PLANIFICACION AUTOMATICA DE TRAYECTORIAS MEDIANTE TECNICAS DE PROCESAMIENTO DE IMAGENES

Autor: Ing.Marisa DE GIUSTI
Investigador Asistente (CICPBA)
Profesor Adjunto (FI-UNLP)

Abstract: El empleo de Robots móviles en un ámbito fabril posibilita el cambio de sistemas rígidos de transporte por esquemas flexibles, donde el flujo de materiales puede acomodarse dinámicamente y sin provocar interrupciones, según las necesidades de máquinas y productos.

Este artículo presenta un enfoque al problema de la planificación de las trayectorias de un Robot Móvil en un ámbito conocido, desde el punto de vista de la aplicación de técnicas de Computer Graphics. Este enfoque, en contraposición al habitualmente empleado de técnicas geométricas, posibilita la obtención de resultados con mucha menor tarea de procesamiento, usando un computador personal tipo PC_AT, y con precisión equivalente. Asimismo, y como conclusión novedosa, se muestra que a los fines del procesamiento lógico de imágenes binarias, un procesador convencional puede ser visto como una máquina SIMD (Single-Instruction Multiple-Data), al procesar múltiples elementos de imagen en paralelo.

Palabras Clave: PARALLEL IMAGE PROCESSING

I.INTRODUCCION

El empleo de Robots móviles en un ámbito fabril posibilita el cambio dinámico de sistemas rígidos de transporte por esquemas flexibles, donde el flujo de materiales puede acomodarse sobre la marcha sin provocar interrupciones, de acuerdo a las necesidades de máquinas y productos [1].

La planificación de las trayectorias de un Robot Móvil en un ámbito conocido es resuelta habitualmente usando técnicas geométricas [2], y usualmente mediante dos metodos: el del espacio de configuracion, que provee un medio efectivo para analizar en profundidad los problemas de planeamiento incluidos los movientos finos de ensamble [3], y el del espacio libre [4] donde una representacion del espacio libre mas sofisticada puede obtenerse mediante el uso de GVDs (Generalized Voronoi Diagrams) [5]. Este planteo, aunque irreprochable desde el punto de vista analítico, consume importante poder de cómputo y con ello tiempo, al requerir gran número de operaciones trigonométricas y aritméticas, usualmente en punto flotante; la precisión obtenida se contradice sin embargo con el empleo de modelos poligonales y de número de caras limitado para cada uno de los obstáculos del ambiente. Existen también trabajos que discretizan el ambiente de trabajo, empleando esquemas tipo OCTREE [6] para detallar eficientemente obstáculos arbitrarios, aunque esta representación conspira en contra a la hora del cálculo de trayectorias.

Este artículo ha encarado el problema mediante técnicas de Image Processing [7] y Computer Graphics [8], y modelizando el ambiente mediante una retícula binaria (bitmap); con este planteo no existe ningún tipo de restricciones en la forma de los objetos, y el uso de aritmética entera posibilita la obtención de resultados rápidos en una computadora tipo PC_AT, con mucha menor tarea de cómputo y similar precisión. En el punto II del artículo se define el problema a resolver y ciertos términos a emplear, en el punto III se da una descripción general de las sucesivas etapas que componen el proceso de búsqueda de la trayectoria, en los puntos IV a X se detalla cada una de estas etapas, y en el punto XI se realiza una evaluación de los resultados obtenidos.

II.PLANTEO DEL PROBLEMA

Dados los siguientes elementos:
- un ámbito de trabajo A de dimensiones U,V (con U y V naturales y distintos de cero) compuesto por un retículo bidimensional finito de U*V puntos.
- un conjunto de puntos {Pf} de acceso prohibido, que representan los obstáculos del ambiente A.
- un punto de partida Pp perteneciente a A no incluído en Pf y de coordenadas (Up,Vp).
- un punto destino Pd también perteneciente a A, no incluído en Pf y de coordenadas (Ud,Vd).

- un robot móvil M, representado en primer instancia por un círculo de radio R, y posteriormente por un rectángulo de dimensiones arbitrarias (L1,L2) donde L1 es la dimensión del lado menor y L2 la del lado mayor.

El problema a resolver consiste en encontrar un camino para el robot móvil M, entre Pp y Pd, que satisfaga las condiciones de ser continuo, no incluir puntos del conjunto {Pf} y satisfacer algún criterio de costo (en este caso, longitud de camino mínima). Dado que el ámbito de trabajo es esencialmente discreto, se define que debe entenderse que dos puntos M,N establecen entre sí una continuidad (son vecinos) si sus coordenadas tanto Um,Un como Vm,Vn difieren entre sí no más que una unidad de distancia.

A fines que serán necesarios, se define el rumbo de la vecindad de un punto Px con cada uno de sus ocho posibles vecinos Pn con la siguiente convención:

- Pn está al ESTE (E) de Px si Un=Uo+1 y Vn=Vo.
- Pn está al NORESTE (NE) de Px si Un=Uo+1 y Vn=Vo-1.
- Pn está al NORTE (N) de Px si Un=Uo y Vn=Vo-1.
- Pn está al NOROESTE (NO) de Px si Un=Uo-1 y Vn=Vo-1.
- Pn está al OESTE (O) de Px si Un=Uo-1 y Vn=Vo.
- Pn está al SUDOESTE (SO) de Px si Un=Uo-1 y Vn=Vo+1.
- Pn está al SUD (N) de Px si Un=Uo y Vn=Vo+1.
- Pn está al SUDESTE (SE) de Px si Un=Uo+1 y Vn=Vo+1.

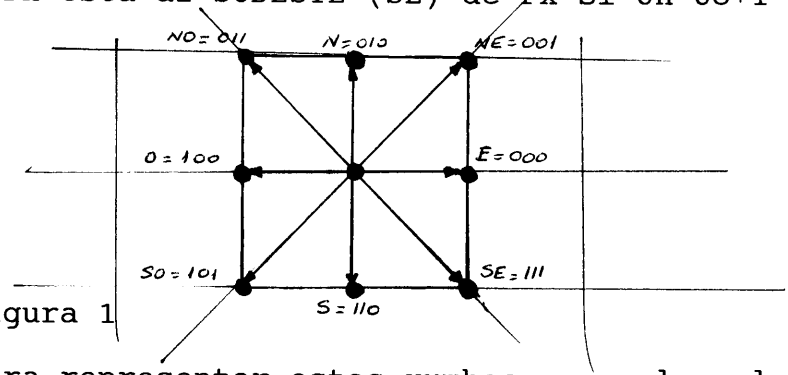

Figura 1

Para representar estos rumbos se emplean los siguientes códigos de tres bits: E=000; NE=001; N=010; NO=011; O=100; SO=101; S=110 y SE=111.

III.PASOS A RESOLVER

La búsqueda de un camino entre Pp y Pd requiere la ejecución de los siguientes pasos de procesamiento:

a) ADQUISICION DEL AMBIENTE: este paso implica definir la dimensión del ambiente, la posición de los puntos de salida y llegada y los obstáculos existentes. En las experiencias realizadas se hizo uso de un ambiente de 256x256 puntos, y se usó un scanner manual para facilitar el proceso de ingreso de obstáculos arbitrarios. Esta decisión es totalmente compatible con el posible ingreso del mapa del ámbito de trabajo mediante técnicas de CAD o similares.
b) ENGROSADO DE OBSTACULOS: este paso requiere que se engrosen los obstáculos del ambiente en R (para el caso del robot móvil circular), o en L1/2 (en el caso de un Robot

móvil rectangular), para poder modelizar la interacción del robot con los obstáculos y detectar las colisiones.
c) ESQUELETIZACION DE CAMINOS: en esta etapa las áreas libres del ambiente son esqueletizadas para definir trayectorias a lo largo de todos los caminos posibles.
d) DETECCION DE ENCRUCIJADAS: este etapa busca los puntos singulares del esqueleto (extremos de segmentos o puntos donde confluyen varios caminos).
e) CREACION DEL GRAFO DE TRAYECTORIAS: a partir de la inspección del esqueleto y los puntos singulares, este paso genera un grafo de conexionado con forma de matriz, que define y pondera los caminos existentes entre los puntos singulares del ambiente.
f) BUSQUEDA DEL CAMINO MINIMO: una vez definido el grafo de conexionado este paso utiliza algoritmos clásicos para buscar el camino mínimo entre los puntos de salida y llegada.
g) VALIDACION DEL CAMINO: si el robot ha sido modelizado en la forma de un rectángulo, este paso verifica que la trayectoria sea válida, y en caso de no serlo modifica el grafo de trayectorias y vuelve a ejecutar la etapa "f".

IV.ADQUISICION DEL AMBIENTE

La definición del ámbito de desplazamiento del Robot móvil con sus obstáculos fué resuelta mediante la digitalización de un dibujo de dicho ámbito a través de un Scanner de mano, tipo DFI HS-3000 Plus [9]. La imagen digitalizada, de 512x512 puntos, puede ser enmarcada luego de la adquisición en un cuadro de 256x256 puntos, y su contenido almacenado en disco.

La selección de una retícula de 256x256 puntos no es arbitraria: un procesador como el de la PC_AT [10], con bus de datos de 16 bits puede emplear 8 bits para la coordenada U y 8 para V; asimismo las distintas tablas de 256x256=65536 puntos (8kbytes cada una) que se emplean para los distintos pasos de procesamiento son fácilmente manejables dentro de mapas de datos de 64kBytes como máximo, tal como debe asumirse para los fines prácticos a una PC, en la que no se realicen permanentes cambios de segmentos.

V.ENGROSADO DE OBSTACULOS

Tal como se observa en la literatura [2], la forma más práctica de considerar el fenómeno de desplazamiento de un móvil circular de radio R en un ambiente con obstáculos es engrosar a los obstáculos en R y considerar luego el desplazamiento de un móvil puntual.

Para el caso de obstáculos rectangulares, en [5] se consideran varios casos particulares que requieren, cada uno, soluciones específicas.

En este trabajo, el problema de los móviles rectangulares se ha resuelto del siguiente modo:

a) Se engrosan los obstáculos en forma circular con un radio L1/2.
b) Se calcula la mejor trayectoria posible.
c) Se valida la solución encontrada verificando si un móvil lineal (ya no puntual) de largo L2 recorre los segmentos de la trayectoria sin colisionar con los obstáculos engrosados.
d) Si se detecta una colisión, se anula dicho segmento de entre las trayectorias posibles y se vuelve al punto b).

Esta solución, aunque puede requerir varias iteraciones en ambientes con espacios muy estrictos, es mucho más rápida y mucho más proxima a la realidad industrial, donde los "cuellos de botella" son situaciones excepcionales.

VI.ESQUELETIZACION DE CAMINOS

Dado el retículo del ambiente y los obstáculos engrosados queda definida toda una región libre del ambiente que es recorrible por el Robot. Esta región libre, sin embargo, no requiere ser analizada en su totalidad, sino que basta considerar un conjunto de líneas, que definen los puntos medios de cada región de circulación, y que componen el esqueleto [11] de la región libre.

El empleo del esqueleto de la región libre es similar al uso del diagrama de Voronoi, con la diferencia fundamental en cuanto hace a la forma de calcularlo.

El cálculo rápido del diagrama de Voronoi requiere el uso de obstáculos poligonales, pues los segmentos de este diagrama surgen de la consideración de la posición de los vértices de dichos obstáculos, y el tiempo de su cálculo crece notablemente con el número de vértices. Por ello, aunque el calculo de las ramas del diagrama se realice con precisión "infinita", la circunstancia de necesitar modelizar a los obtáculos mediante polígonos del menor número posible de lados invalida la precisión obtenida.

En cambio, el empleo del esqueleto es mucho más general, dado que su tiempo de cálculo y complejidad algorítmica es independiente de la forma de los obstáculos.

En este trabajo, el cálculo del esqueleto se ha realizado mediante la técnica de "incendio de pastos", empleando métodos de image processing [12].

En esta solución la decisión sobre si un punto perteneciente a la región libre pertenece al esqueleto o debe borrarse se toma a través de la aplicación iterativa de funciones de esqueletización (SkE, SkN, SkO y SkS) que determinan si ese punto es incendiable desde los rumbos Este, Norte, Oeste o Sud, respectivamente.

Cada una de estas funciones opera considerando el valor de una ventana compuesta por el punto de interés y

sus ocho puntos vecinos y la decisión de si debe borrase o no surge del cálculo de una función lógica [13].

Dado que el procesador de un sistema tipo PC no es apto para el manejo de arreglos bidimensionales de información ni para realizar instrucciones bit a bit, se aprovechó la particular configuración de la imagen en memoria, y la agilidad de estos procesadores para la realización de acciones elementales repetitivas.

La imagen del ámbito del Robot está organizada en memoria por líneas, y cada una de las 256 líneas se forma con paquetes sucesivos de 16 pixels almacenados como palabras de 16 bits en direcciones consecutivas de memoria. Si se numeran los bits de una palabra asignando 15 al bit mas significativo (MSB), que corresponde al pixel de la izquierda del paquete y 0 al menos significativo (LSB), la ventana de 3x3 pixels asociada al bit 'n' de la palabra 'm' (n=1..14) tiene por vecinos:

Vecino NO: bit 'n+1' de la palabra (m-16)
Vecino N: bit 'n' de la palabra (m-16)
Vecino NE: bit 'n-1' de la palabra (m-16)
Vecino O: bit 'n+1' de la palabra m
Vecino E: bit 'n-1' de la palabra m
Vecino SO: bit 'n+1' de la palabra (m+16)
Vecino S: bit 'n' de la palabra (m+16)
Vecino SE: bit 'n-1' de la palabra (m+16)

Para acelerar la tarea de cálculo, a la imagen de trabajo se le agrega una línea inicial y otra final de 16 palabras en '0', y en base a esta información (que se denomina TABLA0) se generan dos nuevas tablas:

Tabla1: la imagen original desplazada un bit hacia la derecha, con todas las palabras asociadas al margen derecho de la imagen con el MSB en '0'.
Tabla2: la imagen original desplazada un bit hacia la izquierda, con todas las palabras asociadas al margen izquierdo de la imagen con el LSB en '0'.

Con estas tres tablas, la ventana de trabajo de 3x3 pixels asociada al bit 'n' de la palabra 'm' de la TABLA0 está dada por:

Vecino NO: bit 'n' de
TABLA1[m-16]
Vecino N: bit 'n' de
TABLA0[m-16]
Vecino NE: bit 'n' de
TABLA2[m-16]
Vecino O: bit 'n' de
TABLA1[m]
Vecino E: bit 'n' de
TABLA2[m]
Vecino SO: bit 'n' de
TABLA1[m+16]
Vecino S: bit 'n' de

```
          TABLA0[m+16]
Vecino SE: bit 'n' de
          TABLA2[m+16]
```

Es también conveniente generar tres nuevas tablas llamadas TABLA_3, TABLA_4 y TABLA_5, que son copia de TABLA_0, TABLA_1 y TABLA_2, respectivamente, con todos sus elementos negados. Gracias a estas tablas los 9 componentes de una ventana corresponden al mismo bit en distintas palabras en las seis tablas y con distinto 'offset'; esto permite aprovechar la capacidad lógica y de direccionamiento del procesador, que ejecuta en una única instrucción el AND(&), OR(|), NOT(~) o XOR(^) bit a bit de 16 bits por vez. Además, se puede ver también que el cálculo de 'q+16' y 'q-16' en los índices de las tablas es innecesario; si se asigna al nombre de una tabla la dirección de inicio de dicha tabla, el valor de TABLA_i[q+16] puede ser visto en realidad como el contenido de una nueva tabla virtual TABLA_iP[q] y el de TABLA_i[q-16] como el contenido de otra tabla virtual TABLA_iM[q], todas ellas superpuestas en memoria, sólo que TABLA_iP[] comienza 16 palabras después de TABLA_i[] y TABLA_iM[] 16 palabras antes.

Dado un pixel con valor '1', ubicado en el centro de una ventana de 3x3 elementos, existen 256 posibles combinaciones de los 8 elementos que los rodean, que definen si ese pixel debe o no ser puesto a '0' en la esqueletización. Si se asigna peso binario a los vecinos del pixel central, de la siguiente forma:

```
+                    +
 a= 128 b = 64 c = 32
+                    +
 d = 16    X   e = 8
+                    +
 f = 4  g = 2  h = 1
+                    +
```

Y se considera la función de esqueletización desde arriba hacia abajo sk_N() para todos los casos, teniendo en cuenta la existencia de simetría central, y con las siguientes reglas:

- Un punto aislado X es borrable.
- Un punto X es borrable solo si su eliminación no interrumpe ninguna línea de espesor unitario.
- Un punto X es borrable solo si no tiene ningún punto inmediatamente arriba suyo.
- Un punto X que sea extremo de línea no debe borrarse.

Resultan los siguientes valores, para los 256 posibles pesos de la ventana:

```
 0...15  0000000000110011
16...31  0011001100110011
32...47  0000000000110011
48...63  0000000000110011
```

```
64...79  0000000000000000
80...95  0000000000000000
96..111  0000000000000000
112..127 0000000000000000
128..143 0000000000000000
144..159 0011001100110011
160..175 0000000000000000
176..191 0000000000110011
192..207 0000000000000000
208..223 0000000000000000
224..239 0000000000000000
240..255 0000000000000000
```

de donde

```
skN () = g * /b *((e * /a)
       + (e * d) + (/c * d))
```

Si se llama:

```
i= ~a; j= ~b; k= ~c; l= ~d;
m= ~e; n= ~f; o= ~g; p= ~h;
```

Resulta:

```
skN () = g*j*((e*i) + (e*d)
       + (k*d));
```

Siendo:

```
     a = TABLA_1[q-16];
     b = TABLA_0[q-16];
     c = TABLA_2[q-16];
     d = TABLA_1[q];
     e = TABLA_2[q];
     f = TABLA_1[q+16];
     g = TABLA_0[q+16];
     h = TABLA_2[q+16];
     i = TABLA_4[q-16];
     j = TABLA_3[q-16];
     k = TABLA_5[q-16];
     l = TABLA_4[q];
     m = TABLA_5[q];
     n = TABLA_4[q+16];
     o = TABLA_3[q+16];
     p = TABLA_5[q+16];
```

De igual forma, las funciones para adelgazar desde otras direcciones se obtienen por permutación de variables, quedando:

```
skE () = d*m*((g*k) + (g*b)
       + (p*b));
skS () = b*o*((d*p) + (d*e)
       + (n*e));
skO () = e*l*((b*n) + (b*g)
       + (i*g))
```

Con estas tablas, la función adelgazo_N() (incendio de pastos desde el Norte) en lenguaje C se define como:

```
for (q=16;q<4112;q++)
```

```
TABLA_D[q] &= ~skN(q);
```

lo que muestra cómo con sólo 4096 iteraciones se procesa toda la imagen de 256x256 pixels. En este aspecto, el procesador opera como una máquina SIMD que procesa en paralelo a 16 pixels independientes por vez.

Si se llama I_0 a la imagen original, e I_q (con q=1..4) a imágenes intermedias, la esqueletización de la región libre del Robot se describe por:

```
do {I_1 = I_0;
    I_2 = adelgazo_N (I_1);
    mantengo Pp y Pd en '1'
    I_3 = adelgazo_E (I_2);
    mantengo Pp y Pd en '1'
    I_4 = adelgazo_S (I_3);
    mantengo Pp y Pd en '1'
    I_0 = adelgazo_O (I_4);
    mantengo Pp y Pd en '1'
} while (I_1 != I_0);
```

A cuyo fin la región libre del Robot se transforma en un esqueleto, donde los posibles caminos entre el punto de partida y de llegada están dados por las ramas de dicho esqueleto.

VII.DETECCION DE ENCRUCIJADAS

Se define "punto de encrucijada (PE) " como:
- Un punto X es PE si es extremo de una línea.
- Un punto X es PE si de él salen más de dos segmentos de trayectoria.

La detección de PEs es similar al problema de la esqueletizacion. Si se calcula el valor de la función PE(X) para los posibles 256 valores de los vecinos a..h del punto X, y con similar definición de i..p como los valores negados de a..h, resulta una función de detección mucho más compleja, dada por:

```
PE(X) = i*j*k*l*m*g + c*d*g
      + i*j*l*e*n*o + d*e*g
      + j*k*d*m*o*p + b*e*g
      + b*l*m*n*o*p + b*e*f
      + c*m*f*o*h + c*d*m*h
      + a*l*f*o*h + b*f*o*h
      + a*j*c*m*h + a*l*e*f
      + a*j*c*l*f + a*j*c*g
      + b*d*h + b*d*g
      + b*d*e + a*e*g
      + i*j*k*l*m*n*o*h
      + i*j*k*l*m*f*o*p
      + i*j*c*l*m*n*o*p
      + a*j*k*l*m*n*o*p
```

La aplicación de este operador a la imagen del esqueleto produce en una única pasada otra imagen con pixels en "1"

sólo en los PEs, por lo que una rápida inspección de esta nueva imagen permite hallar las coordenadas de cada uno de estos puntos, que serán los nodos del grafo de conexionado.

VIII.CREACION DEL GRAFO DE TRAYECTORIAS

Detectados 'n' PEs, ellos definen una matriz de distancias MD [nxn] donde:

```
MC[i,j] = MC[j,i];
MD[i,i] = 0;
```

Dado que la unidad que se use para la medición de la distancia entre dos PEs es intrascendente, y sólo importa la relación entre ellas, se adoptó como unidad de desplazamiento en los rumbos horizontales y verticales 0,2,4,6 el valor "5" y para los rumbos diagonales 1,3,5,7 el valor "7"; esta selección tuvo cuenta que la relación 7::5 es muy próxima a la real entre ellos de SQRT(2)::1.

Con este criterio, el algoritmo de ponderación de la matriz MD es:

```
hago todos los MD[] = oo;
hago todos los MD[i,i] = 0;
while haya algún PE(i) {
 if PE(i) no tiene vecinos
       { lo borro }
 else {
  distancia = 0;
  do {
   me muevo al vecino
   actualizo distancia
   if no estoy en un PE {
    busco un vecino
  }}
  while no esté en un PE(j)
  if (i != j) {
   MD[i,j]=MD[j,i]=distancia
}}}
```

Al finalizar esta etapa, la matriz MD[] tiene en todos sus elementos valores que, si son distintos de oo indican que entre ambos puntos existe un camino de la longitud indicada.

IX.BUSQUEDA DEL CAMINO MINIMO

La búsqueda del camino mínimo que une entre sí a los PEs correspondientes al Pp y al Pd es un problema habitual en teoría de grafos, y en este caso se utilizó el algoritmo de Dijkstra [14], que mediante una técnica de búsqueda amplia garantiza que el camino encontrado es de longitud mínima.

En este algoritmo se asocia a cada nodo una estructura que define:

a) S: estados de un nodo:
   INDEFINIDO (I) o

```
   PERMANENTE (P) o
   TENTATIVO (T);
b) V: nodo vecino en el
   camino hacia el destino.
c) D: distancia al destino.
```

El algoritmo se inicia rotulando al nodo destino como PERMANENTE y a los restantes como INDEFINIDOS, y finaliza cuando todos los nodos quedan PERMANENTES.

Si indico como nodo destino a "d" y de partida a "p", el algoritmo es:

```
para_todo nodo(i) tal que
 (i != p) y (MD[i,d] != oo)
marco: nodo(i).V = p
       nodo(i).D = MD[i,d]
       nodo(i).S = T

while haya uno o mas nodo(i)
  tal que (nodo(i).S = T)
{
 elijo nodo(i) con menor
  valor de nodo(i).D
 marco: nodo(i).S = P
 para todo nodo (j) {
  si ((j <> i)
      y (MD[i,j] <> oo)
      y (nodo(j).S <> P))
   {
    si (nodo(j).S = I) {
      marco:
       nodo(j).S = T
       nodo(j).V = i
       nodo(j).D =
             nodo(i).D
             + MD[i,j]
     }
    si (nodo(j).S = T) {
     si (nodo(j).D >
        nodo(i).D + MD[i,j])
      {
       marco:
        nodo(j).V = i
        nodo(j).D =
              nodo(i).D
              + MD[i,j]
}} } }}
```

Al terminar la ejecucion del algoritmo, el camino minimo esta definido en forma de una lista encadenada, donde cada nodo, con su elemento nodo(i).V indica cual es el proximo hacia el cual moverse, y donde el valor nodo(p).D indica la distancia minima desde el nodo de partida al de destino.

X.VALIDACION DEL CAMINO

Si el robot movil puede ser adecuadamente modelizado mediante un circulo, el camino obtenido mediante el paso IX es el camino deseado.

Si, en cambio, la forma del Robot es de tipo rectangular, la trayectoria obtenida debe validarse, pues hasta el paso IX solo se ha verificado que el camino tenga ancho mayor o igual que la dimension L1 del Robot.

El proceso de validacion implica hacer recorrer las trayectoria elegida por un segmento de recta de longitud L2, que mantenga sus extremos (cabeza y cola) sobre el camino, y verificar a cada paso si ese segmento colisiona o no con la imagen engrosada de los obstaculos. De ser este el caso, ello implica que ese segmento es intransitable para el Robot, por lo que los terminos MD[i,j] y MD[j,i] correspondientes son puestos en oo y se vuele a calcular el camino minimo.

Para ejecutar la tarea de validacion, si se llama al punto de cabeza P(h) y al de cola P(t), y usando dos variables internas l y m, se usa el siguiente algoritmo:

```
P(h) = P(p)
mientras (P(h) <> P(d)) {
 elijo vecino a P(h) tal que
  vecino pertenezca a la
  trayectoria minima
 muevo P(h) al vecino
 l = (P(h).x - P(t).x)^2
   + (P(h).y - P(t).y)^2
 mientras (l > L2^2) {
  elijo vecino a P(t) tal
    que vecino pertenezca a
    la trayectoria minima
  muevo P(t) al vecino
  m = (P(h).x - P(t).x)^2
    + (P(h).y - P(t).y)^2
  si (m > l) --> ERROR
  sino l = m
  }
 para cada punto P(?) del
    segmento P(h)..P(t) {
  si P(?) intersecta a los
      obstaculos --> ERROR
} }
```

El calculo de la tarea de validacion puede hacerse tambien rapidamente mediante aritmetica entera, dado que al utilizar L2 al cuadrado se hace innecesario la funcion Raiz Cuadrada.

Otro elemento a tener en cuenta es la forma de determinar los puntos P(?) intermedios al segmento P(h)..P(t). En este trabajo, explotando la caracteristica discreta del ambiente del Robot, se ha utilizado el algoritmo de Bresenham [_], usual en Computer Graphics, que permite generar los puntos intermedios del segmento con

solo sumas y desplazamientos (shifts), y es por lo tanto sumamente rapido.

Aunque existen casos en los cuales esta tecnica de validacion puede cancelar caminos transitables, en los que ni la cabeza ni la cola se mantengan en la trayectoria, bastan para la mayor parte de los casos.

XI.CONCLUSIONES

Una objeción válida a este trabajo es que para un ámbito de trabajo prefijado, el tiempo que tarde una técnica de cálculo de trayectorias frente a otra es un punto casi sin importancia. Al respecto debe aclararse que la idea clave de este artículo es la explotación inteligente de la capacidad de procesadores convencionales para trabajar con imágenes; el método descripto permite procesar tantos pixels por vez como tenga el 'bus' de datos de la CPU, lo que resulta sumamente interesante para microprocesadores y DSPs de 32 y 64 bits.

XII.BIBLIOGRAFIA

[1] H.Place, L.Marce, M.JULLIERE. "Les Robots Mobiles". Point en Robotique, Vol.1, Chap.14. pp.193-206, Ed.Lavoisier, France, 1983.

[2] T.Lozano Perez & M.A.Wesley. "An algorithm for planning collision-free paths among polyhedral obstacles," Commun.ACM, vol.22, pp.560-570, 1979.

[3] T.Lozano Perez. "Spatial Planning: A Configuration Space Approach". IEEE Transactions On Computers, vol.C32, pp.108-120, February 1983.

[4] R.A.Brooks. "Solving the Find Path Problem by good Representation of Free Space". IEEE Transactions on Systems, Man and Cybernetics, vol SMC13, no.3, pp.190-197, March-April 1983.

[5] O.Takahashi & R.J.Schilling. "Motion Planning in a Plane Using Generalized Voronoi Diagrams," IEEE Transactions on Robotics and Automation, vol.5, no.2, pp.143-150, April 1989.

[6] B. Faverjon. "Obstacle Avoidance Using an OCTREE in the Configuration Space of a Manipulator", IEEE Proceedings, Internal Conference On Robotics, March 1984.

[7] R.Gonzalez & P.Wintz. "DIGITAL IMAGE PROCESSING". Addison Wesley Pub.Co., 6th.Printing, USA, 1979.

[8] W.Newman & R.Sproull. "Principles Of Interactive Computer Graphics, 2d.Ed.", McGraw Hill Book Company, USA, 1979.

[9] "HD-3000 Handy Scanner Users Manual". DFI, USA, 1989.

[10] Assembler 8086/286.
Intel, USA, 1986.

[11] B.Cernuschi ."VISION PARA COMPUTADORAS". III EBAI, Brasil, 1988.

[12] R.Duda & P.Hart. "PATTERN CLASSIFICATION AND SCENE ANALYSIS". John Wiley & Sons, USA, 1973.

[13] M.DE GIUSTI & G.JAQUENOD. "Image Processing and Data Compression for Cattle Brand Applications". Presentado a su publicacion por la Universidad de New Mexico.

[14] Tannenbaum. "Computer Networks & Protocols".