UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA MECÁNICA

ANÁLISIS DEL COMPORTAMIENTO COLECTIVO PRESENTADO POR ROBOTS VIBRACIONALES

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL MECÁNICO

MARCO ANDRÉS VÁSQUEZ DÍAZ

PROFESOR GUÍA:
JUAN ZAGAL MONTEALEGRE

MIEMBROS DE LA COMISIÓN:
CLAUDIO FALCON BEAS
PABLO GUERRERO PÉREZ

SANTIAGO DE CHILE
MARZO 2016

RESUMEN DE LA MEMORIA
PARA OPTAR AL TITULO DE:
Ingeniero Civil Mecánico
POR: Marco Andrés Vásquez Díaz
FECHA:28/03/2016
PROFESOR GUIA: Juan Zagal Montealegre

ANÁLISIS DEL COMPORTAMIENTO COLECTIVO PRESENTADO POR ROBOTS VIBRACIONALES

El informe que a continuación se presenta, expone las directrices que guiarán el trabajo de tesis: *Análisis del comportamiento colectivo presentado por robots vibracionales*. Este trabajo de título tiene por objeto realizar el estudio y modelamiento del comportamiento individual y colectivo presentado por una serie de robots vibracionales autónomos de geometría cúbica.

Como punto inicial, se considera el diseño del cuerpo de los robots en el programa SolidWorks ™para su posterior construcción en las impresoras tridimensionales modelo MakerBot 2, disponibles en el laboratorio de fabricación digital (FABLAB). Motores vibradores de celular servirán como elemento motriz y pilas de reloj serán la fuente de energía de los robots.

La primera etapa del proceso de modelamiento se realizará mediante el rastreo de los cubos utilizando la cámara Optitrack Trio™, gracias a la cual será posible conocer las trayectorias y velocidades de los robots. Con estos datos, se obtendrá un modelo estadístico de la cinemática de los robots (trayectoria y velocidad) mediante el software Matlab™.

En paralelo, se desarrollará una simulación realista de los cubos utilizando la librería Open Dynamics Engine (ODE) perteneciente al lenguaje de programación C++. Esta simulación será validada al enfrentar los datos obtenidos con las mediciones realizadas a los robots por medio de la cámara Optitrack Trio™.

En una segunda etapa, se dotará a estos robots modulares de interacciones para determinar la influencia de las mismas en el comportamiento presentado por los robots. La determinación de patrones de ordenamiento, al igual que en la primera etapa, será determinada mediante el seguimiento obtenido mediante el software Motive™perteneciente a la marca Optitrack™. Tal como en la primera etapa, se implementarán simulaciones realistas de los robots y sus interacciones para luego realizar la validación pertinente.

Finalmente, se procederá a formular un modelo estocástico del movimiento de estos robots que servirá como herramienta para futuras investigaciones dentro del campo de la robótica modular.

Tabla de contenido

Capítulo 1

Introducción

Para nadie es un misterio que la observación de la naturaleza es, ha sido y será una de las principales fuentes de conocimiento para el ser humano. Desde tiempos inmemoriales, el hombre ha sido capaz de adaptar e imitar, dentro de sus posibilidades, el comportamiento de las especies que coexisten con el ser humano dentro del planeta para aumentar sus ventajas comparativas, asegurando su sobrevivencia y confort.

Dentro del último siglo, uno de los temas que ha causado interés en los investigadores de distintas áreas de la ciencia, es el comportamiento exhibido por especies que son capaces de agruparse para obtener el bien colectivo, ejemplos de esto son: abejas, hormigas, aves, lobos, etc., donde se aprecia que lo importante deja de ser el individuo y pasa a preponderar el bienestar del grupo. Es decir, se trabaja en conjunto bajo la premisa de una serie de reglas básicas simples que parecieran formar parte de ellos de manera innata, permitiéndoles: huir de manera efectiva de un depredador, conseguir el camino más corto entre el lugar de origen y la fuente de alimento, migrar de acuerdo a sus necesidades, entre otros.

En un sistema consistente en muchas unidades similares (tales como moléculas o bandadas de pájaros) las interacciones entre las unidades pueden ser simples (atracción o repulsión) o más complejas (combinaciones de interacciones simples) y pueden ocurrir entre vecinos o entre una red interconectada de individuos. Más de un sistema de interacción ha sido identificado, siendo dos los más rescatables: el primero de ellos es de corto alcance y considera la interacción entre los individuos, el segundo comtempla la comunicación a través de los cambios en el ambiente y es conocido como *estigmergia*, el cual se produce cuando los insectos modifican su comportamiento debido a las modificaciones realizadas en el ambiente por sus compañeros. Este tipo de comportamiento puede ser observado en la construcción de los termiteros, donde los cambios en el comportamiento de los trabajadores son determinados por la estructura del termitero.

Las criaturas sociales son capaces de intercambiar información y por ejemplo, comunicar la ubicación de una fuente de alimento, una zona favorable para el pastoreo o la presencia de peligro. Esta interacción entre individuos se basa en el concepto de *localidad*, donde no hay conocimiento de la situación global [64]. Estudios recientes han llegado a demostrar que los individuos constituyentes de estos sistemas no requieren ningún tipo de representación o

conocimiento sofisticado para producir comportamientos complejos [105], es más, en el caso de los insectos sociales, los individuos no se encuentran informados del estado global de la colonia. El conocimiento del enjambre se encuentra distribuido a través de todos los agentes es decir, no existe un líder que guíe a los otros individuos para cumplir sus objetivos [114].

Figura 1.1: Comportamiento Colectivo de Bandada. Neels Castillon, Bird Ballet, 10 de Noviembre de 2013, por Hristiyan Petrov. Imagen tomada del sitio de fotos The Pistrophy: http://www.thepistrophy.com/bird-ballet-swarming-video-by-neels-castillon/

Este comportamiento que, a priori, parece ser muy simple, abre un nuevo mundo de posibilidades al momento de diseñar un nuevo sistema debido a que cambia el dogma de concepción de una herramienta única, especializada y de gran tamaño, por la alternativa de construir una serie de elementos más simples que sean capaces de cumplir con la misma tarea por medio de la agregación de sus funciones.

El estudio de los sistemas auto-organizados, así como, las propiedades de transporte de sistemas consistentes en partículas auto-propulsadas (SPP, por sus siglas en inglés [16]) han permitido el descubrimiento de una serie de propiedades intrínsecas de los sistemas formados por la agregación de individuos, dentro de las cuales cabe destacar: robustez, confiabilidad, adaptabilidad, reconfigurabilidad y escalabilidad.

Por otro lado, las ventajas de diseñar en función de esta serie de elementos modulares, permite entre otras cosas: modificar la tarea en la cual se desempeñan al cambiar la forma en la que interactuán entre ellos, disminuir los costos de implementación por tratarse de unidades más simples y reducir los costos de mantenimiento.

En la actualidad, esta filosofía ha permitido la construcción de modelos descentranlizados en los negocios [44], la creación de robots especializados en la búsqueda y rescate de perso-

nas [84], la optimización de sistemas fotovoltáicos [17] e incluso ha permitido optimizar la planeación de la trayectoria de satélites [65].

Bajo ciertas condiciones, se observan transiciones en las que los individuos adoptan patrones de comportamiento casi completamente determinados por los efectos colectivos. El comportamiento anterior describe la cualidad principal del comportamiento colectivo, vale decir, que el comportamiento de las unidades está determinado por la influencia del resto de los integrantes del conjunto, lo cual produce que las unidades se comporten de manera completamente diferente a como lo hacen de maner individual. Por ejemplo, mientras un grupo de palomas se alimenta en el piso, se encuentran orientadas al azar, pero se alejan en una bandada ordenada cuando se sienten amenazadas por algún tipo de perturbación [120].

La robótica de enjambre investiga formas para lograr la coordinación de grandes grupos de robots relativamente simples por medio de una serie de reglas locales[114]. De este modo, un gran número de robots es capaz de desempeñar tareas de manera más eficiente que un solo robot, dándole robustez y flexibilidad al grupo [114]

La auto-organización significa la espontánea emergencia del orden en sistemas físicos y naturales [75].

Gerardo Beni [25] describe la coordinación de este tipo de robots como sigue:

> *El grupo de robots no es sólo un grupo. Tiene algunas características especiales, que son encontradas en enjambres de insectos, esto es, control descentralizado, carencia de sincronización, miembros simples y casi idénticos.*

Los modelos que describen esta serie de fenómenos son distintivamente modelos fuera del equilibrio, mostrando transiciones de fase cinemáticas y auto-organización, y son de particular interés desde el punto de vista de la mecánica estadística moderna.

1.1. Objetivos

1.1.1. Objetivo General

Desarrollar un modelo de comportamiento individual y colectivo caracterizado por robots modulares de movimientos vibratorios sometidos a distintos grados de interacción sistémica.

1.1.2. Objetivos Específicos

1. Diseñar nuevas plataformas robóticas para el estudio del comportamiento colectivo y robótica modular.
2. Construir familias con un número representativo de individuos para realizar la caracterización cinemática individual y colectiva de las mismas.

3. Caracterizar el comportamiento cinemático individual de cada una de las plataformas robóticas construidas.
4. Análizar el comportamiento colectivo de los robots sometidos a distintos grados de interacción sistémica.
5. Implementar una simulación realista de los robots para escalar los resultados obtenidos con la plataforma, utilizando la librería *Open Dynamics Engine (ODE)*, perteneciente al lenguaje de programación C++.
6. Identificar y explicar comportamientos individuales y colectivos de diversos sistemas sometidos a distintos grados de interacción.

Capítulo 2

Antecedentes

Como humanos, normalmente tendemos a aplicar un acercamiento muy controlado y jerárquico a nuestros procesos productivos. Por ejemplo, el metal que forma parte del chasís de un vehículo es producido mediante el moldeo de un trozo de material en láminas delgadas, que luego son presionadas contra un molde que tiene una forma pre-establecida. Las demás actividades de la vida diaria no escapan a esta estructuración. Ejemplo de esto es la regulación de los medios de transporte: el tráfico de automóviles, trenes y aviones se encuentran controlados por medio de horarios y estructuras de comando centrales (semáforos, torres de control, etc). Este tipo de planificación, si bien es bastante funcional, requiere del conocimiento global de la situación a ser controlada, así como la presencia de un líder que organice las actividades a ser realizadas.

En contraste, en la naturaleza, se construyen geometrías complejas como los termiteros y las bandadas de pájaros que son capaces de moverse ágilmente sin necesidad de un organismo de control. ¿Cómo es esto posible?, la respuesta a esta pregunta se entrega en la primera sección del presente capítulo *Comportamiento colectivo en la naturaleza*, donde se describen los procesos más representativos de la navegación en grandes grupos, toma de decisiones colectivas, coordinación en grandes trabajos de construcción, auto-sellado, crecimiento adaptativo e inteligencia de enjambre. La implicancia de los estudios de los comportamientos naturales anteriores son descritos en la sección *Enjambres robóticos*, donde se define qué es la robótica de enjambre, se describen sus características principales y finalmente se repasan sus aplicaciones potenciales más relevantes.

En el capítulo *Conceptos básicos de la mecánica estadística de Enjambre*, se repasan principios y conceptos tales como parámetros de orden, transiciones de fase y funciones de correlación, para finalmente dar paso a la sección *Modelos básicos para partículas auto-propulsadas*, donde se describen los principales modelos matemáticos para el comportamiento colectivo, dentro de los que cabe mencionar el modelo de Vicsek (SVM) y el modelo de Cucker-Smale (CS).

En un primer acercamiento, el intercambio de información en el enjambre se consideró como una red centralizada. Se supuso que las hormigas y abejas reinas de cada colonia eran las responsables de la transmisión y la asignación de la información a cada agente

[99]. Sin embargo, se ha demostrado que la red en el enjambre está descentralizada [72]. Gracias a las investigaciones recientes los biólogos pueden afirmar con certeza que no hay identificadores únicos u otro almacenamiento global de la información en la red [107]. Ningún agente único puede acceder a toda la información en la red y los marcadores son, por lo tanto, inexistentes. Los biólogos creen que las diversas clases sociales se organizan como un sistema descentralizado, distribuido en todo el entorno que se puede describir a través de un modelo probabilístico. De esta manera los agentes en el enjambre siguen sus propias reglas de acuerdo a la información local. Los comportamientos del grupo surgen de esas leyes locales que se ven afectadas por el intercambio de información y la estructura topológica del enjambre. Las reglas locales son componentes claves para mantener a toda la estructura flexible y robusta, incluso cuando emergen comportamientos sofisticados.

La respuesta a esta pregunta surgió de la observación de los insectos y animales sociales donde se presentan este tipo de procesos. En muchas situaciones, puede resultar ventajoso emular la naturaleza y trasladar la capacidad de planificación desde el nivel superior de la jerarquía a los organismos constituyentes.

Los conceptos de auto-organización generalmente son explicados tomando en cuenta las estructuras emergentes y sus propiedades. Las estructuras emergentes son más que la suma de sus constituyentes, dado que poseen propiedades que no son esperadas al considerar los elementos singulares que se agrupan para formarla. La emergencia es usada para describir fenómenos macroscópicos que no son observables a escala microscópica. Una estructura emergente es gobernada por una combinación de reglas y principios a escala microscópica con la interacción de la escala macroscópica y sus alrededores. Las propiedades emergentes son patrones complejos o procesos derivados de la interacción simple entre multiples agentes. Existen muchos ejemplos de propiedades emergentes en sistemas biológicos, que van desde la construcción de hormigueros y termiteros por medio de reglas de comportamiento simples, hasta los movimientos coordinados y la formación de patrones de bandada. Dentro de los fenómenos a ser descritos en la sección siguiente, se encuentran: el movimiento y navegación en grandes grupos (bandadas y cardúmenes), la coordinación para la construcción de grandes estructuras (termiteros), los métodos de decisión colectiva (abejas y hormigas), el auto-sellado de capilares y crecimiento adaptativo.

Motivación en los insectos sociales e inspiración : En los insectos sociales, los individuos no se encuentran informados acerca del estado global de la colonia [114]. No existe un líder que guíe a los otros individuos para alcanzar sus metas. El conocimiento del enjambre se encuentra distribuido a través de todos los agentes. Los insectos sociales son capaces de intercambiar información y por ejemplo, comunicar la información de una fuente de alimento en una zona favorable para el forrajeo o la presencia de peligro para sus compañeros. La comunicación implícita a través de los cambios realizados en el medio ambiente se llama *estigmergia* [135] [52]. En este proceso, los insectos modifican sus comportamientos debido a los cambios realizados por sus compañeros en el ambiente. Esto puede ser visto en la construcción de nidos de termitas, donde los cambios en el comportamiento de los trabajadores se encuentra determinado por la estructura del nido [43]. Es así como la organización emerge de la interacción entre los individuos y el ambiente.

Inteligencia de enjambre : Como una nueva área de investigación, la inteligencia de enjambre ha atraído la atención de los investigadores desde que el concepto fue propuesto en 1980. Actualmente, se ha convertido en una frontera interdisciplinaria y el foco de muchas áreas de la ciencia incluyendo la inteligencia artificial, economía, sociología y biología, entre otros. Se ha observado hace mucho tiempo que algunas especies sobreviven en la naturaleza, tomando como ventaja el poder de los enjambres, en lugar de la sabiduría de los individuos. Los individuos en los enjambres naturales no son muy inteligentes, pero son capaces de completar tareas complejas a través de la cooperación y la división del trabajo. Estos individuos no son necesariamente insensatos, pero son relativamente simples en comparación con la inteligencia global lograda por el conjunto. Algunos comportamientos nunca observados en un solo individuo pueden surgir cuando varios de ellos comienzan a cooperar o competir. La inteligencia de enjambre es un "biónico suave" de los enjambres naturales, i.e. simula las estructuras sociales y las interacciones de los enjambres más que la estructura de un individuo en la concepción de inteligencia artificial tradicional.

Características de los enjambres naturales : Como los enjambres de robots se encuentran mayoritariamente inspirados en los enjambres que se encuentran en la naturaleza, son una buena referencia para el análisis de los enjambres naturales. La primera hipótesis es un poco personificada [76] y asume que cada individuo tiene una identificación única para la cooperación y comunicación.

2.1. Comportamiento colectivo en la naturaleza

La mayoría de las investigaciones acerca de la inteligencia de enjambre se inspiran en la forma en que los insectos sociales, peces o mamíferos interactúan para formar agrupaciones en la vida real. El número de integrantes dentro de una de estas agrupaciones varían desde unos pocos individuos que viven en pequeñas áreas, a colonias altamente organizadas que pueden ocupar grandes territorios. El comportamiento que nace de las interacciones dentro de estos grupos muestra gran flexibilidad y robustez, en tareas que van desde la planificación de trayectorias, construcción de nidos y muchos otros comportamientos observados en enjambres naturales.

Los individuos que conforman las agrupaciones en la naturaleza muestran muy pocas habilidades, sin embargo, los comportamientos grupales que emergen de las agrupaciones pueden llegar a ser muy complejos, como la migración de las aves, los cardúmenes de peces y las colonias de abejas. Completar dichas actividades por parte de un solo individuo resulta imposible, pero un grupo de individuos coordinados pueden desarrollar las tareas fácilmente.

Los investigadores han centrado sus esfuerzos en la observación de los comportamientos grupales inteligentes que surgen en grupos de individuos a través de la comunicación y transmisión de información. En esta sección se describen seis tipos de comportamiento colectivo presentados en la naturaleza, tales como: la navegación en grandes grupos, toma de decisiones colectivas, coordinación en grandes trabajos de construcción, inteligencia de enjambre, auto-sellado y crecimiento adaptativo.

Desplazamiento colectivo

La agregación de grandes grupos de animales para trasladarse de manera coordinada es uno de los primeros tipos de comportamiento en los que se piensa cuando se habla de comportamiento colectivo. Por ejemplo, los cardúmenes, bandadas y manadas. Uno de los efectos producidos durante la agregación de individuos es que la dependencia de las reglas locales suprime las variaciones del comportamiento individual, permitiéndo que este tipo de comportamiento pueda ser analizado mediante modelos matemáticos y estadísticos empleados para describir fenómenos físicos de objetos inanimados [80]. Por ejemplo, la marea de gente que sale de un tren sigue el camino de menor resistencia y se puede comparar con el agua que fluye cuesta abajo.

Muchas especies de aves tienden a volar en bandadas, así, uno de los ejemplos más ampliamente estudiados es el comportamiento exhibido por los estorninos comunes (*Sturnus Vulgaris*) dado que sus bandadas se vuelven extremadamente grandes, llegando hasta los mil individuos. Dentro de estas agregaciones, a pesar de que los pájaros vuelan muy cerca unos de otros a altas velocidades (aproximadamente 70 $\frac{km}{hr}$), rara vez se ve que las aves se golpeen entre sí. Un fenómeno espectacular producido por las bandadas de estorninos es la ***Murmuración***[1], cuando cerca del atardecer grandes agrupaciones de estas aves son vistas moviéndose como rápidas nubes negras (Ver Figura 2.1). Cuando los halcones de caza atacan la bandada, provocan rápidos cambios a nivel local haciendo que "la nube" cambie de dirección rápida y abruptamente. Este tipo de comportamiento no sólo es visto cuando las aves se encuentran en vuelo, si no que cuando la bandada está aterrizando se puede presenciar un un comportamiento similar y aparentemente muy controlado[80]. A pesar del gran número de aves en la bandada, todos los pájaros aterrizan a corta distancia sin colisionar.

Figura 2.1: Fenómeno de Murmuración que involucra un gran número de estorninos visto en la costa oeste de Dinamarca. Fuente: *https://www.sortsafari.dk*

[1] adaptado del Danés "Sort sol", que significa "sol negro"

Reynolds [129] da una explicación detrás del ágil comportamiento de los estorninos en términos de tres reglas simples que cada pájaro obedece: separación, alineación y cohesión. La regla de separación asegura de que las aves no se hacinen en la bandada. Si el número de aves dentro de una región se torna muy alto, los integrantes que se encuentran en la periferia evitan entrar dentro de esa región. Esto da pie para la repulsión de corto alcance entre los individuos, resultando en una zona de exclusión para cada ave. En un estudio empírico [81], se encontró que la zona de exclusión tiene una medida de 0.38 $[m]$ de radio independiente del tamaño de la bandada. La regla de alineamiento causa que cada ave vuele en la misma dirección que los pájaros adyacentes. Finalmente, la regla de cohesión conduce a las aves en la periferia a una posición de forma tal que la bandada no se rompa o separe. La distancia y la dirección de los individuos dentro del rango de interacción determina la vecindad de cada pájaro. De esta manera, los compañeros de bandada que están fuera de la vecindad local son ignorados. Las tres reglas han reportado dar resultados razonables y conducir a simulaciones realistas, pero modelos más acabados incluyen incluso la evación de obstáculos [120].

La habilidad para actuar de forma coordinada y retener la adherencia del grupo cuando son atacados por un ave de presa tiene ventajas significativas para los individuos dado que el depredador no será capaz de focalizar su ataque. En general, una respuesta colectiva es el sello de una distribución ordenada auto-organizada opuesta al orden centralizado presentado en sistemas jerarquicos donde los individuos siguen a un líder. La habilidad de actuar colectivamente surge de las normas de comportamiento individuales simples descritas en el párrafo anterior: Cada pájaro típicamente interactúa con siete de sus vecinos más cercanos (distancia de interacción). El grado en el cual los pájaros no interactuantes correlacionan su comportamiento depende de que tan bien es transmitida la información a través de las interacciones individuales. En la mayoría de los sistemas físicos y biológicos, la distancia de correlación es significativamente más grande que la distancia de interacción, pero más corta que el tamaño del sistema. Sin embargo, en las bandadas de estorninos la distancia de correlación es tan grande como el grupo, independiente del número de individuos dentro del grupo, el comportamiento de todos los pájaros está correlacionado, lo que se conoce como ***Correlación de escala libre***[2][2]. Quizás sorprendentemente, los grupos no son tan densos como aparecen al observador humano desde el piso. La densidad es de 0.04 - 0.8 pájaros por metro cúbico (comparados con la densidad molecular en los materiales, esto se parece más a un líquido o a un gas) y la distancia de un ave a su vecino más cercano en rangos de 0.7 a 1.5 [m] [81].

Por otro lado, la distancia más cercana en cardúmenes de peces no depende del tamaño del grupo, como sucede en las bandadas de pájaros [100]. Sin embargo, el comportamiento colectivo en los cardúmenes sigue muy de cerca el comportamiento detallado en las bandadas de pájaros. Los peces individuales interactúan con sus vecinos mostrando comportamiento de parejas (emparejan su comportamiento con el del vecino) y preferencia posicional (mantienen una distancia constante o posición relativa con sus vecinos)[73]. Al igual que ocurre en las bandadas de pájaros, se piensa que este comportamiento otorga ventajas a los individuos para protegerse de la predación.

[2] *scale-free correlation*

Toma de decisiones colectivas

Una persona con conocimientos limitados en biología podría pensar que la toma de decisiones en grandes sociedades de insectos siguen la clásica estructura jerarquica conocida desde las cultura humanas pre-democráticas y el mundo de los negocios, donde un líder individual organiza y controla el comportamiento de sus seguidores. Así las grandes decisiones estratégicas como la manera de asignar las tareas, las prioridades de alimentación y la selección de un nuevo nido deberían estar bajo el control directo de la hormiga o abeja reina. De cualquier modo, este no es el caso. Dichas decisiones son alcanzadas por medio de acciones colectivas de individuos siguiendo una serie de reglas de comportamiento. Por ejemplo, la forma en la cual una colonia de hormigas determina el camino más corto a una fuente de alimento, se basa en la concentración crítica de feromonas dejadas por las hormigas recolectoras. La regla simple de comportamiento de seguir la ruta con la mayor concentración de feromonas puede explicar la elección correcta entre el camino más corto y el camino más largo hasta la misma fuente de alimento. Si se piensa en dos hormigas recolectoras que continuamente depositan feromonas a medida que siguen los caminos, con una hormiga recolectando la comida tomando el camino más corto y la otra hormiga tomando el camino más largo, entonces la hormiga que sigue el camino más corto demorará menos y por lo tanto pasará más veces por un mismo punto que la hormiga que tomó el camino más largo. Como las recolectoras irán dejando continuamente un camino de feromonas a medida que se mueven, la concentración de feromonas en el camino más corto se volverá automáticamente más alta a medida que el tiempo pase. Las nuevas recolectoras escogerán por tanto el camino más corto, que será el que tendrá la mayor concentración de feromonas.

Sin embargo, en esta sección se centrará el foco en otro comportamiento bien estudiado, enjambreamiento y selección del nido en las abejas productoras de miel. Las productoras de miel viven en grandes colonias con una reina y más de cien mil trabajadoras. En primavera, la reina comienza a dejar huevos destinados a convertirse en nuevas reinas. Poco después que estas nuevas reinas nazcan, la antigua reina forma un enjambre y deja el antiguo panal. El enjambre usualmente se asienta en un árbol por un par de días mientras las abejas exploradoras son enviadas a buscar nuevos lugares para asentarse. La selección del lugar del nido y el despegue del enjambre se determinan por al menos cuatro métodos de comunicación diferentes: (1) El baile[3], las abejas exploradoras advierten la posición de los posibles lugares para establecer el enjambre a los otros buscadores, mediante una coreografía característica que se muestra en la Figura 2.2. (2) Señales vibratorias, originadas por las abejas exploradoras que hacen vibrar sus cuerpos rapidamente, estimulando a otras abejas y haciéndolas más receptivas a otras señales involucrándolas en las desición para seleccionar el lugar donde anidar y en la selección del momento de despegue del enjambre. (3) Una señal aparece luego de seleccionar el lugar donde anidar y se compone de un ruido agudo producido cuando dos abejas presionan sus cuerpos entre ellas. Esto es una señal para que el resto de las abejas caliente sus músculos de vuelo. (4) Un zumbido se produce poco antes del despegue y se hace por las mismas abejas que realizan las señales de ruido. Este sonido se produce cuando las abejas corren a través de la superficie del enjambre mientras hacen zumbar las alas, lo que eventualmente gatilla el despegue.

[3]Waggle Dance

Figura 2.2: Patrón descrito durante el Waggle-Dance. Fuente: Fotografía tomada desde *http://beeinformed.org/2013/04/6489/*

A continuación se describe el proceso de toma de decisiones, i.e., antes de la señal de ruido y antes que la carrera de los zumbidos se produzca. Una vez que el enjambre se establece en un lugar provisorio, un número de abejas exploradoras deja el enjambre para volar en diferentes direcciones en busca de lugares de anidación definitivos. Una vez localizados, vuelven al enjambre para advertir de la existencia de estos lugares mediante el baile inicial [4]. Mientras mejor sea el sitio, más largo y vigoroso será el baile. Por medio de este baile, se establece una señal de alto para la búsqueda de otros lugares de anidación [124]. La señal consiste en que una abeja toca su cabeza contra la abeja que baila mientras que emite una señal vibratoria de alta frecuencia. Aunque una sola señal de alto no es suficiente para detener el baile inicial, mientras más sean las señales de alto que una abeja danzante recibe, mayor es la tendencia a abandonar el baile inicial. Las señales inihibitorias y de advertencia cambian a través del tiempo, indicando la preferencia por los distintos lugares (medida como el número de exploradoras que se encuentran haciendo el baile inicial para ellas). Eventualmente, dentro del día, un consenso es alcanzado a medida que la mayoría de las exploradoras advierten del mismo lugar [113]. Cuando este consenso es alcanzado, algunas exploradoras comienzan a emitir la señal de ruido agudo, seguida por el zumbido para terminar con la migración del enjambre hasta el sitio de anidación final. Cabe hacer notar que la reina no está involucrada en la selección del nuevo nido. Por el contrario, está decisión está determinada por un proceso democrático donde convergen las preferencias de la mayoría de las abejas exploradoras.

[4] Waggle Dance

Coordinación en grandes trabajos de construcción

Muchas especies diferentes dentro del reino animal construyen estructuras utilizando ya sea materiales de construcción secretados por ellos mismos (como la seda utilizada por las orugas para construir capullos o el caso de las arañas que construyen sus impresionantes telarañas aéreas) o utilizando materiales recolectados en el exterior (como las piedras y hojas en el caso de las larvas de frigáneas[5] o las ramitas de los nidos de las aves). La mayoría de las construcciones animales son fabricadas de forma individual a modo de refugio, protección, atracción de la pareja o para capturar presas. Sin embargo, los ejemplos más impresionantes de arquitectura animal son el resultado de la colaboración de un gran número de individuos. Por ejemplo, los termiteros en la Savana Africana pueden alcanzar alturas de varios metros, lo cual excede en más de 200 veces la altura de las termitas trabajadoras. Al hacer la analogía con los humanos, un termitero de estos es tan alto como el Empire State Building en Nueva York.

Los termiteros no son impresionantes solamente cuando son vistos desde el exterior, sino que ellos también contienen muchas adaptaciones sofisticadas para poder mantener el clima en el interior [133]. Existen sistemas de ventilación complejos a través del nido que cuentan con sistemas cerrados y abiertos (chimeneas y rendijas), donde ocurre el intercambio de gases, en especial en galerías cercanas a otras superficies. La interacción entre las variaciones temporales de velocidad, dirección y turbulencia del aire causada por la morfología del sistema de ventilación en el termitero, ha sugerido causar un intercambio gaseoso como el que ocurre en los pulmones de los humanos [151]. Este sistema de ventilación pasiva tiene un potencial biomimético significativo para el uso en manejo del clima interno en edificios.

Sistemas de ventilación similares inducidos por el viento han sido encontrados en grandes nidos de hormigas, comprometiendo a millones de trabajadores individuales [26]. La estructura de estos nidos gigantes es impresionante, pero igualmente fascinante es la pregunta de como estas elaboradas estructuras pueden ser construidas por hormigas trabajadoras sin supervisión central ni jerárquica. Recientes investigaciones muestran que la estructura de los nidos aparece como una propiedad emergente desde las acciones individuales de miles de trabajadores que indirectamente se comunican a través de modificaciones en el ambiente. El concepto de que las acciones de los inividuos modifican el ambiente, que a su vez, modifica el comportamiento de los individuos, es conocido como ***Estigmergia***. Otra aplicación de este sistema de comunicación en las colonias de hormigas, puede ser visto en los "cementerios de hormigas" encontrados en las afueras de los nidos de muchas especies. Las grandes acumulaciones de hormigas muertas es el resultado del comportamiento de eliminación de los cadáveres, que se produce cuando estos son recogidos y depositados en los lugares donde la densidad de cadáveres es mayor [106]. Esto significa que una vez que una pila de cuerpos se comienza a formar, la probabilidad de que el número de hormigas muertas que son arrojadas ahí aumenta, resultando en que todos los cadáveres son arrojados sólo en ciertos lugares, en grandes pilas de cuerpos.

[5]Tricóptero o frigáneas son un orden de insectos endopterigotos, emparentados con los lepidópteros (mariposas).

Auto-sellado

Tanto los animales como las plantas tienen mecanismos que sellan heridas rápidamente. Primero el cuerpo se asegura de cerrar y sellar la herida para posteriormente iniciar la auto-reparación. Un ejemplo representativo de esto corresponde a la auto-reparación de los capilares sanguíneos humanos que involucra tres pasos: (1) La formación de un tacón en el agujero (llamado *homeostásis primaria*), (2) Sellado del daño (llamado *homeostásis secundaria*) por lo que el tacón existirá solamente hasta que la herida esté sellada y (3) Sellado de la herida. En los humanos (y otros animales), células especializadas cubren los orificios que se producen en los capilares sanguíneos a través de una cascada de eventos. Cuando la membrana interna de los capilares (*endotelio*) es dañada, la sangre entra en contacto con las fibras de colágeno en el tejido. Esto activa pequeños fragmentos de células en la sangre llamadas plaquetas (*trombocitos*) que se hinchan y se vuelven pegajosas para luego liberar varios factores de coagulación. Estos activan más plaquetas que juntas forman un tapón. Además, inician la formación de fibras de fibrina que forman un coágulo que sella el capilar y actúa como soporte para el sellado, donde el tejido de cicatrización se construye. La formación de fibrina incluye una secuencia de acciones. Los factores de coagulación activan la pro-enzima protrombina que circula en la sangre, convirtiéndose en la enzima trombina que causa que la proteína plasmática fibrinógeno se polimerice en fibras de fibrina.

El proceso de coagulación es muy complejo y desde un punto de vista biomimético, podría ser atractivo si un efecto similar pudiese ser alcanzado con agentes más simples. Una parte central de la secuencia de acciones durante la coagulación es cuando una sustancia (plaquetas = agente A) se pone en contacto con otra sustancia (colágeno = agente B) que causa una reacción (las plaquetas se ponen pegajosas y tapan el agujero). Una aplicación posible de este proceso de sellado puede ser en las llantas de bicicletas y automóviles. La llanta podría estar construida como una estructura de doble capa. Un agente líquido ubicado entre ambas capas podría reaccionar y reparar la fuga de aire. Otra posible aplicación podría ser el sellado de las fugas en las cañerías de gas en las cuales un recubrimiento externo sería el encargado de la reparación cuando se vea expuesto al gas.

Crecimiento adaptativo

En parrafos anteriores, se discutió cómo el comportamiento de seres individuales siguiendo reglas simples da pie para que emerjan propiedades óptimas como las decisiones en el comportamiento colectivo para conseguir alimento y selección de un nido. Resultados similares se encuentran si es que se mira el crecimiento de organismos unicelulares y multicelulares simples, donde las reglas para el crecimiento local pueden resultar en patrones de crecimiento óptimos generales sin la necesidad de manejar ningún tipo de información global. Las esponjas marinas son organismos unicelulares eucariotes[6] que previamente se pensaba pertenecían al reino fungi pero que ahora son clasificados como *protistas* tales como pasa con las algas. Las esponjas marinas comienzan su vida como amebas individuales que luego se juntan y se fusionan para formar grandes colonias que usualmente miden varios centímetros de largo

[6]células que contienen estructruas complejas encerradas en membranas, tales como el núcleo o las mitocondrias

pero pueden alcanzar hasta un metro. Ellas se alimentan de mircroorganismos que habitan en plantas muertas y se reproducen con esporas [14, 152].

Los investigadores de la universidad de Hokaido en Japón han encontrado inspiración en las esponjas marinas (*Physarum polycephalum*) para sus simulaciones de redes de tráfico. Los investigadores han desarrollado un modelo matemático biológicamente inspirado que captura el corazón de los mecanismos utilizados en la estructuración de las esponjas marinas [12] [13]. La planificación convencional de las redes requiere control centralizado y que la información global se encuentre disponible. En contraste, las esponjas marinas son auto-organizadas y basan su optimización solamente en la información disponible localmente. Los investigadores compararon el desempeño de las redes de trenes de Tokio, con una red creada por las esponjas marinas. La red de trenes de Tokio fue simulada ubicando copos de avena en las posiciones que correspondían a las estaciones del tren y restringiendo el área de crecimiento usando luz[7]. Los resultados muestran que la esponja marina tiene un desempeño de red que es comparable al que se presenta en la red de ferrocarriles. Tero et al. [14, 12] desarrollaron un modelo matemático que aumenta o disminuye la capacidad de una conexión entre dos nodos dependiendo del flujo. Cuando este modelo se utiliza para generar una red para el trazado de las estaciones de trenes en Tokio, el modelo se comportó un poco mejor que el trazado actual y que el desarrollado por la esponja marina. El modelo descentralizado de planificación puede ser usado para muchos tipos de redes, que van desde los sistemas de ferrocarriles a redes de potencia, sistemas financieros, redes de información y redes de distribución.

Inteligencia de enjambre

La *Inteligencia de Enjambre*[8] se refiere al comportamiento inteligente que emerge desde el comportamiento colectivo de un gran número de agentes autónomos. En biología, este termino es ampliamente usado en referencia al comportamiento a nivel de colonia visto en los insectos sociales. Por ejemplo, mientras las hormigas de fuego (*Solenopsis invicta*) de forma individual se ahogan rápidamente al caer en una piscina de agua, los grupos de hormigas de fuego se unen para formar balsas que pueden flotar en la superficie del agua durante días [92]. De forma similar, la hormigas guerreras en su fase migratoria fabricarán hormigueros temporales (bivouacs) consistentes en varios cientos de miles de trabajadores que juntan sus cuerpos para proteger a la reina y las crías [123]. Sin embargo, donde este comportamiento colectivo es más impresionante y biomiméticamente más relevante es en el control de tráfico a gran escala de las hormigas de forrajeo. La mayoría de las especies de hormigas usan huellas específicas que van desde el lugar de anidación hasta el lugar donde se encuentra el alimento. Debido a la atracción de las feromonas, este será el camino elegido por la mayoría de las hormigas de forrajeo hacia el alimento. Así en cualquier momento, un gran número de forrajeras se están moviendo directo a la fuente de alimento y simultáneamente un número similar se está moviendo en dirección opuesta hacia el nido.

En un grupo de hormigas guerreras, las colisiones potenciales y atascos de tráfico son particularmente grandes. Las colonias de hormigas guerreras, *Eciton burchelli*, tienen más de 200.000 forrajeras y acarrean más de 3.000 porciones de alimento por hora en caminos que

[7]La esponja marina evita la luz y por lo tanto evitará estas áreas
[8]Swarm Intelligence

superan los 100 [m] de longitud [148]. Estas hormigas solucionan el problema de tráfico en una forma que es similar a la forma en la que los humanos estructuramos nuestros sistemas de calles. Ellos forman distintos carriles, donde el tráfico de los forrajeros que vuelven al nido con alimento tiene lugar por un carril central y las hormigas que se van en busca de alimento se mueven por dos carriles paralelos ubicados a ambos lados del camino central. Esto conlleva el beneficio adicional que las hormigas que llevan un cargamento valioso se encuentran más protegidas de los depredadores. La separación de flujo está formada por los trabajadores siguiendo reglas de interacción simples y teniendo velocidades de viraje asimétricas, donde las hormigas que retornan con alimento tienen una velocidad de giro menor que las hormigas que salen del nido, como se puede apreciar en la Figura 2.3. Sin embargo, la mayoría de

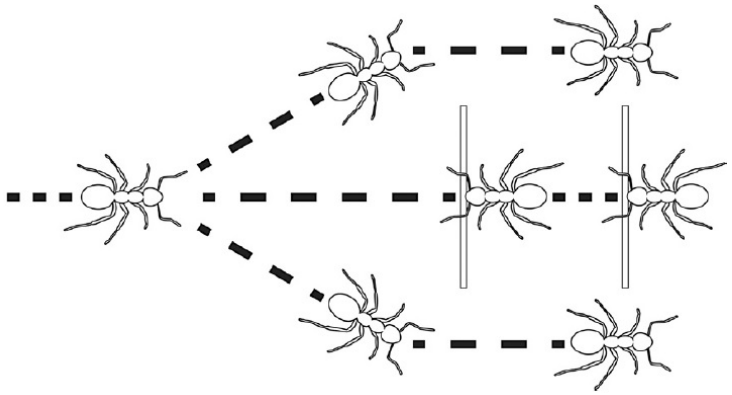

Figura 2.3: Distribución del tránsito en hormigas guerreras. Fuente: Torben Lenau y Thomas Hesselberg [80].

las hormigas no separa sus caminos en carriles como lo hacen las hormigas guerreras. Por el contrario ellas confían en otro tipo de reglas de control de tránsito. La hormiga negra europea de jardín, *Lasius niger*, usa una única ruta cuando hay baja densidad de individuos en el camino, donde las maniobras individuales para evitar las colisiones evita la formación de atascos de tráfico. Sin embargo, a medida que la densidad de trabajadores aumenta, se forma un nuevo camino por separado antes que ocurran retrasos por atascos de tráfico [4]. Nuevamente esta bifurcasión emerge como resultado de las reglas de comportamiento individual de los trabajadores donde a mayores densidades, las hormigas comienzan a empujarse formando nuevos caminos.

2.2. Enjambres robóticos

La robótica de enjambre es un campo de la multi-robótica, que utilizando la inteligencia de enjambre, consigue que un gran número de robots se coordinen de forma distribuida y descentralizada. Se basa en el uso de reglas locales y robots simples en comparación con la tarea a desarrollar. Sus algoritmos de control se encuentran inspirados en los insectos sociales. Lo que se busca es que el comportamiento colectivo complejo surja de la interacción entre individuos y entre los individuos y el medio ambiente. Esto permite que un gran número de robots simples puedan realizar tareas complejas de una manera más eficiente que un solo robot, dando robustez y flexibilidad al grupo.

La investigación sobre la robótica de enjambre se basa en diseñar grandes cantidades de robots relativamente simples, pequeños y de bajo costo a fin de tener una gran población. Un componente clave del sistema es la comunicación entre los individuos del grupo que es normalmente de caracter local y garantiza que el sistema sea escalable y robusto.

Las aplicaciones potenciales de la robótica de enjambre incluyen las tareas que exigen la miniaturización, como las tareas de monitoreo distribuido en micro maquinaria o en el cuerpo humano. La robótica de enjambre puede también participar en tareas que abarcan grandes espacios, requiren mucho tiempo o son peligrosas para el ser humano o para los propios robots, como en caso de desastres naturales, búsqueda y rescate, aplicaciones militares, etc.

En esta sección, se da una visión general de la robótica de enjambre, describiendo sus principales propiedades y características, comparándola con sistemas multi-robot generales.

Características principales

A continuación se listan algunas de las propiedades observadas en los insectos sociales y deseables para los sistemas multi-robot. Estos criterios además permiten establecer una clara diferencia con otros sistemas multi-robot que serán analizados en las secciones siguientes.

- **Interdependencia**: los robots deben ser incapaces o ineficientes respecto a la tarea principal que ellos deben resolver, puesto que necesitan colaborar para poder cumplir o mejorar su desempeño.
- **Robustez**: se refiere a la capacidad de trabajar incluso si algunos de sus integrantes fallan o si hay perturbaciones en el ambiente.
- **Escalabilidad**: el número de integrantes dentro del enjambre no debe ser una limitante. El número de individuos puede variar desde unos pocos hasta miles de ellos, sin afectar el desempeño global.
- **Autonomía**: los individuos en los sistemas robóticos de enjambre deben ser autónomos, es decir, deben ser capaces de interactuar y moverse en el ambiente. Con esas funciones claves, los mecanismos cooperativos inspirados en los enjambres naturales pueden ser introducidos en los enjambres robóticos.
- **Descentralización**: con un buen grupo de reglas cooperativas, los individuos pueden completar tareas sin controles centralizados que comprometan la escalabilidad o flexibilidad del enjambre. Al mismo tiempo, el enjambre puede verse beneficiado en ambientes donde la comunicación es interrumpida o retardada, mejorando la velocidad de reacción y precisión del enjambre.
- **Homogeneidad**: en un sistema de robótica de enjambre, los robots deben dividirse en el menor número de roles posible, mientras que el número de robots que actúa por cada rol debe ser lo más grande posible. El rol asumido indica la estructura física del robot o de otros estados que no se pueden cambiar de forma dinámica durante la tarea.

  Cabe señalar que un estado en una máquina de estados finitos no cuenta en nuestra definición. Ya que esta indica un enjambre, sin importar lo grande que sea, no se considera como robótica de enjambre si los roles de los robots se dividen meticulosamente. Por ejemplo, el fútbol de robots por lo general no se considera robótica de enjambre,

ya que a cada persona en el equipo se le asigna un papel especial durante el juego.

- **Flexibilidad**: el enjambre debe ser capaz de crear diferentes soluciones para diferentes tareas y sus integrantes deben ser capaces de cambiar de rol dependiendo de las necesidades del momento [112]. Luego, un enjambre con alta flexibilidad puede lidiar con diferentes tareas utilizando la misma estructura[9] y con cambios menores en la programación[10]. Los individuos en los enjambres naturales muestran diferentes habilidades y tipos de cooperación estrategica cuando tratan con distintas tareas. La robótica de enjambre debe tener esta flexibilidad, especialmente en tareas similares, como el forrajeo o búsqueda. El enjambre puede cambiar de estratégia de acuerdo al ambiente. Los robots pueden adaptarse al ambiente a través del aprendizaje de movimientos mejorando la estrategia.
- **Comunicación y detección local**: debido a las restricciones de construcción y costo, los robots en el enjambre generalmente tienen un rango limitado de detección y comunicación y, por lo tanto, todo el enjambre se distribuye en el medio ambiente. Y es que el uso de las comunicaciones globales daría lugar a una disminución significativa de la escalabilidad y flexibilidad, ya que el coste de la comunicación aumenta exponencialmente a medida que la población crece. Sin embargo, ciertas comunicaciones globales de control son aceptables, por ejemplo, la actualización de las estrategias de control o envío de las señales de terminales, siempre y cuando no se utilice en la interacción entre individuos.

Ventajas de la robótica de enjambre

Las ventajas y características de un sistema de enjambre de robots son presentados comparando un solo robot especializado y otros sistemas similares compuestos por multiples individuos. Estas características son medianamente similares a las que se encuentran en la naturaleza.

Comparación con robots especializados : Para completar una tarea sofisticada, un solo robot debe ser diseñado con una estructura complicada y modulos de control los que resultan en diseños de alto costo en construcción y mantenimiento. Los robots individuales son vulnerables especialmente cuando se presenta la rotura de una pequeña parte de éste ya que puede afectar todo el sistema, haciendo difícil predecir su comportamiento. El enjambre de robots puede tener las mismas capacidades que un robot especializado mediante la cooperación entre sus agentes, además tiene la ventaja de la reusabilidad y bajos costos de construcción y mantenimiento.

Cabe señalar que otra ventaja corresponde al paralelismo, haciéndolo especialmente apto para tareas a gran escala.

Por otro lado, mientras la construcción de un único robot encuentra inspiración en el comportamieto humano, la robótica de enjambre se inspira en los animales sociales. En

[9] hardware
[10] software

función de lo anterior y teniendo en cuenta las restricciones de la tecnología actual, resulta difícil simular las interacciones humanas usando máquinas o computadores, mientras que los mecanismos de cooperación utilizados en los grupos de animales son más fáciles de aplicar. Esto otorga a la robótica de enjambre un futuro interesante para tratar problemas complejos y de gran escala.

Las ventajas de la robótica de enjambre, al ser comparada con un robot, especializado se resumen a continuación:

- **Paralelismo**: el tamaño de la población de la robótica de enjambre es usualmente grande y puede tratar con múltiples tareas al mismo tiempo. Esto indica que el enjambre puede desarrollar tareas que involucran multiples objetivos distribuidos en un gran rango del ambiente.
- **Escalabilidad**: la interacción en el enjambre es local, lo que permite a los individuos unirse o dejar una tarea en cualquier momento sin interrumpir todo el enjambre. Este puede adaptarse al cambio en la población a través de esquemas de reasignación de tareas implícitos sin la necesidad de ninguna operación externa. Esto también indica que el sistema es adaptable a diferentes tamaños de población sin niguna modificación del software o hardware que es muy útil para aplicaciones en la vida real.
- **Estabilidad**: similar a la escalabilidad, los sistemas de enjambres robóticos no son gravemente afectados incluso cuando parte del sistema abandona la tarea debido a factores de fuerza mayor. El enjambre todavía puede trabajar hacia el objetivo de la tarea aunque el desempeño de los individuos se pueda ver reducido de manera inevitable al tratarse de menos robots. Esta característica es especialmente útil para las tareas en un ambiente peligroso.
- **Económicas**: como se mencionó anteriormente, el costo de la robótica de enjambre es significativamente menor en el diseño, fabricación y mantenimiento. Todo el sistema es más barato que un robot único complejo, incluso si cientos o miles de robots conforman el enjambre. Los individuos en el enjambre pueden ser producidos de forma masiva mientras que un solo robot requiere mecanizado de precisión.
- **Eficiencia Energética**: dado que los individuos en el enjambre son mucho más pequeños y más simples que un robot gigante, el consumo de energía por parte de un robot es muy bajo en comparación con el tamaño de la bateria. Esto significa que el tiempo de vida del enjambre se amplía. En un entorno sin alimentación eléctrica o donde la electricidad por cable está prohibida, la robótica de enjambres puede ser mucho más útil que un solo robot tradicional.

En conclusión, la robótica de enjambre puede ser aplicada a problemas sofisticados que involucran grandes cantidades de tiempo, abarcar grandes espacios, tareas multi-objetivo y ambientes con cierto grado de peligro, cuyas aplicaciones típicas incluyen, control, búsqueda y rescate, minería, geología, aplicaciones militares y transporte cooperativo.

La robótica de enjambre puede completar estas tareas a través del comportamiento cooperativo que emerge de los individuos, mientras que un único robot apenas podría adaptarse a esta situación. Motivo por el cual la robótica de enjambre se ha convertido en un campo de investigación importante en la última década.

Tabla 2.1: Comparación de la robótica de Enjambre y otros sistemas

	Robótica de Enjambre	Sistemas Multi robot	Red de Sensores	Sistemas Multi-agente
Tamaño de la población	Varía en rango amplio	Pequeño	Fijo	En pequeño rango
Control	Descentralizado y autónomo	Centralizado o remoto	Centralizado o remoto	Centralizado o jerárquico o red
Homogeneidad	Homogéneo	Usualmente heterogéneo	Homogéneo	Homogéneo o Heterogéneo
Flexibilidad	Alto	Bajo	Bajo	Medio
Escalabilidad	Alto	Bajo	Medio	Medio
Ambiente	Desconocido	Conocido o Desconocido	Conocido	Conocido
Movimiento	Si	Si	No	Raro
Aplicaciones Típicas	Rescate post desastre	Transporte	Vigilancia	Manejos de redes de recursos
	Aplicaciones militares	detección	Cuidados médicos	Control distribuido
	Aplicaicones peligrosas	Football robótico	Protección ambiental	

Diferencias con otros sistemas multi-agente Existen muchas áreas de investigación inspiradas en los enjambres naturales que a menudo se confunden con la robótica de enjambre, tales como sistemas multi-agente y redes de sensores. Estas áreas de investigación también utilizan el comportamiento cooperativo que surge de múltiples agentes dentro del grupo para desarrollar tareas especializadas. Sin embargo, hay varias diferencias entre estos sistemas, que permiten su distinción de manera fundamental, como se muestra en la Tabla 2.1.

Comparación con sistemas multi-agente : Algunas de las características de los sistemas multi-robóticos pueden ser extrapoladas a los sistemas robóticos de enjambre. Se toma prestado del trabajo de Ronald Arkin [19] un listado de ventajas y desventajas de los sistemas multi-robóticos comparados con los sistemas de un solo robot. Así, las ventajas de los acercamientos de la multi-robótica destacan los siguientes puntos:

- Desempeño mejorado: Si las tareas pueden ser desagregadas utilizando paralelismo, los grupos pueden dividir la tareas para desarrollarlas de manera más eficiente.
- Tareas permisibles: los grupos de robots pueden realizar ciertas tareas que son imposibles para un solo robot.
- Monitoreo distribuido: El rango de monitoreo de un grupo de robots es más amplio que el rango de un solo robot.
- Acción distribuida: Un grupo de robots puede actuar en diferentes lugares al mismo tiempo.
- Tolerancia a los fallos: Bajo ciertas condiciones, la falla de un solo robot dentro de un grupo de ellos no significa que la tarea no pueda ser desarrollada gracias a la redundancia del sistema.

Por otro lado, entre las desventajas se encuentran:

- Interferencia: Los robots en un grupo pueden interferirse entre ellos, debido a las colisiones, oclusiones, etc.
- Incerteza con respecto a la acciones de los otros robots: La coordinación requiere saber qué están haciendo los otros robots. Si esto no está claro, los robots pueden competir en vez de cooperar.
- Costo general del sistema: El hecho de usar más de un robot puede hacer que los costos sean mayores. Este no es el caso ideal de los sistemas de enjambre robóticos, los cuales tratan de usar robots más simples y baratos, con el fin de mantener el costo por debajo

Tabla 2.2: Ejes de la taxonomía presentada por Dudek et al. [55]

Ejes	Descripción
Tamaño del colectivo	Número de robots en el colectivo
Rango de comunicación	Máximo rango de comunicación
Topología de comunicación	De los robots dentro del rango de comunicación, aquellos con los que se puede comunicar
Ancho de banda de comunicación	Cuanta información puede ser enviada entre robots
Reconfigurabilidad Colectiva	La velocidad con la cual la organización colectiva puede ser modificada
Habilidad de procesamiento	Modelo computacional usado por los robots
Composición colectiva	Si es que los robots son homogéneos o heterogéneos

de un único robot más complejo para que desarrolle la misma tarea.

Clasificación de los enjambres robóticos dentro de los sistemas multi-robot

A continuación, se clasifican y caracterizan los enjambres robóticos usando las taxonomías mejor conocidas y las clasificaciones dadas en la literatura se sistemas multi-robóticos. Didek et al. [55] definen una taxonomía en la cual diferentes ejes son usados para caracterizar sistemas multi-robóticos en función de sus propiedades. Los ejes de la taxonomía se encuentran resumidos en la Tabla 2.2, que fue extraída desde el documento de su autor [55]. Usando esta clasificación, las propiedades son asignadas en cada uno de los ejes para un sistema de enjambre robótico genérico, aunque estas propiedades pueden depender del sistema en concreto.

- *Tamaño del colectivo* es *SIZE-INF*, esto es, el número de robots con $N \gg 1$, en oposición a *SIZE-LIM*, donde el número de robots N es pequeño al compararlo con la tarea o ambiente. Esto expresa la escalabilidad pedida en los sistemas de enjambre robótico.
- *Rango de comunicación* es *COM-NEAR*, los robots pueden comunicarse solamente con aquellos que estan lo suficientemente cerca.
- *Topología de comunicación* para un sistema de enjambre puede ser generalmente *TOP-GRAPH*, los robots se encuentran unidos en un gráfico general.
- *Ancho de banda de comunicación* es *BAND-MOTION*, los costos de comunicación son de la misma magnitud que el costo de mover los robots entre locaciones.
- *Reconfigurabilidad Colectiva* es generalmente *ARR-COMM*, esto es, la reagrupación coordinada con miembros que se comunican, pero también puede ser *ARR-DYN*, arreglo dinámico, las posiciones pueden cambiar de forma arbitraria.
- *Habilidad de procesamiento* es *PROC-TIME*, donde el modelo computacional es máquina de "turing" equivalente[11].
- *Composición colectiva* es *CMP-HOM*, lo que significa que los robots son homogéneos.

Locchi et al. [79] presentaron una taxonomía estructurada en diferentes niveles. El primer nivel es *Cooperación* que incluye una situación en la cual muchos robots desarrollan una tarea

[11]Máquina de Turing equivalente o *Turing machine equivalent*, es un artefacto hipotético con capacidad de memoria infinita, propuesto por Alan Turing en 1936.

común. El segundo nivel es *Conocimiento*, el cual distingue si es que los robots saben de la existencia de otros robots consciente o inconscientemente. El tercer nivel es *coordinación*, para diferenciar el grado en el cual los robots toman en cuenta las acciones ejecutadas por otros robots. De acuerdo a los autores esto puede ser: *fuertemente coordinados*, *débilmente coordinado* o *no coordinado*. El último nivel es *organización*, el cual distingue entre sistemas centralizados, donde existe un robot que está a cargo de organizar el trabajo de los otros robots y *sistemas distribuidos*, donde los robots son autónomos en sus decisiones, esto es, no hay líderes. De acuerdo a esta taxonomía, los enjambres robóticos son: *Cooperativos, Conscientes, Fuertemente Coordinados (también pueden ser débilmente coordinados) y Distribuidos*.

Cao et al.[153] definen una taxonomía no muy completa, pero que será detallada dado que resulta bastante intuitiva en su clasificación. En ella se diferencia entre sistemas *Centralizados* y *Descentralizados*. Los *Descentralizados* pueden ser *distribuidos*, si es que todos los robots son iguales con respecto al control o *jerárquicos* si es que existe una centralización local. También identifica entre individuos *homogéneos* y *heterogéneos*. Usando esta taxonomía los sistemas de enjambre son: *descentralizados*, *distribuidos* y *homogéneos*.

Plataformas para la robótica de enjambre

El diseño de la plataforma para desarrollar robótica de enjambre es un elemento clave para el algoritmo cooperativo que controla el comportamiento e interacciones de todos los individuos. En el modelo que describe el comportamiento de estos sistemas, los robots deben tener algunas funciones básicas, como la capacidad de sentir, comunicar, desplazar, etc.

El modelo se divide en tres módulos: el intercambio de información, comportamiento básico y avanzado. El intercambio de información entre los individuos juega el papel más importante en el modelo. Y es que los robots intercambian información entre sí y la propagan a todos sus compañeros a través de comportamientos autónomos que resultan en la cooperación a nivel global. En algunos casos, se introduce algún tipo de comando central, pero el enjambre debe ser capaz de completar la tarea en caso que falle la comunicación central.

Módulo de intercambio de información

El intercambio de información es inevitable cuando los robots cooperan unos con otros y es la parte central para el control del comportamiento de enjambre. Hay tres formas de intercambio de información en el enjambre [27]: Comunicación directa, comunicación a través del ambiente y detección. Más de un tipo de interacción puede ser usado en un enjambre, por ejemplo, cada robot puede monitorear el ambiente y se comunicarse con su vecino.

Comunicación directa : La comunicación directa es similar a las redes inalámbricas y también consiste en dos tipos: *peer-to-peer*[12] y *Broadcast*[13]. Gracias al desarrollo de disposi-

[12]igual a igual
[13]emisión

tivos móviles, muchas tecnologías existentes pueden ser adoptadas inmediatamente. Hawick et al. [156] propusieron un diseño de robots para enjambre de tres ruedas utilizando tanto dispositivos de ethernet y bluetooth. Sin embargo, los sensores inalámbricos cuestan casi la mitad de lo que cuesta el robot completo. Otra desventaja de tal sistema es que el ancho de banda requerido crece de forma exponencial a medida que aumenta el número de individuos. De esta manera, la comunicación directa en el enjambre debe ser limitada.

Aunque varias tecnologías inalámbricas están disponibles, los protocolos y topologías especializadas para la robótica de enjambre permanecen sin descubrir. Las comunicaciones en la robótica de enjambre deben tomar todas las ventajas de la detección local y habilidades de movimiento, mientras prestan especial atención a impulsar el comportamiento cooperativo entre individuos y la topología dinámica del enjambre [41].

Comunicación a través del ambiente El ambiente puede actuar como intermediario para la interacción entre los robots. Los robots dejan sus huellas en el ambiente luego de realizar una acción para estimular a los otros robots que pueden interpretarlas, sin comunicación directa entre los individuos. De esta forma, las acciones subsecuentes tienden a reforzar y construirse entre ellas, promoviendo la generación espontánea de actividades a nivel de enjambre. El cual imita a las hormigas o abejas al interactuar por medio de feromonas virtuales. Este esquema interactivo, no presenta problemas ante la escalabilidad del grupo como en el caso de la comunicación directa, pero se ve limitado por la capacidad del medio ambiente para soportar las feromonas.

Ranjbar-Sahraei et al [98] pusieron en práctica un enfoque de cobertura utilizando marcadores en el medio ambiente sin comunicación directa. Grushin y Reggia [61] resolvieron el problema de auto-ensamblaje de estructuras 3D pre-especificadas a partir de bloques de diferentes tamaños con un enjambre de robots utilizando estigmergia.

Sensores : Los individuos deben ser capaces de diferenciar entre los integrantes del enjambre y el resto de los objetos del ambiente utilizando sus propios sensores, si es que están diseñados para esta función. El tema principal es la forma de integrar todos los sensores en el enjambre de manera eficiente para la cooperación.

Cabe señalar que la principal diferencia entre la comunicación y la detección está en si los individuos envían los mensajes de forma activa o reciben los mensajes de forma pasiva. La comunicación más precisa y abundante requiere de hardware y sincronización más compleja. Por lo anterior, el costo del ancho de banda, la energía y el tiempo de transferencia crecen extremadamente rápido a medida que la población aumenta. Así, el modelo cooperativo de la robótica de enjambre debe tratar de simplificar la comunicación y el uso de la detección tanto como sea posible. La presencia de colores, la luminancia y las posiciones relativas se pueden utilizar para detectar obstáculos y pueden proporcionar información rica sin comunicación. En algunas de las tareas, el enjambre puede intercambiar toda la información sólo con los sensores.

Módulos de comportamiento básico

El comportamiento básico de los individuos incluye funciones como el movimiento y planificación local, la cual es una de las diferencias más significativas de la robótica de enjambre con los sistemas multi-agente o con las redes de sensores. Los robots y el control de su comportamiento son homogéneos y forman los fundamentos del comportamiento de grupo. Basados en lo que se capta por medio de la comunicación o a través de sus sensores, los robots calculan sus movimientos siguientes. Con un excelente módulo de control, el enjambre puede confiar menos en la comunicación con la ayuda de la predicción y las interacciones más directas, en lugar de emisión. El enjambre puede mejorar el rendimiento con menos intercambio de información y teniendo una alta escalabilidad.

Módulos de comportamiento avanzado

Los robots en sistemas robóticos de enjambre complejos pueden tener funciones adicionales, pero no destinados a las tareas de descomposición, asignación de tareas, aprendizaje adaptativo, etc [122]. Los robots con estas funciones en hardware pueden simplificar el diseño del algoritmo y conducir a un diseño físico más complejo del robot real. Los robots también pueden lograr funciones similares con algoritmos de cooperación cuidadosamente diseñados. En ese sentido, la práctica de tales funciones en el hardware o software depende de los diseños físicos de los robots, controladores y sensores con el fin de hacer un mejor uso de los componentes.

La asignación de tareas y el aprendizaje suelen ser bastante importantes para un enjambre de robots. La descomposición de tareas y la asignación pueden mejorar en gran medida la eficiencia de las tareas especialmente complejas. Al comparar los costos y beneficios de los diferentes tipos de enfoques de asignación de tareas en un mundo ruidoso, el aprendizaje también es útil ya que los parámetros del mecanismo de control son difíciles de ser sintonizados. Así con la ayuda del aprendizaje auto-adaptativo y métodos de optimización, el enjambre muestra mejor adaptabilidad en los diferentes ambientes.

Plataformas experimentales para los enjambres robóticos

A continuación se describen las diferentes plataformas robóticas utilizadas en los experimentos más relevantes con respecto a la robótica de enjambre encontrados en la literatura.

1. Robot Khepera [50], diseñados para propósitos de investigación y educacionales, fue desarrollado en la École Polytechnique Fédérale de Lausanne (EPFL, Suiza), ampliamente usado en el pasado, actualmente ha caído en desuso.
2. Robot Khepera III (http://www.k-team.com/) [69], diseñados por K-Team junto con EPFL;
3. Robot e-puck (http://www.e-puck.org/) [51], diseñado en EPFL para propósitos educacionales;

4. El robot miniatura Alice [150] también desarrollado en EPFL;
5. Robot Jasmine (http://www.swarmrobot.org/) [109] desarrollado bajo el proyecto I-Swarm;
6. Robot I-Swarm (http://www.i-swarm.org/) [94] muy pequeño, también desarrollado por el proyecto i-swarm;
7. S-Bot (http://www.swarm-bots.org/) [49], muy versátil, con muchos actuadores, desarrollados en el proyecto Swarm-bot;
8. Kobot (http://www.kovan.ceng.metu.edu.tr/) [6] desarrollado en Middle East Technical University (Turquía);
9. SwarmBot (http://www.irobot.com/) [83], diseñado por la compañía i-Robot para investigadores.

En la tabla 2.3 la columna *Sistema de posicionamiento relativo* indica si los robots poseen la habilidad de determinar la posición relativa de sus vecinos más cercanos. Estos sensores son útiles y necesarios en muchas tareas de los enjambres robóticos. Algunos de ellos están basados en la emisión de una señal infrarroja (por uno de los robots) estimando la distancia de sus vecinos mediante la intensidad de la señal recibida [69] [7]. Otros trabajan emitiendo pulsos de ultra-sonido al mismo tiempo que una señal de radio y estiman la distancia tomando en cuenta la diferencia de tiempo entre la recepción de ambas señales [131] [78]. También hay robots que usan una cámara para detectar y estimar la posición de los robots cercanos que están equipados con marcadores [110] [9].

Modelamiento de enjambres robóticos

El modelamiento es un método usado en muchos campos de investigación para entender mejor el funcionamiento interno de los sistemas que están siendo investigados. En particular, esta práctica ayuda a la robótica de enjambre, dado que el tiempo y dinero suelen ser limitantes para escalar estos experimentos. En cambio al suponer algoritmos escalables para cientos o miles de individuos dentro de la población, las mediciones se pueden hacer de una forma más fácil y rápida. Considerando las características de los enjambres robóticos, los métodos de modelamiento están divididos en cuatro tipos: Microscópicos, macroscópicos, basados en sensores y basados en la inteligencia de enjambre. Estos métodos se detallan a continuación:

Modelamiento microscópico : los robots y sus interacciones son modeladas como máquinas en estado finito. El comportamiento de cada robot está definido por varios estados y las condiciones de transferencia están basadas en los datos de ingreso de la comunicación y de los sensores. Como el modelo se basa en el comportamiento de cada robot, la simulación debe correr varias veces para obtener el comportamiento promedio del enjambre.

En gran parte de las investigaciones de enjambres robóticos, se usa el modelo probabilístico microscópico, dado que el ruido puede ser modelado como una probabilidad dentro del modelo. En el modelo probabilístico microscópico, las probabilidades son evaluadas a partir de los experimentos en los robots reales y el modelo es iterado con esas probabilidades para

Tabla 2.3: Resumen de las principales plataformas robóticas en experimentos de enjambre

Nombre	Size (mm) (diam.) o (lxw)	Actuadores	Capacidad de Computación	Sensores	Comunicación	Sistema de pos. relativo	Desarrollo
Khepera	55	Con ruedas (Acc. Diferencial)	Motorola MC68331	8 IR	RS232 Por cable	—	Investigación Comercial
Khepera III	120	Con ruedas (Acc. Diferencial)	PXA-255 (400MHz) Linux y dsPICs	11 IR 5 ultrasonido	WIFI y Bluetooth Por cable	Expansion (Basado en IR)	Investigación Comercial
e-puck	75	Con ruedas (Acc. Diferencial)	dsPIC	11 IR Anillo de contacto Cámara de color	Bluetooth	Expansion (Basado en IR)	Investigación y Edu. Comercial Open Source
Alice	20 x 20	Con ruedas (Acc. Diferencial)	Microchip PIC	IR proximidad y luz Cámar lineal	Radio (115 kbit/s)	—	Investigación No Comercial
Jasmine	23 x 23	Con ruedas (Acc. Diferencial)	2ATMega microcontroladores	8 IR	IR	Integrado (Basado en IR)	Open Source Research
Robot I-swarm	3 x 3	Micro pies	[a]	[b]	[c]	—	Investigación No Comercial
S-Bot	120	Con ruedas (Acc. Diferencial) Dos sujetadores	XSclae (400 Mhz) Linux PICs	15 Proximidad Omnicámara Micrófono, Temp	WIFI	Basado en cámaras	Investigación No Comercial
Kobot	120	Con ruedas (Acc. Diferencial)	PXA-255 (200 Mhz) y PICs	8 IR Cámara de color	Zigbee	Integrado (Basado en IR)	Investigación No Comercial
SwarmBot	127 x 127	Con ruedas (Acc. Diferencial)	ARM (40 Mhz) y FPGA 200 kgate	IR, sensores de contacto de luz, cámara	Basados en IR (Locales)	Integrado (Basado en IR)	Investigación No Comercial

[a] Inforamción no disponible
[b] Información no disponible
[c] Información no disponible

la transferencia de estados en la simulación con el fin de predecir el comportamiento del enjambre.

Modelamiento macroscópico : es un método de modelamiento opuesto al modelamiento microscópico. En el modelamiento macroscópico, el comportamiento del sistema es definido como una ecuación diferencial y un estado del sistema representa el número promedio de robots en ese estado en un lapso de tiempo [156].

La diferencia principal entre el modelamiento microscópico y el modelamiento macroscópico es la granularidad de los modelos. Y es que el modelamiento microscópico para el comportamiento a nivel individual es usado para simular el comportamiento de grupo, mientras que el modelo macroscópico simula el comportamiento a nivel de enjambre.

Por otro lado, el modelo microscópico itera el comportamiento de enjambre mientras que el modelo macroscópico puede entregar el estado final del enjambre. De esta forma, el modelo macroscópico puede tener una mirada global al enjambre, entretanto el modelo microscópico puede mostrar en detalle el comportamiento de enjambre.

Modelamiento basado en sensores : los sensores y actuadores del robot están modelados como componentes principales del sistema tal como los objetos en el ambiente. Entonces las interacciones de los robots son modeladas de la forma más realista y simple posible. Este método de modelamiento es el más usado y el más antiguo en la experimentación robótica.

Modelamiento a partir de los algoritmos de inteligencia de enjambre : estos esquemas han sido introducidos dentro de la robótica de enjambre por muchos investigadores. Dado que los robots usan los mismos (o muy similares) esquemas que estos algoritmos, los modelos y otros métodos utilizados para analizar estos algoritmos, que son bastante más maduros que la robótica de enjambre, pueden ser usados directamente para la investigación en robótica. El algoritmo usado más utilizado comúnmente en la robótica de enjambre es la optimización de partículas de enjambre (PSO[14]) que imita el proceso de agrupación de los pájaros.

Se puede encontrar, obviamente, que se mantienen muchos beneficios entre PSO y los enjambres robóticos. Ya que un mapeo entre las partículas y los robots se puede presentar fácilmente.

Además de PSO, los investigadores también introducen otro algoritmo de inteligencia de enjambre dentro de la robótica de grupos. Muchos modelos exitosos de enjambre se inspiraron en las colonias de hormigas, sin embargo, todavía hay muchos problemas cuando un esquema cooperativo se introduce en la robótica de enjambre. Los esquemas en estos algoritmos consideran las interacciones más globales e introducen grandes cantidades de movimientos aleatorios de alta diversidad. Algunos planes también contienen las operaciones para restablecer las posiciones de los agentes de búsqueda. Sin embargo, estas operaciones no están

[14]Particle Swarm Optimization

disponibles para la robótica de enjambre.

Modelos disponibles para la robótica de enjambre

Existen muchos simuladores de robots móviles que pueden ser usados para experimentos multi-robóticos y más concretamente para experimentos de robótica de enjambre. Ellos se diferencian no solo en aspectos técnicos si no que también en licencias y costos.

A continuación se resumen los principales simuladores junto con los comentarios de su uso en aplicaciones de robótica de enjambre.

1. *Player/Stage/Gazebo* (http://playerstage.sourceforge.net/) [20] es un simulador de código abierto con capacidades multi-robóticas y un gran grupo de robots y sensores disponibles listos para usar. El uso para experimentos de robótica de enjambre es analizado en simulaciones 2-D [127] con muy buenos resultados. Las escalas de tiempo son aproximadamente lineales con tamaños de población que pueden llegar hasta 100.000 robots simples. Trabaja en tiempo real con 100 robots corriendo un programa simple. Es una buena solución para experimentos de robótica de enjambre.
2. *Webots* (http://www.cyberbotics.com/) [85] es un simulador realista comercial que permite la simulación de multi-robots, con modelos de robots reales ya construidos. Esto es la simulación de elementos 3D y colisiones. De acuerdo a experiencias realizadas, el desempeño decae rápidamente cuando se trabaja con más de 100 robots, haciendo que las simulaciones con un gran número de robots se vuelva complicada.
3. *Microsoft Robotic Studio* [70] es un simulador desarrollado por Microsoft Corporation, que permite la simulación con multi-robots. Requiere de una plataforma Windows para poder funcionar.
4. *SwarmBot3D* [49] es un simulador de multi-robótica diseñado específicamente para robots *S-Bot* del proyecto *SwarmBot*.

Algoritmos de la robótica de enjambre

Un algoritmo de robótica de enjambre debe encajar y hacer pleno uso de las características de la robótica de enjambre. El algoritmo debe explotar la cooperación entre robots y compartir algunas características con un sistema de enjambre de robots. Cinco características de algoritmos de enjambres robóticos son especialmente enfatizados en esta sección: Simpleza, escalabilidad, descentralización, localidad y paralelismo.

Simpleza : Dado que la capacidad de cada robot es limitada, el algoritmo debe ser tan simple como sea posible. Un algoritmo simple puede ayudar a reducir el costo de cada uno de los robots. Incluso el comportamiento complejo de la robótica de enjambre puede emerger a partir de un algoritmo cooperativo simple bien diseñado. En la mayoría de los casos, los robots son considerados como máquinas estáticas finitas con sólo un par de estados.

Escalabilidad : Los algoritmos diseñados para la robótica de enjambre deben ser escalables para cualquier tamaño de enjambre. En un algoritmo, el diseñador debe considerar el permitir que los robots se unan y abandonen el enjambre dinámicamente. Todas las operaciones donde los robots interactúan con el resto del enjambre deben ser diseñadas cuidadosamente para no afectar el desempeño del enjambre cuando la población es muy grande.

Descentralización : Los robots en el enjambre son autónomos y por tanto, el algoritmo también debe serlo. Un algoritmo debe siempre evitar cualquier control centralizado o externo. A pesar que un individuo puede ser afectado por otros, este debe tomar la decisión por sí mismo. Es muy posible que un algoritmo descentralizado sea escalable.

Local : La comunicación local y las interacciones son características especiales de la robótica de enjambre. El algoritmo debe también seguir esta regla al mismo tiempo que las reglas para la escalabilidad. Dado que los robots pueden simular la comunicación global y los sistemas de interacción, usando sistemas locales con esquemas especialmente diseñados, existe un cierto retraso en la propagación de la información en el enjambre, por lo que el uso directo de las operaciones globales debe ser evitado.

Paralelismo : El enjambre normalmente consiste en muchos robots. A pesar de esto, los algoritmos deben permitirle a los individuos desarrollar varias tareas distintas al mismo tiempo, lo cual es una de las ventajas de los enjambres de robots. De acuerdo a esta característica, los científicos han propuesto muchos algoritmos para la robótica de enjambre. De cualquier forma la investigación en este campo se encuentra en sus inicios y el interés de los invetigadores aún se basa en el desempeño en tareas simples, como la formación de patrones o la evasión de obstáculos. Aún no se ha propuesto un marco unificado. Así, y a medida que las investigaciones avancen, se espera que las características presentadas por estos conjuntos sean mejor aprovechadas como escalabilidad, flexibilidad y robustez. En ese momento los investigadores se podrán centrar en aplicaciones más complejas, resultando en algoritmos más aplicables a problemas de la vida real.

Comportamiento experimental básico y tareas fundamentales de la robótica de enjambre

En las décadas pasadas, los enjambres han sido aplicados en varias tareas como la detección de olores, monitoreo móvil de redes, operaciones médicas, vigilancia y operaciones de búsqueda y rescate. Las tareas en estas aplicaciones son muy complicadas y resulta difícil proponer soluciones directas.

A raíz de lo anterior y para llevar a cabo dichas tareas, los investigadores han propuesto una serie de tareas simples como la agregación, navegación y evasión de obstáculos. Dentro de estas tareas, la agregación resulta ser la más importante y fundamental.

Aparentemente, coordinar un gran número de individuos a nivel de enjambre con reglas individuales no es una tarea simple, por lo tanto, el comportamiento de grupo emergente de la interacción de los robots con el medio ambiente y otros robots ha sido el interés principal de la investigación desde que se comenzó a trabajar en esta área. El agrupamiento ha sido ampliamente observado en muchos enjambres naturales e incluso en seres humanos. Dado que las criaturas en los grupos sociales muestran una gran diversidad dentro de su población debido a las diferencias en edad, morfología, estado nutricional, personalidad y estado de liderazgo de los individuos, es sorprendente que ellos logren agruparse con reglas limitadas e interacciones en un grupo tan heterogéneo.

Una colección de las obras experimentales más representativas de enjambres robóticos se presentan en esta sección. Los diferentes resultados experimentales se organizaron agrupándolos en función de las tareas o las conductas realizadas por los enjambres. Algunos de los comportamientos, como la agregación y el movimiento colectivo, son bastante básicas y constituyen un nivel anterior para las tareas más complejas. Ellas se presentan en orden de complejidad creciente.

Agregación : Con el fin de realizar otras tareas, como movimiento colectivo, auto-ensamblaje y la formación de patrones o para intercambiar información, los robots deben ser capaces de reunirse inicialmente. Este problema de agreación ha sido estudiado por varios investigadores. Trianni et al. [126] realizan experimentos utilizando un algoritmo evolutivo en robots simulados *S-Bot*. Las entradas sensoriales son los sensores de proximidad y los micrófonos. Los actuadores son los motores y los altavoces. Una de las soluciones evolucionadas es escalable. Bahceci y Sahin [140] también utilizan algoritmos evolutivos y robots de *S-Bot* simulados, produciendo resultados escalables, aunque su trabajo está más bien centrado en algoritmos evolutivos que en la agregación. Soysal y Sahin [141] utilizan un algoritmo basado en una máquina de estados finitos probabilístico para la agregación. Ellos desarrollaron un modelo macroscópico de la misma y lo compararon con resultados simulados. Dimarogonas y Kyriakopoulos [144] proponen un algoritmo de agregación distribuido basado en funciones potenciales que consisten en: una fuerza repulsiva para evitar obstáculos y una fuerza de atracción para la agregación. Ellos analizaron matemáticamente su convergencia y realizaron simulaciones con nueve robots. Por último, Garnier et al. [108] implementaron un modelo biológico basado en la agregación de robots *Alice*.

Posicionamiento y navegación : Después de la estrategia de agrupación, el control de la dirección del enjambre es el tópico más abordado por los investigadores del área, para aplicaciones de migración y búsqueda. Hasta ahora, un gran número de investigaciones han sido realizadas para dirigir el enjambre a posiciones estratégicas y propagar información en él.

Individuos informados : Una estrategia común para poder dirigir un enjambre consiste en los "individuos informados". Esto fue observado primero en la naturaleza por Couzin y sus colegas [35] quienes condujeron un estudio sobre liderazgo efectivo y toma de decisiones en grupos de animales, publicando su trabajo en *Nature*. En su experimento, sólo unos pocos

individuos eran concientes de la dirección de destino. Los resultados obtenidos muestran que estos individuos informados son capaces de conducir el grupo hacia destino. A partir de entonces, esquemas similares han sido introducidos a la robótica de enjambre. McLurkin [71] desarrolló una estrategia en su tesis de magister para desarrollar la tarea de seguir a un líder dentro de una formación lineal. Los robots se alinean en la topología siguiendo al predecesor y guiando a los sucesores. El líder se guía por otros controles hasta el destino final del grupo. Así, el grupo es capaz de formar la línea sin ningún tipo de orden externa y puede evitar los obstáculos en el medio ambiente así como lidiar con las fallas de comunicación que se puedan experimentar. Nasseri y Asadpour [90] investigaron los efectos de controlar un enjambre cuando solo una pequeña fracción de los individuos tenía conocimiento del destino final. Los portadores de la información no podían transmitir la información directamente, sin embargo, el enjambre es capaz de agruparse hacia el objetivo deseado en la simulación.

Otro comportamiento de agrupación auto-organizado presentado por enjambres de robots fue presentado por Turgut et al. [125] donde no se utilizaron sensores o conocimiento a priori del lugar de destino. La simulación mostró que solo con interacciones locales, los robots podían compartir una dirección común en un proceso auto-organizado hasta que el ruido de detección excedió una cierta medida. En su trabajo de seguimiento [62] estudiaron cómo el enjambre podía ser dirigido hacia la dirección deseada cuando eran guiados por algunos individuos de forma externa. Los resultados son concordantes con los que fueron predichos usando el modelo modal [35]. Los dos trabajos fueron evaluados de forma cualitativa y en simulaciones en un ambiente con obstáculos. Stranieri [102] estudió el comportamiento auto-organizado en agrupaciones de dos tipos de robots: alineados y no alineados. Un robot alineado tiene la capacidad de ponerse de acuerdo en una dirección común con sus vecinos. Un enjambre heterogéneo de estos dos tipos de robots puede alcanzar un buen desempeño de grupo en las simulaciones si la estrategia de control de movimeinto y mecanismos de interacción son cuidadosamente diseñados.

Funciones de campo potencial : Otra de las estrategias de control para la formación de enjambres comúnmente utilizada, es la función de campo potencial.

Ge y Fua [116] presentaron un enfoque escalable y flexible para controlar eficazmente la formación de un grupo de robots. Ellos introdujeron trincheras artificiales de potencial y representaron la formación de estructuras en términos de filas y vértices, en vez de nodos. Los robots eran atraídos y se movían a lo largo del fondo de la trinchera de potencial y se distribuían con respecto a la densidad automáticamente. En su trabajo de seguimiento, Fua et al. [54] investigaron la formación de filas mediante comunicación limitada. Así el intercambio de información es clasificado en dos escalas: la escala rápida y la escala lenta. La escala primera (rápida) implica únicamente la comunicación en tiempo real local, mientras que en la segunda, la información es menos exigente y puede ser recolectada sobre un período de tiempo más prolongado. De esta manera, el enjambre se guía de forma progresiva en la formación específica de una manera más eficiente. La agregación, búsqueda de alimento y el control de la topología fueron investigados por Gazi et al. [56]. Ellos consideraron un entorno significativamente más realista y más complicado con agentes dinámicos no-holonómicos comparados con otros estudios.

Métodos alternativos : Rothermich et al. [101] desarrollaron un método de mapeo y localización distribuida basados en enjambres de robots. Dado que el enjambre no comparte un sistema de coordinación global, debe moverse unido para mantener un sistema virtual. En el enjambre, algunos robots sirven como balizas si se encuentran en una nueva zona de búsqueda y dan marcha atrás en la función de mapeo y búsqueda si es que hay un número suficiente de balizas cercanas. Con tal esquema, el enjambre puede mantener el sistema de coordinación para dibujar un mapa con alta precisión de una manera distribuida.

Correll y Martinolli [91] desarrollaron un sistema de inspección inteligente con sensores locales ubicados en los robots. En la estrategia que propusieron, parte de los robots en el enjambre actuaban como balizas y la estrategia se compara con otros sistemas sin balizas. Ellos también analizaron el sistema con modelos probabilísticos microscópicos y macroscópicos.

Por su parte, Stirling et al. [117] presentaron una nueva metodología de vuelo para la navegación autónoma y vuelos dirigidos a un objetivo en ambientes interiores desconocidos usando un enjambre de robots voladores. El acercamiento es completamente descentralizado y se basa sólo en la detección local sin posicionamiento global, comunicación o información previa del entorno.

Marjovi et al. [82] propusieron un método de navegación que utiliza conexiones inalambricas para guiar al enjambre en la búsqueda de fuentes de olor. Al menos tres individuos dentro del enjambre actúan como balizas que emiten las coordenadas a todo el enjambre para mantener un sistema de coordenadas globales, mientras los otros buscan la fuente de olor. El inconveniente de esta investigación es que las balizas están transmitiendo la coordinación en una gran área mientras que los otros robots deben detectar la distancia con las balizas desde una larga distancia, lo que requiere una implementación de costo elevado.

Evasión de obstáculos : La evasión de obstáculos también es considerada una tarea básica importante en las sociedades de enjambre constituidas por robots. En la mayoría de las investigaciones, algún tipo de función potencial ha sido aplicada a los robots. El enjambre se maneja alrededor de obstáculos de acuerdo a los campos de potencial. Khatib [93] fue el primero en introducir el concepto de evasión de obstáculos en tiempo real en 1986. Él usó un campo de potencial artificial que variaba en el tiempo para los objetos que se movían. Esta solución convertía de forma satisfactoria el problema de planificación tradicional de alto nivel en operaciones distribuidas en el tiempo incluso en ambientes complejos. Algunos ejemplos recientes también utilizan este esquema. Das et al. [39] propusieron un acercamiento que cambiaba entre varios controladores, dependiendo del estado del robot para la evasión de obstáculos. Do [74] utilizó una función potencial para evitar colisiones en el enjambre. La función altera la trayectoria de los robots si es que no se encuentran en la dirección seleccionada. Otra alternativa señala que los obstáculos generan un campo repulsivo similar al de los núcleos atómicos, donde los robots juegan el rol de los electrones que lo orbitan. Un modelo matemático basado en el destino del robot, velocidad y dirección de los obstáculos fue propuesto y optimizado usando PSO. Las simulaciones muestran que el método es mejor que los métodos de campo de potencial artificiales, sin embargo esto requiere una gran cantidad de cómputos dado que cada robot mantiene un modelo de PSO por separado.

Dispersión : El espíritu de la dispersión es distribuir los robots en el espacio para cubrir tanta área como sea posible, sin perder la conectividad entre ellos. El enjambre puede trabajar, cuando está disperso, como un sensor distribuido, pero también sirve para exploración. La dispersión ha sido estudiada por diferentes investigadores usando tanto robots reales como simulados. Howard et al. [8] presentaron un algoritmo de campos potenciales para el despliegue de robots, en el cual los robots se encuentran repelidos por los obstáculos y por los otros robots. El acercamiento es distribuido y no requiere localización centralizada, produciendo una solución escalable. El trabajo fue desarrollado solamente en simulación. En [45], los autores proponen y testean en una simulación un algoritmo distribuido para dispersión, basado en la intensidad de las señales inalámbricas leídas y un enfoque de campo potencial. De acuerdo con ellos y aunque los robots no tienen información sobre la cercanía de los robots vecinos, el algoritmo consigue dispersarlos con éxito.

Ludwing y Gini [146] y Damer et al. [104] también usaron la intensidad de la señal inalámbrica para la dispersión de un enjambre de robots. Ellos utilizaron un algoritmo más elaborado que tiene en cuenta un gráfico de los robots vecinos y la intensidad de la señal recibida. Ellos alcanzaron resultados satisfactorios en entornos más complicados que los utilizados en [45]. El hecho que solamente la intensidad de la señal inalámbrica sea necesaria en estos algoritmos, los hace muy atractivos dado que puede ser aplicada en robots muy simples que no disponen de sistemas de posicionamiento relativo. El problema de la cobertura está relacionado con la dispersión. Los robots necesitan dispersarse y detectar las fronteras del medio ambiente. Correll et al. [87] presentaron un conjunto de algoritmos distribuidos y escalables para cubrir los límites de elementos colocados en un patrón regular. Ellos mostraron, basados en resultados experimentales y con un máximo de 30 robots *Alice* que el rendimiento de cobertura mejora al aumentar el número de robots.

Formación de patrones : La formación de patrones es el problema de crear una forma global cambiando las posiciones de los robots. Dado que esta sección trata de los acercamientos a la robótica de enjambre, los ejemplos explicados contemplan solamente información local. Martinson y Payton [137] describen un algoritmo que usando una orientación de referencia común para los robots y leyes de control local actuando en ejes ortogonales crean patrones cuadrados. Chaimowicz et al. [77] muestran un algoritmo para la colocación de robots en diferentes formas y patrones definidos por funciones implícitas. Los robots usan un enfoque distribuido basado en la información local para colocarse en el contorno deseado. Los algoritmos son probados tanto en simulaciones como en robots reales. En [10] se muestra un marco para montar un enjambre de robots dada cierta morfología. El uso de la capacidad de los robots para unirse, demuestra cómo los robots *S-Bot* se auto-ensamblan formando morfologías globales. El algoritmo se distribuye por completo utilizando solamente información global.

Movimiento colectivo : El movimiento colectivo es el problema de cómo coordinar un grupo de robots y moverlos juntos de manera cohesiva [142]. También puede servir como un comportamiento básico para las tareas más complicadas. Se puede clasificar en dos tipos: las formaciones y agrupaciones. En el primero, los robots deben mantener posiciones y orientaciones predeterminadas entre ellos. En el segundo, en las agrupaciones, las posiciones relativas entre robots no están restringidas. Existen muchas arquitecturas de movimientos

colectivo, pero sólo los que permiten la escalabilidad son de interés en este documento. En [130] [132] se presenta y analiza el marco fisicomimético (PF[15]) que permite crear una formación auto-organizada mediante el uso de leyes inspiradas en la física. El controlador está totalmente descentralizado dado que cada robot percibe las posiciones relativas de sus vecinos y reacciona a fuerzas atractivas o repulsivas, formando retículas triangulares. El algoritmo es escalable, funcionando para docenas de robots. Lee y Nak [154] proponen un algoritmo distribuido para el movimiento colectivo basado en redes. Su convergencia se demuestra mediante el teorema de Lyapunov. En [143], se propone un algoritmo descentralizado para el movimiento colectivo basado en formaciones reticulares. Se prueba la estabilidad del algoritmo en un caso de estudio en particular. La evación de obstáculos es implementada mediante la partición de planes en regiones de Voroni.

Turgut et al. [5] proponen y estudian un algoritmo escalable y disribuído para el agrupamiento de robots. Se basa en la alineación de los robots y el control de la distancia entre ellos. El algoritmo es probado en simulaciones con más de 1000 robots y con un pequeño grupo de robots reales.

Asignación de tareas : El problema de la división del trabajo no es una tarea como las anteriores, pero sí un problema que puede surgir en los sistemas multi-robot y en particular en la robótica de enjambre. Jones y Mataric [147] presentan un algoritmo distribuido y escalable para la división del trabajo en enjambres de robots. Cada robot mantiene un historial de las actividades realizadas por otros robots basados en la observación y de forma independiente realizan una división del trabajo usando esta historia. A continuación, pueden modificar su propio comportamiento para dar cabida a esta división. En [48] los autores proponen dos métodos diferentes para la asignación de tareas en un enjambre robótico. Las tareas son previamente anunciadas por ciertos robots y algunos de ellos tienen que asistir a ellas simultáneamente.

El primer algoritmo se basa en un esquema de comunicación y tiene un mejor rendimiento que el otro, pero debido a la robustez limitada y la pérdida de paquetes de información puede ser menos escalable. El segundo es simple y reactivo y se basa en la interacción a través de señales de luz. McLurkin y Yamins [138] describen cuatro algoritmos diferentes para la asignación de tareas y los prueban utilizando 25 robots reales tipo *SwarmBot*. Los cuatro resultan exitosos y escalables, aunque requeren diferentes tipos de comunicación.

Búsqueda de fuentes : Los enjambres robóticos pueden ser muy útiles en tareas de búsqueda, especialmente en aquellas en que el patrón espacial de la fuente puede ser complejo como en el caso del sonido o el olor. El problema de la localización del olor se estudia en [11], donde los robots buscan la fuente de olor usando un algoritmo distribuido. Los experimentos fueron realizados tanto en simulaciones como con robots reales. En [68] los autores describen y prueban un algoritmo distribuido para localización estacionaria, i.e., fuentes invariantes en el tiempo. Ellos usan control de retroalimentación motivados por la teoría de función de minimización. Ellos exploraron dos situaciones: Una con comunicación global, en la cual los robots son capaces de encontrar la fuente de máximo global y en el segundo están restrin-

[15]Physicomimetics Framework

gidos a comunicación local, donde se encuentra el máximo local. Los experimentos fueron realizados en simulaciones.

Transporte colectivo de objetos : Los enjambres robóticos son muy prometedores en la solución de problemas de transporte de objetos. El uso de muchos robots puede representar una ventaja debido a la cooperación para manejar un objeto. Además, el uso de paralelismo para poder manejar varios objetos al mismo tiempo puede mejorar sustancialmente el desempeño en la tarea.

Kube y Bonabeau [139] se inspiraron en la forma en que las colonias de hormigas transportan sus presas, donde los individuos esperan por otros compañeros cuando deben transportar elementos demasiado pesados. En sus experimentos, desarrollados con robots reales, un grupo de seis robots es capaz de empujar un objeto de forma colectiva hacia un destino especificado mediante un algoritmo distribuido. Groβ y Dorigo [145] solucionan el problema de transportar distintos objetos por grupos de robots tipo *S-Bot* que tienen capacidad de auto-ensamblaje. Los algoritmos fueron sintetizados usando un sistema evolutivo. Los resultados experimentales en la simulación mostraron que el algoritmo es escalable para objetos más pesados usando grupos de robots más grandes (más de 16). Pero el desempeño no escala con el tamaño del grupo, dado que la masa tranportada por cada robot disminuye con el número de robots. En [3] los autores discuten y proponen un sistema de transporte colectivo donde los objetos se coleccionan y se guardan para un transporte posterior. Los robots en el enjambre pueden tener dos tareas distintas: recolectar los objetos y colocarlos en un carro o mover el carro de forma colectiva para trasladar los objetos.

Mapeo colectivo : El problema del mapeo colectivo, no ha sido ampliamente estudiado en la comunidad de investigadores de enjambres robóticos. Rothermich et al. [67] proponen y prueban (mediante simulación y robots reales) un método para la cartografía distribuida mediante un enjambre de robots. Cada robot puede asumir dos roles: moverse o señalar, estos roles son cambiados para el movimiento del enjambre. Además, los robots tienen una cierta confianza en su posición estimada de localización. Con esta información, las estimaciones de localización de otros robots y las mediciones del sensor construyen un mapa colectivo.

Campos de aplicación de la robótica de enjambre

El estudio de la aplicación de la robótica en la búsqueda de objetivos ha crecido sustancialmente en los últimos años, dado que este tipo de tecnología es preferida para las áreas de trabajo peligrosas o inaccesibles. Lo problemas implicados en la investigación de la robótica de enjambre se pueden clasificar en dos clases. Una clase de problema se basa principalmente en los patrones, como la agregación, la cartografía, la migración, las redes auto-organizadas, el despliegue de agentes distribuidos y cobertura de áreas. Otra clase de problemas se centra en las entidades del entorno, por ejemplo, la búsqueda de objetivos, la detección de las fuentes de olor, la localización de las venas de mineral, el forrajeo, el rescate de víctimas en zonas de desastre, etc. Además, la robótica de enjambres también puede estar involucrada en

problemas más sofisticados, en su mayoría híbridos de estas dos clases, incluyendo el transporte cooperativo, remoción de minas, la exploración de planetas y navegación en grandes territorios. Varios ámbitos posibles de aplicación de la robótica de enjambre que son muy adeacuados, se describen a continuación.

1. **Covertura de grandes áreas**: Los sistemas robóticos de enjambre se encuentran distribuidos y especializados para tareas que requieren gran covertura. Los robots en el enjambre se distribuyen en el medio ambiente y pueden detectar el cambio dinámico de toda la zona, tales como fugas de productos químicos o elementos contaminantes. Enjambres robóticos pueden completar estas tareas de mejor manera que una red de sensores, ya que cada robot puede patrullar un área en vez de quedarse quieto. Esto significa que el enjambre puede supervisar el área con menos agentes.

   Además de la supervisión, los robots en el enjambre pueden localizar la fuente, moverse hacia la zona y tomar decisiones rápidas. En caso de urgencia, los robot se pueden unir como un parche para bloquear la fuente como una solución temporal.

2. **Tareas peligrosas para los robots**: Gracias a la escalabilidad y estabilidad, el enjambre ofrece redundancia para hacer frente a tareas peligrosas. Los robots son muy baratos y se prefieren para áreas que probablemente dañen a los trabajadores. En algunas de las tareas, los robots pueden ser irrecuperables después de la tarea por lo que el uso de robots complejos de costo elevado resulta inaceptable, mientras que la robótica de enjambre compuesta por individuos de costo reducido puede proporcionar soluciones razonables. Por ejemplo, Murphy et al. [86] resumió el uso de enjambres robóticos en el rescate y la recuperación en la industria minera.

3. **Tareas que requieren escalamiento de la población**: la carga de trabajo de algunas tareas puede variar con el tiempo por lo que el tamaño del enjambre debe ser escalado en base a la carga de trabajo actual para mantener altos estándares en la eficiencia temporal y económica. Por ejemplo, para limpiar los residuos generados por una fuga luego del rompimiento del tanque contenedor, el enjambre debe mantener una población alta al comienzo y disminuir su población cuando la fuente de la fuga ya se encuentra controlada. El enjambre también se escala entre las distintas regiones si el progreso de estas regiones se desequilibra.

   Stormont [42] describe el potencial para el uso de los enjambres de robots autónomos para reaccionar en un lugar de catástrofe durante las primeras 24 hrs. Él resume que el enjambre puede hacer búsqueda de víctimas con mayor probabilidad de encontrar sobrevivientes e hizo algunas sugerencias para futuras investigaciones en esta área.

4. **Tareas que requieren redundancia**: La robustez de los sistemas de robótica de enjambre en general se beneficia de la redundancia del enjambre, es decir, la eliminación de algunos de los robots no tiene un impacto significativo en el desempeño global.

Una revisión de los diferentes trabajos de investigación y sus correspondientes resultados experimentales junto a una discusión del futuro de la robótica de enjambre en las aplicaciones del mundo real completa esta revisión bibliográfica.

2.3. Dinámica de movimiento en robots vibracionales

En la última década, la microrobótica se ha convertido en un campo de investigación. Dominios de aplicación, como la microfabricación, biotecnología y optoelectrónica [16] demandan plataformas micro-robóticas o miniaturizadas.

Principio de movimiento

El principio de movimiento se demuestra usando un sistema móvil simplificado de un grado de libertad, consistente en una plataforma de masa M. El mecanismo de movimiento utiliza una masa rotatoria excéntrica de masa m adosada al eje de un motor ubicado en O, montada en la plataforma como se muestra en la Figura 2.4. Se asume que la masa m rota en un plano vertical a una velocidad angular constante ω, en el punto O y que la plataforma se encuentra restringida a moverse solamente a lo largo del eje y. Un ciclo de operación se completa cuando la masa ha descrito un ángulo de 360^0 (grados). Las fuerzas gravitacionales y centrípetas producidas por la rotación de la masa se resuelven dentro del plano $y - z$, de la manera que sigue:

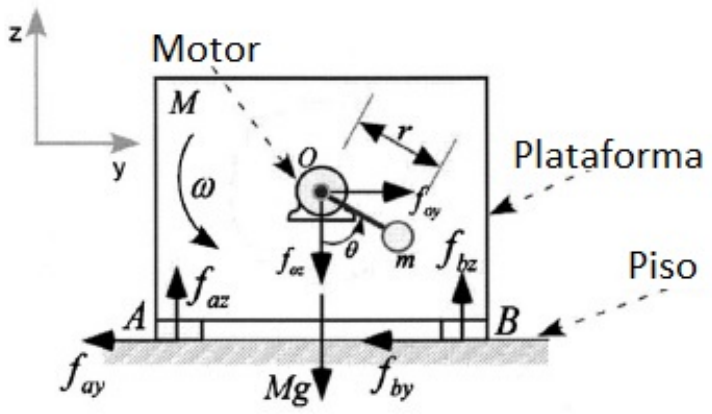

Figura 2.4: Esquema del modelo simplificado de plataforma vibracional.

$$\begin{aligned} f_{oy} &= mr\omega^2 seno(\theta) \\ f_{oz} &= -mg - mr\omega^2 cos(\theta) \end{aligned} \tag{2.1}$$

Donde, g es la aceleración de gravedad y r el largo de la unión del brazo entre m y O. Estas fuerzas también son aplicadas a la platafoma en el punto O, mientras que el momento producido por el movimiento de la pequeña masa excéntrica es despreciado. Cuando la velocidad angular w es baja, la plataforma no se mueve debido al efecto de la fuerza f_{oy} se cancela por las fuerzas de fricción en los puntos de contacto de la plataforma A y B. De cualquier modo, si la velocidad angular ω escede un valor crítico, entonces f_{oy} supera las fuerzas de fricción

[16] La optoelectrónica es el estudio de los componentes electrónicos que responden a la radición de la luz, o que emiten radiación, respondiendo a una frecuencia específica de radiación.

de Coulomb aplicadas en los dos puntos de apoyo, por lo que comienza el deslizamiento de la plataforma. Usando un modelo de fricción simplificado [136], el movimiento de la plataforma a lo largo de los ejes y y z se encuentra descrito por las siguientes ecuaciones:

$$\begin{aligned} M\ddot{y} &= f_{oy} - f_{fr} \\ 0 &= f_{az} + f_{bz} + (-Mg + f_{oz}) \end{aligned} \tag{2.2}$$

Donde todas las fuerzas se encuentran definidas en la Figura 2.4, M es la masa de la plataforma y f_{fr} es la fuerza de fricción. Dejando de lado los efectos de la fricción viscosa, las fuerzas de fricción están dadas por la siguiente ecuación:

$$f_{fr} = \begin{cases} f_C sgn(\dot{y}), & \dot{y} \neq 0 \\ f_{oy}, & \| f_{oy} \| < f_C, \quad \dot{y} = 0, \quad \ddot{y} = 0 \\ f_C sgn f_{oy}, & \| f_{oy} \| > f_C, \quad \dot{y} = 0, \quad \ddot{y} \neq 0 \end{cases} \tag{2.3}$$

Donde, f_C es el nivel de fricción de Coulomb, i.e., la máxima fuerza de fricción que puede existir para la fuerza normal actual, y está dada por,

$$f_C = \mu(f_{az} + f_{bz}) = \mu(Mg - f_{oz}) \tag{2.4}$$

Y la fuerza de fricción viscosa es despreciada. El parámetro μ es el coeficiente de fricción cinética y la función $sign(\dot{y})$ está definida por

$$sgn(\dot{y}) = \begin{cases} +1, & \dot{y} > 0 \\ 0, & \dot{y} = 0 \\ -1, & \dot{y} < 0 \end{cases} \tag{2.5}$$

La física del principio de movimiento se explica en mayor detalle a continuación.

Cuando la masa excéntrica se encuentra en el punto inferior de su trayectoria, las fuerzas normales (y por tanto las fuerzas de roce) son altas, mientras que cuando la masa se encuentra en el punto superior de su trayectoria, las fuerzas de fricción son bajas. Considerando la rotación de la masa m en el sentido de las manecillas del reloj, que se inicia en $\theta = 0^0$, la plataforma tiende a mantenerse en el lugar cuando m se encuentra abajo y se mueve a la derecha cuando la masa se encuentra arriba.

Cuando m pasa el punto más alto $\theta = 180^0$, la plataforma ya tiene una velocidad positiva. Una vez que m ha pasado ese punto, las fuerzas de fricción junto con las fuerzas de actuación tienden a desacelerar la plataforma e incluso a cambiar su dirección. Como la fricción continúa descendiendo, eventualmente lleva a la plataforma a su detención. Luego, las fuerzas de actuación se encuentran apuntando a la izquierda y como resultado, se induce el movimiento en reversa de la plataforma. Dado que la velocidad de la plataforma se vuelve cero pasados los 180^0, hay menos tiempo para que pueda acelerar en la dirección opuesta y finalmente, vuelve a su posición inicial luego de haberse detenido. Por lo tanto, una vez que se ha completado un ciclo, la plataforma exhibe un desplazamiento neto de aproximadamente un milímetro. Es fácil ver que si la velocidad de rotación de la masa excéntrica (m) aumenta, entonces la velocidad de la plataforma se vuelve cero incluso en un punto más allá de los 180^0, aumentando el desplazamiento neto de la plataforma por ciclo. Si la dirección de rotación ω se invierte,

el sentido de movimiento de la plataforma también se invierte. Se debe hacer notar que la resolución del movimiento de este principio de actuación no se encuentra limitado por la fricción de Stribeck, de hecho si se proporciona una velocidad de rotación (ω) adecuada a la masa excéntrica (m), entonces el desplazamiento hacia adelante puede ser igual en magnitud al desplazamiento en reversa y consecuentemente obtener un desplazamiento neto de cero [15].

Movimiento de una plataforma de dos grados de libertad

El principio de actuación descrito anteriormente se utiliza para el diseño de un micro-robot de dos grados de libertad (GDL). El diseño de este robot considera una serie de objetivos que son enumerados en [149]. El primer diseño consideraba 3 actuadores permitiendo el movimiento plano con 3 GDL. Luego, a través de simulaciones e investigación analítica se demostró que cuando operaban los tres actuadores, componentes particulares de las fuerzas generadas por estos se cancelaban entre ellas, reduciendo la eficiencia del robot. Más aún, el movimiento de los tres actuadores generaba el acoplamiento de fuerzas, por lo que se requería un sistema de control más sofisticado. Por lo anterior se decidió que se ocuparán dos actuadores en vez de tres, permitiendo el movimiento plano con dos grados de libertad.

2.4. Plataformas robóticas propulsadas por motores vibracionales

Para el diseño de plataformas robóticas, se considera una serie de factores como los mencionados en [15], en este documento se mencionan los siguientes:

- La plataforma debe ser capaz de moverse a lo largo de tres ejes.
- Debe asegurar la capacidad de posicionamiento del orden de un micrón.
- La plataforma debe ser capaz de navegar grandes distancias, i.e., distacias de a lo menos 10 veces el tamaño de la plataforma.
- Debe ser capaz de desarrollar velocidades de varios milímetros por segundo.
- Su tamaño debe ser tan pequeño como sea posible.
- El costo de construcción y energización de la plataforma debe ser tan bajo como sea posible.

Teniendo en cuenta estas consideraciones, Vartholomeos y Papadopoulos [15], desarrollaron una plataforma consistente en una base de geometría circular y tres actuadores soportados sobre tres pilares rígidos. En estudios posteriores demostraron que la eficiencia de el sistema que habían diseñado se veía mermada cuando los tres actuadores se encontraban en funcionamiento simultáneo. Lo anterior se debía a la interacción de distintos componentes de las fuerzas de locomoción generadas por los actuadores. En vista de los problemas que esta configuración desarrollaba en un segundo modelo, consideraron un sistema compuesto por sólo dos actuadores[149], que corresponde al óptimo funcionamiento para este tipo de sistemas.

Serán este tipo de sistemas los que se analizarán dentro de esta sección. A continuación se presenta un análisis dinámico del comportamiento de este tipo de plataformas.

Análisis Dinámico

La descripción de la dinámica de una plataforma que utiliza dos motores vibracionales y se encuentra apoyada sobre tres soportes rígidos, requiere del uso de tres modelos dinámicos: El primero es la dinámica de la plataforma, que se encuentra expresada a través de las correspondientes ecuaciones de moviemiento. En segundo lugar, un sistema de masa y resortes que modele las deformaciones de la base de la plataforma y en tercer lugar, un modelo dinámico que describa la vibración del micro-motor de corriente contínua.

Dinámica de la plataforma : Las únicas suposiciones para el análisis de la plataforma serán: (i) La masa desbalanceada puede ser considerada como una masa puntual m, que rota a una distancia r del eje del motor. (ii) Los puntos de contacto de la plataforma presentan fricción de Coulomb. No se consideran suposiciones para la velocidad relativa y fase de los actuadores o la distribución de inercia de la plataforma. El análisis de la plataforma involucra el marco de cuerpo {B, x, y} y el marco de inercia { O, x, y}, tal como se muestra en la Figura 2.5

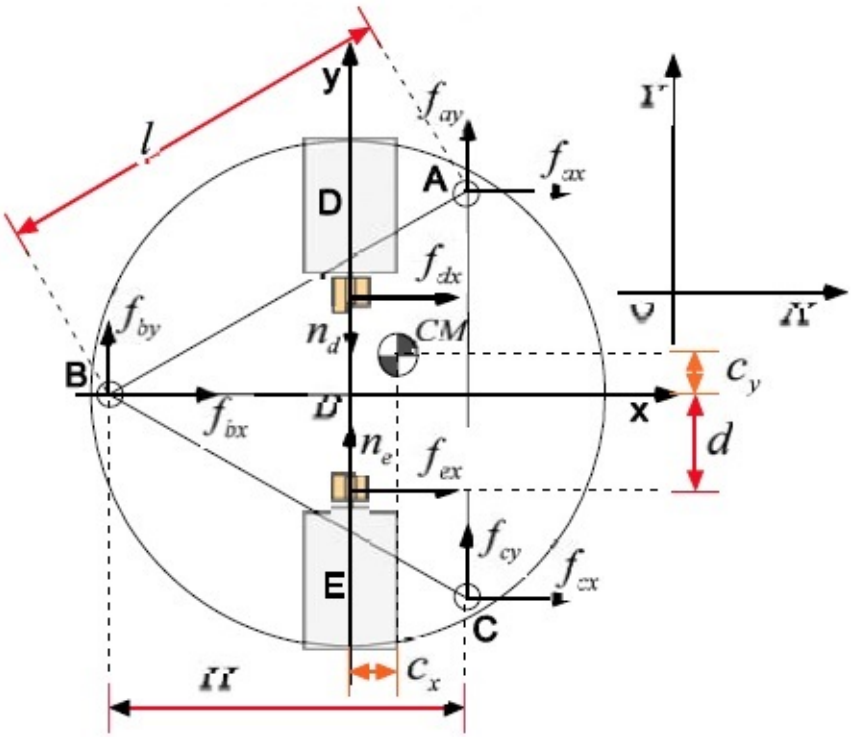

Figura 2.5: Fuerzas aplicadas a la plataforma. Fuente: Vartholomeos y Papadopoulos [149]

La notación adoptada para el análisis es ${}^{\mathrm{i}}f_j$ donde el superíndice i es el índice del marco

y el subíndice j es el índice del componente x, y, z. La b en el superíndice denota el marco B. El marco O no utiliza superíndice. Los vectores de posición de los puntos de contacto A, B y C se denotan por ${}^b r_a$, ${}^b r_b$, ${}^b r_c$ y los vectores de posición de los ejes de los motores, puntos D y E donde las fuerzas de desbalance son aplicadas, son denotados por ${}^b r_{\mathrm{d}}$, ${}^b r_{\mathrm{e}}$. Las fuerzas ${}^b f_a$, ${}^b f_b$, ${}^b f_c$ incluyen las fuerzas normal y de fricción en los puntos A, B y C de manera correspondiente. θ es el ángulo de la masa excéntrica con respecto al eje vertical (ver Figura 2.4). Debido a la rotación de las masas excéntricas, las fuerzas ${}^b f_{\mathrm{d}}$, ${}^b f_{\mathrm{e}}$, son aplicadas en los puntos D y E de la plataforma y los momentos ${}^b n_{\mathrm{d}}$, ${}^b n_{\mathrm{e}}$ son aplicados en los ejes de los motores, (ver Figura 2.5). Los componentes del diagrama de cuerpo rígido están dados por,

$$\begin{gathered} {}^b f_{\mathrm{i}x} = -m_{\mathrm{i}} r_{\mathrm{i}} \omega_{\mathrm{i}}^2 sen(\phi_{\mathrm{i}}) sen(\theta) \\ {}^b f_{\mathrm{i}y} = 0 \\ {}^b f_{\mathrm{i}z} = -m_{\mathrm{i}} g - m_{\mathrm{i}} r_{\mathrm{i}} \omega_{\mathrm{i}}^2 cos(\theta) \\ {}^b n_{\mathrm{i}x} = 0 \\ {}^b n_{\mathrm{i}y} = -m_{\mathrm{i}} g r_{\mathrm{i}} sen(\phi_{\mathrm{i}}) sen(\theta) \\ {}^b n_{\mathrm{i}z} = 0 \end{gathered} \tag{2.6}$$

Donde $\mathrm{i} = \{\mathrm{d}, \mathrm{e}\}$, $\omega_{\mathrm{i}} = \dot{\theta}_{\mathrm{i}}$ es la velocidad angular del motor i, r_{i} es la excentricidad de la masa desbalanceada m_{i} y $\phi_{\mathrm{i}} = \{90^0, -90^0\}$ son los ángulos de los vectores de posición ${}^b r_{\mathrm{d}}$, ${}^b r_{\mathrm{e}}$. Entonces las ecuaciones de Newton-Euler para la plataforma están dadas por [134],

$$M\dot{V} = R \textstyle\sum_{\mathrm{i}} {}^b f_{\mathrm{i}}, \quad \mathrm{i} = \{a, b, c, \mathrm{d}, \mathrm{e}\} \tag{2.7}$$

$$\begin{gathered} {}^b I \dot{\omega}_p + {}^b \omega_p \times {}^b I b_p^{\omega} = \textstyle\sum_{\mathrm{i}} ({}^b r_{\mathrm{i}} \times {}^b f_{\mathrm{i}}) + \sum_{\mathrm{i}} {}^b n_j \\ \mathrm{i} = \{a, b, c, \mathrm{d}, \mathrm{e}\}, \\ j = \{\mathrm{d}, \mathrm{e}\} \end{gathered} \tag{2.8}$$

Donde R es la matriz de rotación entre los marcos B y O, ω_p es la velocidad angular de la plataforma, ${}^b I$, es la matriz de inercia y $V = [\dot{x}, \dot{y}, \dot{z}]^T$ es la posición del centro de masa en el marco de inercia. Durante el análisis, las ecuaciones son simplificadas para el análisis en dos dimensiones.

Dinámica de cuerpo deformable Mientras que la plataforma es estática y las fuerzas de actuación crecen gradualmente, las fuerzas distribuidas en las patas de la plataforma alcanzan el nivel de Coulomb y el movimiento es inminente. Se requiere tener conocimiento de la distribución de fuerzas en cada uno de los tres soportes A, B y C de la plataforma triangular durante las condiciones estáticas. Estas seis variables desconocidas pueden ser determinadas considerando pequeñas deformaciones en la base de la plataforma. Para esto, la plataforma es modelada como un sistema agrupado, consistente en tres masas puntuales unidas por resortes con una rigidez alta como se muestra en la Figura 2.6 Resolviendo las ecuaciones de equilibrio estático de la plataforma para las reacciones verticales de contacto en los puntos A, B y C se encuentra que consecuentemente las masas agrupadas M_1, M_2, M_3 son calculadas:

$$\begin{gathered} M_1 = \frac{2Hl(M+m_{\mathrm{d}}+m_{\mathrm{e}}) - 3lMc_x - 6H(Mc_y - (m_{\mathrm{d}} - m_{\mathrm{e}})\mathrm{d})}{(6Hl)} \\ M_2 = \tfrac{1}{3}(m_{\mathrm{d}} + m_{\mathrm{e}} + M + \tfrac{3Mc_x}{H}) \\ M_3 = \frac{(2Hl(M+m_{\mathrm{d}}+m_{\mathrm{e}})) - 3lMc_x + 6H(Mc_y - (m_{\mathrm{d}} - m_{\mathrm{e}})\mathrm{d})}{(6Hl)} \end{gathered} \tag{2.9}$$

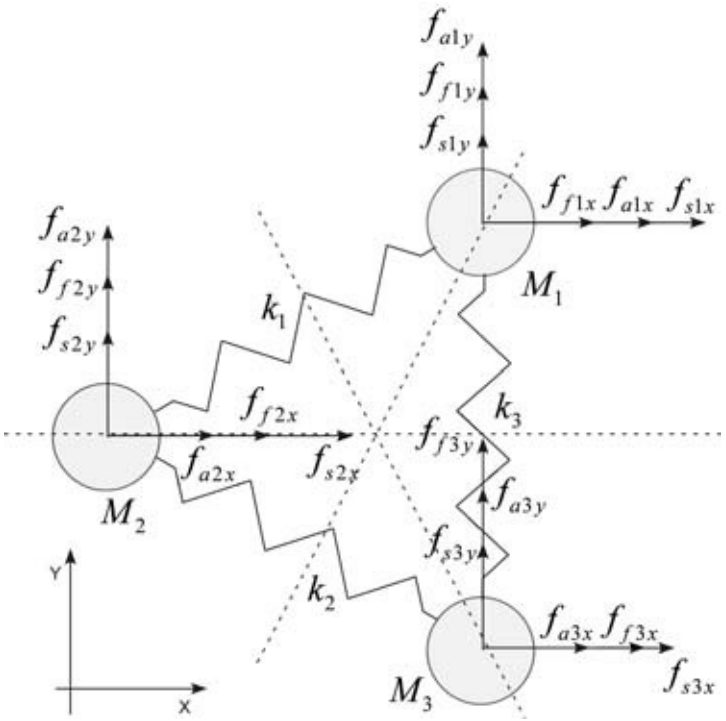

Figura 2.6: Sistema de masas y resortes para la base. Fuente: Vartholomeos y Papadopoulos [149]

Donde l, H, d son parámetros dimensionales mostrados en la Figura 2.5 y (C_x, C_y) Son las coordenadas del centro de masas con respecto al marco B. Las constantes k_1, k_2, k_3 representan las constantes elásticas de los resortes correspondientes y m_{d}, m_{e} son las cargas excéntricas de los motores D y E respectivamente. Las deformaciones producidas son adecuadamente pequeñas de manera que el cambio del ángulo entre los resortes resulta despreciable. Las fuerzas $f_{a\mathrm{i}}$, $f_{f\mathrm{i}}$, $f_{s\mathrm{i}}$ con $\mathrm{i} = 1, 2, 3$, son las fuerzas producidas sobre la masa M_{i} (actuación o motriz, fricción y fuerzas de los resortes). Las ecuaciones dinámicas del sistema masa-resorte son:

$$\begin{aligned} M\ddot{x} &= Ax + f_a + f_f, \\ x(0) &= 0, \\ \dot{x}(0) &= 0 \end{aligned} \tag{2.10}$$

Donde M es la matriz de masas, A es una matriz que contiene las constantes elásticas de los resortes y $x = [x_1, x_2, x_3, y_1, y_2, y_3]^T$ represnta los deplazamientos en el plano $x - y$ de las tres masas. Cuando las masas están estáticas, las incognitas del sistema son las 6 fuerzas de fricción, que son determinadas resolviendo las 6 ecuaciones de equilibrio estático. Cuando algunas o todas las masas se encuentran en movimiento, entonces la magnitud de las fuerzas de roce correspondientes son determinadas por el límite de fricción de Coulomb, mientras que la dirección de las fuerzas de fricción se determina por la velocidad de la masa correspondiente.

Dinámica del actuador : El actuador es modelado por un sistema que compromete un motor de corriente contínua de imanes permanentes y una carga desbalanceada. La variable de ingreso para el actuador es el voltaje entregado por la fuente V_s. La dinámica del actuador

se expresa mediante las siguientes ecuaciones,

$$\ddot{\theta} = -\frac{k_T^2}{RJ}\dot{\theta} - \frac{mgr}{J}seno(\theta) - (\frac{c}{J}sign(\dot{\theta}) + \frac{b}{J}\dot{\theta}) + \frac{k_T}{RJ}V_s \tag{2.11}$$

Donde, k_T es la constante de torque del motor, R es la resistencia Ohmnica, J es la inercia de la masa excéntrica y el termino $(\frac{c}{J}sign(\dot{\theta}) + \frac{b}{J}\dot{\theta})$ corresponde a la fricción viscosa y de Coulomb.

Actuación asíncrona : En el caso asíncrono $\omega_\mathrm{d} \neq \omega_\mathrm{e}$, pero las dos frecuencias deben estar lo suficientemente cerca entre ellas para que el movimiento pueda ser producido, mientras que al mismo tiempo que se evitan los saltos del robot [111]. Esto significa que la superposición de las fuerzas de los actuadores resulta en pulsos sinusoidales. La fuerza de actuación a lo largo de la dimensión x está dada por:

$$\begin{gathered} f_x(t) = A_\mathrm{d}seno(\omega_\mathrm{d}t) - A_\mathrm{e}seno(\omega_\mathrm{e}t) = |f_x(t)|seno(\measuredangle f_x(t)) \\ |f_x(t)| = \sqrt{A_\mathrm{d}^2 + A_\mathrm{e}^2 + 2A_\mathrm{d}A_\mathrm{e}cos(\omega_\mathrm{e} + \omega_\mathrm{d})t} \\ \measuredangle f_x(t) = tan^{-1}\left(\frac{A_\mathrm{d}seno(\omega_\mathrm{d}t) - A_\mathrm{e}seno(\omega_\mathrm{e}t)}{A_\mathrm{d}cos(\omega_\mathrm{d}t) + A_\mathrm{e}cos(\omega_\mathrm{e}t)}\right). \end{gathered} \tag{2.12}$$

Donde, $A_\mathrm{d} = m_\mathrm{d}r_\mathrm{d}\omega_\mathrm{d}^2$, $A_\mathrm{e} = m_\mathrm{e}r_\mathrm{e}\omega_\mathrm{e}^2$. De manera similar, la fuerza de actuación en la componente z está dada por 2.13 y el momento de actuación en el eje z está dado por la ecuación 2.14.

$$\begin{gathered} f_z(t) = -2mg - (A_\mathrm{d}cos(\omega_\mathrm{d}t) + A_\mathrm{e}cos(\omega_\mathrm{e}t)) = -2mg + |g_z(t)|cos(\measuredangle g_z(t) + \pi) \\ |g_z(t)| = \sqrt{A_\mathrm{d}^2 + A_\mathrm{e}^2 + 2A_\mathrm{d}A_\mathrm{e}cos((\omega_\mathrm{d} + \omega_\mathrm{e})t)} \\ \measuredangle g_z(t) = tan^{-1}\left(\frac{A_\mathrm{d}seno(\omega_\mathrm{d}t) - A_\mathrm{e}seno(\omega_\mathrm{e}t)}{A_\mathrm{d}cos(\omega_\mathrm{d}t) + A_\mathrm{e}cos(\omega_\mathrm{e}t)}\right) \end{gathered} \tag{2.13}$$

$$\begin{gathered} M_z(t) = -(s_\mathrm{d}A_\mathrm{d}seno(\omega_\mathrm{d}t) + s_\mathrm{e}A_\mathrm{e}seno(\omega_\mathrm{e}t)) = |M_z(t)|seno(\measuredangle M_z(t) + \pi) \\ |M_z(t)| = (s_\mathrm{d}A_\mathrm{d})^2 + (s_\mathrm{e}A_\mathrm{e})^2 + 2s_\mathrm{d}s_\mathrm{e}A_\mathrm{d}A_\mathrm{e}cos((\omega_\mathrm{d} + \omega_\mathrm{e})t + \pi)^{\frac{1}{2}} \\ \measuredangle M_z(t) = tan^{-1}\left(\frac{s_\mathrm{d}A_\mathrm{d}seno(\omega_\mathrm{d}t) + s_\mathrm{e}A_\mathrm{e}seno(\omega_\mathrm{e}t)}{s_\mathrm{d}A_\mathrm{d}cos(\omega_\mathrm{d}t) - s_\mathrm{e}A_\mathrm{e}cos(\omega_\mathrm{e}t)}\right) \end{gathered} \tag{2.14}$$

Donde, s_d, s_e son las distancias de las cargas m_d, m_e del centro de masa. La magnitud de la actuación se encuentra modulada por la acción de una función envolvente periódica de período $P_b = \frac{2\pi}{\omega_\mathrm{d} + \omega_\mathrm{e}}$. La frecuencia de la señal de actuación resultante recae en alguna parte entre las frecuencias de los motores (ω_d, ω_e) y varía periódicamente con el período P_b. La Tabla 2.4 presenta las fases relativas entre las tres señales de actuación en función de lo calculado en las ecuaciones 2.11 a 2.14. Las fases de las señales portadoras están dadas aproximadamente.

La fuerza $f_z(t)$ tanto como las reacciones verticales en los puntos de contacto de la plataforma están casi en fase con $f_z(t)$. Esto significa, que las fuerzas de fricción también están en

Tabla 2.4: Fases relativas de las tres fuerzas actuantes

Señal	$f_x(t)$	$f_z(t)$	$M_z(t)$
Envolvente	$(\omega_\mathrm{d}+\omega_\mathrm{e})t$	$(\omega_\mathrm{d}+\omega_\mathrm{e})t$	$(\omega_\mathrm{d}+\omega_\mathrm{e})t+\pi$
Portadora	$\frac{(\omega_\mathrm{d}-\omega_\mathrm{e})t}{2}$	$\frac{(\omega_\mathrm{d}-\omega_\mathrm{e})t}{2}+\frac{\pi}{2}$	$\frac{(\omega_\mathrm{d}-\omega_\mathrm{e})t}{2}-\frac{\pi}{2}$

fase con $f_z(t)$. De acuerdo a lo descrito en las secciones anteriores, las fuerzas de actuación interactúan con las fuerzas de fricción resultando en un desplazamiento neto plano. La diferencia de fase entre $f_z(t)$ y $f_x(t)$ o $M_z(t)$ afecta la dirección de movimiento y la magnitud del paso de movimiento. Se induce un desplazamiento considerable por cada paso, en la dirección deseada cuando las fuerzas de actuación horizontales (o sus momentos), se encuentran fuera de fase en ángulos de 180^0 a 90^0 con respecto a las reacciones verticales. Esto significa (ver Tabla 2.4), que esta plataforma puede exhibir tanto desplazamiento lineal como rotacional. Si las señales de actuación tienen una diferencia de fase relativa ϕ, entonces este ángulo es incorporado en las ecuaciones 2.11 a 2.14 y el pulso sinusoidal se ve desplazado en un ángulo ϕ. Este salto de fase no afecta el principio de movimiento, lo cual significa que los actuadores pueden tener cualquier diferencia de fase relativa entre sus giros iniciales. El comportamiento de pulso de la señal desencadena desplazamientos que están compuestos por secuencias alternantes, de regiones estáticas y de movimiento que corresponden a los valles y puntas de la magnitud modulada de las fuerzas de actuación.

Sistemas de enjambres robóticos en la vida real

En los años recientes, los investigadores ya han utilizado los sistemas de enjambres robóticos en varias aplicaciones de la vida real incluyendo la mayoría de las tareas mencionadas anteriormente. William et al. propusieron un marco, llamado Psicomimética[17], para el control distribuido de enjambres [115]. Ellos se centraron en los comportamientos robóticos que son similares a los mostrados en sólidos, líquidos y gases. Las diferentes formaciones son adoptadas por distintas tareas, incluyendo monitoreo distribuido, evasión de obstáculos, vigilancia y propósitos generales. Correll [31] propuso un sistema de inspección utilizando inteligencia de enjambre para las aspas de las turbinas. El sistema está basado en un enjambre de robots miniatura autónomos que utilizan solamente sensores locales montados en su estructura. El Senseable City Lab del MIT desarrolló una flota de robots de bajo costo que son capaces de absorber aceites llamado *Seaswarm* [1] para eliminar la espuma del océano y remover aceite. Cada uno de los robots puede absorber hasta 20 veces su peso en aceite. El sistema proporciona una solución autónoma y de bajo costo para la protección del medio marino.

Roombots [103] es un sistema robótico modular auto-reconfigurable. Los robots modulares autónomos pueden alterar su forma para adaptarse a una tarea determinada y medio ambiente de trabajo. Por su parte, Formica [46] es una plataforma robótica escalable biológicamente inspirado. Este diseño mecánico permite la producción de líneas estándar de montaje de placas de circuito. El sistema tiene como ventaja el tratarse de robots pequeños, baratos y con larga vida, es compatible con los periféricos y se puede escalar a varios cientos de individuos. Los científicos creen que estos enjambres son soluciones adecuadas para tareas

[17] Physicomimetics

como el reconocimiento de Marte, recuperación luego de un terremoto, etc. La robótica de enjambre puede ser útil para la aplicación militar. Pettinaro et al. [96] propusieron un sistema de robots auto-reconfigurables para la búsqueda de alimento y tareas de búsqueda y rescate, el cual tiene capacidad de hacer frente al fracaso ocasional. Los expertos militares creen que los vehículos aerodinámicos biónicos inspirados en la tecnología de inteligencia de enjambre serán aplicables dentro de unos años. Se puede prever que las abejas robóticas o cucarachas con equipos de reconocimiento y bombas podrían aparecer en las guerras del futuro.

2.5. Conceptos básicos de la mecánica estadística de enjambre

El concepto de enjambre ha sido usado sistemáticamente como un sinónimo de movimiento coherente de unidades individuales. En varios modelos de enjambre, éste emerge a través de una especie de transición (desde el desorden al orden) en función del (de los) parámetro(s) relevante(s) dentro del modelo [120]. Para comprender la naturaleza de estos modelos, es recomendable comenzar con algunas definiciones derivadas de la mecánica estadística o física estadística (física de un conjunto de moléculas interactuantes).

Antes de comenzar la discusión de mecánica estadística, se comenzará con una lista de características del enjambre en general. Asumiendo que un sistema que presenta moviento colectivo está compuesto por unidades que:

1. Son más bien similares
2. Se mueven a una velocidad absoluta relativamente constante y son capaces de cambiar su dirección
3. Interactúan dentro de un rango específico de rotación, cambiando su dirección de movimiento, de forma que consiguen un alineamiento efectivo
4. Están sujetos a un cierto ruido de magnitud variable.

Princípios y conceptos

En un sentido general, la transición de fase es un proceso, durante el cual un sistema, consistente en un gran número de partículas interactuantes, cambia desde una fase a otra, como una función de uno o más parámetros externos [120]. Las fases con las que uno se encuentra más familiarizado, son las fases de sólido, líquido y gas encontradas en la materia regularmente. Un ejemplo de cambio de fase típico, consiste en el congelamiento de un fluído (agua) cuando la temperatura desciende. Para el caso anterior, la temperatura es el parámetro externo o de control. Las transiciones de fase son definidas por el cambio de una o más variables del sistema, llamadas ***parámetros de orden***. Este nombre, parámetro de orden, proviene de la observación que las transiciones de fase usualmente involucran un cambio abrupto en las propiedades simétricas de sistema. Por ejemplo, en el estado sólido de la materia, los átomos tienen una posición promedio bien definida en los sitios de una red ordenada, pero en el caso

de las fases líquido y gas son desordenadas y aleatorias. De acuerdo con lo anterior, el parámetro de orden se refiere al grado de simetría que caracteriza una fase. Matemáticamente, este valor es usualmente cero en una fase (en las fases desordenadas) y son distintos de cero en en las otras (en las fases ordenadas). En el caso del movimiento colectivo, el parámetro de orden naturalmente elegido (pero no necesariamente) es la velocidad promedio normalizada φ, cuya definición se presenta en la ecuación 2.15.

$$\varphi = \frac{1}{N v_0} \sum_{\mathrm{i}=1}^{N} \vec{v_{\mathrm{i}}} \tag{2.15}$$

Donde N es el número total de unidades y v_0 es la velocidad absoluta promedio de las unidades del sistema.

Si el movimiento es desordenado, las velocidades de las unidades individuales, apuntan en direcciones aleatorias y la velocidad promedio será un vector de magnitud pequeña. Para movimientos ordenados, donde los vectores de velocidad apuntan en una dirección particular, la magnitud del vector de velocidad promedio será aproximadamente $N v_0$, Por lo que el parámetro de orden elegido de acuerdo a la definición anterior, para N grande, oscilará dentro del rango $(0 \to 1)$.

Si el parámetro de orden cambia de forma discontinua durante la transición de fase, hablamos de una ***transición de fase de primer orden***. Por ejemplo, el volumen de agua cambia de esta forma (discontinuamente) cuando se transforma en hielo. En contraste, durante la ***transición de fase de segundo orden*** (o contínua) el parámetro de orden cambia continuamente, mientras que su derivada, con respecto al parámetro de control, es discontinua. Las transiciones de fase de segundo orden son siempre acompañadas por grandes fluctuaciones de algunas de las cantidades relevantes.

Durante la transición de fase ocurre quiebre espontáneo de la simetría, es decir, la simetría del sistema cambia tan pronto como se supera un valor crítico del parámetro de control (presión, temperatura, etc.). En el caso de uno de los modelos más simples de transición de fases continua (el modelo de Ising), unas flechas que sólo pueden apuntar hacia arriba o abajo, son puestas en una red y son capaces de interactuar con sus vecinas más próximas. Para altas temperaturas, las flechas apuntan en direcciones aleatorias (con probabilidad $\frac{1}{2}$ de apuntar hacia arriba o hacia abajo), mientras que para bajas temperaturas (por debajo del punto crítico) la mayoría de ellas apunta en la misma dirección (que puede ser hacia arriba o hacia abajo), seleccionada de forma espontánea. Este es un ejemplo para el quiebre de la simetría (existente a altas temperaturas) durante la transición, dado que una de las direcciones se vuelve dominante. Si existieran infinitas direcciones posibles para las flechas (simetría continua) aún así existiría una dirección preferente que se manifestaría espontáneamente.

Las transiciones de fase, pueden ocurrir tanto en sistemas en equilibrio como en no equilibrio. En el contexto del movimiento colectivo (a pesar de tratarse de un fenómeno claramente fuera del equilibrio), existen razones para la preferencia de analogías con las transiciones de fase presentadas por sistemas en equilibrio más que aquellas encontradas comúnmente en el marco del estudio del comportamiento de fenómenos fuera del equilibrio [120]. Una razón importante es que las investigaciones dedicadas a los sistemas fuera del equilibrio han derivado en sub-disciplinas con sus formalismos y lenguaje propio, incluyendo los procesos en los

cuales se basa la investigación.

En particular, la mayoría de las investigaciones dedicadas a los sistemas fuera del equilibrio están concentradas en temas como:

1. Estados de absorción de los sistemas reactivo - difusivo.
2. Mapeo para el comportamiento universal de los modelos de crecimiento de la interfaz.
3. Escalamiento dinámico para el comportamiento de la relajación lejos del equilibrio y el envejecimiento.
4. Ampliación de los sistemas fuera del equilibrio mediante la percolación dirigida como el paradigma principal de absorción de la transición de fase.

Otro aspecto a ser considerado en los fenómenos observados durante el movimiento colectivo es que en contraste con lo que se asume de manera estándar en la mecánica estadística, el número de unidades involucradas en el comportamiento colectivo típicamente varía desde unas pocas docenas hasta unos pocos cientos (rara vez decenas de miles) [120]. Por otro lado, se tiene que las transiciones de fase en el marco de la mecánica estadística son realmente significativas solamente para sistemas de gran tamaño (cuyas unidades tienden a infinito). Las cantidades como los exponentes críticos no pueden ser interpretadas de forma correcta para el enjambre incluso en cantidades moderadas de individuos. De cualquier forma, una transición simple desde un estado desordenado a un estado ordenado puede ocurrir incluso en casos cuando el número de unidades consiste en un par de docenas. La mayoría de las observaciones en la vida real y los experimentos implican esta llamada "*escala mesoscópica*". Los estados, rotación por ejemplo y las transiciones (de orden a desorden, por ejemplo) pueden ser asociados con el fenómeno que tiene lugar en sistemas mesoscópicos.

Definiciones y expresiones

Así como para las definiciones utilizadas en la mecánica estadística, los fenómenos asociados con la transición de fase contínua son normalmente referidos como ***fenómenos críticos*** debido a su relación con un ***punto crítico*** en el cual la transición de fase tiene lugar ("Crítico", porque ahí es cuando el sistema es extremadamente sensible a pequeños cambios o perturbaciones). Cerca del punto crítico, el comportamiento de las cantidades que describen el sistema (presión, densidad, capacidad calorífica, etc.) son caracterizadas por los así llamados ***exponentes críticos***. Por ejemplo, la compresibilidad isotermal k_T de una sustancia líquida, cerca de su punto crítico, puede ser expresada por

$$k_T \sim |T - T_C|^{-\gamma} \tag{2.16}$$

Donde T es la temperatura, T_C es la temperatura crítica (a la cual ocurre la transición de fase), $\sim$ denota proporcionalidad y γ es el exponente crítico. En sistemas de partículas auto-propulsadas, el ruido (η) juega el rol de la temperatura (T): Un parámetro externo que destruye el orden. La fluctuación del parámetro de orden, $\sigma^2 = \langle \phi^2 \rangle - \langle \phi \rangle^2$, es descrito por

$$\sigma \sim |1 - \eta/\eta_C|^{-\gamma} \tag{2.17}$$

Donde η es el ruido, η_C es el ruido crítico que separa las fases ordenadas y desordenadas y γ es nuevamente la susceptibilidad del exponente crítico. Mediante la introducción de $\chi = \sigma^2 L^2$, donde L es el tamaño lineal del sistema, tenemos

$$\chi \sim (\eta - \eta_C)^{-\gamma} \tag{2.18}$$

k_T en la ecuación 2.16 corresponde a χ en la ecuación 2.18.

Otro ejemplo, es la expresión que describe el cambio de densidad entre las fases de líquido y gas, $\rho_l - \rho_g$

$$\rho_l - \rho_g \sim (T_C - T)^{\beta} \tag{2.19}$$

Donde β es el exponente crítico. Para sistemas de partículas auto-propulsadas, cuando $L \to \infty$, la ecuación correspondiente es

$$\phi \sim \begin{cases} (1 - \eta/\eta_C)^{\beta} & para \quad \eta < \eta_C \\ 0 & para \quad \eta > \eta_C \end{cases} \tag{2.20}$$

En cuanto a la relación entre el parámetro de orden ϕ y el campo de polarización externo h ("viento"), ϕ escala en función de h según la ley de potencia

$$\phi \sim h^{1/\delta} \tag{2.21}$$

Para $\eta > \eta_C$, donde δ es el exponente crítico relevante. Varias expresiones similares pueden ser formuladas involucrando otras cantidades y exponentes críticos. Es interesante notar que sistemas físicos diferentes que presentan distintos tipos de cambio de fase, siguen leyes similares. Por ejemplo, la magnetización de un elemento ferromagnético M sujeto a transición de fase cerca de una temperatura llamada "Punto de Curie", obedece a $M \sim (T_C - T)^{\beta}$.

Otras observaciones sorprendentes es que estos exponentes críticos están relacionados entre ellos, esto es, expresiones como $\alpha + 2\beta + \gamma = 2$ o $\delta = 1 + \frac{\gamma}{\beta}$ pueden ser formuladas, las cuales mantienen la independencia de los exponentes críticos $(\alpha, \beta, \gamma, \delta)$ de los sistemas físicos a los cuales pertenecen [120].

Otro fenómeno que acompaña las transiciones de fase son la formación de cúmulos de unidades ("*clusters*") que se comportan de manera similar. Las unidades que pueden ser alcanzadas a través de unidades vecinas, pertenecen al mismo cúmulo, donde las vecindades de cúmulos, se mantienen de acuerdo a un criterio de proximidad. Por lo tanto, el comportamiento de las unidades en un mismo cúmulo suele estar altamente correlacionado. En general, las funciones de correlación representan una herramienta muy útil para caracterizar el nivel de orden en un sistema.

Funciones de correlación

Hablando en general, dos series de datos (X e Y) se encuentran correlacionados si es que existe cierto tipo de relación entre sus elementos. Una ***Función de Correlación*** mide la similaridad entre las secuencias de datos, o en los casos contínuos, la similaridad entre dos señales o funciones. ***Auto-Correlación*** es la correlación de una señal consigo misma,

tipicamente como una función del tiempo. Esto es usado a menudo para revelar patrones repetitivos, como la presencia de una señal periódica cubierta por ruido. Si las dos señales comparadas son diferentes, se considera la ***Correlación-Cruzada***.

Por ejemplo, consideremos dos series de datos f_1 y f_2, que difieren solamente en un salto entre el número de elementos, por ejemplo, el elemento número cinco en la primera serie es el mismo que el elemento número doce en la segunda serie, el sexto se corresponde con el décimo tercero, etc. En este caso el salto es de siete elementos es s=7, esto es, la primera serie debe desplazarse siete lugares para ser congruente con la segunda. La correspondiente función de correlación cruzada mostrará un máximo en 7. Formalmente, para secuencias de datos discretas f_1 y f_2, la correlación está dada por:

$$c(s) = \sum_{n=-\infty}^{\infty} {f_1}^*[n] f_2[n+s] \tag{2.22}$$

Donde '*' se refiere a la operación de complejo conjugado. De forma correspondiente, en el caso continuo, donde f_1 y f_2 son funciones continuas (o senãles), la función de correlación cruzada revelará la magnitud en la que las funciones se encuentran desplazadas (a lo largo del eje horizontal) para ser congruentes con el caso anterior, la escritura formal es:

$$c(\tau) = \int_{\tau=-\infty}^{\infty} {f_1}^*(t) f_2(t+\tau) \mathrm{d}t \tag{2.23}$$

La ecuación 2.23 tiene un salto en la f_2 a lo largo del eje horizontal (en este ejemplo, corresponde al eje del tiempo), y calcula el producto de estas funciones desplazadas en el tiempo. Este valor es máximo cuando f_1 y f_2 son congruentes, porque cuando los montes (áreas positivas) se encuentran alineadas, ellos contribuyen a hacer la integral más grande, y de forma similar, cuando las depresiones (áreas negativas) se encuentran alineadas, ellas hacen un aporte positivo, dado que el producto de dos valores negativos es positivo. Con esta introducción podemos formular algunas funciones de correlación específica que son normalmente usadas en el campo del movimiento colectivo.

La función de correlación *velocidad-velocidad*, c_{vv}, es una función de auto-correlación que muestra qué tan cerca están las velocidades de una partícula (unidad, individuo, etc.) en el tiempo t está correlacionado con la velocidad de referencia. Que es definido como sigue:

$$c_{vv}(t) = \frac{1}{N} \sum_{\mathrm{i}=1}^{N} \frac{\langle \vec{v_\mathrm{i}}(t) \cdot \vec{v_\mathrm{i}}(0) \rangle}{\langle \vec{v_\mathrm{i}}(0) \cdot \vec{v_\mathrm{i}}(0) \rangle} \tag{2.24}$$

Donde, $\vec{v_\mathrm{i}}(0)$ es el punto de inicio (referencia) y N es el número de partículas dentro del sistema y $\langle ... \rangle$ denota el promedio sobre un grupo de tiempos de inicio. La forma en la que $c_{vv}(t)$ cae hasta cero muestra cómo las velocidades en tiempos posteriores se muestran independiente de los tiempos iniciales.

La ***función de correlación pareada***, $c_p(r)$, (o función de distribución radial, $g(r)$), describe cómo la densidad de unidades varía como una función de la distancia desde un elemento en particular. Para ser más preciso, si una partícula se encuentra en el origen y si $n = N/V$ es la densidad promedio (N es el número de unidades en un volumen V),

entonces la densidad local a una distancia r desde el origen es $ng(r)$. Esta magnitud puede ser interpretada como una medida del ordenamiento espacial local. La ecuación 2.25 entrega la fórmula exacta

$$c_p(r) = \frac{V}{4\pi r^2 N^2} \left\langle \sum_{\mathrm{i}} \sum_{j \neq \mathrm{i}} \delta(r - r_{\mathrm{i}j}) \right\rangle \tag{2.25}$$

La función de correlación direccional,

$$c_{\mathrm{i}j}(\tau) = \langle \vec{v_{\mathrm{i}}}(t) \cdot \vec{v_j}(t+\tau) \rangle \,, \tag{2.26}$$

indica en qué grado la velocidad de la partícula i en el tiempo t está correlacionada con la de la partícula j en el tiempo $t + \tau$. $\langle ... \rangle$ denota el promedio sobre el tiempo, y $\vec{v_{\mathrm{i}}}(t)$ es la velocidad normalizada de la partícula auto-propulsada i, (Notar que $c_{\mathrm{i}j}(\tau) = c_{\mathrm{i}j}(-\tau)$). El desfase de la correlación direccional es usada primeramente para determinar la relación entre el líder y los seguidores en las bandadas de pájaros, peces o más en general, en un conjunto de partículas auto-propulsadas (SPP's). El retraso en la correlación direccional para un par de SPP's (i y j, donde, $\mathrm{i}, j = 1, 2, ..., N$ y N es el número de SPP's dentro del conjunto) es calculado de acuerdo a 2.26. Entonces el valor máximo de la función de correlación $c_{\mathrm{i}j}(\tau)$ es, $\tau^*_{\mathrm{i}j}$. Valores negativos significan que la dirección de movimiento de la i-ésima SPP cae por debajo de la j-ésima, lo cual también puede ser interpretado como que la partícula j-ésima está liderando. La función de correlación direccional para una SPP con respecto al resto del grupo, es $c_{\mathrm{i}}(\tau) = \langle \vec{v_{\mathrm{i}}}(t) \cdot \vec{v_j}(t+\tau) \rangle_{\mathrm{i},j}$. Para terminar esta sección, se procede con una breve discusión respecto de dos fenómenos característicos que ocurren en sistemas de partículas auto-propulsadas pero que no tienen analogía directa en los procesos que ocurren en el equilibrio. El primero, es que dentro de una geometría confinada de cualquier grupo de unidades que se mantienen en movimiento de tamaño finito, tienen una posibilidad de ser atrapados o atascados. La presencia de paredes o incluso una "salida" muy estrecha, tenderán inevitablemente a producir este fenómeno[120].

Otro antecedente interesante de nombrar es una forma muy específica de cambio de densidades llamadas "fluctuaciones gigantes de número" (GNF, por sus siglas en inglés). Esta expresión denota la siguiente propiedad de un sistema de unidades auto-propulsadas: La fluctuación del número de unidades en un área en expansión crece linealmente con el número de unidades (N). Además, estas fluctuaciones se relajan de manera anómalamente lenta [120]. Este es un claro contraste con los resultados teóricos válidos para sistemas en equilibrio en los cuales las fluctuciones crecen como la raíz cuadrada de N. Esta interesante propiedad fue descrita por primera vez por Narayan et al. [89] para un sistema de barras alargadas que se hicieron vibrar y fue comentado posteriormente por Aranson et al. [18] mostrando que las colisiones inelásticas pueden producir GNF incluso para partículas esféricas. Como el sacudir las partículas introduce una velocidad media y las colisiones inelásticas o nemáticas implica una tendencia a que las partículas se alineen, se espera que todos estos hallazgos ocurran en una variedad de sistemas con movimiento colectivo.

2.6. Modelos básicos para partículas auto-propulsadas (SPP)

El modelamiento de los sistemas que presentan comportamiento colectivo ha sido inicialmente estudiado por áreas divergentes como los especialistas gráficos, biólogos y físicos. La que quizás fue la primera simulación mundialmente conocida fue publicada por Reynolds[129], quién se inspiró en la apariencia visual de una docena de objetos que volaban de manera coherente, entre ellos aves imaginarias o naves espaciales. Estos objetos con apariencia de ave, que el llamó "boids", se mueven a través de trayectorias determinadas por ecuaciones diferenciales tomando en cuenta tres tipos de interacciones: evitar las colisiones, mantener la orientación de los vecinos y finalmente, tratar de mantenerse cerca del centro de masas del grupo, tal como se muestra en la imagen 2.7. El modelo era determinístico y tenía un número de parámetros facilmente ajustables.

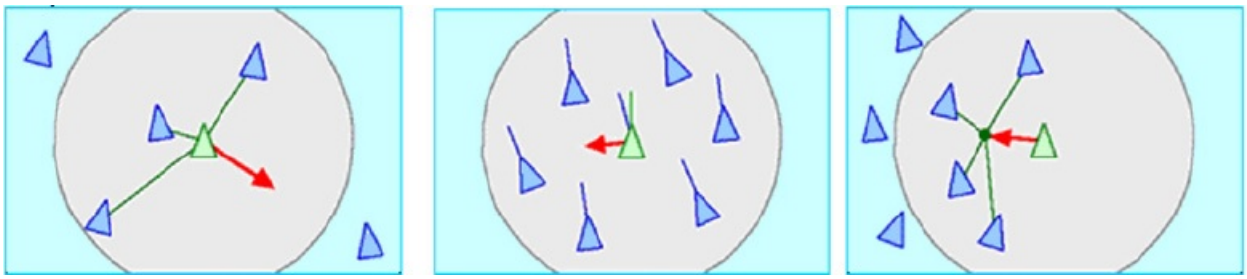

Figura 2.7: Bases para el movimiento en el modelo de Reynolds. Izquierda, rango de interacción. Centro, alineamiento. Derecha, cohesión. Fuente:*http://www.red3d.com/cwr/boids/*

El modelo de Vicsek estándar (SVM)

Con el objetivo de establecer una interpretación cuantitativa del comportamiento de grandes grupos en la presencia de perturbaciones, una aproximación proveniente de la física estadística fue introducida por Viscek et al. [128], el cual en la actualidad es normalmente referido como el "*Modelo de Viscek*". En lo que sigue se hará referencia a este modelo como "SVM", que corresponde al "*Modelo de Viscek Estándar*, por sus siglas en inglés[18]. En este modelo, las perturbaciones son consideradas como una consecuencia natural de una variedad de factores estocásticos y determinísticos que afectan el movimiento de los elementos que pertenecen al conjunto y se representan por la adición de un ángulo aleatorio de acuerdo a la dirección promedio. En esta aproximación para partículas auto-propulsadas (SPPs) las unidades se mueven con una velocidad absoluta ajustada v_0 y se asume una dirección promedio de los individuos que se encuentran dentro de una distancia R. Las ecuaciones de movimiento para la velocidad $\vec{v_{\mathrm{i}}}$ y la posición $\vec{x_{\mathrm{i}}}$ de la partícula i que tiene unos vecinos j vienen dadas por:

$$\vec{v_{\mathrm{i}}}(t+1) = v_0 \frac{\langle \vec{v_j}(t) \rangle_R}{|\langle \vec{v_j}(t) \rangle_R|} + perturbacion \tag{2.27}$$

$$\vec{x_{\mathrm{i}}}(t+1) = \vec{x_{\mathrm{i}}}(t) + \vec{v_{\mathrm{i}}}(t+1) \tag{2.28}$$

[18]Standard Vicsek Model

Donde $\langle ... \rangle_R$ indica el promedio (o suma) de las velocidades dentro de un círculo de radio R que rondan la partícula i. La expresión $\frac{\langle \vec{v_j}(t) \rangle_R}{|\langle \vec{v_j}(t) \rangle_R|}$ entrega un vector unitario que apunta en la dirección promedio de movimiento (caracterizada por su ángulo $\vartheta(t)$) dentro de este círculo.

Se debe hacer notar que los procesos explicados por esta regla de alineamiento pueden tener orígenes muy diferentes (adhesividad, hidrodinámica, pre-programación, procesamiento de la información, etc.). Las perturbaciones pueden ser tomadas en cuenta en varias formas, en la versión estándar, están representadas por la adición de un ángulo aleatorio al ángulo correspondiente a la dirección promedio de movimiento de los vecinos de la partícula i. El ángulo de la dirección de movimiento $\vartheta_{\mathrm{i}}(t+1)$ en el tiempo $t+1$, es obtenido mediante $\vartheta_{\mathrm{i}}(t) = arctan\left[\frac{\langle v_{j,x} \rangle_R}{\langle v_{j,y} \rangle_R}\right]$, como

$$\vartheta_{\mathrm{i}}(t+1) = \vartheta_{\mathrm{i}}(t) + \triangle_{\mathrm{i}}(t), \tag{2.29}$$

Donde $v_{j,x}$ y $v_{j,y}$ son las coordenadas x e y de la velocidad de la partícula $j-$ésima en la vecindad de la partícula i, y las parturbaciones están representadas por $\triangle_{\mathrm{i}}(t)$, es un número aleatorio obtenido de una distribución uniforme dentro del rango $[-\eta\pi, \eta\pi]$.

Esto es, la dirección final de la partícula i es obtenida después de la rotación de la dirección promedio de los vecinos en un ángulo aleatorio. Los únicos parámetros del modelo son: la densidad ρ (número de partículas en un volumen R^{d}, donde d es la dimensión), la velocidad v_0 y el nivel de perturbaciones $\eta < 1$. Como parámetro de orden φ, se acomoda la velocidad promedio normalizada, de acuerdo a lo definido en la ecuación 2.15.

$$\varphi \equiv \frac{1}{N v_0} \left| \sum_{\mathrm{i}=1}^{N} \vec{v_{\mathrm{i}}} \right| \tag{2.30}$$

Este modelo extremadamente simple permite la simulación de varios cientos de grupos de partículas y exhibe una transición de fase de segundo orden desde el estado de desorden al orden (partículas moviéndose en paralelo) a medida que el nivel de perturbación decrece. En el punto de transición, características tanto del estado ordenado como desordenado se encuentran presentes de forma simultánea para grupos de partículas de todos tamaños (y una función de correlación algebraica decreciente para la velocidad).

Como se verá en las secciones venideras, variando algunos parámetros, condiciones iniciales y ajustes, las simulaciones exhiben una gran variedad de patrones de movimiento colectivo, tales como "grupos de marcha", molinos, cadenas rotatorias, bandadas, etc. En algunos casos (vale decir, aplicando ciertos parámetros y ajustes de condiciones iniciales) estas fases ordenadas pueden mostrar algunas características, tales como fluctuaciones de número gigantes (GNF) o formación de bandadas. De acuerdo a los estudios, el ruido, la densidad, el tipo de interacción (atractiva - repulsiva, polar - apolar), el rango de interacción y las condiciones de borde (en caso de modelos de tamaño finito) han probado ser preponderantes en la formación de ciertos patrones.

Transición de fase para SVM

En el artículo que introduce la variante original del modelo de Vicsek (SVM)[121], se ha observado una transición de segundo orden desde el movimiento ordenado al desordenado. En particular, en el límite termodinámico, el modelo ha presentado transiciones de fase cinéticas análogas a las de los sistemas continuos del equilibrio, esto es,

$$\begin{aligned} \varphi &\sim [\eta_c(\rho) - \eta]^{\beta} \\ \varphi &\sim [\rho - \rho_c(\eta)]^{\delta} \end{aligned} \tag{2.31}$$

El cual define el comportamiento del parámetro de orden en el punto crítico, en el caso de la transición estándar de segundo orden. β y δ son exponentes críticos, η es el ruido (en forma de perturbaciones aleatorias), ρ es la densidad de partículas y $\eta_c(\rho)$ y $\rho_c(\eta)$ son el ruido crítico y la densidad crítica, respectivamente, para $L \to \infty$, donde L es el tamaño lineal del sistema.

De cualquier modo, la naturaleza continua de la transición ha sido cuestionada por Grégoire y Chaté [58], resultando en un número de estudios significativos de este aspecto fundamental del movimiento colectivo. Chaté et al. [29], en sus estudios sucesivos, presentaron resultados numéricos indicando que existe un "cruzamiento" en el tamaño del sistema, el cual llamó L^*, más allá del cual el carácter discontinuo de la transición parece ser independiente de la magnitud de la velocidad. Ellos demostraron que este carácter discontinuo es el "verdadero" comportamiento asintótico en el comportamiento del límite de tamaño infinito. Lo importante, es que Chaté et al. [29] mostró y presentó resultados en favor de su postura que L^* diverge en varios límites: tanto en el límite inferior de densidad, como en el límite superior, así como en el límite inferior de velocidad [120]. En particular, una extrapolación de sus estimaciones a través de los regímenes para pequeñas velocidades considerados en trabajos previos entregan valores de L^* tan grandes que hacen que las simulaciones no sean factibles.

Los estudios que buscan revelar la naturaleza de las transiciones de fase (tanto de primer como de segundo orden) han encontrado que el ruido (o más bien, la forma en la que es introducido al sistema) y la velocidad con la que las partículas se mueven, juegan un rol clave. Mientras las simulaciones muestran que para velocidades relativamente grandes ($v_0 > 0{,}5$) la transición es discon'inua, Baglietto y Albano [22] demostraron que para velocidades pequeñas, incluso en el límite en que las velocidades tienden a cero (excepto cuando son exactamente cero), la transición de orden es continua (es independiente del valor actual de la velocidad).

Escalabilidad de tamaño finito para SVM

El estudio más completo acerca del comportamiento de la escalabilidad de sistemas de partículas auto-propulsadas que exhiben alineamiento simple agregando perturbaciones, ha sido llevado a cabo por Baglietto y Albano [21]. Ellos llevaron a cabo extensas simulaciones de SVM y las analizaron por un sistema de escalamiento finito (método usado para determinar los valores de los exponentes críticos y el punto crítico mediante la observación de cómo las cantidades medidas varían para distintos tamaños de red) y mediante un acercamiento por escalamiento dinámico. Ellos observaron que la transición era contínua. Además demostraron la existencia de un conjunto completo de exponentes críticos para el caso bidimensional

(incluyendo aquellos correspondientes al escalamiento de tamaño finito) del mismo modo que fueron capaces de determinar sus valores numéricamente. En particular, dentro del marco de escalamiento de tamaño finito teórico, el escalamiento para el parámetro de orden φ del SVM ha sido reescrito como:

$$\varphi(\eta, L) = L^{-\beta/\upsilon}\tilde{\varphi}(\eta - \eta_c)L^{1/\upsilon} \tag{2.32}$$

Donde L es el tamaño finito del sistema, $\tilde{\varphi}$ es una función de escalabilidad ad-hoc, β y υ son dos de los exponentes críticos en cuestion: β es uno perteneciente a los parámetros de orden y υ es el exponente crítico de largo de correlación. De forma similar, la fluctuación del parámetro de orden, $\chi = \sigma^2 L^2$, toma la forma:

$$\chi(\eta, L) = L^{\gamma/\upsilon}\tilde{\chi}((\eta - \eta_c)L^{1/\upsilon}), \tag{2.33}$$

donde, $\tilde{\chi}$ es una función de escalabilidad apropiada, γ es el exponente crítico de susceptibilidad y $\sigma \equiv \langle\varphi^2\rangle - \langle\varphi\rangle^2$ es la varianza del parámetro de orden. En el límite termodinámico, χ obedece $\chi \sim (\eta - \eta_c)^{-\gamma}$ (ver 2.18).

Las ecuaciones 2.31 y 2.32 son convenientes para determinar los exponentes críticos dentro del marco de la teoría de escalabilidad para tamaño finito. Como resultado crucial, los autores encontraron que los exponentes que calcularon satisfacían la así llamada *relación de hiperescalabilidad* 2.34, la cual, en general, es válida para fenómenos estándar (en equilibrio).

$$\mathrm{d}\upsilon - 2\beta = \gamma \tag{2.34}$$

donde, d denota la dimensión $\mathrm{d} = 2$.

Variantes del modelo SPP original

Muchas variantes para el modelo más simple de SPP han sido propuestos a lo largo de los años. Una de las directrices principales compromete aquellos estudios que investigan sistemas en los cuales las partículas (unidades) no siguen ningún tipo de regla de alineamiento explícita, sólo las colisiones existentes entre individuos en presencia de algún tipo de interacción potencial. Este tipo de sistemas y los modelos que asumen algún tipo de regla de alineamiento para las unidades, serán tratados a continuación.

Modelos sin reglas explícitas de alineamiento

Como se mencionó en la sección 2.6, en la mayoría de los modelos simples para SPP, se asume un término de alineamiento. De cualquier modo, de acuerdo a estudios recientes, el movimiento de partículas se puede tornar ordenado incluso si es que no se aplica una regla de alineamiento explícita, pero el alineamiento es introducido por medio de las colisiones en una forma indirecta por las reglas de interacción locales. El modelo más simple (minimalista) de movimiento ordenado que emerge en un sistema de partículas auto-propulsadas tiene la siguiente configuración: Las partículas tratan de mantener una cierta velocidad absoluta y la

única interacción entre ellas es una fuerza lineal repulsiva ($\vec{F}$) de corto alcance (es decir, ellas no "calculan" el promedio de la velocidad de los vecinos, y la única interacción es a través de una fuerza central de a pares). Las ecuaciones correspondientes son:

$$\frac{\mathrm{d}\vec{v}_\mathrm{i}}{\mathrm{d}t} = \vec{v}_\mathrm{i}\left(\frac{v_0}{|\vec{v}_\mathrm{i}|} - 1\right) + \vec{F}_\mathrm{i} + \vec{\xi}_\mathrm{i} \tag{2.35}$$

Donde $\vec{\xi}_\mathrm{i}$ es el ruido (perturbaciones aleatorias, típicamente ruido blanco)

$$\vec{F}_\mathrm{i} = \sum_{\mathrm{i}\neq j} \vec{F}_{\mathrm{i}j} + \vec{F}_\mathrm{i} \quad (Bor\mathrm{de}) \tag{2.36}$$

$$\vec{r}_{\mathrm{i}j} = \vec{x}_\mathrm{i} - \vec{x}_j \tag{2.37}$$

$$\vec{F}_{\mathrm{i}j} = \begin{cases} C\vec{r}_{\mathrm{i}j}\left(\frac{r_0}{|\vec{r}_{\mathrm{i}j}|} - 1\right), & si\ |\vec{r}_{\mathrm{i}j} \leq r_0| \\ 0, & \text{para los otros casos} \end{cases} \tag{2.38}$$

Las simulaciones sobre el modelo mínimo resultan en transiciones de orden desde el desorden al movimiento colectivo coherente [40].

Resultados análogos fueron recientemente obtenidos para otros modelos simples asumiendo solamente una forma específica de colisión inelástica entre las partículas[60]. En sus simulaciones numéricas, una serie de partículas isotrópicas auto-propulsadas se mueven y colisionan en una superficie bidimensional plana libre de roce. Imponiendo condiciones de borde reflectivas producen una serie de fenómenos colectivos: migración ordenada, formación de vórtices y movimiento aleatorio de apariencia caótica en sub-grupos. Cambiando la densidad de partículas y el borde físico del sistema (desde una forma circular a una forma elíptica, por ejemplo) nuevamente resultan distintos tipos de movimiento colectivo. Para ciertas densidades y tipos de borde el sistema exhibe comportamiento espacio-temporales no triviales de subgrupos compactos de unidades.

Lo que produce la aparición del movimiento colectivo coherente en este tipo de sistemas, es que cada una de estas colisiones inelásticas entre partícuas isotrópicas produce alineamiento, resultando en un aumento de la velocidad de correlación global (se puede demostrar que las colisiones no conservan el momentum, pero producen a lo menos un pequeño aumento cada vez). Estos experimentos numéricos son fundamentales en la clarificación de las preguntas que se hacen respecto a los requerimientos mínimos para que un sistema presente movimiento colectivo, basado solamente en interacciones físicas.

Strömbom[38] también consideró un modelo de SPP en el cual solamente una regla de interacción social fue tomada en cuenta: atracción. Por medio de las simulaciones encontró una serie de patrones, como *enjambres* (un grupo de partículas con poca y variada alineación), Molinos indireccionados (grupo en el cual las partículas se mueven en un patrón circular alrededor de un centro común) y grupos alineados movedizos (en los cuales las unidades se mueven de manera altamente alineada). De forma importante, estas estructuras fueron estables solamente en presencia de ruido.

La introducción de un ángulo ciego[19] tiene un efecto fundamental en los patrones emergentes: molinos indireccionales se vuelven direccionales y aparecen "cadenas rotantes". En estas

[19]Un ángulo ciego está en una región detrás de cada unidad, en la cual las otras partículas son "invisibles", donde se incorpora algún tipo de alineamiento dentro del sistema.

cadenas las sub-unidades se mueven en curvas cerradas teniendo cero, una o dos junturas. Estas formaciones son llamadas "rotantes" porque las cadenas con cero o dos intersecciones seguidas rotan alrededor de un eje que se mueve lentamente.

Modelos con regla de alineamiento explícita

Las unidades en un sistema que exhibe cualquier tipo de comportamiento colectivo interactúan entre sí. En la SVM original, esta interacción se produce en el llamado camino "métrico", es decir, cada unidad interactúa sólo con aquellas partículas que están más cerca que la distancia predefinida, llamada "rango de interacción".

Una alternativa a este enfoque es la representación "topológica", en la que cada partícula se comunica con sus n vecinas más cercanas (un típico valor para n es alrededor de 6 - 7). La diferencia importante aquí es que, dado que el enfoque métrico se puede prescribir, el número de las partículas que caen en el rango de interacción podría cambiar también. Hay un punto sutil que ocurre de nuevo aquí.

Según la comparación de los dos enfoques realizada por Ginelli y Chaté [57], la principal diferencia radica en que el modelo de distancia topológica presenta una transición de fase de segundo orden, mientras que en el modelo SVM la transición de fase es de primer orden. Se ha discutido este punto en la sección 2.6 donde se argumentó que la resolución de la controversia radica en la altamente específica característica de la SVM, es decir, en el modelo métrico para bajas velocidades la transición es continua (como en el modelo topológico), mientras que para grandes velocidades la transición es de primer orden[88].

Comparado con el SVM original, una característica adicional importante ha sido introducida por Grégoire et al. [59], quien añadió adhesión entre las partículas para evitar la "evaporación" de grupos aislados en las simulaciones con condiciones de frontera abierta. La adición de esta nueva característica cambió el orden de transición y el ordenamiento observado fue discontinuo y dependiente de las perturbaciones.

La forma más común de introducir la cohesión a un sistema sin recurrir a interacciones globales, es complementar las reglas de interacción que definen el comportamiento de las unidades con algún tipo de mecanismo de atracción-repulsión de pares. En este espíritu, Chaté et al. [28] añadiún nuevo término a la ecuación 2.27, que determina una fuerza de atracción-repulsión de pares entre las partículas.

Otra generalización ha sido considerada por Szabó et al. [119] al extender los factores que influyen en el orden. El modelo asume que la velocidad de las partículas depende tanto de la velocidad y la aceleración de las partículas vecinas. En el modelo original, la velocidad de las partćulas depende solamente de la velocidad de sus vecinas. Cambiando el valor de un parámetro de peso determinó la influencia relativa de los términos de velocidad y aceleración. Se demostró que por debajo de un valor crítico el sistema exhibe movimiento desordenado, mientras que por encima de dicho valor la dinámica se asemeja a la del modelo original de SPP.

Otro punto de vista interpreta las partículas del SVM como unidades polares, ya que

llevan un vector de velocidad. Chaté et al. [30] consideraron una versión bipolar de la SVM, en la que después de determinar el ángulo correspondiente a la velocidad media de la zona, las partículas pueden "decidir" si se mueven a lo largo de esta dirección o en una dirección opuesta a la misma. Este modelo surge al considerar que las partículas auto - propulsadas alargadas se mueven preferentemente a lo largo de su eje principal. Los autores encontraron una transición desorden-orden distintivamente diferente que implica fluctuaciones gigantes de densidad (GNF), frente a los casos considerados previamente.

Mediante el uso de un nuevo conjunto de herramientas de diagnóstico relacionados con la distribución espacial de las partículas, Huepe y Aldana [66] compararon tres modelos simples que reproducen cualitativamente el comportamiento emergente de varios tipos de animales. El objetivo fue dar a conocer las características y diferencias cualitativas no reportadas previamente (que eran poco claras) entre los distintos modelos en cuestión. Al comparar sólo los parámetros de orden estándar (que miden el grado de alineación), los autores encuentran las transiciones de fase orden-desorden, en función del ruido, muy similares en los modelos investigados. Ellos demostraron que la distribución de tamaños de clúster, que es normalmente exponencial para valores de ruido altos, se acerca a una distribución de ley exponencial en los niveles de ruido reducidos y que esta tendencia ocasionalmente se invierte cerca del valor crítico de ruido, lo que sugiere un comportamiento crítico no trivial.

Recientemente Peruani et al. [95] registraron varios tipos de patrones espaciales auto-organizados mediante el uso de un modelo simple: una red de dos dimensiones con "exclusión de volumen". Una red con, "exclusión de volumen" significa que un nodo podría ser ocupado por un máximo de una partícula y que la simetría de rotación se rompe, al contrario de lo que ocurre en el resto de los modelos que asumen dimensiones espaciales contínuas. Las unidades tenían tendencia a la alineación ferromagnética. Como la susceptibilidad de las partículas para alinearse con sus vecinos aumentó, el sistema pasó a través de algunas fases distintas (Figura 2.8): Primero, para la fuerza de alineación débil, las unidades auto-segregadas en el desorden, se agregan (Figura 2.8 a), y luego, mediante el fortalecimiento de la alineación, se convierten en localmente ordenadas formando regiones de alta densidad, que los autores llamanron "atascos" (figura 2.8 b). Al mejorar aún más la susceptibilidad de alineación, emergen agregados triangulares de alta densidad (llamados "planeadores") que migran en una dirección bien definida (figura 2.8 c). Por último, dentro de estas estructuras auto-organizadas y altamente ordenadas, se encuentran las regiones enlongadas de alta densidad, "bandas" (figura 2.8 d). El código de colores en la Figura 2.8 es para las cuatro orientaciones posibles: rojo es para "derecha", negro es para "izquierda", verde es para "abajo" y azul es para "arriba".

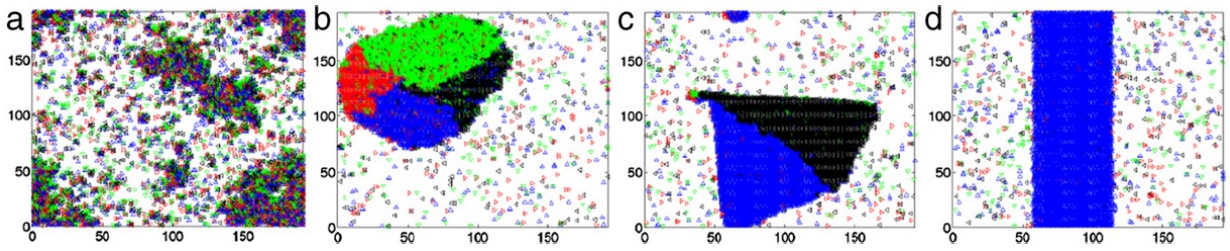

Figura 2.8: Fases de ordenamiento: (a) desorden orientacional, (b) atasco, (c) planeador y (d) bandada. Fuente: Peruani y Cia. (2011).

Resultados Exactos

Resultados exactos se refiere a aquellos obtenidos con una mínima o nula cantidad de aproximaciones de cualquier tipo concernientes al comportamiento de las unidades que se mueven (más allá de las reglas/ definiciones que obedecen). Originalmente, la mayoría de los resultados en esta área han sido obtenidos solamente para sistemas en los cuales el ruido (otra parte escencial de las manadas) fue completamente descuidado. Así, se puede considerar los sistemas relacionados como completamente determinísticos. De todas formas, se ha demostrado recientemente que los teoremas revisados con anterioridad son válidos en la mayoría de los casos con un nivel bajo de perturbaciones.

El modelo de Cucker - Smale (CS)

Una formulación exacta de la convergencia al consenso en una población de agentes autónomos fue llevado a cabo por Cucker y Smale [37] basados en su modelo (CS). Siguiendo su corriente de pensamiento, considérense pájaros, denotados por $\mathrm{i} = 1, ..., k$ moviéndose en un espacio tridimensional (Euclideano), $\Re^3$, procurando alcanzar una dirección común, la cual es en este caso el tema de "consenso". La posición del i-ésimo pájaro está dada por $x_\mathrm{i}(\in \Re^3)$ (Por supuesto, $x_\mathrm{i} = x_\mathrm{i}(t)$). Definiendo la matriz de proximidad $A = (a_{\mathrm{i}j})$, donde el elemento $a_{\mathrm{i}j}$ mide la habilidad de los pájaros i y j para comunicarse entre ellos, o como se puede decir, la "influencia" que ejercen entre ellos. Los elementos de A pueden tomar valores desde el intervalo (0..,1], donde la unidad indica la mayor cercanía (donde la influencia entre ellos es mayor). β es un "parámetro de ajuste" que afecta la magnitud de la influencia. Una expresión apropiada para los requerimientos anteriores, puede ser:

$$a_{\mathrm{i}j} = \frac{1}{(1 + \|x_\mathrm{i} - x_j\|^2)^\beta}, \tag{2.39}$$

Donde $\beta \geq 0$ (no debe ser confundido con el exponente crítico presentado en la sección 2.5). La ventaja principal de este modelo es que la dependencia de la distancia de interacción es una función suave permitiendo el tratamiento analítico. Es importante hacer notar que esta matriz de proximidad cambia con el tiempo, así como las posiciones de las aves.

Para un uso manifiesto de estos gráficos los autores introdujeron la *Matriz Laplaciana* de A, $L = D - A$, donde D es una matriz diagonal de kxk en la cual el elemento i-ésimo de la diagonal es definido como $\mathrm{d_i} = \sum_{j=1}^{k} a_{\mathrm{i}j}$. La matriz laplaciana (forma mediante la cual un gráfico puede ser representado en forma de matriz) es usualmente usado para encontrar propiedades en un gráfico. En particular, como se verá más adelante, los *autovalores* de L entregarán información importante.

Denotando la velocidad de un pájaro i en el tiempo t por $v_\mathrm{i}(t)(\in \Re^3)$,

$$v_\mathrm{i}(t + h) - v_\mathrm{i}(t) = h \sum_{j=1}^{k} a_{\mathrm{i}j}(v_j - v_\mathrm{i}). \tag{2.40}$$

Nuevamente, el valor $a_{\mathrm{i}j}$ mide la magnitud de la comunicación entre los pájaros i y j, así el lado derecho de la ecuación 2.40 representa un *promedio* local alrededor del pájaro i.

Las *ecuaciones de manada* son obtenidas haciendo que h tienda a cero:

$$\begin{aligned} x' &= \upsilon \\ \upsilon' &= -L\upsilon \end{aligned} \tag{2.41}$$

en $(\Re^3)^k$ x $(\Re^3)^k$, donde L entrega el promedio local. Notar que las matrices A y L están actuando en $(\Re^3)^k$ mapeando $(\upsilon_1, ..., \upsilon_k)$ a $(a_{\mathrm{i}1}\upsilon_1 + ... + a_{\mathrm{i}k}\upsilon_k)_{\mathrm{i}\leq k}$.

Luego de las consideraciones anteriores, se puede preguntar ¿bajo qué condiciones un sistema (descrito por las condiciones anteriores) presenta comportamiento de manada? o en otras palabras, ¿cuándo las soluciones de $\upsilon_{\mathrm{i}}(t)$ convergen a un $\upsilon^*(\in \Re^3)$?

Uno de los resultados más importantes de Cucker y Smale [37] es que la emergencia de comportamiento de manada depende de β. A saber si β es lo sufientemente pequeño ($\beta < \frac{1}{2}$) entonces la manada siempre emerge. Formalmente,

Teorema. *Para las ecuaciones de manada (Ecuación 2.41) existe una solución única para todo* $t \in \Re$. *Si* $\beta < \frac{1}{2}$ *entonces las velocidades* $\upsilon_{\mathrm{i}}(t)$ *tienden a un límite común* υ^* $(\in \Re^3)$ *a medida que* $t \to \infty$, *donde* υ^* *es independiente de* i, *y los vectores* $x_{\mathrm{i}} - x_j$ *tienden al vector límite* $\hat{x_{\mathrm{i}j}}$ *para todo* i, i, $j \leq k$, *a medida que* $t \to \infty$, *esto es, las posiciones relativas se mantienen encerradas.*

Si $\beta \geq \frac{1}{2}$ hay dispersión, la ruptura de la manada es posible. Pero se puede presentar comportamiento de manada, siempre y cuando algunas ciertas condiciones iniciales sean satisfechas.

Para obtener resultados más generales para las condiciones que determinan comportamiento de manada, se pueden investigar los valores propios de la correspondiente matriz laplaciana L. Denotando por G un gráfico y A la correspondiente matriz de atracción, definida de la manera usual, esto es

$$a_{\mathrm{i}j} = \begin{cases} 1 & \text{si i y j están conectados} \\ 0 & \text{si no lo están} \end{cases} \tag{2.42}$$

Siendo D una matriz diagonal con las mismas dimensiones de A, definidas por $\mathrm{d}(\mathrm{i}, \mathrm{i}) = \sum_j a(\mathrm{i}, j)$. Entonces la forma general de la matriz laplaciana de G, está dada por $L = L(G) = D - A$. Los valores propios de L pueden ser expresados por

$$0 = \lambda_1 \leq \lambda_2 \leq \lambda_3 \leq ... \tag{2.43}$$

λ_1 el primer valor propio es siempre cero. El segundo, λ_2, es conocido como *número de Fiedler*, F, el cual es cero si el gráfico G está separado (en este caso la manada se desintegra en dos o más manadas más pequeñas) y distinto de cero si y solo si G está conectada. Este número es una medida descriptiva crucial de las condiciones necesarias para satisfacer la emergencia de la manada.

De forma importante, en el caso del comportamiento de manada, $F = F(t)$ (es una función del tiempo), porque los elementos de G dependen de las posiciones individuales de los individuos x_{i}. Usando el *número de Fiedler*, se puede decir que uno obtiene movimiento de manada, si y solo si

$$0 < const < F = F(x(t)) \tag{2.44}$$

De otra manera la manada se dispersa. Las definiciones anteriores también pueden ser extendidas a matrices ponderadas generalizadas.

Además, Cucker y Dong [36] extendieron el modelo añadiéndole una fuerza repulsiva entre partículas. Ellos mostraron que, para este modelo modificado, la convergencia para el comportamiento de manada es establecido a lo largo las mismas líneas mientras, en adición, se evitan las colisiones, es decir, el respeto de una distancia mínima entre partículas está asegurada.

Las diferencias principales entre los sistemas descritos por Cucker y Smale y los descritos por SVM son, por una parte, la definición del rango de interacción y por otra parte, la existencia (o ausencia) de ruido. El SVM compromete el ruido, mientras que el modelo original de Cucker-Smale no lo hace. Respecto al rango de interacción, en el presente modelo (CS) se trata de un efecto de largo alcance que decae con la distancia de acuerdo a β (ver ecuación 2.39), mientras en el SVM tiene la misma intensidad para todas las unidades vecinas alrededor de una partícula dada, pero solo dentro de un rango bien definido (ver ecuación 2.27).

Modelos con segregación de unidades

La clasificación de células denota un tipo especial de movimiento colectivo durante el cual una mezcla heterogénea de células se segrega en dos (o más) grupos de células homogéneas sin ningún tipo de campo externo. Esto se puede observar, por ejemplo, durante el desarrollo de los órganos en un embrión o durante la regeneración después de la ruptura del tejido. Para simular este fenómeno, Belmonte et al. [24] consideraron dos tipos de células, que difieren en sus intensidades de interacción. De acuerdo al modelo, N partículas se mueven en un espacio de dos dimensiones con velocidad constante v_0. La velocidad y el ángulo de orientación de la partícula n en el tiempo t es denotado por $\vec{\overset{t}{n}}$ y θ_n^t, respectivamente. La nueva orientación θ_n^{t+1} de la partícula n es

$$\theta_n^{t+1} = arg\left[\sum_m \left(\alpha_{nm}\frac{\vec{v}_m^t}{v_0} + \beta_{nm} f_{nm}^t \vec{e}_{nm}^t\right) + \vec{u}_n^t\right] \tag{2.45}$$

Donde la sumatoria se refiere a las partículas (m) que están dentro de un radio r_0. Estas "células" producen una fuerza $f_{nm}^t \vec{e}_{nm}^t$ en n, a lo largo de la dirección $\vec{e}_{nm}^t$. El ruido es tomado en cuenta por $\vec{u}_n^t$, el cual es un vector unitario aleatorio con orientación uniformemente distribuida. α_{nm} y β_{nm} son los parámetros de control: α controla la ponderación relativa de la interacción de alineamiento y β muestra la magnitud de las fuerzas radiales entre dos cuerpos f_{nm}, la cual está definida como

$$f_{nm} = \begin{cases} \infty & \text{si } r_{nm} < r_c \\ 1 - \frac{r_{nm}}{r_e} & \text{si } r_c < r_{nm} < r_0 \\ 0 & \text{si } r_{nm} > r_0 \end{cases} \tag{2.46}$$

Esto es, para distancias más pequeñas que un radio del núcleo r_c, existe una importante fuerza repulsiva, alrededor del radio de equilibrio r_e se trata de una interacción de tipo armónica y para distancias más grandes que el rango de la interacción r_0 se establece en cero.

La segregación también ocurre en varios sistemas tridimensionales, tales como bandadas de pájaros o cardúmenes de peces. Mayormente, los modelos asumen partículas *idénticas* para simular el movimiento colectivo. Al mismo tiempo, esas simulaciones que suponen diversas partículas, exhiben comportamiento selectivo[34]. Esto significa que las diferencias en el comportamiento y/o motivación entre los animales afecta la estructura del grupo, ya que los individuos cambian sus posiciones relativas a los demás de acuerdo con su estado interior real. Esto implica que si las variaciones individuales son persistentes entonces el grupo se rearmará a su estado original después de las perturbaciones[33]. El fenómeno de clasificación depende principalmente en las diferencias relativas entre las unidades.

Modelos inspirados por patrones de comportamiento animal

El objetivo principal de los primeros modelos para sistemas específicos tratando de simular el movimiento de animales moviéndose en tres dimensiones (primeramente pájaros y peces) fueron para producir movimiento colectivo que parezca real [129]. En estos modelos se tomaron en cuenta muchos parámetros, por ejemplo, la separación con el vecino más cercano, la densidad, etc. Posteriormente, varios otros aspectos y características fueron también estudiados, tales como la función y el mecanismo de formación de cúmulos en bandadas de aves [23], el tamaño y parentesco de los peces para correlacionarlos con las características espaciales[63], el efecto de las conexiones sociales ("*Redes Sociales*"[20]) sobre el movimiento colectivo del grupo, el despegue cohorte de bandadas de pájaros o el efecto de la amenaza percibida [120].

En cuanto a la metodología de la mayoría de las simulaciones, el acercamiento basado en el individuo ha probado ser el más popular (aunque existen métodos alternativos [120]). La razón detrás de esto es que este acercamiento entrega un vínculo entre el comportamiento de los individuos y las propiedades emergentes del enjambre como un todo. Por lo tanto, resulta adecuado investigar cómo las propiedades del sistema dependen de la conducta real de los individuos. Las reglas más comunes aplicadas en estos modelos son:

1. Fuerzas repulsivas de corto alcance para evitar la colisión entre compañeros y con los bordes
2. Ajustar el vector de velocidad de acuerdo a la dirección de las unidades vecinas
3. Una fuerza que evite quedarse solo
4. Ruido
5. Algún tipo de fuerza de arrastre si es considerado el medio donde se mueven los individuos (que usualmente no es tomado en cuenta en el caso de pájaros y peces).

Entonces, los modelos concretos difieren en las reglas que se aplican y en los parámetros del sistema.

Couzin et al. [33] categorizaron el movimiento colectivo emergente como una función de los parámetros del sistema. En este marco, los individuos obedecen a las siguientes reglas básicas:

[20] *Social Network*

1. Ellos continuamente tienden a mantener cierta distancia entre ellos y sus compañeros.
2. Si no están realizando una maniobra de evación (descrita en la regla 1), entonces ellos son atraídos hacia sus compañeros.
3. Ellos alinean su dirección a sus vecinos.

Su zona de percepción (en la cual ellos interactúan con los otros) es dividida en tres zonas no superpuestas. El radio de estas esferas (zona de repulsión, zona de orientación y zona de atracción) son R_r, R_o y R_a, respectivamente. De este modo, el ancho de los dos anillos más externos son $\Delta R_o = R_o - R_r$ y $\Delta R_a = R_a - R_o$. α denota el campo de percepción, de este modo, el "*volumen ciego*" está detrás del individuo, con un ángulo interior $(360 - \alpha)$. Con el fin de explorar el comportamiento global del sistema, los autores analizaron las consecuencias de variar ciertos parámetros sistémicos, como el núnmero de individuos, velocidad preferente, tasa de giro, ancho de las zonas, etc. Para todos los casos, las siguientes propiedades globales fueron calculadas:

$$\varphi(t) = \frac{1}{N}\left|\sum_{\mathrm{i}=1}^{N} \vec{v}_{\mathrm{i}}^{u}(t)\right| \tag{2.47}$$

Donde N es el número de individuos dentro del grupo, $(\mathrm{i} = 1, 2, ..., N)$ y $\vec{v}_{\mathrm{i}}^{u}$ es el vector unitario del i-ésimo individuo en el tiempo t. El vector unitario $\vec{v}_{\mathrm{i}}^{u}$ es un vector equivalente con el parámetro de orden definido por la Ec. 2.15.

La otra medida, "*Momento angular del grupo*", es la suma de los momentos angulares de los miembros del grupo con respecto al centro del grupo, $\vec{r}_{Gr}$. Esta expresión mide el grado de rotación del grupo con respecto a su centro.

$$m_{Gr} = \frac{1}{N}\left|\sum_{\mathrm{i}=1}^{N} \vec{r}_{\mathrm{i}-Gr}(t) \times \vec{v}_{\mathrm{i}}^{u}(t)\right| \tag{2.48}$$

Donde $\vec{r}_{\mathrm{i}-Gr} = \vec{r}_{\mathrm{i}} - \vec{r}_{Gr}$, es la diferencia vectorial de la posición de un individuo i, $\vec{r}_{\mathrm{i}}$ y la posición del centro del grupo, $\vec{r}_{Gr}$.

$$r_{Gr}(t) = \frac{1}{N}\sum_{\mathrm{i}=1}^{N} \vec{r}_{\mathrm{i}}(t). \tag{2.49}$$

En la Figura 2.9 se resumen los cuatro "tipos básicos" de movimiento colectivo que emerge de acuerdo a varias configuraciones de los parámetros:

1. "*Enjambre*": Tanto el parámetro de orden (φ) como el momento angular (m_{Gr}) son pequeños, lo que significa poco o ningún grado de paralelismo en la orientación. (Figura 2.9(a)).
2. "*Toro*" o "*Molino*" Los individuos rotan alrededor de un centro vacío con una dirección aleatoriamente elegida. El parámetro de orden (ϕ) es pequeño, pero el momento angular (m_{Gr}) es grande. Esto ocurre cuando Δr_a es grande, pero Δr_o es pequeño (Figura 2.9(b)).
3. "*Grupo Dinámico en Paralelo*" ocurre para valores intermedios o altos de Δr_a y valores intermedios de Δr_o. Esta formación es mucho más móvil que cualquiera de las previamente descritas. El parámetro de orden (φ) es alto, pero el momentum angular $(||m_{Gr}||)$ es pequeño (Figura 2.9(d)).

4. "*Grupo Altamente Paralelo*" Al incrementar Δr_o emerge una formación altamente alineada caracterizada por un valor muy alto del parámetro de orden (φ) y bajo mementum angular (m_{Gr}) (Figura 2.9(d)).

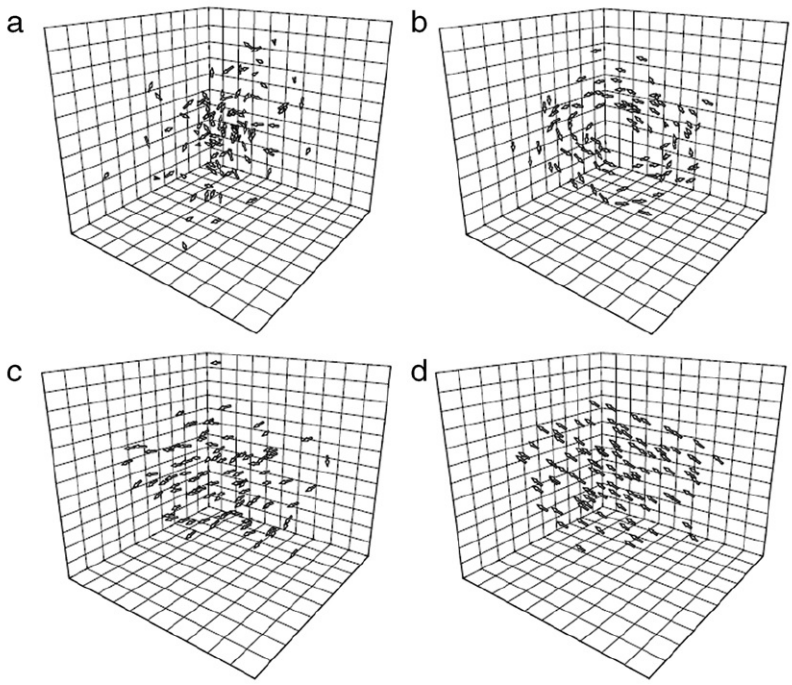

Figura 2.9: Tipos básicos de movimiento colectivo. (a) Enjambre, (b) Toro, (c) Grupo dinámico en paralelo y (d) Grupo altamente paralelo. Fuente: Couzin et al. [33]

Sin embargo, el enfoque basado en el modelado individual (o agente) tiene sus propias limitaciones o "trampas", Tal como lo mencionó Eriksson et al. [47]. Es decir, que las diferentes combinaciones de reglas y parámetros pueden producir el mismo (o muy similar) patrón y comportamiento colectivo. En consecuencia, con el fin de demostrar que el comportamiento emergente de un determinado sistema biológico obedece principios dados, no es suficiente para proporcionar una regla y un conjunto de parámetros indicando que ellos reproducen el comportamiento observado. Por otro lado, Couzin et al. [33] demostraron que la misma regla y grupo de parámetros pueden resultar en comportamientos colectivos distintos en sistemas muy distintos, dependiendo de su pasado reciente denominado "historia"). Yates et al. [155] notaron que muchos modelos tienen un débil poder predictivo concerniente a varios aspectos relevantes de movimiento colectivo, incluyendo, por ejemplo, la tasa a la cual los grupos repentinamente cambian de dirección.

Rol del líder para encontrar el consenso

Cuando los animales viajan juntos tienen que desarrollar un método para tomar desiciones de acuerdo a los lugares para alimentarse, descansar y anidar, rutas de migración, etc. Modificando ligeramente los modelos (típicamente extendiéndolos) un grupo de individuos puede desarrollar dichas actividades. En consecuencia, Couzin et al. [32] sugirieron un modelo simple en el cual los individuos no requerían saber cuántos y cuales individuos tenían

la información, no necesitaban tener un mecanismo de señalización y tampoco requerían reconocimiento individual entre los individuos del grupo. Los individuos informados no necesitaron saber nada sobre el nivel de información de sus compañeros ni como la calidad de la información que ellos poseían fue comparada con la de los demás. El modelo es el siguiente:

$$\vec{\mathrm{d}}_{\mathrm{i}}(t+\Delta t) = -\sum_{j\neq \mathrm{i}} \frac{\vec{r}_j(t) - \vec{r}_{\mathrm{i}}(t)}{|\vec{r}_j(t) - \vec{r}_{\mathrm{i}}(t)|} \tag{2.50}$$

Esta regla tiene la más alta prioridad. Si no hay compañeros dentro de su rango, el individuo intentará alinearse con los vecinos j, que están dentro del rango de interacción ρ. Si es así, la dirección deseada se define como:

$$\vec{\mathrm{d}}_{\mathrm{i}}(t+\Delta t) = -\sum_{j\neq \mathrm{i}} \frac{\vec{r}_j(t) - \vec{r}_{\mathrm{i}}(t)}{|\vec{r}_j(t) - \vec{r}_{\mathrm{i}}(t)|} + \sum_{j\neq \mathrm{i}} \frac{\vec{v}_j(t)}{|\vec{v}_j(t)|} \tag{2.51}$$

Se utilizará el correspondiente vector unitario, $\hat{\mathrm{d}}_{\mathrm{i}}(t) = \frac{\vec{\mathrm{d}}_{\mathrm{i}}(t)}{\left|\vec{\mathrm{d}}_{\mathrm{i}}(t)\right|}$.

Con el objetivo de estudiar la influencia de los individuos que portan información, una porción del grupo, p, se le ha dado información respecto a una dirección preferida, descrita por el vector unitario $\vec{g}$. El resto de los individuos puede ser considerado como "ingenuo" dado que no poseen una dirección preferente. Los individuos portadores de información balancean su alineamiento social y su dirección preferente con el factor de peso ω.

$$\vec{\mathrm{d}}_{\mathrm{i}}(t+\Delta t) = \frac{\hat{\mathrm{d}}_{\mathrm{i}}(t+\Delta t) + \omega\vec{g}_{\mathrm{i}}}{\left|\hat{\mathrm{d}}_{\mathrm{i}}(t+\Delta t) + \omega\vec{g}_{\mathrm{i}}\right|} \tag{2.52}$$

(ω puede exceder 1; en este caso los individuos son influenciados en mayor medida por sus propias preferencias, más que por la influencia de sus compañeros). *La precisión del grupo* (que describe la calidad de la información transferida) es caracterizada por la desviación angular normalizada de la dirección del grupo alrededor de la dirección preferida $\vec{g}$, Similar al término descrito en la ecuación 2.48.

Los autores encontraron que para un tamaño de grupo ajustado, la presición aumenta asintóticamente a medida que la porción p de individuos con información aumenta. Esto significa que mientras más grande el grupo, menor es la porción de miembros informados necesaria para poder guiar al grupo a través de la dirección preferente.

Sin embargo, los individuos con información también pueden diferir en cuanto a su dirección preferida. Si el número de individuos que prefiere una u otra dirección es igual, entonces la dirección del grupo dependerá del grado en el cual estas direcciones preferentes difiere la una de la otra: Si estas preferencias son similares, entonces el grupo avanzará en la dirección preferente promedio de todos los individuos que portan información. A medida que la diferencia entre las direcciones promedio aumenta, los individuos comienzan a seleccionar aleatoriamente una u otra dirección preferente. Si el número de individuos con una dirección preferente aumenta, el grupo entero irá en la dirección preferida por la mayoría, incluso si es que esa mayoría es pequeña.

Freeman y Biro [53] Extendieron este modelo mediante la inclusión de un "*Factor de Importancia Social*", h, que describe la "*Magnitud del efecto*" de un individuo dado en el

movimiento del grupo. Esto es, h varía con cada agente, y a medida que aumenta este valor, mayor es la influencia que una determinada unidad produce en el grupo. La ecuación 2.51 es modificada de acuerdo a:

$$\vec{\mathrm{d}}_{\mathrm{i}}(t+\Delta t) = \sum_{j \neq \mathrm{i}} h_j \frac{\vec{r}_j(t) - \vec{r}_{\mathrm{i}}(t)}{|\vec{r}_j(t) - \vec{r}_{\mathrm{i}}(t)|} + \sum_{j \neq \mathrm{i}} h_j \frac{\vec{v}_j(t)}{|\vec{v}_j(t)|} \tag{2.53}$$

Estos modelos muestran que el liderazgo puede emerger desde las diferencias en el nivel de información poseído por los miembros del grupo. De manera importante, como la información puede ser transiente y diferentes integrantes del grupo pueden tener información pertinente en diferentes tiempos o diferentes contextos, el liderazgo puede ser transiente y transferible.

Otros estudios también respaldan estos resultados. Quera et al. [97] usaron otro tipo de reglas que eran seguidas por sus agentes y observaron lo mismo: ciertos agentes se convertían en líderes sin que ninguno de los parámetros o condiciones iniciales, permitiera predecir ese hecho. Incluso los modelos más simples pueden producir decisiones de consenso. Por ejemplo, la regla del quórum severa (En la que la probabilidad de que un individuo siga una determinada opción, aumenta brúscamente cuando el número de miembros de otro grupo que toma una decisión, alcanza un umbral) resultaron en decisiones de grupo precisas [118].

Capítulo 3

Metodología

En el presente capítulo se describe la metodología utilizada para la construcción y posterior caracterización cinemática del comportamiento colectivo de dos plataformas robóticas. La primera de estas plataformas lleva por nombre *Cube* y corresponde a una plataforma que cumple doble función. Por un lado, fue utilizada para la exploración de los mecanismos de activación y locomoción en robots modulares y por otro, se utilizó para estudiar comportamiento colectivo, dado que se trata de una plataforma fácil de construir. La segunda plataforma lleva por nombre *CheaViBot*[1] y corresponde a una plataforma robótica mejorada, concebida para permitir la realización de experiencias en el campo de la robótica modular.

A continuación se procede a dar una breve descripción del capítulo.

En la primera sección, Procedimiento general se realiza la descripción de las experiencias realizadas con las plataformas robóticas.

En la segunda sección llamada Construcción de plataformas robóticas, se describen los procedimientos involucrados en la construcción de las plataformas robóticas estudiadas. En primer lugar se presenta el proceso de construcción de la plataforma robótica *Cube*. Dentro de esta descripción se incluyen las etapas necesarias para la confección de piezas utilizando impresoras tridimensionales. En particular, esta descripción se basa en la utilización de la impresora modelo *MakerBot 2*. La sección concluye con la descripción del procedimiento empleado para la confección de la segunda plataforma robótica utilizada durante las experiencias realizadas (*CheaViBot*).

La tercera sección lleva por nombre Obtención de trayectorias y tiene por objetivo detallar el procedimiento mediante el cual se obtuvieron las funciones itinerario de los robots. Este proceso contempla la utilización de la cámara *OptiTrack Trio*™, para la obtención de la posición de cada uno de los robots en el tiempo y el software *Matlab*™, para el análisis de dichos datos y la generación de los gráficos pertinentes.

La cuarta sección, Modelo Realista, se encarga de describir el desarrollo y experiencias realizadas con una simulación computacional realista. Esta simulación permitió estudiar el

[1] Acrónimo de su nombre en inglés: "Cheap Vibrating Robots" (Robots Vibracionales Económicos)

comportamiento colectivo presentado en ambas plataformas robóticas para poblaciones de varias decenas de individuos.

Al final del capítulo se hace referencia a etapas que no corresponden a los objetivos de la presente investigación pero que fueron necesarias para la correcta ejecución de las experiencias realizadas.

3.1. Procedimiento general

El trabajo realizado se divide en dos grandes tareas. La primera corresponde a la obtención de las características cinemáticas de las plataformas robóticas: *Cubes* y *CheaViBots*, tanto para su comportamiento individual como colectivo, mientras que la segunda corresponde al escalamiento de dichos resultados utilizando la simulación computacional desarrollada para este fin.

El proceso de caracterización cinemática de las plataformas robóticas comenzó con los robots *Cube*. En primer lugar se determinaron las propiedades cinemáticas individuales promedio presentadas por estos robots y posteriormente, se estudió el comportamiento colectivo (ver Figura 3.1).

La caracterización del comportamiento individual se realizḿediante dos experiencias. En la primera, se obtuvo la velocidad promedio con la cual se desplazan estos robots tanto frontal como lateralmente cuando se trabaja con una fuente de energía constante. Lo anterior se determinó al posicionar uno de los robots *Cube*, conectado a una fuente de alimentación constante, en una superficie acanalada que sólo permitía desplazamiento unidireccional. Posteriormente, se procedió a registrar mediante la cámara *OptiTrack Trio* ™, el tiempo que demoraba en recorrer desde un extremo al otro dicha superficie. Este proceso fue realizado posicionando el robot de costado y de frente, para determinar ambas velocidades.

En la segunda sub-etapa, se obtuvo la función itinerario de 10 robots, al registrar sus trayectorias mediante la cámara *OptiTrack Trio* ™, por un período de 30 [seg]. Para esto, cada uno de estos individuos fue depositado sobre una base de roce despreciable con total libertad de desplazamiento. Por cierto, la energización de los robots se realizó por medio de una batería de botón. Al final de cada experiencia, el robot utilizado fue apartado del grupo original para evitar realizar las mediciones dos veces sobre el mismo robot. A partir de esta información se determinó: trayectoria, desplazamiento medio, velocidad promedio y la existencia de orientaciones preferentes.

En una segunda fase, se caracterizó el comportamiento colectivo presentado por los robots *Cubes* cuando presentan distintos niveles de densidad poblacional y/o grados de interacción entre individuos. Esto último, se ejecutó variando la configuración de los robots, al dotarlos o no, de elementos anexos. Durante las primeras experiencias, se permitió que grupos de robots interactuaran mediante colisiones elásticas sobre una plataforma de roce despreciable y se obtuvo la función itinerario de cada uno de los robots dentro de estos grupos. En las etapas sucesivas este número se incrementó mediante la adición de 5 robots cada vez. Según

lo anterior, la primera medición se realizó con 5 robots, la segunda con 10, la tercera con 15, hasta completar un máximo de 25 robots interactuando a la vez. En cada una de las experiencias anteriores, se reemplazaron las baterías ya utilizadas por baterías nuevas, para que la carga de las baterías no afectara las mediciones.

En las experiencias sucesivas, los robots disponían de imanes esféricos en las caras laterales, produciendo fuerzas de atracción o repulsión entre individuos dependiendo de la polaridad con la que se encontraban estos imanes. Al igual que en caso anterior, durante la realización de estas experiencias, se varió la cantidad de individuos presentes en el área de interacción, comenzando con 5 robots hasta completar un total de 25 aumentando de 5 en 5 cada vez. En ambos casos, las mediciones fueron realizadas durante 60 [seg] y se registró mediante la cámara *OptiTrack Trio™* la trayectoria de cada uno de los individuos para su posterior análisis. Durante este aálisis, que corresponde al análisis del comportamiento colectivo, se determinaron las características cinemáticas del centro de masas tal y como se tratase de una partícula puntual. Las características determinadas fueron: trayectoria, desplazamiento medio, velocidad media y existencia de orientaciones preferentes.

Las experiencias realizadas con los robots *CheaViBot* siguen la misma lógica del proceso anterior, con la diferencia, que no se realizó la caracterización de la velocidad individual cuando la trayectoria del robot se encuentra restringida. Descartando dicha etapa, el proceso fue el siguiente: En primer lugar, utilizando la cámara *OptiTrack Trio™*, se registraron las trayectorias de 5 robots que poseían total libertad de movimiento durante un período de 30 [seg]. Cada uno de estos robots se encontraba conectado a la fuente de alimentación que se utilizó para determinar el comportamiento colectivo (pila de botón de 3 [V]). Con estas trayectorias se procedió a determinar el comportamiento cinemático medio presentado por estos robots, es decir, trayectoria, desplazamiento medio, velocidad promedio y la existencia de orientaciones preferentes.

La determinación de las propiedades cinemáticas durante el comportamiento colectivo, corresponde a un proceso análogo al realizado con los robots *Cube*. En primer lugar, se registró la trayectoria de cada uno de estos robots (*CheaViBots*) cuando se les permite interactuar por colisiones simples. Comenzando con un grupo compuesto por dos robots e incrementando la población de 2 en 2, hasta completar un total de 8. La última etapa, al igual que en el caso de las experiencias realizadas con los robots *Cube*, corresponde a la interacción de los robots *CheaViBots* cuando cuentan con imanes adosados a su estructura (en este caso, cada uno de los robots contaba con 4 imanes). A partir de los datos recopilados en ambas experiencias, se procedió a determinar las características del centro de masas, es decir, trayectoria, desplazamiento medio, velocidad promedio y existencia de orientaciones preferentes, para cada una de las experiencias.

3.2. Construcción de plataformas robóticas

Para la construcción de ambas plataformas robóticas se consideró el uso de elementos comerciales tanto para su sistema motriz como para su sistema de activación. En el caso de los robots *Cube* fueron considerados los siguientes elementos: un motor con una masa excén-

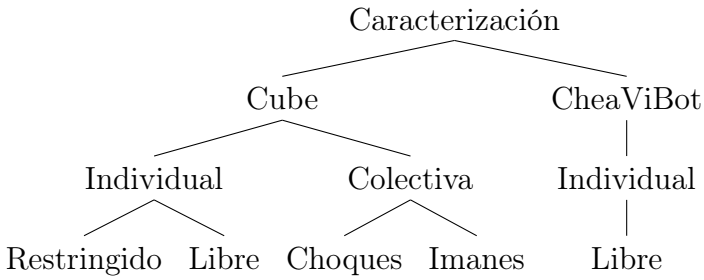

Figura 3.1: Diagrama para caracterización

trica adosada a su eje que produce la vibración del robot, una batería que energiza al motor, una fotorresistencia y un transistor, (que componen el sistema de encendido del robot) y una estructura que contiene los elementos anteriores. Para los robots *CheaViBot*, se duplican algunos de estos elementos (2 motores, 2 fotorresistencias y 2 transistores), agregándose una placa PCB [2] que contiene el circuito de encendido y un porta-pilas. Dado que los requerimientos energéticos y espaciales de este robots aumentaron, se utilizó una batería de mayor capacidad y se modificó la estructura de soporte.

De acuerdo a lo señalado anteriormente, solamente las estructuras de soporte fueron diseñadas y construidas. El proceso de construcción de las bases de soporte, sigue los pasos de para la construcción de elementos en impresoras tridimensionales. El diseño fue realizado en el programa *SolidWorks*™, dado que posee mayor compatibiliada con los procesos de fabricación digital. Se comienza con el diseño de la pieza a ser fabricada, teniendo la precaución de guardar un respaldo en formato STero Lithography (STL). Respaldar el archivo en este formato es un requisito para poder convertir el archivo a una extensión que pueda ser interpretada por la impresora para construir la(s) pieza(s). La conversión antes mencionada se realizó por medio del programa MakerWare™en su versión 2.4.1. Este programa, entre otros, genera documentos con la extensión apropiada ("*.x3g") para la impresora MakerBot2™, que es la que se ocupó en esta ocasión.

El proceso de impresión comienza con la selección y posicionamiento de la(s) pieza(s) a ser impresa(s). En la Figura 3.2, se observa una captura de pantalla del software MakerWare 2.4.1. En esta imagen se observa un esquema que emula la plataforma de impresión real. En ella se puede establecer con claridad la ubicación y disposición que tendrá(n) la(s) pieza(s) durante la impresión. En particular, se presenta la forma en la cual se dispusieron las bases de los *Cubes* para su impresión.

Una vez especificada la disposición que tendrán las piezas durante la impresión, se procede a generar el archivo que será interpretado por la impresora. Al generar este archivo es necesario especificar una serie de parámetros que se indican en la Figura 3.3. En la imagen de la izquierda de esta Figura (3.3) se presenta la primera pantalla para selección de parámetros. Recorriendo la imagen desde el la parte superior, se encuentra:

- La acción a realizar con el archivo generado por el programa (Imprimir directamente el archivo conectando la impresora o exportar el archivo para su impresión posterior).

[2] Acrónimo del inglés: *Printed Circuit Board* o tarjeta de circuito impreso.

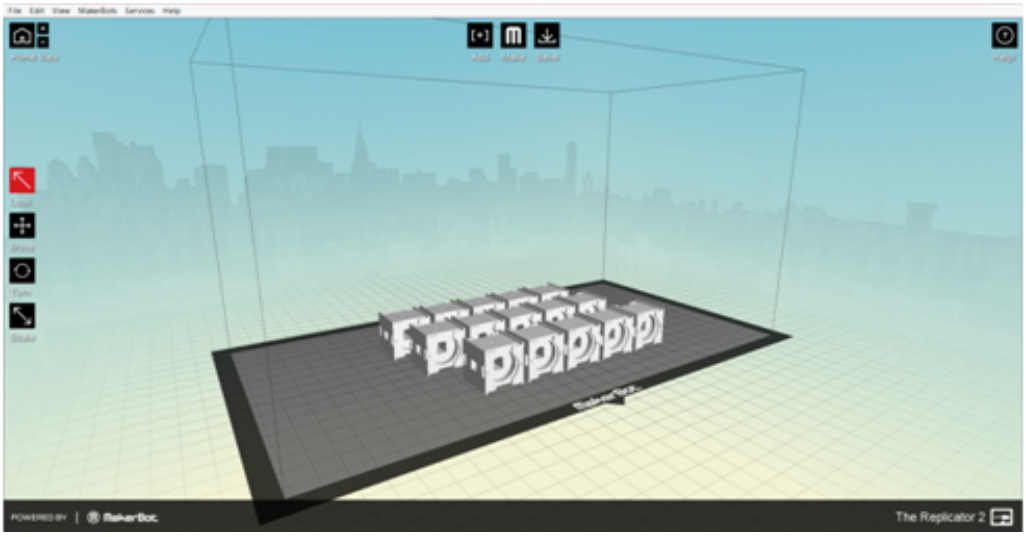

Figura 3.2: Captura del programa MakerWare 2.4.1.

- Selección del modelo de impresora con el que se construirá la pieza.
- El material a ser utilizado.
- La resolución de la impresión , que varía en tres niveles (alta, media y baja calidad).
- Selección de la presencia o ausencia de una superficie base para la construcción de la pieza y la presencia o ausencia de una estructura auxiliar de soporte para las secciones de la pieza que se encuentran en voladizo durante la construcción.

En la parte inferior de esta misma captura, se encuentra desplegada la viñeta que define la calidad de la impresión. Primero se selecciona el porcentaje de relleno que tendrán las piezas a ser construidas, este rango varía desde 0 hasta 100 %. Donde, 0 % indica que la pieza solo se forma por la capa más externa y 100 % define una pieza maciza. En el centro de este menú se encuentra la selección del número de capaz que conforman la capa más externa. Al final de esta viñeta se especifica el espesor para la capa más externa.

En la captura del centro se especifica la temperatura a la cual será extruído el plástico. Para teminar con la decripción de los parámetros a ser seleccionados, en la captura de la derecha, se selecciona la velocidad de extrusión del plástico y la velocidad con la que se desplazaá el extrusor mientas construye la pieza. Estos parámetros fueron seleccionados de acuerdo a las recomendaciones del fabricante.

Una vez reunidas las piezas necesarias, se procedió con el ensamblaje de los robots. Este ensamblaje, en el caso de los robots *Cube* requirió la soldadura directa, mediante estaño, de los elementos constituyentes para conformar el circuito de encendido. En el caso de los robots *CheaViBot* el proceso de ensamblaje consistió en la soldadura de los componentes a la placa PCB. Los imanes descritos en la sección anterior fueron incorporados una vez realizadas las pruebas correspondientes a las colisiones simples.

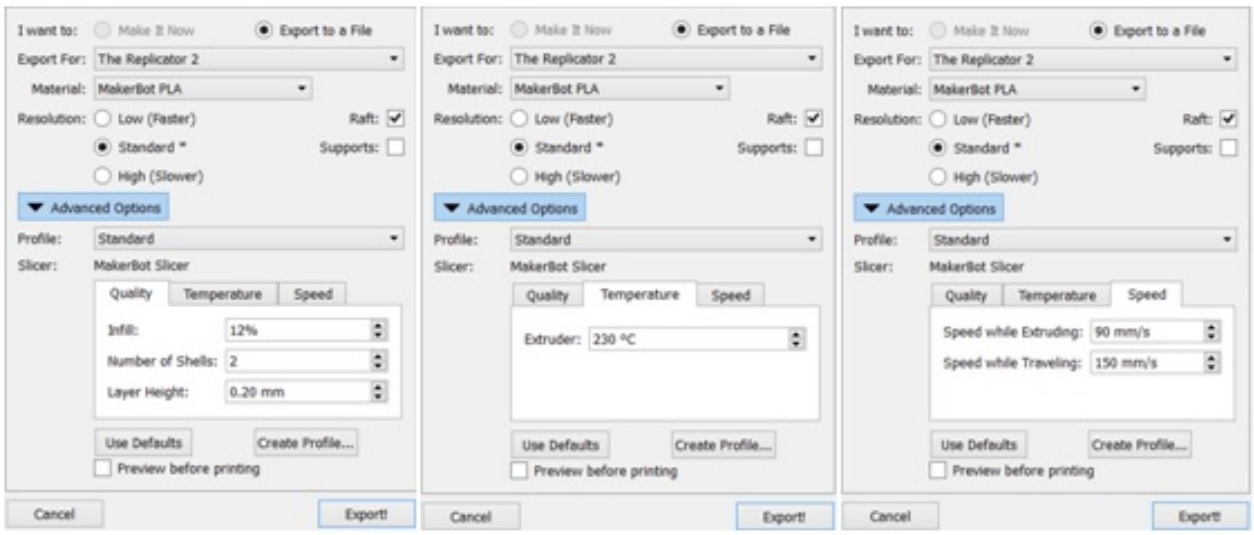

Figura 3.3: Parámetros de Impresión en el programa MakerWare 2.4.1. Fuente: Captura de pantalla del programa MakerWare ™versión 2.4.1.

3.3. Obtención de trayectorias

A lo largo de las secciones precedentes se hizo mención a la obtención de las trayectorias de los robots mediante la utilización de la cámara *OptiTrack Trio* ™. La cámara es en realidad un complejo de tres cámaras de la marca OptiTrack ™, (Figura 3.4). La acción combinada de estas tres cámaras entrega, mediante un programa proporcionado por el fabricante, la posición espacial de una partícula que se encuentran dentro de su cono de visión (FOV[3]), con una frecuencia de muestreo de 200 [fps]. Para que lo anterior sea posible, cada elemento a ser muestreado debe contar con un marcador reflectante que lo haga visible para la cámara. Si bien la cámara cuenta con marcadores comerciales ofrecidos por el fabricante, se desestimó la compra de los mismos, dado que su volumen es proporcional al tamaño de los robots que se consideraron para realizar las experiencias. Por lo tanto, fue necesario realizar una sub-etapa para determinar empíricamente el tamaño óptimo de los marcadores con los que fue dotado cada robot.

El procedimiento para obtener la trayectoria de un marcador en el tiempo fue el siguiente. Mediante la interfaz para el usuario proporcionada por el fabricante de la cámara, *Motive*™, se seleccionó la opción para obtener trayectorias en cada una de las cámaras. Una vez que dicha opción fue seleccionada, se realizó una grabación pulsando el botón dispuesto para este fin dentro del programa. Concluida la grabación, se procedió a la trayectorización de la grabación, al seleccionar dicha opción dentro del programa, para finalmente, se exportar el resultado de este proceso en un archivo tipo "*.CSV[4]", para su posterior análisis.

Una vez que las funciones itinerario fueron obtenidas en formato "*.CSV", se procedió al análisis de los datos obtenidos. Para esto, se utilizó el programa *Matlab*™, dentro del cual se implementaron las funciones correspondientes para la realización de cálculos y el desarrollo

[3]Acrónimo del inglés: "Field Of View"
[4]Comma Separated Values

Figura 3.4: Cámara Optitrack modelo Trio. Fuente: Imágenes proporcionadas por el fabricante (http://www.optitrack.com).

de los gráficos pertinentes. La descripción de las funciones implementadas se encuentra en el capítulo Resultados y el código se adjunta en el **??**. El análisis realizado a los datos obtenidos para el comportamiento individual contempló: Obtención de su trayectoria, desplazamiento medio, desplazamiento según las abscisas, desplazamiento según las ordenadas, obtención de la rapidez media, la función velocidad y la velocidad media, calculo de las funciones de distribución de probabilidades para el eje de las abscisas, las ordenadas, del desplazamiento medio y ángulo de orientación. En el caso de las experiencias grupales, el análisis fue el mismo, solo que esta vez se agregó el análisis de las propiedades grupales, que fueron tratadas como si se tratara de un solo robot cuya función itinerario correspondiera a las coordenadas del centro de masas de los integrantes del grupo.

El análsis de los datos generados mediante la simulación computacional, fue mucho más directa, dado que en la implementación del programa, se creó una rutina de comandos que exportaba los datos directamente a un formato legible por el programa *Matlab™*.

3.4. Modelo Realista

Junto con la realización de las mediciones experimentales a las plataformas robóticas, se implementó un modelo computacional de los robots. Este modelo tuvo por objetivo la realización de experiencias que contemplaban una mayor cantidad de individuos. Para que lo anterior tuviese sentido, se validaron los resultados obtenidos para estas simulaciones al compararlos con los datos obtenidos durante las experiencias realizadas con las plataformas robóticas.

El modelo se desarrolló utilizando la librería[5] *Open Dynamics Engine (ODE)* que pertenece al lenguaje de programación *C++*. Esta librería permite realizar simulaciones de sólidos tridimensionales dado que cuenta con módulos para la asignación de fuerzas, detección de colisiones y un variado set de geometrías básicas para el modelado de sistemas dinámicos. Otra de la garantías ofrecidas por esta librería es la posibilidad de ejecutar una interfaz gráfica donde se pueden observar los fenómenos que ocurren durante la ejecución del programa. El detalle de la implementación se encuentra en la sección **??** en el capítulo Resultados y el código de programación está disponible en el **??**.

[5]Programa que contiene funciones específicas

3.5. Accesorios

En la presente sección se describen procesos anexos a los realizados durante la investigación. Estos procesos anexos permitieron determinar una serie de parámetros de diseño utilizados en la construcción y caracterización de las plataformas robóticas.

Determinación del tamaño de los marcadores De acuerdo a lo mencionado en las sección Obtención de trayectorias, para poder registrar las trayectorias de una partícula utilizando la cámara *OptiTrack™*, se requiere que dicha partícula cuente con un marcadro adosado. En el caso de las plataformas que fueron estudiadas, se desestimó la utilización de los marcadores ofrecidos por el fabricante de la cámara, dado que su tamaño era excesivo en comparación con el tamaño de los robots. En función de lo anterior, se realizaron pruebas para determinar el tamaño óptimo de los marcadores que serían utilizados. Para esto se impusieron dos condiciones: La primera hace referencia al tamaño excesivo del marcador, es decir, la cámara debía ser capaz de distinguir con claridad que se trataba de dos partículas independientes en todo momento. La segunda restringe el tamaño mínimo del marcador, de modo que los robots debían ser visibles en todo momento. La primera restricción fue estudiada posicionando dos robots *Cube* a la mínima distancia a la que se podrían encontrar durante la experimentación y se fueron probando distintos tamaños de marcadores, comenzando desde uno que cubría la totalidad de la superficie superior del robot hasta llegar a un tamaño en el cual la cámara ya no se confundía. La segunda restricción se inspeccionó al ubicar el robot con el último marcador utilizado en la fase anterior, en las posiciones más lejanas a las que podían acceder los robots dentro del área de experimentación.

Capítulo 4

Resultados

En este capítulo se presentan los resultados obtenidos en cada una de las etapas descritas en el capítulo anterior.

La primera sección describen y caracterizan geometricamente las dos plataformas robóticas construidas durante la realización del presente trabajo (*Cube* y *CheaViBot*) además de presentar el montaje utilizado para la realización de las experiencias.

En la segunda sección se presentan los resultados obtenidos durante la caracterización cinemática individual de ambas plataformas.

En la tercera sección se describen los modelos computacionales que permitieron escalar las experiencias a un número mayor de individuos, para el caso de las experiencias realizadas en *Cubes* y estudiar la maniobrabilidad de la plataforma *CheaViBots*. Estos modelos fueron programados en lenguaje C++ utilizando la librería *Open Dynamics Engine (ODE)*.

En la cuarta sección, se presentan los resultados obtenidos para la caracterización cinemática individual del modelo computacional de la plataforma *Cube*.

Finalmente, se presentan los resultados obtenidos para el comportamiento colectivo conseguido con ambas plataformas robóticas. Al igual que en las secciones precedentes primero se detallarán los resultados de las experiencias realizadas con *Cubes* y en segundo lugar para las experiencias realizadas con *CheViBots*. La exposición, se realiza en orden creciente, en función de la complejidad de la interacción presente entre los individuos y de cantidad de individuos participantes.

4.1. Plataformas robóticas

La primera plataforma robótica construida corresponde a los *Cubes*. Como se mencionó en el capítulo anterior, estos robots corresponden a una plataforma que cumplirá una doble función. Por un lado deben ser útiles para realizar experiencias de comportamiento colectivo

y por otro lado deben dilusidar cuestionamientos prácticos referentes al trabajo con este tipo de robots. Por otra parte, la plataforma *CheaViBot* se considera como el producto final del presente trabajo, por lo tanto, presenta todas las mejoras que se consideraron competentes para mejorar el desempeño presentado por los *Cubes*. A continuación se procede a describir ambas plataformas.

4.1.1. Construcción Cubes

De acuerdo a lo presentado en el capítulo de antecedentes, los robots a ser construidos para el estudio del comportamiento colectivo deben cumplir con una serie de características. Para la construcción del primer conjunto de robots, fueron considerados los siguientes parámetros de diseño:

- Activación automática, es decir, que no fuese necesario ponerlos en funcionamiento individuo por individuo.
- Geometría simple.
- Capacidad de locomoción auto-propulsada.
- Posibilidad de incluir accesorios para la interacción entre individuos.
- Necesidad de incluir un marcador que permita seguir la trayectoria de las unidades utilizando la cámara *OptiTrack Trio*.
- Sistema de energización independiente para cada individuo.
- Reducir los costos de producción.

De acuerdo con los criterios anteriores, se definió que el robot tendría la siguiente configuración:

- La activación de cada individuo se realizará por medio de un circuito que encendiera el robot cuando detecte la presencia de luz.
- La geometría de soporte de los componentes de cada robot será un cubo de 22 x 22 x 22 $[mm^3]$ construido utilizando impresoras tridimensionales.
- El sistem de locomoción sería por medio del fenḿeno "Stick-slip", detallado en la sección de antecedentes.
- Los accesorios a ser utilizados serían: imanes de Neodimio ubicados en las caras laterales del cubo.
- El marcador será un cuadrado de cinta reflectante de 3 x 3 $[mm^2]$ ubicado en la cara superior del cubo.
- La energización de cada robot se conseguirá por una batería de boton de 3 [V].

En función de lo anterior, se definieron dos categorías de componentes. Los elementos comerciales y los que serían fabricados. Los elementos en ambas categorías debían satisfacer la restricción de minimización del peso, volumen y costo. Los elementos comerciales fueron:

- Motor: 1 Motor vibrador para celulares Nokia de la serie 3000 (Figura 4.1), masa: 1.20 [gr].

- Batería: 1 pila de botón Maxell CR2016 Micro Lithium Cell, 3[V] (Figura 4.2), masa:1.7 [gr].
- Fotorresistencia: 1 Fotorresistencia modelo FC 107 (Figura 4.4), masa: 0.35 [gr].
- Transistor: 1 transistor PN2222a (Figura 4.3), masa: 0.20 [gr].
- Imán: 2 imanes esféricos de neodimio, radio: 2.5 [mm], masa: 0.50 [gr] cada uno.

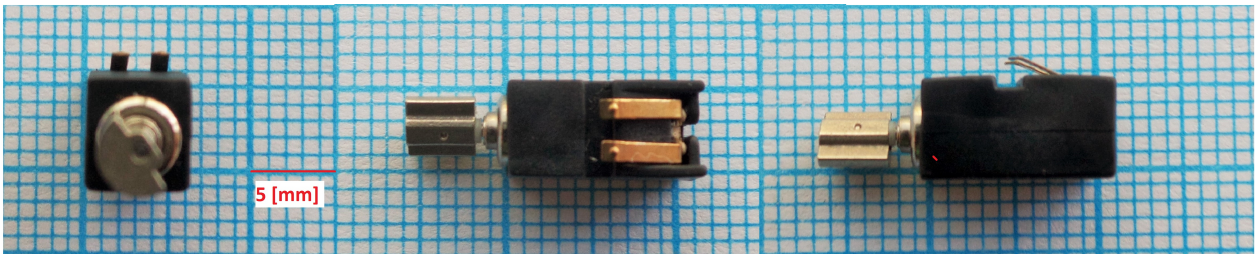

Figura 4.1: Motor utilizado para la construcción de los robots modulares.

Figura 4.2: Bateria de botón utilizada en los robots modulares.

Los elementos construidos fueron:

- Estructura de soporte de los componentes: Construida por medio de impresoras tridimensionales.
- Porta-pila: Construida por medio de dos placas de cobre.

Base de soporte Cubes Esta estructura tiene por objetivo contener a los demás constituyentes del robot *Cube*. En la Figura 4.5 se señalan los elementos de esta base. En la esquina superior izquierda (Figura4.5 1), se encuentra una vista isométrica de la estructura. En la imagen que se encuentra en la esquina superior derecha (Figura 4.5 2), corresponde a la vista lateral de la base. En esta vista se pueden observar dos componentes. El primero, señalado con la letra A indica una cavidad de 3 x 3 x 3 $[mm^3]$ destinada a contener un imán de Neodimio. El segundo, identificado con la letra B indica las ranuras donde se depositan

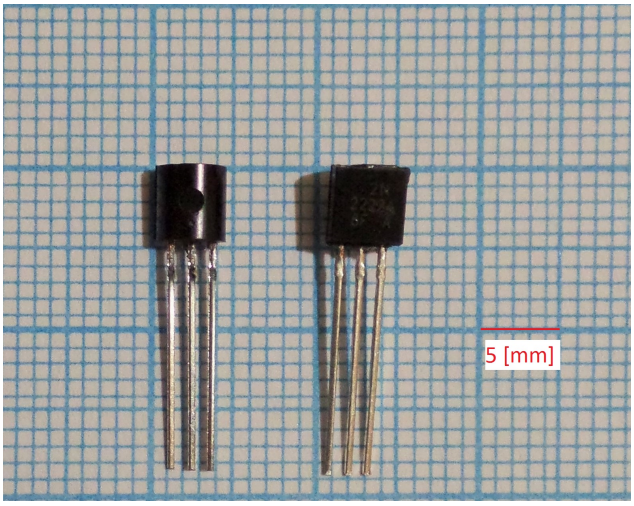

Figura 4.3: Transistor utilizado en la construcción de los robots *Cube*

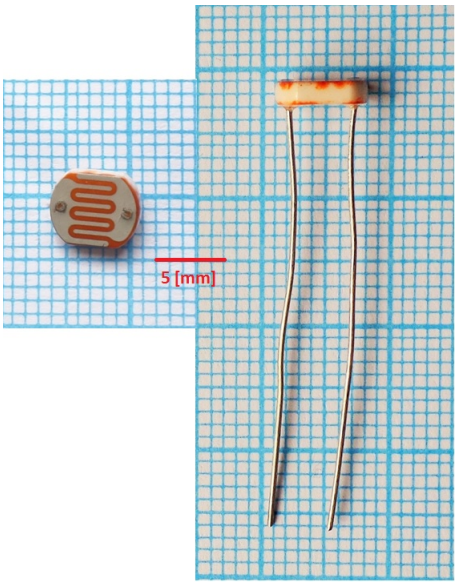

Figura 4.4: Fotorresistencia utilizada en la construcción de los robots *Cube*

los contornos de la pila de botón, esto tienen por objetivo evitar el deslizamiento de la pila fuera del robot. La imagen correspondiente a la Figura 4.5 3 (esquina inferior izquierda), corresponde a la vista superior del cubo. En ella se puede identificar el marcador que indica el centro geométrico de la cara superior, señalado con la letra E. Este marcador tiene por objetivo indicar el lugar donde se situará el marcador que hará que el cubo sea visible por la cámara *OptiTrack Trio*™. Finalmente, en la imagen inferior derecha, se presenta una vista frontal de la base. En ella se pueden distinguir dos perforaciones señaladas con la letra D y una cavidad, señalada con la letra C. Las primeras dos perforaciones, de 1 [mm] de diámetro cada una, se encuentran tanto en la cara anterior como en la cara posterior de la base (4 en total). Su objetivo es permitir incluir dispositivos de acople elástoméricos entre los *Cubes*, durante las pruebas de comportamiento colectivo a ser realizadas. La cavidad señalada con la letra C, es una perforación pasante cuya geometría corresponde a la del contorno del motor. Esto permite que el motor permanezca firmemente acoplado, dado que el motor cuenta con

un recubrimiento plástico (ver Figura4.1).

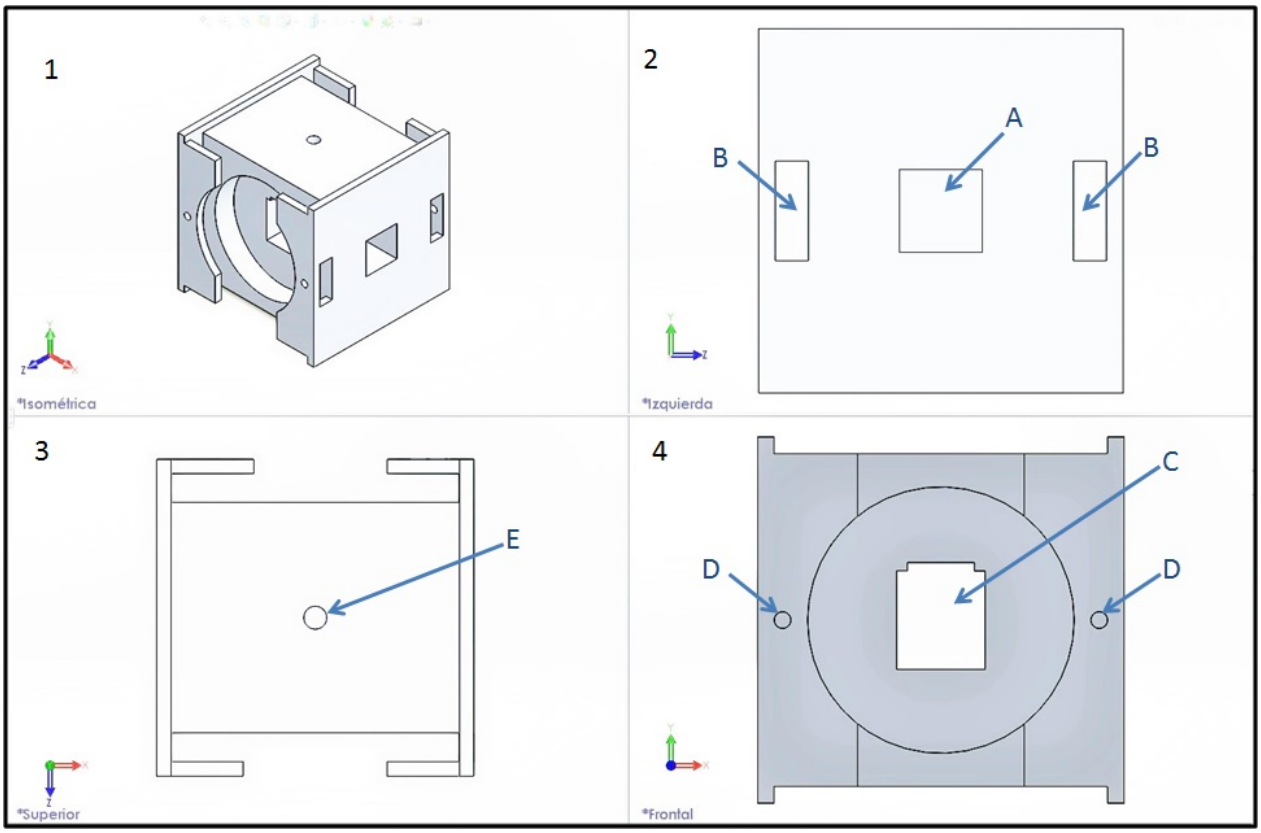

Figura 4.5: Base de soporte para *Cubes* impresa en PLA. fuente: Captura de pantalla desde el programa SolidWorks™

Los parámetros seleccionados en el programa MakerWare 2.4.1 para la construcción de esta pieza, son los que se presentan en la tabla 4.1.

El resultado del proceso anterior se muestra en la Figura 4.6. Al igual que en la Figura 4.5, se presentan las vistas isométrica, lateral, alzado y frontal de una de las bases para los robots *Cube*.

Porta-pila El segundo componente a ser construido en los robots *Cube*, corresponde al porta-pilas. Este elemento se compone de dos placas de cobre de 16 [mm] de diámetro, con un espesor de 0.1 [mm]. Estas placas cumplen la función de terminales para la pila de botón, al ubicarse una en el terminal positivo de la batería y una en el terminal negativo.

Ensamblado de robots *Cube* El ensamble de los robots *Cube*, se realiza de acuerdo al circuito que controla su encendido. El circuito se presenta en la Figura 4.8 y se compone de 4 componentes: Una fuente de alimentación (bateria de botón de 3 [V]), un motor vibrador de celular, un transistor modelo PN-2222A y una fotorresistencia FC 107. El ensamblado se realiza en dos etapas paralelas. En la primera de ellas, se soldan dos cables de 0.6 [mm] a los terminales del motor, para luego ubicar el conjunto motor-cables en la cavidad destinada

Tabla 4.1: Parámetros constructivos base *Cube*.

Parámetro	Valor asignado
Resolución	estándar
Capa de soporte	Activada
Soportes auxiliares	Desactivados
Archivo	estándar
Relleno	12 %
Número de capas	2
Ancho de capas	0.20 [mm]
Temperatura extrusor	230 C
Velocidad de extrusión	90 $[\frac{mm}{seg}]$
Velocidad de viaje extrusor	150 $[\frac{mm}{seg}]$

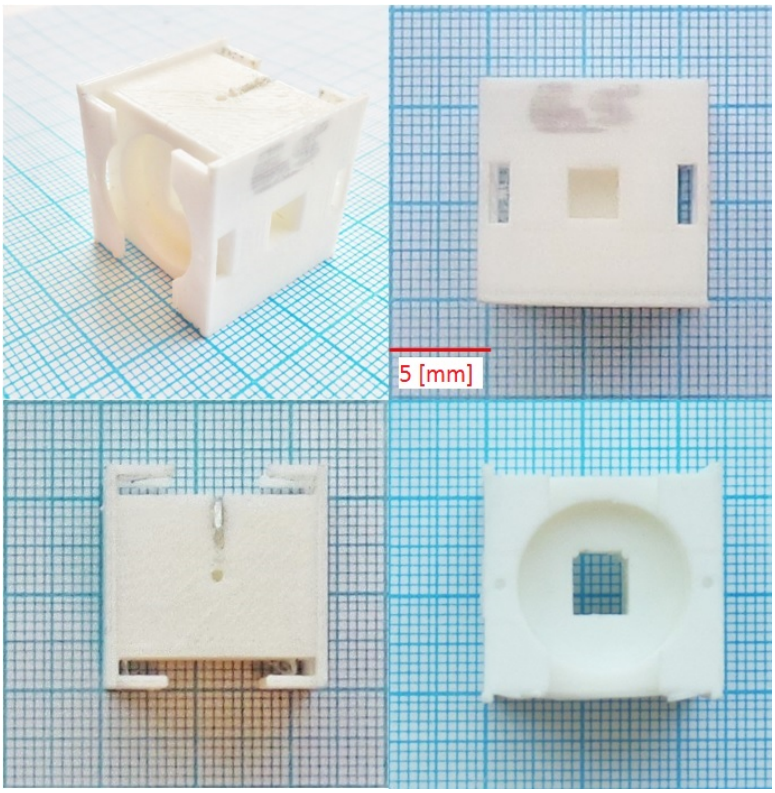

Figura 4.6: Estructura de soporte para los robots *Cube*

para este fin en la base de los *Cubes*. En paralelo se soldan los terminales correspondientes de la fotorresistencia con los terminales del transistor. Finalmente se procede a conectar el porta-pila (placas de cobre). Finalmente, se conecta el extremo negativo del porta-pilas con el emisor del transistor. El resultado es el que se muestra en la Figura 4.7.

Funcionamiento de *Cubes* El funcionamiento de los robots *Cube*, se rige por el principio "Stick-slip". Este movimiento es producido por el giro de un motor que posee una masa excéntrica adosada a su eje. Este sistema puede ser implementado de muchas maneras, la más simple de ellas consiste en conectar directamente el motor a la batería. Sin embargo,

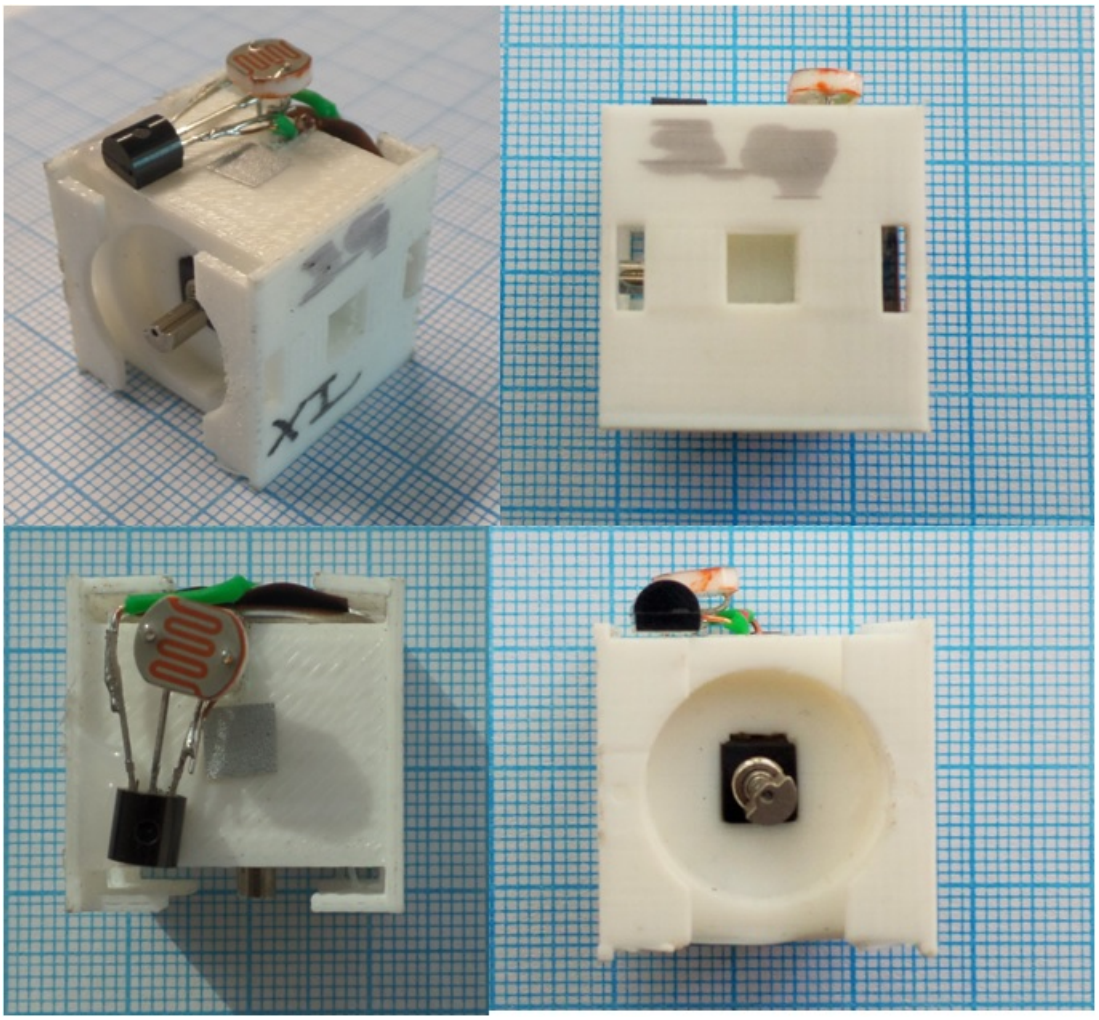

Figura 4.7: Robot *Cube* ensamblado.

la implementación de este sistema implica que para iniciar y/o detener el funcionamiento de cada robot, sería necesario remover/introducir una batería. Convirtiendo esta tarea en algo tedioso e impracticable al momento de trabajar con varios robots. Es por esto, que se implementó un sistema de activación "automático" que permite controlar la puesta en marcha de muchos individuos de forma simultánea. Este sistema corresponde a un circuito electrónico de activación en función del nivel de luz percibido por una fotorresistencia. El funcionamiento se detalla a continuación: En la parte superior de cada uno de los robtos *Cube*, se encuentra una fotorresistencia conectada de acuerdo al circuito detallado en la subsección anterior. Dependiendo de la cantidad de luz que este elemento reciba, el transistor alternará entre sus estados de corte y saturación. El primer estado o estado de corte, corresponde a la no conducción de electricidad por parte del transistor. En este circuito, este estado se alcanza cuando no incide luz sobre la fotorresistencia. Esto se debe a que la oposición al paso de corriente que presenta este elemento en ausencia de luz es muy alta impidiendo el flujo de corriente entre los terminales base y colector del transistor. Este estado puede ser asimilado como un circuito abierto, bajo el cual el motor no gira, y por tanto, el robot se mantiene en estado de reposo.

Por el contrario, cuando incide una cantidad suficiente de luz sobre la fotorresistencia, la oposición al paso de corriente eléctrica disminuye drásticamente. En consecuencia, existe flujo de corriente entre los terminales base y colector. Al mismo tiempo, el flujo de corriente entre colector y emisor se encuentra activo, por lo que el motor puede funcionar.

Cabe destacar que estados de luz intermedios, producirán el giro del motor con una intensidad menor. Este hecho será aprovechado para el control de los robots *CheaViBots*.

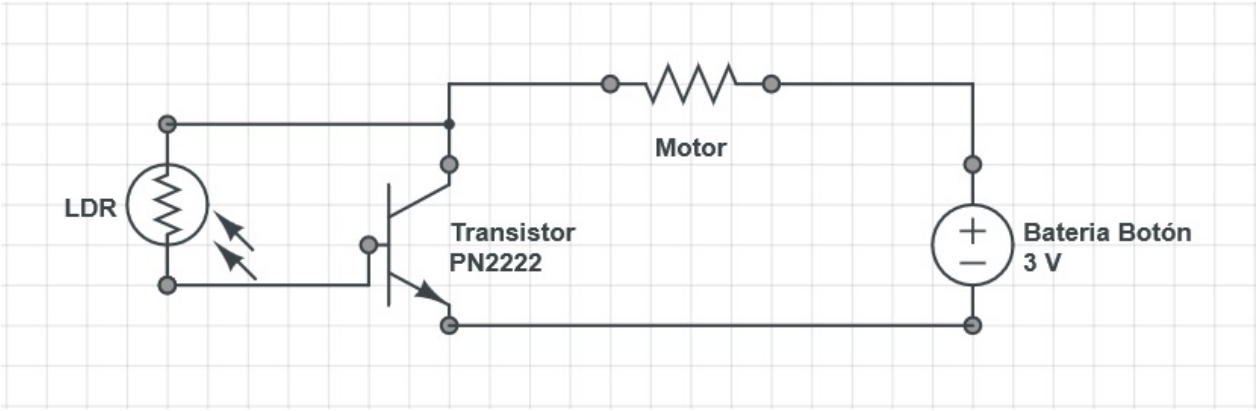

Figura 4.8: Circuito utilizado en los robots *Cube*. Fuente: Modelo desarrollado, en el servidor online http://www.circuitlab.com

4.1.2. Construcción CheaViBots

Este robot fue concebido para desarrollar experiencias dentro del campo de la robótica modular y el comportamiento colectivo. En función de lo anterior su diseño tiene características de modularidad, es decir, sus componentes pueden ser intercambiados dependiendo de la función para la cual se desea emplear.

Teniendo como precedente la construcción de los robots *Cube* y las experiencias realizadas con los mismos, este robot incorpora la totalidad de las mejoras que fueron consideradas pertinentes para optimizar los resultados obtenidos con la primera plataforma. Teniendo esto en mente, se definieron los siguientes objetivos para esta nueva plataforma:

- Superar la velocidad de desplazamiento de los robots *Cube*.
- Permitir estratégias de control externo.
- Aumentar la cantidad de interacciones a ser incluidas.

Por lo tanto, y de acuerdo a las experiencias de otros investigadores de la materia, se determinó contruir una plataforma robótica con las siguientes características:

- Tener solamente 3 apoyos.
- Tener dos motores para mejorar la maniobrabilidad.
- Contar con dos sistemas de activación independientes para cada motor.

- Aumentar a 4 la cantidad de imanes que puede albergar cada base.
- Contar con una placa de circuito impresa (PCB[1]), para facilitar su ensamblado.

La construcción de la plataforma *CheaViBot* siguió la misma lógica empleada para la construcción de los robots *Cube*, es decir, se definieron los componentes a ser comprados (comerciales) y los componentes a ser construidos. Los componentes comerciales fueron:

- Motor: 2 Motores vibradores para celulares Nokia de la serie 3000 (Figura 4.1), masa: 1.20 [gr] cada uno.
- Batería: Maxell CR2032 Micro Lithium Cell, 3[V] (Figura 4.2), masa:2 [gr].
- Fotorresistencia: 2 fotorresistencias modelo FC 107 (Figura 4.4), masa: 0.35 [gr] cada una.
- Transistor: 2 transistores PN2222a (Figura 4.3), masa: 0.20 [gr] cada uno.
- Imán: 4 Imanes esférico de neodimio, radio: 2.5 [mm], masa: 0.50 [gr] cada uno.
- Portapilas: 1 portapila comercial para pilas de botón tipo 2032 (Figura 4.9), masa: 2.5 [gr].
- Circuito impreso: Disco de 30 [mm] de diámetro que contiene el circuito diseñado para el funcionamiento del robot (Figura 4.10). masa 3 [gr].

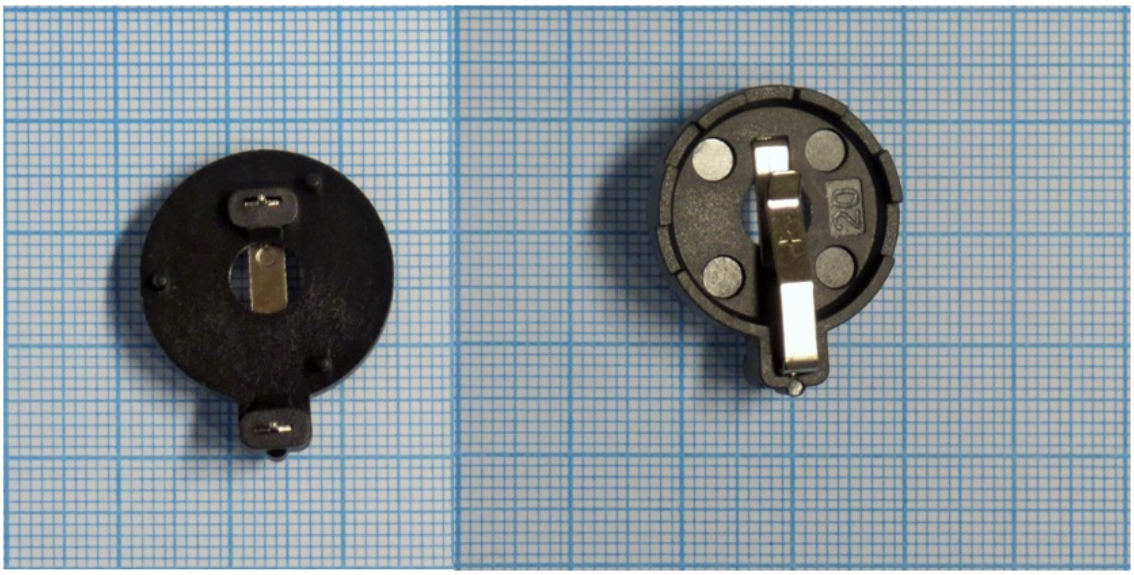

Figura 4.9: Porta-pila utilizado en la construcción de la plataforma *CheaViBot*.

En función de lo anterior, solamente la base de soporte para los *CheaViBots* fue diseñada y construida. Esta base fue construida utilizando impresión tridimensional, siguiendo la rutina ya descrita anteriormente. En la Figura 4.11 se presenta el diseño se presenta el diseño a partir del cual se construyeron las bases para los robots *CheaViBots*. En la imagen superior izquierda, se presenta una vista isométrica de la pieza. En la imagen superior derecha se observa una vista lateral, en la cual se pueden distinguir los siguientes componentes: con la letra A se señalan las cavidades acondicionadas para la inserción de los imanes esféricos, con la letra B se distingue el espacio acondicionado para el motor y con la letra C se señala una de las 3 patas de soporte.

[1]Acrónimo del inglés: Printed Circuit Board

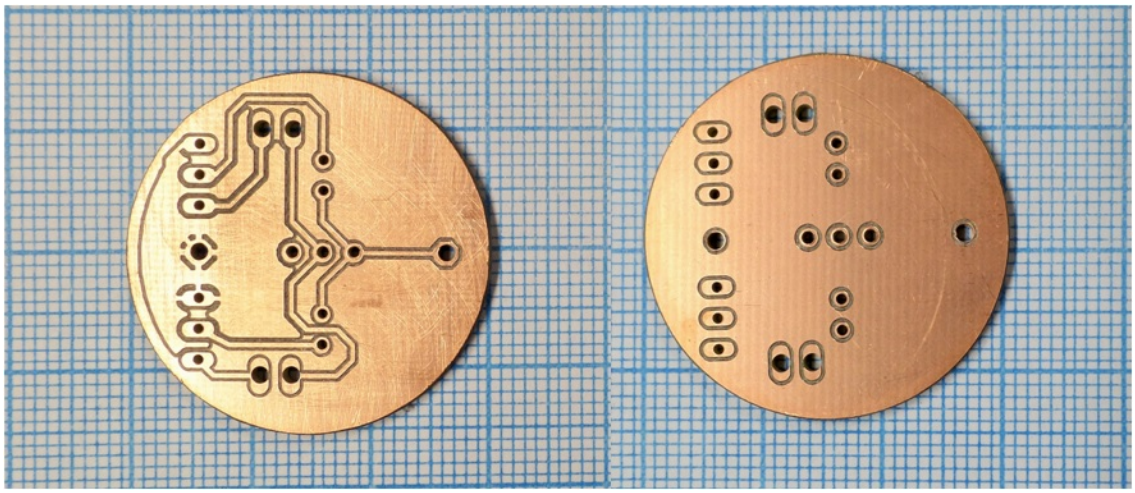

Figura 4.10: Circuito impreso para robots *CheaViBots*

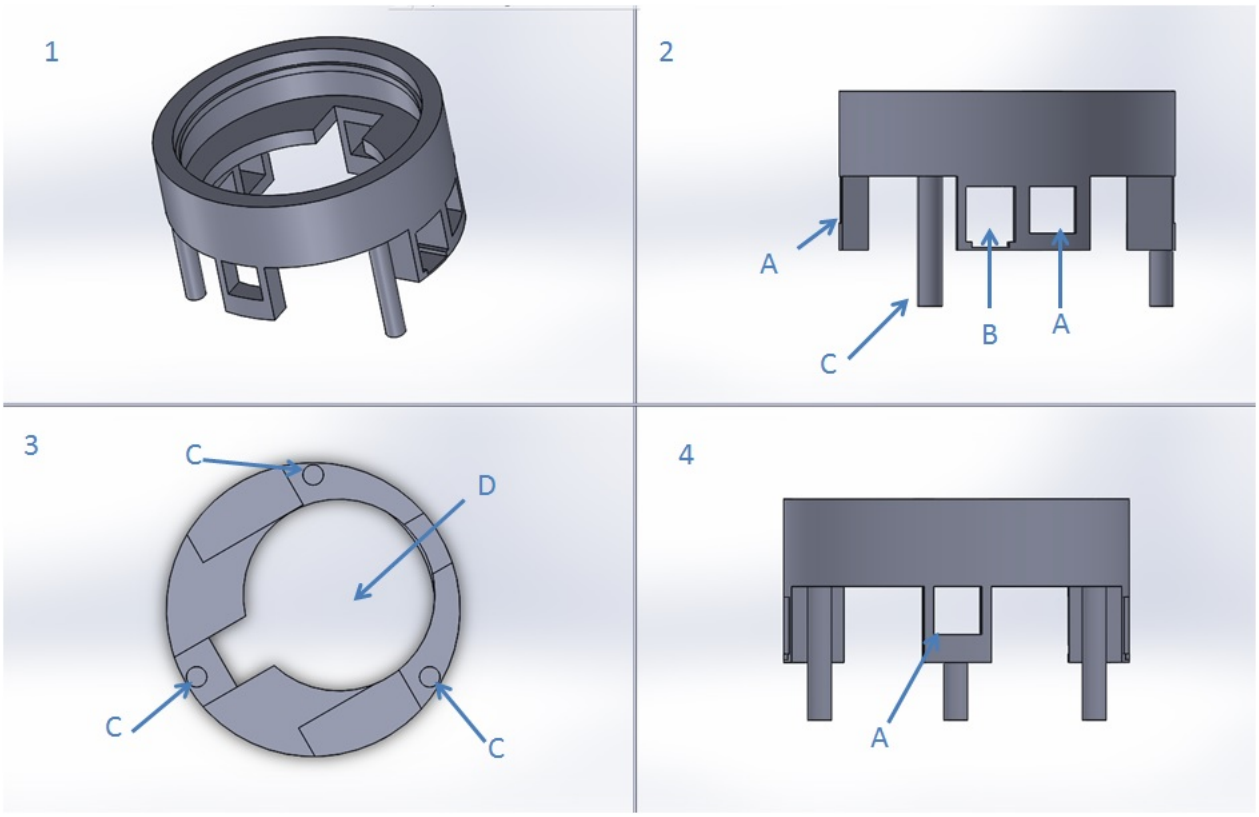

Figura 4.11: Componentes base de soporte *CheaViBot*.

En la imagen inferior izquierda se presenta una vista superior, en la cual se pueden distinguir las tres patas de soporte señaladas con la letra C y la cavidad acondicionada para recibir el porta-pilas, indicada con la letra D. Finalmente, en la imagen inferior derecha, se presenta una vista posterior de la pieza, en la cual se puede observar el cuarto espacio acondicionado para insertar uno de los imanes esféricos.

Los parámetros para la impresión de la pieza fueron los que se presentan en la tabla 4.2. A

Tabla 4.2: Parámetros constructivos base *CheaViBot*.

Parámetro	Valor asignado
Resolución	estándar
Capa de soporte	Activada
Soportes auxiliares	Desactivados
Archivo	estándar
Relleno	12 %
Número de capas	3
Ancho de capas	0.20 [mm]
Temperatura extrusor	240 C
Velocidad de extrusión	85 $[\frac{mm}{seg}]$
Velocidad de viaje extrusor	150 $[\frac{mm}{seg}]$

diferencia de los parámetros asignados para la construcción de la estructura de soporte de los robots *Cube*, en este caso se aumentó la temperatura de extrusión, se disminuyó la velocidad con la que se entrega el material de aporte durante la impresión y se aumentó la cantidad de capas externas. Esto tuvo como objetivo conseguir una pieza más rígida, para así mejorar la transmisión de la vibración inducida por el motor.

El resultado de la impresión de esta pieza, es el que se presenta en la Figura 4.12

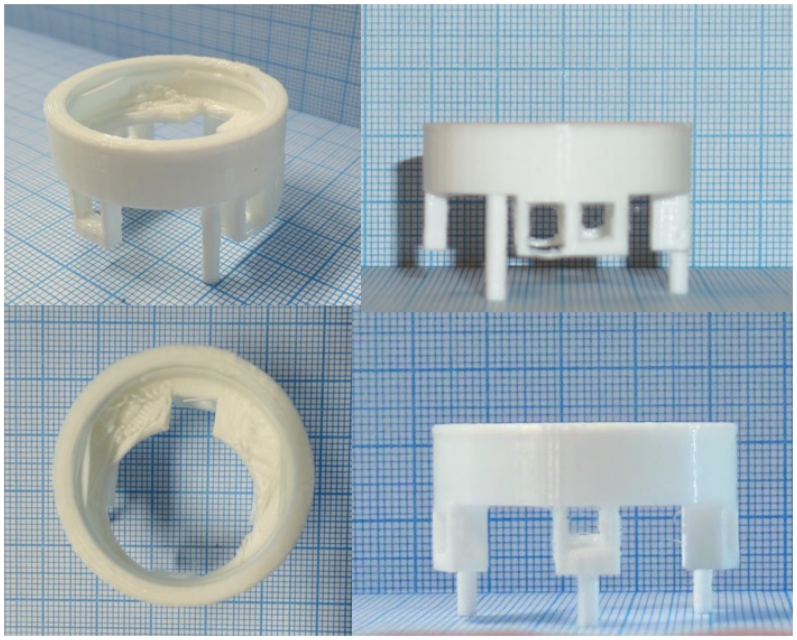

Figura 4.12: Base de soporte *CheaViBot*.

Ensamblado de robots *CheaViBot* El ensamblado de estos robots consta de dos partes, la primera consiste en la soldadura de los componentes electrónicos al circuito impreso y la segunda al montaje de este complejo sobre la estructura de soporte. El resultado de la soldadura de los componentes electrónicos es el que se muestra en la Figura 4.13. Una vez

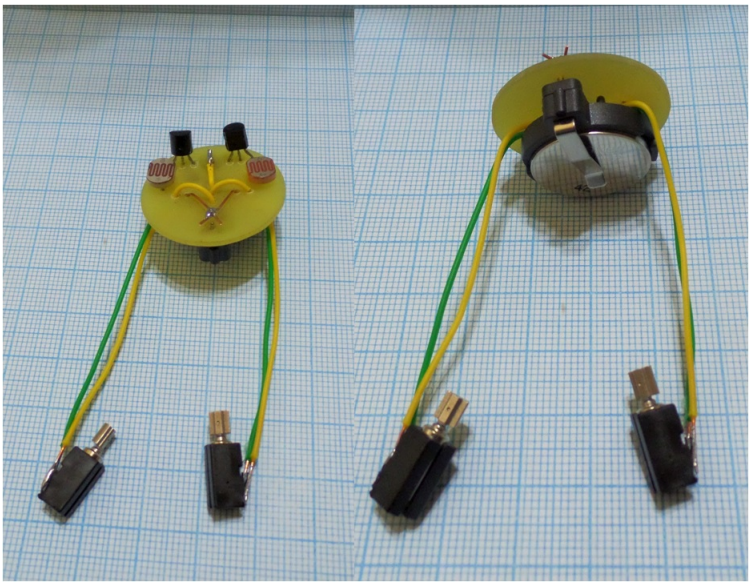

Figura 4.13: Ensamble de componentes electrónicos para *CheaViBot*.

concluido el proceso de soldadura de componentes, la fijación de este complejo sobre la base de soporte se realiza por medio de un sistema de ajuste que presiona el contorno del circuito impreso contra las paredes interiores de la base. El resultado de este proceso es el que se muestra en la Figura 4.14.

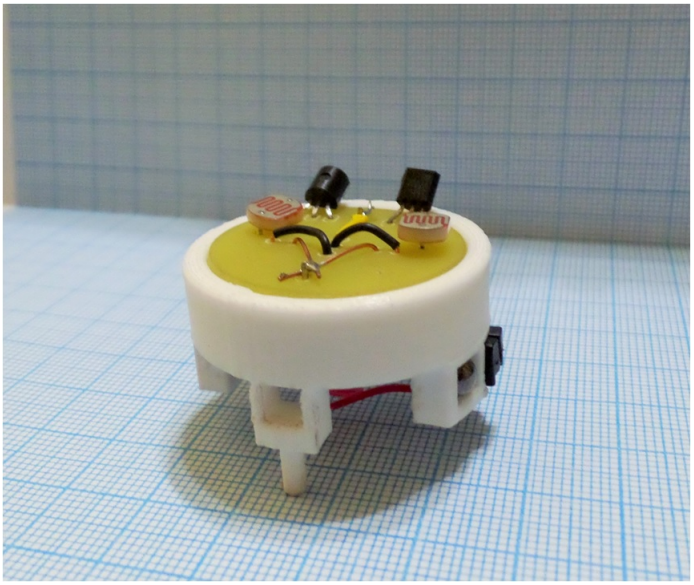

Figura 4.14: Robot *CheaViBot*.

Funcionamiento de *CheaViBots* El circuito que gobierna el comportamiento de este robot para las experiencias que se presentan en este trabajo, consiste en la duplicidad del circuito incorporado en los robots *Cube*, es decir, cada *CheaViBot* corresponde a dos *Cubes* conectados a un sistema de energización y base de soporte comunes. La principal diferencia

con un robot *Cube* radica en la disposición de los motores dentro de la base de soporte, dado que en este caso los motores se encuentran ubicados en el extremo opuesto al que se encuentra la fotorresistencia que controla su activación. En función de lo anterior, además de conseguir que cada robot se active en presencia de luz, estos serán capaces de seguir la fuente lumínica mediante la alternancia en la activación de sus motores, ya que cuando incida luz sobre la fotorresistencia que se encuentra en el costado derecho, se activará el motor del lado izquierdo provocando el giro del robot en dirección a la fuente de luz. En caso de que la fuente lumínica se encuentre del lado izquiero el proceso es análogo. En caso de que la fuente de luz incida de igual forma sobre ambas fotorresistencias, el robot avanzará en línea recta.

4.1.3. Estructura para realizar mediciones

La estructura para realizar las mediciones se compone de dos partes: La cámara *OptiTrack Trio™* y una superficie para el desplazamiento de los robots. La cámara tiene por función, entregar la posición de cada uno de los robots durante la experimentación. El procedimiento mediante el cual esto es posible fue detallado en el capítulo 3. En esta subsección se procederá a detallar las condiciones necesarias para que las mediciones sean realizadas correctamente. El primer punto a ser garantizado es que los marcadores ubicados en la parte superior de los robots (*Cubes* y *CheaViBots*) sean visibles en todo momento por la cámara. Para que lo anterior sea posible, la superficie sobre la cual serán depositados los robots debe encontrarse dentro del cono de visión de la cámara. Este cono de visión se encuentra en las especificaciones técnicas de la cámara y se presenta en la Figura 4.15. A partir de esta figura y considerando una superficie de experimentación de 1 x 1 $[m^2]$, se determinó que la distancia entre la plataforma para las mediciones y la cámara debe ser de al menos 1311,4 [mm]. Para efectos prácticos se ocupará una distancia de 1500 [mm], es decir, 1,5 [m].

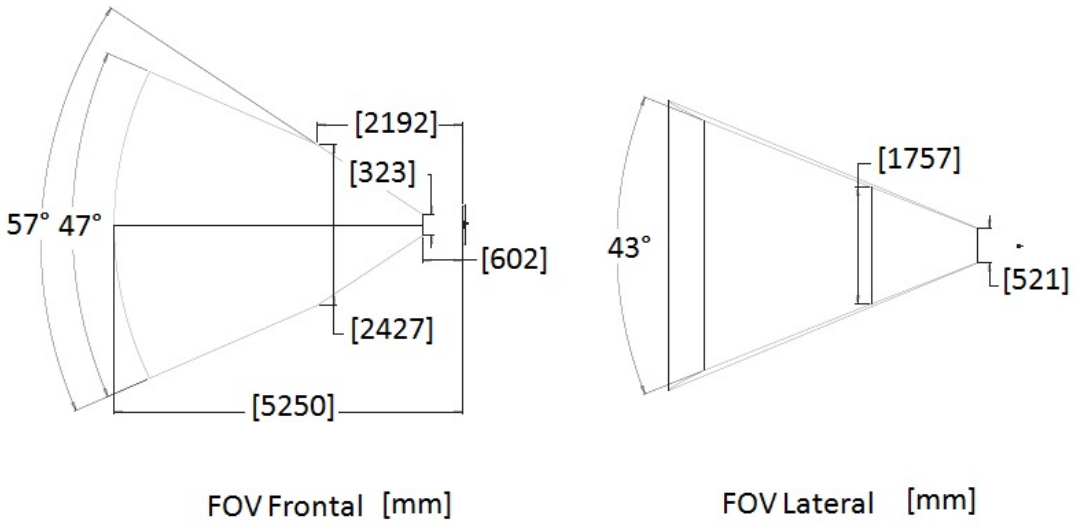

Figura 4.15: Campo de visión de la cámara *OptiTrack Trio™*. Fuente: Hoja de datos de la cámara proporcionada por el fabricante.

Tabla 4.3: Configuración de OptiTrack™

	Exposición (Exposure)	Umbral (Threshold)	Intensidad (Intensity)
Cámara 1	120	120	1
Cámara 2	235	238	8
Cámara 3	55	75	8

Por otro lado, la superficide donde interacturán los robots, debe cumplir con una serie de requisitos:

- Superficie libre para experimentación de al menos 1 x 1 [m^2].
- La rugosidad debe ser despreciable para facilitar el desplazamiento de los robots.
- La cara de la superficie que enfrenta los lentes de la cámara debe ser opaca, no reflectante, para garantiza que la cámara no detecte falsos marcadores.
- Debe ser nivelable, para garantizar que las experiencias no se vean alteradas por la inclinación de la misma.

En función de lo anterior, se optó por una superficie de madera trupan de 1 x 1 [m^2], la cara expuesta a la cámara fue pintada con aerosol negro opaco y finalmente se agregaron cuatro pernos de precisión que permitían nivelar la superficie antes de realizar cada experiencia.

Configuración de la cámara

Dadas las condiciones iluminación presentes en el laboratorio, fue necesario modificar los parámetros de percepción de las cámaras tal y como se muestra en la tabla 4.3.

4.2. Funciones de cálculo

Debido a que el formato en el cual se entregan los datos de las trayectorias por el programa *Motive* no permite realizar un análisis directo de los mismos, se implementó una rutina de trabajo que permitiera cumplir con esta tarea. En primer lugar, el archivo entregado por el programa *Motive*, debió ser editado mediante el programa *Microsoft Excel™*. Con este programa se eliminaron las columnas donde se identifican los marcadores y el texto descriptivo previo, donde se explica la forma en la cual se encuentra dispuesta la información dentro del archivo. Posteriormente, el archivo es respaldado en formato "Excel Worksheet", para luego importarlo al programa *Matlab™*, con el cual se realizó el análisis de los datos. El mismo consistió en el estudio de las propiedades cinemáticas de los robots, mediante una serie de funciones que se describirán a continuación.

4.2.1. Funciones de análisis en *Matlab™*

El primer paso para analizar los datos obtenidos correspondió a la eliminación de los espacios en blanco entregados por el programa *Motive™*. Dichos espacios se producen debido a la no identificación momentánea del marcador que está siendo registrado por la cámara.

Función para eliminar los espacios en blanco Esta función lleva por nombre "completar.m" y tiene por objetivo completar los espacios en blanco por interpolaciones lineales. Para esto, recibe como parámetro una matriz y entrega una matriz con las mismas dimensiones que la original, pero sin ceros entre datos relevantes. Lo anterior se realiza al identificar el primer y último término de la sección de valores nulos y posteriormente completar dicha sección con la interpolación lineal realizada entre los valores no nulos que limitan la sección sin valores. Los casos límites corresponden al comienzo y término de una serie de datos con valores nulos. En el primer caso, se reemplazan los valores al inicio por el primer término no nulo de la serie de datos y el segundo caso es eliminado en la etapa de edición mediante el programa *Microsoft Excel™*.

Una vez que se cuenta con la matriz resultado del proceso anterior, se procede a reagrupar los datos en un nuevo archivo, a fin de facilitar el posterior trabajo de análisis.

Función para ordenar los datos El nombre de esta función corresponde a "matricear.m" y tal como se menciona en el párrafo anterior, tiene por objetivo reagrupar los datos para su posterior análisis. Los parámetros que recibe esta función pueden ser: dos, cuatro o seis columnas, dependiendo del tipo de medición realizada. Cada pareja de columnas corresponde a los pares ordenados (abcisas y ordenadas) de un marcador (robot).

Se comienza con la descripción de los pasos realizados al recibir 2 columnas. En primer lugar, se crea una columna de tiempo en segundos, mientras los datos en las columnas que contienen desplazamientos, son convertidos a centímetros. Posteriormente se llama a una función auxiliar de nombre "calcAngDesp.m" que recibe como parámetro un matriz de dos columnas y entrega como resultado una matriz de 4 columnas con un número de filas igual al de las columnas entregadas como parámetro. Esta matriz resultado, tiene en sus primeras columnas los desplazamientos en las abscisas y ordenadas, en ángulo de desplazamiento en la tercera columna y la rapidez en la cuarta. El detalle de las operaciones realizadas por esta función se presentan en el párrafo Función calcAngDesp.

Finalmente, la matriz resultado entregada por la función "matricear.m", para el caso de recibir dos columnas como parámetro, contiene las siguientes columnas: tiempo de muestreo, abscisas, ordenadas, desplazamiento en las abscisas, desplazamiento en las ordenadas y el ángulo de desplazamiento.

En caso de entregar cuatro columnas como parámetro, se trata de una simulación computacional. En función de lo anterior, los pasos que se realizan son los siguientes. Se crea la columna con el tiempo de muestreo y se convierten los datos de las columnas entregadas a centímetros. Posteriormente, se llama a una función auxiliar de nombre "calcAngDespSimula-

cion", que recibe una matriz de dos columnas como parámetros y entrega como resultado una matriz de tres columnas, donde la primera corresponde a los desplazamientos en las abscisas, la segunda al desplazamiento en las ordenadas y la tercera al ángulo de desplazamiento.

En el paso siguiente, se llama a una segunda función auxiliar de nombre "calcAngMarcSimulacion". A esta función se le entregan las columnas correspondientes a los pares ordenados de la posición de dos puntos (cuatro columnas con datos en total). Como resultado, esta función entrega una matriz de tres columnas, donde las dos primeras corresponden a las diferencias de posición respecto a la orientación y la tercera al ángulo de orientación. Los resultados entregados por ambas funciones auxiliares, son agregados en una única matriz que corresponde al resultado de la función "matricear". El orden de las columnas dentro de la matriz resultado es: tiempo, abscisas y ordenadas del primer punto, diferencias de desplazamiento en abscisas y ordenadas, ángulo de desplazamiento, abscisas y ordenadas del segundo punto, diferencias de desplazamiento en la orientación para abscisas y ordenadas y finalmente, el ángulo de orientación. El detalle de estas funciones se encuentra en los párrafos **??** y calcAngMarcSimulacion.

Finalmente, el caso donde se entregan seis columnas como parámetros, corresponde a una plataforma robótica dotada de tres marcadores, por lo que se ejecuta la siguiente serie de comandos. Se crea la columna correspondiente al tiempo de muestreo en segundos y se convierten los datos de posición a centímetros. Posteriormente llama a la función "CalcAngDesp", mencionada anteriormente, a la cual se le entregan como parámetros las dos primeras columnas. Como se mencionó anteriormente, las columnas resultado son: la diferencia de desplazamiento en las abscisas, la diferencia de desplazamiento en las ordenadas y el ángulo de desplazamieto.

Posteriormente, se llama a la función "calcAngMarc", a la cual se le entregan como parámetros las columnas correspondientes a los pares ordenados de la posición del segundo y tercer punto, obteniéndose como resultado del proceso anterior, tres columnas, la diferencia de las coordenadas en las abscisas, diferencia en las coordenadas de las ordenadas y el ángulo de orientación del robot.

La matriz que se obtiene como resultado para este caso, contiene como primera columna el tiempo en segundos, los pares ordenados (abscisas y ordenadas) del primer marcador, la diferencia en las abscisas y ordenadas para el desplazamiento, el ángulo de desplazamiento, los pares ordenados para el segundo y tercer marcador, la diferencia en las coordenadas para ambos ejes y el ángulo de orientación.

Función calcAngDesp En esta función se realizan tres cálculos fundamentales para concretar el análisis de los robots. Para esto recibe como parámetros dos vectores de dimensión $nx1$ y entrega como resultado un matriz de cuatro columnas ($nx4$). Los vectores columna que entregados como parámetros corresponden a las abscisas y ordenadas obtenidas mediante el programa *Motive™*.

El primer cálculo realizado por esta función, corresponde a las diferencias móviles entre los datos obtenidos, que son entregadas en las dos primeras columnas de la matriz resultado. La obtención de estas diferencias se obtiene de acuerdo al diagrama presentado en la Figura

4.16. En esta Figura se observan dos pasos consecutivos de la rutina de cálculo, a partir de un punto genérico $r(t-\varepsilon)$. La primera medición corresponde a la diferencia de posición entre el punto antes citado $(r(t-\varepsilon))$ y un punto posterior desfasado en un tiempo arbitrario. En el ejemplo, el desfase corresponde a 2 veces ε, por lo que la diferencia se obtiene entre los puntos $r(t+\varepsilon)$ y $r(t-\varepsilon)$. La segunda medición, correspondiente al tiempo $t+1$, nuevamente corresponde a la diferencia entre dos posiciones distanciadas en 2 veces ε. Este proceso se realiza hasta completar la serie de datos disponibles.

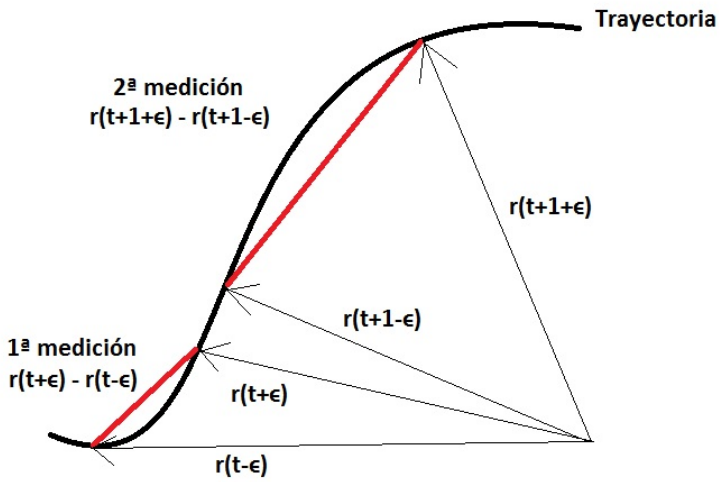

Figura 4.16: Esquema para el cálculo del desplazamiento.

El cálculo del ángulo de orientación con respecto al origen del sistema de coordenadas de la cámara, se obtiene mediante la función "angle" provista por el programa *Matlab™*. A esta función se le entrega como parámetro un número complejo y esta retorna el ángulo correspondiente. En este caso, se le entregaron las coordenadas de la posición del robot en formato complejo. Los ángulos obtenidos a partir de esta función se entregan en el tercer vector columna de la matriz resultado.

Finalmente, el cálculo de la rapidez se obtiene como el cuociente entre la distancia recorrida por el robot, a intervalos regulares y el tiempo empleado en recorrer dicha distancia. El resultado es entregado en la cuarta columna de la matriz resultado.

CalcAngDespSimulacion Esta función, orientada al tratamiento de los datos provenientes de la simulación computacional, recibe dos vectores columna y entrega una matriz de $nx4$. Las dos primeras columnas corresponden a las diferencias móviles para abscisas y ordenadas de acuerdo al procedimiento detallado en el párrafo anterior.

La tercera columna corresponde al ángulo de orientación obtenido mediante la siguiente la ecuación 4.1

$$\sphericalangle\theta(\mathrm{i}) = atan\left(\frac{\mathrm{d}Y(\mathrm{i})}{\mathrm{d}X(\mathrm{i})}\right) \tag{4.1}$$

La cuarta y última columna de la matriz resultado, corresponde a la rapidez, que corresponde al cuociente entre la distancia recorrida por el robot y el tiempo empleado en

recorrerla, al igual que en el caso de la función anterior.

calcAngMarcSimulacion Esta función al igual que la detallada en el párrafo anterior, tiene por función tratar los datos obtenidos a partir de las simulaciones computacionales realizadas. Recibe como parámetros cuatro vectores columna, a partir de los cuales, obtiene vectores diferencia para las abscisas y ordenadas junto con el ángulo de orientación del robot. El cálculo de las diferencias, se realiza mediante la sustracción, componente a componente, de los vectores que contienen los pares ordenados. Posteriormente, se obtiene el ángulo de orientación del robot por medio de la función "angle" proporcionada por *Matlab™*, la cual recibe como parámetros los componentes de los vectores diferencia calculados en la etapa previa.

4.3. Caracterización cinemática individual

Como punto de partida para entender el comportamiento de los robots vibracionales, se comenzó con la caracterización del comportamiento cinemático individual de las plataformas robóticas.

4.3.1. Caracterización cinemática individual unidireccional robots *Cube*

Cabe recordar que esta experiencia se realizó al ubicar uno de los robots *Cube*, sobre una superficie acanalada, cuyas dimensiones son las que se muestran en la Figura 4.17. Esta superficie fue construida utilizando impresión tridimensional. En la figura 4.18, se presenta

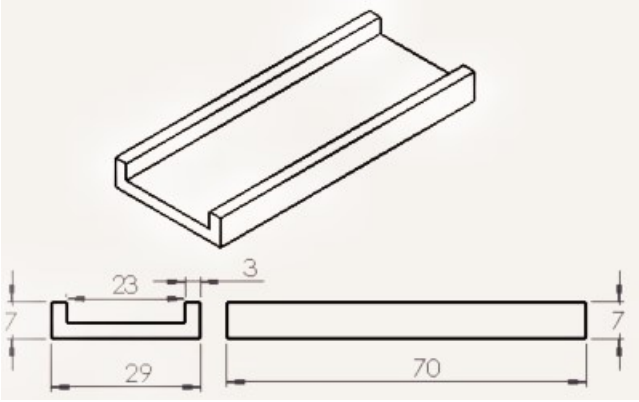

Figura 4.17: Perfil acanalado para medición de voltaje

el esquema global con el cual se caracterizó la velocidad lateral y frontal de los robots al trabajar con una fuente de voltaje constante de 3 [V].

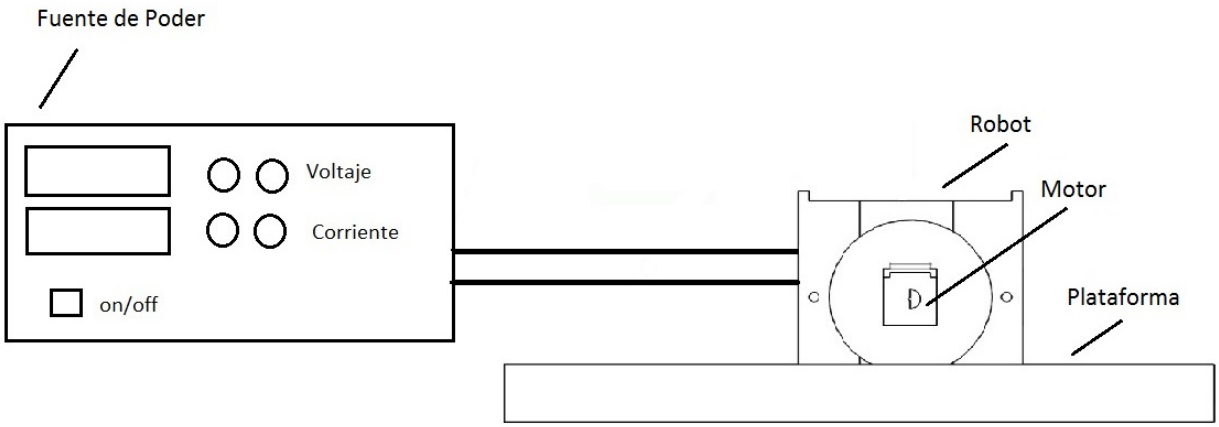

Figura 4.18: Esquema para caracterización de desplazamiento unidireccional.

Los resultados de la caracterización se presentan en la Figura 4.19. La velocidad frontal promedio es de 0.7211 $[\frac{mm}{seg}]$, mientras que la velocidad lateral promedio es de 0.62032 $[\frac{mm}{seg}]$.

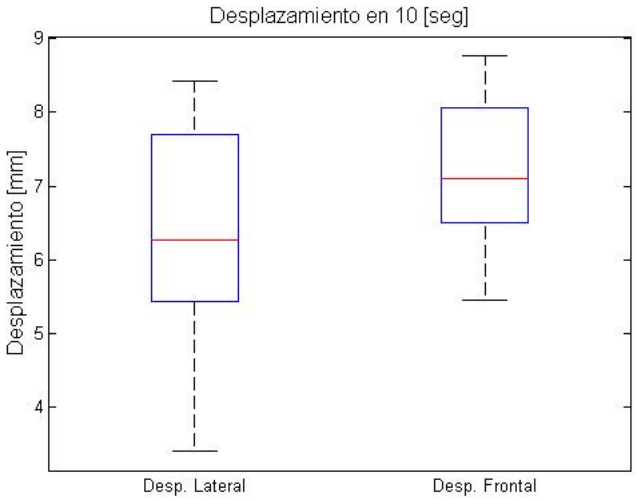

Figura 4.19: Desplazamientos unidireccional lateral y frontal para *Cubes*

Tiempo de Muestreo Una de las variables a ser consideradas antes de comenzar las experiencias de comportamiento colectivo corresponde al tiempo de muestreo, es decir, el tiempo que se dispone para realizar las mediciones antes de que el robot deje de funcionar debido a la descarga de la batería. Para determinar esta variable se realizó un montaje en una protoboard, que permitió medir el voltaje y corriente entregados por la batería durante el movimiento del robot. El montaje consistió en un porta-pilas comercial conectado a un multímetro configurado para medir corriente y otro multímetro configurado para medir voltaje.

Los resultados son los que se muestran en las Figuras 4.20 y 4.21. En el primer gráfico

4.20, se presenta el voltaje medido en Volts en el eje de las ordenadas, mientras que en el eje de las abscisas se presenta el tiempo en segundos. En color azul es posible observar los valores obtenidos de manera experimental, mientras que en color azul, se presenta una función de ajuste obtenido mediante la herramienta de ajuste exponencial disponible en el software *Matlab™*. La función de ajuste es la que se presenta en la ecuación 4.2.

$$v(t) = 2{,}109\mathrm{e}^{-0{,}2531t} + 1{,}201\mathrm{e}^{-0{,}001164} \tag{4.2}$$

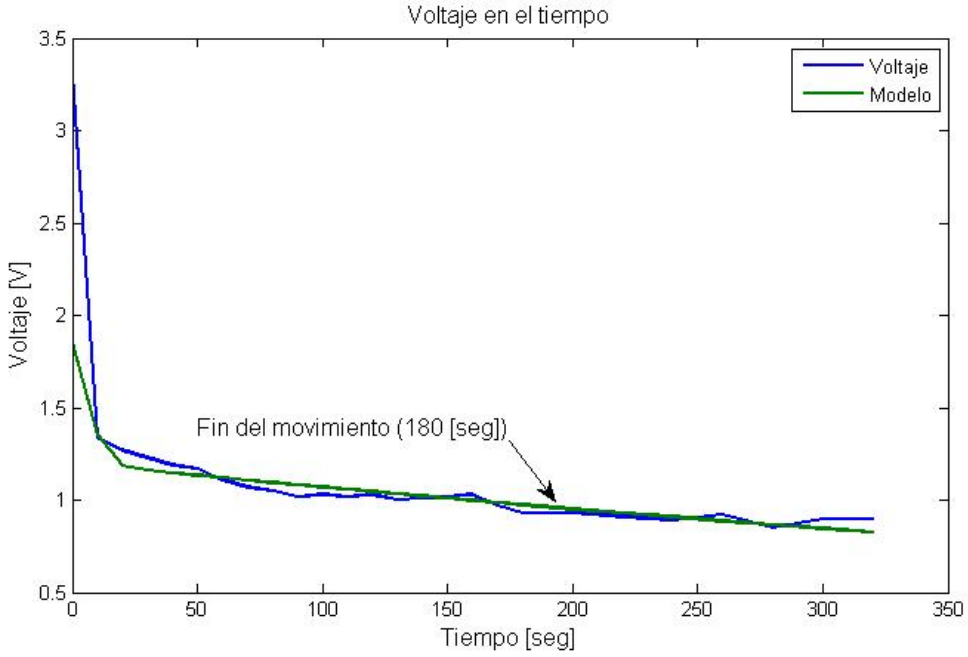

Figura 4.20: Disminución del voltaje en la batería

En la Figura 4.21, se presenta tanto la curva obtenida de forma empírica (en color azul) como el modelo correspondiente (en color verde), obtenido mediante la herramienta de ajuste disponible en el programa *Matlab™*. En el eje de las ordenadas se presenta la corriente eléctrica en miliAmperios, mientras que en el eje de las abscisas, se presenta el tiempo transcurrido. El modelo de ajuste sigue la ecuación 4.3

$$\mathrm{i}(t) = 0{,}02271\mathrm{e}^{-0{,}02457t} + 0{,}04738\mathrm{e}^{-0{,}001452t} \tag{4.3}$$

En función de los datos obtenidos durante esta experiencia, se determinó que el tiempo máximo de muestreo corresponde a 180 [seg], que corresponde al momento en que el robot deja de desplazarse. Sin embargo, para fines prácticos, los robots sólo fueron examinados por un tiempo máximo de 100 [seg], dado que una vez transcurrido ese período de tiempo, la rapidez con la que se desplazan los robots disminuye considerablemente.

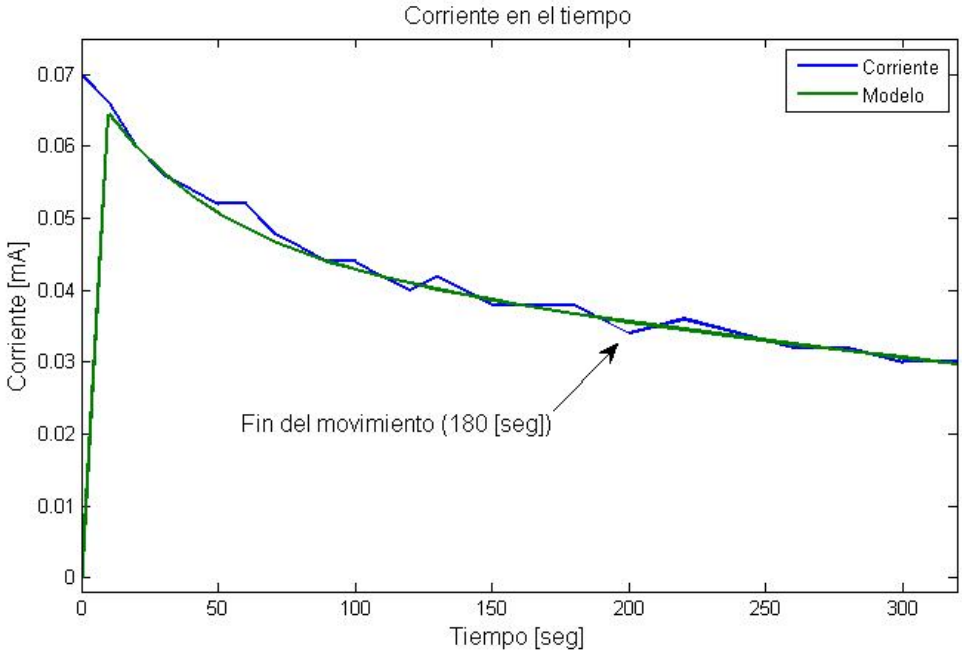

Figura 4.21: Disminución de la corriente en la batería

4.3.2. Caracterización cinemática individual libre robots *Cubes*

De acuerdo a lo detallado en el capítulo Metodología, se presentan los resultados obtenidos para las trayectorias de diez robots *Cube* en las Figuras 4.22 a 4.24.

Si bien las tomas fueron planificadas en 30 [seg], algunas de estas experiencias fueron prolongadas en su duración por las características presentadas por el individuo testeado.

Estas trayectorias se presentan en gráficos tridimensionales para permitir al lector comprender el movimiento realizado por los robots durante el período de tiempo que fueron testeados. En estos gráficos, tanto las ordenadas como las abcisas se encuentran en pixeles, dado que la conversión a unidades métricas fue realizada en etapas de caracterización posteriores y lo que se busca es dar una caracterización cualitativa del movimiento presentado. En el tercer eje se presenta el tiempo.

En color azul es posible observar las trayectorias descritas por los robots durante el tiempo que fueron estudiados, mientras que en color rojo, se encuentra una recta que representa el desplazamiento efectivo del robot. Esta recta de color rojo fue obtenida mediante el uso de la función "detrend" del programa *Matlab™* en su forma "detrend (x)", mediante la cual se elimina el mejor ajuste lineal del vector "x", para posteriormente realizar la sustracción de este valor a la variable original (x-detrend(x)).

Los robots, como era de esperar, de acuerdo a las diferencias de velocidades encontradas en la etapa de caracterización restringida, describen trayectorias tubulares en el tiempo. Las trayectorias rectilíneas se entienden como singularidades del individuo testeado.

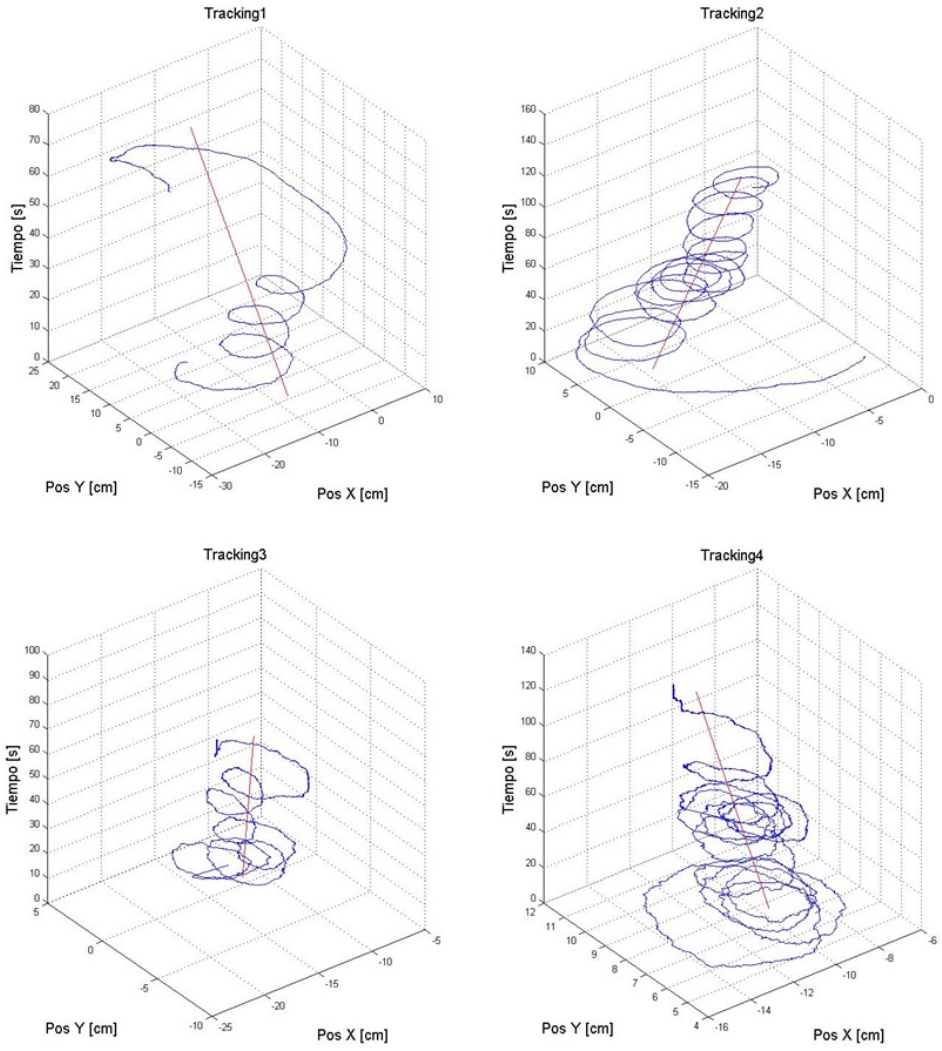

Figura 4.22: Trayectorias de los *Cubes* testeados.

Una vez que fueron graficadas las trayectorias, se procedió a utilizar las funciones descritas en la sección Funciones de cálculo. En primer lugar, se modificaron las tomas, de modo que todas tengan tantos elementos como la medición con la menor cantidad de datos.

Esto tiene por objetivo unificar el tamaño de las curvas a ser generadas para que las muestras sean comparables entre ellas y facilitar el cálculo de las medias a ser obtenidas en etapas posteriores.

A partir de las trayectorias, se determinaron los desplazamientos medios, que se muestran en la Figura 4.25. Esta gráfica se presenta en coordenadas log-log, donde en el eje de las abcisas bajo la letra τ representa los intervalos de tiempo considerados para calcular las diferencias de posición, que se encuentran en indicadas de acuerdo al eje de las abcisas.

Esta gráfica fue obtenida mediante la función "calcDeltaW" implementada en el programa *Matlab™*. La misma puede recibir como parámetros una matriz de dos columnas y un vector columna o dos vectores columna, entregando como resultado una matriz de dos columnas

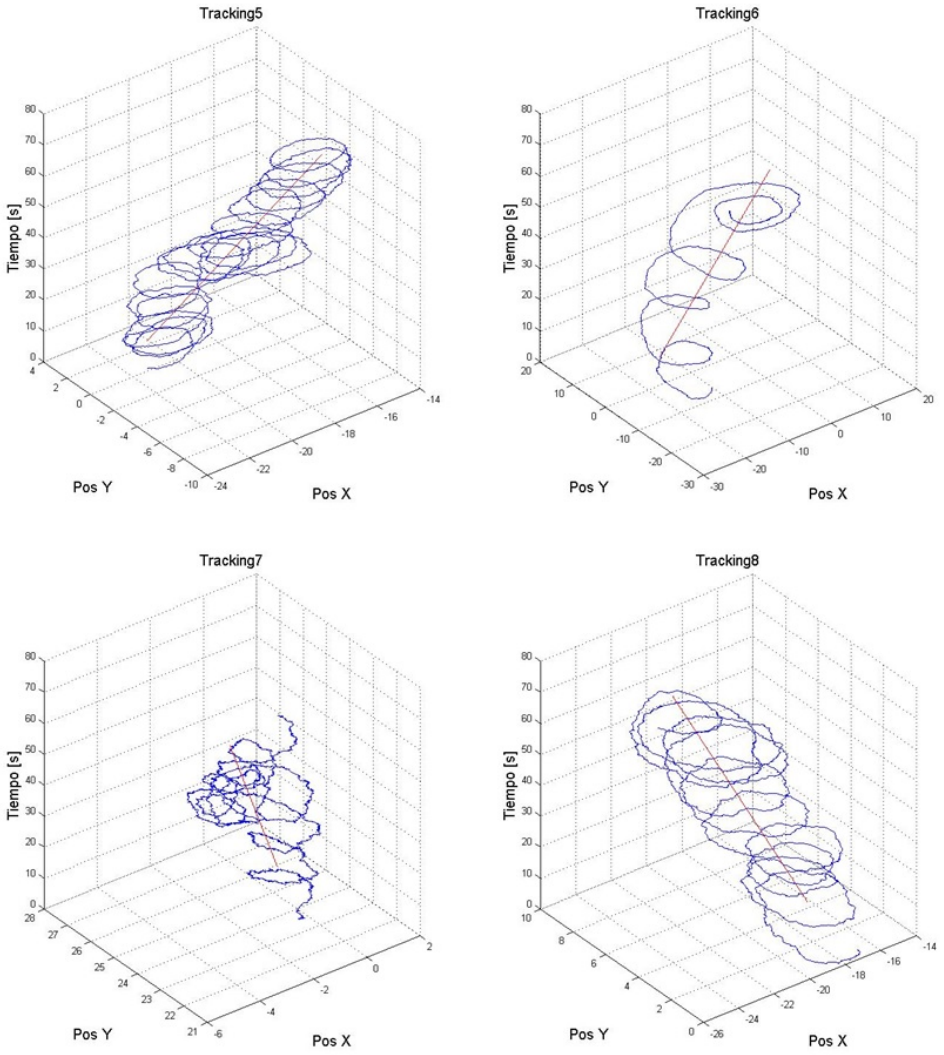

Figura 4.23: Trayectorias de los *Cubes* testeados.

que contienen el desplazamiento para cada uno de los puntos a contar del punto de origen y la serie de datos considerada ("taus"). El primer caso consiste en entregar una matriz que contiene las abscisas y ordenadas de un marcador para su posterior análisis (esto fue implementado para realizar un análisis inmediato de los datos obtenidos luego del tratamiento realizado en *Microsoft Excel*™). Siendo el programa el encargado de seleccionar las columnas correspondientes para continuar con el análisis.

En caso de recibir dos vectores columna, estos deben corresponder a las abscisas y ordenadas de un marcador. Este es el caso que se utiliza en el análisis final de los datos, entregando como parámetros a la función las columnas dos y tres de la matriz resultado obtenida mediante la función "matricear.m".

El primer paso dentro de la función "calcDeltaW" corresponde a la determinación del rango dentro del cual se definirá la variable independiente ("taus") contra la cual serán analizados los datos. Esta variable se define entre 1 y el número de datos dentro de la toma. El paso

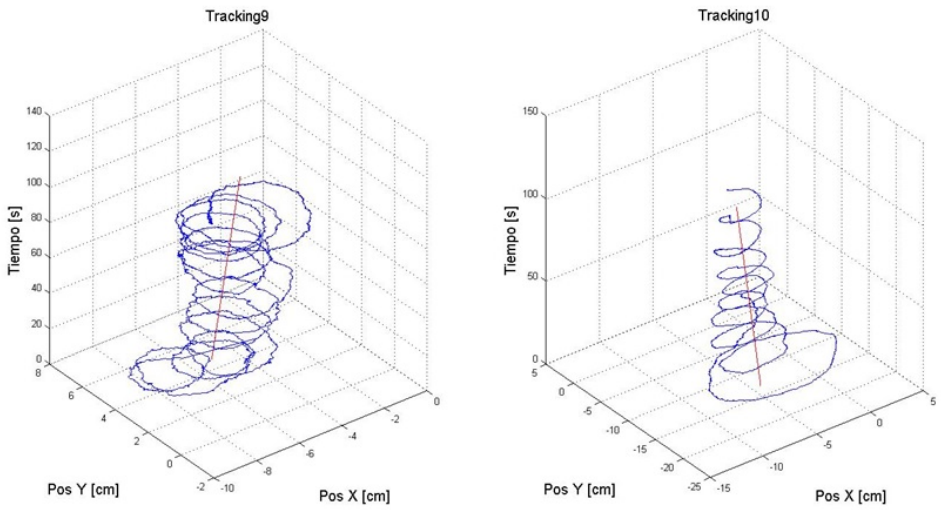

Figura 4.24: Trayectorias de los *Cubes* testeados.

corresponde a la unidad.

Acto seguido, se procede con el cálculo del desplazamiento $W(\tau)$, de acuerdo a la ecuación 4.4, que se encuentra implementada dentro de una función "for" para *tau*, donde esta variable comienza en 1 terminando cuando el valor de *tau* alcanza el número de datos contenidos en la muestra.

$$W(\tau) = \sqrt{x(\tau) - x(1)^2 + y(\tau) - y(1)^2} \tag{4.4}$$

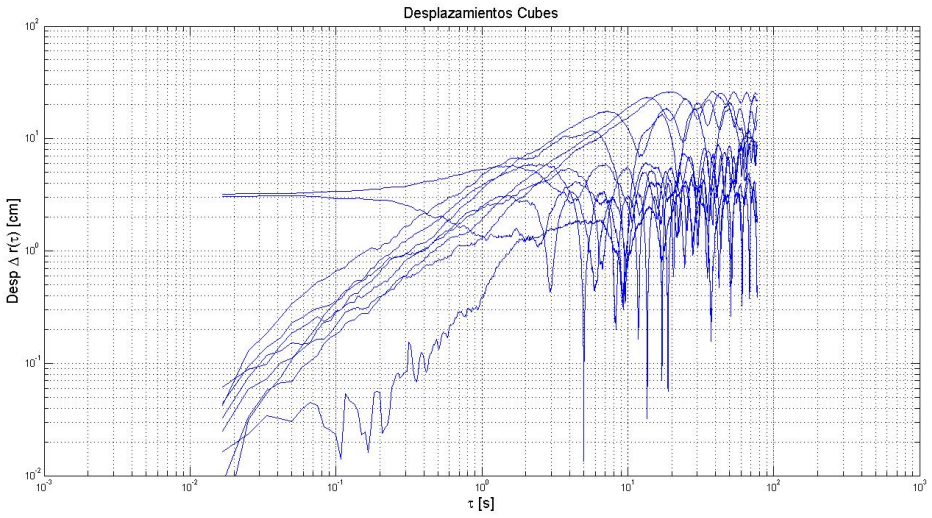

Figura 4.25: Desplazamientos medios *Cubes*

La determinación de desplazamientos preferentes se determinó por medio de las funciones de distribución de probabilidad o (PDF) para las coordenadas x e y. La función de distribución de probabilidad se presenta en la Figura 4.26.

Esta curva fue determinada por medio de la función "calculPDFAdqui", la que recibe como parámetros, un vector columna (sobre los cuales se calculará la PDF) y un número entero, que indica la cantidad de intervalos considerados para el análisis. El resultado de la misma corresponde a una matriz de $nx4$, donde serán relevantes para el presente análisis solamente la segunda y tercera columna, correspondientes a la PDF y a las abscisas correspondientes. El detalle de la implementación de esta función se encuentra en el Anexo B.

En la parte superior de esta figura, se presenta la función de distribución de probabilidad o PDF (por sus siglas en inglés), para el desplazamiento de acuerdo al eje X. Mientras que en la parte inferior, se observa la función de distribución de probabilidad para el desplazamiento en el eje Y.

En el eje de las abscisas se encuentra

Como primera observación relevante, se observa que para estos robots, el desplazamiento final en ambos ejes tiende a ser cero.

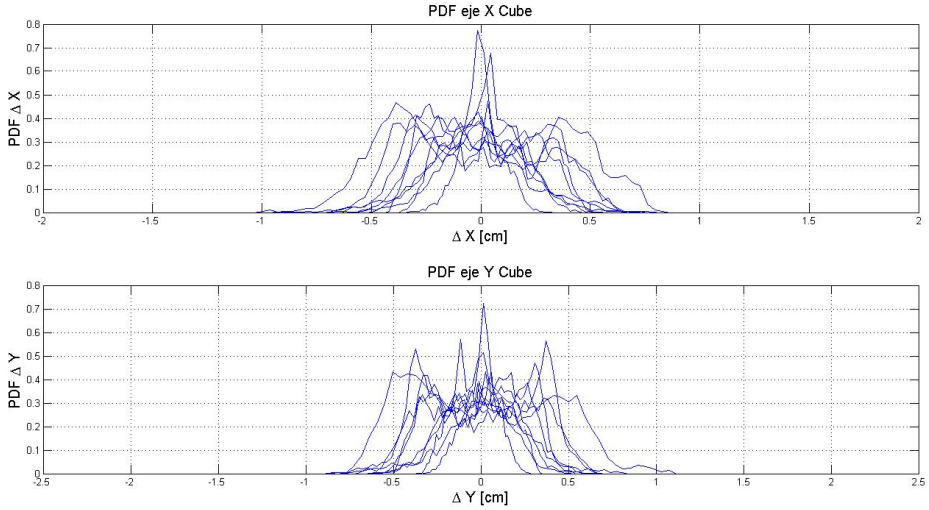

Figura 4.26: Función de distribución de probabilidad para x e y en *Cubes*

La Figura 4.27 presenta dos funciones de distribución de probabilidad. En la parte superior de la Figura, se presenta la función de distribución de probabilidad (PDF), para el ángulo de desplazamiento del cubo, es decir, para la trayectoria del robot. En el eje de las ordenadas se presenta la PDF, mientras que en el eje de las abscisas se presenta el ángulo medido en grados. Esta gráfica busca determinar si es que el robot tenderá a desplazarse en alguna dirección preferente.

Por otro lado, en la parte inferior de la imagen se presenta la PDF para el ángulo de orientación del robot, a saber, el ángulo con el cual el robot encara la trayectoria. Presentándose la PDF en el eje de las ordenadas, mientras que en el eje de las abscisas se presenta el ángulo medido en grados. Esta gráfica permite saber si es que el robot avanza encarando la trayectoria "hacia adelante", entendiendo que el frente del robot corresponde a la cara en la cual se encuentra la masa excéntrica, o si es que se desplaza encarando su trayectoria de otra manera.

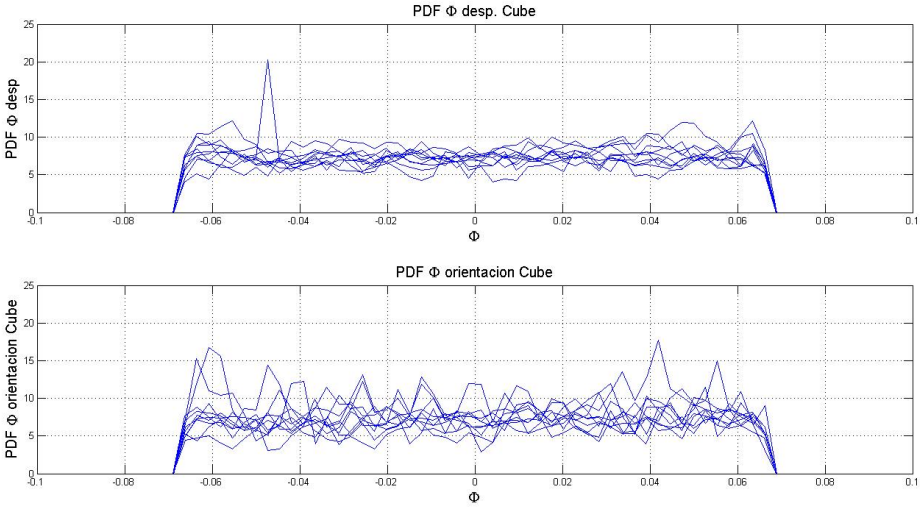

Figura 4.27: Función de distribución de probabilidad para angulo en *Cubes*

En la Figura 4.28, se presenta un gráfico de correlación entre el ángulo de desplazamiento, en color azul y el ángulo de orientación, en color rojo. En el eje de las ordenadas se presentan el ángulo de desplazamiento y el ángulo de orientación, mientras que en el eje de las abscisas se presenta la variación temporal. Los datos para el ángulo de desplazamiento corresponden a la media aritmética de los datos obtenidos a partir de la función "matricear" para los diez robots estudiados. Esto es, se tomaron las diez columnas obtenidas para el ángulo de desplazamiento y se calculo la media aritmética de las mismas. Del mismo modo, para el caso del ángulo de orientación, se tomaron las columnas correspondientes de la matriz resultado de la función "matricear", para posteriormente promediarlas e incluirlas en el gráfico.

A partir de este gráfico se desprende un grado de correlación directa entre la orientación del robot y la dirección en la cual se desplaza. Es decir, si bien no cuenta con un ángulo preferente, si es posible concluir que se mueve en la dirección hacia donde se orienta.

Finalmente, en la Figura 4.29, se presenta un gráfico resumen para la velocidad en ambos ejes (X e Y) y la rapidez de los robots. Se presentan superpuestas las curvas obtenidas para los 10 *Cubes* que fueron estudiados en esta ocasión. El objetivo detrás de este gráfico, es presentar visualmente, la banda sobre la cual varían estas dimensiones en esta plataforma.

4.3.3. Caracterización cinemática individual robots *CheaViBots*

En el caso de la plataforma robótica *CheaViBot*, se buscó determinar la capacidad de maniobrabilidad del robot, por lo que fueron sometidos a tres pruebas básicas. La primera de ellas consistió en mantener una trayectoria rectilínea a lo largo de una plataforma cuadrada cuyo lado media cuarenta centímetros. El análisis de los datos recopilados durante esta experiencia, fueron analizados mediante la función "`Rutina_Cheavibot_Lineal`". Los resultados de dicha experiencia se presentan en la Figura 4.30. A diferencia de los gráficos presentados para las trayectorias en *Cubes*, en este caso, la trayectoria se muestra mediante una vista

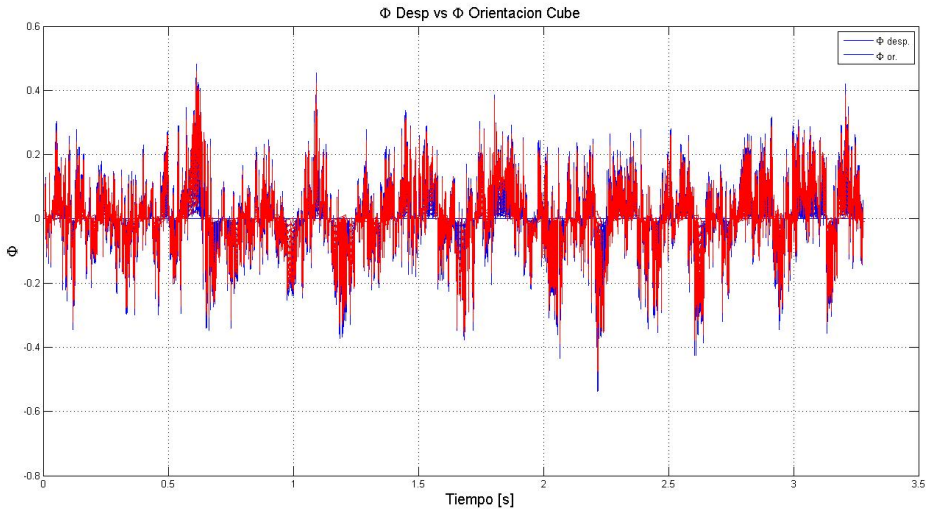

Figura 4.28: Correlación entre el ángulo de orientación y el ángulo de desplazamiento en *Cubes*

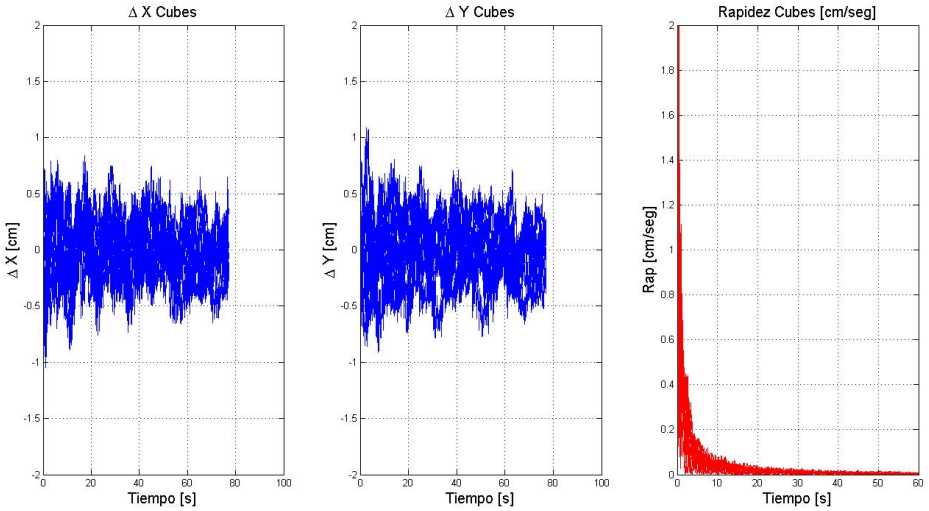

Figura 4.29: Velocidades y rápidez en *Cubes*

en alzado para facilitar la comprensión del lector. En el eje de las abscisas se presenta la coordenada X y en el eje de las ordendas se presenta la coordenada Y, ambas en unidades de centímetros. En color azul se pueden observar las trayectorias obtenidas durante las experiencias, mientras que en color rojo se presenta la trayectoria ideal a ser desarrollada por el robot.

A continuación se procede a presentar los gráficos con los que se determinaron las propiedades del movimiento antes presentado. En primer lugar, se realizó el ordeanamiento y conversión de unidades de los datos recopilados mediante la función "matricear". Posteriormente, los datos obtenidos en el proceso anterior fueron analizados mediante la función "calcDeltaW". Los resultados del proceso anterior son presentados en la Figura 4.31, en la cual los ejes coordenados son presentados en formato log-log. El eje de las ordenadas corresponde al

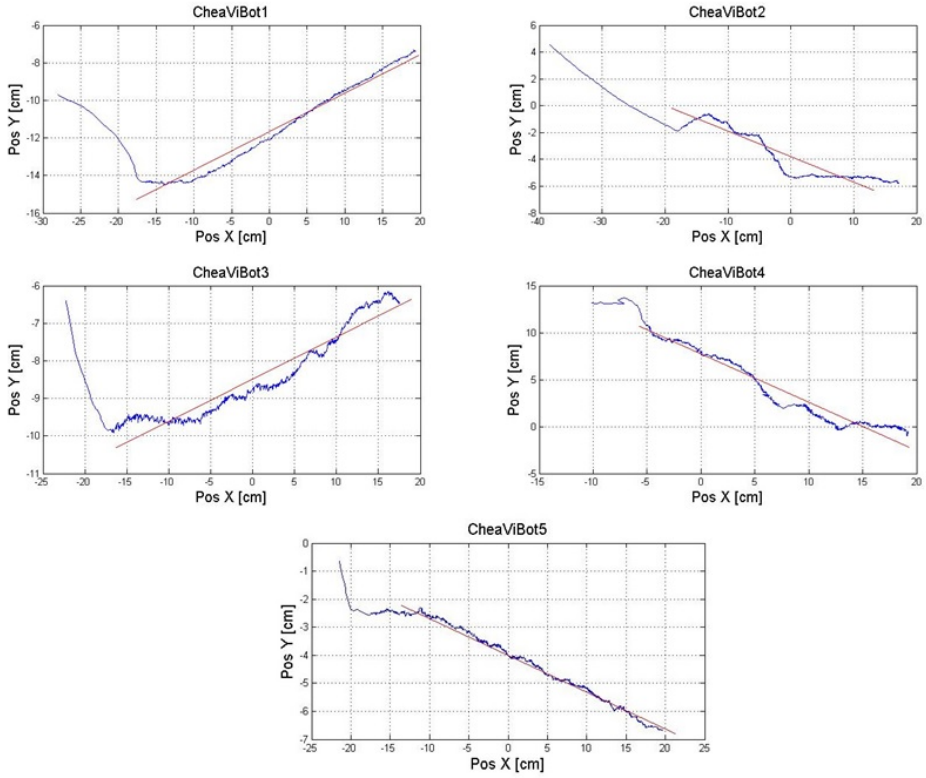

Figura 4.30: Trayectorias de los *CheViBots* testeados.

desplazamiento en centímetros y el eje de las ordenadas corresponde al tiempo transcurrido durante la experiencias en segundos.

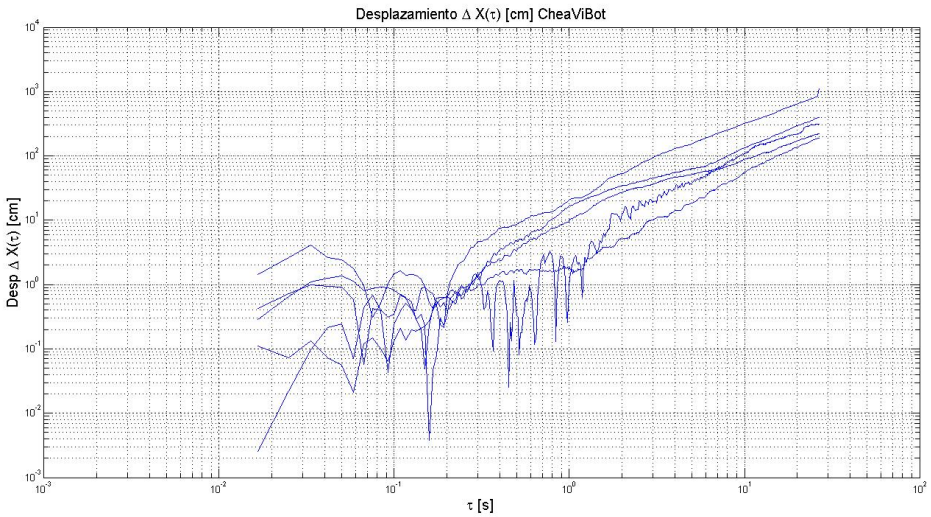

Figura 4.31: Desplazamientos medios en *CheaViBots*

Posteriormente, utilizando la función "`Rutina_Cheavibot_Lineal`" se procedió a promediar las series de datos obtenidas anteriormente. Dicho promedio se presenta en la Figura 4.32. Los ejes del gráfico se encuentran en formato log-log, donde el eje de las ordenadas

corresponde al desplazamiento medio en centímetros y el eje de las abscisas corresponde al tiempo medio empleado empleado en completar la experiencia. Además en la presente Figura, se presenta un ajuste exponencial del promedio de los datos obtenidos, el cual se presenta en la Ecuación 4.5

$$r(\tau) = 0{,}4\mathrm{e}^{1{,}16} \tag{4.5}$$

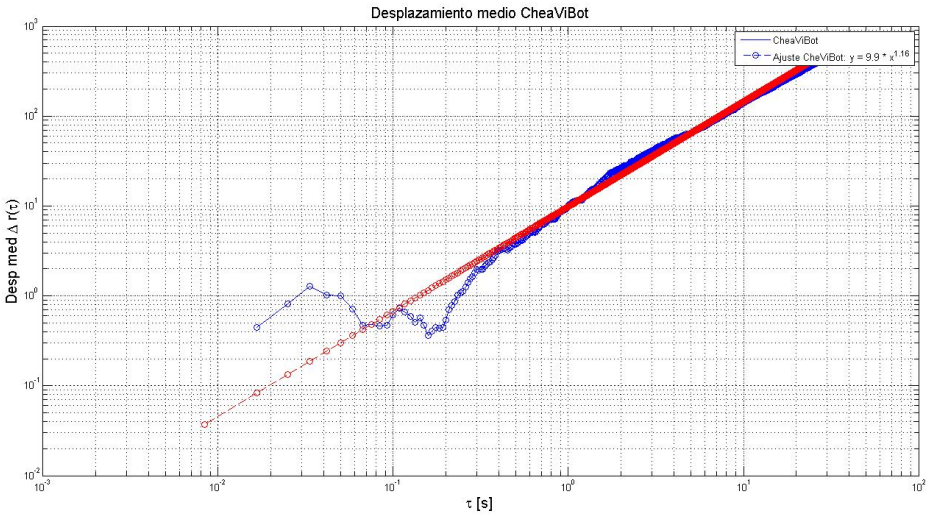

Figura 4.32: Desplazamiento medio en *CheaViBots*

Siguiendo con el análisis, se procede con la caracterización del desplazamiento medio por eje. Para esto, se calculó el promedio aritmético de las cuartas columnas de las matrices resultado de la función "matricear" aplicada a cada una de las muestras. El resultado del proceso anterior se presenta en la imagen de la izquierda dentro de la Figura 4.33. Del mismo modo, se calculó la media aritmética de las quintas columnas, presentando este resultado en la imagen central de la misma Figura. En ambos casos el eje de las abscisas representa el tiempo medio de empleado para desarrollar la experiencia, mientras que en el eje de las ordenadas se presenta el desplazamiento en el eje X en el caso de la imagen de la izquierda se presenta en la imagen central. En la imagen de la derecha, en color rojo, se presenta la evolución de la curva de rapidez media en centímetros por segundo.

Experiencias con cambio de dirección La primera prueba propuesta para determinar la capacidad del robot *CheaViBot* para desarrollar trayectorias con cambios de dirección, consistió en guiarlo a través de una trayectoria con forma de "L" a través de la misma plataforma cuadrada de 40 centímetros de lado utilizada para el desarrollo de las experiencias de desplazamiento lineal.

El resultado de esta experiencia se encuentra en la Figura 4.34. En la presente se observa la trayectoria realizada por el robots mientras era guiado por medio de una linterna en un ambiente sin luz. En el eje de las ordenadas se presenta el desplazamiento realizado a lo largo del eje de Y mientras que en eje de las ordenadas se presenta el desplazamiento realizado en el

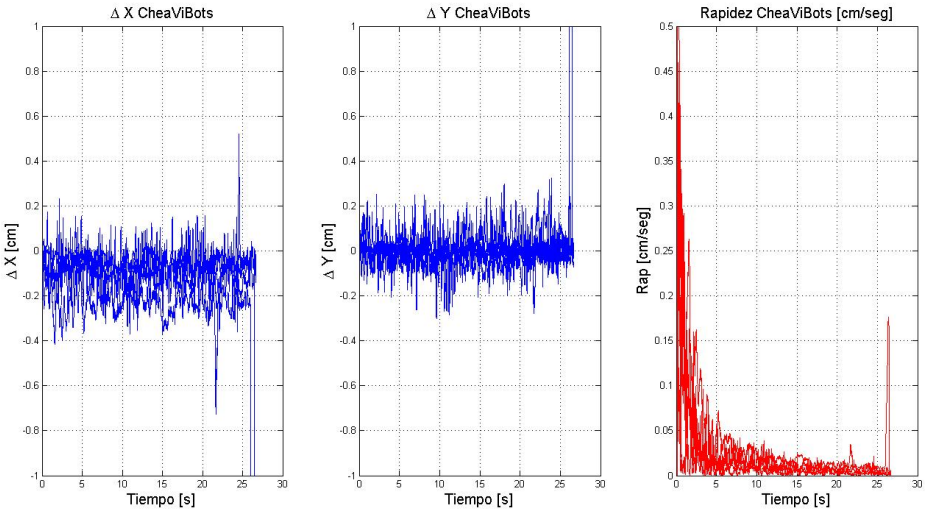

Figura 4.33: Desplazamientos y rapidez para trayectorias lineales en *CheaViBots*

eje X. Cada una de las caras de la trayectoria descrita tiene un largo de 25 [cm], sin embargo, la trayectoria conseguida bordea los 23[cm] por lado. El tiempo empleado para desarrollar esta experiencia fue de 79.55 [seg]. En función de lo anterior, la rapidez media es de 0.575 [cm/seg].

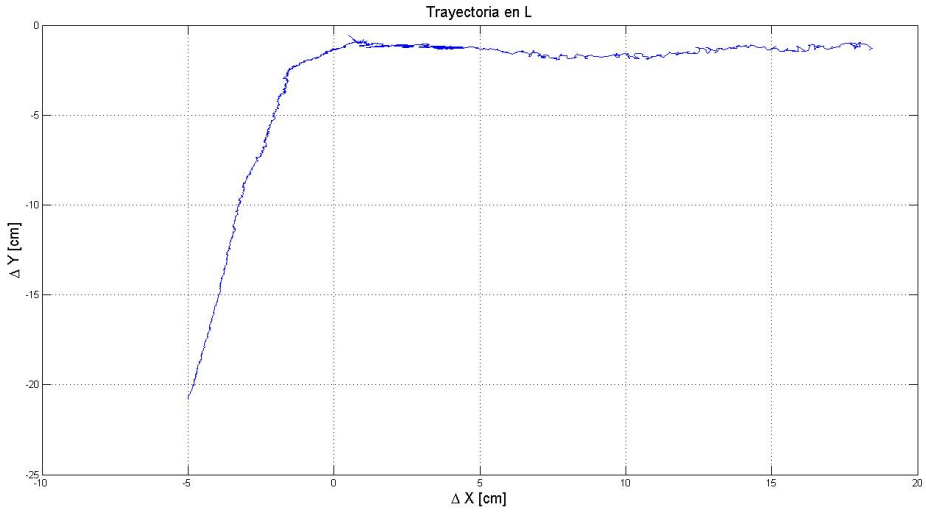

Figura 4.34: Trayectoria en forma de "L" en *CheViBots*

Los resultados para el desplazamiento realizado en cada uno de los ejes, así como la curva de rapidez media en el tiempo, son las que se presentan en la Figura 4.35

Además se realizó una segunda prueba, en donde se estudió la capacidad del robot de realizar una trayectoria curva. En este caso, la trayectoria seleccionada corresponde a una trayectoria en forma de "U". El resultado de esta experiencia, es el que se muestra en la Figura 4.36. Al igual que en el caso de la prueba anterior (trayectoria en forma de "L"), los ejes coordenados de la gráfica corresponden a las coordenadas descritas por el plano

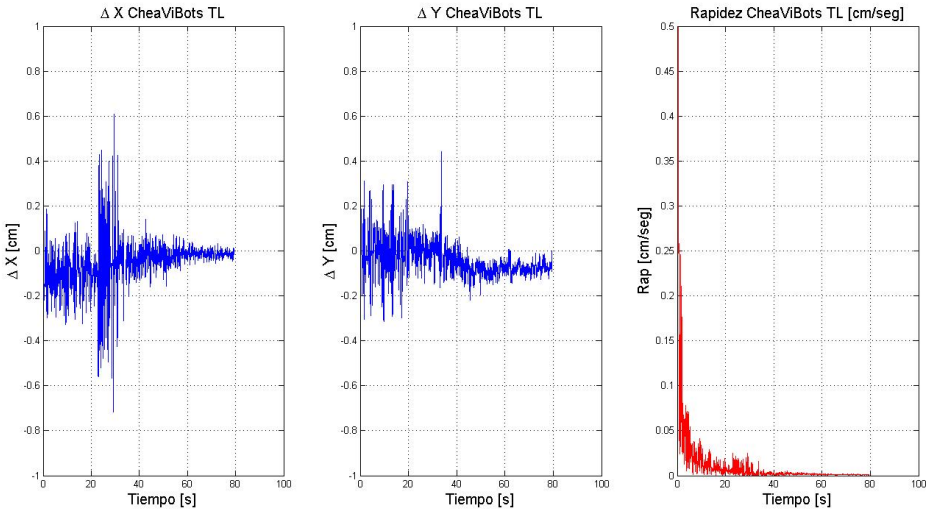

Figura 4.35: Desplazamiento en ejes coordenados y rapidez media en *CheViBots*, para trayectoria en forma de L.

descrito por la plataforma donde se realizó la experiencia. El tiempo empleado en desarrollar esta experiencia corresponde a 110.76 [seg]. A partir de esta segunda prueba, además se

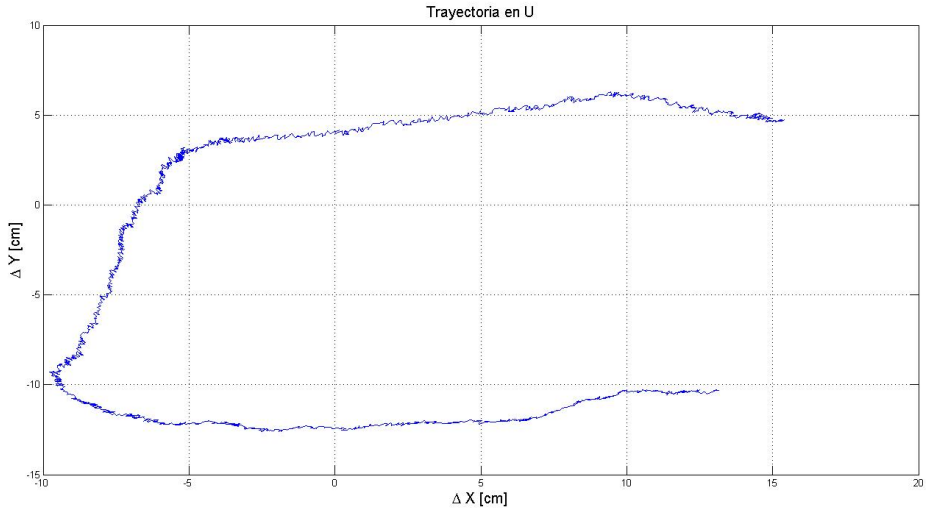

Figura 4.36: Trayectoria en forma de "U" en *CheViBots*

determinaron los desplazamientos en los ejes coordenados y la curva de rapidez media en centímetros por segundo. Estos gráficos fueron obtenidos siguiendo el mismo procedimiento desarrollado para la prueba anterior y los ejes coodenados se encuentran en centímetros en el caso de los desplazamientos y el tiempo en segundos.

Con las pruebas anteriormente descritas, se logró demostrar que el robot *CheaViBot*, es una plataforma capaz de seguir trayectorias predefinidas.

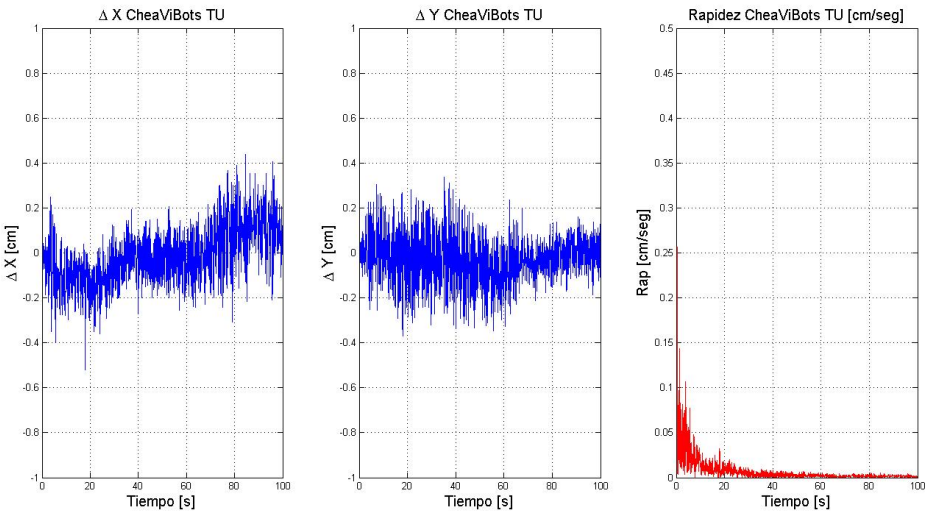

Figura 4.37: Desplazamiento en ejes coordenados y rapidez media en *CheViBots*, para trayectoria en forma de U.

4.4. Simulación Computacional

La simulación computacional se realizó por medio del software de programación C++, este programa ampliamente utilizado para diversas aplicaciones, cuenta con una librería, o conjunto de pequeños programas, que permite realizar simulaciones dinámicas realistas. Esta librería de nombre Open Dynamics Engine, ODE de ahora en adelante, permite configurar distintos tipos de elementos a través de la unión de "geometrías" más simples como son: esferas, cubos, cilindros de caras planas y cilindros de caras esféricas. Estas geometrías son utilizadas para configurar la detección de colisiones.

Por otro lado, cada una de estas geometrías define un "cuerpo", cuya función es realizar el seguimiento y configuración de fuerzas sobre la geometría. Este conjunto de "geometría" y "cuerpo" permiten la caracterización de un cuerpo con su respectiva masa, volumen, densidad y propiedades de interacción.

Un tercer elemento que toma importancia al momento de configurar un cuerpo vibratorio, son las articulaciones. En esta aplicación se utilizarán principalmente articulaciones de tipo bisagra, dado que dentro de la gama de articulaciones disponibles en la librería es la que más se asemeja al tipo de interacción que presenta el motor con la masa descentrada que produce la vibración.

4.4.1. Programación de los robots *Cube*

Los robots fueron simulados por medio de la interacción de dos cubos y un cilindro. Un cubo principal que reemplazó al cuerpo del robot, un cubo pequeño que reemplazó a la masa excéntrica que se encuentra adosada al motor y finalmente, un disco que se encuentra en la

cara opuesta del cubo a la cual se encuentra adosada la masa excéntrica. Este disco busca emular la distribución de masas del cubo original, al ser caracterizado con las propiedades de la pila que energiza el robot. Las conexiones entre estos elementos es la siguiente: El cubo principal y el cubo pequeño se encuentran unidos por medio de una interacción tipo bisagra, el cubo principal y el cilindro (pila), se encuentran conectados por una interacción fija, que no permite movimiento relativo entre estos elementos. Entre el cilindro y la masa pequeña, no existe interacción alguna.

Accionamiento de los robots *Cube* El movimiento de los robots dentro de la simulación, así como pasa en los modelos reales, se produce debido al movimiento de la masa pequeña en torno al eje de bisagra que la une al cubo principal. La frecuencia de oscilación de la masa pequeña emula la frecuencia con la que oscila la masa de los cubos reales. Esta frecuencia de oscilación fue caracterizada por medio de un banco de pruebas en el cual se medía tanto la intensidad de corriente como el voltaje de la pila.

Interacciones entre *Cubes* En una primera etapa, los robots serán simulados aislados, por lo que su única interacción será contra la superficie sobre la cual se encuentran apoyados. En una segunda etapa, cuando la simulación busque estudiar el comportamiento de los *Cubes* cuando se encuentran de forma agregada, a la interacción con la superficie se sumarán las colisiones entre ellos. En una tercera etapa, además de la interacción con la superficie que los contiene y las colisiones, se sumarán, en ciertos grupos de cubos, fuerzas de interacción entre los robots.

Implementación de la simulación

Por tratarse de una simulación en lenguaje de programación C++, la filosofía de programación sigue la lógica de la programación orientada a objetos, esto quiere decir, que la definición de los elementos resulta invisible a los ojos del programa principal que combina los distintos subprogramas en una plataforma común. Desde este punto de vista el acceso a los componentes de cada uno de estos subprogramas se realizará por medio de funciones especializadas, que serán capaces de entregar el valor o componente que se requiera.

Clase Cube La clase Cube se definió en dos archivos. El primero corresponde a Cube.h donde se realiza la declaración de las variables y funciones (métodos) pertenecientes a la clase. Mientras, el segundo archivo corresponde a Cube.cpp donde se implementan las funciones de la clase. El detalle de la implementación se encuentra disponible en el Anexo A. La clase Cube se compone de:

- Un constructor.
- Un destructor.
- Funciones para declarar y asignar valores a las variables de la clase.
- Función que permite dibujar las geometrías.

- Función que reproduce el movimiento de la masa excéntrica.
- Función que determina la variación angular de los robots.
- Funciónque exporta la posición del cubo a un archivo tipo texto.
- Función que construye el nombre del archivo donde será exportada la posición del robot.
- Función que exporta la variación angular presentada por cada robot.
- Función que construye el nombre del archivo donde será exportada la variación angular del robot.
- Función que emula la descarga de la batería.

A continuación se procede a detallar la forma en la cual se implementan las distintas funciones pertenecientes a la clase Cube.

El constructor El constructor inicializa dos cuerpos de la clase ode "bodyaz "bodyb", que corresponden a la estructura de soporte y la masa excéntrica adosada al motor. Estos cuerpos son asociados a dos geometrías cúbicas pertenecientes al arreglo "box". Con el fin de simular la distribución de masas de los robots reales, se incluye un tercer cuerpo, "bodyc", que representa la pila de botón que energiza el sistema. Este cuerpo es asociado a una geometría cilíndrica de nombre "Battery". La unión entre estas geometrías se logra mediante articulaciones definidas dentro de la clase ode, en particular, los cubos que representan el cuerpo del robot y la masa excéntrica se unen por una articulación de tipo bisagra y entre el cuerpo del robot y la pila, existe una articulación estática.

El destructor Debido a que la implementación de la clase es bastante simple, no se requiere un destructor que se encargue de la liberación de memoria asignada a elementos específicos, si no que basta con la declaración de la función pre-establecida perteneciente al lenguaje C++.

Funciones para declaración y extracción de variables (setters y getters) El objetivo de estas funciones es poder conocer el valor asignado a cada una de las variables pertenecientes a la clase, sin necesidad de llamar a la variable directamente, con el fin de evitar modificar de forma involuntaria el valor original asignado a dicha variable. La implementación de estas funciones se consigue mediante la asignación (variable = valor obtenido como parámetro por la función) y mediante el retorno del valor de la variable (return variable).

Función que permite dibujar las geometrías La función que permite dibujar las geometrías, lleva por nombre DrawC y recibe como parámetro un entero. La función tiene por objeto llamar a las funciones pertenecientes a la clase ode que se encargan de dibujar las geometrías pre-establecidas en la simulación. Dentro de esta función se asigna además, el color que tendrán los distintos elementos al momento de ser representados en la pantalla. El entero que recibe como parámetro tiene por objeto, asignar un color distintivo para los distintos objetos creados como clases heredadas de la clase Cube. Conforme avanza la complejidad de la simulación, esta función tendrá por objeto advertir al programador que los elementos se

encuentran en las posiciones que deben estar, pero no será llamada cuando se registren los datos experimentales, para acortar los períodos de espera.

Función que produce el movimiento de la masa excéntrica La función que se encarga de producir el movimiento de la masa excéntrica, lleva por nombre vibrate(). Esta función no requiere se le entregue parámetro alguno dado que se encuentra dentro del loop que produce la simulación y básicamente consiste en el aumento progresivo del ángulo mediante el cual se establece la posición de la masa pequeña con respecto al eje que la une a la masa grande.

Función que determina la variación angular de los robots Esta función lleva por nombre Calc_angC() y su función es entregar dos puntos de referencia para poder posteriormente calcular el ángulo del robot. El análisis del ángulo se lleva a cabo por medio de un programa implementado en *Matlab™*, cuya descripción se entrega en el capítulo correspondiente.

Función que exporta la posición del cubo a un archivo tipo texto Esta función lleva por nombre Stream_Pos_cube() y entrega los valores de las variables x_ e y_ que son asignadas como una copia de las dos primeras componentes del arreglo entregado por la función dBodyGetPosition() perteneciente a la clase ode.

Función que construye el nombre del archivo donde será exportada la posición del robot Esta función lleva por nombre build_name_pos(int bidx) y tal como se describe en el título, tiene por objetivo construir el nombre del archivo donde serán exportados los datos de cada robot. Toma como parámetro el índice del arreglo de elementos pertenecientes a la clase Cube y por lo tanto entrega nombres del estilo "Track robots n^0 X", donde X representa el índice del arreglo correspondiente.

Función que exporta la variación angular presentada por cada robot El nombre de esta función es Stream_ang_cube() y exporta, a un archivo tipo texto, los datos recopilados por la función Calc_angC().

Función que construye el nombre del archivo donde será exportada la variación angular del robot Al igual que la función encargada de la construcción del nombre del archivo que contendrá los datos correspondientes a la posición de cada robot, la función build_name_ang(int bidx) recibe como parámetro el índice del arreglo donde son definidos los elementos de la clase Cube incluidos dentro de la simulación.

Función que representa la descarga de la batería Esta función se basa en la relación existente entre potencia eléctrica y resultados mecánicos. Por lo tanto, la función

Download_Battery() toma valores para el torque y velocidad angular, definida en la función vibrate(), y le asigna los valores ajustados a la descarga de la pila obtenida de manera experimental.

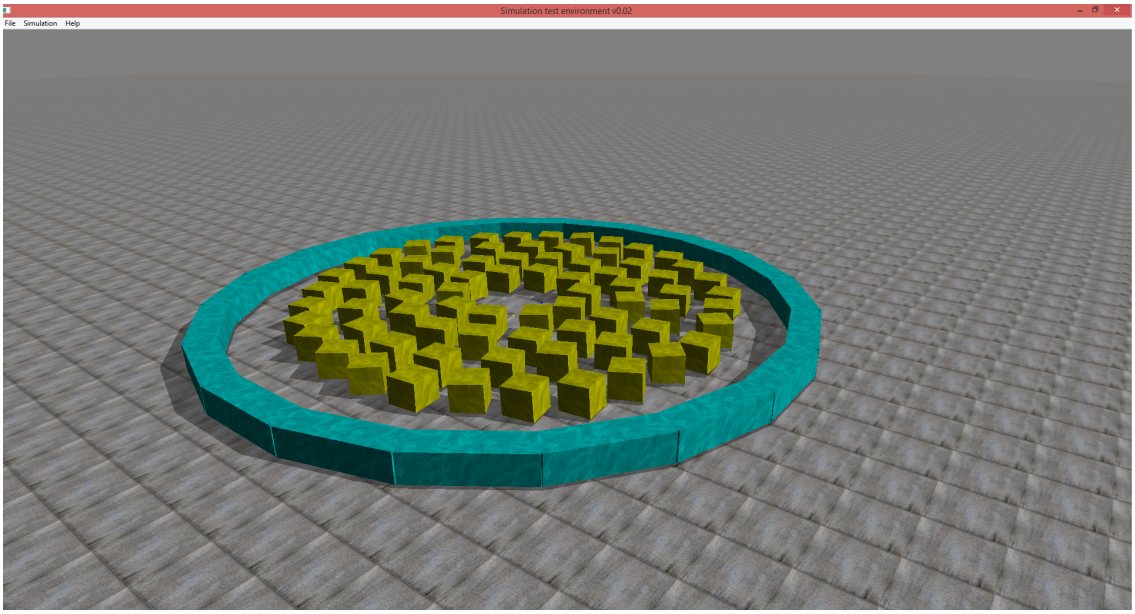

Figura 4.38: Captura de pantalla del modelo para los robots modulares.

4.4.2. Programación de los robots *CheaViBots*

La simulación de estos robots se realizó por medio de la interacción de 5 discos y dos cubos. El disco principal representó el cuerpo del robot y fue el cuerpo de referencia para la ubicación de los demás componentes. Tres cilindros emularon las patas del robot y un quinto cilindro fue ubicado en la posición que corresponde a la pila de botón. Los dos cubos al igual que en la simulación de los robots *Cube*, emulan las masas excéntricas adosadas a los ejes de los motores.

Las conexiones entre estos elementos fue la siguiente: El cilindro principal y el cilindro que representa la pila de botón se encuentran unidos por una articulación fija, que impide el movimiento relativo entre ellos. Los cilindros que emulan las patas del robot real, al igual que el cilindro que representa la bateria, se encuentran adheridos mediante articulaciones fijas. Finalmente, los cubos que representan las masa excéntricas de los motores, se encuentran unidos mediante una articulación de tipo bisagra con el disco principal. Entre las patas, la pila y las masas excéntricas no existe interacción, más que la conexión por medio del cilindro principal.

Accionamiento de los robots *CheaViBots* Al igual que en la simulación de los robots *Cube* descrita anteriormente, el movimiento en este caso se produce al modificar la posición relativa de las masas cúbicas que se encuentran adosadas al disco principal. La frecuencia

con la cual varía la posición de las masas es la misma que fue determinada para el caso de los *Cubes*.

Interacciones entre *CheaViBots* Para la primera etapa, de caracterización individual, los robots serán considerados como aislados, siendo su única interacción el contacto con la superficie que lo soporta. Durante la segunda etapa, de caracterización colectiva, se agregarán colisiones elásticas entre los robots.

Implementación de la simulación

Esta simulación a diferencia de la desarrollada para los robots *Cube*, busca explorar las condiciones que presenta la plataforma *CheaViBot* para el estudio del comportamiento colectivo. En función de lo anterior las funciones implementadas en el diseño de este programa, son orientadas al manejo externo, es decir, que sea el usuario a través de una consola el encargado de maniobrar con el robot. Lo anterior busca emular el comportamiento que tendría un sistema de inteligencia a bordo que gobernara el funcionamiento de los motores. Para estudiar la respuesta al control externo, se implementaron funciones que permiten exportar la posición de la plataforma en función del tiempo, con el fin de estudiar el tiempo de respuesta de estos robots.

Clase Drive_Cube La clase Drive_Cube se encuentra dividida en dos archivos. El primero corresponde a Drive_Cube.h donde se realizó la declaración de las variables (métodos) pertenecientes a la clase. El segundo archivo corresponde a Drive_Cube.cpp que es donde se implementaron las funciones de la clase. El detalle de la implementación se encuentra disponible en el Anexo A. Esta clase hereda las funciones desarrolladas en la clase Cube.cpp, por lo que su implementación solamente requiere incluir seis funciones:

- Un constructor genérico.
- Un constructor.
- Un destructor.
- Una función para dibujar el robot.
- Una función para activar el motor derecho.
- Una función para activar el motor izquierdo.

Constructor genérico Este constructor corresponde a un requisito de las clases heredadas, en este caso solamente fue declarado por concepto de compilación del programa.

Constructor Corresponde al constructor efectivo de la simulación, es decir, es la función que es llamada cuando se requiere utilizar uno de los objetos de la clase. Recibe como parámetros dos enteros que corresponden a las coordenadas, en los ejes X e Y, del centro del

disco principal, la ubicación de los demás componentes del robot quedan definidos en función de las uniones definidas entre ellos.

La construcción del robot se consigue por medio de arreglo de las clases "body", "cylinder" y "box". La definición de las patas del robot corresponde a los tres primeros elementos del arreglo "Cylinders", mientras que la batería corresponde al cuarto y el cuerpo del robot corresponde al quinto. El arreglo de la clase "box" se compone de dos elementos que corresponden a las masas excéntricas que produciran el movimiento del robot. Finalmente, cada una de las geometrías anteriores es asociada a uno de los elementos dentro del arreglo "body".

Función para activar el motor derecho Mientras esta función se encuentra activa, el ángulo de la masa derecha será incrementado de manera constante en $0{,}05^0$, produciendo la vibración del robot. La activación de esta función no requiere se le entreguen parámetros adicionales, pero si que mantenga presionada la flecha derecha del teclado.

Función para activar el motor izquierdo De manera análoga a lo que ocurre con la función que produce el movimiento de la masa de la derecha, esta función se mantendrá activa mientras se mantenga presionada la flecha izquierda del teclado.

4.5. Caracterización cinemática individual para modelos

En esta sección se presenta la caracterización cinemática individual del modelo computacional construido para escalar las experiencias colectivas realizadas con la plataforma *Cube*. Esta caracterización tiene por objetivo validar el modelo desarrollado por medio de la repetición de las pruebas y mediciones realizadas a los robots *Cube*.

4.5.1. Caracterización cinemática individual modelo *Cube*

En primer lugar se procedió a obtener las trayectorias individuales de diez robots simulados, de ahora en adelante "*Sim. Cube*". Para esto los robots fueron ubicado en una posición central de la base virtual configurada para la medición de los *Sim. Cube*. Mediante la función "`Stream_Pos_Cube()`", implementada en el programa que se encargó de la simulación, se exportaron las coordenadas de la posición del centro geométrico del cubo principal. En paralelo, mediante la función "`Calc_angC()`" se determinó la posición angular del robot simulado. Los resultados entregados por ambas funciones fueron exportados a un archivo de texto ("*.txt") separados por comas para faclitar su análisis posterior. Dicho análisis fue realizado por medio del programa *Matlab™* mediante la función "`Rutina_Cubes_Simulacion`". En primer lugar se realizó la conversión y ordenamiento de los datos obtenidos por medio de la función "matricear". Acto seguido se procedió a graficar las trayectorias, las cuales se presentan en las Figuras 4.39 a 4.41. Estas trayectorias corresponden a la segunda y tercera columna de la matriz resultado entregada por la función "matricear". En esta imagen se presenta en color

azul la trayectoria descrita por los *Sim. Cube*, mientras que en color rojo se presenta el desplazamiento. Esta magnitud, el desplazamiento, se obtuvo de la misma forma que se obtuvo para los robots *Cubes*.

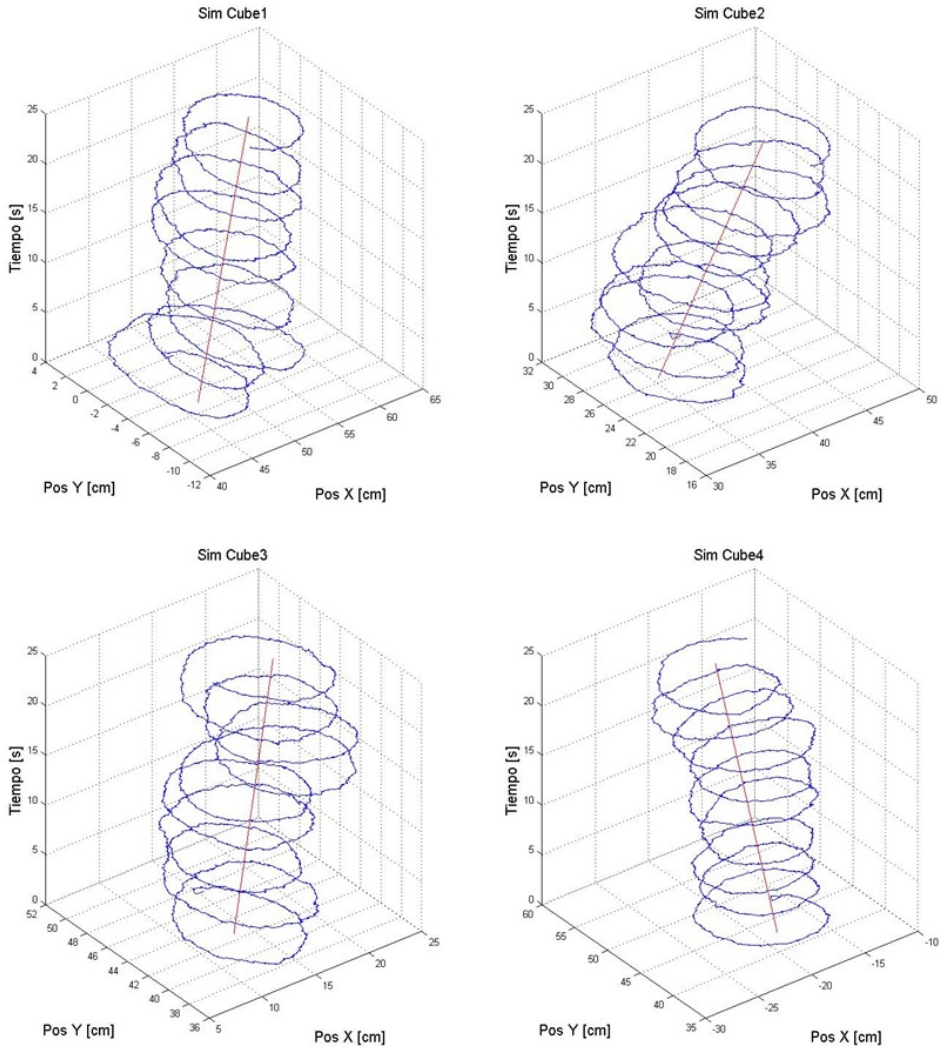

Figura 4.39: Trayectorias para los *Cubes* simulados.

Una vez obtenidas las trayectorias, mediante el programa "`calc_DeltaW`", se obtuvieron los desplazamientos móviles para cada una de las muestras. Lo anterior se consiguió al entregarle como parámetros la segunda y tercera columna de la matriz resultado entregada por la función "matricear". Dichas curvas se presentan en la Figura 4.42. Al igual que en el caso de los *Cubes*, estos resultados son presentados en formato log-log. En el eje de las ordenadas se presenta el desplazamiento móvil, mientras que en el eje de las abscisas, se presenta el tiempo de muestreo.

Una vez realizado el cálculo anterior, se procedió con la obtención de las funciones de distribución de probabilidad para el desplazamiento tanto en el eje X como en el eje Y. Estas funciones fueron obtenidas por medio de la función "calculoPDFadqui", entregándole como parámetros las columnas cuarta y quinta de la matriz resultado de la función "matricear".

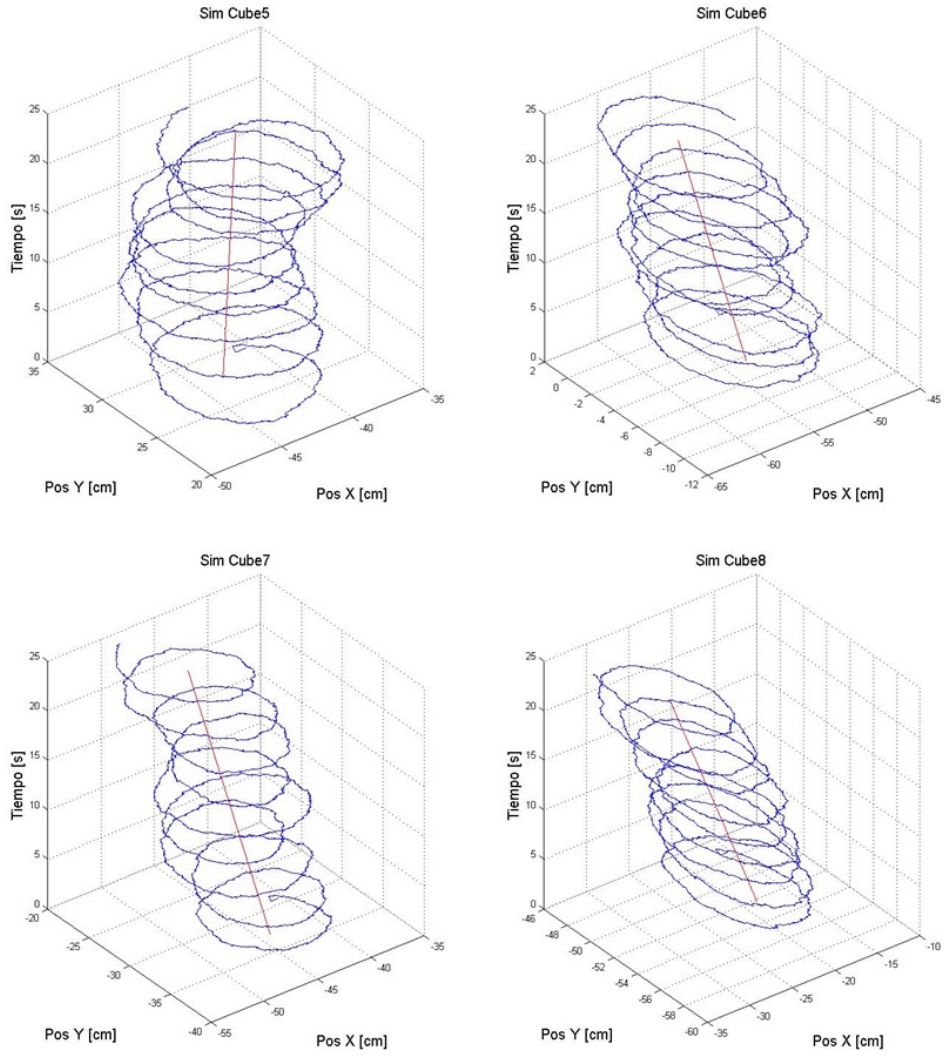

Figura 4.40: Trayectorias para los *Cubes* simulados.

Los resultados son presentados en la Figura 4.43. En dicha figura, se presenta en la imagen superior la función de distribución de probabilidad para el desplazamiento en el eje X, mientras que en la imagen inferior, se presenta la función de distribución de probabilidad para el eje Y. En el eje de las ordenadas se presenta la PDF propiamente tal, mientras que en el eje de las abscisas se presenta el desplazamiento en centímetros. En esta gráfica, se pueden observar dos puntas claramente marcadas para ambos desplazamientos en -0.5 [cm] y +0.5 [cm].

El siguiente paso en busca de la validación del modelo computacional para los robots *Cubes*, consistió en la obtención de la función de distribución de probabilidad para los ángulos de desplazamiento y orientación de los robots simulados. Esta distribución se obtuvo mediante la función "calculoPDFadqui", al entregarle como parámetro la sexta columna de la matriz resultado de la función "matricear" para el caso del ángulo de desplazamiento y la undécima columna, de la misma matriz, para el ángulo de orientación. Los resultados del proceso antes descrito son presentados en la Figura 4.44. En esta imagen se presenta en la parte superior la

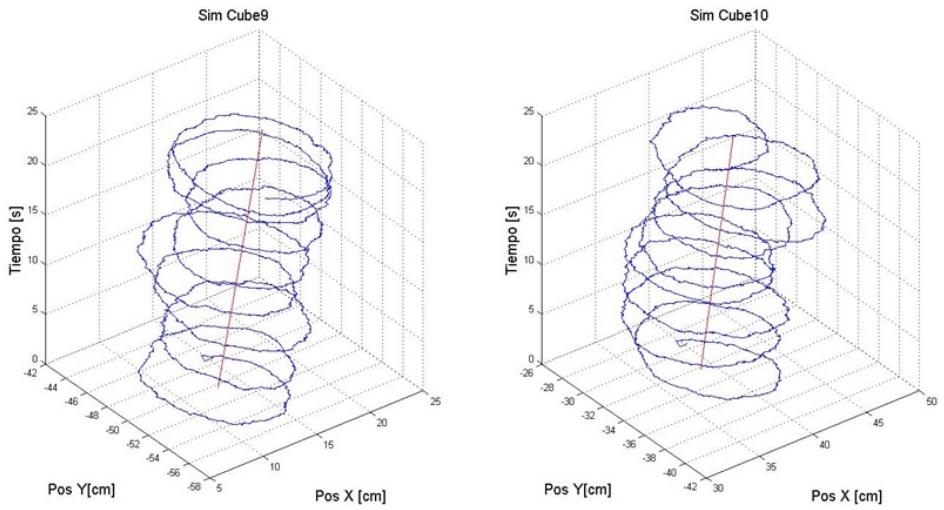

Figura 4.41: Trayectorias para los *Cubes* simulados.

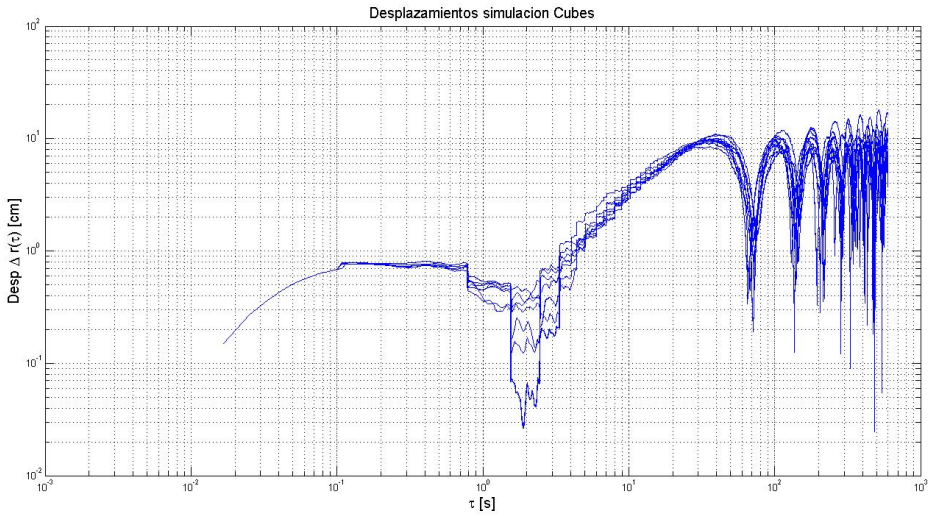

Figura 4.42: Desplazamiento medio para la simulación de *Cubes*.

PDF para el ángulo de desplazamiento, mientras que en la parte inferior se presenta el ángulo de orientación. En ambos casos el eje de las abscisas se encuentra en grados, mientras que en el eje de las ordenadas se presentan las funciones de probabilidad anteriormente descritas.

A partir de esta gráfica se puede observar que los robots, al igual que en el caso de los robot *Cubes*, no presentan orientaciones preferentes tanto para su desplazamiento como para su orientación.

A continuación se procedió a unificar la evolución de los ángulos de desplazamiento y orientación del robot simulado. Para esto, se calculó la media aritmética de los resultados obtenidos para ambos ángulos. El resultado de esta operación es el que se presenta en la Figura 4.45. Esta imagen cuenta en su eje de las ordenadas el gráfico en grados y en el eje de las abscisas el tiempo transcurrido durante la simulación.

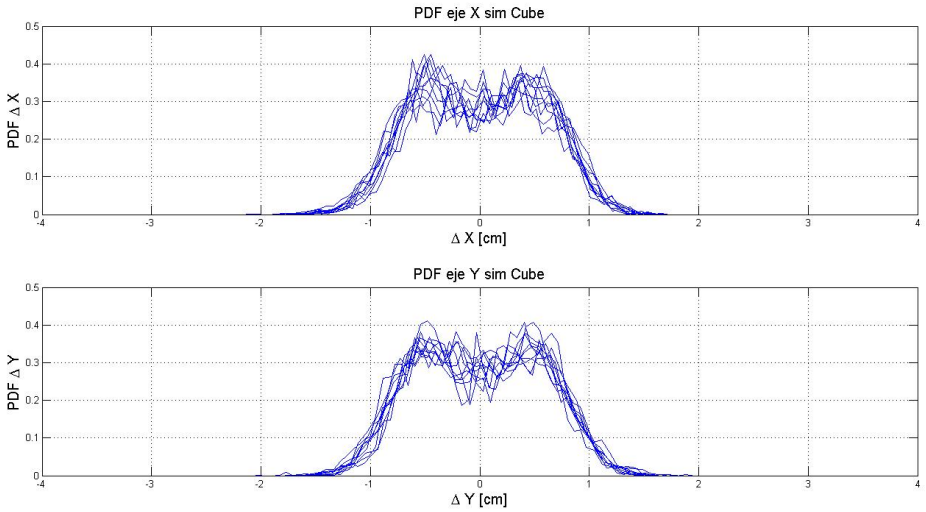

Figura 4.43: PDF desplazamiento para x e y en las simulaciones de *Cubes*.

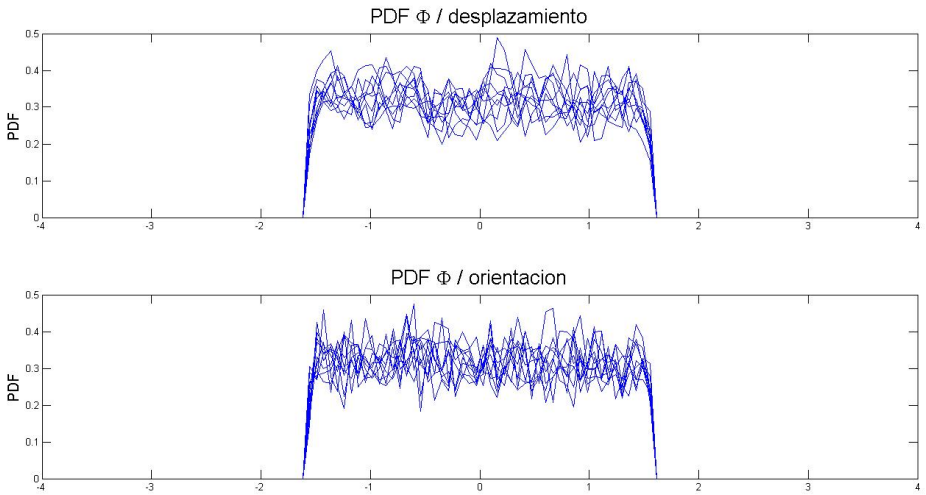

Figura 4.44: PDF ángulo desplazamiento y ángulo orientación en las simulaciones de *Cubes*.

Al superponer ambas gráficas, se puede observar una relación directa entre ambos ángulos, el análisis correspondiente será realizado en el capítulo de conclusiones.

Siguiendo con la etapa de validación de los resultados obtenidos para el comportamiento individual presentado por la simulación de los robots *Cube*, se procedió con la obtención de los desplazamientos en ambos ejes coordenados y la rapidez media. Para esto se graficaron las columnas número cuatro y cinco de las matrices resultado de la función matricear contratadas con el tiempo que se encuentra en la columna uno. Los resultados del proceso anterior generó las gráficas que se encuentran a la izquierda y el centro de la Figura 4.46. En las cuales los desplazamientos se encuentran en centímetros mientras que el tiempo se encuentra en centímetros.

La tercera gráfica presentada en la Figura 4.46, representa la curva de rapidez en el tiempo

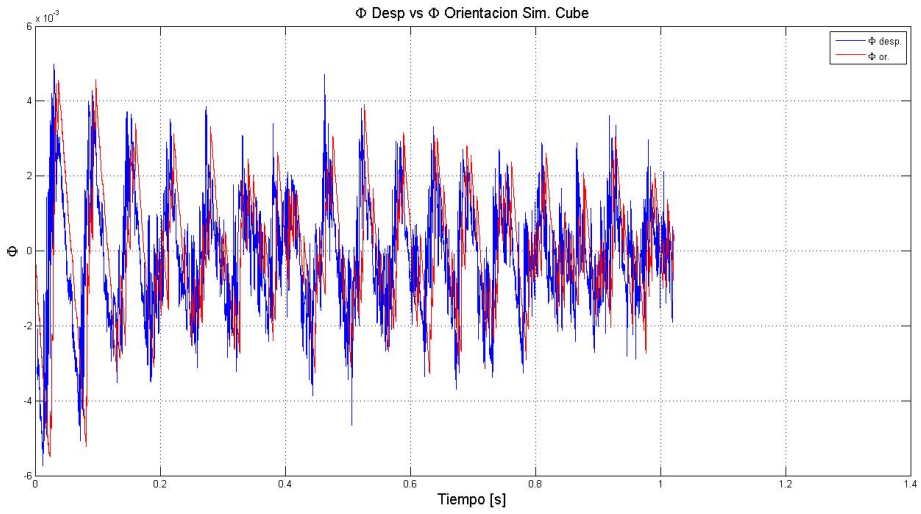

Figura 4.45: Ángulo de desplazamiento v/s ángulo de orientación.

para los robots estudiados. Esta curva fue obtenida a partir de las columnas cuatro y cinco anteriormente mencionadas a las cuales se les aplicó la ecuación 4.6.

$$v(t) = \frac{sqrtx(t)^2 + y(t)^2}{t} \tag{4.6}$$

En esta gráfica, la curva de rapidez se encuentra en centímetros por segundo, mientras que el tiempo se encuentra en segundos.

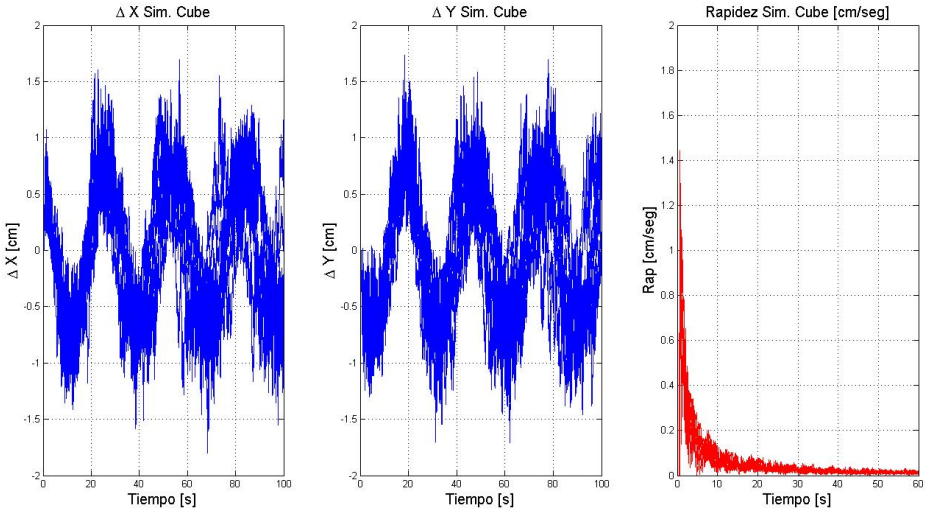

Figura 4.46: Velocidades y rapidez para la simulación de *Cubes*.

4.5.2. Caracterización cinemática individual modelo *CheaViBot*

Siguiendo con la lógica bajo la cual se estudió el comportamiento de los robots *CheaViBots*, las pruebas desarrolladas en utilizando el modelo computacional desarrollado, apuntaron a

obtener un comportamiento guiado por parte del usuario del programa. Para esto se implementaron una serie de funciones que permitieron controlar el apagado y encendido de los motores con los que cuenta la plataforma real al presionar las flechas izquierda y/o derecha que contiene el teclado del computador.

A continuación se procede a presentar los resultados para trayectorias rectilíneas, las cuales fueron conseguidas al mantener presionadas ambas teclas (flecha izquierda y derecha). En la Figura 4.47, se pueden observar tres de las trayectorias conseguidas. En las imagenes presentadas en la figura antes citadas, la única variación consistió en la posición inicial en la cual se posicionó el robot. Para todas las gráficas, los ejes coordenados representan una vista en alzado de la plataforma virtual donde se realizaron las experiencias. Las unidades tanto del eje de las ordenadas como el eje de las abscisas se encuentran en centímetros.

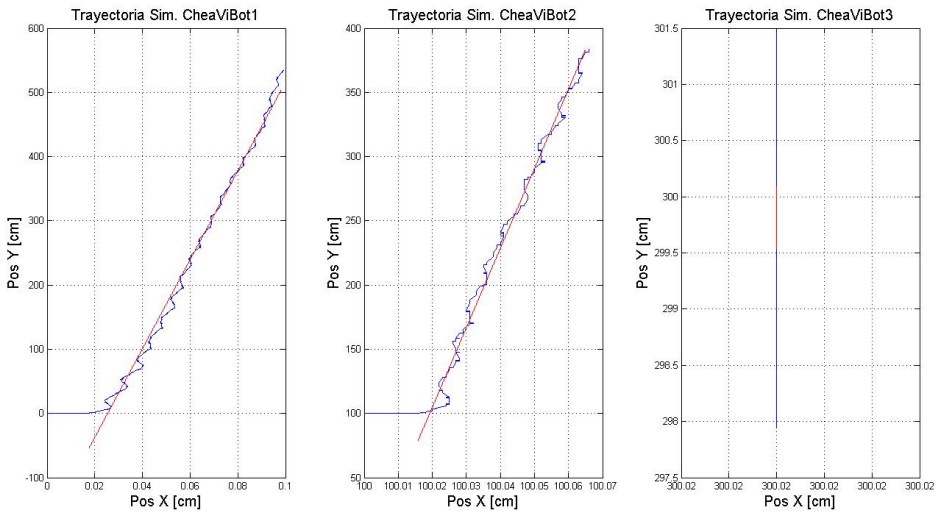

Figura 4.47: Trayectorias obtenidas en tres experiencias realizadas con *Sim CheaViBot*.

Una vez obtenidas las trayectorias, se procedió a obtener las curvas de desplazamientos móviles explicadas en párrafos anteriores. El resultado es el que se presenta en la Figura 4.48. Tal y como se han presentado en etapas anteriores, los ejes de la gráfica se encuentran en formato log-log y representan, en el caso de las ordenadas el desplazamiento antes mencionado y en el caso del eje de las abscisas el tiempo transcurrido durante las experiencias. Este proceso corresponde a la etapa previa mediante la cual se consiguió determinar el desplazamiento medio para la simulación computacional. A modo de resumen, se presenta en la Figura **??**, el desplazamiento obtenido para la simulación computacional. La curva correspondiente es la que se entrega en color azul, mientras que en color rojo, se presenta un modelo exponencial ajustado de los datos obtenidos. La ecuación de este modelo se presenta en 4.7

$$y = 0{,}01 * x^{1{,}4} \tag{4.7}$$

Finaliza el análisis de los datos obtenidos durante estas experiencias, la Figura 4.49, en la cual se presentan las curvas de evolución en el tiempo para el desplazamiento en ambos ejes coordenados (X e Y) y la curva de evolución en el tiempo para la velocidad desarrollada

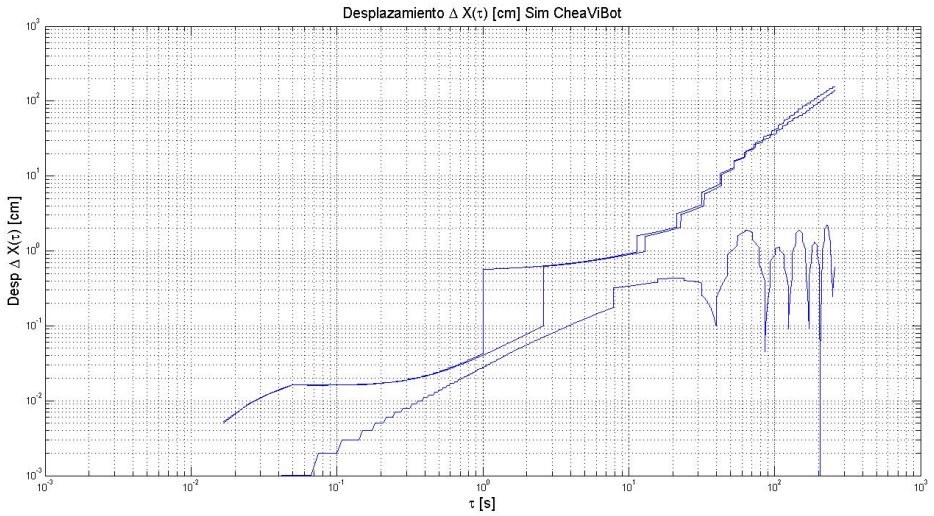

Figura 4.48: Gráfico de desplazamiento móvil para *Sim CheaViBot*.

por la plataforma computacional. En el caso de las tres gráficas, las curvas corresponden a la media aritmética de los datos obtenidos en las tres experiencias realizadas. En la imagen de la izquierda de la Figura 4.49, se presenta la curva de evolución en el tiempo para el eje X, en la imagen del centro se presenta la curva de evolución en el tiempo para el desplazamiento en el eje Y. En ambas imágenes los ejes de las ordenadas se encuentran en centímetros, mientras que el eje de las abscisas, que corresponde al tiempo de evolución, se encuentra en segundos.

Finalmente en la imagen de la derecha, se presenta la curva de evolución de la velocidad en el tiempo, donde los ejes se presentan en unidades de centímetros por segundo en el caso de las ordenadas y en segundos en el caso de las abscisas.

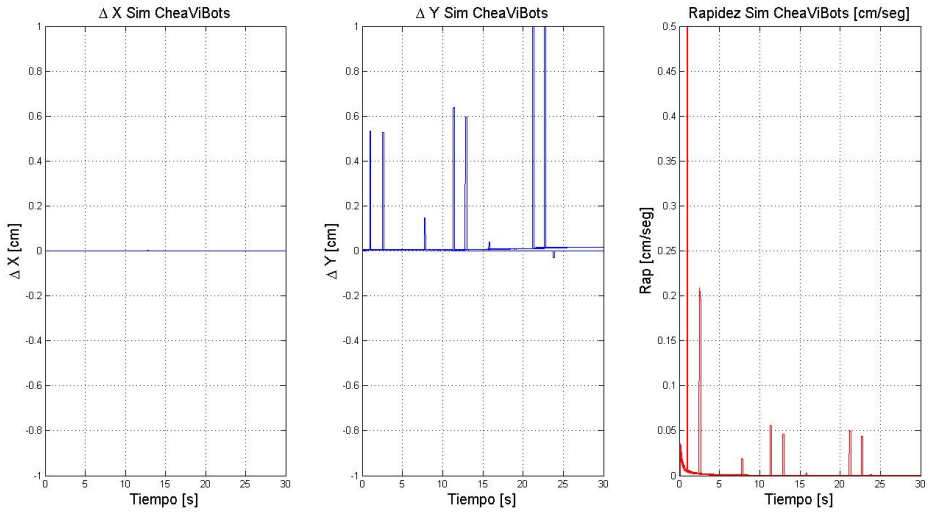

Figura 4.49: Desplazamiento y rapidez para la simulación de *Cubes*.

4.6. Comparación entre plataformas robóticas y modelos computacionales

En la presente sección se procede a contrastar los resutados obtenidos para la plataforma robótica *Cube* con los datos entregados por su simulación computacional *Sim. Cube*.

En primer lugar se procedió con el cálculo de los desplazamientos medios tanto para la plataforma como para su simulación. Para esto se tomaron las columnas pertinentes de la matriz resultado entregadas por la función "calcDeltaW" que fueron almacenadas en un arreglo de matrices durante su cálculo en la primera etapa y posteriormente se procedió a obtener la media aritmética de estos valores.

Los resultados obtenidos en el proceso anterior son los que se presentan en color azul en la Figura 4.6. Esta gráfica tiene sus ejes en formato log-log, donde en el eje de las ordenadas se presenta la evolución de la curva de desplazamiento medio en centíemtros y en el eje de las abscisas se presenta el tiempo en segundos. Junto con el resultado de la media aritmética, se presentan los correspondientes ajustes exponenciales, que en la gráfica se encuentran en color rojo. Estas curvas fueron obtenidas por medio de la función "polyfit" entregada por el programa *Matlab*, a la cual se le entregaron como parámetros el logaritmo de las matrices promedio obtenidas en el paso anterior. El resultado de la curva de ajuste en el caso de la plataforma robótica *Cube* es el que se presenta en la ecuación 4.8

$$y(t) = 3{,}2x(t)^{0}{,}32 \tag{4.8}$$

En el caso del ajuste realizado para la simulación *Sim. Cube*, el resultado es el que se presenta en la ecuación 4.9

$$y(t) = 2{,}2x(t)^{0}{,}2 \tag{4.9}$$

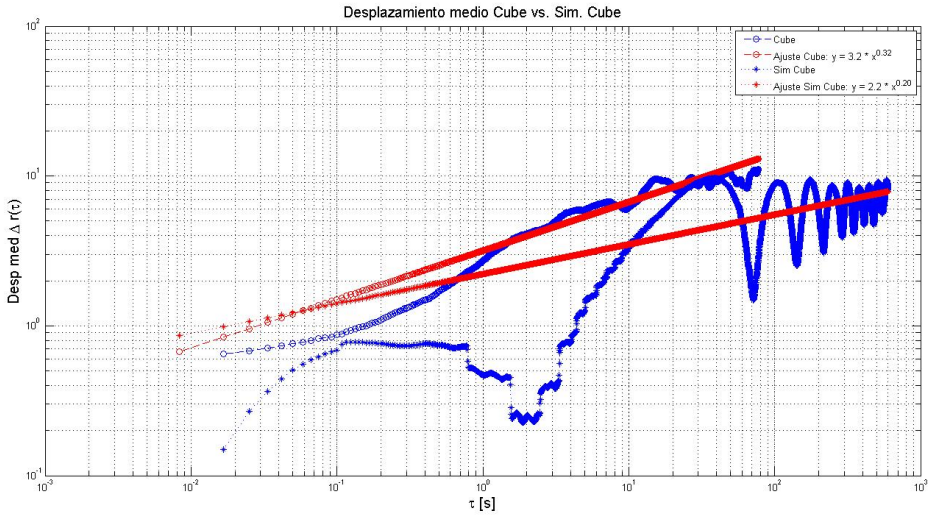

Figura 4.50: Comparación para el desplazamiento medio entre *Cube* y Sim. Cube

Siguiendo con la lógica empleada en la caracterización individual realizada tanto para los robots *Cube* y *Sim. Cube*, se procede con el cálculo de las funciones de distribución de probabilidad para los desplazamientos en los ejes X e Y. Esto se consiguió al crear dos matrices auxiliares. La primera de ellas se utilizó para acumular los datos de las trayectorias de los robots *Cubes* y la segunda se ocupó para juntar los datos correspondientes a *Sim.Cube*. Una vez que se contaó con todos los datos reunidos en estas matrices auxiliares, se procedió con el cálculo de las funciones de distribución de probabilidad mediante la función "calculoPDFadqui".

Los resultados fueron resumidos en la Figura 4.51. En esta imagen es posible observar en color azul los datos correspondientes a los robots *Cubes* y en color azul los de la simulación *Sim. Cube*. En la parte superior se presentan los resultados correspondientes a las funciones de distribución de probabilidad para el desplazamiento en X, mientras que en la parte inferior, se exponen las funciones de distribución de probabilidad para el desplazamiento en el eje Y.

En ambas imágenes las distancias ubicadas en el eje de las abscisas se encuentran en centímetros y el eje de las ordenadas corresponden a la función de probabilidad.

El análisis que continúa corresponde a las funciones de distribución de probabilidad, o PDF, para ambas mediciones. Lo anterior se realizó al promediar las distribuciones de probabilidad obtenidas. Para esto, se utilizaron matrices auxiliares donde se respaldaron los datos obtenidos en la primera etapa de análisis, para posteriormente calcular la media aritmética en ambos casos.

Los resultados para el ángulo de desplazamiento son los que se muestran en la parte superior de la Figura 4.52, donde la curva de color azul corresponde al promedio obtenido para los robots *Cubes*, mientras que en color rojo se presenta el resultado obtenido para *Sim. Cube*. En la parte inferior de la Figura 4.52, se presentan los resultados obtenidos al promediar los datos obtenidos para el ángulo de orientación en ambos casos. En color azul se presentan los datos obtenidos para *Cubes* y en color rojo los obtenidos para *Sim. Cube*.

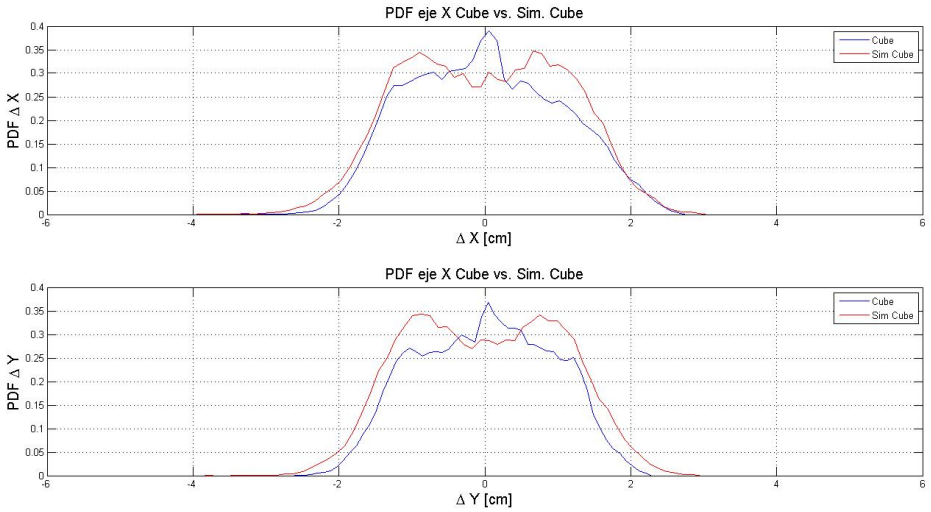

Figura 4.51: Comparación para las PDF de desplazamiento entre *Cube* y Sim. Cube

Las unidades utilizadas en los gráficos para ambos casos corresponden a grados en el caso de las abscisas. Como primera conclusión se tiene que la dispersión de los datos para los

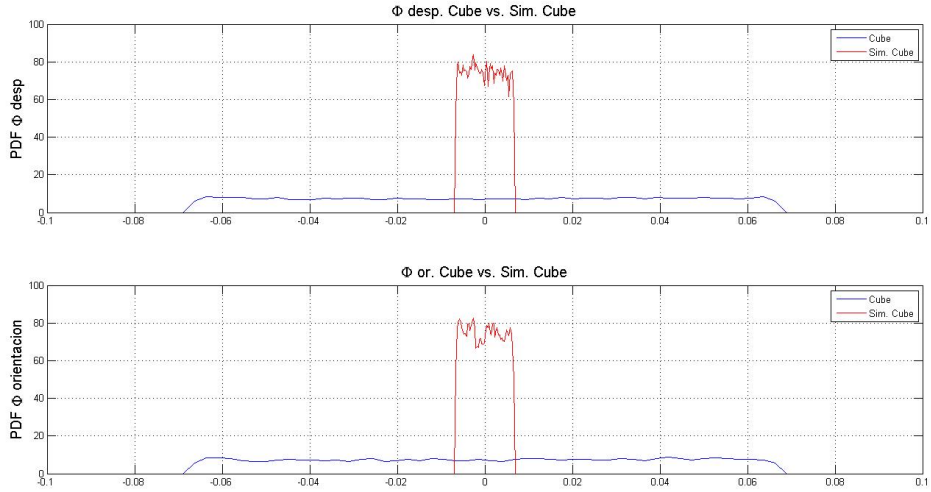

Figura 4.52: Comparación de las PDF para los ángulos de desplazamiento y orientación entre *Cube* y Sim. Cube

robots *Cubes* es mucho mayor que la presentada por su simil computacional en ambos casos. Además, se puede observar que no existen ángulos preferentes en ninguno de los dos casos, dado que las curvas son planas, lo cual indica equiprobabilidad de los ángulos posibles.

Finalmente, se procede con la comparación de los datos obtenidos para el desplazamiento en ambos ejes coordenados (X e Y) y la rapidez media obtenida a partir de las experiencias. Los desplazamientos y la rapidez presentadas en la Figura 4.53, corresponden a las medias aritméticas de los datos obtenidos. El procedimiento utilizado para su obtención es el mismo que se empleó en los casos anteriores, es decir, se recurrió a la creación de matrices auxiliares

para respaldar los datos obtenidos y posteriormente, se procedió al cálculo de las medias aritméticas correspondientes. Los resultados del proceso anterior, se presentan en la Figura 4.53 donde el color azul corresponde a las medias conseguidas para los robots *Cubes*, mientras que en color rojo se presentan los datos correspondientes a *Sim. Cube*.

En el caso de los desplazamientos los ejes de las ordenadas se encuentran en centímetros y el eje de las abscisas se encuentran en segundos. En el caso de las curvas de rapidez en el tiempo, se mantiene el código de colores empleado para el caso de los desplazamientos y las unidades utilizadas fueron centímetros por segundo en el caso de las ordenadas y segundos en el caso de las abscisas.

La diferenia temporal entre las mediciones hace referencia a que durante la simulación, por tratarse de un caso ideal, permitió el movimiento de los *Sim Cube*, por al menos el doble del tiempo que fueron capaces de desplazarse los *Cubes*.

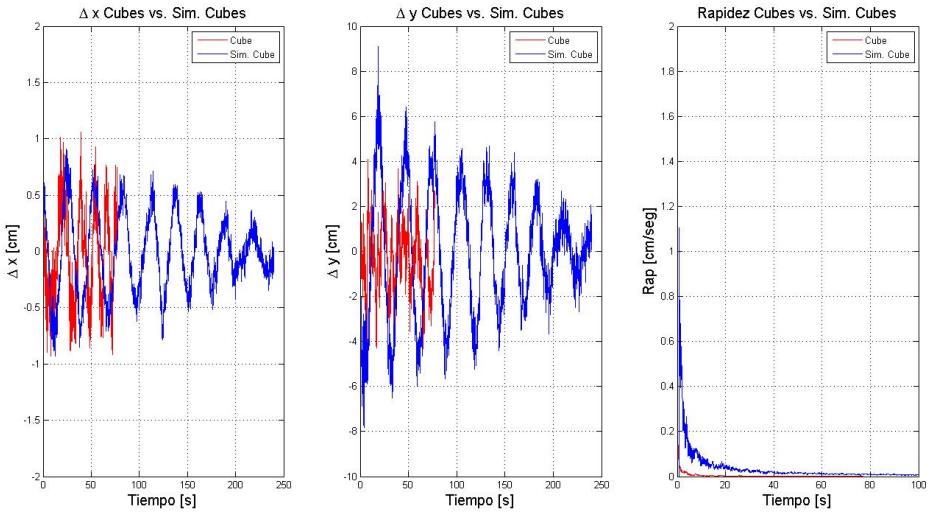

Figura 4.53: Comparación de desplazamientos y rapidez entre *Cube* y Sim. Cube

sectionCaracterización del comportamiento Colectivo en *Cubes* La caracterización del comportamiento colectivo en las plataformas robóticas se llevó a cabo en dos etapas. En primer lugar, se estudiaron las colisiones elásticas para finalmente estudiar el comportamiento de los robots cuando poseen imanes en su contorno.

4.6.1. Comportamiento colectivo en *Cubes*

El análisis global del comportamiento colectivo se realiza por medio del estudio de las propiedades cinemáticas del centro de masas del conjunto. A continuación se presentan los resultados obtenidos al estudiar el comportamiento colectivo de dos grupos de 5 robots *Cubes* que interactuaron por colisiones simples hasta el término en la carga de sus baterías. Al inicio de la experiencia los robots se encontraban cubiertos para evitar que la luz incidiera sobre ellos. Una vez retirada la cubierta que los mantenía en reposo, se comenzó con la grabación.

El análisis de estos datos se realizó por medio de la función `Analisis_Grupal_Cubes.m`. Por medio de esta función se importaron los archivos correspondientes a dos mediciones en las cuales participaban 5 robots. Dichas mediciones fueron tratadas por medio de la función "matricear.m", para posteriormente igualar el largo de las tomas para facilitar el análisis correspondiente.

En la Figura 4.54, se presentan las trayectorias de los centros de masas durante ambas experiencias. En esta gráfica tanto los ejes de las abscisas como el eje de las ordenadas se encuentran en centímetros.

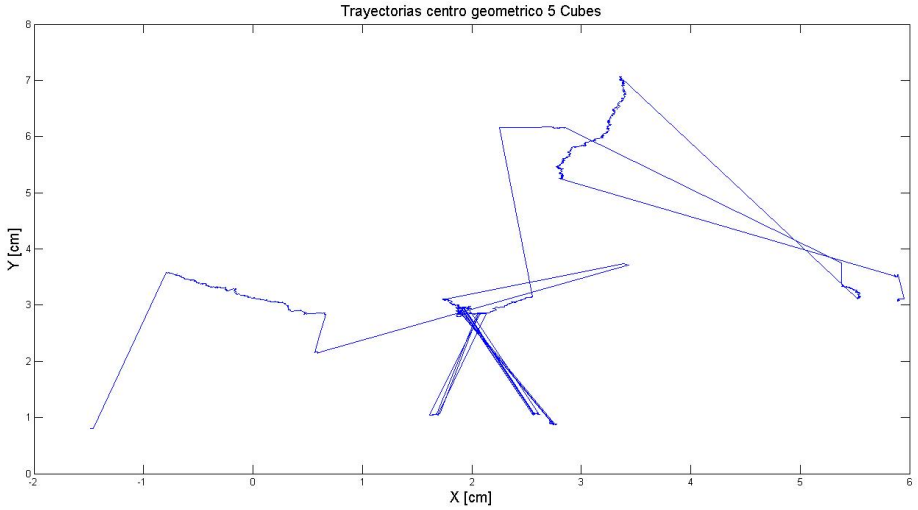

Figura 4.54: Trayectorias para centros geometricos de 5 *Cubes*.

Posteriormente, pero siempre utilizando la función `Analisis_Grupal_Cubes.m`, se procedió al estudio de los desplazamientos medios de los centros geométricos. Para esto se recurrió a la función "calcDeltaW", entregándole como parámetros las columnas segunda y tercera de la matriz resultado de la función "matricear" aplicada sobre las coordenadas de ambos centros geométricos. Este gráfico, como ya es de saber, se presenta en coordenadas log-log, junto con un ajuste logarítmico que se presenta en de color rojo en la Figura 4.55. El resultado de este ajuste es el que se presenta en la ecuación 4.10

$$y(t) = 210{,}1x(t)^{0{,}17} \tag{4.10}$$

Siguiendo con el análisis del centro geométrico, se procedió con el estudio de los desplazamientos en el tiempo en ambos ejes coordenados y la rapidez media. Los gráficos con las curvas de desplazamiento en el tiempo se encuentran a la izquierda y al centro de la Figura 4.56. En ambos gráficos el eje de las ordenadas, que representa el desplazamiento, se encuentra en centímetros, mientras que el eje del tiempo se encuentra en segundos.

En el caso de la curva de rapidez en el tiempo, ubicada a la derecha de la Figura 4.56, las unidades corresponden a centímetro por segundo en el caso de las ordenadas y a segundos en el caso de las abscisas.

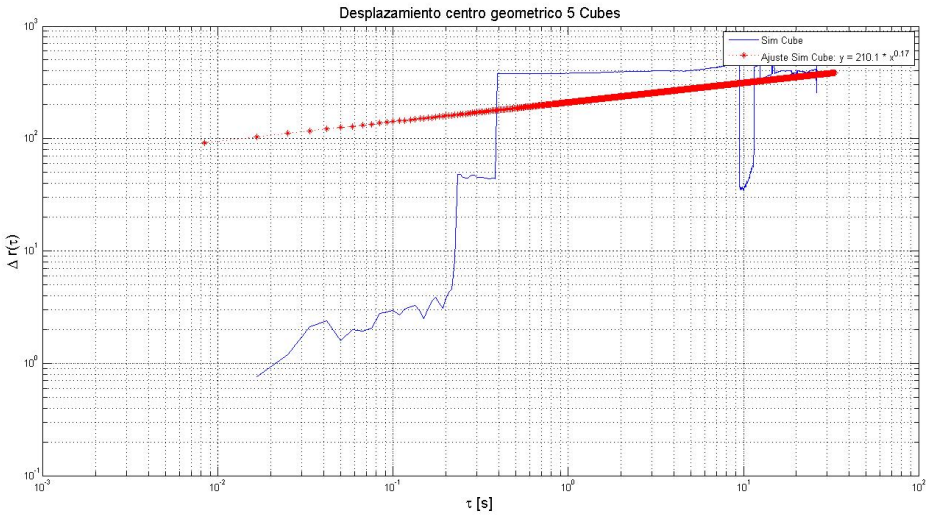

Figura 4.55: Desplazamiento medio para centros geometricos de 5 *Cubes*.

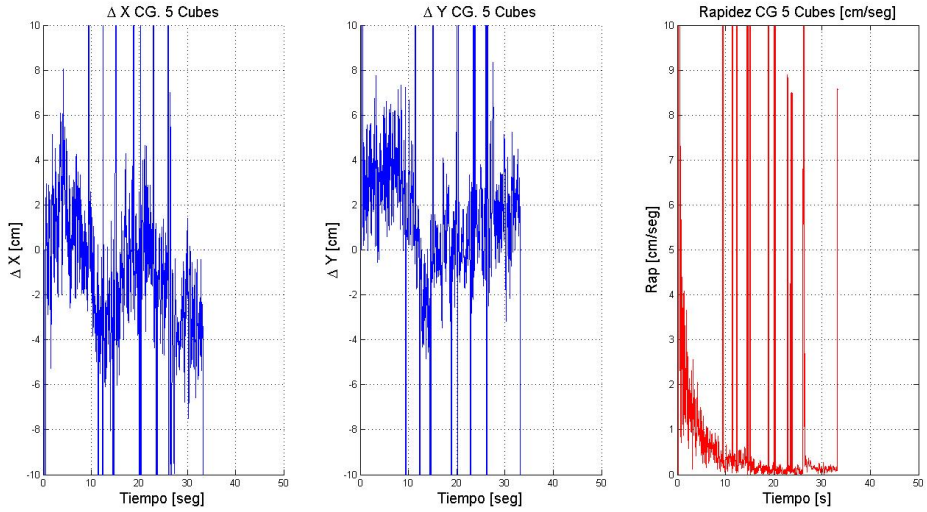

Figura 4.56: Desplazamientos y rapidez media para centros geometricos de 5 *Cubes*.

El último análisis realizado para los datos obtenidos durante esta experiencia, son los que se presentan en la Figura 4.57. En esta imagen se presenta la gráfica de la función de distribución de probabilidad para el ángulo de desplazamiento del centro geométrico. Los ejes coordenados en esta imagen se encuentran en grados, en el caso de las abscisas y la PDF se encuentra en el eje de las ordenadas.

Imanes

En esta segunda etapa, cada uno de los robots interactuantes del conjunto poseen dos imanes en total. Uno de ellos se ubica en la cara lateral derecha, mientras que el otro se ubica en la cara lateral izquierda. Al igual que en el caso de la primera experiencia para el

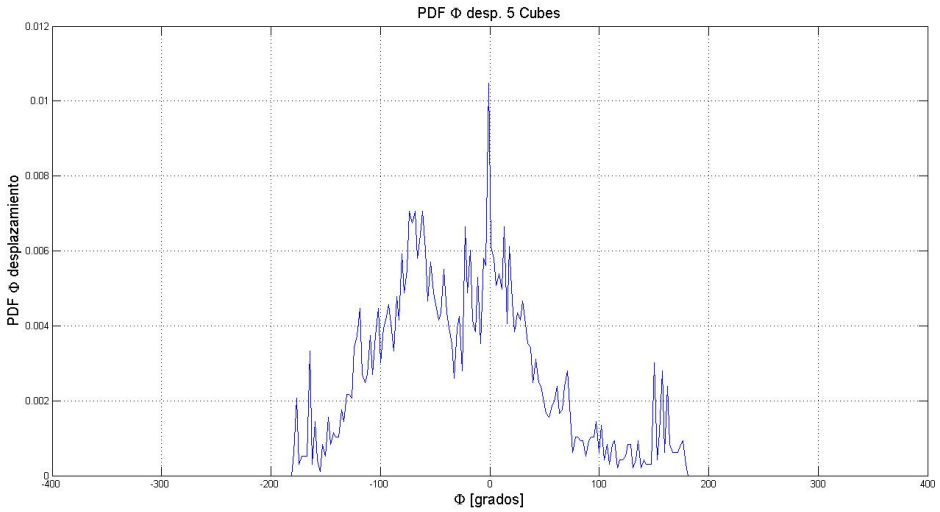

Figura 4.57: PDF ángulo de desplazamiento para centros geometricos de 5 *Cubes*.

comportamiento colectivo utilizando esta plataforma, se procede con el análisis del centro geométrico de los robots.

En primer lugar se presentan en la Figura 4.58, las trayectorias presentadas durante las experiencias realizadas con cinco, diez y quince robots. En primer lugar se procedió con el arreglo de la información recopilada mediante la función "matricear.m", con la cual se realizó la transformación de las unidades de longitud a centímetros, junto con crear una columna que representa el tiempo de medición. La gráfica corresponde al plano definido por la superficie donde se realizaron las experiencias y las curvas, en color azul, representan las trayectorias descritas por los centros geométricos durante las experiencias. El eje de las ordenadas, que representa la coordenada Y se encuentra en centímetros, al igual que el eje de las abscisas que representa la coordenada X.

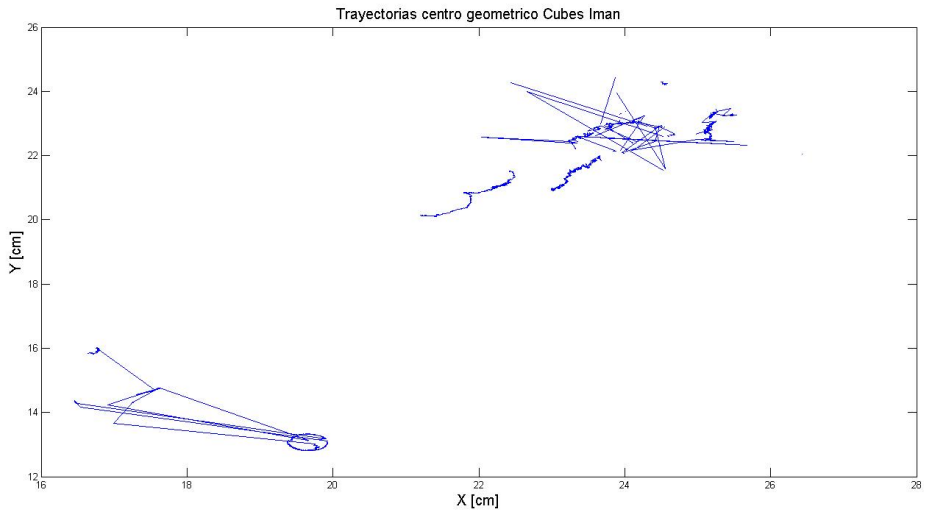

Figura 4.58: Trayectoria para centros geometricos de *Cubes* con imanes.

Siguiendo con la pauta definida para el análisis de los datos obtenidos para los centros geométricos, se procedió por medio de la función "calclDeltaW" a obtener los desplazamientos medios. Los resultados obtenidos por medio de este proceso, son los que se presentan en la Figura 4.59.

Junto con la gráfica para el desplazamiento medio, en la imagen antes citada, se presenta el ajuste exponencial correspondiente, cuya ecuación se presenta en la ecuación 4.11.

$$y(t) = 24{,}4x(t)^{1{,}21} \tag{4.11}$$

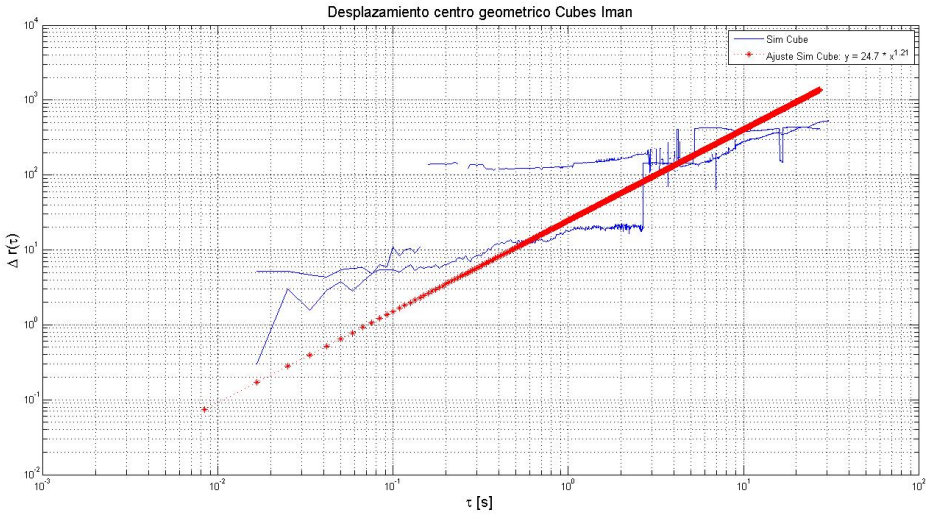

Figura 4.59: Desplazamiento medio para centros geometricos de *Cubes* con imanes.

El paso siguiente correspondió a la obtención de las curvas de desplazamiento en los ejes coordenados en función del tiempo, junto con la rapidez en función del tiempo. Los resultados de dicho proceso, son los que se presentan en la Figura 4.60. En la imagen de la izquierda se presenta la curva de desplazamiento a lo largo del eje X en el tiempo, mientras que en la imagen del centro se presenta la curva de desplazamiento en el eje Y en el tiempo. En ambas gráficas las unidades de desplazamiento corresponden a centímetros, mientras que las unidades de tiempo (abscisas) se encuentran en segundos.

En el caso de la gráfica que representa la curva de rapidez en el tiempo, se tiene que el eje de las ordenadas se encuentra en centímetros por segundo, mientras que el eje de las abscisas se encuentra en segundos. Finalmente el análisis de esta etapa del trabajo concluye con la presentación de la función de distribución de probabilidad para el ángulo de desplazamiento del centro geométrico. Dicha curva es la que se presenta en la Figura 4.61. En esta imagen se tiene que en el eje de las ordenadas se encuentran los valores correspondientes a la PDF mientras que en el eje de las abscisas se encuentra en grados.

Esta curva presenta una punta muy acentuada en el centro, lo que demuestr que el desplazamiento del centro geométrico no presenta un ángulo preferente para su movimiento.

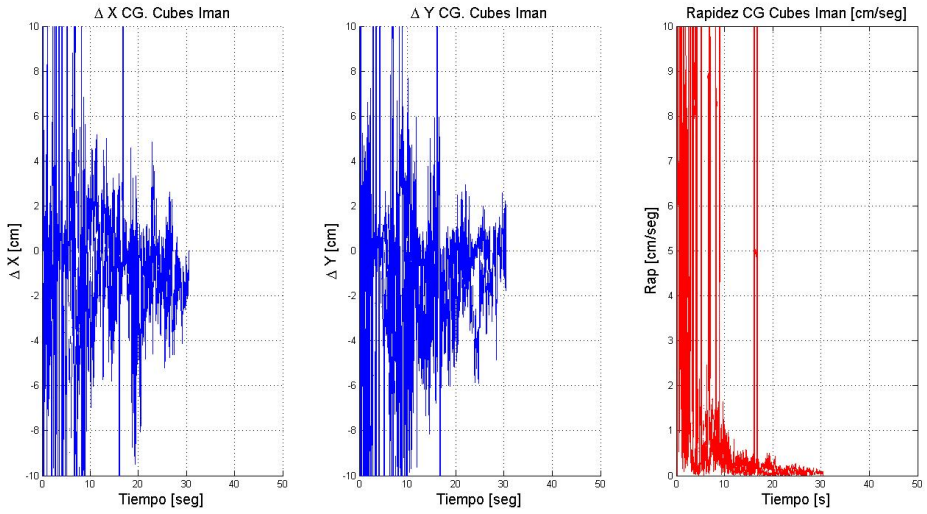

Figura 4.60: Curvas de desplazamiento y rapidez para centros geometricos de *Cubes* con imanes.

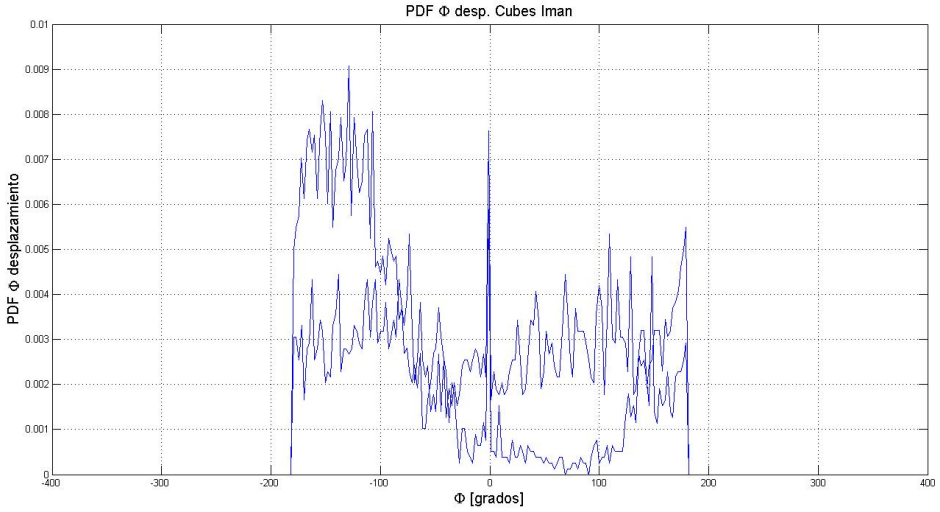

Figura 4.61: PDF ángulo de desplazamiento para centros geometricos de *Cubes* con imanes.

4.6.2. Comportamiento colectivo en la simulación de *Cubes*

En esta sección se presentan los resultados obtenidos al escalar las experiencias realizadas con los robots *Cube* utilizando la plataforma computacional *Sim. Cube*. Estas experiencias fueron desarrolladas utilizando la plataforma computacional desarrolladas para este fin. En la figuras que se presentan a continuación se expone el comportamiento obtenido al estudiar el comportamiento del centro geométrico de los *Sim. Cube*. Dicho estudio se realizó empleando la función "`Analisis_Grupal_Juanitos`". Mediante esta función se realizó la importación directa de los archivos correspondientes. Posteriormente se procedió a unificar el largo de las tomas para posteriormente comenzar con el estudio de los centros geométricos.

En la Figura 4.62 se presentan las trayectorias de los centros geométricos al comenzar desde un conjunto de dos robots hasta completar los cincuenta en incrementos de dos en dos, es decir, se comenzó con un conjunto de dos robots, luego cuatro, seis, hasta completar un conjunto de cincuenta robots *Sim Cube*. El tiempo de muestreo para estas experiencias corresponde a 5 minutos.

Al igual que en el caso de las experiencias individuales, los datos obtenidos durante estas experiencias fueron tratados mediante la función "matricear" a objeto de realizar la conversión de unidades correspondientes y un ordenamiento que facilitara el análisis que se presenta. En la Figura 4.62, el eje de las ordenadas corresponde a la posición en la coordenada Y, mientras que en el eje de las abscisas se presenta la coordenada X, ambas en centímetros.

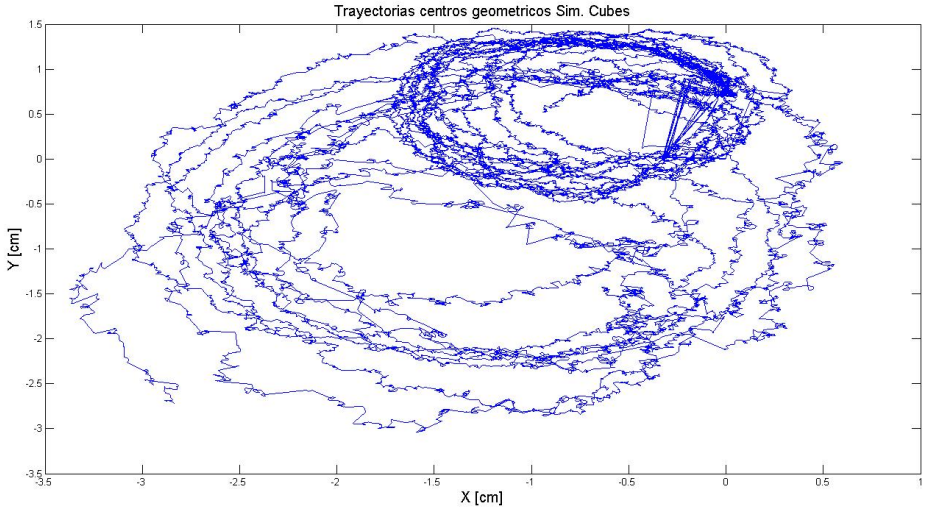

Figura 4.62: Trayectorias para centros geometricos.

Siguiendo con la lógica del análisis desarrollado con las experiencias individuales, el segundo paso conistió en la determinación del desplazamiento incremental del centro geométrico. Lo anterior se realizó por medio de la función "calcDeltaW", al entregarle como parámetros la segunda y tercera columna de la matriz resultado de la función "matricear" para cada una de las muestras estudiadas. Los resultados del proceso anterior, son los que se muestran en la Figura 4.63. Como es costumbre para este tipo de gráficos, los resultados se entregan en formato log-log, donde el eje de las abscisas corresponde al tiempo de muestreo y el eje de las ordenadas corresponde al desplazamiento incremental.

Además, en este gráfico, se presenta un ajuste exponencial (en color rojo), cuya ecuación se encuentra en 4.12.

$$y(t) = 4{,}4x^{0{,}1} \tag{4.12}$$

Posteriormente, se estudió el comportamiento de las funciones de distribución de probabilidad en ambos ejes coordenados. Para esto se utilizó la función "calculoPDFadqui" entregándole como parámetros la cuarta y quinta columnas de la matriz resultado de la función

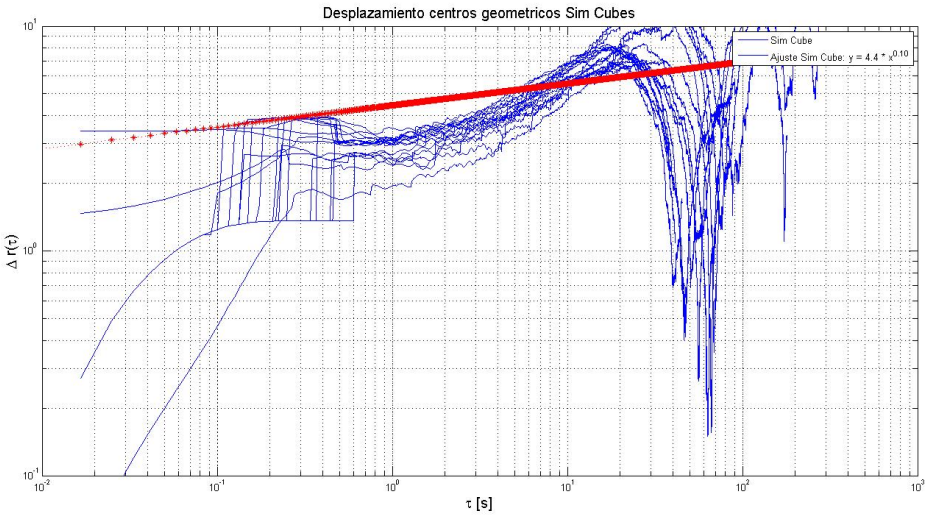

Figura 4.63: Desplazamientos medios en *Cubes* simulados.

matricear aplicada sobre las coordenadas del cenrtro geométrico. El resultado de este proceso es el que se muestra en la Figura 4.64, donde en la imagen de la izquierda se presenta la función de distribución de probabilidad para la coordenada X y en la imagen de la derecha se presenta su equivalente para la coordenada Y. En ambos casos, el eje de las ordenadas presenta la función de distribución de probabilidad, mientras que en el eje de las abscisas se presenta el desplazamiento en centímetros. A partir de este resultado se puede observar la tendencia a la estanqueidad del centro geométrico, es decir, tiende a permanecer en reposo, lo cual viene a confirmar lo observado en la Figura 4.62.

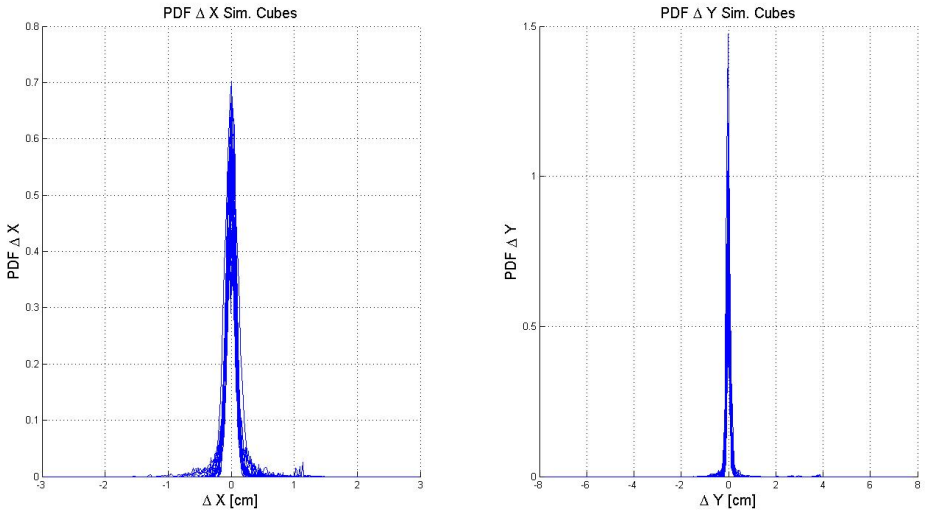

Figura 4.64: PDF para desplazamientos en x e y.

Una vez analizados los datos correspondientes al desplazamiento en los ejes coordenados, se procedió a estudiar el comportamiento angular del desplazamiento. Para esto se analizó el ángulo de desplazamiento presentado por el centro geométrico mediante la función "calcDeltaW". Como parámetro para esta función se entregó la sexta columna de la función

"matricear" aplicada sobre las coordenadas de los centro geométricos. El resultado del proceso anterior es el que se presenta en la Figura 4.65 en la cual el eje de las abscisas corresponde al ángulo de desplazamiento en grados y el eje de las ordenadas corresponde a la PDF. En esta imagen es posible distinguir dos puntas, la primera de ellas se ubica en -90 grados mientras que la otra se ubica cerca de 180 grados.

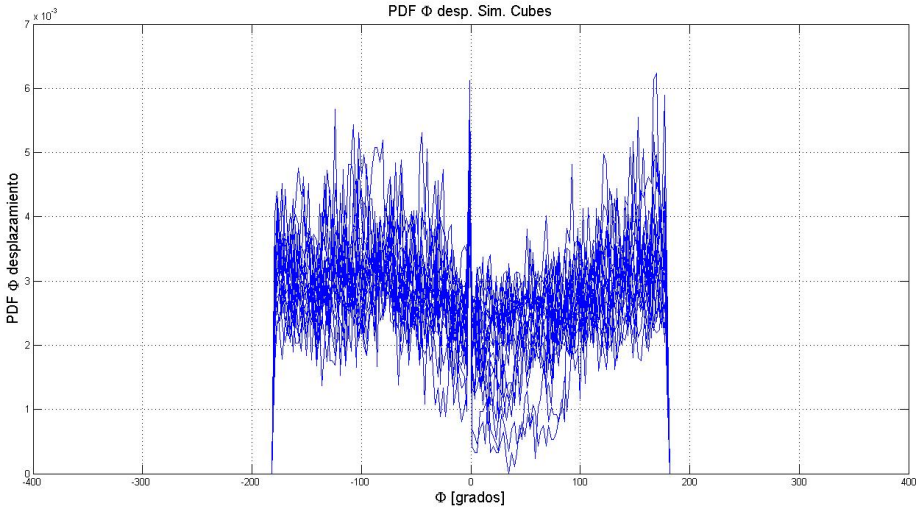

Figura 4.65: PDF para desplazamientos en x e y.

El paso siguiente dentro de la función de análisis corresponde a la obtención de los desplazamientos por eje y la rapidez media del centro de geométrico. Para esto se graficaron la cuarta y quinta columna de la matriz resultado de la función matricear aplicada a las columnas de posición de la matriz auxiliar que contiene las coordenadas de los centros geométricos. Como resultado se obtuvo la Figura 4.66. En ella se puede distinguir de izquierda a derecha las curvas de desplazamiento en el tiempo de las coordenadas X e Y, mientras que en la imagen de la derecha se presenta la curva de rapidez en el tiempo en color rojo. En las curvas de desplazamiento en el tiempo las abscisas se encuentran en segundos, mientras que las ordenadas se encuentran en centímetros. En el caso de la curva de rapidez en el tiempo, el eje de las ordenadas se encuentra en centímetros por segundo y el eje del tiempo se encuentra en segundos.

A partir de esta imagen se puede determinar que el comportamiento colectivo presentado por estos robots tiende a ser el mismo que ocurre en el caso individual, desplazamientos oscilantes en los ejes coordenados, mientras que la rapidez tiene una tendencia decreciente en el tiempo conforme la carga de la batería (que se encuentra incluida en la simulación) disminuye.

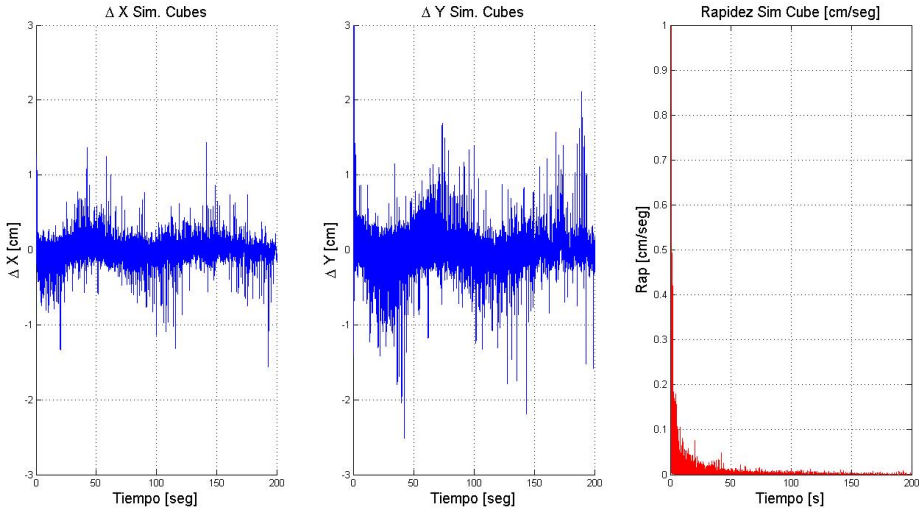

Figura 4.66: Desplazamiento en X e Y junto con la rapidez media para los centros geométricos en Sim. Cube

Capítulo 5

Conclusiones

En contraste con la forma en que los humanos controlan, producen y reparan objetos en una forma de control arriba-abajo, la naturaleza hace uso extensivo de la auto-organización descentralizada y auto-sellado. Resulta atractivo adoptar los principios y métodos ocupados por la naturaleza para los productos desarrollados por los humanos y la forma en la que controlan las actividades en las que se ven envueltos. En particular, en el caso de la micro y nano producción de estructuras delicadas estas tecnologías resultan muy útiles dado que resulta complicado adoptar esquemas de construcción arriba-abajo.

Otra área significativa, son los sistemas computacionales que controlan objetos y actvidades. Los procedimientos de control son a menudo diseñados de forma jerárquica para asegurar que las cosas pasen de la manera que se pretende y en el tiempo que corresponde. Los beneficios de la construcción arriba-abajo es la predictibilidad, pero la parte negativa es el costo administrativo de los recursos de control no productivos y la vulnerabilidad del sistema. En sistemas de control jerárquico, la resistencia no es mejor que el elemento más débil. En los sistemas descentralizados auto-organizados, por otro lado, tienen la ventaja de tener costos de administración muy bajos y ser muy robustos, dado que no hay estructuras de comando centralizadas que puedan ser perturbadas. Por el contrario, las metas de los sistemas descentralizados tienen que ser construidas como propiedades emergentes dentro de las unidades simples de los sistemas auto-organizados.

Los principios para el control descentralizado pueden ser aplicados en un amplio rango de actividades de nuestra vida diaria: producción y mantenimiento de productos, programas computacionales, organización de sistemas de servicio y organización de estructuras sociales. Resulta, por tanto, muy interesante estudiar como la naturaleza se auto-organiza con el objetivo de entender los principios centrales y poder implementarlos cuando se están diseñando nuevos productos. Como los procesos de diseño son llevados a cabo entre expertos de distintas disciplinas, la comunicación de los principios centrales de la bio-imitación es un elemento vital en la transferencia del conocimiento. Muchos de los procesos de auto-organización y auto-sellado son bien entendidos dentro de la comunidad de biólogos, pero el conocimiento no se encuentra en una forma que sea entendible y utilizable por los diseñadores.

5.1. Comportamiento Colectivo en las plataformas estudiadas

A lo largo de las experiencias realizadas, se ha podido observar como el comportamiento colectivo en las plataformas estudiadas se ve afectado por la forma en la cual interactúan los individuos, sin embargo, no ha sido posible desarrollar comportamiento colectivo de acuerdo a los principios estudiados a lo largo de la bibliografía. Lo anterior se debe en gran medida la gran cantidad de variables que requieren ser estudiadas para este fin.

Comportamiento individual de la plataforma *Cube*

Conforme a las experiencias previas realizadas donde se estudió la distribución de velocidades al encontrarse en una plataforma que restringía su movimiento, las trayectorias observadas corresponden a cilindros al ser observados en función del tiempo. Al estudiar la curva de desplazamiento medio para esta plataforma y conforme al modelo de ajuste desarrollado, el movimiento exhibido corresponde al tipo Browniano. En las funciones de distribución de probabilidad obtenidas para el desplazamiento en los ejes X e Y se observo una clara tendencia al desplazamiento nulo. El ángulo de desplazamiento por su parte no muestra una tendencia clara, por lo que se concluye que el ángulo de desplazamiento tiende a ser aleatorio. Cuando se observa la curva que relaciona el ángulo de orientación del robot con el ángulo de desplazamiento, se puede concluir que el robot se desplaza en la dirección en la cual se posiciona. Las curvas de desplazamiento en función del tiempo para los ejes X e Y presentan comportamiento oscilatorio, lo cual tiene directa relación con el mecanismo de movimiento del robot. En el caso de la curva de rapidez media en función del tiempo, se observa una tendencia decreciente en el tiempo, lo cual tiene directa relación con el tiempo de descarga de la batería que impulsa los motores, dado que el tiempo de muestreo, tal y como fue demostrado en la sección de caracterización individual para esta plataforma, corresponde a no más de cien segundos.

Comportamiento individual de la plataforma *CheaViBot*

A diferencia de las experiencias realizadas durante el estudio de la plataforma *Cube*, el objetivo del estudio de esta plataforma correspondió a demostrar la capacidad con la que esta cuenta para futuros estudios en el campo de la robótica modular. En este sentido, el robot demostró una gran maniobrabilidad al ser guiado en ambientes carentes de luz, pero gran sensibilidad a las variaciones de luz en su entorno, lo cual no causa mayor sorpresa dado que el mecanismo de control corresponde a la presencia o ausencia de luz en los receptores adecuados para la activación de los motores que producen su desplazamiento.

Comportamiento colectivo presentado en robot *Cube*

Al estudiar el comportamiento colectivo de esta plataforma, se logró determinar que el movimiento de su centro geométrico tiende a mantener el comportamiento exhibido por la plataforma individual. Básicamente al observar la trayectoria de su centro geométrico para distintos conjuntos de robots, se pudo observar que la tendencia es a la estanqueidad.

La curva de desplazamiento medio presentada corresponde a un movimiento del tipo Browniano, donde su desplazamiento a lo largo de los ejes es de tipo oscilatorio y la curva de rapidez media tiende a decrecer conforme la carga de las baterías que propulsan los robots tienden a descargarse.

El ángulo de desplazamiento del centro geométrico no muestra tendencias claras al estudiar la función de distribución de probabilidad, hecho que se acentúa conforme se acumulan los resultados obtenidos para distintos conjuntos de robots.

En el caso de las experiencias realizadas con imanes, el centro geométrico acentúa su tendencia a la estanqueidad, disminuyendo el desplazamiento medio conseguido. La función de distribución de probabilidad para el ángulo de desplazamiento no muestra una tendencia clara, tal y como sucede con las experiencias anteriores.

Simulación computacional para robots *Cube* (*Sim. Cube*)

La simulación computacional desarrollada con el objetivo de escalar los resultados obtenidos durante las experiencias realizadas con los robots *Cube*, demostró ser una herramienta potente, al replicar de manera satisfactoria el comportamiento presentado por dicha plataforma tanto en el comportamiento individual como colectivo.

Bibliografía

[1] Homepage of seaswarm project [en línea] < http://senseable.mit.edu/seaswarm/index.html > [consulta: 03/12/2014].

[2] I. Giardina G. Parisi R. Santagati y M. Viale A. Cavagna, A. Cimarelli. Scale-free correlations in starling flocks. *Proc Natl Acad Sci*, 107:11865–11870, 2010.

[3] A Campo et al. A Decugniere, B Poulain. Enhancing the cooperative transport of multiple objects. *Proceedings if the 6th International Conference on An Colony Optimization and Swarm Intelligence (ANTSÂ´08)*, 5217:307–314, 2008.

[4] D. Helbing y J.-L. Deneubourg A. Dussotour, V. Fourcassie. Optimal traffic organization in ants under crowded conditions. *Nature*, 428:70–73, 2004.

[5] F Gokce y E Sahin A E Turgut, H Celikkanat. Self organized flocking in mobile robot swarms. *Swarm Intelligence*, 2:97–120, 2008.

[6] H C Celikkanat L Bayindir y E Sahin A E Turgut, F Gokce. Kobot:a mobile robot designed specifically for swarm robotics search. *Tech Rep, KOVAN Research Lab, Department of computer engineering. Middle East Technical University*, 2007.

[7] M Doringo D Amor L Magdalena y F monasterio-Huelin A Gutierrez, A Campo. An open localization and local communication embodied sensor. *Sensors*, 8:7545–7568, 2008.

[8] J Maja Mataric y S Gaurav Sukhatme A Howard. Mobile sensor network deployment using potential elds:a distributed, scalable solution to the area coverage problem. *Proceedings of the 6th International Symposium on Distributed Autonomous Robotic Systems (DARS02), Fukuoka, Japón*, Junio 2002.

[9] V Kumar J P Ostrowki J Spletzer y C J Taylor A K Das, R Fierro. A vision-based formation control framework. *IEEE Transactions on robotics and automation*, 18:813–825, 2002.

[10] R OÂ´Grady y M Dorigo A L Christensen. Morphology control in a mutirobot system. *IEEE Robotic and Automation Magazine*, 14:18–25, 2007.

[11] A Martinoli y R M Goodman A T Hayes. Swarm robotic odor localization: off-line optmization and validation with real robots. *Robotica*, 21:427–441, 2003.

[12] R. Kobayashi T. Saigusa y T. Nakagaki A. Tero, K. Yumiki. Flow-network adaption in *Physarum Amoebae*, journal = Theor Biosci, year = 2008, volume = 127, pages = 89-94,.

[13] R. Kobayashi y T. Nakagaki A. Tero. A mathematical model for adaptive transport network in path finding by true slime mold. *J Theor Biol*, 224:553–564, 2007.

[14] T. Saigusa K. Ito D.P. Bebber M.D. Fricker K. Yumiki R.. Kobayashi y T. Nakagaki A. Tero, S. Takagi. Rules for biologically inspired adaptive network design. *Science*, 327:439–442, 2010.

[15] Design Analysis and Control of a Planar Micro-robot Driven by Two Centripetal-Force Acruators. Panagiotis vartholomeos y evangelos papadopoulos. *Proceedings of the 2006 IEEE International Conference on Robotics and Automation*, pages 649–654, 2006.

[16] Alber-László Barabási András Czirók and Tamás Vicsek. Collective motion of self-propelled particles: Kinetic phase transition in one dimension. *Physical Review Letters*, 82:209–212, 1999.

[17] Saroj Rangnekar Anula Khare. A review of particles swarm optimization and its applications in solar photovoltaic system. *Applied soft Computing*, 13:2997–3006, 2013.

[18] Olafsen J S Urbach J S Aranson I S, Snezhko A. Comment on long-lived giant number fluctuations in a swarming granular nematic. *Science*, 320:612, 2008.

[19] C Ronald Arkin. Behavior based robotics. *MIT Press, Cambridge Mass, USA*, 1998.

[20] T Vaughan y A Howard B P Gerkey. The player/stage project: Tools for multirobot and distributed sensor systems. *Proceedings of the international conference on advanced robotics*, pages 317–323, 2003.

[21] Albano E V Baglietto G. Finite-zise scaling analysis and dynamic study of the critical behavior of a model for the collective displacement of self-driven individuals. *Physical Review*, 78, 2008.

[22] Albano E V Baglietto G. Computer simulations of the collective displacement of self-propelled agents. *Computer Physics Communications*, 180:527–531, 2009.

[23] Heppner F H Bajec I L. Organized flight in birds. *Animal Behavior*, 78:777–779, 2009.

[24] Brunnet K G de Almeida R M C Chaté H Belmonte J M, Thomas G L. Self-propelled particle model for cell storing phenomena. *Physical Review Letters*, 100, 2008.

[25] Gerardo Beni. From swarm intelligence to swarm robotics. *Lecture Notes in Computer Science*, 3038:8–16, 2004.

[26] R. Ernst y F. Roces C. Kleineindan. Wind induces ventilation of the giant nests of the leaf cuttting ant atta vollenweideri. *Naturwissen*, 88:301–305, 2001.

[27] Kahng A Cao YU, Fukunaga AS. Cooperative mobile robotics: antecedents and directions. *Auton Robot*, 4:7 – 27, 1997.

[28] Grégoire G Peruani F Raynaud F Chaté H, Ginelli F. Modeling collective motion: variations on the vicsek model. *The European Physical Journal B*, 64:451–456, 2008.

[29] Gregoiré G Raynaud F Chaté H, Ginelli F. Collective motion of self-propelled particles interacting without cohesion. *Physical Review*, 77, 2008.

[30] Montagne R Chaté H, Ginelli F. Simple model for active nematics:quasi-long-range order and giant fluctuations. *Physical Review Letters*, 96, 2006.

[31] Martinoli A Correll N. Towards optimal control of self-organized robotic inpection systems. In *8th International IFAC symposium on robot control*, Citeseer, 2006.

[32] Franks N R Levin S A Couzin I D, Krause J. Effective leadership and decision-making in animal groups on the move. *Nature*, 433:513–516, 2005.

[33] James R Ruxton G D Franks N R Couzin I D, Krause J. Collective memory and spatial sorting in animal groups. *Journal of Theoretical Biology*, 218:1–11, 2002.

[34] Krause J Couzin I D. Self-organization and collective behavior in vertebrates. *Advances in the study if Behavior*, 32:1–75, 2003.

[35] Franks NR Levin SA Couzin ID, Krause J. Effective leadership and decision-making in animal groups on the move. *Nature*, 433:513–6, 2005.

[36] Dong J-G Cucker F. Avoiding collisions in flocks. *IEEE Transactions on Automatic Control*, 55:1283–1243, 2010.

[37] Smale S Cucker F. Emergent behavior in flocks. *IEEE Transactions on Automatic Control*, 52:852–862, 2007.

[38] Strombom D. Collective motion from local attraction. *Journal of Theorizal Biology*, 283:145–251, 2011.

[39] Kumar V Ostrowski JP Spletzer J Taylor CJ Das AK, Fierro R. A vision-based formation control framework. *IEEE Trans Robot Autom*, 18:813–25, 2002.

[40] Vicsek T Derzsi A, Szolosi G. Most minimal spp model. *http://hal.elte.hu/ vicsek/SPP-minimal/.*, 2009.

[41] Groβ R Trianni V Labella TH Nouyan S et al. Dorigo M, Tuci E. The swarm-bots project. *Swarm robotics. Lecture notes in computer science*, 3342:31–44, 2005.

[42] Stormont DP. Autonomous rescue robots swarms for first responders. In *Proceedings if the 2005 IEEE international conference on computational intelligence for homeland security and personal safety*, pages 151 – 7, IEEE, 2005.

[43] M Dorigo y G Theraulaz E Bonaneau. swarm intelligence: From natural to artificial

system. *Oxford university press, NY, USA*, 1999.

[44] V. Botti E. del Val, M. Rebollo. Combination of self-organization mechanisms to enhance service discovery in open systems. *Elsevier*, 279:138–162, 2014.

[45] A E Turgut y E Sahin E Ugur. Dispersion of a swarm of robot based on realistic wireless intensiti signals. *Proceedings of the 22nd International Symposium on Computer and Information Sciences*, pages 1–6, 2007.

[46] Johnson A Spanton R Sun JA English S, Gough J. Formica: a swarm robotics project. *Technical report. University of Southampton*, 2008.

[47] Nystrom J Tunstrom K Eriksson A, Jacobi M N. Determining interaction rules in animal swarms. *Behavioral Ecology*, 21:1106–1111, 2010.

[48] G A Di Caro y L M Gambardella F Ducatelle, A Forester. New task allocation methods for robotic swarms. *Proceedings of the 9th IEEE/RAS Conference on Autonomous Robot Systems and Competitions, Castelo Branco, Portugal, Mayo*, 2009.

[49] D Floreano S Nolfi J L Deneubourg y M Doringo F Mondada, L M Gambardella. The cooperation of swarm-bots: physical interactions in collective robotics. *IEEE Robotics an automation magazine*, 12:21–28, 2005.

[50] E Franzi y A Guignard F Mondada. The development of khepera. *Proceedings of the 1st international Khepera Workshop*, 64:7–14, 1999.

[51] M Bonani y X Raemy F Mondada. The e-puck,a robot designed for education in engineering. *Proceedings of the 9th Conference on Autonomous Robot systems and competitions*, 1:59–65, 2009.

[52] S Franklin. Coordination without communication. *http://www.msci.memphis.edu/franklin/coord.html*, 2010.

[53] Biro D Freeman R. Modelling group navigation: dominance and democracy in homing pigeons. *The Journal of Navigation*, 62:33–40, 2009.

[54] Do KD Lim KW Fua CH, ge SS. Multirobot formations based on the queue-formation scheme with limited communication. *IEEE Trans Robot*, 23:1160–9, 2007.

[55] E milios y D Wilkes G Dudek, M R M Jenkin. A taxonomy for multi-agent robotics. *Autonomous robots*, 3:375–397, 1996.

[56] SinanHanay Y Ilter M Koksal Gazi V, Fidan B. Aggregation, foraging, and formation control of swarms with non-holonomic agents using potential functions and sliding mode techniques. *Turk K Elec Engin*, 15:149–68, 2007.

[57] Chaté H Ginelli F. Relevance of metric-free interactions in flocking. *Physical Review Letters*, 205, 2010.

[58] Chaté H Grégoire G. Onset of collective and cohesive motion. *Physical Review letters*, 92, 2004.

[59] Tu Y Grégoire G, Chaté H. Moving and staying together without a leader. *Physica D*, 181:157–170, 2003.

[60] Ben Jacob E Grossman D, Aranson I S. Emergence of agent swarm migration and vortex formation through inelastic collisions. *New Journal of Physics*, 10, 2008.

[61] Reggia JA Grushin A. Stigmergic self-assembly of prespecified artificial structures in a constrained and continuos environment. *Integr Comput Aided Eng*, 13:298 – 312, 2006.

[62] Erol S Hande C. Steering self-organized robot flockks through externally guided individuals. *Neural Comput Appl*, 9:849–65, 2010.

[63] Kunz H Hemelrijk C K. Density distribution and size sorting in fish schools: an individual-based model. *Behavioral Ecology*, 16:178–187, 2005.

[64] O. Holland and C. Melhuish. Stigmergy, self organization, and sorting in collective robotics. *Artificial Life*, 5:173–202, 1999.

[65] Yan-wei Zhu Yuan-wen Zhang Huan Huang, Le-ping Yang. Collective trajectory planning for satellite swarm using inter-satellite electromagnetic force. *Acta ASTRONAUTICA*, 104:220–230, 2014.

[66] Aldana M Huepe C. New tools for characterizing swarming systems: A comparison of minimal models. *Physica A*, 387:2809–2822, 2008.

[67] M I Ecemis y P Gaudiano J A Rothermich. Distributed localization and mapping with a robotic swarm. *Lectures Notes in Computer Science, Springer, Berlín, Alemania*, pages 58–69.

[68] C R Dohrmann y S Y Goldsmith J E Hurtado, R D Robinett. Decentralized control for a swarm of vehicles performing source localization. *Journal of Intelligent and Robotic Systems*, 41:1–18, 2004.

[69] C Favre R Falconi y A Martinoli J pugh, X Raemy. A fast onboard relative positioning module for multirobot systems. *IEEE/ASME Transaction on Mechatronics*, 14:2009, 151-162.

[70] J Jackson. Microsoft robotics studio: a technical introduction. *IEEE Robotics and automation magazine*, 14:82–87, 2007.

[71] McLurkin JD. Stupid robot tricks:a behavior based distributed algorithm library for programmin swarm of robots. *Massachusetts: Masschusetts Institute of Technology*, 2004.

[72] Jes SP Thomas GP Rita C David CQ et al. Jha S, Casey-Ford RG. The queen is not a pacemaker in the small-colony wasps polistes instabilis and p dominulus. *Anim Behav*,

71:1197–203, 2006.

[73] S.V. Viscido y D. Grunbaum J.K. Parrish. Self-organized fish schools: an examination of emergent properties. *Biol Bull*, 202:296–305, 2002.

[74] Duc Do K. Formation tracking control of unicycle-type mobile robots with limited sensing ranges. *IEEE transactions on control systems technology*, 16:527–38, 2008.

[75] S.A. Kauffman. *The Origins of order:Self organization and Selection in Evolution.* Oxford University, Nueva York, 1st edition, 1993.

[76] Büchner L. La vie psychique des bêtes. *Paris: C Reinwald*, 1881.

[77] N Michael y V Kumar L Chaimowicz. Controlling swarms of robots using interpolated implici functions. *Proceedings od the IEEE International Conference on Robotics and Automation (ICRA Â´05)*, pages 2487–2492, 2005.

[78] C J J Paredis y p k Khosla L E Navarro-Serment. A beacon system for the localization of distributed robotic teams. *Proceedings of the international conference on field and service robots*, pages 232–237, 1999.

[79] D Nardi y M Salerno L Iocchi. Reactivity and deliberation: a survey on multi-robot systems. *Balancing Reactivity and social deliberation in multi-agent systems. From Robocup to real-world applications*, pages 9–32, 2001.

[80] Torben Lenau and Thomas Hesselberg. *Engineered Biomimicry.* Elsevier, Denmark, 3nd edition, 2013.

[81] R. Candelier A. Cavagna E. Cisbani I. Giardina A. Orlandi G. Parisi A. Procaccini M. Viale y V. Zdravkovic M. Ballerini, N. Cabibbo. Empirical investigation of starling flocks: a benchmark study in collective animal behavior. *Anim Behav*, 76:201–215, 2008.

[82] Sousa P Faria R Marques L Marjovi A, Nunes J. An olfactory based robto swarm navigation method. *IEEE international conference on robotics and automation, IEEE*, 6234:155–66, 2010.

[83] J McLurkin. Stupid robot tricks: a behavior-based distributed algorithm library for programming swarms of robots [m.s. thesis], massachusetts institute of technology. 2004.

[84] Rui P. Rocha Nuno M.F. Ferreira Micael S. Couceiro, Patricia A. Vargas. Benchmark of swarm robotic distributed techniques in a search task. *Robotics and autonomous systems*, 62:200–213, 2014.

[85] O Michel. Webots: professional mobile robot simulation. *journal of advanced robotics systems*, 1:39–42, 2004.

[86] Stover SL Shoureshi R Murphy RR, Kravitz J. Mobile robots in mine rescue and

recovery. *Robot Autom Mag IEEE*, 16:91–103, 2009.

[87] S Rutishauser y A Martinoli N Correll. Comparing coordination schemes for miniature robotic swarms: a case study in boundary coverage of regular structures. *Proceedings of the 10th International Symposium on Experimental Robotics (ISER'08), Springer*, 2006.

[88] Vicsek T Nagy M, Daruka I. New aspects of the continuous phase transition in the scalar noise model (snm) of collective motion. *Physica A*, 373:445–454, 2007.

[89] Menon N Narayan V, Ramaswamy S. Long-lived giant number fluctuations ins a swarming granular nematic. *Science*, 317:105–108, 2007.

[90] Aasadpour M Nasseri MA. Control of flocking behavior using informed agents:an experimental study. *IEEE symposium on swarm intelligence*, pages 1–6, 2011.

[91] Alcherio M Nikolaus C. Modeling and analysis of beaconless and beacon-based policies for a swarm-intelligent inspection system. *Proceedings of the 2005 IEEE international conference on robotics and automation. IEEE*, pages 2477 – 82, 2005.

[92] C.A. TOvey y D.L. Hu N.J. Mlot. Fire ants self-assemble into waterproof rafts to survive floods. *Proc Natl Acad Sci*, 108:1669–7673, 2011.

[93] Khatib O. Real-time obstacle avoidance form manipulators and mobile robots. *Int J Robot Res*, 5:90–8, 1986.

[94] A Menciassi et al. P Valdastri, P Corradi. Micromanipulation, communication and swarm intelligent issues in a swarm microrobotic platform. *Robotic and automatous systems*, 54:789–804, 2006.

[95] Deutsch A Voss-Boehme A Peruani F, Klauss T. Traffic jams, gliders and bands in the quest for collective motion of self-propelled particles. *Physical Review Letters*, 106, 2011.

[96] Gambardella LM mondada F Floreano D Nolfi S et al. Pettinaro GC, Kwee IW. Swarm robotics: a different approach to service robotics. *In: Proceedings of the 33rd international smposium on robotics*, pages 71–6, 2002.

[97] Dolado R Quera V, Beltran F S. Flocking behavior: Agent-based simulation and hierarchical leadership. *Journal of Artificial Societies and Social Simulation (ISSN: 1460-7425)*, 13:8, 2010.

[98] Nakisaee A Ranjbar-Sahraei B, Weiss G. A multi-robot coverage approach based on stigmergic communication. *Multiagent system technologies. Lecture notes in computer science*, 7598:126–38, 2012.

[99] Gamboa GJ Reeve HK. Queen regulation of worker foraging in paper wasps: a social feedback control system (polistesfuscatus, hymenoptera vispedae). *Behaviour*, 102:147–67, 1987.

[100] Reynolds. Boids-background and update. http://www.red3d.com/cwr/boids/, accessed 30 november 2014.

[101] Gaudiano P Rothermich J, Ecemis M. Distributed localization and mapping with a robotic swarm. *swarm robotics, lecture notes in computer science, Springer*, 3342:58–69, 2000.

[102] Alessandro S. Self-organizing flocking in behaviorally heterogeneous swarms. *Bruxelles: UniversitéLibre de Bruxelles*, 2011.

[103] Alexander S. Roombots: Design and implementation of a modular robot for reconfiguration and locomotion. *Lausanne: École Poly technique Fédérale de Lausanne*, 2010.

[104] M A Lapoint M Gini N Papanikolopoulos y J Budenske S Damer, L Ludwing. Dispersion and exploration algorithms for robots in unknown environments. *Proceedings of the Unmanned Systems Technology VIII*, 6230, 2006.

[105] J. Gautrais S. Garnier and G. Theraulaz. The biological principles of swarm intelligence. *Swarm Intelligence*, 1:3–31, 2007.

[106] J. Gautrais y G. Theraulaz S. Garnier. The biologial principles of swarm intelligence. *Swarm Intell*, 1:3–31, 2007.

[107] J Gautrais y G Theraulaz S. Garnier. The biological principles of swarm intelligence. *Swarm intelligence*, 1:3–31, 2007.

[108] R Jeason et al. S Garnier, C Jost. Aggregation behaviour as a source of collective decision in a group of cockroach-like-robots. *Lecture notes in Computer Science*, 3630:169–178, 2005.

[109] O Kornienko y P Levi S Kornienko. Minimalistic approach towards communication and perception in micro-robotic swarm. *Proceedings of the IEEE IRS/RSJ International conference on intelligent robots and systems (IORS 05)*, pages 2228–2234, 2005.

[110] A Campo y M Dorigo S Nouyan. Path formation in a robot swarm:self organized strategies to find your way home. *Swarm intelligence*, 2:1–23, 2008.

[111] UmitUyar M Conner M Holkelek I Bertoli G et al. SafakSahin C, Urrea E. Self-deployment of mobile agents in manets form military applications. *New York: The City College of New York*, 2008.

[112] E Sahin. Swarm robotics: from sources of inspiration to domains of application. *Swarm robotics workshop: state of the art survey*, 3342:10–20, 2005.

[113] T.D. Seeley. Honeybee democracy. *Princeton University Press, NJ, USA*, 2010.

[114] C. A. G. Soerensen and A. Zavala-Rio. An introduction to swarm robotics. *Hindawi Publishing Corporation*, 2013:1–11, 2013.

[115] Heil R Kerr W Hettiarachi S. Spears WM, Spears DF. An overview of physicomimetics. *Swarm robotics, lecture notes in computer science*, 3342:84–97, 2005.

[116] Fua CH Sshuzhi SG. Queues and artificial potential trenches for multirobots formations. *IEEE Trans Robot*, 21:646–56, 2005.

[117] Zufferey J Floreano D Stirling T, Roberts J. Indoor navigation with a swarm of flying robots. *IEEE international conference on robotics and automation, IEEE*, pages 4641–7, 2012.

[118] James R Couzin I Ward A Sumpter D, Krause J. Consensous decision making by fish. *Current Biology*, 18:1773–1777, 2008.

[119] Vicsek T Szabó P, Nagy M. Transitions in a self-propelled-particles model with coupling of acceleration. *Physical Review E*, 79, 2009.

[120] Anna Zafeiris Tamás Vicsek. Collective motion. *Elsevier*, 517:71–140, 2012.

[121] Eshel Ben-Jacob Inon Cohen Tamás Vicsek, András Czirók and Ofer Shochet. Novel type of phase transition in a system of self-driven particles. *Physical review letters*, 75:1226–1230, 1995.

[122] Xu GH Tan M, Fan Y. Research on control and co-operation for swarm robot systems. *Robot*, 23:178–82, 2001.

[123] R.Z. Brown y F.C. Brown T.C. Schneirla. The bovouac or temporary nest as an adaptative factor in certain terrestrial species of army ants. *Ecol Monogr*, 24:269–296, 1954.

[124] T. Schlegel P.M. Hogan N.R. Franks y J.A.R. Marshall T.D. Seeley, P.K. Visscher. Stop signals provide cross-inhibition in collective decision-making by honeybee swarms. *Science*, 335:108–111, 2012.

[125] Celikkanat H Gokce F Sahin E Turgut A, Huepe C. Modelling phase transition in self organized mobile robot flocks. *Ant colony optimization and swarm intelligence, lecture notes incomputer science*, 5217:108–19, 2008.

[126] T H Labella E Sahin y M Dorigo V Trianni, R Grob. Evolving aggregation behaviors in a swarm of robots. *in proceedings of the 7th European conference on artificial life (ECAL 03)*, 2801:865–874, 2003.

[127] R Vaughan. Massively multi-robot simulation in stage. *swarm intelligence*, 2:189–208, 2008.

[128] Ben-Jacob E Cohen I I Shochet O Vicsek T, Czirók A. Novel type of phase transition in a system of self-driven particles. *Pshysical Review Letters*, 75:1226, 1995.

[129] Reynolds C W. Flocks, herds and schools: A distributed behavioral model. *Computer Graphics*, pages 25–34, 1987.

[130] D F Spears y R Heil W M Spears. A formal analysis fo potential energy in a multi-agent system. *Proceedings of the 3rd International Workshop on Formal Approaches to Agent-Based Systems (FAABSÂ ´04)*, pages 131–145, 2004.

[131] P M Maxim et al W M Spears, J C Hamann. Where are you? *Lecture notes in computer science*, 4433:129–143, 2007.

[132] R Heil W Kerr y S Hettiarachchi W M Spears, D F Spears. An overview of physico-mimetics. *Swarm Robotics Workshop: State-of-the-Art survey*, 3342:84–97, 2005.

[133] M. Worall. Homeostasis in nature: nest building termites and intelligent buildings. *Intel Buildings*, 3:87–95, 2011.

[134] L Sciavicco y B Siciliano. Modelling and control of robot manipulators. *Springer-Verlag, London Ltd*, 2001.

[135] O Holland y C Melhuish. Stigmergy, self-organization and sorting in collective robotics. *Artificial Life*, 5:173–202, 1999.

[136] Papadopoulos E y Chasparis G. Analysis and model-based control of servomechanisms with friction. *ASME J Dyn Syst, Meas, Control*, 126:911–915, 2004.

[137] E Martinson y D Payton. Lattice formation in mobile autonomous sensor arrays. *Proceedings of the swarm robotic workshop: state-of-the-art survey, E Sahin y W Spears*, 3342:98–111, 2005.

[138] J McLurkin y D Yamins. Dynamic task assignment in robot swarms. *Proceedings of Robotics: Science and systems, Cambridge, Mass, USA, Junio*, 2005.

[139] C R Kube y E Bonabeau. Cooperative transport by ants and robots. *Robotics and Autonomous Systems*, 30:85–101, 2000.

[140] E Bahceci y E Sahin. Evolving aggregation behaviors for swarm robotic systems: a systematics case study. *Proceedings of the IEEE swarm intelligence symposium*, pages 333–340, 2005.

[141] O Soysal y E Sahin. A macroscopic model for self organized aggregation in swarm robotic systme. *Lecture notes in computer science*, 4433:27–42, 2007.

[142] I Navarro y F Matí. A survey of collective movement of mobile robots. *Tech. Rep., Universidad Polotécnica de Madrid*, 2010.

[143] M Lindhé y K H Johansson. A formation control algorithm using voronol regions. *CTS-HYCON Workshop on Nonlinear and Hybrid Control*, 2005.

[144] D V Dimarogonas y K J Kyriakopoulos. Connectedness preserving disrtibuted swarm aggregation for multiple kinematic robots. *IEEE Transactions on Robotics*, 24:1213–1223, 2008.

[145] R Grob y M Dorigo. Towards group transport by swarms of robots. *International Journal of Bio-Inspired Computation*, 1:1–13, 2009.

[146] L Ludwing y M Gini. Robotic swarm dispersion using wireless intensity signals. *Distributed Autonomous Robotic Systems*, 7:135–144, 2007.

[147] C Jones y M J Mataric. Adaptative division of labor in large-scale minimalist multi-robot systems. *Proceedings of the IEEE/RSJ International Conference on Intelliget Robots and Systems, Las Vegas, Nev. USA, Octubre 2003*, pages 1969–1974.

[148] I.D. Couzin y N.R. Franks. Self-organized lane formation and optimized traffic flow in army ants. *Proc R Soc Lond B*, 70:139–146, 2003.

[149] Papadopoulos E y Panagiotis Vartholomeos. Dynamicas, design and simulation of a novel microrobotic platform employing vibration microactuators. *Transactions of the ASME*, 128:122–133, 2006.

[150] G Caprari y R Siegwart. Mobile micro-robots ready to use: alice. *Proceedings of the IEEE IRS/RSJ international conference on intelligent robots and systems (IORS 05)*, pages 3845–3850, 2005.

[151] J.S. Turner y R.C. Soar. Beyond biomimicry: what termites can tell us about realizing the living building. In *Proceedings of the 1st International conference of industrialized, intelligent construction (I3CON) (T. Hansen y J. Ye, eds.)*, Loughborough University, UK, 14-16 May 2008.

[152] T. Nakagaki y R.D. Guy. Intelligent behaviors of amoeboid movement based on complex dynamics of soft matter. *Soft Matter*, 4:57–67, 2008.

[153] A S Fukunaga y A B Kahng Y U Cao. Cooperative mobile robots: antecedents and direction. *Autonomous Robots*, 4:226–234, 1997.

[154] G Lee y Y C Nak. Self-configurable mobile robot swarms with hole repair capability. *Proceedings of the IEEE/RSJ International Conference on Intelligent Robot and Systems (IROS'08)*, pages 1403–1408, 2008.

[155] Escudero C Couzin I D Buhl J Kevrekidis I G Maini P K Sumpterh D J T Yates C A, Erban R. Inherent noise can facilitate coherence in collective swarm motion. *Proceedings of the National Academy of Sciences of the United States of America*, 106:5464–5469, 2009.

[156] Zhong-yang ZHENG Ying TAN. Research advance in swarm robotics. *Defence Technology*, 9:18–39, 2013.

Apéndice A

Anexo A
Programa en C++

*

Función `Rutina_Cubes_Simulacion.m`

```
clc;
Simulacion={Replicant0;Replicant1;Replicant2;Replicant3;Replicant4;...
    Replicant5;Replicant6;Replicant7;Replicant8;Replicant9};
nS=length(Simulacion);
Tracking={JuanI1;JuanI2;JuanIV1;JuanV1;JuanVI1;JuanVII2;JuanIX2;JuanX1;...
    JuanXI1;JuanXII1};
nT=length(Tracking);

bins=100;


%---- trayectorias temporales ----
figure(1)
c=1;
%
for i=1:2
    juan=Tracking{i}*0.0426;
    subplot(1,2,c); plot3(juan(:,2),juan(:,3),juan(:,1)/0.0426)
    grid on
    hold on
    plot3(juan(:,2)-detrend(juan(:,2)),juan(:,3)-detrend(juan(:,3)),...
        juan(:,1)/0.0426,'r')
    zlabel('Tiempo [s]','fontsize',15);
    xlabel('Pos X [cm]','fontsize',15);
    ylabel('Pos Y [cm]','fontsize',15)
    titulo=strcat('Tracking ',num2str(i));
    title(titulo,'fontsize',15)
    c=c+1;
end
```

```
30
31 figure(2)
32 c=1;
33 %
34 for i=3:4
35     juan=Tracking{i}*0.0426;
36     subplot(1,2,c); plot3(juan(:,2),juan(:,3),juan(:,1)/0.0426)
37     grid on
38     hold on
39     plot3(juan(:,2)-detrend(juan(:,2)),juan(:,3)-detrend(juan(:,3))...
40         ,juan(:,1)/0.0426,'r')
41     zlabel('Tiempo [s]','fontsize',15);
42     xlabel('Pos X [cm]','fontsize',15);
43     ylabel('Pos Y [cm]','fontsize',15)
44     titulo=strcat('Tracking ',num2str(i));
45     title(titulo,'fontsize',15)
46     c=c+1;
47 end
48
49
50 figure(3)
51 c=1;
52 %
53 for i=5:6
54     juan=Tracking{i}*0.0426;
55     subplot(1,2,c); plot3(juan(:,2),juan(:,3),juan(:,1)/0.0426)
56     grid on
57     hold on
58     plot3(juan(:,2)-detrend(juan(:,2)),juan(:,3)-detrend(juan(:,3)),...
59         juan(:,1)/0.0426,'r')
60     zlabel('Tiempo [s]','fontsize',15);
61     xlabel('Pos X [cm]','fontsize',15);
62     ylabel('Pos Y [cm]','fontsize',15)
63     titulo=strcat('Tracking ',num2str(i));
64     title(titulo,'fontsize',15)
65     c=c+1;
66 end
67
68
69 figure(4)
70 c=1;
71 %
72 for i=7:8
73     juan=Tracking{i}*0.0426;
74     subplot(1,2,c); plot3(juan(:,2),juan(:,3),juan(:,1)/0.0426)
75     grid on
76     hold on
77     plot3(juan(:,2)-detrend(juan(:,2)),juan(:,3)-detrend(juan(:,3)),...
78         juan(:,1)/0.0426,'r')
79     zlabel('Tiempo [s]','fontsize',15);
80     xlabel('Pos X [cm]','fontsize',15);
81     ylabel('Pos Y [cm]','fontsize',15)
82     titulo=strcat('Tracking ',num2str(i));
83     title(titulo,'fontsize',15)
84     c=c+1;
85 end
```

```
86
87
88  figure(5)
89  c=1;
90  %
91  for i=9:10
92      juan=Tracking{i}*0.0426;
93      subplot(1,2,c); plot3(juan(:,2),juan(:,3),juan(:,1)/0.0426)
94      grid on
95      hold on
96      plot3(juan(:,2)-detrend(juan(:,2)),juan(:,3)-detrend(juan(:,3)),...
97          juan(:,1)/0.0426,'r')
98      zlabel('Tiempo [s]','fontsize',15);
99      xlabel('Pos X [cm]','fontsize',15);
100     ylabel('Pos Y [cm]','fontsize',15)
101     titulo=strcat('Tracking ',num2str(i));
102     title(titulo,'fontsize',15)
103     c=c+1;
104 end
105
106 %Trayectorias temporales Sim_Cube
107
108 figure(6)
109 c=1;
110 %
111 for i=1:2
112     juan=Simulacion{i}*0.00426;
113     subplot(1,2,c); plot3(juan(:,2),juan(:,3),juan(:,1)/0.0426)
114     grid on
115     hold on
116     plot3(juan(:,2)-detrend(juan(:,2)),juan(:,3)-detrend(juan(:,3)),...
117         juan(:,1)/0.0426,'r')
118     zlabel('Tiempo [s]','fontsize',15);
119     xlabel('Pos X [cm]','fontsize',15);
120     ylabel('Pos Y [cm]','fontsize',15)
121     titulo=strcat('Sim Cube ',num2str(i));
122     title(titulo,'fontsize',15)
123     c=c+1;
124 end
125
126 figure(7)
127 c=1;
128 %
129 for i=3:4
130     juan=Simulacion{i}*0.00426;
131     subplot(1,2,c); plot3(juan(:,2),juan(:,3),juan(:,1)/0.0426)
132     grid on
133     hold on
134     plot3(juan(:,2)-detrend(juan(:,2)),juan(:,3)-detrend(juan(:,3)),...
135         juan(:,1)/0.0426,'r')
136     zlabel('Tiempo [s]','fontsize',15);
137     xlabel('Pos X [cm]','fontsize',15);
138     ylabel('Pos Y [cm]','fontsize',15)
139     titulo=strcat('Sim Cube ',num2str(i));
140     title(titulo,'fontsize',15)
141     c=c+1;
```

```
142 end
143
144
145 figure(8)
146 c=1;
147 %
148 for i=5:6
149     juan=Simulacion{i}*0.00426;
150     subplot(1,2,c); plot3(juan(:,2),juan(:,3),juan(:,1)/0.0426)
151     grid on
152     hold on
153     plot3(juan(:,2)-detrend(juan(:,2)),juan(:,3)-detrend(juan(:,3)),...
154         juan(:,1)/0.0426,'r')
155     zlabel('Tiempo [s]','fontsize',15);
156     xlabel('Pos X [cm]','fontsize',15);
157     ylabel('Pos Y [cm]','fontsize',15)
158     titulo=strcat('Sim Cube ',num2str(i));
159     title(titulo,'fontsize',15)
160     c=c+1;
161 end
162
163
164 figure(9)
165 c=1;
166 %
167 for i=7:8
168     juan=Simulacion{i}*0.00426;
169     subplot(1,2,c); plot3(juan(:,2),juan(:,3),juan(:,1)/0.0426)
170     grid on
171     hold on
172     plot3(juan(:,2)-detrend(juan(:,2)),juan(:,3)-detrend(juan(:,3)),...
173         juan(:,1)/0.0426,'r')
174     zlabel('Tiempo [s]','fontsize',15);
175     xlabel('Pos X [cm]','fontsize',15);
176     ylabel('Pos Y [cm]','fontsize',15)
177     titulo=strcat('Sim Cube ',num2str(i));
178     title(titulo,'fontsize',15)
179     c=c+1;
180 end
181
182
183 figure(10)
184 c=1;
185 %
186 for i=9:10
187     juan=Simulacion{i}*0.00426;
188     subplot(1,2,c); plot3(juan(:,2),juan(:,3),juan(:,1)/0.0426)
189     grid on
190     hold on
191     plot3(juan(:,2)-detrend(juan(:,2)),juan(:,3)-detrend(juan(:,3)),...
192         juan(:,1)/0.0426,'r')
193     zlabel('Tiempo [s]','fontsize',15);
194     xlabel('Pos X [cm]','fontsize',15);
195     ylabel('Pos Y[cm]','fontsize',15)
196     titulo=strcat('Sim Cube ',num2str(i));
197     title(titulo,'fontsize',15)
```

```
198     c=c+1;
199 end
200 
201 
202 %Determinar la muestra con el largo minimo Cubes
203     largo_min_Cube=100000000000;
204     for i=1:nT
205         Cube=Tracking{i};
206         if (length( Cube ) < largo_min_Cube)
207                 largo_min_Cube = length( Cube );
208         end
209     end
210 
211 %Igualar el largo de las tomas Cubes
212     Tomas_largo_comun = {};
213     for i=1:nT
214         Cube=Tracking{i};
215         if (length(Cube)>=largo_min_Cube)
216             Tomas_largo_comun{i} = zeros(largo_min_Cube,13);
217             Tomas_largo_comun{i}= Cube(1:largo_min_Cube, 1:13);
218         end
219     end
220 
221 Tracking = Tomas_largo_comun;
222 
223 %Determinar la muestra con el largo minimo Simulacion Cubes
224     largo_min_SimCube=10000000000;
225     for i=1:nS
226         Cube=Simulacion{i};
227         if (length( Cube ) < largo_min_SimCube)
228                 largo_min_SimCube = length( Cube );
229         end
230     end
231 
232 %Igualar el largo de las tomas Simulacion Cubes
233     Tomas_largo_comun_SimCubes = {};
234     for i=1:nS
235         Cube=Simulacion{i};
236         if (length(Cube)>=largo_min_SimCube)
237             Tomas_largo_comun_SimCubes{i} = zeros(largo_min_SimCube,11);
238             Tomas_largo_comun_SimCubes{i} ...
239            (1:largo_min_SimCube, 1:11) = Cube(1:largo_min_SimCube, 1:11);
240         end
241     end
242 
243 Simulacion = Tomas_largo_comun_SimCubes;
244 
245 %---- desplazamiento medio ----
246 figure(11)
247 for i=1:nT
248     juan=Tracking{i};
249     [desplazamiento_medio,taus]=...
250         calcDeltaW(juan(:,2)*0.0426,juan(:,3)*0.0426);
251     loglog(taus,desplazamiento_medio)
252     axis([10^-3 10^3 10^-2 10^2]);
253     grid on
```

```
    hold on
    xlabel('\tau [s]','fontsize',15);
    ylabel('Desp \Delta r(\tau) [cm]','fontsize',15);
    titulo=strcat('Desplazamientos Cubes');
    title(titulo,'fontsize',15)
    desp_med{i}=desplazamiento_medio;
    end


figure(12)
for i=1:nS
    juan=Simulacion{i};
    [desplazamiento_medio,taus]=calcDeltaW(juan(:,2)*0.00426,...
        juan(:,3)*0.00426);
    loglog(taus,desplazamiento_medio)
    axis([10^-3 10^3 10^-2 10^2]);
    grid on
    hold on
    xlabel('\tau [s]','fontsize',15);
    ylabel('Desp \Delta r(\tau) [cm]','fontsize',15)
    titulo=strcat('Desplazamientos simulacion Cubes');
    title(titulo,'fontsize',15)
    desp_med_Sim_Cube{i}=desplazamiento_medio;
end

%------ Promediar desplazamietos ----
Suma_desp_medio = zeros(1,largo_min_Cube);
for i=1:nT
Suma_desp_medio=Suma_desp_medio+desp_med{i};
end
Desp_medio=Suma_desp_medio/10;


Suma_desp_medio_Sim_Cube = zeros(1,largo_min_SimCube);
for i=1:nT
Suma_desp_medio_Sim_Cube=Suma_desp_medio_Sim_Cube+desp_med_Sim_Cube{i};
end
Desp_medio_Sim_Cube=Suma_desp_medio_Sim_Cube/10;

figure(13)
x1=taus(1:largo_min_Cube);
y1=Desp_medio;
x2=taus(1:largo_min_SimCube);
y2=Desp_medio_Sim_Cube;

%ajuste lineal 1
p11=log(x1(5:round(length(x1)/2)));
p12=log(y1(5:round(length(y1)/2)));
p1=polyfit(p11,p12,1);
tau1=p1(1);
k1=exp(p1(2));
%ajuste lineal 2
p21=log(x2(5:round(length(x2)/2)));
p22=log(y2(5:round(length(y2)/2)));
p2=polyfit(p21,p22,1);
tau2=p2(1);
```

```
k2=exp(p2(2));

%graficar
loglog(x1,y1,'--bo',x1,k1*x1.^tau1,'--ro',x2,y2,...
    ':b*',x2,k2*x2.^tau2,':r*');
legend('Cube',sprintf('Ajuste Cube: y = %.1f * x^{%.2f}',...
    k1, tau1),'Sim Cube',...
    sprintf('Ajuste Sim Cube: y = %.1f * x^{%.2f}', k2, tau2));
title('Desplazamiento medio Cube vs. Sim. Cube','fontsize',15)
xlabel('\tau [s]','fontsize',15);
ylabel('Desp med \Delta r(\tau)','fontsize',15);
grid on;
hold on;


%---- dX, dY, Rapidez ----
%----  Tracking ----
figure(14)
Suma_vel=zeros(length(Tracking{1}),1);
Suma_dx=Suma_vel;
Suma_dy=Suma_vel;
for i=1:nT
    juan=Tracking{i};
    subplot(1,3,1);
    plot(juan(:,1),juan(:,4)*0.0426)   %*120/20
    axis([0 100 -2 2]);
    grid on
    hold on; %plot(juan(:,1),juan(:,11),'r'); hold off
    xlabel('Tiempo [s]','fontsize',15);
    ylabel('\Delta X [cm]','fontsize',15);
    titulo=strcat('\Delta X Cubes');
    title(titulo,'fontsize',15)
    Suma_dx=Suma_vel+juan(:,4);

    subplot(1,3,2);
    plot(juan(:,1),juan(:,5)*0.0426)  %*120/20
    axis([0 100 -2 2]);
    grid on
    hold on; %plot(juan(:,1),juan(:,12),'r'); hold off
    xlabel('Tiempo [s]','fontsize',15);
    ylabel('\Delta Y [cm]','fontsize',15);
    titulo=strcat('\Delta Y Cubes');
    title(titulo,'fontsize',15)
    Suma_dy=Suma_dy+juan(:,5);

    velocidad=sqrt((juan(:,4)*0.0426).^2+(juan(:,5)*0.0426).^2)./juan(:,1);
    subplot(1,3,3);
    plot(juan(:,1),velocidad,'r')  %*120/20
    axis([0 60 0 2]);
    grid on
    hold on; %plot(juan(:,1),juan(:,12),'r'); hold off
    xlabel('Tiempo [s]','fontsize',15);
    ylabel('Rap [cm/seg]','fontsize',15);
    titulo=strcat('Rapidez Cubes [cm/seg]');
    title(titulo,'fontsize',15)
    Suma_vel=Suma_vel+velocidad;
```

```
end
clear velocidad
figure(15)
%----- Simulacion -----
Suma_vel_sim=zeros(length(Simulacion{1}),1);
Suma_dx_sim=Suma_vel_sim;
Suma_dy_sim=Suma_vel_sim;
for i=1:nS
    juan=Simulacion{i};
    subplot(1,3,1);
    plot(juan(:,1),juan(:,4)*0.00426)
    axis([0 100 -2 2]);
    grid on
    hold on; %plot(juan(:,1),juan(:,9),'r'); hold off
    xlabel('Tiempo [s]','fontsize',15);
    ylabel('\Delta X [cm]','fontsize',15);
    titulo=strcat('\Delta X Sim. Cube');
    title(titulo,'fontsize',15)
    Suma_dx_sim=Suma_dx_sim+juan(:,4)*0.00426;

    subplot(1,3,2);
    plot(juan(:,1),juan(:,5)*0.00426)
    axis([0 100 -2 2]);
    grid on
    hold on; %plot(juan(:,1),juan(:,10),'r'); hold off
    xlabel('Tiempo [s]','fontsize',15);
    ylabel('\Delta Y [cm]','fontsize',15);
    titulo=strcat('\Delta Y Sim. Cube');
    title(titulo,'fontsize',15)
    Suma_dy_sim=Suma_dy_sim+juan(:,5)/30;

    velocidad=sqrt((juan(:,4)*0.00426).^2+ ...
        (juan(:,5)*0.00426).^2)./juan(:,1);
    subplot(1,3,3);
    plot(juan(:,1),velocidad,'r')
    axis([0 60 0 2]);
    grid on
    hold on
    xlabel('Tiempo [s]','fontsize',15);
    ylabel('Rap [cm/seg]','fontsize',15);
    titulo=strcat('Rapidez Sim. Cube [cm/seg]');
    title(titulo,'fontsize',15)
    Suma_vel_sim=Suma_vel_sim + velocidad;
end


Promedio_dx = Suma_dx/10;
Promedio_dy = Suma_dy/10;
Promedio_vel = Suma_vel/10;
Promedio_dx_sim = Suma_dx_sim/10;
Promedio_dy_sim = Suma_dy_sim/10;
Promedio_vel_sim = Suma_vel_sim/10;

figure(16)
subplot(1,3,1);
plot(Tracking{1}(:,1),Promedio_dx,'r',...
```

```
422     Simulacion{1}(:,1),Promedio_dx_sim,'b');
423 grid on
424 xlabel('Tiempo [s]','fontsize',15);
425 ylabel('\Delta x [cm]','fontsize',15);
426 axis([0 250 -2 2]);
427 title('\Delta x Cubes vs. Sim. Cubes','fontsize',15);
428 legend('Cube','Sim. Cube')
429 subplot(1,3,2);
430 plot(Tracking{1}(:,1),Promedio_dy,'r',...
431     Simulacion{1}(:,1),Promedio_dy_sim,'b');
432 grid on
433 xlabel('Tiempo [s]','fontsize',15)
434 ylabel('\Delta y [cm]','fontsize',15)
435 axis([0 250 -10 10]);
436 title('\Delta y Cubes vs. Sim. Cubes','fontsize',15)
437 legend('Cube','Sim. Cube')
438 subplot(1,3,3);
439 plot(Tracking{1}(:,1),Promedio_vel,'r',...
440     Simulacion{1}(:,1),Promedio_vel_sim,'b');
441 grid on
442 xlabel('Tiempo [s]','fontsize',15)
443 ylabel('Rap [cm/seg]','fontsize',15)
444 axis([0 100 0 2]);
445 title('Rapidez Cubes vs. Sim. Cubes','fontsize',15)
446 legend('Cube','Sim. Cube')
447
448 figure(17) % PDF dX, PDF dY tracking
449 Savex_Cube={};
450 Savey_Cube={};
451 auxX=[];
452 auxY=[];
453 for i=1:nT
454     juan=Tracking{i}*0.0426;
455     auxX=[auxX, juan(:,4)'];
456     auxY=[auxX, juan(:,5)'];
457     [myhistnc,absmyhistnc,myhist,absmyhist]= ...
458         calculoPDFadqui(juan(:,4),bins);
459     subplot(2,1,1); plot(absmyhistnc,myhist)
460     xlabel('\Delta X [cm]','fontsize',15);
461     ylabel('PDF \Delta X','fontsize',15);
462     grid on
463     hold on
464     titulo=strcat('PDF eje X Cube');
465     title(titulo,'fontsize',15)
466 Savex_Cube{i}=myhist;
467     absmyhistx_Cube = absmyhist;
468     [myhistnc,absmyhistnc,myhist,absmyhist]= ...
469         calculoPDFadqui(juan(:,5),bins);
470     subplot(2,1,2); plot(absmyhistnc,myhist)
471     xlabel('\Delta Y [cm]','fontsize',15);
472     ylabel('PDF \Delta Y','fontsize',15);
473     grid on
474     hold on
475     titulo=strcat('PDF eje Y Cube');
476     title(titulo,'fontsize',15)
477 Savey_Cube{i}=myhist;
```

```
    absmyhisty_Cube = absmyhist;
end

figure(18)  % PDF dX, PDF dY Simulacion
Savex_SimCube={};
Savey_SimCube={};
auxX=[];
auxY=[];
for i=1:nS
    juan=Simulacion{i}*0.00426;
    auxX=[auxX, juan(:,4)'];
    auxY=[auxX, juan(:,5)'];
    [myhistnc,absmyhistnc,myhist,absmyhist]= ...
        calculoPDFadqui(juan(:,4),bins);
    subplot(2,1,1); plot(absmyhistnc,myhist)
    xlabel('\Delta X [cm]','fontsize',15);
    ylabel('PDF \Delta X','fontsize',15);
    grid on
    hold on
    titulo=strcat('PDF eje X sim Cube');
    title(titulo,'fontsize',15)
    Savex_SimCube{i}=myhist;
    absmyhistx_SimCube = absmyhist;
    [myhistnc,absmyhistnc,myhist,absmyhist]= ...
        calculoPDFadqui(juan(:,5),bins);
    subplot(2,1,2); plot(absmyhistnc,myhist)
    xlabel('\Delta Y [cm]','fontsize',15);
    ylabel('PDF \Delta Y','fontsize',15);
    grid on
    hold on
    titulo=strcat('PDF eje Y sim Cube');
    title(titulo,'fontsize',15)
    Savey_SimCube{i}=myhist;
    absmyhisty_SimCube = absmyhist;
end

%Comparacion Cube vs Sim Cube

Sumax_Cube=zeros(length(Savex_Cube{1}),1);
Sumay_Cube=zeros(length(Savex_Cube{1}),1);
Sumax_Simulacion=zeros(length(Savex_SimCube{1}),1);
Sumay_Simulacion=zeros(length(Savex_SimCube{1}),1);
for i=1:10
Sumax_Cube = Sumax_Cube + Savex_Cube{i}';
Sumay_Cube = Sumay_Cube + Savey_Cube{i}';
Sumax_Simulacion = Sumax_Simulacion + Savex_SimCube{i}';
Sumay_Simulacion = Sumay_Simulacion + Savey_SimCube{i}';
end

Promx_Cube=Sumax_Cube/10;
Promy_Cube=Sumay_Cube/10;
Promx_SimCube=Sumax_Simulacion/10;
Promy_SimCube=Sumay_Simulacion/10;

% Comparacion PDF dX, PDF dY Cube vs. Sim_Cube
figure(19)
```

```
534
535     subplot(2,1,1);
536     plot(absmyhistx_Cube,Promx_Cube,'b',...
537         absmyhistx_SimCube,Promx_SimCube,'r')
538     xlabel('\Delta X [cm]','fontsize',15);
539     ylabel('PDF \Delta X','fontsize',15);
540     grid on
541     hold on
542     titulo=strcat('PDF eje X Cube vs. Sim. Cube');
543     title(titulo,'fontsize',15)
544     legend('Cube', 'Sim Cube');
545
546
547     [myhistnc,absmyhistnc,myhist,absmyhist]= ...
548         calculoPDFadqui(juan(:,5),bins);
549     subplot(2,1,2);
550     plot(absmyhisty_Cube,Promy_Cube,'b',...
551         absmyhisty_SimCube,Promy_SimCube,'r')
552     xlabel('\Delta Y [cm]','fontsize',15);
553     ylabel('PDF \Delta Y','fontsize',15);
554     grid on
555     hold on
556     titulo=strcat('PDF eje X Cube vs. Sim. Cube');
557     title(titulo,'fontsize',15)
558     legend('Cube', 'Sim Cube');
559
560     %--- orientaci?n ----
561 % PDF angulo Cube
562 figure(20)
563 auxO=[];
564 auxD=[];
565 for i=1:nT
566     juan=Tracking{i}*0.0426;
567     auxO=[auxO, juan(:,13)'];
568     auxD=[auxD, juan(:,6)'];
569     [myhistnc,absmyhistnc,myhist,absmyhist]= ...
570         calculoPDFadqui(juan(:,6),bins);
571     subplot(2,1,1);
572     plot(absmyhistnc,myhistnc)
573     axis([-0.1 0.1 0 25]);
574     grid on
575     hold on
576     xlabel('\Phi','fontsize',15);
577     ylabel('PDF \Phi desp','fontsize',15);
578     titulo=strcat('PDF \Phi desp. Cube');
579     title(titulo,'fontsize',15)
580
581
582     [myhistnc,absmyhistnc,myhist,absmyhist]=...
583         calculoPDFadqui(juan(:,13),bins);
584     subplot(2,1,2);
585     plot(absmyhistnc,myhistnc)
586     axis([-0.1 0.1 0 25]);
587     grid on
588     hold on
589     xlabel('\Phi','fontsize',15)
```

```
    ylabel('PDF \Phi orientacion Cube','fontsize',15);
    titulo=strcat('PDF \Phi orientacion Cube');
    title(titulo,'fontsize',15)
end


%PDF angulo Sim Cube
figure(21)
auxP=[];
auxQ=[];
for i=1:nT
    juan=Simulacion{i}*0.00426;
    auxP=[auxP, juan(:,11)'];
    auxQ=[auxQ, juan(:,6)'];
    [myhistnc,absmyhistnc,myhist,absmyhist]= ...
        calculoPDFadqui(juan(:,6),bins);
    subplot(2,1,1);
    plot(absmyhistnc,myhistnc)
    axis([-0.01 0.01 0 150]);
    grid on
    hold on
    xlabel('\Phi','fontsize',15);
    ylabel('PDF \Phi desp','fontsize',15);
    axis([-0.1 0.1 0 25]);
    titulo=strcat('PDF \Phi desp. Sim. Cube');
    title(titulo,'fontsize',15)


    [myhistnc,absmyhistnc,myhist,absmyhist]= ...
        calculoPDFadqui(juan(:,11),bins);
    subplot(2,1,2);
    plot(absmyhistnc,myhistnc)
    axis([-0.01 0.01 0 150]);
    grid on
    hold on
    xlabel('\Phi','fontsize',15)
    ylabel('PDF \Phi orientacion Cube','fontsize',15);
    titulo=strcat('PDF \Phi orientacion Sim. Cube');
    title(titulo,'fontsize',15)
end

figure(22)
[myhistnc_Cubedsp,absmyhistnc_Cubedsp,myhist_Cubedsp,...
    absmyhist_Cubedsp]=calculoPDFadqui(auxD,bins);
[myhistnc_SimCubedsp,absmyhistnc_SimCubedsp,myhist_SimCubedsp,...
    absmyhist_SimCubedsp]=calculoPDFadqui(auxP,bins);
subplot(2,1,1);
plot(absmyhistnc_Cubedsp,myhistnc_Cubedsp,'b',...
    absmyhistnc_SimCubedsp,myhistnc_SimCubedsp,'r');
axis([-0.1 0.1 0 100]);
grid on
hold on
ylabel('PDF \Phi desp','fontsize',15);
title('\Phi desp. Cube vs. Sim. Cube','fontsize',15)
legend('Cube', 'Sim. Cube');
```

```
[myhistnc_Cubeor,absmyhistnc_Cubeor,...
    myhist_Cubeor,absmyhist_Cubeor]=calculoPDFadqui(auxO,bins);
[myhistnc_SimCubeor,absmyhistnc_SimCubeor,...
    myhist_SimCubeor,absmyhist_SimCubeor]=calculoPDFadqui(auxQ,bins);
subplot(2,1,2);
plot(absmyhistnc_Cubeor,myhistnc_Cubeor,'b',...
    absmyhistnc_SimCubeor,myhistnc_SimCubeor,'r');
axis([-0.1 0.1 0 100]);
grid on
hold on
ylabel('PDF \Phi orientacion','fontsize',15);
title('\Phi or. Cube vs. Sim. Cube','fontsize',15)
legend('Cube', 'Sim. Cube');
clear auxX auxY myhistnc absmyhistnc myhist absmyhist

%orientacion vs desplazamiento
%Cubes
Sumadsp_Cube = zeros(length(Tracking{1}),1);
Sumaor_Cube = Sumadsp_Cube;
for i=1:nT
    juan=Tracking{i}*0.0426;
    Sumadsp_Cube = Sumadsp_Cube + juan(:,6);
    Sumaor_Cube = Sumaor_Cube + juan(:,13);
end
Promdsp_Cube=Sumadsp_Cube/10;
Promor_Cube=Sumaor_Cube/10;

figure(23)
plot(juan(:,1),Promdsp_Cube,'b',juan(:,1),Promor_Cube,'r')
    grid on
    hold on;
    xlabel('Tiempo [s]','fontsize',15);
    ylabel('\Phi','fontsize',15);
    titulo=strcat('\Phi Desp vs \Phi Orientacion Cube');
    title(titulo,'fontsize',15)
    legend('\Phi desp.','\Phi or.');

%Sim Cubes
Sumadsp_SimCube = zeros(length(Simulacion{1}(:,1)),1);
Sumaor_SimCube = Sumadsp_SimCube;

for i=1:nS
    juan=Simulacion{i}*0.00426;
    Sumadsp_SimCube = Sumadsp_SimCube + juan(:,6);
    Sumaor_SimCube = Sumaor_SimCube + juan(:,11);
 end
Promdsp_SimCube=Sumadsp_SimCube/10;
Promor_SimCube=Sumaor_SimCube/10;

figure(24)
plot(juan(:,1),Promdsp_SimCube,'b',juan(:,1),Promor_SimCube,'r')
    grid on
    hold on;
    xlabel('Tiempo [s]','fontsize',15);
    ylabel('\Phi','fontsize',15);
    titulo=strcat('\Phi Desp vs \Phi Orientacion Sim. Cube');
```

```
    title(titulo,'fontsize',15)
    legend('\Phi desp.','\Phi or.');
```

Función Rutina_Cheavibot_Lineal.m

```
clc;
bins=100;
Lineal={Cheavibot_lineal1,Cheavibot_lineal2,...
    Cheavibot_lineal3,Cheavibot_lineal4,Cheavibot_lineal5};
nL=length(Lineal);
Trayectoria_U=matricear2(U(:,1),U(:,2));
Trayectoria_L=matricear2(L(:,1),L(:,2));
%Trayectorias temporales
figure(1)
c=1;
for i=1:5
    Cheavibot=Lineal{i};
    subplot(3,2,c); plot(Cheavibot(:,2),Cheavibot(:,3))
    grid on
    hold on
    plot(Cheavibot(:,2)-detrend(Cheavibot(:,2)),...
        Cheavibot(:,3)-detrend(Cheavibot(:,3)),'r')
    xlabel('Pos X [cm]','fontsize',15); ylabel('Pos Y [cm]','fontsize',15)
    titulo=strcat('CheaViBot ',num2str(i));
    title(titulo,'fontsize',15)
    c=c+1;
end

figure(2)
for i=5:5
    Cheavibot=Lineal{i};
    plot(Cheavibot(:,2),Cheavibot(:,3))
    grid on
    hold on
    plot(Cheavibot(:,2)-detrend(Cheavibot(:,2)),...
        Cheavibot(:,3)-detrend(Cheavibot(:,3)),'r')
    xlabel('Pos X [cm]','fontsize',15);
    ylabel('Pos Y [cm]','fontsize',15)
    titulo=strcat('CheaViBot ',num2str(i));
    title(titulo,'fontsize',15)
end

figure(3)
plot(Trayectoria_U(:,2),Trayectoria_U(:,3));
grid on
xlabel('\Delta X [cm]','fontsize',15)
ylabel('\Delta Y [cm]','fontsize',15)
title('Trayectoria en U','fontsize',15);

figure(4)
plot(Trayectoria_L(:,2),Trayectoria_L(:,3));
grid on
```

```
xlabel('\Delta X [cm]','fontsize',15);
ylabel('\Delta Y [cm]','fontsize',15);
title('Trayectoria en L','fontsize',15);

%Determinar la muestra con el largo minimo Cubes
    largo_min_Cube=100000000000;
    for i=1:nL
        Cube=Lineal{i};
        if (length( Cube ) < largo_min_Cube)
                largo_min_Cube = length( Cube );
        end
    end

%Igualar el largo de las tomas Cubes
    Tomas_largo_comun = {};
    for i=1:nL
        Cube=Lineal{i};
        if (length(Cube)>=largo_min_Cube)
            Tomas_largo_comun{i} = zeros(largo_min_Cube,6);
            Tomas_largo_comun{i}= Cube(1:largo_min_Cube, 1:6);
        end
    end

Lineal = Tomas_largo_comun;


%---- desplazamiento medio ----

%Determinar la muestra con el largo min
largo_min=1000000000000000000000;
    for j=1:nL
        if (length( Lineal{j} ) < largo_min)
                largo_min = length( Lineal{j} );
        end
    end

%Igualar el largo de las tomas
    Tomas_largo_comun = {};
    Centro_aux = zeros(largo_min,2);
    Centro_geom = zeros(largo_min,2);
    for k=1:nL
        if (length(Lineal{k})>=largo_min)
            Tomas_largo_comun{k,1} = zeros(2,largo_min);
            Tomas_largo_comun{k,1} (1:largo_min, 1:2) = ...
                Lineal{k}(1:largo_min, 1:2);
        end
    end
Suma_desp_medio = zeros(1,largo_min);
figure(5)
for i=1:nL
    Cheavibot=Tomas_largo_comun{i};
    [desplazamiento_medio,taus]=calcDeltaW(Cheavibot(:,2),Cheavibot(:,3));
    loglog(taus,desplazamiento_medio)
    grid on
    hold on
    xlabel('\tau [s]','fontsize',15);
```

```
104     ylabel('Desp \Delta X(\tau) [cm]','fontsize',15)
105     titulo=strcat('Desplazamiento \Delta X(\tau) [cm] CheaViBot');
106     title(titulo,'fontsize',15)
107     Suma_desp_medio = Suma_desp_medio + desplazamiento_medio;
108 end
109 Promedio_desp_medio=Suma_desp_medio/nL;
110 
111 figure(6)
112 loglog(taus,Promedio_desp_medio)
113 grid on
114 hold on
115 xlabel('\tau [s]'); ylabel('Desp \Delta X(\tau) [cm]')
116 titulo=strcat('Desplazamiento medio \Delta X(\tau) [cm] CheaViBot');
117 title(titulo)
118 
119 %ajuste lineal 1
120 x1=taus(1:largo_min);
121 y1=Promedio_desp_medio;
122 p11=log(x1(5:round(length(x1)/2)));
123 p12=log(y1(5:round(length(y1)/2)));
124 p1=polyfit(p11,p12,1);
125 tau1=p1(1);
126 k1=exp(p1(2));
127 
128 %graficar
129 loglog(x1,y1,'--bo',x1,k1*x1.^tau1,'--ro');
130 legend('CheaViBot',sprintf('Ajuste CheViBot: y = %.1f * x^{%.2f}',...
131     k1, tau1));
132 title('Desplazamiento medio CheaViBot','fontsize',15)
133 xlabel('\tau [s]','fontsize',15);
134 ylabel('Desp med \Delta r(\tau)','fontsize',15);
135 grid on;
136 hold on;
137 
138 %---- dX, dY, Rapidez ----
139 %----  Tracking ----
140 figure(7)
141 Suma_vel=zeros(length(Lineal{1}),1);
142 Suma_dx=Suma_vel;
143 Suma_dy=Suma_vel;
144 for i=1:nL
145     juan=Lineal{i};
146     subplot(1,3,1);
147     plot(juan(:,1),juan(:,4))
148     axis([0 30 -1 1]);
149     grid on
150     hold on;
151     xlabel('Tiempo [s]','fontsize',15);
152     ylabel('\Delta X [cm]','fontsize',15);
153     titulo=strcat('\Delta X CheaViBots');
154     title(titulo,'fontsize',15)
155     Suma_dx=Suma_vel+juan(:,4);
156 
157     subplot(1,3,2);
158     plot(juan(:,1),juan(:,5))
159     axis([0 30 -1 1]);
```

```
    grid on
    hold on;
    xlabel('Tiempo [s]','fontsize',15);
    ylabel('\Delta Y [cm]','fontsize',15);
    titulo=strcat('\Delta Y CheaViBots');
    title(titulo,'fontsize',15)
    Suma_dy=Suma_dy+juan(:,5);

    velocidad=sqrt(juan(:,4).^2+juan(:,5).^2)./juan(:,1);
    subplot(1,3,3);
    plot(juan(:,1),velocidad,'r')
    axis([0 30 0 0.5]);
    grid on
    hold on;
    xlabel('Tiempo [s]','fontsize',15);
    ylabel('Rap [cm/seg]','fontsize',15);
    titulo=strcat('Rapidez CheaViBots [cm/seg]');
    title(titulo,'fontsize',15)
    Suma_vel=Suma_vel+velocidad;
end

%---- desp en ejes coordenados y curva de rapidez
%media para Trayectoria en L ----
figure(8)
for i=1:1
    juan=Trayectoria_L;
    subplot(1,3,1);
    plot(juan(:,1),juan(:,4))   %*120/20
    axis([0 100 -1 1]);
    grid on
    hold on; %plot(juan(:,1),juan(:,11),'r'); hold off
    xlabel('Tiempo [s]','fontsize',15);
    ylabel('\Delta X [cm]','fontsize',15);
    titulo=strcat('\Delta X CheaViBots TL');
    title(titulo,'fontsize',15)

    subplot(1,3,2);
    plot(juan(:,1),juan(:,5)) %*120/20
    axis([0 100 -1 1]);
    grid on
    hold on; %plot(juan(:,1),juan(:,12),'r'); hold off
    xlabel('Tiempo [s]','fontsize',15);
    ylabel('\Delta Y [cm]','fontsize',15);
    titulo=strcat('\Delta Y CheaViBots TL');
    title(titulo,'fontsize',15)

    velocidad=sqrt(juan(:,4).^2+juan(:,5).^2)./juan(:,1);
    subplot(1,3,3);
    plot(juan(:,1),velocidad,'r') %*120/20
    axis([0 100 0 0.5]);
    grid on
    hold on; %plot(juan(:,1),juan(:,12),'r'); hold off
    xlabel('Tiempo [s]','fontsize',15);
    ylabel('Rap [cm/seg]','fontsize',15);
    titulo=strcat('Rapidez CheaViBots TL [cm/seg]');
    title(titulo,'fontsize',15)
```

```
end

%---- desp en ejes coordenados y curva de
%rapidez media para Trayectoria en U ----
figure(9)
for i=1:1
    juan=Trayectoria_U;
    subplot(1,3,1);
    plot(juan(:,1),juan(:,4))   %*120/20
    axis([0 100 -1 1]);
    grid on
    hold on; %plot(juan(:,1),juan(:,11),'r'); hold off
    xlabel('Tiempo [s]','fontsize',15);
    ylabel('\Delta X [cm]','fontsize',15);
    titulo=strcat('\Delta X CheaViBots TU');
    title(titulo,'fontsize',15)

    subplot(1,3,2);
    plot(juan(:,1),juan(:,5))  %*120/20
    axis([0 100 -1 1]);
    grid on
    hold on; %plot(juan(:,1),juan(:,12),'r'); hold off
    xlabel('Tiempo [s]','fontsize',15);
    ylabel('\Delta Y [cm]','fontsize',15);
    titulo=strcat('\Delta Y CheaViBots TU');
    title(titulo,'fontsize',15)

    velocidad=sqrt(juan(:,4).^2+juan(:,5).^2)./juan(:,1);
    subplot(1,3,3);
    plot(juan(:,1),velocidad,'r')  %*120/20
    axis([0 100 0 0.5]);
    grid on
    hold on; %plot(juan(:,1),juan(:,12),'r'); hold off
    xlabel('Tiempo [s]','fontsize',15);
    ylabel('Rap [cm/seg]','fontsize',15);
    titulo=strcat('Rapidez CheaViBots TU [cm/seg]');
    title(titulo,'fontsize',15)

end
```

Función Analisis_Grupal_Juanitos.m

```
  clc;
%Calculo de centro de gravedad
num_cubos = 2;
bins = 300;
A = {};
% for a=4:2:42
for i=1:num_cubos
    %Importar archivo
    aux = i-1;
    filename = strcat('Track_ang_cubo',num2str(aux),'.txt');
    delimiterIn = ' ';
```

```
12      headerlinesIn = 0;
13      b = strcat(num2str(num_cubos),' Juanitos 5_min');
14      A{i,1} = importdata(fullfile(b,filename),delimiterIn,headerlinesIn);
15  end
16
17  %matricear las muestras
18  aux={};
19  for i=1:num_cubos
20  aux{i,1}=matricear2(A{i}(:,1),A{i}(:,2),A{i}(:,3),A{i}(:,4));
21  end
22  A=aux;
23
24  %Determinar la muestra con el largo m?nimo
25      largo_min=10000000000;
26      for j=1:num_cubos
27          if (length( A{j,1}(:,1) ) < largo_min)
28                  largo_min = length( A{j,1}(:,1) );
29          end
30      end
31
32  %Igualar el largo de las tomas
33      B = {};
34      Centro_aux = zeros(largo_min,11);
35      Centro_geom = zeros(largo_min,1);
36      for k=1:num_cubos
37          if (length(A{k,1})>=largo_min)
38              B{k,1} = zeros(largo_min,11);
39              B{k,1} (1:largo_min, 1:11) = A{k,1}(1:largo_min, 1:11);
40          end
41      end
42
43  %Calculo efectivo
44      for i=1:num_cubos
45      Centro_aux (:,1) = Centro_aux(:,1)+B{i,1}(:,2)*0.0426;
46      Centro_aux (:,2) = Centro_aux(:,2)+B{i,1}(:,3)*0.0426;
47      end
48      Centro_geom (:,1) = Centro_aux(:,1)/num_cubos;
49      Centro_geom (:,2) = Centro_aux(:,2)/num_cubos;
50
51  %An?lisis del centro geometrico
52      Centro_geom_matriceado=matricear2(Centro_geom(:,1)*0.0426,...
53          Centro_geom(:,2)*0.0426);
54
55  %Graficar la trayectoria del centro geometrico
56      figure(1)
57      plot(Centro_geom(:,1),Centro_geom(:,2));
58      xlabel('X [cm]','fontsize',15);
59      ylabel('Y [cm]','fontsize',15)
60      titulo=strcat('Trayectorias centros geometricos Sim. Cubes');
61      title(titulo,'fontsize',15)
62      hold on
63
64  %Desplazamiento medio
65  %ajuste lineal
66  figure(2)
67      [desplazamiento_medio,taus]=calcDeltaW(Centro_geom_matriceado(:,2),...
```

```
        Centro_geom_matriceado(:,3));
    loglog(taus,desplazamiento_medio)
    grid on
    hold on
    xlabel('\tau [s]','fontsize',15);
    ylabel('\Delta r(\tau)','fontsize',15)
    axis([10^-2 10^3 10^-1 10^1])
    titulo=strcat('Desplazamiento centros geometricos Sim Cubes');
    title(titulo,'fontsize',15);


x2=taus(1:length(taus));
y2=desplazamiento_medio;
p21=log(x2(5:round(length(x2)/2)));
p22=log(y2(5:round(length(y2)/2)));
p2=polyfit(p21,p22,1);
tau2=p2(1);
k2=exp(p2(2));

%graficar
loglog(x2,k2*x2.^tau2,':r*');
legend('Sim Cube',sprintf('Ajuste Sim Cube: y = %.1f * x^{%.2f}',...
    k2, tau2));

%dX, dY (velocidades)
    figure(3)
    subplot(1,3,1);
    grid on
    hold on;
    plot(Centro_geom_matriceado(:,1),Centro_geom_matriceado(:,4))
    hold on;
    xlabel('Tiempo [seg]','fontsize',15);
    ylabel('\Delta X [cm]','fontsize',15);
    axis([0 200 -3 3])
    titulo=strcat('\Delta X Sim. Cubes');
    title(titulo,'fontsize',15)

    subplot(1,3,2);
    grid on;
    hold on;
    plot(Centro_geom_matriceado(:,1),Centro_geom_matriceado(:,5)) %*120/20
    xlabel('Tiempo [seg]','fontsize',15);
    ylabel('\Delta Y [cm]','fontsize',15);
    axis([0 200 -3 3])
    titulo=strcat('\Delta Y Sim. Cubes');
    title(titulo,'fontsize',15)

    velocidad=sqrt((Centro_geom_matriceado(:,4)).^2+ ...
        (Centro_geom_matriceado(:,5)).^2)./Centro_geom_matriceado(:,1);
    subplot(1,3,3);
    plot(Centro_geom_matriceado(:,1),velocidad,'r') %*120/20
    axis([0 200 0 1]);
    grid on
    hold on; %plot(juan(:,1),juan(:,12),'r'); hold off
    xlabel('Tiempo [s]','fontsize',15);
    ylabel('Rap [cm/seg]','fontsize',15);
```

```
124     titulo=strcat('Rapidez Sim Cube [cm/seg]');
125     title(titulo,'fontsize',15)
126
127
128 % PDF dX, PDF dY tracking
129     figure(4)
130
131     [myhistnc,absmyhistnc,myhist,absmyhist]= ...
132         calculoPDFadqui(Centro_geom_matriceado(:,4),bins);
133     subplot(1,2,1);
134     grid on;
135     hold on;
136     plot(absmyhistnc,myhist)
137     xlabel('\Delta X [cm]','fontsize',15)
138     ylabel('PDF \Delta X','fontsize',15);
139     titulo=strcat('PDF \Delta X Sim. Cubes');
140     title(titulo,'fontsize',15)
141
142
143     [myhistnc,absmyhistnc,myhist,absmyhist]= ...
144         calculoPDFadqui(Centro_geom_matriceado(:,5),bins);
145     subplot(1,2,2);
146     grid on;
147     hold on;
148     plot(absmyhistnc,myhist)
149     xlabel('\Delta Y [cm]','fontsize',15)
150     ylabel('PDF \Delta Y','fontsize',15);
151     titulo=strcat('PDF \Delta Y Sim. Cubes');
152     title(titulo,'fontsize',15)
153
154 % PDF ?ngulo desplazamiento
155     figure(5)
156     [myhistnc,absmyhistnc,myhist,absmyhist]= ...
157         calculoPDFadqui(Centro_geom_matriceado(:,6),bins);
158     plot(absmyhistnc,myhistnc)
159     hold on
160     grid on
161     xlabel('\Phi [grados]','fontsize',15)
162     ylabel('PDF \Phi desplazamiento','fontsize',15);
163     titulo=strcat('PDF \Phi desp. Sim. Cubes');
164     title(titulo,'fontsize',15)
```

Función `calcDeltaW`

```
1   function [W,taus] = calcDeltaW(toma, Y)
2
3 tau_max=length(toma(:,1));
4 taus=1:tau_max;
5 if nargin==1
6     X=toma(:,1);
7     Y=toma(:,2);
8 else
9     X=toma;
```

```
end
W=zeros(1,length(taus));
for tau = taus
    W(tau)=sqrt( (X(tau)-X(1))^2+(Y(tau)-Y(1))^2 );
end
taus=taus/120;
```

Función `matricear.m`

```
function matriz=matricear(X1,Y1,X2,Y2,X3,Y3)

if nargin > 6
    error('matricear:TooManyInputs', ...
        'inputs: X1,Y1,opcional:X2,Y2,X3,Y3');

elseif nargin==2
    %----tiempo y posicion----
    X1=X1*62/0.0264583;
    Y1=Y1*62/0.0264583;
    tiempo=(1:length(X1))/120; %conversiï¿½n a segundos del tiempo (fm=120Hz)

    %----dX, dY, ï¿½ngulo desplazamiento----
    [dX_desplazamiento,dY_desplazamiento,angulo_desplazamiento] = ...
        calcAngDesp(X1,Y1);

    %----desfinicion de matriz de salida----
    matriz=[tiempo', X1, Y1, dX_desplazamiento', dY_desplazamiento',...
        angulo_desplazamiento'];

elseif nargin==6 %tracking
    %----tiempo y posicion----
    tiempo=(1:length(X1))/120; %conversiï¿½n a segundos del tiempo (fm=120Hz)
    X1=X1*62/0.0264583;
    Y1=Y1*62/0.0264583;
    X2=X2*62/0.0264583;
    Y2=Y2*62/0.0264583;
    X3=X3*62/0.0264583;
    Y3=Y3*62/0.0264583;
    %----dX, dY, ï¿½ngulo desplazamiento----
    [dX_desplazamiento,dY_desplazamiento,angulo_desplazamiento] = ...
        calcAngDesp(X1,Y1);

    %----ï¿½ngulo orientaciï¿½n----
    [dX_orientacion, dY_orientacion, angulo_orientacion]=...
        calcAngMarc(X2,Y2,X3,Y3);

    %----desfinicion de matriz de salida----

    matriz=[tiempo', X1, Y1, dX_desplazamiento', dY_desplazamiento',...
        angulo_desplazamiento', ...
        X2, Y2, X3, Y3, dX_orientacion, dY_orientacion,...
        angulo_orientacion'];

```

```
elseif nargin==4 %simulaciones
    %---tiempo y posicion----
    tiempo=(1:length(X1))/300; %conversiï¿½n a segundos del tiempo (fm=120Hz)
    X1=X1*62/0.0264583;
    Y1=Y1*62/0.0264583;
    X2=X2*62/0.0264583;
    Y2=Y2*62/0.0264583;


    %---dX, dY, ï¿½ngulo desplazamiento----

    [dX_desplazamiento,dY_desplazamiento,angulo_desplazamiento] =...
        calcAngDespSimulacion(X1,Y1);

    %---ï¿½ngulo orientaciï¿½n----
    [dX_orientacion, dY_orientacion, angulo_orientacion]=...
        calcAngMarcSimulacion(X1,Y1,X2,Y2);

    %---desfinicion de matriz de salida----

    matriz=[tiempo', X1, Y1, dX_desplazamiento', dY_desplazamiento',...
        angulo_desplazamiento', ...
        X2, Y2, dX_orientacion, dY_orientacion, angulo_orientacion'];

else
    error('matricear:InvalidInput', ...
        'input: X1,Y1,opcional:X2,Y2,X3,Y3');
end
```

Función Voltaje_function.m

```
function v = Voltaje_funcion(t)
a=2.109;
b=-0.2531;
c=1.201;
d=-0.001164;
    if (t==0)
        v = a+b;
        else
        v = a*exp(b*t) + c*exp(d*t);
    end
end
```

Función Corriente_function.m

```
function i = Corriente_funcion(t)
a=0.02271;
b=-0.02457;
c=0.04738;
d=-0.001452;
```

```
    if (t==0)
        i = a+b;
        else
        i = a*exp(b*t) + c*exp(d*t);
    end
end
```

Función calcAngMarcSimulacion.m

```
function [dX, dY, angulos] = calcAngMarcSimulacion(X0,Y0,X1,Y1)
%X0,Y0 punto central; X1,Y1 ancla de masa chica
dX=X1-X0;
dY=Y1-Y0;

angulos=zeros(1,length(dX));
for i=1:length(angulos);
    angulos(i)=radtodeg(angle(dY(i)+dX(i)*j));
end
```

Función calcAngDesp.m

```
function [dX, dY, angulos, rapidez] = calcAngDesp(X,Y)
s=20;  %para reducir ruido

dX=zeros(1,length(X));
dY=dX;
angulos=dX;
rapidez=dX;

for i=s:length(X)-s
    dX(i)=X(i+(s/2))-X(i-(s/2));
    dY(i)=Y(i+(s/2))-Y(i-(s/2));
end
% dX=filtro(dX);  %El (-) no tiene justificacion p
% dY=filtro(dY);

for i=1:length(dX)
    angulos(i)=radtodeg(angle(dY(i)+dX(i)*j));
end
s=20;
%tiempo entre 20 mediciones:
for i=s:length (X)-s
    rapidez(i)= sqrt((dX(i)-dX(i-1))^2-(dY(i)-dY(i-1))^2)/0.1667;
end
end
```

Función Analisis_Grupal_Cubes

```
clc;
 %Calculo de centro de gravedad
A = {VJuanitos,VJuanitos2};
Juanitos={};
bins=300;
for i=1:5
    Juanitos{i}=[A{1}(:,i+(i-1)),A{1}(:,2*i)];
    Juanitos{i+5}=[A{2}(:,i+(i-1)),A{2}(:,2*i)];
end
Juanitos_aux={};
for i = 1:10
Juanitos_aux{i}=matricear2(Juanitos{i}(:,1),Juanitos{i}(:,2));
end
Juanitos=Juanitos_aux;

num_cubos=10;
 %Determinar la muestra con el largo m?nimo
    largo_min=10000000000;
    for j=1:num_cubos
        if (length( Juanitos{j}(:,1) ) < largo_min)
                largo_min = length( Juanitos{j}(:,1) );
        end
    end

 %Igualar el largo de las tomas
    B = {};
    Centro_aux = zeros(largo_min,6);
    Centro_geom = zeros(largo_min,2);
    for k=1:num_cubos
        if (length(Juanitos{k})>=largo_min)
            B{k,1} = zeros(largo_min,6);
            B{k,1} (1:largo_min, 1:6) = Juanitos{k}(1:largo_min, 1:6);
        end
    end
 %Calculo efectivo
    for i=1:num_cubos
    Centro_aux (:,1) = Centro_aux(:,1)+B{i,1}(:,2);
    Centro_aux (:,2) = Centro_aux(:,2)+B{i,1}(:,3);
    end
    Centro_geom (:,1) = Centro_aux(:,1)/num_cubos;
    Centro_geom (:,2) = Centro_aux(:,2)/num_cubos;

 %An?lisis del centro geometrico
    Centro_geom_matriceado=matricear2(Centro_geom(:,1),...
        Centro_geom(:,2));

 %Graficar la trayectoria del centro geometrico
    figure(1)
    plot(Centro_geom(:,1),Centro_geom(:,2));
    xlabel('X [cm]','fontsize',15);
    ylabel('Y [cm]','fontsize',15)
    titulo=strcat('Trayectorias centro geometrico 5 Cubes');
    title(titulo,'fontsize',15)
    hold on

     %Desplazamiento medio
```

```
%ajuste lineal
figure(2)
    [desplazamiento_medio,taus]=calcDeltaW(Centro_geom_matriceado(:,2),...
        Centro_geom_matriceado(:,3));
    loglog(taus,desplazamiento_medio)
    grid on
    hold on
    xlabel('\tau [s]','fontsize',15);
    ylabel('\Delta r(\tau)','fontsize',15)
    %axis([10^-2 10^3 10^-1 10^1])
    titulo=strcat('Desplazamiento centro geometrico 5 Cubes');
    title(titulo,'fontsize',15);


x2=taus(1:length(taus));
y2=desplazamiento_medio;
p21=log(x2(5:round(length(x2)/2)));
p22=log(y2(5:round(length(y2)/2)));
p2=polyfit(p21,p22,1);
tau2=p2(1);
k2=exp(p2(2));

%graficar
loglog(x2,k2*x2.^tau2,':r*');
legend('Sim Cube',sprintf('Ajuste Sim Cube: y = %.1f * x^{%.2f}',...
    k2, tau2));

%dX, dY (velocidades)
    figure(3)
    subplot(1,3,1);
    grid on
    hold on;
    plot(Centro_geom_matriceado(:,1),Centro_geom_matriceado(:,4))
    hold on;
    xlabel('Tiempo [seg]','fontsize',15);
    ylabel('\Delta X [cm]','fontsize',15);
    axis([0 50 -10 10])
    titulo=strcat('\Delta X CG. 5 Cubes');
    title(titulo,'fontsize',15)

    subplot(1,3,2);
    grid on;
    hold on;
    plot(Centro_geom_matriceado(:,1),Centro_geom_matriceado(:,5)) %*120/20
    xlabel('Tiempo [seg]','fontsize',15);
    ylabel('\Delta Y [cm]','fontsize',15);
    axis([0 50 -10 10])
    titulo=strcat('\Delta Y CG. 5 Cubes');
    title(titulo,'fontsize',15)

    velocidad=sqrt((Centro_geom_matriceado(:,4)).^2+ ...
        (Centro_geom_matriceado(:,5)).^2)./Centro_geom_matriceado(:,1);
    subplot(1,3,3);
    plot(Centro_geom_matriceado(:,1),velocidad,'r') %*120/20
    axis([0 50 0 10]);
    grid on
```

```
    hold on; %plot(juan(:,1),juan(:,12),'r'); hold off
    xlabel('Tiempo [s]','fontsize',15);
    ylabel('Rap [cm/seg]','fontsize',15);
    titulo=strcat('Rapidez CG 5 Cubes [cm/seg]');
    title(titulo,'fontsize',15)


% PDF ?ngulo desplazamiento
    figure(4)
    [myhistnc,absmyhistnc,myhist,absmyhist]= ...
        calculoPDFadqui(Centro_geom_matriceado(:,6),bins);
    plot(absmyhistnc,myhistnc)
    hold on
    grid on
    xlabel('\Phi [grados]','fontsize',15)
    ylabel('PDF \Phi desplazamiento','fontsize',15);
    titulo=strcat('PDF \Phi desp. 5 Cubes');
    title(titulo,'fontsize',15)
```

Función Analisis_Grupal_Cubes_iman

```
clc;
%Calculo de centro de gravedad
%A = {VJuanitos_iman,VJuanitos_iman2};
%A={VIIJuanitos_iman,VIIJuanitos_iman2};
nJ=length(A{1}(1,:))/2;
nJ=10;
Juanitos={};
bins=300;
for i=1:nJ
    Juanitos{i}=[A{1}(:,i+(i-1)),A{1}(:,2*i)];
    Juanitos{i+round(nJ/2)-1}=[A{2}(:,i+(i-1)),A{2}(:,2*i)];
end
Juanitos_aux={};
for i = 1:round(nJ)-1
Juanitos_aux{i}=matricear2(Juanitos{i}(:,1),Juanitos{i}(:,2));
end
Juanitos=Juanitos_aux;

num_cubos=nJ/2;
%Determinar la muestra con el largo m?nimo
    largo_min=10000000000;
    for j=1:num_cubos
        if (length( Juanitos{j}(:,1) ) < largo_min)
                largo_min = length( Juanitos{j}(:,1) );
        end
    end

%Igualar el largo de las tomas
    B = {};
    Centro_aux = zeros(largo_min,6);
    Centro_geom = zeros(largo_min,2);
    for k=1:num_cubos
```

```
        if (length(Juanitos{k})>=largo_min)
            B{k,1} = zeros(largo_min,6);
            B{k,1} (1:largo_min, 1:6) = Juanitos{k}(1:largo_min, 1:6);
        end
    end
%Calculo efectivo
    for i=1:num_cubos
    Centro_aux (:,1) = Centro_aux(:,1)+B{i,1}(:,2);
    Centro_aux (:,2) = Centro_aux(:,2)+B{i,1}(:,3);
    end
    Centro_geom (:,1) = Centro_aux(:,1)/num_cubos;
    Centro_geom (:,2) = Centro_aux(:,2)/num_cubos;

%An?lisis del centro geometrico
    Centro_geom_matriceado=matricear2(Centro_geom(:,1),...
        Centro_geom(:,2));

%Graficar la trayectoria del centro geometrico
    figure(1)
    plot(Centro_geom(:,1),Centro_geom(:,2));
    hold on
    xlabel('X [cm]','fontsize',15);
    ylabel('Y [cm]','fontsize',15)
    titulo=strcat('Trayectorias centro geometrico Cubes Iman');
    title(titulo,'fontsize',15)


    %Desplazamiento medio
%ajuste lineal
figure(2)
    [desplazamiento_medio,taus]=calcDeltaW(Centro_geom_matriceado(:,2),...
        Centro_geom_matriceado(:,3));
    loglog(taus,desplazamiento_medio)
    grid on
    hold on
    xlabel('\tau [s]','fontsize',15);
    ylabel('\Delta r(\tau)','fontsize',15)
    %axis([10^-2 10^3 10^-1 10^1])
    titulo=strcat('Desplazamiento centro geometrico Cubes Iman');
    title(titulo,'fontsize',15);


x2=taus(1:length(taus));
y2=desplazamiento_medio;
p21=log(x2(5:round(length(x2)/2)));
p22=log(y2(5:round(length(y2)/2)));
p2=polyfit(p21,p22,1);
tau2=p2(1);
k2=exp(p2(2));

%graficar
loglog(x2,k2*x2.^tau2,':r*');
legend('Sim Cube',sprintf('Ajuste Sim Cube: y = %.1f * x^{%.2f}',...
    k2, tau2));

%dX, dY (velocidades)
```

```
    figure(3)
    subplot(1,3,1);
    grid on
    hold on;
    plot(Centro_geom_matriceado(:,1),Centro_geom_matriceado(:,4))
    hold on;
    xlabel('Tiempo [seg]','fontsize',15);
    ylabel('\Delta X [cm]','fontsize',15);
    axis([0 50 -10 10])
    titulo=strcat('\Delta X CG. Cubes Iman');
    title(titulo,'fontsize',15)

    subplot(1,3,2);
    grid on;
    hold on;
    plot(Centro_geom_matriceado(:,1),Centro_geom_matriceado(:,5)) %*120/20
    xlabel('Tiempo [seg]','fontsize',15);
    ylabel('\Delta Y [cm]','fontsize',15);
    axis([0 50 -10 10])
    titulo=strcat('\Delta Y CG. Cubes Iman');
    title(titulo,'fontsize',15)

    velocidad=sqrt((Centro_geom_matriceado(:,4)).^2+ ...
        (Centro_geom_matriceado(:,5)).^2)./Centro_geom_matriceado(:,1);
    subplot(1,3,3);
    plot(Centro_geom_matriceado(:,1),velocidad,'r') %*120/20
    axis([0 50 0 10]);
    grid on
    hold on; %plot(juan(:,1),juan(:,12),'r'); hold off
    xlabel('Tiempo [s]','fontsize',15);
    ylabel('Rap [cm/seg]','fontsize',15);
    titulo=strcat('Rapidez CG Cubes Iman [cm/seg]');
    title(titulo,'fontsize',15)


% PDF ?ngulo desplazamiento
    figure(4)
    [myhistnc,absmyhistnc,myhist,absmyhist]= ...
        calculoPDFadqui(Centro_geom_matriceado(:,6),bins);
    plot(absmyhistnc,myhistnc)
    hold on
    grid on
    xlabel('\Phi [grados]','fontsize',15)
    ylabel('PDF \Phi desplazamiento','fontsize',15);
    titulo=strcat('PDF \Phi desp. Cubes Iman');
    title(titulo,'fontsize',15)
```

Apéndice B

Anexo B
Programas en *Matlab™*

Aisle.cpp

```
#include "Aisle .h"
#include <iostream>
using namespace std;


Aisle ::Aisle (float l, float w)
{
    l_=l;
    w_=w;

            SIZE_LONG_SIDE = l_;
            x1_=0;
            y1_=w_;

            dMatrix3 rotac;
            dRFromAxisAndAngle (rotac, 0, 0, 1, M_PI_2);

%CREACION DE LA PRIMERA BARRA

            body[0].create (world);
            body[0].setPosition (x1_,y1_,SIZE_SHORT_SIDE/2.0);

            m.setBox
            (100.0,SIZE_SHORT_SIDE,SIZE_LONG_SIDE,SIZE_SHORT_SIDE);
            m.adjust
            (SIZE_SHORT_SIDE*SIZE_LONG_SIDE*SIZE_SHORT_SIDE);

            body[0].setMass (&m);

            box[0].create
             (space,SIZE_SHORT_SIDE,SIZE_LONG_SIDE,SIZE_SHORT_SIDE);
```

```
32              box[0].setBody (body[0]);
33
34              box[0].setRotation(rotac);
35              body[0].setRotation(rotac);
36
37              joint1[0].create (world);
38              joint1[0].attach (body[0],0);
39              joint1[0].setAnchor
40              (x1_+SIZE_SHORT_SIDE/2.0,y1_,SIZE_SHORT_SIDE/2.0);
41              joint1[0].setAxis(0,1,0);
42
43  %CREACION DE LA SEGUNDA BARRA
44              x2_=0;
45              y2_=-1.0;
46
47              body[1].create (world);
48              body[1].setPosition (x2_,y2_,SIZE_SHORT_SIDE/2.0);
49
50              m.setBox
51              (100.0,SIZE_SHORT_SIDE,SIZE_LONG_SIDE,SIZE_SHORT_SIDE);
52              m.adjust
53              (SIZE_SHORT_SIDE*SIZE_LONG_SIDE*SIZE_SHORT_SIDE);
54
55              body[1].setMass (&m);
56
57              box[1].create
58              (space,SIZE_SHORT_SIDE,SIZE_LONG_SIDE,SIZE_SHORT_SIDE);
59              box[1].setBody (body[1]);
60
61              box[1].setRotation(rotac);
62              body[1].setRotation(rotac);
63
64              joint1[1].create (world);
65              joint1[1].attach (body[0],0);
66              joint1[1].setAnchor
67              (x2_+SIZE_SHORT_SIDE/2.0,y2_,SIZE_SHORT_SIDE/2.0);
68              joint1[1].setAxis(0,1,0);
69
70  }
71
72  Aisle ::~Aisle ()
73  {
74      %dtor
75  }
76
77  void Aisle ::draw(void)
78  {
79    dReal sidelarge[3] = {SIZE_SHORT_SIDE,
80    SIZE_LONG_SIDE,SIZE_SHORT_SIDE};
81    dsSetColor (0,1,8);
82    dsSetTexture (DS_WOOD);
83     for (int bidx = 0; bidx < NUM_AISL_BODIES; bidx ++ )
84        dsDrawBox (body[bidx].getPosition(),
85        body[bidx].getRotation(),sidelarge);
86  }
```

caterpillar.cpp

```
#include "Caterpillar.h"

Caterpillar::Caterpillar()
{
    x_=0.0;
    y_=0.0;
}
Caterpillar::Caterpillar(float x, float y)
{
    x_=x;
    y_=y;

    for (int idx = 0; idx < LONG_CAT; idx++)
     %UBICACION DE LOS CUBOS QUE VIBRAN
        {
        Cat[idx] = new Cube(x_,y_+idx*(LARGESIDE+0.3));
        }
    %Join_Cat();
}
void Caterpillar::Join_Cat()
{

   /* for (int idx = 0; idx < LONG_CAT-1; idx++)
        {
        JointCat[idx].create (world);
        JointCat[idx].attach (Cat[idx]->bodya,Cat[idx+1]->bodya);
        JointCat[idx].setAnchor (x_+m_large_side/2,y_,m_large_side/2.0);
        %JointCat[idx].setAxis(0,0,1);
        }*/
}
void Caterpillar::drawCat()
{

    for (int idx = 0; idx < LONG_CAT; idx++)
    {
        Cat[idx]->drawC(1);
    }

}
void Caterpillar::vibrateCat()
{
    for (int idx = 0; idx < LONG_CAT; idx++)
    {
        Cat[idx]->vibrate();
    }
}
void Caterpillar::AddForceCat()
{
    static dReal fx=0;
    static dReal fy=5;
    static dReal fz=0;

```

```
53      for (int idx = 0; idx < LONG_CAT; idx++)
54      %UBICACION DE LOS CUBOS QUE VIBRAN
55      {
56          Cat[idx]->bodya.addRelForce(fx=0,fy=5,fz=0);
57      }
58
59  }
```

chess.cpp

```
1   #include "Chess.h"
2
3   Chess::Chess()
4   {
5        for (int idx = 0; idx <
6       CUBOS_HORIZONTALES; idx++ ){
7          for (int idx2 = 0; idx2 <
8          CUBOS_VERTICALES; idx2++){
9          Chess_cubes [idx][idx2] =
10         new Cube ((0-CUBOS_HORIZONTALES/2)*LARGESIDE*1.2 +
11         idx*(LARGESIDE*1.2),
12         (0-CUBOS_VERTICALES/2)*LARGESIDE*1.2 + idx2*(LARGESIDE*1.2));
13         }
14      }
15
16  }
17  void Chess::Join_Chess()
18  {
19
20      /*for (int idx = 0; idx < NUM_CHESS-1; idx++)
21          {
22          JointChess[idx].create (world);
23          JointChess[idx].attach
24          (Chess_cubes[idx]->bodya,Chess_cubes[idx+1]->bodya);
25          JointChess[idx].setAnchor
26          (x_+m_large_side/2,y_,m_large_side/2.0);
27          %JointCat[idx].setAxis(0,0,1);
28          }*/
29  }
30  void Chess::drawChess()
31  {
32
33      for (int idx = 0; idx < CUBOS_HORIZONTALES; idx++){
34          for (int idx2 = 0; idx2 < CUBOS_VERTICALES; idx2++){
35          Chess_cubes[idx][idx2]->drawC(2);
36          }
37      }
38
39  }
40  void Chess::vibrateChess()
41  {
42      for (int idx = 0; idx < CUBOS_HORIZONTALES; idx++){
43          for (int idx2 = 0; idx2 < CUBOS_VERTICALES; idx2++){
```

```
44          Chess_cubes[idx][idx2]->vibrate();
45          }
46      }
47  }
48
49  void Chess:: Stream_pos_Chess(char str [], float num_cubos)
50  {
51
52      if (Take_Data())
53      {
54      string cubos = Conv_to_String(num_cubos);
55      strcpy (str,"C:%Users/monte_000/Desktop/Caracterizacion/
56      %Mediciones/Mediciones_Automaticas");
57      strcat (str, cubos.c_str());
58      strcat (str, " Cubos_chess");
59      strcat (str, "_20_min/");
60      strcat (str, s4.c_str());
61
62      ofstream myfile;%writing to a file
63      myfile.open(str,ios::in | ios::app | ios::ate);
64
65      %CREA PUNTERO AL CUBO GRANDE Y ENTREGA LA POSICION
66      const dReal *a=dBodyGetPosition(bodya);
67      x_ = a[0];
68      y_ = a[1];
69      myfile << x_ << "," << y_ << endl;
70      myfile.close();
71      }
72  }
```

textttCube.cpp

```
1  #include "Cube.h"
2
3  using namespace std;
4
5  Cube::Cube()
6  {
7
8  }
9  Cube::Cube(float x, float y)
10 {
11     torque = 0.2;
12     vel_ang = 3;
13
14     Get_Initial_Time();
15
16     m_small_side = SMALLSIDE;
17     m_large_side = LARGESIDE;
18
19     x_ = x;
20     y_ = y;
21
```

```
22      %CTOR CUBO GRANDE
23      bodya.create (world);
24      bodya.setPosition (x_,y_,1*m_large_side);
25      m.setBox (density_Cube,m_large_side,m_large_side,m_large_side);
26      %la densidad del pla son 1.3 kg/m3
27      m.adjust (m_large_side*m_large_side);
28      bodya.setMass (&m);
29      %body[0].setData ((void*)(size_t)i);
30      box[0].create (space,m_large_side,m_large_side,m_large_side);
31      box[0].setBody (bodya);
32
33      %CTOR CUBO CHICO
34      bodyb.create (world);
35      bodyb.setPosition
36       (x_+0.20*m_large_side + 0.35*m_small_side,y_,1.0*m_large_side);
37      m.setBox (1,m_small_side,m_small_side,m_small_side);
38      m.adjust (2*m_small_side*m_small_side);
39      bodyb.setMass (&m);
40      %body[0].setData ((void*)(size_t)i);
41      box[1].create (space,m_small_side,m_small_side,m_small_side);
42      box[1].setBody (bodyb);
43
44      %CTOR BATERIA
45      bodyc.create(world);
46      bodyc.setPosition(x-0.3*m_large_side,y,m_large_side);
47      m.setCylinder
48      (density_Bat,direction_Bat,radius_Bat,length_Bat);
49      m.adjust(2.0*M_PI*(radius_Bat*radius_Bat)
50      +length_Bat*2.0*M_PI*radius_Bat);
51      bodyc.setMass(&m);
52      Battery.create (space,radius_Bat,length_Bat);
53      Battery.setBody (bodya);
54      %ESTA CON LA REFERENCIA CRUZADA PARA
55      %PEGAR LA PILA AL CUBO GRANDE.
56
57      %CREACION DE LA ARTICULACION ENTRE LOS CUBOS
58      jointC.create (world);
59      jointC.attach (bodya,bodyb);
60      jointC.setAnchor (x+0.5*m_large_side,y,1.05*m_large_side);
61      jointC.setAxis(1,0,0);
62
63      jointB.create(world);
64      jointB.attach(bodya,bodyc);
65
66
67
68  }
69  Cube::~Cube()
70  {
71      %dtor
72  }
73  void Cube::vibrate()
74  {
75      static double angle = 0;
76      angle += 0.05;
77      %Download_Battery();
```

```
78      dJointSetHingeParam(jointC,dParamVel,3);
79      % original 3 %vel_ang
80      dJointSetHingeParam(jointC,dParamFMax,0.2);
81       %original 0.2 %torque
82  }
83  void Cube::vibrate2()
84  {
85      static double angle2=5000*M_PI;
86      angle2 -=0.05;
87      dJointSetHingeParam(jointC,dParamVel,4);
88      dJointSetHingeParam(jointC,dParamFMax,0.5);
89  }
90  void Cube::drawC(int i)
91  {
92      dReal sidesmall[3] =
93      {m_small_side,m_small_side,m_small_side};
94      dReal sidelarge[3] =
95      {m_large_side,m_large_side,m_large_side};
96      switch (i)
97              {
98                  case 1 :{dsSetColor (7,10,0); break;}
99                  case 2 :{dsSetColor (3,4,0); break;}
100                 default:{dsSetColor (1,1,0); break;}
101             }
102     dsSetTexture (DS_WOOD);
103     dsDrawBox
104     (bodya.getPosition(),bodya.getRotation(),sidelarge);
105     %dsDrawCylinder
106     (bodya.getPosition(),bodya.getRotation(),0.6,0.6);
107     dsSetColor(5,5,5);
108     dsDrawBox
109     (bodyb.getPosition(),bodyb.getRotation(),sidesmall);
110     dsDrawCylinder
111     (bodyc.getPosition(),bodyc.getRotation(),length_Bat,radius_Bat);
112 }
113 void Cube:: Stream_Pos_cube()
114 {
115     if (Take_Data())
116     {
117             %CREA PUNTERO AL CUBO GRANDE Y ENTREGA LA POSICION
118             const dReal *a=dBodyGetPosition(bodya);
119
120             x_ = a[0];
121             y_ = a[1];
122
123             ofstream myfile;
124             myfile.open(s4.c_str(), ios::in | ios::app | ios::ate);
125             myfile << x_ << "," << y_ << endl;
126             myfile.close();
127         }
128 }
129 void Cube:: Stream_pos_cube2(char str [], float num_cubos)
130 {
131     if (Take_Data())
132     {
133     string cubos = Conv_to_String(num_cubos);
```

```
134     strcpy (str,"C:
135     %Users/monte_000/Desktop/Caracterizacion/
136     %Mediciones/Mediciones_Automaticas");
137     strcat (str, cubos.c_str());
138     strcat (str, " Cubos");
139     strcat (str, "_20_min/");
140     strcat (str, s4.c_str());
141
142     ofstream myfile; %writing to a file
143     myfile.open(str,ios::in | ios::app | ios::ate);
144
145     %CREA PUNTERO AL CUBO GRANDE Y ENTREGA LA POSICION
146     const dReal *a=dBodyGetPosition(bodya);
147     x_ = a[0];
148     y_ = a[1];
149     myfile << x_ << "," << y_ << endl;
150     myfile.close();
151     }
152 }
153 void Cube:: Stream_pos_cube3(char str [], float num_cubos)
154 {
155     if (Take_Data())
156     {
157     string cubos = Conv_to_String(num_cubos);
158     strcpy (str,"C:
159     %Users/monte_000/Desktop/Caracterizacion/
160     %Mediciones/Mediciones_Automaticas");
161     strcat (str, cubos.c_str());
162     strcat (str, " Cubos_Chess");
163     strcat (str, "_20_min/");
164     strcat (str, s4.c_str());
165
166     ofstream myfile; %writing to a file
167     myfile.open(str,ios::in | ios::app | ios::ate);
168
169     %CREA PUNTERO AL CUBO GRANDE Y ENTREGA LA POSICION
170     const dReal *a=dBodyGetPosition(bodya);
171     x_ = a[0];
172     y_ = a[1];
173     myfile << x_ << "," << y_ << endl;
174     myfile.close();
175     }
176 }
177 void Cube::build_name_pos_Chess(int bidx, int bidx2)
178 {
179     s1 = "Track_Cubo";
180     s2 = Conv_to_String((float) bidx);
181     s9 = Conv_to_String((float) bidx2);
182     s3 = ".txt";
183     s4 = s1 + s2+ s9 + s3;
184 }
185 void Cube::build_name_pos(int bidx)
186 {
187     s1 = "Track_Cubo";
188     s2 = Conv_to_String((float) bidx);
189     s3 = ".txt";
```

```
190     s4 = s1 + s2 + s3;
191     %cout <<"iteraciOn" << endl;
192 }
193 void Cube::Stream_ang_cube()
194 {
195     if (Take_Data())
196     {
197         ofstream myfile;
198         myfile.open(s8.c_str(), ios::in | ios::app | ios::ate);
199         Calc_angC();
200         myfile << ang1_ << endl;
201         myfile.close();
202     }
203 }
204 void Cube::Stream_ang_cube2(char str [], float num_cubos)
205 {
206     if (Take_Data())
207     {
208     string cubos = Conv_to_String(num_cubos);
209 
210     strcpy (str,"C:
211     %Users/monte_000/Desktop/Caracterizacion/
212     %Mediciones/Mediciones_Automaticas ");
213     strcat (str, cubos.c_str());
214     strcat (str, " Cubos");
215     strcat (str, "_20_min/");
216         strcat (str, s8.c_str());
217         ofstream myfile;
218         myfile.open(str, ios::in | ios::app | ios::ate);
219         Calc_angC();
220         myfile << ang1_ << endl;
221         myfile.close();
222     }
223 }
224 void Cube::build_name_ang(int bidx)
225 {
226     s5 = "Track_ang_cubo";
227     s6 = Conv_to_String((float)bidx);
228     s7 = ".txt";
229     s8 = s5 + s6 + s7;
230 }
231 void Cube::Stream_ang_Chess(char str [], float num_cubos)
232 {
233     if (Take_Data())
234     {
235     string cubos = Conv_to_String(num_cubos);
236 
237     strcpy (str,"C:
238     %Users/monte_000/Desktop/Caracterizacion/
239     %Mediciones/Mediciones_Automaticas");
240     strcat (str, cubos.c_str());
241     strcat (str, " Cubos_chess");
242     strcat (str, "_20_min/");
243         strcat (str, s8.c_str());
244         ofstream myfile;
245         myfile.open(str, ios::in | ios::app | ios::ate);
```

```
246         Calc_angC();
247         myfile << ang1_ << endl;
248         myfile.close();
249     }
250 }
251 void Cube::build_name_ang_Chess(int bidx, int bidx2)
252 {
253     s5 = "Track_ang_cubo";
254     s6 = Conv_to_String((float)bidx);
255     s10 = Conv_to_String((float)bidx2);
256     s7 = ".txt";
257     s8 = s5 + s6 + s10 + s7;
258 }
259 void Cube::Calc_angC()
260 {
261     const dReal *a=dBodyGetPosition(bodya);
262     dVector3 coordAnchor;
263     dJointGetHingeAnchor(jointC,coordAnchor);
264     ang1_ = Conv_to_String(a[0])+" "+Conv_to_String(a[1])+" "+
265     Conv_to_String(coordAnchor[0])+" "+Conv_to_String(coordAnchor[1]);
266 }
267 string Cube::Conv_to_String(float w)
268 {
269     float Number = w;       % number to be converted to a string
270     string Result;          % string which will contain the result
271     ostringstream convert;   % stream used for the conversion
272     convert << Number;      % insert the textual representation of
273     % 'Number' in the characters in the stream
274     Result = convert.str();
275     return Result;
276 }
277 bool Cube::Take_Data()
278 {
279  time(&second_time);
280     if ((second_time-start_time)%2==0)
281         return true;
282     else
283         return false;
284
285 }
286 void Cube::Get_Initial_Time()
287 {
288     time(&start_time);
289 }
290 void Cube::Download_Battery()
291 {
292     vel_ang=2000;
293     torque=5.0;
294     /*dReal descarga = torque/(1.0e15);
295     torque-=descarga;
296     vel_ang=POTENCIA/torque;*/
297 }
298 void Cube::Save_Position(int idx){
299     const dReal *K = bodya.getPosition();
300     Position_x [idx] = K[0];
301     Position_y [idx] = K[1];
```

```
302 }
303 void Cube::Get_Distance(){
304     for (int idx = 0; idx < NUMCUBES+NUMCUBES2+
305     NUMCUBES3+NUMCUBES4; idx++){
306             for (int idx2 = 0; idx2 < NUMCUBES+NUMCUBES2+
307             NUMCUBES3+NUMCUBES4; idx2++){
308             distance_cubes[idx][idx2] =
309             sqrt(powf((Position_x[idx]-Position_x[idx2]),2.0f)+
310             powf((Position_y[idx]-Position_y[idx2]),2.0f));
311             }
312     }
313 }
314 void Cube::Spring(float a){
315         if (a>3){}
316         if (a<3){}
317         else {}
318 }
```

Drive_Cube.cpp

```
1  #include "\texttt{Drive\_Cube.h}"
2  using namespace std;
3
4  Drive_Cube::~Drive_Cube()
5  {
6  }
7
8  Drive_Cube::Drive_Cube()
9  {
10 }
11 Drive_Cube::Drive_Cube(float x, float y)
12 {
13     x_ = x;
14     y_ = y;
15     m_small_side=SIDE_SMALL_CUBE;
16
17     %CTOR CUERPO JUANITO
18     body[0].create (world);
19     body[0].setPosition (x_,y_,height_leg+height_Cube/2);
20     m.setCylinder (1,1,radius_Cube,height_Cube);
21     m.adjust (2*M_PI*radius_Cube*radius_Cube+
22     height_Cube*2*M_PI*radius_Cube);
23     body[0].setMass (&m);
24     %body[0].setData ((void*)(size_t)i);
25     Cylinders[0].create (space,radius_Cube,height_Cube);
26     Cylinders[0].setBody (body[0]);
27
28     %CTOR 1Â° PATA
29     body[1].create (world);
30     body[1].setPosition (x_+radius_Cube,y_,height_leg/2);
31     m.setCylinder (1,1,radius_leg,height_leg);
32     m.adjust (2*M_PI*radius_leg*radius_leg+
33     height_leg*2*M_PI*radius_leg);
```

```
34      body[1].setMass (&m);
35      %body[0].setData ((void*)(size_t)i);
36      Cylinders[1].create (space,radius_leg,height_leg);
37      Cylinders[1].setBody (body[1]);
38
39      %CTOR 2Â° PATA
40      body[2].create (world);
41      body[2].setPosition (x_-radius_Cube*sin(M_PI/6),y_+
42      radius_Cube*cos(M_PI/6),height_leg/2);
43      m.setCylinder (1,1,radius_leg,height_leg);
44      m.adjust (2*M_PI*radius_leg*radius_leg+
45      height_leg*2*M_PI*radius_leg);
46      body[2].setMass (&m);
47      %body[0].setData ((void*)(size_t)i);
48      Cylinders[2].create (space,radius_leg,height_leg);
49      Cylinders[2].setBody (body[2]);
50
51      %CTOR 3Â° PATA
52      body[3].create (world);
53      body[3].setPosition (x_-radius_Cube*sin(M_PI/6),
54      y_-radius_Cube*cos(M_PI/6),height_leg/2);
55      m.setCylinder (1,1,radius_leg,height_leg);
56      m.adjust (2*M_PI*radius_leg*radius_leg+
57      height_leg*2*M_PI*radius_leg);
58      body[3].setMass (&m);
59      %body[0].setData ((void*)(size_t)i);
60      Cylinders[3].create (space,radius_leg,height_leg);
61      Cylinders[3].setBody (body[3]);
62
63      %CTOR BATERIA
64      body[4].create(world);
65      body[4].setPosition(x_,y_,height_Cube+height_leg);
66      m.setCylinder(density_Bat,1,radius_Bat,length_Bat);
67      m.adjust(2.0*M_PI*(radius_Bat*radius_Bat)+length_Bat*
68      2.0*M_PI*radius_Bat);
69      body[4].setMass(&m);
70      Battery.create (space,radius_Bat,length_Bat);
71      Battery.setBody (body[0]);
72      %ESTA CON LA REFERENCIA CRUZADA PARA
73      %PEGAR LA PILA AL CUBO GRANDE.
74
75      Cylinders[4].create (space,radius_Bat,length_Bat);
76      Cylinders[4].setBody (body[4]);
77
78      %CTOR 1Â° MASA PEQUENA
79      body[5].create (world);
80      body[5].setPosition
81      (x_+(radius_Cube-0.5*m_small_side)*sin(M_PI/6),y_
82      +(radius_Cube-0.5*m_small_side)*
83      cos(M_PI/6),height_leg+height_Cube/2);
84      m.setBox (1,m_small_side,m_small_side,m_small_side);
85      m.adjust (6*m_small_side*m_small_side);
86      body[5].setMass (&m);
87      Cubes[0].create (space,m_small_side,m_small_side,m_small_side);
88      Cubes[0].setBody (body[5]);
89      dMatrix3 rotac;
```

```
90      dRFromAxisAndAngle (rotac, 0, 0, 1, M_PI/3);
91      Cubes[0].setRotation(rotac);
92      body[5].setRotation(rotac);
93
94      %CTOR 2Â° MASA PEQUENA
95      body[6].create (world);
96      body[6].setPosition (x_+(radius_Cube-0.5*m_small_side)*sin(M_PI/6),y_-
97      (radius_Cube-0.5*m_small_side)*cos(M_PI/6),height_leg+height_Cube/2);
98      m.setBox (1,m_small_side,m_small_side,m_small_side);
99      m.adjust (6*m_small_side*m_small_side);
100     body[6].setMass (&m);
101     Cubes[1].create (space,m_small_side,m_small_side,m_small_side);
102     Cubes[1].setBody (body[6]);
103     dMatrix3 rotac1;
104     dRFromAxisAndAngle (rotac1, 0, 0, 1, -M_PI/3);
105     Cubes[1].setRotation(rotac1);
106     body[6].setRotation(rotac1);
107
108     %CREACION DE LA ARTICULACION PATAS Y BASE
109     for (int idx=0; idx<3; idx++){
110         jointH_Legs[idx].create(world);
111         jointH_Legs[idx].attach(body[idx+1],body[0]);
112     }
113
114     jointH_Legs[0].setAnchor (x_+radius_Cube,y_,height_leg+height_Cube/2.0);
115     jointH_Legs[1].setAnchor (x_-radius_Cube*sin(M_PI/6),y_+
116     radius_Cube*cos(M_PI/6),height_leg+height_Cube);
117     jointH_Legs[2].setAnchor (x_-radius_Cube*sin(M_PI/6),y_
118     -radius_Cube*cos(M_PI/6),height_leg+height_Cube);
119
120     jointH_Legs[0].setAxis(0,0,1);
121     jointH_Legs[1].setAxis(0,0,1);
122     jointH_Legs[2].setAxis(0,0,1);
123
124     for (int idx=0; idx<2; idx++){
125         jointH_Motors[idx].create(world);
126         jointH_Motors[idx].attach(body[idx+5],body[0]);
127     }
128     jointH_Motors[0].setAnchor(x_+(radius_Cube-m_small_side)*sin(M_PI/6),y_+
129     (radius_Cube-m_small_side)*cos(M_PI/6),height_leg+1.1*(height_Cube/2));
130     jointH_Motors[1].setAnchor(x_+(radius_Cube-m_small_side)*sin(M_PI/6),y_
131     -(radius_Cube-m_small_side)*cos(M_PI/6),height_leg+1.1*(height_Cube/2));
132     jointH_Motors[0].setAxis(0.6*cos(M_PI/6),0.6*sin(M_PI/6),0);
133     jointH_Motors[1].setAxis(-0.6*cos(M_PI/6),0.6*sin(M_PI/6),0);
134
135     joint_Battery.create(world);
136     joint_Battery.attach(body[0],body[4]);
137     joint_Battery.setAnchor(x_,y_,height_Cube+height_leg);
138     joint_Battery.setAxis(0,0,1);
139 }
140
141 void Drive_Cube::vibrate_DriveCube_left()
142 {
143     static double angle_left = 0;
144     angle_left += 0.05;
145     %Download_Battery();
```

```
146     dJointSetHingeParam(jointH_Motors[0],dParamVel,2);
147     % original 3
148     dJointSetHingeParam(jointH_Motors[0],dParamFMax,0.1);
149     %original 0.2
150 }
151
152 void Drive_Cube::vibrate_DriveCube_right()
153 {
154     static double angle_right = 0;
155     angle_right += 0.05;
156     dJointSetHingeParam(jointH_Motors[1],dParamVel,3);
157     % original 3
158     dJointSetHingeParam(jointH_Motors[1],dParamFMax,0.2);
159     %original 0.2
160 }
161
162 void Drive_Cube::draw_DriveCube()
163 {
164     dReal sidesmall[3] = {m_small_side,m_small_side,m_small_side};
165     dsSetColor (7,10,0);
166     dsSetTexture (DS_WOOD);
167     dsDrawBox (body[5].getPosition(),
168     body[5].getRotation(),sidesmall);
169     dsDrawBox (body[6].getPosition(),
170     body[6].getRotation(),sidesmall);
171     dsSetColor(8,1,1);
172     dsDrawCylinder(body[0].getPosition(),
173     body[0].getRotation(),height_Cube,radius_Cube);
174     dsDrawCylinder(body[1].getPosition(),
175     body[1].getRotation(),height_leg,radius_leg);
176     dsDrawCylinder(body[2].getPosition(),
177     body[2].getRotation(),height_leg,radius_leg);
178     dsDrawCylinder(body[3].getPosition(),
179     body[3].getRotation(),height_leg,radius_leg);
180     dsDrawCylinder(body[4].getPosition(),
181     body[4].getRotation(),length_Bat,radius_Bat);
182 }
```

Farmyard.cpp

```
1  #include "Farmyard.h"
2  #include <iostream>
3
4  using namespace std;
5
6  Farmyard::Farmyard(float r){
7      r_ = r;
8      float angle = (2.0*M_PI)/NUM_FARM_BODIES;
9      long_side_ = 2.0*r_*sin(M_PI/NUM_FARM_BODIES);
10
11         for (int bidx = 0; bidx < NUM_FARM_BODIES; bidx ++ ){
12
13             float slice_angle = bidx*angle;
```

```
14
15              %cout <<slice_angle << endl; cin.get();
16
17              x_ = posiciones[bidx][0] = (r_-SHORT_SIDE/2)*cos(slice_angle);
18              y_ = posiciones[bidx][1] = (r_-SHORT_SIDE/2)*sin(slice_angle);
19
20              %cout << "posicion inicial:" <<posiciones[bidx][0] << " "
21              %<< posiciones[bidx][1] << endl;
22
23              body[bidx].create (world);
24              body[bidx].setPosition (x_,y_,SHORT_SIDE/2.0);
25
26              joint1[bidx].create (world);
27
28              m.setBox (1300*0.2,SHORT_SIDE,long_side_,SHORT_SIDE);
29              %el primer parÃ¡metro se asocia a densidad de la pieza
30              m.adjust (SHORT_SIDE*long_side_*SHORT_SIDE);
31
32              body[bidx].setMass (&m);
33
34
35              box[bidx].create (space,SHORT_SIDE,long_side_,SHORT_SIDE);
36              box[bidx].setBody (body[bidx]);
37
38              dMatrix3 rotac;
39              dRFromAxisAndAngle (rotac, 0, 0, 1, slice_angle);
40              box[bidx].setRotation(rotac);
41              body[bidx].setRotation(rotac);
42
43          }
44          for (int bidx = 0; bidx < NUM_FARM_BODIES; bidx ++ )
45              {
46                  switch (bidx)
47                  {
48  case (NUM_FARM_BODIES-1)
49  :{unir(body[bidx],body[0],bidx);
50  /*cout << "Ãºltima uniOn creada"
51   <<endl;*/ break;}
52  default:
53   {unir(body[bidx],body[bidx+1],bidx); /*
54    cout <<
55   "uniones intermedias creadas" <<endl;*/ break;}
56                  }
57          }
58  }
59
60  Farmyard::~Farmyard()
61  {
62      %dtor
63  }
64
65  void Farmyard::draw(void)
66  {
67    dReal sidelarge[3] =
68     {SHORT_SIDE,long_side_,SHORT_SIDE};
69    dsSetColor (0,1,1);
```

```
70   dsSetTexture (DS_WOOD);
71    for (int bidx = 0; bidx < NUM_FARM_BODIES; bidx ++ )
72       dsDrawBox (body[bidx].getPosition(),
73       body[bidx].getRotation(),sidelarge);
74 }
75
76 void Farmyard::unir (dBody &a, dBody &b,int bidx)
77 {
78             joint1[bidx].attach(a,b);
79             joint1[bidx].setAnchor
80             ( posiciones[bidx][0]+SHORT_SIDE/2,
81             posiciones[bidx][1],
82             SHORT_SIDE/2 );
83             %coord globales
84             joint1[bidx].setAxis(0,0,1);%direccion generica
85 }
86
87 void Farmyard:: Stream_Pos_Farm()
88 {
89     if (Take_Data())
90     {
91     %CREAR PUNTERO A LOS CENTROS DE LOS CUBOS
92     % Y ENTREGARLOS EN UNA SOLA LINEA
93     ofstream myfile;
94     const dReal *a;
95     myfile.open(s4.c_str(), ios::in | ios::app | ios::ate);
96     for (int idx = 0; idx < NUM_FARM_BODIES; idx++){
97          a = dBodyGetPosition(body[idx]);
98          x_ = a[0];
99          y_ = a[1];
100         myfile << x_ << "," << y_ << ",";
101         }
102         myfile << endl;
103         myfile.close();
104     }
105 }
106 void Farmyard:: Stream_Pos_Farm2(char str [], float num_farm)
107 {
108     if (Take_Data())
109     {
110     string farm_element = Conv_to_String(num_farm);
111     strcpy (str,"C:
112     %Users/monte_000/Desktop/Caracterizacion/
113     %Mediciones/Mediciones_Automaticas");
114     strcat (str, farm_element.c_str());
115     strcat (str, " Cubos");
116     strcat (str, "_20_min/");
117     strcat (str, s4.c_str());
118     %CREAR PUNTERO A LOS CENTROS DE LOS CUBOS
119     % Y ENTREGARLOS EN UNA SOLA LINEA
120     ofstream myfile;
121     const dReal *a;
122     myfile.open(str, ios::in | ios::app | ios::ate);
123     for (int idx = 0; idx < NUM_FARM_BODIES; idx++){
124          a = dBodyGetPosition(body[idx]);
125          x_ = a[0];
```

```
126           y_ = a[1];
127           myfile << x_ << "," << y_ << ",";
128           }
129           myfile << endl;
130           myfile.close();
131       }
132   }
133
134   void Farmyard::build_name_pos(int bidx)
135   {
136       s1 = "Track_Farmyard";
137       s2 = Conv_to_String((float) bidx);
138       s3 = ".txt";
139       s4 = s1 + s2 + s3;
140   }
141   void Farmyard::Stream_ang_Farm()
142   {
143       if (Take_Data())
144       {
145           ofstream myfile;
146           myfile.open(s8.c_str(), ios::in | ios::app | ios::ate);
147           for (int idx = 0; idx < NUM_FARM_BODIES; idx++){
148           Calc_angF(idx);
149           myfile << ang1_;
150           }
151           myfile << endl;
152           myfile.close();
153       }
154   }
155   void Farmyard::Stream_ang_Farm2(char str [], float num_farm)
156   {
157       if (Take_Data())
158       {
159           string farm_element = Conv_to_String(num_farm);
160           strcpy (str,"C:%Users/monte_000/Desktop/Caracterizacion/
161           %Mediciones/Mediciones_Automaticas");
162           strcat (str, farm_element.c_str());
163           strcat (str, " Cubos");
164           strcat (str, "_20_min/");
165           strcat (str, s8.c_str());
166           ofstream myfile;
167           myfile.open(str, ios::in | ios::app | ios::ate);
168           for (int idx = 0; idx < NUM_FARM_BODIES; idx++){
169           Calc_angF(idx);
170           myfile << ang1_<<" ";
171           }
172           myfile << endl;
173           myfile.close();
174       }
175   }
176   void Farmyard::build_name_ang(int bidx)
177   {
178       s5 = "Track_ang_Farmyard";
179       s6 = Conv_to_String((float)bidx);
180       s7 = ".txt";
181       s8 = s5 + s6 + s7;
```

```
}
void Farmyard::Calc_angF(int bidx)
{
    const dReal *a=dBodyGetPosition(body[bidx]);
    dVector3 coordAnchor;
    dJointGetHingeAnchor(joint1[bidx],
    coordAnchor);
    ang1_ = Conv_to_String(a[0])+" "+
    Conv_to_String(a[1])+" "+
    Conv_to_String(coordAnchor[0])+" "+
    Conv_to_String(coordAnchor[1]);
}
string Farmyard::Conv_to_String(float w)
{
    float Number = w;
     % number to be converted to a string
    string Result;
     % string which will contain the result
    ostringstream convert;
     % stream used for the conversion
    convert << Number;
    % insert the textual representation of
    %'Number' in the characters in the stream
    Result = convert.str();
    return Result;
}
bool Farmyard::Take_Data()
{
  time(&second_time);
    if ((second_time-start_time)%2==0)
        return true;
    else
        return false;
}
```

main.cpp

```
#include <Needed.h>

static void nearCallback (void *data, dGeomID o1, dGeomID o2)
{
  % exit without doing anything if the two bodies are connected by a joint
  dBodyID b1 = dGeomGetBody(o1);
  dBodyID b2 = dGeomGetBody(o2);
  if (b1 && b2 && dAreConnected (b1,b2)) return;

  % @@@ it's still more convenient to use the C interface here.

  dContact contact;
  contact.surface.mode = 0;
  contact.surface.mu = dInfinity;
  if (dCollide (o1,o2,1,&contact.geom,sizeof(dContactGeom)))
  %dCollide:intersects two geoms and generates contact points.
```

```
17      {
18      dJointID c = dJointCreateContact (world.id(),contactgroup.id(),&contact);
19      dJointAttach (c,b1,b2);
20      }
21  }
22
23  static void start()  %seteo de cÃ¡mara y punto de observaciOn
24  {
25    dAllocateODEDataForThread(dAllocateMaskAll);
26
27    static float xyz[3] = {2.1640f,-1.3079f,1.7600f};
28    static float hpr[3] = {125.5000f,-17.0000f,0.0000f};
29    dsSetViewpoint (xyz,hpr);
30  }
31
32
33  % simulation loop
34
35  static void simLoop (int pause)
36  {
37      finish_simulation();
38    if (!pause)
39      {
40
41          %Drive_Cube1.vibrate_DriveCube_left();
42          %Drive_Cube1.vibrate_DriveCube_right();
43
44      /*for (int idx = 0; idx < NUMCUBES; idx++)
45          {
46          mycube[idx]->vibrate();
47          mycube[idx]->build_name_pos(idx);
48          mycube[idx]->Stream_pos_cube2(str, Suma_NUMCUBES);
49          mycube[idx]->build_name_ang(idx);
50          mycube[idx]->Stream_ang_cube2(str, Suma_NUMCUBES);
51          %const dReal* k = dBodyGetPosition(mycube[idx]->bodya);
52          mycube[idx]->Save_Position(idx);
53          mycube[idx]->Get_Distance();
54          %mycube[idx]->Put_ring();
55          }
56      for (int idx = 0; idx < NUMCUBES2; idx++)
57      {
58          mycube2[idx]->vibrate();
59          mycube2[idx]->build_name_pos(idx+NUMCUBES);
60          mycube2[idx]->Stream_pos_cube2(str, Suma_NUMCUBES);
61          mycube2[idx]->build_name_ang(idx+NUMCUBES);
62          mycube2[idx]->Stream_ang_cube2(str, Suma_NUMCUBES);
63      }
64          for (int idx = 0; idx < NUMCUBES3; idx++)
65      {
66          mycube3[idx]->vibrate();
67          mycube3[idx]->build_name_pos(idx+NUMCUBES2+NUMCUBES);
68          mycube3[idx]->Stream_pos_cube2(str, Suma_NUMCUBES);
69          mycube3[idx]->build_name_ang(idx+NUMCUBES2+NUMCUBES);
70          mycube3[idx]->Stream_ang_cube2(str, Suma_NUMCUBES);
71      }
72
```

```
73      for (int idx = 0; idx < NUMCUBES4; idx++)
74      {
75          mycube4[idx]->vibrate();
76          mycube4[idx]->build_name_pos(idx+NUMCUBES3+NUMCUBES2+NUMCUBES);
77          mycube4[idx]->Stream_pos_cube2(str, Suma_NUMCUBES);
78          mycube4[idx]->build_name_ang(idx+NUMCUBES3+NUMCUBES2+NUMCUBES);
79          mycube4[idx]->Stream_ang_cube2(str, Suma_NUMCUBES);
80      }*/
81     /*for (int idx = 0; idx < NUM_ROTAT; idx++)
82          {
83          Rotador[idx]->vibrateR();
84          Rotador[idx]->build_name_pos_Rot(idx);
85          Rotador[idx]->Stream_Pos_Rot2(str2, NUM_ROTAT);
86          Rotador[idx]->build_name_ang_Rot(idx);
87          Rotador[idx]->Stream_ang_Rot2(str2, NUM_ROTAT);
88          }*/
89
90         /* Cuadricula->vibrateChess();
91      for (int idx = 0; idx < CUBOS_HORIZONTALES; idx++){
92          for (int idx2 = 0; idx2 < CUBOS_VERTICALES; idx2++){
93          Cuadricula->Chess_cubes[idx][idx2]->build_name_pos_Chess(idx, idx2);
94          Cuadricula->Chess_cubes[idx][idx2]->
95          Stream_pos_cube3(str3, CUBOS_HORIZONTALES*CUBOS_VERTICALES);
96          Cuadricula->Chess_cubes[idx][idx2]->
97          build_name_ang_Chess(idx, idx2);
98          Cuadricula->Chess_cubes[idx][idx2]->
99          Stream_ang_Chess(str3, CUBOS_HORIZONTALES*CUBOS_VERTICALES);
100         }*/
101     %}
102  %ENTREGAR LA POSICION DE LOS ELEMENTOS DE LA BARRERA
103 /*        Barrera.build_name_pos(NUM_FARM_BODIES);
104         Barrera.Stream_Pos_Farm2(str, Suma_NUMCUBES);
105         Barrera.build_name_ang(NUM_FARM_BODIES);
106         Barrera.Stream_ang_Farm2(str, Suma_NUMCUBES);*/
107
108 /*   for (int idx = 0; idx < NUM_ROTAT2; idx++)
109         {
110         Rotador2[idx]->vibrateR2();
111         }*/
112
113    % Cater.vibrateCat();
114     space.collide (0,&nearCallback);
115     world.step (0.05);
116     contactgroup.empty();
117     }
118
119     /*for (int idx = 0; idx < NUMCUBES; idx++)
120         {
121         mycube[idx]->drawC(0);
122         }
123     for (int idx = 0; idx < NUMCUBES2; idx++)
124     {
125         mycube2[idx]->drawC(0);
126     }
127     for (int idx = 0; idx < NUMCUBES3; idx++)
128     {
```

```
129         mycube3[idx]->drawC(0);
130     }
131     for (int idx = 0; idx < NUMCUBES4; idx++)
132     {
133         mycube4[idx]->drawC(0);
134     }*/
135    /* for (int idx = 0; idx < NUM_ROTAT; idx++)
136         {
137         Rotador[idx]->drawR();
138         }*/
139    % Cuadricula->drawChess();
140
141  /*   for (int idx = 0; idx < NUM_ROTAT2; idx++)
142         {
143         Rotador2[idx]->drawR2();
144         }*/
145
146 %Barrera.draw();
147     %Pasillo.draw();
148 %    Cater.drawCat();
149 Drive_Cube1.draw_DriveCube();
150 }
151
152
153 int main (int argc, char **argv)
154 {
155  Build_Folder(Suma_NUMCUBES);
156  Build_Folder_Rotat(NUM_ROTAT);
157  Build_Folder_Chess(CUBOS_HORIZONTALES * CUBOS_VERTICALES);
158  start_simulation();
159
160   % setup pointers to drawstuff callback functions
161   dsFunctions fn;
162   fn.version = DS_VERSION;
163   fn.start = &start;
164   fn.step = &simLoop;
165   fn.command = 0;
166   fn.stop = 0;
167   fn.path_to_textures = DRAWSTUFF_TEXTURE_PATH;
168
169   % create world
170   dInitODE();
171
172   contactgroup.create ();
173   world.setGravity (0,0,-9.8);
174   dWorldSetCFM (world.id(),1e-5);
175   dPlane planebottom (space,0,0,1,0);
176   %dPlane planetop (space,0,0,1,PLANEHEIGHT);
177
178   %TEMPORIZADOR PARA LA SIMMULACION
179 /*
180     for (int idx = 0; idx < NUMCUBES; idx++) %UBICACION DE LOS CUBOS QUE VIBRAN
181         {
182         float angle = 2*3.14*((float)idx/(float)NUMCUBES);
183         float radius = 0.5*RADIUS_FARMYARD;
184         mycube[idx] = new Cube(radius*cos(angle),radius*sin(angle));
```

```
185         }
186     for (int idx = 0; idx < NUMCUBES2; idx++)
187     {
188         float angle = 2*M_PI*((float)idx/(float)NUMCUBES2);
189         float radius = 0.7*RADIUS_FARMYARD;
190         mycube2[idx] = new Cube(radius*cos(angle), radius*sin(angle));
191     }
192    for (int idx = 0; idx < NUMCUBES3; idx++)
193     {
194         float angle = 2*M_PI*((float)idx/(float)NUMCUBES3);
195         float radius = 0.3*RADIUS_FARMYARD;
196         mycube3[idx] = new Cube(radius*cos(angle), radius*sin(angle));
197     }
198     for (int idx = 0; idx < NUMCUBES4; idx++)
199     {
200         float angle = 2*M_PI*((float)idx/(float)NUMCUBES4);
201         float radius = 0.9*RADIUS_FARMYARD;
202         mycube4[idx] = new Cube(radius*cos(angle), radius*sin(angle));
203     }*/
204     /*for (int idx = 0; idx < NUM_ROTAT; idx++)
205         {
206         float angle = 2*3.14*((float)idx/(float)NUM_ROTAT);
207         float radius = 0.9*RADIUS_FARMYARD;
208         Rotador[idx] = new Rotator (radius*cos(angle),radius*sin(angle));
209         }*/
210 %    Cuadricula = new Chess();
211   % run simulation
212   dsSimulationLoop (argc,argv,352,288,&fn);
213   dCloseODE();
214
215   return 0;
216 }
217 void start_simulation(){
218 time(&start_simulation_time);
219 }
220 void finish_simulation(){
221  time(&finish_simulation_time);
222     if ((finish_simulation_time-start_simulation_time)>1200)
223         exit(0);
224 }
225 void Build_Folder(float total_cubos)
226 {
227     string cubos = Conv_to_String(total_cubos);
228
229     strcpy (str,"C:
230     %Users/monte_000/Desktop/Caracterizacion/
231     %Mediciones/Mediciones_Automaticas");
232     strcat (str, cubos.c_str());
233     strcat (str, " Cubos");
234     strcat (str, "_20_min");
235     mkdir(str);
236 }
237 void Build_Folder_Rotat(float total_Rotat)
238 {
239     string Rotadores = Conv_to_String(total_Rotat);
240
```

```
241     strcpy (str2,"C:
242     %Users/monte_000/Desktop/Caracterizacion/
243     %Mediciones/Mediciones_Automaticas");
244     strcat (str2, Rotadores.c_str());
245     strcat (str2, " Rotat");
246     strcat (str2, "_20_min");
247     mkdir(str2);
248 }
249 void Build_Folder_Chess(float cubos_en_chess)
250 {
251     string Total_chess = Conv_to_String(cubos_en_chess);
252 
253     strcpy (str3,"C:
254     %Users/monte_000/Desktop/Caracterizacion/
255     %Mediciones/Mediciones_Automaticas");
256     strcat (str3, Total_chess.c_str());
257     strcat (str3, " Cubos_chess");
258     strcat (str3, "_20_min");
259     mkdir(str3);
260 }
261 string Conv_to_String(float w)
262 {
263     float Number = w;        % number to be converted to a string
264     string Result;           % string which will contain the result
265     ostringstream convert;    % stream used for the conversion
266     convert << Number;       % insert the textual representation
267     %of 'Number' in the characters in the stream
268     Result = convert.str();
269     return Result;
270 }
```