UNIVERSIDAD NACIONAL DEL SUR

Tesis de Doctor en Ingeniería

Integración de SLAM y planificación para el control de vehículos terrestres

Juan Martín Pinna Cortiñas

BAHÍA BLANCA ARGENTINA

2012

Prefacio

Esta Tesis se presenta como parte de los requisitos para optar al grado Académico de Doctor en Ingeniería, de la Universidad Nacional del Sur y no ha sido presentada previamente para la obtención de otro título en esta Universidad u otra. La misma contiene los resultados obtenidos en investigaciones llevadas a cabo en el Instituto de Investigaciones en Ingeniería Eléctrica «Alfredo Desages», dentro del Departamento de Ingeniería Eléctrica y de Computadoras (DIEC), durante el período comprendido entre el 18 de julio de 2006 y el 29 de diciembre de 2010, bajo la dirección de los Profesores Dres. Favio Román Masson y Jorge Luis Moiola, Profesores Adjunto y Asociado del Departamento de Ingeniería Eléctrica y de Computadoras (DIEC).

10 de junio de 2012

Juan Martín Pinna Cortiñas

INSTITUTO DE INVESTIGACIONES
EN INGENIERÍA ELÉCTRICA
«ALFREDO DESAGES»

DEPARTAMENTO DE INGENIERÍA
ELÉCTRICA Y DE COMPUTADORAS

UNIVERSIDAD NACIONAL DEL SUR

UNIVERSIDAD NACIONAL DEL SUR
Secretaría General de Posgrado y Educación Continua

La presente tesis ha sido aprobada el ..../..../....., mereciendo la calificación de ..... (.....).

Resumen

En esta tesis, el algoritmo de SLAM —localización y mapeo simultáneo— ha sido integrado con técnicas de planificación, con el fin de obtener trayectorias para vehículos terrestres en ambientes exteriores. El algoritmo de SLAM permite que el vehículo obtenga una representación del ambiente en un mapa variante en el tiempo al que podrían asociarse múltiples capas de datos, donde estas capas pueden incluir información diversa tal como: presencia de obstáculos, pendiente o altura del terreno, tipo de suelo, etc. Este mapa es dividido en regiones con alta correlación interna utilizando técnicas de geometría altamente eficientes desde el punto de vista computacional, con estructuras de datos y abstracciones que permitan manejar grandes cantidades de información. Estas regiones a su vez son divididas logrando una resolución en dos niveles de la información. Una que se denomina global y divide a todo el mapa conocido, y otra local que fracciona cada una de esas regiones. Con esta representación del ambiente, o en forma simultánea con la adquisición de ésta, es posible hallar trayectorias mediante técnicas de planificación. Con este fin, se definen criterios que permitan utilizar en la planificación las múltiples capas de información permitiendo la utilización de cualquier algoritmo existente para este propósito. Esta aproximación permite lograr un algoritmo que es más eficiente que cualquier algoritmo de planificación existente para ambientes en exploración como se enfrentan en aplicaciones de SLAM. Finalmente, como ejemplo de la estrategia propuesta, se generan trayectorias a partir de un modelo cinemático de un vehículo de cuatro ruedas y una estrategia de control. Se elige la velocidad lineal deseada del vehículo y se observa la limitación en el ángulo de giro del volante, con el objetivo de evaluar el desempeño del mismo mediante simulaciones numéricas y datos experimentales.

Abstract

In this thesis, the SLAM algorithm —Simultaneous Localization and Mapping— has been integrated with planning strategies in order to achieve trajectories for land vehicles in outdoor environments. The SLAM algorithm allows the vehicle to obtain a representation of the environment in a time varying map. This time varying map could contain multiple layers of information, where these layers may include many types of data such as obstacles, slope or height of the terrain, etc. The map is divided into regions that have a high correlation using computational geometry techniques, which are highly efficient from the computational point of view. Towards this end, data structures and abstractions which can handle a significant amount of data are defined. Additionally, these regions are divided achieving two levels of inform: global and local. At global level, the whole known map is divided and at local level, each region is partitioned. Either once the representation of the environment is achieved or simultaneously with this process, it is possible to find trajectories using planning techniques. Criteria are defined to make use of the multiple layers of information at the planning stage and a particular type of planning algorithm is used, but there is a wide range of possible algorithms. This approach allows to achieve an algorithm that is more efficient than any existing planning algorithm for environments under exploration. Finally, to exemplify the proposed strategy, a kinetic model of a car–like vehicle is used with a control strategy to obtain trajectories. In these trajectories, the speed of the vehicle is chosen roughly constant and the limits of the steering angle are observed. The performance of the whole scheme is assessed through numerical simulations and experimental data.

Índice general

1 Introducción **1**
1.1 Contexto y motivación 1
1.2 Contribuciones 3
1.3 Organización de la tesis 6

2 Conceptos preliminares **9**
2.1 Navegación y mapeo simultáneo 9
2.1.1 Navegación y mapeo mediante probabilidad 10
2.1.2 Requisitos de tiempo y memoria 12
2.1.3 Información auxiliar 13
2.2 Triangulación de las regiones 13
2.2.1 Tiangulación de un conjunto de puntos 14
2.2.2 La triangulación de Delaunay 14
2.3 Montículo de Fibonacci y búsqueda 20
2.3.1 Montículo de Fibonacci 20
2.3.2 Búsqueda óptima 22
2.4 Uso dinámico de la memoria 25
2.5 Arquitectura 25
2.6 Conclusiones del capítulo 26

3 Triangulación del mapa y planificación **27**
3.1 Definición de las regiones 27
3.1.1 Triangulación en el interior de las regiones 32
3.2 Enlazando regiones y subdivisiones 37
3.3 Planificación óptima 39
3.3.1 Planificación local 40
3.3.2 Planificación global 43
3.4 Complejidad computacional 48
3.5 Mediciones de complejidad computacional 52
3.6 Conclusiones del capítulo 59

4 Generación de trayectorias y control **61**
4.1 Reconstrucción de trayectorias 62
4.2 Modelo cinemático y limitaciones 64
4.3 Estrategia de control 69
4.4 Unificación de planificación y control 71
4.5 Expansión del mapa 79
4.6 Conclusiones del capítulo 81

5 Simulaciones y datos experimentales **83**
5.1 Planificación en un ambiente no estructurado 83
5.2 Trayectorias controladas 90
5.3 Implementación del controlador 95
5.4 Criterio de planificación 97
5.5 Conclusiones del capítulo 97

6 Conclusiones generales **101**

Apéndices **107**

A Triángulos auxiliares **107**

B Implementación de regiones y subdivisiones **111**

C Puntos trayectoria **117**

Índice de figuras

2.1 Cuatro puntos en el plano permiten obtener dos triangulaciones posibles. 14
2.2 Triangulación de Delaunay del conjunto de puntos. 15
2.3 Casos posibles de triangulación con un triángulo existente y un punto nuevo. 19
2.4 División de un triángulo por un punto interior. 20
2.5 Tiempo de ejecución del algoritmo D* Básico y D* Enfocado. . 24

3.1 Primer punto o mojón c a procesar dentro del triángulo auxiliar inicial. 29
3.2 Punteros de los nodos que representan los tres triángulos que resultan de procesar el primer mojón interior al triángulo auxiliar. 29
3.3 Un triángulo existente con vértices p_1, p_2 y p_3 es modificado por un nuevo mojón c. 31
3.4 Los nuevos triángulos son representados por tres nodos nuevos en el grafo. Los nuevos nodos se conectan entre ellos y con los nodos existentes. 31
3.5 Se agregan puntos en el borde de la región local triangular con el fin de hacerla accesible desde los vecinos. Los puntos se encuentran espaciados uniformemente de acuerdo con una densidad elegida en forma previa. 33
3.6 A partir de los puntos que se encuentran en el interior y en los bordes de la región triangular local, se genera una subdivisión de la misma usando una triangulación. 35
3.7 Tres regiones triangulares: $p_1\,p_2\,p_3$, $p_1\,p_4\,p_2$ y $p_1\,p_5\,p_4$. La subdivisión resaltada en la región triangular $p_1\,p_4\,p_2$ tiene subdivisiones adyacentes en dos regiones distintas. 38
3.8 La subdivisión indicada con un triángulo es adyacente a dos subdivisiones en dos regiones distintas. Punteros para enlazar subdivisiones en diferentes regiones. 39
3.9 Las subdivisiones en los bordes de la región son hipotéticamente numeradas con el objeto de construir la matriz de costos de transición entre ellas. 42

3.10 Ejemplo de camino óptimo entre las subdivisiones 1 y 2 de los bordes de la región. 43
3.11 Ejemplo de camino óptimo entre las subdivisiones 1 y 3 de los bordes de la región. 44
3.12 Las tres regiones se representan mediante los nodos de las subdivisiones en los bordes. 46
3.13 Dos regiones adyacentes apuntadas por los vectores $xPtr$ e $yPtr$. Índices de entrada y salida. 47
3.14 El símbolo γ indica la posición inicial del vehículo y la letra χ representa el destino deseado. Camino inicial y final agregado a la trayectoria. 49
3.15 La subdivisión resaltada que se encuentra en la esquina de la región tiene dos punteros hacia subdivisiones adyacentes mientras que una subdivisión del centro cuenta con tres punteros. 51
3.16 Costo temporal en triangular las regiones. 53
3.17 Costo temporal predicho y experimental de triangular las subdivisiones interiores. 55
3.18 Costo de unir regiones de acuerdo al número de accesos. Resultados de predicciones y experimentos. Los valores absolutos del eje vertical no son representativos pero sí lo es la tendencia de los puntos. 56
3.19 Costo temporal predicho de la planificación local de todos los caminos dependiendo del número de accesos. Los valores absolutos del eje vertical no son representativos pero sí lo es la tendencia de los puntos. 57
3.20 Costo temporal experimental de la planificación local de todos los caminos dependiendo del número de accesos. Los valores absolutos del eje vertical no son representativos pero sí lo es la tendencia de los puntos. 58
3.21 Costo temporal predicho y experimental en obtener la trayectoria óptima global mediante el algoritmo de Dijkstra. 59

4.1 Coordenadas x_d e y_d, el ángulo del chasis θ_d, las velocidades lineal v_{d1} y angular v_{d2} del chasis para un vehículo de cuatro ruedas. Vista superior de la parte delantera del móvil. 66
4.2 Ángulo de la dirección ϕ_d, distancia entre ejes l_m y a_m es la anchura para un vehículo de cuatro ruedas. 67
4.3 Las ruedas de la izquierda del vehículo se encuentran describiendo el radio mínimo de giro r_m y el centro del mismo describe un círculo de radio inferior, $R = r_m - a_m/2$. 68

4.4 Los asteriscos indican los puntos medios de los lados adyacentes de las subdivisiones que conforman el camino óptimo entre γ y χ. Trayectoria única a partir de la planificación. . . 73
4.5 El eje horizontal indica los trechos desde el comienzo hasta el final del camino. El eje vertical muestra la distancia en metros. 75
4.6 Componente de velocidad lineal $z_{d1}^{\cdot}$ a utilizarse como referencia en el controlador. 76
4.7 Componente de velocidad lineal $z_{d2}^{\cdot}$ a utilizarse como referencia en el controlador. 77
4.8 Para cada una de las posiciones que resultan de filtrar los puntos medios de los lados adyacentes de la trayectoria, se agrega un vector cuya longitud es proporcional a la velocidad lineal requerida. Se observa que la misma es uniforme adrede. 78
4.9 Expansión del mapa. 80
4.10 Nodos resultantes inmediatamente después de agregar el nuevo mojón y antes de calcular si los lados de los nuevos triángulos son válidos. 80
4.11 Los enlaces que se agregan inmediatamente después de incluir el nuevo mojón y antes de verificar si la triangulación resultante es válida. 80

5.1 Lugar de ejecución del algoritmo de SLAM: «Victoria Park», «Sydney». 85
5.2 Puntos elegidos para obtener la altitud del parque. 86
5.3 Datos de altitud extraídos de información de GPS para la zona de interés. 87
5.4 Resultados de planificación óptima global y simulación numérica a partir de un conjunto de datos experimentales. . . . 88
5.5 Amplificación de la figura anterior. Se muestra solamente una parte de la trayectoria planificada y controlada. 89
5.6 Resultado de implementar el controlador. Ángulo del chasis del vehículo para velocidad uniforme. 91
5.7 Resultado de implementar el controlador. Ángulo de la dirección del vehículo para velocidad uniforme. 92
5.8 Resultado de implementar el controlador. Velocidad lineal del vehículo para referencia de velocidad uniforme. 93
5.9 Resultado de implementar el controlador. Velocidad angular del vehículo para referencia de velocidad uniforme. 94
5.10 Resultado de implementar el controlador. Ángulo de la dirección del vehículo a partir de integración numérica mediante la regla de Euler. 96

5.11 Tres trayectorias óptimas con diferentes valores de k_2. Los fondos de las figuras representan la altura del terreno y las zonas más claras se corresponden con alturas mayores. Se indican el origen y el destino. 98

A.1 Tres puntos c_1, c_2 y c_3 son procesados mediante la triangulación de Delaunay. 108
A.2 Para determinar si un triángulo es auxiliar, pueden darse cuatro casos al momento de dividir un triángulo existente ijk con un punto c. El punto l es el vértice de un triángulo adyacente. 108

C.1 Baricentro o centro de masa del triángulo ABC. Los ejes representan la distancia en metros. 118
C.2 Tres regiones triangulares y sus subdivisiones interiores. . . . 119

Índice de tablas

2.1 Un punto puede encontrarse en el interior, exterior o sobre los bordes de un triángulo: casos posibles. 18
2.2 Tiempos de ejecución de las operaciones utilizando un montículo de Fibonacci. 21

4.1 Resultado de ejecutar la planificación global para el caso de la Fig. 3.14 63
4.2 Resultado de ejecutar la planificación global para el caso de la Fig. 3.14. Versión amplificada de la tabla a los fines de incluir los tramos iniciales y finales. Se resaltan tres tramos de trayectoria: γ–3, 3–8 y 3–χ. 64

B.1 Implementación computacional en C++ de los nodos que representan las regiones. 113
B.2 Implementación computacional en C++ de los nodos que representan subdivisiones de las regiones. 114

Notación

I, II y III	Triángulos.
a	Parámetro Ec. círculo.
a_m	Ancho chasis.
a_t	Constante interior triángulo.
`alti,altj,altk`	Punteros C++. Altura vértices.
b_t	Constante interior triángulo.
b_x	Parámetro Ec. círculo.
b_y	Parámetro Ec. círculo.
c	Parámetro Ec. círculo.
c_0	Constante.
c_{ij}	Componente de la matriz anterior.
$\mathbf{C}$	Matriz de costos entre subdivisiones.
$\mathscr{C}_{z_1 z_2}$	Costo de desplazarse entre nodos z_1 y z_2.
`ch1,ch2,ch3`	Punteros C++ a descendientes de subdivisión.
d	Densidad de puntos/m.
D_i	Orden del cálculo de todos los caminos.
E	Número de arcos del grafo.
E_i	Número de arcos del grafo interior.
E_r	Número de enlaces entre regiones.

$f(n)$	Función.
$g(n)$	Función.
`G`	Puntero C++ al *grafo local*.
$\mathbf{h}$	Vector.
id, id_0	Índices que indican por donde se entró y salió.
ir_1, ir_2	Índices a subdivisiones adyacentes.
k_1, k_2	Constantes en la función de costo de transición.
k_i	Números de puntos borde región.
k_m	Número de mojones en el borde del mapa.
k_{pi}	Ganancias de control.
l	Índice $\in \mathbf{Z}$.
l_m	Distancia entre ejes.
$\mathbf{l}_1, \mathbf{l}_2, \mathbf{l}_3$	Lados de un triángulo.
m	Índice $\in \mathbf{Z}$.
$\mathbf{m} = \{\mathbf{m}_0, \mathbf{m}_1, ..., \mathbf{m}_n\}$	Vector de todos los mojones.
$\mathbf{m}_i$	Mojón i-ésimo.
n_0	Constante.
n_i	Número de puntos interiores región i-ésima.
n_i^a	Número de subdivisiones bordes región i-ésima.
n_m	Número de mojones del mapa.
$O(\cdot)$	Orden de un algoritmo.
`obstaculo`	Variable C++ que indica si es un obstáculo.
`op_i,op_j,op_k`	Punteros C++ subdivisiones adyacentes.
$\mathbf{p_i}$	Punto.
`P_{x}`	Vector C++ grilla x.
`P_{y}`	Vector C++ grilla y.

$\mathscr{P} = \{\mathbf{p_1}, \mathbf{p_2}, ..., \mathbf{p_n}\}$	Conjunto de puntos.
`p1x,p2x,p3x`	Punteros C++. Vértices de las regiones.
`p1y,p2y,p3y`	Punteros C++. Vértices de las regiones.
`ptrCostos`	Puntero C++ a arreglo de dos dimensiones (costos).
q	Índice $\in \mathbf{Z}$.
Q	Cola de prioridades.
r	Radio del círculo.
$\boldsymbol{r}$	Vector de acciones de control.
r_i	Acciones de control.
r_m	Radio mínimo de giro.
R	Radio de trayectoria circular.
$\mathscr{S}$	Subdivisión.
$T_{01}, \ldots, T_{04}$	Triángulos auxiliares.
T_a	Triángulo adyacente a T_n en la triangulación.
T_i	Triángulo.
T_n	Nuevo triángulo en la triangulación.
$\mathscr{T}$	Triangulación.
$\mathbf{u}$	Vector.
$\mathbf{u}_k$	Vector de acciones de control.
$\mathbf{U}_{0:k} = \{\mathbf{u}_0, \mathbf{u}_1, ..., \mathbf{u}_k\}$	Vector de todas las acciones de control.
$\mathbf{v}$	Punto interior triángulo.
v_{d1}	Velocidad lineal vehículo.
v_{d2}	Velocidad angular vehículo.
V	Número de vértices del grafo.
V_i	Número de vértices del grafo interior.
V_r	Número de regiones.

`vectorB`	Puntero C++ a vector de transiciones a subdivisiones en regiones adyacentes.
`vectorRegionAdy`	Elemento C++: vector de dos dimensiones que indica las subdivisiones en los bordes de las regiones adyacentes.
w_{pi}	Puntos en el borde de una región.
$\mathbf{w}_i$	Vector.
x_c	Coordenada x del centro del círculo.
x_d e y_d	Posición vehículo plano.
$\mathbf{x}_k$	Vector de estados.
$\mathbf{X}_{0:k} = \{\mathbf{x}_0, \mathbf{x}_1, ..., \mathbf{x}_k\}$	Vector de todas las posiciones hasta instante k.
y_c	Coordenada y del centro del círculo.
z_{di}	Posición deseada (controlador).
$\dot{z_{di}}$	Velocidad deseada (controlador).
$\mathbf{z}_{ik}$	Mojón i-ésimo en el instante k.
$\mathbf{z}_k$	Observación.
$\mathbf{Z}_{0:k} = \{\mathbf{z}_0, \mathbf{z}_1, ..., \mathbf{z}_k\}$	Vector de todas las observaciones de mojones.
$*pr_1$	Puntero a regiones adyacentes.
$*pr_2$	ídem.

Letras griegas

γ	Posición inicial del vehículo.
δ	Escalar. Distancia al punto control.
θ_d	Ángulo del chasis vehículo.
$\boldsymbol{\mu}$	Posición fuera del vehículo (control).
ϕ_d	Ángulo de la dirección del vehículo.

$\overline{\phi_d}$	Ángulo máximo de la dirección del vehículo.
χ	Posición final del vehículo.

Capítulo 1

Introducción

1.1 Contexto y motivación

Los vehículos autónomos en ambientes exteriores resultan de interés en aplicaciones tales como la agricultura, la minería, la exploración espacial, transporte en puertos y los vehículos subacuáticos. En el caso de la agricultura robótica, los vehículos autónomos resultan de particular interés [3, 30, 31, 24] y además permiten reducir el impacto ambiental [44]. En el caso de la minería [41], se utilizan vehículos de gran porte y se dan situaciones que son inaccesibles para los seres humanos por razones de seguridad que requieren de vehículos con cierta autonomía. Por otro lado, las aplicaciones espaciales involucran temperaturas extremas, vacío, radiación y grandes distancias, haciéndolas hostiles para los seres humanos [55], y en estos casos la operación a distancia no siempre es viable presentando riesgos para los robots [4]. Asimismo, en las actividades relacionadas con el transporte de cargas y las aplicaciones marítimas, existe en la actualidad interés en planificar y supervisar las tareas mediante vehículos robó-

ticos [11] y en estos ambientes también tendrían aplicación los vehículos subacuáticos autónomos.

Actualmente existen técnicas que permiten generar trayectorias para vehículos terrestres a partir de modelos conocidos de los mismos [16], otras que permiten representar el ambiente mediante localización y mapeo [18] y algoritmos que permiten planificar y hallar trayectorias óptimas [22]. La combinación de estas dos últimas técnicas (*localización y mapeo* con *planificación*) permitiría que los vehículos se desplazaran por ambientes desconocidos y generaran un mapa del ambiente pero al mismo tiempo, se podrían planificar trayectorias óptimas; lo que le permitiría alcanzar objetivos, adaptándose a los cambios del ambiente y la incertidumbre, presentando ventajas sobre algoritmos existentes tales como D* [45]. Es importante tener en cuenta que el algoritmo D* sólo permiten cambios locales en mapas conocidos y sin incertidumbre. Además, el algoritmo D* tiene un costo computacional considerable. Finalmente, una vez que se halla la trayectoria óptima, la misma puede ser ejecutada por las técnicas de control existentes.

Es interesante observar que existe una clasificación de los algoritmos de planificación. En general, se dividen en dos categorías: combinatorios y basados en muestras [22]. Los primeros se refieren a los algoritmos exactos y los segundos se consideran algoritmos aproximados. En el desarrollo de esta tesis no se realizarán aproximaciones en la etapa de planificación por lo que puede afirmarse que el algoritmo presentado es de tipo combinatorio.

1.2 Contribuciones

Debido a la naturaleza multidisciplinaria de la robótica, las contribuciones de esta tesis son diversas y se detallan a continuación:

- Se aplicaron con éxito técnicas de computación geométrica que permiten generar eficientemente las entidades definidas como regiones. Para esto se definen estructuras de datos abstractas que permiten manejar grandes cantidades de información en tiempo constante sin agregar complejidad a la que requiere generar una triangulación de un conjunto de puntos.
- Se utilizaron técnicas de computación geométrica que permiten subdividir las regiones. Con este fin, se define una serie de puntos artificiales que permiten que las regiones queden enlazadas en forma unívoca y de esta forma facilitar la planificación.
- Las estructuras definidas permiten incorporar datos sensados en el ambiente en forma eficiente y sin alterar la estructura generada por lo que se pueden incorporar múltiples capas de información sobre el ambiente [38]. Esto permite planificar con cualquier técnica de planificación existente en la literatura.
- La representación del ambiente utilizada se basa en el algoritmo de SLAM, que tiene vasta difusión y aceptación, lo que permite trabajar con mapas con incertidumbre [38].
- Se define un criterio de planificación que combina múltiples capas de información y permite ponderar qué tan relevante resulta cada capa al momento de hallar una trayectoria óptima [36].
- Se define un concepto de planificación local que permite trabajar den-

tro de entidades conocidas como regiones, de forma tal que la información almacenada dentro de las regiones puede cambiar en forma local y lo planeado a este nivel puede evaluarse nuevamente pero sin afectar lo planificado en las demás regiones [38]. Esto presenta una ventaja ya que no se requiere ejecutar la planificación sobre todo el mapa. Esto es similar a lo que se obtiene utilizando el algoritmo conocido como D*, aunque éste se limita a aceptar mapas variables puntualmente y no maneja mapas con incertidumbre, que es el caso presentado. Es importante aclarar que la incertidumbre no va a ser utilizada explícitamente por la técnica de planificación sino que se encuentra hereda al representar el mapa con técnicas de SLAM.

– El concepto de planificación global permite hallar una trayectoria óptima entre la posición actual del vehículo y cualquier destino que se encuentre dentro o fuera del mapa. Esta planificación reutiliza todas las planificaciones previas, por lo que resulta altamente eficiente.
– El concepto de planificación local y global no distorsiona la trayectoria obtenida respecto de la que se obtendría ejecutando la planificación con todos los datos disponibles al momento. Esto se debe a que cuando se planifica a nivel local, no se descarta ninguna trayectoria posible (no hay ningún tipo de aproximación) y por lo tanto el óptimo obtenido a partir de los óptimos locales coincide con el que se obtendría ejecutando la planificación sobre todo el mapa.
– Con respecto a las trayectorias, se ideó una técnica que permite reconstruir las trayectorias que fueron generadas en la etapa de planificación.
– Se obtuvo una forma de combinar las referencias generadas por un

módulo de planificación con estrategias de control existentes en la literatura [36, 33]. Se unificaron de dos formas distintas: la primera resulta en una velocidad lineal variable para un vehículo terrestre y la segunda permite obtener una velocidad prácticamente uniforme. Esta última presenta la ventaja que permitiría un desplazamiento uniforme del vehículo. La técnica que se presenta no aparta al vehículo de la trayectoria óptima hallada y no genera acciones de control exageradas o difíciles de implementar.

– Se esboza una técnica que permite que el algoritmo de triangulación de Delaunay pueda ser ejecutado sobre subconjuntos del conjunto de puntos a triangular. La triangulación final es exactamente igual a la que se obtendría ejecutando la triangulación sobre el total de los puntos. Esta forma de trabajo permite aplicar el esquema completo en forma creciente sobre datos parciales del ambiente y alterar lo menos posible la representación ya lograda.
– Desde el punto de vista geométrico computacional, se ideó una forma de obtener los puntos medios de una tira de triángulos en tiempo constante y refinar este resultado con un filtro existente para obtener una trayectoria suave que no se aparte significativamente de la trayectoria planificada.
– El módulo de planificación diseñado no asume características del vehículo ni del mapa (salvo que el mapa debe contar con mojones) por lo que sigue los principios modernos de la robótica de trabajar con módulos que sean reutilizables para diversas plataformas y mapas. La aplicación se limitó a vehículos terrestres pero, por lo dicho anteriormente, el planteo permitiría vehículos de diversa naturaleza.

– Conocimientos previos en robótica fueron publicados en [20, 35] mientras que estudios sobre compensadores PID fueron presentados en [39, 34].
– Por otra parte, el modelo cinemático de un vehículo terrestre fue estudiado en [37]. Además, una aplicación de una cámara y de un sensor láser pueden verse en [27].

1.3 Organización de la tesis

El Capítulo 2 presenta técnicas y conceptos existentes, necesarios para comprender y entender los principios en los cuales se basa el algoritmo propuesto tales como: el algoritmo de localización y mapeo simultáneo, la triangulación de un conjunto de puntos, los montículos de Fibonacci, las técnicas de búsqueda, el uso dinámico de la memoria y la arquitectura del software.

El Capítulo 3 presenta la forma en que se definen las entidades conocidas como regiones locales, la manera en que estas regiones se enlazan entre sí y se dividen internamente, la estrategia de planificación en dos etapas —local y global— y la complejidad computacional.

El Capítulo 4 expone el modo en que las trayectorias pueden ser reconstruidas a partir de la planificación propuesta. A continuación, se presenta un modelo cinemático de un vehículo tipo auto de vasta aceptación en la literatura y se estudian sus limitaciones. Posteriormente se emplea una técnica de control, pero para esto se propone una forma de combinar la planificación y la técnica de control. Luego, se esboza una forma en que el mapa se puede expandir más allá de sus límites.

En el Capítulo 5, se muestran los caminos óptimos sobre un conjunto de datos experimentales y se obtienen las curvas de acciones de control para luego variar el criterio de planificación óptima.

Finalmente, en el Capítulo 6, se proponen conclusiones generales.

Capítulo 2

Conceptos preliminares

Este capítulo presenta conceptos que son necesarios para comprender el desarrollo de la tesis. En primer lugar, se introducirá la idea de navegación y mapeo simultánea (SLAM). Luego, interesará saber cómo puede dividirse el mapa obtenido mediante el algoritmo de SLAM en regiones que cubran todo el mapa. A continuación, se mostrarán algoritmos de búsqueda y estructuras de datos asociadas. Finalmente, se presentarán conceptos relacionados con la implementación mediante software.

2.1 Navegación y mapeo simultáneo

La idea de localización y mapeo simultáneo (SLAM) permite a un robot, que se encuentra en un ambiente desconocido, construir un mapa y simultáneamente ubicarse dentro del mismo [18]. Si bien hoy en día el problema de SLAM puede considerarse resuelto —ya sea desde el punto de vista conceptual o teórico—, existen extensiones de este algoritmo, tales como el algoritmo de DenseSLAM [29], que permiten utilizar mapas ricos en infor-

mación. Este algoritmo permite obtener una representación del ambiente que no sólo conserva la información esencial que permite realizar la localización y el mapeo, sino también conserva datos del ambiente adquiridos mediante sensores. Esta representación se vuelve relevante al momento de tomar decisiones o de planificar una trayectoria para el vehículo.

En el algoritmo de SLAM se definen una serie de cantidades [18], que a modo de reseña se presentan a continuación:

- $\mathbf{x}_k$: el vector de estados que describe la posición y la orientación del vehículo,
- $\mathbf{u}_k$: el vector de acciones de control aplicado en el tiempo $k-1$ y que conduce al vehículo al estado $\mathbf{x}_k$ en el tiempo k,
- $\mathbf{m}_i$: el vector que describe la ubicación del i-ésimo mojón, cuya ubicación real se asume invariante,
- $\mathbf{z}_{ik}$: el vector de observación del mojón i-ésimo en el instante k,
- $\mathbf{X}_{0:k} = \{\mathbf{x}_0, \mathbf{x}_1, ..., \mathbf{x}_k\}$: que representa el vector de todas las posiciones hasta el instante k,
- $\mathbf{U}_{0:k} = \{\mathbf{u}_0, \mathbf{u}_1, ..., \mathbf{u}_k\}$: el vector de todas las acciones de control hasta el momento k,
- $\mathbf{m} = \{\mathbf{m}_0, \mathbf{m}_1, ..., \mathbf{m}_n\}$: el vector de todos los mojones,
- $\mathbf{Z}_{0:k} = \{\mathbf{z}_0, \mathbf{z}_1, ..., \mathbf{z}_k\}$: el vector de todas las observaciones de los mojones.

2.1.1 Navegación y mapeo mediante probabilidad

El algoritmo de SLAM requiere que la distribución de probabilidad

$$\mathbf{P}(\mathbf{x}_k, \mathbf{m} | \mathbf{Z}_{0:k}, \mathbf{U}_{0:k}, \mathbf{x}_0) \tag{2.1}$$

sea calculada en cada instante k. Ésta representa la probabilidad conjunta a posteriori de los mojones y de la ubicación del vehículo en el tiempo k dadas las observaciones y las acciones de control hasta el tiempo k junto con la condición inicial del vehículo. En general se busca una representación recurrente comenzando con una estimación de la distribución en el tiempo $k-1$, la distribución a posteriori se calcula a partir de la acción de control $\mathbf{u}_k$ y la observación $\mathbf{z}_k$ utilizando la regla de Bayes.

El modelo de observación representa la probabilidad de que se produzca una observación $\mathbf{z}_k$ cuando la ubicación del vehículo y de los mojones son conocidas y se pueden describir como

$$\mathbf{P}(\mathbf{z}_k|\mathbf{x}_k, \mathbf{m}). \tag{2.2}$$

Además, el modelo del vehículo se representa como

$$\mathbf{P}(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{u}_k), \tag{2.3}$$

asumiendo que es un proceso de Markov, en el cual, el estado $\mathbf{x}_k$ depende sólo del estado inmediato anterior $\mathbf{x}_{k-1}$ y la acción de control aplicada $\mathbf{u}_k$ y no depende de la observación ni del mapa.

El algoritmo de SLAM puede ser implementado de forma recurrente en dos pasos: una predicción y luego una corrección. La predicción y la corrección son implementadas para calcular la distribución de probabilidad a posteriori de la Ec. 2.1 a partir del modelo de observación y del modelo de desplazamiento del vehículo de las Ecs. 2.2 y 2.3.

Es interesante observar que el mapa puede construirse a partir de la probabilidad condicional

$$\mathbf{P}(\mathbf{m}|\mathbf{X}_{0:k}, \mathbf{Z}_{0:k}, \mathbf{U}_{0:k}), \tag{2.4}$$

asumiendo que la posición actual del vehículo $\mathbf{x}_k$ es conocida en todo momento y también es conocida la posición inicial. Entonces, el mapa $\mathbf{m}$ es construido fusionando observaciones. Además, la posición del vehículo puede calcularse a partir de la distribución de probabilidad

$$\mathbf{P}(\mathbf{x}_k|\mathbf{Z}_{0:k}, \mathbf{U}_{0:k}, \mathbf{m}). \tag{2.5}$$

Esta última ecuación asume que la ubicación de los mojones es conocida con certeza y el objetivo es calcular la estimación de la posición del vehículo con respecto a estos mojones.

2.1.2 Requisitos de tiempo y memoria

Se puede decir que la implementación del algoritmo de SLAM utilizando el filtro de Kalman extendido requiere $O(n_m^2)$, donde n_m es el número de mojones del mapa y se sigue la notación dada en [42]; donde $O(\cdot)$ indica que una función $g(n)$ es de orden $O(f(n))$ si existen constantes c_0 y n_0 tales que $g(n) < c_0\, f(n)$ para todo $n > n_0$. Esta notación permite clasificar los algoritmos de acuerdo a la cota superior de su tiempo de ejecución y a la cota superior de su espacio ocupado.

Mucho esfuerzo se ha concentrado en reducir el tiempo de ejecución del algoritmo de SLAM [12]. Algunos métodos se basan en limitar el área de trabajo a una región, otros sacrifican precisión y otros trabajan con mapas parciales en vez de mapas completos [1].

2.1.3 Información auxiliar

En el algoritmo de SLAM, además de buscar información que sea relevante para el mapeo, es posible asociar información con la posición tal como: salinidad del terreno, humedad, temperatura, características del terreno, etc. Esta información puede ser utilizada para ayudar a la asociación de datos o para tareas que no estén relacionadas con el mapeo tales como generación de trayectorias mediante planificación o recolección de datos [1].

Este concepto de agregar información auxiliar es el utilizado en un método denominado DenseSLAM [29]. En este caso, a medida que el robot se desplaza, la información auxiliar es almacenada en una estructura de datos tal como las grillas de ocupación [19] dentro de regiones que se encuentran determinadas mediante un conjunto de mojones en el mapa de SLAM. A medida que el mapa evoluciona y los mojones se mueven (ya sea porque el algoritmo de SLAM actualice alguno de los mojones o por movimiento relativo entre mojones), las zonas se desplazan pero la información representada en forma local dentro de ellas se mantiene consistente utilizando las estimaciones de los mojones dadas por el algoritmo de SLAM [19].

2.2 Triangulación de las regiones

El algoritmo de SLAM permite obtener el mapa m que contiene la posición de los mojones. Para obtener una representación densa del ambiente, interesa dividir el mapa en regiones con el fin de representar la información auxiliar [38]. Para esto se introducen conceptos como la triangulación de un conjunto de puntos y una triangulación en particular, conocida como la triangulación de Delaunay.

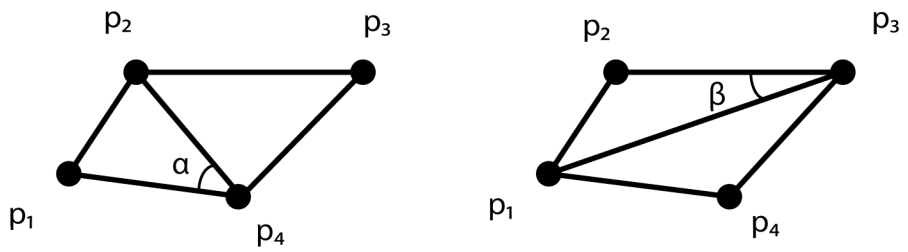


Figura 2.1: Cuatro puntos en el plano permiten obtener dos triangulaciones posibles. En el caso de la izquierda, el ángulo interno mínimo de los triángulos α, resulta mayor que el ángulo mínimo de la derecha β.

2.2.1 Tiangulación de un conjunto de puntos

Dado un conjunto de puntos $\mathscr{P} = \{\mathbf{p_1}, \mathbf{p_2}, ..., \mathbf{p_n}\}$ en el plano, una subdivisión máxima en el plano de los puntos se define como una subdivisión $\mathscr{S}$ en la cual no existe ningún arco que conecte dos vértices que pueda ser agregado a $\mathscr{S}$ sin que se produzca una intersección con alguno de los arcos existentes. Una triangulación de $\mathscr{P}$ se define como una máxima división en el plano cuyos vértices se encuentran en $\mathscr{P}$ [15].

Otro concepto interesante es el de triangulación válida. Cuatro puntos en un plano pueden formar dos triángulos y existen dos combinaciones posibles. De estas dos combinaciones posibles, una se denomina no válida si la otra combinación produce un ángulo interno mínimo mayor que el ángulo mínimo actual, como se observa en la Fig. 2.1. Este concepto se utiliza para definir la triangulación de Delaunay.

2.2.2 La triangulación de Delaunay

Dado un conjunto de puntos en el plano $\mathscr{P}$, y sea $\mathscr{T}$ una triangulación de $\mathscr{P}$. Se define a $\mathscr{T}$ como una triangulación de Delaunay de $\mathscr{P}$ sí y sólo sí el círculo formado por los vértices de cualquier triángulo de $\mathscr{T}$ no contiene

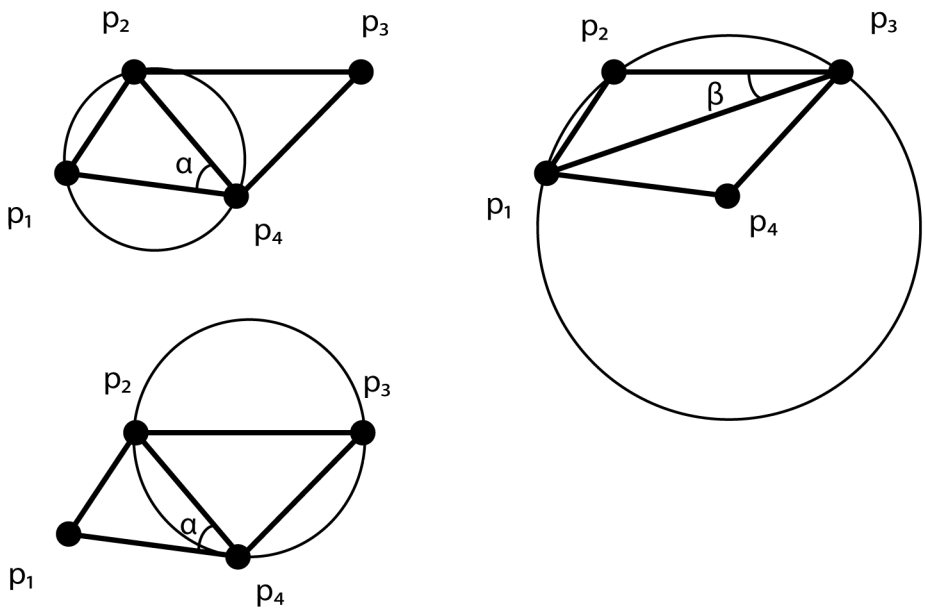


Figura 2.2: En la figura se observa que en la triangulación de la izquierda, los círculos formados por los vértices de los triángulos no contienen ningún punto en su interior mientras que en la figura de la derecha, no ocurre lo mismo. Se observa que la triangulación de la izquierda es una triangulación de Delaunay del conjunto de puntos.

ningún punto de $\mathscr{P}$ en su interior. En la Fig. 2.2 se observa que en el caso de la triangulación de la derecha, el círculo formado por los vértices de los triángulos contiene un punto en su interior, mientras que en la triangulación de la izquierda ocurre lo contrario. Se observa que en el caso de la izquierda, se tiene una triangulación de Delaunay del conjunto de puntos.

Inmediatamente de lo anterior surge que cualquier triangulación de Delaunay es válida y luego una triangulación óptima en ángulo de $\mathscr{P}$ es una triangulación de Delaunay de $\mathscr{P}$. Estos conceptos son los que permiten definir un algoritmo para calcular este tipo de triangulación.

Interior círculo

Para determinar qué puntos se encuentran en el interior de un círculo se utiliza el siguiente procedimiento:

– Primero se determina la ecuación de un círculo que pasa por tres puntos [48]

$$a(x^2 + y^2) + b_x\, x + b_y\, y + c = 0, \tag{2.6}$$

$$\begin{aligned}
a &= \begin{vmatrix} x_{\mathbf{p_1}} & y_{\mathbf{p_1}} & 1 \\ x_{\mathbf{p_2}} & y_{\mathbf{p_2}} & 1 \\ x_{\mathbf{p_3}} & y_{\mathbf{p_3}} & 1 \end{vmatrix}, \\
b_x &= -\begin{vmatrix} x^2_{\mathbf{p_1}} + y^2_{\mathbf{p_1}} & y_{\mathbf{p_1}} & 1 \\ x^2_{\mathbf{p_2}} + y^2_{\mathbf{p_2}} & y_{\mathbf{p_2}} & 1 \\ x^2_{\mathbf{p_3}} + y^2_{\mathbf{p_3}} & y_{\mathbf{p_3}} & 1 \end{vmatrix}, \\
b_y &= \begin{vmatrix} x^2_{\mathbf{p_1}} + y^2_{\mathbf{p_1}} & x_{\mathbf{p_1}} & 1 \\ x^2_{\mathbf{p_2}} + y^2_{\mathbf{p_2}} & x_{\mathbf{p_2}} & 1 \\ x^2_{\mathbf{p_3}} + y^2_{\mathbf{p_3}} & x_{\mathbf{p_3}} & 1 \end{vmatrix}, \\
c &= -\begin{vmatrix} x^2_{\mathbf{p_1}} + y^2_{\mathbf{p_1}} & x_{\mathbf{p_1}} & y_{\mathbf{p_1}} \\ x^2_{\mathbf{p_2}} + y^2_{\mathbf{p_2}} & x_{\mathbf{p_2}} & y_{\mathbf{p_2}} \\ x^3_{\mathbf{p_3}} + y^3_{\mathbf{p_3}} & x_{\mathbf{p_3}} & y_{\mathbf{p_3}} \end{vmatrix}.
\end{aligned}$$

– Se calcula el centro del círculo y su radio como

$$x_c = -\frac{b_x}{2\, a}, \tag{2.7}$$

$$y_c = -\frac{b_y}{2\,a}, \tag{2.8}$$

$$r = \frac{\sqrt{b_x^2 + b_y^2 - 4\,a\,c}}{2\,|a|}. \tag{2.9}$$

– Finalmente, se calcula la distancia del centro del círculo al punto y si la distancia es inferior al radio del círculo, se trata de un punto interior.

Interior triángulo

A los fines de la triangulación, es necesario saber si un punto v se encuentra dentro de una región triangular. Para esto se sigue un procedimiento similar al descripto en [52]; en donde a partir de las coordenadas de los vértices y del punto que se quiere saber si se encuentra dentro del triángulo, pueden calcularse dos constantes a_t y b_t en la ecuación $\mathbf{v} = \mathbf{p_1} + a_t\,\mathbf{v_1} + b_t\,\mathbf{v_2}$, donde $\mathbf{v_1} = \mathbf{p_2} - \mathbf{p_1}$, $\mathbf{v_2} = \mathbf{p_3} - \mathbf{p_1}$ y a_t y b_t son dos escalares dados por

$$a_t = \frac{\det(\mathbf{v}\,\mathbf{v_2}) - \det(\mathbf{p_1}\,\mathbf{v_2})}{\det(\mathbf{v_1}\,\mathbf{v_2})}, \tag{2.10}$$

$$b_t = -\left[\frac{\det(\mathbf{v}\,\mathbf{v_1}) - \det(\mathbf{p_1}\,\mathbf{v_1})}{\det(\mathbf{v_1}\,\mathbf{v_2})}\right], \tag{2.11}$$

$$\det(\mathbf{u}\,\mathbf{h}) = \mathbf{u_x}\,\mathbf{h_y} - \mathbf{u_y}\,\mathbf{h_x}. \tag{2.12}$$

En nuestro caso, interesa no sólo determinar si el punto se encuentra dentro o fuera del triángulo sino también si se encuentra sobre los bordes.

Tabla 2.1: Un punto puede encontrarse en el interior, exterior o sobre los bordes de un triángulo: casos posibles.

a_t	b_t	$a_t + b_t$	Caso
> 0	> 0	< 1	Interior
$= 0$	> 0	< 1	Lado 2
> 0	$= 0$	< 1	Lado 1
> 0	> 0	$(= 1)$	Lado 3

Por esto, dependiendo de los valores que tomen estas constantes pueden argüirse los casos que se detallan en la Tabla 2.1 y que se muestran en la Fig. 2.3.

Construcción

Para realizar la triangulación de un conjunto de puntos, se utiliza el criterio descripto en la Tabla 2.1. Si el punto se encuentra en el interior de un triángulo, el triángulo dentro del que se encuentra es dividido en tres triángulos y en caso de que un punto se encuentre sobre uno de los lados, el triángulo es dividido en dos. Para lograr una representación abstracta eficiente, se utiliza un *grafo acíclico dirigido*. Por *grafo* se entiende una estructura que contiene *nodos* y *arcos* que los conectan, siendo los nodos objetos que tienen nombres y pueden contar con otros campos de información asociados; *acíclico* quiere decir que no se puede regresar al punto de partida si se recorren los nodos y *dirigido* ya que dado un arco, existen un nodo inicial y uno final [13].

Para implementar la triangulación, se comienza con un triángulo que abarca todos los puntos y se van procesando sucesivamente todos los puntos

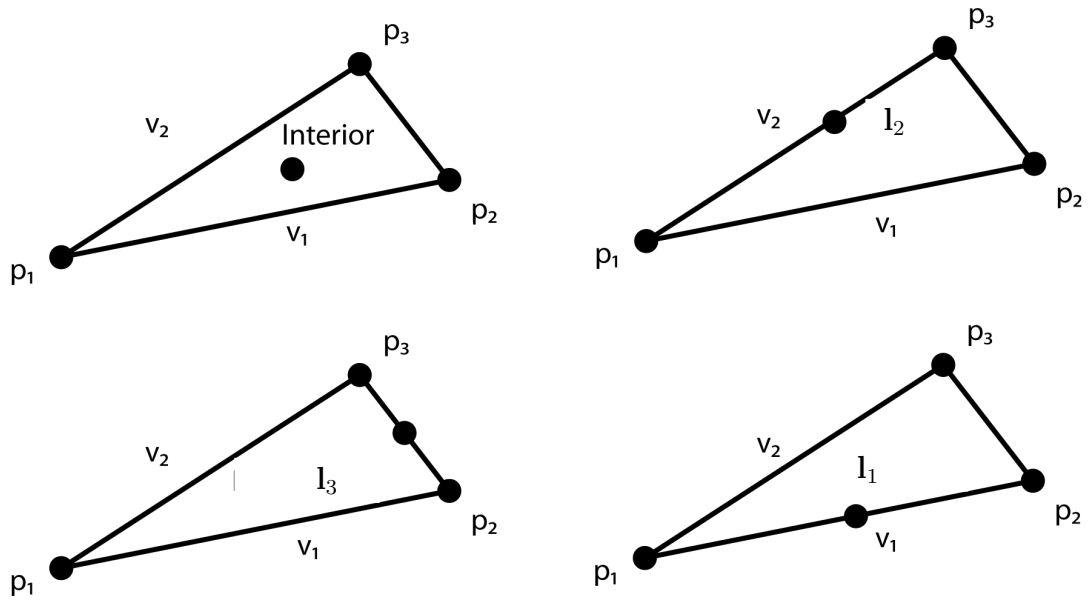


Figura 2.3: Casos posibles de triangulación con un triángulo existente y un punto nuevo. En el sentido de las agujas del reloj comenzando por la esquina superior izquierda: el punto puede encontrarse dentro, sobre el lado l_2 , sobre el lado l_1 o sobre el lado l_3.

a triangular. Dado que el triángulo inicial abarca todos los puntos, ya que se comienza con dos puntos artificiales y un mojón, se pueden producir los casos mencionados en la Tabla 2.1 y geométricamente ocurre lo mostrado en la Fig. 2.4. Si el punto se encuentra en el interior del triángulo; la figura se divide en tres mientras que si se encuentra en uno de los lados, ésta se divide en dos. En el grafo, se inicia con un nodo que representa el triángulo inicial y dependiendo del caso, se agregarán dos o tres descendientes [15]. De esta forma, se obtiene un grafo que representa una triangulación de los puntos. Si además de triangular, se realizan las operaciones de la Ec. 2.6, se puede determinar si los triángulos que se están formando son válidos y en caso que no lo fuesen cambiar la triangulación. La ventaja de este procedimiento es que puede obtenerse la triangulación de Delaunay de los

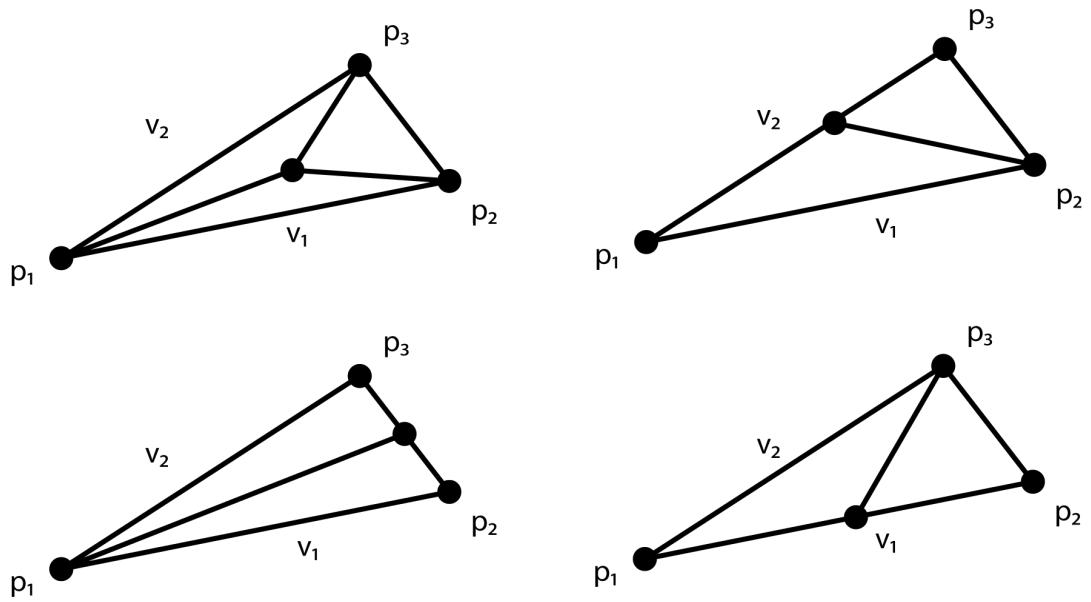


Figura 2.4: División de un triángulo por un punto interior. Comenzando por la esquina superior izquierda en sentido horario: punto interior, sobre el lado l_2, sobre el lado l_1 y sobre el lado l_3. Cuando el punto se encuentra en el interior, la figura se divide en tres; mientras que en los restantes casos se divide en dos.

puntos en un tiempo $O(n_p \log n_p)$ y utilizando un espacio $O(n_p)$, donde n_p representa el número de puntos a triangular.

2.3 Montículo de Fibonacci y búsqueda

Esta sección presenta definiciones sobre la estructura de datos a utilizar y el algoritmo necesario para realizar una búsqueda óptima dentro de la estructura.

2.3.1 Montículo de Fibonacci

Los montículos de Fibonacci consisten en una colección de árboles con raíz. Por *árbol* se entiende una colección de *vértices* o *nodos* y *arcos* que

Tabla 2.2: Tiempos de ejecución de las operaciones utilizando un montículo de Fibonacci.

Operación	Orden
Insertar	$O(1)$
Acceder Mínimo	$O(1)$
Quitar Mínimo	$O(\log n)$
Union	$O(1)$
Disminuir Clave	$O(1)$
Borrar	$O(\log n)$

cumplen ciertos requisitos: la propiedad que define un árbol es que exista un solo camino que conecte dos nodos. Otra definición importante es la de *árbol con raíz* donde uno de los nodos del árbol es designado como la *raíz* y existe un único camino entre ella y todos los demás nodos [42].

En el caso de los montículos de Fibonacci, sus árboles no están restringidos a ser árboles binarios, es decir que cada nodo no necesariamente cuenta exactamente con dos descendientes, y estos árboles no están restringidos a encontrarse ordenados o sea a cumplir ciertas propiedades respecto de cuál es el descendiente de la derecha y cuál el de la izquierda. Las operaciones que se realizan en el montículo requieren un tiempo de $O(1)$ mientras que no haya que borrar un nodo o elemento como puede verse en la Tabla 2.2. La ventaja de utilizar esta estructura de datos se obtiene cuando la cantidad de operaciones de tipo *Quitar Mínimo* y *Borrar* es relativamente baja comparada con el número de ejecuciones de las otras operaciones. La desventaja que posee es su alta complejidad de ser programada [13].

Cada nodo m dentro del montículo posee un puntero $p[m]$ a su padre

y un puntero $hijo[m]$ a cualquiera de sus hijos. Los hijos de m se encuentran unidos en una lista doblemente circular. Cada hijo w tiene punteros $izquierda[w]$ y $derecha[w]$ que apuntan a los hermanos de la izquierda y de la derecha y el orden en el cual aparecen los hijos es arbitrario. El esquema de la estructura de datos y el análisis completo de la implementación y el tiempo de ejecución de las operaciones de la Tabla 2.2 se encuentra en [13].

2.3.2 Búsqueda óptima

Un algoritmo *recurrente* es uno que permite solucionar un problema dividiéndolo en uno o varios problemas más pequeños. Una implementación posible del algoritmo de Dijkstra se basa en este tipo de técnica.

El algoritmo de Dijkstra

El algoritmo de Dijkstra permite encontrar un único camino posible entre el origen y el destino elegido [17, 22, 33] de forma sistemática y óptima. Este algoritmo es una forma de *programación dinámica*. La *programación dinámica* es una técnica que permite que cualquier cálculo *recurrente*, donde puedan almacenarse los valores calculados anteriormente, reduzca el tiempo de ejecución del algoritmo. Esta técnica permite una reducción drástica en la cantidad de llamadas de *recurrencia* [42].

La implementación del algoritmo requiere que se utilice una cola de prioridades Q para analizar los nodos del grafo. Si la cola se implementa utilizando un montículo de Fibonacci el tiempo de ejecución está dado por $O(V \log V + E)$, donde V representa el número de vértices del grafo y E representa el número de arcos. En este caso se asume que todas las demás operaciones tales como determinar si un estado ya ha sido visitado se

resuelven en tiempo constante. El algoritmo en sí, en sus diferentes implementaciones posibles, puede verse en [22].

El algoritmo de Dijkstra y otros algoritmos de planificación tales como A*, que es una variante del algoritmo de Dijkstra, permiten no solamente hallar el costo entre el origen y el destino sino que también permite reconstruir el camino recorrido. Esto es de particular interés para aplicaciones en el campo de la robótica [22].

El algoritmo D*

El algoritmo conocido como D* introduce la idea de planificación en ambientes desconocidos, parcialmente conocidos y cambiantes [45]. Su implementación requiere del concepto de arcos que contengan el costo de desplazarse y de estados que tienen un costo de transición entre ellos asociado si son vecinos. Este algoritmo está pensado para cambios que se produzcan en un entorno del robot. Si se produce un cambio en el mapa, actualiza en forma local la planificación. Por lo que se aprecia que se limita a cambios en un entorno del robot y no tiene aplicación en cambios que se produzcan en grandes partes del mapa como se daría en una representación del mapa basada en SLAM. Además, a partir de datos de tiempo de ejecución publicados por el autor del algoritmo [46], se observa en la Fig. 2.5 que la actualización es de orden $O(n^2)$ en el caso del algoritmo D* básico. La versión *enfocada* de este algoritmo ofrece una mejora pero se aprecia la misma tendencia $O(n^2)$.

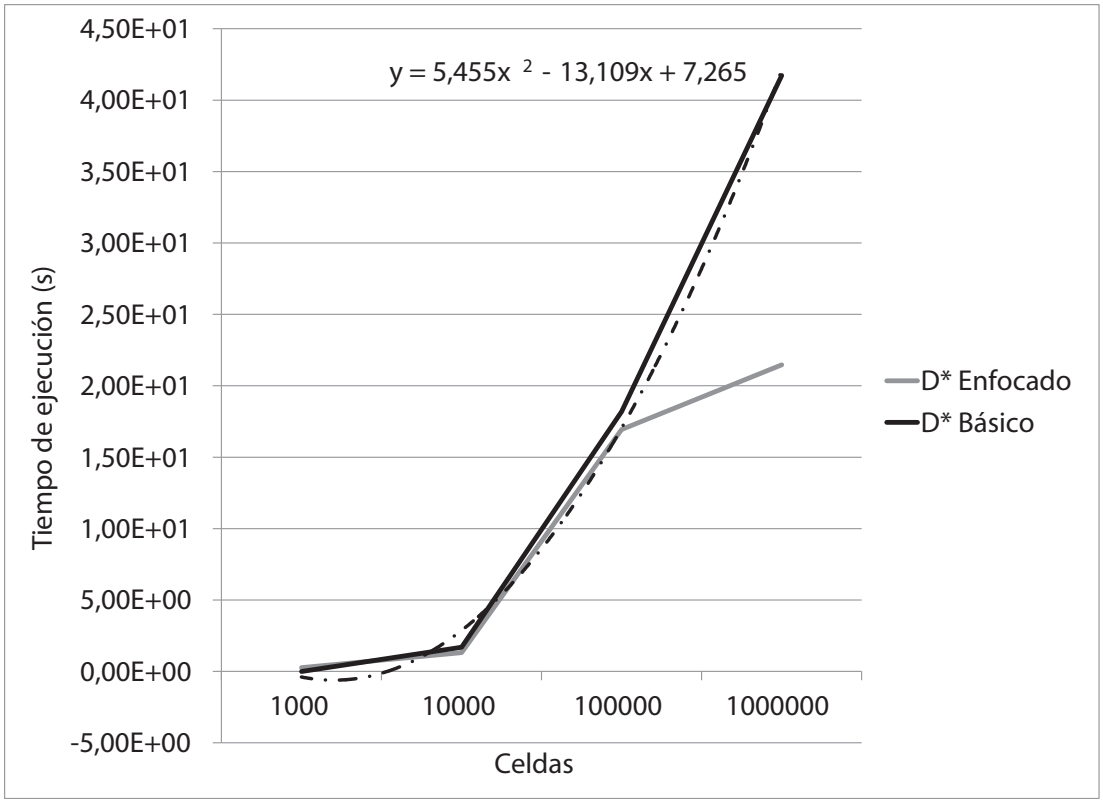


Figura 2.5: Tiempo de ejecución del algoritmo D* Básico y D* Enfocado. Se observa que el ajuste de los puntos del algoritmo D* Básico resulta de tipo $O(n^2)$. Además se observa que la versión Enfocada de este algoritmo ofrece una mejora pero se sigue observando la misma tendencia.

2.4 Uso dinámico de la memoria

Si el uso de la memoria en un sistema informático es *estático*, los elementos que se ubican en la memoria tienen un tamaño fijo, se conoce su tamaño en el momento de compilación y no pueden extenderse más allá de su tamaño [26]. Sin embargo, algunas aplicaciones no pueden existir con estas limitaciones y requieren que la memoria sea administrada de forma *dinámica*.

Existen básicamente dos formas de administrar la memoria en forma dinámica [14]:

- *Explícita*, donde se realiza una invocación para ubicar y destruir elementos.
- *Implícita*, que al no ser explícito, las partes de la memoria que no se utilizan más son liberadas automáticamente.

Hoy en día las aplicaciones han incrementado su complejidad y requieren mayor flexibilidad en el uso de la memoria [32], por lo que el uso dinámico de la memoria resulta necesario en algunos casos tales como en esta tesis.

2.5 Arquitectura

Esta propuesta va a requerir de herramientas de la *ingeniería del software* para poder ser implementada eficientemente tales como la *arquitectura del software*, que permite reducir la complejidad de los desarrollos para vehículos.

Un *sistema* puede considerarse como una serie de componentes que se relacionan o son independientes y forman un todo que provee una serie

de funciones con un propósito definido [8]. Se considera *arquitectura* a las principales decisiones de diseño de un *sistema* [47]. La arquitectura facilita el diseño de un sistema y el manejo de un proyecto [8] y de allí deriva su importancia.

2.6 Conclusiones del capítulo

Este capítulo presentó ideas y conceptos que permiten comprender el desarrollo de la tesis tales como el algoritmo de SLAM, que permite representar un mapa variante en el tiempo y con incertidumbre. Esta incertidumbre se manifiesta en la posición de los mojones y está manejada por el algoritmo de SLAM. A partir de este mapa y los algoritmos de triangulación, es posible dividir este mapa en regiones y luego en subdivisiones más pequeñas. Esta metodología será desarrollada en el próximo capítulo. Además, interesa planificar dentro de las regiones utilizando algún algoritmo de planificación tales como el presentado en este capítulo. Finalmente, se expusieron criterios asociados a la implementación mediante software tales como el uso de la memoria y la arquitectura.

Capítulo 3

Triangulación del mapa y planificación

Las grillas de ocupación [19] permiten obtener una representación del ambiente. Sin embargo, resta analizar una serie de cosas tales como: la forma en que se realiza la triangulación de las regiones, la división interna de las regiones, cómo se conectan las regiones con sus vecinas, las estructuras de datos, el cálculo de los costos de atravesar los caminos óptimos y el criterio que permite hallar un camino óptimo en el mapa. Todo lo enumerado anteriormente debe ser tratado para lograr esta representación.

3.1 Definición de las regiones

A partir del mapa que se obtiene en la Ec. 2.4 y de la triangulación de Delaunay, interesa realizar una triangulación del mapa en regiones locales. La elección de este tipo de triangulación se basa en que es óptima en ángulo. Esto permite obtener regiones triangulares donde el ángulo interior

mínimo de cada región es lo más grande posible. Esto quiere decir que se optimiza en este sentido permitiendo obtener triángulos lo menos agudos posible y que se prestan para representar información dentro de ellos.

Los mojones a triangular pueden elegirse entre todos los mojones disponibles. Un criterio de selección estaría basado en elegir los mojones más *rígidos*, entendiéndose por *rígidos* aquellos mojones que tienden a desplazarse menos. Este criterio es provisto por el algoritmo de SLAM.

Para triangular los mojones, se emplea un grafo acíclico dirigido. Este grafo se inicia con un nodo definido a partir de un punto del conjunto de puntos a triangular $\mathscr{P}$, que en este caso estaría dado por el vector de mojones. Del conjunto de puntos representados en el vector de mojones, se elige el punto $\mathbf{p_0}$, el cual tiene su coordenada en y de máximo valor. Esto asegura que todos los restantes puntos van a ser triangulados apropiadamente. Dos puntos restantes $\mathbf{p_{-1}}$ y $\mathbf{p_{-2}}$ se definen como puntos auxiliares a la triangulación. El primer punto auxiliar $\mathbf{p_{-1}}$ se elige con un valor positivo y relativamente grande de la coordenada x y coordenada y inferior al valor de coordenada y de todos los puntos en $\mathscr{P}$. El segundo punto auxiliar $\mathbf{p_{-2}}$ se elige con el mismo criterio para y y con coordenada en x negativa y relativamente grande también. El triángulo que resulta a partir de estos tres puntos (ver Fig. 3.1) es el que se emplea como primer nodo en el grafo acíclico dirigido.

Una vez que se ha generado el primer nodo, se procesan los restantes puntos en $\mathscr{P}$. Para esto, se elige cualquiera de los restantes puntos y se designa con c al mojón o punto de interés. En el caso del primer punto, el mismo siempre resulta dentro del triángulo auxiliar y éste se divide en tres triángulos como muestra la Fig. 3.1. En esta figura, el punto c resul-

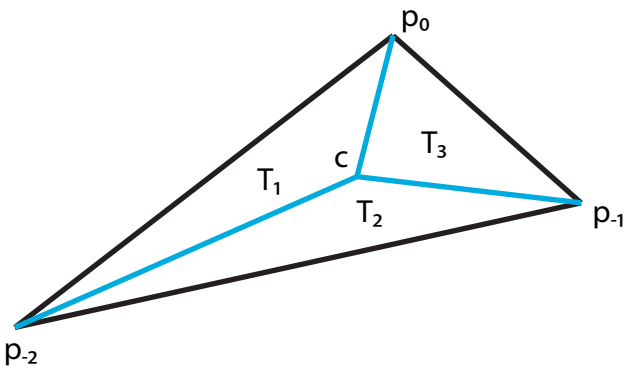


Figura 3.1: Triangulo auxiliar formado por p_0, p_{-1} y p_{-2} y el primer punto o mojón c a procesar. El triángulo auxiliar se divide en tres triángulos.

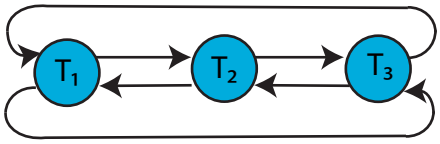


Figura 3.2: Punteros de los nodos que representan los tres triángulos que resultan de procesar el primer mojón interior al triángulo auxiliar.

ta siempre en el interior del triángulo auxiliar debido a como se eligen los tres puntos de los vértices. Los tres triángulos se colocan como descendientes del nodo que representa el triángulo auxiliar. De esta forma se va construyendo el grafo acíclico dirigido que permite triangular el conjunto de mojones o puntos.

En forma simultánea, interesa que las regiones queden enlazadas con las regiones adyacentes o vecinas para permitir en una etapa posterior la planificación. Los enlaces necesarios para el caso del primer mojón interior se muestran en la Fig. 3.2.

Cuando se procesan los restantes puntos, lo primero que se hace es determinar dentro de qué región triangular se encuentra el punto actual. Pa-

ra esto se determinan las constantes a_t y b_t en las Ecs. 2.10 y 2.11 y a partir de la Tabla 2.1 se determina si el punto se encuentra en el interior o en los bordes del triángulo con el que se calculan las constantes. Para esto, se procede en forma recurrente: se comienza con el triángulo auxiliar y si el punto se encuentra dentro del triángulo, se determina dentro de cuáles de los descendientes se encuentra y así sucesivamente hasta que se llega al triángulo más pequeño posible que lo contenga. Una vez que se alcanza este triángulo, el mismo es dividido en tres o en dos según corresponda. El caso en el cual el punto se halla en el interior del triángulo puede verse en la Fig. 3.3. Allí, el triángulo original T_0, formado por los puntos p_1, p_2 y p_3, es dividido en tres triángulos T_1, T_2 y T_3 que se agregan como tres descendientes del triángulo original T_0. De la misma forma se procede con todos los demás puntos en $\mathscr{P}$.

Una vez que el triángulo $p_0\, p_{-1}\, p_{-2}$ ha sido dividido, los nuevos triángulos deben ser enlazados no solamente entre sí, sino también con los triángulos adyacentes que ya existen en la estructura, como puede observarse en la Fig. 3.4. Los nuevos nodos se conectan entre sí y también con los nodos existentes representados por I, II y III.

Todo el procedimiento descripto anteriormente permite generar una triangulación. Pero si además interesa obtener una triangulación de Delaunay, cada vez que se genera un nuevo triángulo T_n, el mismo debe ser validado. Con este fin, se determinan los coeficientes de la Ec. 2.6 a partir de los tres vértices de T_n y un vértice del triángulo adyacente T_a y si se llega a la conclusión que el triángulo generado no es válido, se generan dos triángulos nuevos y válidos que se colocan como descendientes de los triángulos T_n y T_a. Estos a su vez deben ser validados con sus respectivos adyacentes.

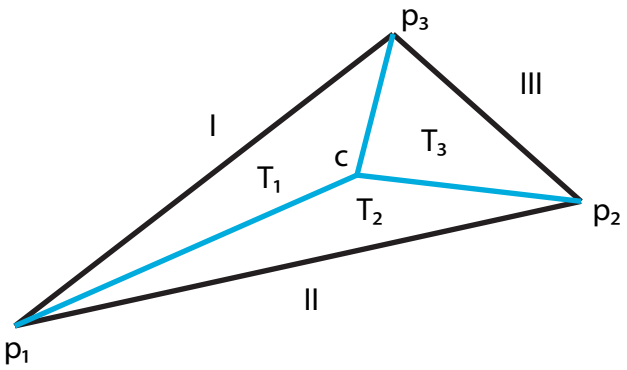


Figura 3.3: Un triángulo existente con vértices p_1, p_2 y p_3 es modificado por un nuevo mojón c. Tres nuevos triángulos T_1, T_2 y T_3 se agregan a la triangulación. Los triángulos vecinos se encuentran representados por I, II y III.

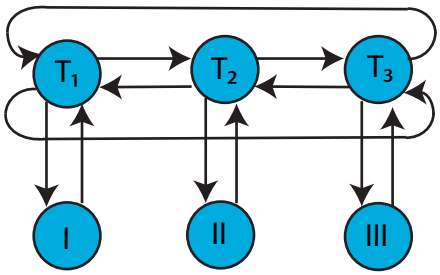


Figura 3.4: Los nuevos triángulos T_1, T_2 y T_3 son representados por tres nodos nuevos en el grafo. Los nuevos nodos se conectan entre ellos y con los nodos existentes I, II y III.

De esta forma, a medida que se construye el grafo acíclico dirigido que permite triangular los puntos que representan los mojones del mapa, se establece un grafo que permite determinar qué regiones locales son adyacentes. Es importante notar que durante todo este proceso, solamente se deben actualizar dos punteros por cada nodo que se modifica o se genera —y estas operaciones pueden realizarse en tiempo constante u $O(1)$— por lo que la generación de este segundo grafo no agrega mayor complejidad a la que ya existía debido a utilizar la triangulación de Delaunay. Este grafo va a ser de gran importancia posteriormente, al momento de planificar trayectorias óptimas.

Una vez que todos los puntos fueron procesados, interesa en algún momento determinar cuáles de los triángulos resultantes corresponden a la triangulación y cuáles son auxiliares. Este tema es tratado en el Apéndice A.

3.1.1 Triangulación en el interior de las regiones

Las regiones triangulares locales deben ser divididas. Con este fin se aplica una grilla de puntos que puede ser elegida en forma arbitraria o siguiendo algún tipo de patrón. Además, la resolución de los puntos de la grilla dentro de la región triangular local debe ser elegida apropiadamente ya que una grilla con una muy baja resolución puede llegar a sobrestimar el tamaño de los obstáculos en una representación del tipo de grillas de ocupación y puede llegar a fallar en proveer una representación útil del ambiente a los fines de la planificación. Por el contrario, una grilla con una muy alta resolución puede aumentar la complejidad de la planificación debido a que se podrían llegar a dar casos en que el vehículo no quepa

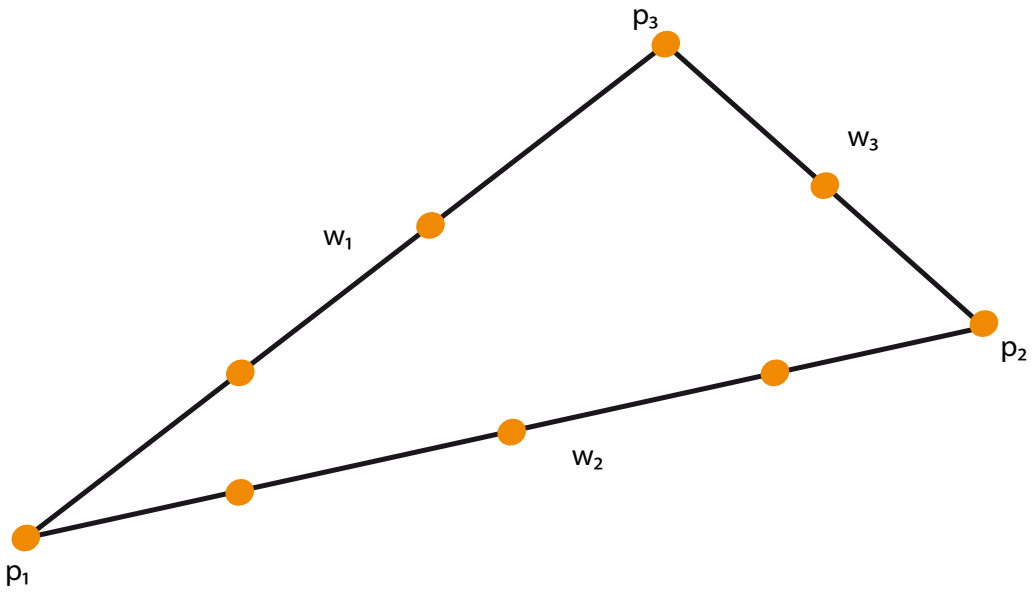


Figura 3.5: Se agregan puntos en el borde de la región local triangular con el fin de hacerla accesible desde los vecinos. Los puntos se encuentran espaciados uniformemente de acuerdo con una densidad elegida en forma previa. Los puntos p_1, p_2 y p_3 representan los vértices de la región, sobre tres mojones, a partir de los cuales los vectores w_1, w_2 y w_3 son formados.

dentro de las subdivisiones de la región o aún peor, haga desplazarse al vehículo muy cerca de obstáculos o genere situaciones poco favorables para él mismo.

Con objeto de que el vehículo pueda desplazarse entre las regiones locales triangulares, deben agregarse puntos sobre los bordes de las regiones. Estos puntos son elegidos de forma tal que permitan al vehículo desplazarse de una región triangular local a otra de forma unívoca ya que si la densidad de puntos sobre un lado de una región difiriese de la densidad de puntos del lado de la región adyacente, podría llegar a darse el caso en el cual existieran dos transiciones posibles. Esto dificultaría la planificación y la posterior generación de trayectorias viables para un vehículo. En la Fig. 3.5 puede verse que los vértices se indican por p_1, p_2 y p_3; mientras

que los puntos sobre los lados $\mathbf{w_1}$, $\mathbf{w_2}$ y $\mathbf{w_3}$ se notan $\mathbf{w_{p1}}$, $\mathbf{w_{p2}}$ y $\mathbf{w_{p3}}$ y están dados por

$$\begin{aligned}
\mathbf{w_{p1}} &= p_1 + \frac{l}{\lceil \|\mathbf{w_1}\| d \rceil} \mathbf{w_1}, \\
\mathbf{w_{p2}} &= p_1 + \frac{m}{\lceil \|\mathbf{w_2}\| d \rceil} \mathbf{w_2}, \\
\mathbf{w_{p3}} &= p_1 + \mathbf{w_1} + \frac{q}{\lceil \|\mathbf{w_3}\| d \rceil} \mathbf{w_3},
\end{aligned} \tag{3.1}$$

en este caso, la densidad de puntos por metro está dada por d y los índices l, m y $q \in \mathbb{Z}$ toman valores en los intervalos

$$\begin{aligned}
0 \leq l &\leq \lceil \|\mathbf{w_1}\| d \rceil, \\
0 \leq m &\leq \lceil \|\mathbf{w_2}\| d \rceil, \\
0 \leq q &\leq \lceil \|\mathbf{w_3}\| d \rceil,
\end{aligned} \tag{3.2}$$

donde el símbolo $\lceil \cdot \rceil$ representa el número entero superior y $\| \cdot \|$ denota el módulo del vector. La Fig. 3.5 muestra los puntos resultantes de las Ecs. 3.1 y 3.2.

Los puntos resultantes de la combinación de la grilla interior y los puntos agregados sobre los bordes de la región son triangulados con el fin de obtener una subdivisión de la región y un grafo local que represente esta subdivisión. La triangulación se realiza en forma análoga a la triangulación de las regiones y los nodos del grafo a utilizar en este caso representan subdivisiones de la región local. El resultado de aplicar este procedimiento puede verse en la Fig. 3.6, donde se observa que la combinación de los

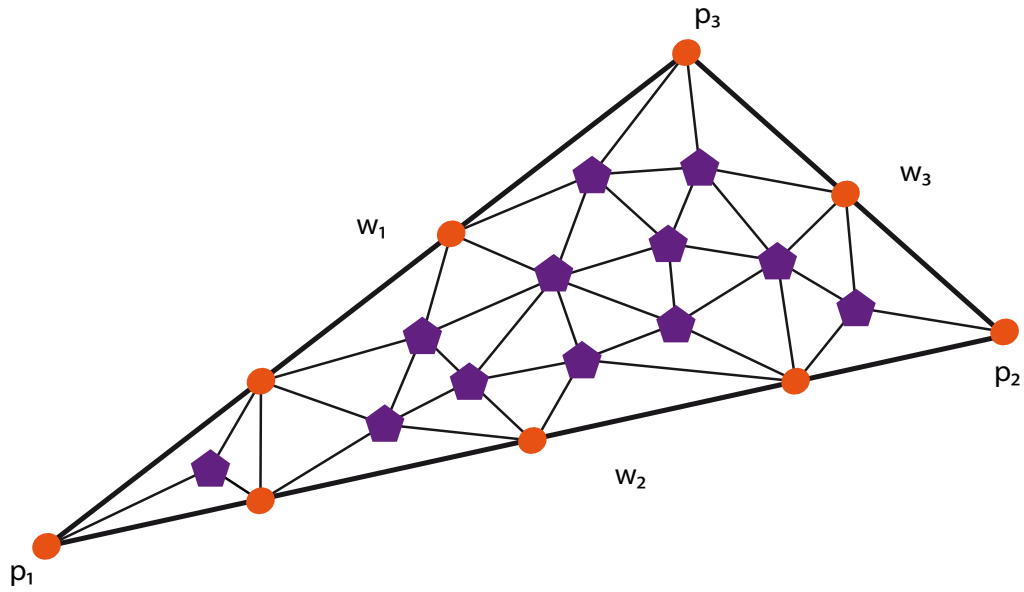


Figura 3.6: A partir de los puntos que se encuentran en el interior y en los bordes de la región triangular local, se genera una subdivisión de la misma usando una triangulación. Los pentágonos representan los puntos de la grilla, que podrían llegar a ser mediciones de alguna característica del ambiente, y los círculos los puntos sobre los bordes. Los puntos p_1, p_2 y p_3 son mojones del mapa.

puntos de los bordes y de la grilla da lugar a una triangulación interior. Finalmente, en forma análoga a lo que ocurre cuando se triangulan las regiones, aparecen triángulos auxiliares cuando se triangula en el interior de las regiones. En el Apéndice A se explica cómo discriminar los dos tipos de triángulos.

De esta forma se obtiene un *grafo local* que representa la subdivisión de la región local. Este grafo local va a permitir planificar dentro de la región en una etapa posterior. Es importante resaltar que si se incrementa el conocimiento del ambiente, se puede agregar esta información en forma altamente eficiente.

Agregando la información sensada a la región

A medida que se imcrementa el conocimiento del ambiente al aplicar el algoritmo de SLAM, esta información es volcada en los *grafos locales*. Para esto, lo primero que se realiza es determinar a qué región local pertenece el dato mediante el *grafo de regiones*. Este paso implica determinar cuál es la región que contiene a las coordenadas $x - y$ del dato. Una vez que se determina la región local, debe buscarse dentro del *grafo local* de esta región y finalmente el campo del nodo que almacena el dato es actualizado. Es importante notar que la búsqueda dentro de estos grafos es altamente eficiente: por ejemplo, en el caso del grafo de las regiones se comienza por el triángulo auxiliar y se determina dentro de cuál de sus tres descendientes se encuentra el punto y así en forma recurrente hasta llegar a la región dentro de la cual se encuentra el dato a agregar. Luego se procede en forma análoga con el *grafo local*.

3.2 Enlazando regiones y subdivisiones

A medida que las regiones se encuentran bien definidas, interesa saber de qué forma se encuentran unidas aquellas que son adyacentes mediante las subdivisiones que se hallan en los bordes de las mismas. Como se muestra en la Fig. 3.7, una subdivisión que se encuentra sobre los bordes de la región pero que no se ubica en la esquina de la misma, cuenta con exactamente una única subdivisión adyacente en la región vecina. En cambio, para el caso de las subdivisiones que se encuentran en las esquinas de una región; pueden llegar a tener dos subdivisiones vecinas en dos regiones diferentes. El procedimiento que se va a emplear para representar esto consiste en utilizar para cada una de las subdivisiones en los bordes un puntero y un índice. El puntero indica a qué región pertenece la subdivisión adyacente y el índice identifica la subdivisión dentro de la región.

En el caso de la subdivisión 1 de la región definida por el triángulo $p_1\,p_4\,p_2$ en la Fig. 3.7, la misma es adyacente a la subdivisión 1 de la región $p_1\,p_2\,p_3$ y a la subdivisión 5 de la región $p_1\,p_5\,p_4$. Para representar esto se utilizan dos punteros a estas regiones y dos índices como muestra la Fig. 3.8. Desde las subdivisiones en las regiones adyacentes ocurre algo similar ya que desde la subdivisión 5 del triángulo $p_1\,p_5\,p_4$ existe un puntero y un índice que la enlazan con la subdivisión 1 del triángulo $p_1\,p_4\,p_2$. Algo análogo ocurre con la subdivisión 1 de $p_1\,p_2\,p_3$.

Estos punteros e índices son los que en definitiva van a permitir la transición entre regiones, al momento de planificar la trayectoria óptima global, y de allí deriva su importancia.

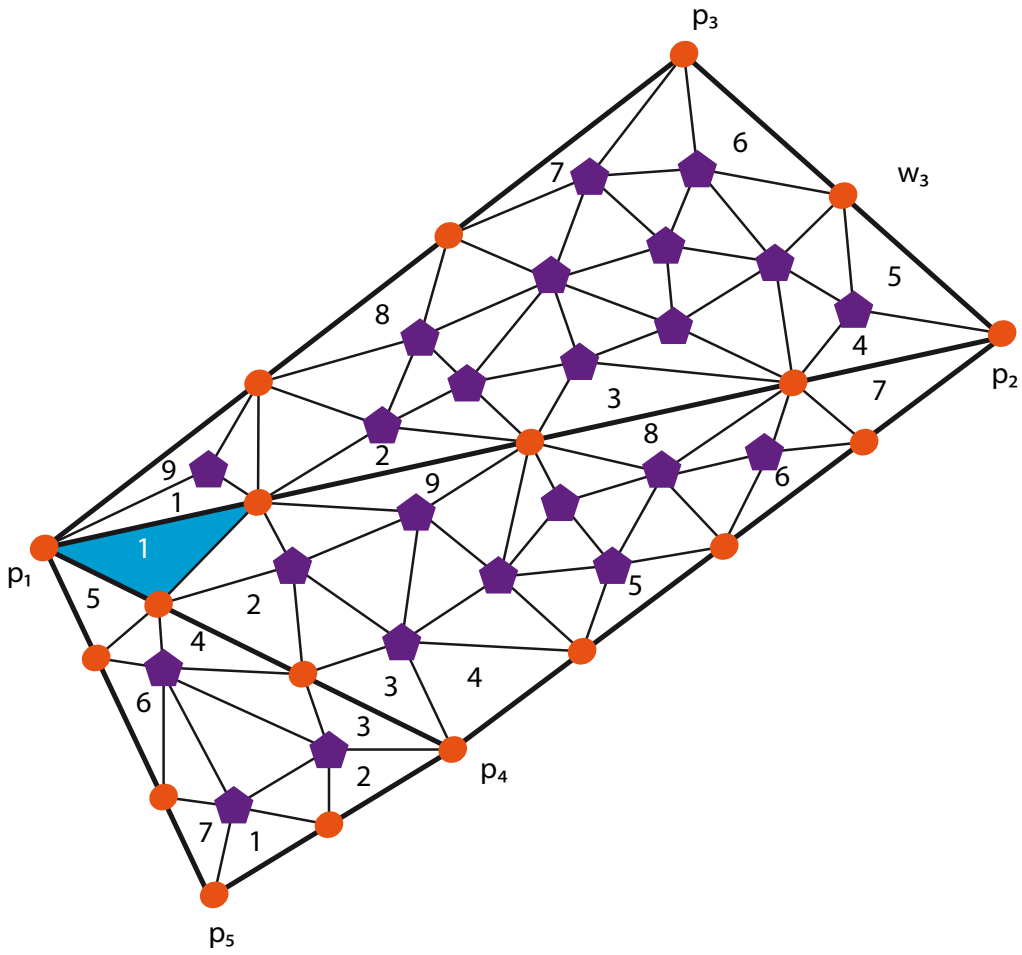


Figura 3.7: Tres regiones triangulares: $p_1\,p_2\,p_3$, $p_1\,p_4\,p_2$ y $p_1\,p_5\,p_4$. La subdivisión resaltada en la región triangular $p_1\,p_4\,p_2$ tiene subdivisiones adyacentes en dos regiones distintas.

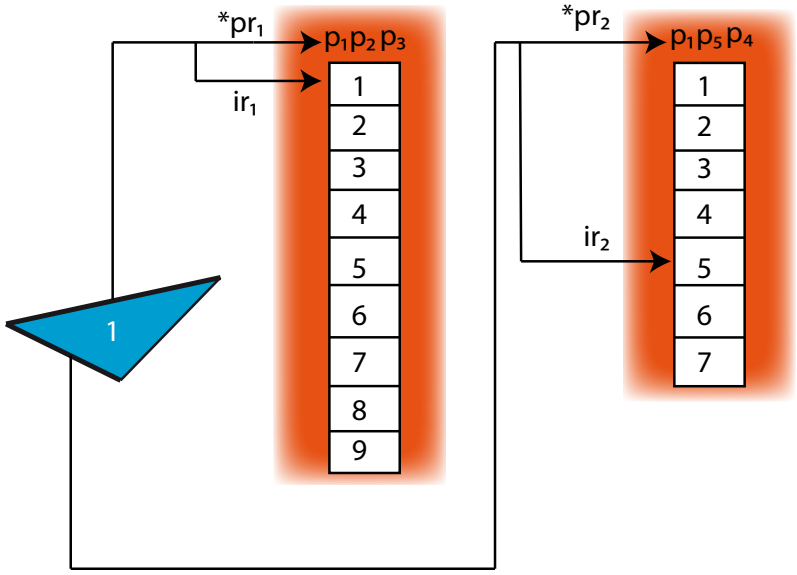


Figura 3.8: La subdivisión indicada con un triángulo es adyacente a dos subdivisiones en dos regiones distintas. Las regiones adyacentes son $p_1\, p_2\, p_3$ y $p_1\, p_5\, p_4$. Una región cuenta con nueve subdivisiones en los bordes y la otra siete. Para enlazarlas se utilizan dos punteros: $*pr_1$ y $*pr_2$; y dos índices: ir_1 e ir_2.

3.3 Planificación óptima

Existen básicamente tres etapas en el proceso que permite hallar el camino óptimo entre la ubicación actual y el destino. En primer lugar, es necesario construir las regiones para poder representar la información obtenida a través de los sensores mediante grillas de ocupación. El segundo paso consiste en hallar los caminos óptimos entre todas las posibles combinaciones de entradas y salidas para cada una de las regiones locales o sea, se realiza una planificación a nivel local. Finalmente, se halla el camino óptimo global entre la ubicación inicial y el destino.

3.3.1 Planificación local

En este caso, las grillas de ocupación se encuentran definidas mediante los puntos que se encuentran dentro de las regiones auxiliares y los bordes de las mismas; y la triangulación de Delaunay que se realiza en el interior de las regiones. Una vez que las regiones tienen definidas estas grillas y los grafos locales para cada una de ellas han sido construidos, se ejecuta el algoritmo de Dijkstra a nivel local, o sea, dentro de las regiones.

El algoritmo de Dijkstra requiere una función de costo para determinar el camino óptimo entre el origen y el destino. En este caso, las subdivisiones de la región que se está procesando y se encuentran marcadas como obstáculos no son insertadas en el montículo de Fibonacci ya que no pueden ser recorridas. Las demás subdivisiones son consideradas como posibles de ser recorridas y son agregadas al montículo cuando el algoritmo las alcanza a través de una transición posible. La función que determina el costo de la transición entre dos subdivisiones adyacentes se implementa combinando la información de las diferentes capas del mapa o simplemente utilizando la capa de obstáculos. Por ejemplo, si el objetivo es evitar los obstáculos y minimizar la distancia del viaje, el costo de alcanzar una subdivisión o nodo del grafo local consiste en el costo previo incrementado en una unidad. Otra función de costo podría incluir un término asociado con la pendiente o altura del terreno de la siguiente manera

$$\mathscr{C}_{z_1 z_2} = k_1 + k_2 \left|\Delta h\right|, \tag{3.3}$$

donde $\mathscr{C}_{z_1 z_2}$ denota el costo de desplazarse entre los nodos z_1 y z_2, k_1 indica el costo estrictamente asociado al desplazamiento entre z_1 y z_2 —en

definitiva, es el término destinado a disminuir la longitud del camino—, $\Delta h = h_{z_1} - h_{z_2}$ es el cambio en la altura entre las dos celdas adyacentes, $|\cdot|$ indica el valor absoluto, y k_2 es una constante que regula cuánto se pondera el cambio en la altura del terreno. Luego, los valores de las constantes k_1 y k_2 pueden elegirse adecuadamente de acuerdo a cuánto se quiera ponderar la longitud del camino recorrido y los cambios en la altura del terreno. Mayor complejidad puede darse a la Ec. 3.3 agregando términos que incluyan los efectos de otras capas del mapa, resultando en una planificación más sofisticada.

El algoritmo se ejecuta una única vez por cada una de las subdivisiones en los bordes de la región y se almacena el costo en llegar a cada una de las subdivisiones en los bordes. Como se muestra en la Fig. 3.9, si las subdivisiones se numeraran como se indica, se ejecutaría el algoritmo tantas veces como el número de subdivisiones en los bordes y para cada ejecución se almacenan los costos en una fila completa de la matriz que los almacena. Entonces, para el caso de la Fig. 3.9 la matriz de costos resulta

$$\mathbf{C} = \begin{pmatrix} c_{11} & c_{12} & \dots & c_{19} \\ c_{21} & c_{22} & \dots & c_{29} \\ \vdots & \vdots & \vdots & \vdots \\ c_{91} & c_{92} & \dots & c_{99} \end{pmatrix}, \tag{3.4}$$

donde las componentes c_{ij} representan el costo de desplazarse desde la subdivisión i-ésima a la j-ésima. Debe notarse que las componentes c_{ii} toman un valor nulo ya que, es este caso, el origen y el destino son coincidentes. Además, es interesante observar que en general, $c_{ij} \neq c_{ji}$ ya que el costo de desplazarse en una dirección no tiene por qué ser el mismo al de moverse

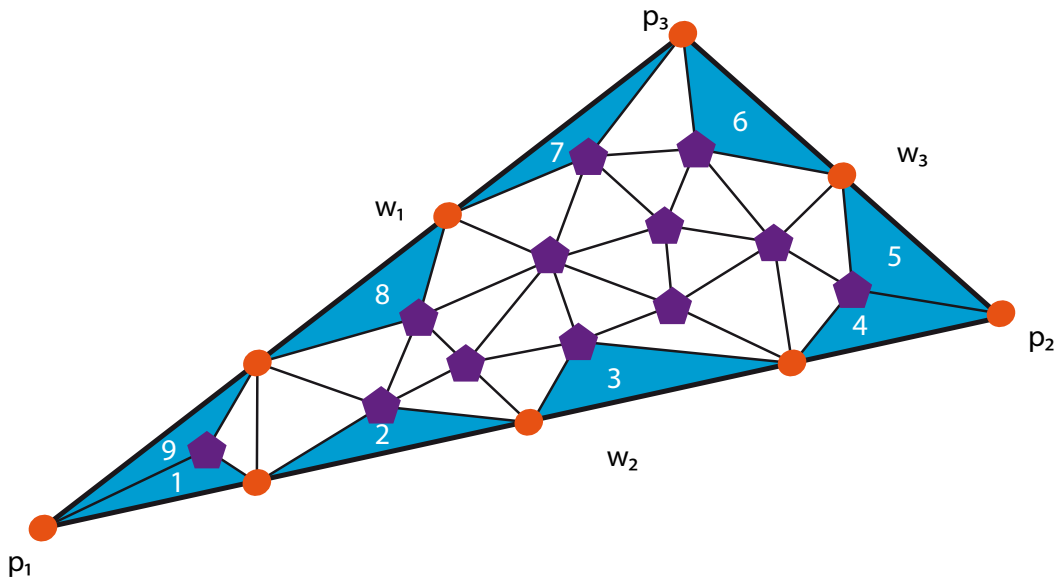


Figura 3.9: Las subdivisiones en los bordes de la región son hipotéticamente numeradas con el objeto de construir la matriz de costos de transición entre ellas.

en la dirección opuesta. Este caso podría darse si en la Ec. 3.3 se eliminaran las barras de valor absoluto. Con las barras de valor absoluto, las componentes del costo $\mathscr{C}_{z_1 z_2}$ se incrementan siempre que se produzca un cambio en la altura del terreno. Por el contrario, sin las barras se representaría el caso en el cual es más fácil para el vehículo desplazarse en una dirección descendente que ascendente y en este último caso, la matriz de la Ec. 3.4 resulta asimétrica.

Además de los costos de transición entre subdivisiones en los bordes, interesa almacenar los caminos recorridos. Para esto se dispone de una estructura tridimensional donde las primeras dos dimensiones se corresponden al origen y al destino en forma análoga a la matriz de la Ec. 3.4 y la tercera dimensión contiene las subdivisiones interiores de la región que se encuentran entre el punto de partida y el de llegada. Dos ejemplos de

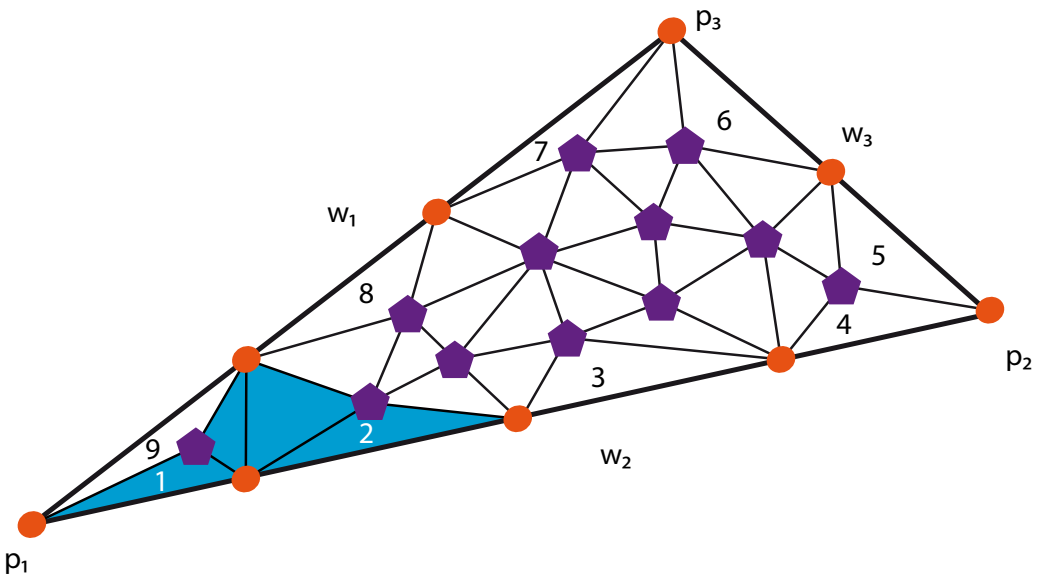


Figura 3.10: Ejemplo de camino óptimo entre las subdivisiones 1 y 2 de los bordes de la región.

caminos se observan en las Figs. 3.10 y 3.11. En la primera se muestra el camino óptimo entre las subdivisiones 1 y 2; mientras que en la segunda se observa el camino entre las subdivisiones 1 y 3.

Implementación computacional

Los detalles de la implementación computacional de las regiones y sus subdivisiones interiores pueden consultarse en el Apéndice B.

3.3.2 Planificación global

Si las regiones se encuentran definidas, los grafos locales están construidos y las regiones existentes se encuentran enlazadas; interesa hallar el camino óptimo global entre el origen y un destino. Para esto, se supone que existe una región definida donde se encuentra el vehículo y que existe una región en la zona de destino; que las matrices de costo fueron calcula-

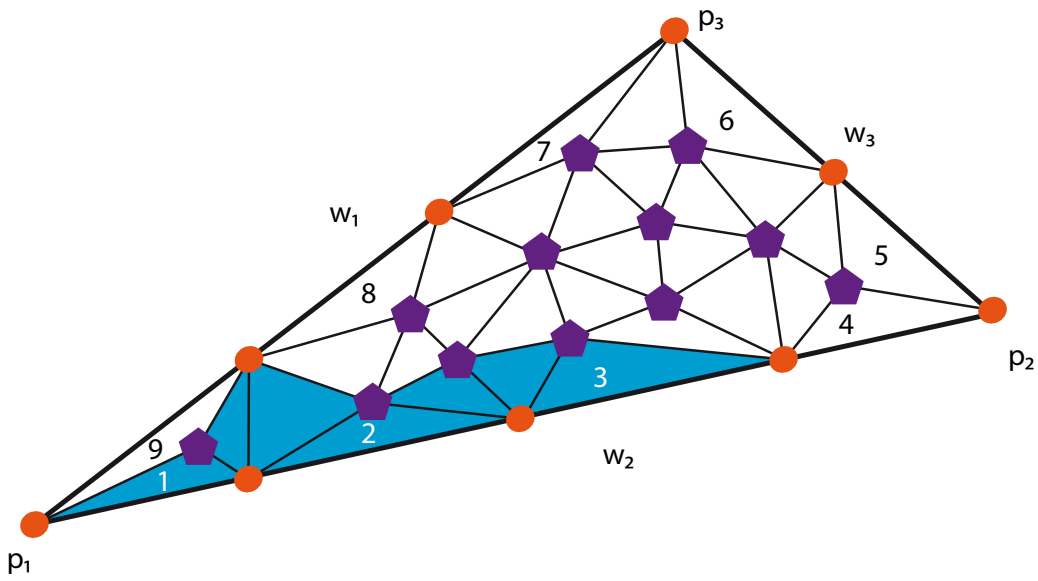


Figura 3.11: Ejemplo de camino óptimo entre las subdivisiones 1 y 3 de los bordes de la región.

das y que las regiones están apropiadamente enlazadas.

Puede calcularse el camino óptimo global de la siguiente forma: en este momento no interesa la representación de las regiones sino que las matrices de costo fueron calculadas para cada una de las regiones y almacenadas en una matriz como la presentada en la Ec. 3.4. Además, los enlaces entre las subdivisiones adyacentes entre regiones ya se encuentran implementados. En este momento, interesa obtener el camino óptimo entre la posición actual del vehículo y un destino y para esto, se recurre nuevamente al algoritmo de Dijkstra, aunque otros algoritmos de planificación óptima tales como A* o D* serían válidos también. La preferencia por este algoritmo se basa en el bajo costos computacional que posee si se implemente utilizando montículos de Fibonacci. El algoritmo elegido requiere de una función que determine el costo de transición entre dos nodos o celdas adyacentes. Para empezar, se asume que el vehículo se encuentra ubicado sobre una de

las subdivisiones que se hallan en los bordes de una región, por ejemplo la subdivisión 1 de la región $p_1\,p_4\,p_2$ en la Fig. 3.7. Los únicos dos tipos de transiciones posibles que se van a admitir son: un desplazamiento dentro del interior de la región entre una subdivisión de la frontera hacia otra del mismo tipo y una transición hacia otra subdivisión de la frontera de una región adyacente. En el caso del primer tipo y para la Fig. 3.7, los desplazamientos posibles serían hacia las subdivisiones $2, 3, \ldots, 8$ ó 9 de la misma región. En el segundo caso, las transiciones posibles son hacia la subdivisión 5 de la región $p_1\,p_5\,p_4$ ó hacia la subdivisión 1 de la región $p_1\,p_2\,p_3$. Estas transiciones posibles se muestran en la Fig. 3.12 en la cual, los nodos o subdivisiones en los bordes de la región de partida se encuentran en color naranja mientras que los nodos de las subdivisiones adyacentes se muestran en amarillo y celeste. Como se observa, hay ocho transiciones posibles hacia subdivisiones en el interior de la región y dos transiciones posibles hacia nodos en regiones adyacentes.

Los costos de transición entre nodos, necesarios para el algoritmo de Dijkstra, dependen de qué tipo de transición sea. Si si trata de una transición entre nodos internos, los costos se obtienen de la matriz de la Ec. 3.4 y para este caso, como el nodo de partida es el número 1, los costos corresponden a los indicados en la primer fila de la matriz. Así, si el algoritmo requiriese conocer el costo de la transición hacia la subdivisión 2, el costo correspondiente es el del elemento c_{12}. En cambio, si el algoritmo requiriese el costo de la transición hacia el nodo 5 de la región $p_1\,p_5\,p_4$, el mismo sería el costo de transición entre dos celdas adyacentes como descripto en la Ec. 3.3. De esta forma se ejecuta el algoritmo con estos costos hasta que alcance el destino deseado.

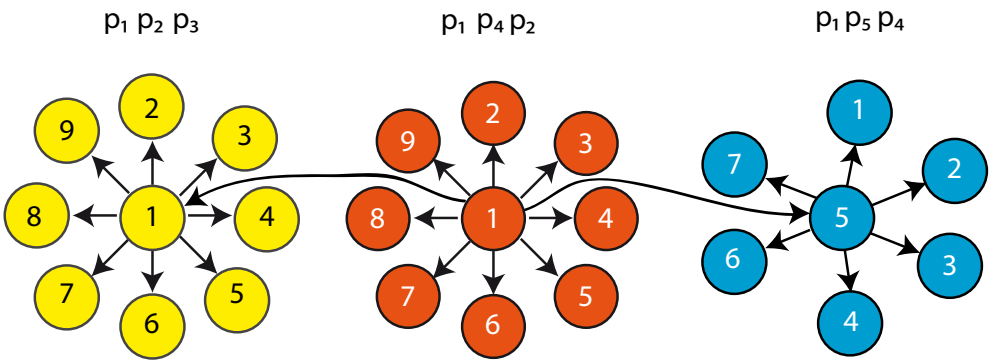


Figura 3.12: Las tres regiones se representan mediante los nodos de las subdivisiones en los bordes. La subdivisión 1 de la región del centro se encuentra enlazada con las ocho subdivisiones restantes de la misma región y además con las subdivisiones 1 y 5 de las regiones adyacentes. De la región que se muestra en el centro, sólo se indican todos los enlaces de la subdivisión 1 aunque existen más enlaces entre los demás nodos.

Debe notarse que para poder lograr almacenar el camino óptimo entre el origen y el destino, deben registrarse dos cosas ante cada transición. Si la misma se produjo entre dos regiones diferentes, no interesa registrar esta transición ya que las de este tipo no genera un tramo de camino. En cambio, una transición dentro del interior de la región produce un tramo de camino, por lo que interesa registrarla y almacenar por qué nodo se entró a la región y por cuál se salió. Para esto se emplea un método que registra mediante índices por dónde se ingresó y por dónde se abandonó la región.

El método consiste en dos índices. El primero, id, es un índice que indica por qué subdivisión del borde del mapa se ingresó a la región y el segundo, id_0, es un índice que indica por qué subdivisión del borde de la región previa se salió. Es importante enfatizar que los índices se registran de esta forma porque es lo que va a permitir más adelante reconstruir el camino óptimo ya que en definitiva, se está registrando como se llegó a la región. En la

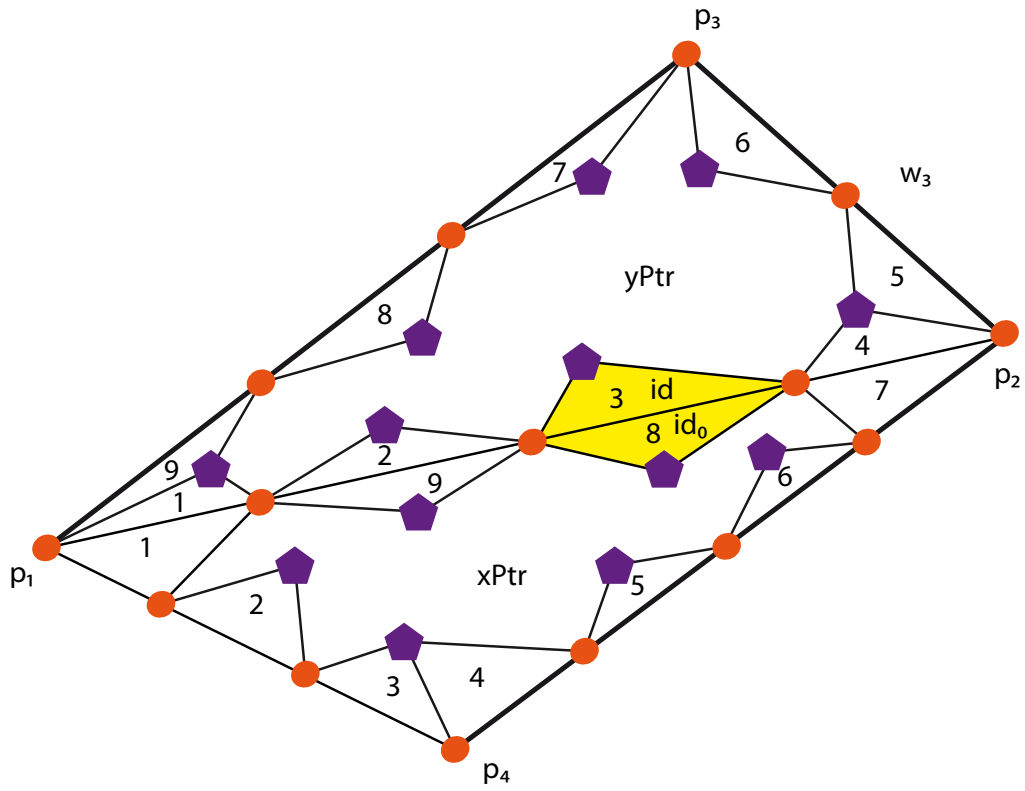


Figura 3.13: Dos regiones adyacentes apuntadas por los vectores $xPtr$ e $yPtr$. La región actual es la $yPtr$ y la previa $xPtr$. Los índices id e id_0 indican por qué subdivisiones se ingresó y se salió de la región previa. No se muestran las divisiones internas de las regiones a los fines de claridad.

Fig. 3.13 se observa un esquema que indica los índices para dos regiones adyacentes donde la región actual es la `yPtr` y la región previa es la `xPtr`.

Con respecto a la suposición de que el vehículo se encuentra en una subdivisión del borde de una región al comenzar, la misma puede salvarse agregando un tramo de trayectoria entre la posición inicial real del vehículo y la subdivisión que resulte como salida de la región en la cual se encuentra. Debe recalcarse que esta suposición es teórica y a los fines de comprender el funcionamiento del algoritmo y no es necesaria en la práctica ya que la etapa que se encarga de buscar la trayectoria óptima global sólo tiene en

cuenta regiones y registra por donde se ingresó a la región actual y por donde se salió de la región anterior. Jamás registra la posición inicial del vehículo en forma exacta como una subdivisión dentro de la región inicial, y de ahí deriva la necesidad de agregar un tramo de trayectoria entre la posición inicial del móvil —la subdivisión dentro de la región local inicial— y la subdivisión de salida de la región en la que se encuentra. En la Fig. 3.14 se ilustra este procedimiento y se observa como se incluye un tramo de trayectoria entre la posición inicial del robot γ, y la subdivisión por la cual se sale de la región inicial, id_0^1.

Además, es necesario agregar un tramo al final de la trayectoria entre la subdivisión de entrada a la última región y el punto de destino χ. Esto se debe a que el algoritmo de planificación global se detiene cuando alcanza la región dentro de la cual se encuentra el destino. Este tramo de trayectoria se agrega en forma análoga al tramo del comienzo y puede observarse en la Fig. 3.14 entre la subdivisión id^2 y χ.

3.4 Complejidad computacional

De acuerdo a lo visto, la triangulación de un número de mojones n_m tiene un coste $O(n_m \log n_m)$ y ocupa un espacio $O(n_m)$. Esto se debe a que se utiliza la triangulación de Delaunay.

Luego, dentro de cada una de las regiones existe una serie de puntos que se utilizan en conjunción con otros que se agregan en las fronteras de las regiones, con el fin de triangular dentro de ellas. Si se denomina n_i a la cantidad total de puntos en la región i-ésima, el coste de triangular dentro de esta región está dado por $O(n_i \log n_i)$.

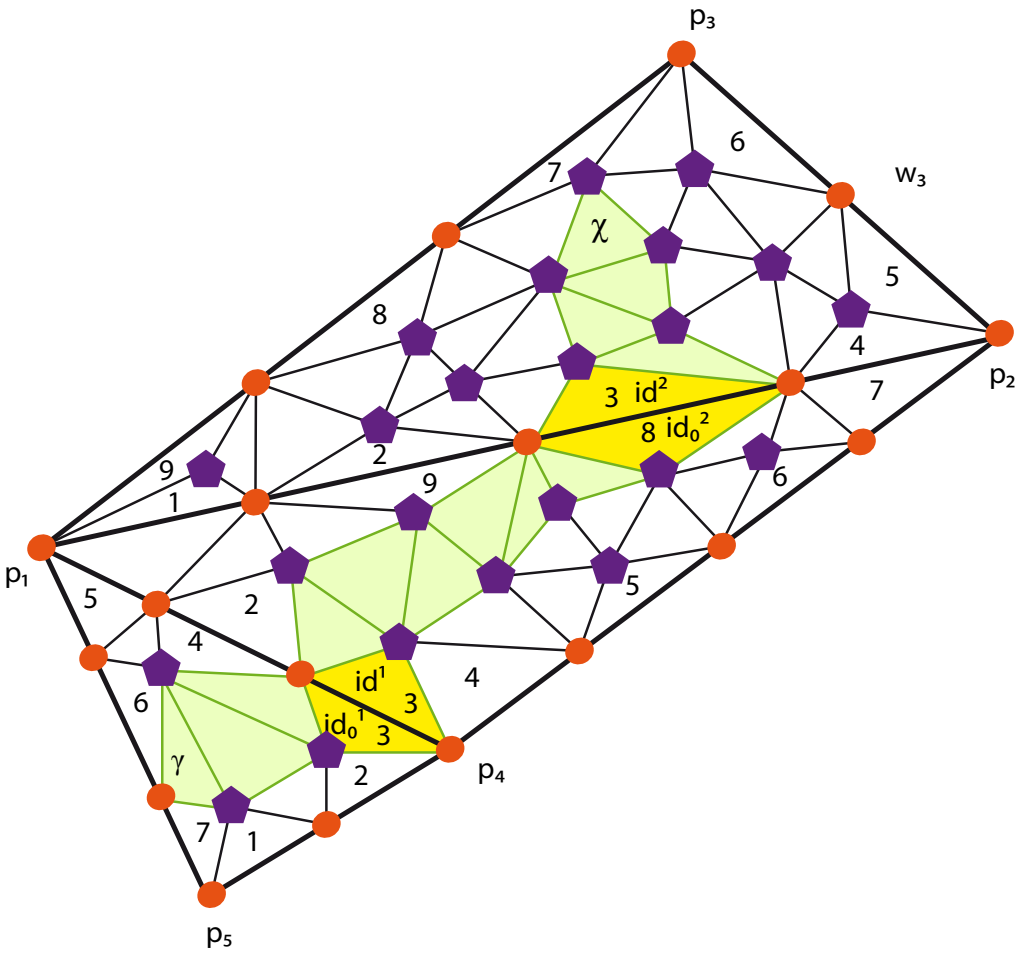


Figura 3.14: El símbolo γ indica la posición inicial del vehículo y la letra χ representa el destino deseado. Los índices id_0^1, id^1, id_0^2 e id^2 indican las subdivisiones de salida, entrada, salida y entrada respectivamente. El camino inicial es agregado entre γ e id_0^1 mientras que el camino final agregado se encuentra entre id^2 y χ. En este caso, el algoritmo de planificación hallaría la trayectoria entre id^1 e id_0^2.

Cada región resulta con un determinado número de accesos n_i^a y se ejecuta una vez el algoritmo de Dijkstra por cada subdivisión en los bordes de la región; con el objetivo de hallar un camino óptimo hacia cada una de las salidas de la región, que son idénticas a las entradas. Por lo tanto, el coste de calcular estos trechos está dado por

$$\begin{aligned} D_i &= O(n_i^a[V_i \log V_i + E_i]), \\ V_i &= 2n_i - 2 - k_i, \\ E_i &\leq 3(V_i - n_i^a) + 2n_i^a, \end{aligned} \tag{3.5}$$

donde D_i representa el coste total, V_i indica el número de vértices de la región i-ésima que equivale al número de subdivisiones, E_i el número de arcos de la misma región y k_i denota el número de puntos en la frontera de la región —que depende de la longitud de los lados de la región y de la densidad d de puntos elegida—. Además, cada subdivisión del interior de una región está conectada con las adyacentes de la forma que muestra la Fig. 3.15, donde se observa que las subdivisiones de los bordes cuentan con exactamente dos punteros mientras que las restantes —o sea, las subdivisiones en el interior de la región— tienen tres punteros. De la observación anterior se deduce la última de las Ecs. 3.5.

Las regiones que resultan de triangular los mojones y sus respectivas subdivisiones en los bordes deben ser enlazadas a fin de permitir la planificación. El coste de enlazar o unir dos regiones es cuadrático en el número de subdivisiones que se encuentran en los bordes, o sea $O(n_i^a\, n_j^a)$ donde n_i^a y n_j^a indican los números de accesos o de subdivisiones en los bordes de las

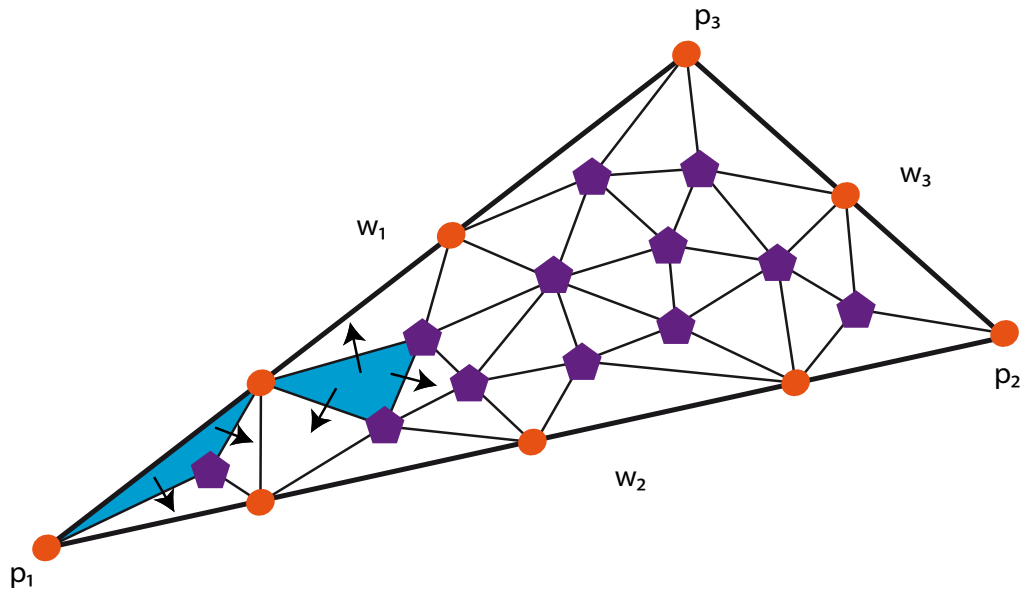


Figura 3.15: La subdivisión resaltada que se encuentra en la esquina de la región tiene dos punteros hacia subdivisiones adyacentes mientras que una subdivisión del centro cuenta con tres punteros.

regiones i-ésima y j-ésima, respectivamente. Esto se debe a que las subdivisiones en los bordes comparten puntos en común pero no se encuentran unidas mediante punteros. Este costo podría llegar a reducirse bajo otro tipo de implementación ya que, en definitiva, se trata de una búsqueda y pueden aplicarse todos los algoritmos existentes para este fin.

Como el algoritmo de Dijkstra se utiliza para hallar la secuencia óptima de regiones, el coste estaría dado por

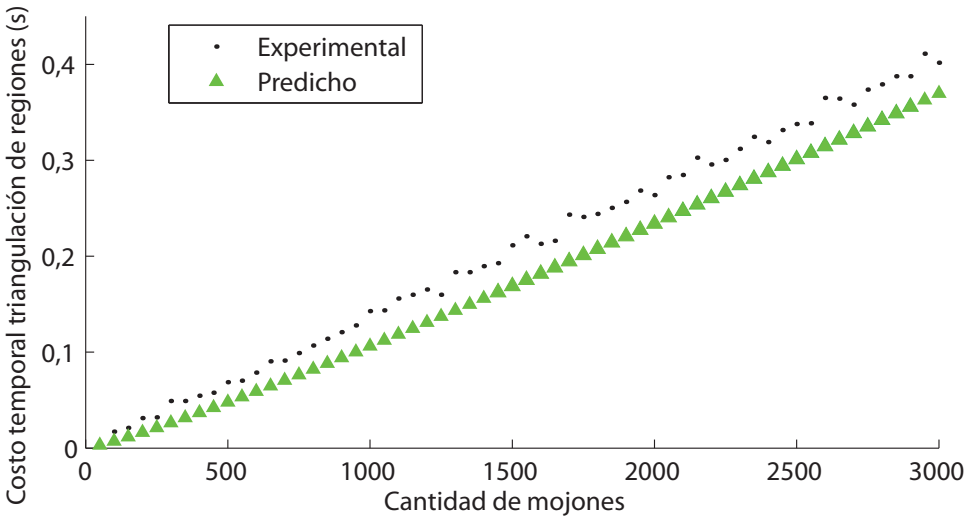
$$\begin{aligned} D_r &= O(V_r \log V_r + E_r), \\ V_r &= 2\,n_m - 2 - k_m, \\ E_r &= \sum_{i=1}^{V_r} n_i^a, \end{aligned} \tag{3.6}$$

donde D_r es el costo de hallar esta secuencia. Esto se debe a que en esta etapa la búsqueda se realiza sobre las regiones, que representan los nodos, y las transiciones posibles están dadas por el número de subdivisiones en los bordes ya que por cada subdivisión existe un único enlace hacia otra subdivisión.

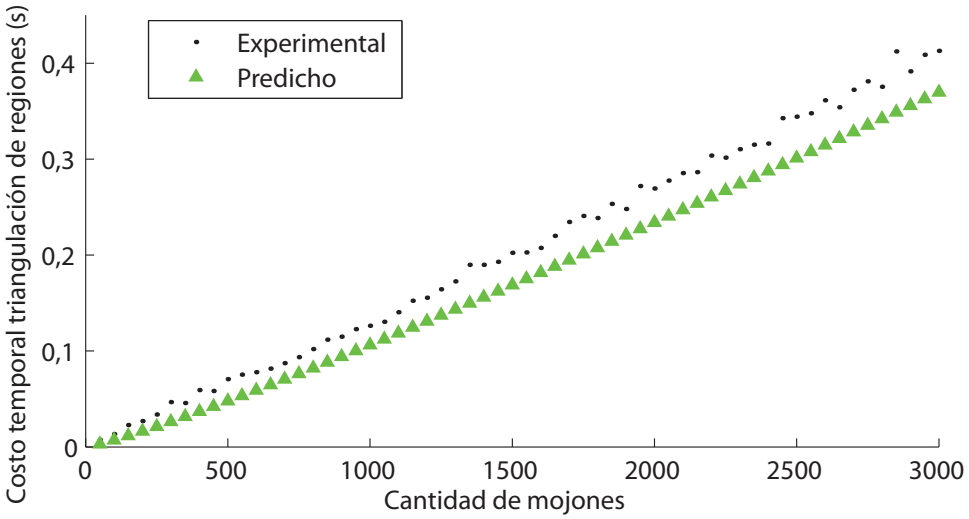
3.5 Mediciones de complejidad computacional

A los fines de verificar los cálculos de orden computacional ($O[\cdot]$), se calculan en forma teórica y experimental los órdenes de los tiempos de cálculo. Las Figs. 3.16(a) y 3.16(b) muestran el costo computacional en segundos de generar las regiones a medida que se incrementan los mojones. Los puntos experimentales fueron obtenidos midiendo los tiempos promedio (promediando veinte y treinta ejecuciones para cada punto respectivamente) que tardaba la rutina encargada de generar la triangulación de los mojones. Los tiempos medidos corresponden al tiempo que tarda la rutina y fueron cuidadosamente medidos de forma tal que no influyeran los tiempos de ejecución de otros procesos que puedan llegar a estar siendo ejecutados por el sistema operativo (ver en [25] la definición de procesos e hilos). Además, a modo de comparación, se incluye la tendencia dada por la predicción efectuada por cálculo. Esta referencia es obtenida en base al cálculo del orden y escalada de forma que muestre si la tendencia seguida por los datos experimentales es la correcta y no pretende ser un ajuste de los datos experimentales.

Luego, se midió el tiempo de ejecución de la rutina encargada de generar



(a) Veinte puntos promediados.



(b) Treinta puntos promediados.

Figura 3.16: Costo temporal en triangular las regiones. La triangulación se genera a partir de los mojones. Cada punto de la curva es un promedio de varias ejecuciones. El eje vertical se encuentra en segundos y el horizontal representa la cantidad de puntos.

la triangulación en el interior de las regiones. Primero se intentó proceder de forma análoga al método anterior pero en este caso, como los gránulos de tiempos del microprocesador empleado son del orden de 16 ms y debido a que el tiempo de ejecución de la rutina resultó ser inferior a estos; se introdujo una pausa de 50 ms cada vez que se ejecutaba uno de los pasos recurrentes de la rutina. Debido a esta modificación, los valores absolutos de los ejes verticales de las Figs. 3.17 a 3.21 no son relevantes en sí mismos pero sí son interesantes las tendencias de las curvas. Se considera que una pausa de 50 ms es suficientemente larga aún si el sistema operativo se encuentra ejecutando otro proceso. De esta forma, pueden medirse los pulsos del microprocesador cuando comienza a ejecutar la rutina, cuando termina de ejecutarla y efectuar la diferencia entre estas dos cantidades y de esta forma estar infiriendo de forma indirecta el tiempo de ejecución sin ser afectados por el sistema operativo. Debe destacarse que esto es una aproximación práctica a la medición. En la Fig. 3.17 se observa que los puntos experimentales siguen la misma tendencia que la predicción teórica ya que el costo se incrementa en forma $n_i \; log \, n_i$ con el número de puntos en el interior de la región.

A continuación, la Fig. 3.18 muestra el costo de unir cada una de las regiones en el eje vertical mientras que en el eje horizontal se indica el número de accesos a las región n_i^a. Se aprecia que para algunos valores de n_i^a existe más de un punto y esto se debe a que en estos casos, existe más de una región con el mismo número de accesos. Los costos para estos casos resultan similares pero no exactamente iguales —y esto puede apreciarse leyendo la figura para un mismo valor de n_i^a—.

Las Figs. 3.19 y 3.20 corresponden a predicciones y resultados experi-

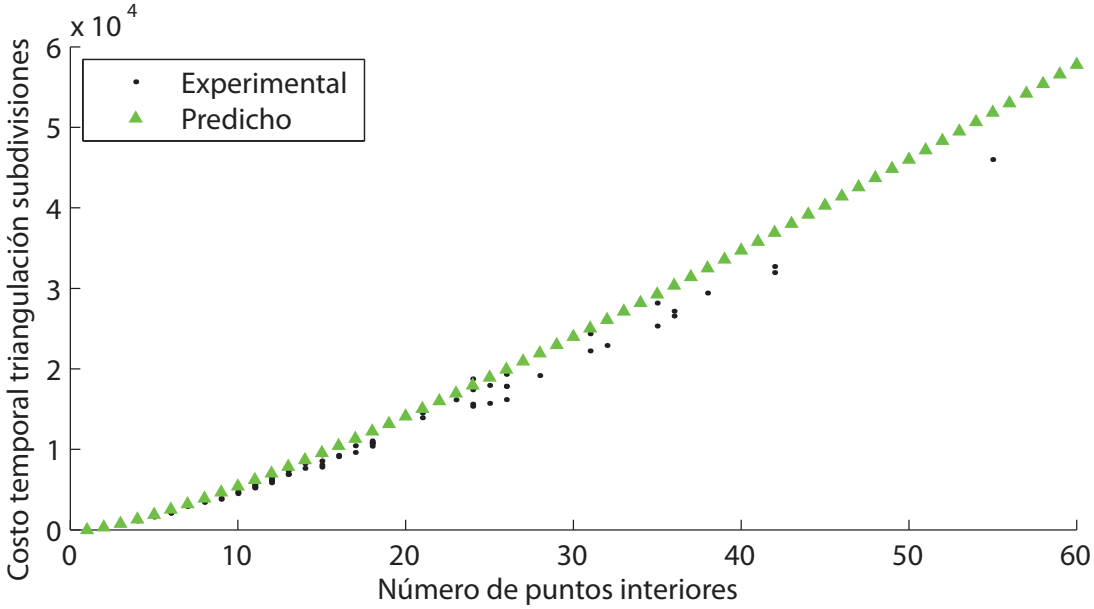


Figura 3.17: Costo temporal predicho y experimental de triangular las subdivisiones interiores. Los valores absolutos del eje vertical no son representativos pero sí lo es la tendencia de los puntos.

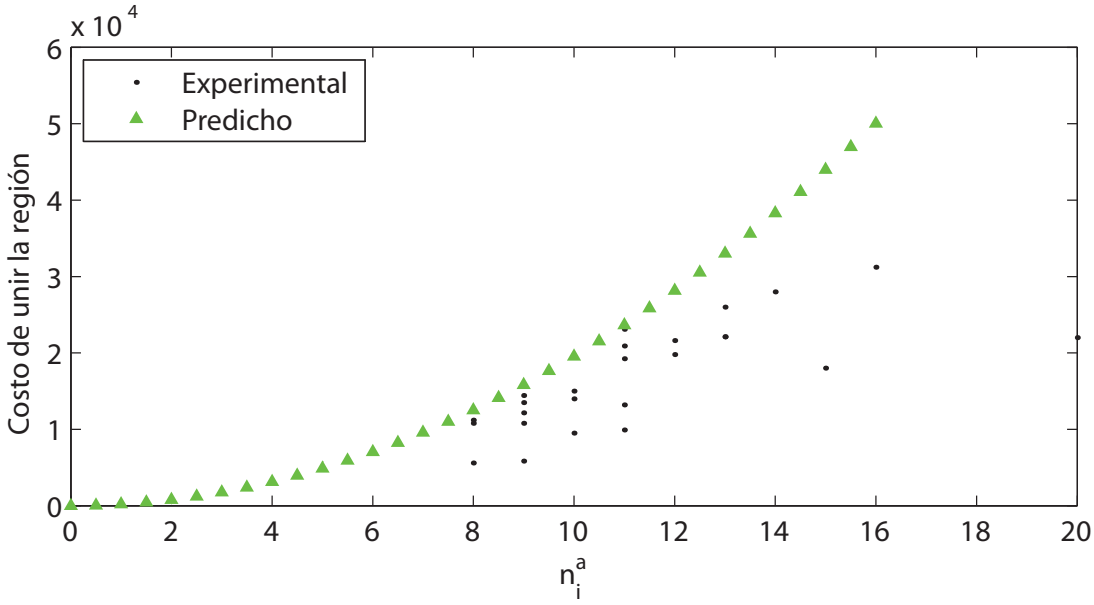


Figura 3.18: Costo de unir regiones de acuerdo al número de accesos. Resultados de predicciones y experimentos. Los valores absolutos del eje vertical no son representativos pero sí lo es la tendencia de los puntos.

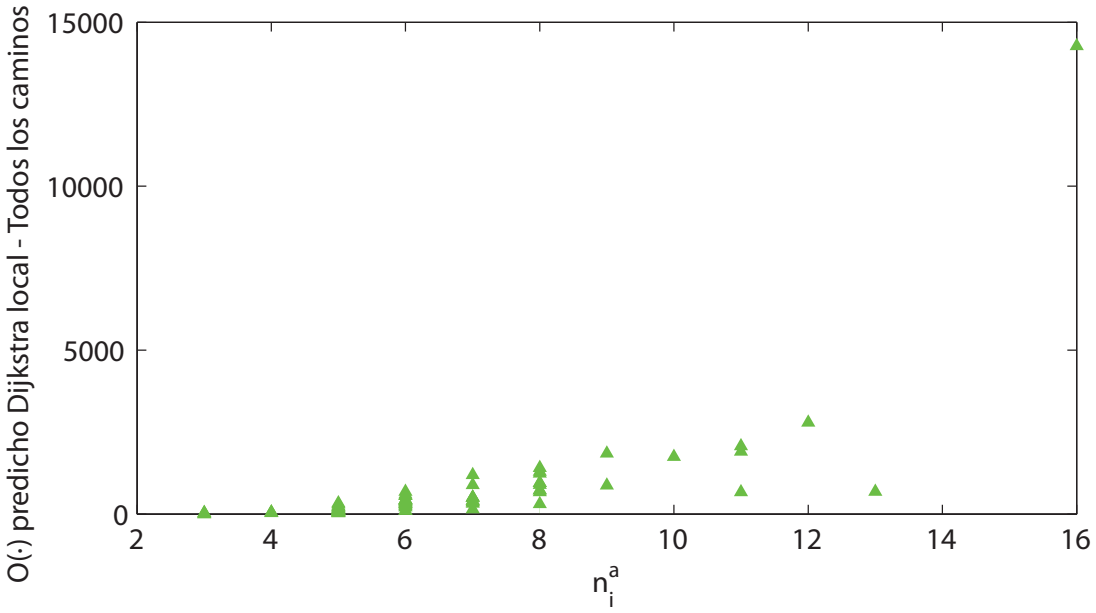


Figura 3.19: Costo temporal predicho de la planificación local de todos los caminos dependiendo del número de accesos. Los valores absolutos del eje vertical no son representativos pero sí lo es la tendencia de los puntos.

mentales para el caso de la obtención de todos los caminos en el interior de las regiones. La primera figura corresponde a predicciones realizadas mediante las Ecs. 3.5 y conociendo el número de accesos y de puntos dentro de las regiones. La segunda figura fue obtenida en forma experimental midiendo los tiempos para cada una de las regiones introduciendo las pausas de 50 ms en cada rutina que involucrase una recurrencia. Como ejes horizontales fueron elegidos los números de accesos para observar el incremento lineal en el costo con este parámetro.

Finalmente, se llevó a cabo una predicción en calcular la trayectoria óptima global y se midió en forma experimental este tiempo como se aprecia en la Fig. 3.21. Para lo primero se utilizó la Ec. 3.6 y para lo segundo se

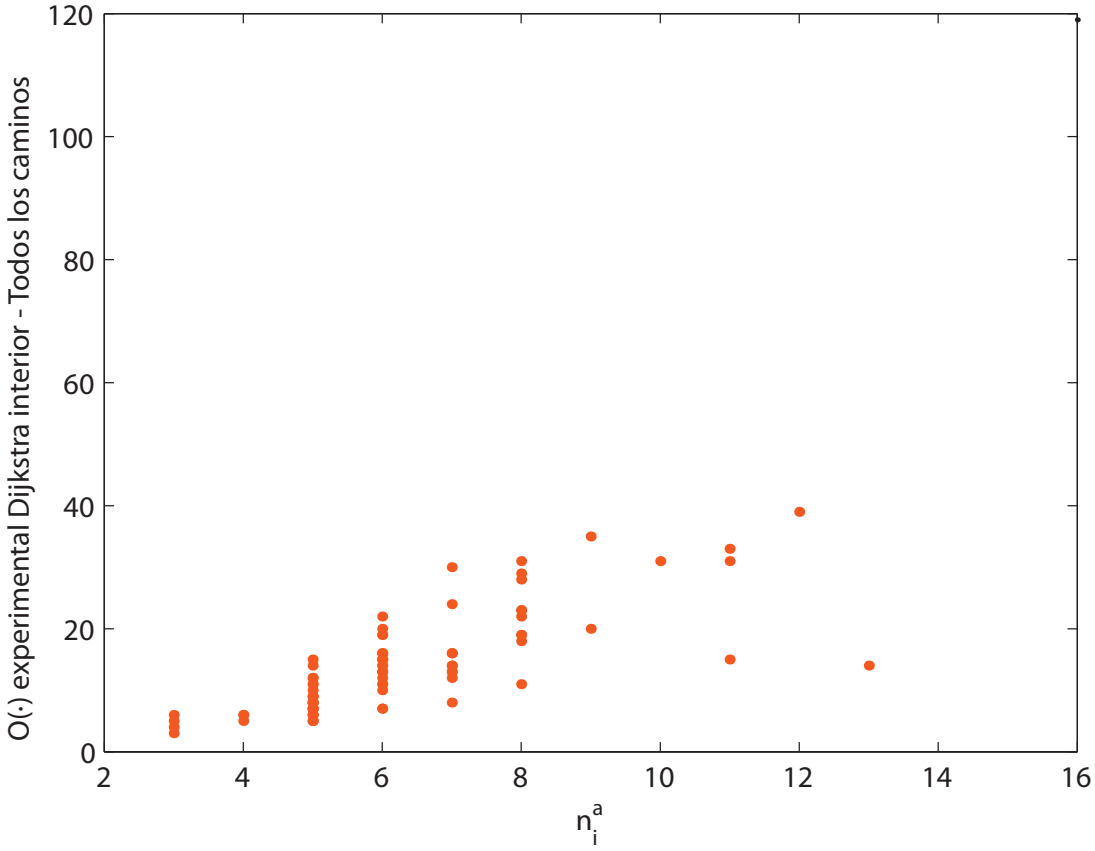


Figura 3.20: Costo temporal experimental de la planificación local de todos los caminos dependiendo del número de accesos. Los valores absolutos del eje vertical no son representativos pero sí lo es la tendencia de los puntos.

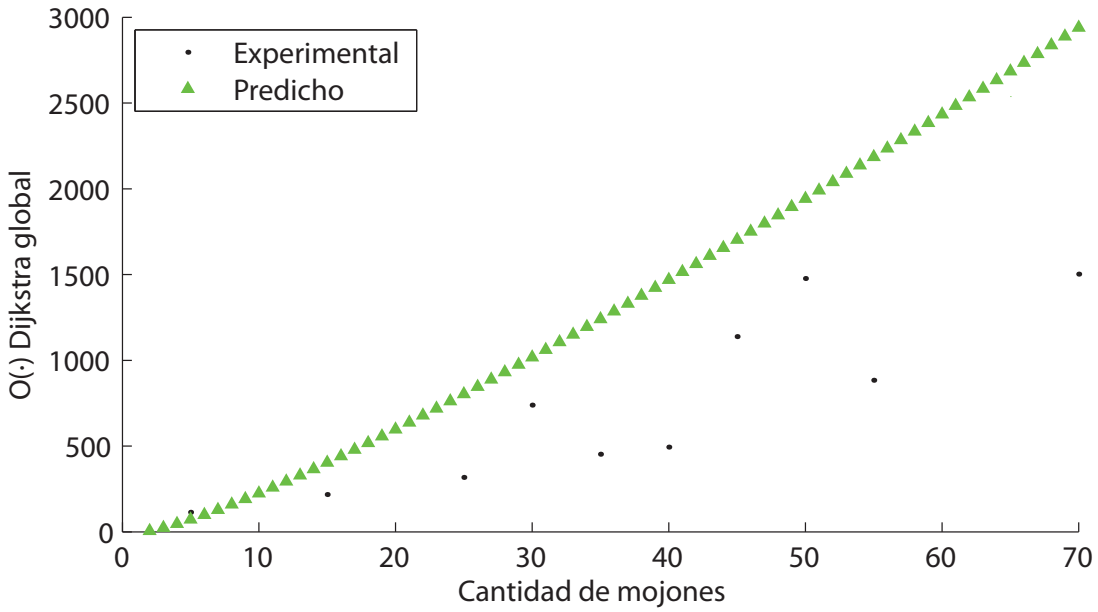


Figura 3.21: Costo temporal predicho y experimental en obtener la trayectoria óptima global mediante el algoritmo de Dijkstra. Los puntos experimentales fueron obtenidos realizando un promedio de tres mediciones. Los valores absolutos del eje vertical no son representativos pero sí lo es la tendencia de los puntos.

midieron los tiempos de ejecución de la rutina, introduciendo pausas en caso de recurrencia. Los puntos de la curva experimental fueron obtenidos realizando un promedio de tres mediciones para cada uno.

3.6 Conclusiones del capítulo

A partir de la triangulación de Delaunay, se definió una forma de dividir el mapa en regiones y en subdivisiones internas. Luego, las regiones y las subdivisiones en los bordes fueron enlazadas con las subdivisiones en regiones vecinas. A continuación, fueron definidos dos niveles de planifica-

ción: *local* y *global*. El primero se refiere a planificar en el interior de las regiones y el segundo apunta a la planificación entre regiones. Finalmente, la complejidad computacional de todas las operaciones fue estimada y medida mediante dos estrategias distintas.

Resta analizar cómo generar una trayectoria única a partir del resultado obtenido mediante la planificación global y la estrategia de control a emplear. Además, el vehículo en sí debe ser analizado para tener en cuenta sus limitaciones.

Capítulo 4

Generación de trayectorias y control

En el Capítulo 3 se presentaron todos los elementos necesarios que deben implementarse para poder planificar, tanto dentro de las entidades definidas como regiones como también a nivel global, permitiendo obtener una planificación entre el punto donde se ubica el vehículo y el destino. Sin embargo, resta analizar cuestiones tales como la forma en que se reconstruye la trayectoria planificada; la complejidad computacional asociada a la representación en secciones, subdivisiones y la planificación; el modelo del vehículo en sí; la estrategia de control a emplear; las limitaciones del vehículo y la forma en que el mapa y la representación del ambiente pueden expandirse más allá de los límites que tengan en ese momento.

Hasta este punto, cosas tales como el vehículo, sus restricciones, la estrategia de control y cuestiones sobre el mapa no han sido deliberadamente tenidas en cuenta en el módulo de planificación; ya que de esta forma el módulo de planificación puede ser utilizado en diferentes tipos de vehícu-

los, con variadas técnicas de control y para diferentes técnicas de mapeo —siempre y cuando los mapas admitan tener mojones—. Esta forma de trabajo sigue la tendencia actual del campo de la ingeniería de la robótica [9, 10]. En este capítulo se aplicará el módulo de planificación a un vehículo en particular y se empleará una técnica de control a los fines de mostrar su aplicabilidad.

Primero, se presentará un método que permite fácilmente recuperar y unir los tramos de la trayectoria global a partir de los índices almacenados en la etapa de planificación global.

A continuación, se presentará un modelo cinemático de un vehículo terrestre y se discutirán sus limitaciones.

Posteriormente, se expondrá una estrategia de control viable y la forma en que se integra con la trayectoria global planificada y finalmente, se bosquejará una estrategia que permite que el mapa se expanda más allá de sus límites, pero alterando lo menos posible la representación lograda hasta ese momento.

4.1 Reconstrucción de trayectorias

El resultado de ejecutar la planificación global consiste en un vector de punteros a regiones. Cada región contiene dos índices: id e id_0, donde el primero indica por qué subdivisión se ingresó a la región y el segundo indica por qué subdivisión del borde de la región previa se salió. Para el caso de la Fig. 3.14, el resultado de la planificación global puede verse en la Tabla 4.1 y consiste en dos punteros a dos subdivisiones y a dos pares de índices. La tabla debe leerse de abajo hacia arriba si se quieren seguir las

Tabla 4.1: Resultado de ejecutar la planificación global para el caso de la Fig. 3.14.

Región	id	id_0
$p_1\, p_2\, p_3$	3	8
$p_1\, p_4\, p_2$	3	3

regiones desde el origen hacia el destino.

Debe recordarse que es necesario agregar dos tramos de trayectoria: uno al principio y otro al final. El tramo del comienzo se extiende entre la posición inicial γ y el índice que muestra por qué subdivisión se salió de la primer región —o sea, la región donde se encuentra el vehículo—; y el tramo de terminación prolonga el trayecto entre la subdivisión de entrada a la región de destino y la posición final χ. A los fines prácticos, se incluye la Tabla 4.2 que muestra el resultado de la planificación global y se añaden en la parte inferior y superior las prolongaciones del principio y del final. Los índices se encuentran coloreados para indicar la forma en que deben leerse los tramos de la trayectoria. Si se lee la tabla de abajo hacia arriba y siguiendo los colores, pueden reconstruirse los tramos de trayectoria de la Fig. 3.14. Debe recordarse que los tramos de trayectoria entre todas las combinaciones de entrada y salida a una región local a través de las subdivisiones de los bordes son almacenados en una estructura tridimensional. De forma que, teniendo los índices y la estructura, sólo resta leer de la última los tramos e incorporar los trechos del comienzo y del final.

Tabla 4.2: Resultado de ejecutar la planificación global para el caso de la Fig. 3.14. Versión amplificada de la tabla a los fines de incluir los tramos iniciales y finales. Se resaltan tres tramos de trayectoria: γ–3, 3–8 y 3–χ.

Región	id	id_0
—	—	χ
$p_1\ p_2\ p_3$	3	8
$p_1\ p_4\ p_2$	3	3
—	γ	—

4.2 Modelo cinemático y limitaciones

Si bien existen casos en los cuales los modelos dinámicos son esenciales [6, 7, 20, 35]; en general, los modelos cinemáticos de vehículos son preferibles a los dinámicos por una serie de motivos [43]:

– Los modelos cinemáticos son más simples que los dinámicos. No se requiere hallar valores precisos dentro de ecuaciones matriciales.

– Los *actuadores* eléctricos que consisten en motores pueden regular la velocidad y permiten interpretarla como una variable de control.

A los fines de utilizar el esquema de planificación presentado en esta tesis sobre un vehículo terrestre de cuatro ruedas, se presenta un modelo cinemático para robots tipo auto, difundido y aceptado en la literatura [43, 21, 23, 16, 37] y que consiste en

$$\begin{pmatrix} \dot{x_d} \\ \dot{y_d} \\ \dot{\theta_d} \\ \dot{\phi_d} \end{pmatrix} = \begin{pmatrix} \cos\theta_d\, v_{d1} \\ \sin\theta_d\, v_{d1} \\ v_{d1}/l_m\ \tan\phi_d \\ v_{d2} \end{pmatrix}, \tag{4.1}$$

donde x_d e y_d representan la posición en el plano del punto medio de las ruedas traseras del vehículo, θ_d es la orientación del chasis del vehículo, como se aprecia en la Fig. 4.1; v_{d1} es la intensidad de la velocidad longitudinal, v_{d2} es la intensidad de la velocidad angular instantánea del chasis y ϕ_d es el ángulo de la dirección del vehículo y l_m es la distancia entre ejes. Estas dos últimas variables y la anchura a_m pueden observarse en la Fig. 4.2 para un vehículo todo terreno de marca «Yamaha», modelo «Grizzly 125».

Debido a la motivación en integrar el método de planificación con una estrategia de control [36], interesa analizar las limitaciones del vehículo. Algo que se observa en las Ecs. 4.1 es que el ángulo de giro de la dirección ϕ_d no se encuentra limitado en el modelo. Por esto, interesa deducir qué rangos de valores puede tomar el mismo.

A partir de las especificaciones técnicas del vehículo [54], se observa que la distancia entre ejes toma un valor de $l_m = 1,080$ m, la anchura total es de $a_m = 0,990$ m y el radio mínimo de giro es $r_m = 2,9$ m.

Por radio mínimo de giro se entiende que es el radio del círculo descripto por la rueda que se encuentra más alejada del centro de giro [53]. En el caso de la Fig. 4.3, el vehículo se encuentra girando hacia la derecha y entonces el radio mínimo de giro está definido por la rueda de la izquierda.

A fin de poder calcular el ángulo máximo, se utiliza el concepto de radio mínimo de giro y las Ecs. 4.1. Para este caso, se asume que el ángulo de la dirección se mantiene en su máximo valor, o sea $\phi_d = \overline{\phi_d}$; la velocidad angular es nula por lo que $v_{d2} = 0$; y las ecuaciones de movimiento se reducen a

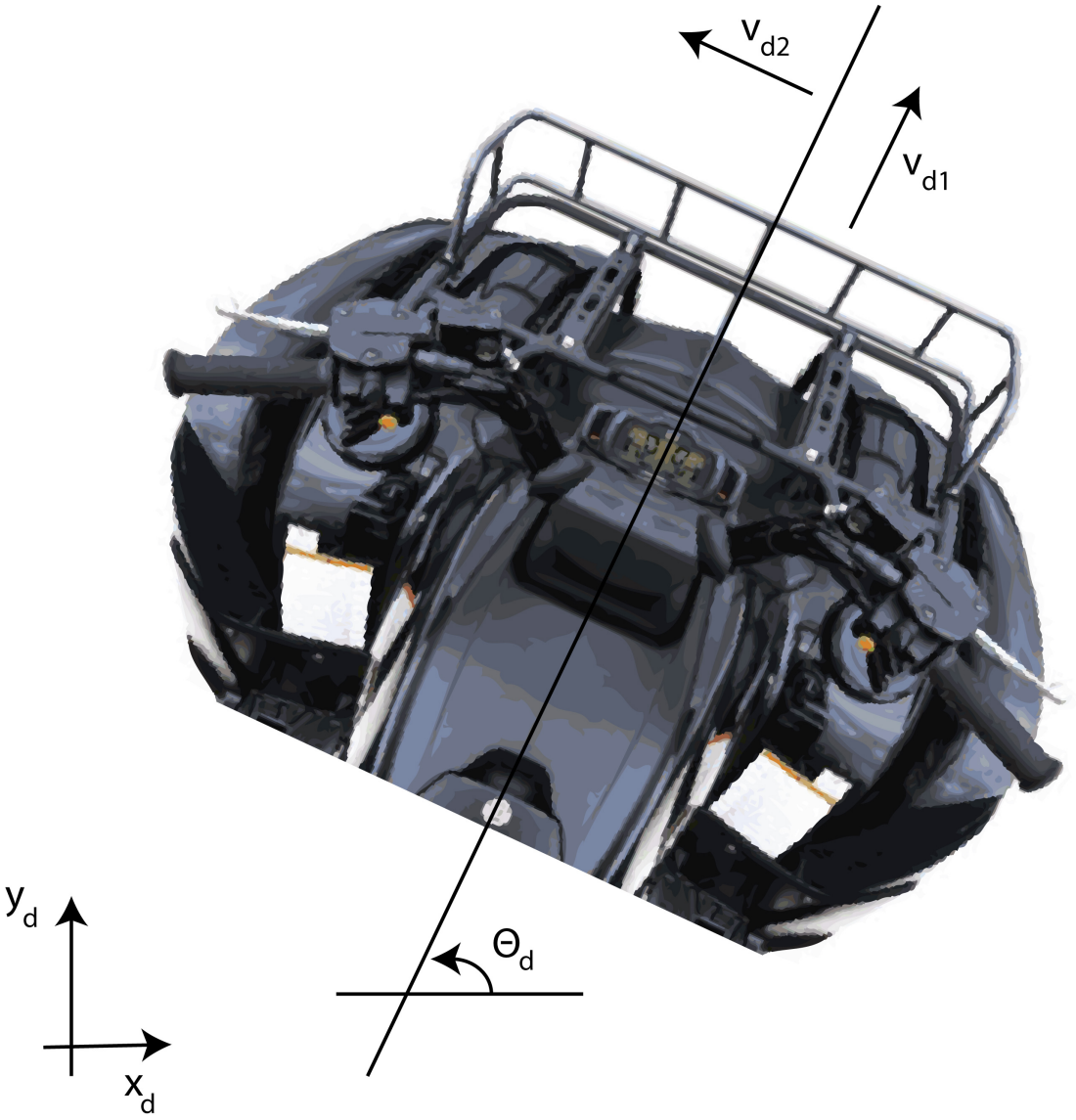


Figura 4.1: Coordenadas x_d e y_d, el ángulo del chasis θ_d, las velocidades lineal v_{d1} y angular v_{d2} del chasis para un vehículo de cuatro ruedas. Vista superior de la parte delantera del móvil.

Figura 4.2: Ángulo de la dirección ϕ_d, distancia entre ejes l_m y a_m es la anchura para un vehículo de cuatro ruedas.

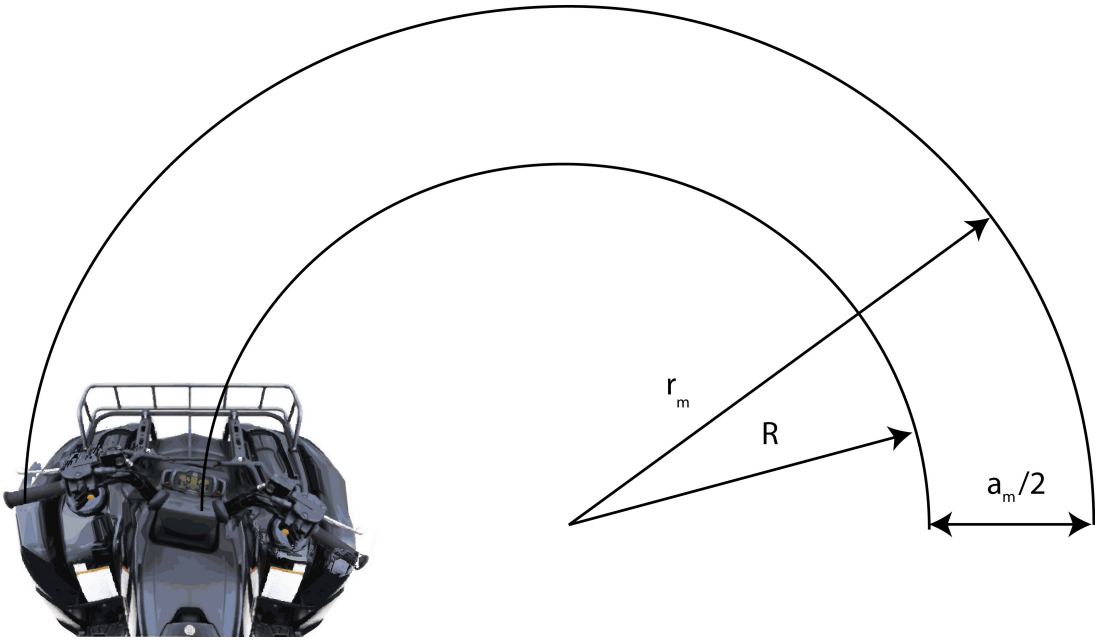


Figura 4.3: Las ruedas de la izquierda del vehículo se encuentran describiendo el radio mínimo de giro r_m y el centro del mismo describe un círculo de radio inferior, $R = r_m - a_m/2$.

$$\begin{pmatrix} \dot{x_d} \\ \dot{y_d} \\ \dot{\theta_d} \\ \dot{\phi_d} \end{pmatrix} = \begin{pmatrix} \cos\theta_d \, v_{d1} \\ \sin\theta_d \, v_{d1} \\ v_{d1}/l_m \, \tan\overline{\phi_d} \\ 0 \end{pmatrix}, \tag{4.2}$$

y el vehículo se encuentra describiendo una trayectoria circular de radio R, como muestra la Fig. 4.3, y de valor $R = r_m - a_m/2$. Se propone la siguiente solución para las dos primeras ecuaciones

$$\begin{pmatrix} x_d \\ y_d \end{pmatrix} = \begin{pmatrix} R\sin\theta_d \\ R\cos\theta_d \end{pmatrix}, \tag{4.3}$$

y luego diferenciando las Ecs. 4.3 y reemplazando el valor de $\dot{\theta}_d$ de las Ecs. 4.2, se obtienen

$$\begin{pmatrix} x_d \\ y_d \end{pmatrix} = \begin{pmatrix} \left(\frac{R\tan\overline{\phi_d}}{l_m}\right) v_{d1}\cos\theta_d \\ \left(\frac{R\tan\overline{\phi_d}}{l_m}\right) v_{d1}\sin\theta_d \end{pmatrix}, \tag{4.4}$$

y para que se respeten las Ecs. 4.1, el témino entre paréntesis debe ser igual a la unidad o

$$R = \frac{l_m}{\tan\overline{\phi_d}}, \tag{4.5}$$

y entonces

$$\overline{\phi_d} = \arctan\left(\frac{l_m}{R}\right) = \arctan\left(\frac{1,080}{2,9-0,990/2}\right) = 24°11', \tag{4.6}$$

por lo que el ángulo máximo de la dirección del vehículo no puede superar esta cantidad.

4.3 Estrategia de control

Debido a que mediante la planificación propuesta se obtiene una «tira de triángulos», los cuales representan la trayectoria óptima —ver la Fig. 3.14—, interesa procesarla para que el vehículo siga esta trayectoria lo mejor posible. Se observa que la trayectoria de referencia estará dada por una secuencia de posiciones z_{di} y velocidades $\dot{z_{di}}$. A los fines obtener las acciones de control necesarias para que el vehículo se desplace sobre la trayectoria hallada, se analiza *a modo de ejemplo* una estrategia de control aceptada en la literatura [43, 21]. Sin embargo, existen estrategias de seguimiento de trayectorias [43] que permiten seguir el camino con una velocidad lineal fija y elegida.

Según la estrategia a emplear en esta tesis, se define como punto a seguir por el controlador el siguiente

$$\boldsymbol{\mu} = \begin{pmatrix} x_d \\ y_d \end{pmatrix}, \tag{4.7}$$

y luego se calculan las velicidades como

$$\dot{\boldsymbol{\mu}} = \begin{pmatrix} \cos\theta_d & 0 \\ \sin\theta_d & 0 \end{pmatrix} \begin{pmatrix} v_{d1} \\ v_{d2} \end{pmatrix}, \tag{4.8}$$

dado que no interesa seguir la posición propuesta como trayectoria de referencia con el punto medio del eje trasero del vehículo; sino con uno que se encuentre fuera de éste —para evitar singularidades matemáticas—. Para esto se define un nuevo punto de salida como el siguiente

$$\boldsymbol{\mu} = \begin{pmatrix} x_d + l_m\cos\theta_d + \delta\cos(\theta_d + \phi_d) \\ y_d + l_m\sin\theta_d + \delta\sin(\theta_d + \phi_d) \end{pmatrix}, \tag{4.9}$$

donde $\delta \neq 0$. Diferenciando esta nueva salida se obtiene

$$\dot{\boldsymbol{\mu}} = \begin{pmatrix} \cos\theta_d - \tan\phi_d(\sin\theta_d + \delta\sin(\theta_d + \phi_d)/l_m) & -\delta\sin(\theta_d + \phi_d) \\ \sin\theta_d + \tan\phi_d(\cos\theta_d + \delta\cos(\theta_d + \phi_d)/l_m) & \delta\cos(\theta_d + \phi_d) \end{pmatrix} \begin{pmatrix} v_{d1} \\ v_{d2} \end{pmatrix}, \tag{4.10}$$

$$\dot{\boldsymbol{\mu}} = A(\theta_d, \phi_d) \begin{pmatrix} v_{d1} \\ v_{d2} \end{pmatrix}, \tag{4.11}$$

$$\begin{pmatrix} v_{d1} \\ \\ v_{d2} \end{pmatrix} = A^{-1}(\theta_d, \phi_d)\dot{\boldsymbol{\mu}}. \tag{4.12}$$

Luego, siguiendo la estrategia propuesta en [43, 21], se puede hacer $\dot{\boldsymbol{\mu}} = \boldsymbol{r}$ (entrada auxiliar) y el problema de seguimiento de trayectorias está dado por

$$r_i = \dot{z_{di}} + k_{pi}(z_{di} - \mu_i),\ k_{pi} > 0,\ i = 1, 2, \tag{4.13}$$

donde z_{di} y $\dot{z_{di}}$ representan la posición y la velocidad deseada, μ_i son los estados presentados en la Ec. 4.9 y k_{pi} son ganancias. El error está dado por $(z_{di} - \mu_i)$ y debe decirse que las dos variables de salida μ_i convergen de forma exponencial pero que el comportamiento de las variables θ_d y ϕ_d no puede ser determinado.

Se reitera que la trayectoria de referencia está dada por una secuencia de posiciones z_{di} y velocidades $\dot{z_{di}}$. Debido a esto, se requiere procesar en forma previa la «tira de triángulos», hallada mediante la estrategia de planificación, de forma tal que se pueda obtener una secuencia de posiciones y una secuencia de velocidades a partir de una serie de subdivisiones de forma triangular.

4.4 Unificación de planificación y control

La estrategia de planificación permite hallar una «tira de triángulos» que representa la trayectoria óptima. Además, a partir de una secuencia de posiciones y velocidades, es posible que un vehículo de cuatro ruedas siga esta trayectoria; por lo que resta unificar estas dos cosas.

En el campo de la computación gráfica, se utilizan las curvas sobre superficies generadas por mallas con el fin de visualizar características y existen algoritmos que generan y refinan estas curvas [5]. A los fines de generar una trayectoria viable desde el punto de vista cinemático, no interesa refinar sucesivamente la curva, como ocurre en [2], sino simplemente generar una en un solo paso y que sea lo mejor posible a los fines del control. Para esto, se toma el punto medio de los lados de la tira de triángulos, de forma tal que el vehículo se desplace lo más alejado posible de los obstáculos y dentro de esta tira que representa la trayectoria óptima. En la Fig. 4.4 se observa una secuencia de asteriscos que representan el punto central de los lados de las subdivisiones. Estos puntos son unidos mediante una línea de trazos. Sin embargo, existe una salvedad con respecto a la última subdivisión de cada uno de los trechos y es tratada en el Apéndice C donde se explica por qué de la última subdivisión no se toma la mitad del lado de la subdivisión sino su baricentro.

Observando la misma figura se aprecia que la trayectoria en línea de trazos tiene giros relativamente abruptos para un vehículo tipo auto con restricciones en el giro del volante como el visto. Por esto, interesa procesar los puntos de forma tal que resulte una trayectoria de referencia más suave para el vehículo. Con este fin, se procesan los puntos mediante un filtro del tipo «media móvil» de orden tres [50]. Una vez que los puntos fueron procesados, son unidos mediante una línea continua que puede apreciarse en la Figura 4.4. La trayectoria a utilizarse como referencia para el controlador resulta mucho más suave. La desventaja observada ahora es que el vehículo va a desplazarse en forma más cercana a subdivisiones externas a las que conforman la trayectoria óptima. Por esto, podría llegar a desplazarse

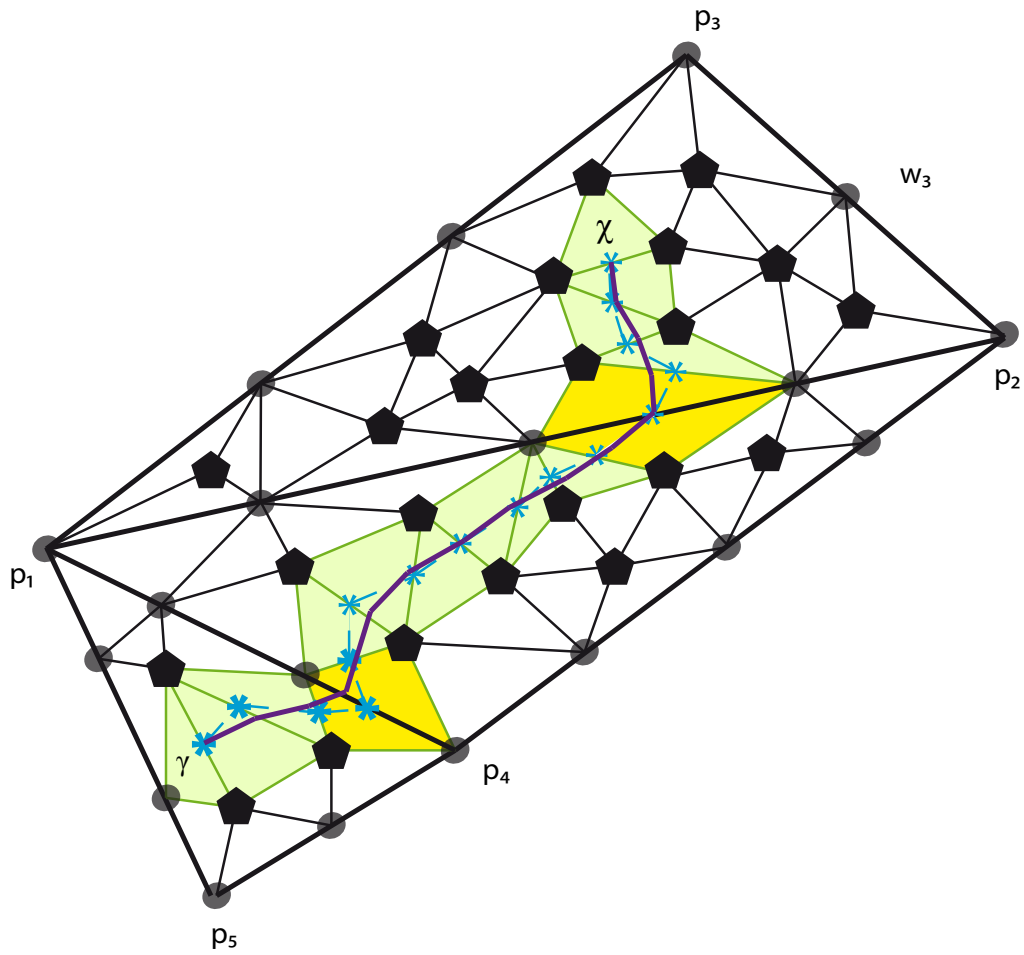


Figura 4.4: Los asteriscos indican los puntos medios de los lados adyacentes de las subdivisiones que conforman el camino óptimo entre γ y χ. Una linea discontinua une los asteriscos, generando una única trayectoria. Las coordenadas de los puntos tipo asterisco son filtradas con un filtro de tipo «media móvil» de orden tres y son unidas mediante una línea continua.

más cerca de obstáculos o situaciones desfavorables.

Una vez que las referencias de posición han sido generadas y suavizadas, resta generar una referencia de velocidad. Para esto se proponen diferentes opciones:

- La primera consistiría en hacer que el vehículo se desplace entre los puntos de referencia de posición, tomando un tiempo constante entre cada par de puntos adyacentes. Esto presenta la desventaja que la velocidad del vehículo va a resultar variable y va a ser más desafiante desde el punto de vista del control.
- La segunda, consistiría en tratar de predecir la longitud de arco entre dos puntos adyacentes de la referencia de posición mediante una heurística, como por ejemplo, a partir de la distancia geométrica entre dos puntos adyacentes en la referencia de posición.

La Fig. 4.5 muestra la distancia geométrica entre los asteriscos de la Fig. 4.4 procesados por un filtro de media móvil de orden tres. Estos valores de distancia d_j son los que se van a utilizar, junto con los valores de referencia de posición, para definir las referencias de velocidad. Así, si se desea que el vehículo tenga una velocidad longitudinal aproximadamente de v^o, se eligen el vector de tiempos $t^o = \{t_0, t_1, ..., t_n\}$ de forma tal que la velocidad lineal resulte lo más uniforme posible. Luego, las componentes del vector de tiempos se eligen en forma directamente proporcional a las distancias de los trechos de la trayectoria e inversamente proporcional a la velocidad deseada, y entonces $t_j = d_j/v^o$.

Con los valores de d_j y una velocidad longitudinal deseada de 0,5 m/s, se calculan los t_j; y con estos últimos y los valores de z_{d1} y z_{d2} se calculan las componentes $\dot{z_{d1}}$ y $\dot{z_{d2}}$, que se aprecian en las Figs. 4.6 y 4.7. Los últimos

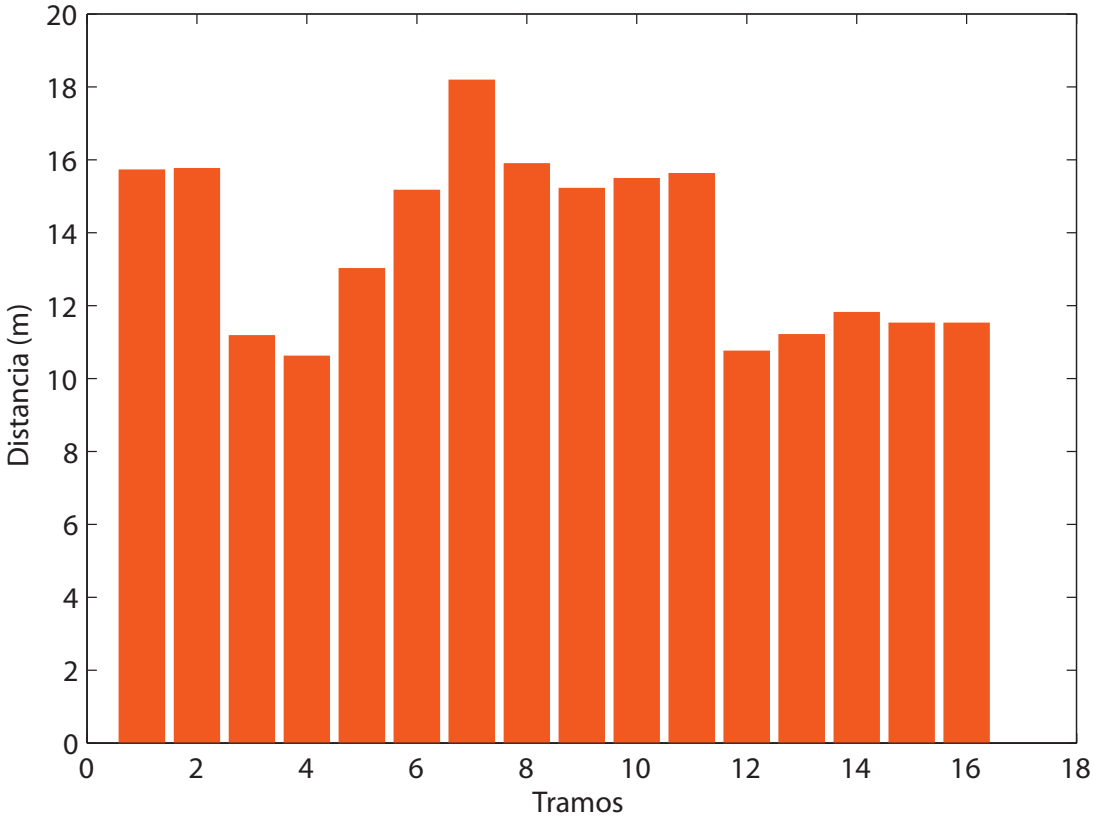


Figura 4.5: El eje horizontal indica los trechos desde el comienzo hasta el final del camino. El eje vertical muestra la distancia en metros.

cuatro valores negativos en la Figs. 4.6 se deben a que el vehículo se desplaza hacia la izquierda en el plano cartesiano hacia el final de la trayectoria. Además, combinando la información de posición: z_{d1} y z_{d2}, con la de velocidad: $\dot{z_{d1}}$ y $\dot{z_{d2}}$, se obtiene la Fig. 4.8. Allí, en cada una de las posiciones se coloca un vector cuya longitud es proporcional a la velocidad y se aprecia que la misma es uniforme.

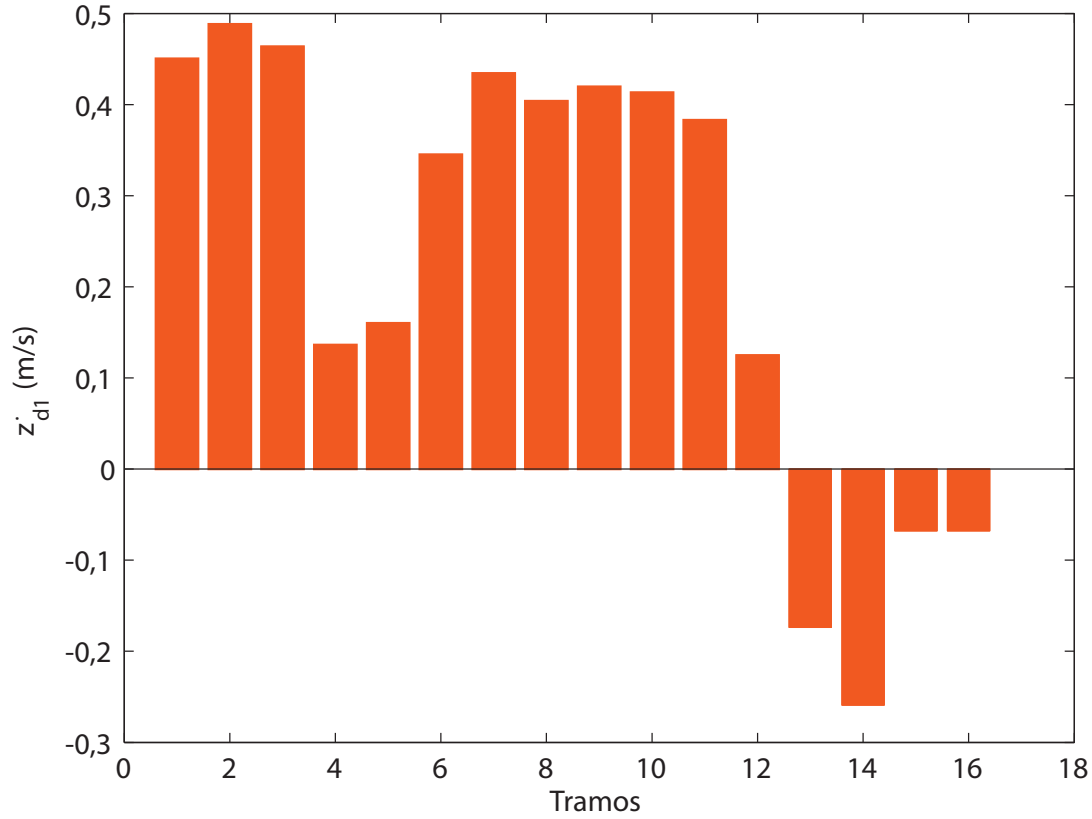


Figura 4.6: Componente de velocidad lineal $\dot{z_{d1}}$ a utilizarse como referencia en el controlador. Los últimos cuatro valores de esta velocidad son negativos ya que el vehículo se desplaza hacia la izquierda en la componente horizontal hacia el final de la trayectoria.

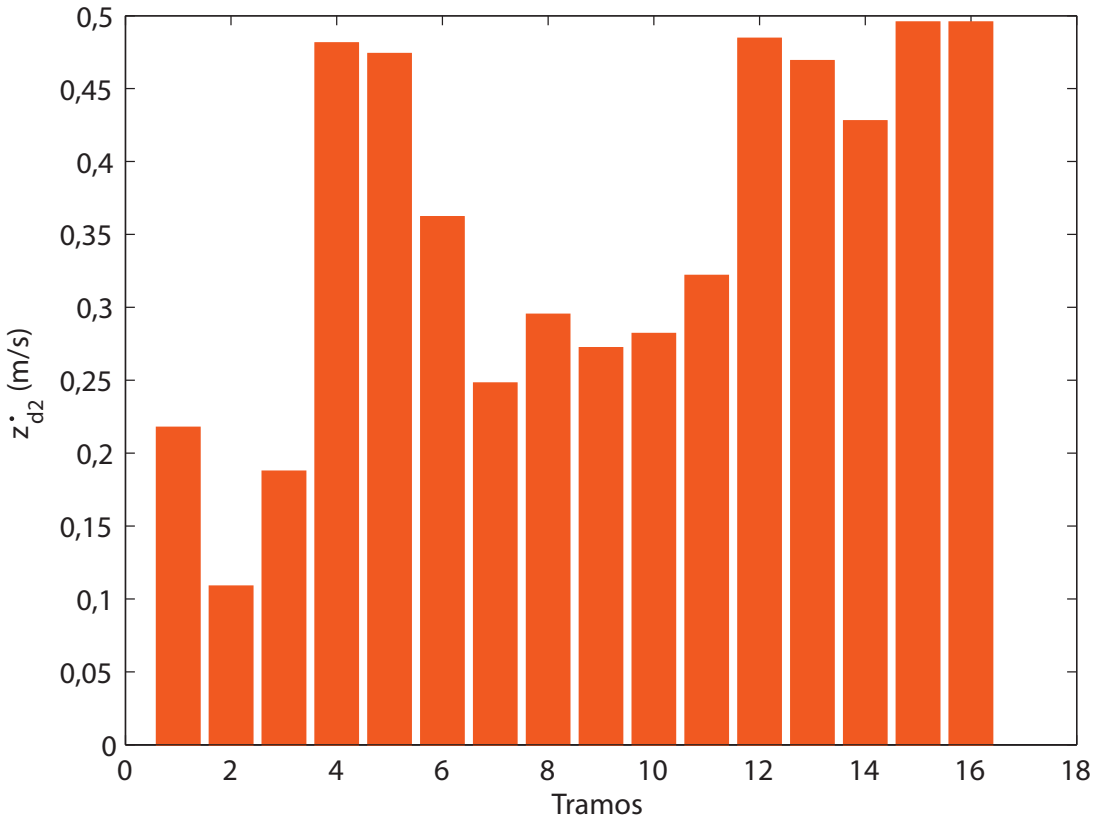


Figura 4.7: Componente de velocidad lineal $\dot{z_{d2}}$ a utilizarse como referencia en el controlador.

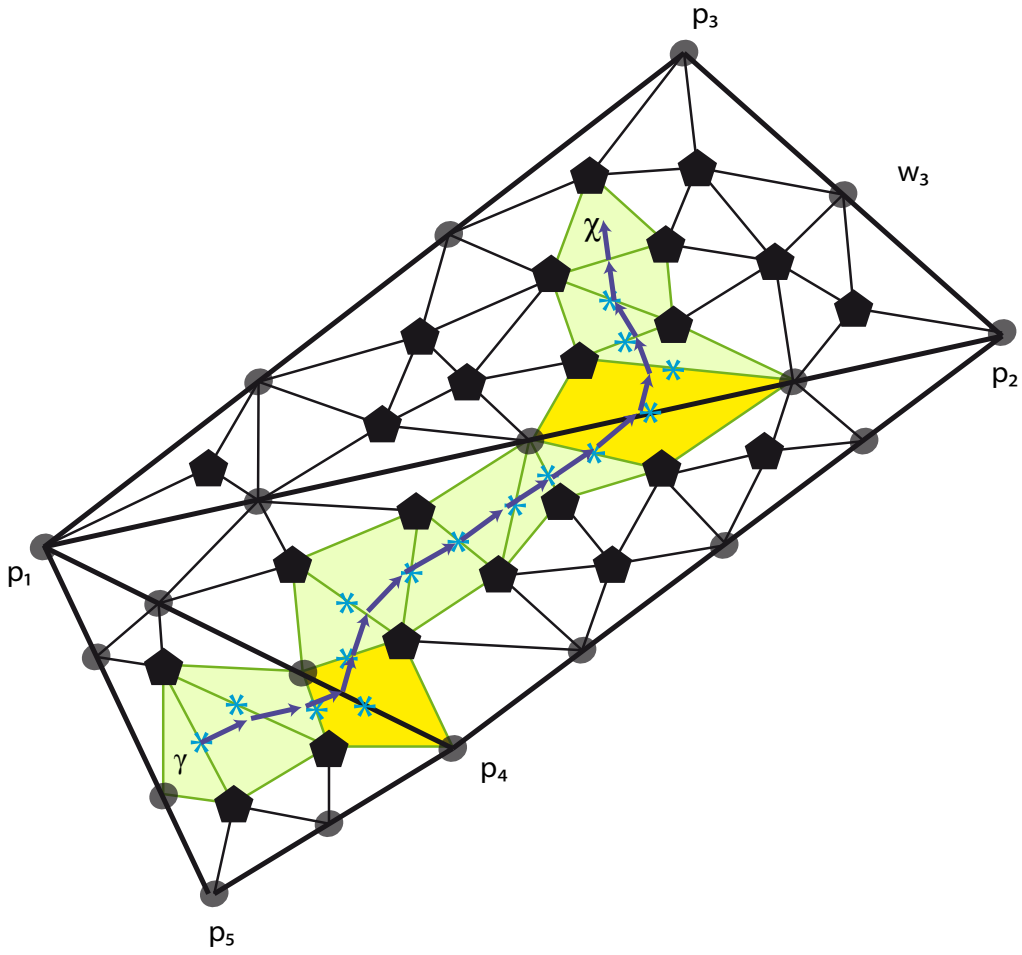


Figura 4.8: Para cada una de las posiciones que resultan de filtrar los puntos medios de los lados adyacentes de la trayectoria, se agrega un vector cuya longitud es proporcional a la velocidad lineal requerida. Se observa que la misma es uniforme adrede.

4.5 Expansión del mapa

Si bien el método de triangulación utilizado para generar las regiones locales permite incluir mojones que aparezcan posteriormente a haber generado la triangulación, puede llegar a darse el caso en el cual el vehículo explore y halle un mojón nuevo p_0^n que tiene un valor de coordenada y mayor al del punto p_0. En principio, esto representaría una limitación del método por lo que se plantea el siguiente razonamiento: el grafo acíclico dirigido se expande hacia abajo a medida que aparecen nuevos mojones y cabría pensar que el mismo grafo podría llegar a expandirse en la dirección opuesta desde la raíz. Para que se diese este escenario, el nodo que sería el padre del nodo raíz, debería representar un triángulo T_{-1} formado por los puntos auxiliares p_{-1} y p_{-2} y un punto cuya coordenada en y sea superior a la del punto p_0. Si este último se elige como el triángulo $p_{-2}\,p_{-1}\,p_0^n$, el nodo raíz anterior pasaría a ser uno de sus nodos hijos. Resta ver cuales serían sus otros dos descendientes. La Fig. 4.9 muestra el nuevo triángulo $p_{-2}\,p_{-1}\,p_0^n$ y tres triángulos interiores: el triángulo que era el nodo raíz y los triángulos T_{04} y T_{03}. Cada uno de estos dos últimos surgen de unir un punto auxiliar con el nuevo mojón y el punto p_0.

En la Fig. 4.10 se observa como queda conformado el nuevo grafo después de haber agregado el mojón p_0^n. Se observa que el nuevo nodo raíz representa el triángulo T_{-1} y los triángulos T_{04}, T_0 y T_{03} son sus descendientes. Además, la Fig. 4.11 muestra los enlaces que deben agregarse antes de analizar si los lados de los triángulos agregados son válidos, de acuerdo a la definición vista.

Resta analizar si los lados de los nuevos triángulos T_{04} y T_{03} son válidos,

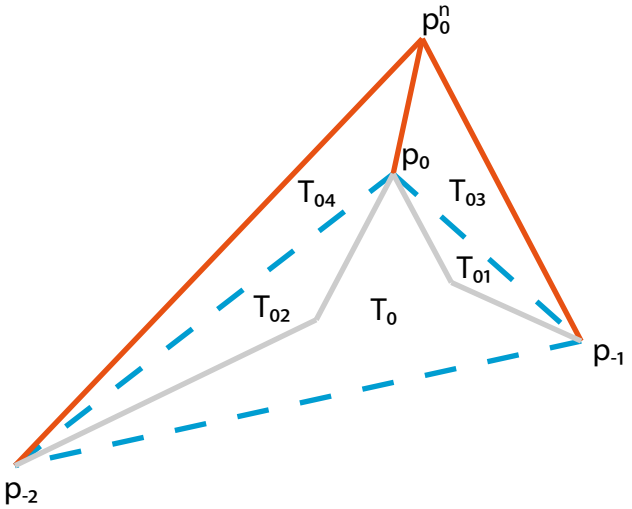


Figura 4.9: En línea de trazos se muestra el triángulo que representa el nodo raíz de la triangulación anterior T_0; además, el nuevo mojón p_0^n genera los triángulo T_{04} y T_{03} y finalmente, los triángulos T_{02} y T_{01} se encuentran en los bordes de la triangulación anterior.

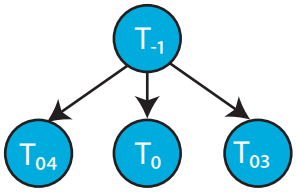


Figura 4.10: Nodos resultantes inmediatamente después de agregar el nuevo mojón y antes de calcular si los lados de los nuevos triángulos son válidos. Los enlaces son los que corresponden al grafo acíclico dirigido.

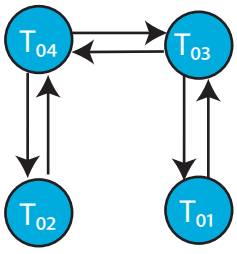


Figura 4.11: Los enlaces que se agregan inmediatamente después de incluir el nuevo mojón y antes de verificar si la triangulación resultante es válida.

de forma tal que la triangulación resultante siga siendo una triangulación de Delaunay. En el caso del lado adyacente entre T_{04} y T_{03}, sería absurdo que el mismo no fuese válido ya que no existe otra posibilidad de formar un lado. Con respecto al lado adyacente entre T_{04} y T_{02}, el mismo debe ser validado y debe procederse en forma recurrente con los dos lados restantes de T_{02}. Algo análogo ocurre con los triángulos T_{01} y T_{03}.

4.6 Conclusiones del capítulo

En este capítulo, se presentó la forma en que una trayectoria puede ser reconstruida a partir de los resultados arrojados por el módulo de planificación global. A continuación, se presentó un modelo cinemático de un vehículo tipo auto y se observaron sus limitaciones. Luego, se analizó una estrategia de control existente y se propuso un método que permite integrar en forma eficiente el módulo de control con el de planificación. Finalmente, se presentó una técnica que permite que el mapa se expanda y se genere una triangulación de Delaunay en forma parcial (es decir, se construye con subconjuntos de todos los puntos a triangular y se llega a la misma triangulación que procesando la totalidad de los puntos).

El próximo capítulo presentará resultados de planificación y control utilizando un conjunto de datos experimentales obtenidos en un ambiente no estructurado.

Capítulo 5

Simulaciones y datos experimentales

El objetivo de este capítulo es presentar una serie de resultados experimentales basados en la implementación del esquema de planificación y su aplicación al control de vehículos terrestres.

En primer lugar, se expondrán resultados de planificar con un conjunto de datos obtenidos en un ambiente exterior no estructurado.

A continuación, se mostrarán las acciones de control necesarias para que un vehículo siga la trayectoria planificada.

5.1 Planificación en un ambiente no estructurado

En esta sección, interesa analizar los resultados arrojados por el algoritmo de planificación al aplicarlo a un caso concreto con datos experimentales para luego generar las acciones de control requeridas para desplazar

un vehículo todo terreno.

El conjunto de datos corresponde a la ejecución de una implementación del algoritmo de SLAM. El lugar de trabajo se encuentra en «Victoria Park», en la ciudad de «Sydney» que se observa en la Fig. 5.1. El algoritmo entrega un vector que representa la ubicación de mojones pero no provee la altura del terreno, por lo que se procedió de la siguiente manera: un conjunto de puntos de coordenadas —latitud y longitud— del parque fue extraído y mediante un software de mapeo de GPS [28] (ver la Fig. 5.2), la altura de cada uno de los puntos fue obtenida y representada en la Fig. 5.3. De esta forma, se cuenta con un conjunto de mojones y un conjunto de puntos que representan la altitud del parque. Estos dos conjuntos de datos son los que se utilizan para corroborar y observar el funcionamiento del algoritmo propuesto.

Algunos de los puntos que representan los mojones fueron elegidos para triangular el terreno en regiones locales, mientras que los demás fueron elegidos como obstáculos. Además, los puntos que representan la altura del terreno son los que se utilizaron para dividir las regiones en subdivisiones. En la Fig. 5.4(a) se observan las regiones locales con forma de triángulo y sólo se incluyen las subdivisiones en los bordes de las regiones a los fines de claridad. El punto de partida tiene coordenadas $(\gamma) = (26;\ -26)$ y el de llegada es $(\chi) = (30;\ -200)$.

La trayectoria calculada mediante el algoritmo de planificación resulta ser la tira de triángulos que une el origen con el destino. Esta trayectoria debe ser procesada con el fin de hallar las acciones de control necesarias para poder recorrerla con un vehículo. En primer lugar, se genera una secuencia de puntos dada por el punto medio de los lados de la tira de triángulos y

Figura 5.1: Lugar de ejecución del algoritmo de SLAM: «Victoria Park», «Sydney». Fuente: *Servicio de imágenes del planeta Google Earth*™.

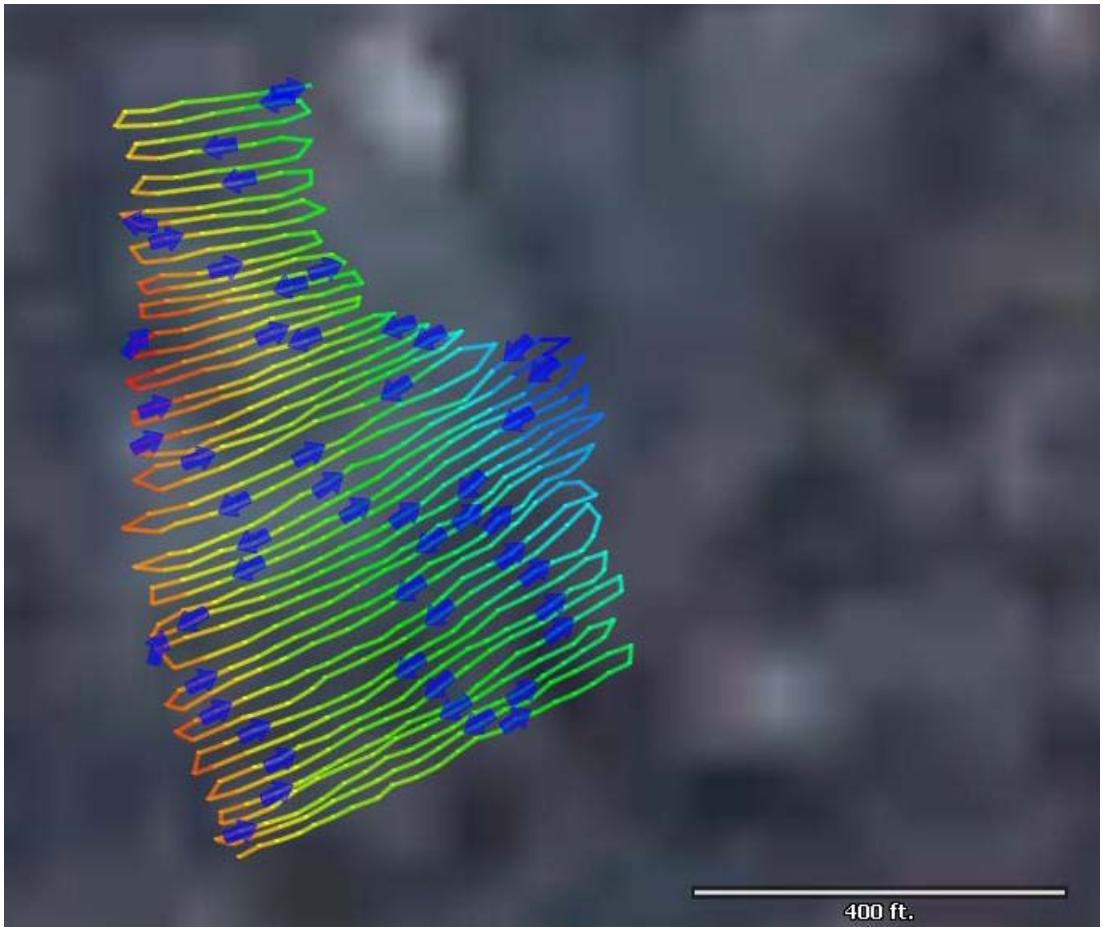


Figura 5.2: Algunos puntos fueron elegidos sobre la superficie del parque a los fines de obtener la altitud del terreno y contar con una capa de información densa del mismo. Los puntos de referencia se unen con una línea continua desde abajo hacia arriba. El software de mapeo de GPS cuenta con la información topológica de la zona. La escala se encuentra en la esquina inferior derecha en unidades inglesas (400 pies = 121,92 m). Fuente: *TopoFusion Pro*.

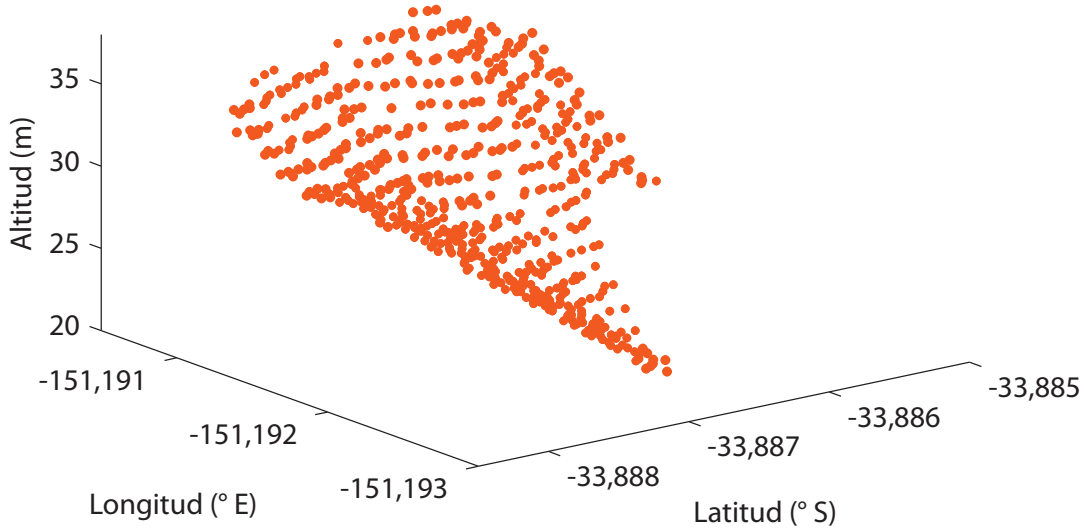
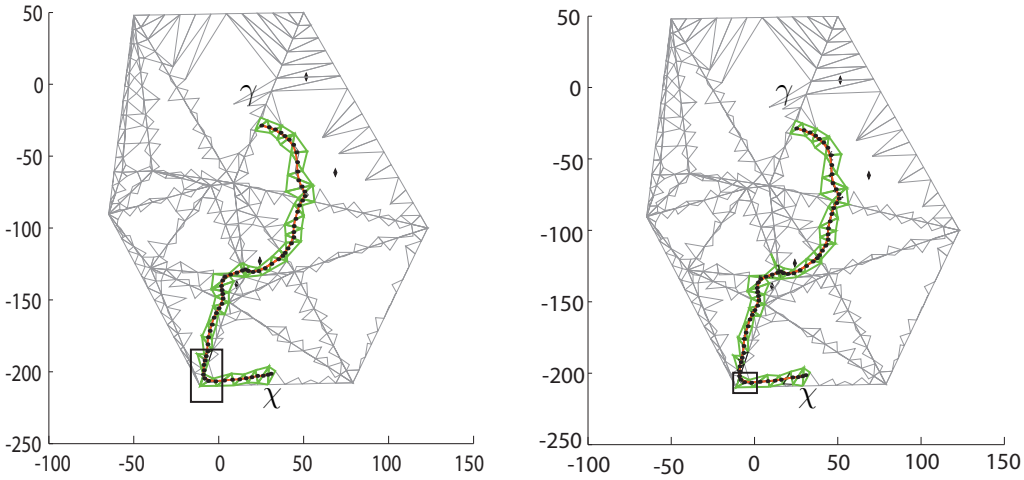


Figura 5.3: Datos de altitud extraídos de información de GPS para la zona de interés. Uno de los ejes representa la latitud en grado, el otro la longitud en la misma unidad y el tercero la altitud en metros. A los fines de orientación, se nota que el óvalo que aparece en la parte superior de la foto del parque se encuentra en esta figura sobre el borde de la derecha como un arco.

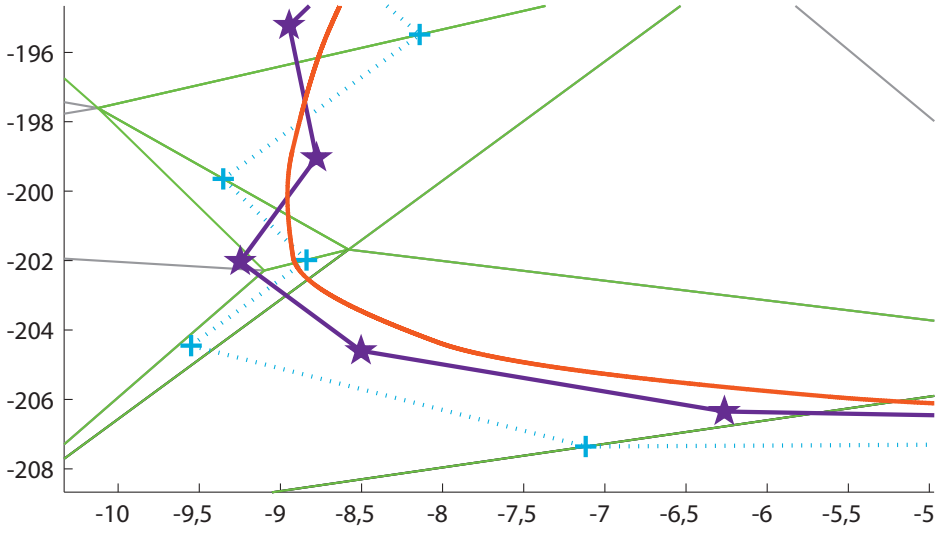


(a) Velocidad lineal no uniforme. (b) Velocidad lineal uniforme.

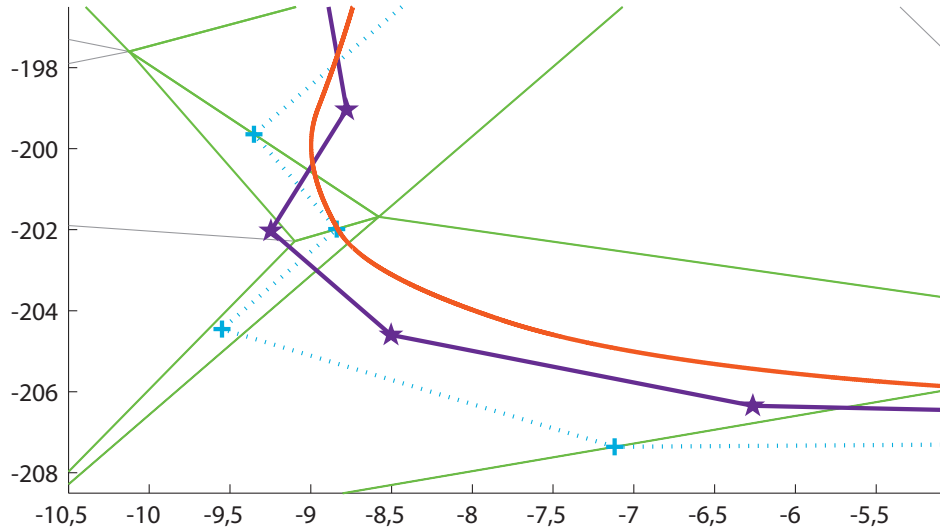
Figura 5.4: Resultados de planificación óptima global. Las regiones triangulares muestran las subdivisiones en los bordes. Los obstáculos se indican mediante pequeños rombos. El punto de partida tiene coordenadas $(\gamma) = (26; -26)$ y el de llegada es $(\chi) = (30; -200)$. La trayectoria óptima global entre el origen y el destino está representada por una «tira de triángulos». A partir de la trayectoria se genera una secuencia de puntos dada por el punto medio de los lados de la tira de triángulos y se suaviza por un filtro de media móvil de orden tres. Finalmente, se genera la trayectoria y las acciones de control mediante simulación numérica y se muestra mediante línea gruesa. El recuadro indica una zona que va a ser amplificada en la próxima figura. Los ejes están en metros.

se observa en la Fig. 5.5(a) como una línea punteada con signos más. Luego, la secuencia de puntos anterior se suaviza mediante un filtro de media móvil de orden tres y se observa en la misma figura como una línea continua y estrellas. Finalmente, la figura muestra con una línea continua la trayectoria generada por el controlador.

Las Figs. 5.4(a) y 5.5(a) resultan ser trayectorias donde la velocidad lineal del vehículo es variable. Pueden generarse trayectorias donde la velocidad lineal del vehículo es uniforme (esto se hacía determinando el vector de tiempos a partir de la distancia geométrica entre dos puntos sucesivos



(a) Velocidad lineal no uniforme.



(b) Velocidad lineal uniforme.

Figura 5.5: Amplificación de una parte de la figura anterior. La línea de puntos y los signos '+' indican la trayectoria que surge de tomar el punto medio de los lados de la secuencia de triángulos que representa la trayectoria óptima. La línea continua a tramos y las estrellas indican la trayectoria que surge de aplicar un filtro de media móvil de orden tres a los puntos anteriores. Finalmente, la trayectoria controlada y simulada se representa con una línea continua. Los ejes representan las distancias en metros.

de la referencia de posición). La trayectoria resultante y una amplificación de la misma se aprecian en las Figs. 5.4(b) y 5.5(b). Se puede argüir de estas cuatro figuras que el hecho de modificar el vector de tiempos para obtener este tipo de velocidad no afecta significativamente la forma de la trayectoria.

5.2 Trayectorias controladas

Como fue presentado en el Capítulo 4, a partir de una referencia de posición dada por una secuencia de puntos; se puede generar mediante una heurística y una velocidad lineal deseada, una referencia de velocidad para un controlador de seguimiento de trayectorias. A partir de los puntos de referencia entregados por el módulo de planificación global, y una velocidad lineal deseada $v^o = 3$ m/s, y de simulaciones numéricas se obtienen que la orientación del chasis del vehículo resulta como se aprecia en la Fig. 5.6; mientras que el ángulo del volante resulta como se observa en la Fig. 5.7. Además, las acciones de control v_{d1} y v_{d2} se observan en las Figs. 5.8 y 5.9.

Con respecto al ángulo de la dirección del vehículo ϕ, se aprecia que los valores del ángulo son inferiores a los límites del vehículo ($\overline{\phi_d} = 24°11'$) y por lo tanto esta acción de control es viable. Además, se observa en la Fig. 5.8 que la velocidad lineal del vehículo resultó muy cercana al valor deseado de 3 m/s; pero no es exacta ya que el vector de tiempos entre dos referencias de posición fue inferido a partir de la distancia entre estas dos referencias. La única forma de que resultase la velocidad lineal exactamente igual a la deseada sería conociendo la longitud de arco de la trayectoria entre los puntos de referencia de posición, pero esto no es posible ya que la

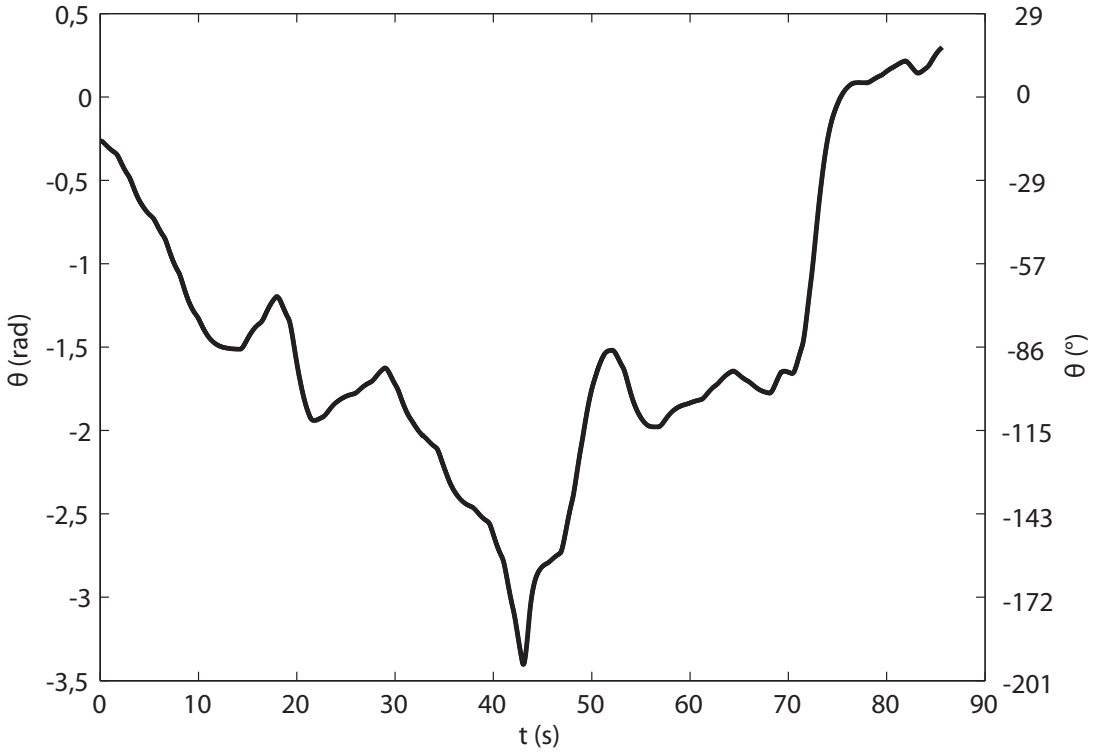


Figura 5.6: Resultado de implementar el controlador. Ángulo del chasis del vehículo para velocidad uniforme.

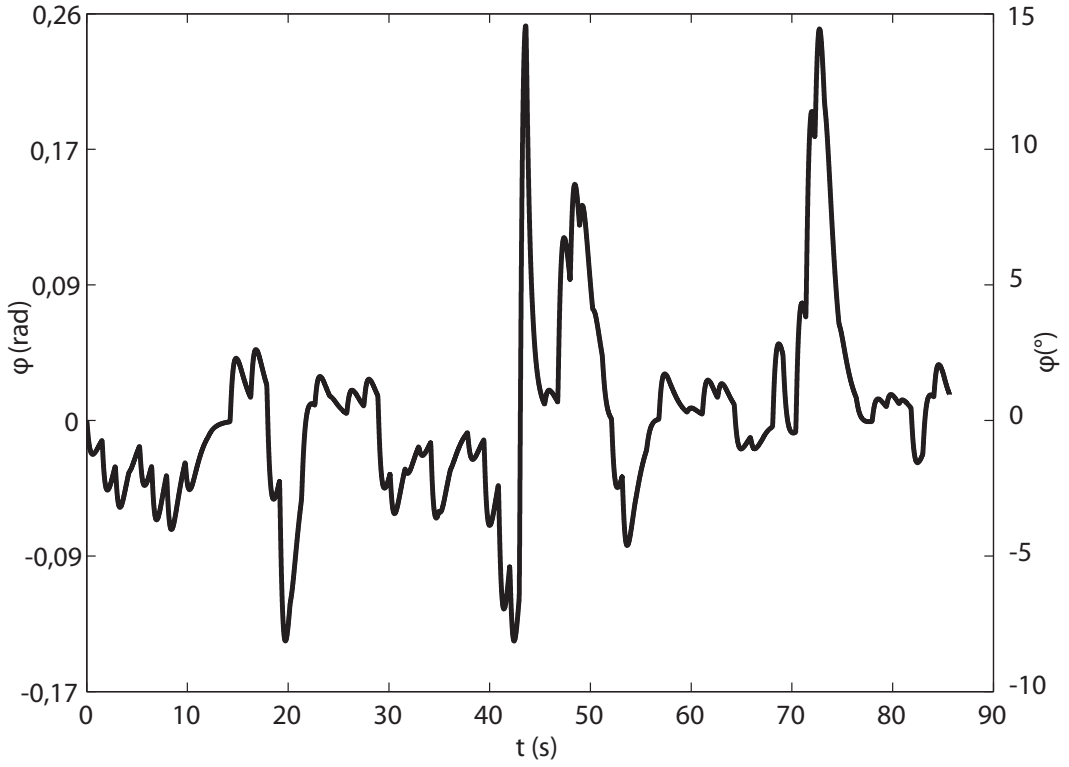


Figura 5.7: Resultado de implementar el controlador. Ángulo de la dirección del vehículo para velocidad uniforme.

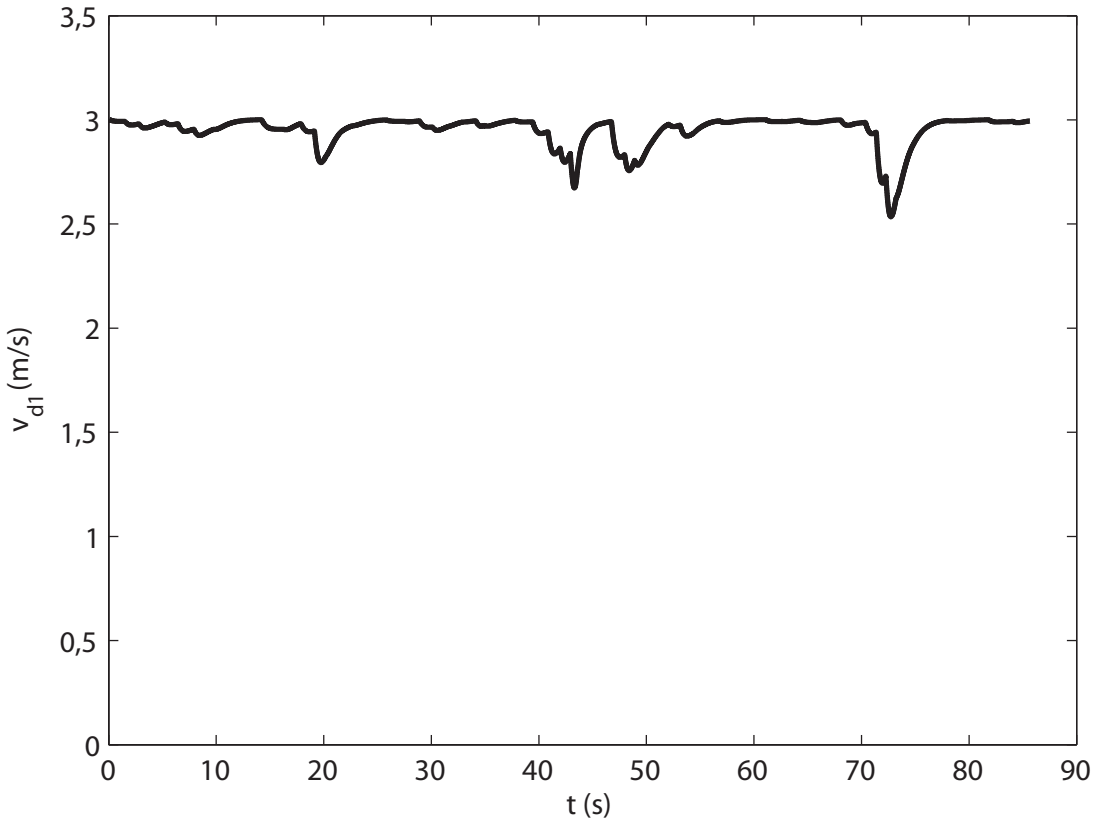


Figura 5.8: Resultado de implementar el controlador. Velocidad lineal del vehículo para referencia de velocidad uniforme.

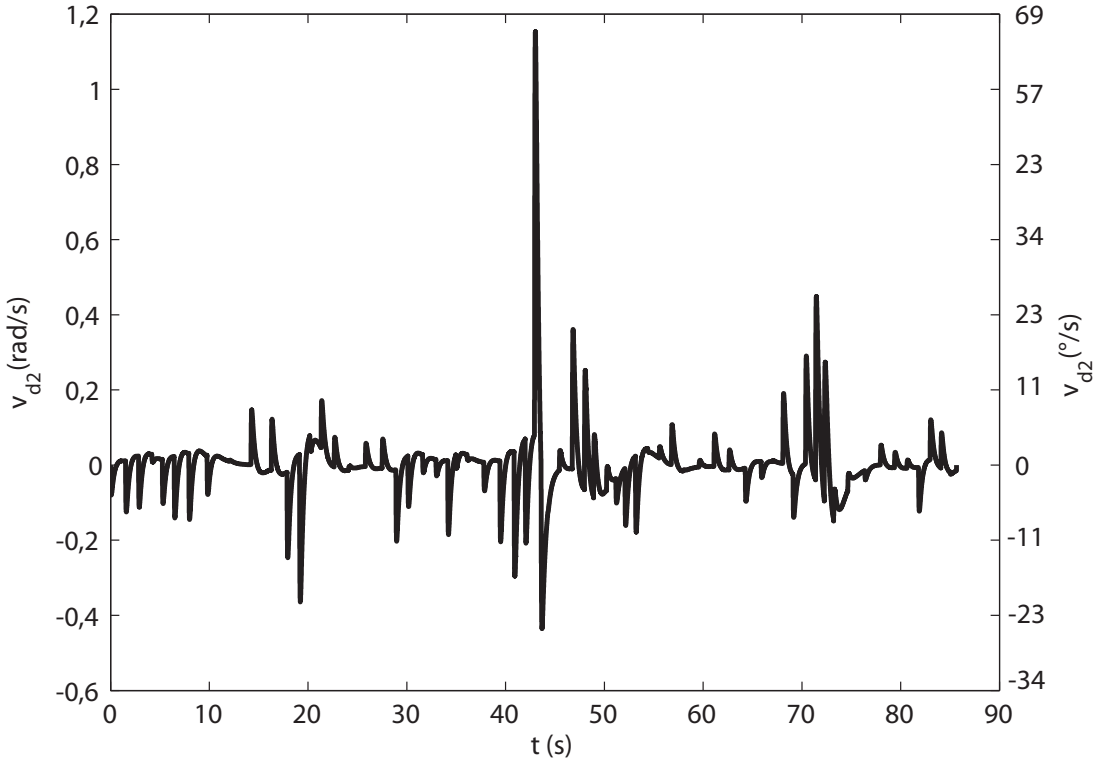


Figura 5.9: Resultado de implementar el controlador. Velocidad angular del vehículo para referencia de velocidad uniforme.

trayectoria no es conocida de antemano. Resta analizar como generar las acciones de control en forma práctica y no mediante las simulaciones numéricas que se utilizaron para analizar los resultados de la planificación y su unificación con la estrategia de control.

5.3 Implementación del controlador

Observando la Ec. 4.9 se observa que las variables x_d, y_d y ϕ_d estarían dadas por la posición del vehículo y el ángulo de su dirección. La información de ubicación de la plataforma es provista por el algoritmo de SLAM.

Una vez que se tienen los valores anteriores se calculan las acciones de control v_{d1} y v_{d2}, por lo que resta calcular el valor futuro de ϕ_d. Para esto se observa la última de las Ecs. 4.1 y teniendo en cuenta la regla de Euler [49], se desprende que

$$\phi_d[k+1] = T\, v_{d2}[k] + \phi_d[k], \tag{5.1}$$

donde T es el intervalo de muestreo; por lo que se deduce que el control se reduce desde el punto de vista práctico a obtener desde el algoritmo de SLAM la ubicación, calcular las velocidades v_{d1} y v_{d2} y finalmente calcular el ángulo de la dirección. En la Fig. 5.10 se muestra el valor del ángulo ϕ_d pero esta vez no calculado mediante una simulación numérica sino a partir de la Ec. 5.1 para un $T = 1$ ms. Comparando las Figs. 5.7 y 5.10 no se observan diferencias por lo que se supone que este intervalo de muestreo es suficiente.

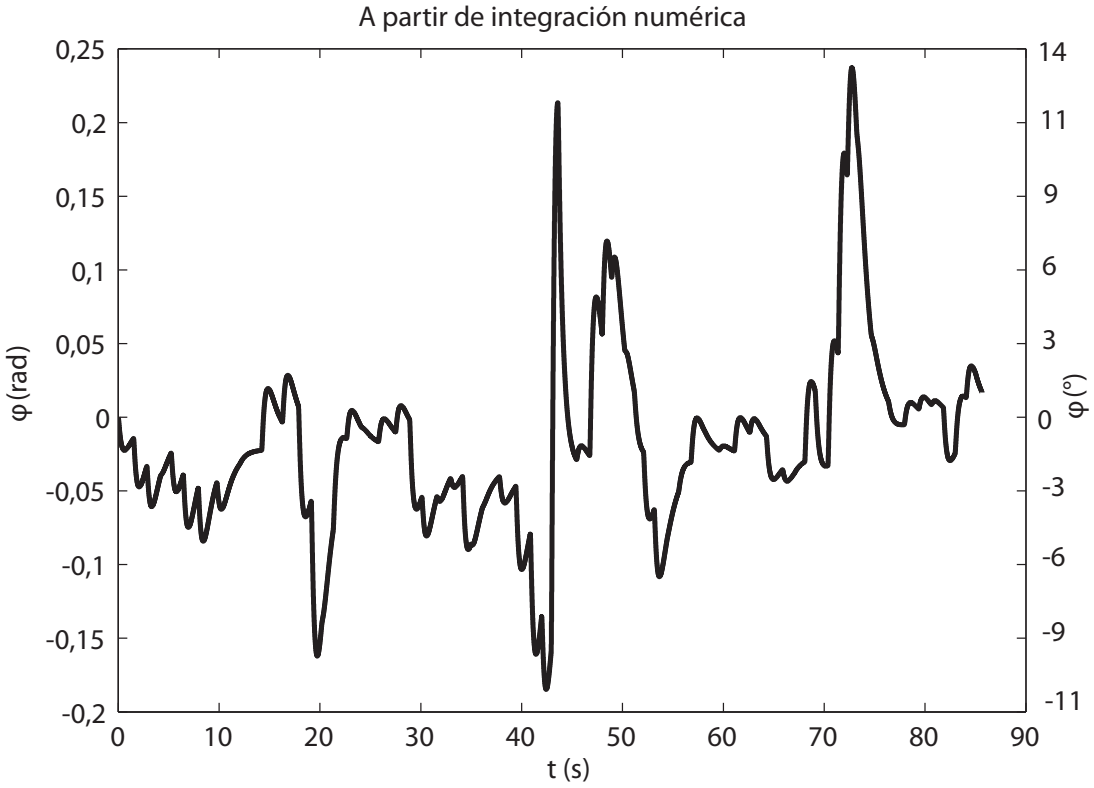


Figura 5.10: Resultado de implementar el controlador. Ángulo de la dirección del vehículo a partir de integración numérica mediante la regla de Euler.

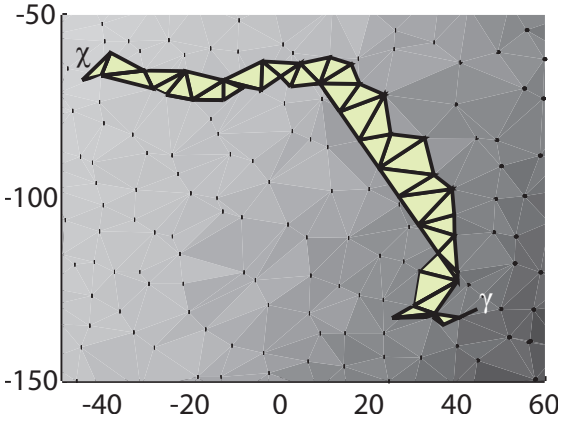
5.4 Criterio de planificación

Además del ejemplo con datos experimentales presentado anteriormente, interesa agregar otro más en el que se aprecie el efecto de variar el criterio de optimización a nivel local, o sea, dentro de las regiones. Para esto, se recuerda que la Ec. 3.3 permite incluir los efectos en los cambios de la altura del terreno. En este caso, se va a utilizar esta misma ecuación y el término k_2 va a tomar los valores $k_2 = \{0;\ 0{,}1;\ 1\}$. En la Fig. 5.11 el punto de partida tiene coordenadas $(\gamma) = (45;\ -135)$ y el destino es $(\chi) = (-45;\ -65)$ y la intensidad del color del fondo indica la altura ya que las zonas más claras representan zonas más altas. En la Fig. 5.11(a) se observa que como la constante k_2 es nula, el criterio de optimización consiste en hallar el camino más corto posible entre el origen y el destino mientras que en las Figs. 5.11(b) y 5.11(c), se incrementa el parámetro y se observa como el algoritmo de planificación está más dispuesto a sacrificar distancia con el fin de desplazarse de una forma más conveniente desde el punto de vista de cambio en la altura del terreno. Se aprecia además que las Figs. 5.11(b) y 5.11(c) son muy similares y solo se observan cambios en torno a $x = -20$.

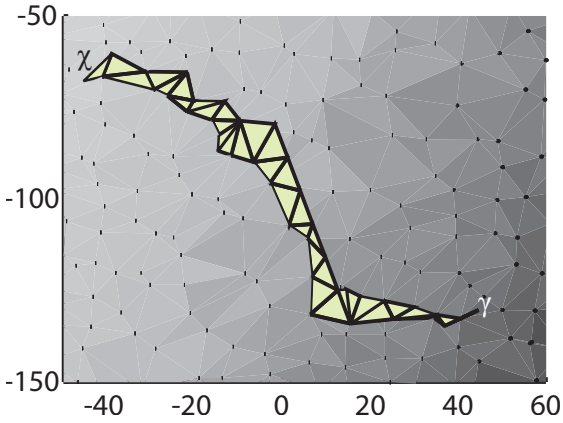
El algoritmo propuesto permite incorporar diversas capas de información a la planificación y de esta forma se aprecia como el camino óptimo variaría a medida que se incorporan datos al mapa.

5.5 Conclusiones del capítulo

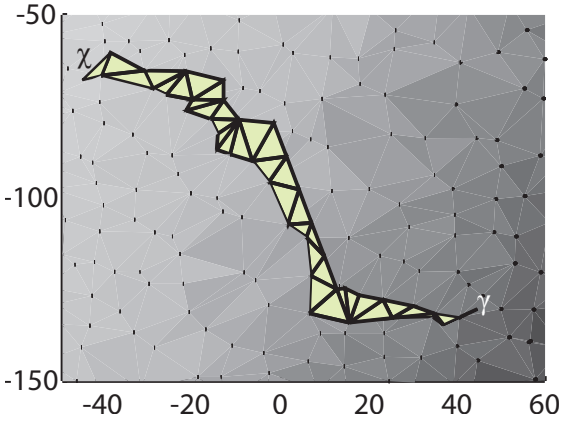
Este capítulo presentó resultados de simulaciones y datos experimentales obtenidos en un ambiente no estructurado. Se mostraron las mediciones y se planificó utilizando el algoritmo propuesto. Luego, a partir de las tra-



(a) $k_2 = 0$.



(b) $k_2 = 0{,}1$.



(c) $k_2 = 1$.

Figura 5.11: Tres trayectorias óptimas para diferentes valores de k_2. Los fondos de las figuras representan la altura del terreno y las zonas más claras se corresponden con alturas mayores. El origen y el destino se representan mediante γ y χ, respectivamente.

yectorias generadas por el algoritmo, se implementó la estrategia de control y se analizaron los resultados arrojados por la misma.

Se observa que, en general, el ángulo de giro del volante requerido por el controlador se encuentra dentro de los límites admisibles.

Luego, se varió la estrategia de planificación. Se observa que modificando la misma, se llega a diferentes soluciones. Cada una pondera de distinta forma la información existente en el mapa.

El próximo capítulo presentará conclusiones generales sobre los resultados obtenidos en esta tesis.

Capítulo 6

Conclusiones generales

La motivación de esta tesis fue integrar las técnicas que permiten la localización y el mapeo en forma simultánea con los algoritmos de planificación. Esto permite que un vehículo se desplace sobre un ambiente desconocido y construya un mapa y al mismo tiempo cumpla un objetivo. Se concluye que aplicando técnicas de computación geométrica se pueden generar regiones y subdivisiones internas a partir de un vector de mojones. Las regiones resultantes son la mejor subdivisión posible del mapa desde el punto de vista de los ángulos internos de las regiones por lo que se logra una muy buena división del mapa, como se observó utilizando datos experimentales.

Además, las subdivisiones interiores que surgen a partir de los puntos sensados y agregados artificialmente permiten subdividir las regiones para agregar las múltiples capas de información a medida que los datos son sensados. Las funciones de costo definidas permiten combinar las múltiples capas de información de los mapas de acuerdo a un criterio elegido de antemano. Si bien la implementación de la triangulación parecía tener una

limitación con respecto a que no podrían agregarse mojones más allá del mojón cuya coordenada en y toma el máximo valor, debido a las ideas presentadas, los mojones pueden ser triangulados en etapas y pueden tomar cualquier valor de coordenadas.

Al mismo tiempo, interesa que el algoritmo planifique; permitiéndose alcanzar objetivos, adaptarse a los cambios del ambiente y la incertidumbre —que es manejada por el algoritmo de SLAM y no es utilizada de forme explícita en la planificación—. En este sentido, el algoritmo propuesto permite ir construyendo los mapas en forma creciente con respecto a los mojones. La forma en que se construyen las regiones y sus subdivisiones internas permite ir construyendo en forma simultánea grafos de regiones y grafos dentro de las mismas sin agregar complejidad a la que ya se requiere para triangular, debido a que las operaciones que permiten su construcción se ejecutan en tiempo constante u $O(1)$. Además, al utilizar mapas obtenidos por el algoritmo de SLAM, el esquema presentado puede manejar la incertidumbre y no solamente los cambios puntuales sobre un mapa conocido y sin incertidumbre como lo hace el algoritmo D*. También, la estrategia de planificación en dos etapas, permite hallar los caminos óptimos en el interior de las regiones y luego hallar el camino óptimo entre la posición actual del vehículo y uno o múltiples destinos. Esta división en dos etapas permite modificar la información del ambiente en una de las regiones y sólo hace falta ejecutar la planificación local en la región actualizada. Otra ventaja que presenta es que se puede ejecutar la segunda etapa si hay cambios en los objetivos a alcanzar y la planificación necesaria es mínima.

Al planificar en forma local los caminos son almacenados y una vez ejecutada la planificación global, mediante índices puede reconstruirse en for-

ma inmediata el camino óptimo global, gracias a la idea de los dos índices que registra por qué subdivisión se ingresó a una región y por qué subdivisión se salió de la región previa.

Asimismo, las trayectorias halladas pueden integrarse con técnicas de control existentes. Con el fin de aplicar el algoritmo de planificación propuesto a un vehículo todo terreno, se analizó y calculó el ángulo máximo de la dirección de las ruedas delanteras. Se observa a partir de simulaciones que en este caso en particular el ángulo requerido por el controlador es inferior al máximo admisible por lo que esto no representa una limitación. Esto puede deberse a que este tipo de vehículos permiten ser maniobrados fácilmente. Además, se unificó el resultado arrojado por el módulo de planificación con una técnica de control existente. De esta forma, se calculan las acciones de control necesarias para el vehículo todo terreno. Esta unificación surge a partir de una heurística que permite que la velocidad lineal resulte prácticamente uniforme y se basa en determinar el vector de tiempos a partir de las referencias de posición generadas por el módulo, obteniendo referencias de velocidad. Las mismas conducen a que la velocidad lineal del vehículo resulte uniforme. Estas acciones de control se calculan en forma sencilla y pueden ser implementadas fácilmente mediante sumas y multiplicaciones con un intervalo de muestreo razonable. Por todo esto se discurre que la interfaz definida entre los módulos de planificación y control resulta eficiente, práctica y posible de ser implementada con requisitos mínimos de cálculo computacional.

Puede afirmarse que el algoritmo propuesto sigue los principios de la robótica moderna, según los cuales, se deben emplear módulos que puedan utilizarse nuevamente con diferentes tipos de mapas y diferentes tipos de

robots. En este caso, lo único que se supuso sobre los mapas es que los mismos cuentan con mojones y no se hizo ninguna consideración sobre el vehículo.

Todos los costos computacionales fueron calculados o estimados. Las curvas de complejidad computacional obtenidas en forma experimental se condicen con las predicciones teóricas y de esto se colige que los tiempos de ejecución de las rutinas son razonables y factibles de ser ejecutadas en una aplicación práctica utilizando en forma dinámica la memoria, empleando objetos, módulos y los principios de la arquitectura del software.

Los vehículos pueden ser de diversa naturaleza: terrestres, espaciales o acuáticos. Estas técnicas pueden aplicarse en supervisión, rescate, agricultura, minería o cualquier otra que requiriese de vehículos autónomos y que admita una representación del ambiente en un mapa con mojones.

Apéndices

Apéndice A

Triángulos auxiliares

Una vez que un conjunto de puntos fue triangulado mediante la triangulación de Delaunay, interesa determinar cuáles de los triángulos son relevantes a los fines de la triangulación y cuáles son auxiliares ya que como se aprecia en la Fig. A.1, resultan de ambos tipos luego de triangular. Para esto se debe tener en cuenta la Fig. 3.1 donde los puntos $\mathbf{p_0}$, $\mathbf{p_{-1}}$ y $\mathbf{p_{-2}}$ son puntos auxiliares y se procede de la siguiente manera, dado un triángulo de vértices i, j y k y un vértice l de un triángulo adyacente, pueden darse cuatro casos: el punto cae en el interior del triángulo, en el lado ij, en el lado ik o en el lado jk.

Cuando aparece un punto nuevo que divide un triángulo existente, interesa determinar si los triángulos resultantes son de tipo auxiliares (o sea que uno de sus vértices es un punto auxiliar). Pueden llegar a darse cuatro casos que se aprecian en la Fig. A.2. El primer caso se observa en la Fig. A.2(a) y se implementa de la siguiente manera

Punto interior

```
1 Si el triángulo es tipo auxiliar e ((i == i0) ó (i == j0))
```

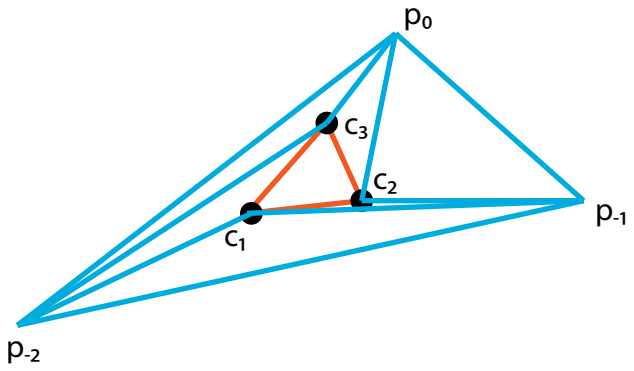
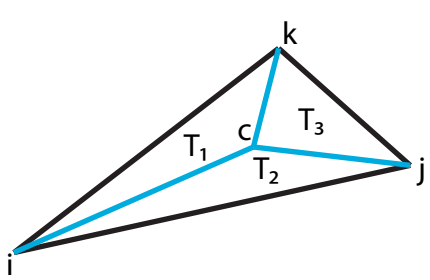
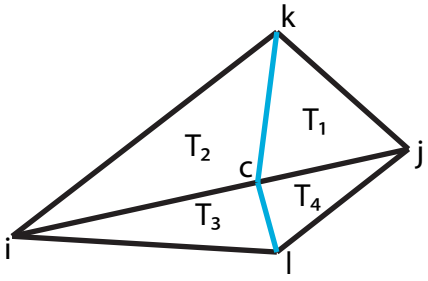


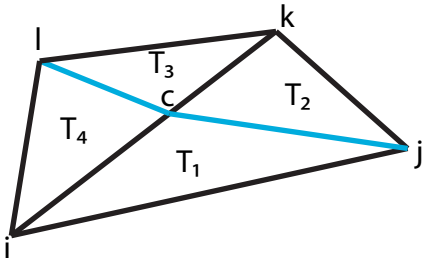
Figura A.1: Tres puntos c_1, c_2 y c_3 son procesados mediante la triangulación de Delaunay. Para esto se agregan tres puntos auxiliares p_0, p_{-1} y p_{-2}. Los triángulos azules son auxiliares y el naranja es el resultante.



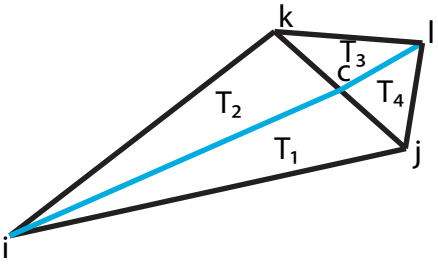
(a) Punto interior.



(b) Sobre lado ij.



(c) Sobre lado ik.



(d) Sobre lado jk.

Figura A.2: Para determinar si un triángulo es auxiliar, pueden darse cuatro casos al momento de dividir un triángulo existente ijk con un punto c. El punto l es el vértice de un triángulo adyacente.

```
        entonces T1 y T2 son auxiliares
Si el triángulo es tipo auxiliar y ((j == i0) ó (j == j0))
        entonces T2 y T3 son auxiliares
Si el triángulo es tipo auxiliar y ((k == i0) ó (k == j0))
        entonces T1 y T3 son auxiliares
```

Además, si el punto cae sobre el lado ij, como se observa en la Fig. A.2(b), se analiza como a continuación

```
Sobre lado ij
Si i es punto auxiliar
        entonce T2 y T3 son auxiliares
Si j es punto auxiliar
        entonces T1 y T4 son auxiliares
Si k es punto auxiliar
        entonces T1 y T2 son auxiliares
Si l es punto auxiliar
        entonces T3 y T4 son auxiliares
```

Luego, si el punto se encuentra en el lado ik (ver la Fig.A.2(c)) se analiza de la siguiente manera

```
Sobre lado ik
Si i es punto auxiliar
        entonces T1 y T4 son auxiliares
Si j es punto auxiliar
        entonces T1 y T2 son auxiliares
Si k es punto auxiliar
```

```
6           entonces T2 y T3 son auxiliares
7   Si l es punto auxiliar
8           entonces T3 y T4 son auxiliares
```

Finalmente, si el punto se ubica en el lado jk (ver la Fig.A.2(d)) resulta

```
                        Sobre lado jk
1   Si i es punto auxiliar
2           entonces T1 y T2 son auxiliares
3   Si j es punto auxiliar
4           entonces T1 y T4 son auxiliares
5   Si k es punto auxiliar
6           entonces T2 y T3 son auxiliares
7   Si l es punto auxiliar
8           entonces T3 y T4 son auxiliares
```

De los cuatro análisis se desprende que se puede implementar en forma computacional como dos casos posibles: si el punto se encuentra en el interior o si el mismo se encuentra sobre uno de los lados.

Apéndice B

Implementación de regiones y subdivisiones

En la actualidad, se requiere que los componentes de software para aplicaciones de robótica sean diseñados e implementados de forma tal que puedan reutilizarse y mantenerse. La *tecnología orientada a objetos* es la forma más conveniente de cumplir con estos requisitos [9].

Las regiones y las subdivisiones de estas regiones requieren una representación abstracta. Para esto se emplean *objetos* dentro de lo que se conoce como *programación orientada a objetos*. Los *objetos* permiten definir estas abstracciones y las operaciones que se quieran realizar sobre ellos. Los *objetos* pueden pensarse como cápsulas dentro de las cuales se encuentran los *estados* que son accedidos y manipulados mediante funciones [47]. El estilo de programación que emplea este nivel de abstracción es la *programación orientada a objetos* que se encuentra soportada dentro del lenguaje C++ mediante *clases* [42, 40].

En definitiva, la clase que representan las regiones locales cuentan con

los siguientes elementos:

- Grilla: para esto se utilizan dos vectores `P_{x}` y `P_{y}`,
- Grafo local: utilizando un puntero al nodo raíz `G` del gráfo acíclico dirigido,
- Matriz de costos: mediante un puntero `ptrCostos` a un arreglo de dos dimensiones,
- Vector tridimensional de caminos: punteros `caminos` a las subdivisiones que forman un camino,
- Vector de subdivisiones en el borde de la región: `vectorB`,
- Vector de dos dimensiones `vectorRegionAdy` que indica cuáles son las regiones adyacentes,
- Índice de subdivisión en región adyacente `indiceAdy`: indica cuál es la subdivisión adyacente dentro de la región adyacente.

Una implementación posible en C++ de los elementos enumerados arriba puede verse en la Tabla B.1. El este caso, la grilla queda representada por dos vectores cuyos tipos numéricos son tipo `double`, que tienen una precisión de 64 bits y ofrece al menos 10 dígitos significativos. Esta elección se basa en que en general, un tipo numérico de menor precisión —como sería el tipo `float` de 32 bits— no cuenta con la precisión suficiente que se requiere para realizar la mayoría de los cálculos [26]. Las limitaciones en la precisión se hacen evidentes al momento de determinar mediante cálculos en las Ecs. 2.10 y 2.11 si un punto se encuentra sobre el borde de una región. En este caso, una baja precisión podría llegar a indicar que un punto se encuentra fuera de la región o en el interior de la misma cuando en realidad se halla sobre los lados de ella.

En la Tabla B.1 se observa que la grilla resulta implementada mediante

Tabla B.1: Implementación computacional en C++ de los nodos que representan las regiones.

Grilla x	`vector<double> *Px;`
Grilla y	`vector<double> *Py;`
Raíz del grafo acíclico dirigido	`DAG G;`
Matriz de costos	`double **ptrCostos;`
Vector tridimensional de caminos	`vector<vector<vector< Triangulo*>>> *caminos;`
Vector subdivisiones borde	`vector<Triangulo*> *vectorB;`
Vector de dos dimensiones regiones adyacentes	`vector<vector< region*>> *vectorRegionAdy;`
Índice de subdivisión en región adyacente	`vector<vector<unsigned>> *indiceAdy;`

dos vectores de punteros a números de tipo `double`, el grafo acíclico dirigido local se representa mediante una clase propia llamada `DAG` que consiste en un puntero a un nodo y este nodo es la raíz del grafo, la matriz de costos se implementa mediante un arreglo de dos dimensiones que se encuentra apuntado mediante `ptrCostos`, los caminos mediante un vector de tres dimensiones y apuntado mediante `caminos`, las subdivisiones en los bordes de la región están apuntadas mediante un vector de punteros `vectorB` y las subdivisiones adyacentes cuentan con punteros en un vector de dos dimensiones `vectorRegionAdy` e índices `indiceAdy` de tipo `unsigned` en un vector de dos dimensiones también.

Las subdivisiones que se encuentran dentro de las regiones también

Tabla B.2: Implementación computacional en C++ de los nodos que representan subdivisiones de las regiones.

Vértices	`double *p1x; double *p1y;`
	`double *p2x; double *p2y;`
	`double *p3x; double *p3y;`

Descendientes	`Triangulo *ch1;`
	`Triangulo *ch2;`
	`Triangulo *ch3;`

Subdivisiones adyacentes	`Triangulo *op_i;`
	`Triangulo *op_j;`
	`Triangulo *op_k;`

Obstáculo (Dijkstra)	`int obstaculo;`

Altura vértices (Dijkstra)	`double *alti;`
	`double *altj;`
	`double *altk;`

requieren una representación abstracta mediante una clase. Para esto, se definen los siguientes elementos dentro de la clase:

- Vértices: los vértices que definen la subdivisión `p1x, p1y, p2x, p2y, p3x, p3y`,
- Punteros a los descendientes por la triangulación: `ch1, ch2, ch3`,
- Punteros a las subdivisiones adyacentes: `op_i, op_j, op_k`,
- Una variable que indica si la subdivisión o nodo representa un obstáculo: `obstaculo`,
- La altura de los vértices de la subdivisión: `alti`.

En la Tabla B.2 se observa que los vértices de la subdivisión se encuentran implementados mediante seis punteros a números de tipo `double`, los descendientes de la triangulación de Delaunay son tres punteros a nodos de tipo `Triangulo`, las subdivisiones o nodos adyacentes mediante tres

punteros a nodos mediante `op_i`, `op_j` y `op_k`, si la subdivisión es un obstáculo se implementa mediante un número entero `obstaculo` y la altura de los vértices mediante tres punteros a números de tipo `double` y son `alti`, `altj` y `altk`.

Apéndice C

Puntos trayectoria

La «tira de triángulos», que es la respuesta que arroja el módulo de optimización, debe ser procesada para poder pasársela al módulo de control. De los triángulos del camino se tiene que tomar la mitad de los lados adyacentes para que la trayectoria suavizada caiga dentro de este camino.

La única salvedad es la subdivisión final de cada uno de los tramos (cada tramo se encuentra dentro de una región). De esta última, no se puede tomar la mitad del lado porque no es posible saber en forma inmediata cuál es la subdivisión adyacente. A los fines prácticos, se toma el baricentro de la última subdivisión, siendo el baricentro el centro de masa del triángulo que representa la subdivisión [51]. Se observa en la Fig. C.1 que el baricentro resulta conveniente ya que independientemente de por qué lado se ingrese a la región, la trayectoria que se genera se halla dentro de la subdivisión, que es el objetivo que se persigue. En la Fig. C.2 se observa el caso presentado en la Fig. 4.4 pero habiendo modificado la posición de los asteriscos de la última división de cada uno de los trechos (excepto del trecho que llega al objetivo χ). En esta figura el origen γ y el destino χ están unidos por una

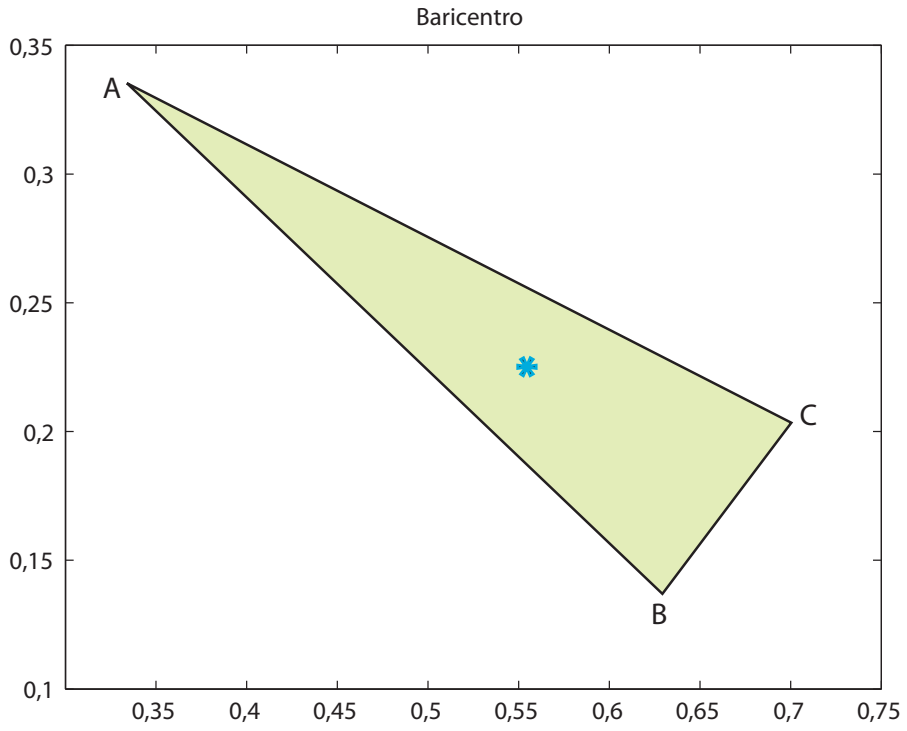


Figura C.1: Baricentro o centro de masa del triángulo ABC. Los ejes representan la distancia en metros.

serie de puntos, representados por asteriscos. Se toma el punto medio de todos los lados de los triángulos que forman la secuencia de subdivisiones salvo de las dos subdivisiones últimas de cada uno de los trechos, de las cuales se toma el baricentro.

Desde el punto de vista computacional se calculan los puntos medios de la siguiente manera: si el opuesto al vértice de una subdivisión que forma parte del camino es el triángulo siguiente en la lista de triángulos, se toma el punto medio de los restantes dos vértices de la subdivisión actual. Sino, se hace la misma pregunta con los otros dos vértices hasta detectarlo. Por lo tanto, esto se puede calcular en tiempo constante u $O(1)$.

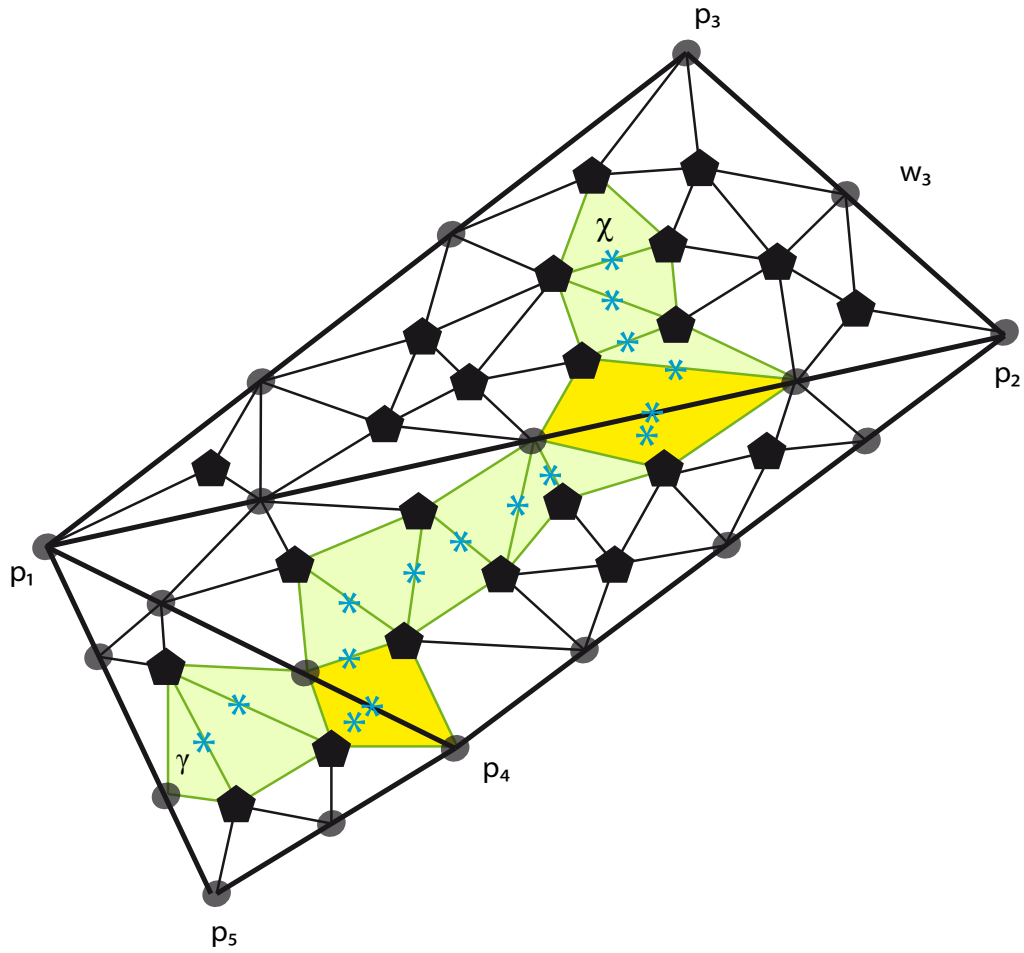


Figura C.2: Tres regiones triangulares y sus subdivisiones interiores. El origen γ y el destino χ son unidos por una secuencia de puntos que se representan mediante asteriscos, que son los puntos medios de los lados de la «tira de triángulos» a excepción de las subdivisiones finales de cada uno de los trechos de las cuales se toman los baricentros.

Bibliografía

[1] Bailey, T. and H. Durrant-Whyte: *Simultaneous localization and mapping (SLAM): part II*. Robotics & Automation Magazine, IEEE, 13(3):108–117, September 2006, ISSN 1070-9932.

[2] Bhattacharya, P. and M.L. Gavrilova: *Roadmap–based path planning - using the Voronoi diagram for a clearance–based shortest path*. Robotics & Automation Magazine, IEEE, 15(2):58–66, June 2008, ISSN 1070-9932.

[3] Billingsley, J., D. Oetomo, and J. Reid: *Agricultural robotics [tc spotlight]*. Robotics Automation Magazine, IEEE, 16(4):16 –16, 19, December 2009, ISSN 1070-9932.

[4] Birk, A., N. Vaskevicius, K. Pathak, S. Schwertfeger, J. Poppinga, and H. Buelow: *3-d perception and modeling*. Robotics & Automation Magazine, IEEE, 16(4):53–60, December 2009, ISSN 1070-9932.

[5] Bonneau, G.P. and S. Hahmann: *Smooth polylines on polygon meshes*. In Brunnett, G., B. Hamann, H. Müller, and L. Linsen (eds.): *Geometric Modeling for Scientific Visualization*, pp. 69–84. Springer–Verlag GmbH Berlin Heidelberg, Germany, 2004, ISBN 3-540-40116-4.

[6] Boyer, F.: *Optimisation de trajectorire pour la conception automobile*. Dans *7ème Congrès des doctorants EDSYS 2006*, Toulouse, France, mai 2006.

[7] Boyer, F. and F. Lamiraux: *Trajectory deformation applied to kinodynamic motion planning for a realistic car model*. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pp. 487–492, Orlando, FL, United States of America, May 2006.

[8] Broy, M., M. Gleirscher, S. Merenda, D. Wild, P. Kluge, and W. Krenzer: *Toward a holistic and standardized automotive architecture description*. Computer, 42(12):98–101, December 2009, ISSN 0018-9162.

[9] Brugali, D. and P. Scandurra: *Component–based robotic engineering (part I) [tutorial]*. Robotics & Automation Magazine, IEEE, 16(4):84–96, December 2009, ISSN 1070-9932.

[10] Brugali, D. and A. Shakhimardanov: *Component–based robotic engineering (part II)*. Robotics & Automation Magazine, IEEE, 17(1):100–112, March 2010, ISSN 1070-9932.

[11] Casalino, G., A. Turetta, and E. Simetti: *A three-layered architecture for real time path planning and obstacle avoidance for surveillance USVs operating in harbour fields*. In *OCEANS 2009-EUROPE, 2009. OCEANS '09*, pp. 1–8, Bremen, Germany, May 2009.

[12] Castellanos, J.A., J. Neira, and J.D. Tardós: *Map building and SLAM algorithms*. In Ge, S.S. and F.L. Lewis (eds.): *Autonomous Mobile Robots: Sensing, Control, Decision–Making, and Applications*, pp. 335–371. CRC Press, Boca Ratón, FL, United States of America, 2006, ISBN 0-8493-3748-8.

[13] Cormen, T.H., C.E. Leiserson, R.L. Rivest, and C. Stein: *Introduction to Algorithms*. The MIT Press, United States of America, third ed., September 2009, ISBN 0262033844.

[14] Craciunas, S.S., C.M. Kirsch, H. Payer, A. Sokolova, H. Stadler, and R. Staudinger: *A compacting real–time memory management system*. In *ATC'08: 2008 USENIX Annual Technical Conference*, pp. 349–362, Berkeley, CA, USA, 2008. USENIX Association.

[15] de Berg, M., O. Cheong, M. van Kreveld, and M. Overmars: *Computational geometry: algorithms and applications*. Springer–Verlag GmbH Berlin Heidelberg, Germany, third ed., 2008, ISBN 978-3-540-77973-5.

[16] Delsart, V., T. Fraichard, and L. Martinez: *Real–time trajectory generation for car–like vehicles navigating dynamic environments*. In *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, pp. 3401–3406, May 2009.

[17] Dijkstra, E.W.: *A note on two problems in connexion with graphs*. Numerische Mathematik, 1:269–271, 1959, ISSN 0029599X.

[18] Durrant-Whyte, H. and T. Bailey: *Simultaneous localization and mapping: part I*. Robotics & Automation Magazine, IEEE, 13(2):99–110, June 2006, ISSN 1070-9932.

[19] Guivant, J., E. Nebot, and J. Nieto: *Navigation and mapping in large unstructured environments*. The International Journal of Robotics Research, 23(4-5):449–472, 2004.

[20] Jordán, M.A., J. Bustamante, and J.M. Pinna Cortiñas: *Design of adaptive control systems for ROVs using inverse dynamics and state/disturbance observation*. In *6th Argentine Symposium on Computing Technology*, Rosario, Argentina, September 2005.

[21] Laumond, J.P.: *Robot motion planning and control*. Tech. Rep. 97438, Laboratoire d'Analye et d'Architecture des Systèmes du Centre National de la Recherche Scientifique, Toulouse, France, 1998. Chapter 3.

[22] LaValle, S.M.: *Planning Algorithms*. Cambridge University Press, Cambridge, U.K., 2006, ISBN 0-521-86205-1. Available at `http://planning.cs.uiuc.edu/`.

[23] Lefebvre, O.: *Navigation autonome sans collision pour robots mobiles nonholonomes*. Thèse de doctorat, Laboratoire d'Analyse et d'Architecture des Systèmes du Centre National de la Recherche Scientifique, Toulouse, France, juillet 2006.

[24] Lenain, R., B. Thuilot, C. Cariou, and P. Martinet: *Mobile robot control in presence of sliding: Application to agricultural vehicle path tracking*. In *Decision and Control, 2006 45th IEEE Conference on*, pp. 6004–6009, San Diego, CA, United States of America, December 2006.

[25] Lewis, B. and D.J. Berg: *PThreads Primer–A Guide to Multithreaded Programming*. SunSoft Press, A Prentice Hall Title, Mountain View, California, United States of America, 1996, ISBN 0-13-443698-9.

[26] Lippman, S.B., J. Lajoie, and B.E. Moo: *C++ Primer*. Addison–Wesley Professional, Stoughton, Massachusetts, United States of America, fourth ed., 2005, ISBN 0-201-72148-1.

[27] Moreyra, M., J. M. Pinna Cortiñas y F. Masson: *Aplicación de procesos gaussianos a la predicción de distancia utilizando imágenes monoculares*. En *Reunión de Trabajo en Procesamiento de la Información y Control*, Rosario, Argentina, septiembre 2009.

[28] Morris, A. and S. Morris: *TopoFusion Pro: GPS mapping software for Windows*. Web, 2002–2009. Version 3.90, `http://www.topofusion.com/`.

[29] Nieto, J., J. Guivant, and E. Nebot: *DenseSLAM: Simultaneous Localization and Dense Mapping*. The International Journal of Robotics Research, 25(8):711–744, 2006, ISSN 0278-3649.

[30] Oksanen, T.: *Path planning algorithms for agricultural field machines*. PhD thesis, Helsinki University of Technology, Department of Automation and Systems Technology, Automation Technology Laboratory, Helsinki, December 2007, ISBN 978-951-22-9080-2.

[31] Oksanen, T. and A. Visala: *Coverage path planning algorithms for agricultural field machines.* J. Field Robot., 26(8):651–668, 2009.

[32] Payer, H.E.: *A compacting real-time memory management system*. Master's thesis, University of Salzburg, Salzburg, Austria, 2007.

[33] Pinna, J.M., F.R. Masson, and J. Guivant: *Motion planning strategies for non–holonomic vehicles*. In *XII Reunión de Trabajo en Procesamiento de la Información y Control*, Río Gallegos, Argentina, October 2007.

[34] Pinna Cortiñas, J. M., R. Coppo y C. Delrieux: *Controladores PID: programación orientada a objetos*. En *XX^{o} Congreso Argentino de Control Automático — Sección Estudiantil*, Buenos Aires, Argentina, agosto 2006.

[35] Pinna Cortiñas, J. M., M. A. Jordán y J. Bustamante: *Estudio de perturbaciones en vehículos subacuáticos con cable*. En *XX^{o} Congreso Argentino de Control Automático — Sección Estudiantil*, Buenos Aires, Argentina, agosto 2006.

[36] Pinna Cortiñas, J. M. y F. Masson: *Control de vehículos terrestres mediante la integración de planificación y SLAM*. En *VI Jornadas Argentinas de Robótica*, Buenos Aires, Argentina, noviembre 2010.

[37] Pinna Cortiñas, J. M., F. Masson y O. Agamennoni: *Modelo cinemático de un vehículo autónomo*. En *Jornadas Argentinas de Robótica*, Córdoba, Argentina, noviembre 2006.

[38] Pinna Cortiñas, J. M., F. R. Masson y J. Guivant: *Planeamiento de trayectoria en mapas extensos mediante programación dinámica de dos niveles*. En *Actas de las V Jornadas Argentinas de Robótica 2008*, Bahía Blanca, Argentina, noviembre 2008.

[39] Pinna Cortiñas, J. M. y J. L. Moiola: *Técnicas de sintonizado para controladores PID*. En *III Congreso Internacional de Matemática Aplicada a la Ingeniería y Enseñanza de la Matemática en Ingeniería (INMAT 2005)*, Buenos Aires, Argentina, octubre 2005.

[40] Prata, S.: *C++ Primer Plus*. Sams Publishing, Indianapolis, Indiana, United States of America, fifth ed., 2005, ISBN 0-672-32697-3.

[41] Roberts, J.M., E.S. Duff, P.I. Corke, P. Sikka, G.J. Winstanley, and J. Cunningham: *Autonomous control of underground mining vehicles using reactive navigation*. In *Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on*, vol. 4, pp. 3790–3795 vol.4, San Francisco, CA, United States of America, 2000.

[42] Sedgewick, R.: *Algorithms in C++, Parts 1–4: Fundamentals, Data Structure, Sorting, Searching*. Addison–Wesley Professional, Stoughton, Massachusetts, United States of America, third ed., 2008, ISBN 0-201-35088-2.

[43] Siciliano, B. and O. Khatib (eds.): *Springer Handbook of Robotics*. Springer–Verlag GmbH Berlin Heidelberg, Germany, 2008, ISBN 978-3-540-23957-4. Chapter 34.

[44] Slaughter, D.C., D.K. Giles, and D. Downey: *Autonomous robotic weed control systems: A review*. Computers and Electronics in Agriculture, 61(1):63–78, April 2008, ISSN 0168-1699.

[45] Stentz, A.: *Optimal and efficient path planning for partially–known environments*. In *Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on*, vol. 4, pp. 3310–3317, May 1994.

[46] Stentz, A.: *The focussed d* algorithm for real–time replanning*. In *In Proceedings of the International Joint Conference on Artificial Intelligence*, pp. 1652–1659, 1995.

[47] Taylor, R.N., N. Medvidović, and E.M. Dashofy: *Software Architecture: Foundations, Theory, and Practice*. John Wiley & Sons, Inc., United States of America, 2010, ISBN 978-0470-16774-8.

[48] Weisstein, E.W.: *Circumcircle*. From MathWorld–A Wolfram Web Resource. http://mathworld.wolfram.com/Circumcircle.html.

[49] Weisstein, E.W.: *Euler integral*. From MathWorld–A Wolfram Web Resource. http://mathworld.wolfram.com/EulerIntegral.html.

[50] Weisstein, E.W.: *Moving average*. From MathWorld–A Wolfram Web Resource. http://mathworld.wolfram.com/MovingAverage.html.

[51] Weisstein, E.W.: *Triangle centroid*. From MathWorld–A Wolfram Web Resource. http://mathworld.wolfram.com/TriangleCentroid.html.

[52] Weisstein, E.W.: *Triangle interior*. From MathWorld–A Wolfram Web Resource. http://mathworld.wolfram.com/TriangleInterior.html.

[53] Wright, M.: *The complete auto repair glossary–car repair terms defined for you*. http://autorepair.about.com/library/glossary/bldef-667.htm, 2002.

[54] Yamaha: *Gama ATV*. Barcelona, España, 2009. Especificaciones técnicas, catálogo, modelo Grizzly 125,.

[55] Yoshida, K.: *Achievements in space robotics*. Robotics & Automation Magazine, IEEE, 16(4):20–28, December 2009, ISSN 1070-9932.