UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

PROGRAMA DE MAESTRIA Y DOCTORADO EN INGENIERIA

FACULTAD DE INGENIERIA

DESARROLLO E INVESTIGACIÓN DEL SISTEMA DE CONTROL PARA UN ROBOT MOVIBLE

T E S I S

QUE PARA OPTAR POR EL GRADO DE:

MAESTRO EN INGENIERIA MECANICA

P R E S E N T A:

DAVID VÁZQUEZ RAZO

TUTOR:

Dr. ERNST KUSSUL

Co- director

Dra. TATIANA BAYDYK

México D.F., Ciudad Universitaria 2007

JURADO ASIGNADO:

Presidente: (**Dr. Marcelo López Parra**)

Secretario: (**Dr. Ernst Kussul**)

Vocal: (**Dr. Vicente Borja Ramírez**)

1er. Suplente: (**Dra. Tatiana Baydyk**)

2do. Suplente: (**Dr. Ángel A. Rojas Salgado**)

Centro de Ciencias Aplicadas y Desarrollo Tecnológico

TUTOR DE TESIS:

DR. ERNST KUSSUL

_____

FIRMA

CO. DIRECTOR:

DRA. TATIANA BAYDYK

_____

FIRMA

Dedico el presente trabajo a:

Eusebio Vázquez Conde
Matilde Razo Durán

Mis papás
Que sin importar las dificultades que conlleva formar
una familia numerosa, en tiempos difíciles, siempre
estuvo presente en Ellos el darme su amor y sus mejores
cuidados.

Lo que despertó en mí: admiración, respeto y amor.
No olvido los momentos bellos y aun los difíciles que vivimos juntos.
El recuerdo revive en mí con mayor intensidad
cada vez que pienso en Ellos, doy gracias a Dios por la gran
familia que me dio.

Su ejemplo es el motor que me impulsa a seguir adelante
y ver la vida como una oportunidad para ser una persona honorable.

A mi esposa **María Isabel** y a mis hijos
David Alberto y **Joel**

Al Dr. Ernst Kussul y a la Dra. Tatiana Baydyk

Por el interés y disposición plena para el logro
del presente trabajo. Fue interesante conocerlos y trabajar juntos.
Sin olvidar que el presente trabajo fue una propuesta del Dr. Ernst
y únicamente me dediqué a dar forma a su idea, con ayuda de la Dra. Tatiana

A la **Universidad Tecnológica de Querétaro**

Por el apoyo que me brindó.
A través de:

Rector.
M.I. Raúl Noriega Ponce

Secretario Académico
M.E. Héctor Julián Vázquez Ramírez

Director de la Carrera de Electrónica y Automatización
M.C. José Felipe Aguilar Pereyra

Y a todas las personas que han contribuido
con su valiosa ayuda para el logro del presente trabajo

Gracias,

CONTENIDO

RESUMEN

El problema del control de un robot autónomo viene resolviéndose últimamente utilizando un control local de bajo nivel que proporciona competencias elementales [Domingo Gallardo y otros] – evitar obstáculos, seguir paredes, entrar en puertas – y una capa superior (programa principal) que utiliza dicha funcionalidad para la navegación efectiva y para la solución del problema completo. Bajo este planteamiento, diferentes investigadores han trabajado en la obtención de técnicas eficientes para el control local que permita al robot obtener una navegación segura.

En este trabajo proponemos el diseño del algoritmo de esquemas locales de control basado en la generación de una conducta óptima utilizando técnicas de computación, que permite la generación de la trayectoria que el robot seguirá desde el punto de inicio hacia la meta, y evitar obstáculos que se presente durante su recorrido.

Para lo cual, se realiza un mapeo bidimensional para delimitar la frontera de cada uno de los obstáculos que están presentes en el ambiente en el que el robot se moverá.

A continuación y en cada punto de movimiento el robot emite un haz por medio de un barrido vectorial de forma circular con un campo de acción de 360^{o} (cuya finalidad es representar la función que desempeña uno o varios sensores ópticos montados a bordo del robot) y cuyo alcance está limitado por el mismo.

Se determina el valor de las coordenadas (x, y) – en el plano cartesiano en dos dimensiones – en la intersección del haz que el robot emite y el obstáculo. Se proponen tres formas para generar la trayectoria que el robot a de seguir:

- La primera utiliza la información de la frontera del obstáculo y se le indica al robot que siempre se moverá alrededor del objeto y posteriormente se dirigirá al punto destino
- La segunda utiliza la nuestra representativa de los diferentes puntos detectados y con el uso de interpolación lineal por mínimos cuadrados [Steven C. Chapra, Raymond P.], se ajusta la distribución de los puntos por medio de la ecuación de la recta; a su vez se calcula la media de los puntos detectados en ambas direcciones y se calcula la desviación estándar. Valores que permiten determinar las dimensiones del objeto y que se emplean para determinar el radio mínimo de giro que permite definir puntos de control para generar Curva de Bézier [Bézier, P. 1972], trayectoria que seguirá el robot para esquivar el obstáculo
- La tercera forma, - que es la aportación del trabajo – durante cada movimiento que ha de realizar el robot compara la distancia y orientación que existe entre la posición actual en la que se encuentra el robot y la meta ó destino. Los valores obtenidos se introducen en una fórmula matemática que determina la distancia máximo y mínimo entre el robot y el obstáculo, permitiendo determinar el punto de fuga que hará posible el primer movimiento del robot durante su recorrido. Este proceso se repite

cada vez que el robot se mueve hacia la nueva posición que permite esquivar los obstáculos hasta llegar próximo al destino

Para detección de obstáculos y generar el movimiento del robot se utiliza el algoritmo de líneas de Bresenham [Donald Hearn, M. Pauline B, pp 59-75], se acondiciona para determina la posición del píxel que delimita la frontera del obstáculo –que permite localizar la posición del objeto detectado y a su vez, realizar el movimiento del robot sobre la trayectoria determinada –. El algoritmo se encarga de halla las coordenadas enteras más próximas a la trayectoria real de la recta utilizando solamente aritmética entera. Empieza con una recta cuya pendiente es positiva y menor que uno. Las posiciones de los píxeles a lo largo de la trayectoria de la línea pueden trazarse después tomando etapas unitarias en la dirección *x* y determinado el valor de la coordenada *y* del píxel más cercano a la recta en cada etapa, el procedimiento se repite para cada valor y sentido de la pendiente.

ABSTRACT

The past few decades have witnessed an increased research effort in the area of motion control of autonomous vehicles. A typical motion control problem is trajectory – tracking, which is concerned with the design of control laws that force a vehicle to reach and follow a time parameterized reference (i.e., a geometric path with an associated timing law).

We use an approach to simultaneous localization and mapping to determine a path between two points for a mobile robot, from to star point to the gold. Simply instructing a robot to move and send the robot to the goal via a sequence of way points, called meet points, whose locations are determined by algorithm that guarantees local convergence to the target while at the same time the generated path

The three major motion planning problems are point-to-point, mapping, and coverage. Point-to-point determines a path between two prescribed locations; mapping determines a geometric structure which a robot can use to determine a path between two points; and coverage determines a path that directs the robot to pass over all points in a target region.

The environment in which the robot is operating is a 2-dimensional manifold. It has one starting point P_{start} and one target point P_{finish}. It also has a finite number of static obstacles.

The robot is considered to be a point. It knows the coordinates of its current position P_{start} and the target P_{finish}. It is equipped with limited range sensors, which provide readings 360^{o} around it. Therefore, the sensor range is a disc of radius R centered at P_{start}. The robot can measure the distance to the closest obstacles which are within the sensor range.

Where (x_in, y_in) is the position of the robot, and F_{in_end} is its orientation. The setup corresponds to a path planning problem with incomplete information. The robot doesn't know the location and shapes of the obstacles until they are within the sensor range. Local sensory information is used for calculated the distance between position of the robot and

obstacle as the same time orientation. This model is attractive because many practical robots operate in unknown and changing environments.

With this information the robot moves away from the closest obstacle until it is. At this point the robot has a choice of two directions to move along. Robot chooses the direction that locally decreases its distance to the goal.

The robot then searches for the next meet point, where it then chooses among the out-going edges of the meet point that locally decreases the robot's distance to the goal. This procedure is repeated until the distance to the goal is less than the distance to any of the obstacles. This point is termed the departure point from the departure point, the robot moves in a straight line towards the goal. The access point, the intermediate meet points, and the departure point serve as the way points through which the mobile robot will pass to achieve a goal location.

To made that the robot moves in a straight line towards the goal and with limited range sensors, which provide readings 360^{o} around it. We used Bresenham's [Donald Hearn, M. Pauline B, pp 59-75], line algorithm is an algorithm that determines which points on a 2-dimensional raster should be plotted in order to form a close approximation to a straight line between two given points. It is commonly used to draw lines on a computer screen, as it uses only integer addition, subtraction and bit shifting all of which are very cheap operations in standard computer architectures.

CAPÍTULO 1
INTRODUCCIÓN

1.1 Introducción.

A principios de los años sesenta se introducen en la industria, de modo significativo, los robots manipuladores como un elemento más del proceso productivo. Esta proliferación, motivada por la amplia gama de posibilidades que ofrecía, suscitó el interés de los investigadores para lograr manipuladores más rápidos, precisos y fáciles de programar. La consecuencia directa de este avance originó un nuevo paso en la automatización industrial, y que a su vez flexibilizó la producción con el nacimiento de la noción de célula de fabricación robotizada o manufactura flexible.

Los trabajos desarrollados por robots manipuladores consistían frecuentemente en tareas repetitivas, como la alimentación de las distintas máquinas componentes de la célula de fabricación robotizada. Ello exigía ubicarlas en el interior de un área accesible para el manipulador, caracterizada por la máxima extensión de sus articulaciones, lo cual podría resultar imposible a medida que la célula sufría progresivas ampliaciones. Una solución a este problema se logra al desarrollar un vehículo móvil sobre rieles para proporcionar un transporte eficaz de los materiales entre las distintas zonas de la cadena de producción. De esta forma, aparecen los primeros *vehículos guiados automáticamente* (AGV´s). Una mejora con respecto a su concepción inicial estriba en la sustitución de los rieles como referencia de guiado en la navegación por cables enterrados, reduciéndose, con ello, los costos de instalación.

La posibilidad de estructurar el entorno industrial permite la navegación de vehículos con una capacidad sensorial y de razonamiento mínimas. De este modo, la tarea se estructura en una secuencia de acciones en la que a su término el vehículo supone que ha alcanzado el objetivo para el que está programado. Ante cualquier cambio inesperado en el área de trabajo que afecte el desarrollo normal de la navegación, el sistema de navegación del vehículo se encontrará imposibilitado para ejecutar acciones alternativas que le permitan reanudar su labor. Sin embargo, por sus potenciales aplicaciones fuera del ámbito industrial, donde resulta costoso o imposible estructurar el entorno, se les dotó, en la búsqueda de un vehículo de propósito general apto para desenvolverse en cualquier clase de ambiente, de un mayor grado de inteligencia y percepción. Una definición correcta de robot móvil plantea la capacidad de movimiento sobre entornos no estructurado, de los que se posee un conocimiento incierto, mediante la interpretación de la información proporcionada a través de sus sensores y del estado actual del vehículo.

El uso de robots móviles está justificado en aplicaciones en las que se realizan tareas molestas o arriesgadas para el trabajador humano. Entre ellas, el transporte de material peligroso, las excavaciones mineras, la limpieza industrial o la inspección de plantas nucleares son ejemplos donde un robot móvil puede desarrollar su labor y evita exponer, la salud del trabajador. Otro grupo de aplicaciones donde este tipo de robots complementa la

actuación del operador lo componen las labores de vigilancia, de inspección o asistencia a personas incapacitadas.

El robot móvil autónomo se caracteriza por una conexión inteligente entre las operaciones de percepción y acción, que define su comportamiento y le permite llegar al logro de los objetivos programados sobre entornos con cierta incertidumbre. El grado de autonomía depende en gran medida de la facultad del robot para abstraer el entorno y convertir la información obtenida en órdenes, de tal modo que, aplicadas sobre los actuadores del sistema de locomoción, garantice la realización eficaz de su tarea. De este modo, las dos grandes características que lo alejan de cualquier otro tipo de vehículo se relacionan a continuación [Lozano – Pérez, 1990]:

- Percepción: Determina la relación del robot con su entorno de trabajo, mediante el uso de los sensores de a bordo.
- Razonamiento: Determina las acciones que se han de realizar en cada momento, según el estado del robot y su entorno, para alcanzar las metas asignadas.

De este modo, la capacidad de razonamiento del robot autónomo móvil se traduce en la planificación de unas trayectorias seguras que le permitan realización de los objetivos encomendados. La ejecución de la tarea debe realizarla en bucle cerrado para adaptarse a la navegación sobre entornos no estructurados.

Por tanto, resulta necesario el uso de un planificador con capacidad de análisis geométrico que conozca el estado del entorno y del robot junto a sus características cinemáticas y dinámicas. De este modo, se realiza la transformación de los datos suministrados por la percepción en referencias de control adecuadas que no traspasen ninguna de las limitaciones físicas del vehículo, y que definan una trayectoria libre de obstáculos que garantice el logro de las metas determinadas en la tarea.

1.2 Alcance del trabajo de tesis.

La tarea que el robot debe realizar se define mediante el uso de un mapa del entorno y un conjunto de puntos objetivos que se desean alcanzar. Con estos datos el planificador construye una trayectoria con buenas propiedades para ser utilizada por el seguidor y que minimiza el esfuerzo de control sobre el vehículo. La construcción de la trayectoria se lleva a cabo en un primer paso de planificación espacial. Así, la contribución de esta tesis se refieren a:

Desarrollo de un método de generación de caminos locales con buenas propiedades especializado en el sorteo de obstáculos.

La capacidad de un robot autónomo móvil para seguir un camino depende en gran medida de las características del mismo. Por tanto, resulta necesario construirlo de forma adecuada para minimizar los errores en la navegación. El método propuesto se basa en el desplazamiento por medio de trayectorias rectilíneas.

1.3 Marco de realización.

La realización de este trabajo de tesis ha estado determinada por las actividades de investigación que realizan en el **Departamento de Micro-mecánica y Robótica del Centro de Ciencias Aplicadas y Desarrollo Tecnológico de la Universidad Nacional Autónoma de México**, en la línea de los robots móviles y, en particular, en el diseño y construcción del Robot Autónomo Móvil.

El desarrollo de todos los algoritmos, obtenidos como resultado del trabajo de investigación y desarrollo, han sido realizados mediante el uso del software Borland Builder [Francisco Charte O., Builder 5] con programación C++ [Walter Savitch], bajo el sistema operativo Windows

1.4 Estructura de la tesis.

La tesis se encuentra dividida en siete capítulos, conclusiones, un apéndice con el listado del programa y las referencias bibliográficas. La organización de la tesis responde a la resolución progresiva del problema de planificación de trayectorias, y su contenido se describe en los siguientes párrafos del presente apartado:

Capítulo 1 **"Introducción",** capítulo que ha sido descrito en la sección anterior.

Capítulo 2, "**Navegación en robots móviles**", con fundamento en la investigación realizada se introduce los conceptos fundamentales que definen la realización de trabajos por parte de los robots móviles. Así, en primer lugar se presenta la estructura general de un navegador estratégico, para, con posterioridad, ocuparse de la formalización del tema central de la tesis: la planificación de trayectorias en dos dimensiones. De este modo, en segundo lugar, se concretan los diferentes aspectos derivados de esta última cuestión. La planificación de una ruta a través de un entorno con obstáculos, el ajuste de los puntos intermedios para la especificación del camino y la conversión de éste en trayectoria recta ó curva, para asegurar la navegación segura del robot móvil autónomo, son los aspectos introducidos en este capítulo.

Capítulo 3, "**Gráficas por computadora**", se aborda de forma general los conocimientos que permiten entender que las computadoras se han convertido en una herramienta poderosa para la producción de ilustraciones animadas y a su vez un factor importante en el control de robots móviles, ya que forma parte fundamental del sistema de a bordo del robot móvil. Prácticamente son pocas las áreas del conocimiento científico y tecnológico en las cuales no puedan utilizarse los despliegues gráficos; así como, lo requerimientos necesarios para su adecuada utilización.

Capítulo 4, "**Sistemas de gráficas**", los sistemas de computación pueden adaptarse a aplicaciones gráficas en varias formas y conforme a los recursos de software y hardware de que se dispone. Cualquier computadora de uso general puede utilizarse para hacer gráficas de caracteres utilizando elementos del conjunto de caracteres del sistema con objeto de formar modelos o patrones. Para aplicaciones más complejas, en especial para aplicaciones

científicas y tecnología, se dispone de una gran variedad de paquetes de software y dispositivos de hardware.

Capítulo 5, "**Propiedades de las primitivas de salida**", debido a que el robot móvil dispone de una computadora de abordo; por lo que, se tendrán que diseñar los programas de control. Los procedimientos que despliegan primitivas de salida dirigen un dispositivo de salida para producir estructuras geométricas especificadas en la localidad designada. Dichos algoritmos toman coordenadas de la entrada y realizan despliegues para construir una figura geométrica en un dispositivo de salida seleccionado, para este trabajo el dispositivo de salida será el monitor de la PC.

Capítulo 6, "**Sistema de control robot movible**", el problema de control de un robot autónomo viene resolviéndose utilizando un control local de bajo nivel que proporciona competencias elementales como: evitar un obstáculo, seguir el contorno de la pared, entrar en una puerta, evitar zonas de inestabilidad. Se desarrolla una aplicación utilizando control local que permite evitar obstáculos; por medio de la generación de trayectorias que permiten realizar una navegación segura. Se describe la forma de cómo generar la trayectoria por medio de tres propuestas diferentes. Cabe indicar que la tercera propuesta es la esencia de este trabajo

Capítulo 7, "**Experimentos y resultados obtenidos"**, en este apartado se realizar varios experimentos, variando las geometrías de los obstáculos, tamaño, ubicación y cantidad de obstáculos, con la finalidad de conocer el comportamiento del robot bajo diferentes condiciones de operación y a su vez observar las trayectoria generadas para diferente puntos coordenados de inicio y destino. Es importante mencionar que el algoritmo se va modificando conforme a los resultados obtenidos y así lograr el cumplimiento del objetivo propuesto. Con los resultados obtenidos se estima que son 64 nodos cada uno de los cuales corresponde a una de las posibles direcciones de movimiento

Conclusiones

Referencias bibliográficas

Apéndice **A, listado de programa**

CAPÍTULO 2
NAVEGACIÓN EN ROBOT MÓVILES

2.1 Introducción

Se define navegación como la metodología (o arte) que permite guiar el curso de un robot móvil a través de un entorno con obstáculos. Existen diversos esquemas, pero todos ellos poseen en común el afán por llevar el vehículo a su destino de forma segura. La capacidad de reacción ante situaciones inesperadas debe ser la principal cualidad para desenvolverse, de modo eficaz, en entornos no estructurados.

Las tareas involucradas en la navegación de un robot móvil son: la percepción del entorno a través de sus sensores, de modo que le permita crear una abstracción del mundo; la planificación de una trayectoria libre de obstáculos, para alcanzar el punto destino seleccionado; y el guiado del vehículo a través de la referencia construida. De forma simultánea, el vehículo puede interaccionar con ciertos elementos del entorno. Así, se define el concepto de operación como la programación de las herramientas de a bordo que le permiten realizar la tarea especificada. Un ejemplo de esta última noción es el transporte automático de materiales y herramientas dentro de una célula de manufactura flexible (FMS), lo que implica no sólo el movimiento físico de dichos elementos desde la estación de entrada de material hasta la máquina que lo requiera, sino que además pueda realizar operaciones como el cambio automático de la herramienta o la descarga automática del material en la máquina que lo haya solicitado [Newman y Kempf, 1985].

2.2 Concepto de misión, navegación y operación.

El robot móvil se caracteriza por realizar una serie de desplazamientos (navegación) y por llevar a cabo una interacción con distintos elementos de su entorno de trabajo (operación), que implican el cumplimiento de una serie de objetivos impuestos según cierta especificación. Así, formalmente el concepto de misión en el ámbito de los robots móviles [Levi, 1987] se define como la realización conjunta de una serie de objetivos de navegación y operación.

En consecuencia con las definiciones del párrafo anterior, el robot móvil debe poseer una arquitectura que coordine los distintos elementos de a bordo (sistema sensorial, control de movimiento y operación) de forma correcta y eficaz para la realización de una misión. El diseño de esta arquitectura depende mucho de su aplicación en particular, pero un esquema básico de los principales módulos que la componen y la interacción que existe entre los mismos es el presentado en la figura 2.1.

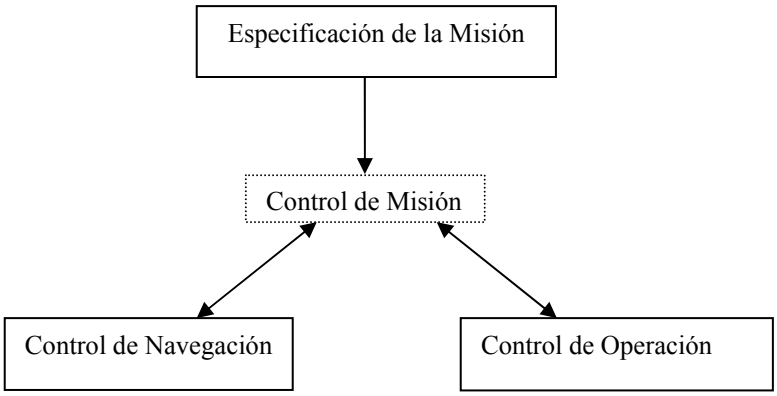


Figura 2.1. Esquema básico de la arquitectura necesaria en un robot móvil para realizar una misión.

En la figura 2.1, se presenta un módulo de *control de misión* [Ollero, 1994] dedicado a coordinar al controlador de desplazamientos (*control de navegación*) con el controlador del elemento que interacciona con el entorno de trabajo (*control de operación*). Esta coordinación debe efectuarse de forma perfecta para cumplir los objetivos impuestos por la misión, definida de acuerdo con ciertas especificaciones de entrada. Formalmente, el control de misión debe analizar el problema y encontrar una estrategia para resolverlo, de suerte que el resultado de este análisis será un plan de navegación y otro de operación, los cuales se entregan a los módulos correspondientes de la parte inferior de la figura 2.1.

2.3 Esquemas de navegación en robots móviles.

Realizar una tarea de navegación para un robot móvil significa recorrer un camino que lo conduzca desde una posición inicial hasta otra final, pasando por ciertas posiciones intermedias. El problema de la navegación se divide en las siguientes cuatro etapas:

- *Percepción del mundo*: Mediante el uso de sensores externos, creación de un mapa o modelo del entorno donde se desarrollará la tarea de navegación [González, 1993]

- *Planificación de la ruta*: Crea una secuencia ordenada de objetivos o submetas que deben ser alcanzadas por el vehículo. Esta secuencia se calcula utilizando el modelo o mapa de entorno, la descripción de la tarea que debe realizar y algún tipo de procedimiento estratégico.

- *Generación del camino*1: En primer lugar define una función continua que interpola la secuencia de objetivos construida por el planificador. Posteriormente procede a la discretización de la misma información a fin de generar el camino.

- *Seguimiento del camino*: Efectúa el desplazamiento del vehículo, según el camino generado mediante el adecuado control de los actuadores del vehículo [Martínez, 1994].

Estas tareas pueden llevarse a cabo de forma separada, aunque en el orden especificado. La interrelación existente entre cada una de estas tareas conforma la estructura de control de navegación básica en un robot móvil [Shin – Singh, 1990] y se muestra en la figura 2.2.

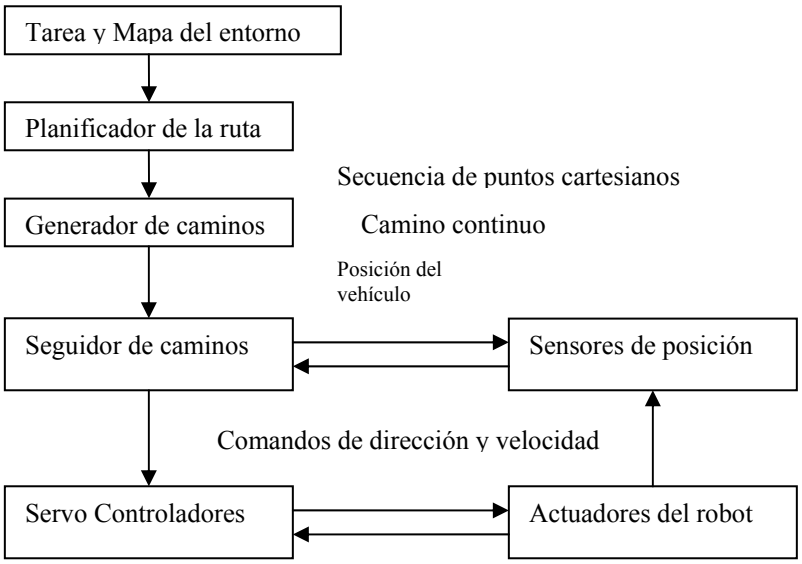


Figura 2.2. Estructura de control de navegación básica para un robot móvil.

En el esquema de la figura 2.2 se parte de un mapa de entorno y de las especificaciones de la tarea de navegación. De estos datos se realiza la planificación de un conjunto de objetivos representados como una secuencia de puntos cartesianos dispersos que definen la ruta. Dicho conjunto cumple los requisitos de la tarea impuesta asegurándose de que la ruta asociada está libre de obstáculos. Mediante el uso del generador del camino se construye la referencia que utilizará el seguidor de caminos para generar los comandos de direccionamiento y velocidad que actuarán sobre los servo-controladores del vehículo. Por último, mediante el uso de los sensores internos del vehículo (sensores de posición) en conjunción con técnicas odométricas (medir la velocidad al moverse), se produce una estimación de la posición actual [Cox, 1991], la cual será realimentada al seguidor de caminos.

La complejidad del sistema necesario para desarrollar esta tarea depende principalmente del conocimiento que se posee del entorno de trabajo. Así, la figura 2.2 considera que se cuenta con un mapa del entorno que responde de forma fiel a la realidad.

Mediante el uso adecuado del mismo se puede construir un camino que cumpla los requisitos impuestos por la tarea de navegación, sin que el vehículo colisione con algún elemento del entorno. Sin embargo, es posible que el modelo del entorno del que dispone el robot adolezca de ciertas imperfecciones al omitir algunos detalles del mismo. El esquema presentado en la figura 2.2 resulta ineficaz, al no asegurar la construcción de un camino libre de obstáculos. Por ello se necesita introducir en la estructura de control básica nuevos elementos que mejoren este defecto. Un esquema de navegador utilizado en aplicaciones, donde la información acerca del entorno de trabajo varía desde un perfecto conocimiento del mismo hasta poseer un cierto grado de incertidumbre, es el mostrado en la figura 2.3.

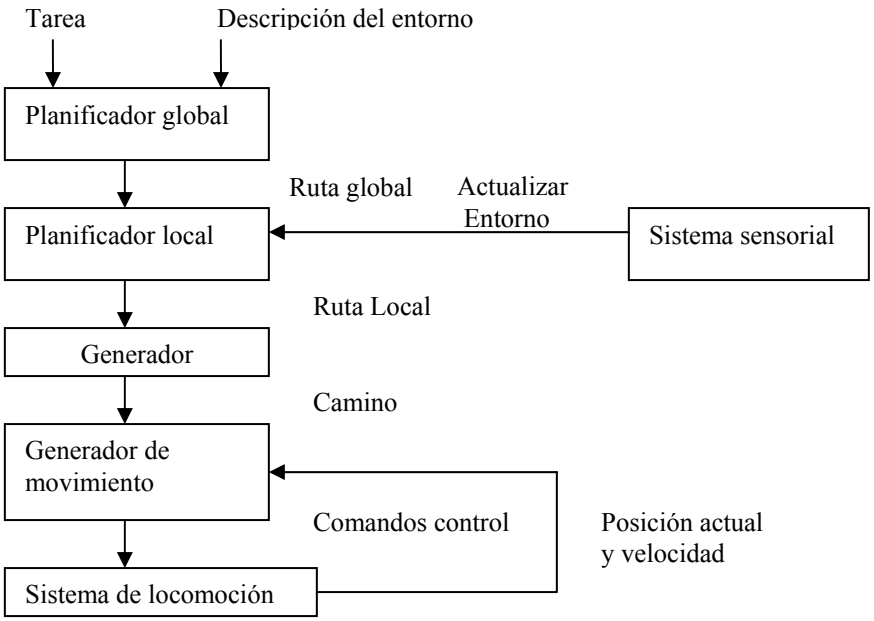


Figura 2.3. Navegador implantado en el robot móvil Blanche de AT&T [Nelson – Cox, 1990]

El esquema presentado contiene en líneas generales, el funcionamiento de la estructura de navegación básica. Lo novedoso reside en el desdoblamiento de la tarea de planificación en dos subtareas: planificación global y local. La primera de estas subtareas es análoga al módulo de planificación de la figura 2.2 y construye una ruta sobre la cual se puede definir un camino libre de obstáculos según la información que a priori se posee del entorno. Si la descripción del entorno introducida fuese perfecta, la ruta calculada sería de forma directa la entrada de la tarea *generador*. Sin embargo, al no serlo, puede dar lugar a la construcción de un camino que no esté libre de obstáculos, con el consiguiente peligro de que el vehículo impacte con algún elemento del entorno. La tarea de *planificación local* recibe información del *sistema sensorial* sobre el entorno local del robot, según el radio de alcance de los sensores externos de a bordo. Mediante el análisis de estos datos actualiza el modelo preliminar del entorno y decide si se precisa replanificar la ruta local del robot. La clave del esquema presentado en la figura 2.3 para adaptarse a diversos entornos, aunque no se posea un conocimiento exhaustivo del mismo, reside en la distinción efectuada entre planificación global y local. Ambos conceptos se pueden definir con mayor precisión:

- **Planificación global:** Construir o planificar la ruta que lleve al robot a cada una de las submetas determinadas por el control de misión, según las especificaciones del problema que debe resolverse. Esta planificación es una aproximación al camino final que se va a seguir, ya que en la realización de esta acción no se consideran los detalles del entorno local al vehículo.

- **Planificación local:** Resolver las obstrucciones sobre la ruta global en el entorno local al robot para determinar la ruta real que será seguida. El modelo del entorno local se construye mediante la fusión de la información proporcionada por los sensores externos del robot móvil.

La construcción de la ruta global puede realizarse antes de que el vehículo comience a ejecutar la tarea, mientras que la planificación local se lleva a cabo en tiempo de ejecución. En el caso de realizar una navegación sobre entornos totalmente conocidos es obvio que resulta innecesario proceder a una planificación local, pero a medida que disminuye el conocimiento de la zona por la cual el robot móvil realiza su tarea, aumenta la relevancia de la misma.

En aplicaciones de navegación en exteriores o campo a través, del conocimiento que se posee del entorno es pobre y por tanto se necesita hacer un uso intensivo de la planificación local. En el robot móvil la navegación está confiada totalmente al planificador local, de suerte que el camino que debe seguir se construye de forma dinámica a medida que se navega. El esquema empleado recurre a un uso más intenso del sistema sensorial, y se responsabiliza de la coordinación de la percepción, planificación y control del vehículo para guiarlo por el camino especificado, mientras verifica el entorno, y realizar el sorteo de obstáculos [Martínez J.L. 1994]. La interacción de cada uno de los módulos en este esquema de navegación local queda representada en la figura 2.4.

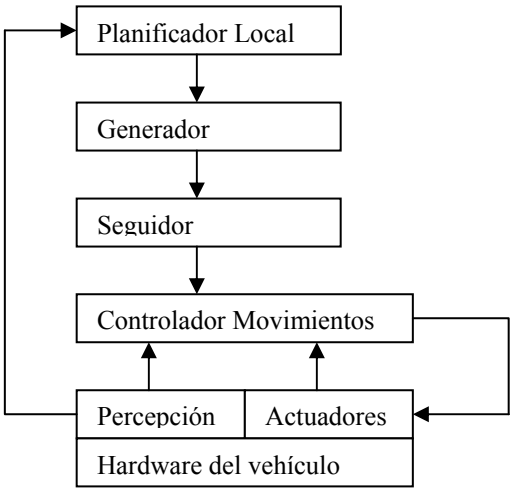


Figura 2.4. Navegador local implantado en el Navlab II de CMU.

El funcionamiento de este navegador local está basado en la realización de un ciclo de construcción del mapa local del entorno inmediato al robot móvil, la elección de una ruta segura por la cual puede pasar el vehículo, de acuerdo con la información suministrada, para, a continuación construir el camino, y por último, realizar el seguimiento. La iteración de este ciclo ocurre cada vez que el *control de movimientos* termina de seguir el camino actual, pasándose en ese momento a la construcción del próximo mapa local de entorno. La diferencia entre los esquemas de navegador presentados radica en la adaptación que debe poseer el robot móvil para moverse por su entorno de trabajo según el conocimiento de que disponga sobre la estructura del mismo, dándose mayor ponderación a la planificación global o local. Sin embargo, mantienen un nexo común en la realización de forma secuencial y continua las operaciones de percepción, planificación-generación y seguimiento. Este grupo de acciones permite el desarrollo de la navegación minimizando cierto índice de coste, como puede ser la distancia recorrida o el consumo energético del

vehículo. Se habla en este caso de una descomposición jerárquica en módulos funcionales que encadenados en forma de ciclo realizan la denominada navegación estratégica. Además, este tipo de navegación se caracteriza por la necesidad de conocer con el mínimo error posible la posición actual del vehículo, ya que la realimentación de esta información es la base para el cálculo de la próxima acción de control. Mediante el uso de la odometría del vehículo se puede realizar esta acción, pero debido a la naturaleza del método y a las características de los sensores utilizados, la estimación efectuada se ve afectada por errores acumulativos [Watanabe y Yuta, 1990]. Cuando dichos errores alcanzan niveles indeseables se hace necesario eliminarlos mediante la utilización de algoritmos de estimación de la posición basados en referencias externas. La navegación estratégica tiene sus limitaciones en entornos dinámicos no conocidos, ya que requiere un completo conocimiento de la dinámica de los posibles obstáculos móviles, además de una adecuada actualización del mapa de entorno.

La filosofía alternativa a la navegación estratégica consiste en el uso intensivo de sensores de bajo coste (transductores ultrasónicos, sensores infrarojos, sensores táctiles, etc.) con el fin de reaccionar dinámicamente ante el entorno, con lo cual pierde relevancia el concepto de planificación y seguimiento de caminos. Esta tendencia se ha basado en la *subsumption architecture* [Brooks, 1986], una arquitectura descompuesta en módulos especializados en realizar tareas individuales, denominados comportamientos. Se trata de una descomposición vertical del problema de navegación que se comporta de forma eficiente en entornos dinámicos donde se posee un conocimiento impreciso del mismo. Este esquema aparece mostrado en la figura 2.5.

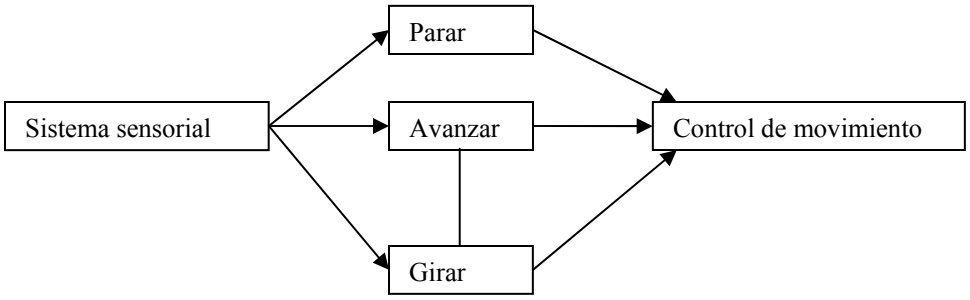


Figura 2.5. Navegación reactiva.

En cada intervalo de navegación el sistema sensorial, según la información extraída del entorno local del robot, activa uno o varios comportamientos simples que suman sus actuaciones, de suerte que el comportamiento final resulta una mezcla de los simples activados.

La navegación reactiva basada en comportamientos ha sido implantada en múltiples aplicaciones, entre las que predominan los comportamientos de supervivencia, dando lugar a robots errantes que se mueven con libertad por entornos desconocidos e incluso dinámicos, sin colisionar con los obstáculos, pero que raramente obedecen a un plan establecido, imprescindible en misiones reales [Arkin, 1987; Anderson y Donath, 1988].

2.4 Planificación de la ruta.

Una primera definición del problema de la planificación, ya sea global o local, consiste en encontrar una ruta segura capaz de llevar al vehículo desde la posición actual hasta la especificada meta o de destino. El concepto de ruta segura implica el cálculo de un camino al menos continuo en posición, que sea libre de obstáculos. En virtud de esta ruta, el generador construirá las referencias que se le entregan al control de movimientos. Por ello, en la especificación de esta ruta se obvian las características cinemáticas y dinámicas del vehículo, ya que el cómputo de una referencia adecuada que cumpla con estos atributos es tarea del generador de caminos. Por tanto, la ruta tan sólo asegurar continuidad en posición, supone que únicamente los robots móviles omnidireccionales puedan seguir una referencia de tales características.

2.4.1 Formalización del problema de la planificación.

El entorno de trabajo en el cual un robot móvil realizará su tarea, puede considerarse como un conjunto de configuraciones en las cuales puede encontrarse el robot en un determinado instante de tiempo. Entre ellas existirá un subconjunto inalcanzable, al estar ocupado por los obstáculos del entorno [Lozano – Pérez, 1990]. Se define una configuración q de un robot como un vector cuyas componentes proporcionan información completa sobre el estado actual del mismo. Un robot es un objeto rígido al cual se le puede asociar un sistema de coordenadas móvil. La localización del vehículo en un determinado instante de tiempo queda definida por la relación existente entre el sistema de coordenadas global F_g, en virtud del cual está definido todo el entorno de trabajo y su sistema de coordenadas locales asociado F_r (Figura 2.6.).

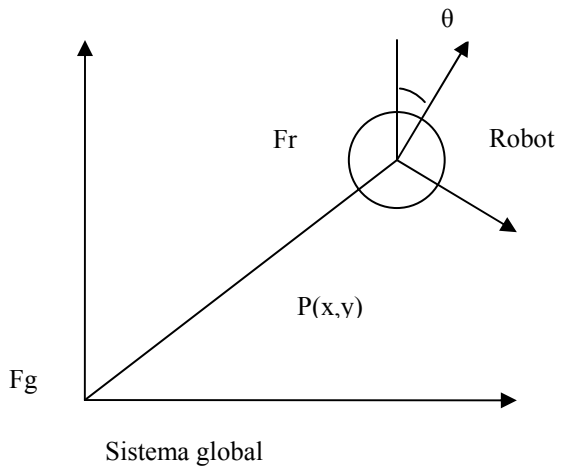


Figura 2.6. Sistema de coordenadas global, y sistema local asociado al robot.

El vector que proporciona información sobre el estado actual del robot viene dado, en principio, por dos componentes: la posición p y la orientación θ. Por tanto, se puede definir configuración como [Martínez J.L. 1994, pp 29-30]:

$$q = (p, \theta) = (x, y, \theta) \qquad (2.1)$$

Se denomina espacio de configuraciones *C* del robot *R* a todas las configuraciones *q* que puede tomar el robot en su entorno de trabajo. El subconjunto de *C* ocupado por el robot *R* cuando este se encuentra en *q*, se denota por *R(q)*. Si el robot se modela de forma circular con radio ρ, *R(q)* se define como:

$$R(q) = \{ q_i \in C \,/\, \| q, q_i \|, \leq \rho \} \qquad (2.2)$$

En el caso de un robot puntual, en la expresión (2.2), ρ es nulo, con lo cual se cumple:

$$R(q) = \{q\} \qquad (2.3)$$

En el espacio de trabajo, donde el robot realizará su tarea, se encuentran distribuidos una serie de obstáculos definidos como un conjunto de objetos rígidos *B* y que se encuentran distribuidos por el espacio de configuraciones *C*.

$$B = \{ b_1, b_2, \ldots, b_q \} \qquad (2.4)$$

El conjunto de configuraciones del espacio *C* ocupadas por un obstáculo se define por $b_i(q)$, de tal forma que el subconjunto de configuraciones de *C*, que especifican el espacio libre de obstáculos viene dado por:

$$C_l = \{ p \in C \,/\, R(q) \cap (\cup_{i=1} b_i(q)) = \varnothing \} \qquad (2.5)$$

Según esta metodología, el problema de la planificación, tal y como se ha definido, queda transformado en la búsqueda de una sucesión de posturas *q* tal que la primera de ellas sea la postura actual del robot q_a y la última de esta sucesión la postura objetivo q_f. Todas las posturas de la serie deben pertenecer al subconjunto C_l definido en la expresión (2.5). Es decir, una ruta Q_r que conecta la postura inicial q_a con la final q_f es:

$$Q_r = \{ q_a, \ldots, q_f \,/\, q_i \in C_l \} \qquad (2.6)$$

La especificación de este conjunto Q_r, implica la construcción de una función ruta definida de la siguiente forma:

$$\tau : [0,1] \rightarrow C_l \qquad (2.7)$$

tal que:

$$\tau(0) = q_a \qquad \tau(1) = q_f \qquad (2.8)$$

Además de la restricción mostrada en (2.7), se le exige a la función τ, teniendo en cuenta la suposición de robot un omnidireccional, la continuidad. Este concepto se refleja en la siguiente expresión:

$$\lim_{s \to s_0} \| \tau(s), \tau(s_0) \| = 0 \qquad (2.9)$$

2.4.2 Métodos clásicos de planificación.

Todos ellos se fundamentan en una primera fase de construcción de algún tipo de grafo sobre el espacio libre, según la información poseída del entorno, para posteriormente emplear un algoritmo de búsqueda en grafos que encuentra el camino óptimo según cierta función de coste.

2.4.2.1 Planificación basada en grafos de visibilidad.

Los grafos de visibilidad proporcionan un enfoque geométrico útil para resolver el problema de la planificación [Nilsson, 1969]. Supone un entorno bidimensional en el cual los obstáculos están modelados mediante polígonos. Para la generación del grafo este método introduce el concepto de *visibilidad*, según el cual define dos puntos del entorno como *visibles* si y solo si se pueden unir mediante un segmento rectilíneo que no cruce ningún obstáculo (si dicho segmento resulta tangencial a algún obstáculo se consideran los puntos afectados como visibles). En otras palabras, el segmento definido debe yacer en el espacio libre del entorno *Cl*.

Así, si se considera como nodos del grafo de visibilidad la posición inicial, la final y todos los vértices de los obstáculos del entorno, el grafo resulta de la unión mediante arcos de todos aquellos nodos que sean visibles.

En la figura 2.7 se muestra el grafo de visibilidad construido con los obstáculos poligonales existentes en el entorno y las configuraciones inicial q_a y final q_f.

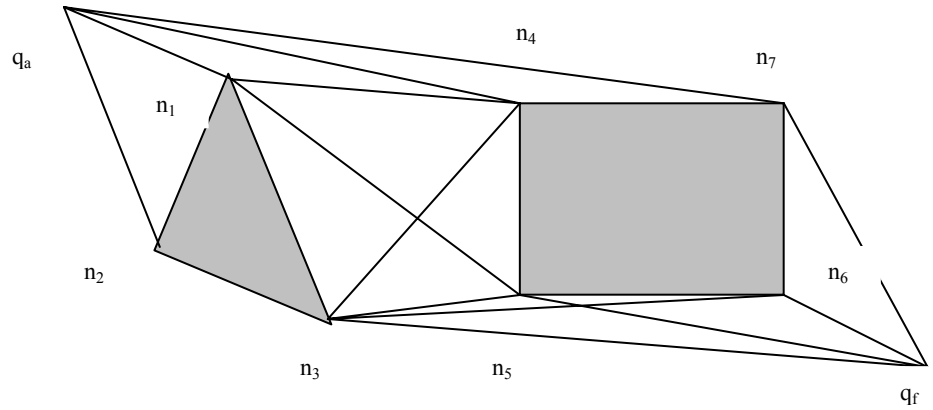


Figura 2.7. Grafo de visibilidad en un entorno de dos obstáculos.

En el grafo mostrado (Figura 2.7), se puede observar cómo sólo están unidos los nodos directamente visibles, de tal forma que el conjunto de arcos estará formado por las aristas

de los obstáculos, más el resto de líneas que relacionan los vértices de los diferentes polígonos.

Mediante un algoritmo de búsqueda en grafos se elige la ruta que una la configuración inicial con la final minimizando alguna función de coste. La ruta que cumple el objetivo de la navegación queda definida como una sucesión de segmentos que siguen los requisitos especificados (Figura 2.8).

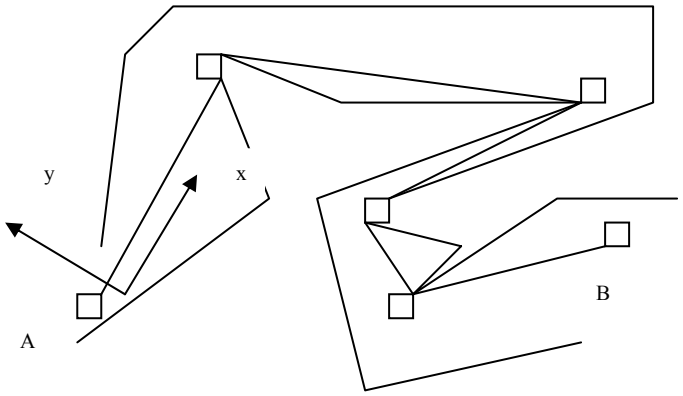


Figura 2.8. Planificación con el espacio libre de obstáculos modelado mediante cadenas.

2.4.2.2 Planificación basada en diagramas de Voronoi.

Al contrario que los métodos basados en grafos de visibilidad, la planificación basada en diagramas de Voronoi sitúa la ruta lo más alejada posible de los obstáculos. Con ello elimina el problema presentado por los grafos de visibilidad de construir rutas semi-libres de obstáculos.

Los diagramas de Voronoi se definen como una proyección del espacio libre del entorno en una red de curvas unidimensionales yacientes en dicho espacio libre. Formalmente se definen como una retracción con preservación de la continuidad. Si el conjunto C_l define las posiciones libres de obstáculos de un entorno, la función retracción RT construye un subconjunto C_v continuo de C_l [Janich, 1984].

$$RT(q): C_l \rightarrow C_v / C_v \subset C_l \qquad (2.10)$$

De esta forma, se dice que existe un camino desde una configuración inicial q_a hasta otra final q_f, supuestas ambas libres de obstáculos, si y solo si existe una curva continua desde $RT(q_a)$ hasta $RT(q_f)$.

La idea fundamental es ampliar al máximo la distancia entre el camino del robot y los obstáculos. Por ello, el diagrama de Voronoi resulta el lugar geométrico de las

configuraciones que se encuentran a igual distancia de los dos obstáculos más próximos del entorno. El diagrama estará formado por dos tipos de segmentos: rectilíneos y parabólicos.

La elección de la modalidad de segmento corresponde con la clase de elementos de los obstáculos más cercanos que se encuentren enfrentados entre sí. De esta forma, el lugar geométrico de las configuraciones que se hallan a igual distancia de dos aristas de dos obstáculos diferentes es una línea recta, mientras que en el caso de tratarse de un vértice y una arista resulta una parábola.

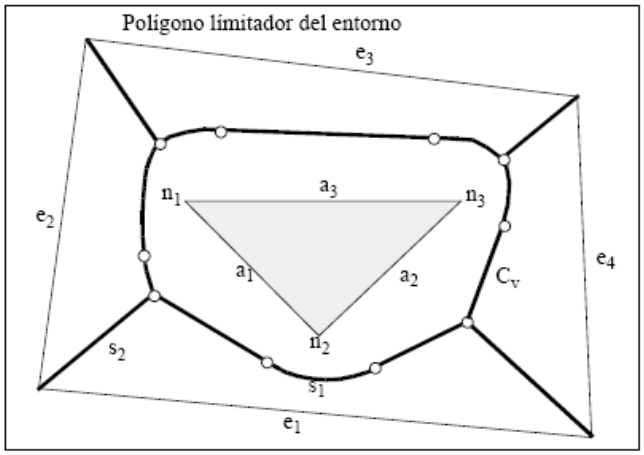


Figura 2.9. Retracción del espacio libre en un diagrama de Voronoi.

En la figura 2.9 se muestra un entorno delimitado por un polígono de aristas *{e1,e2,e3,e4}* y un obstáculo triangular de vértices *{n1,n2,n3}* y aristas *{a1,a2,a3}*. La retracción del espacio libre en una red continua de curvas es el diagrama de Voronoi *Cv*, representado mediante las líneas de trazo grueso. Los dos tipos de segmento utilizados en la construcción del diagrama pueden distinguirse en la figura, así, el segmento *s1* es el lugar geométrico de los puntos equidistantes entre la arista *e1*, y el vértice *n2*. Por otra parte, puede observarse como el segmento rectilíneo *s2* cumple la misma condición pero con respecto a las aristas *e1* y *e2*.
Dado una configuración *q* no perteneciente a *Cv*, existe un único punto *p* más cercano perteneciente a un vértice o arista de un obstáculo. La función *RT(q)* se define como el primer corte con *Cv* de la línea que une *p* con *q* (Figura 2.10.).

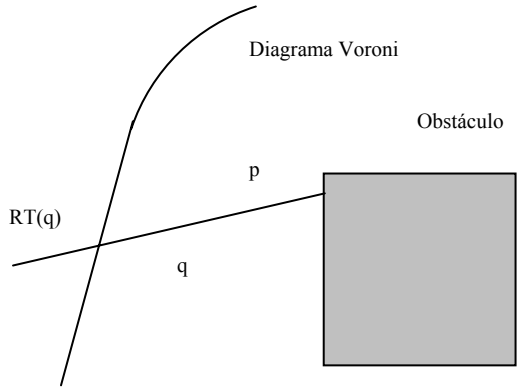


Figura 2.10. Imagen de una configuración q en el diagrama de Voronoi.

2.4.2.3 Planificación basada en modelado del espacio libre.

Se aplica a arquetipos de entornos con obstáculos poligonales, y la planificación en este caso se realiza mediante el modelado del espacio libre. Esta acción se lleva a cabo por los denominados cilindros rectilíneos generalizados (*CRG*). Al igual que los diagramas de Voronoi, se pretende que el vehículo navegue lo más alejado de los obstáculos. De forma que la ruta que lleve al robot desde una configuración inicial hasta otra final estará compuesta por una serie de puntos interconectados, de tal modo que la configuración de partida se encuentre en el primer punto de la sucesión y la final en el último.

La construcción se realiza a partir de las aristas de los distintos obstáculos que se encuentran en el entorno. Para que un par de aristas $_1a_i$ y $_2a_j$ pertenecientes a los obstáculos b_1 y b_2 respectivamente puedan formar un cilindro generalizado, deben cumplir las siguientes condiciones:

i. La arista $_1a_i$ está contenida en una recta que divide al plano en dos regiones. La arista $_2a_j$ debe yacer por completo en la región opuesta en la que se encuentra situada b_1. Este criterio es simétrico.
ii. El producto escalar de los vectores normales con dirección hacia el exterior del obstáculo que contiene cada arista debe resultar negativo. Si se cumplen estas condiciones significa que ambas aristas se encuentran enfrentadas, y por tanto se puede construir un *CRG* con ellas (Figura 2.11.).

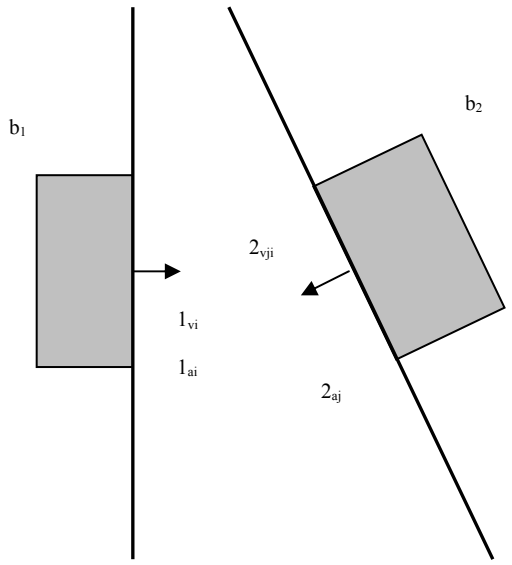


Figura 2.11. Condiciones que deben cumplir dos aristas para construir un *CRG*.

Una vez detectadas dos aristas, el siguiente paso será construirlo.

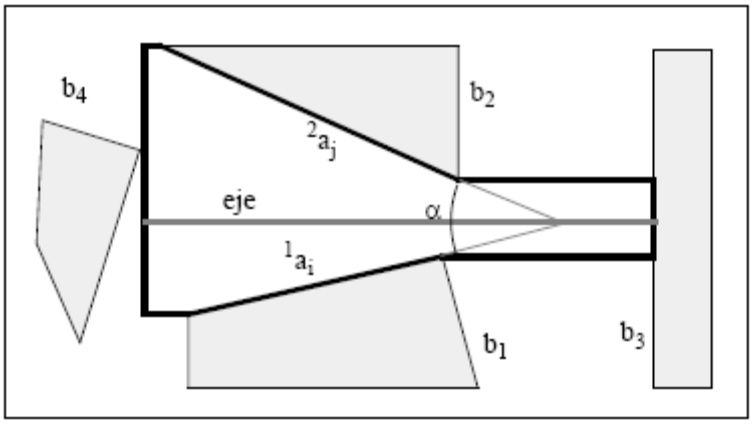


Figura 2.12. Construcción de un *CRG*.

El primer paso es el cálculo del *eje* del *CRG*, el cual se define como la bisectriz del ángulo α formado por el corte de las rectas que contienen las aristas $_1a_i$ y $_2a_j$ que cumplen las condiciones i) y ii) expuestas más arriba. Por ambos lados de dichas aristas se construyen segmentos rectilíneos paralelos al *eje*, con origen en los vértices de las aristas implicadas y con extremo señalado por la proyección del primer obstáculo que corta el *eje*.

Repitiendo este proceso, se construye una red *CRG* en el entorno del robot que modela el espacio libre del mismo. El robot navegará por el eje del cilindro, en el cual se encuentran anotadas para cada punto en el rango de orientaciones admisibles. El paso de un *CRG* a otro se produce siempre y cuando sus ejes intersecten y la intersección del rango de orientaciones admisibles en el punto de corte de ambos *ejes* no sea nulo.

2.4.2.4 Planificación basada en la descomposición en celdas.

Este tipo de métodos se fundamenta en una descomposición en celdas del espacio libre [Thorpe, 1984]. Así, la búsqueda de una ruta desde una postura inicial q_a hasta otra final q_f, consiste en encontrar una sucesión de celdas que no presente discontinuidades, tal que la primera de ellas contenga a q_a y la última a q_f. Al contrario que los métodos expuestos a lo largo de este apartado, no encuentra una serie de segmentos que modele la ruta, sino una sucesión de celdas; por ello, se hace necesario un segundo paso de construcción de un *grafo de conectividad*, encargado de definir la ruta.

Para la planificación según el método de descomposición en celdas, se precisa la resolución de dos problemas: la descomposición del espacio libre en celdas y la construcción de un grafo de conectividad. El primero de ellos implica construir unas celdas eje α con determinada forma geométrica tal que resulte fácil de calcular un camino entre dos configuraciones distintas pertenecientes a la celda, y la comprobación para averiguar si dos celdas son adyacentes debe disfrutar de la mayor simpleza posible. Aparte de estas características, la descomposición global del espacio libre implica que no deben existir solapamientos entre celdas y que la unión de todas ellas corresponde exactamente al espacio libre.

El grafo de conectividad es un grafo no dirigido, y su construcción está asociada a la descomposición en celdas efectuada en el paso anterior, del tal forma, que los nodos van a ser cada una de las celdas, existiendo un arco entre dos celdas si y solo si son adyacentes (Figura 2.13.).

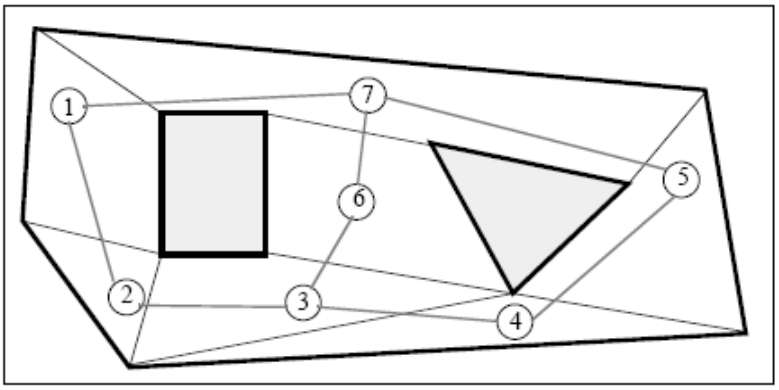


Figura 2.13. Descomposición en celdas y grafo de conectividad.

Una vez especificado el grafo de conectividad, sólo queda emplear un algoritmo de búsqueda en grafos, para la detección de la celda que contiene la postura a la cual se desea llegar, tomando como partida la que contiene la postura inicial.

Los distintos métodos basados en este principio, se distinguen por la forma en la cual realizan la descomposición en celdas y como se construye el grafo de conectividad. El método más sencillo de descomposición del espacio libre del entorno en celdas resulta la *descomposición trapezoide* [Latombe, 1991]. Este método se basa en la construcción de segmentos rectilíneos paralelos al eje *Y* del sistema global F_g a partir de los vértices de cada uno de los elementos del entorno. El final del segmento queda delimitado por el primer corte de la línea con un elemento del entorno. Esta descomposición es la mostrada en la figura 2.14.

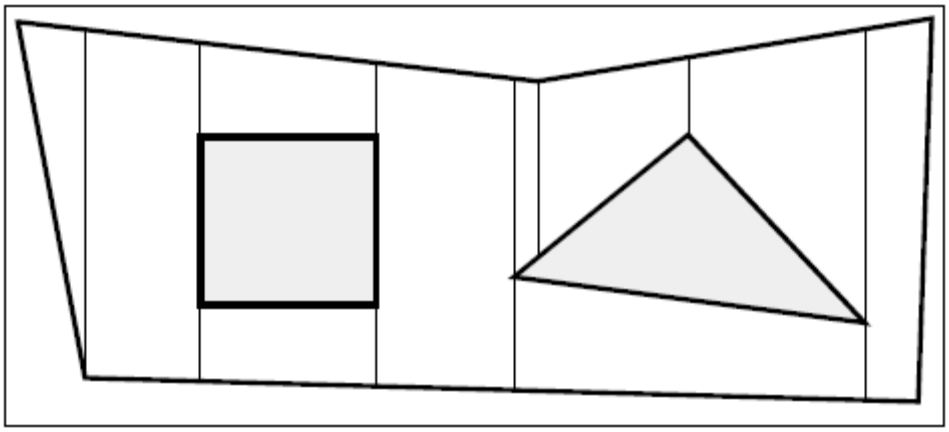


Figura 2.14. Descomposición trapezoidal del espacio libre.

El grafo de conectividad se construye por medio de la unión de los puntos medios de los segmentos verticales definidos (Figura 2.15.)

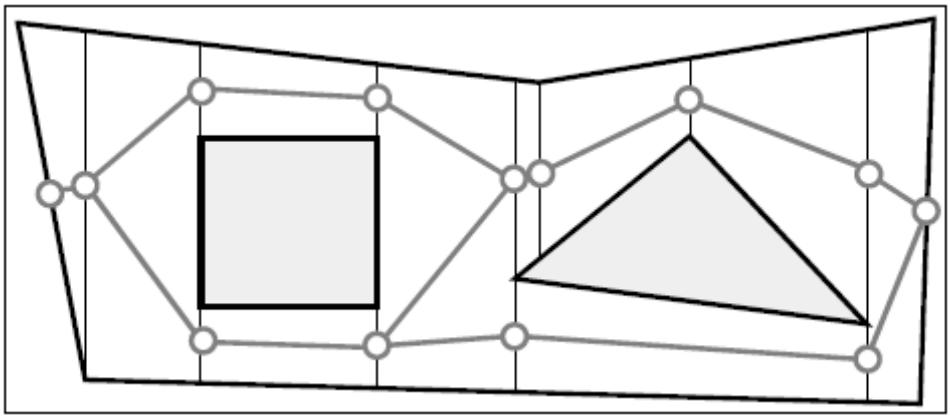


Figura 2.15. Grafo de conectividad de una descomposición trapezoidal.

2.4.2.5 Planificación basada en campos potenciales.

Los métodos basados en campos potenciales poseen una concepción totalmente distinta a los expuestos con anterioridad al estar basados en técnicas reactivas de navegación. El ámbito de uso de esta técnica se centra en la planificación local en entornos desconocidos, como puede ser el sorteo en tiempo real de obstáculos o de los que no se tiene constancia [Borentein y Koren, 1989].

La teoría de campos potenciales considera al robot como una partícula bajo la influencia de un campo potencial artificial, cuyas variaciones modelan el espacio libre. La función potencial U en un punto p del espacio euclídeo, se define sobre el espacio libre y consiste en la composición de un potencial atractivo $U_a(p)$, que atrae al robot hacia la posición destino, y otro repulsivo $U_r(p)$ que lo hace alejarse de los obstáculos, es decir:

$$U(p) = U_a(p) + U_r(p) \qquad (2.11)$$

La fuerza artificial $F(p)$ a la que afecta el vehículo en la posición p, por el potencial artificial $U(p)$ resulta:

$$F(p) = -\nabla U(p) \qquad (2.12)$$

Al igual que la función potencial, la fuerza artificial es el resultado de la suma de una fuerza de atracción $F_a(p)$, proveniente de la posición destino, y otra fuerza de repulsión $F_r(p)$ debidas a los obstáculos del entorno de trabajo:

$$F(p) = F_a(p) + F_r(p) \qquad (2.13)$$

Así, la navegación basada en campos potenciales se basa en llevar a cabo la siguiente secuencia de acciones:

i. Calcular el potencial $U(p)$ que actúa sobre el vehículo en la posición actual p según la información recabada de los sensores.
ii. Determinar el vector fuerza artificial $F(p)$ según la expresión (2.12).
iii. En virtud del vector calculado construir las consignas adecuadas para los actuadores del vehículo que hagan que éste se mueva según el sentido, dirección y aceleración especificadas por $F(p)$.

La iteración continua del ciclo expuesto proporciona una navegación reactiva basada en campos potenciales. El comportamiento del vehículo está muy ligado a la definición que se efectúe de los potenciales de atracción y repulsión. El potencial de atracción debe ir en función de la distancia euclídea a la posición destino, de forma, que a medida que el robot móvil se acerca, este disminuya su influencia. Por otra parte, el potencial repulsivo conviene que sólo influya en el movimiento del vehículo cuando éste se encuentre demasiado próximo a un obstáculo, de forma que la fuerza debida a este hecho tenga una dirección tal que lo aleje del mismo. En la posición destino es necesario que la suma de ambos potenciales resulte nula.

En el caso de conocer todo el entorno de trabajo y realizando una simulación del movimiento del robot a través del mismo, resulta posible construir una ruta que lleve al vehículo desde la posición inicial hasta la final. Dada la posición actual p_i, la próxima posición que debe alcanzar en un ciclo de simulación p_{i+1} resulta:

$$p_{i+1} = p_i + \delta_i \ \mathbf{J}(U(p)) \qquad (2.14)$$

donde δ_i es un factor de escalado y $\mathbf{J}(U(p))$ representa al jacobiano de la función potencial en el punto p. El factor de escalado define la longitud del segmento con origen en p_i y final en p_{i+1}, y debe ser tal que dicho segmento esté libre de obstáculos.

2.5 Generación de caminos.

El camino es el grupo de consignas que se entregarán al seguidor de caminos para la ejecución de la tarea de navegación. Se construye en función de la ruta definida por la tarea de planificación y debe estar libre de obstáculos. Además de esta característica básica, al utilizarse como referencia del seguidor de caminos, debe poseer ciertas cualidades que faciliten la acción de esta última tarea. La importancia de la definición de un camino con buenas propiedades reside en la capacidad del seguidor para realizar una ejecución del camino con el menor error posible. La acción del generador consiste en la conversión de una ruta en un camino, es decir construir una sucesión de configuraciones que lleve al vehículo de la posición inicial a la final, de forma que elimine la restricción de omnidireccionabilidad inherente a la definición de ruta.

El camino se define como la discretización de una curva continua que interpola ciertos *puntos elegidos* de la ruta calculada por el planificador. Por tanto, el problema de la

definición de un camino con buenas propiedades pasa por la construcción de la función camino adecuado que las posea. Las características buscadas son aquellas que hacen posible el seguimiento del camino especificado según el comportamiento cinemático y dinámico del vehículo.

2.5.1 Propiedades deseables de un camino.

El análisis de las características cinemáticas que debe tener el camino para que sea seguible por un robot móvil, necesita disponer de un modelo cinemático preciso de éste. Como arquetipo para el mencionado análisis se puede utilizar un modelo simplificado denominado *modelo de la bicicleta*, el cual se emplea con fiabilidad para el estudio de la cinemática en la mayoría de los robots móviles no omnidireccionales con ruedas, y de dos grados de libertad: direccionamiento y propulsión [Shin y Sing, 1990; Martínez, 1994].

Se denomina punto de guía del modelo del vehículo al punto que se desea controlar para seguir el camino. La elección de la situación de este punto es una decisión importante ya que afecta a los valores requeridos de direccionamiento y a la propulsión para realizar el seguimiento del camino a una velocidad dada. Por lo general, se elige el punto de guía en el modelo, situado en el punto medio del eje trasero, según se muestra en la figura 2.16.

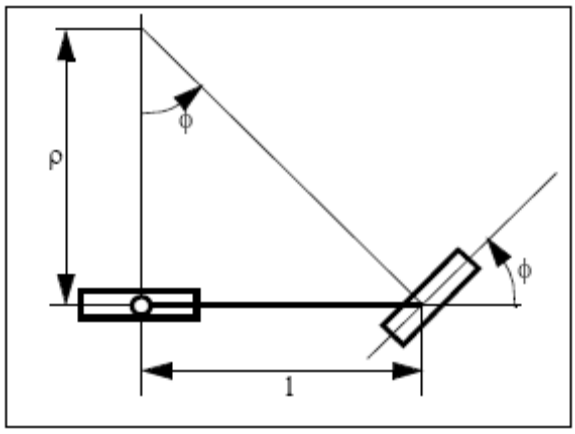


Figura 2.16. Modelo cinemático de la bicicleta.

Las ventajas de optar por la antedicha elección aparecen descritas a continuación:

- El ángulo de direccionamiento ϕ en cualquier punto del camino se determina de forma geométrica con independencia de la velocidad, de la siguiente forma:

$$\phi = a\tan\left(\frac{l}{\rho}\right) \qquad (2.15)$$

donde, l es la distancia entre ejes y y ρ el radio de giro actual.

- La velocidad angular ω de la rueda trasera motora queda determinada por la velocidad del vehículo v y el radio de la rueda R_w.:

$$\omega = \frac{v}{R_w} \qquad (2.16)$$

  Si el punto de guía se encontrase situado en cualquier otro lugar, las expresiones para el direccionamiento ϕ y la velocidad angular ω serían mas complejas que las mostradas en las ecuaciones (2.15) y (2.16).

- El vehículo posee la capacidad para realizar el mínimo radio de giro ρ_{min} con el valor máximo del ángulo de dirección ϕ_{max} [Nelson, 1988]
- La orientación del vehículo está alineada con la dirección de la tangente del punto actual del camino.
- El radio de giro actual coincide con el radio del círculo de osculación del punto presente del camino.

Estas dos últimas cuestiones quedan reflejadas en la figura 2.17.

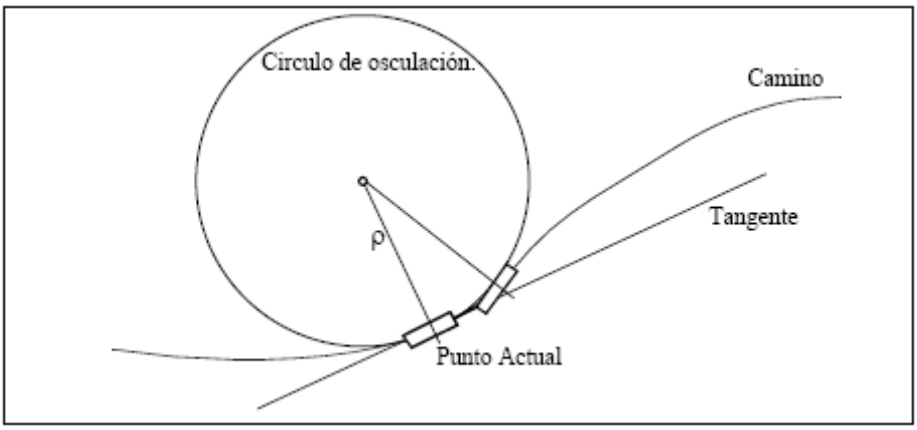


Figura 2.17. Orientación y curvatura del modelo a lo largo de un camino.

Mediante el uso de este modelo se solventa la necesidad de que la función sobre la cual se define el camino ofrezca las siguientes propiedades:

i. *Poseer continuidad en posición, orientación y curvatura*: Discontinuidades en la orientación del camino conllevan la necesidad de imprimir un cambio brusco en la orientación del vehículo, el cual no lo puede efectuar debido a la restricción de no holonomicidad. Por otra parte, una discontinuidad en la curvatura requeriría una aceleración infinita de la rueda de dirección.
ii. *Acotación de los valores que puede tomar la curvatura*: Debido a la definición de la curvatura como la inversa del radio del círculo de osculación existe una radio mínimo que puede realizar el vehículo según el ángulo máximo de direccionamiento.

iii. *Variación lineal de la curvatura*: Una variación suave y lineal de la curvatura minimiza el esfuerzo de control que se debe ejercer sobre los actuadores del vehículo. Como consecuencia se reducen los errores que se producen en el seguimiento del camino.

El cumplimiento de las condiciones expuestas por parte de la función sobre la cual se construye el camino, hace que este último resulte admisible desde el punto de vista cinemático. Además, las características de la curvatura de continuidad y variación lineal proporcionan cambios suaves en la fuerza centrífuga que actúa sobre robot móvil cuando éste se encuentra desplazándose sobre un camino especificado. Por tanto, las condiciones antes indicadas son también recomendables desde el punto de vista dinámico.

La literatura especifica un amplio grupo de métodos de generación de caminos. La técnica más simple para la generación de caminos se fundamenta en la unión de los puntos elegidos de la ruta mediante la concatenación de segmentos y círculos [Nelson, 1990]. La principal desventaja de los métodos basados en esta premisa, según las conclusiones del párrafo anterior, se constituye en la aparición de discontinuidades del ángulo de direccionamiento en ciertas configuraciones en la rueda directriz. Otros métodos evitan este problema con la construcción de caminos continuos en curvatura utilizando alguna clase de curva de interpolación cúbica como por ejemplo curvas cúbicas tipo Bezier [Segovia y Rombaut, 1993], pero la mayoría de ellos no asegura una limitación de la curvatura. Las curvas clotoidales [Kanayama y Miyake, 1985] o espirales [Kanayama y Hartman, 1990] sí aseguran una curvatura acotada a lo largo del camino, incluso su variación de forma lineal, pero no poseen una fórmula cerrada para su cálculo, lo cual dificulta su ejecución en tiempo real.

CAPÍTULO 3
GRÁFICAS POR COMPUTADORA

3.1 Introducción

Un computador contemporáneo es una máquina calculadora electrónica rápida que acepta como entradas información digitalizada, la procesa de acuerdo a una lista de instrucciones almacenadas internamente y produce la información de salida resultante de acuerdo a la información solicitada. La información solicitada puede ver en forma de gráficos

Gráficas por computadora es uno de los campos disponibles para su desarrollo dentro de las computadoras. Sistemas de computación complejos que se usan hoy día están diseñados para la generación de despliegues gráficos. Se utilizan figuras como eficaz medio de comunicación, y la capacidad de conversar en forma gráfica con las computadoras está revolucionando la forma en que las computadoras se están utilizando en todas las áreas del acontecer moderno

Las computadoras se han convertido en una herramienta poderosa para la producción de ilustraciones animadas. Prácticamente son pocas las áreas del conocimiento científico y tecnológico en las cuales no puedan utilizarse los despliegues gráficos. Hoy día, se puede advertir que estas gráficas se utilizan rutinariamente en áreas diversas como [Donald – Pauline Baker, pp 1-3]: gobierno, administración, industria, investigación, capacitación, arte, publicidad, entretenimiento, medicina, robótica, automatización, instrumentación, etc.

Como ejemplo de aplicación se realiza un breve recorrido a los diferentes campos de aplicación de las gráficas por computadora, a través de una galería de usos y aplicaciones de las gráficas.

- Despliegues generados por computadora que se usan en el modelado y simulación, demostraciones de capacitación y la graficación de informes
- Proyectos de dibujo mecánico en la realización bidimensional y tridimensional de partes y ensamble de una máquina con sistemas CAD (diseño asistido por computadora)
- Proyección CAD de terminación a máquina tridimensional controlada numéricamente de una parte, la superficie de la parte se despliega en un color y la trayectoria de la herramienta en otro color
- Proyectos con estructura de alambre de diseño de cuerpo completo y en conjunto con análisis de elemento finito
- Modelado de sólidos de un automóvil, del casco de un barco y en el diseño virtual de teatro, edificios, etc.
- Galaxias graficadas a partir de datos de la observación astronómica
- Despliegue gráfico a partir de datos de llamas solares
- Imagines que muestran depósitos de petróleo en el subsuelo

- Exámenes PET que despliega el flujo sanguíneo en un corte horizontal a través del músculo cardiaco pulsante, el cual se utiliza para medir el dolor torácico tal como ocurrió
- Examen PET (tomografía por emisión de la posición) que muestra la activación cerebral a partir del estímulo del sonido. Los muestran que el estímulo verbal produce la activación de la parte izquierda del cerebro y la música produce la activación de la parte derecha
- Técnica de procesamiento de imagines aplicadas a una imagen de una célula cancerosa, con la ventaja de agrandar y realzar la imagen de la célula cancerosa
- En el manejo de fotografías borrosas que se vuelven legibles después de la aplicación del método de procesamiento de imágenes
- Escenas generadas por un simulador de manejo, el cual se emplea para estudiar las reacciones de los conductores en situaciones críticas
- Despliegue de gráficas de terreno, edificios y aviones que se utilizan como parte de una secuencia de animación que permite la práctica de formación de vuelo con un simulador de vuelo
- Vista generada por computadora de una pista utilizada por pilotos que operan un simulados de vuelo para practicar aterrizajes
- Simulación con gráficas del recorrido de la cápsula espacial Viajero, pasando la superficie del planeta Saturno. El lado oscuro de Saturno con la tierra
- Aplicación de la gráficas en arte comercial para publicidad de televisión y cine
- Las técnicas de manejo de proyectos se sirven de diagramas de tiempos y proyectos de redes de tareas para programar y monitorear proyectos
- El comportamiento de los sistemas físicos a menudo se estudian construyendo gráficas y modelos.

El comportamiento de los sistemas físicos a menudo se estudian construyendo gráficas y modelos. Cuando deben analizarse grandes cantidades de datos, una gráfica codificada en color puede ayudar a los investigadores a entender la estructura de un sistema. Sin la ayuda de estas gráficas, sería muy difícil para un investigador interpretar tablas de datos con millones de registros. Los avances en la tecnología de la computación han hecho que las gráficas interactivas por computadora sean una herramienta práctica y de amplio crecimiento para el futuro

3.2 Procesamiento de imágenes

La técnica de graficación que se usa para producir despliegues visuales a partir de fotografías o exploraciones de TV se llama procesamiento de imágenes. Aunque las computadoras se utilizan con estos despliegues, los métodos de procesamiento de imágenes difieren de los métodos convencionales de gráficas de computadora. En las gráficas de computadora tradicional, una computadora se usa para crear la imagen. Las técnicas de procesamiento de imágenes usan una computadora para digitalizar los modelos de sombreado y color a partir de una imagen ya existente. La información digitalizada se transfiere después a la pantalla de un monitor de video. Tales métodos son útiles para visualizar muchos sistemas u objetos que no se pueden apreciar directamente, como las

exploraciones de TV desde una nave espacial o las imágenes visuales del ojo de un robot industrial.

Una vez que se ha digitalizado una imagen, puede aplicarse otra técnica de procesamiento de imágenes para reacomodar las partes de la figura, en la que se realiza separación de colores y tonalidad o mejoramiento de la calidad de las sombras. El procesamiento de imágenes se utiliza ampliamente en aplicaciones de arte comercial que implica el retoque y reacomodación de secciones de fotografías y otros trabajos artísticos. Las aplicaciones médicas se apoyan de técnicas de procesamiento de imágenes para realizar imágenes y en la tomografía de células cancerosas [Hawrylyshyn, Tasker y Organ, 1977]

3.3 Interfases con el usuario de gráficas

Las opciones de entrada a muchos programas de computadora se diseñan como un con junto de iconos, símbolos gráficos que realizan la opción de procesamiento que deben representar. Los usuarios seleccionan opciones de procesamiento señalando el icono adecuado. La ventaje de estos sistemas es que los iconos ocupan menos espacio en la pantalla que la descripción textual correspondiente de las funciones pueden entenderse más rápido, sí están bien diseñado.

Como resultado del reconocimiento generalizado de la utilidad de las gráficas por computadora existe actualmente una gran variedad de equipos de hardware y sistemas de software para aplicaciones gráficas. Con capacidad de realizar aplicaciones bidimensionales así como tridimensionales con una gran aceptación en varios departamentos que forman parte de la empresa metalmecánica, como es el departamento de Ingeniería de producto. El proceso de manufactura también se asocia con la descripción por computadora de objetos diseñados para automatizar máquinas, acorde al personal que dirige construcción de uno o varios productos. Por ejemplo, se puede convertir el esquema de un tablero de circuitos electrónicos en una descripción de los procesos individuales para facilitar la elaborar del producto

3.4 Educación y capacitación

A menudo, se utilizan como instrumentos de ayuda educativa modelos de sistemas físicos, financieros y económicos, los cuales se generan por computadora. Modelos de sistemas físicos, sistemas fisiológicos, tendencias de población o equipo, pueden ayudar a los estudiantes a comprender la operación del sistema. En el caso de algunas aplicaciones de capacitación, se diseñan sistemas especiales, como los simuladores para sesiones de práctica o capacitación de capitanes de barco, pilotos de avión, operadores de equipo pesado y el personal de control de tráfico aéreo. Algunos simuladores no tiene pantallas de video; por ejemplo, un simulador de vuelo que sólo tiene un panel de control como instrumento de vuelo. No obstante, la mayor parte de los simuladores cuenta con pantallas gráficas para la operación visual.

3.5 Visualización

Científicos, ingenieros, personal médico, analistas comerciales y otros con frecuencia necesitan analizar grandes cantidades de información o estudiar el comportamiento de ciertos procesos. Las simulaciones numéricas efectuadas en supercomputadoras a menudo producen archivos de datos que contienen miles e incluso millones de valores de datos. De modo similar, cámaras vía satélite y otras fuentes acumulan grandes archivos de datos más rápido de lo que se pueden interpretar. El rastreo de estos grandes conjuntos de números para determinar tendencias y relaciones es un proceso tedioso e ineficaz. Pero si se convierten los datos a una forma visual, es frecuente que se perciban de inmediato las tendencias y los patrones. Por lo regular, la producción de representaciones gráficas para conjuntos de datos y procesos científicos de ingeniería y de medicina se conoce como visualización científica. El término visualización empresarial se emplea en relación con conjuntos de datos que se asocian con el comercio, la industria y otras áreas no científicas. Existen muchas clases de conjuntos de datos y los esquemas de visualización efectivos dependen de las características de los datos. Una compilación de datos contiene valores escalares, vectores, tensores de orden superior o cualquier combinación de estos tipos de datos. Y los conjuntos de datos pueden ser bidimensionales o tridimensionales. La codificación de colores es sólo una manera de visualizar un conjunto de datos. Las técnicas adicionales incluyen trazos, gráficas y diagramas de contorno, presentaciones de superficie y visualización de interiores de volumen. Además, se combinan técnicas de procesamiento de imágenes con gráficas por computadora para crear muchas de las visualizaciones de datos. Las comunidades de matemáticos, científicos físicos y otros utilizan técnicas visuales para analizar funciones matemáticas y procesos o sólo con el propósito de crear representaciones gráficas interesantes.

CAPÍTULO 4
SISTEMA DE GRÁFICAS

4.1 Introducción

Los sistemas de computación pueden adaptarse a aplicaciones gráficas en varias formas y conforme a los recursos de software y hardware de que se dispone. Cualquier computadora de uso general puede utilizarse para hacer gráficas de caracteres utilizando elementos del conjunto de caracteres del sistema con objeto de formar modelos o patrones. A pesar de ser una técnica simple, las gráficas de caracteres pueden producir diseños complicados. Si un monitor de video y los comandos de trazo son parte de nuestro sistema, podemos crear y manipular en forma interactivo estos trazos en la pantalla del monitor. Para aplicaciones más complejas, en especial para aplicaciones científicas y tecnología aplicada, se dispone de una gran variedad de paquetes de software y dispositivos de hardware

A continuación se hace una breve explicación del hardware y software que se requiere para realizar gráficas en computadora

4.2 Dispositivos de despliegue

Los sistemas interactivos emplean algún tipo de monitor de video como dispositivo principal de salida [Sherr, 1979]. La operación de muchos monitores de video se basa en el diseño estándar del tubo de rayos catódicos (CRT). Un haz de electrones emitidos por un disparador de electrones, atraviesa los sistemas de enfoque y deflexión que dirigen el haz hacia puntos específicos en la pantalla cubierta de fósforo. El fósforo emite después una pequeña mancha de luz en cada punto contactado por el haz de electrones, se dispone de varios tipos de fósforo para usarse en un CRT. Además de los colores, una diferencia importante entre los fósforos es su persistencia

La deflexión del haz de electrones se realiza con campos eléctricos o bien campos magnéticos. El haz de electrones pasa entre dos pares de placas de metal: un par vertical y otro horizontal. Se aplica una diferencia de tensión a cada par de acuerdo con la cantidad de haz que se reflexiona primero en cada dirección. A medida que el haz de electrones pasa entre cada par de placas, se dobla hacia la placa con la mayor tensión positiva.

El número máximo de puntos que puede exhibir sin superposición en un CRT de puntos por centímetro se conoce como resolución. La resolución de depende del tipo de sustancia fosforescente que se utiliza y de los sistemas de enfoque y deflexión. Los sistemas de alta precisión pueden desplegar un máximo de 4000 puntos en cada dirección, para hacer un total de 16 millones de puntos direccionables en la pantalla

La renovación de un sistema de rastreo con rastreador se lleva a cabo a intensidades de 30 a 60 cuadros por segundo. A razón de 30 cuadros por segundo, el haz de electrones recorre todas las líneas de la apantalla de arriba hacia abajo 30 veces por segundo. Una taza de renovación menor de cerca de 25 cuadros por segundo hace que la figura se desvanezca

Otra propiedad de los monitores de vídeo es la razón de aspecto. Este número da la proporción de los puntos verticales con respecto de los puntos horizontales necesarios para producir líneas con una longitud igual en ambas direcciones de la pantalla, expresada en términos de la razón de puntos horizontales respecto a verticales. Por ejemplo una razón de aspecto – de tres a cuatro – implica que una línea vertical trazada con tres puntos tiene la misma longitud que una línea horizontal que se traza con cuatro puntos.

4.3 Monitor CRT a color

Un monitor CRT despliega figuras de color utilizando una combinación de sustancias fosforescentes que emiten luz de diferentes colores. Las dos técnicas básicas para producción de despliegues a color con un CRT son el método de penetración del haz y el método de la máscara de sombra.

El método de penetración del haz utiliza dos capaz de sustancia fosforescente por lo general rojo y verde, se colocan en la pantalla y el color exhibido depende de cuánto penetre el haz de electrones en las capaz fosforescentes. Un haz de electrones lento excita solamente la capa roja exterior. Un haz de electrones muy rápidos penetra a través de la capa roja y excita la capa verde interior. A velocidades intermedias del haz, se emiten combinaciones de luz roja y verde para mostrar dos colores adicionales, amarillos y naranjas.

Un CRT de máscara con sombra cubre la pantalla con modelos triangulares pequeños, cada uno de los cuales contiene tres puntos fosforescentes diferentes con muy poco espacio entre ellos. Un punto fosforescente de cada triangulo emite una luz azul. Este tipo de CRT tiene tres disparadores de electrones, uno para cada punto de color y una retícula de máscara con sombra apenas detrás de la pantalla cubierta con sustancia luminosa. Los tres haces de electrones son desviados y enfocados como grupo sobre la máscara con sombra, la cual contiene una serie de orificios alineados con los modelos de puntos fosforescentes. Cuando los tres haces atraviesan un orificio en la máscara con sombra, éstos activan un triángulo formado con puntos, el cual aparece como una pequeña mancha de color en la pantalla. Las variaciones de color con máscara se obtienen combinando varios niveles de intensidad de los tres haces de electrones

Aunque muchos monitores de gráficas contienen CRT, otras tecnologías que se emplean en algunos sistemas. Los despliegues en paneles de plasma se construyen llenando la región entre dos placas de vidrio con gas neón. Una serie de electrodos verticales y horizontales, colocados en los paneles de vidrio frontal y anterior respectivamente, se utilizan para iluminar puntos individuales en el neón

Otras dos tecnologías que se usan en el diseño de monitores de gráficas son los diodos emisores de luz (LED) y los despliegues en cristal líquido (LCD). Estos dispositivos emplean luz emitida de diodos o cristales en vez de sustancias fosfóricas o gas neón para el despliegue de gráficas o imágenes.

Una técnica adicional que no es el CRT para generar una salida de gráficas consiste en trazar modelos sobre película foto cromática que se oscurece temporalmente por la

exposición a la luz. Los modelos se forman con un haz de rayo láser, se hace deflexión por espejos controlados electromecánicamente. Después se utiliza otra fuente de luz para proyectar las imágenes en una pantalla. Un cambio de las imágenes de la pantalla se obtiene enrollando el carrete de película en el siguiente cuadro vacío y repetir el proceso.

Los monitores de gráficas para el despliegue de escenas tridimensionales se han ideado aplicando una técnica que refleja una imagen del CRT desde un espejo flexible vibratorio. Conforme el espejo (llamado espejo unifocal) vibra, cambia la longitud focal. Estas vibraciones se sincronizan con el despliegue de un objeto en un CRT de manera que cada punto del objeto se refleja desde el espejo en una posición a la profundidad de ese punto. Otra técnica de representación de objetos tridimensionales es el despliegue de vistas estereoscópicas. Este método no produce imágenes tridimensionales verdaderas, sino que proporciona un efecto tridimensional presentando una vista diferente a cada ojo del observador. Una manera de lograr esto consiste en desplegar las dos vistas de una escena en diferentes mitades de la pantalla de un CRT. Un filtro polarizante se coloca sobre cada mitad de la pantalla y el usuario observe la pantalla a través de un par de anteojos polarizados

4.4 Dispositivos de copia dura

Muchos sistemas de gráficas están equipados para producir salidas en copia dura directamente de un monitor de video en la forma de acetato o transparencia d. Las figuras en copia dura también pueden obtenerse dirigiendo la salida gráfica a una impresora o graficadora

Se dispone de dos métodos para realizar la impresión:

- Método de impacto
- Método de no impacto

Respecto a graficado se dispones de:

- Pluma de tinta para generar el trazo
- Haces de rayo láser
- Atomizador de chorro de tinta
- Métodos electrostáticos

4.5 Dispositivos de entrada interactivos

Estos dispositivos incluyen ratón, bola palmar, esfera de control, palanca de control, digitalizadores, discos marcadores y botones. Algunos otros dispositivos se utilizan en aplicaciones particulares con los guantes de datos, paneles de tacto, rastreadores de imágenes y sistemas de voz.. Otros dispositivos de posicionamiento del cursor, como la bola palmar a la palanca de control, se incluyen en algunos teclados. Con el ratón Z podemos levantar un objeto, hacerlo girar y moverlo en cualquier dirección o podemos desplazar nuestra posición de vista y orientación a través de una escena tridimensional. Las

aplicaciones del ratón Z incluyen realidad virtual, CAD y animación. Mientras que la bola palmar es un dispositivo de posicionamiento bidimensional, una esfera de control ofrece seis grados de libertad.

Los teclados se incluyen en muchos sistemas de gráficas para realizar entrada de coordenadas de caracteres y valores de datos. En un teclado se pueden incluir varios tipos de llaves, discos e interruptores para manejar diversas aplicaciones. Los valores coordenados que especifican posiciones en la pantalla se introducen con mayor frecuencia con tablas de gráficas, pluma luminosa o pluma interna o una palanca de mando, se usan comúnmente para trazar figuras o hacer selecciones de menús. Una tableta introduce posiciones coordenadas activando un cursor manual o un estilo en posición de selección en la superficie de la tableta. Una pluma luminosa indicada en un monitor de video registra posiciones coordenadas, respondiendo a la luz emitida de sustancias fosforescentes en la pantalla.

4.6 Procesadores de despliegue

Los sistemas de gráficas interactivos emplean dos o más unidades de procesamiento. A demás de la unidad central de procesamiento (UCP), se utiliza un procesador de despliegue de uso general para interactuar con la UCP y controlar la operación del dispositivo de despliegue. Básicamente el proceso de despliegue se utiliza para convertir información digital de la UCP en valores de tensión correspondiente que necesita el dispositivo de despliegue [Haber y Wilkinson, 1982]. La forma en la cual se realiza esta conversión de digital a analógico depende del dispositivo de despliegue que se usa y de las funciones de gráficas particulares que se instrumentan en hardware

Los programas de aplicación de sistemas de gráficas interactivos ofrecen información de figuras al procesador de despliegue en términos de niveles de intensidad de luz de puntos coordenados de la pantalla. En muchos monitores de gráficas, el origen se define en la equina inferior izquierda de la pantalla. La superficie de la pantalla se representa después como el primer cuadrante de un sistema coordenado bidimensional, con valores positivos de x que crecen hacia la derecha y valores positivos de y que crecen de abajo hacia arriba. En otras computadoras, el origen se refiere a la esquina superior izquierda de la pantalla, de manera que los valores de y se invierten. Los sistemas de gráficas a menudo permiten que los programas de aplicación definan puntos de una figura utilizando cualquier referencia de coordenadas que convenga al usuario. Después una transformación es realizada por el sistema para convertir las coordenadas del usuario en valores de la pantalla. Una tarea fundamental para el procesador de despliegue es la exhibición de segmentos de líneas. Los niveles de intensidad (o valores de color) que se usarán en posición coordenadas de graficación a lo largo de una línea son proporcionados por el programa de aplicación y se convierten en nivel de tensión, que después se aplican al dispositivo de despliegue. Para sistemas simples de blanco y negro, no necesita especificarse ninguna información de intensidad en el programa, ya que cualquier punto está encendido o apagado, los sistemas de mayor calidad permiten que la intensidad de los puntos de la pantalla sea variada de manera que puedan desplegarse sombras grises

Los procesadores de despliegue avanzado están diseñados para realizar otras operaciones. Estas funciones incluyen la generación de varios estilos de líneas, despliegues de áreas con color, producción de líneas curvas y realización de ciertas transformaciones y manipulación de objetos desplegados

4.7 Sistema de rastreo al azar

Los comandos de gráficas de un programa de aplicaciones se traducen en un programa de archivo de despliegue, que tiene acceso por el procesador de despliegue para renovar la pantalla. El procesador de despliegue entra en un ciclo a través de cada comando del programa de archivo de despliegue una vez durante cada ciclo de renovación. Esta actualización debe sincronizarse por el proceso de renovación, de manera que no se distorsione la imagen mientras está en proceso de renovación.

Aquí un CRT dirige el haz de electrones sólo a las partes de la pantalla donde se debe crear la imagen. Los monitores de trazado aleatorio trazan una imagen, una línea a la vez. Las líneas que componen una imagen se pueden trazar y refrescar o enfriar mediante un sistema de trazado aleatorio en cualquier orden específico. Los sistemas de trazado aleatorio están diseñados para aplicaciones de trazo de líneas y no pueden desplegar escenas sombreadas realistas.

Los modelos de gráficas se trazan en un sistema de rastreo al azar dirigiendo el haz de electrones a lo largo de las líneas componentes de la figura. Las líneas se definen por los valores de sus extremos coordenados y estos valores coordenados de entrada se convierten en tensión de deflexión x ψ y. Después se traza una escena, una línea a la vez, posicionando el haz para llenar la línea entre los puntos extremos especificados.

Las líneas rectas se trazan en un sistema de rastreo al azar con un generador de vectores, que produce tensiones de deflexión del haz de electrones. Estas tensiones de deflexión pueden generarse en una de dos formas. Un generador de vectores analógicos envía el haz de electrones directamente desde un punto extremo de la línea al otra variante linealmente las tensiones de reflexión, se produce una línea recta alisada entre los dos puntos. Un generador de vectores digital calcula puntos sucesivos a lo largo de la línea, comenzando en un extremo y convierte estos valores coordenados en tensión. Y así se construye una línea recta como un conjunto de puntos. Aunque el método digital no produce una línea alisada como el método analógicos; por lo general, es más rápido y menos costoso.

4.8 Sistemas de rastreo con rastreador

Funciona recogiendo el haz de electrones a través de cada línea, activándose o desactivando la intensidad del haz para crear un patrón de manchas iluminadas. La definición de la imagen se almacena en una área de memoria llamada buffer de repasado o buffer de marco o estructura, que contiene el conjunto de valores de intensidad para todos los puntos de la pantalla.

Cada posición del buffer de estructura se llama elemento de la figura o bien píxel. Las figuras se pintan en la pantalla desde el buffer de estructura, una hilera a la vez, de arriba

hacia abajo. Cada línea horizontal de píxeles se conoce como línea de rastreo y el proceso de generación de información de píxeles en el buffer de estructura del programa de aplicación se conoce como conversión de rastreo. Los valores de intensidad se almacenan en el rastreador durante el tiempo de retrazo vertical

Las posiciones de los píxeles en el buffer de estructura se organizan como un arreglo bidimensional de valores de intensidad correspondientes a posiciones coordenadas de la pantalla. El número de posiciones de píxeles en el rastreador se denomina resolución del procesador de despliegue (o bien resolución del buffer de estructura). Los sistemas rastreadores de buena calidad tienen una resolución de cerca de 1024 por 1024 requiere 3Mb de almacenamiento para el buffer de imagen, aunque se dispone de sistemas de resolución más alta. Como la resolución de un monitor CRT depende del punto fosforescente que se puede producir, un sistema de gráficas puede tener dos resoluciones, una para el monitor que se usa con el sistema y una para el buffer de estructura. Para generar las imágenes de la mejor calidad, la resolución del monitor de video debe ser igual o mayor que la resolución del buffer de estructura.

En un sistema de blanco y negro, cada punto de la pantalla está apagado o encendido, de manera que sólo se necesita un bit por píxel para controlar la intensidad de las posiciones de la pantalla. Se necesitan otros bits cuando pueden desplegarse variaciones de color e intensidad.

Hasta 24 o más bits por píxel se incluyen en los sistemas de alta calidad, aunque los requerimientos de almacenamiento del buffer de estructura se vuelve después muy elevados. Cada línea se almacena como un conjunto de puntos y los caracteres se almacenan como un modelo de matriz de puntos. Una ventaja importante del almacenamiento de valores de intensidad en buffer de estructura es la representación de áreas que serán llenadas con modelos de color o de sombreado.

Una manera de lograrlo consiste en almacenar cada línea de rastreo como un conjunto de pares enteros. Un número de cada par indica un valor de intensidad y el segundo número especifica el número de píxel adyacente en la línea de rastreo que tiene esa intensidad. Esta técnica denominada codificación de la longitud del tramo, puede lograr un ahorro considerable de almacenamiento sí una imagen se construye principalmente con tramos largos de un solo color cada una

4.9 Software de gráficas

Hay dos clasificaciones generales para el software

- paquetes generales de programación
- paquetes de aplicaciones especificas

Los comandos de programación para desplegar y manipular salidas de gráficas están diseñados como extensiones de lenguajes existentes. Las funciones básicas de que se dispones en un paquete diseñado para el programa de gráficas incluye aquellas para la

generación de componentes de una figura (líneas rectas, polígonos, circunferencias y otras figuras), fijación de color e intensidad, selección de vistas y aplicación de transformaciones. En cambio, los paquetes de gráficas de aplicación diseñados para no programadores se forman de manera que los usuarios puedan producir gráficas sin preocuparse por la forma en que lo hagan. La interfase con las rutinas de gráficas en estos paquetes les permite establecer comunicación con los programas en términos propios.

Muchos paquetes de gráficas están diseñados para utilizar sistemas de coordenadas cartesianas. Más de un sistema cartesiano puede ser referido por un paquete, ya que diferentes dispositivos de salida pueden requerir diferentes sistemas de coordenadas, por lo general permiten que se construyan definiciones de imágenes en cualquier sistema de referencia cartesiano que convenga a la aplicación que se tiene disponible. Las coordenadas referidas por un usuario se denominan coordenadas mundiales y las coordenadas que utiliza un dispositivo de salida particular reciben el nombre de coordenadas de dispositivo o bien coordenadas de la pantalla en caso que se trate de un monitor de video. Un procedimiento común que se utiliza en los paquetes de gráficas consiste primero en convertir las definiciones de las coordenadas mundiales en coordenadas de dispositivo normalizados antes de la conversión final a coordenadas de dispositivo específico

4.10 Funciones de gráficas

Las estructuras básicas de imágenes se conocen como primitivas de salida, entre las que se incluyen cadenas de caracteres y entidades geométricas: puntos, líneas, rectas, polígonos y circunferencias. Los atributos son las propiedades de las primitivas de salida. Incluyen especificaciones de color e intensidad, estilo de línea, estilo de texto y modelos de llenado de áreas

Las transformaciones de vistas se utilizan para especificar la vista que se presentará y la porción del área en el despliegue de salida que se usará, las imágenes pueden subdividirse en partes componentes o segmentos. Una escena con varios objetos podría definir la construcción de cada objeto en un segmento nombrado por separado.

Un paquete de gráficas contiene comúnmente varias tareas de mantenimiento, como el borrado de la pantalla o la inicialización de parámetros, se recomienda agrupar las funciones para realizar estos trabajos rutinarios en el encabezado operaciones de control

4.11 Normas de software

Organizaciones nacionales e internacionales de planeación de estándares, desarrollan un estándar para gráficas por computadora. Sistema gráfico de Kernel (GKS; Graphical Kernel System) [Hopgood y otros, 1983]. La ISO (Organización Internacional de Estándares) incluyendo el American National Standards Institute (ANSI) adoptaron este sistema como el primer estándar de software de gráficas. Aunque, al principio el GKS se diseño como un paquete de gráficas bidimensionales posteriormente se desarrollo una extensión tridimensional. El segundo estándar que se aprobó fue el PHIGS (Estándar Jerárquico de Graficas Inter-activas para el Programador). Las funciones gráficas estándar se definen

como un conjunto de especificaciones [véase Graphics Standards Planning Committee, 1977 y 1979], que es independiente de cualquier lenguaje de programación. Una vinculación del lenguaje se define entonces para un lenguaje particular de programación.

Las funciones de gráficas estándar se definen como un conjunto de especificaciones abstractas, independientes de cualquier lenguaje de programación. Para instrumentar una norma de gráficas en un lenguaje de programación determinado, debe definirse una vinculación de lenguaje, esta vinculación define la sintaxis para acceder las varias funciones de gráficas que se especifican dentro de la norma.

CAPÍTULO 5
PROPIEDADES DE LAS PRIMITIVASA DE SALIDA

5.1 Algoritmo para el trazo de primitiva de salida

Los procedimientos que despliegan primitivas de salida dirigen un dispositivo de salida para producir estructuras geométricas especificadas en la localidad designada. Dichos algoritmos toman coordenadas de la entrada y realizan despliegues para construir una figura geométrica en un dispositivo de salida seleccionado, para este trabajo el dispositivo de salida será el monitor de la PC.

Los componentes geométricos más simples de una imagen son los puntos (píxeles) y la unión de los mismos por líneas rectas [Donald H., y M. Pauline, 1995]. Otros tipos de primitivas de salida son las áreas poligonales, figuras curvas y cadenas de caracteres. En cada tipo de primitivas se considera las técnicas de algoritmos básicos para desplegar la primitiva en diferentes tipos de sistemas gráficos, como los sistemas rastreadores y los vectoriales

Cuando vamos a realizar una aplicación gráfica, necesitamos recurrir a unos elementos llamados *primitivas* para poder llevar a cabo los dibujos. Las primitivas de dibujo son los elementos básicos de los que disponemos para realizar cualquier representación gráfica: puntos, líneas, círculos. La primera de estas primitivas es muy dependiente del hardware de la máquina, del sistema operativo, del modo de vídeo, y dado que siempre podemos usar la función para este fin que viene incluida en alguna librería de gráficos, vamos a tomarla como básica disponible, por ejemplo (pixels[i][j]=clColor) a una función que vamos a tener a nuestra disposición siempre que la necesitemos. Significa "pintar un punto en la posición (x,y) de la pantalla del color 'color'". Hay que hacer notar que en la pantalla, la esquina superior izquierda tiene coordenadas (0,0), y que la coordenada *x* crece hacia la derecha mientras que la coordenada *y* crece hacia abajo [Foley, J. D. y otros, 1994].

En la resolución en la que trabajemos, tendremos un máximo de x_max puntos de ancho por y_max puntos de alto, por tanto, el rango permitido para dibujar es el rectángulo de esquina superior izquierda (0, 0) y de esquina inferior derecha (xmax-1, ymax-1).

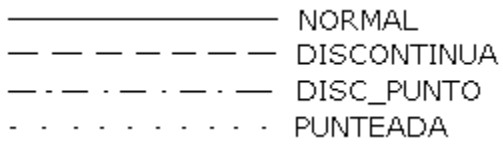
Una vez inicializado el modo, tenemos dos posibilidades: modificar el contexto gráfico o usar alguna función de dibujo. Se entiende por **contexto gráfico** un conjunto de propiedades que determinan con qué estilo se van a pintar las primitivas en pantalla.

El estilo más básico es el **color**, que es aplicable a todas las primitivas, pues todas ellas son susceptibles de poder ser dibujadas con algún color.

También podemos establecer la forma en que las líneas serán dibujadas: grosor y trazo continuo o discontinuo, pudiendo elegir entre varias posibilidades. El estilo de trazo de líneas pone de manifiesto que cualquier estilo no es susceptible de ser aplicable a cualquier

primitiva. De todas formas, no hay que preocuparse por qué primitivas admiten un estilo y qué primitivas no lo admiten: el esquema general de una librería gráfica en este aspecto es, primero, fijar el estilo, para después dibujar la primitiva con ese estilo, si es posible aplicarlo, y si no es posible, la dibuja sin aplicarlo, únicamente empleará aquello parámetros aplicable a la primitiva en cuestión.

Además de tener grosores distintos, las líneas pueden tener varios tipos de trazo. A continuación mostramos unos ejemplos de trazos y los identificadores que usaremos (las librerías gráficas dan completa información y ejemplos sobre las posibilidades):



Posibles patrones de relleno son, por ejemplo, sólido, que hace referencia al relleno normal. Trazo, que hace una trama de líneas horizontales y verticales en lugar del relleno sólido, Trazo oblicuo, que hace una trama de líneas oblicuas. Punteado, que hace una trama de puntos, transparente que no rellenará la figura.

5.2 Sistema visual u ocular

Las imágenes que percibimos son debidas a nuestro sistema visual [Agur, 1994.]: ojo, cerebro, y nervio óptico, principalmente. El ojo, que forma parte de este sistema visual, se caracteriza por sus componentes oculares. El iris controla la cantidad de luz que es permitida entrar al ojo. La lente ocular concentra la luz permitida en el ojo para formar una imagen. La proyección de la imagen es situada en la retina - una estructura de dos dimensiones. La retina se compone principalmente de conos y bastoncillos, que son células especiales. Estos conos y bastoncillos tienen la funcionalidad de recibir estímulos de luz para nuestra visión diurna y nocturna, respectivamente. La luz es descompuesta en las intensidades que corresponden a los colores rojo, verde, y azul. Este trío de colores se denomina los colores primarios. A partir de los colores primarios podemos conseguir otros tres colores: cian, amarillo, y magenta, llamados los colores complementarios. Cian, amarillo, y magenta son complementarios a los colores rojo, verde, y azul, respectivamente, ya que combinados formarán el color blanco.

5.3 Resolución gráfica

En un sistema gráfico se utilizan colores para formar una imagen. Los colores son combinados para formar tonos y otros colores, y se ubican en un área pequeña. Esta área es llamada píxel, que viene del inglés: *picture element* o elemento pictográfico. Este elemento o píxel es la parte fundamental más pequeña que contiene información para crear una

imagen en un sistema gráfico. Un conjunto de píxeles forma una malla para crear una imagen completa.

La mayoría de los sistemas gráficos usa una malla rectangular con las mismas dimensiones de una superficie rectangular: anchura y altura. La anchura es el número de columnas y la altura, el número de filas de tal superficie gráfica. El número de píxeles en la anchura y en la altura constituyen la resolución (gráfica) de la imagen. Por decir, 640 x 480 significa que la resolución gráfica de la imagen constituye 640 columnas y 480 filas de píxeles formando una malla de 307.200 píxeles. Analizando otro caso: 800 x 600, comprobamos que obtenemos una imagen de 480.000 píxeles. Esta imagen contiene muchos más píxeles y por tanto mayor resolución pictográfica. Con una mayor resolución gráfica se obtiene una mejor calidad de imagen.

La resolución y el concepto de una malla de áreas de colores son análogos a un mosaico. En tal forma de arte, un mosaico intenta representar una imagen a partir de azulejos pequeños de determinados colores y tonos. Cada azulejo intenta aproximarse en color a un área de la imagen, combinando los colores en área indicada.

5.4 Representación de colores

En sistemas gráficos, los colores son descompuestos y representados por combinaciones de valores numéricos. Generalmente, se habla de un color diferente para cada combinación de valores. En realidad, se está hablando de tonos diferentes de colores, pero en el argot informático cada tono es tratado como un color diferente. Por ejemplo, si se habla de un sistema gráfico de 15 bits por píxel (bpp) tenemos a nuestra disposición 32.768 colores, aunque en verdad sean unos cuantos colores y varios miles de tonos. De igual forma, podemos manipular 24 bpp obteniendo un conjunto de 16.777.216 colores.

Existen varios modelos de descomposición de colores [Forsyth, D.A. y Ponce, J, 2003], especialmente usados en tecnología, los modelos más populares y usados son RGB/A, CMYK, HLS, e YUV

RGB. Estas siglas provienen del inglés *red*, *green*, y *blue*; esto es, *rojo*, *verde*, y *azul*, respectivamente. Este trío de colores intenta modelar el sistema visual humano. La mayoría de los monitores y televisores se basan en este modelo, debido a que están basados en la emisión de luz. Este modelo también se le atribuye la propiedad de *sistema aditivo*, ya que se añaden las intensidades para formar un color.

RGBA. Se trata del mismo modelo RGB, pero con otra propiedad: canal *alfa*. Este canal se usa como un índice de la transparencia en un píxel. Esto nos sirve a la hora de mezclar varios colores designados para un solo píxel. Este modelo se suele usar para crear efectos y técnicas visuales como: niebla, llamas, y objetos semi-transparentes: cristal, agua, vidrieras, etcétera.

CMYK. Las siglas representan, en inglés, *cyan*, *magenta*, *yellow*, y *black*; esto es, *cian*, *magenta*, *amarillo*, y *negro*, respectivamente. Los tres primeros colores son complementarios a los de RGB. Para obtener un color más vivo, se le añade la tinta negra también. Este modelo tiene la propiedad de *sistema sustractivo*, ya que se restan las intensidades a la luz.

HLS. En inglés, las siglas son *hue*, *lightness* o *luminance*, y *saturation*; esto viene a ser, *matiz*, *brillo*, y *saturación*, respectivamente. Este modelo se basa en lo empírico de nuestra percepción visual. En lugar de usar un modelo tricolor para formar otros colores y tonos, el modelo HLS se basa en tres propiedades que sirven para definir los colores que percibimos. El matiz es el color que solemos denominar: azul, violeta, rojo, dorado, etcétera. El brillo describe la intensidad y brillo de un tono de color..

YUV también escrito **YCbCr** o incluso **YPbPr**. Este modelo se basa en el modelo RGB, pero restringiendo y descomponiendo algunos valores del RGB. YUV se usa en señales de televisión y en equipos de vídeo y grabación como S-Vídeo, y MPEG-2 y por tanto DVD. Originalmente, la principal razón de usar este modelo fue para mantener compatibilidad con los televisores analógicos en blanco y negro, cuando se introdujeron los televisores en color.

5.5 Representación de nuestra realidad a través del color

Debemos aproximarnos a la imagen deseada usando varias técnicas visuales. En definitiva, estamos tratando de mostrar a nuestro sistema visual para que nuestra imagen aparente ser real. Dicho de otra forma, tenemos que visualizar una imagen de resolución infinita en un área de resolución limitada. Éste es el concepto principal de la representación gráfica. Todas nuestras técnicas visuales y manipulaciones de una o varias imágenes se basan en presentar una imagen mostrando al usuario lo suficientemente como para que la imagen final aparente ser el objeto real. Sin embargo, a veces debemos hacer sacrificios en nuestra tarea de crear ilusiones. Ya que existen limitaciones según el hardware de nuestro sistema gráfico, es posible que decidamos reducir la resolución por falta de memoria, tiempo de computación, falta de tiempo y recursos.

5.6 Modelar imágenes

Esto implica que tendremos que trabajar con valores numéricos y por consiguiente deberemos realizar cálculos para conseguir nuestra imagen [Francisco J. Portilla M, Tesis Doctoral]. Los valores numéricos que usaremos representan vértices, líneas, vectores, vórtices (vértices en tres dimensiones), y demás conceptos matemáticos, al igual que la manipulación de bits para colores, combinación de imágenes, etcétera. Siguiendo la misma lógica de la representación gráfica, debemos representar un objeto usando conceptos matemáticos. Este concepto de modelado se basa en la composición de objetos primitivos como vértices y líneas. Si queremos representar un cuadrado, tendremos que guardar información acerca de los vértices y líneas para poder recrear la imagen de un cuadrado.

Dibujamos una línea desde un vértice a otro representando un lado de este cuadrado. Luego, dibujamos una segunda línea desde el último vértice a uno nuevo; y así sucesivamente hasta dibujar los cuatro lados del cuadrado.
Acabamos de definir, en términos de objetos primitivos, el modelo de un cuadrado. Del mismo modo, podemos definir las características de un cubo: un conjunto de ocho vértices, cuatro aristas y seis caras o planos.

Con esta información, podemos crear una imagen desde cualquier punto de vista: de lado, desde arriba, abajo, etc. También podemos girar el objeto, mudarlo, cambiarlo de tamaño, y deformarlo, con tan sólo aplicar fórmulas a la información definida. Al tratar con modelos de objetos de dos o tres dimensiones o más, nos aproximamos a un nivel más matemático, y por tanto, el ordenador nos ayuda a realizar los cálculos requeridos.

5.7 Mapeo de imágenes

Debido a que estamos trabajando con conjuntos de píxeles para representar imágenes, a veces es necesario que la imagen represente dimensiones especiales; por ejemplo, se desea crear una imagen de un mapa. Requerimos que nuestro mapa nos dé información precisa. Por ello, cada línea en nuestro mapa debe representar una longitud determinada; por ejemplo, cada tramo de carretera equivale a 10 kilómetros. Si esto es cierto para todos los tramos de nuestra imagen, entonces nuestro mapa está a escala. Análogamente, podemos tener varias imágenes pero de distintos tamaños. Necesitamos combinar estas imágenes para que tengan la misma escala. Esto se denomina mapear, del inglés *mapping*, o cambiar de escala.

El cambio de escala también puede producirse debido al modelo matemático. Por ejemplo, si queremos representar una ecuación matemática: $y = 2x + 3$, podemos calcular los valores de y a partir de valores escogidos de x. Estas coordenadas se usarán para poder seleccionar un píxel que las represente. Sin embargo, no todos los valores van a ser posibles representar, según los valores escogidos. Realmente estamos hablando de representar una imagen en dimensiones matemáticas en un sistema gráfico de píxeles. Debemos realizar un cambio de escala y seguramente de coordenadas.

El mapeo es necesario al tratar con objetos de diferentes dimensiones y orientaciones. Se debe transformar el estado de todos los objetos a una base para que sean compatibles. En el ejemplo de la ecuación matemática, acabaríamos transformando las coordenadas cartesianas calculadas a coordenadas en píxeles.

5.8 Figuras geométricas

Uno de los principales objetos o entidades en el campo de la programación gráfica es la figura geométrica. Algunas figuras se usan más que otras, pero todas son útiles a la hora de representarlas gráficamente. Algunas API's gráficas contienen todas o algunas de las siguientes figuras geométricas.

5.8.1 Vértices

No son figuras geométricas como tales, pero forman la base de toda figura geométrica. En algunas ocasiones, nos interesa mostrar los vértices como puntos en la pantalla: v_0, v_1, v_2, ..., v_n.

5.8.2 Línea Recta

Matemáticamente hablando, no se trata de una línea recta, sino de un segmento. Aunque no se trate de una figura geométrica, como tal, la línea recta forma la base de los trazados de las figuras. Requerimos dos vértices para crear una línea o segmento: v_0, y v_1.

5.8.3 Triángulo

Esta figura geométrica también forma la base de la representación de objetos gráficos, especialmente a la hora de representar objetos en tres dimensiones: v_0, v_1, y v_2. Podemos crear un objeto complejo a partir de triángulos. Necesitamos tres vértices para formar un triángulo.

5.8.4 Abanico de Triángulos

Se trata de una serie de triángulos unidos entre sí con un vértice común como el centro del abanico. Como mínimo debe haber tres vértices para formar el primer triángulo: v_0, v_1, y v_2. Los siguientes triángulos se crean a partir de un vértice nuevo, el vértice central del abanico, y el vértice perteneciente al triángulo contiguo.

5.8.5 Tira de Triángulos

Se trata de otra serie de triángulos unidos entre sí para formar un cinta o tira. Como mínimo debe haber tres vértices para formar el triángulo inicial: v_0, v_1, y v_2. Los siguientes triángulos se forman a partir de un vértice nuevo y dos vértices que pertenecen al triángulo contiguo.

5.8.6 Rectángulo

Otra figura geométrica importante es el rectángulo que también se usa para dibujar cuadrados. Debido a su simetría, sólo requerimos conocer dos vértices, v_0, y v_1, que forman dos esquinas opuestas. En total, obtendremos cuatro números diferentes, con los cuales podemos averiguar las coordenadas de los otros dos vértices.

5.8.7 Cuadrilátero

Esta figura geométrica también es usada con frecuencia, tanto para representar objetos en dos como en tres dimensiones. Necesitamos cuatro vértices para formar el cuadrilátero, v_0, v_1, v_2, y v_3.

5.8.8 Tira de Cuadriláteros

Se trata de una serie de cuadriláteros unidos entre sí para formar un cinta o tira. Como mínimo necesitamos cuatro vértices para el cuadrilátero inicial: v_0, v_1, v_2, y v_3. Los siguientes cuadriláteros, en la tira, se forman con dos vértices nuevos y con dos vértices pertenecientes al cuadrilátero contiguo.

5.8.9 Polígonos

Generalmente, las figuras que nos interesan suelen ser irregulares y con muchos más vértices que 2, 3, ó 4. Además, algunas gráficas ofrecen la funcionalidad de describir figuras geométricas según una lista de vértices. Típicamente, la funcionalidad de crear polígonos requiere n vértices, donde automáticamente se une el último vértice, v_n, con el primero, v_0, para poder cerrar la figura y formar un polígono cerrado.

5.8.10 Elipse - Círculo

La elipse al igual que el círculo son otras figuras que nos pueden servir en ocasiones, dependiendo de la escena o "dibujo" que queramos crear. El círculo, o la circunferencia, si la preferimos, es un caso especial de la elipse. Algunos gráficos describen las elipses y círculos de formas diferentes. Existen principalmente dos maneras de describir tales figuras:

1. Punto central, c; y dos longitudes para ambos radios, r_1 y r_2; o
2. Un rectángulo, v_0 y v_1, que circunscribe la elipse o el círculo.

5.8.11 Arco

El arco es una línea curva que describe el perímetro total o parcial de una elipse o circunferencia. Como se trata de una línea, no podemos hablar de un área ni rellenarla. La mayoría de las gráficas ofrecen tal elemento gráfico con una de dos formas para su descripción:

1. Punto central, c; dos radios, r_1 y r_2; y dos ángulos, para describir el ángulo inicial, α, y el final, β
2. Un rectángulo, v_0 y v_1, que circunscribe el arco; y dos ángulos, para describir el ángulo inicial, α, y el final, β

5.8.12 Sector - cuña

Podemos crear cuñas o sectores que básicamente se parecen a un "trozo de tarta". Esta superficie se basa en una elipse, como el arco, mencionado previamente. Creamos el sector como el arco, pero uniendo cada extremo al centro de la elipse o círculo con un segmento. Existen dos formas populares para describir un sector:

1. Punto central, c; dos radios, r_1 y r_2; y dos ángulos, para describir el ángulo inicial, α, y el final, β
2. Un rectángulo, v_0 y v_1, que circunscribe el sector; y dos ángulos, para describir el ángulo inicial, α, y el final, β

5.8.13 Segmento elíptico

El segmento elíptico se basa en una elipse como el arco y el sector. Esta figura se parece a un sector, pero en lugar de unir cada extremo al centro con segmentos, unimos los extremos entre sí con un segmento. Existen dos formas comunes para describir un segmento elíptico:

1. Punto central, c; dos radios, r_1 y r_2; y dos ángulos, para describir el ángulo inicial, α, y el final, β;
2. Un rectángulo, v_0 y v_1, que circunscribe el segmento elíptico; y dos ángulos, para describir el ángulo inicial, α, y el final, β.

5.8.14 Curva de Bézier

Esta curva fue descubierta por Pierre Bézier, mientras trabajaba para la compañía de automóviles, Renault, en los años 1960. Para poder diseñar adecuadamente las superficies curvas de los automóviles modernos, necesitaba una forma matemática, pero sencilla. Encontró la solución con una ecuación paramétrica definida como ecuaciones cúbicas. Por esta razón, también se denomina esta curva como *curva de Bézier del tercer orden*, al usar una ecuación cúbica. Se describe esta curva con cuatro puntos: dos puntos indican el punto inicial, (x_0, y_0), y final, (x_3, y_3), de la curva, mientras que los otros dos sirven como puntos de control: (x_1, y_1) y (x_2, y_2). Estos puntos de control modifican la curva dependiendo de su posición relativa a los puntos iniciales y finales. Se puede pensar en estos puntos de control como si describieran una atracción, como la gravedad, tirando parte de la curva hacia su centro.

6.1 Introducción

El problema de la generación de un camino a seguir por un robot móvil para llegar de una posición inicial a otra final evitando obstáculos del entorno ha sido tratado ampliamente [Latombe, 1991], encontrándose soluciones eficientes tanto mediante técnicas algorítmicas como mediante evolutivas. La mayor parte de estos enfoques resuelve el problema en el espacio de configuración del robot (definido normalmente por su posición x, y y su orientación θ). En este caso la solución es una secuencia continua de configuraciones espaciales, esto es, una trayectoria en el plano que no colisione con los obstáculos y que conecte el punto inicial con el final.

Existen dos tipos básicos de enfoques para controlar la navegación de un robot móvil: técnicas globales y locales [Domingo Gallardo y otros]. En las técnicas globales; como son los métodos geométricos, la programación dinámica o los métodos de campo de potencial, se asume conocida la descripción geométrica del entorno en el que se va a mover el robot. Se trata de métodos potentes y eficaces para generar trayectorias a seguir como secuencias de comandos a ejecutar. Son métodos usados para el control de robot que trabajan en entornos sin ninguna variabilidad y que realizan tareas repetitivas en las que se conoce en todo momento el valor de todas sus variables de estado

Los métodos locales o reactivos consideran que el robot va a moverse en un entorno no conocido a priori y proporcionar una conducta estándar para reaccionar ante lecturas de los sensores del robot (evitar obstáculos, seguir paredes, entrar en puertas, alinearse con objetos). Estas conductas [Arkin, 1987] son aplicables en gran número de entornos distintos y se suelen usar junto con un control de alto nivel que se encarga de secuenciarlas.

6.2 Técnicas para el control local

Entre las propuestas de control local, cabe destacar el enfoque del histograma de campo vectorial [Borenstenin y Korem, 1989], el método de velocidad – curvatura [Simmons, 1996], el método de ventana dinámica y el método de esquemas motores[Arkin, 1987].

Método de control local	**Esquemas**
Histograma de campo de potencial	Avanzar – evitando – obstáculos
	ir – a – objeto
	entrar – por – puerta
Velocidad – curvatura	Avanzar – evitando – obstáculos
	ir – a – objeto
Ventana dinámica	Avanzar – evitando – obstáculos
	ir – a – objeto
Esquemas motores	Esquemas configurables

Tabla 6.2.1; esquemas de actuación susceptibles de ser implementados con cada uno de los métodos de control local.

Es importante indicar que todos los enfoques propuestos trabajan con información proporcionada por los sensores de rango de baja densidad y limitado alcance, provocando el problema denominado *aliasing perceptual*, en el que situaciones del robot distintas, en las que se deberían tomar acciones distintas, se soluciona con la misma percepción.

6.3 Mapa de colores

Para asignarle el valor de uno a los píxeles que forman el contorno del objeto fue necesario crear un mapa de colores. Tal mapa nos permite asociar un valor entero a un color determinado, se escogió el valor de uno a los píxeles que forman el contorno del objeto y el valor de cero a toda los píxeles que no son ocupados por el contorno del objeto. El procedimiento para realizar tal mapeo consiste en generar el intervalo de colores elegidos a partir de una fórmula, basta con comprobar el intervalo del entero dado, y generar el color deseado.

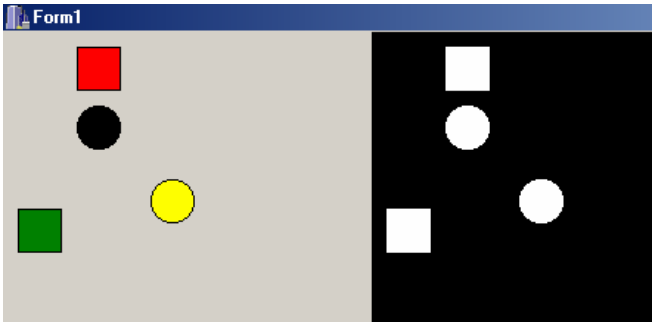


figura 6.3.1- asignación de valor de 1 a contorno de objeto

6.4 Algoritmo de movimiento del robot y detección de objetos

Se utiliza un algoritmo de líneas más efectivo para determinar las posiciones del píxel, creado por Bresenham [Donald Hearn, M. Pauline B, pp 59-75], halla las coordenadas enteras más próximas a la trayectoria real de la recta utilizando solamente aritmética entera. Empieza con una recta cuya pendiente es positiva y menor que uno. Las posiciones de los píxeles a lo largo de la trayectoria de la línea pueden trazarse después tomando etapas unitarias en la dirección x y determinado el valor de la coordenada y del píxel más cercano a la recta en cada etapa. Fue necesario ampliar el algoritmo a pendientes positivas mayores que uno intercambiando los papeles de las coordenadas x y y. Es decir, nos dirigimos por la dirección y en etapas unitarias y calculamos posiciones sucesivas de x. También para pendientes negativas, donde los procedimientos son similares excepto que ahora una coordenada decrece mientras la otra crece. Es decir, se implementan ocho etapas para el movimiento en cualquier dirección dentro del plano cartesiano comprendido desde el origen ubicado en la esquina superior izquierda $P_1(0,0)$ del monitor hasta el píxel $P_2(600, 500)$, ubicado en la esquina inferior derecha. En la figura 6.4.1 se nuestra la trayectoria curva que sigue el robot empleando el algoritmo de movimiento, para que el robot siga la trayectoria deseada es necesario introducir la función que describe su comportamiento. En este caso se introduce la ecuación matemática que describe la figura de la **limacom** ($r=b+a*\cos(\theta)$)

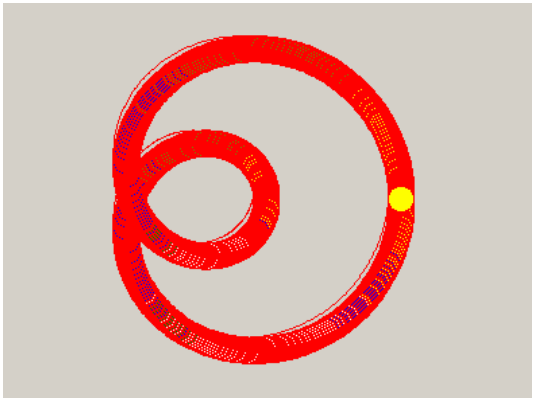


figura 6.4.1- limacom

6.5 Algoritmo de trayectoria con fronteras de objeto predefinidas

En este nuevo espacio, los obstáculos son definidos geométricamente, por lo que se limita la técnica algorítmica a seguir el mismo comportamiento sin importar la ubicación del objeto, como se muestra en la figura (6.5.1). Debido a que se imponen restricciones que reducen localmente la dimensionalidad del espacio ocupado por el obstáculo y cuyos datos son utilizados para general las restricciones lógicas y que a su vez se emplean para generar la trayectoria a seguir. Es importante mencionar que la primer característica de este algoritmo es que el robot se dirige en primera instancia hacia la frontera izquierda del obstáculo y a continuación sigue el recorrido de la frontera inferior, una vez concluido este recorrido inferior el robot se dirige hacia la esquina superior derecha en donde se determina la separación que a seguir para realizar el recorrido nuevamente hacia la frontera inferior y como ultimo dirigirse hacia el punto destino. Como se puede observar el uso de este algoritmo reduce la capacidad de movilidad del robot, debido a que la ruta más corta a seguir entre el punto de inicio el punto destino no se considera, además del movimiento de ida y regreso que realiza al detectar la frontera del lado derecho del obstáculo.

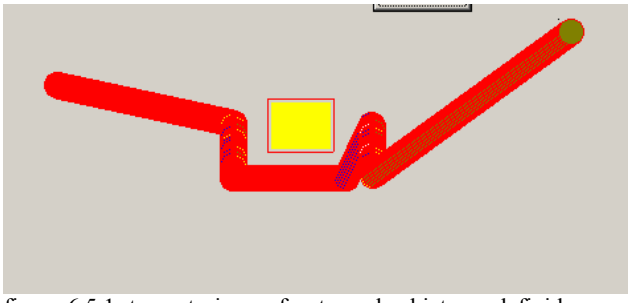


figura 6.5.1- trayectoria con fronteras de objeto predefinidas

6.6 Control local de robots móviles (curvas de Bézier)

El problema de control de un robot autónomo viene resolviéndose utilizando un control local de bajo nivel que proporciona competencias elementales como son: evitar obstáculos,

seguir paredes, entrar en puertas, etc. Y una capa superior que utiliza dichas funcionalidades para la navegación efectiva y para la solución de problemas completos. Por lo que se busca la obtención de técnicas eficientes para el control local que permitan obtener una navegación correcta. En las técnicas globales se asume totalmente conocida la descripción geométrica del entorno en el que se va a mover el robot. Se trata de métodos potentes y eficaces para generar trayectorias a seguir como secuencias de comandos a ejecutar. Son métodos utilizados para el control de robots que trabajan en un entorno sin ninguna variabilidad y que realizan tareas repetitivas en las que se conoce en todo momento el valor de todas sus variables de estado. Se busca que el robot proporcione una conducta estándar para reaccionar ante lecturas de los sensores del robot. Estas conductas son aplicables en gran número de entornos distintos y se suelen usar junto con un control de alto nivel que se encarga de secuencias. Con base en lo expuesto se realizó la siguiente propuesta que se presenta en este trabajo.

Entre las propuestas de control se utiliza: avanzar – evitando – obstáculo ir – a – objetivo. Hay que hacer notar también que este enfoque se trabaja con información proporcionada por los sensores de rango de baja densidad y limitado alcance. Esto provoca que en situaciones del robot distintas, en las que se debería tomar acciones también distintas, se resulten en una misma percepción. Debido a que es la primera etapa de desarrollo del proyecto **"Investigación y desarrollo de un sistema de control para robot movibles"** se considera que la información que proporciona los sensores ya se dispone y con base en la información la ubicación del obstáculo ya quedó definida.

Las variables de estado del robot móvil son su posición inicial (x1, y1), su posición final (x2, y2), el radio de detección (R) y su orientación (θ). Se utiliza una técnica que realiza la búsqueda en espacios, una vez realizada la detección se define una distancia mínima al objeto y se obtiene información que parten de una población inicial que representa un conjunto de puntos detectados, a continuación se utiliza **Regresión por mínimos cuadrados** [Steven C. Chapra, Raymond P.] se determina la ecuación de la recta que mejor representa la distribución del conjunto de puntos.

$$y = a_0 + a_1 x + \varepsilon \qquad (6.1)$$

$$a_1 = \frac{n\sum x_i y_i - \sum x_i \sum y_i}{n\sum x_i^2 - \left(\sum x_i\right)^2} \qquad (6.2)$$

$$a_0 = \bar{y} - a_1 \bar{x} \qquad (6.3)$$

Las rectas se utiliza para determina las fronteras del obstáculo en dirección vertical como horizontal y el punto centro del objeto, se compara la ubicación de ambas rectas con el punto centro del obstáculo y se determina la distancia más corta entre el robot y la frontera para determinar el lado por el cual el robot esquivará al obstáculo; así como, los puntos necesarios que se requieren para realizar el movimiento siguiendo l curva de Bézier entre el punto considerado de detección del robot y el punto destino.

La curva cúbica de Bézier [Bézier, P. 1972], es definida por cuatro puntos. Dos son puntos finales, (x_o, y_o) es el origen y (x_3, y_3) el destino. Los puntos (x_1, y_1) y (x_2, y_2) son puntos de

control. Son dos ecuaciones que definen los puntos sobre la curva. Ambas son evaluadas por un número arbitrario de valores entre 0 y 1, una ecuación define los valores en el dominio de x y la otra en el dominio de y. Con el incrementa en el valor de t se definen los puntos x(t), y(t) que permiten el movimiento de origen a destino. A continuación se presentan las ecuaciones que define el comportamiento de la Curva de Bézier.

$$x(t) = a_x t^3 + b_x t^2 + c_x t + x_0 \qquad (6.4)$$

$$x_1 = x_0 + c_x/3$$
$$x_2 = x_1 + (c_x + b_x)/3$$
$$x_3 = x_0 + c_x + b_x + a_x$$

$$y(t) = a_y t^3 + b_y t^2 + c_y t + y_0 \qquad (6.5)$$

$$y_1 = y_0 + c_y/3$$
$$y_2 = y_1 + (c_y + b_y)/3$$
$$y_3 = y_0 + c_y + b_y + a_y$$

Este método de definición puede emplear la ingeniería de reversa. Tal que, se obtiene el valor de los coeficientes basados en los punto que definen la curva.

$$c_x = 3(x_1 - x_0)$$
$$b_x = 3(x_2 - x_1) - c_x$$
$$a_x = x_3 - x_0 - c_x - b_x$$

$$c_y = 3(y_1 - y_0)$$
$$b_y = 3(y_2 - y_1) - c_y$$
$$a_y = y_3 - y_0 - c_y - b_y$$

A continuación se presenta un conjunto de cuatro gráficas que representan la trayectoria que sigue el robot al detecta un obstáculo rectangular y un obstáculo circular.

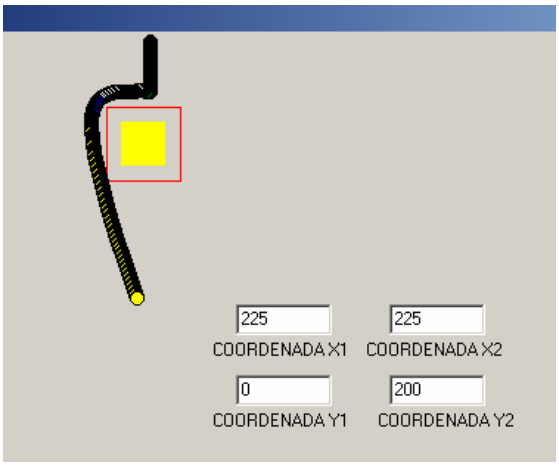


figura 6.6.1- trayectoria con fronteras de objeto predefinidas

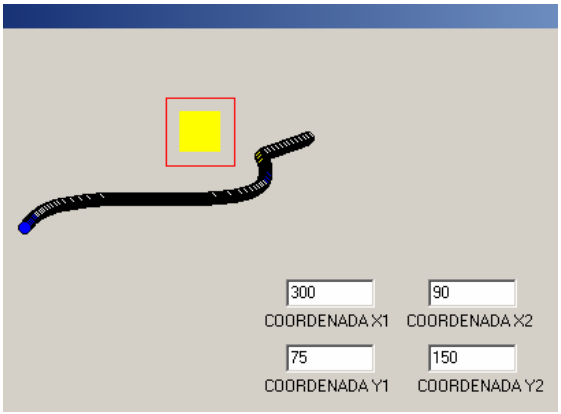


figura 6.6.2- trayectoria con fronteras de objeto predefinidas

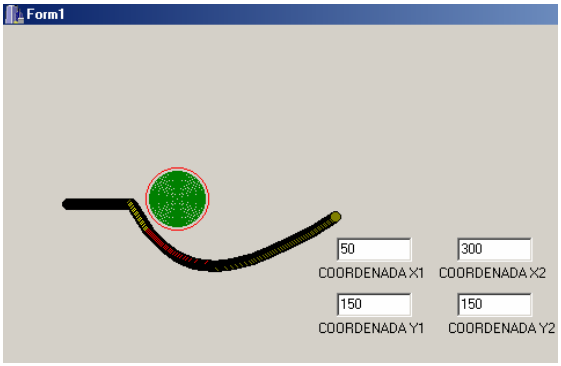


figura 6.6.3- trayectoria con fronteras de objeto predefinidas

6.7 Control local de robots móviles con varios obstáculos

Se retoma la propuesta de control donde se realizan los cálculos de distancia y orientación entre el punto inicial o ubicación actual del robot (x_in, y_in) y el punto meta (x_end, y_end), en punto intermedio durante la trayectoria de movimiento.

$$R\_in\_end = \sqrt{(x\_end - x\_in)^2 + (y\_end - y\_in)^2} \qquad (6.7.1)$$

$$Fi\_in\_end = a\sin\left(\frac{y\_end - y\_in}{R\_in - end}\right) \qquad (6.7.2)$$

Donde:
R_in_end es la distancia entre punto ubicación actual del robot y meta
Fi_in_end es la orientación entre punto ubicación actual del robot y meta

A continuación el robot realiza un barrido circular con radio R y alcance de 360^o con centro en el punto inicial, lo que hará posible la detección del objeto, a su vez determinar el valor de las coordenadas de intersección entre el haz y el objeto. Datos que se utilizaran para calcular la distancia y orientación de cada uno de los puntos detectados y el punto inicial del robot, empleando las siguientes ecuaciones.

$$R\_\det = \sqrt{(x\_\det - x\_in)^2 + (y\_\det - y\_in)^2} \qquad (6.7.3)$$

$$Fi\_\det = a\sin\left(\frac{y\_\det - y\_in}{R\_\det}\right) \qquad (6.7.4)$$

$$Ang\_SI = Fi\_\det - Fi\_in\_end \qquad (6.7.5)$$

Donde:
R_det es la distancia entre el robot y puntos detectados del objeto
Fi_det es la orientación ubicación actual del robot y puntos detectados del objeto
Ang_SI es la diferencia entre la orientación ubicación actual del robot y meta vs. ubicación actual del robot y puntos detectados del objeto

Una vez obtenida la información antes definida se calcula la distancia máxima y mínima entre el robot y los puntos detectados en el objeto, con la finalidad de determinar la distancia más próxima entre el robot y el objeto. Comparando la orientación entre el punto inicial y el punto meta vs. la orientación entre el punto inicial y el punto detectado. Con la finalidad de orientar al robot para el primer movimiento que realizará hacia el punto meta. Para lo cual se utilizó la siguiente fórmula:

$$Q = C1 * \cos(Ang\_SI) + C2 + \left(\frac{1}{R\_\det + R\min}\right) \qquad (6.7.6)$$

Donde:
C1, C2 y Rmin son constantes y adquieren los siguientes valores: 20, 10 y 1 respectivamente

El valor máximo de Q determina el valor de las coordenadas (x2_max, y2_max) más convenientes para que definir la distancia mínima que el robot requiere para esquivar el objeto y utilizando las ecuaciones (6.7.7 y 6.7.8) se define el punto coordenado para el primer movimiento.

$$X\_mid = x2\_\max - dp2 * \cos(Ang\_SI\_\max) \qquad (6.7.7)$$

$$Y\_mid = y2\_\max - dp1 * \sin(Ang\_SI\_\max) \qquad (6.7.8)$$

Donde:
dp1 y dp2 son constantes con los valores de 1 y 50 respectivamente.

Con estos valores se le indica al robot moverse en línea recta del punto inicial al primer punto intermedio. Una vez iniciado en primer movimiento el robot se detiene e inicia de nuevo el mismo proceso hasta haber llegado al objetivo planeado.

Como ilustración se muestra la trayectoria que sigue el robot del punto inicial P_in(30, 130) y punto meta P_end(450, 200) con seis obstáculos con coordenadas en los vértices:

Obstáculo	Coordenada Esquina superior izquierda	Coordenada Esquina inferior derecha
1	100, 118	150, 168
2	200, 70	250, 120
3	280, 50	330, 100
4	180, 130	230, 180
5	270, 130	320, 180
6	210, 195	260, 245

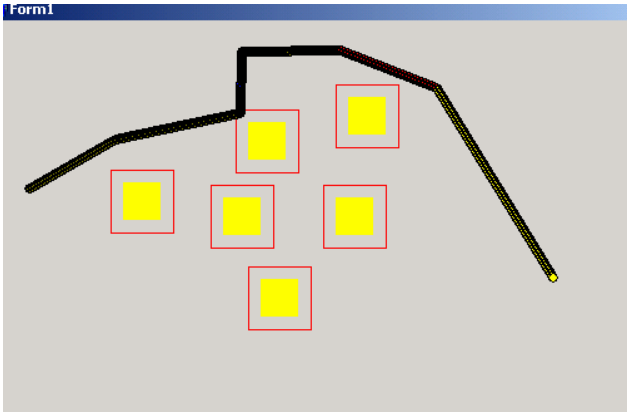


Figura 6.7.1- Trayectoria con varios obstáculos

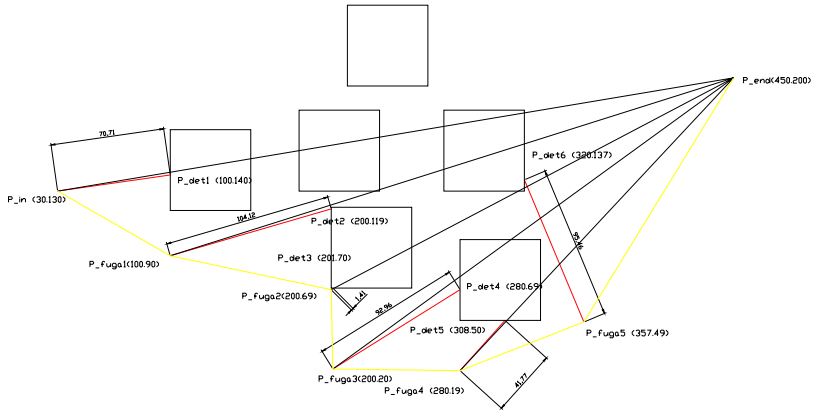


Figura 6.7.2 – trayectoria con puntos intermedios de movimiento con base en la información obtenida en la tabla 6.7.1

X_det	Y_det	X_mid	Y_mid	dmax	Qmax	Fi_det	Ang_SI
100	140	100	90	70.71	20.13	0.139	0.0255
200	119	200	69	104.12	20.08	0.279	0.0255
201	70	201	20	1.41	24.08	0.558	-0.075
280	69	280	19	92.96	20.06	0.558	0.067
308	50	357	49	41.77	20.22	0.837	-0.021
320	137	450	200	95.46	11.96	1.954	-0.935

Tabla 6.7.1- Datos obtenidos durante el movimiento del punto inicial P_in(30,130) al punto meta P_end(450,200)

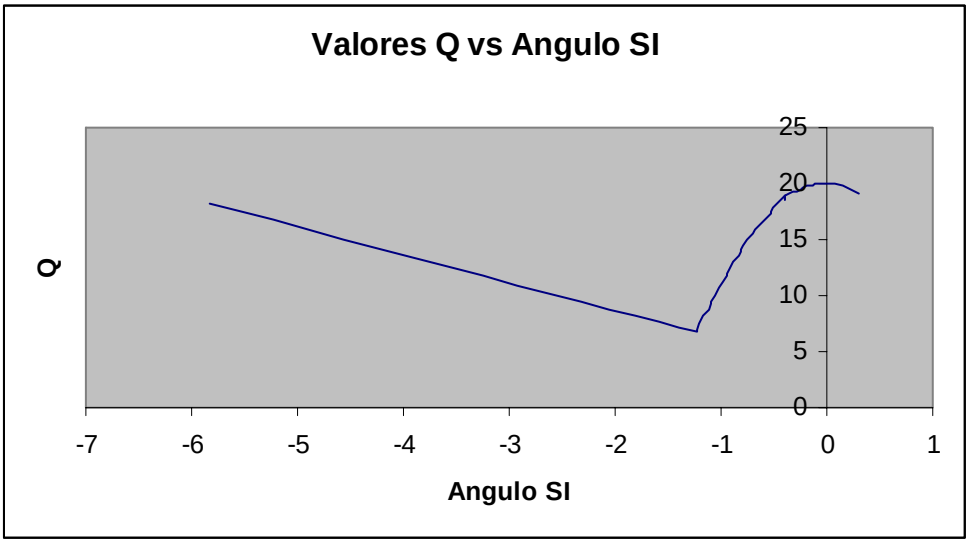


Figura 6.7.3- datos de Q respecto a Angulo SI, determinados durante el segundo movimiento

7.1 Descripción del movimiento para esquivar un objeto circular

Figura 7.1- El robot esquiva objeto de forma circular

El primer experimento que realizó el robot fue esquivar un objeto de forma circular con centro en las coordenadas (150, 150), diámetro de circunferencia sólida (color verde) de 20 y frontera (color rojo) de 30 unidades. El robot al inicio del movimiento se encuentra en el punto coordenado (X1=50, Y1=150) y como punto meta ó destino se encuentra en la coordenada (X2=250, Y2=150).

El movimiento que el robot tiene que realizar para esquivar el obstáculo es del lado izquierdo al derecho, como se muestra en la figura 7.1. La información proporcionada por el espacio ocupado por el obstáculo, punto de inicio de movimiento y punto destino se utilizan en los cálculos matemáticos para generar la trayectoria. Se inicia en primera instancia con el dibuja del obstáculo, como se observa en la figura 7.1 se emplean dos colores el de color verde para definir el área que ocupa el obstáculo y el color rojo para definir la frontera, para permitir que el robot disponga de un margen para el movimiento.

A continuación el procesador de despliegue para interactuar con la UCP y controlar la operación del dispositivo de despliegue emplea el sistema de rastreo al azar con la finalidad de que el programa de aplicación ofrezca información del contorno del obstáculo al procesador de despliegue en términos de nivel de intensidad de luz de puntos coordenados de la pantalla, considerando un 1 lógico a todos los puntos coordenados ocupados por el contorno del obstáculo y un 0 lógico para los restantes puntos coordenados. Los valores de los puntos coordenados (X_p, Y_p) ocupados por el contorno del obstáculo y que fueron identificados por el rastreados se muestran en la figura 7.2

Una vez definidos los puntos ocupados por el contorno del obstáculo y con ayuda de los niveles de intensidad de luz, se procede a calcular la distancia y orientación entre el punto inicio y destino, los valores calculados se indica la tabla 7.1

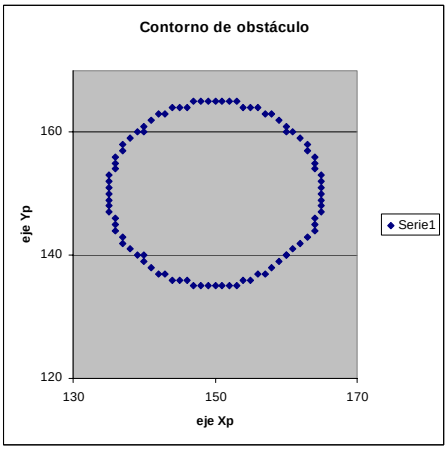


Figura 7.2.- puntos coordenados del contorno del obstáculo

Punto inicio	Punto destino	Distancia calculada	Orientación calculada
P_{in} (50, 150)	P_{end} (250, 150)	200 unidades	0 radianes

Tabla 7.1- datos de distancia y orientación obtenidos entre punto inicial y destino

A continuación se realiza un barrido circulas con radio de 200 unidades y con punto centro en la coordenada P_{in} (50, 150) y con incrementos de 3 grados, como se observa en la figura 7.3. En esta actividad se buscó la forma para que el barrido circular se inicie con la misma orientación entre el punto destino y la meta

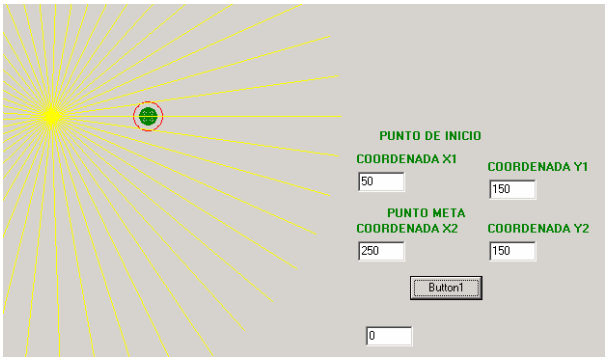


Figura 7.3- Detección radial de objeto

Con la finalidad de identificar y conocer los puntos coordenados que se presentaran durante el recorrido, la información proporcionada se emplea para determinar los puntos coordenados que definen la trayectoria que el robot seguirá, con la encomienda de que se deben evitar colisiones. Los puntos coordenados detectados por el robot en el primer ciclo de movimiento son los siguientes:

Puntos coordenados detectados												
X_det	135	165	141	142	143	154	155	156	142	143	146	153
Y_det	150	150	162	163	163	164	164	164	137	137	136	135
Orientación en radianes												
Fi_det	0	0	0.139	0.139	0.139	0.139	0.139	0.139	6.14	6.14	6.14	6.14

Tabla 7.2.- Puntos coordenados detectados

Se observa en los puntos detectados por el robot que; los dos primeros valores (X_det = 135, Y_det =150) y (X_det = 165, Y_det =150) corresponden al primer recorrido o movimiento radial realizado por el robot y con orientación de 0 radianes. Los siguientes puntos se detectan con orientación de 0.1396 y 6.1435 radianes, respectivamente, como se muestra en la figura 7.4

Los resultados obtenidos corresponden con los puntos de intersección entre la línea amarilla y la circunferencia roja, cabe mencionar que este primer apartado tiene como finalidad documentar los resultados obtenidos y con el apoyo de las figuras mostradas corroborar los mismos. Y a su vez, verificar que los resultados obtenidos correspondan con los esperados.

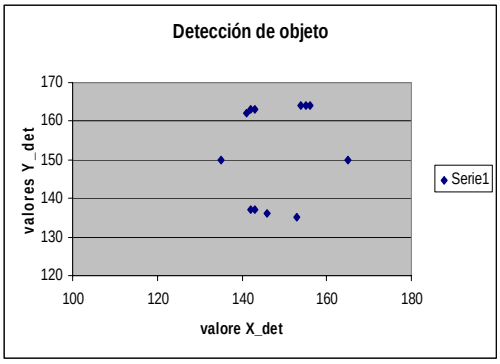


Figura 7.4- Puntos coordenados detectados del objeto

Se calcula la distancia y orientación entre el punto coordenado ocupa por robot – centro de la circunferencia que representa al robot – y los puntos identificados durante el recorrido o movimiento radial. En la tabla 7.3 se muestran los valores calculados.

Distancia calculada (unidades)												
R_det	85	115	91.78	92.91	93.40	104.93	105.92	106.92	92.91	93.90	97.01	104.08
Orientación calculada (radianes)												
Fi_det	0	0	0.139	0.139	0.139	0.139	0.139	0.139	6.14	6.14	6.14	6.14

Tabla 7.3- distancia y orientación entre el robot y puntos detectados

Una forma fácil de verificar, al menos los dos primeros cálculos, es restar la componente X1_det y X2_det de la componente Xc_robot y cuyo cálculo resulta.

X_1_det – X_c_robot = R_1_det (7.1)
X_2_det – X_c_robot = R_1_det (7.2)

Al sustituir valores en las ecuaciones (7.1) y (7.2) se obtienen los siguientes resultados

135- 50 = 85 (7.3)
165- 50 = 115 (7.4)

Los resultados obtenidos se deben a que se encuentra sobre la misma orientación y la distancia es la diferencia que existe entre ambos puntos. Respecto a la orientación calculada corresponde exactamente a los valores identificados durante el recorrido o movimiento circular. Por lo que, una vez más se corrobora que los resultados obtenidos por los cálculos y gráficas proporcionadas por software propuesto corresponden con lo esperado.

Hasta este momento el robot tiene que determinar cual es la distancia más próxima y cual es la más alejada entre el robot y los puntos identificados ocupados por el obstáculo; por lo que, se estima el valor de Q (ver ec. 6.7.6 A 6.7.8), condicionante para determinar el valor de distancia máximo y mínimo. Los valores obtenidos son:

Q_min = 20.16
d_min=85
Punto coordenado que determinan la distancia mínima

X_min = 135
Y_min= 150

Punto coordenado que determinan punto medio ó punto de fuga

X_mid=136
Y_mid= 125

Con la obtención de los datos descritos el robot está listo para realizar el primer movimiento, como se observa en la figura 7.5. Hasta el momento se ha descrito la secuencia de operaciones que el algoritmo realiza para determinar el primer movimiento. En seguida se inicia un nuevo ciclo; pero, ahora se introduce como valor inicial ó nuevo punto de inicio de movimiento el punto (X_mid, Y_mid) para calcular el nuevo punto fuga, reiniciando así hasta llegar al punto destino, se determina que el robot no llega exactamente al punto destino sino que se aproxima.

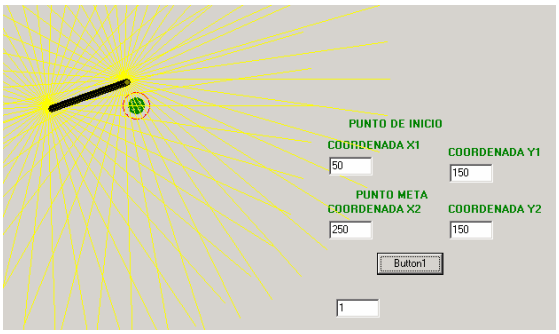


Figura 7.5- primer movimiento realizado por el robot

7.2 Descripción del movimiento para esquivar un objeto cuadrado

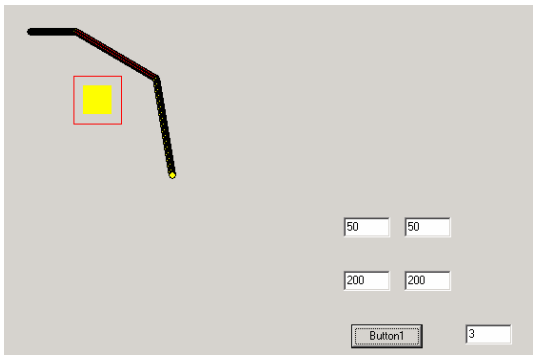


Figura 7.6- El robot esquiva objeto de forma cuadrada

El segundo experimento que realizó el robot fue esquivar un objeto de forma cuadrada con esquina superior izquierda en la coordenada (100,100), y esquina inferior derecha en la coordenada (150,150).

El robot al inicio del experimento se encuentra en el punto coordenado (50, 50) y punto meta ó destino en la coordenada (250, 150). Durante el recorrido el robot tiene que esquivar el obstáculo del lado izquierdo al derecho, como se muestra en la figura 7.6

Con la información proporcionada del obstáculo, punto de inicio de movimiento y punto destino, lo primero que el algoritmo realiza es dibuja el obstáculo y a continuación el procesador de despliegue para interactuar con la UCP y controlar la operación del dispositivo de despliegue, el sistema de rastreo informa del contorno del obstáculo al procesador de despliegue en términos de nivel de intensidad de luz de puntos coordenados de la pantalla, considerando un 1 lógico a todos los puntos coordenados ocupados por el contorno del obstáculo. Los valores de los puntos coordenados (X_p, Y_p) ocupados por el contorno corresponden a los puntos coordenados empleados para dibujar el obstáculo.

El contorno queda definido por 200 puntos coordenados; por lo que, se tiene pensado para el siguiente trabajo manejar bases de datos que permitan generar diferentes archivos con la información obtenida en diferentes experimentos e interactuar con diferentes alternativas de software, por ejemplo Excel y LabView. En especial LabView, debido a que es una herramienta útil para la adquisición y control de datos.

La datos de distancia y orientación calculada entre el punto inicio y destino, se muestra en la tabla 7.5. Se observa que se obtienen los mismos resultados obtenidos en el experimento 1 y es debido a que se introdujo la misma información, mismo punto de inicio y mismo punto destino

Punto inicio	Punto destino	Distancia calculada	Orientación calculada
P_{in} (50, 150)	P_{end} (250, 150)	200 unidades	0 radianes

Tabla 7.5- datos de distancia y orientación calculados entre punto inicial y destino

A continuación el robot realiza un barrido circulas con radio de 200 unidades con centro en el punto inicial P_{in} (50, 150) y con incrementos de 3 grados. Se obtiene 380 puntos detectados. Como sea podido apreciar: el objeto de forma cuadrada proporciona más información para los cálculos, ya que es mayor el perímetro ocupado

Hasta este momento el robot tiene que determinar cual es la distancia más próxima y cual es la más alejada entre el robot y los puntos identificados por el obstáculo; se determina el valor mínimo de Q, condicionante para determinar el valor de distancia máximo y mínimo. Los valores obtenidos son:

Q_min = 20.1960
d_min=50

Punto coordenado que determinan la distancia mínima

X_min = 100
Y_min= 100

Punto coordenado que determinan en punto medio ó punto de fuga

X_mid= 99
Y_mid= 20

Con base en la información obtenida el robot está listo para realizar el primer movimiento, como se muestra en la figura 7.7. Hasta el momento se ha descrito la secuencia de operaciones que el algoritmo realiza para determinar el primer movimiento. En seguida se itera un nuevo ciclo; pero, ahora se introduce como valor inicial ó nuevo punto inicio de movimiento el punto (X_mid, Y_mid) para determinar el siguiente punto fuga: así hasta aproximarse al punto destino.

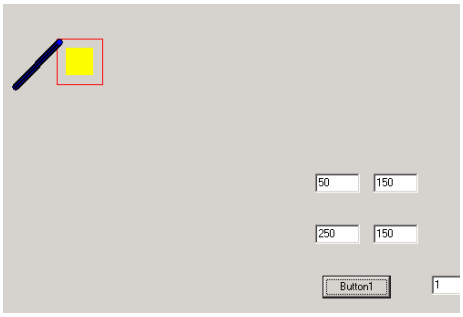


Figura 7.7- primer movimiento realizado por el robot

Hasta este momento los dos experimentos realizados arrojan casi la misma información y se puede decir que el robot se comporta de forma muy parecida.

7.3 Resultados obtenidos al esquivar un objeto rectangular con dimensiones mayores

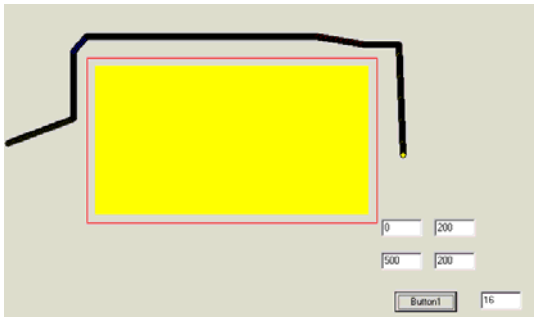


Figura 7.8- trayectoria para esquivar un obstáculo rectangular de dimensión mayor

El tercer experimento que realizó el robot fue para observar el comportamiento del robot y la trayectoria que se genera, se utiliza un obstáculo rectangular con esquina superior izquierda en la coordenada (100,100), y esquina inferior derecha en la coordenada (450,300).

El robot al inicio del experimento se encuentra en el punto coordenado (0, 200) y punto meta ó destino en la coordenada (500, 200), ver figura 7.8.
Durante el recorrido el robot tiene que esquivar el obstáculo del lado izquierdo al derecho. Para realizar el movimiento se generan 17 iteraciones y se registran los valores coordenados (X_mid, Y_mid), puntos coordenados que reciben el nombre de punto de fuga, como lo muestra la tabla 7.6

Puntos de fuga																
Componente X_mid																
80	80	80	80	96	168	224	267	300	326	346	361	373	432	473	478	440
Componente Y_mid																
170	141	114	88	70	70	70	70	70	70	70	70	70	80	80	214	113

Tabla 7.6- Puntos de fuga

Para comprobar los datos obtenidos se introduce la información y se obtiene la figura 7.9

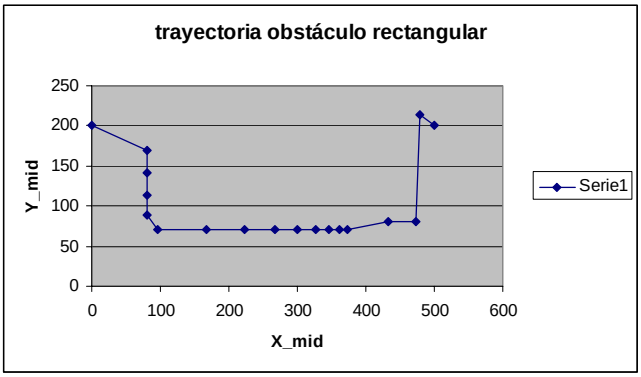


Figura 7.9.- primer movimiento realizado por el robot

A continuación se realizan varias pruebas para observar el comportamiento del robot, se inicia en la coordenada (0,0) como punto inicio y la coordenada (500,0) como meta. El robot realiza cuatro iteraciones para aproximarse al punto destino y termina el la coordenada (420, 90), como se observa en la figura 7.10

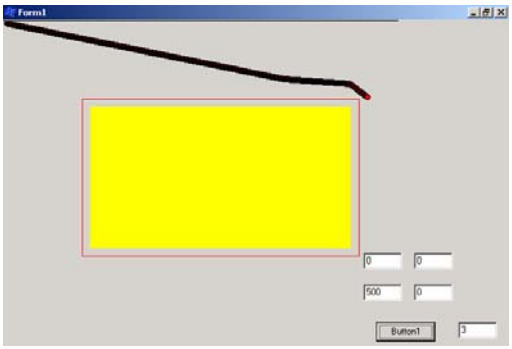


Figura 7.10.- Trayectoria P_in(0,0), P_end(500,0), cuatro iteraciones

Se continúa con las pruebas, sin mover el punto inicio y variando el punto meta con los siguientes valores coordenadas, ver tabla 7.7

Durante las diez pruebas realizadas el robot se comporta como se observa en la figura 7.11. El robot sigue el contorno del obstáculo como se definió con anterioridad. Pero cuando se posiciona en la coordenada (290,350) el robot rompe con la protección y ocupa el espacio ocupado por el área del obstáculo, como muestra la figura 7.12

P_in (X_in, Y_in)		P_end (X_end, Y_end)	
X_in	Y_in	X_end	Y_end
0	0	500	50
		500	100
		500	150
		500	200
		500	250
		500	300
		400	350
		350	350
		300	350
		291	350
		290	350

7.7.- Datos coordenados

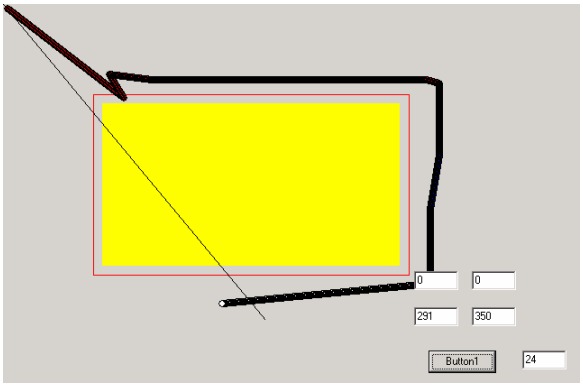


Figura 7.11.- Prueba 10

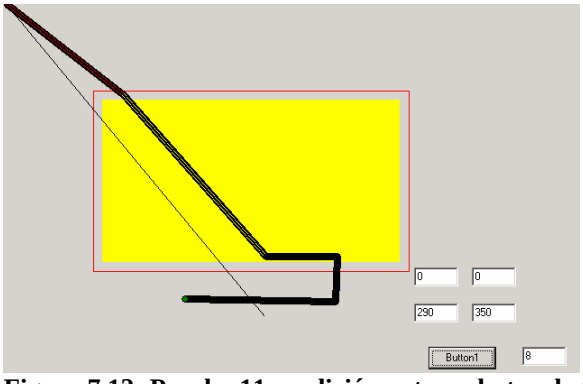


Figura 7.12- Prueba 11, coalición entre robot y obstáculo

7.4 Resultados obtenidos al esquivar un objeto circular con dimensiones mayores

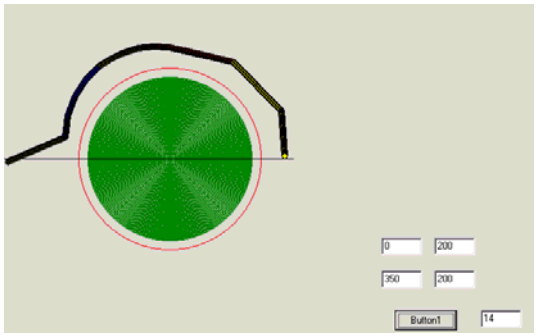


Figura 7.13 - trayectoria para esquivar un obstáculo circular de dimensión mayor

Durante el cuarto experimento que realizó el robot se observa el comportamiento del robot y la trayectoria que se genera, se utiliza un obstáculo circular con punto centro en la coordenada (200,200), y radio de 100 unidades

El robot al inicio del experimento se encuentra en el punto coordenado (0, 200) y punto meta ó destino en la coordenada (350, 200), ver figura 7.13

Durante el recorrido el robot tiene que esquivar el obstáculo del lado izquierdo al derecho. Para realizar el movimiento se generan 13 iteraciones y se registran los valores coordenados (X_mid, Y_mid), puntos coordenados que reciben el nombre de punto de fuga, como lo muestra la tabla 7.8

Puntos de fuga																
Componente X_mid																
70	73	82	95	111	127	140	158	177	195	213	273	331	335			
Componente Y_mid																
170	144	120	100	84	74	68	62	60	61	65	79	138	193			

Tabla 7.8- Puntos de fuga

Para comprobar los datos obtenidos se grafica la información calculada y se obtiene la figura 7.14

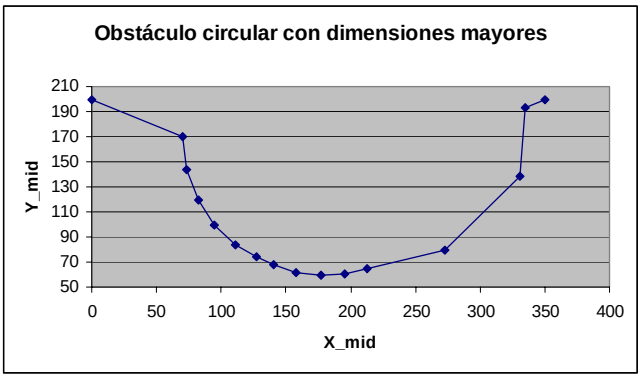


Figura 7.14 – gráfica (X_mid, Y_mid) objeto circular con dimensiones mayores

Se realizan varias pruebas para observar el comportamiento del robot. En la primera prueba se define como punto inicio la coordenada (0,0) y coordenada destino (400, 0), la trayectoria generada se muestra en la figura 7.15. Se realiza cuatro iteraciones para aproximarse al punto destino y termina en la coordenada (280, 118). Los puntos coordenados definidos como punto fuga son: P_{f1}(183, 62), P_{f2}(223, 75), P_{f3}(259, 100), P_{f4}(280, 118)

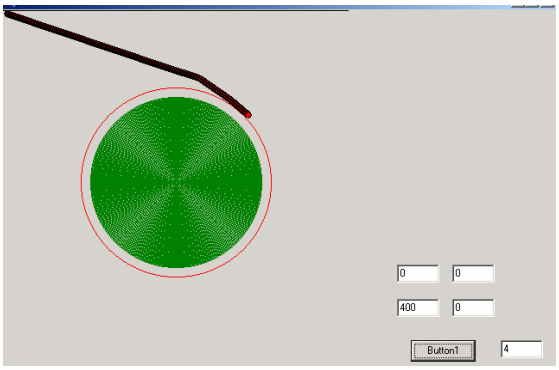


Figura 7.15 - Trayectoria P_in(0,0) P_end(400,0) con cuatro iteraciones

Se continúa con las pruebas, sin mover el punto inicio y variando el punto meta con los siguientes valores coordenadas, ver tabla 7.9

P_in (X_in, Y_in)		P_end (X_end, Y_end)	
X_in	Y_in	X_end	Y_end
0	0	400	50
		400	100
		400	150
		400	200
		400	250
		400	300
		400	350
		391	350
		390	350

7.9.- Datos coordenados

Se realizan nueve pruebas, la trayectoria generada se comporta como se observa en la figura 7.16. El robot sigue el contorno del obstáculo como se definió con anterioridad. Pero cuando se posiciona en la coordenada (390,350) el robot rompe con la protección y ocupa el espacio ocupado por el área del obstáculo, como muestra la figura 7.17

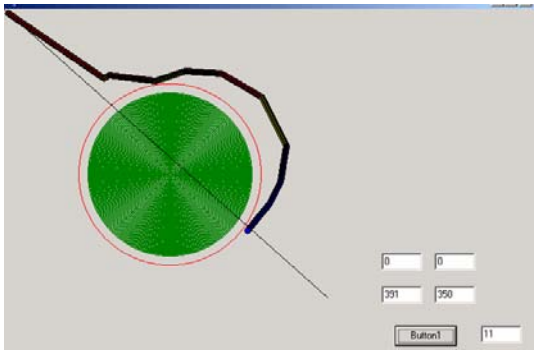


Figura 7.16 - Trayectoria P_in(0,0) P_end(400,0) con cuatro iteraciones

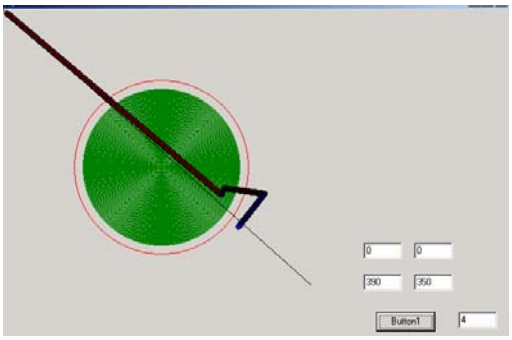


Figura 7.17- Prueba 9, coalición entre robot y obstáculo

La orientación para el movimiento de la figura 8, es 41.83^{o} y para el de la figura 9, es de 41.9^{o} sólo una diferencia de 0.07^{o} hace que el comportamiento cambie drásticamente.

Durante la elaboración del algoritmo son varias las pruebas que sean realizado y se a podido distinguir con claridad que no se presenta ningún problema cuando se le indica al robot que el punto inicio sea perpendicular a una de las caras del obstáculo y como punto destino la cara opuesta al objeto, ver figuras (7.18 a 7.20). Los problemas comienzan cuando se indican movimientos en las esquinas del obstáculo. Qué significa, que el robot desconoce como actuar bajo ciertas condiciones; por lo que, es necesario adicionar más instrucciones en la toma de decisión que permitan que el robot se nueva en ambos sentidos y no solamente movimientos con el sentido de las manecillas del reloj

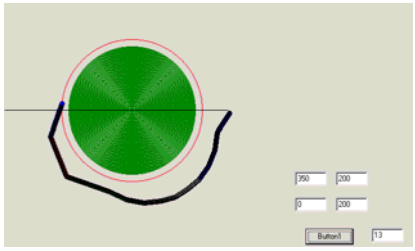


Figura 7.18- movimiento de derecha a izquierda

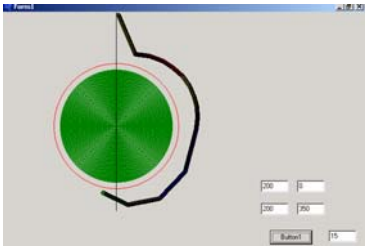


Figura 7.19- movimiento de arriba hacia abajo

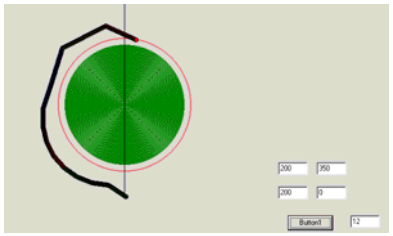


Figura 7.20- movimiento de abajo hacia arriba

7.5 Resultados obtenidos al esquivar dos obstáculos, un circular y otro rectangular

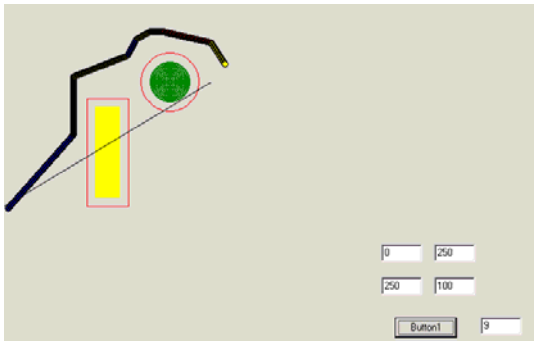


Figura 7.21- Trayectoria para esquivar dos obstáculos

Se pensaría en este apartado que se repiten los experimentos antes descritos, y en cierta forma sí, ya que el comportamiento que realiza el robot seguirá lo antes descrito. Lo que se pretende es realizar varias pruebas para observar el comportamiento que presenta al cambiar simplemente el número de obstáculos, y a su vez, utilizando las dos geometrías, circular y rectangular para definir el contorno del obstáculo, ver figura 7.21

El obstáculo rectangular se ubica en las coordenadas (100, 120) esquina superior izquierda y en la coordenada (150, 250) esquina inferior derecha, la circunferencia queda definida como punto centro la coordenada (200, 100) y radio 50. El movimiento inicia en el punto coordenado P_in (0, 250) y punto destino P_end (250, 100), los puntos coordenados que definen la trayectoria se muestran en la tabla 7.10.

Sin problemas para esquivar obstáculos									
X_mid									
80	80	80	145	156	172	188	246	263	179
Y_mid									
160	123	90	65	45	36	36	49	75	226

Tabla 7.10 – puntos coordenados que definen trayectoria correspondiente figura 7.21

A continuación se deja fijo el punto inicio y se varía el punto destino para observar el comportamiento, los diferentes puntos destino que se emplean se indican en la tabla 7.11.

P_in (X_in, Y_in)		P_end (X_end, Y_end)	
X_in	Y_in	X_end	Y_end
0	250	250	0
		250	50
		250	75
		250	100
		250	125
		250	150
		250	170
		250	174
		250	174

Tabla 7.11 – puntos inicio y destino para diferentes pruebas realizadas

Durante las primeras siete pruebas el robot no presenta ningún problema para esquivar los obstáculos, hasta que se define como punto destino el punto coordenado P_end (250, 173), con dificultad logra esquivar el obstáculo circular, ver figura 7.22. Los puntos coordenados que definen la trayectoria se muestran en la tabla 7.12

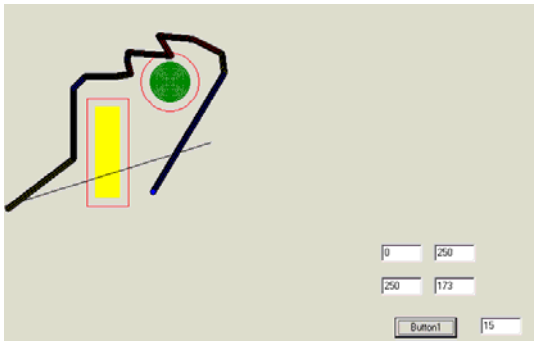


Figura 7.22- Inicia problema para esquivar segundo obstáculo

Inicia problemas para esquivar segundo obstáculo P_in (0,250) P_end (250, 173)															
X_mid															
80	80	80	80	94	130	149	145	146	198	183	225	259	262	176	201
Y_mid															
189	157	129	104	90	90	87	71	61	65	41	45	62	86	229	138

Tabla 7.12 – puntos coordenados que definen trayectoria antes de invadir área del segundo obstáculo

Se define el siguiente punto destino en el punto coordenado P_end (250, 174) el robot se comporta como los dos casos antes descritos: rompe con la frontera del objeto e invade el área del obstáculo circular, ver figura 7.23. Los puntos coordenados que definen la trayectoria se muestran en la tabla 7.13

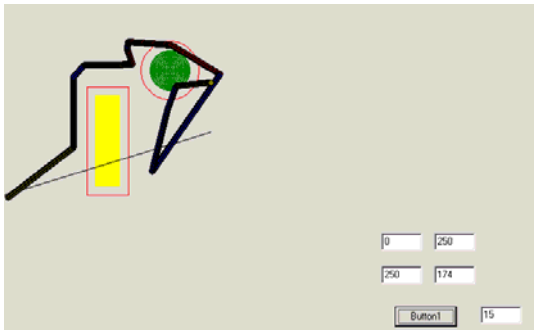


Figura 7.23- Se invade área del obstáculo

Inicia problemas para esquivar segundo obstáculo P_in(0,250) P_end(250, 174)															
X_mid															
80	80	80	80	93	129	151	145	146	198	257	175	193	204	246	173
Y_mid															
189	157	129	104	90	90	89	71	62	66	101	128	145	115	111	230

Tabla 7.13 – puntos coordenados que definen trayectoria cuando invadir área del segundo obstáculo

Como se observa en las figuras el primer obstáculo – de forma rectangular – no sufre ninguna invasión, sólo el obstáculo circular. Al analizar los datos obtenidos en las tablas 7.12 y 7.13 se observa que: en la iteración número 11 los valores cambian bruscamente.

Al calcula la pendiente entre puntos coordenados que da origen al cambio brusco y punto destino obtenemos los siguientes valores.

$\theta_1 = 64.08^o$
$\theta_2 = 64.29^o$

Con mínima diferencia en la orientación, 0.21^o, causa cambios bruscos en el movimiento del robot

7.6 Resultados obtenidos al moverse dentro de un pasillo

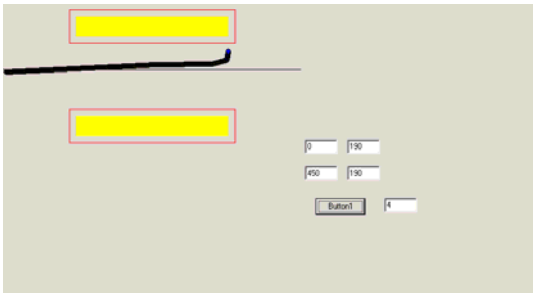


Figura 7.24- Movimiento dentro de un pasillo

Se hace una representación del movimiento que realizaría el robot dentro de un pasillo; por lo que, se representa el pasillo con dos obstáculos rectangulares alineados y se parados por una distancias de 100 unidades, el primer obstáculo se localiza en la coordenada (100, 100) esquina superior izquierda y la coordenada (350, 150) esquina inferior derecha, ver figura 7.24. El movimiento inicia en la coordenada (0, 190) y termina en la coordenada (450, 190), como se observa en la figura 7.24 no presenta problemas el cruzar el pasillo. A continuación se varía la distancia que existe entre obstáculos y para ello se deja fijo el obstáculo superior y se mueve el obstáculo inferior, buscando que se aproximen hasta que interfiera con la distancia de movilidad que se define para el movimiento del robot que es de 30 unidades. Los puntos coordenados que definen las nuevas posiciones del obstáculo en movimiento se muestran en la tabla 7.14

esquina superior izquierda		esquina inferior derecha	
X_{eq1}	Y_{eq1}	X_{eq2}	Y_{eq2}
100	200	350	250
100	185	350	235
100	180	350	230
100	175	350	225

Tabla 7.14 – puntos coordenados que definen movimiento del segundo obstáculo

Para la segunda prueba se define como punto de inicio la coordenada (0, 175) y como punto destino la coordenada (400, 174), ver figura 7.25. Durante el recorrido dentro de pasillo se acortó la distancia entre obstáculos a 50 unidades, la separación permite la maniobralidad del robot; pero, se observa que se desvía hacia la pared del obstáculo inferior y trata de

compensar y se dirige hacia la pared superior, hasta el momento no se presentan problemas. Pero, que pasa al acortar la distancia de separación a 35 unidades, que es la tercera prueba, donde se define como coordenada de inicio (0, 167) y coordenada meta (400, 165), ver figura 7.26.

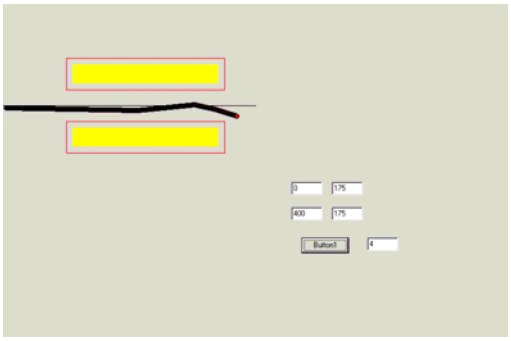


Figura 7.25- segunda prueba de movimiento dentro de un pasillo

En el primer movimiento el robot se dirige hacia la protección del obstáculo inferior, trata inmediatamente compensar y se dirige hacia la protección del obstáculo superior comienza a perder control sobre la trayectoria y realiza tres oscilaciones sin control.

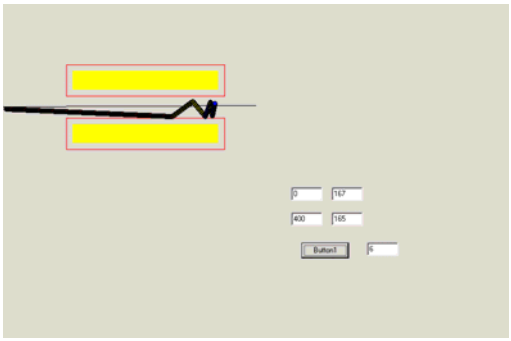


Figura 7.26- Tercera prueba de movimiento dentro de un pasillo

En la cuarta prueba la distancia de separación es de 30 unidades, es la misma distancia de maniobralidad que se asignó para el movimiento del robot. Se define como coordenada de inicio (0, 165) y coordenada meta (400, 165), ver figura 7.27

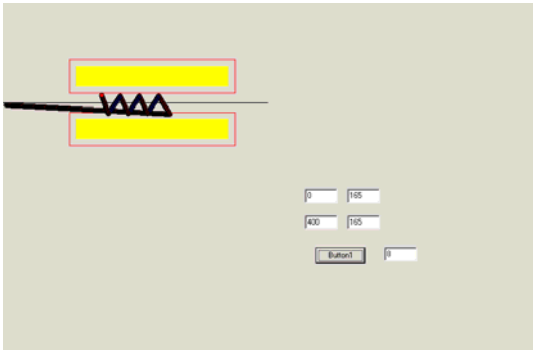


Figura 7.27- Cuarta prueba de movimiento dentro de un pasillo

El robot pierde el control comienza a oscilar y en vez de dirigirse hacia el punto destino retrocede la distancia que había avanzado, es un resultado de esperar ya que no existe el espacio suficiente para actual con libertad de movimiento.

En la quinta prueba se define como distancia de separación 25 unidades, con coordenada de inicio (0, 165) y coordenada meta (400, 165), ver figura 7.26

El robot rompe con la protección e invade el área del obstáculo superior, en condiciones reales el robot se impacta con el obstáculo desde el inicio del pasillo y continuaría ejerciendo fuerza contra los obstáculos, lo que ocasionaría daños al robot y a la instalación. Bajo estas condiciones es necesario acondicionar los interiores para que pueda moverse con liberta de movimiento

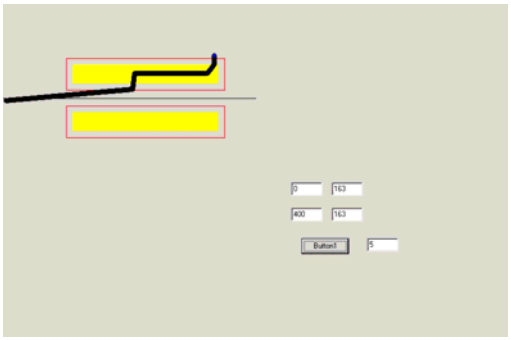


Figura 7.28- Quinta prueba de movimiento dentro de un pasillo

7.7 Resultados obtenidos al moverse dentro de varios obstáculos

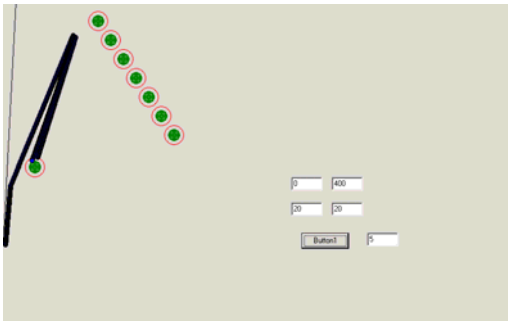


Figura 7.29- Primera prueba de movimiento dentro de varios obstáculos

Como última serie de experimento se realizan 11 pruebas, las 7 primeras pruebas el robot se mueve desde la parte inferior hacia la parte superior, en la tabla 7.15 se listan los valores coordenados que definen el punto inicio y el punto meta.

Punto inicio		Punto meta	
X_in	Y_in	X_end	Y_end
0	400	20	20
50	400	20	20
100	400	20	20
150	400	20	20
200	400	20	20
300	400	20	20
400	400	20	20

Tabla 7.17- valores coordenados de las primeras 7 pruebas

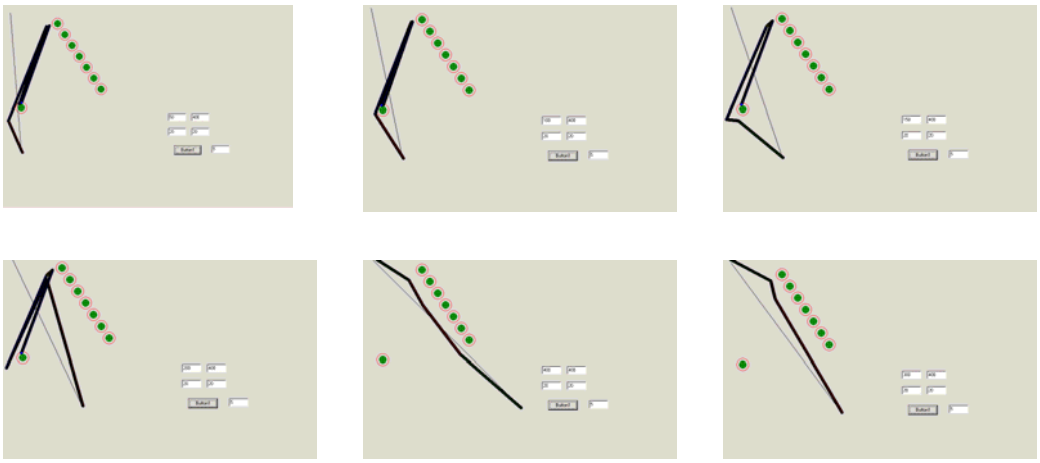


Figura 7.30- Prueba de movimiento dentro de varios obstáculos

Como se puede observar en las tres primeras figuras el robot esquiva el obstáculo que se encuentra en la esquina inferior izquierda, a continuación se dirige al obstáculo superior y comienza a oscilar entre ambos obstáculos. En la figura cuatro, el robot se dirige al obstáculo de la parte al obstáculo de la parte superior; pero, permanece oscilando entre ambos obstáculos. Se diría que lo único que cambio fue la trayectoria que seguía hacia el obstáculo que se ubica en la parte inferior

En la figura cinco y seis, la trayectoria cumple con lo requerido: se mueve desde el punto inicio al punto destino sin impacto con obstáculos y sin oscilaciones. Después de la realización de las pruebas podemos concluir que el algoritmo desarrollado cumple en forma parcial con lo requerido en la generación de trayectoria. Ya que bajo ciertas circunstancias se comporta "bien", pero en otras "mal"

Por lo que, se tiene que elaborar un nuevo planteamiento en el que se le den instrucciones de actuar bajo circunstancias que hagan en lo posible que invada áreas de obstáculos y que defina una distancia mínima al moverse entre obstáculos. El algoritmo desarrollado hasta el momento cumple con las expectativas sólo falta planear la estrategia de mejora. La pregunta es ¿cómo solucionar el problema? y pensé si el algoritmo de líneas de Bresenham ha servido para generar las geometrías, detección de frontera de obstáculo y movimiento del robot en el plano cartesiano -¿será posible utilizarlo para determinar una distancia mínima de protección y realizar el ciclo completo de movilidad del robot punto por punto?- los resultados obtenidos se muestran a continuación

7.8 Resultados obtenidos al moverse dentro de varios obstáculos, con algoritmo modificado

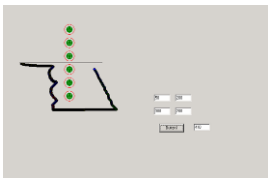


Figura 7.31.1

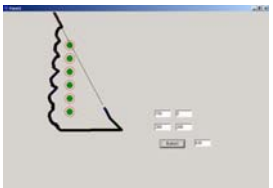


Figura 7.31.2

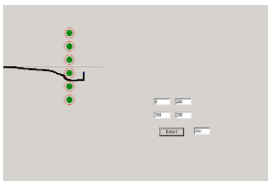


Figura 7.31.3

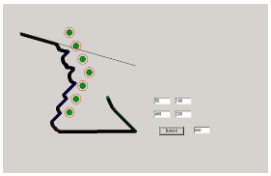


Figura 7.31.4

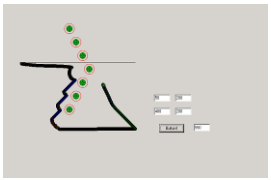


Figura 7.31.5

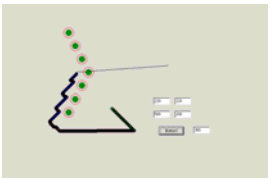


Figura 7.31.6

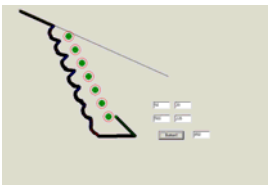


Figura 7.31.7

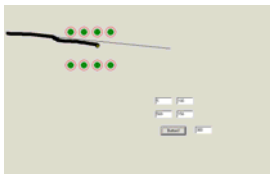


Figura 7.31.8

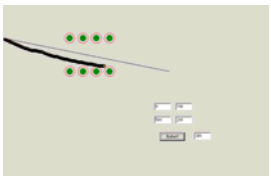


Figura 7.31.9

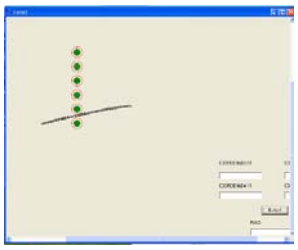
Figura 7.31- Prueba de movimiento dentro de varios obstáculos, con algoritmo modificado

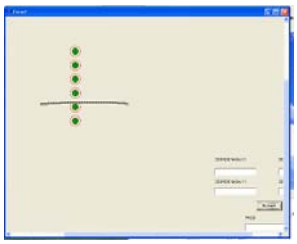
Se observa en la figura 7.31, varias pruebas realizadas con diferentes conjuntos de arreglos. Donde se destaca las condiciones antes mencionadas: se define una distancia de proximidad a obstáculos, con condiciones de movilidad que hacen posible que el robot analice sí es posible cruzar entre objetos o regresar a otra posición diferente cuando se llegue a la distancia de proximidad y que a su vez permita realizar una nueva operación para corregir trayectoria hacia punto destino. En el caso de ser posible cruzar entre objetos se hará, como se muestra en la figura 7.31.6

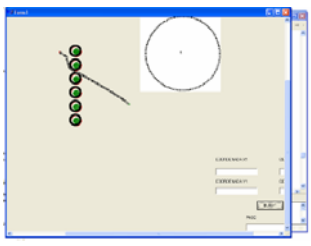
Los detalles se pueden ver en el listado de programa que se encuentra en el anexo A

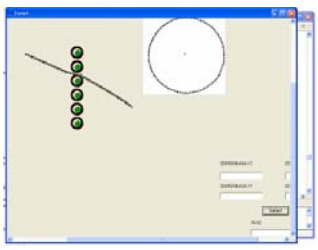
7.9 Resultados obtenidos, propuesta de la Dra. Tatiana

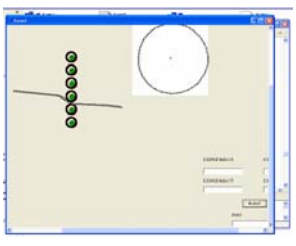
A continuación se muestra una recopilación de los resultados obtenidos con el algoritmo propuesto por la Dra. Tatiana

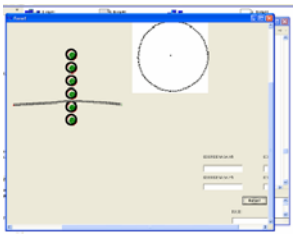


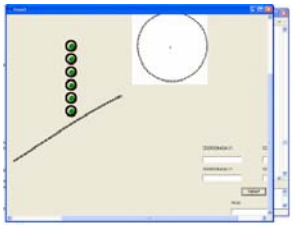


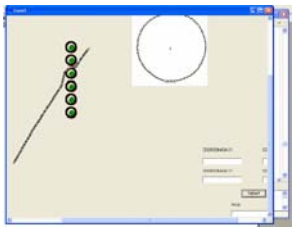


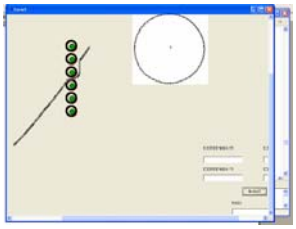


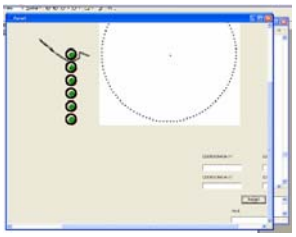


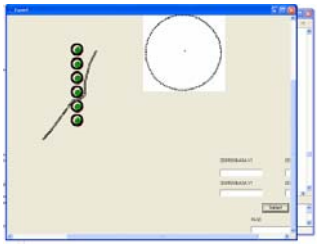


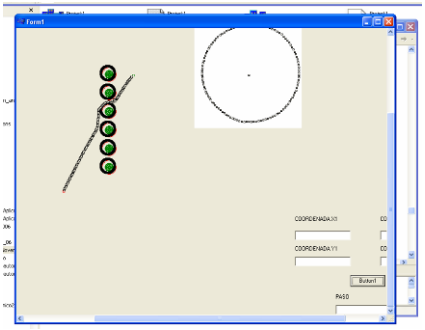












CONCLUSIONES

Se ha presentado un enfoque en el que es posible manejar y adecuar esquemas locales de conducta que guían la navegación de un robot móvil, aplicándose al ejemplo concreto de avanzar evitando obstáculos. Se ha llevado a cabo utilizando técnicas de algoritmos para detección y generación de trayectoria.

En este trabajo se puede resaltar como puntos importantes los siguientes:

1. Se adecua e implementa el algoritmo de movimiento y detección de obstáculos, creado por Bresenham. Para los cuatro cuadrantes que conforman el espacio de movimiento del robot. Se realizaron diferentes pruebas y los resultados obtenidos son satisfactorios.

2. Los sensores no sólo detectan la presencia o ausencia de un objeto sino que también se emplea la información que proporcionan para medir la distancia a la que se encuentra el obstáculo con respecto al robot, determinando así la cercanía o alejamiento de un obstáculo y no sólo su presencia sino también se busca optimizar la ruta. Con el uso de regresión lineal por mínimos cuadrados se puede calcular las dimensiones del obstáculo cuando sean objetos rectangulares y en el caso de objetos no simétricos se puede determinar el máximo y mínimo de sus dimensiones.

3. Se ha abordado el problema de la generación local de caminos aplicado al caso concreto del sorteo de obstáculos bajo el reconocimiento del entorno en cada punto intermedio de movimiento sobre la trayectoria que ha de seguir el robot. Esta problemática resulta frecuente en muchos casos debido a que el conocimiento del entorno es, en ocasiones, incompleto.

4. El método propuesto permite construir un camino admisible. Para ello, emplea movimientos rectilíneos o curvas de tipo Bézier. Por otra parte, se han definido tres propuestas factibles de utilizar en la generación de trayectorias en robot móviles.

5. El presente trabajo sirvió para identificar los 64 nodos, cada uno de los cuales corresponde a una de las posibles direcciones de movimiento, planteamiento que el Dr. Ernst describe en el artículo “El modelo de laboratorio el niño”. Este planteamiento es el rumbo que debe seguir el proyecto, debido a que se comprobó que bajo ciertas circunstancias el robot libra el ó los obstáculos sin altercados y en otras invade el área del obstáculo. Esto es debido a que se utiliza un solo criterio para resolver la generación de trayectorias, este criterio es útil para esquivar obstáculos que se presentan durante el recorrido y el robot tiene que cruzar de un lado hacia el otro lado del obstáculo.

6. Dentro de las siguientes posibles direcciones de movimiento se deben considerar: el movimiento del robot sin que existan obstáculos durante el recorrido, el movimiento del robot para moverse en direcciones opuesta al obstáculo, - en este apartado se contempla en movimiento en dirección perpendicular, paralela e inclinada-, el movimiento entre obstáculos, etc.

Propuesta para trabajos futuros

Como trabajos futuros se plantea modificar la función de evaluación para generar diferentes trayectorias que correspondan a esquemas de evaluación local, como es el seguimiento de paredes, o la navegación por el centro de pasillos, con la finalidad de utilizar estas trayectorias obtenidas como muestra para la generación de un control local que responda a estos esquemas y a su vez considerar al robot como un objeto no puntual.

También se plantea la construcción de un sensor óptico que permita la detección de los objetos, utilizar interfases que permitan que el computador reciba la información que proporciona el sensor y se realice el registro de ubicación de los mismos en el plano cartesiano delimitado por el monitor de la computadora y que, a su vez, permita corroborar la trayectoria generado para esa situación en especial. Añadir sensores en la parte trasera del robot con el fin de que pueda navegar marcha atrás sin peligro de colisión

APENDICE “A”

```
{  //****************INICIO DE PROGRAMA***********
{  // INICIO CIRCUNFERENCIA 1
       //CONTORNO
              //CIRCUNFERENCIA
  int cir, y_center,x_center;
  int prop, x_cir, y_cir, radius, k;
  x_center =200;   y_center=100;   cir=20;

  x_cir=0;
  y_cir=5+cir/2;
  prop=3-2*y_cir;
  while (x_cir<y_cir)
  {
   Canvas->Pixels[x_center+x_cir][y_center+y_cir]=clRed;
   Canvas->Pixels[x_center-x_cir][y_center+y_cir]=clRed;
   Canvas->Pixels[x_center+x_cir][y_center-y_cir]=clRed;
   Canvas->Pixels[x_center-x_cir][y_center-y_cir]=clRed;
   Canvas->Pixels[x_center+y_cir][y_center+x_cir]=clRed;
   Canvas->Pixels[x_center-y_cir][y_center+x_cir]=clRed;
   Canvas->Pixels[x_center+y_cir][y_center-x_cir]=clRed;
   Canvas->Pixels[x_center-y_cir][y_center-x_cir]=clRed;

   if (prop<0)
    prop= prop + 4*x_cir + 6;
  else
   { prop= prop + 4*(x_cir-y_cir)+10;
     y_cir=y_cir-1;}
     x_cir=x_cir+1;
   }
  if (x_cir=y_cir)
  {
   Canvas->Pixels[x_center+x_cir][y_center+y_cir]=clRed;
   Canvas->Pixels[x_center-x_cir][y_center+y_cir]=clRed;
   Canvas->Pixels[x_center+x_cir][y_center-y_cir]=clRed;
   Canvas->Pixels[x_center-x_cir][y_center-y_cir]=clRed;
   Canvas->Pixels[x_center+y_cir][y_center+x_cir]=clRed;
   Canvas->Pixels[x_center-y_cir][y_center+x_cir]=clRed;
   Canvas->Pixels[x_center+y_cir][y_center-x_cir]=clRed;
   Canvas->Pixels[x_center-y_cir][y_center-x_cir]=clRed;
  }
     // NUCLEO SOLIDO

 for ( k=1; k<(cir/2); k++)
 {
   x_cir=0;
   y_cir=k;
   prop=3-2*y_cir;
  while (x_cir<y_cir)
   {
    Canvas->Pixels[x_center+x_cir][y_center+y_cir]=clGreen;
    Canvas->Pixels[x_center-x_cir][y_center+y_cir]=clGreen;
    Canvas->Pixels[x_center+x_cir][y_center-y_cir]=clGreen;
    Canvas->Pixels[x_center-x_cir][y_center-y_cir]=clGreen;
    Canvas->Pixels[x_center+y_cir][y_center+x_cir]=clGreen;
    Canvas->Pixels[x_center-y_cir][y_center+x_cir]=clGreen;
    Canvas->Pixels[x_center+y_cir][y_center-x_cir]=clGreen;
    Canvas->Pixels[x_center-y_cir][y_center-x_cir]=clGreen;

     if (prop<0)
      prop= prop + (4*x_cir) + 6;
     else
     {
      prop= prop + (4*(x_cir-y_cir))+10;
      y_cir=y_cir-1;
     }
    x_cir=x_cir+1;
    }
  if (x_cir=y_cir)
```

```
  {
   Canvas->Pixels[x_center+x_cir][y_center+y_cir]=clGreen;
   Canvas->Pixels[x_center-x_cir][y_center+y_cir]=clGreen;
   Canvas->Pixels[x_center+x_cir][y_center-y_cir]=clGreen;
   Canvas->Pixels[x_center-x_cir][y_center-y_cir]=clGreen;
   Canvas->Pixels[x_center+y_cir][y_center+x_cir]=clGreen;
   Canvas->Pixels[x_center-y_cir][y_center+x_cir]=clGreen;
   Canvas->Pixels[x_center+y_cir][y_center-x_cir]=clGreen;
   Canvas->Pixels[x_center-y_cir][y_center-x_cir]=clGreen;
  }
 } }//CIRCUNFERENCIA 1
//************************************************************
{  // INICIO CIRCUNFERENCIA 2
      //CONTORNO
            //CIRCUNFERENCIA
  int cir, y_center,x_center;
  int prop, x_cir, y_cir, radius, k;
  x_center =200;   y_center=140;   cir=20;

  x_cir=0;
  y_cir=5+cir/2;
  prop=3-2*y_cir;
  while (x_cir<y_cir)
  {
   Canvas->Pixels[x_center+x_cir][y_center+y_cir]=clRed;
   Canvas->Pixels[x_center-x_cir][y_center+y_cir]=clRed;
   Canvas->Pixels[x_center+x_cir][y_center-y_cir]=clRed;
   Canvas->Pixels[x_center-x_cir][y_center-y_cir]=clRed;
   Canvas->Pixels[x_center+y_cir][y_center+x_cir]=clRed;
   Canvas->Pixels[x_center-y_cir][y_center+x_cir]=clRed;
   Canvas->Pixels[x_center+y_cir][y_center-x_cir]=clRed;
   Canvas->Pixels[x_center-y_cir][y_center-x_cir]=clRed;

   if (prop<0)
    prop= prop + 4*x_cir + 6;
   else
   { prop= prop + 4*(x_cir-y_cir)+10;
     y_cir=y_cir-1;}
     x_cir=x_cir+1;
   }
  if (x_cir=y_cir)
  {
   Canvas->Pixels[x_center+x_cir][y_center+y_cir]=clRed;
   Canvas->Pixels[x_center-x_cir][y_center+y_cir]=clRed;
   Canvas->Pixels[x_center+x_cir][y_center-y_cir]=clRed;
   Canvas->Pixels[x_center-x_cir][y_center-y_cir]=clRed;
   Canvas->Pixels[x_center+y_cir][y_center+x_cir]=clRed;
   Canvas->Pixels[x_center-y_cir][y_center+x_cir]=clRed;
   Canvas->Pixels[x_center+y_cir][y_center-x_cir]=clRed;
   Canvas->Pixels[x_center-y_cir][y_center-x_cir]=clRed;
  }
     // NUCLEO SOLIDO

 for ( k=1; k<(cir/2); k++)
 {
  x_cir=0;
  y_cir=k;
  prop=3-2*y_cir;
  while (x_cir<y_cir)
   {
    Canvas->Pixels[x_center+x_cir][y_center+y_cir]=clGreen;
    Canvas->Pixels[x_center-x_cir][y_center+y_cir]=clGreen;
    Canvas->Pixels[x_center+x_cir][y_center-y_cir]=clGreen;
    Canvas->Pixels[x_center-x_cir][y_center-y_cir]=clGreen;
    Canvas->Pixels[x_center+y_cir][y_center+x_cir]=clGreen;
    Canvas->Pixels[x_center-y_cir][y_center+x_cir]=clGreen;
    Canvas->Pixels[x_center+y_cir][y_center-x_cir]=clGreen;
    Canvas->Pixels[x_center-y_cir][y_center-x_cir]=clGreen;

     if (prop<0)
```

```
      prop= prop + (4*x_cir) + 6;
     else
     {
      prop= prop + (4*(x_cir-y_cir))+10;
      y_cir=y_cir-1;
     }
    x_cir=x_cir+1;
    }
  if (x_cir=y_cir)
  {
   Canvas->Pixels[x_center+x_cir][y_center+y_cir]=clGreen;
   Canvas->Pixels[x_center-x_cir][y_center+y_cir]=clGreen;
   Canvas->Pixels[x_center+x_cir][y_center-y_cir]=clGreen;
   Canvas->Pixels[x_center-x_cir][y_center-y_cir]=clGreen;
   Canvas->Pixels[x_center+y_cir][y_center+x_cir]=clGreen;
   Canvas->Pixels[x_center-y_cir][y_center+x_cir]=clGreen;
   Canvas->Pixels[x_center+y_cir][y_center-x_cir]=clGreen;
   Canvas->Pixels[x_center-y_cir][y_center-x_cir]=clGreen;
  }
 } }//CIRCUNFERENCIA 2
//************************************************************

{  // INICIO CIRCUNFERENCIA 3
       //CONTORNO
              //CIRCUNFERENCIA
  int cir, y_center,x_center;
  int prop, x_cir, y_cir, radius, k;
  x_center =200;   y_center=180;   cir=20;

  x_cir=0;
  y_cir=5+cir/2;
  prop=3-2*y_cir;
  while (x_cir<y_cir)
  {
   Canvas->Pixels[x_center+x_cir][y_center+y_cir]=clRed;
   Canvas->Pixels[x_center-x_cir][y_center+y_cir]=clRed;
   Canvas->Pixels[x_center+x_cir][y_center-y_cir]=clRed;
   Canvas->Pixels[x_center-x_cir][y_center-y_cir]=clRed;
   Canvas->Pixels[x_center+y_cir][y_center+x_cir]=clRed;
   Canvas->Pixels[x_center-y_cir][y_center+x_cir]=clRed;
   Canvas->Pixels[x_center+y_cir][y_center-x_cir]=clRed;
   Canvas->Pixels[x_center-y_cir][y_center-x_cir]=clRed;

   if (prop<0)
    prop= prop + 4*x_cir + 6;
  else
   { prop= prop + 4*(x_cir-y_cir)+10;
     y_cir=y_cir-1;}
     x_cir=x_cir+1;
   }
  if (x_cir=y_cir)
  {
   Canvas->Pixels[x_center+x_cir][y_center+y_cir]=clRed;
   Canvas->Pixels[x_center-x_cir][y_center+y_cir]=clRed;
   Canvas->Pixels[x_center+x_cir][y_center-y_cir]=clRed;
   Canvas->Pixels[x_center-x_cir][y_center-y_cir]=clRed;
   Canvas->Pixels[x_center+y_cir][y_center+x_cir]=clRed;
   Canvas->Pixels[x_center-y_cir][y_center+x_cir]=clRed;
   Canvas->Pixels[x_center+y_cir][y_center-x_cir]=clRed;
   Canvas->Pixels[x_center-y_cir][y_center-x_cir]=clRed;
  }
    // NUCLEO SOLIDO

 for ( k=1; k<(cir/2); k++)
 {
  x_cir=0;
  y_cir=k;
  prop=3-2*y_cir;
  while (x_cir<y_cir)
   {
```

```
   Canvas->Pixels[x_center+x_cir][y_center+y_cir]=clGreen;
   Canvas->Pixels[x_center-x_cir][y_center+y_cir]=clGreen;
   Canvas->Pixels[x_center+x_cir][y_center-y_cir]=clGreen;
   Canvas->Pixels[x_center-x_cir][y_center-y_cir]=clGreen;
   Canvas->Pixels[x_center+y_cir][y_center+x_cir]=clGreen;
   Canvas->Pixels[x_center-y_cir][y_center+x_cir]=clGreen;
   Canvas->Pixels[x_center+y_cir][y_center-x_cir]=clGreen;
   Canvas->Pixels[x_center-y_cir][y_center-x_cir]=clGreen;

    if (prop<0)
     prop= prop + (4*x_cir) + 6;
    else
    {
     prop= prop + (4*(x_cir-y_cir))+10;
     y_cir=y_cir-1;
    }
   x_cir=x_cir+1;
   }
  if (x_cir=y_cir)
  {
   Canvas->Pixels[x_center+x_cir][y_center+y_cir]=clGreen;
   Canvas->Pixels[x_center-x_cir][y_center+y_cir]=clGreen;
   Canvas->Pixels[x_center+x_cir][y_center-y_cir]=clGreen;
   Canvas->Pixels[x_center-x_cir][y_center-y_cir]=clGreen;
   Canvas->Pixels[x_center+y_cir][y_center+x_cir]=clGreen;
   Canvas->Pixels[x_center-y_cir][y_center+x_cir]=clGreen;
   Canvas->Pixels[x_center+y_cir][y_center-x_cir]=clGreen;
   Canvas->Pixels[x_center-y_cir][y_center-x_cir]=clGreen;
  }
 } }//CIRCUNFERENCIA 3
//*************************************************************
{  // INICIO CIRCUNFERENCIA 4
      //CONTORNO
            //CIRCUNFERENCIA
  int cir, y_center,x_center;
  int prop, x_cir, y_cir, radius, k;
  x_center =200;   y_center=220;   cir=20;

  x_cir=0;
  y_cir=5+cir/2;
  prop=3-2*y_cir;
  while (x_cir<y_cir)
  {
   Canvas->Pixels[x_center+x_cir][y_center+y_cir]=clRed;
   Canvas->Pixels[x_center-x_cir][y_center+y_cir]=clRed;
   Canvas->Pixels[x_center+x_cir][y_center-y_cir]=clRed;
   Canvas->Pixels[x_center-x_cir][y_center-y_cir]=clRed;
   Canvas->Pixels[x_center+y_cir][y_center+x_cir]=clRed;
   Canvas->Pixels[x_center-y_cir][y_center+x_cir]=clRed;
   Canvas->Pixels[x_center+y_cir][y_center-x_cir]=clRed;
   Canvas->Pixels[x_center-y_cir][y_center-x_cir]=clRed;

   if (prop<0)
    prop= prop + 4*x_cir + 6;
  else
   { prop= prop + 4*(x_cir-y_cir)+10;
     y_cir=y_cir-1;}
     x_cir=x_cir+1;
   }
  if (x_cir=y_cir)
  {
   Canvas->Pixels[x_center+x_cir][y_center+y_cir]=clRed;
   Canvas->Pixels[x_center-x_cir][y_center+y_cir]=clRed;
   Canvas->Pixels[x_center+x_cir][y_center-y_cir]=clRed;
   Canvas->Pixels[x_center-x_cir][y_center-y_cir]=clRed;
   Canvas->Pixels[x_center+y_cir][y_center+x_cir]=clRed;
   Canvas->Pixels[x_center-y_cir][y_center+x_cir]=clRed;
   Canvas->Pixels[x_center+y_cir][y_center-x_cir]=clRed;
   Canvas->Pixels[x_center-y_cir][y_center-x_cir]=clRed;
  }
```

```
    // NUCLEO SOLIDO

  for ( k=1; k<(cir/2); k++)
  {
   x_cir=0;
   y_cir=k;
   prop=3-2*y_cir;
   while (x_cir<y_cir)
    {
     Canvas->Pixels[x_center+x_cir][y_center+y_cir]=clGreen;
     Canvas->Pixels[x_center-x_cir][y_center+y_cir]=clGreen;
     Canvas->Pixels[x_center+x_cir][y_center-y_cir]=clGreen;
     Canvas->Pixels[x_center-x_cir][y_center-y_cir]=clGreen;
     Canvas->Pixels[x_center+y_cir][y_center+x_cir]=clGreen;
     Canvas->Pixels[x_center-y_cir][y_center+x_cir]=clGreen;
     Canvas->Pixels[x_center+y_cir][y_center-x_cir]=clGreen;
     Canvas->Pixels[x_center-y_cir][y_center-x_cir]=clGreen;

      if (prop<0)
       prop= prop + (4*x_cir) + 6;
      else
      {
       prop= prop + (4*(x_cir-y_cir))+10;
       y_cir=y_cir-1;
      }
     x_cir=x_cir+1;
     }
   if (x_cir=y_cir)
   {
    Canvas->Pixels[x_center+x_cir][y_center+y_cir]=clGreen;
    Canvas->Pixels[x_center-x_cir][y_center+y_cir]=clGreen;
    Canvas->Pixels[x_center+x_cir][y_center-y_cir]=clGreen;
    Canvas->Pixels[x_center-x_cir][y_center-y_cir]=clGreen;
    Canvas->Pixels[x_center+y_cir][y_center+x_cir]=clGreen;
    Canvas->Pixels[x_center-y_cir][y_center+x_cir]=clGreen;
    Canvas->Pixels[x_center+y_cir][y_center-x_cir]=clGreen;
    Canvas->Pixels[x_center-y_cir][y_center-x_cir]=clGreen;
   }
  } }//CIRCUNFERENCIA 4
//*****************************************************************
{  // INICIO CIRCUNFERENCIA 5
       //CONTORNO
              //CIRCUNFERENCIA
   int cir, y_center,x_center;
   int prop, x_cir, y_cir, radius, k;
   x_center =200;   y_center=260;   cir=20;

   x_cir=0;
   y_cir=5+cir/2;
   prop=3-2*y_cir;
   while (x_cir<y_cir)
   {
    Canvas->Pixels[x_center+x_cir][y_center+y_cir]=clRed;
    Canvas->Pixels[x_center-x_cir][y_center+y_cir]=clRed;
    Canvas->Pixels[x_center+x_cir][y_center-y_cir]=clRed;
    Canvas->Pixels[x_center-x_cir][y_center-y_cir]=clRed;
    Canvas->Pixels[x_center+y_cir][y_center+x_cir]=clRed;
    Canvas->Pixels[x_center-y_cir][y_center+x_cir]=clRed;
    Canvas->Pixels[x_center+y_cir][y_center-x_cir]=clRed;
    Canvas->Pixels[x_center-y_cir][y_center-x_cir]=clRed;

    if (prop<0)
     prop= prop + 4*x_cir + 6;
    else
     { prop= prop + 4*(x_cir-y_cir)+10;
       y_cir=y_cir-1;}
       x_cir=x_cir+1;
     }
    if (x_cir=y_cir)
    {
```

```
  Canvas->Pixels[x_center+x_cir][y_center+y_cir]=clRed;
  Canvas->Pixels[x_center-x_cir][y_center+y_cir]=clRed;
  Canvas->Pixels[x_center+x_cir][y_center-y_cir]=clRed;
  Canvas->Pixels[x_center-x_cir][y_center-y_cir]=clRed;
  Canvas->Pixels[x_center+y_cir][y_center+x_cir]=clRed;
  Canvas->Pixels[x_center-y_cir][y_center+x_cir]=clRed;
  Canvas->Pixels[x_center+y_cir][y_center-x_cir]=clRed;
  Canvas->Pixels[x_center-y_cir][y_center-x_cir]=clRed;
 }
    // NUCLEO SOLIDO

 for ( k=1; k<(cir/2); k++)
 {
  x_cir=0;
  y_cir=k;
  prop=3-2*y_cir;
 while (x_cir<y_cir)
  {
   Canvas->Pixels[x_center+x_cir][y_center+y_cir]=clGreen;
   Canvas->Pixels[x_center-x_cir][y_center+y_cir]=clGreen;
   Canvas->Pixels[x_center+x_cir][y_center-y_cir]=clGreen;
   Canvas->Pixels[x_center-x_cir][y_center-y_cir]=clGreen;
   Canvas->Pixels[x_center+y_cir][y_center+x_cir]=clGreen;
   Canvas->Pixels[x_center-y_cir][y_center+x_cir]=clGreen;
   Canvas->Pixels[x_center+y_cir][y_center-x_cir]=clGreen;
   Canvas->Pixels[x_center-y_cir][y_center-x_cir]=clGreen;

    if (prop<0)
     prop= prop + (4*x_cir) + 6;
    else
    {
     prop= prop + (4*(x_cir-y_cir))+10;
     y_cir=y_cir-1;
    }
   x_cir=x_cir+1;
   }
 if (x_cir=y_cir)
 {
  Canvas->Pixels[x_center+x_cir][y_center+y_cir]=clGreen;
  Canvas->Pixels[x_center-x_cir][y_center+y_cir]=clGreen;
  Canvas->Pixels[x_center+x_cir][y_center-y_cir]=clGreen;
  Canvas->Pixels[x_center-x_cir][y_center-y_cir]=clGreen;
  Canvas->Pixels[x_center+y_cir][y_center+x_cir]=clGreen;
  Canvas->Pixels[x_center-y_cir][y_center+x_cir]=clGreen;
  Canvas->Pixels[x_center+y_cir][y_center-x_cir]=clGreen;
  Canvas->Pixels[x_center-y_cir][y_center-x_cir]=clGreen;
 }
 } }//CIRCUNFERENCIA 5
//*************************************************************
{  // INICIO CIRCUNFERENCIA 6
      //CONTORNO
            //CIRCUNFERENCIA
  int cir, y_center,x_center;
  int prop, x_cir, y_cir, radius, k;
  x_center =200;   y_center=300;   cir=20;

  x_cir=0;
  y_cir=5+cir/2;
  prop=3-2*y_cir;
  while (x_cir<y_cir)
  {
   Canvas->Pixels[x_center+x_cir][y_center+y_cir]=clRed;
   Canvas->Pixels[x_center-x_cir][y_center+y_cir]=clRed;
   Canvas->Pixels[x_center+x_cir][y_center-y_cir]=clRed;
   Canvas->Pixels[x_center-x_cir][y_center-y_cir]=clRed;
   Canvas->Pixels[x_center+y_cir][y_center+x_cir]=clRed;
   Canvas->Pixels[x_center-y_cir][y_center+x_cir]=clRed;
   Canvas->Pixels[x_center+y_cir][y_center-x_cir]=clRed;
   Canvas->Pixels[x_center-y_cir][y_center-x_cir]=clRed;
```

```
   if (prop<0)
    prop= prop + 4*x_cir + 6;
   else
    { prop= prop + 4*(x_cir-y_cir)+10;
      y_cir=y_cir-1;}
      x_cir=x_cir+1;
    }
   if (x_cir=y_cir)
   {
    Canvas->Pixels[x_center+x_cir][y_center+y_cir]=clRed;
    Canvas->Pixels[x_center-x_cir][y_center+y_cir]=clRed;
    Canvas->Pixels[x_center+x_cir][y_center-y_cir]=clRed;
    Canvas->Pixels[x_center-x_cir][y_center-y_cir]=clRed;
    Canvas->Pixels[x_center+y_cir][y_center+x_cir]=clRed;
    Canvas->Pixels[x_center-y_cir][y_center+x_cir]=clRed;
    Canvas->Pixels[x_center+y_cir][y_center-x_cir]=clRed;
    Canvas->Pixels[x_center-y_cir][y_center-x_cir]=clRed;
   }
      // NUCLEO SOLIDO

  for ( k=1; k<(cir/2); k++)
  {
    x_cir=0;
    y_cir=k;
    prop=3-2*y_cir;
   while (x_cir<y_cir)
    {
     Canvas->Pixels[x_center+x_cir][y_center+y_cir]=clGreen;
     Canvas->Pixels[x_center-x_cir][y_center+y_cir]=clGreen;
     Canvas->Pixels[x_center+x_cir][y_center-y_cir]=clGreen;
     Canvas->Pixels[x_center-x_cir][y_center-y_cir]=clGreen;
     Canvas->Pixels[x_center+y_cir][y_center+x_cir]=clGreen;
     Canvas->Pixels[x_center-y_cir][y_center+x_cir]=clGreen;
     Canvas->Pixels[x_center+y_cir][y_center-x_cir]=clGreen;
     Canvas->Pixels[x_center-y_cir][y_center-x_cir]=clGreen;

      if (prop<0)
       prop= prop + (4*x_cir) + 6;
      else
      {
       prop= prop + (4*(x_cir-y_cir))+10;
       y_cir=y_cir-1;
      }
     x_cir=x_cir+1;
     }
   if (x_cir=y_cir)
   {
    Canvas->Pixels[x_center+x_cir][y_center+y_cir]=clGreen;
    Canvas->Pixels[x_center-x_cir][y_center+y_cir]=clGreen;
    Canvas->Pixels[x_center+x_cir][y_center-y_cir]=clGreen;
    Canvas->Pixels[x_center-x_cir][y_center-y_cir]=clGreen;
    Canvas->Pixels[x_center+y_cir][y_center+x_cir]=clGreen;
    Canvas->Pixels[x_center-y_cir][y_center+x_cir]=clGreen;
    Canvas->Pixels[x_center+y_cir][y_center-x_cir]=clGreen;
    Canvas->Pixels[x_center-y_cir][y_center-x_cir]=clGreen;
   }
  } }//CIRCUNFERENCIA 6
//************************************************************

//************************************************************
//++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 // INICIA DETECCION DE CAMPO
 //---DATOS DETECCION DE CAMPO---------------------------------

   int i,j, iter;
   int col1, col2, tono;
   int const w=500,h=500;
   int campo[w][h];
   tono=clGreen;
   int x_obj[1000], y_obj[1000] ;
```

```
   iter=0;
//----------CALCULO PARA DETECCION DE CAMPO----------------------

  for( i=0; i<w; i++)
  for(j=0; j<h; j++)
  {   col1=Canvas->Pixels[i][j];
      col2=((col1&0xFFFFF)+((col1>>8)&0xFFFFF)+((col1>>16)&0xFFFFF));
        if (col2>tono)
     {  campo[i][j]=0;}
        else
      {
        campo[i][j]=1;
        iter++;
        x_obj[iter]=i;
        y_obj[iter]=j;
      }
  }
// -------TERMINA DETECCION DE CAMPO**********************

//---DATOS IINICIO Y META DESEADA PARA MOVIMIENTO DE ROBOT
 int  xp_in, yp_in, xp_end, yp_end, ciclo=0;
 int  x_in, y_in;

 float  R_in_end;
 float  Fi_in_end;
 float  dxp, dyp;
 float const PI=3.141592653589;
  x_in  =0;//Edit1->Text.ToInt();
  y_in  =150;//Edit2->Text.ToInt();
  xp_end=350;//Edit3->Text.ToInt();
  yp_end=220;//Edit4->Text.ToInt();

//---DATOS PARA DETECCIÒN RADIAL DEL OBJETO

  int x1,y1,x2,y2;
  int dx,dy,x,y,x_end,p,cons1,cons2,ahora;
  int evento,y_end;
  int const diam=10;
  int const R_dis=500;
  float Fi;
  float const dFi=3*PI/180;  // TRES GRADOS DE INCREMENTO

//----DATOS DE DISTANCIA E INCLINACION INICIO PUNTO DETECTADO

 int pass;
 int x_det1, y_det1;
 float  angle, distance, angle_SI;
 int const Rmin=1;
 int const C1= 20;
 int const C2= 10;


 //-----DATOS PARA EL PUNTO DE FUGA

 float Q;
 int   x_mid, y_mid;
 int   x_mid_rec[1000], y_mid_rec[1000];
 int const dp2=30, dp1=20;

 float Qmax=0;

 int  const espera=15;

   xp_in=x_in;
   yp_in=y_in;
//--- CALCULO DE LA DISTANCIA E INCLINACION ENTRE PUNTO INICIAL Y FINAL
 int const prox=30;
 int paso;
 paso=470;//Edit5->Text.ToInt();
 for (int n=0; n<=paso; n++)
```

```
 {
  ciclo++;
  dxp=labs(xp_in-xp_end);
  dyp=labs(yp_in-yp_end);
  R_in_end= sqrt(pow(xp_end-xp_in,2)+ pow(yp_end-yp_in,2));
//-------------------------------------------------------------------------
// SE EVALUA PENDIENTE POSITIVA Y NEGATIVA
if (((xp_in<xp_end)&&(yp_in<=yp_end))||((xp_end<xp_in)&&(yp_end<=yp_in)))
 {  //PENDIENTE POSITIVA
 //PENDIENTE POSITIVA ENTRE  0 y 1
  if(dyp<=dxp)
  {
 //*********************************************************
 //UNO---CASO 1 (MOVIMIENTO DE IZQUIERDA A DERECHA)
     if (xp_in<xp_end)
      Fi_in_end= asin((yp_end-yp_in)/R_in_end);
//DOS---CASO DOS (MOVIMIENTO DE DERECHA A IZQUIERDA)
 else
   Fi_in_end= asin((yp_in-yp_end)/R_in_end) + PI;
 }//*********************************************************
//+++++++++++++++++++++++++++++++++++++++++++++++++
//INICIA PENDIENTE POSITIVA MAYOR QUE 1
  else
   {//TRES---CASO 1 (MOVIMIENTO DE IZQUIERDA A DERECHA)
       if (yp_in<yp_end)
         Fi_in_end= (PI/2)-asin((xp_end-xp_in)/R_in_end);
    else
  //CUATRO------CASO DOS (MOVIMIENTO DE DERECHA A IZQUIERDA)
       Fi_in_end= (3*PI/2.0)-asin((xp_in-xp_end)/R_in_end);
    }
}
//+++++++++++++++++++++++++++++++++++++++++++++++++
//-------------------------------------------------------------------------
// SE EVALUA PENDIENTE NEGATIVA
 else
  {
  if (dyp<dxp)
//PENDIENTE NEGATIVA ENTRE 0 y 1
   {
 // CINCO-----CASO UNO (MOVIMIENTO DE IZQUIERDA A DERECHA)
    if (xp_in<xp_end)
     Fi_in_end= (2*PI)-asin((yp_in-yp_end)/R_in_end);
// SEIS-----CASO DOS  (MOVIMIENTO DE  DERECHA A IZQUIERDA)
else
     Fi_in_end= PI-asin((yp_end-yp_in)/R_in_end);
  }
//*****************************************************
//INICIA PENDIENTE NEGATIVA MAYOR QUE 1
  else
   {//SIETE---CASO 1 (MOVIMIENTO DE IZQUIERDA A DERECHA)
       if (yp_in<yp_end)
     Fi_in_end= (PI/2)+asin((xp_in-xp_end)/R_in_end);
     else
    //OCHO------CASO DOS (MOVIMIENTO DE DERECHA A IZQUIERDA)
     Fi_in_end= (3*PI/2.0)+asin((xp_end-xp_in)/R_in_end);
   }
}// ---TERMINA CALCULO DE DISTANCIA E INCLINACIÓN PUNTO INICAL Y META

//******************INICIA DETECCION RADIAL DE OBJETO************
//                                                                ***
//                                                                ***
//*****************************************************************


for (Fi=Fi_in_end; Fi<=2*PI+Fi_in_end; Fi=Fi+dFi)
{      //INICIA FOR DE DETECCION RADIAL
 x1=xp_in;
 y1=yp_in;
 x2=x1+R_dis*cos(Fi);
 y2=y1+R_dis*sin(Fi);
```

```
 // SE EVALUA LA PENDIENTE
  dx=labs(x1-x2);
  dy=labs(y1-y2);
  p=2*dy-dx;
  cons1=2*dy;
  cons2=2*(dy-dx);
// SE EVALUA PENDIENTE POSITIVA Y NEGATIVA
 if (((x1<x2)&&(y1<=y2))||((x2<x1)&&(y2<=y1)))
  {  // INICIA PENDIENTE POSITIVA
  if(dy<=dx)
    { // INICIA PENDIENTE POSITIVA ENTRE  0 y 1
    if (x1<x2)
      {     //CASO 1 (MOVIMIENTO DE IZQUIERDA A DERECHA)
       x=x1;        y=y1;        x_end=x2;
       while (x<x_end)
       {
         x=x+1;
         if (p<0)
          p=p+cons1;
         else
         {
         y=y+1;         p=p+cons2;
         }
//       Canvas->Pixels[x][y]=clYellow;
   if (x<0) break;
   if (y<0) break;
   if (x>w-1) break;
   if (y>h-1) break;
//**********INICIA IF DE VERIFICACION
        if ( campo [x][y] == 1 )
        {
          x_det1=x;
          y_det1=y;
          distance= sqrt(pow(x-x1,2)+ pow(y-y1,2));
          angle = Fi;
          angle_SI =Fi_in_end -Fi;
          Q = C1*cos(angle_SI)+ C2*( 1/(distance+Rmin));
         if (Q>Qmax)
          { Qmax=Q;
           if (prox<=distance)
            { x_mid=xp_in+1;
              y_mid=yp_in;
             if (p<0)
             p=p+cons1;
             else
             {
              y_mid=yp_in+1;
              p=p+cons2;
             }
            }//  TERMINA IF DISTANCE
            else
            { x_mid=xp_in-1;
              y_mid=yp_in+1;
            }
            x_mid_rec[ciclo]=x_mid;
            y_mid_rec[ciclo]=y_mid;
          }//TERMINA IF Q
        }//TERMINA IF CAMPO
       }//TERMINA WHILE
      }//TERMINA CASO 1
     else
      { //INICIA  CASO 2 (MOVIMIENTO DE DERECHA A IZQUIERDA)
      x=x1;         y=y1;         x_end=x2;
      while (x_end<x)
      {
        x=x-1;
        if (p<0)
         p=p+cons1;
        else
```

```
          {
          y=y-1;          p=p+cons2;
          }
//     Canvas->Pixels[x][y]=clYellow;
   if (x<0)   break;
   if (y<0)   break;
   if (x>w-1) break;
   if (y>h-1) break;
//**********INICIA IF DE VERIFICACION
        if ( campo [x][y] == 1 )
        {
          x_det1=x;
          y_det1=y;
          distance= sqrt(pow(x-x1,2)+ pow(y-y1,2));
          angle= Fi;
          angle_SI =Fi-Fi_in_end;
          Q = C1*cos(angle_SI)+ C2*( 1/(distance+Rmin));

         if (Q>Qmax)
          { Qmax=Q;
            if (prox<=distance)
            { x_mid=xp_in+1;
              y_mid=yp_in;
             if (p<0)
             p=p+cons1;
             else
             {
              y_mid=yp_in-1;
              p=p+cons2;
             }
            }//  TERMINA IF DISTANCE
            else
            { x_mid=xp_in-1;
              y_mid=yp_in+1;
            }
            x_mid_rec[ciclo]=x_mid;
            y_mid_rec[ciclo]=y_mid;
           }//TERMINA IF Q
        }//TERMINA IF DE CAMPO
      } //TERMINA WHILE
     }//TERMINA CASO DOS
    }//TERMINA PENDIENTE POSITIVA ENTRE CERO Y UNO
    else
   { // INICIA PENDIENTE POSITIVA MAYOR A 1
     p=2*dx-dy;
     cons1=2*dx;
     cons2=2*(dx-dy);
   if (x1<x2)
    { //CASO TRES MOVIMIENTO DE IZQUIERDA A DERECHA
      x=x1;       y=y1;       y_end=y2;
      while (y<y_end)
      {
        y=y+1;
      if (p<0)
        p=p+cons1;
      else
        {
       x=x+1;       p=p+cons2;
        }
//       Canvas->Pixels[x][y]=clYellow;
   if (x<0) break;
   if (y<0) break;
   if (x>w-1) break;
   if (y>h-1) break;
//**********INICIA IF DE VERIFICACION
        if ( campo [x][y] == 1 )
        {
          x_det1=x;
          y_det1=y;
          distance = sqrt(pow(x-x1,2)+ pow(y-y1,2));
```

```
          angle = Fi;
          angle_SI =Fi-Fi_in_end;
          Q = C1*cos(angle_SI)+ C2*( 1/(distance+Rmin));
         if (Q>Qmax)
          { Qmax=Q;
           if (prox<=distance)
            { x_mid=xp_in;
              y_mid=yp_in+1;
             if (p<0)
             p=p+cons1;
             else
             {
              x_mid=xp_in+1;
              p=p+cons2;
             }
            }//  TERMINA IF DISTANCE
            else
            { x_mid=xp_in-1;
              y_mid=yp_in;
            }
            x_mid_rec[ciclo]=x_mid;
            y_mid_rec[ciclo]=y_mid;
           }//TERMINA IF Q
        }//TERMINA IF DE CAMPO
       } //TERMINA WHILE
      } // TERMINA CASO TRES MOVIMIENTO DE IZQUIERDA A DERECHA
     else
     { // CASO CUATRO MOVIMIENTO DE DERECHA A IZQUIERDA
       x=x1;       y=y1;       y_end=y2;
       while (y_end<y)
       {
        y=y-1;
        if (p<0)
          p=p+cons1;
        else
         {
         x=x-1;         p=p+cons2;
         }
//         Canvas->Pixels[x][y]=clYellow;
    if (x<0) break;
    if (y<0) break;
    if (x>w-1) break;
    if (y>h-1) break;
//**********INICIA IF DE VERIFICACION
        if ( campo [x][y] == 1 )
        {
          x_det1=x;
          y_det1=y;
          distance = sqrt(pow(x-x1,2)+ pow(y-y1,2));
          angle = Fi;
          angle_SI =Fi_in_end -Fi;
          Q = C1*cos(angle_SI)+ C2*( 1/(distance+Rmin));
         if (Q>Qmax)
          {Qmax=Q;
           if (prox<=distance)
            { x_mid=xp_in;
              y_mid=yp_in-1;
              p=2*dx-dy;
              cons1=2*dx;
              cons2=2*(dx-dy);
             if (p<0)
             p=p+cons1;
             else
             {
              x_mid=xp_in-1;
              p=p+cons2;
             }
            }//  TERMINA IF DISTANCE
            else
            { x_mid=xp_in+1;
```

```
            y_mid=yp_in+1;
           }
           x_mid_rec[ciclo]=x_mid;
           y_mid_rec[ciclo]=y_mid;
          }//TERMINA IF DE Q
         }//TERMINA IF DE CAMPO
        }//TERMINA WHILE
      } // TERMINA CASO CUATRO MOVIMIENTO DE DERECHA A IZQUIERDA
    } // TERMINA PENDIENTE POSITIVA MAYOR A 1
  }  // TERMINA PENDIENTE POSITIVA
  //+++++++++++++++++++++++++++++++++++++++++++
  //+++++++++++++++++++++++++++++++++++++++++++
  //+++++++++++++++++++++++++++++++++++++++++++
 else
  {
    // INICIA PENDIENTE NEGATIVA
     if (dy<=dx)
      {// INICIA PENDIENTE NEGATIVA ENTRE 0 y 1
      if (x1<x2)
       { // CASO UNO MOVIMIENTO DE IZQUIERDA A DERECHA)
        x=x1;      y=y1;      x_end=x2;
        while (x<x_end)
        {
         x=x+1;
         if (p<0)
          p=p+cons1;
         else
          {
           y=y-1;         p=p+cons2;
          }
//        Canvas->Pixels[x][y]=clYellow;
    if (x<0) break;
    if (y<0) break;
    if (x>w-1) break;
    if (y>h-1) break;
//**********INICIA IF DE VERIFICACION
        if ( campo [x][y] == 1 )
        {
          x_det1=x;
          y_det1=y;
          distance = sqrt(pow(x-x1,2)+ pow(y-y1,2));
          angle = Fi;
          angle_SI =Fi_in_end-Fi;
          Q = C1*cos(angle_SI)+ C2*( 1/(distance+Rmin));
         if (Q>Qmax)
          {Qmax=Q;
           if (prox<=distance)
            { x_mid=xp_in+1;
              y_mid=yp_in;
             if (p<0)
             p=p+cons1;
             else
             {
              y_mid=yp_in+1;
              p=p+cons2;
             }
            }//  TERMINA IF DISTANCE
            else
            { x_mid=xp_in-1;
              y_mid=yp_in+1;
            }
            x_mid_rec[ciclo]=x_mid;
            y_mid_rec[ciclo]=y_mid;
           }//TERMINA IF Q
         } //TERMINA IF CAMPO
        }//TERMINA WHILE
       }// TERMINA CASO UNO MOVIMIENTO DE IZQUIERDA A DERECHA
       else
       {// CASO DOS MOVIMIENTO DE IZQUIERDA A DERECHA)
       x=x1;       y=y1;       x_end=x2;
```

```
        while (x_end<x)
         {
          x=x-1;
          if (p<0)
           p=p+cons1;
          else
           {
            y=y+1;          p=p+cons2;
           }
//        Canvas->Pixels[x][y]=clYellow;
    if (x<0) break;
    if (y<0) break;
    if (x>w-1) break;
    if (y>h-1) break;
//**********INICIA IF DE VERIFICACION
         if ( campo [x][y] == 1 )
         {
           x_det1=x;
           y_det1=y;
           distance = sqrt(pow(x-x1,2)+ pow(y-y1,2));
           angle = Fi;
           angle_SI =Fi-Fi_in_end;
           Q = C1*cos(angle_SI)+ C2*(1/(distance+Rmin));
          if (Q>Qmax)
           {Qmax=Q;
            if (prox<=distance)
             { x_mid=xp_in+1;
               y_mid=yp_in;
              if (p<0)
              p=p+cons1;
              else
              {
              y_mid=yp_in+1;
              p=p+cons2;
              }
             }//  TERMINA IF DISTANCE
             else
             { x_mid=xp_in-1;
               y_mid=yp_in+1;
             }
             x_mid_rec[ciclo]=x_mid;
             y_mid_rec[ciclo]=y_mid;
            }//TERMINA IF DE Q
          }//TERMINA IF DE CAMPO
        }//TERMINA WHIEL
       }//TERMINA CASO DOS
      }//TERMINA PENDIENTE NEGATIVA ENTRE 0 Y 1
      else
      {// INICIA PENDIENTE NEGATIVA MAYOR A 1
        p=2*dx-dy;
        cons1=2*dx;
        cons2=2*(dx-dy);
       if (x2<x1)
        {// CASO TRES (MOVIMIENTO DE DERECHA A IZQUIERDA)
        x=x1;       y=y1;       y_end=y2;
        while (y<y_end)
        {
          y=y+1;
          if (p<0)
           p=p+cons1;
          else
           {
            x=x-1;          p=p+cons2;
           }
//        Canvas->Pixels[x][y]=clYellow;
    if (x<0) break;
    if (y<0) break;
    if (x>w-1) break;
    if (y>h-1) break;
//**********INICIA IF DE VERIFICACION
```

```
        if ( campo [x][y] == 1 )
        {
          x_det1=x;
          y_det1=y;
          distance = sqrt(pow(x-x1,2)+ pow(y-y1,2));
          angle = Fi;
          angle_SI =Fi_in_end-Fi;
          Q = C1*cos(angle_SI)+ C2*( 1/(distance+Rmin));
         if (Q>Qmax)
          {Qmax=Q;
           if (prox<=distance)
            { x_mid=xp_in;
              y_mid=yp_in+1;
             if (p<0)
             p=p+cons1;
             else
             {
              x_mid=xp_in-1;
              p=p+cons2;
             }
            }//  TERMINA IF DISTANCE
            else
            { x_mid=xp_in-1;
              y_mid=yp_in-1;
            }
            x_mid_rec[ciclo]=x_mid;
            y_mid_rec[ciclo]=y_mid;
           }//TERMINA IF Q
         }//TERMINA IF DE CAMPO
        }//TERMINA WHILE
       }// TERMINA CASO TRES (MOVIMIENTO DE DERECHA A IZQUIERDA)
       else
        {// CASO CUATRO (MOVIMIENTO DE IZQUIERDA A DERECHA)
        x=x1;       y=y1;       y_end=y2;
        while (y_end<y)
         {
          y=y-1;
          if (p<0)
           p=p+cons1;
          else
           {
           x=x+1;         p=p+cons2;
           }
//       Canvas->Pixels[x][y]=clYellow;
      if (x<0) break;
      if (y<0) break;
      if (x>w-1) break;
      if (y>h-1) break;
//**********INICIA IF DE VERIFICACION
        if ( campo [x][y] == 1 )
        {
          x_det1=x;
          y_det1=y;
          distance = sqrt(pow(x-x1,2)+ pow(y-y1,2));
          angle = Fi;
          angle_SI =Fi_in_end -Fi;
          Q = C1*cos(angle_SI)+ C2*( 1/(distance+Rmin));

         if (Q>Qmax)
          { Qmax=Q;
           if (prox<=distance)
            { x_mid=xp_in;
              y_mid=yp_in-1;
              dx=labs(x_mid-xp_end);
              dy=labs(y_mid-yp_end);
              p=2*dx-dy;
              cons1=2*dx;
              cons2=2*(dx-dy);
             if (p<0)
             p=p+cons1;
```

```
            else
            {
             x_mid=xp_in+1;
             p=p+cons2;
            }
           }//  TERMINA IF DISTANCE
           else
           { x_mid=xp_in+1;
             y_mid=yp_in+1;
           }
           x_mid_rec[ciclo]=x_mid;
           y_mid_rec[ciclo]=y_mid;
          }//TERMINA IF DE Q
        }//TERMINA IF DE CAMPO
       }//TERMINA WHILE
      }// TERMINA CASO CUATRO (MOVIMIENTO DE IZQUIERDA A DERECHA)
    }// TERMINA PENDIENTE NEGATIVA MAYOR A 1
   }// TERMINA PENDIENTE NEGATIVA
   //-------------------------------------------
   //-------------------------------------------
   //-------------------------------------------
  }//TERMINA FOR DE DETECCION RADIAL
  xp_in=x_mid;
  yp_in=y_mid;
  Qmax=0;
}//TERMINA FOR DE CICLO
   Canvas->MoveTo(x_in,y_in);
   Canvas->LineTo(xp_end,yp_end);
 //************************INICIO DE PREOGRAMA****************
//******         MOVIMIENTO DEL ROBOT                 ****

//************************************************************
   x1=x_in;
   y1=y_in;

  for (int k=1;k<=paso; k++)
{
   x2=x_mid_rec[k];
   y2=y_mid_rec[k];

// SE EVALUA PENDIENTE
  dx=labs(x1-x2);
  dy=labs(y1-y2);
  p=2*dy-dx;
  cons1=2*dy;
  cons2=2*(dy-dx);

// SE EVALUA PENDIENTE POSITIVA Y NEGATIVA
 if (((x1<=x2)&&(y1<y2))||((x2<x1)&&(y2<y1)))
  {  //PENDIENTE POSITIVA

//PENDIENTE POSITIVA ENTRE  0 y 1
   if(dy<=dx)
    {
//CASO 1 (MOVIMIENTO DE IZQUIERDA A DERECHA)
     if (x1<x2)
      {
       x=x1;       y=y1;       x_end=x2;
       while (x<x_end)
       {
         x=x+1;
         if (p<0)
          p=p+cons1;
         else
         {
         y=y+1;        p=p+cons2;
         }
         Canvas->Brush->Color=clRed;
         Canvas->RoundRect(x,y,x+diam,y+diam,x+diam,y+diam);
         ahora=GetTickCount();
```

```
        do; while(GetTickCount()-ahora<espera);
       }
      }
// TERMINA PRIMER CUADRANTE

//CASO DOS (MOVIMIENTO DE DERECHA A IZQUIERDA)
     else
      { x=x1;        y=y1;        x_end=x2;
       while (x_end<x)
       {
         x=x-1;
         if (p<0)
          p=p+cons1;
         else
           {
          y=y-1;         p=p+cons2;
           }
         Canvas->Brush->Color=clGreen;
         Canvas->RoundRect(x,y,x+diam,y+diam,x+diam,y+diam);
         ahora=GetTickCount();
         do; while(GetTickCount()-ahora<espera);
       }
      }
// TERMINA CUARTO CUADRANTE
    }
// TERMINA PENDIENTE POSITIVA ENTRE 0 Y 1

//INICIA PENDIENTE POSITIVA MAYOR QUE 1
   else
    {
      p=2*dx-dy;
      cons1=2*dx;
      cons2=2*(dx-dy);
//CASO UNO (MOVIMIENTO DE IZQUIERDA A DERECHA)
      if (y1<y2)
      {
       x=x1;       y=y1;       y_end=y2;
      while (y<y_end)
      {
        y=y+1;
      if (p<0)
        p=p+cons1;
      else
        {
       x=x+1;       p=p+cons2;
        }
       Canvas->Brush->Color=clYellow;
       Canvas->RoundRect(x,y,x+diam,y+diam,x+diam,y+diam);
       ahora=GetTickCount();
       do; while(GetTickCount()-ahora<espera);
      }
     }
// TERMINA PRIMER CUADRANTE

//CASO DOS (MOVIMIENTO DE DERECHA A IZQUIERDA)
    else
      {
       // Caso dos
       x=x1;       y=y1;       y_end=y2;
       while (y_end<y)
       {
        y=y-1;
        if (p<0)
          p=p+cons1;
        else
         {
         x=x-1;         p=p+cons2;
         }
        Canvas->Brush->Color=clRed;
        Canvas->RoundRect(x,y,x+diam,y+diam,x+diam,y+diam);
```

```
        ahora=GetTickCount();
        do; while(GetTickCount()-ahora<espera);
       }
     } // TERMINA CUARTO CUADRANTE
    } // TERMINA PENDIENTE POSITIVA MAYOR A 1
  }   //TERMINA PENDIENTE POSITIVA

// SE EVALUA PENDIENTE NEGATIVA
 else
  {
   if (dy<dx)
//PENDIENTE NEGATIVA ENTRE 0 y 1
    {
 // CASO UNO (MOVIMIENTO DE IZQUIERDA A DERECHA)
     if (x1<x2)
     {
      x=x1;     y=y1;     x_end=x2;
      while (x<x_end)
      {
       x=x+1;
       if (p<0)
        p=p+cons1;
       else
        {
         y=y-1;        p=p+cons2;
        }
       Canvas->Brush->Color=clOlive;
       Canvas->RoundRect(x,y,x+diam,y+diam,x+diam,y+diam);
        ahora=GetTickCount();
        do; while(GetTickCount()-ahora<espera);
      }
     }
// TERMINA SEGUNDO CUADRANTE

// CASO DOS (MOVIMIENTO DE DERECHA A IZQUIERDA)
     else
     {
      x=x1;      y=y1;      x_end=x2;
      while (x_end<x)
      {
       x=x-1;
       if (p<0)
        p=p+cons1;
       else
        {
         y=y+1;         p=p+cons2;
        }
       Canvas->Brush->Color=clWhite;
       Canvas->RoundRect(x,y,x+diam,y+diam,x+diam,y+diam);
       ahora=GetTickCount();
       do; while(GetTickCount()-ahora<espera);
      }
     }
// TERMINA CUARTO CUADRANTE
    }
// TERMINA PENDIENTE NEGATIVA ENTRE 0 Y 1

//PENDIENTE NEGATIVA MAYOR A 1
   else
    {
      p=2*dx-dy;
      cons1=2*dx;
      cons2=2*(dx-dy);
    if (x2<x1)
// CASO UNO (MOVIMIENTO DE DERECHA A IZQUIERDA)
      {
       x=x1;      y=y1;      y_end=y2;
       while (y<y_end)
       {
        y=y+1;
```

```
        if (p<0)
         p=p+cons1;
        else
         {
          x=x-1;          p=p+cons2;
         }
         Canvas->Brush->Color=clBlue;
         Canvas->RoundRect(x,y,x+diam,y+diam,x+diam,y+diam);
         ahora=GetTickCount();
         do; while(GetTickCount()-ahora<espera);
       }
      }
//TERMINA SEGUNDO CUADRANTE

// CASO DOS (MOVIMIENTO DE  IZQUIERDA A DERECHA)
      else
      {
       x=x1;      y=y1;      y_end=y2;
       while (y_end<y)
        {
         y=y-1;
         if (p<0)
          p=p+cons1;
         else
          {
          x=x+1;        p=p+cons2;
          }
          Canvas->Brush->Color=clBlue;
          Canvas->RoundRect(x,y,x+diam,y+diam,x+diam,y+diam);
          ahora=GetTickCount();
          do; while(GetTickCount()-ahora<espera);
         }
      }
// TERMINA CUARTO CUADRANTE
     }
// TERMINA PENDIENTE NEGATIVA MAYOR A 1
  }
// TERMINA PENDIENTE NEGATIVA
  x_mid_rec[paso]=xp_end;
  y_mid_rec[paso]=yp_end;
   x1=x_mid_rec[k];
   y1=y_mid_rec[k];
   x2=x_mid_rec[k+1];
   y2=y_mid_rec[k+1];
 } //FIN DE MOVIMIENTO DEL ROBOT
} //*****************FIN DE PROGRAMA***************
//---------------------------------------------------------------------------
```

REFERENCIAS

Agur. Atlas de Anatomía de Grant. Ed. Panamericana. 9ª Edición. 1994.

Anderson T.L., Donath M. (1.988) " A Computational Structure for Enforcing Reactive Behavior in a Mobile Robot". Mobile Robot III. Proc. of SPIE conference Vol. Cambridge, MA.

Arkin, R. (1987) "Motor Schema Based Navigation for a Mobile Robot: An Approach to Programming by Behaviours". Proc. of 1987 IEEE Conference on Robotics and Automation.

Bézier, P. (1972) "Numerical Control" mathematics and Applications. A. R. Forrest, Trans. London: Wiley

Borenstein J., Koren Y. (1989) "Real-Time Avoidance for Fast Mobile Robots". IEEE Transactions on Systems, Man, and Cibernetics. Vol. 19, No 5, septiembre/octubre de 1.989. pp 1.179-1.234.

Brian Salomon Maxim Garber Ming C. Lin Dinesh Manocha. Interactive Navigation in Complex Environments Using Path Planning

Cox J.I. (1991)"Blanche-An Experiment in Guidance of Autonomous Robot Vehicle". IEEE Transactions on Robotic and Automation. Vol 7, No 2, abril de 1.991, pp 193-204.

Domingo G., Otto C., Francisco F., Ramón R., Generación de trayectorias robustas mediante computación evolutiva (Tecnología Informática y Computación, Universidad de Alicante, España)

Domingo G., Otto C., Francisco F., Pilar A., Patricia C., Ramón R., Control local de robots móviles basado en métodos estadísticos y algoritmos genéticos (Tecnología Informática y Computación, Universidad de Alicante, España)

Donald Hearn, M. Pauline Baker, "Gáficas por Computadora"

Foley, J. D., Van Dam, A., Feiner, S. K., Hughes, J. F. y Phillips, R. L.: "Introduction to Computer Graphics". Addison-Wesley, Publishing Company, Reading, MA, 1994.

Forsyth, D.A. y Ponce, J.; "Computer Vision: A Modern Approach", capítulo 4, Prentice Hall, 2003.

Francisco Charte Ojeda, "Guía práctica para usuarios, C++ Builder 5"

Francisco J. Portilla M, Tesis Doctoral: "Representación local multiescala de imágenes. Modelado y síntesis de texturas " Universidad Politécnica de Madrid

Goltsev A.D., Kussul E.M., 1987, Control de transporte robot basada a información de medición de distancias. *Automática*, V.26, N4, pp.25-29.

Goltsev A.D., Kussul E.M., 1982, Red neuronal para móvil robot control. *Automática y telemecánica*. V.36, N5, pp. 44-51.

González J. (1993) "Estimación de la Posición y Construcción de Mapas para un Robot Móvil Equipado con un Escáner Láser Radial". Tesis Doctoral. Universidad de Málaga.

Haber, R. N., and L. Wilkinson (1982) "Perceptual Components of Computer Displays" IEEE Computer Graphics and Applications

Harrington, S. (1983) "Computer Graphics: A Programming Approach. New York": McGraw-Hill

Hawrylyshyn, P. A., R. R. Tasker, and L. W. Oregan (summer 1977). "Computer Assisted Stereotaxic Surgery"

Hopgood, F. R. A., (1983). "Introduction to the Graphical Kernel System (GKS)" London: Academic Press

Howie Choset, Marco La Civita, Jong Chul Park. Path Planning between Two Points for a Robot Experiencing Localization Error in Known and Unknown Environments

Humberto M., Miguel Ángel Z., German V., Juan Pedro C., Quaky – II: Robot Móvil para Interiores (Ingeniería de la Informática y las Comunicaciones, Universidad de Murcia, España)

J. Pomares, F. Torres, Miembro, Control Visual Basado en Flujo de Movimientos para el Seguimiento de Trayectorias con Oclusiones (IEEE)

Janich K. (1984) "Topology". Springer-Verlag, New-York.

Kanayama Y., Miyake (1985) "Trajectory Generation for Mobile Robots". Proc. of 3rd International Symposium on Robotic Research. Editores: O. D. Faugeras y G. Giralt. Gouvierux, Francia pp 333-340.

Kanayama Y., Hartman B.I. (1990) "Smooth Local Path Planning for Autonomous Vehicles". Autonomous Robot Vehicles. Editores I.J. Cox y G.T. Wilfong. Springer-Verlag. pp 62-67.

Latombe J.C. (1991) "Robot Motion Planning". Kluwer academic publishers. ISBN: 0-7923-9129-2.

Levi P. (1987) "Principles of Planing and Control Concepts for Autonomous Mobile Robots". Proc. of 1987 IEEE International Conference on Robotics and Automation. pp 74-81.

Lozano – Pérez T. (1990) “Foreword: Mobile Robot and Robotics”. Autonomous Robot Vehicle. Editores I. J. Cox y G.T Wilfong springer – Verlag. Pp 7 – 11

María Concepción M., Roberto Guzmán M., Rocío Alaiz R., Autoguiado de robots móviles mediante redes neuronales (Escuela de Ingeniería e Informática. Universidad de León)

Martínez J.L. (1994) “Seguimiento Automático de Caminos en Robots Móviles”. Tesis Doctoral. Universidad de Málaga.

Nelson L. W., Cox I. J. (1990) “Local Path Control for an Autonomous Vehicle”. Autonomous Robot Vehicles. Editores I.J. Cox y G.T. Wilfong. Springer-Verlag. pp 38-44.

Newman P. A., Kempf K.G. (1985) “Opportunistic Scheduling for Robotic Machine Tending”. Proc. of Second Conference on Artificial Intelligence Applications. Pp 168-175.

Nelson L. W. (1988) “Continuous Steering Function Control of Robot Cart”. IEEE Transactions on Industrial Electronics.

Nelson L. W., Cox I. J. (1990) “Local Path Control for an Autonomous Vehicle”. Autonomous Robot Vehicles. Editores I.J. Cox y G.T. Wilfong. Springer-Verlag. pp 38-44.

Nilsson N.J. (1.969) “ A Mobile Automaton: An Application of Artificial Intelligence Techiques”. Proc. of the 1st. International Joint Conference on Artificial Intelligence. pp 509-520.

Ollero A., Mandow A., Gómez de Gabriel J., Muñoz V. (1994) “Autonomous Mobile Robot Operation and Navigation in Industrial Environments”. Proc. of European Robotics and Intelligent Systems Conference (EURISCON’94). Vol. 1, pp 150- 158.

Shangming Wei and Miloˇs ˇ Zefran. Smooth Path Planning and Control for Mobile Robots

Sherr, S. (1979). “Electronic Displays” New York: Wiley

Shin D. H., Singh S. (1990) “ Path Generation for Robot Vehicles Using Composite Clothoid Segments”. The Robotics Institute, Carnegie-Mellon University. Internal Report CMU-RI-TR-90-31.

Segovia A., Rombaut M. (1993) “Continuous Curvature Path Finding for a Non-Holonomic Mobile Robot“. 1st International Workshop on Intelligent Autonomous Vehicles (IAV’93). pp 481-486.

Steven C. Chapra, Raymond P. Canale, “Métodos Numéricos para Ingenieros” cuarte edición

Thorphe C. (1984) “FIDO: Vision and Navigation for a Robot Rover”. Ph. D. Thesis, Carnegie Mellon University.

Tomás M., David S., Celestino M., José María M., Planificación de Movimientos en Vehículos no Holonómicos Mediante Aprendizaje por Refuerzo (Departamento de Física, Ingeniería de Sistemas y Teorías de la Salud, Universidad de Alicante, España)

Traducción al Español del libro (Ed. Kussul E.):
N.M.Amosov, T.N.Baidyk, A.D.Goltsev, A.M.Kasatkin, L.M.Kasatkina, E.M.Kussul, D.A.Rachkovski, 1991, Neurocomputadoras y robots inteligentes. Traductores Baidyk T., Velasco G., México, D.F., 2006 (preparado para publicación).

Walter Savitch, "Solución de problemas con C++, El objetivo de la programación", segunda edición

Watanabe Y., Yuta S. (1990) "Position Estimation of Mobile Robots with Internal and External Sensors Using Uncertainty Evolution Technique". Proc. of 1990 IEEE International Conference on Robotics and Automation. pp 2.011-2.016.