003. ISBN 0486428095

[12]Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C., "*Introduction to Algorithms*," Second edition. MIT Press and McGraw-Hill, 2001. ISBN 0262032937

[13]Larson, R. E. and Casti, J. L., "*Principles of Dynamic Programming, Part II, Advanced Theory and Applications*," New York, Marcel Dekker, Inc., 1982. ISBN 0824765907

[14]Asseo, S. J., "In-Flight Replanning of Penetration Routes to Avoid Threat Zones of Circular Shapes," *Aerospace and Electronics Conference*, July 1998, pp. 383–391.

[15]Holdsworth, R., Lambert, J., and Harle, N., "Inflight Path Planning Replacing Pure Collision Avoidance, Using ADS-B," *Aerospace and Electronic Systems Magazine*, Feb. 2001, pp. 27–32.

[16]Kwok, K. S. and Driessen, B. J., "Path Planning for Complex Terrain Navigation Via Dynamic Programming," *Proceedings of American Control Conference*, S.D., Cal., 1999, pp. 2941–2944.

[17]U.S. Geological Survey (USGS), GTOPO30 Digital Elevation Model, Sioux Falls, South Dakota, USGS EROS Data Center, http://edc.usgs.gov/products/elevation/gtopo30/gtopo30.html [1997]

[18]Guth, P., Naval Academy, MICRODEM is a Free Software Package With a Built-in DEM Converter. MICRODEM is available at http://www.usna.edu/Users/oceano/pguth/website/microdemdown.htm

[19]Anderson, J. D., "*Introduction to Flight*," McGraw-Hill International, 1989. ISBN 0072990716

[20]Gifford, K. K. and Murphy, R. R., "Incorporating Terrain Uncertainties in Autonomous Vehicle Path Planning," *Proceeding IROS*96, 1996, pp. 1134–1140.

Ellis Hitt
Associate Editor