INDUSTRIAL DATA 2002, (5) 1 : 44-48

EL PROBLEMA DE LA PLANEACIÓN DEL MOVIMIENTO PARA VEHÍCULOS O ROBOTS

Eduardo Raffo L*., Edgar Ruiz L.**

RESUMEN

El artículo es una introducción al estudio del problema del planeamiento del movimiento de un vehículo o robot. Para ello, primero se plantea una solución para determinar la dificultad del terreno y luego planear el movimiento a seguir dentro de él.

Palabras clave: Movimiento. Planeamiento. Vehículos. Robots. Geometría computacional.

ABSTRACT

The article is an introduction to the study of the problem of the planning of the movement of a vehicle or robot. For it, first thinks about a solution to determine the difficulty of the terrestrial one and then to plan the movement to continue inside him.

Key words: Motion. Planning. Vehicles. Robots. Computational geometry.

INTRODUCCION

Sin duda la robótica se ha constituido en un tema de gran interés para los investigadores en ciencia y tecnología.

La navegación sobre un terreno es un componente clave en el diseño de vehículos piloteados a control remoto (VPCR); estos vehículos pueden viajar en tierra tales como un robot o un auto, otros pueden volar sobre tierra, tal como una avioneta o similar. Un VPCR es un sistema que contiene una computadora que almacena la información del terreno en el cual va a ser operado.

El problema de diseñar un software para resolver el problema de planeación del movimiento para un vehículo o robot puede afrontarse en dos fases: Fase I: Determinar la dificultad del terreno y Fase II: Planeación del movimiento.

PLANEACION DEL MOVIMIENTO

Fase I: Determinacion de la dificultad del terreno

Para estudiar esta fase, se asume que se tiene un terreno donde el vehículo se moverá. El software de computadora que guía estos vehículos debe ser probado sobre una amplia variedad de formaciones y topología del terreno. La información sobre grandes extensiones de terreno (rejilla o grid) esta disponible un una base de datos. Una manera de medir la "*dificultad*" de una rejilla o grid del terreno con respecto a la navegación sobre él; es determinar el número de colinas (cerros, picos o montañas) en la rejilla en estudio; donde una colina es un punto tal que los alrededores tienen menor elevación que él.

Para esta primera fase, se asume que los valores en las cuatro posiciones adyacentes a cierta posición [i, j] en la rejilla nos permitirán determinar si un valor en la posición [i, j] es una colina o no. Para ello se tiene en cuenta lo siguiente:

	rejilla[i-1, j]	
rejilla[i, j-1]	rejilla[i, j]	rejilla[i, j+1]
	rejilla[i+1, j]	

* Ingeniero Industrial: Instituto de Investigación - UNMSM
E-mail: eraffol@unmsm.edu.pe
** Ingeniero Industrial: Instituto de Investigación - UNMSM
E-mail: eruizl@unmsm.edu.pe

Lo cual implica que cada pico potencial debe compararse con sus cuatro puntos vecinos. Si todos los cuatro puntos vecinos son menores que el pico potencial entonces se trata de un pico real. Los datos de las elevaciones del terreno se leerán desde un archivo de datos para luego ser trasladados a una matriz o arreglo bidimensional. En este momento es importante observar que no todos los puntos de la matriz o grid de datos pueden ser picos potenciales; esto sucede con los bordes de la matriz pues cada uno de estos puntos no tienen sus cuatro vecinos completos; por ello, estos puntos no intervendrán en el proceso de evaluación.

El algoritmo básico es el siguiente

a. Leer los datos del terreno desde un archivo.
b. Cargar los datos leídos a un array bidimensional.
c. Determinar e imprimir la localización y el número de los picos.

El paso 1, consiste en leer los datos desde el archivo y el paso 2, es almacenar la data leída en el array o matriz. El paso 3 es un par de bucles para evaluar todos los picos potenciales, imprimir su localización y cuantos picos hay en la rejilla o grid. A continuación se presenta un refinamiento del algoritmo

a. Leer los datos del terreno desde un archivo
b. Cargar los datos leídos a un array bidimensional
c. Determinar e imprimir la localización y el número de los picos.

```
Contador = 0
for i = 1 to i<= n – 2 do
    for j = 1 to j<= n – 2 do
        if (elevacion[i-1, j]<elevacion[i, j]  &&
          elevacion[i+1, j]<elevacion[i, j] &&
          elevacion[i, j-1]<elevacion[i, j] &&
          elevacion[i, j+1]<elevacion[i, j] )
          imprimir("Pico en la fila ", i ,"columna ", j )
          contador++
    endfor
endfor
```

La implementación del algoritmo en cuestión puede realizarse utilizando algún lenguaje de programación como Basic, Pascal, C, o C++. Para éste estudio se ha escogido el software MATLAB®, por su potencia, versatilidad, eficiencia y por poseer un ambiente gráfico. Adicionalmente MATLAB posee un lenguaje (parecido al C) que facilita la tarea de programación de diversos algoritmos, útiles en ciencia e ingeniería y cuyos resultados sirven para un adecuado análisis e interpretación.

Para probar el algoritmo se asumen datos hipotéticos sobre un terreno considerando una matriz de 6 filas por 7 columnas, donde cada celda es una determinada elevación del terreno. Al respecto ver el código MATLAB en la figura 1.

```
function navigation
 grid=[5039 5127 5238 5259 5248 5310 5299;
       5150 5392 5410 5401 5320 5820 5321;
      5290 5560 5490 5421 5530 5831 5210;
      5110 5429 5430 5411 5459 5630 5319;
      4920 5129 4921 5821 4722 4921 5129;
      5023 5129 4822 4872 4794 4862 4245];
% grafica
   mesh(grid);
   colormap(jet);
   axis([1 8 1 6 3000 6000]);
   xlabel('x');
   ylabel('y');
   zlabel('z');

   title('Grafica de elevaciones');
% impresión de los picos
     dc
     [n,m]=size(grid);
   total=0;
   fprintf('Impresión de los picos\n');
   fprintf('======================\n');

   for i=2:n-1
     for j=2:m-1
       if((grid(i-1,j)<grid(i,j))&(grid(i+1,j)<grid(i,j))
         &(grid(i,j-1)<grid(i,j))& (grid(i,j+1)<grid(i,j))
         fprintf('fila :%5d columna :%5d\n',i,j);
         total=total+1;
       end
     end
   end
   fprintf('En total son %2d picos\n\n',total);
% los puntos extremos
   mini=1;
   minj=1;
   maxi=1;
   maxj=1;
   for i=2:n
     for j=2:m
       if(grid(i,j)<grid(mini,minj))
         mini=i;
         minj=j;
       else
         if(grid(i,j)>grid(maxi,maxj));
           maxi=i;
           maxj=j;
         end
       end
     end
   end
     fprintf('Punto minino: fila :
           %5d columna :%5d\n',mini,minj);
   fprintf('Punto maximo: fila :
          %5d columna :%5d\n',maxi,maxj);

% ingreso del plan de navegación
     maximo=0;
     fprintf('\nEntrada de datos :\n');
```

Figura 1. Código para el algoritmo que determina el número de colinas en un terreno.

Tanto los puntos encontrados como colinas en los datos de prueba y la gráfica de superficie correspondiente se muestran en las figuras 2 y 3 respectivamente. Observe que según los datos, la altura mínima para realizar el vuelo sobre el terreno es de 5039, existiendo en dicha superficie un total de 3 picos o colinas.

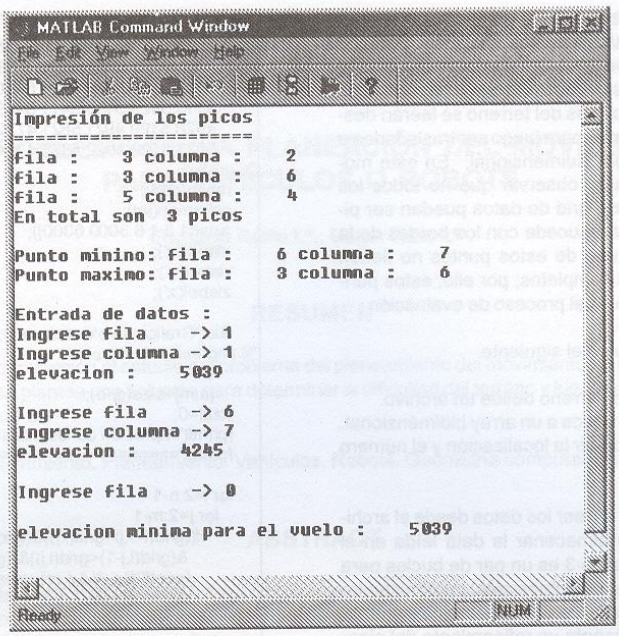

Figura 2. Salida para el código de la figura 1.

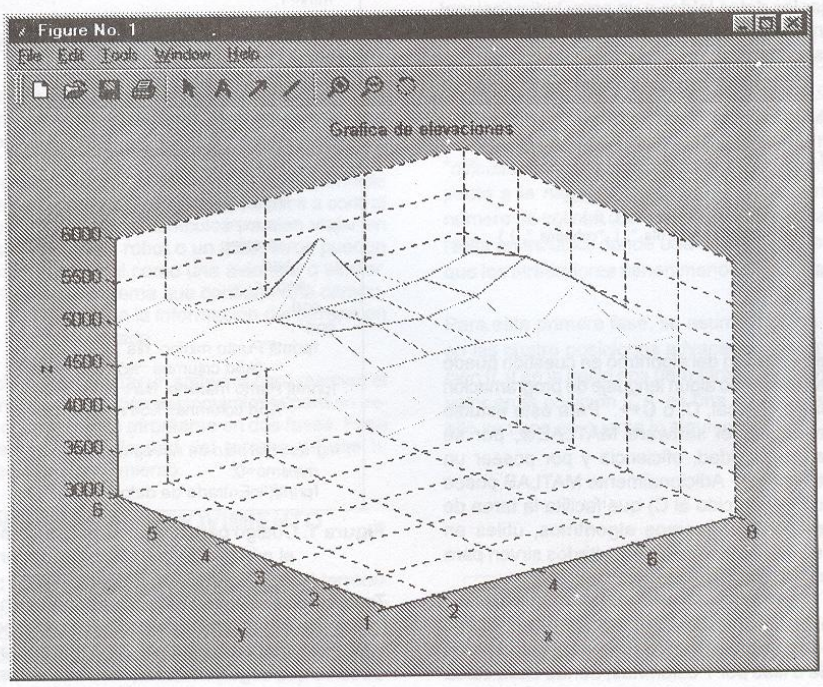

Figura 3. Superficie obtenida con los datos de prueba.

Fase II: Planeamiento del movimiento de un robot

Una de las metas recientes en robótica, es el diseño de robots autónomos; con la finalidad de que realicen una determinada labor, sin indicarles como hacerla. Otras veces se dice que el robot tiene preparado un plan para su movimiento.

Planificar el movimiento; significa, que el robot tiene conocimiento acerca del ambiente en el que se está moviendo. Así; un robot en una factoría deberá conocer los obstáculos, y donde estos, están localizados. Esta información la constituyen las paredes y la localización de las máquinas. El robot también posee dispositivos sensores; con la finalidad de detectar obstáculos que no se encuentran en el plan de movimiento(personas, por ejemplo). Con esta información el robot tiene que moverse desde un punto inicial o de referencia, hacia su posición meta(punto de llegada); sin tener que colisionar con ningún obstáculo.

Se denomina problema del planeamiento de movimiento (*Motion Planning Problem*); a la situación de resolver el dilema de cualquier clase de robot o vehículo que quiera moverse en un espacio físico. Esta definición asume en principio, que se trata de un robot autónomo moviéndose en el ambiente de la factoría; y no, de un brazo de robot (*arm robot*) o robot articulado.

El MPP es un problema difícil y en las siguientes líneas se realizaran algunos supuestos, con la finalidad de contar con un modelo simplificado.

Para empezar, se debe suponer que el MPP se realiza en un plano 2D; siendo esta una región plana, en donde los obstáculos son polígonos; así mismo el robot es otro polígono. El entorno es estático y no existe gente caminando alrededor del robot. Los robots poseen movimientos que dependen de sus mecanismos; algunos se mueven en cualquier dirección y otros tienen restringidos sus movimientos.

Sea *R* un robot que se mueve en un ambiente 2D o en un espacio de trabajo (*work space*), consistente de un conjunto S ={P_1,P_2, ..} de obstáculos. Se sume que *R* es un polígono simple. El lugar o configuración del robot, puede ser especificado por un vector traslación. Para el caso de ser el robot, un polígono (como por ejemplo un cuadrado), con vértices (1,-1), (1,1), (-1,1), (-1,-1); entonces los vértices de *R*(3,2) son: (4,1), (4,3), (2,3), (2,1). El robot inicialmente puede estar en *R* (0,0). En la figura 4, se presenta una trayectoria bajo este enfoque; y en la figura 5, el M-file de MATLAB correspondiente.

Un camino alternativo para visualizar esta situación, es mediante un punto de referencia. Intuitivamente si el origen es (0,0); éste corresponde al vector de traslación *R* (0,0), entonces para el par (*x*,*y*), el vector de traslación es *R* (*x*, *y*). Adicionalmente; si su orientación es de rotación; es necesario un nuevo parámetro que llamaremos F, donde *R* (*x*, *y*, F) denota el punto de referencia en (*x* ,*y*) rotado un ángulo F. Inicialmente es *R* (0, 0, 0).

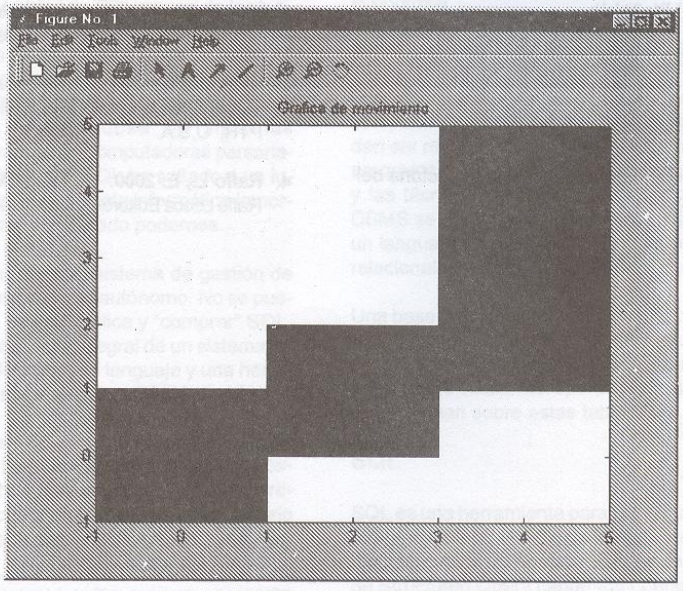

Figura 4. Trayectoria realizada por el robot.

El parámetro espacio para un robot R denotado por C(R) se le llama espacio de configuración; así un punto *p* en este espacio de configuración, corresponde a un lugar R (*p*) del robot en el espacio de trabajo.

El espacio de configuración de la traslación de un robot, es el plano Euclidiano dimensional 2D y es idéntico al espacio de trabajo para nuestro caso.

```
function Robot1
 R=[1 -1;1 1;-1 1;-1 -1];
 % grafica
  xlabel('x');ylabel('y');
  set(gca,'XTick',-1:1.0:5);
  set(gca,'yTick',-1:1.0:5);
  fill(R(:,1),R(:,2),'r');title('Grafica de movimiento');
% ingreso del plan de navegación
     fprintf('\nEntrada de datos :\n');
     contador=0;
     while 1
    fila=input('Ingrese fila   -> ');
    if(fila==0)
       break;
    end
                columna=input('Ingrese columna -> ');
   if(columna==0)
      break;
    end
    contador=rem(contador+1,2);
   P(:,1)=R(:,1)+fila;P(:,2)=R(:,2)+columna;
    hold on
    if contador==0
     fill(P(:,1),P(:,2),'r');
    else
    fill(P(:,1),P(:,2),'b');
    end
  end
```

Figura 5. Código para obtener la trayectoria del robot

CONCLUSIONES

El planeamiento del movimiento, es sólo un aspecto de un problema más amplio denominado el problema del planeamiento de tareas (Problem of task planning). Para determinar la dificultad de un terreno se requiere de una base de datos donde se consigne los puntos integrantes del grid o malla, cualquier punto es confrontado con sus cuatro vecinos para determinar si se trata de una colina o no. Sin embargo, si lo que se requiere es una mayor precisión o certeza puede compararse cualquier punto de la malla con sus ocho vecinos (si es que los tiene, por supuesto).

El uso de MATLAB en estos problemas es adecuado por su facilidades de programación y por sus comandos que permiten plotear los resultados en un gráfico; permitiendo recrear la superficie o terreno en cuestión.

El campo del estudio de la robótica es el diseño y uso de robots. En el mundo real, un robot es conceptualizado como un objeto geométrico que opera en 3D.

La geometría computacional estudia el MPP; siendo también de su dominio: computer graphics, geographic information systems (GIS), robótica y CAD/CAM.

BIBLIOGRAFÍA

1. **De Berg, M. , Van Kreveld, Overmars M. y Schwarzkopf O. 1997.** Computational Geometry: Algorithms and Applications. 1ra. Ed. Springer-Verlag, Berlin. Heidelberg.

2. **Etter, D. 1995.** Engineering Problem Solving whith C. 1ra. Ed., Prentice Hall PTR, United States of America

3. **Nakamura, S. 1996.** Numerical Analysis and Graphic Visualization with MATLAB. 1ra. Ed., Prentice Hall PTR, U.S.A.

4. **Raffo L., E. 2000.** Mi Primer MATLAB. 1ra. Ed., Raffo Lecca Editores, Lima Perú.