Universidad Nacional Autónoma de México

Facultad de Ingeniería

Robots Móviles Colaborativos para el Transporte Autónomo de Objetos Vía Visión

T E S I S

QUE PARA OBTENER EL TÍTULO DE:
INGENIERO MECATRÓNICO

PRESENTA:
Eduardo Daniel Ruiz Libreros

DIRECTOR DE TESIS:
Dr.Víctor Javier González Villela

2013

Dedicatorias

A mi madre por todo su amor, confianza y consejo.
Por ti he llegado hasta donde estoy.

A mi novia por haberme brindado su cariño y
compresión a lo largo de este proceso.

A mis amigos por todos los momentos vividos dentro
y fuera de las aulas.

Agradecimientos

A la Universidad Nacional Autónoma de México por
las excelentes oportunidades brindadas para mi formación.

A los profesores de la Facultad de Ingeniería
por todo el conocimiento transmitido.

Al Dr. Víctor Javier González Villela por su asesoría y
el apoyo brindado a lo largo de este proyecto.

A la DGAPA por el apoyo brindado para la realización de este trabajo, a través del proyecto PAPIIT IN115811 con Título: "Investigación y desarrollo en sistemas mecatrónicos: robótica móvil, robótica paralela, robótica híbrida y teleoperación".

Resumen

En este trabajo se presenta el desarrollo de un sistema colaborativo de transporte de objetos a partir de dos robots móviles. Esta tarea de transporte se analiza en dos etapas: antes y después de tomar el objeto a transportar. En la primera parte se utiliza el control por campos potenciales con la aproximación de González Villela V. La segunda parte se resuelve utilizando modificaciones al control por campos potenciales, estas permiten planear el transporte de cierto objeto considerando su orientación y la configuración del entorno de trabajo. También se diseña un algoritmo de coordinación que permite a los robots navegar hacia el punto para enganchar al objeto que se encuentre más cercano y libre, evitando interferir entre ellos.
La adquisición de datos se realiza mediante raecTIVision, sistema de visión artificial basado en el reconocimiento de símbolos. Usando el mencionado sistema se obtiene información sobre la postura de los robots móviles, objeto a transportar y posibles obstáculos a evadir. Utilizando SIMULINK se procesa está información y planean las trayectorias de cada robot para más tarde, enviar inalámbricamente la velocidad necesaria de cada motor de los robots para seguir esa trayectoria. Estos datos de velocidad son procesados en cada robot mediante un microcontrolador ATmega328 (Arduino UNO). Una vez identificados, los valores son enviados a los motores utilizando una tarjeta controladora MD25, que recibe información vía I2C desde el microcontrolador. La solución de comunicación inalámbrica utilizada consiste en módulos XBee Series 2 de Digi.
Finalmente para observar el comportamiento del sistema se realizan las simulaciones y experimentos pertinentes.

Índice general

Índice de figuras

Capítulo 1

Introducción

1.1. Apectos Generales

En 1921 el dramaturgo Karel Capek dió origen al término robot, al introducir en su obra R.U.R. (Rossum Universal Robots) la palabra *robota* que en checo y más idiomas eslavos significa trabajo o tarea. Capek se refería con este término a seres artificiales creados para realizar ciertas labores e incluso capaces de tener sentimientos como los seres humanos.
Hoy día la palabra robot hace referencia a un dispositivo o sistema capaz de realizar tareas de forma similar a los seres humanos e incluso de forma autónoma. Los robots poseen características o capacidades mecánicas, electrónicas,de cómputo y de control, en otras palabras mecatrónicas, que van de acuerdo a las tareas que necesitan desempeñar. Podemos encontrar diferentes configuraciones como los robots manipuladores empleados en la industria, los robots humanoides ,los robots móviles e incluso robots híbridos.
La robótica móvil es la rama que estudia todos aquellos robots que son capaces de desplazarse en el medio, pudiendo ser dentro del agua, de forma aérea o por tierra. Algunas de las principales aplicaciones de la robótica móvil son: labores de rescate, exploración y búsqueda, manipulación de residuos peligrosos, transporte, vigilancia, entre otras. En estas tareas es necesario contar con un sistema robusto y de precisión, y contemplar factores importantes como confiabilidad, velocidad,costos del sistema, etc. Es ahí cuando los sistemas integrados por múltiples robots surgen como una posible alternativa.
De forma general, un sistema multi-robot (SMR)esta formado por varios robots que permiten realizar de manera conjunta una tarea específica. Estos sistemas pueden estar integrados por robots con distintas capacidades: manipuladores, humanoides, vehículos, etc. El tema de los sistemas muti-robot ha sido estudiado extensamente durante las últimas dos décadas. El objetivo de diseño de estos sistemas es permitir que un grupo de robots resuelva tareas de una mejor forma que un solo robot.
De acuerdo con [1] los principales causas que han permitido el estudio de estos sistemas han sido:

- El diseño de *hardware* sofisticado capaz de realizar tareas demandantes
- El desarrollo de sensores y sistemas de percepción que proveen al robot de conocimiento detallado sobre su entorno.
- El desarrollo de sistemas de comunicación robustos y confiables que permiten a los robots compartir la información.
- El diseño de software de control inteligente que permite a los robots lograr resultados globales coherentes basados en acciones de control locales.

La robótica colaborativa estudia las ventajas que representa tener equipos formados por múltiples robots para la realización de tareas y como mejorar el desempeño en la realización de estas.

1.2. Trabajo Previo

Desde las primeras investigaciones hechas en el área de la róbotica móvil el campo de estudio ha cambiado y ha crecido de forma importante. En el campo de la robótica móvil colaborativa algunos temas han sido explorados ámpliamente y de acuerdo con [2] los principales áreas de investigación son:

- Inspiraciones Biológicas
- Comunicaciones
- Arquitecturas, asignación de tareas y control
- Localización, mapeo y exploración
- Manipulación y transporte de objetos
- Coordinación de movimientos
- Robots reconfigurables

Este trabajo se centra en la manipulación y transporte de objetos, es decir el permitir que múltiples robots carguen, empujen o manipulen objetos comunes de forma colaborativa. Buena cantidad de investigaciones se han centrado en este problema y sólo algunas han hecho demostraciones físicamente. Existen algunas variaciones de esta tarea, algunos enfoques por ejemplo se centran en restricciones de movimiento, grupos pequeños de robots contra grandes enjambres, variaciones en el entorno, modelos centralizados contra modelos distribuidos, entre otras.
En la década de los 1980's la investigación en robótica colaborativa se volvió muy activa,

en [3] Fukuda y Nakagawa proponen una arquitectura jerárquica y descentralizada inspirada en la organización celular de algunos seres vivos llamada CEBOT (Cellular roBOTics system). Beni en [4] hace un análisis de la estructura de los sistemas celulares, resaltando las diferencias con los autómatas y las redes neuronales. En [5], Asama et al. proponen un sistema descentralizado de agentes heterogéneos capaces de comunicarse, asignar tareas y planear movimientos llamado ACTRESS (ACTor-based Robot and Equipments Sythetic System). Caloud et al. en [6] desarrollan una arquitectura orientada a resolver problemas como tolerancia ante contingencias, planeación y control de movimientos y planeación de tareas.

Dentro del área de Transporte y Manipulación se han hecho varios acercamientos desde diferentes enfoques; por ejemplo en [7], Parker se enfoca en tolerancia a fallas, aprendizaje y asignación de tareas. Por otro lado Donald et al. en [8], analiza dos protocolos diferentes para empujar cajas enfocándose en las comunicaciones y los requerimientos de *hardware*. La tarea de empujar objetos mediante múltiples robots es quizás de las tareas más estudiadas, por ejemplo [9] y [10]. Otra propuesta es cuando los robots cargan o agarran los objetos para llevarlos hasta su posición final como en [11], [12] y un variante con sogas en [13].

En el año 2002, Asama et al. en [14] muestran una propuesta que permite a los robots mover obstáculos que interfieran durante el transporte. Además de las respectivas simulaciones, implementan físicamente su sistema con dos robots navegando en un ambiente estático. También en ese año Kumar et al. en [15], hacen un acercamiento a la manipulación colaborativa basándose en el control de la formación de tres robots. En su propuesta los objetos a manipular son rodeados o "encerrados" por la formación de los robots.

Entre los acercamientos más importantes a la robótica móvil colaborativa más recientes está [16] y [17] , ambas propuestas hacen el transporte de cajas cargándolas con un par de robots. Los robots cuentan con una plataforma con rotación libre que les permite alinearse en todo momento para soportar mejor la caja. Ambos trabajos utilizan sensores ópticos, mientras que en [16] utilizan sensores infrarrojos, en [17] utilizan laser.

En 2008 Fink et al. en [18], proponen una sistema de transporte descentralizado en un ambiente con obstáculos, el transporte se realiza rodeando el objeto a transportar (Caging en ingles). El sistema descentralizado se basa en que los robots no comparten ninguna información, la tarea se realiza con la información local de cada robot y se confía en ella. Finalmente en 2011 en [19] proponen un modelo difuso de control para transportar objetos empujándolos (box pushing).Entre las ventajas del modelo es que se adapta a prácticamente cualquier número de robots que participen en el transporte.

1.3. Planteamiento del problema

Si bien se han citados diferentes acercamientos al tema de esta tesis, es importante señalar que la gran mayoría parte de los sistemas existentes implementan arquitecturas de alto costo, que representan una gran desventaja o contradicción a una de las bases de

la robótica colaborativa; realizar una tarea compleja con múltiples robots de arquitectura simple. La necesidad de contar con un sistema colaborativo de robots tiene grandes aplicaciones en la industria, ejemplo de ellas son el transporte de sustancias peligrosas, pesadas o muy grandes.
Tomando en cuenta que los sistemas multi-robot brindan redundancia y contribuyen a resolver las tareas de una forma más confiable, rápida e incluso de menor costo, es importante desarrollar algoritmos y modelos para el control sistemas multi robot, en específico del transporte colaborativo. En el ámbito local, existen importantes teorías en el campo de la robótica móvil como [20]; resulta interesante verificar la aplicación de estas teorías al ámbito del transporte colaborativo y desarrollar nuevos algoritmos en está importante área.

1.4. Objetivo

El objetivo de este trabajo es desarrollar un sistema de transporte de objetos con formas geométricas regulares, utilizando robots móviles colaborativos capaces de planear y seguir trayectorias,así como de evadir obstáculos. Se utilizará la teoría unificadora de Gonźalez Villela en [20] así como el desarrollo planteado en [21]. Se desarrollará una algoritmo capaz de coordinar y planear la trayectoria de dos robots móviles de forma que estos puedan: navegar hasta el objeto a transportar, orientarse correctamente para la toma o enganche de este objeto y finalmente transportarlo hasta una meta final.

1.5. Organización

Para el desarrollo de esta tesis, el trabajo se ha organizado en 7 capítulos. A continuación se hace una breve descripción de su contenido:

Capítulo 1: En este capítulo de tratan a manera de introducción aspectos generales, trabajos e investigaciones previas que se tomaron en cuenta para esta tesis así como los objetivos de la misma.

Capítulo 2: Aquí se trata más a fondo el tema de los robots colaborativos. Se plantean las arquitecturas de hardware, de comunicación, etc. Los principios de funcionamiento, mecanismos de interacción e inclusive se hacen algunas clasificaciones.

Capítulo 3: Se describe el algoritmo utilizado para coordinar a los robots y planear las trayectorias. También se explica el funcionamiento del control por campos potenciales empleado.

Capítulo 4: En este capítulo se presentan los modelos matemático que representan a los robots en cada una de las configuraciones empleadas, así como los modelos cinemáticos empleados para el control de los robots móviles.

Capítulo 5: Aquí se describe como funciona la simulación generada para comprobar el algoritmo de coordinación y navegación. Se detallan las consideraciones necesarias para el funcionamiento correcto del *software*. Además se describen los elementos del modelo funcional con el que se harán experimentos.

Capítulo 6: En esté capítulo se especifican las pruebas realizadas y los resultados arrojados por estas. Se presentan gráficas y datos útiles para la documentación de las pruebas tanto hechas en simulación como experimentales.

Capítulo 7: Finalmente, en este apartado se comentan las conclusiones generadas a partir de los resultados de las pruebas y simulaciones realizadas. Adicionalmente se sugieren trabajos a realizar en el futuro para complementar o mejorar el trabajo aquí realizado.

Capítulo 2

Sistema de Robots Colaborativos

2.1. Robots Colaborativos

De acuerdo con [22] las principales razones para el estudio de los sistemas multi-robot son:

- La complejidad de la tarea es muy elevada para un solo robot.
- La tarea es inherentemente distributiva.
- Construir varios robots con recursos limitados es más fácil que un solo robot muy complejo.
- Múltiples robots pueden resolver los problemas más rápido utilizando paralelismo.
- El introducir múltiples robots incrementa la robustez mediante la redundancia.

De una forma muy general general, [22] hace una aproximación que solo considera principalmente dos categorías: Enjambre (*swarm* en ingles) o sistema colectivo y sistemas intencionalmente colectivos. Los primeros están basados en robots que ejecutan sus tareas con la mínima necesidad de información sobre el resto de los miembros de sistema(también llamado comportamiento eusocial). En cambio, en los sistemas intencionalmente colectivos los robots se basan en el estado, las acciones y las capacidades de los demás miembros para ejecutar sus tareas(llamado comportamiento cooperativo).

Otra clasificación se basa en el mecanismo de interacción, el cual dará características y capacidades particulares al sistema. Según [1] los principales mecanismos de interacción en estos sistemas multi-robot son:

- Colectivo: Involucra robots simples que no están al tanto del resto de los integrantes del sistema, a pesar de compartir metas comunes. En estos sistemas las acciones individuales resultan en el beneficio del sistema.

- Cooperativo: Involucra robots que están "conscientes" de los demás integrantes. Las acciones individuales de los robots se traducen en un beneficio para el equipo.
- Colaborativo: Involucra robots también "conscientes"del resto y con objetivos individuales. Aunque sus objetivos no sean los mismos, están dispuestos a ayudar al resto en sus objetivos individuales.
- Coordinación: Involucra robots cuyo objetivo principal es minimizar la interferencia con los demás. En este caso en lugar tratar activamente de ayudar a otros, el robot trata a toda costa evitar interferir con el resto.

2.2. Taxonomía

Para aclarar las variantes que existen dentro los SMR se hará una breve clasificación de las arquitecturas, comunicación y localización existentes. De esta forma será más fácil comprender el tipo de sistema a desarrollar y sus características, limitaciones y aplicaciones.

Figura 2.1: Taxonomía

2.3. Arquitecturas

Se refiere a la forma que interactuan, funcionan y están organizados los distintos componentes del robot, tanto en *hardware* como *software*. De modo que podemos identificar tres distintas:

- Deliberativa: Aquí el robot tiene un conjunto de sensores y una representación del entorno. Con base en esta representación y a partir de la información de los sensores se toman decisiones.

- Reactiva: A partir de la información de los sensores mide o adquiere las características del entorno, de forma que no es necesario contar con una representación del entorno.
- Híbrida: Como su nombre lo indica, representa una combinación entre los casos anteriores. Combina la capacidad de respuesta de la arquitectura reactiva con la representación de la deliberativa.

Ahora bien, con base en las arquitecturas mencionadas los SMR presentan arquitecturas más complejas, que en un nivel más alto le permitirán a un conjunto de robots organizarse para desempeñar sus tareas. Estas dos arquitecturas son:

1. Centralizada:Existe un agente que tiene el control de los robots. En esta arquitectura se requiere que dicho agente tenga un conocimiento global del entorno.
2. Descentralizada: Aquí los robots perciben el entorno y toman sus propias decisiones sin necesidad de un control o agente central. Esta arquitectura presenta dos subtipos:
   a) Jerárquico: Los robots trabajan en diferentes niveles del problema.
   b) Distribuido: Los robots dividen la tarea en subproblemas. Diferentes robots trabajan en el mismo nivel de abstracción o complejidad de la tarea.

Cabe mencionar que no todos los sistemas multi robot son completamente centralizados o descentralizados.

2.4. Comunicación

Gracias a la comunicación los elementos del SMR pueden intercambiar información sobre el entorno, el estado de la tarea o información específica para algún robot. En este rubro la SMR se pueden clasificar como:

- Directa: Es cuando la comunicación es establecida intencionalmente de un elemento a otro. El intercambio de información se puede llevar a cabo por diferentes medios: Bluetooth, IEEE 802.11, infrarrojo, etc. y puede ser mediante *broadcast* (uno a muchos) o *unicast* (uno a uno).
- Indirecta: Se da implícitamente y ocurre como efecto de las acciones de los agentes sobre el entorno o área de trabajo. Este tipo de comunicación se puede observar en grupos o enjambres de insectos.

2.5. Localización

La localización se refiere a estimar la posición de un robot dentro de un área de trabajo. Ya sea ambientes internos o externos los robots necesitan conocer su posición para completar de manera satisfactoria su objetivo. Existen diferentes técnicas para realizar dicha medición, la mayoría se basan en la información proveniente de los sensores, por ejemplo la odometría. También puede ser que los sensores reconozcan marcas en el entorno que permitan hacer algún tipo de triangulación para conocer la posición del robot. Podemos clasificar la localización como:

- Local: Consiste en hacer un seguimiento del movimiento del robot a partir del conocimiento de la posición inicial del mismo. Una de las técnicas más empleadas es la odometría. Este tipo de localización genera una buena cantidad de errores y existen diferentes técnicas para corregir la información generada, dichas técnicas pueden ser el uso de marcas.
- Global: Este tipo de localización permite determinar la posición del robot sin tener la referencia de la posición inicial. En este caso se utilizan enfoques probabilísticos como teoría de Bayes, filtro de Kalman o cadenas de Markov.

Capítulo 3

Planeación de las trayectorias

El problema de transporte colaborativo mediante robots móviles plantea en general dos etapas principales. La primera de ellas contempla desde el inicio, en dónde cada robot se encuentra en su posición inicial y aleatoria hasta el momento en que llega a una meta (objeto a transportar). Una vez que se encuentra en la posición ideal para el transporte comienza la segunda etapa, en la que la tarea consiste en llevar el objeto a un punto final. Ahora bien, para llevar a cabo dichas tareas de forma correcta se necesita de un proceso planeación de tareas y trayectorias para cada robot. Este capítulo se enfoca en explicar como se lleva a cabo este proceso para realizar el transporte de manera colaborativa.
El sistema de transporte propuesto se basa en un planeador global encargado de planear las trayectorias a seguir por cada robot. El planeador global tiene la capacidad de conocer la postura de los robots móviles, del objeto a transportar y los posibles obstáculos en todo momento.

3.1. Navegación por campos potenciales

Para la navegación en el ambiente de trabajo se emplea la técnica de campos potenciales, en donde se consideran sumas de fuerzas de atracción(meta) y repulsión (obstáculos) en todo el espacio de trabajo. Este método se emplea con el enfoque visto en [20], en donde los campos potenciales artificiales son tomados como vectores que se encargan de guiar al robot hasta alcanzar una meta con la posibilidad de evadir los obstáculos que encuentre en su camino.
El problema de llegar a la meta se traduce entonces en encontrar la orientación y la velocidad que el robot debe tener para dirigirse a la meta.
Primero se define el vector de distancia a la meta $d_g = (x_f - x_i, y_f - y_i)$ cuya magnitud queda expresada como $d_g = \sqrt{(x_f - x_i)^2 + (y_f - y_i)^2}$ y la orientación como $\theta_g = \tan^{-1}(\frac{y_f - y_i}{xf_- x_i})$. También se define un ángulo de error como $\theta_e = \theta_g - \theta_i$.

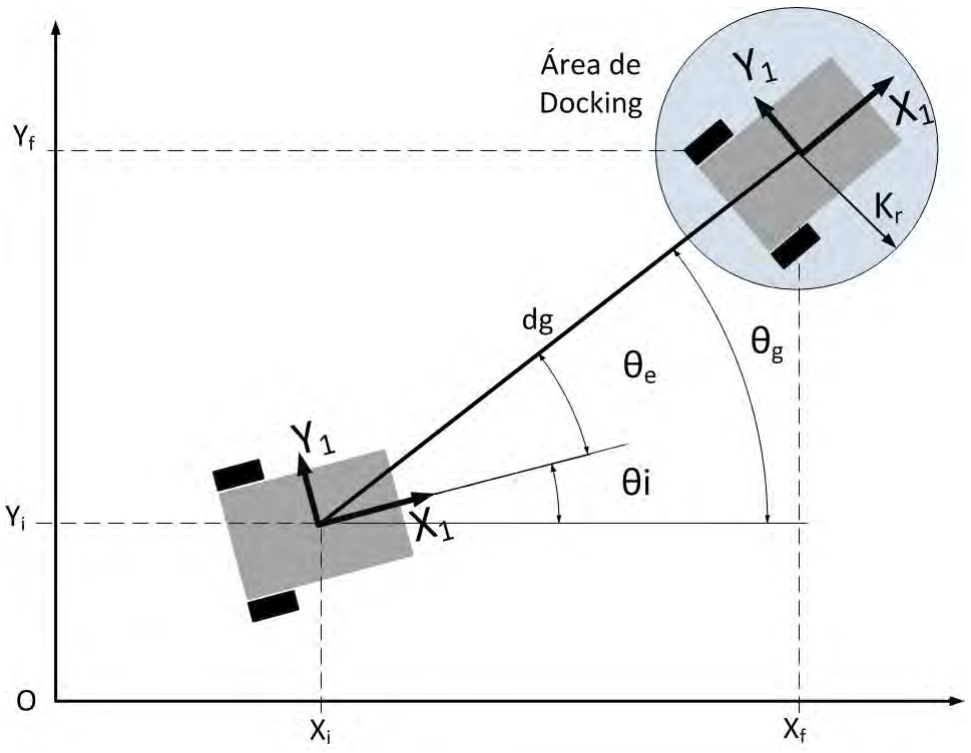


Figura 3.1: Campos potenciales Artificiales

El robot es llevado de su posición inicial hasta la meta bajo las siguientes reglas:

$$v = \begin{cases} v_{max} & \text{si } |d_g| > k_r \\ \frac{v_{max}}{k_r} d_g & \text{si } |d_g| \leq k_r \end{cases} \tag{3.1}$$

$$\omega = \omega_{max} \sin \theta_e \tag{3.2}$$

Donde ω_{max} y v_{max} son las velocidades máximas con las que se desplaza el robot y k_r es el radio del área de *docking*.
Lo que dice 3.1 es que el robot es guiado a la meta con su velocidad máxima cuando esta fuera del área de *docking*, cuando entre en esta se aproximará más lento. Por otro lado 4.6 dice que el robot corregirá su orientación hacia la meta en función al seno del ángulo de error θ_e.
Hasta ahora el robot sería capaz de navegar de una posición inicial hasta un punto meta, sin embargo falta tener una representación de los obstáculos para poder evadirlos. La representación empleada es tomada de [23], en donde el campo potencial repulsivo U_o se considera como un vector fuerza de repulsión y se hace uso de la distancia más corta al obstáculo ρ.
Para efectos de simplicidad los obstáculos son tomados como circulares con radio ρ_r por lo que la distancia ρ se traduce en la distancia al centro del obstáculo ρ_c menos el radio

del mismo ρ_r. La función de campo potencial que modela entonces a los obstáculos es:

$$U_o = \begin{cases} \frac{1}{2}\eta(\frac{1}{\rho_c - \rho_r} - \frac{1}{\rho_o}) & \text{si } \rho_c \leq \rho_o + \rho_r \\ 0 & \text{si } \rho_c > \rho_o + \rho_r \end{cases} \tag{3.3}$$

En donde η es una ganancia constante y ρ_o representa la distancia límite sobre la cual el campo potencial tiene efecto. Estas constantes se eligen con base en las velocidades v_{max} y ω_{max} que posibilitan al robot móvil evadir los obstáculos.
El sistema esta diseñado para que ambos robots naveguen hasta el punto de *docking* de manera simultanea, para evitar que colisionen entre ellos se utilizan igualmente los campos potenciales. Una diferencia importante es que uno de los robots solo evade obstáculos y trata de llegar a la meta, mientras que el segundo robot además de evadir los obstáculos tiene la capacidad de evadir a su compañero. Esto se puede hacer gracias al planeador global mencionada anteriormente, este hace posible conocer en todo momento la ubicación de los robots. El segundo robot si tiene acceso a la ubicación de su compañero, esto le posibilita evadirlo en caso de que sus trayectorias se crucen o interfieran. Este modelado se asimila al hecho por campos potenciales, pues de cierta forma el robot a evadir es un obstáculo que varía su posición en el entorno de trabajo.
No obstante se puede implementar otro forma de coordinación en la que un robot se mueve primero hasta llegar a su destino, al encontrarse posicionado envía una señal al segundo robot para que este último pueda comenzar su trayecto.

3.2. Aproximación al objeto

Un aspecto importante es la aproximación al objeto a transportar, pues no basta con llegar a cualquier punto cercano al objeto, se deben definir ciertos puntos sobre el objeto en donde los robots podrán engancharse o sujetarse y así realizar el transporte. Además, el enfoque empleado para la navegación y planeación de trayectorias (campos potenciales) no permite modelar la meta como un cuerpo sino como puntos o partículas, por lo que se complica la aproximación a un punto sobre el objeto a mover y al mismo tiempo evitar chocar con secciones del mismo.
La solución empleada es generar puntos alrededor del objeto a mover (predocking), que sirvan como guía a los robots tanto para aproximarse y orientarse correctamente como para evitar chocar con partes del objeto. Una posible forma de verlo es generar un mapa o grafo alrededor del objeto a transportar, cada nodo representa un punto de *pre-docking*. Estos puntos, dependiendo de su ubicación, serán para orientar o para guiar al robot alrededor del objeto. El número y ubicación de los nodos dependerá de la geometría del objeto a transportar.
Se presenta el caso de un paralelogramo de dimensiones l por a cuyos puntos de sujeción o *docking* son (x_{t1}, y_{t1}) y (x_{t2}, y_{t2}).

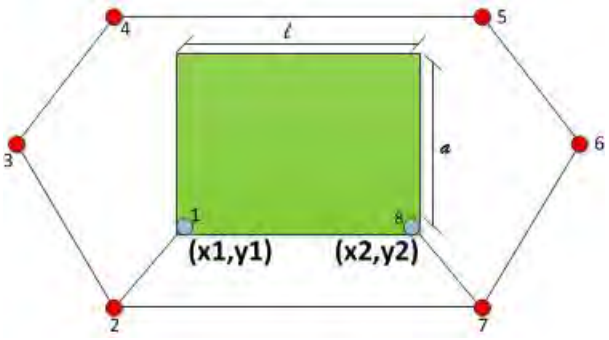


Figura 3.2: Puntos de *predocking*

Se observan 6 nodos alrededor del objeto además de los puntos de *docking*.También se puede observar la conexión entre los nodos, estas indican las posibles trayectorias a seguir desde cada uno de ellos. La trayectoria a seguir una vez alcanzado alguno de estos nodos se genera con el siguiente algoritmo.

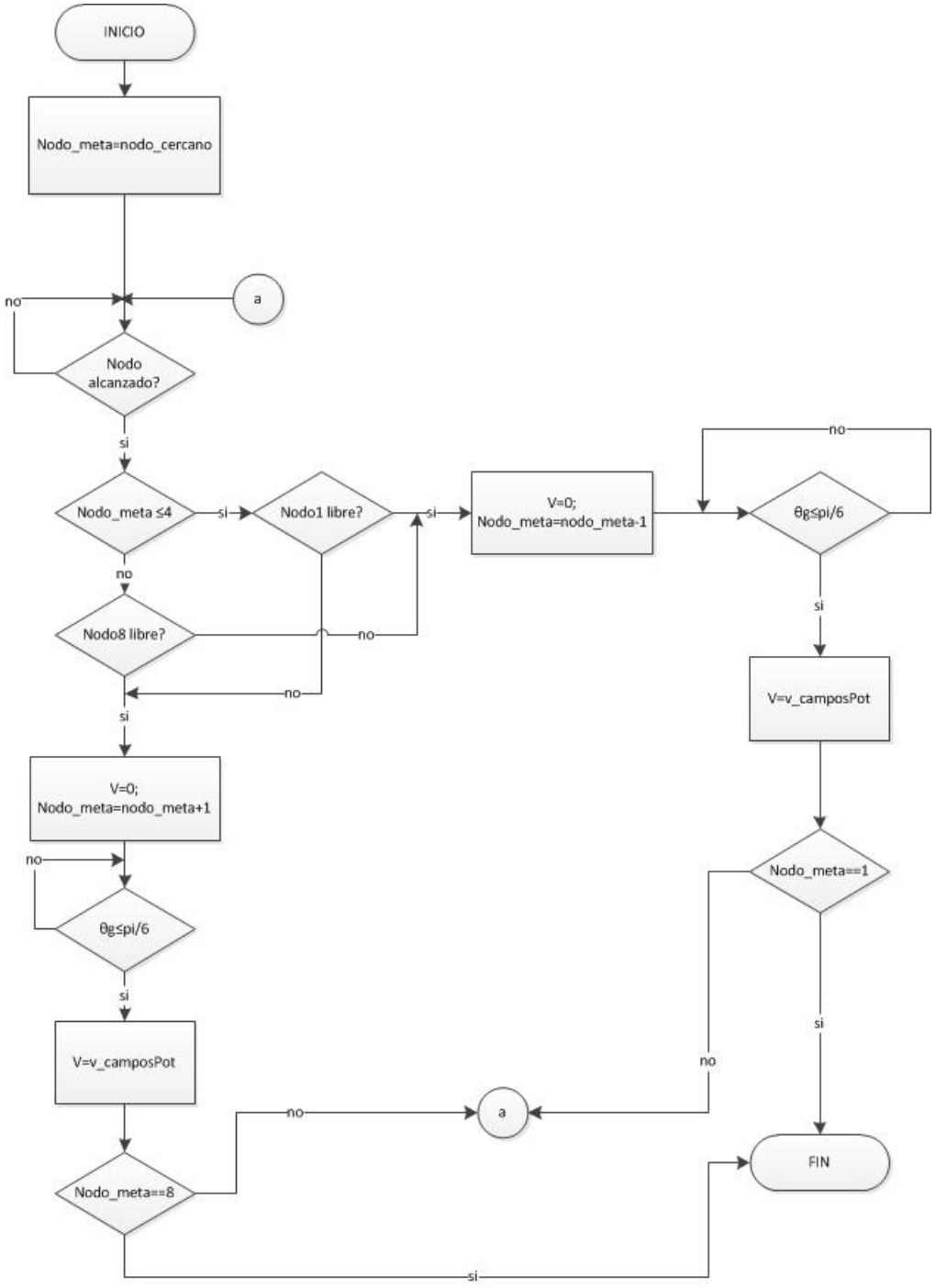


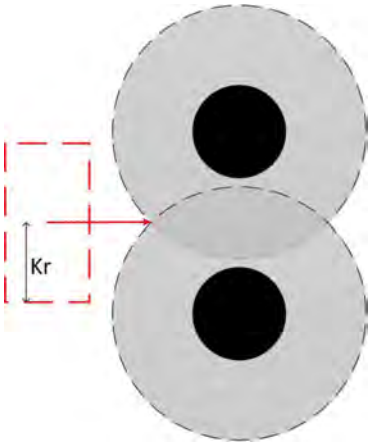
Figura 3.3: Algoritmo para navegación *predocking*

Cabe destacar que las velocidades calculadas se generan con el método de campos potenciales, con la única condición de que el robot primero corrige su orientación antes de avanzar al siguiente nodo. Esto se debe a que el espacio de trabajo es reducido y las

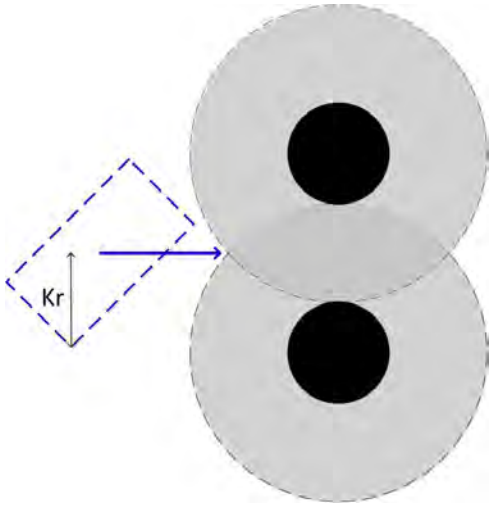
velocidades de desplazamiento del robot no permitirían un movimiento más natural.

3.3. Transporte evadiendo obstáculos

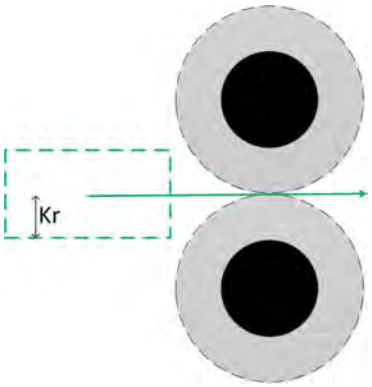
Una vez que los robots han llegado a la meta y se han acoplado al objeto a transportar se inicia la segunda etapa del problema; encontrar la trayectoria a seguir para llevar dicho objeto hasta el punto deseado. Para esta etapa también se hace uso de la aproximación por campos potenciales descrita anteriormente con algunas modificaciones.
La modificación más importante es la representación de los obstáculos, ya que el área sobre la cuál su campo repulsivo tendrá efecto cambia de acuerdo a la orientación del objeto a transportar y su proximidad al obstáculo.
Para el caso del paralelogramo estudiado hasta el momento se contempla que dependiendo de su orientación se tienen dimensiones diferentes. Es decir el radio sobre el que influye el campo potencial dependerá, como se muestra en la siguiente figura, de la orientación con la que se aproxime el objeto a la zona:



(a)



(b)



(c)

Figura 3.4: Orientación en el transporte

Recordando que el control por campos potenciales trata al robot como una partícula

transfiriendo las dimensiones a los obstáculos. La forma de calcular el radio de influencia del obstáculo es la siguiente:

1. Se traza una recta hacia el centro del obstáculo.
2. Se encuentra la intersección de esa recta con el contorno del obstáculo.
3. La distancia del centro del robot a la intersección es el valor del radio de influencia del obstáculo

En la siguiente figura se ilustra un ejemplo de distintos radios de influencia para obstáculos de iguales dimensiones.

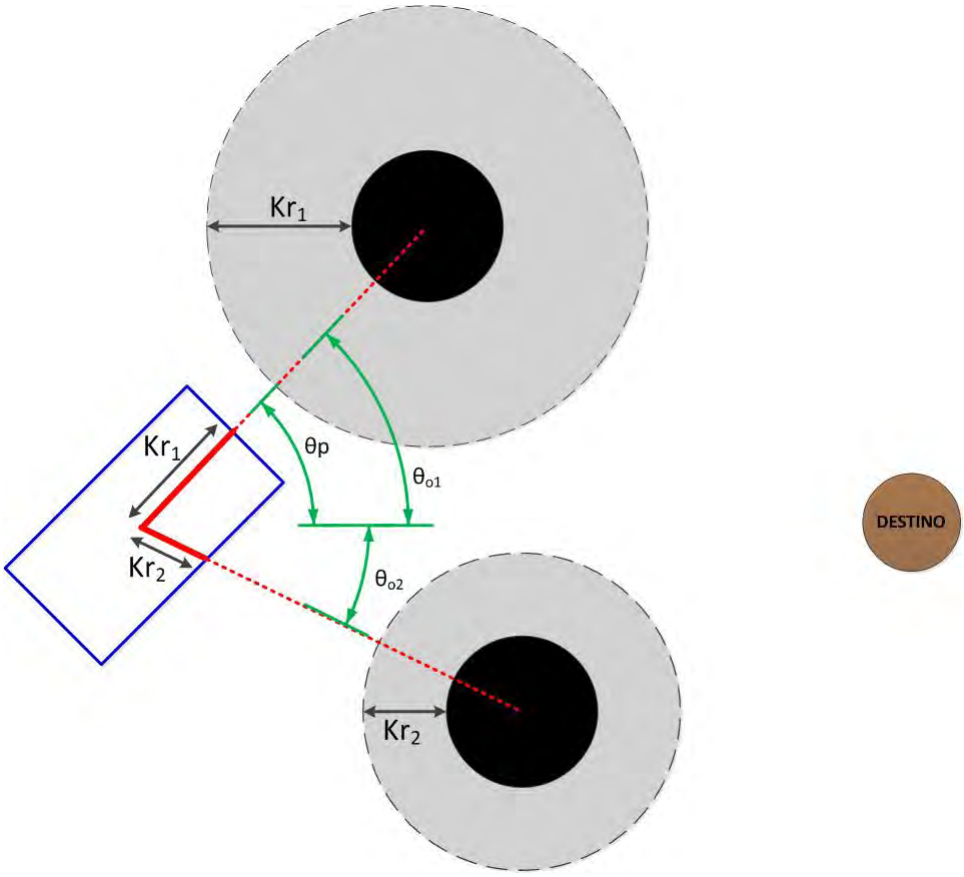


Figura 3.5: Ejemplos de distintos valores de Kr

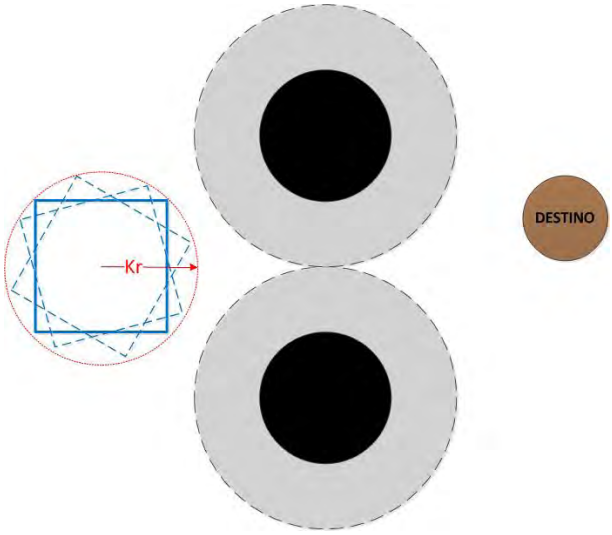
Solamente se agrega al cálculo la dimensión d del robot, pues en todo momento el robot queda por fuera del objeto a transportar, no obstante esta valor es constante. La variación de k_r se aproxima mediante la siguiente ley:

$$k_r = \frac{d}{2} + \frac{|\cos\theta_o - \sin\theta_p| * l}{2} + \frac{|\cos\theta_o - sin\theta_p| * a}{2}$$

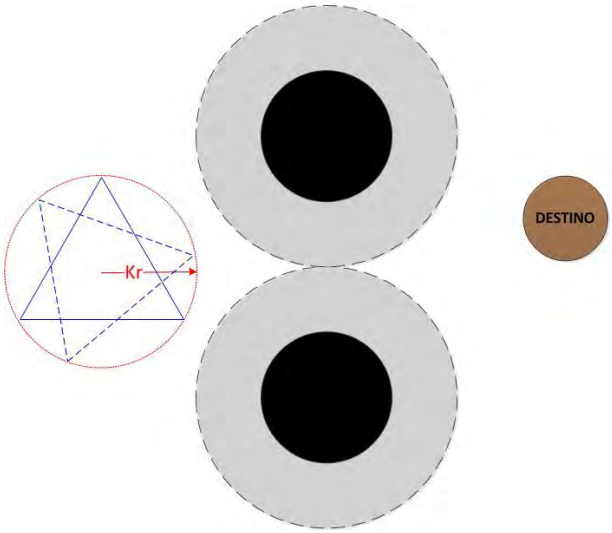
En donde:

- d es la distancia máxima entre las llantas de cada robot móvil.
- θ_o es el ángulo del obstáculo.
- θ_p es al ángulo del objeto transportado.

El caso expuesto representa el de mayor complejidad a tratar, pues los casos de figuras geométricas como cuadrados o triángulos, se pueden considerar dentro de un circulo de radio igual a la longitud de total de la configuración. Esto hace que el radio de influencia de los obstáculos se mantenga constante.



(a)



(b)

Figura 3.6: Transporte de figuras regulares

Capítulo 4

Modelos matemáticos del sistema

Una vez planeada la trayectoria a seguir es necesario tener una representación matemática de los robots a utilizar para poder controlar su movimiento de manera idónea. Para este trabajo se utilizaron dos robots del tipo (2,0), los cuales son bien descritos en [20].
Como se comentó anteriormente, la tarea se divide en dos casos: el primero es cuando los robots necesitan llegar y sujetar al objeto, el segundo cuando lo transportan al destino final.

4.1. Caso 1: Dos robots móviles independientes

4.1.1. Postura del robot móvil

Antes del acoplamiento al objeto se puede modelar al robot como un diferencial. Bastará con hacer algunas suposiciones para adecuarse al trabajo realizado por [20], tales suposiciones serán:

- El robot móvil es rígido y no deformable.
- Las llantas no se deslizan.
- El robot se mueve sobre una superficie horizontal.

Para hacer el análisis de un robot móvil se toman en cuenta dos sistemas de referencia.El primero es el sistema inercial o base sobre el cuál se mueve el robot (X,Y) y el segundo esta centrado en la posición del robot (X_1, Y_1).

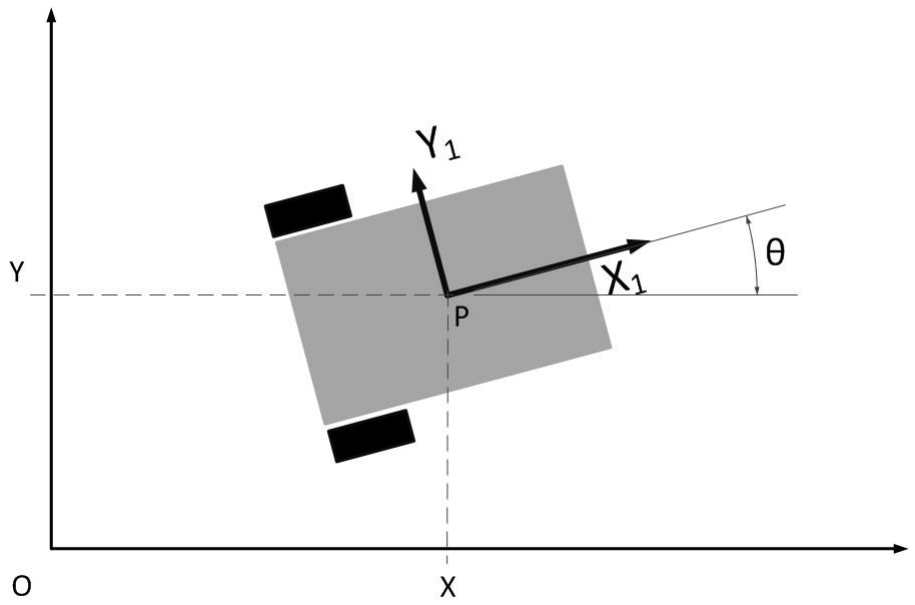


Figura 4.1: Postura de un robot móvil

De aquí que la postura del robot sera representada por :

$$\xi = [x, y, \theta]'$$

Donde x e y son la posición en el sistema de referencia (X,Y) y θ es la orientación sobre el mismo sistema. También podemos definir la matriz ortonormal de rotación del sistema que nos servirá para mapear vectores del sistema referencial al sistema del robot y viceversa.

$$R(\theta) = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Lo más importante es que también nos permite trasladar vectores de velocidad. Ahora podemos obtener el modelo del robot móvil diferencial representado en variables de estado

$$\begin{bmatrix} \dot{x_1} \\ \dot{y_1} \\ \dot{\theta_1} \end{bmatrix} = \begin{bmatrix} \cos\theta_1 & d * \sin\theta_1 \\ -\sin\theta_1 & -d * \cos\theta_1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v_1 \\ \omega_1 \end{bmatrix} \tag{4.1}$$

Para el segundo robot se sigue el mismo análisis por lo que se presenta el mismo modelo ahora con el sistema (X_2, Y_2)

$$\begin{bmatrix} \dot{x_2} \\ \dot{y_2} \\ \dot{\theta_2} \end{bmatrix} = \begin{bmatrix} \cos\theta_2 & d * \sin\theta_2 \\ -\sin\theta_2 & -d * \cos\theta_2 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v_2 \\ \omega_2 \end{bmatrix} \tag{4.2}$$

4.1.2. Modelo Cinemático

De igual forma que el desarrollo anterior se toma en cuenta el trabajo realizado en [20] para obtener el modelo cinématio del robot diferencial tipo (2,0).

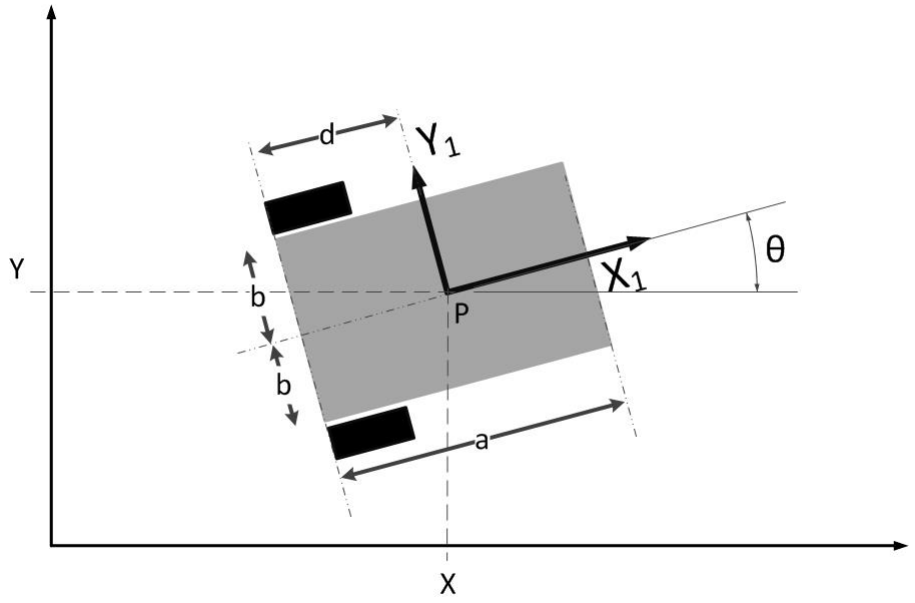


Figura 4.2: Robot tipo 2.0

De acuerdo con el trabajo de González Villela , el número de coordenadas de configuración $q = [q_1, ..., q_{n_T}]^T$ que presenta un robot móvil esta dado por la ecuación

$$n_T = 3 + N_T + N_c + N_{oc} \tag{4.3}$$

En donde:

- N_T es el número total de ruedas
- N_c es el número ruedas centradas
- N_{oc} es el número ruedas descentradas

Estas coordenadas generalizadas se encuentran distribuidas entre las coordenadas de postura ξ, los ángulos de las ruedas centradas α_c, los ángulos de las ruedas descentradas α_{oc} y de rotación de las ruedas ϕ. De acuerdo con esto las coordenadas generalizadas para el robot son:

$$q = \begin{bmatrix} x & y & \theta & \phi_r & \phi_l \end{bmatrix}$$

Donde x,y son la posición y θ la orientación en el sistema de referencia del robot (X_1, Y_1). ϕ_l y ϕ_r son los ángulos de rotación de las llantas izquierda y derecha respectivamente

alrededor de su eje.
Haciendo el desarrollo matemático necesario se puede obtener la representación en variables de estado del modelo cinemático, quedando:

$$\dot{q} = \begin{bmatrix} \cos\theta & d * \sin\theta \\ \sin\theta & -d * \cos\theta \\ 0 & 1 \\ \frac{1}{r} & \frac{b}{r} \\ \frac{1}{r} & -\frac{b}{r} \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{4.4}$$

En donde el vector de velocidades generalizadas $\dot{q}$ esta dado por la ecuación 4.3 y para el caso de un robot diferencial tipo (2,0) es:

$$\dot{q} = \begin{bmatrix} \dot{x}_1 & \dot{y}_1 & \dot{\theta}_1 & \dot{\phi}_{1_l} & \dot{\phi}_{1_r} \end{bmatrix}^T$$

Para el robot 2 el modelo es exactamente el mismo

$$\dot{q} = \begin{bmatrix} \cos\theta_2 & d * \sin\theta_2 \\ \sin\theta_2 & -d * \cos\theta_2 \\ 0 & 1 \\ \frac{1}{r} & \frac{b}{r} \\ \frac{1}{r} & -\frac{b}{r} \end{bmatrix} \begin{bmatrix} v_2 \\ \omega_2 \end{bmatrix} \tag{4.5}$$

con

$$\dot{q} = \begin{bmatrix} \dot{x}_2 & \dot{y}_2 & \dot{\theta}_2 & \dot{\phi}_{2_l} & \dot{\phi}_{2_r} \end{bmatrix}^T$$

Es importante considerar que el desarrollo matemático completo se puede encontrar en [20].

4.2. Caso 2: Dos robots móviles acoplados

4.2.1. Postura de la configuración

Hasta ahora se tiene la representación del robot en la primera parte del problema. Falta por desarrollar el modelo que se consideró para la segunda parte, el transporte del objeto. Para esta parte se plantea que existe un conjunto de robots móviles unidos por un cuerpo central, que para este caso será el objeto a transportar.

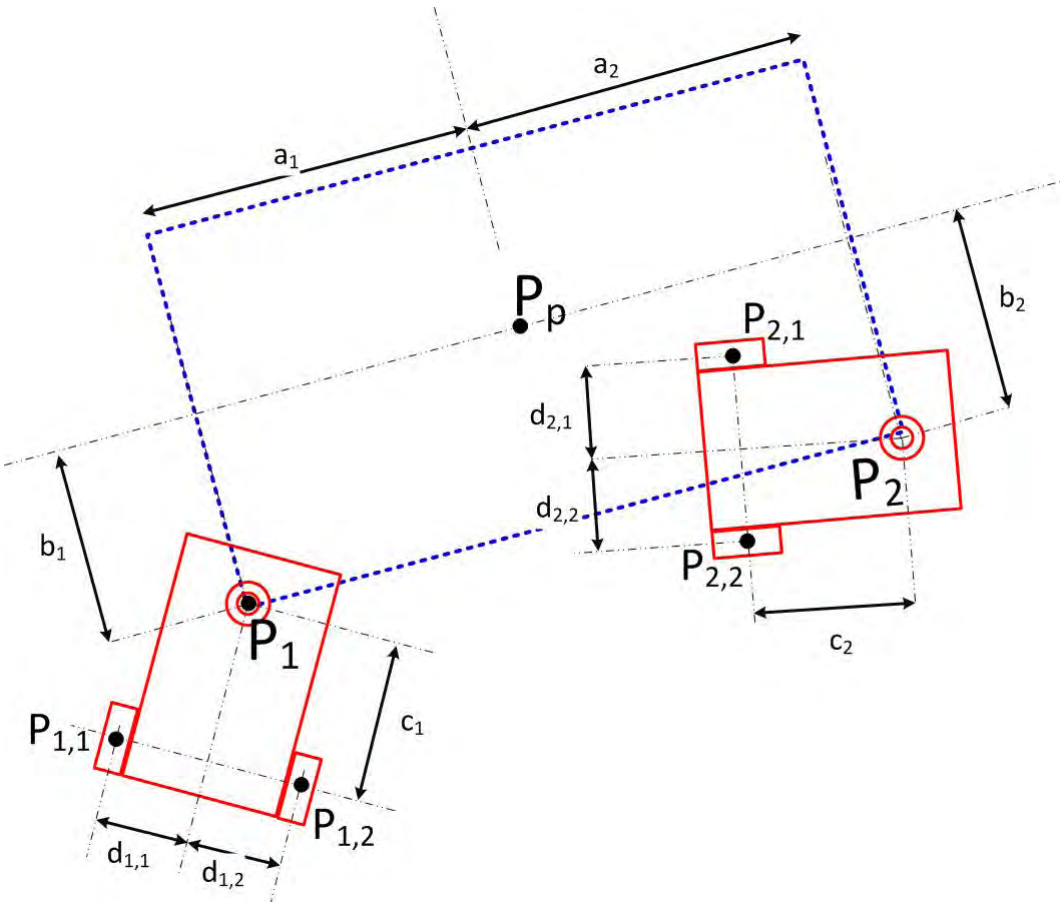


Figura 4.3: Configuración propuesta para el transporte

En dónde a2 será la longitud del objeto a transportar divida entre dos y b el la profundidad o ancho del objeto. El resto de las dimensiones especificada en la figura 4.3 ya se vieron anteriormente y están relacionadas con las dimensiones de cada robot móvil de tracción.
Para realizar la cinemática inversa de la configuración se utilizan el método de la propagación de velocidades. El desarrollo matemático completo es tomado de [21].
Este método nos permite hacer la propagación de los diferentes subsistemas que encontramos en la configuración propuesta. Lo que el método sugiere es que la velocidad angular del eslabón i+1 relativa a su propio sistema de referencia es igual a la velocidad angular del eslabón i con respecto al sistema i+1 más una componente propia del eslabón i+1.

$$ {}^{i+1}_{i+1}\omega = {}^{i+1}_{i}R \times {}^{i}_{i}\omega + {}^{i}_{i}\dot{\theta} \tag{4.6} $$

Por otro lado la velocidad lineal del eslabón i+1 en relación con sigo mismo es igual a la velocidad lineal del eslabón i más la velocidad lineal provocada por el brazo de palanca ${}^{i}_{i+1}P$ y la velocidad angular de i_p. Todo referenciado al sistema i+1

$$ {}^{i+1}_{i+1}v = {}^{i+1}_{i}R \times ({}^{i}_{i}v + {}^{i}_{i}\omega \times {}^{i}_{i+1}P) \tag{4.7} $$

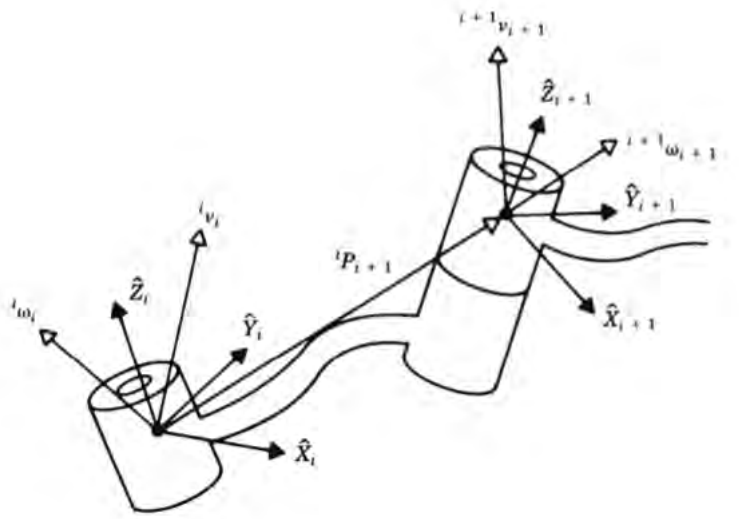


Figura 4.4: Metodo de propagación de velocidades del eslabón i al i+1

Ahora bien, el punto de análisis será el centro del objeto a mover P_p, por lo que se procede a hacer la propagación de velocidades desde P_p a cada una de las ruedas.
En la siguiente figura podemos observar los diferentes sistemas de referencia considerados para el análisis cinemático.

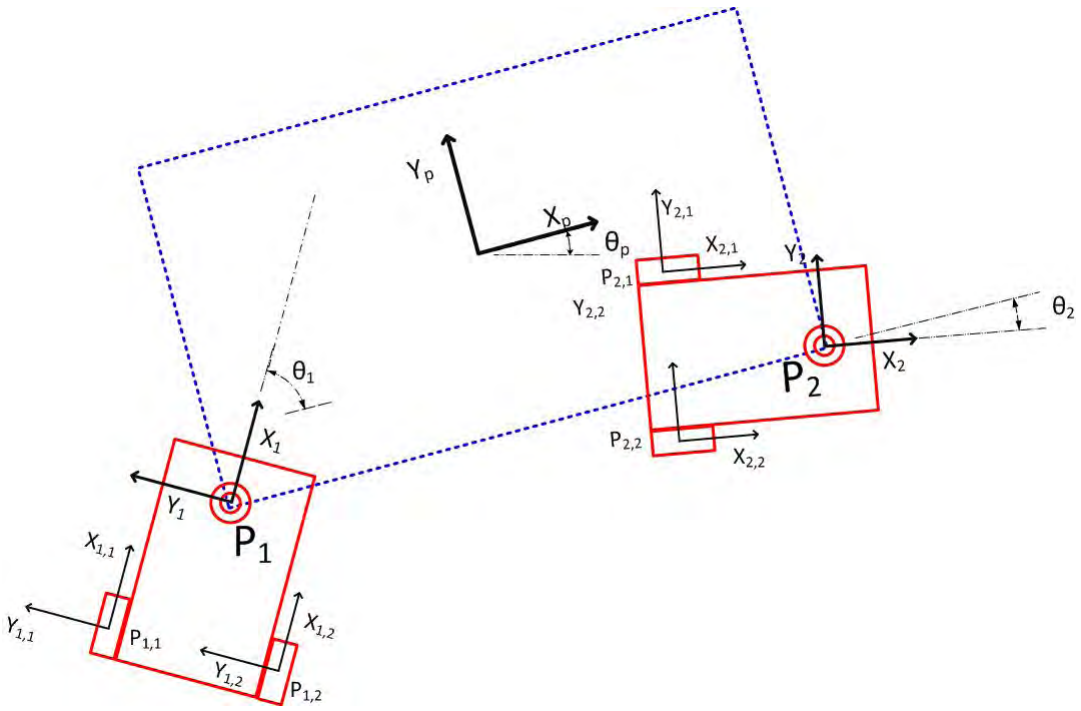


Figura 4.5: Postura de la configuración para el transporte

En dónde el sistema (X_p, Y_p) es el del objeto a transportar, (X_1, Y_1) y (X_2, Y_2) son los sistemas del robot de tracción 1 y 2 respectivamente. También aparece los sistemas $(X_{1,1}, Y_{1,2})$ correspondientes a las llantas del robot 1 y $(X_{2,1}, Y_{2,2})$ de las llantas del robot 2.

De forma que utilizando las ecuaciones 4.6 y 4.7 como se muestra en [21] se pude obtener la representación para cada sistema, obteniéndose un total de ocho ecuaciones que expresan el modelo matemático y señalan las restricciones para las ruedas de los robots de tracción. Agrupando las ecuaciones de la propagación de velocidades de cada rueda al punto P_p y expresándolas en la forma propuesta en [20], es decir una ecuación de la forma $A_\tau(q)\dot{q} = 0$, obtenemos el modelo matemático de la configuración propuesta. Aún más se obtiene la forma general, entonces al sustituir las dimensiones y ángulos de la figura 4.5 obtenemos el modelo de nuestro caso.

$$\begin{bmatrix} -\sin\theta_1 & \cos\theta_1 & a_1\cos\theta_1 + b\sin\theta_1 & -c & 0 & 0 & 0 & 0 & 0 \\ -\sin\theta_1 & \cos\theta_1 & a_1\cos\theta_1 + b\sin\theta_1 & -c & 0 & 0 & 0 & 0 & 0 \\ -\sin\theta_2 & \cos\theta_2 & a_2\cos\theta_2 + b\sin\theta_2 & 0 & -c & 0 & 0 & 0 & 0 \\ -\sin\theta_2 & \cos\theta_2 & a_2\cos\theta_2 + b\sin\theta_2 & 0 & -c & 0 & 0 & 0 & 0 \\ \cos\theta_1 & \sin\theta_1 & -b\cos\theta_1 + a_1\sin\theta_1 & -d & 0 & -r & 0 & 0 & 0 \\ \cos\theta_1 & \sin\theta_1 & -b\cos\theta_1 + a_1\sin\theta_1 & d & 0 & 0 & -r & 0 & 0 \\ \cos\theta_2 & \sin\theta_2 & -b\cos\theta_2 + a_2\sin\theta_2 & 0 & -d & 0 & 0 & -r & 0 \\ \cos\theta_2 & \sin\theta_2 & -b\cos\theta_2 + a_2\sin\theta_2 & 0 & d & 0 & 0 & 0 & -r \end{bmatrix} \dot{q} = 0 \qquad (4.8)$$

4.2.2. Modelo cinemático

Con la nueva configuración se tienen tres grados de libertad, es pertinente seleccionar tres variables para el control por retroalimentación de estados. Es evidente seleccionar las velocidades lineales V_{xp}, V_{yp} y la angular ω_p. Estas velocidades serán las variables de estado del sistema y de acuerdo con la teoría unificadora de González Villela V. se deben expresar como:

$$\dot{q_1} = \begin{bmatrix} \sum_{\xi}(\alpha_s) & 0 \\ 0 & \sum_{c}(\alpha_s) \\ \sum_{oc}(\alpha_s, \alpha_{oc}) & 0 \\ \sum_{\phi}(\alpha_s, \alpha_{oc}) & 0 \end{bmatrix} \begin{bmatrix} \eta \\ \zeta \end{bmatrix} \tag{4.9}$$

Donde $\sum_{\xi}$ es la matriz asociada a las coordenadas de postura; $\sum_{oc}$ es la matriz asociada a las coordenadas de ruedas descentradas; $\sum_{\phi}$ la matriz asociada a las coordenadas de rotación de las ruedas y $\sum_{c}$ la matriz de ruedas centradas. Además de η y ζ que corresponden a la movilidad y direccionabilidad respectivamente.
En el caso de esta configuración de robots los elementos de la ecuación 4.9 $\sum_{oc}$, $\sum_{c}$ y ζ no existen, y η es el vector de estado $[v_{xp}\ v_{yp}\ \omega_p]^T$.
Ahora podremos obtener la cinemática interna de la configuración, pero si recordamos el análisis se hizo a partir del punto P_p por lo que es necesario multiplicar por la matriz de rotación de la configuración (X_p, Y_p) para expresar la cinemática referenciada a nuestro universo o sistema (X,Y)

$$R_q = \begin{bmatrix} {}^0_pR(\theta_p) & 0 & 0 \\ 0 & I_{N_{DR}} & 0 \\ 0 & 0 & I_{N_C} \end{bmatrix}$$

En donde $R_q \in R^{(3+N_{DR}+N_C)\times(3+N_{DR}+N_C)}$ y tanto $I_{N_{DR}} \in R^{N_{DR}\times N_{DR}}$ como $I_{N_C} \in R^{N_C\times N_C}$ son matrices identidad. Esta matriz de rotación esta definida en el trabajo desarrollado en [20].
Una vez multiplicada la ecuación 4.9 con la matriz de rotación obtenemos las velocidades generalizadas de la configuración propuesta en términos de las variables de estado. Para obtener dicha expresión seguimos la forma:

$$\dot{q} = S_\tau(q)u \equiv \begin{bmatrix} \dot{\xi} \\ \dot{\theta_{DR}} \\ \dot{\phi} \end{bmatrix} \equiv R_q\dot{q_1}$$

Para finalmente obtener, las velocidades generalizadas en términos de las variables de entrada con respecto al sistema coordenado (X,Y). Es necesario señalar que la controla-

bilidad de este sistema se encuentra demostrada con álgebra de Lie en [21].

$$\dot{q} = \begin{bmatrix} \cos\theta_p & -\sin\theta_p & 0 \\ \sin\theta_p & \cos\theta_p & 0 \\ 0 & 0 & 1 \\ -\frac{\sin\theta_1}{c} & \frac{\cos\theta_1}{c} & \frac{a_1\cos\theta_1 + b\sin\theta_1}{c} \\ -\frac{\sin\theta_2}{c} & \frac{\cos\theta_2}{c} & \frac{a_2\cos\theta_2 + b\sin\theta_2}{c} \\ \frac{c\cos\theta_1 + d\sin\theta_1}{cr} & \frac{c\sin\theta_1 - d\cos\theta_1}{cr} & \frac{(a_1c - bd)\sin\theta_1 - (bc - a_1d)\cos\theta_1}{cr} \\ \frac{c\cos\theta_1 - d\sin\theta_1}{cr} & \frac{c\sin\theta_1 + d\cos\theta_1}{cr} & \frac{(a_1c + bd)\sin\theta_1 + (-bc + a_1d)\cos\theta_1}{cr} \\ \frac{c\cos\theta_2 + d\sin\theta_2}{cr} & \frac{c\sin\theta_2 - d\cos\theta_2}{cr} & \frac{(a_2c - bd)\sin\theta_2 - (bc - a_2d)\cos\theta_2}{cr} \\ \frac{c\cos\theta_2 - d\sin\theta_2}{cr} & \frac{c\sin\theta_2 + d\cos\theta_2}{cr} & \frac{(a_2c + bd)\sin\theta_2 + (-bc + a_2d)\cos\theta_2}{cr} \end{bmatrix} \begin{bmatrix} v_{xp} \\ v_{yp} \\ \omega_p \end{bmatrix} \tag{4.10}$$

En donde el vector de velocidades generalizadas $\dot{q}$ se obtiene de la ecuación 4.3 haciendo una modificación, pues existen otras dos componentes que es necesario tomar en cuenta. Estas nuevas componentes corresponden a los ángulos de orientación de los robots de tracción θ_1 y θ_2. La nueva ecuación que expresa el número de coordenadas generalizadas es

$$n_T = 3 + N_T + N_c + N_{oc} + N_{DR} \tag{4.11}$$

El vector de coordenadas generalizadas para nuestro sistema queda entonces de la siguiente forma

$$q = \begin{bmatrix} {}^p_p x & {}^p_p y & \theta_p & \theta_1 & \theta_2 & \phi_{1,1} & \phi_{1,2} & \phi_{2,1} & \phi_{2,2} \end{bmatrix}^T$$

Y las velocidades generalizadas

$$\dot{q} = \begin{bmatrix} {}^p_p \dot{x} & {}^p_p \dot{y} & \dot{\theta}_p & \dot{\theta}_1 & \dot{\theta}_2 & \dot{\phi_{1,1}} & \dot{\phi_{1,2}} & \dot{\phi_{2,1}} & \dot{\phi_{2,2}} \end{bmatrix}^T$$

Los valores de las velocidades que se utilizan en ambos modelos (caso 1 y caso 2) son calculados con la aproximación por campos potenciales explicada un capitulo atrás.

Capítulo 5

Descripción de la simulación e implementación del sistema

5.1. Simulación

Para probar el funcionamiento del sistema de transporte colaborativo se realizan las simulaciones y pruebas pertinentes. Es importante verificar antes de implementar con un modelo funcional el control, la planeación de trayectorias y el algoritmo de coordinación; es por esto que se desarrolla una simulación que permita comprobar gráficamente el funcionamiento del sistema. El software elegido para realizar esta simulación es Simulink ®. Simulink®es un ambiente gráfico que permite hacer simulaciones y diseño basado en modelos. También cuenta con una gran cantidad de herramientas de control, funciones embebidas y generación automática de código; además de ser *software* ya conocido y usado con anterioridad por el autor de este trabajo. Otra gran ventaja de Simulink es que también permite controlar en tiempo real a los robots móviles, siendo una opción ideal para el planeador global; solo será necesario acondicionar las señales de entrada y salida necesarias para el modelo cinemático, es decir, señales del sistema de visión y señales para los motores de los robots.
El primer componente de la simulación es un programa hecho con Matlab que permite establecer todos los parámetros iniciales necesarios, parámetros de los robots móviles como:

- Dimensiones
- Velocidades máximas
- Condiciones iniciales para el modelo cinemático

También parámetros del entorno de trabajo como:

- Dimensiones del entorno

- Dimensiones y ubicación del objeto a transportar
- Dimensiones y ubicación de los obstáculos
- Ubicación del destino final del objeto

El código de este programa se puede encontrar los apéndices de este trabajo. Para lograr condiciones iniciales lo más arbitrarias posibles, este código cuenta con un generador de números pseudo-aleatorios para generar las condiciones iniciales de postura de los robots móviles, del objeto a transportar, así como de la ubicación de los obstáculos en el espacio de trabajo. Además de este programa, se implementó una segunda versión capaz de obtener dichas condiciones iniciales a partir de los datos del sistema de visión. Esto permite realizar una simulación de la prueba a realizar y así comparar de mejor forma el comportamiento del sistema.
En el siguiente diagrama se presenta el funcionamiento general de la simulación

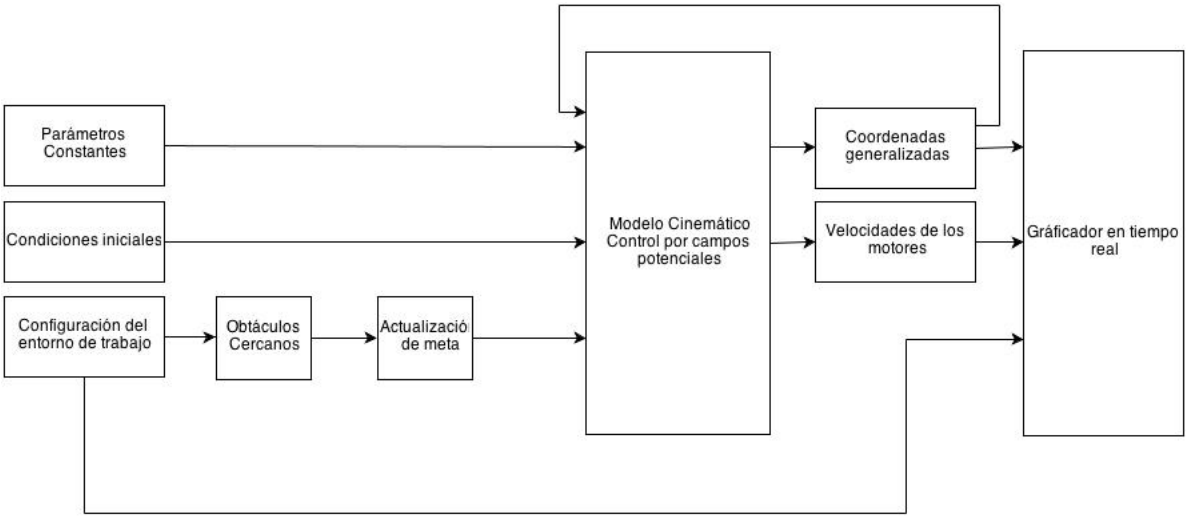


Figura 5.1: Funcionamiento general de la Simulación

El bloque obstáculos cercanos se encarga de actualizar el objeto más cercano a cada robot durante su navegación para en caso de ser necesario evadirlo. El bloque actualización de meta se encarga de dirigir a los robots cuando están aproximándose al objeto, se hace siguiendo el algoritmo descrito en el capitulo 3 (aproximación al obstáculo).
Como se ha mencionado a lo largo de este trabajo la tarea a resolver consiste de dos partes, antes y después de tomar el objeto. Para el caso de la simulación el funcionamiento es el mismo para ambas partes, lo único que cambiará es el modelo cinemático a resolver para cada caso y la aproximación por campos potenciales, todo esto fue detalladamente explicado en los capítulos 3 y 4.
El código de ambos robots se encuentra dentro del mismo bloque; sin embargo, de ser necesario, es posible transferir éste código al microcontrolador a bordo de cada robot.

Esto gracias a que en la programación del modelo cinemático y los campos potenciales se implementaron funciones simples compatibles con lenguaje C.
El bloque graficador permite ver la posición de obstáculos, el objeto a mover y los robots en todo momento. También gracias Simulink ®es posible obtener por separado el gráfico correspondiente a las velocidades tanto del robot como de las llantas de cada uno de ellos. A continuación se muestra un ejemplo del gráfico generado:

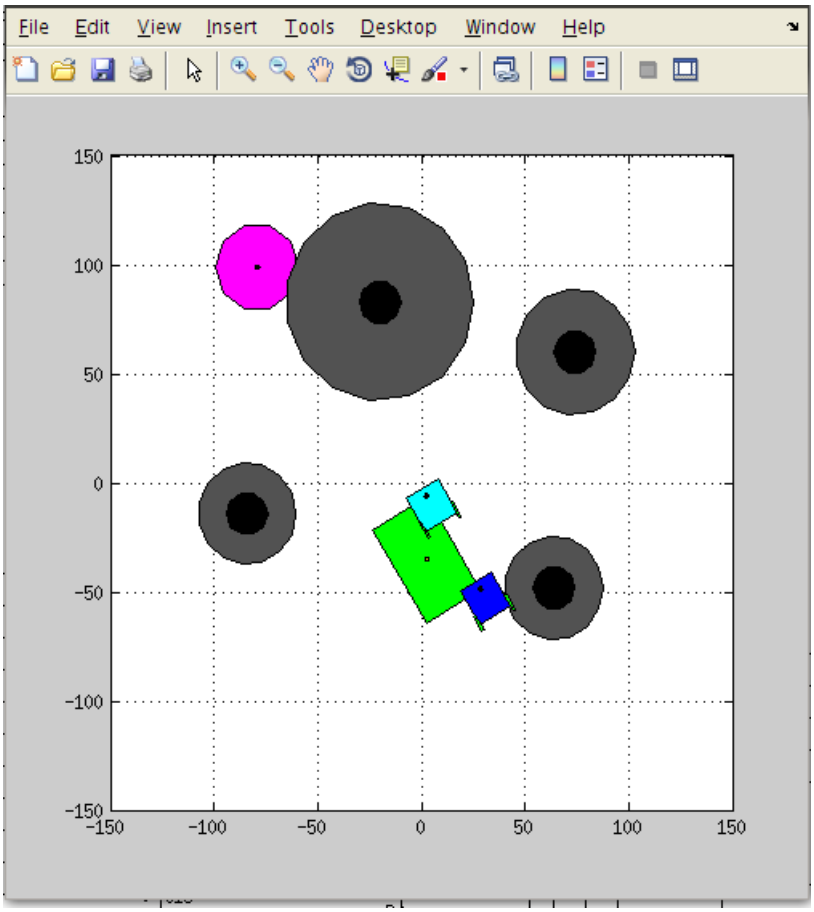


Figura 5.2: Graficador en tiempo real

5.2. Implementación del sistema

5.2.1. Sistema de visión

Como se comentó en los primeros capítulos el robot necesita de elementos de censado para conocer el medio en el que trabaja. Para este caso el único elemento de censado es la visión artificial. Gracias al sistema de visión empleado se puede conocer la ubicación y orientación de los robots, obstáculos y del objeto a transportar. El software empleado

para el sistema de visión es reacTIVsion, que es una plataforma de software libre utilizado para rastreo de símbolos *fiducials* colocados en objetos.
ReacTIVision surgió como el componente de censado base de Reactable, un sintetizador modular y tangible. ReacTIVision es una aplicación que emite mensajes TUIO utilizando el protocolo UDP (*User Data Protocol*) en el puerto 3333 a cualquier cliente que se encuentre habilitado. Este *framework* funciona bajo plataformas Windows, Linux y MacOS X, lo cual representa una gran ventaja para su utilización. Otra de las caracteristicas importantes de reacTIVision es que se puede implementar con gran variedad de lenguajes de programación como: JAVA, C++, C# y *Processing*. De forma que dependiendo del lenguaje empleado se contará con un conjunto de funciones(métodos), objetos y clases que mediante el protocolo TUIO, hacen peticiones sobre información especifica de los *fiducials* como puede ser su posición, orientación, velocidad, identificador (ID),etc. Todos ellos de vital importancia para el rastreo de los robots y demás elementos involucrados en el sistema de transporte.
ReacTIVision puede funcionar con cámaras USB, USB2 y FireWire, siempre y cuando el sistema operativo cuente con el controlador correcto. La cámara utilizada para las pruebas es una webcam USB LifeCam Studio.
Para el rastreo de los fiducials se coloca la cámara en la parte superior del espacio de trabajo, de forma que pueda captar todo lo que acontezca en el entorno (visión global). Todo elemento del cual se requiera su información deber tener un *fiducial* para que reacTIVision lo pueda localizar. Ejemplos de estos símbolos se pueden observar a continuación:

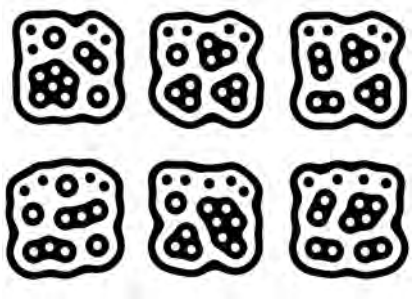


Figura 5.3: Ejemplos de los símbolos *fiducial*

Una vez colocados los símbolos en el espacio de trabajo se puede obtener información sobre ellos. Lo primero es lanzar la aplicación reacTIVision, para este caso será suficiente con introducir en la terminal el comando reacTIVision y presionar la tecla Enter. La aplicación será llamada a ejecutar y se hará la detección de la(s) cámara(s) conectada(s), la preferencias de la cámara, el número de cámara o dispositivo a utilizar, resolución y demás se pueden editar en el archivo camera.xml

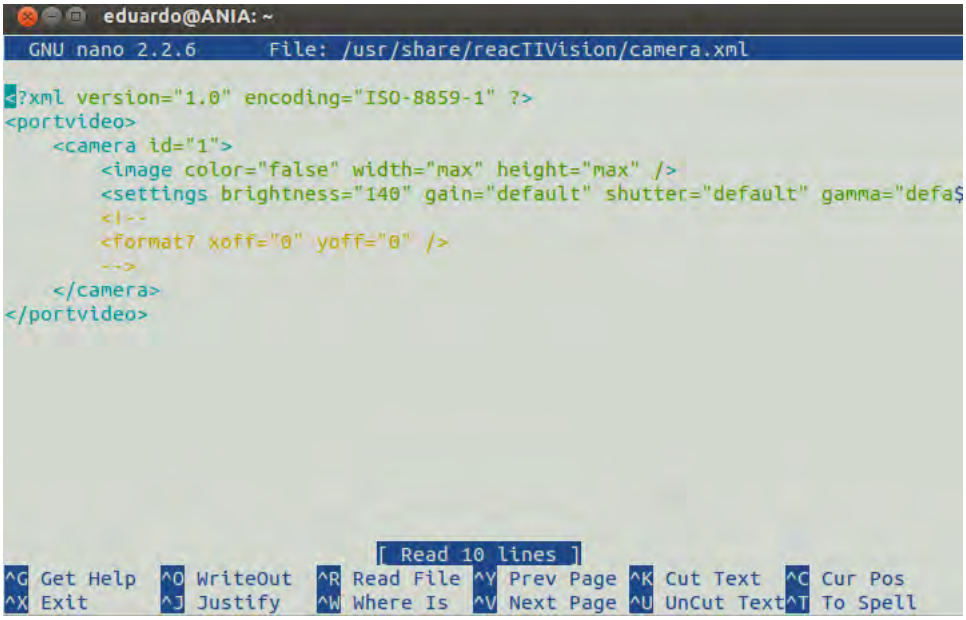


Figura 5.4: Archivo de configuración de cámara

Al ejecutar la aplicación se observan en la terminal los comandos que se pueden utilizar con la aplicación de reacTIVision, algunos de los más importantes son para calibrar la cámara, variar parámetros de la imagen como el brillo e inclusive alternar entra las imágenes antes y después del procesado.

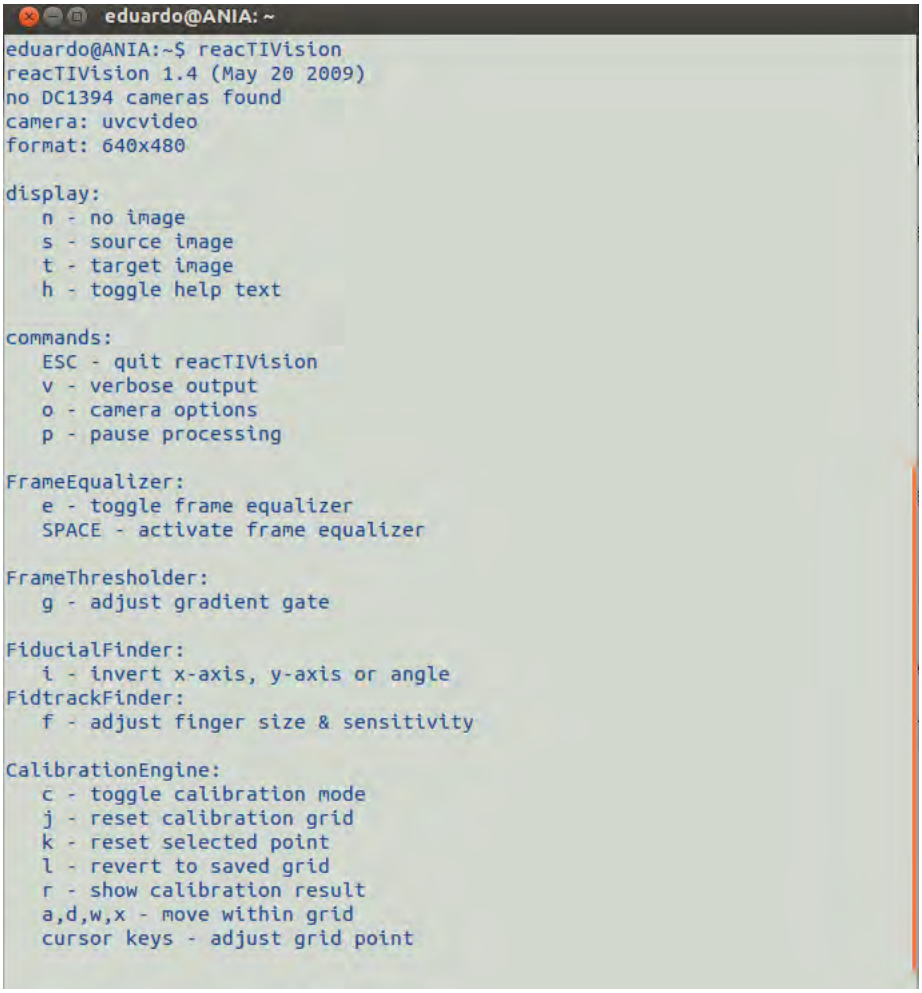


Figura 5.5: Comandos para aplicación reacTIVision

Al mismo tiempo se genera y se muestra la ventana con la imagen que capta la cámara, es decir el entorno de trabajo reacTIVision.

Figura 5.6: Entorno de trabajo reacTIVision

Para poder acceder a la información de los símbolos es necesario, como se mencionó anteriormente, contar con un cliente que pida dicha información. Debido a qué el sistema colaborativo se implementó con SIMULINK/Matlab se deberá utilizar un cliente capaz de correr en este software. Para solucionar esta situación se hace uso del soporte que tiene Matlab para código en JAVA. Los pasos para establecer el cliente Java en SIMULINK/Matlab son los siguientes:

1. Descargar de la página del proyecto reacTIVision el cliente TUIO_JAVA.zip
2. Descomprimir la carpeta en una ubicación que no deberá cambiar a lo largo del trabajo.
3. En la ventana de comandos de Matlab escribir el comando "edit classpath.txt"para editar el path de Matlab.
4. Una vez abierto el archivo, agregar la ruta del archivo "libTUIO.jar"que se encuentra en la carpeta del punto 2.
5. Guardar y cerrar el archivo.

Ya con estos pasos será posible implementar funciones de reacTIVision escritas en Java. Para tener funcionalidad completa ahora se deberá indicar a Matlab que utilizaremos la biblioteca de funciones TUIO, crear y conectar al cliente. Lo anterior se traduce en agregar al inicio del código las siguientes lineas:

- import TUIO.*;
- tc=TuioClient();
- tc.connect();

Acciones similares se deberán realizar en caso de utilizar cualquier otro lenguaje de programación, por supuesto respetando la sintaxis adecuada.
Es importante mencionar que reacTIVision trabaja con una imagen con resolución de AxB pixeles(dependiendo del archivo camera.xml y las características del dispositivo), entonces excluyendo la orientación de los símbolos, la información que nos brinda tiene como unidad pixeles y como referencia la esquina superior izquierda de la imagen.

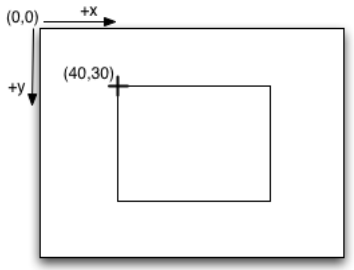


Figura 5.7: Ejemplo de sistema de coordenadas de imagen

En donde los valores máximos de (X,Y) son (1,1). Por lo tanto necesitamos convertir este sistema de coordenadas a uno que represente de mejor forma al entorno de trabajo,un sistema coordenado común es el cartesiano en dos dimensiones. Para este trabajo solo se contempla el primer cuadrante.
Para esta conversión hacemos uso de una aproximación lineal. En principio se podrían asignar valores máximos y mínimos para X e Y y hacer la conversión de coordenadas, sin embargo para este trabajo se procede de la siguiente forma:

1. Se colocan tres *fiducials* en posiciones conocidas del entorno de trabajo.
2. Se obtiene la información de posición mediante reacTIVision de estos *fiducials*.
3. Debido a que los *fiducials* se encuentran en posiciones conocidas, se procede a hacer la conversión o equivalencia pixeles-metros, esto mediante la aproximación lineal.

Con este procedimiento se evita tener que verificar cuál es el campo de visión de la cámara (en metros) para hacer la conversión de coordenadas. En este caso la aproximación lineal queda como lo siguiente.

$$X[cm] = 234.94X[px] - 112.77$$
$$Y[cm] = -197.23Y[px] + 100.64$$

Estos valores corresponden a una resolución de 640x480 pixeles con la cámara colocada a aproximadamente 2.48 metros de altura.

5.2.2. Robots Móviles

El sistema de transporte colaborativo propuesto está compuesto por dos robots diferenciales: Estos robots diferenciales están conformados principalmente de acero, placas de acrílico y los elementos que le dan movilidad los motores y llantas.

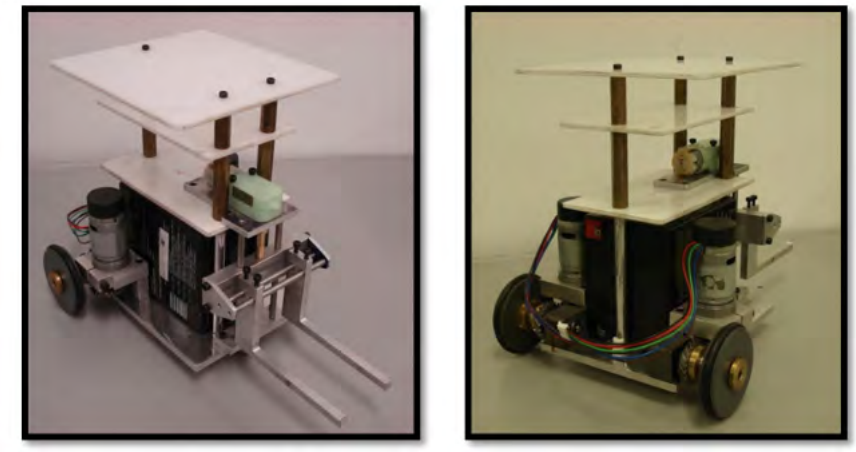


Figura 5.8: Robots utilizados para las pruebas

Del capitulo anterior se sabe que las dimensiones importantes para el modelo matemático son la distancia entre las llantas, la distancia entre el eje de las llantas y el punto P de análisis así como el radio de las llantas. Para esta caso dichas dimensiones son:

- Distancia entre llantas 11 [cm]
- Distancia más corta al punto P 7.5 [cm]
- Radio de las llantas 3 [cm]

Las placas de acrílico funcionan como entrepaños para colocar la circuitería utilizada, así como algunos otros elementos indispensables para el experimento como el fiducial y el motor del sinfin.

Mecanismo de ensamble

Una vez que los robots han llegado al punto de *docking* deben engancharse al objeto para poder llevarlo al punto final. Para este caso se utilizó un perno que se engancha en una argolla, la argolla es capaz de moverse verticalmente para enganchar o liberar el perno gracias al giro de un tornillo sinfin, este tornillo sinfin gira con un motorredutor de CD.

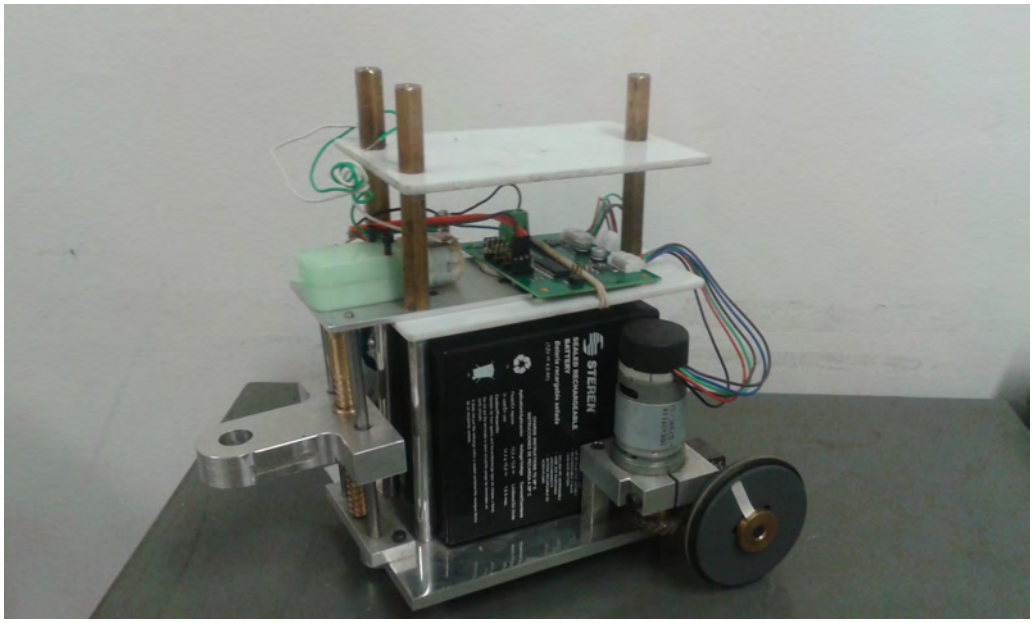


Figura 5.9: Mecanismo de ensamble

Tarjeta Arduino UNO

El sistema esta controlado por un planeador global el cual, a partir de la información del sistema de visión, decide la trayectoria a seguir por cada robot, y con el modelo empleado calcula las velocidades de los motores para seguir dicha trayectoria. Después, envía la información de las velocidades a cada robot inalámbricamente. Cada robot cuenta con un microcontrolador ATMEL en una tarjeta Arduino UNO, gracias a este microcontrolador se puede establecer la comunicación con el módulo de radiofrecuencia.

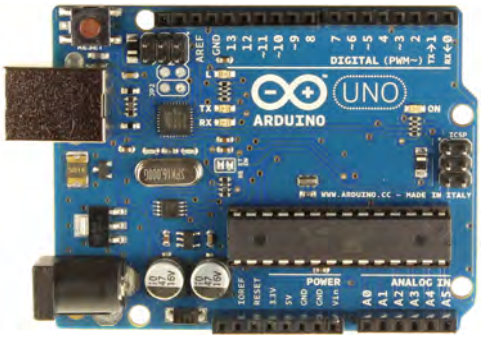


Figura 5.10: Tarjeta de desarrollo Arduino Uno

A esta tarjeta se le agrega un *shield* o expansión que permite conectar fácilmente el módulo de radiofrecuencia. De esta forma el microcontrolador puede transmitir en la red ZigBee emulando una comunicación serial.

Figura 5.11: Xbee shield

Otra función del microcontrolador es la de activar el motor del sin fin una vez que el robot se localiza en la posición correcta para enganchar al objeto. Esto se hace mediante uno de los puertos digitales de la tarjeta Arduino UNO. Esta salida digital funciona como *trigger* para un temporizador que activa el motor durante seis segundos. Este temporizador se implemento de manera externa ya que el microcontrolador necesita escuchar todo el tiempo los comandos remotos, lo que no permite implementar una rutina dentro el mismo para activar o desactivar el motor.

Tarjeta MD25

Una vez que el microcontrolador recibe la información sobre las velocidades que deben tener los motores, estas velocidades se deben enviar a los motores. Los motores utilizados son de CD modelo EMG30 que pueden ser controlados de manera práctica con la tarjeta MD25. Esta tarjeta es una tarjeta de doble puente H y esta diseñada para controlar específicamente dos motores de modelo mencionado. Algunas de las características de la tarjeta MD25 son:

- Alimentación única de 12 VCD.
- Contiene un regulador de +5VCD a 300 mA para alimentar circuitería externa.
- Es capaz de proporcionar hasta 3 A a cada motor.
- Comunicación serial e I^2C

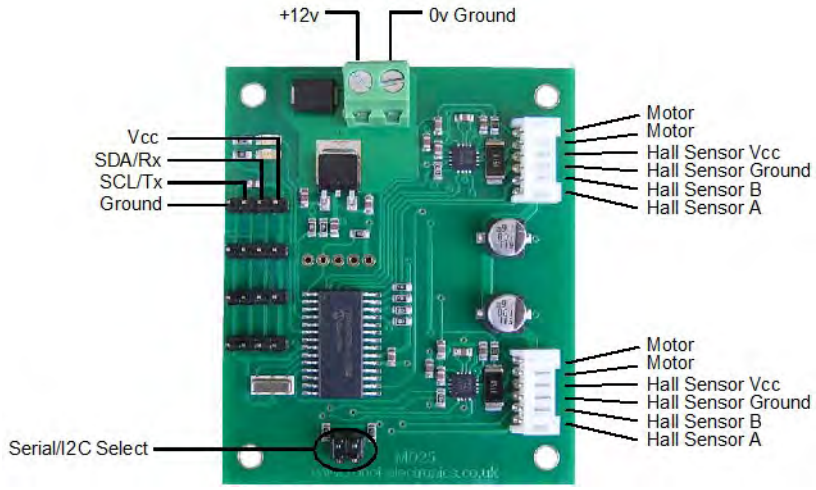


Figura 5.12: Tarjeta MD25

Por otro lado el microcontrolador empleado también posee comunicación serial e I^2C, sin embargo el puerto serial ya se emplea para comunicarse con el modulo XBee, por lo que la comunicación entre la tarjeta MD25 y el microcontrolador es mediante el protocolo I^2C. El microcontrolador enviará las velocidades de cada motor a la tarjeta MD25 mediante este protocolo y la tarjeta será encargada de escribir esas velocidades en cada motor. La tarjeta MD25 cuenta con varios registros internos en dónde se escriben las operaciones a realizar. La forma básica de funcionamiento es:

1. Indicar que se establecerá comunicación.
2. Escribir la velocidad de cada motor en el registro correspondiente.
3. Indicar que se cierra la comunicación.

La tarjeta puede trabajar en 4 modos diferentes, se emplea el determinado modo 0. En este modo los motores de controlan de forma independiente: un valor de 0 representa la máxima velocidad en un sentido, con 128 el motor se detiene y 255 es la máxima velocidad en el otro sentido. Información sobre los otros modos de operación se puede encontrar en [24].
Ahora bien un valor de 255 escrito sobre el motor corresponde a 190 RPM y un valor de 0 corresponde a 50 RPM en sentido contrario. Nuestro modelo cinemático arroja las velocidades de los motores en radianes por segundo, por lo que resulta necesario establecer una relación entre ellas

Velocidad Motores (RPM)	Rad/s	Velocidad Tarjeta
190	19.896753473	255
0	0	128
-50	-19.896753473	0

Figura 5.13: Correspondencia de velocidades

Utilizando un aproximación lineal se pude obtener la función que convierte los radianes por segundo arrojados del modelo en un número entre 0 y 255, que a su vez se traducirá en RPMs escritas sobre los motores.

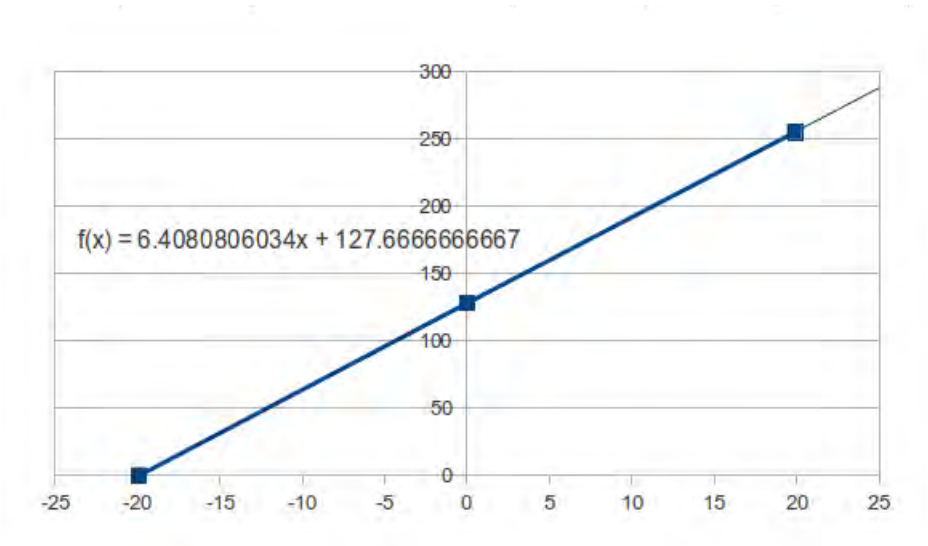


Figura 5.14: Función para convertir velocidades MD52

5.2.3. Comunicación

La comunicación es un parte importante del sistema colaborativo, al tratarse de robots móviles resulta adecuado utilizar un sistema inalámbrico para no limitar el espacio de trabajo y no entorpecer el movimiento de los robots. Para el modulo de comunicación inalámbrica se decide utilizar módulos de radiofrecuencia XBee Serie 2 de Digi [25]. Estos módulos están diseñados para trabajar bajo el protocolo ZigBee y representan una opción de bajo costo y bajo consumo de potencia para redes inalámbricas.
ZigBee es una plataforma estandarizada abierta bajo la norma 802.15.4 de la IEEE. Provee comunicación en ambientes hostiles de radiofrecuencia que son comunes hoy en día, tanto en ambientes comerciales como industriales. Además de que posee soporte para diversas topologías de red como: punto a punto, punto a multi-punto e incluso redes de malla. En esta plataforma se definen tres tipos de dispositivos: coordinador, router y dispositivo final(*coodinator,router,end device*) con las siguientes características:

- **Coordinador**
  - Permite a routers y dispositivos finales unirse a la red.
  - Siempre está activo.

- Puede enrutar datos.
- Selecciona el canal y PAN ID para comenzar la red.

- **Router**
  - Debe unirse a una red antes de poder transmitir.
  - Puede permitir a otros dispositivos unirse a la red.
  - Puede enrutar datos.
  - Siempre activo.

- **Dispositivo Final**
  - Debe unirse a una red antes de transmitir o recibir.
  - No puede unir a otros dispositivos a la red.
  - No puede enrutar datos
  - Puede entrar en modo ahorrador de energía(dormir).

En toda red ZigBee debe haber por lo menos un coordinador para que los nodos se puedan comunicar. Por lo tanto una red personal (PAN) consiste de un coordinador y uno o más routers o dispositivos finales. La red se crea cuando el coordinador elige un canal y un identificador de red (PAN ID). Una vez creada la red se puede dar permiso a otros dispositivos para unirse a ella.
En la siguiente figura se puede ver un ejemplo de topología ZigBee:

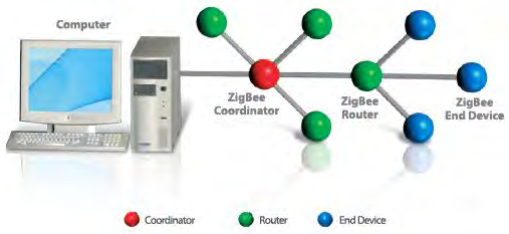


Figura 5.15: Ejemplo de red ZigBee

Estos módulos de radiofrecuencia permiten dos modos de configuración: el modo API y el modo transparente o modo AT. En el modo AT los módulos transmiten o dejan pasar toda la información que se les envía, funcionan como un simple substituto de RS232. En el modo API el módulo se configura para enviar una trama específica que, además de la cadena original, contiene información del nodo emisor, nodo destino, *checksum*, entre otros.
Debido a que el sistema solo cuenta con dos dispositivos finales que escucharan al planeador global se elige el modo de operación AT por su sencillez. Una consideración especial

es que en primera instancia parecería adecuada una comunicación punto-multipunto, sin embargo en los módulos Xbee Series 2 este tipo de comunicación solo permite tasas de transferencia de .5 [Hz] por lo que no resulta conveniente. La solución consiste entonces en una red punto a punto, en donde el coordinador enviará información al dispositivo final 1 y este a su vez le enviará la información al dispositivo final 2. A continuación se describen los pasos necesarios para configurar los módulos de radiofrecuencia.

1. Como primer paso descargar e instalar el software X-CTU de la página del fabricante. Este software permite de manera gráfica y sencilla configurar todos los parámetros de la red.

2. Se procede a conectar el primer módulo a configurar (coordinador), en la pestaña principal (*PC Settings*) del programa X-CTU dar click sobre el boton Test/Query para verificar la conexión del modulo.

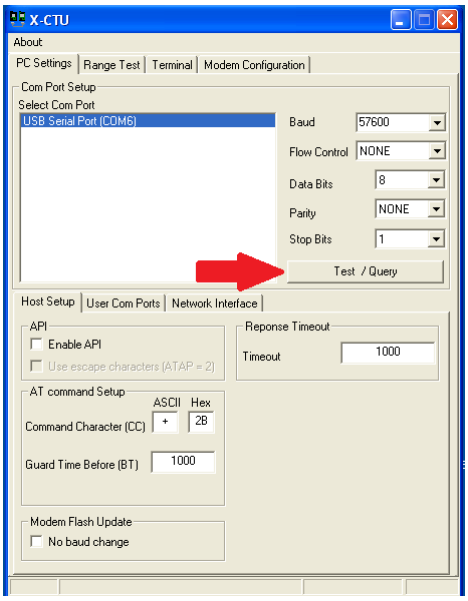


Figura 5.16: X-CTU configuración de PC

Puede generarse un error en caso de que el Baud Rate este mal configurado, intentar con otro hasta tener éxito.

3. Ya asegurada la conexión, en la pestaña Configuración de Modem (*Modem Configuration*) presionar el botón *Read* para leer los parámetros por defecto del módulo.

4. En el menú *Function Set* seleccionar la opción **ZNET 2.5 COORDINATOR AT**, en caso de que no se encuentre la opción verificar que este seleccionado el modo XB24-B en el menú *Modem XBEE*

5. Elegir y escribir un PAN ID arbitrario en la opción correspondiente.

6. Ahora se introduce la dirección del nodo destino, esta dirección es el número de 64 bits que se encuentra impreso en la parte inferior del módulo que servirá como dispositivo final 1.

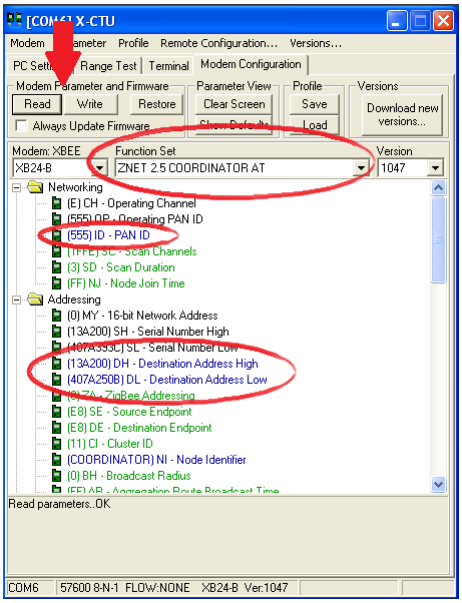


Figura 5.17: Número de dirección del nodo destino

   Este número está escrito en dos partes y deberá ser introducido en las opciones *Destination Address High - DH* y *Destination Address Low -DL* correspondientes.

7. Presionar el botón *Write* para grabar la nueva configuración.

8. Una vez que se ha guardado la información, presionar el botón *Read* y poner atención en el canal generado pues este número nos permitirá unir después a los otros dos dispositivos en la red.

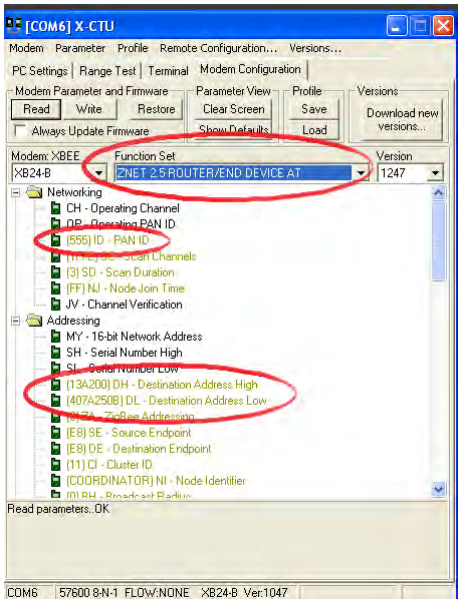


Figura 5.18: Configuracion del modem

Después para configurar los dispositivos finales se realizan los mismos pasos con los siguientes cambios para el dispositivo final 1:

- En el paso 4, seleccionar el *Function Set* con la opción **ZNET 2.5 ROUTER/END DEVICE AT**.
- En la dirección del nodo destino, escribir el número escrito en el dispositivo final 2

Para el último modulo (dispositivo final 2) solo se deberá configurar el PAN ID y el canal. Es de gran importancia asegurarse que los tres módulos tengan el mismo PAN ID, canal y *BaudRate* para lograr la comunicación entre ellos.

Capítulo 6

Resultados de Simulaciones y Pruebas

6.1. Simulación

Se presenta los resultados divididos en las partes que se han mencionado a lo largo de este trabajo. En una primera prueba se observa el comportamiento del sistema antes de tomar el objeto a transportar; aquí se pretende evaluar el algoritmo de navegación y aproximación al objeto.
Por otro lado también se evalúa el comportamiento del sistema al transportar el objeto evadiendo obstáculos; con esta prueba se pretende evaluar el algoritmo de transporte, es decir, la modificación hecha al control por campos potenciales.

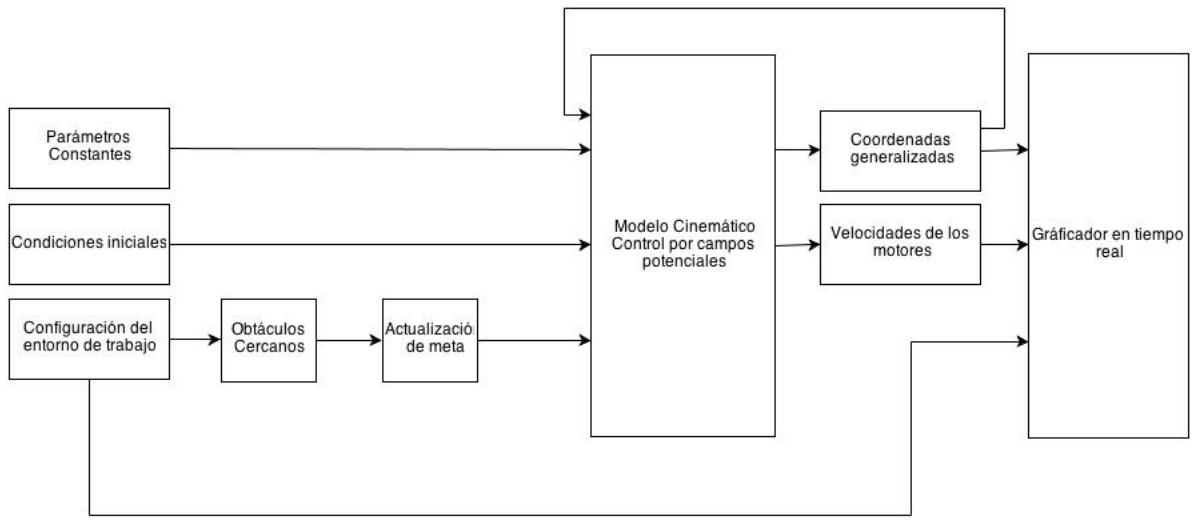


Figura 6.1: Funcionamiento general de la Simulación

Una importante consideración es la referente a los parámetros o restricciones bajos los

cuales se harán las pruebas. Esto se refiere, por ejemplo, a las restricciones del espacio de trabajo en cuanto a la ubicación o configuración de obstáculos y objetos a transportar.Aquí una restricción importante es que no puede haber obstáculos dentro de las trayectorias de aproximación al objetivo, es decir entre los 6 nodos que se generan alrededor del objeto a transportar. A pesar de que los robots cuentan con la capacidad de evadir obstáculos, una vez que empieza a trabar el algoritmo de aproximación, no es posible evadirlos. La segunda restricción importante es que el obstáculo más cercano al destino final del transporte, debe estar a por lo menos una distancia igual a la longitud máxima del objeto a transportar. De lo contrario el objeto podría nunca llegar a su destino por la presencia de un campo repulsivo.

6.1.1. Aproximación al objeto

En esta primera parte se reporta el funcionamiento del algoritmo de navegación y coordinación de los robots, este permite navegar hasta el objeto evadiendo obstáculos, elegir el punto de ensamble libre y más cercano y orientarse correctamente para llegar a él.
Los parámetros iniciales importantes para esta primera parte son:

- Dimensiones del objeto: $l = .4[m]$ $a = .2[m]$
- Dimensiones del espacio de trabajo: 2.5[m] x 2.5[m]
- Radio de los obstáculos: $\rho_r = .1[m]$
- Máxima velocidad de avance: $v_{max} = .1[\frac{m}{s}]$
- Máxima velocidad angular: $\omega_{max} = \frac{\pi}{5}\ [\frac{rad}{s}]$

Estos valores se introducen en el código que genera condiciones iniciales de postura para los robots y el objeto a transportar, la ubicación de los obstáculos y la meta final, se genera un espacio de trabajo como el siguiente:

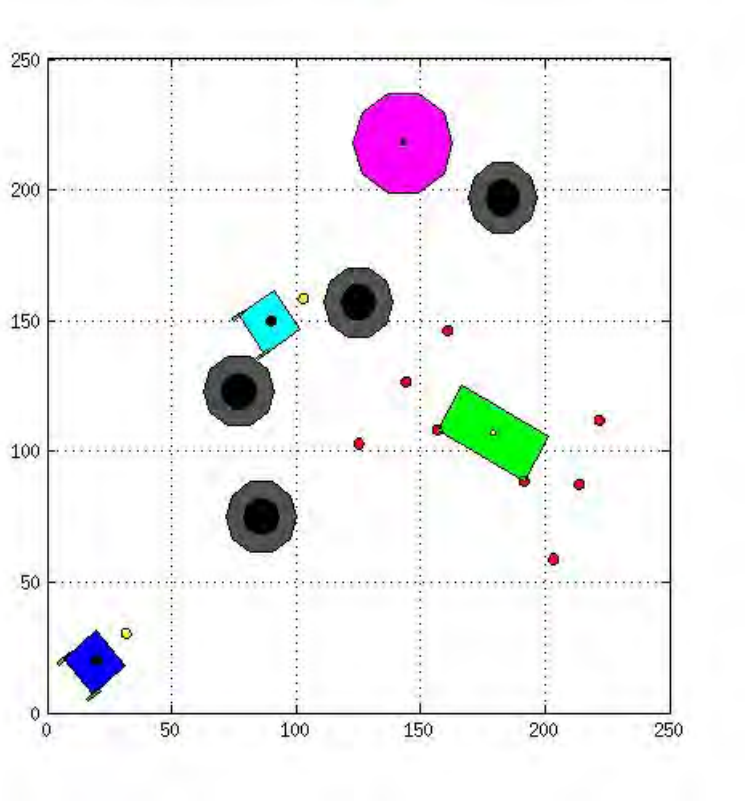
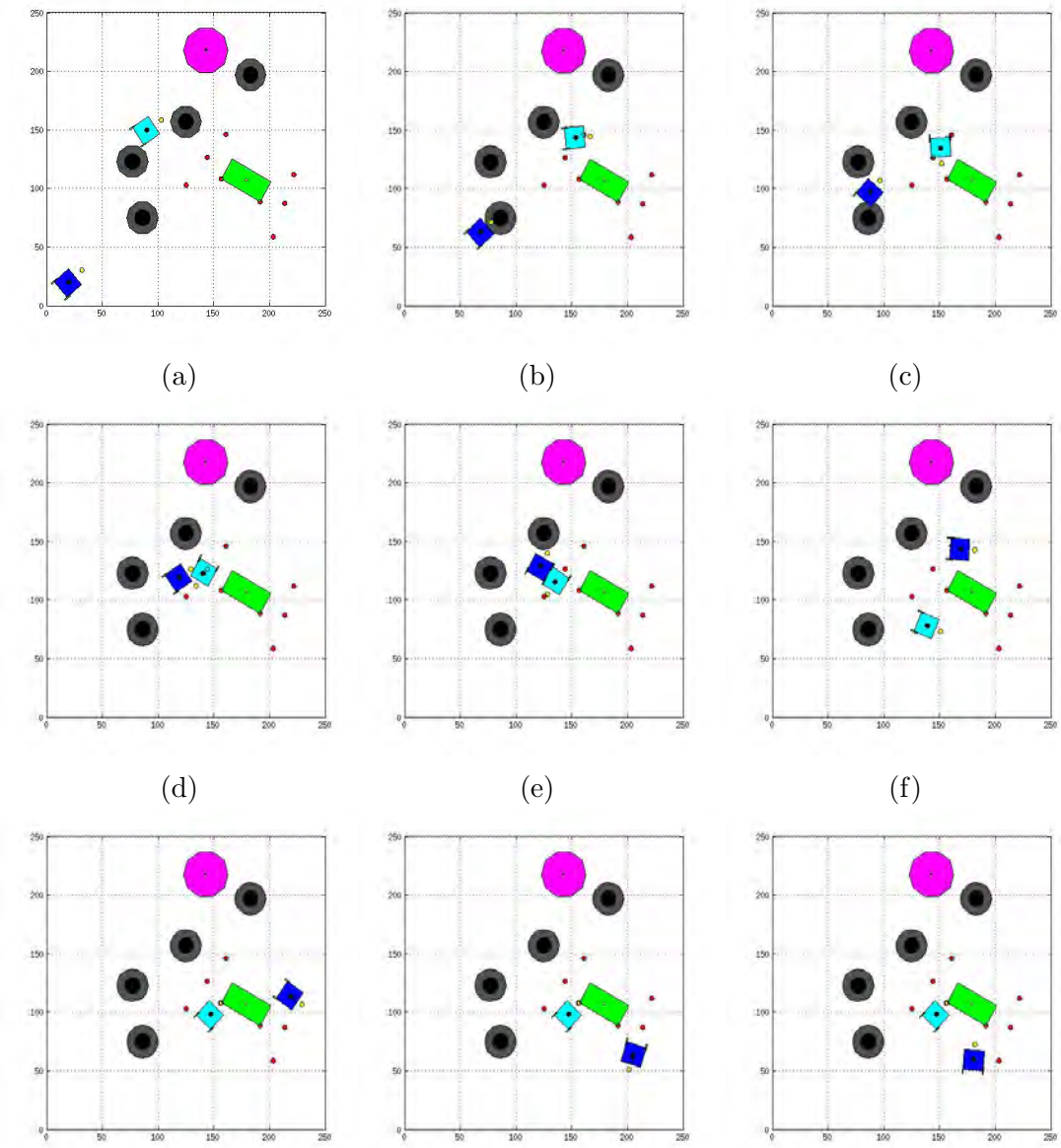


Figura 6.2: Condiciones iniciales en la Simulación

Con esta configuración del espacio de trabajo se busca ilustrar el funcionamiento del algoritmo para llevar a cada robot hasta su punto de ensamble y orientarlo correctamente. En la siguiente secuencia se aprecia el funcionamiento de buena forma:



(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

Figura 6.3: Navegación y aproximación al objetivo

Se puede apreciar en la figura 6.3e como el robot 2 (azul oscuro) evita chocar con el robot 1 (cyan) y sigue con su camino. También se aprecia como, a pesar de que ambos tienen como punto de ensamble el nodo 1 (esquina inferior izquierda del objeto), se coordinan para ir a diferentes puntos. Esto demuestra el comportamiento efectivo de algoritmo de coordinación diseñado. A continuación se presentan algunos gráficos que se generan también la simulación. Para el robot 1:

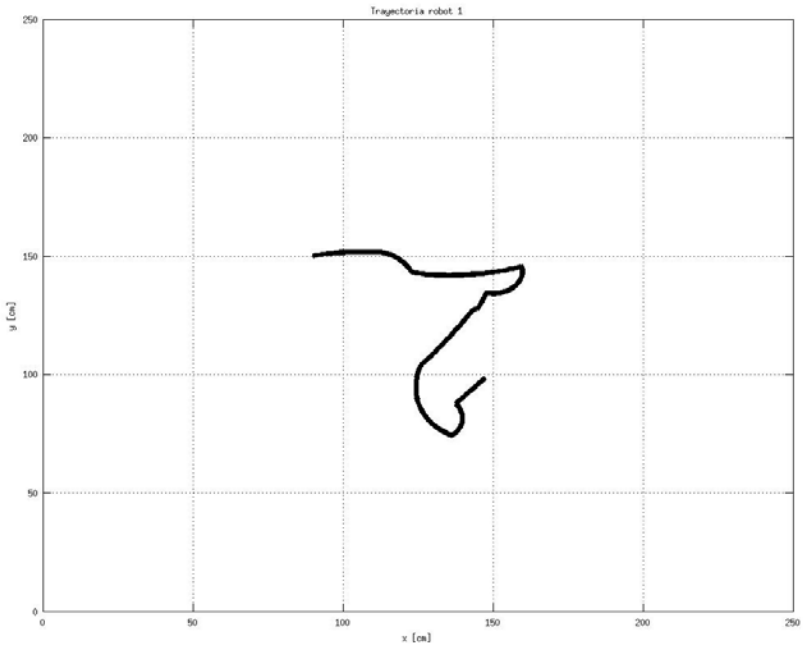


Figura 6.4: Trayectoria del robot 1

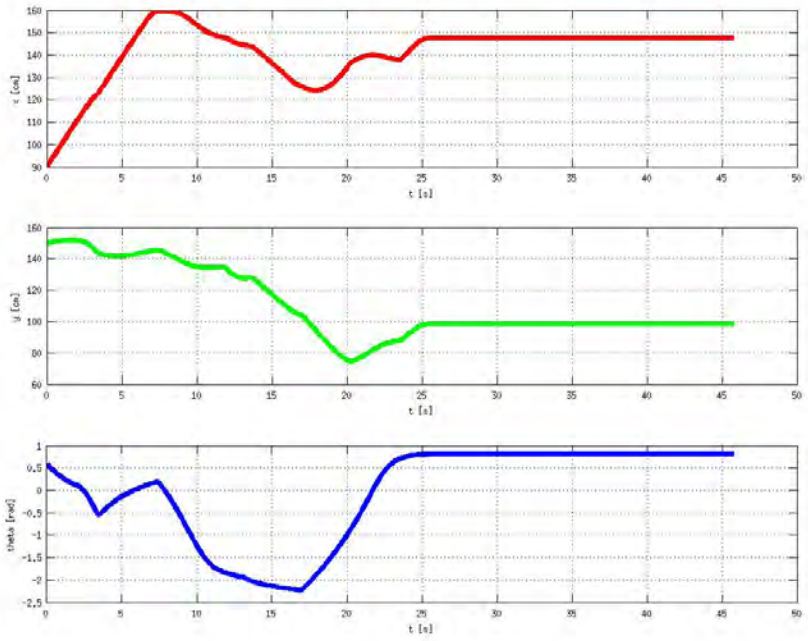


Figura 6.5: Postura del robot 1

Lo mismo para el robot 2:

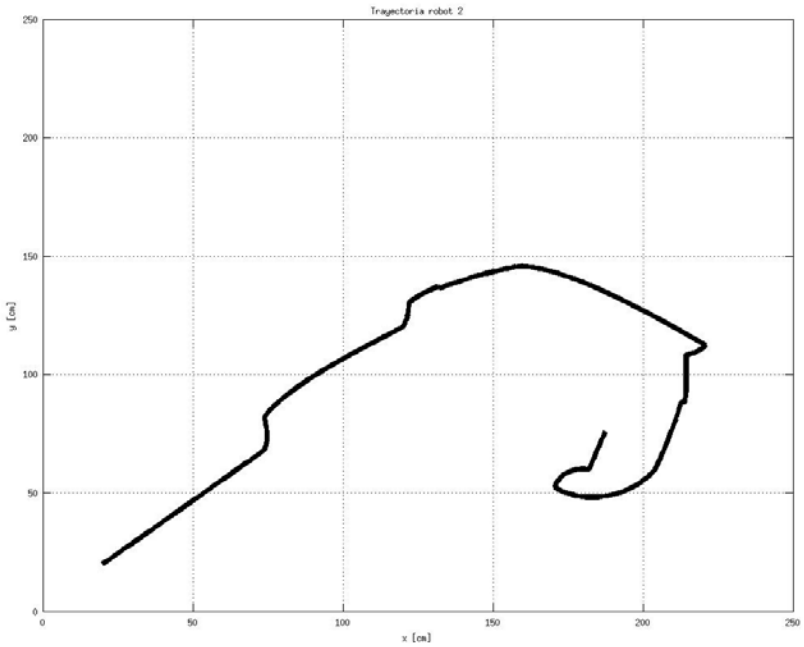


Figura 6.6: Trayectoria del robot 2

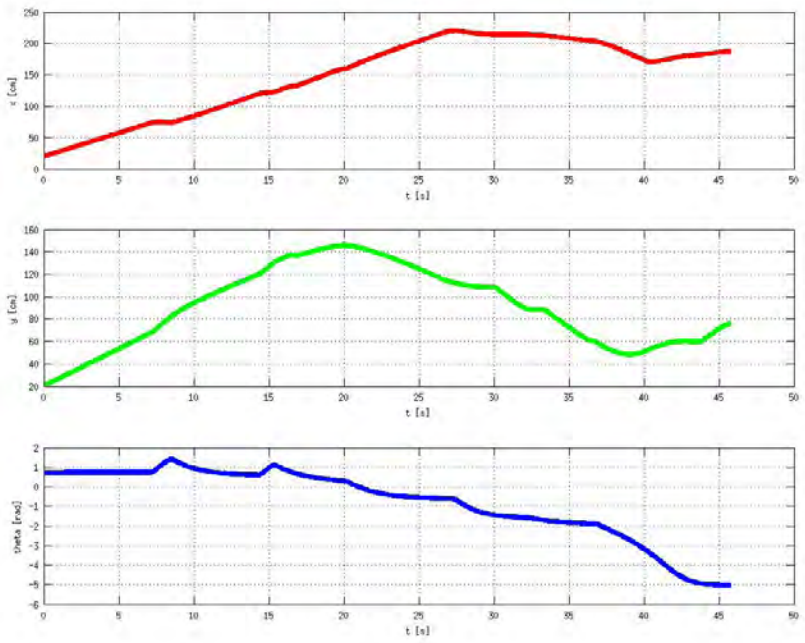
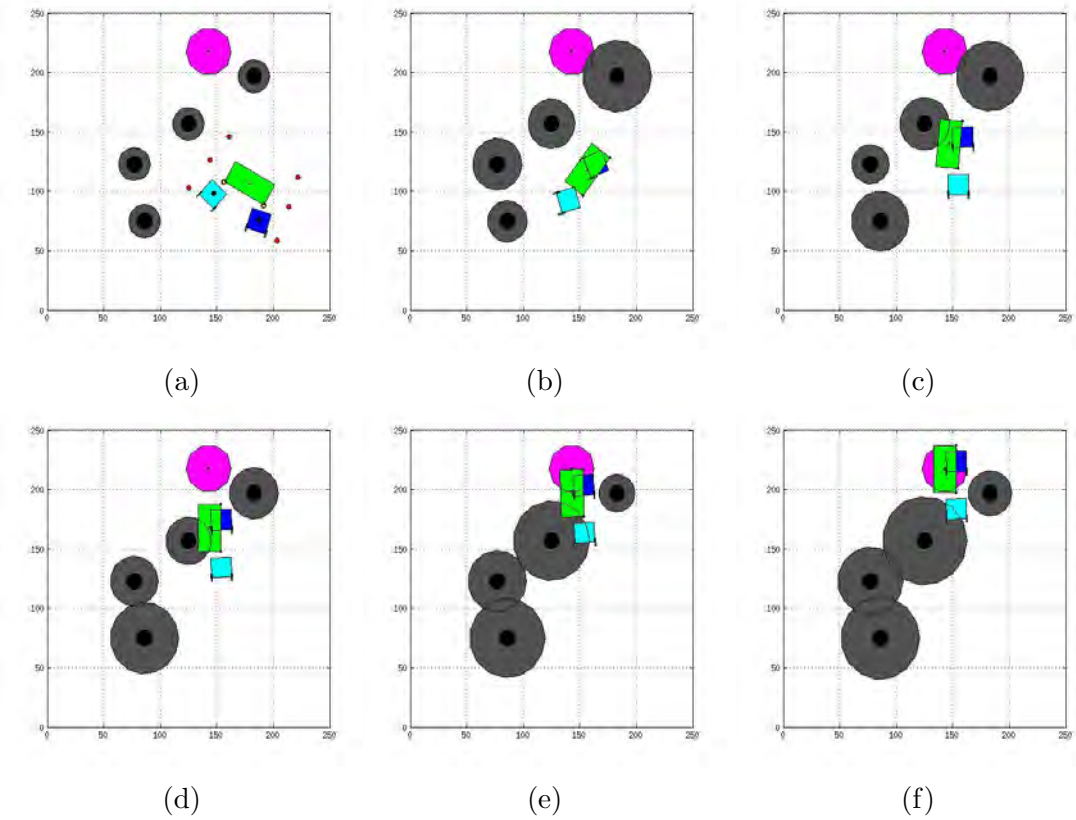


Figura 6.7: Postura del robot 2

La misma prueba se realiza con la segunda forma de coordinación, aquella en la que el robot 1 inicia su trayecto hasta el punto de ensamble para luego dar la señal al robot 2 de poder iniciar su trayectoria. Esta segunda prueba arrojo resultados similares, lo único que cambia es la trayectoria del robot 2. Este robot 2 no tiene la necesidad de evadir a su compañero, únicamente se enfoca evitar colisionar con los obstáculos. Debido a la sencillez de esta prueba no se reportan resultados de ella.

6.1.2. Transporte evadiendo obstáculos

Una de las cualidades del algoritmo que controla este sistema de transporte, es la capacidad de evadir los obstáculos que se encuentren en el camino hacia la meta final. Esta característica, como se explicó en capítulos anteriores, tiene su grado más complejo en el paralelogramo que se está tomando en cuenta para la simulación. Esto se debe a que es bien importante la orientación con la que se transporte el objeto. Gracias a que el planeador conoce en todo momento la ubicación de los elementos, es posible orientar al objeto para transitar por espacios estrechos. La secuencia de posturas de esta parte del algoritmo se muestra en la siguiente figura:



(a) (b) (c)

(d) (e) (f)

Figura 6.8: Transporte evadiendo obstáculos

Gracias al graficador en tiempo real, de donde se han obtenido las imágenes mostradas hasta ahora, es posible observar la variación en el radio de influencia del campo repulsivo de los objetos. Esta variación, como se comentó con anterioridad, depende del ángulo del objeto y del ángulo de este respecto a los obstáculos. Se presenta a continuación la información la variación de los campos repulsivos de cada objeto a lo largo del transporte.

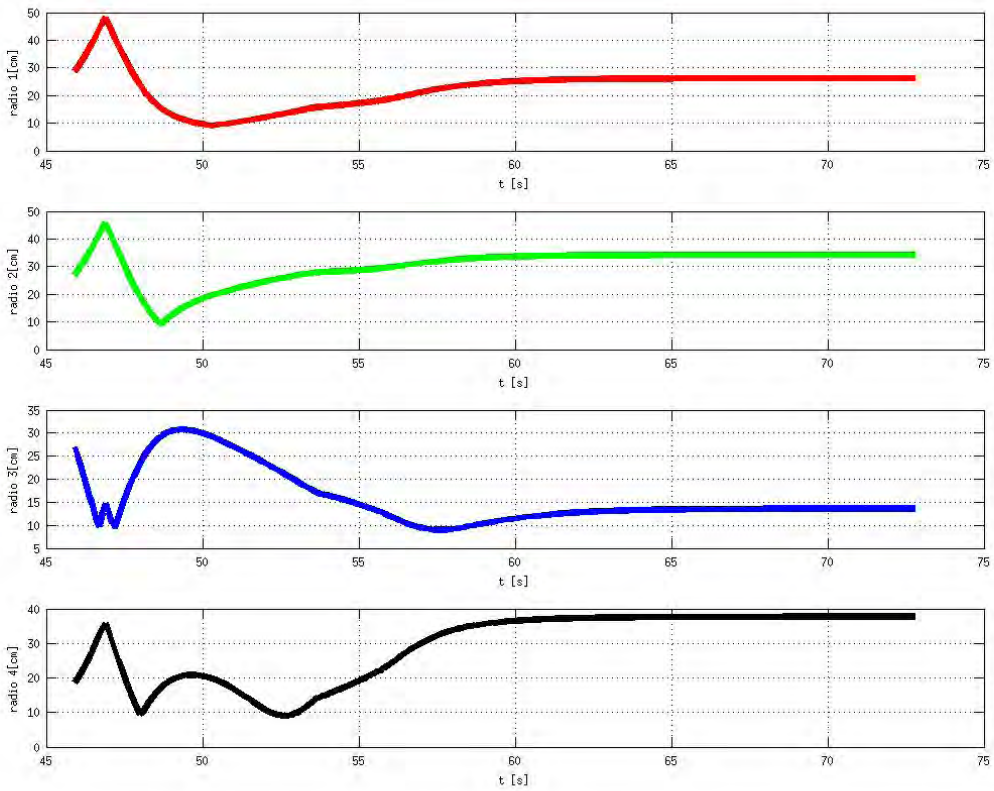


Figura 6.9: Variación en el radio de campos repulsivos

Es también importante observar información sobre la postura de la configuración utilizada para el transporte, de forma que también se generan las siguientes gráficas con la información obtenida.

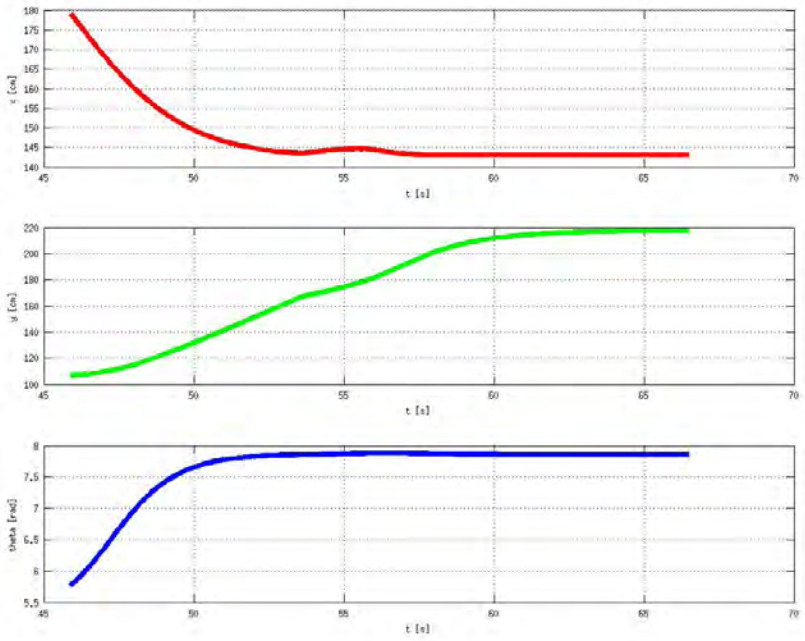


Figura 6.10: Postura del objeto durante el transporte

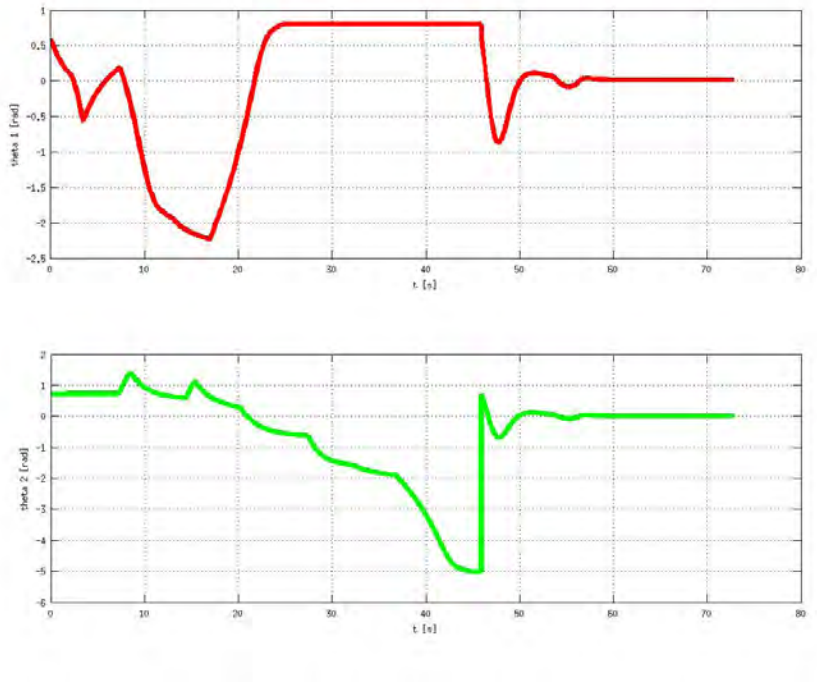


Figura 6.11: Orientación de los robots durante el transporte

Todos los parámetros y condiciones iniciales anteriores fueron establecidos mediante la ayuda del código de un programa en Matlab. Este código aprovecha la posibilidad en Matlab de generar números pseudo aleatorios, así se generan posiciones y orientaciones de robots, objeto a transportar, meta final y ubicación de obstáculos. El programa mencionada se puede encontrar en la sección de apéndices.

Para dejar más claro la importancia de la orientación en el transporte, se generan ciertas condiciones para que el entorno de trabajo permita demostrar del algoritmo. Resulta evidente que esta configuración no se genera con el programa mencionado recientemente, sino condiciones iniciales específicas para esta demostración gráfica.

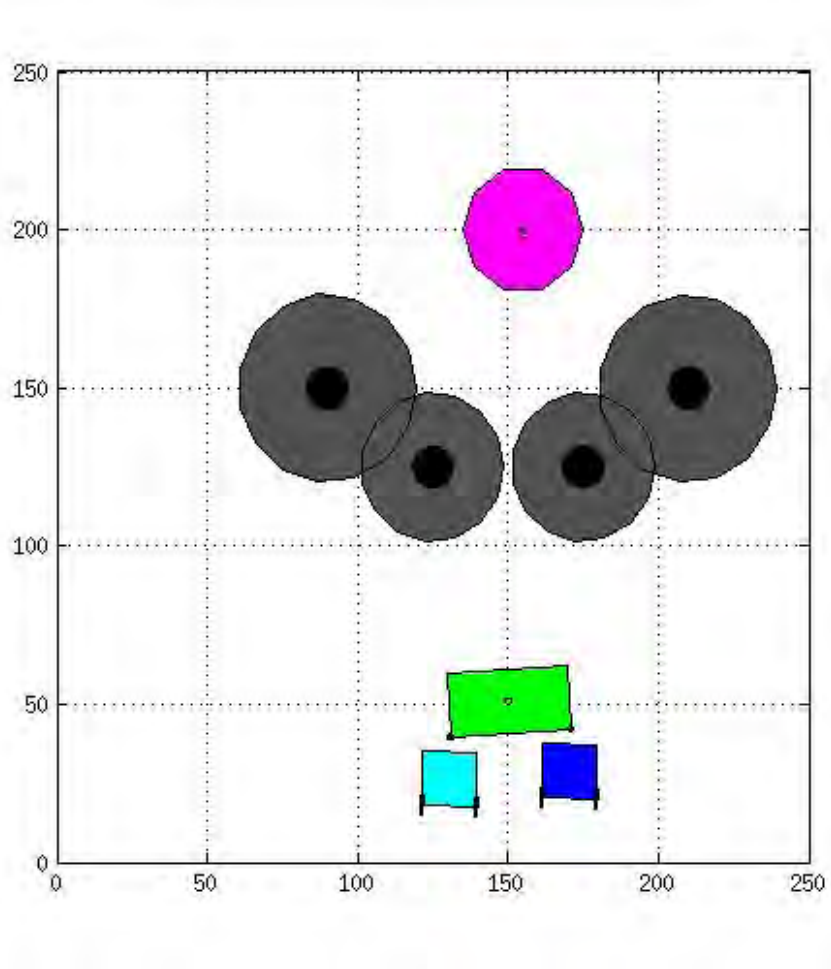
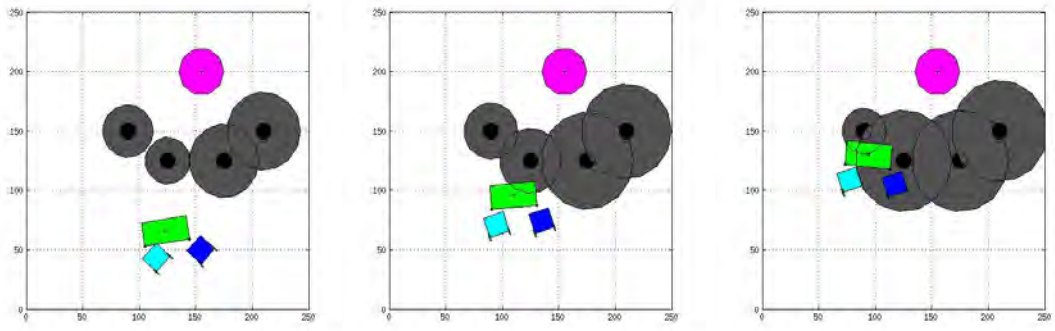
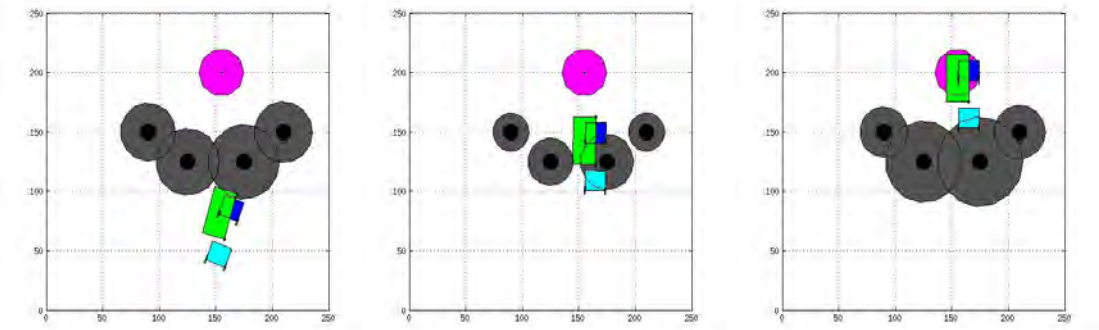


Figura 6.12: Entorno de trabajo con condiciones adversas para el transporte



(a) (b) (c)

Figura 6.13: Transporte sin considerar orientación



(a) (b) (c)

Figura 6.14: Transporte considerando orientación

En estas condiciones iniciales de transporte existe solo un camino considerado posible, pasar por el centro de los obstáculos 2 y 3 (de izquierda a derecha). Esto bajo la restricción de que no sea pertinente transitar por la parte externa a estos obstáculos. De forma que, si se suprime el código que permite al planeador global encontrar una trayectoria variando la orientación, no es se encuentra una solución al transporte.

6.2. Modelo Funcional

6.2.1. Entorno de la prueba

El experimento para corroborar el funcionamiento del sistema de transporte se llevó a cabo en el laboratorio de Mecatrónica. Este espacio cuenta con una lugar reservado para la experimentación en robótica móvil y cuenta con el instrumental necesario para realizara dichos experimentos. En el lugar mencionado se encuentra un arnés para colocar una cámara de video en el techo, el cual se encuentra a una altura aproximada de 2.30[m].

6.2.2. Instrumental

Sin considerar a los robots móviles y a lo descrito en el capítulo anterior, consiste en la computadora sobre la que corre el planeador global, la cámara para el sistema de visión y el cable con extensión para conectar estos elementos. La computadora utilizada es un computadora Laptop, modelo Sony Vaio VPCEG23EL corriendo un sistema operativo Ubuntu 12.04 (Precise Pangolin). Este equipo cuenta con un procesador Intel Core i3-2330M (2.20 GHz) y una memoria RAM DDR3 de 4 GB.
Por otro lado la cámara de video utilizada para el sistema de visión(reacTIVision) es una cámara web LifeCam Studio de Microsoft ®, la cual cuenta con una conexión USB 2.0 y una resolución máxima de 1920x1080 pixeles. No obstante debido al controlador de video utilizado por el sistema operativo se utiliza la resolución de 640x480. Finalmente, también

se utiliza una extensión USB para conectar la cámara a la computadora, pues al colocar la cámara a la altura mencionada resulta necesario más cable.

6.2.3. Consideraciones Adicionales

Una consideración importante es que a lo largo del trayecto de los robots, el análisis y la planeación de trayectorias se hacen con respecto al punto P en el centro del robot; sin embargo para realizar el enganche con el objeto el análisis se debe realizar con dicho punto proyectado o trasladado a la argolla en donde se ensambla el robot al objeto. Para ello se hace una proyección simple utilizando el angulo θ_R del robot:

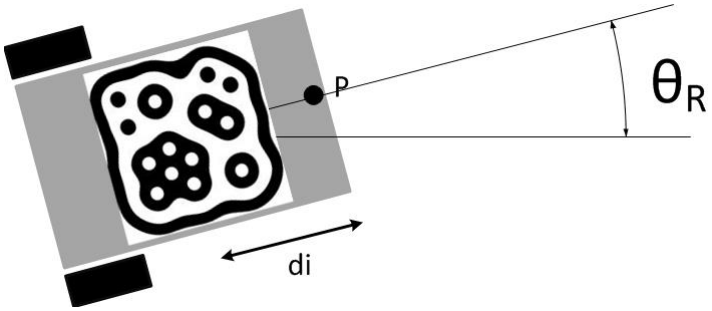


Figura 6.15: Ajuste de posición para enganche

$$X_R = X_V + di\cos(\theta_R)$$
$$Y_R = Y_V + di\sin(\theta_R)$$

En donde (X_R, Y_R) son las coordenadas de posición del dispositivo de enganche y (X_v, Y_v) es la posición del robot obtenida del sistema de visión.
Un segundo aspecto a considerar es que, debido a que el *fiducial* para reconocer a los robots y al objeto a transportar se encuentra en la parte superior de estos; los datos provenientes de la visión corresponden a un plano paralelo al piso, siendo este último donde se realiza la calibración. Además, debido al funcionamiento de la cámara, entre más lejos se encuentren los *fiducuals* del foco mayor distorsión tendrán. Lo anterior se traduce en un valor erróneo de posición y se observa en la siguiente figura

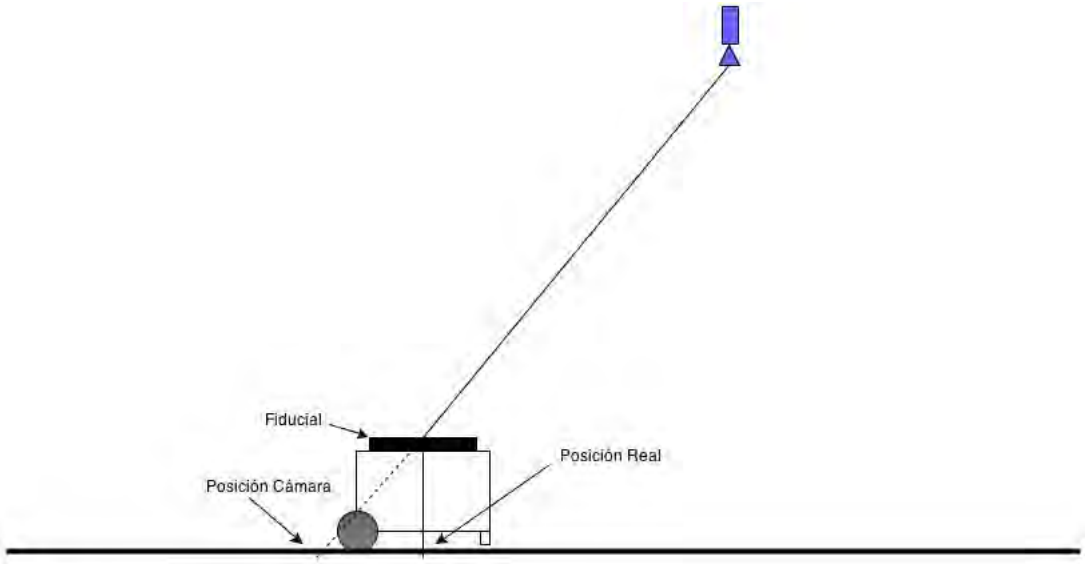


Figura 6.16: Ajuste de posición por deformación

Se aprecia que la posición real del *fiducial* es una proyección del dato obtenido de reacTIVision sobre el plano del piso. Este ajuste se realiza de la siguiente forma:

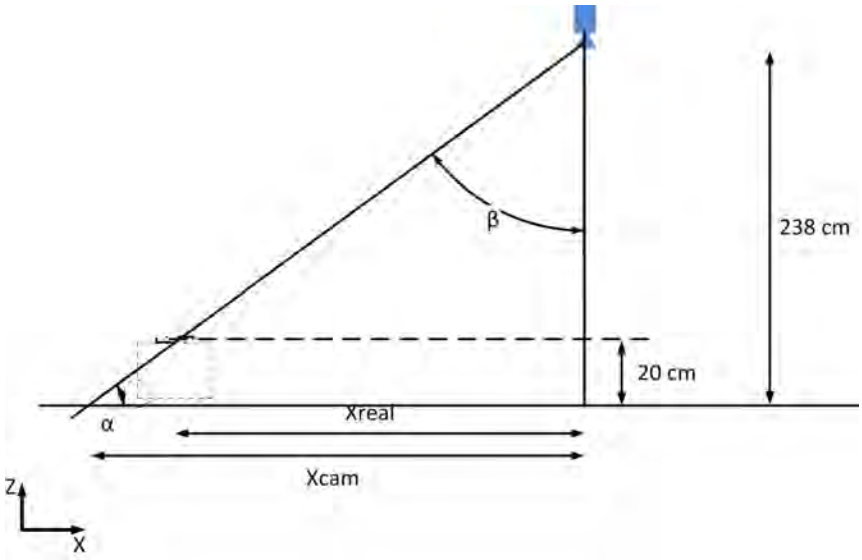


Figura 6.17: Correción de coordenadas de cámara plano XZ

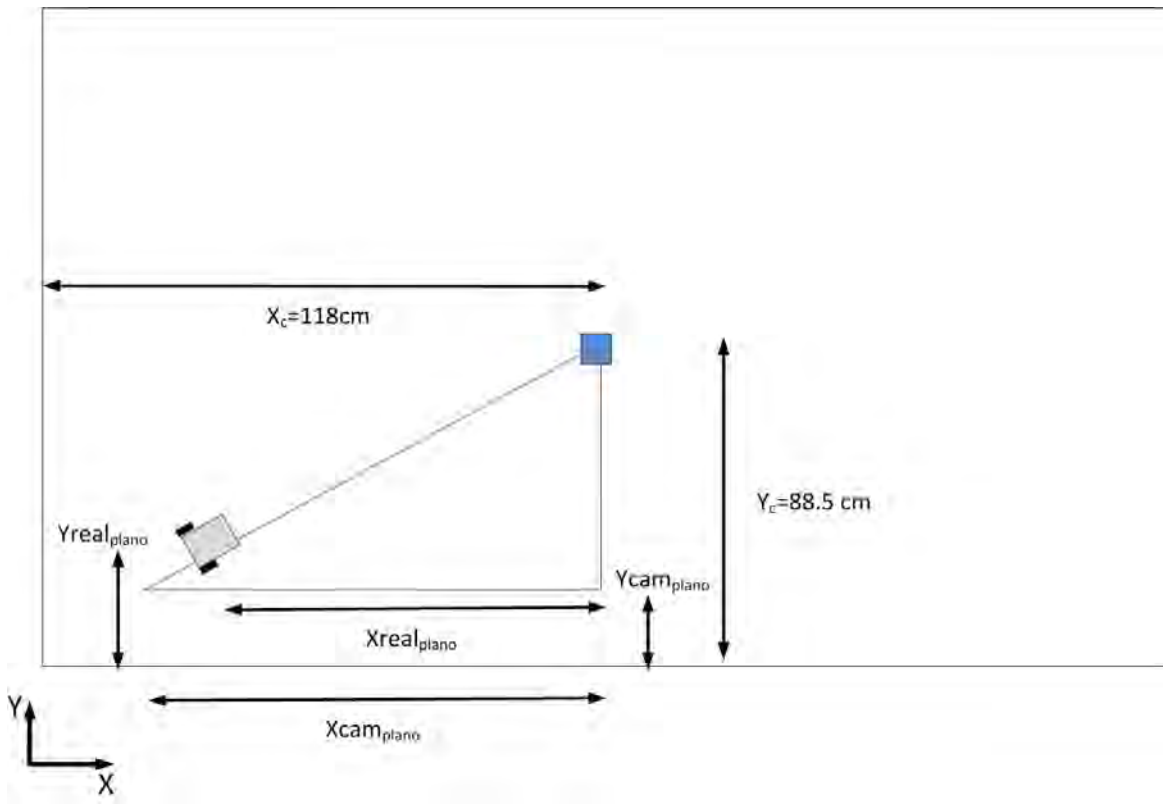


Figura 6.18: Correción de coordenadas de cámara plano XY

De la figura 6.17 se tienen las siguientes relaciones:

$$\frac{Xcam}{Xreal} = \frac{238}{238 - 20}$$

Y de la figura 6.18

$$\frac{Xcam_{plano}}{Xreal_{plano}} = \frac{Y_c - Ycam_{plano}}{Y_c - Yreal_{plano}}$$

De donde

$$Xcam_{plano} = Xcam \cos\eta$$

$$Xreal_{plano} = \|Xcam\| \cos\eta$$

En donde η representa al ángulo que hay entre las coordenadas de reacTIVision y las coordenadas del centro del espacio de trabajo (foco de la cámara).

También es pertinente aclarar que el planeador global que corre sobre SIMULINK funciona igualmente que la simulación, con la diferencia de que los datos de postura de los robots no se realimenta del modelo; se obtiene a partir del sistema de visión. Por otro lado se elimina el bloque graficador pues alenta el funcionamiento del programa.

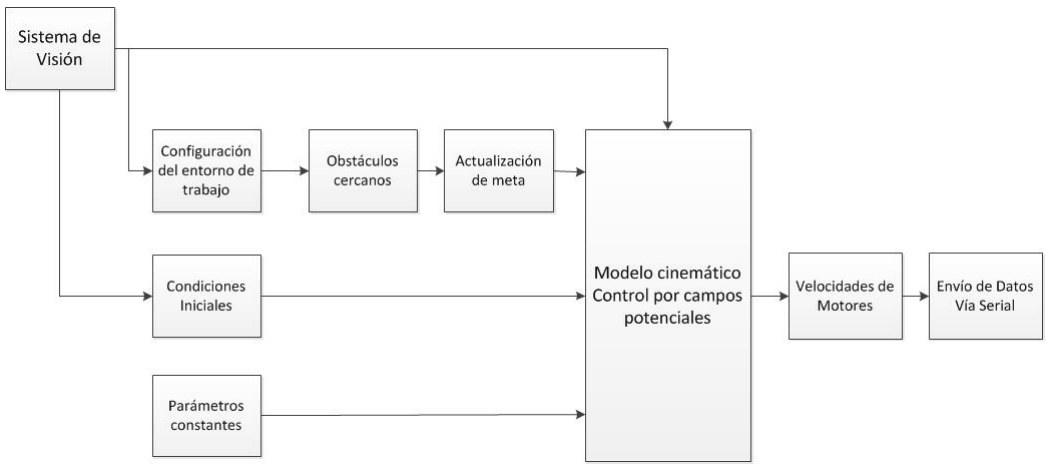


Figura 6.19: Funcionamiento Control en Tiempo Real

6.3. Experimentos

Para corroborar el funcionamiento y observar el comportamiento del sistema de transporte colaborativo se realizan los experimentos pertinentes. Además es importante mencionar que, dadas las dimensiones del entorno para pruebas, no es posible realizar configuraciones como las que se reportan en las simulaciones. Debido al sistema de visión empleado no se prueba el caso de ambos robots desplazándose al mismo tiempo, en el siguiente capítulo parte de este trabajo se discutirá más al respecto. Es así que son dos los experimentos a reportar:

- En el primer experimento se coloca el objeto a transportar y los robots en posiciones arbitrarias. Con esta prueba se busca comprobar la capacidad de los robots aproximarse al objeto,orientarse y engancharlo.

- En el segundo, se colocan dos objetos que fungirán como obstáculos. En esta prueba se podrá observar el comportamiento de los robots ante la presencia de obstáculos en el camino antes y después de tomar el objeto.

6.3.1. Primer Experimento: Aproximación al objeto

Por las condiciones antes mencionadas para los experimentos, se elige una configuración sencilla pero que permita ver el comportamiento del algoritmo para navegación y aproximación, esta configuración es:

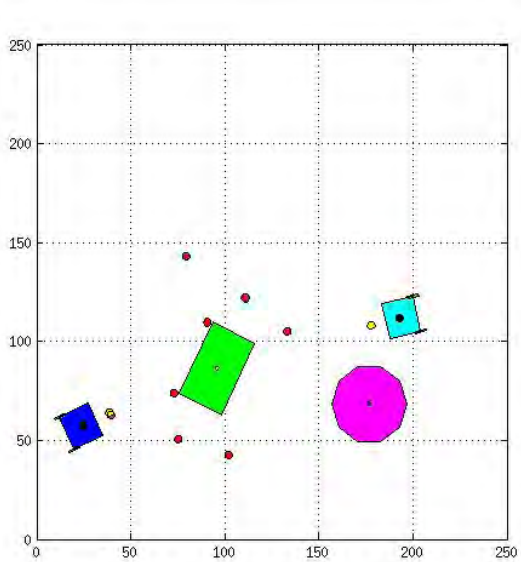
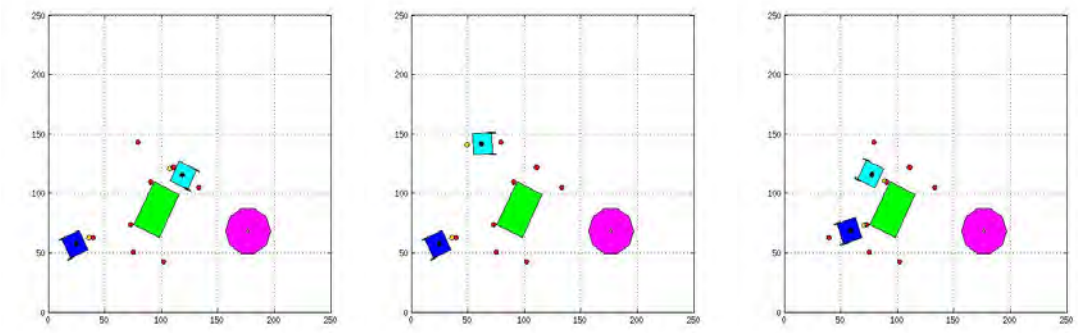


Figura 6.20: Condiciones iniciales experimento 1

Para tener una referencia se ejecuta la simulación con las condiciones iniciales del experimento antes de llevarlo a cabo con el sistema funcional. Estás condiciones iniciales son:

$$\begin{array}{lll} x_p = .95[m] & x_1 = 1.92[m] & x_2 = .246[m] \\ y_p = .86[m] & y_1 = 1.11[m] & y_2 = .576[m] \\ \theta_p = 4.25[rad] & \theta_1 = 3.37[rad] & \theta_2 = .434[rad] \end{array}$$

Esto valores son obtenidos del sistema de visión, el programa utilizado se encuentra en la sección de apéndices bajo el nombre de ExperimentParameters.m . Aquí los *fiducials* con ID 6 y 7 representan a los robots 1 y 2 respectivamente. El *fiducial* con ID 0 representa al objeto a transportar. Al ejecutar la simulación se obtienen el siguiente resultado



(a) (b) (c)

Figura 6.21: Simulación de la configuración del experimento 1

Una vez que se ha observado el comportamiento de los algoritmos de navegación y aproximación se procede a llevar a cabo el experimento. A continuación se presentan los resultados obtenidos.

(a) Condición inicial del experimento 1 (b) Robots orientados

Figura 6.22: Experimento 1

Los robots siguen la trayectoria adecuada para cada caso. Las trayectorias seguidas por los robots son:

Figura 6.23: Trayectoria robot 1

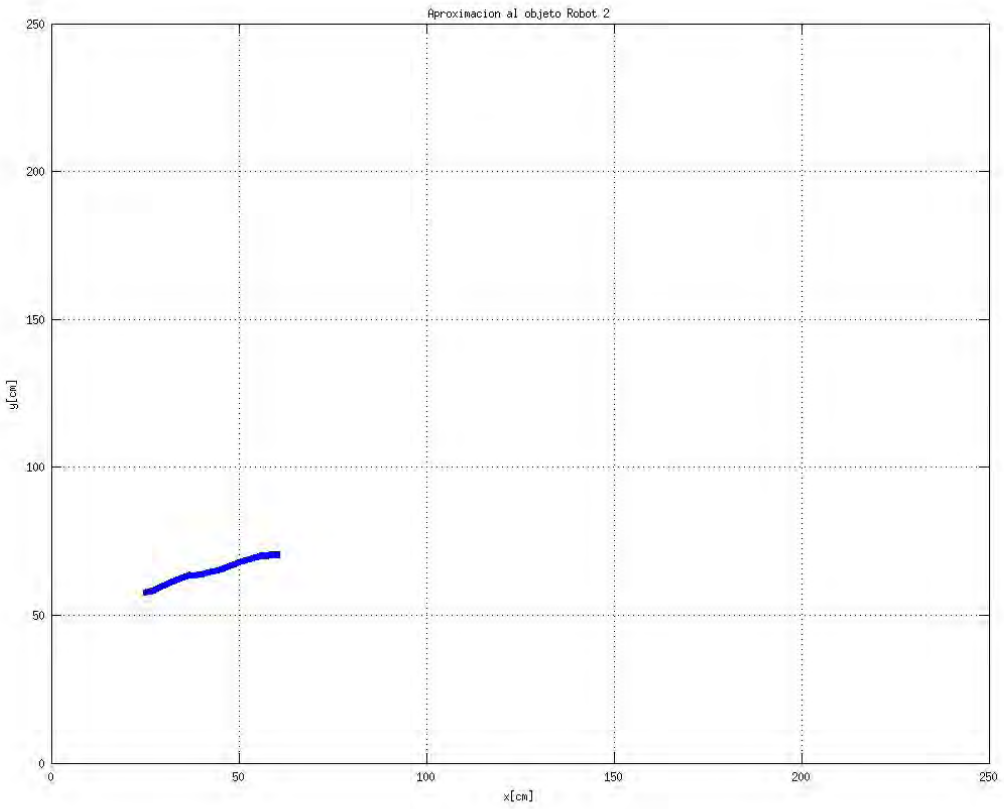
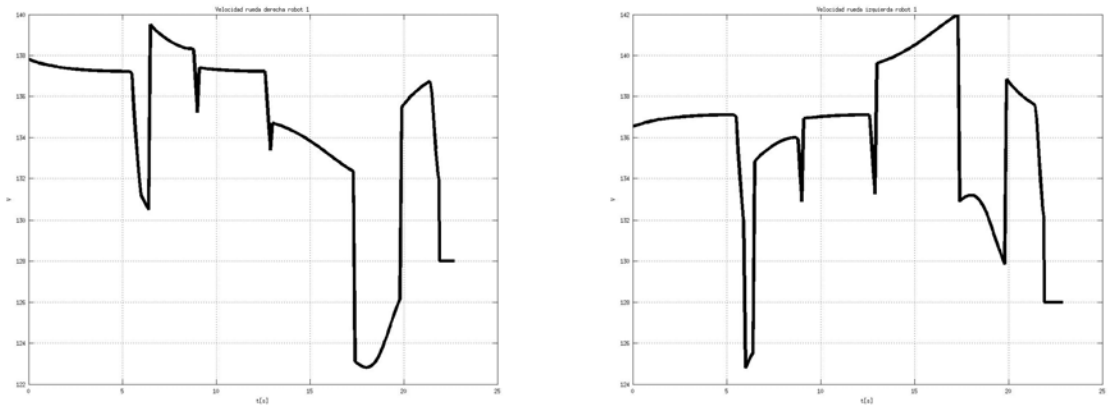


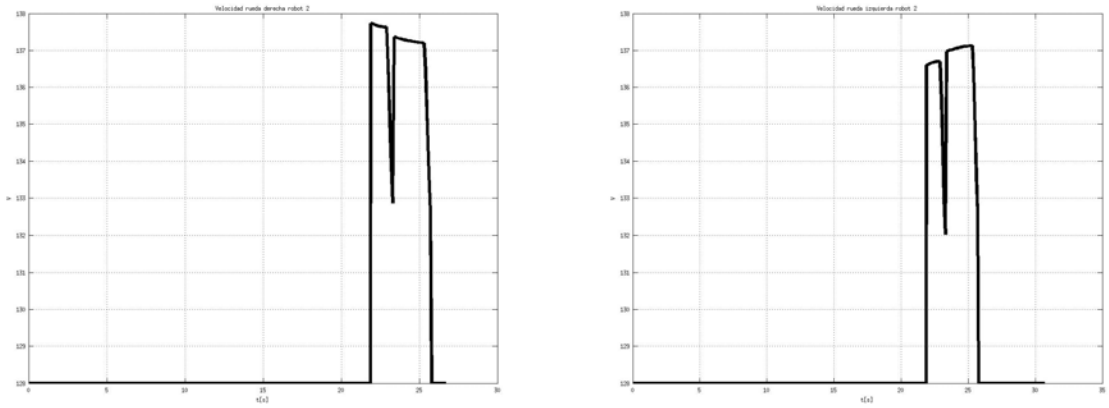
Figura 6.24: Trayectoria robot 2

También se muestran los datos de velocidad enviada a los motores de los robots. En las gráficas se aprecia el correcto escalamiento de los datos y la transformación de rad/s a números enteros de 0 a 255 entendidos por la tarjeta MD25.



(a) Derecha (b) Izquierda

Figura 6.25: Velocidades de motores en robot 1



(a) Derecha (b) Izquierda

Figura 6.26: Velocidades de motores robot 2

Durante la realización del experimento se notó que había problemas cuando ambos robots se movían simultáneamente. En prácticamente todas las ocasiones había saltos o valores erróneos en la información, haciendo que reacTIVision no viera a los robots y esto a su vez provocaba el envió de valores equívocos de velocidad para los motores. Por esta razón se realizó el experimento moviendo un robot a la vez. Esto no es problema para el algoritmo, pues este tipo de coordinación tenia considerada como opcional en el algoritmo desarrollado. Aquí el robot 1 hace su recorrido y una vez que ha llegado a su meta envía la señal de inicio al robot 2.

6.3.2. Segundo Experimento: Transporte evadiendo obstáculos

El segundo experimento a realizar será la tarea de transporte en si, es decir, desde el momento en que los robots se encuentran ensamblados al objeto hasta que lo llevan a su meta final. Como en el experimento anterior se restringen las condiciones del espacio de trabajo. Además, el mecanismo de ensamble empleado limita en gran medida el movimiento de los robots, por lo que el experimento realizado en para observar el comportamiento del sistema se limita un transporte simple con dos obstáculos. Las condiciones iniciales de este experimento son:

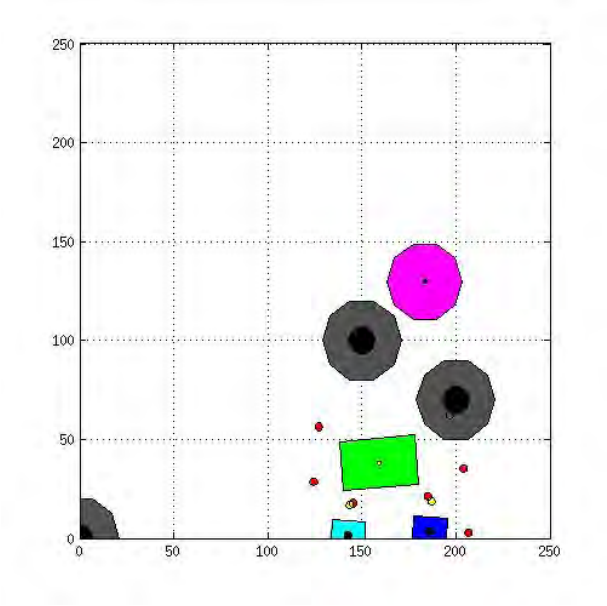
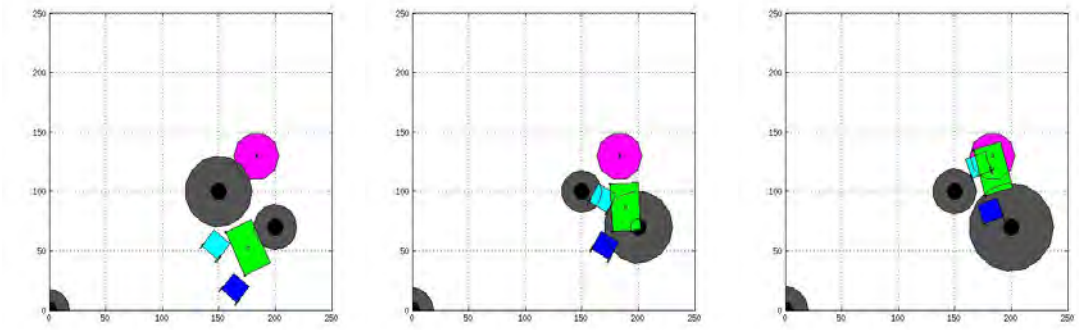


Figura 6.27: Condiciones iniciales experimento 2

Al igual que en el experimento anterior, con el sistema de visión se obtienen la configuración del entorno de trabajo; a partir de ahí se prueba la simulación antes de realizar el experimento.



(a) (b) (c)

Figura 6.28: Simulación del entorno para experimento.

Una vez comprobado el funcionamiento en la simulación se procede a realizar el experimento. Los resultados se muestran a continuación:

(a) Condición inicial del experimento 2 (b) Resultado del transporte

Figura 6.29: Experimento 2

Los *fiducials* con ID 1 y 2 son empleados como obstáculos. El *fiducial* con ID 5 se utiliza como meta final del objeto. Al igual que el caso anterior los *fiducial* 6 y 7 representan a los robots y el número 0 al objeto a transportar.

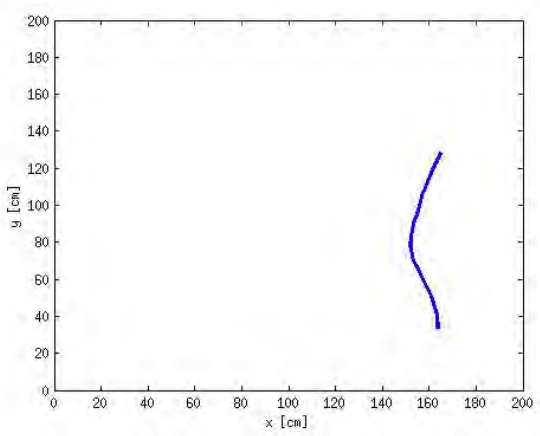


Figura 6.30: Trayectoria del objeto durante el transporte

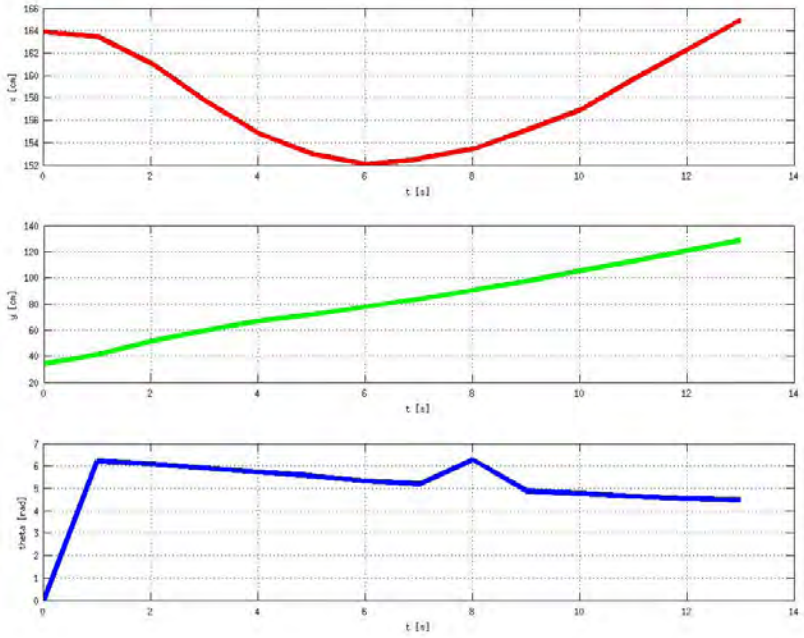


Figura 6.31: Postura del objeto durante el transporte

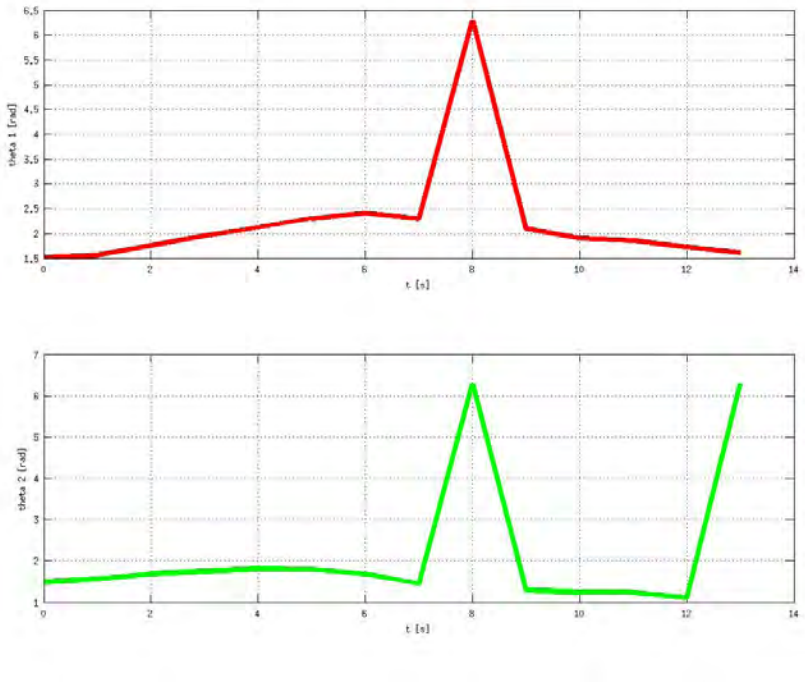


Figura 6.32: Orientación de los robots durante el transporte

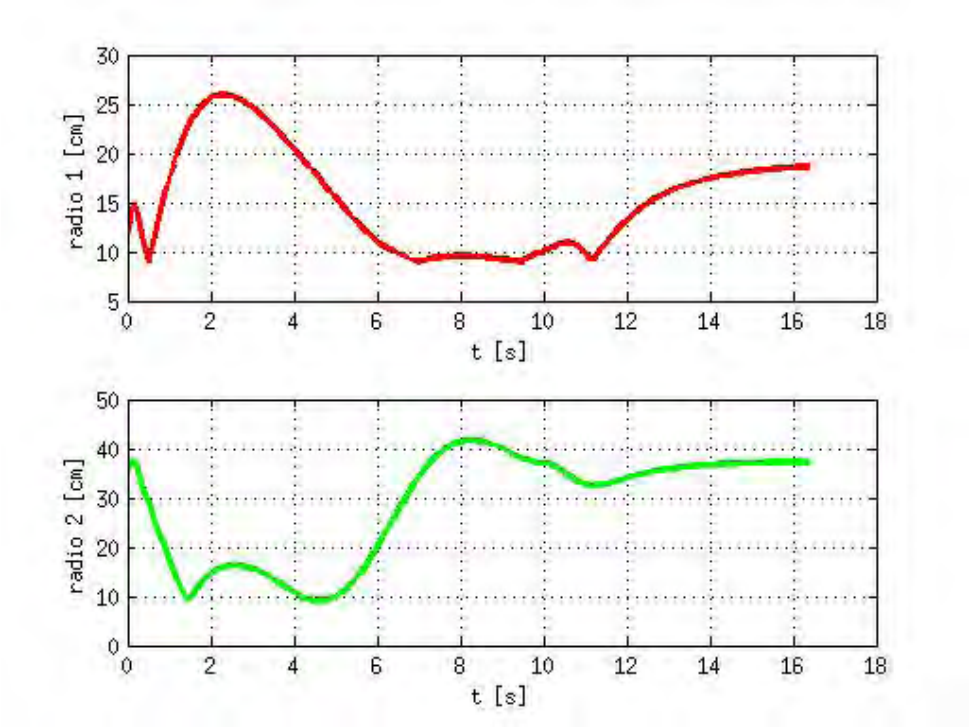


Figura 6.33: Radio de campo repulsivo de los objetos durante el transporte

Una observación de los resultados de este experimento es que el giro de los robots está limitado por el mecanismo de enganche. Para ejecutar el transporte con cierta orientación el robot debe tomar ángulos que el modelo físico no permite. Por otro lado, a pesar de que el sistema calcula de forma correcta los valores del campo repulsivo en el control por campos potenciales, en la implementación física del sistema es necesario sintonizar algunas ganancias correctamente, pues a pesar de que el cálculo de campos repulsivos y atrayentes es correcto no se logra la evasión esperada.
Desde luego que la comparación se hace con la simulación en dónde todos los parámetros son ideales. En cambio con el modelo funcional existen diversos factores que alteran el comportamiento esperado del sistema. Dentro de los factores más importantes están las condiciones mecánicas (ensamble) y la adquisición de datos (visión).

Capítulo 7

Conclusiones y Trabajo Futuro

7.1. Conclusiones

Con este trabajo, se probó que la teoría desarrollada por González Villela en [20] junto con el trabajo desarrollado en [21] funcionan efectivamente para desarrollar sistemas robóticos colaborativos. Se comprobó que el modelo cinemático es suficiente para planear y ejecutar las trayectorias en el transporte de objetos con formas geométricas regulares ponderando su orientación. El algoritmo implementado en los robots les permite, gracias a los puntos de *pre-docking*, navegar y orientarse correctamente para tomar al objeto a transportar. El modelo de evasión y transporte funciona correctamente cuando hay un cálculo correcto de los datos de entrada.

Gracias al sistema diseñado es posible simular diferentes situaciones o configuraciones del entorno de trabajo y ver como el algoritmo resuelve el transporte. Además, con el programa generado para establecer las condiciones iniciales, es posible tomar estos parámetros de una configuración real o experimental. Esto trae beneficios como el poder observar el comportamiento del sistema antes de realizar experimentos, incluso tener una referencia para hacer comparaciones de desempeño.

El sistema de visión empleado presentó problemas referentes a los tiempos de muestreo, cuando se agregan más *fiducials* y estos se mueven por el entorno de trabajo, el tiempo necesario para localizarlos correctamente aumenta considerablemente; inclusive en ocasiones se pierden o se altera los datos. Otra condición con gran influencia sobre el funcionamiento de este sistema es la iluminación, pues la cámara trabaja más lento cuando hay poca luz. Una consecuencia importante es la utilización de bajas velocidades en el desplazamiento de los robots y limitar el número de objetos con símbolos en el entorno. A pesar de haber diseñado el sistema y hecho simulaciones con velocidad máxima de $.1[\frac{m}{s}]$, al momento de hacer el experimento esta se redujo a $.03[\frac{m}{s}]$. Además, el *hardware* utilizado limitó en gran medida tanto el espacio de trabajo como la resolución para la captura de datos.

En el primer experimento se corroboró, a pesar de los problemas con el sistema de visión y las restricciones del espacio de trabajo, que el algoritmo de navegación junto con el de aproximación al objetivo funcionan correctamente. Dicho algoritmo es capaz coordinar y planear las trayectorias de ambos robots para llegar a los puntos de enganche con la orientación adecuada. A partir de los resultados de la primera prueba se observa como los robots se aproximan al punto de pre-docking más cercano, a partir de ahí navegan hacia el punto libre para ensamble más cercano si este se encuentra libre.

En un principio el sistema se planteó para ser completamente autónomo, sin embargo, debido al diseño del mecanismo de ensamble y a las limitantes de la visión , se vio en la necesidad de hacer un ajuste o ensamble manual de los robots al objeto. No obstante, suponiendo la existencia de una solución comercial para este mecanismo, el sistema de transporte colaborativo puede funcionar correctamente. Esta fue la razón principal por la que se reportan los experimentos de forma separada y con configuraciones simples en el entorno de trabajo. El mecanismo de ensamble implementado limita en gran medida el movimiento de los robots. Primero porque físicamente el giro tiene como límite 0 y π radianes respecto al objeto a transportar. Además, por la ubicación de los *fiducials* sobre los robots, en el momento de que estos toman una orientación mayor a $\frac{5\pi}{6}$ o menor a $\frac{pi}{6}$ radianes respecto al objeto, el sistema de visión no es capaz de obtener la información sobre ellos debido a que parte del *fiducial* se oculta.

A pesar de funcionar de forma correcta, el *software* que se implementó para el control (planeador global) de los robots, presentó algunos problemas como el tiempo de procesamiento. Para resolver los modelos cinemáticos, la planeación de las trayectorias y demás cálculos, Simulink toma demasiado tiempo; desde luego todos estos cálculos se realizan con gran precisión y con pocas adecuaciones el mismo modelo se puede utilizar para el control en tiempo real. Este problema, aunado al comentado del sistema de visión, limitó los tiempos de muestreo a entre 4 y 7 Hz; que cuando se compara con trabajos similares resulta aceptable, pues en la literatura encontrada se reportan muestreos entre 7 y 20 [Hz], siendo que en esos trabajos se implementan *hardware* dedicado (visión, procesamiento, comunicación) y otras tecnologías de alto costo.

7.2. Trabajo Futuro

En este trabajo se demostró el funcionamiento del algoritmo generado para coordinar, planear trayectorias y transportar objetos mediante dos robots móviles; todo desde el punto de vista cinemático. Una primera tarea para desarrollar en el futuro es contar con un modelo dinámico y observar el comportamiento del algoritmo.
Ligado a lo anterior y a manera de complemento, sería interesante desarrollar y aplicar

una metodología de diseño para obtener especificaciones como materiales, dimensiones y componentes idóneos para transportar determinados objetos o materiales. Un ejemplo de esto sería obtener el par necesario (motores) para transportar una masa o volumen específico. Resalta también, de manera especial, diseñar el mecanismo de ensamble o enganche ideal para cada caso.

Como se planteó en un principio, los sistemas multi robot tiene como uno de sus fundamentos, el repartir o delegar las tareas para poder desarrollarlas con robots de arquitecturas simples. Entonces, seria conveniente desarrollar un algoritmo que contemplara más robots participando en la tarea. Incluso contemplar un algoritmo que permita adaptar o variar el número de participantes para diferentes condiciones.

Por otro lado, como se comenta en las conclusiones, sería pertinente implementar otro sistema de visión que permitiera una mayor libertad respecto al uso de *hardware*. Además, que tenga una mayor resolución que posibilite localizar mejor a los elementos del sistema, así como realizar movimientos más precisos. Ejemplos de estos son los sistemas de captura de movimientos como OptiTrack o Vicon.

Finalmente, sería conveniente estudiar y mejorar el algoritmo para diversas configuraciones y restricciones del entorno de trabajo. Además, una mejora importante sería contemplar el transporte de objetos con formas irregulares o inclusive en tres dimensiones.

Apéndice A

Adquisición de Datos

```
function sys=mdlOutputs(t,x,u, flag)

%Datos de Reactivision
%ID 0: objeto a transportar
%ID 1 al 4: Obstaculos
%ID 5: Destino del objeto a transportar
%ID 6: Robot 1
%ID 7: Robot 2
global t_1 x_1 y_1 t_2 x_2 y_2 x_P y_P t_P x_ob y_ob x_t y_t;
di=13;
x_ob=zeros(1,4);
y_ob=zeros(1,4);
import TUIO.*;
tc = TuioClient();
tc.connect();
for i = 1 : 10
    fiducials=tc.getTuioObjects();
    for j=1 : fiducials.size
        fid=TuioObject(fiducials.elementAt(j-1));
        id=fid.getSymbolID();
        switch id
            case 0
                t_P = 2*pi-(fid.getAngle());
                x_P = (fid.getX())*234.9415-112.7782;
                y_P = (fid.getY())*-197.2313+100.64;
            case 5
                x_t = (fid.getX())*234.9415-112.7782;
                y_t = (fid.getY())*-197.2313+100.64;
            case 6
                t_1 = 2*pi-(fid.getAngle());
                x_1 = (fid.getX())*234.9415-112.7782;
                y_1 = (fid.getY())*-197.2313+100.64;
            case 7
                t_2 = 2*pi-(fid.getAngle());
                x_2 = (fid.getX())*234.9415-112.7782;
                y_2 = (fid.getY())*-197.2313+100.64;
            otherwise
                x_ob(id) = (fid.getX())*234.9415-112.7782;
                y_ob(id) = (fid.getY())*-197.2313+100.64;
        end
    end
    pause(.01)
end
tc.disconnect();
sys = [x_1+cos(t_1)*di y_1+sin(t_1)*di t_1 x_2+cos(t_2)*di y_2+sin(t_2)*di t_2 x_t y_t x_P y_P t_P x_ob
```

Apéndice B

Código Arduino

Robot 1

```
#include <Wire.h>
#define  CMD       (byte)0x00
#define  MD25ADDRESS  0x58
#define  SOFTWAREREG  0xOD
#define  SPEED1     (byte)0x00
#define  SPEED2     0x01
#define  ENCODERONE  0x02
#define  ENCODERTWO  0x06
#define  VOLTREAD    0x0A
#define  RESETENCOD  0x20

int led =9;
int contador=0;
int contador2=0;
byte datos[7]={0,0,0,0,0,0,0};
void setup(){
  pinMode(led,OUTPUT);
  Wire.begin();
  encodeReset();
  delay(50);
  Serial.begin(57600);
}

void loop(){
  digitalWrite(led,LOW);
  if(Serial.available()==7){
    for(int i=0;i<7;i++){
      datos[i]=Serial.read();
    }
    if (datos[4]==1){
      contador+=1;
      if (contador<100){
        analogWrite(led,255);
        delay(40);
      }
      else
        analogWrite(led,0);
    }
    if (datos[5]==1){
      contador2+=1;
```

```
    if (contador<50){
      datos[1]=128;
      datos[0]=128;
     }
    }
  int chksum= datos[0]^datos[1]^datos[2]^datos[3]^datos[4]^datos[5];
  if(chksum==datos[6]){
    Wire.beginTransmission(MD25ADDRESS);
    Wire.write(SPEED2);
    Wire.write(datos[1]);
    Wire.endTransmission();
    Wire.beginTransmission(MD25ADDRESS);
    Wire.write(SPEED1);
    Wire.write(datos[0]);
    Wire.endTransmission();
  }
  Serial.flush();
 }

}
void encodeReset(){
 Wire.beginTransmission(MD25ADDRESS);
 Wire.write(SPEED2);
 Wire.write(128);
 Wire.endTransmission();
 Wire.beginTransmission(MD25ADDRESS);
 Wire.write(SPEED1);
 Wire.write(128);
 Wire.endTransmission();
}
```

Robot 2

```
#include <Wire.h>
#define  CMD       (byte)0x00
#define  MD25ADDRESS  0x58
#define  SOFTWAREREG  0xOD
#define  SPEED1     (byte)0x00
#define  SPEED2     0x01
#define  ENCODERONE  0x02
#define  ENCODERTWO  0x06
#define  VOLTREAD    0x0A
#define  RESETENCOD  0x20

int led =4;
int contador=0;
int contador2=0;
byte datos[7]={0,0,0,0,0,0,0};
void setup(){
  pinMode(led,OUTPUT);
  Wire.begin();
  encodeReset();
  delay(50);
  Serial.begin(57600);
}

void loop(){
  digitalWrite(led,HIGH);
  if(Serial.available()==7){
    for(int i=0;i<7;i++){
      datos[i]=Serial.read();
    }
    if (datos[5]==1){
      contador+=1;
      if (contador=10){
        digitalWrite(led,LOW);
        delay(10);
      }
    }
```

```
   if (datos[4]==1){
      int chksum= datos[0]^datos[1]^datos[2]^datos[3]^datos[4]^datos[5];
      if(chksum==datos[6]){
        digitalWrite(led,HIGH);
        Wire.beginTransmission(MD25ADDRESS);
        Wire.write(SPEED2);
        Wire.write(datos[3]);
        Wire.endTransmission();
        Wire.beginTransmission(MD25ADDRESS);
        Wire.write(SPEED1);
        Wire.write(-datos[2]+255);
        Wire.endTransmission();
        digitalWrite(led,LOW);
     }
    }
    else
    encodeReset();
  }
 Serial.flush();
}
void encodeReset(){
  Wire.beginTransmission(MD25ADDRESS);
  Wire.write(SPEED2);
  Wire.write(128);
  Wire.endTransmission();
  Wire.beginTransmission(MD25ADDRESS);
  Wire.write(SPEED1);
  Wire.write(128);
  Wire.endTransmission();
}
```

Apéndice C

Bloques Simulink (S-Functions)

C.1. Actualización de Meta

```
function sys=mdlOutputs(t,x,u)

    global xod yod xod2 yod2
    doX=[u(1);u(2);u(3);u(4);u(5);u(6);u(7);u(8)];
    doY=[u(9);u(10);u(11);u(12);u(13);u(14);u(15);u(16)];
    x1=u(17);
    y1=u(18);
    x2=u(19);
    y2=u(20);
    dot=round(u(21));
    dott=round(u(22));
    distm1=100000;
    distm2=100000;
    if dot>0
        if dot>8
                xod=doX(8);
                yod=doY(8);
                dott=8;
            else
                xod=doX(dot);
                yod=doY(dot);
        end
    else
        for i=1:8
            distancia=sqrt((doX(i)-x1)^2+(doY(i)-y1)^2);
            if distancia<distm1
                distm1=distancia;
                dot=i;
            end
         end
         xod=doX(dot);
         yod=doY(dot);
    end
    if dott>0
        if dott>8
                xod2=doX(8);
                yod2=doY(8);
                dott=8;
            else
                xod2=doX(dott);
                yod2=doY(dott);
        end
    else
        for i=1:8
            distancia2=sqrt((doX(i)-x2)^2+(doY(i)-y2)^2);
            if distancia2<distm2
                distm2=distancia2;
                dott=i;
            end
         end
         xod2=doX(dott);
         yod2=doY(dott);
    end
    sys = [xod;yod;dot;xod2;yod2;dott];
```

C.2. Modelo cinemático y campos potenciales robots diferenciales

```
function sys=mdlOutputs(t,x,u)


    global xR yR xR2 yR2 tR tR2 xt1 yt1 xt2 yt2 kr1 xo1 xo2 yo1 yo2 rho01
    global rhor1 rhoc1 n1 tkn1 rho02 rhor2 rhoc2;
    xR = u(1);
    yR = u(2);
    tR = u(3);
    xo1 = u(4);
    yo1 = u(5);
    rhor1 = u(7);
    rho01 = u(6);
    n1 = u(8);
    xt1 = u(9);
    yt1 = u(10);
    kr1 = u(11);
    vxmax = u(12);
    vymax = u(13);
    wmax = u(14);


% Dimensiones de la plataforma

    global b1 d1 r1 nodo1 nodo8;

    b1 = u(17);
    d1 = u(15);
    r1 = u(16);
    tkn1 = u(18);
    xR2=u(19);
    yR2=u(20);
    tR2=u(21);
    xo2=u(22);
    yo2=u(23);
    rhor2=u(25);
    rho02=u(24);
    xt2=u(26);
    yt2=u(27);
    dt=u(28);
    dt2=u(29);
    xobjeto=u(30);
    yobjeto=u(31);
% Aqui cambio la bandera de tkn, si el robot ya se encuentra en posicion
% para tomar el objeto.
   tknU=tkn1;
   tknU2=tkn1;
   if ((abs(xR-xt1)<1) && (abs(yR-yt1)<1))
       if dt<5
           if dt~=1
               ready=dt-1;
           else
               tknU=1;
               ready=1;
               nodo1=1;
               nodo8=0;
           end
       else
           if dt~=8
                ready=dt+1;
           else
               tknU=1;
               nodo8=1;
```

```
nodo1=0;
              ready=8;
           end
       end
   else
       ready=dt;
   end

   if tknU~=1
% Calculo de las velocidades de entrada para un Robot
    dg = sqrt((xt1-xR)^2 + (yt1-yR)^2);
    rhoc1 = sqrt((xo1-xR)^2 + (yo1-yR)^2);
    if yt1==yR && xt1==xR
        tg = tR;
    else
        tg = atan2((yt1-yR),(xt1-xR));
    end
    if rhoc1 > rho01+rhor1
        drep = 0;
        to = 2*tg;
    else
        drep = (n1/2)*((1/(rhoc1-rhor1))-(1/rho01))^2;
        to = atan2((yR-yo1),(xR-xo1));
    end
    dres = dg + drep;
    te = (to-tg)-tR;
    texx=abs(cos(abs(tg))-cos(abs(tR)));
    dob = sqrt((xobjeto-xR)^2 + (yobjeto-yR)^2);
    if dres > kr1
        if (dob<55) && abs(texx)>.15
            vxpp=0;
        else
            vxpp=vxmax;
        end
    else

        vxpp=vxmax*dres/kr1;
    end
    wpp = wmax*sin(te);
    U = [vxpp;wpp];
    Sq=[cos(tR) d1*sin(tR);
        sin(tR) -d1*cos(tR);
        0 1;
        1/r1 b1/r1;
        1/r1 -b1/r1];
    else
           U=[1;0];
            Sq=[0 0;
           0 0;
           0 0;
           0 0;
           0 0];
   end
   %%Robot 2
   if ((abs(xR2-xt2)<1) && (abs(yR2-yt2)<1))
        if dt2<5 && nodo1==0
            if dt2~=1
                ready2=dt2-1;
            else
                tknU2=1;
                ready2=1;
            end
        else
```

```
        if dt2~=8 && nodo8==0
            ready2=dt2+1;
        else
           tknU2=1;
           ready2=8;
        end
    end
else
   ready2=dt2;
end

 if tknU2~=1 && tknU==1
     dg2 = sqrt((xt2-xR2)^2 + (yt2-yR2)^2);
     rhoc2 = sqrt((xo2-xR2)^2 + (yo2-yR2)^2);
 if yt2==yR2 && xt2==xR2
     tg2 = tR2;
 else
     tg2 = atan2((yt2-yR2),(xt2-xR2));
 end
 if rhoc2 > rho02+rhor2
     drep2 = 0;
     to2 = 2*tg2;
 else
     drep2 = (n1/2)*((1/(rhoc2-rhor2))-(1/rho02))^2;
     to2 = atan2((yR2-yo2),(xR2-xo2));
 end
 dres2 = dg2 + drep2;
 te2 = (to2-tg2)-tR2;
 tex=abs(cos(abs(tg2))-cos(abs(tR2)));
 dob2 = sqrt((xobjeto-xR2)^2 + (yobjeto-yR2)^2);
 if dres2 > kr1
     if (dob2<56) && abs(tex)>.15
         vxpp2=0;
     else
         vxpp2=vxmax;
     end
 else
     vxpp2=vxmax*dres2/kr1;
 end
 wpp2 = wmax*sin(te2);
 U2 = [vxpp2;wpp2];
 Sq2=[cos(tR2) d1*sin(tR2);
     sin(tR2) -d1*cos(tR2);
     0 1;
     1/r1 b1/r1;
     1/r1 -b1/r1];
 else
        U2=[1;0];
         Sq2=[0 0;
        0 0;
        0 0;
        0 0;
        0 0];
end
X = (Sq*U);
X2= (Sq2*U2);
sys = [X;X2;tknU;tknU2;ready;ready2];
```

C.3. Obstáculo cercano

```
function [xo,yo,rho0,rhor] = ObstacleUpdate(xp,yp,x,y,r0,rr,tkn,hayObstacles,xr2,yr2)
%#codegen
if tkn==0
xob=x;
yob=y;
rob=r0;
rrob=rr;
distm=10000;
cercano=1;
distr2=sqrt((xr2-xp)^2+(yr2-yp)^2);
if hayObstacles
    distm=sqrt((xob(1)-xp)^2+(yob(1)-yp)^2);
    for i=2:length(xob)
        dist=sqrt((xob(i)-xp)^2+(yob(i)-yp)^2);
        if dist<distm
            distm=dist;
            cercano=i;
        end
    end
    if distr2>distm
        xo=xob(cercano);
        yo=yob(cercano);
        rho0=rob(cercano);
        rhor=rrob(cercano);
    else
        xo=xr2;
        yo=yr2;
        rho0=10;
        rhor=10;
    end
else
        xo=xr2;
        yo=yr2;
        rho0=10;
        rhor=10;

end
else
    xo=0;
    yo=0;
    rho0=0;
    rhor=0;
end
```

C.4. Modelo cinemático y campos potenciales robots acoplados

```
function sys=mdlOutputs(t,x,u)


% Variables de velocidad de entrada

    vxmax = u(1);
    vymax = u(2);
    wmax = u(3);


% Variables de retroalimentaciï¿½n angular

    tP = u(4);
    t1 = u(5);
    t2 = u(6);


% Dimensiones de la plataforma

    global a1 a2 b c d r;

    a1 = u(7);
    a2 = u(8);
    b = u(9);
    c = u(10);
    d = u(11);
    r = u(12);

% Coordenadas del robot, del objetivo y del obstï¿½culo

    global xp yp xt yt kr xo yo rho0 rhor rhoc n;

    xp = u(13);
    yp = u(14);
    xt = u(15);
    yt = u(16);
    kr = u(17);
    xo = [u(18);u(19);u(20);u(21)];
    yo = [u(22);u(23);u(24);u(25)];
    rho0 = [u(26);u(27);u(28);u(29)];
    rhor = [u(30);u(31);u(32);u(33)];
    n = u(34);
    tkn=u(35);

    if tkn==1
% Cï¿½lculo de las velocidades de entrada

    drep=zeros(1,4);
    to=zeros(1,4);
    dg = sqrt((xt-xp)^2 + (yt-yp)^2);
    %fax=xt-xp;
    %fay=yt-yp;
    for co=1:length(xo);
        rhoc(co) = sqrt((xo(co)-xp)^2 + (yo(co)-yp)^2);
```

```
end
    if yt==yp && xt==xp
        tg = tP;
    else
        tg = atan2((yt-yp),(xt-xp));
    end
    for co=1:length(rhoc)
        if rhoc(co) > rho0(co)+rhor(co)
            drep(co) = 0;
            to(co) = 2*tg;
        else
            drep(co) = (n/2)*((1/(rhoc(co)-rhor(co)))-(1/rho0(co))^2);
            to(co) = atan2((yp-yo(co)),(xp-xo(co)));
        end
    end
    dres = dg + sum(drep);
    te = (sum(to)-tg)-tP;
    teg = tg-tP;

    if dres > kr
        vxpp = vxmax*cos(te);
        vypp = vymax*sin(te);
    else
        vxpp = vxmax*dres*cos(te)/kr;
        vypp = vymax*dres*sin(te)/kr;
    end

    wpp = wmax*sin(teg-(pi/2));
    %wpp = wmax*sin(teg);
    %wpp = wmax*sin(-tP);         %Orientado a 0

    U = [vxpp; vypp; wpp];


% Cï¿½lculo de las velocidades generalizadas

    Sq=[cos(tP),-sin(tP),0;
        sin(tP),cos(tP),0;
        0,0,1;
        -sin(t1)/c,cos(t1)/c,a1*cos(t1)/c+b*sin(t1)/c;
        -sin(t2)/c,cos(t2)/c,a1*cos(t2)/c+b*sin(t2)/c;
        (c*cos(t1)+d*sin(t1))/(c*r),-d*cos(t1)/(c*r)+sin(t1)/r,-b*cos(t1)/r-
a1*d*cos(t1)/(c*r)+a1*sin(t1)/r-b*d*sin(t1)/(c*r);
        (c*cos(t1)-d*sin(t1))/(c*r),d*cos(t1)/(c*r)+sin(t1)/r,-
b*cos(t1)/r+a1*d*cos(t1)/(c*r)+a1*sin(t1)/r+b*d*sin(t1)/(c*r);
        (c*cos(t2)+d*sin(t2))/(c*r),-d*cos(t2)/(c*r)+sin(t2)/r,-b*cos(t2)/r-
a2*d*cos(t2)/(c*r)+a2*sin(t2)/r-b*d*sin(t2)/(c*r);
        (c*cos(t2)-d*sin(t2))/(c*r),d*cos(t2)/(c*r)+sin(t2)/r,-
b*cos(t2)/r+a2*d*cos(t2)/(c*r)+a2*sin(t2)/r+b*d*sin(t2)/(c*r)];
    else
        Sq=zeros(9,3);
        U=ones(3,1);
    end
    sys = Sq*U;
```

C.5. Graficador en tiempo real

```
function sys=mdlUpdate(t,x,u, Axis_Width, ON)

sys = [];


% Dibujo de las figuras


if (ON==0)||(t==0)
    hold off;
end

figure(1)
set(1,'Position', [490 150 500 500]);

% Fijacion de ejes

    axis([-Axis_Width Axis_Width -Axis_Width Axis_Width]);
    grid on;

    xlabel('x [cm]')
    ylabel('y [cm]')
    title('Figure 3: Sequence of posture')


% Entradas

    xP      = u(1);
    yP      = u(2);
    thetaP  = u(3);
    theta1  = u(4);
    theta2  = u(5);
    a1      = u(6);
    a2      = u(7);
    b       = u(8);
    c       = u(9);
    d       = u(10);
    r       = u(11);
    x1      = u(38);
    y1      = u(39);
    x2      = u(40);
    y2      = u(41);
    xobjetos = [u(15);u(16);u(17);u(18)];
    yobjetos = [u(19);u(20);u(21);u(22)];
    rhobjetos = [u(23);u(24);u(25);u(26)];
    robjetos = [u(27);u(28);u(29);u(30)];
    xt      = u(12);
    yt      = u(13);
    kr      = u(14);
    xot = u(31);
    yot = u(32);
    kor = u(33);
    tot = u(34);
    largo = u(35);
    alto = u(36);
    tkn = u(37);
    hayObjetos= u(42);
    predockX=[u(43);u(44);u(45);u(46);u(47);u(48);u(49);u(50)];
    predockY=[u(51);u(52);u(53);u(54);u(55);u(56);u(57);u(58)];
    %radio=u(59);
% Dimensiones extra de la plataforma
```

```
    extraP_x1=1.9;
    extraP_x2=5;
    extraP_x3=16.75;
    extraP_x4=5.6;
    extraP_y1=1.9;
    extraP_y2=15.8;
    a2=largo/2;
    a1=-a2;
    b=-alto/2;
% Dimensiones extra de las ruedas

    wheel_y         = 1;

% Cï¿½lculos ï¿½tiles

    cos_thP = cos(thetaP);
    sin_thP = sin(thetaP);

    cos_th1 = cos(theta1);
    sin_th1 = sin(theta1);

    cos_th2 = cos(theta2);
    sin_th2 = sin(theta2);

    cos_tot = cos(tot);
    sin_tot = sin(tot);


% Realizaciï¿½n del ï¿½rea de aproximaciï¿½n al objetivo

    Point_Sides = 10;
    Point_Sides_Angles = (0:1/Point_Sides:1-1/(2*Point_Sides))'*2*pi;

    Point_Size = kr;
    Point_Size_y = Point_Size*sin(Point_Sides_Angles);
    Point_Size_x = Point_Size*cos(Point_Sides_Angles);

    Point_y = Point_Size_y + yt;
    Point_x = Point_Size_x + xt;


% Dibujo del ï¿½rea de aproximaciï¿½n al objetivo

    hd_Point = fill(Point_x, Point_y, 'm');
    hold on;
    set(hd_Point,'EraseMode','none');

    axis([-Axis_Width Axis_Width -Axis_Width Axis_Width]);
    grid on;
    if tkn==0
%Realizacion de puntos de predocking
    Point_Sides = 10;
    Point_Sides_Angles = (0:1/Point_Sides:1-1/(2*Point_Sides))'*2*pi;
    Point_Size = 2;
    Point_Size_y = Point_Size*sin(Point_Sides_Angles);
    Point_Size_x = Point_Size*cos(Point_Sides_Angles);
    for j=1:length(predockX)
    Point_y = Point_Size_y + predockY(j);
    Point_x = Point_Size_x + predockX(j);
```

```
% Dibujo de predocking
    hd_Point = fill(Point_x, Point_y, 'r');
    set(hd_Point,'EraseMode','none');
    end
    end



% Realizaciï¿½n del ï¿½rea de aproximaciï¿½n al obstï¿½culo
if hayObjetos
    for j=1:length(yobjetos)
    xo=xobjetos(j);
    yo=yobjetos(j);
    if tkn==1
    objetivo=atan2(yobjetos(j)-yP,xobjetos(j)-xP);
    rho0=robjetos(j)+d+abs(abs(cos(objetivo))-
abs(sin(thetaP)))*alto/2+abs(abs(cos(objetivo))-abs(sin(thetaP)))*largo/2;
    %rho0=radio+robjetos(j);
    %rho0=40+robjetos(j);
    else
    rho0=rhobjetos(j)+robjetos(j);
    end
    rhor=robjetos(j);

    Point_Sides = 15;
    Point_Sides_Angles = (0:1/Point_Sides:1-1/(2*Point_Sides))'*2*pi;

    Point_Size = rho0;
    Point_Size_y = Point_Size*sin(Point_Sides_Angles);
    Point_Size_x = Point_Size*cos(Point_Sides_Angles);

    Point_y = Point_Size_y + yo;
    Point_x = Point_Size_x + xo;


% Dibujo del ï¿½rea de aproximaciï¿½n al obstï¿½culo

    hd_Point = fill(Point_x, Point_y, [0.325 0.325 0.325]);
    set(hd_Point,'EraseMode','none');


% Realizaciï¿½n del obstï¿½culo

    Point_Sides = 10;
    Point_Sides_Angles = (0:1/Point_Sides:1-1/(2*Point_Sides))'*2*pi;

    Point_Size = rhor;
    Point_Size_y = Point_Size*sin(Point_Sides_Angles);
    Point_Size_x = Point_Size*cos(Point_Sides_Angles);

    Point_y = Point_Size_y + yo;
    Point_x = Point_Size_x + xo;


% Dibujo del obstï¿½culo

    hd_Point = fill(Point_x, Point_y, 'k');
    set(hd_Point,'EraseMode','none');
    end
end
```

```
% Realizaciï¿½n del objetivo

    Point_Sides = 10;
    Point_Sides_Angles = (0:1/Point_Sides:1-1/(2*Point_Sides))'*2*pi;

    Point_Size = 1;
    Point_Size_y = Point_Size*sin(Point_Sides_Angles);
    Point_Size_x = Point_Size*cos(Point_Sides_Angles);

    Point_y = Point_Size_y + yt;
    Point_x = Point_Size_x + xt;


% Dibujo del objetivo

    hd_Point = fill(Point_x, Point_y, 'k');
    set(hd_Point,'EraseMode','none');
    % Realizaciï¿½n del ï¿½rea de aproximaciï¿½n al objeto a mover

    wmr1_contour_x = [-(c+1.1); -(c+1.1); 3.9; 3.9];
    wmr1_contour_y = [-d; d; d; -d];
    wmr2_contour_x = [-(c+1.1); -(c+1.1); 3.9; 3.9];
    wmr2_contour_y = [-d; d; d; -d];
    wheel_contour_x = [-r; -r; r; r];
    wheel_contour_y = [-(wheel_y/2); (wheel_y/2); (wheel_y/2); -(wheel_y/2)];
    if (tkn==0)
% Realizaciï¿½n del robot de traccii¿½n 1
    wmr1_pos_x = (cos_th1*wmr1_contour_x-sin_th1*wmr1_contour_y)+x1;
    wmr1_pos_y= (sin_th1*wmr1_contour_x+cos_th1*wmr1_contour_y)+y1;
% Realizaciï¿½n del robot de traccii¿½n 2
    wmr2_pos_x = (cos_th2*wmr2_contour_x-sin_th2*wmr2_contour_y)+x2;
    wmr2_pos_y=(sin_th2*wmr2_contour_x+cos_th2*wmr2_contour_y)+y2;
% Realizaciï¿½n de las ruedas

    wheel11_pos_x = cos_th1*(-c + wheel_contour_x)- sin_th1*(d + wheel_contour_y)+x1;
    wheel11_pos_y = sin_th1*(-c + wheel_contour_x)+ cos_th1*(d + wheel_contour_y)+y1;
    wheel12_pos_x = cos_th1*(-c + wheel_contour_x)- sin_th1*(-d + wheel_contour_y)+x1;
    wheel12_pos_y = sin_th1*(-c + wheel_contour_x)+ cos_th1*(-d + wheel_contour_y)+y1;
    wheel21_pos_x = cos_th2*(-c + wheel_contour_x)- sin_th2*(d + wheel_contour_y)+x2;
    wheel21_pos_y = sin_th2*(-c + wheel_contour_x)+ cos_th2*(d + wheel_contour_y)+y2;
    wheel22_pos_x = cos_th2*(-c + wheel_contour_x)- sin_th2*(-d + wheel_contour_y)+x2;
    wheel22_pos_y = sin_th2*(-c + wheel_contour_x)+ cos_th2*(-d + wheel_contour_y)+y2;

%Realizacion del punto (objeto) a mover
    ob_contour_x = [-largo/2; -largo/2; largo/2; largo/2];
    ob_contour_y = [-alto/2; alto/2; alto/2; -alto/2];

    ob_pos_x = (cos_tot*ob_contour_x-sin_tot*ob_contour_y)+xot;
    ob_pos_y= (sin_tot*ob_contour_x+cos_tot*ob_contour_y)+yot;

    else
    % Realizaciï¿½n del robot de traccion 1
    wmr1_pos_x = xP + cos_thP*(a1 + cos_th1*wmr1_contour_x - sin_th1*wmr1_contour_y) -
sin_thP*(b + sin_th1*wmr1_contour_x + cos_th1*wmr1_contour_y);
    wmr1_pos_y = yP + sin_thP*(a1 + cos_th1*wmr1_contour_x - sin_th1*wmr1_contour_y) +
cos_thP*(b + sin_th1*wmr1_contour_x + cos_th1*wmr1_contour_y);

% Realizaciï¿½n del robot de tracciïon 2
```

```
wmr2_pos_x = xP + cos_thP*(a2 + cos_th2*wmr2_contour_x - sin_th2*wmr2_contour_y) -
sin_thP*(b + sin_th2*wmr2_contour_x + cos_th2*wmr2_contour_y);
    wmr2_pos_y = yP + sin_thP*(a2 + cos_th2*wmr2_contour_x - sin_th2*wmr2_contour_y) +
cos_thP*(b + sin_th2*wmr2_contour_x + cos_th2*wmr2_contour_y);


% Realizaciï¿½n de las ruedas

    wheel11_pos_x = xP + cos_thP*(a1 + cos_th1*(-c + wheel_contour_x) - sin_th1*(d +
wheel_contour_y)) - sin_thP*(b + sin_th1*(-c + wheel_contour_x) + cos_th1*(d +
wheel_contour_y));
    wheel11_pos_y = yP + sin_thP*(a1 + cos_th1*(-c + wheel_contour_x) - sin_th1*(d +
wheel_contour_y)) + cos_thP*(b + sin_th1*(-c + wheel_contour_x) + cos_th1*(d +
wheel_contour_y));
    wheel12_pos_x = xP + cos_thP*(a1 + cos_th1*(-c + wheel_contour_x) - sin_th1*(-d +
wheel_contour_y)) - sin_thP*(b + sin_th1*(-c + wheel_contour_x) + cos_th1*(-d +
wheel_contour_y));
    wheel12_pos_y = yP + sin_thP*(a1 + cos_th1*(-c + wheel_contour_x) - sin_th1*(-d +
wheel_contour_y)) + cos_thP*(b + sin_th1*(-c + wheel_contour_x) + cos_th1*(-d +
wheel_contour_y));
    wheel21_pos_x = xP + cos_thP*(a2 + cos_th2*(-c + wheel_contour_x) - sin_th2*(d +
wheel_contour_y)) - sin_thP*(b + sin_th2*(-c + wheel_contour_x) + cos_th2*(d +
wheel_contour_y));
    wheel21_pos_y = yP + sin_thP*(a2 + cos_th2*(-c + wheel_contour_x) - sin_th2*(d +
wheel_contour_y)) + cos_thP*(b + sin_th2*(-c + wheel_contour_x) + cos_th2*(d +
wheel_contour_y));
    wheel22_pos_x = xP + cos_thP*(a2 + cos_th2*(-c + wheel_contour_x) - sin_th2*(-d +
wheel_contour_y)) - sin_thP*(b + sin_th2*(-c + wheel_contour_x) + cos_th2*(-d +
wheel_contour_y));
    wheel22_pos_y = yP + sin_thP*(a2 + cos_th2*(-c + wheel_contour_x) - sin_th2*(-d +
wheel_contour_y)) + cos_thP*(b + sin_th2*(-c + wheel_contour_x) + cos_th2*(-d +
wheel_contour_y));

    %Realizacion del objeto a mover

    ob_contour_x = [-largo/2; -largo/2; largo/2; largo/2];
    ob_contour_y = [-alto/2; alto/2; alto/2; -alto/2];

    ob_pos_x = (cos_thP*ob_contour_x-sin_thP*ob_contour_y)+xP;
    ob_pos_y= (sin_thP*ob_contour_x+cos_thP*ob_contour_y)+yP;


% Dibujo del ï¿½rea de aproximaciï¿½n al objeto a mover
    end

    hd_PointO = fill(ob_pos_x, ob_pos_y, 'g');
    hold on;
    set(hd_PointO,'EraseMode','none');
    axis([-Axis_Width Axis_Width -Axis_Width Axis_Width]);
    grid on;
    % Realizaciï¿½n del punto P

    Point_Sides = 10;
    Point_Sides_Angles = (0:1/Point_Sides:1-1/(2*Point_Sides))'*2*pi;

    Point_Size = 1;
    Point_Size_y = Point_Size*sin(Point_Sides_Angles);
    Point_Size_x = Point_Size*cos(Point_Sides_Angles);

    Point_y = Point_Size_y + yP;
    Point_x = Point_Size_x + xP;
```

```
    % Dibujo del punto P

    hd_Point = fill(Point_x, Point_y, 'y');
    set(hd_Point,'EraseMode','none');

    % Dibujo del robot de tracciï¿½n 1

    hd_wmr1 = fill(wmr1_pos_x, wmr1_pos_y, 'c');
    set(hd_wmr1,'EraseMode','none');

    % Dibujo del robot de traciï¿½n 2

    hd_wmr2 = fill(wmr2_pos_x, wmr2_pos_y, 'b');
    set(hd_wmr2,'EraseMode','none');

    % Dibujo de la rueda 1,1

    hd_wheel11 = fill(wheel11_pos_x, wheel11_pos_y, 'g');
    set(hd_wheel11,'EraseMode','none');

    % Dibujo de la rueda 1,2

    hd_wheel12 = fill(wheel12_pos_x, wheel12_pos_y, 'g');
    set(hd_wheel12,'EraseMode','none');

% Dibujo de la rueda 2,1

    hd_wheel21 = fill(wheel21_pos_x, wheel21_pos_y, 'g');
    set(hd_wheel21,'EraseMode','none');

% Dibujo de la rueda 2,2

    hd_wheel22 = fill(wheel22_pos_x, wheel22_pos_y, 'g');
    set(hd_wheel22,'EraseMode','none');
    if (tkn==0)
%Realizacion del punto P1
    Point_Sides = 10;
    Point_Sides_Angles = (0:1/Point_Sides:1-1/(2*Point_Sides))'*2*pi;
    Point_Size = 2;
    Point_Size_y = Point_Size*sin(Point_Sides_Angles);
    Point_Size_x = Point_Size*cos(Point_Sides_Angles);
    Point_y = Point_Size_y + y1;
    Point_x = Point_Size_x + x1;
% Dibujo del punto P1
    hd_Point = fill(Point_x, Point_y, 'y');
    set(hd_Point,'EraseMode','none');

%Realizacion del punto P2
    Point_Sides = 10;
    Point_Sides_Angles = (0:1/Point_Sides:1-1/(2*Point_Sides))'*2*pi;
    Point_Size = 2;
    Point_Size_y = Point_Size*sin(Point_Sides_Angles);
    Point_Size_x = Point_Size*cos(Point_Sides_Angles);
    Point_y = Point_Size_y + y2;
    Point_x = Point_Size_x + x2;


% Dibujo del punto P2
    hd_Point = fill(Point_x, Point_y, 'y');
    set(hd_Point,'EraseMode','none');
```

```
else
        % Realizaciï¿½n del punto P1

    Point_Sides = 10;
    Point_Sides_Angles = (0:1/Point_Sides:1-1/(2*Point_Sides))'*2*pi;

    Point_Size = 1;
    Point_Size_y = Point_Size*sin(Point_Sides_Angles);
    Point_Size_x = Point_Size*cos(Point_Sides_Angles);

    Point_y = Point_Size_y + yP + sin_thP*a1 + cos_thP*b;
    Point_x = Point_Size_x + xP + cos_thP*a1 - sin_thP*b;
    % Dibujo del punto P1

    hd_Point = fill(Point_x, Point_y, 'k');
    set(hd_Point,'EraseMode','none');


% Realizaciï¿½n del punto P2

    Point_Sides = 10;
    Point_Sides_Angles = (0:1/Point_Sides:1-1/(2*Point_Sides))'*2*pi;

    Point_Size = 1;
    Point_Size_y = Point_Size*sin(Point_Sides_Angles);
    Point_Size_x = Point_Size*cos(Point_Sides_Angles);

    Point_y = Point_Size_y + yP + sin_thP*a2 + cos_thP*b;
    Point_x = Point_Size_x + xP + cos_thP*a2 - sin_thP*b;


% Dibujo del punto P2

    hd_Point = fill(Point_x, Point_y, 'k');
    set(hd_Point,'EraseMode','none');


    end

% Guardado de los objetos

    set_param(gcb,'UserData',[hd_Point]);% hd_wmr1; hd_wmr2; hd_wheel11; hd_wheel12;
hd_wheel21; hd_wheel22; hd_Point]);
```

C.6. Acondicionamiento de velocidades

```
function sys=mdlOutputs(t,x,u)


    vel(1)=8.14*u(1)+128;
    vel(2)=8.14*u(2)+128;
    vel(3)=8.14*u(3)+128;
    vel(4)=8.14*u(4)+128;
    vel(5)=u(5);
    vel(6)=u(6);
    if vel(1) < 0
        vel(1)=0;
    elseif vel(1) > 255
        vel(1)=255;
    end
    if vel(2) < 0
        vel(2)=0;
    elseif vel(2) > 255
        vel(2)=255;
    end
    if vel(3) < 0
        vel(3)=0;
    elseif vel(3) > 255
        vel(3)=255;
    end
    if vel(4) < 0
        vel(4)=0;
    elseif vel(4) > 255
        vel(4)=255;
    end
    vel(1)=uint8(u(1));
    vel(2)=uint8(u(2));
    vel(3)=uint8(u(3));
    vel(4)=uint8(u(4));
    vel(5)=uint8(u(5));
    vel(6)=uint8(u(6));
    checksumB=bitxor(vel(1),bitxor(vel(2),bitxor(vel(3),bitxor(vel(4),bitxor(vel(5),vel(6))))));
    sys = [vel checksumB];
```

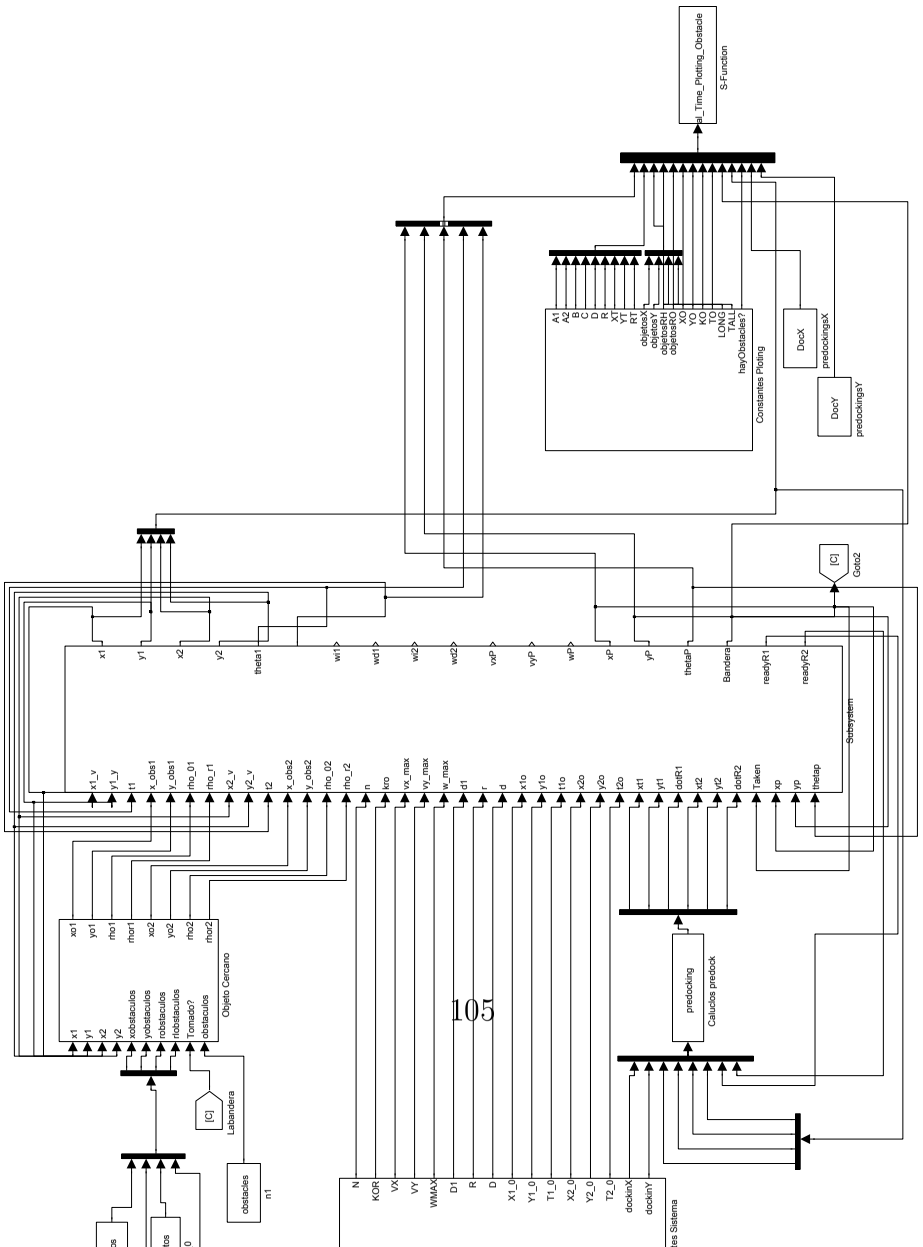
C.7. Envío de datos vía serial

```
function sys=mdlUpdate(t,x,u)
v(1)=u(1);
v(2)=u(2);
v(3)=u(3);
v(4)=u(4);
v(5)=u(5);
v(6)=u(6);
chksum=u(7);
delete(instrfindall)
s=serial('/dev/ttyS101','BaudRate',57600);
fopen(s);
fwrite(s,v(1),'uint8');
fwrite(s,v(2),'uint8');
fwrite(s,v(3),'uint8');
fwrite(s,v(4),'uint8');
fwrite(s,v(5),'uint8');
fwrite(s,v(6),'uint8');
fwrite(s,chksum,'uint8');
fclose(s);
pause(.05);
delete(instrfindall)
sys = [ ];
```

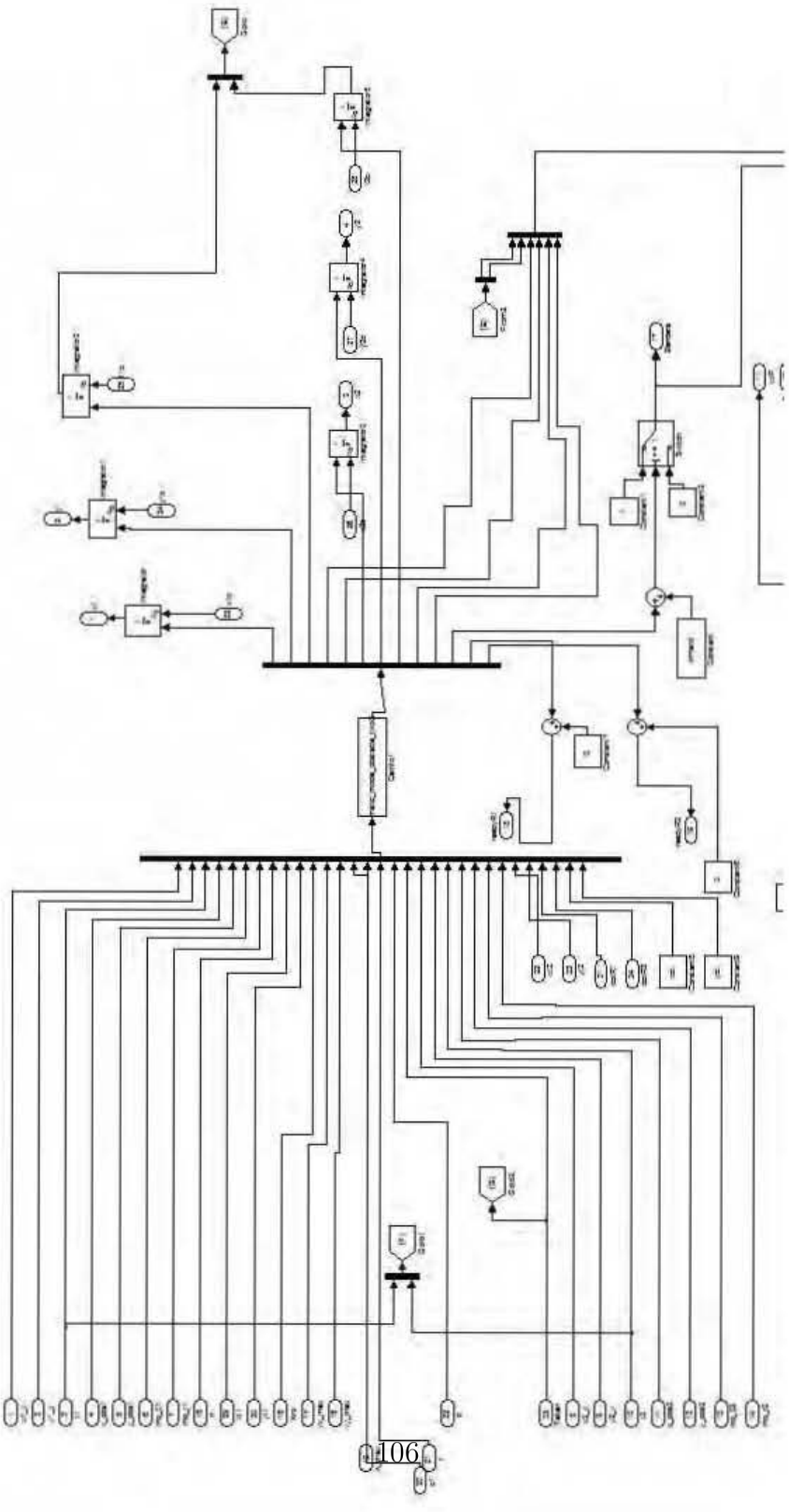
Apéndice D

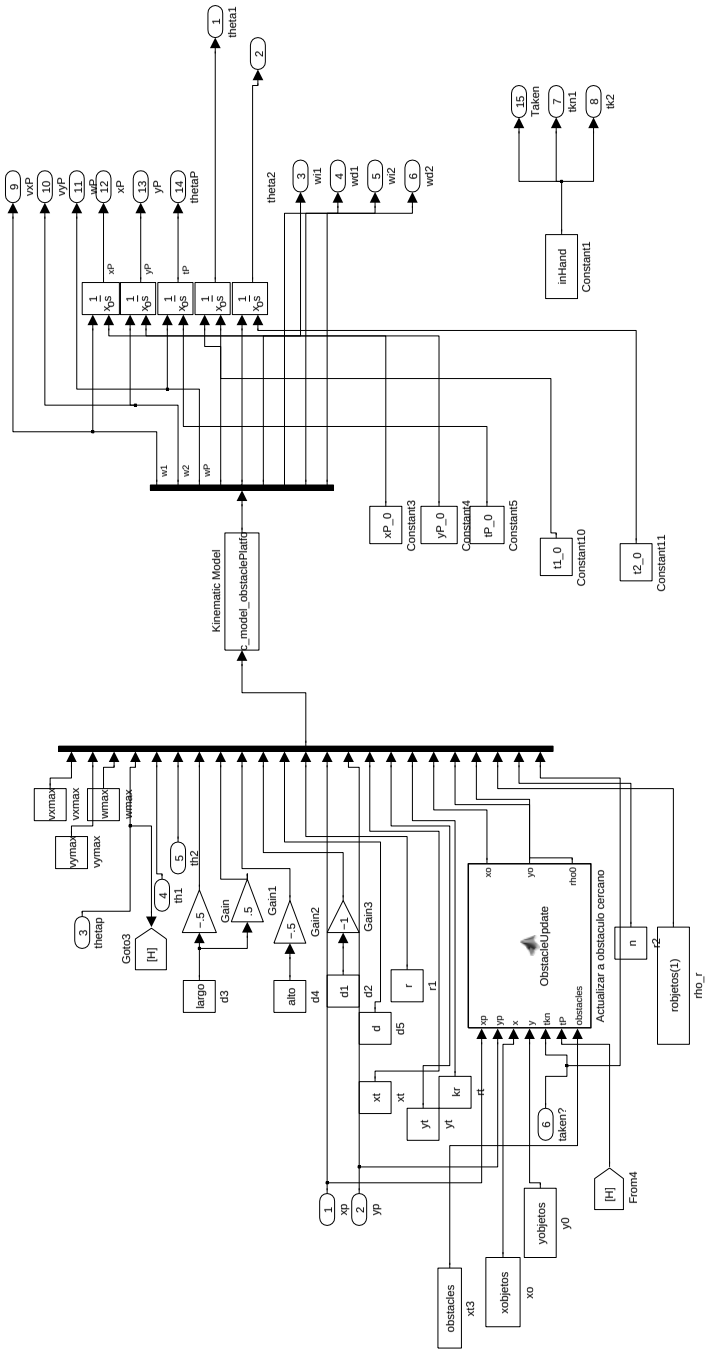
Diagramas de bloques de Simulink

D.1. Diagrama General



D.2. Diagrama Subsistema Modelos Cinemáticos




theta1
theta2
wi1
wd1
wi2
wd2
vxP
vyP
wP
xP
yP
thetaP
taken
tkn1
tk2
inHand
Constant1
xP_0
yP_0
tP_0
t1_0
t2_0
Constant3
Constant4
Constant5
Constant10
Constant11
Kinematic Model
r_model_obstaclePlati
vxmax
vymax
wmax
th1
th2
Gain
Gain1
Gain2
Gain3
thetap
Goto3
largo
alto
d1
d2
d3
d4
d5
r
r1
d
xt
yt
kr
rt
ObstacleUpdate
Actualizar a obstaculo cercano
xo
yo
rho0
obstacles
n
r2
robjetos(1)
rho_r
taken?
From4
xp
yp
x
y
tkn
tP
xobjetos
yobjetos
y0
obstacles
xt3

Bibliografía

[1] Lynne Parker. Decision making as optimization in multi-robot teams. In R. Ramanujam and Srini Ramaswamy, editors, *Distributed Computing and Internet Technology*, volume 7154 of *Lecture Notes in Computer Science*, pages 35–49. Springer Berlin / Heidelberg, 2012.

[2] T. Arai, E. Pagello, and L.E. Parker. Guest editorial advances in multirobot systems. *Robotics and Automation, IEEE Transactions on*, 18(5):655 –661, oct. 2002.

[3] T. Fukuda, S. Nakagawa, Y. Kawauchi, and M. Buss. Self organizing robots based on cell structures - ckbot. In *Intelligent Robots, 1988., IEEE International Workshop on*, pages 145 –150, oct-2 nov 1988.

[4] G. Beni. The concept of cellular robotic system. In *Intelligent Control, 1988. Proceedings., IEEE International Symposium on*, pages 57 –62, aug 1988.

[5] H. Asama, A. Matsumoto, and Y. Ishida. Design of an autonomous and distributed robot system: Actress. In *Intelligent Robots and Systems '89. The Autonomous Mobile Robots and Its Applications. IROS '89. Proceedings., IEEE/RSJ International Workshop on*, pages 283 –290, sep 1989.

[6] P. Caloud, Wonyun Choi, J.-C. Latombe, C. Le Pape, and M. Yim. Indoor automation with many mobile robots. In *Intelligent Robots and Systems '90. 'Towards a New Frontier of Applications', Proceedings. IROS '90. IEEE International Workshop on*, pages 67 –72 vol.1, jul 1990.

[7] Parker L.E. *Heterogeneous multi-robot cooperation.* PhD thesis, MIT, 1994.

[8] B.R. Donald, J. Jennings, and D. Rus. Analyzing teams of cooperating mobile robots. In *Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on*, pages 1896 –1903 vol.3, may 1994.

[9] D. Rus, B. Donald, and J. Jennings. Moving furniture with teams of autonomous robots. In *Intelligent Robots and Systems 95. 'Human Robot Interaction and Cooperative Robots', Proceedings. 1995 IEEE/RSJ International Conference on*, volume 1, pages 235 –242 vol.1, aug 1995.

[10] D.J. Stilwell and J.S. Bay. Toward the development of a material transport system using swarms of ant-like robots. In *Robotics and Automation, 1993. Proceedings., 1993 IEEE International Conference on*, pages 766 –771 vol.1, may 1993.

[11] O. Khatib, K. Yokoi, K. Chang, D. Ruspini, R. Holmberg, and A. Casal. Vehicle/arm coordination and multiple mobile manipulator decentralized cooperation. In *Intelligent Robots and Systems '96, IROS 96, Proceedings of the 1996 IEEE/RSJ International Conference on*, volume 2, pages 546 –553 vol.2, nov 1996.

[12] Zhi Dong Wang, Y. Kimura, Takayuki Takahashi, and Eiji Nakano. A control method of a multiple non-holonomic robot system for cooperative object transportation. In *DARS'00*, pages 447–456, 2000.

[13] B. Donald, L. Gariepy, and D. Rus. Distributed manipulation of multiple objects using ropes. In *Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on*, volume 1, pages 450 –457 vol.1, 2000.

[14] N. Miyata, J. Ota, T. Arai, and H. Asama. Cooperative transport by multiple mobile robots in unknown static environments associated with real-time task assignment. *Robotics and Automation, IEEE Transactions on*, 18(5):769 – 780, oct 2002.

[15] A.K. Das, R. Fierro, V. Kumar, J.P. Ostrowski, J. Spletzer, and C.J. Taylor. A vision-based formation control framework. *Robotics and Automation, IEEE Transactions on*, 18(5):813 – 825, oct 2002.

[16] M. Udomkun and P. Tangamchit. Cooperative overhead transportation of a box by decentralized mobile robots. In *Robotics, Automation and Mechatronics, 2008 IEEE Conference on*, pages 1161 –1166, sept. 2008.

[17] Chia Choon Loh and A. Trachtler. Laser-sintered platform with optical sensor for a mobile robot used in cooperative load transport. In *IECON 2011 - 37th Annual Conference on IEEE Industrial Electronics Society*, pages 203 –208, nov. 2011.

[18] J. Fink, M.A. Hsieh, and V. Kumar. Multi-robot manipulation via caging in environments with obstacles. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 1471 –1476, may 2008.

[19] Yifan Cai and S.X. Yang. Fuzzy logic-based multi-robot cooperation for object-pushing. In *Information and Automation (ICIA), 2011 IEEE International Conference on*, pages 273 –278, june 2011.

[20] González Villela V.J. *Research on a semiautonomous mobile robot for loosely structures environments focused on transporting mail trolleys.* PhD thesis, Loughborough University, 2006.

[21] Martinez Carrillo A. Plataforma móvil omnidireccional a partir de dos robots móviles diferenciales, 2011.

[22] Oussama Khatib Bruno Siciliano, editor. *Handbook of robotics*. Springer, 2008.

[23] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. In *Robotics and Automation. Proceedings. 1985 IEEE International Conference on*, volume 2, pages 500 – 505, mar 1985.

[24] Devantech. Md25 - dual 12volt 2.8amp h bridge motor drive. http://pishrobot.com/files/products/datasheets/md25.pdf, Febrero 2013.

[25] Digi International Inc. Xbee series 2 (zigbee mesh). http://www.digi.com/products/wireless-wired-embedded-solutions/zigbee-rf-modules/point-multipoint-rfmodules/xbee-series1-moduleoverview, Febrero 2013.