UNA ARQUITECTURA BASADA EN MULTI-AGENTES PARA LA PLANIFICACIÓN DEL AGARRE ESTABLE EN UNA MANO ROBÓTICA

EDUAR DORADO DAZA

EDWAR DARIO AÑASCO PARRA

UNIVERSIDAD DEL CAUCA
FACULTAD DE INGENIERÍA ELECTRÓNICA Y TELECOMUNICACIONES
INGENIERÍA EN AUTOMÁTICA INDUSTRIAL
POPAYÁN
2010

UNA ARQUITECTURA BASADA EN MULTI-AGENTES PARA LA PLANIFICACIÓN DEL AGARRE ESTABLE EN UNA MANO ROBÓTICA

EDUAR DORADO DAZA

EDWAR DARIO AÑASCO PARRA

Proyecto de grado presentado como requisito para optar al título de Ingenieros en Automática Industrial

Director:
Ing. Jaime Díaz Chávez
Docente de la Universidad del Cauca

UNIVERSIDAD DEL CAUCA
FACULTAD DE INGENIERÍA ELECTRÓNICA Y TELECOMUNICACIONES
INGENIERÍA EN AUTOMÁTICA INDUSTRIAL
POPAYÁN
2010

CONTENIDO

CONTENIDO ____ ***i***

LISTA DE FIGURAS ____ ***ivv***

LISTA DE TABLAS ____ ***v***

RESUMEN ____ ***vi***

INTRODUCCIÓN ____ ***1***

1. ANÁLISIS DEL AGARRE ____ ***3***

1.1 CONTACTOS ____ **3**

1.1.1 Contactos punto-punto sin fricción ____ 4

1.1.2 Contactos punto-punto con fricción rígidos ____ 4

1.1.3 Contactos punto-punto con fricción blandos ____ 5

1.2 AGRUPACIÓN DE CONTACTOS EN UN AGARRE ____ **7**

1.2.1 Base de *Wrenches* ____ 7

1.2.2 *Grasp map* (mapa del agarre) ____ 8

1.3 EL ESPACIO *WRENCH* DEL AGARRE ____ **9**

1.3.1 Construcción del espacio *Wrench* del agarre ____ 10

1.3.2 Construcción en *GraspIt!* del espacio *Wrench* del agarre ____ 13

1.4 DOS MEDIDAS DE CALIDAD ____ **14**

1.5 VISUALIZACIÓN DEL ESPACIO *WRENCH* ____ **16**

1.5.1 Visualización del espacio *Wrench* usando *GraspIt!* ____ 16

2. PLANIFICACIÓN DEL AGARRE ESTABLE EN UNA MANO ROBÓTICA ____ ***19***

2.1 ANTECEDENTES DE LA PLANIFICACIÓN DEL AGARRE ____ **20**

2.1.1 Planificadores que utilizan sensores de visión ____ 20

2.1.2 Planificadores que utilizan sensores de contacto ____ 21

2.1.3 Planificadores que utilizan sensores ópticos o de campo eléctrico ____ 22

2.2 SISTEMA PROPUESTO DE PLANIFICACIÓN DEL AGARRE BASADO EN MULTI-AGENTES ____ **23**

2.2.1 Multi-agentes en el control del agarre y la manipulación ____ 23

2.2.2 Antecedentes ____ 25

2.2.3 El enfoque propuesto ____ 26

3. AGENTES Y SISTEMAS MULTI-AGENTE ____ ***29***

3.1 AGENTES INTELIGENTES 29

3.2 CONCEPTOS CENTRALES PARA LA CONSTRUCCION DE AGENTES 31

3.2.1 Agentes situados en un ambiente: acciones y percepciones 31

3.2.2 Eventos y objetivos: agentes proactivos y reactivos 31

3.2.3 Ambientes difíciles: planes y creencias 32

3.3 SISTEMAS MULTI-AGENTE (MAS) 32

3.4 MÉTODOS DE COMUNICACIÓN EN SISTEMAS MULTI-AGENTE 33

3.5 PROMETHEUS: UNA METODOLOGIA PARA EL DESARROLLO DE SISTEMAS MULTI-AGENTE 35

3.5.1 Fase 1: especificación del sistema 36

3.5.2 Fase 2: diseño de la arquitectura 37

3.5.3 Fase 3: diseño detallado 38

3.5.4 Prometheus en comparación con otras metodologías 39

4. FASE DE ESPECIFICACIÓN DEL SISTEMA ***40***

4.1 ESPECIFICACIÓN DE OBJETIVOS 40

4.1.1 Determinación del agarre estable 41

4.1.2 Identificación de los sub-objetivos 43

4.1.3 Notación abreviada de los objetivos y sub-objetivos 46

4.2 ESPECIFICACION DE LAS FUNCIONALIDADES 48

4.3 ESPECIFICACIÓN DE LOS ESCENARIOS 52

4.3.1 Descripción detallada del escenario de configuración 53

4.3.2 Descripción detallada del escenario de adaptación por cierre 54

4.3.3 Descripción detallada del escenario de adaptación por alineación 56

4.3.4 Descripción detallada del escenario de contacto 58

4.3.5 Ejecución de los escenarios para los diferentes tipos de agarre 58

4.4 DESCRIPCIÓN DE LA INTERFAZ 61

4.4.1 Acciones y percepciones 61

4.4.2 Las bases de datos 62

5. FASE DE DISEÑO DE LA ARQUITECTURA. ***63***

5.1 ESPECIFICACIÓN DE LOS TIPOS DE AGENTES 63

5.1.1 Agrupación de funcionalidades 63

5.1.2 Diagrama de acoplamiento de datos 63

5.1.3 Revisión del acoplamiento de los agentes - diagrama de relaciones 64

5.1.4 Desarrollo de los descriptores de agente 65

5.2 ESPECIFICACIÓN DE LAS INTERACCIONES 67

5.2.1 Diagramas de interacción a partir de los escenarios 67

5.2.2 Desarrollo de protocolos de interacción a partir de los diagramas de interacción 70

5.2.3 Desarrollo de los descriptores de protocolos y mensajes 73

5.3 DISEÑO FINAL DE LA ARQUITECTURA 75

5.3.1 Identificación de los límites del sistema de agentes 75
5.3.2 Desarrollo de los descriptores de las percepciones y acciones 76
5.3.3 Definición de los objetos de datos compartidos (*Shared Data Objects*) 77
5.3.4 Diagrama de visión general del sistema 78

6. FASE DE DISEÑO DETALLADO 81

6.1 DIAGRAMA DE VISIÓN GENERAL DEL AGENTE 81

6.2 DIAGRAMAS DE VISIÓN GENERAL DE LAS CAPACIDADES 83
6.2.1 Capacidad de flexión-extensión_ind 83
6.2.2 Capacidad de aducción-abducción_ind 84

6.3 PROCESOS 85
6.3.1 Proceso de configuración del agente índice 86
6.3.2 Proceso de flexión del agente índice 86
6.3.3 Proceso de aducción-abducción del agente índice. 87

7. CONTROL DE FUERZA EN EL AGARRE 89

8. PARÁMETROS DE IMPLEMENTACIÓN DE LA ARQUITECTURA DE PLANIFICACIÓN. 93

8.1 PLATAFORMAS PARA LA IMPLMENTACIÓN DE AGENTES 93

8.2 CONEXIÓN DE *JACK* Y *JADEX* CON *GraspIt!* 94

9. CONCLUSIONES Y TRABAJOS FUTUROS 99

9.1 CONCLUSIONES 99

9.2 TRABAJOS FUTUROS 100

LISTA DE FIGURAS

Figura 1.1. Tipos de contacto según su geometría. 4
Figura 1.2. Contacto punto-punto sin Fricción 4
Figura 1.3. Componentes de fuerza en el plano tangente en un contacto punto-punto 5
Figura 1.4. Cono de fricción que se origina en el plano tangente 5
Figura 1.5. Contacto punto-punto con fricción blando 6
Figura 1.6. Par originado debido al modelo de contacto blando 6
Figura 1.7. Transformación del marco de referencia del contacto al marco de referencia del objeto. 9
Figura 1.8. Cono de fricción aproximado con una pirámide de 8 lados. 10
Figura 1.9. Ejemplo de un espacio unitario *Wrench* del agarre. 15
Figura 1.10. Ventana de diálogo para crear el espacio *Wrench* del agarre. 17
Figura 1.11. Agarres de diferentes objetos con una mano robótica. 18
Figura 2.1. Tipos De Agarre 19
Figura 2.2. Configuraciones de los sensores ópticos según el enfoque de pre-contacto. 23
Figura 2.3. Movimientos de los dedos que se proponen para la mano robótica. 27
Figura 3.1. Estructura de un sistema de pizarra 34
Figura 3.2. Arquitecturas de comunicación en sistemas multi-agente. 34
Figura 3.3. Las fases de la metodología Prometheus. 36
Figura 4.1. Ejemplo representativo que muestra como la superficie de la punta del dedo índice logra quedar lo mas paralela posible a la superficie local del objeto. 45
Figura 4.2. Diagrama de Objetivos para el sistema de planificación. 49
Figura 4.3. Diagrama de funcionalidades para el sistema de planificación del agarre. 50
Figura 5.1. Diagrama de acoplamiento de datos. 64
Figura 5.2. Diagrama de relaciones 65
Figura 5.3. Diagrama de interacción de acuerdo con el escenario de adaptación por alineación. 69
Figura 5.5. Protocolo para la flexión. 72
Figura 5.4. Protocolo para la aducción-abducción. 73
Figura 5.6. Símbolos gráficos utilizados en el diagrama de visión general del sistema. 78
Figura 5.7. Diagrama de visión general del sistema para la planificación de agarre en una mano robótica. 79
Figura 5.8. Arquitectura basada en agentes 81
Figura 6.1. Diagrama de visión general del agente índice. 82
Figura 6.2. Arquitectura interna de los agentes (índice, medio o anular) 83
Figura 6.3. Diagrama de visión general de la capacidad de flexión-extensión_ind 84

Figura 6.4 Diagrama de visión general de la capacidad de aducción-abducción_ind 85
Figura 6.5 Ilustración de algunas notaciones para la especificación de los procesos. 85
Figura 6.6 proceso de configuración del agente índice. 86
Figura 6.7 Proceso de flexión-extensión del agente índice. 87
Figura 6.8 Proceso para la aducción-abducción del agente índice 88
Figura 7.1. Dos dedos y su correspondiente dedo virtual 90
Figura 7.2. Espacios de oposición 90
Figura 8.1 Conexión de la plataforma *JACK* con *GraspIt!* 95
Figura 8.2. Conexión de *JADEX* con *GraspIt!,* pasando antes por *MATLAB*. 96

LISTA DE TABLAS

Tabla 1. Valores de los coeficientes de fricción entre los materiales definidos en *GraspIt!* 13
Tabla 2. Valores que identifican el tipo de agarre. 26
Tabla 3. Escenario de configuración 54
Tabla 4. Escenario adaptación por cierre 55
Tabla 5. Escenario de adaptación por alineación 57
Tabla 6. Escenario de contacto 58
Tabla 7. Percepciones y acciones realizadas en cada dedo 61

RESUMEN

En este documento se presenta una arquitectura de planificación distribuida para el agarre estable de objetos con una mano robótica de 12 grados de libertad (GDL). El diseño de la arquitectura es realizado usando un enfoque de pre-contacto combinado un enfoque basado en multi-agentes, en el cual cada dedo es considerado como un agente que interactúa con los demás para desempeñar ajustes de agarre en línea sin requerir contacto prematuro con el objeto o estrategias de re-agarre. Los ajustes se hacen de acuerdo con la información extraída de un arreglo de sensores de proximidad ópticos ubicados en la punta y falange proximal de cada dedo. Esta información contiene la orientación y la distancia de la superficie local del objeto que permite a cada dedo adaptarse a la geometría del objeto en orden de aumentar la calidad del agarre. En este documento también se muestra como con la ayuda de los sensores de proximidad puede ser posible aplicar la hipótesis de los dedos virtuales para simplificar el problema de control del agarre de una mano multi-dedo a un problema, más sencillo, donde solo se controlan dos dedos virtuales.

INTRODUCCIÓN

En la actualidad se han desarrollado muchos trabajos para solucionar el problema de la planificación del agarre estable de objetos. En estos trabajos lo primero que se hace es analizar el contorno del objeto para determinar el conjunto de puntos de contacto que garantizan un buen balance entre fuerza y par. En la mayoría de estos enfoques para encontrar dichos puntos se hace uso de algoritmos de procesamiento de imágenes y reconstrucción de objetos en 3D [1]-[7]. En muchos casos el procesamiento se hace fuera de línea por lo que el gasto computacional es bastante alto.

En otros enfoques lo que se hace es mover los dedos por el contorno del objeto buscando maximizar alguna medida de calidad [8], [9], [10]. Pero al igual que en los enfoques basados en visión el gasto computacional es alto y no se garantiza que el agarre sea estable. Un tercer enfoque deja de lado la visión y la planificación se realiza en la fase de pre-agarre (fase previa del agarre). Este tipo de planificadores usan sensores de proximidad ópticos o de campo eléctrico para lograr que la superficie de la punta de los dedos de una mano robótica logre contactar, de forma paralela, la superficie local del objeto, es decir, la superficie que se encuentra enfrente de la punta de cada dedo, y que además, todos los dedos logren contactar el objeto al mismo tiempo con el fin de que el objeto no se desplace por el contacto prematuro de algún dedo. En este grupo de planificadores se destaca el trabajo realizado por *Hsiao* y sus colegas quienes recientemente han propuesto el sensado *pretouch* con sensores ópticos [11]. La puesta en práctica de este enfoque muestra como objetos de diferente forma, tamaño y peso son agarrados en tiempo real y de manera estable por una mano robótica. Una limitación del enfoque, es que su diseño es orientado como una herramienta de análisis científico y no es aplicado, al menos explícitamente, al caso de una prótesis de mano robótica.

Como los humanos, los robots necesitan un cierto nivel de autonomía, reactividad, pro-actividad y habilidad social para desarrollar sus tareas [12]. Una mano robótica multi-dedo, que sirva como prótesis de mano robótica, necesita autonomía para enfrentarse ante situaciones imprevistas, como: el agarre de objetos desconocidos, el deslizamiento del objeto cuando se esta agarrando, perturbaciones por parte del ambiente y muchos otros inconvenientes que se pueden dar al tratar de agarrar un objeto; La reactividad en una mano robótica es la respuesta de ésta en tiempo real ante eventos críticos de tiempo para realizar un buen agarre; la pro-actividad esta representada en la persecución de un agarre estable aún en caso de fallas; la habilidad social se puede ver cuando se la toma como un conjunto de dedos unidos a una palma que interactúan entre si para llevar a cabo un agarre. Estas características son normalmente citadas como razón para adoptar la tecnología de agentes [12], ya que precisamente, entre las principales características de un agente se encuentran la autonomía, la reactividad y la pro-actividad. Estas características toman aún mas sentido cuando los agentes

están situados en ambientes dinámicos, es decir, en ambientes que cambian rápidamente. Para este tipo de ambientes se necesita que varios agentes interactúen entre si para lograr solucionar problemas de gran complejidad. Dado que un problema complejo, es divido sub-problemas que son resueltos, mas fácilmente, por cada agente de manera individual, obteniéndose así sistemas más autónomos, robustos, escalables y modulares [12]. Aprovechando las cualidades que ofrece la tecnología de agentes, muchos investigadores de robótica la han aplicado para hacer sus robots más inteligentes. Específicamente los robots son diseñados para que los procesos de planificación y control sean distribuidos en varios agentes quienes se encargan de una parte de cada uno de estos sub-procesos.

Debido a la ventajas que ofrecen los sistemas multi-agente en la solución de problemas complejos y a las ventajas que ofrece el enfoque *pretouch* en la planificación del agarre estable, se propone, en la primera parte de este trabajo, una arquitectura que se basa en el enfoque *pretouch* y considera a cada dedo como un agente que se coordina y coopera con los demás para lograr un agarre estable del objeto.

Para el desarrollo de la arquitectura, en el primer capítulo se estudian los diferentes tipos de contacto que existen, se hace una agrupación de estos considerando un solo marco de referencia, se construye el espacio *Wrench* en base a esta agrupación y se hace uso del software *GraspIt!* para realizar diferentes agarres con una mano robótica con las mismas características a la descrita en [14]; en el segundo capítulo se revisan los antecedentes sobre planificación del agarre estable, se hace un estudio sobre la aplicación de sistemas multi-agente en control del agarre y manipulación de objeto y por último se describe de manera detallada el enfoque propuesto para la planificación del agarre. En capítulo 3 se hace un marco teórico de agentes, sistemas multi-agentes y métodos de comunicación entre ellos, además, se escoge en base a criterios de aplicabilidad e información detallada la metodología Prometheus como la metodología más idónea para el desarrollo del sistema multi-agente aplicado a una mano robótica. Por último, en los capítulos 4, 5 y 6 se desarrollan cada una de las fases descritas en la metodología Prometheus y como resultado se obtiene una arquitectura basada en multi-agentes para la planificación del agarre.

1. ANÁLISIS DEL AGARRE

Agarrar de manera correcta un objeto es un aspecto importante para satisfacer una tarea en particular. En robótica el desarrollo de manipuladores cada vez más complejos, tal como las manos robóticas, ha aumentando la necesidad de obtener algoritmos que no solo generen agarres de manera automática, sino que la calidad de estos pueda ser cuantificada con el fin de optimizar dichos agarres [13]. En la actualidad, para cuantificar la calidad del agarre se hace uso de diferentes métodos de análisis, entre estos se encuentran el *mapa de agarre* y *el espacio Wrench del agarre,* los cuales han sido ampliamente usados por diferentes autores que enfocan sus trabajos en el análisis del agarre robótico.

En este capítulo se hace un estudio sobre la naturaleza del agarre y los diferentes tipos de contacto que existen, además, se describen dos medidas de calidad del agarre definidas sobre el *espacio Wrench del agarre*. Por último, se usa el software *GraspIt!* [15] para visualizar y estudiar los resultados que se producen al agarrar diferentes objetos con una mano robótica que tiene las mismas características a la desarrollada en [14].

Mucha de la información que se presenta en este capítulo aprovecha el desarrollo y los resultados del trabajo realizado por Andrew Miller, quien en 2001 desarrollo *GraspIt!* como parte de su proyecto de doctorado en la Universidad de Columbia [15].

1.1 CONTACTOS

Por el contacto, el movimiento relativo de dos cuerpos que se tocan es obstruido completamente en algunas direcciones. En esta situación es posible que entre ellos se trasmitan fuerzas y velocidades a través de las regiones de contacto. Si existe contacto entre dos cuerpos, A y B, entonces entre ellos se define un plano tangente. Si se consideran las fuerzas que el cuerpo A puede aplicar al cuerpo B, lo primero que se debe considerar es que el cuerpo A solamente puede aplicar fuerzas al otro lado del plano tangente. Es decir, un contacto es siempre unilateral ya que el cuerpo A puede empujar al cuerpo B pero no puede arrastrarlo. El espacio de fuerzas que pueden ser aplicadas a través del plano tangente, como veremos a continuación, es definido por la geometría del contacto y la presencia de fuerzas de fricción.

Según su geometría, los tipos de contactos son tres: contacto punto-punto, contacto borde-borde y contacto cara-cara. En la figura 1.1 se puede ver una representación gráfica de estos tipos de contacto. Las coordenadas que se encuentran menos repintadas de negro son la dirección y grados de libertad que posee el contacto. El plano con el que contacta el objeto es el plano tangente que se genera entre los dos objetos.

Figura 1.1. Tipos de contacto según su geometría. [3]

El tratamiento de los contactos cara-cara y borde-borde son más complejos que los contactos punto-punto. Para simplificarlos es posible introducir la noción propuesta por [3] en la cual los contactos cara-cara y borde-bode pueden ser representados como una suma de contactos punto-punto, ubicados en cada uno de los vértices de la superficie contactada, es decir, un contacto borde-borde es equivalente a tener dos contactos punto-punto ubicados al final de los bordes en contacto, en tanto que un contacto cara-cara es equivalente a tener contactos punto-punto ubicados en cada uno de los vértices de la caras que se encuentran en contacto.

1.1.1 Contactos punto-punto sin fricción

Se define un contacto punto-punto sin fricción cuando no hay fricción en el punto de contacto entre los dos objetos (ver figura 1.2). En este caso, cualquier componente de fuerza ubicada en el plano tangente provocaría un movimiento relativo o deslizamiento entre los objetos. Por lo tanto, para que persista el contacto, las fuerzas solo pueden ser aplicadas en la dirección normal ($\boldsymbol{f}_{\perp}$) a la superficie del objeto. Los contactos sin fricción solo tienen interés puramente teórico ya que dos superficies en contacto siempre contienen cierta cantidad de fricción.

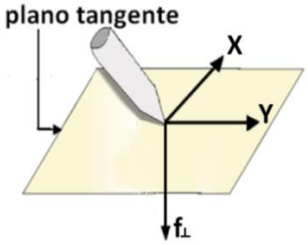


Figura 1.2. Contacto punto-punto sin Fricción

1.1.2 Contactos punto-punto con fricción rígidos

Un punto de contacto con fricción se produce cuando existe fricción entre dos objetos. La fricción genera componentes de fuerza que se oponen al deslizamiento del contacto y que están ubicadas en el plano tangente formado en el punto de contacto. La magnitud máxima de las fuerzas de fricción que se pueden generar en el plano tangente de un contacto punto-punto se determina comúnmente usando el modelo de Coulomb [16]. El modelo de Coulomb establece:

$$\boldsymbol{ft \leq \mu f_{\perp}} \qquad (1.1)$$

Donde ft es la componente de fuerza tangencial debida a la fricción, $f_{\perp}$ es la componente de fuerza normal, y μ es el coeficiente de fricción entre los dos materiales en contacto. Cuando se trabaja en 3 dimensiones la ecuación puede ser escrita como:

$$\boldsymbol{f_x^2 + f_y^2 \leq \mu^2 f_{\perp}^2}, \qquad (1.2)$$

Donde f_x y f_y son componentes de fuerza perpendiculares entre sí ubicadas en el plano tangente. En la figura 1.3 se observa un ejemplo de este tipo de contacto, f_x y f_y son las componentes de fuerza perpendiculares. ft es la componente de fuerza tangencial y $\boldsymbol{f_{\perp}}$ es la componente de fuerza normal.

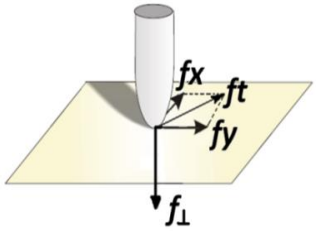


Figura 1.3. Componentes de fuerza en el plano tangente en un contacto punto-punto

De la ecuación (1.2), es posible ver que las fuerzas que pueden ser aplicadas al contacto, están localizadas dentro de un cono alineado con la normal del contacto, comúnmente conocido como *cono de fricción.* En la figura 1.4 se observa un cono de fricción que se origina en el plano tangente ante la presencia de una fuerza normal de magnitud μf⊥. Para prevenir el deslizamiento, la fuerza de contacto total, *f*, debe estar localizada dentro del cono de posibles direcciones y el ángulo máximo entre el borde del cono y la normal de contacto es igual a $\tan^{-1}\mu$

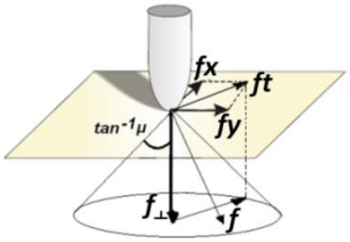


Figura 1.4. Cono de fricción que se origina en el plano tangente

1.1.3 Contactos punto-punto con fricción blandos

Cuando en la región de contacto los cuerpos se ven sometidos a deformación, el contacto se realiza realmente en un área. El área de contacto puede tener un incremento directamente

proporcional a la fuerza normal, pero si la deformación es leve esta área es pequeña y, por simplicidad, es posible asumir que el contacto se realiza en un punto. Se dice entonces que el contacto es blando (en ingles *soft*). En la figura 1.5 se observa dos cuerpos que ejercen presión sobre una superficie deformable. La superficie que esta marcada con rayas representa el área de contacto que se establece entre los cuerpos y el objeto al cual se le ejerce presión.

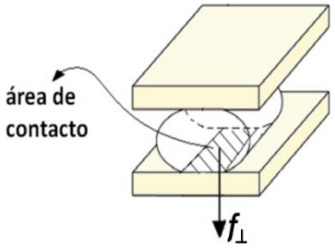


Figura 1.5. Contacto punto-punto con fricción blando

El resultado importante de la presencia de esta área de contacto es que permite al dedo ejercer, además de fuerzas puras dentro del cono de rozamiento, un par respecto al eje normal al plano tangente que es proporcional a la fuerza normal aplicada (ver figura 1.6). Si el par es excesivo el dedo resbala en el área de contacto y, de manera similar a la fuerza, la magnitud del par estará limitada por un coeficiente de fricción torsional γ. Las ecuaciones que describen el contacto blando con friccion son las de la ecuación (1.2) más las condiciones que describen el par, se muestran en la ecuación 1.3.

$$\begin{gathered} f_x^2 + f_y^2 \leq \mu^2 {f_\perp}^2 \\ f_\perp \geq 0 \\ |fT| \leq \gamma f_\perp \end{gathered} \qquad (1.3)$$

Donde fT es el par generado alrededor de la componente de fuerza normal del contacto

Un modelo analítico que reduce la complejidad del tratamiento matemático del contacto blando representa al área de contacto mediante un conjunto de vértices situados en el límite del área obtenidos por medio de una expresión matemática. Este es el modelo usado en *GraspIt!* Y es descrito en [17].

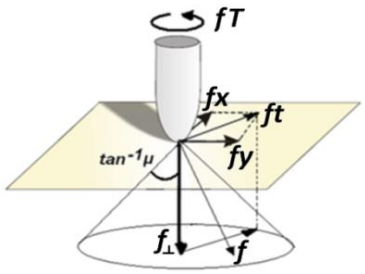


Figura 1.6. Par originado debido al modelo de contacto blando

1.2 AGRUPACIÓN DE CONTACTOS EN UN AGARRE

Antes de profundizar sobre la agrupación de contactos en un agarre, se hace necesario que primero definamos las nociones de agarre y agarre estable. Un agarre se define como un conjunto de contactos sobre un objeto en los cuales se aplica alguna cantidad de fuerza [18]. El objetivo principal de un agarre es lograr que el objeto quede estable ante la presencia de fuerzas externas (fuerzas de disturbio) o la fuerza debida a su propio peso, es decir, que el agarre sea estable. De manera más formal se puede definir que el agarre estable realizado por una mano robótica es aquel en el cual las fuerzas ejercidas por los dedos pueden balancear pares y fuerzas arbitrarias ejercidas sobre el objeto.

En la sección anterior se hizo un análisis del espacio de fuerzas que pueden ser aplicados en un solo punto de contacto. En estos puntos de contacto también se pueden aplicar pares, los cuales junto con las fuerzas de contacto determinan que tan eficaz es el agarre actual. Por simplicidad matemática, estas cantidades (fuerza y par en un punto de contacto) son agrupadas en un Vector llamado *Wrench* el cual será descrito con más detalle en esta sección.

1.2.1 Base de *Wrenches*

Un contacto es una interfaz entre dos cuerpos a través de la cual pueden ser aplicadas fuerzas y pares. Con el propósito de describir matemáticamente las interacciones en los contactos en ecuaciones compactas en lugar de tratar estas cantidades como un par de vectores, comúnmente son combinadas en un solo vector llamado *Wrench.*[1].

$$w = \begin{bmatrix} f \\ T \end{bmatrix} \quad (1.4)$$

Con objetos bidimensionales $\omega \, \epsilon \, \Re^3$, dos dimensiones correspondientes a las dos fuerzas ortogonales situadas en el plano tangente del objeto, y la otra correspondiendo al par perpendicular a este plano. Para objetos tridimensionales $\omega \, \epsilon \, \Re^6$, se tienen 3 variables que representan el par total que actúa sobre el objeto y 3 más para representar la fuerza total.

La geometría del contacto (De borde-borde, de cara-cara o de punto-punto) y la presencia de fricción definen el espacio de *Wrenches* que pueden ser generado a través del contacto. Este espacio *Wrench* puede ser descrito con una base. Matemáticamente,

$$\omega_c = B_c f_c \qquad f_c \, \epsilon \, FC_c \quad (1.5)$$

[1] En la actualidad no existe un consenso universal del significado de la palabra inglesa *Wrench.* Algunos autores de habla hispana lo traducen como fuerza extendida, en la cual se contiene un par que es el resultado de multiplicar una fuerza por un coeficiente. En ocasiones se utiliza el término fuerza como sinónimo de *Wrench*.

Donde ω_c es el espacio *Wrench* que puede ser transmitido, respecto al origen de coordenadas del contacto c, B_c es la denominada matriz base de *Wrenches*, f_c es un vector que representa la intensidad de las fuerzas ejercidas en el contacto, y FC_c son las restricciones del cono de fricción. En un *contacto punto-punto con fricción rígida* la base de *Wrenches* está dada por la ecuación (1.6):

$$B_c = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad FC_c = \left\{ f \in \Re^3 : \sqrt{f1^2 + f2^2} \leq \mu f_3, f_3 \geq 0 \right\} \quad (1.6)$$

La presencia de las tres últimas filas de la matriz B_c muestra que con esta base no se pueden trasmitir pares. Los contactos en *GraspIt!* son todos considerados como contactos punto-punto con fricción rígidos.

1.2.2 *Grasp map* (mapa del agarre)

Para saber cómo afecta el aplicar fuerzas, (*Wrenches*), en los contactos sobre el objeto es necesario expresar tales fuerzas con respecto a un solo marco de referencia. Aunque este marco de referencia puede ser arbitrario, la selección más común es el centro de masa del objeto. La transformación del marco de referencia de un contacto al marco de referencia del objeto se realiza utilizando la siguiente transformación:

$$\omega_{0i} = \begin{pmatrix} R_{0c_i} & 0 \\ P_{0c_i} R_{0c_i} & R_{0c_i} \end{pmatrix} B_{c_i} f_{c_i} = G_i . f_{c_i} \quad (1.7)$$

En donde R_{0c_i} denota la matriz de orientación del marco de referencia del contacto con respecto al marco de referencia del objeto, 0. P_{0c_i} Es el vector que representa la posición del punto de contacto i relativo al marco de referencia del objeto. f_{ci} Es un vector que representa la intensidad de las fuerzas ejercidas en el contacto i. B_{c_i} Es la matriz base de *Wrenches* para el contacto i, esta matriz multiplicada por la matriz que contiene a R_{0c_i} y P_{0c_i}, genera una matriz representada como G_i que se le conoce como mapeo del contacto i. La figura 1.7 es un ejemplo representativo de la función que ejerce el mapa de agarre. En esta figura un primer objeto de forma alargada contacta un segundo objeto de forma ovalada. Este contacto representado como c_i con las coordenadas z_i, x_i, y_i es llevado al centro de masa del objeto por medio del mapa de agarre G. El centro de masa del objeto es representado con las coordenadas z_o, x_o, y_o.

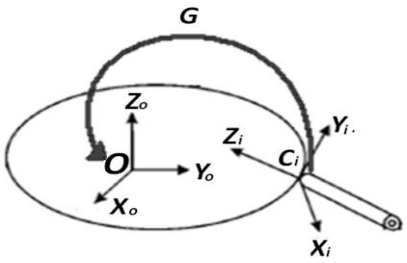


Figura 1.7. Transformación del marco de referencia del contacto al marco de referencia del objeto.

Por lo tanto, la suma de todas las fuerzas o *Wrenches* que se están aplicando al objeto resultan en la siguiente ecuación:

$$\omega_0 = G_1 f_{c1} + \cdots + G_n f_{cn} = [G_1 \dots G_n].\begin{bmatrix} f_{c1} \\ \vdots \\ f_{cn} \end{bmatrix} = G.\begin{bmatrix} f_{c1} \\ \vdots \\ f_{cn} \end{bmatrix} \quad (1.8)$$

Finalmente, el mapeo del agarre es la colección de todos los mapeos de contacto G_i de cada punto de contacto, definido como la matriz $\boldsymbol{G}$**:**

$$\boldsymbol{w_0 = G.f_c} \quad \boldsymbol{f_c} \in \boldsymbol{FC} \quad (1.9)$$

1.3 EL ESPACIO *WRENCH* DEL AGARRE

Una fuerza de contacto f_{ci} se dice que es permisible si satisface las restricciones impuestas por la fricción, dadas por una de las ecuaciones (1.1), (1.2), o (1.3), dependiendo del tipo de contacto. Además, se dice que la fuerza total del contacto, $\boldsymbol{f_c}$, es permisible si cada una de las fuerzas f_{ci} actuando sobre el objeto, es permisible. Asi mismo, un *Wrench* resultante ω_0 se dice que es permisible si existe una fuerza f_c permisible tal que $w_0 = G.f_c$. Finalmente, el espacio *Wrench* del agarre se define como el espacio completo de fuerzas y pares que pueden ser aplicados a un objeto por un conjunto de fuerzas de contacto permisibles.

Sin embargo, para un análisis más completo de las propiedades de un agarre es necesario examinar el conjunto de fuerzas externas que puede resistir una determinada configuración de agarre. Un agarre se dice que tiene cierre de fuerza, en ingles *force-closure*, si para cualquier ω_{ext} existe un ω_0 permisible tal que $\omega_0 = -\omega_{ext}$. En otras palabras, un agarre con cierre de fuerza puede generar *Wrenches* en todo el espacio. Es preciso mencionar que aunque un agarre con cierre de fuerza puede resistir *Wrenches* de disturbio arbitrarios, algunos de estos disturbios pueden ser más fáciles de resistir que otros.

1.3.1 Construcción del espacio *Wrench* del agarre

En la planificación de agarres óptimos expuesta en [19] se describen dos métodos para construir el espacio *Wrench* del agarre. En ambos casos se construye una envoltura convexa[2]. En un caso, la envoltura convexa se normaliza respecto a la magnitud de la suma de las fuerzas normales de contacto, y en el otro, se normaliza respecto a la magnitud máxima de las fuerzas normales. Ambos métodos comienzan con una aproximación lineal del cono de fricción cuadrático. Esto significa que cualquier fuerza de contacto permisible *f* está acotada por la suma convexa de un número finito *m* de fuerzas situadas en el borde del cono de fricción. En estas condiciones, una fuerza situada en uno de los bordes del cono de fricción aproximado esta dada por:

$$f \approx \sum_{j=1}^{m} \alpha_j f_j, \quad (1.10)$$

$$f_j = \hat{f}_{\lrcorner} + \mu_s \cos\left(\frac{2\pi j}{m}\right) \hat{f}_x + \mu_s \sin\left(\frac{2\pi j}{m}\right) \hat{f}_y, \quad (1.11)$$

$$\alpha_j \geq 0,$$

$$\sum_{j=1}^{m} \alpha_j = \|f_{\lrcorner}\| \quad (1.12)$$

Donde cada $\boldsymbol{f_j}$ es un vector de fuerza a lo largo de un borde de la pirámide de m-lados descrita por los vectores unitarios $\hat{\boldsymbol{f}}_{\lrcorner}$, $\hat{\boldsymbol{f}}_{\boldsymbol{x}}$, $\hat{\boldsymbol{f}}_{\boldsymbol{y}}$.

El cono de fricción que se muestra en la figura 1.8 se ha aproximado con una pirámide de 8 lados. *f* es representada como una combinación convexa de 8 vectores de fuerza alrededor del borde del cono de fricción. Cada uno de estos 8 vectores de fuerza es obtenido mediante la ecuación 1.11.

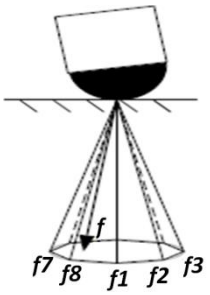


Figura 1.8. Cono de fricción aproximado con una pirámide de 8 lados. [15]

El siguiente paso para construir el *espacio Wrench del agarre*, que se describe en los dos métodos mencionados anteriormente, es encontrar el espacio total de *Wrenches* que pueden

[2] En matemática se define la envoltura convexa de un conjunto de puntos *X* de dimensión n como la intersección de todos los conjuntos convexos que contienen a *X*.

ser ejercidos por todos los contactos del agarre sobre el objeto. Dado el espacio de fuerzas que pueden ser aplicados al punto de contacto i, se debe determinar el espacio correspondiente de *Wrenches* que pueden ser ejercidos por este contacto sobre el objeto. Para lograr esto, se requiere un multiplicador para el par, ⋋, que relacione unidades de par a unidades de fuerza.

La elección de ⋋ es algo arbitraria debido a que la magnitud del par puede ser escalada de manera independiente con respecto a la magnitud de fuerza. En *GraspIt!* al igual que en [18], ⋋ se ha escogido de tal manera que la calidad de agarre sea independiente de la escala del objeto. Para ello se ha escogido ⋋ = *1/r,* donde *r* es la distancia máxima desde el origen del espacio *Wrench,* a menudo el centro de gravedad, a cada uno de los contactos. Lo anterior da como resultado que la magnitud del par no sea mayor a la magnitud de fuerza, es decir, $\|T\| \leq \|f\|$.

Cada uno de los m vectores que representan el borde del cono de fricción es trasladado al origen del espacio *Wrench* calculando el par correspondiente tal que:

$$\omega_{i,j} = \begin{bmatrix} f_{i,j} \\ \lambda\,(d_i \times f_{i,j}) \end{bmatrix}, \quad (1.13)$$

Donde $f_{i,j}$ es uno de los m vectores sobre el borde del cono de fricción del contacto i y d_i es el vector que representa la distancia desde el origen del *espacio Wrench* al punto de contacto i. Cada *Wrench* $\omega_{i,j}$ en (1.13) determina el valor máximo del *Wrench* que puede ser aplicado en el punto de contacto *i* por la fuerza $f_{i,j}$, dada una fuerza normal unitaria.

Antes de continuar con la construcción del espacio *Wrench* del agarre es útil para el cálculo de las medidas de calidad la definición del denominado *espacio Wrench unitario del agarre*. Ahora bien, si se toma *g* como el vector compuesto de las magnitudes de las fuerzas normales aplicadas en cada contacto, se obtiene el vector de fuerza:

$$g = \begin{bmatrix} \|f_{\lrcorner 1}\| \\ . \\ . \\ . \\ \|f_{\lrcorner n}\| \end{bmatrix} \quad (1.14)$$

El espacio *Wrench* unitario del agarre se puede definir como el espacio de los *Wrenches* que pueden ser aplicados por un agarre que tiene un vector de fuerza *g* con magnitud igual a 1. Sin embargo, es posible construir el espacio unitario de distintas maneras dependiendo de la forma de especificar cómo se determina la magnitud unitaria del vector de fuerza. Por ejemplo, en [19] se ofrecen dos posibilidades, en ambos casos se normaliza respecto a la magnitud del vector de fuerza que ha sido definido según la expresión (1.14). En el primer caso la magnitud se define como la suma de las fuerzas normales de contacto, es decir se usa la norma L_1.

$$\|g\| = (\|f_{\lrcorner 1}\| + \cdots . + \|f_{\lrcorner n}\|) \;\; ; \qquad \|g\| = 1 \; para \; una \; norma \; L_1 \; (L_1 \; norm) \; (1.15)$$

y en el otro, la magnitud es definida como la componente máxima del vector de fuerza, es decir se usa la norma L_∞, la cual establece:

$$\|g\| = \max \; (\|f_{\lrcorner 1}\| + \cdots . + \|f_{\lrcorner n}\|) \quad ; \qquad \|g\| = 1 \; para \; una \; norma \; L_\infty \; (L_\infty \; norm) \;\; (1.16)$$

Otra posible aproximación para construir el espacio *Wrench* unitario, como es el caso de *GraspIt!*, es asumir que las fuerzas de contacto son independientes y se normalizan individualmente usando la norma L_∞. Si la fuerza normal ejercida por cada contacto es normalizada como $\|f_{\lrcorner i}\|$=1, entonces el conjunto total de *Wrenches*, W_i, que resulta de todos los contactos se puede expresar de la siguiente manera:

$$W_i = \left\{ \omega_i \mid \omega_i = \sum_{j=1}^{m} \alpha_{i,j} \omega_{i,j} \,, \;\; \alpha_{i,j} \geq 0, \;\; \sum_{j=1}^{m} \alpha_{i,j} \leq 1 \right\}, (1.17)$$

De esta manera, el espacio *Wrench* unitario del agarre W_{L_∞}, que es el conjunto de posibles *Wrenches,* que pueden ser ejercidos sobre el objeto, es la suma de Minkowski[3] de los conjuntos convexos de todos los contactos W_i :

$$W_{L_\infty} = \bigoplus_{i=1}^{n} \omega_i \;\; (1.18)$$

Una manera de implementar el conjunto de posibles *Wrenches,* W_{L_∞} que pueden ser ejercidos sobre el objeto seria realizar la envoltura convexa de los puntos generados a partir de la suma de Minkowski de cada uno de los conos de fricción.

$$W_{L_\infty} = ConvexHull\left(\bigoplus_{i=1}^{n} \{\omega_{i,1}, \ldots . , \omega_{i,m}\}\right) (1.19)$$

La suma de Minkowski producirá $m^n \; puntos,$ los cuales son pasados a la operación de envoltura convexa.

Si, por otro lado, se normaliza la magnitud de la suma de las fuerzas normales de los contactos según la ecuación (1.15):

$$\sum_{i=1}^{n} \|f_{\lrcorner i}\| \leq 1, \;\; (1.20)$$

Entonces se obtendrá la siguiente expresión,W_{L_1}, que muestra el conjunto de *Wrenches* que pueden ser aplicados al objeto:

[3] En geometría, la suma de Minkowski - también conocida como la dilatación - de dos conjuntos A y B en el espacio Euclídeo es el resultado de sumar cada elemento de A con cada elemento de B, esto es, $A + B = \left\{ a + b \middle| a \in A, b \in B \right\}$

$$W_{L_1} = \{\omega \mid \omega = \sum_{i=1}^{n}\sum_{j=1}^{m}\alpha_{i,j}\omega_{i,j}, \quad \alpha_{i,j} \geq 0, \quad \sum_{i=1}^{n}\sum_{j=1}^{m}\alpha_{i,j} \leq 1 \}, \text{ (1.21)}$$

Lo cual es equivalente a:

$$W_{L_1} = ConvexHull\left(\bigcup_{i=1}^{n}\{\omega_{i,1}, \ldots., \omega_{i,m}\}\right), \text{ (1.22)}$$

Este cálculo de envoltura convexa es solamente sobre mn puntos.

Se debe notar que $W_{L_\infty} \supseteq W_{L_1}$ por que el conjunto de puntos sobre los cuales fue calculado la segunda envoltura convexa es un subconjunto del conjunto de puntos que fueron calculados en la primera operación de envoltura convexa.

1.3.2 Construcción en *GraspIt!* del espacio *Wrench* del agarre

Para construir el espacio *Wrench* del agarre (GWS), *GraspIt!* aproxima el cono de fricción de cada punto de contacto con 8 vectores de fuerza que están espaciados simétricamente alrededor del borde del cono. El ángulo máximo de los vectores de fuerza con respecto a la normal de contacto es determinado por el coeficiente de fricción estática μ_s. Ya que este valor puede ser diferente para cada par de materiales, en el simulador se ha permitido que el material de los enlaces y el objeto puedan ser especificados individualmente, incrementando así la flexibilidad y el realismo del sistema. La tabla 1 es una matriz que contiene los coeficientes de fricción que se producen entre diferentes materiales y que actualmente se encuentran definidos en *GraspIt!.* Estos coeficientes de fricción fueron escogidos de acuerdo a los rangos establecidos en [20].

Tabla 1. Valores de los coeficientes de fricción entre los materiales definidos en *GraspIt!* [15]

material	metal	glass	plastic	wood	rubber
metal	0.2	0.2	0.2	0.3	1.0
glass	0.2	0.2	0.2	0.3	1.0
plastic	0.2	0.2	0.3	0.4	1.0
wood	0.3	0.3	0.4	0.4	1.0
rubber	1.0	1.0	1.0	1.0	2.0

El *Wrench 6D* que corresponde a cada vector de fuerza en el borde del cono correspondiente a un contacto se calcula usando el vector que representa el radio desde el centro de masa del objeto al punto de contacto. El componente de par del vector *Wrench* se escala con $1/r_{max}$ para que los elementos del *Wrench* tengan las mismas unidades y para que el tamaño del espacio *Wrench* del agarre no dependa de la escala del objeto.

Después de agrupar todos los *Wrenches* del agarre, el sistema llama a la librería **Qhull** [4] [21] para calcular la envoltura convexa de dimensión 6. Esta envoltura es el espacio *Wrench* unitario del agarre definido bajo la norma L_1. Actualmente, en *GraspIt!* no se construye el espacio W_{L_∞} por el tiempo que tomaría calcular la envoltura convexa de 8^n puntos, donde el numero de contactos n puede ser del orden de 6 a 10 para un agarre medianamente complejo.

1.4 DOS MEDIDAS DE CALIDAD

Para evaluar un agarre se debe primero considerar las propiedades de un buen agarre. En general, un agarre se genera para desempeñar alguna tarea en particular, así que una manera de describir el agarre seria en términos del espacio de los *Wrenches* que pueden ser aplicados al objeto para llevar a cabo esta tarea [15]. En [22] Este espacio es llamado el *espacio Wrench de la tarea* (*task Wrench space*).

El espacio de la tarea podría ser tan simple como el *Wrench* requerido para sostener un objeto en contra de la fuerza de gravedad, con una componente de fuerza en la dirección vertical hacia arriba y ninguna componente de par, o tan complicado como el espacio *Wrench* requerido para una tarea de manipulación.

Si el espacio *Wrench* de la tarea es un sub-espacio del espacio *Wrench* del agarre, entonces el agarre es valido para esa tarea. Sin embargo, no es necesariamente el agarre más eficiente que se puede realizar para la tarea determinada, ya que las fuerzas internas ejercidas sobre el objeto pueden ser mucho más grandes que las necesarias. Por lo tanto, para definir la calidad del agarre un método posible es hallar la proporción entre la magnitud de los *Wrenches* de tarea y la magnitud de las fuerzas de contacto.

Las medidas de calidad más generales asumen que no se sabe nada sobre el espacio *Wrench de la tarea*. En [22] y [19] el espacio *Wrench* de la tarea se modela como una esfera de dimensión 6 ya que se asume que los *Wrenches* de disturbio pueden llegar desde cualquier dirección. En este escenario, la magnitud máxima de cualquier *Wrench* de tarea dentro de esta esfera es igual a su radio. Ya que la calidad del agarre es definida como una relación de magnitudes, ésta puede ser re-definida como:

La medida de calidad es el radio de la esfera más grande, centrada en el origen, y que puede encajarse dentro del espacio unitario del Wrench del agarre.

En la figura 1.9 se muestra un ejemplo del espacio unitario *Wrench* del agarre, al interior de este espacio se observa una esfera que está centrada en el origen de dicho espacio. Esa esfera

[4] Qhull es una librería que, además de otras funciones, calcula envolturas convexas 2-d, 3-d y de dimensiones más altas. Esta envoltura convexa es representada como un conjunto de caras las cuales tienen forma de triángulos.

corresponde a la esfera más grande entre todas las esferas que pueden contenerse al interior de ese espacio y por lo tanto su radio corresponderá a la medida de calidad del agarre.

El radio ϵ_1 que se muestra en la figura 1.9 tiene la misma magnitud al vector, ω_{min}, el cual es el vector mas corto que puede hallarse desde el origen del espacio *Wrench* al borde externo de la envoltura. En *GraspIt!* se define al radio mínimo del *espacio unitario del Wrench del agarre* bajo la norma L_1 como ϵ_1.

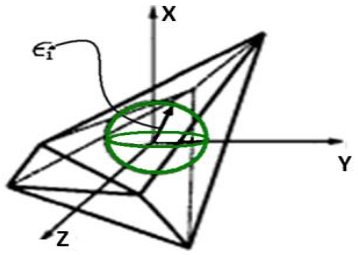


Figura 1.9. Ejemplo de un espacio unitario *Wrench* del agarre.

Cuando se ha construido el espacio *Wrench* del agarre, W_{L_1}, se puede observar que las fuerzas normales de contacto pueden resistir mínimo un *Wrench* de disturbio cuya magnitud sea igual a la magnitud de ω_{min}. Pero si la magnitud del *Wrench* de disturbio es más grande y esta en dirección contraria a ω_{min} las fuerzas normales de contacto se verán obligadas a aumentar su magnitud para poder contrarrestar el disturbio. La medida de calidad ϵ_1 nos dice que tanto deben aumentar estas fuerzas normales de contacto, para lo cual la suma de la magnitud de las fuerzas normales de contacto debe ser $1/\epsilon_1$ veces la magnitud del disturbio *Wrench.* Esto significa que el agarre más eficiente será el que tenga el ϵ_1 más cercano a 1.

En la ecuación (1.13) se puede observar que la magnitud del par depende de d_i, el cual es un vector que representa la distancia desde el origen del espacio *Wrench* al punto de contacto i. Si se cambia el origen del espacio *Wrench* se estará también modificando la distancia d_i lo cual a su ves modifica la magnitud y dirección del par generado en el punto de contacto i. El cambio de magnitud y dirección del par hace que haya un cambio en la magnitud y dirección del *Wrench* generado en el punto de contacto i y por lo tanto también habrá un cambio en el espacio *Wrench* del agarre (según ecuaciones (1.19) y (1.22)). Lo anterior, al igual que en [23], conlleva a concluir que las medidas de calidad varían de-acuerdo la selección del origen del par el cual es el mismo origen del espacio *Wrench* del agarre. Por lo tanto, el volumen del espacio unitario *Wrench* del agarre puede ser usado como una medida de calidad que es invariante a la selección del origen del par. En *GraspIt!* esta medida de calidad, es denotada como $v1$ y corresponde al volumen del espacio unitario *Wrench* del agarre W_{L_1}.

El calculo de los valores de las medidas de calidad resulta fácil una vez se haya construido el espacio *Wrench* del agarre. En *GraspIt!*, la librería **Qhull** reporta directamente el volumen de la envoltura, el cual servirá como el caso promedio de la calidad del agarre y se muestra como la medida de calidad "*v*". El sistema también puede determinar la cara de la envoltura que se encuentra más cercana al origen examinando el valor del *offset* de la ecuación del hiperplano

correspondiente a cada una de las caras. El vector que es normal a esta cara es el *Wrench* más pequeño que se encuentra sobre la superficie del **GWS,** ω_{min}, y es considerado el punto más débil del agarre. La magnitud de este vector (el *offset* del hiperplano) es usado como el peor caso de la medida de calidad ϵ_1 y es mostrado en *GraspIt!* como la medida de calidad "***e****"*.

1.5 VISUALIZACIÓN DEL ESPACIO *WRENCH*

Si bien las medidas de calidad numéricas suministran algún tipo información sobre la eficiencia del agarre, la representación gráfica puede suministrar información adicional. Un objeto en 2D tendrá un espacio *Wrench* 3D; sin embargo, para mostrar la envoltura convexa 6D que resulta de un objeto 3D, esta debe ser proyectada dentro de un espacio 3D, y para ello se deben fijar tres de las coordenadas *Wrench*.

El sistema de *GraspIt!* permite al usuario escoger cual de las coordenadas serán fijas. Si en el espacio *Wrench* se fijan las coordenadas de par a 0, la proyección resultante muestra el espacio de fuerzas que pueden ser aplicadas por el agarre sin ejercer un par neto sobre el objeto; si se fijan las coordenadas de fuerza a 0, se puede visualizar el espacio de par que pueden ser aplicados al objeto sin una fuerza neta actuando sobre él.

Ajustando las coordenadas de par a los valores en el punto sobre el borde de la envoltura que esta más cerca al origen, ω_{min}^{T}, se puede mostrar el espacio de fuerzas que pueden ser aplicadas si se necesita aplicar el peor caso de par. Similarmente se puede mostrar el espacio de par que pueden ser aplicados si debe ser aplicado el peor caso de fuerza, ω_{min}^{f}.

En *GraspIt!* se muestran las componentes del peor caso de disturbio *Wrench* ,$-\omega_{min}$, como un par de indicadores púrpura surgiendo desde el marco de referencia del objeto. Esto ayudará a mostrar en donde es más débil el agarre.

1.5.1 Visualización del espacio *Wrench* usando *GraspIt!*

En los siguientes ejemplos se determinará cuantitativamente la calidad del agarre de distintos objetos utilizando el modelo de la mano del grupo de investigación "**Prótesis robótica de Mano**" expuesto en [14]. Los objetos que serán utilizados para determinar la calidad del agarre son: una esfera, un vaso de café y un zapato.

Cuando la mano agarra a un objeto, en la ventana de *GraspIt!* se muestran las medidas de calidad $(\epsilon_1, v1)$ y la envoltura convexa que se genera en cada agarre.

En la figura 1.11, para cada uno de los objetos ejemplo, (A), (B), (C), (D), en la ventana al lado derecho, donde se muestra la mano, se pueden ver las medidas de calidad “e” y “v”, que corresponden a la magnitud del *Wrench* más pequeño con respecto a la superficie de la unidad del Espacio *Wrench* del Agarre y el volumen de la unidad de dicho espacio, respectivamente. Los indicadores de color púrpura que se muestran al interior del objeto, que está siendo agarrado, revelan los componentes de fuerza y par del peor caso de disturbio que se puede llegar a tener en la configuración de agarre actual. La ventana que se muestra en la parte de arriba-izquierda es una proyección del Espacio *Wrench* del agarre que muestra el espacio de fuerzas que pueden ser aplicados al objeto creando un par neto igual a cero. Para obtener esta proyección lo que se debe hacer es fijar las tres coordenadas de par a cero como se muestra en la figura 1.10.

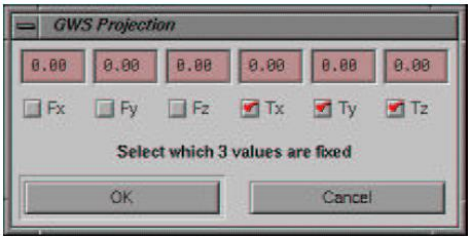
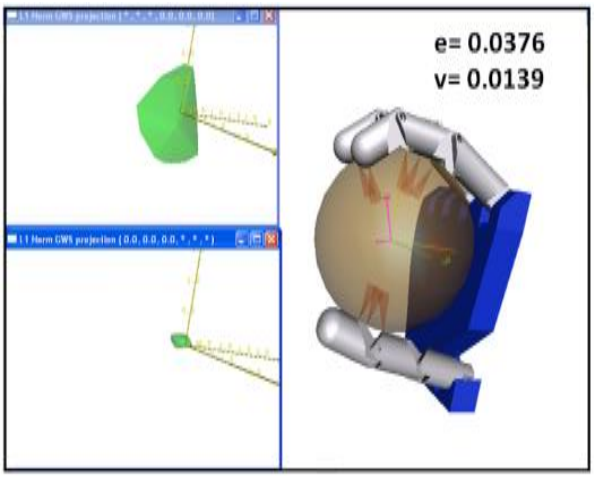


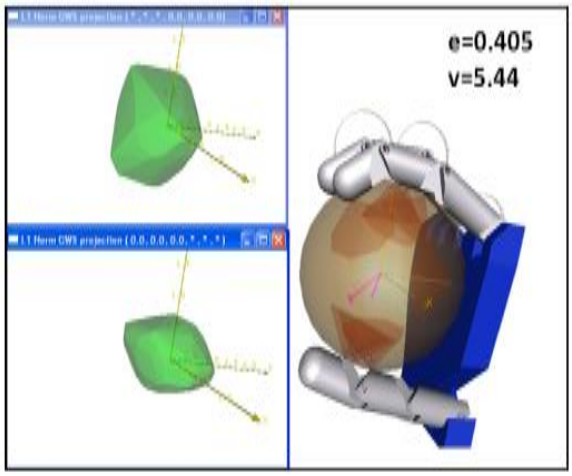
Figura 1.10. Ventana de diálogo para crear el espacio *Wrench* del agarre. [15]

La ventana de la parte de abajo-izquierda es una proyección que muestra el espacio de par que pueden ser aplicados sin aplicar ninguna fuerza sobre el objeto. Para obtener esta proyección se deben fijar las coordenadas f_x, f_y, f_z, que aparecen en la figura 1.10, a cero. En la figura 1.11 C), la mano esta agarrando un vaso de café. Los indicadores purpuras muestran que el agarre es más débil cuando una fuerza de disturbio es aplicada a lo largo de un eje que es casi perpendicular a la palma de la mano y cuando un par es aplicado en una dirección casi vertical (con relación a la vertical del vaso). En la ventana de arriba-izquierda se puede observar la proyección del espacio *Wrench* del espacio de fuerzas actuando sobre el objeto, en este espacio se observa que mientras el agarre puede aplicar una variedad de fuerzas en el plano horizontal, este no puede aplicar grandes fuerzas en la dirección vertical. Esto es porque los contactos están organizados alrededor del perímetro del vaso, es decir, que no existe ningún contacto en la parte de arriba o abajo del vaso. Por lo tanto, el agarre requerirá grandes fuerzas de fricción para resistir una fuerza de disturbio en la dirección vertical. En la figura 1.11 a) se muestra la mano agarrando una esfera de metal. En este modelo de la mano cada uno de los enlaces son del mismo material de la esfera. Si observamos la tabla 1 el coeficiente de fricción que existe entre estos dos materiales es de 0.2 y como se mencionó anteriormente, este coeficiente de fricción afecta directamente el radio del cono de fricción. Ahora si se observa la figura 1.11 B), en donde la esfera se asume de goma y los enlaces de la mano siguen siendo de metal, se puede observar que el radio del cono de fricción se ha incrementado, esto se debe a que el coeficiente de fricción entre estos dos materiales se incremento a 1.0. Una vez se incrementa el coeficiente de fricción entre los contactos, el agarre puede soportar fuerzas de disturbio mayores y la calidad del agarre tiene un incremento bastante significativo. Este incremento también se ve reflejado en el espacio de fuerzas y par que actúan sobre el objeto

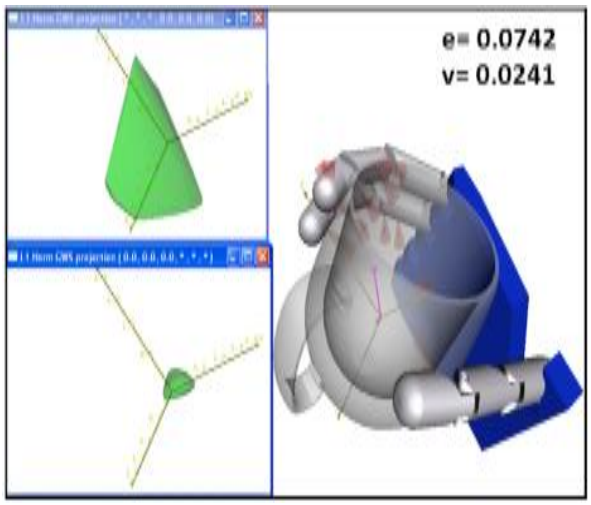
como se muestra en las dos ventanas de la parte izquierda de las figuras 1.11 B) y C). En la figura 1.11 D) se muestra el agarre de un zapato y las medidas de calidad correspondientes.



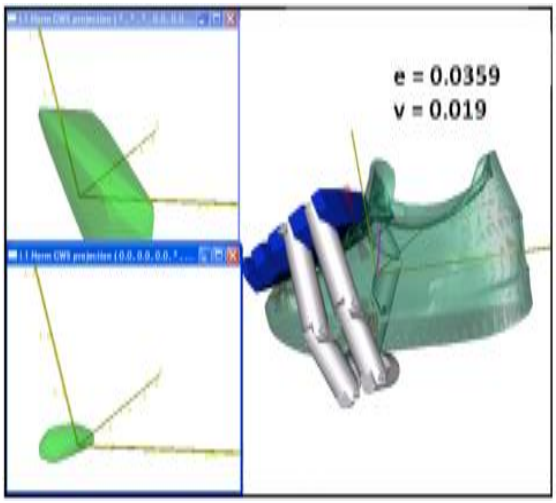
(A): Modelo de la mano del grupo agarrando una esfera de vidrio



(B): Modelo de la mano del grupo agarrando una esfera de goma



(C): Modelo de la mano del grupo garrando un vaso de café



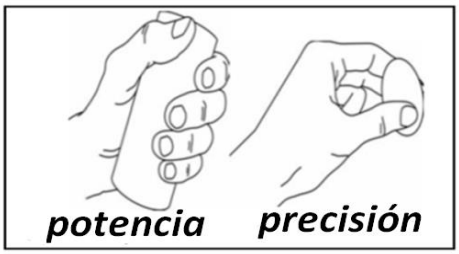
(D): Modelo de la mano del grupo agarrando zapato

Figura 1.11. Agarres de diferentes objetos con una mano robótica.
La mano tiene las mismas características a la desarrollada en [14].

2. PLANIFICACIÓN DEL AGARRE ESTABLE EN UNA MANO ROBÓTICA

En el capitulo anterior se estudia el efecto de aplicar fuerzas y pares en los diferentes puntos de contacto sobre el objeto. El número de estos puntos de contacto y la forma de cómo se distribuyen alrededor del objeto depende, en cierta medida, del tipo de agarre que se esta realizando.

El tipo de agarre de un objeto se puede clasificar de diferentes formas. Según Napier [26] los agarres pueden ser de dos tipos: agarre de poder (potencia) y agarre de precisión (ver figura 2.1 a)). Un agarre de precisión es utilizado para agarrar un objeto únicamente con la punta de los dedos. Esto permite que el objeto sea manipulado más fácilmente, ya que la punta de los dedos puede desplazarse en cualquier dirección a lo largo de la superficie del objeto. El agarre de poder rodea el objeto de tal manera se produzcan contactos con cualquier parte de los dedos y posiblemente con la palma de la mano. Este tipo de agarre es usado para aumentar las probabilidades de un agarre estable. La clasificación de Schelsinger [27] es más extensa, y clasifica los agarres en seis tipos: agarre cilíndrico, palmar, de punta, lateral, esférico y de gancho (ver figura 2.1 b)). Estos tipos de agarre pueden encajarse en una de las clasificaciones de Napier. Los agarres de tipo palmar y de punta tienen características de un agarre de precisión, mientras que un agarre cilíndrico tiene características de un agarre de poder. En [28] se fusiona la clasificación que hace Napier y Schelsinger y describe que los tipos de agarre esférico y cilíndrico pueden ser de precisión o de poder. Por ejemplo, un agarre tipo cilíndrico es de poder si la persona acerca la mano de tal manera que los dedos contacten el objeto no solo con su punta. Por el contrario un agarre tipo cilíndrico es de precisión si la persona acerca la mano al objeto de tal manera que solamente la punta de los dedos contacten el objeto.



a) Clasificación según Napier [26]

b) Clasificación según Schelsinger [27]

Figura 2.1. Tipos De Agarre

Para hacer del agarre una operación correcta, no solo se debe considerar el tipo de agarre que se desea realizar, sino que también se necesita que se cumplan algunos criterios fundamentales: el agarre debe ser estable, el agarre debe ser alcanzable y el agarre debe estar libre de colisión [24]. Un agarre estable realizado por una mano robótica es aquel en el cual las

fuerzas ejercidas por los dedos pueden balancear par y fuerzas arbitrarias ejercidas sobre el objeto [18]. Un agarre libre de colisión es cuando solo se producen contactos entre la mano y el objeto que se desea agarrar. Un sistema de planificación del agarre debe cumplir con los anteriores criterios fundamentales y según se expone en [25] para lograrlo se necesita de la combinación de varios tipos de movimientos por parte de los dedos.

En una mano robótica la planificación del agarre puede ser vista como la fase previa al agarre, conocida como fase de pre-agarre, la cual se representa como un conjunto de vectores o arreglos que contienen las secuencias angulares de las articulaciones de cada dedo antes de contactar el objeto [29]. Antes de iniciar la fase de pre-agarre la mano es configurada a una postura inicial que depende del tipo de agarre que se desea realizar sobre el objeto [30]. En muchos trabajos esta postura inicial es denominada como *Preshape de la mano* (***hand preshape***) o *Preshape del agarre* (***grasp preshape***). En [1] el *Preshape* del agarre es denominado como postura de pre-agarre. Esta postura es identificada con el mismo nombre del tipo de agarre que produce. Por ejemplo, una postura de pre-agarre esférica produce un agarre esférico sobre el objeto.

Para identificar la postura de pre-agarre se debe conocer, parcial o totalmente, la geometría del objeto y de esa manera posicionar las articulaciones de los dedos para producir el agarre. Una versión simplificada de la geometría del objeto se puede obtener utilizando sensores. Los investigadores han desarrollado una gran variedad de planificadores haciendo usos de sensores que se pueden clasificar en tres tipos:

- Planificadores que utilizan sensores de visión
- Planificadores que utilizan sensores de contacto.
- Planificadores que utilizan sensores ópticos o de campo eléctrico.

La característica esencial que diferencia a los tres tipos de sensores es la amplitud del alcance. Así, los sensores de visión se consideran de largo alcance, los táctiles o de contacto de corto y de medio alcance los ópticos y de campo eléctrico. En general, la arquitectura de los planificadores depende del tipo de sensor. A continuación se hace una descripción de diferentes trabajos realizados sobre planificación del agarre y que encajan en alguna de las categorías de los sensores mencionados.

2.1 ANTECEDENTES DE LA PLANIFICACIÓN DEL AGARRE

2.1.1 Planificadores que utilizan sensores de visión

Entre los enfoques más comunes que utilizan los sensores de visión se encuentra el propuesto por [1]. En este enfoque se usa un subsistema de visión para obtener una versión simplificada

del objeto, que consiste únicamente de un conjunto de formas primitivas: esferas, cilindros, conos y cajas. De acuerdo a la versión simplificada del objeto, el planificador genera un conjunto de posiciones iniciales de agarre. Cada posición inicial de agarre consiste de una postura de pre-agarre y de una posición y orientación de la palma de la mano.

Otros enfoques, como los propuestos por [2], [3], [4], usan componentes de visión para hacer una reconstrucción parcial del objeto y sobre su contorno determinan varios conjuntos de puntos de contacto que garantizan un buen balance entre fuerza y par. Lo siguiente que hacen es chequear cada uno de los conjuntos de puntos encontrados y por medio de una medida de calidad del agarre, determinan cuál de estos conjuntos es el más apropiado para ser utilizado. En esta categoría también se encuentran los enfoques de planificación que utilizan bases de datos que contienen posturas de pre-agarre ya definidas y que, de acuerdo a la geometría reconstruida del objeto, se usa solo una de ellas [5], [1], [6]. También se encuentran planificadores que no reconstruyen totalmente el objeto si no que toman imágenes para ubicar puntos de contacto estratégicos que podrían resultar en un agarre estable [7]. Esta clase de enfoques le dan al sistema una mayor velocidad de respuesta, ya que no se hace una reconstrucción total del objeto sino que directamente se usa la imagen para ubicar los puntos de contacto. En un enfoque distinto encontramos los planificadores de agarre implementados en prótesis de manos robóticas. Estos planificadores hacen uso de la visión del paciente para generar señales electromiografías, las cuales, por medio de un sistema de clasificación, determinan el tipo de agarre que pretende realizar el paciente [31].

2.1.2 Planificadores que utilizan sensores de contacto

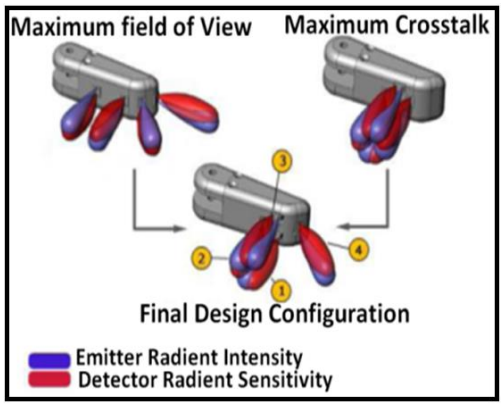
La idea general en la que se basan esta clase de planificadores es llevar los dedos hasta ponerlos en contacto con el objeto y por medio de técnicas empíricas o analíticas moverlos por la superficie del objeto para maximizar alguna medida de calidad. Limitándose a los enfoques que usan técnicas empíricas se encuentra el trabajo desarrollado por [8]. En este trabajo la tarea de agarre y manipulación son simplificadas como una serie de comportamientos tales como: mover el dedo siguiendo la superficie del objeto, mover la mano hasta que haya contacto con el objeto, o mover la mano debajo del objeto mientras se mantiene contacto. Normalmente para seleccionar el tipo comportamiento se acude a un mecanismo de toma de decisiones de alto nivel que selecciona los comportamientos correctos a ejecutar [9]. Una manera de lograrlo es con la ayuda de técnicas de aprendizaje como: el aprendizaje por refuerzo (reinforcement learning) propuesto por [32]. Otro trabajo en el cual se exponen técnicas empíricas es el enfoque basado en control como lo propone Coelho [10]. En su trabajo formula el agarre como un problema de control robusto donde las variaciones de las propiedades como la geometría y la fricción de contacto son modeladas como disturbios. El controlador de agarre es diseñado para ser robusto a estas variaciones. El enfoque propuesto por *Coelho* expresa el proceso de planificación del agarre en la forma de procesos de control lineal múltiples y asíncronos.

2.1.3 Planificadores que utilizan sensores ópticos o de campo eléctrico

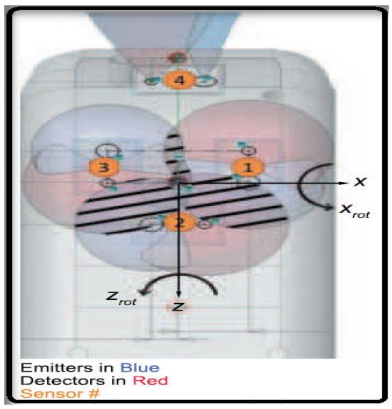
La idea general en la cual se basan este tipo de planificadores es la que se toma como referencia para realizar este trabajo de grado.

Recientemente se ha propuesto un nuevo enfoque de planificación conocido como el enfoque de pre-contacto (pretouch). Éste se caracteriza por la utilización de sensores de medio alcance (sensores ópticos y de campo eléctrico) ubicados en la punta de los dedos de una mano robótica. En la figura 2.2. se muestran las diferentes configuraciones de sensores ópticos ubicados en la punta de los dedos de una mano robótica. En la figura 2.2 a) se muestran las diferentes configuraciones que pueden usarse. Si los sensores están demasiado separados generan un gran campo de visión (*máximum field of view*) permitiendo detectar obstáculos. Esta separación reduce el traslape de las señales lo cual incrementa la señal de ruido. Si los sensores están muy cercanos (*máximum crosstalk*) aumenta el traslape, pero se incrementa la posibilidad de contactar un obstáculo. Una configuración final, como se muestra en la figura 2.2 b), organiza los sensores de tal manera que formen un triangulo manteniendo un traslape o cruce entre las señales. Un sensor adicional se ubica en la punta de los dedos para que se puedan detectar obstáculos. El área de color azul es la señal emitida por los sensores, el área de color rojo es la señal detectada y el área a rayas es la región de cruce entre las señales.

Al ubicar estos sensores en la punta de cada dedo se obtienen dos medidas: la distancia hasta el objeto (desde el centro de los tres sensores) y la orientación de la superficie del objeto vista localmente por la punta del dedo. La primera medida es usada por el planificador para acercar los dedos al objeto de tal manera que todos ellos lo toquen al mismo tiempo. La segunda medida es usada para que cada dedo forme su propia postura de agarre y así entre todos contribuyan a formar la postura final de agarre. En [11] se propone el uso de un controlador reactivo y sensores de proximidad ópticos. Los datos de estos sensores son interpretados usando modelos que han sido obtenidos empíricamente. Estos modelos son usados por un algoritmo, el cual estima la distancia y orientación de la superficie del objeto, y con esta información ajusta la postura del agarre, que al hacerla corresponder con la orientación de la superficie del objeto, incrementa la probabilidad de un agarre estable. La arquitectura de control es jerárquica, compuesta de dos elementos de control reactivos que se ejecutan de forma asíncrona y controlan aspectos separados del proceso de agarre. En el nivel inferior, el controlador de distancia de los dedos asegura que éstos se mantengan a una distancia aproximadamente igual con relación a la superficie del objeto. En la parte superior un controlador de acondicionamiento cinemático controla que el brazo lleve la mano robótica al centro del objeto y que la punta de los dedos quede paralela a la superficie local del objeto. También en [33] y [34] se exponen enfoques que usan el sensado de pre-contacto. Estos enfoques son básicamente iguales al enfoque visto anteriormente. La principal diferencia de estos enfoques es que en vez de usar sensores ópticos utilizan sensores de campo eléctrico para el alineamiento y el formado de la postura de la mano robótica.



a) Diferentes configuraciones de los sensores ópticos en la punta de un dedo.



b) Configuración final de los sensores ópticos.

Figura 2.1. Configuraciones de los sensores ópticos según el enfoque de pre-contacto. [11]

2.2 SISTEMA PROPUESTO DE PLANIFICACIÓN DEL AGARRE BASADO EN MULTI-AGENTES

2.2.1 Multi-agentes en el control del agarre y la manipulación

Como los humanos, los robots necesitan un cierto nivel de autonomía, reactividad, pro-actividad y habilidad social para desarrollar sus tareas [12]. Una mano robótica multi-dedo necesita autonomía para que poder enfrentarse ante situaciones imprevistas, como: el agarre de objetos desconocidos, el deslizamiento del objeto cuando se está agarrando, perturbaciones por parte del ambiente y muchos otros inconvenientes que se pueden dar al tratar de agarrar un objeto. La reactividad en una mano robótica es la respuesta de ésta en tiempo real ante eventos críticos de tiempo para realizar un buen agarre. Por otro lado la pro-actividad está representada en la persecución de un agarre estable y en caso de fallas la mano siempre intentara conseguir su objetivo. En una mano, la habilidad social se puede ver cuando se la toma como un conjunto de dedos unidos a una palma que interactúan entre si para llevar a cabo un agarre. Estas características son normalmente citadas como razón para adoptar la tecnología de agentes [12]. Precisamente, una de las principales características de un agente es ser autónomo, en un ambiente dinámico. Pero en esta clase de ambientes, normalmente, existe más de un agente, dando lugar a los sistemas multi-agente que han demostrado ser de gran ayuda para resolver problemas de gran complejidad. Dado que un problema complejo, es divido sub-problemas que son resueltos por los agentes individualmente, más fácilmente, obteniéndose así sistemas más autónomos, robustos, escalables y modulares [12]. Aprovechando las cualidades que ofrece la tecnología de agentes, muchos investigadores de robótica la han aplicado para hacer sus

robots más inteligentes. Específicamente los robots son diseñados para que los procesos de planificación y control sean distribuidos en varios agentes quienes se encargan de una parte de cada uno de estos sub-procesos.

En el campo de las manos robóticas son pocos los trabajos con agentes para el agarre y manipulación de objetos. En [35] se describe un enfoque multi-agente para el agarre que combina evasión de obstáculo y estrategias de *Visual Servoing*. El control es distribuido entre cinco tipos diferentes de agentes: agentes de enlace, agentes de articulación, agentes de efector final, agentes de tarea y agentes de objeto. Como cada agente intenta conseguir un comportamiento individual deseado, el manipulador mismo presenta un comportamiento emergente que evita obstáculos mientras se acerca al objeto a ser agarrado. Cuando el objeto es alcanzado, los agentes de objeto que representan el objeto a ser agarrado y el efector que desarrollan el agarre son instanciados. La instanciación de estos dos agentes, mientras son guiados por realimentación visual, automáticamente cambian la estructura del control del manipulador. Una vez ambos agentes están activos, durante la etapa final del agarre, se realiza el control del efector final en malla cerrada con *Visual Servoing*. Por su parte en [36] se presenta una arquitectura multi-agente para controlar un robot multi-dedo. En esta arquitectura existen dos niveles de tareas, uno bajo y otro alto. Cada unidad de control, llamada agente, esta compuesta de los dos niveles de tarea. La tarea de nivel bajo se encarga del desarrollo de cálculos en tiempo real mientras que la tarea de nivel alto se encarga de la comunicación con otros agentes. Él sistema es implementado en 4 procesadores y se plantea una red de agentes para las tareas de agarre. En la manipulación de objetos cabe mencionar el trabajo realizado en [62]. En su resumen se menciona que las puntas de los dedos de una mano robótica hábil pueden ser tratadas como agentes de manipulación con restricciones de movimiento. En su propuesta, durante la manipulación los agentes deben trasladar los puntos de contacto para reconfigurar el agarre.

Las propuestas mencionadas se enfocan hacia el control del agarre y la manipulación. Los trabajos realizados hasta ahora en planificación propiamente dicha del agarre con sistemas multi-agentes han sido muy limitados, una de las razones posiblemente porque las manos robóticas hasta hace poco construidas no eran lo suficientemente hábiles para permitir que sus dedos tuvieran cierto grado de habilidad y autonomía, impidiendo pensar en las habilidades de cada dedo como un agente autónomo que contribuye junto con los otros dedos al agarre. Por otra parte, el surgimiento del enfoque de pre-contacto (ver sección 2.1.3), proporciona a cada dedo un nuevo sentido mediante la incrustación de uno o varios sensores de pre-contacto, permitiendo a cada dedo disponer de información importante, como distancia y orientación de la ubicación en donde se establecerá contacto con el objeto a ser agarrado. Con esta información, cada dedo es capaz de acercarse al objeto, determinar el mejor lugar de contacto y a partir de la información de la forma del objeto generar una muy buena postura basada en el objeto real y no en algún tipo de modelo, como ocurre con los enfoques basados en primitivas. Bajo esta premisa, cada dedo de la mano tiene la posibilidad de poseer autonomía y es posible aplicar la tecnología de los sistemas multi-agente en la planificación del agarre.

Bajo este enfoque de planificación del agarre, cada dedo se encarga de una parte del problema del agarre y coopera con los otros dedos para cumplir con un objetivo común que es el agarre estable de un objeto. Otra ventaja es el uso de control distribuido, haciendo al sistema más robusto, adaptativo y tolerante a fallas, ya que no hay una fuerte dependencia con algún ente específico en particular, dando la descentralización una mayor velocidad de respuesta [37]. Este enfoque basado en multi-agentes para la planificación de agarre es propuesto en el presente trabajo.

2.2.2 Antecedentes

En la actualidad el grupo de investigación en Automática Industrial de la Universidad del Cauca desarrolla el proyecto "**Prótesis robótica de Mano**". El objetivo del grupo es construir un prototipo de prótesis de mano controlada a partir de señales electromiográficas captadas del propio paciente, que permite devolver funciones específicas de agarre y movimiento. En la actualidad se han desarrollado algunos trabajos que aportan un conocimiento importante al proyecto "Prótesis robótica de Mano" mencionado. En el trabajo de maestría desarrollado por Cabrera, [29], se propone un modelo de los componentes del simulador de una prótesis de mano robótica, tratando de garantizar modularidad del sistema. El modelo general de análisis tiene como finalidad detallar las interfaces en cada uno de los componentes sin especificar su implementación. Debido a su complejidad el modelo se fragmenta en 7 sub-módulos. El Sub**-módulo Controlador del Dispositivo** (MCD) realiza en esencia dos acciones: ubicar los dedos en las posiciones deseadas según el tipo de agarre y posteriormente entrar en contacto con el objeto, posiblemente controlando la fuerza. Específicamente, este sub-módulo recibe como entrada la información del tipo de agarre que desea realizar el paciente y las señales que indican las posiciones articulares de cada dedo, así como la detección de deslizamiento y la fuerza ejercida sobre el objeto. Estas señales sirven para controlar los movimientos y la fuerza del agarre a través de los motores instalados en el dispositivo. La información necesaria para la determinación de la geometría y puntos de contacto con el objeto se asume que se adquiere a través de la visión del paciente, quien es el responsable de generar las señales mioeléctricas a través de los músculos del antebrazo. A partir de las señales ME se reconoce la intención del movimiento y es aquí donde el sistema de control comienza a trabajar.

En el módulo controlador del dispositivo se hace énfasis, principalmente, en dos fases: la fase previa del agarre, o pre-agarre, y la fase del agarre [25]. Cada opción de agarre representa un movimiento de los dedos de la mano como extensión, flexión, oposición, etc. La combinación de varios tipos de movimientos que deben realizar los dedos para asir un objeto define una planificación del agarre. Cuando la geometría del objeto se especifica, es necesario definir o calcular las trayectorias aproximadas de los dedos que contactan con el objeto. En esta fase debe identificarse el tipo de agarre a ser empleado. La segunda fase corresponde a la fase del agarre, que consiste en el agarre propiamente dicho. Empieza en el momento que la mano toca y sostiene el objeto de una manera estable. En este trabajo se propone una solución a la primera fase, esto es, a partir de la intención de movimiento recibida del modulo gestor de

reconocimiento de intención de movimiento (MGRIM) generar las trayectorias de las articulaciones de los dedos para lograr el contacto con el objeto y obtener un agarre estable.

La intención de movimiento se asume ya ha sido identificada y al igual que en [29] a cada uno de los tipos de agarre se le asigna un numero tal y como se muestra en la tabla 2.

Tabla 2. Valores que identifican el tipo de agarre. [29]

Tipo	Palmar.	Cilíndrico	Gancho	Punta	Lateral	Esférico	Reservado	Reservado
valor	1	2	4	8	16	32	64	128

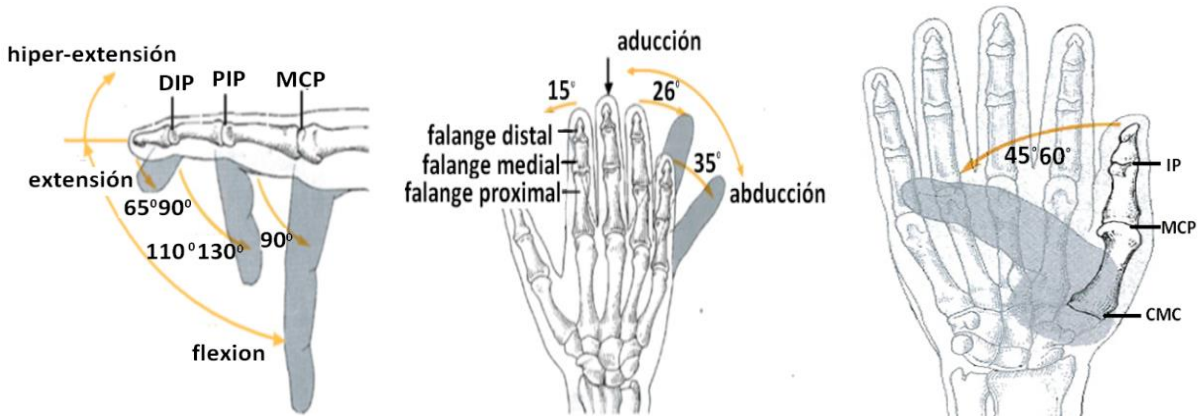
En la tabla anterior los tipos de agarre expuestos por Napier son confinados en un valor entero de un *byte.* A cada *bit* se le asigna un tipo de agarre. Así por ejemplo el tipo de garre palmar es representado por el número 1, el tipo de agarre cilíndrico por el número 2, y así sucesivamente. Las dos últimas columnas de la tabla se dejan como valores reservados los cuales por ahora solo sirven para completar el *byte.*

2.2.3 El enfoque propuesto

El enfoque para la planificación del agarre abordado en el presente trabajo está basado en multi-agentes y utiliza sensores de pre-contacto en su diseño. El enfoque asume que se cumplen las siguientes condiciones:

En este trabajo se propone una mano robótica que realice movimientos parecidos a los de la mano humana. En la figura 2.3 a) se muestra el movimiento de flexión-extensión realizado por la mano humana. Este movimiento consiste básicamente de una apertura y cierre de la mano. El eje que se toma como referencia para describir este tipo de movimientos es el que se genera al extender el dedo. Cuando el dedo se encuentra extendido se considera que el ángulo que forman las articulaciones interfálangica proximal (PIP), interfálangica distal (DIP) y metacarpofalangica (MCP) con el eje de referencia es igual a cero. Si este ángulo comienza a incrementarse en cada una de las articulaciones se dice entonces que se está realizando un movimiento de flexión. Si por el contrario este ángulo disminuye entonces se está realizando un movimiento de extensión. En la figura 2.3 b) se muestra el movimiento de aducción-abducción. El eje de referencia para describir este tipo de movimientos es el dedo medio. Si aumenta el ángulo de separación de algún dedo con relación al dedo medio entonces se habla de un movimiento de abducción. Por el contrario si este ángulo disminuye se habla de un movimiento de aducción. En la figura 2.3 c) se muestra el movimiento de oposición del dedo pulgar. La articulación carpometacarpiana (CMC) es la encargada de realizar esta clase de movimiento de tal manera que el dedo pulgar se oponga siempre al dedo meñique. Las articulaciones: interfalangica (IP) y metacarpofalángica (MCP) son las encargadas de realizar los movimientos

de flexión-extensión del dedo pulgar. Por último, los ángulos que se muestran en la figura 2.3 son los limites articulares descritos para cada uno de los movimientos de la mano humana.



a) flexión-extensión b) aducción-abducción c) oposición

Figura 2.3. Movimientos de los dedos que se proponen para la mano robótica. [39]

- En este trabajo se propone la siguiente configuración: una mano de cuatro dedos (pulgar, índice, medio y anular). Cada dedo con tres grados de libertad (GDL). Los dedos: índice, medio y anular usan dos GDL, ubicados en las articulaciones (PIP) y (MCP), para realizar el movimiento de flexión-extensión. El último GDL, ubicado en la articulación (MCP), realiza el movimiento de aducción-abducción. El dedo pulgar tiene la capacidad de realizar los movimientos de oposición y flexión-extensión. El movimiento de flexión-extensión se realiza gracias a los dos GDL ubicados en las articulaciones (IP) y (MCP). La articulación (CMC) tiene un GDL adicional que realiza el movimiento de oposición.

  Los ángulos que se muestran en la figura 2.3, describen los límites articulares de cada uno de los dedos de la mano robótica.

- El paciente es el encargado de generar la señal mioeléctrica que determina el tipo de agarre y de posicionar la prótesis de mano robótica cerca del objeto de manera tal que pueda agarrarlo. La etapa de clasificación de las señales mioeléctricas no será tenida en cuenta en este trabajo. Se asume que al planificador llega el dato (un número) que describe el tipo de agarre deseado (ver tabla 2).

- Cada uno de los dedos: índice, medio y anular, contará con 4 sensores ópticos. 3 sensores se ubicarán en la punta del dedo para detectar la distancia y orientación de la superficie que está en frente de cada uno de ellos (superficie local del objeto). Un sensor adicional se colocará en la falange proximal para detectar la distancia desde ésta hasta el objeto. El dedo pulgar contará con dos sensores ópticos ubicados en su punta y falange proximal. Las características de estos sensores y su configuración son básicamente iguales a las encontradas en [11].

- Se asume que la mano robótica está cubierta de un material que se deforma debido a la presencia de fuerzas de contacto. Esta consideración se hace debido a que el tipo de contacto (contacto *soft-finger* [38]) que se genera con una superficie deformable aumenta la fuerza de rozamiento y por consiguiente disminuye las probabilidades de que el objeto se resbale. Este tipo de contacto se explica con más detalle en la sección 1.1.3.

Los sensores pueden generar grandes cantidades de datos, lo que dificulta su manejo en tiempo real [40]. Por esta razón se propone una arquitectura de control distribuido, asumiendo cada dedo como una única entidad con características de autonomía y reactividad. Cada dedo determina su comportamiento (la posición de sus articulaciones) de manera autónoma.

El funcionamiento del planificador se puede resumir en los siguientes ítems:

- Inicialmente le llega el valor del tipo de agarre que se desea realizar. Este valor se convierte en una postura de pre-agarre. Así por ejemplo, si el valor es 2, correspondiente al agarre cilíndrico, la postura de pre-agarre que formará la mano será la cilíndrica (ver tabla 2). El valor de la posición de las articulaciones para cada una de las posturas de pre-agarre está contenida en una base de datos.

- Luego de alcanzar la postura de pre-agarre se inician en paralelo dos movimientos. El primero es el movimiento de flexión. La heurística usada para realizar este movimiento es el lograr que los dedos toquen la superficie del objeto al mismo tiempo. Con esto se evita que el objeto se desplace debido al primer contacto de alguno de ellos. EL movimiento de flexión de los dedos se detiene cuando se detecta contacto con el objeto o con alguna parte de la superficie de la propia mano robótica. El segundo movimiento se realiza si se cumple que la distancia[5] de la punta de los dedos al objeto es menor que un valor, determinado por las características del sensor. Es este un movimiento de aducción-abducción con el fin de lograr que la superficie de la punta de los dedos quede lo más paralela posible a la superficie local del objeto. El movimiento de aducción-abducción finaliza en el instante en que se detecta una distancia[6] umbral de los dedos al objeto, menor o igual a 0.5 cm. Esta distancia es para evitar colisión con algún dedo, que produzca un desplazamiento del objeto, dando como posible resultado un agarre infructuoso.

- Para generar una postura de agarre más simétrica alrededor del objeto el movimiento del dedo pulgar siempre se opone al dedo anular, ya que la mano robótica considerada en este trabajo no tiene dedo meñique [39]. Para lograr la oposición, el dedo pulgar re-configura su articulación de oposición cada vez que el dedo anular realiza movimientos de aducción-abducción.

[5]En [11] esta distancia es de 3.5 cm, debido a los sensores usados

[6]Esta distancia es igual a 0.5 cm, debido a que si se realizan movimientos de aducción-abducción en una distancia menor a ésta, puede ocurrir un desplazamiento del objeto.

3. AGENTES Y SISTEMAS MULTI-AGENTE

En este capítulo se describe de forma general el significado de agente, sistemas multi-agente y las posibles arquitecturas que pueden encontrarse en un sistema diseñado con agentes. Al final, se hace un breve resumen de algunas metodologías usadas para el diseño de sistemas multi-agente y se escoge la metodología utilizada en el diseño del planificador del agarre.

3.1 AGENTES INTELIGENTES

No existe un consenso universal sobre la definición de agente [41]. Según [42] un agente es: "*un sistema autónomo situado dentro y haciendo* par*te de un ambiente, que percibe este ambiente y actúa sobre él".* Esta definición abarca mucho de lo que se entiende por agente, y cubre diversos tipos de agentes, según su aplicación: agentes robóticos, agentes computacionales, agentes biológicos, etc. Otra definición también ampliamente adoptada, es la que se da en [43], que define a un agente como: "*un sistema computacional que está situado en cierto ambiente, y que es capaz de ejercer acciones autónomas en este ambiente* para *llevar a cabo sus objetivos de diseño*". Estas dos definiciones tienen las dos propiedades básicas de los agentes: **La autonomía** y estar **situados** en un ambiente. La primera propiedad significa que los agentes son independientes y que toman sus propias decisiones. La segunda propiedad no restringe la definición de agente, ya que virtualmente todo software debe estar situado en un ambiente, la diferencia radica en el tipo de ambiente que los agentes habitan. Los agentes son normalmente utilizados en ambientes difíciles; más específicamente, en ambientes *dinámicos, impredecibles* y poco *fiables*. Los ambientes *dinámicos* son aquellos que cambian rápidamente, y los agentes deben adaptarse a estos cambios cuando están intentando lograr un objetivo. Los ambientes *impredecibles* son aquellos en los cuales no es posible predecir sus estados futuros; a menudo esto es posible debido a que los agentes no disponen de la información correcta y completa o porque el ambiente está siendo modificado más allá de la influencia y del conocimiento del agente. Los ambientes *poco fiables*, son aquellos en los que las acciones que un agente realiza pueden fallar por razones que se escapan del control del agente. Por ejemplo, un robot que intenta levantar un objeto puede fallar por varias razones, entre ellas por que el objeto es demasiado pesado. Los agentes necesitan ser ***Reactivos***. Es decir, responder rápidamente ante cambios en su ambiente. Por ejemplo, un agente que controla un robot que esta jugando futbol, puede realizar planes basados en la posición actual del balón y de otros jugadores. Pero debe estar preparado para adaptarse o abandonar sus planes debido a los movimientos del balón o de los jugadores. Otra propiedad clave de los agentes es que persiguen objetivos a través del tiempo, es decir, son ***proactivos****.* Esta característica es muy útil ya que un agente tratara de cumplir con sus objetivos a través de la ejecución de un plan. Si este plan falla el agente intentará con otro plan de tal manera que se dé cumplimiento al objetivo planteado. En general, un agente debe balancear su reactividad y pro-actividad. Si el agente es demasiado reactivo, entonces constantemente estará ajustando sus planes y no logrará nunca sus objetivos. Por otro lado, si el agente no es lo suficientemente reactivo,

entonces gastará tiempo tratando de seguir planes que no tienen gran relevancia o aplicabilidad. Debido a que las acciones y los planes pueden fallar los agentes deben ser capaces de recuperarse de tales fallas, es decir, deben ser **robustos**. Un enfoque natural para lograr la robustez es ser **flexible**. Lo anterior significa, tener un amplio rango de maneras de lograr un objetivo dado, es decir, que el agente tiene alternativas que pueden ser usadas si un plan falla.

Finalmente, los agentes casi siempre necesitan interactuar con otros agentes, es decir, ser **sociables.** Frecuentemente estas interacciones se encuentran en un alto nivel: en lugar de decir que los agentes intercambian mensajes, la interacción de los agentes se enmarca en términos de compromisos, tales como 'informe', 'petición', 'estar de acuerdo' etc. De esta manera, la interacción entre agentes es vista como algún tipo de interacción humana tal como negociación, coordinación, cooperación y trabajo en equipo. Con base en las consideraciones presentadas, un agente puede ser visto como **una pieza de software** que es:

- **Situado:** existe en un ambiente.
- **Autónomo**: independiente, no controlado externamente
- **Reactivo**: responde a los cambios en su ambiente de una manera oportuna.
- **Proactivo**: persistente en la consecución de sus objetivos.
- **Flexible**: tiene varias maneras de conseguir sus objetivos.
- **Robusto:** se recupera de las fallas.
- **Social:** interactúa con otros agentes.

En la persecución de sus objetivos, se quiere que los agentes sean **racionales.** Parte de ser racional es que un agente no haga cosas "tontas" tal como mandar a realizar dos acciones que se contradicen. Un análisis detallado de lo que significa ser "**racional**" puede ser encontrado en [44]. La racionalidad es la base del modelo **BDI** (Belief-Desire-Intention) [45]. En este modelo la toma de decisiones se basa en un proceso de razonamiento que parte de las creencias que el agente tiene del mundo para realizar unas determinadas acciones. El razonamiento práctico, en el cual se basa este modelo, involucra dos procesos importantes: decidir qué metas se desean conseguir, proceso que se conoce como deliberación; y cómo se lograrán estas metas, procedimiento que se denomina razonamiento de fines y medios. El proceso de decisión empieza típicamente tratando de comprender que opciones están disponibles; una vez generado este conjunto de alternativas, se debe elegir entre ellas y comprometerse con una; esta opción escogida se convierte en una intención, la cual determina las acciones del agente. Una vez adoptada una intención, el agente debe perseverarla, y solo debe sustituirla cuando la razón por la cual tenía la intención ha cambiado; o cuando el agente sabe con certeza que no podrá cumplir con ella. Finalmente, las intenciones están estrechamente relacionadas con las creencias acerca del futuro. Cuando tiene una intención, el agente debe creer que tiene una gran posibilidad de cumplirla.

Aunque la capacidad de **aprendizaje** puede ser útil para algunas aplicaciones, para otras puede no ser conveniente. De cualquier manera, el aprendizaje no es una característica esencial de los agentes y por eso no es mencionada.

3.2 CONCEPTOS CENTRALES PARA LA CONSTRUCCION DE AGENTES

Con el propósito de precisar los términos que aparecen al tratar de definir el comportamiento de un agente, en las siguientes secciones se examinan algunos de estos, tomando como referencia aquellos usados en la metodología Prometheus [41]. Esta discusión es útil en el momento de diseñar los agentes que constituyen el sistema, en nuestro caso el planificador de agarre.

3.2.1 Agentes situados en un ambiente: acciones y percepciones

Una **acción** es algo que un agente hace, tal como mover o expulsar. Una acción es básicamente la habilidad del agente para cambiar el ambiente en el cual está situado. Las acciones pueden ser *instantáneas o duraderas* y pueden fallar o ser exitosas. Las acciones duraderas son las que toman cierta cantidad de tiempo para realizar un cambio significativo en el ambiente. Por otro lado las acciones instantáneas realizan un cambio en el ambiente de manera rápida. En muchas de las ocasiones, el inicio de una acción viene como consecuencia de una **percepción**, que normalmente es un ítem de información recibida del ambiente, por algún sensor, por ejemplo.

3.2.2 Eventos y objetivos: agentes proactivos y reactivos

Un **evento** es un hecho significativo al que el agente debe responder de alguna forma. Los eventos son normalmente extraídos de las percepciones, aunque también pueden ser generados internamente por el propio agente, siendo estos vistos como una clase particular de eventos (propiocepción).

¿Qué puede ser considerado un **objetivo** para un agente? No todo lo que se piensa que es un objetivo en realidad lo es. Por ejemplo mover un carro podría ser considerado un objetivo, pero la verdad es que pudiera ser una simple acción que ayuda a alcanzar una meta. Se puede considerar un objetivo como cierto estado del mundo que el agente debe obtener. Típicamente el agente detecta un evento y ejecuta diversas acciones que procuran modificar parcial o totalmente el ambiente.

Los eventos son importantes en la creación de agentes reactivos ya que el agente reacciona inmediatamente ante ellos. Los objetivos, por el contrario dan al agente pro-actividad. En pocas palabras, los agentes proactivos usan los objetivos y los agentes reactivos responden a los eventos.

3.2.3 Ambientes difíciles: planes y creencias

Los agentes en aplicaciones reales, usualmente, tienen recursos computacionales limitados así como restricciones para sensar su ambiente. Una **creencia** es una información que el agente supone acerca del ambiente, de si mismo o de otros agentes.

Un **plan** es la conducta necesaria para lograr un objetivo. La noción de plan es bastante general y abarca una amplia gama de plataformas, incluyendo los agentes basados en planes en el modelo **BDI** (Belief-Desire-Intention). Un plan para conseguir un objetivo provee: (a) una función que indica si vale la pena probar el plan en la situación actual, es decir, si el plan es aplicable, y (b) un cuerpo del plan que es ejecutado para conseguir un objetivo. La estructuración del cuerpo del plan se da en términos de los pasos necesarios para conseguir sub-objetivos. En tiempo de ejecución, el agente seleccionará un plan para conseguir un sub-objetivo dado.

En resumen, en la construcción de agentes inteligentes son importantes los siguientes conceptos centrales:

Creencias: información acerca del ambiente.
Deseos/Objetivos: Objetivos a cumplir.
Intenciones: el curso de acción elegido.
Planes: maneras de conseguir ciertos estados futuros del mundo o maneras de conseguir objetivos.
Acciones: maneras en las que el agente puede operar en el ambiente.
Percepciones: información relevante del ambiente y del propio agente.
Eventos: información relevante acerca de un cambio en la situación (las percepciones son un subconjunto de los eventos).
Mensajes: Información necesaria para que los agentes interactúen.
Protocolos: especificaciones de reglas de interacción. Usualmente asociadas con la consecución de objetivos.

3.3 SISTEMAS MULTI-AGENTE (MAS)

La solución de un problema utilizando un solo agente causa grandes restricciones. Si el agente es capaz de resolver un problema, con seguridad se presentaran cuellos de botella en lo que se refiere a velocidad, fiabilidad, flexibilidad y modularidad. En el peor de los casos, el problema puede ser tan complejo que un agente no puede encontrar una solución útil. Estas razones motivaron la aparición de los sistemas multi-agente (**MAS**: multi-agent systems). Al Igual que para el caso de los agentes, no existe una definición que todos acepten. Una definición muy general dada por [46] es:"*Un sistema multi-agente tiene una representación explicita del ambiente sobre el cual los agentes pueden razonar y predecir eventos futuros*". Otras definiciones no son tan generales y dependen del punto de vista de donde lo abordemos.

Específicamente, para los sistemas de manipulación se puede tomar como referencia la definición dada en [47]: "*Un sistema multi-agente puede ser visto como una clase de sistema multi-dedo, una vez los dedos son distribuidos espacialmente alrededor del objeto, necesitando ser coordinados* para *trabajar juntos y así desempeñar tareas de manipulación correctas".*

Los sistemas Multi-agente forman una comunidad social de miembros inter-dependientes que actúan individualmente. Cada agente tiene información o capacidad incompleta para solucionar el problema y no hay un sistema global de control, los datos están descentralizados y la computación es asíncrona [48]. Desde el momento en que los agentes asumen formar una comunidad se hace necesaria la capacidad de negociar y coordinar diferentes tareas [49]. Los agentes deben ser capaces de combinar sus esfuerzos, para solucionar problemas que el propio agente, por si mismo, no es capaz de solucionar, dando lugar a comportamientos de *cooperación* o de *competencia* [48], [50].

3.4 MÉTODOS DE COMUNICACIÓN EN SISTEMAS MULTI-AGENTE

En un sistema multi-agente existen situaciones de *cooperación y* de *competencia.* En el primer caso, los agentes deben ser capaces de combinar sus esfuerzos, para solucionar problemas que el propio agente, por si mismo no es capaz de encontrar. Por otro lado, cuando los agentes tienen objetivos distintos y tienen que competir entre ellos tendrán que utilizar estrategias de negociación para evitar conflictos. En ambos casos, los agentes necesitan comunicarse entre si para obtener las ventajas que ofrece la cooperación y librarse de conflictos que genera la competencia.

La comunicación forma la base de la coordinación y está formada por los protocolos de comunicación y los métodos de comunicación [48]. Los protocolos de comunicación son un conjunto de reglas que deben cumplir los agentes para comunicarse entre sí. Dentro de los métodos de comunicación se pueden encontrar los *sistemas de pizarra y sistemas de mensaje/dialogo* [48].

Sistemas de pizarra: La arquitectura de pizarra es ampliamente usada en Inteligencia Artificial. La pizarra es una estructura de datos que es usada como mecanismo general de comunicación entre las múltiples fuentes de conocimiento y es gestionada y arbitrada por un controlador. De esta manera los agentes pueden intercambiar, datos y conocimiento a través de un área de trabajo común. Al igual que nuestras aulas, las pizarras son dispositivos para compartir información, con múltiples escritores y múltiples lectores. En la Figura 3.1 se ilustra este tipo de arquitectura. En esta figura se muestra la organización de diferentes agentes alrededor de una pizarra, en la cual se lee y escribe información que será compartida por todos los agentes.

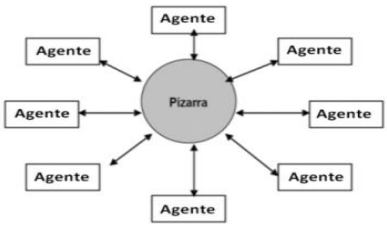


Figura 3.1. Estructura de un sistema de pizarra [48].

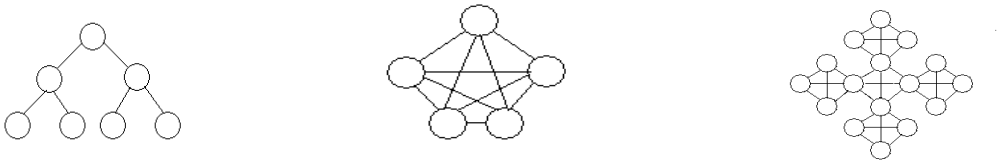
Un ejemplo práctico del uso de este tipo de arquitectura, es el sistema de control distribuido basado en multi-agentes para un robot inteligente desarrollado en [64]. El sistema consiste de la integración de muchos módulos funcionales (agentes), que intercambian información por medio de un sistema de pizarra distribuido.

Sistema de mensajes/dialogo: Por medio de protocolos los agentes intercambian mensajes. Estos mensajes, además de ser utilizados para establecer comunicaciones y mecanismos de cooperación entre agentes forman una base de comunicación flexible que ayuda a la implementación de estrategias de coordinación complejas. Un agente emisor transfiere un mensaje específico a otro agente, el receptor. Al contrario que en los sistemas de pizarra, los mensajes son intercambiados directamente entre dos agentes. Algunas arquitecturas de sistemas multi-agente que se forman utilizando el sistema de comunicación mensaje/dialogo son:

- **Árbol fijo:** Son sistemas que usan arquitecturas de control jerárquicas. Los agentes solo se comunican con el agente que está más arriba o más abajo de ellos en la jerarquía. ver figura 3.2 a).

- **Red de Agentes:** Cada agente establece conexión con cada uno de los otros agentes, ver figura 3.2 b).

- **Colección de Grupos de agentes:** Los agentes que componen un grupo interactúan con otros grupos vía un agente escogido como representante del grupo, y a su vez, todos los agentes representantes se comunican entre sí, ver figura 3.2 c).



a) Árbol fijo b) Red de agentes c) Colección de grupos de agentes

Figura 3.2. Arquitecturas de comunicación en sistemas multi-agente. [51]

3.5 PROMETHEUS: UNA METODOLOGIA PARA EL DESARROLLO DE SISTEMAS MULTI-AGENTE

El desarrollo de sistemas complejos distribuidos basados en tecnologías multi-agente requiere el uso de alguna metodología durante las fases de análisis y diseño. La principal ventaja reside en que el conocimiento adquirido al desarrollar aplicaciones se puede reutilizar o adaptar a nuevos proyectos. Las primeras metodologías que se desarrollaron tuvieron una concepción muy cercana al paradigma orientado a objetos y en algunas de ellas se proponía el uso del Lenguaje de Modelado Unificado (*Unified Modeling Language*, *UML)*. Estas propuestas reportaron limitaciones ya que los agentes no son especificaciones de objetos, son mucho más que eso, lo que significaba buscar otras alternativas que contemplaran la mayor parte de las características de este nuevo paradigma. En los últimos años han aparecido diversas metodologías propias para el desarrollo de Sistemas Multi-agente, unas a partir de modificaciones de propuestas existentes y otras que presentan concepciones completamente nuevas [48].

En este trabajo se ha optado por escoger PROMETHEUS como metodología idónea para diseñar el sistema de planificación basado en multi-agentes. La elección de esta metodología se debe a las ventajas (descritas al final de la sección) que ofrece, en comparación con otras metodologías. Una descripción general de estas metodologías se encuentra en el anexo A. Prometheus define un proceso detallado para la especificación, diseño, implementación y prueba de sistemas software basados en agentes. Además del proceso detallado se define un rango de artefactos que son producidos a lo largo del diseño. Estos artefactos, que pueden ser gráficos o textos estructurados, son producidos para capturar información acerca del sistema y su diseño, tal como lo hacen los diagramas UML, por ejemplo. El proceso de desarrollo comienza de lo más general a lo más especifico, pero a medida que avanza el proceso del diseño puede cambiar a de lo más especifico a lo más general y luego retomar la metodología inicial. La metodología es iterativa, si es necesario devolver atrás para corregir o actualizar ciertas ideas relacionadas con el desarrollo en general.

La metodología Prometheus propone tres fases: especificación del sistema, diseño de la arquitectura y diseño detallado.

1. La *fase de especificación del sistema* se enfoca en identificar los objetivos y las funcionalidades básicas del sistema, junto con las entradas (percepciones) y salidas (acciones).

2. La *fase de diseño de la arquitectura* se basa en la fase anterior y determina los tipos de agentes del sistema y sus interacciones.

3. La *fase de diseño detallado* se encarga del interior de cada agente y de cómo éste conseguirá sus tareas en el sistema completo.

La fase de implementación, común en otras metodologías, es omitida porque depende de la plataforma escogida, y Prometheus no llega hasta estos detalles.

En la figura 3.3 se muestra un gráfico que describe cada una de las fases del sistema. Para cada fase se muestran los componentes que se deben desarrollar para obtener los artefactos finales que ayudarán a especificar componentes de otras fases. Así por ejemplo, los escenarios ayudan a la creación de los diagramas de interacción y protocolos. Estos a su vez ayudan a la creación de los procesos.

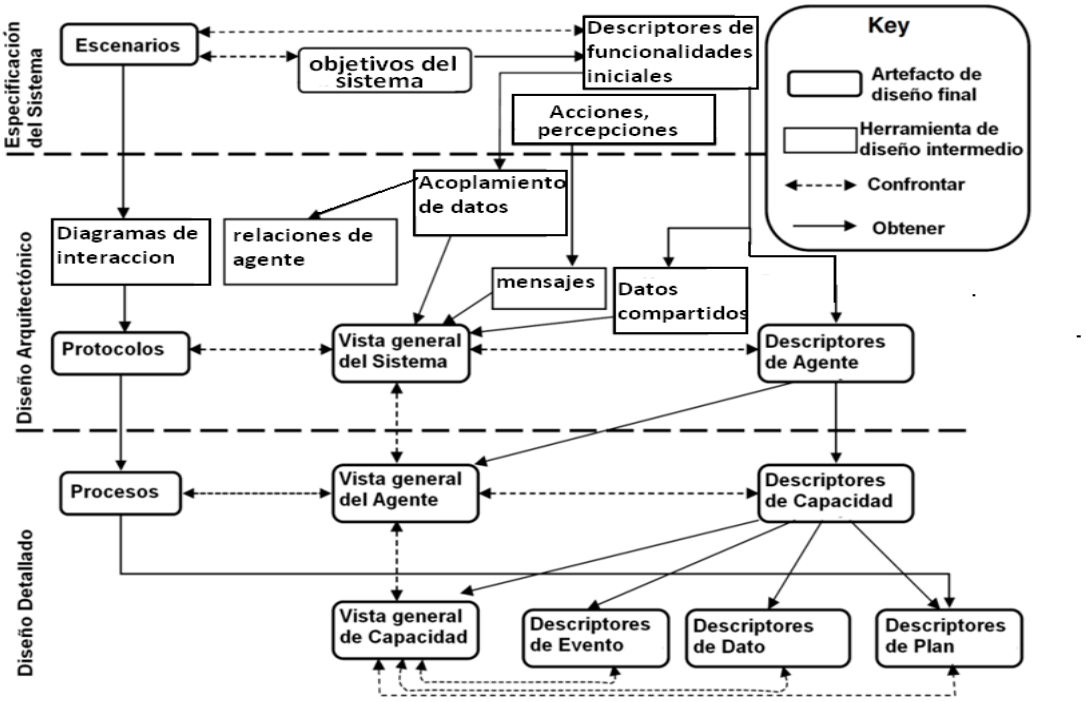


Figura 3.3. Las fases de la metodología Prometheus.
Imagen traducida según [41].

En aras de hacer más claras las etapas de diseño de la arquitectura propuesta para la planificación del agarre, en las siguientes sub-secciones se hace una breve descripción de las fases de desarrollo de la metodología Prometheus.

3.5.1 Fase 1: especificación del sistema

La fase de especificación del sistema se enfoca en los siguientes aspectos:

- Identificar de los objetivos del sistema**.**

- Desarrollar los escenarios de caso de uso para mostrar la operación del sistema.
- Identificar las funcionalidades básicas del sistema.
- Especificar la interfaz entre el sistema y su ambiente en términos de percepciones y acciones.

Los **objetivos** del sistema capturan, en un alto nivel, lo que el sistema tiene que ser capaz de hacer. Dado que los objetivos son descripciones de alto nivel, tienden a tener poca probabilidad de cambiar en el tiempo. La identificación de objetivos da como resultado una colección de objetivos, cada uno con un nombre y una descripción, así como de un conjunto de sub-objetivos.

Los **escenarios** casos de uso consisten de una secuencia de pasos ordenados que pueden ocurrir durante la operación del sistema, junto con una descripción del contexto en el cual esta secuencia de pasos ocurre. Son útiles en el sentido de que son más específicos y concretos que los objetivos y de que, además, tienden a ser más accesibles y fáciles de visualizar. Sin embargo, los casos de uso no capturan toda la operación de un sistema.

Funcionalidad es un término que se usa para describir un comportamiento del sistema en particular. En una funcionalidad se incluye un grupo de objetivos relacionados, así como percepciones, las acciones y los datos relevantes. Una funcionalidad debe ser coherente en el sentido de poder ser descrita en una o dos frases y asignarle un nombre adecuado de modo que capte toda su esencia. En comparación con los objetivos, las funcionalidades tienden a cambiar en el tiempo.

3.5.2 Fase 2: diseño de la arquitectura

La fase de diseño arquitectónico se enfoca en los siguientes aspectos:

- Decidir qué tipos de agentes serán implementados y desarrollar los descriptores de los agentes.
- Captar la estructura total del sistema usando el diagrama de visión general del sistema.
- Describir el comportamiento dinámico del sistema usando los diagramas y protocolos de interacción.

La decisión de los **tipos de agentes** se realiza mediante el agrupamiento de las funcionalidades. Cada agente consta de una o más funcionalidades. Existen múltiples maneras de realizar el agrupamiento de las funcionalidades. Por ejemplo, el agrupamiento puede estar guiado por consideraciones de acoplamiento y cohesión. Por *acoplamiento* se entiende el grado de dependencia entre una funcionalidad y otra. Un acoplamiento alto produce sistemas difíciles de modificar, ya que el cambio en una funcionalidad posiblemente requiere el cambo de otras funcionalidades. La *cohesión* es una propiedad que afecta a un solo agente. Un agente es cohesivo si todas sus funcionalidades están estrechamente relacionadas.

Una vez se ha hecho la agrupación de los agentes, cada grupo o tipo de agente se describe usando un **descriptor de agente**. Los descriptores de agente reúnen toda la información detallada del agente, como su nombre, las funcionalidades que contienen, las acciones que realiza etc. Estos descriptores se realizan para que sea más fácil el ajuste a diferentes plataformas de implementación. En cuanto se haya decidido qué agentes hacen parte del sistema, se identifican las percepciones a las cuales los agentes reaccionan y las acciones que el agente desempeña en el ambiente externo. Además, se especifican los mensajes entre agentes y se determinan los depósitos de **datos compartidos** (que la mayoría de veces son bases de datos utilizadas para transferir información entre los agentes).

El resultado de la fase de diseño arquitectónico es capturado en el **diagrama de visión general del sistema**, en donde se muestran: la relación entre agentes, las percepciones, las acciones, los depósitos de datos compartidos, y los caminos directos de comunicación entre agentes. Siempre que la actividad de un agente dependa de las actividades otro agente, debe existir explícita o implícitamente una comunicación entre ellos. En el diagrama de visión general del sistema no se muestra la secuencia de transmisión de los mensajes. Los **diagramas de interacción del agente** son más detallados y muestran explícitamente las posibles secuencias de mensajes entre los agentes, de conformidad con los escenarios de uso. Finalmente, los **protocolos de interacción**, detallan todas las posibles interacciones entre agentes y son descritos usando la herramienta Agent UML. Para mayor información consultar [52].

3.5.3 Fase 3: diseño detallado

El diseño detallado se enfoca en los siguientes aspectos:

- Construir el diagrama de visión general del agente en términos de sus capacidades.
- Desarrollar las especificaciones de los procesos.
- Diseñar los planes dentro de una capacidad y los eventos generados y manejados por estos planes. Todo lo anterior debe ser capturado en los diagramas de visión general de la capacidad.
- Especificación del algoritmo dentro de cada plan, así como también de los datos asociados y la especificación detallada de los eventos. Esto es capturado en los descriptores de planes, datos y eventos.

La **capacidad** de un agente es la agrupación de funcionalidades al interior del agente. Esta agrupación se basa en la estrecha relación que pueda existir entre algunas funcionalidades.

En relación con los protocolos de interacción**,** la **especificación de procesos** muestra de manera aún mas detallada la secuencia en la cual se transmiten los mensajes entre agentes.

3.5.4 Prometheus en comparación con otras metodologías

Las fases de desarrollo de PROMETHEUS crean diferencia con otras metodologías. Muchas de las metodologías orientadas a agentes aun no son usadas por los desarrolladores industriales: ellas están todavía bajo investigación, y tampoco se enfocan en aspectos específicos del diseño de agentes tal como el trabajo en equipo (es decir, no suministran una metodología completa) o no son descritas con suficiente detalle. Por ejemplo, muchas de las metodologías están descritas solamente en pocas páginas (8 a 15 Págs.). Adicionalmente, muchas de estas no suministran una herramienta de soporte.

La metodología **Gaia,** al igual que Prometheus, ha sido desarrollada desde hace algunos años por personas experimentadas en la construcción de sistemas de agentes. Sin embargo, se ha encontrado que le hace falta un proceso de diseño detallado – intencionalmente ausente debido a un deseo de generalidad- por lo que no suministra una herramienta de soporte a las necesidades que surgen en este trabajo de grado. En Prometheus y Gaia La especificación del sistema y el diseño arquitectural son similares. Los diagramas de relación entre agentes que se propone en Prometheus son esencialmente los mismos a los usados en Gaia, y los roles en Gaia son similares en concepto a las funcionalidades en Prometheus, aunque existen ligeras diferencias.

En la metodología **Tropos** los requerimientos para diseño detallado son cubiertos anticipadamente. Su diseño detallado es muy orientado hacia JACK como una plataforma de implementación. Comparada con Prometheus, Tropos suministra un fase anticipada de requerimientos, lo cual Prometheus no hace (pero es posible adaptar dichos requerimientos en Prometheus). Prometheus suministra un proceso de diseño más detallado - particularmente en la fase de diseño arquitectural, además, suministra una herramienta de soporte y verificación; actualmente la herramienta de soporte para Tropos está en la forma de un editor de diagramas, a diferencia de la herramienta de diseño de Prometheus en la cual se pueden verificar consistencias y se pueden generar automáticamente algunas partes del diseño.

La metodología **MaSE** en una de las pocas metodologías que parece tener una buena herramienta de soporte. Sin embargo, MaSE es inadecuado para algunas aplicaciones ya que los agentes son vistos, simplemente, como una abstracción conveniente los cuales pueden tener o no tener inteligencia. Por lo tanto, MaSE (intencionalmente) no soporta la construcción de agentes basados en planes que son capaces de suministrar una mezcla flexible de comportamiento reactivo y pro-activo. MaSE es muy general y trata a los agentes como “simples agentes software que interactúan unos con otros para satisfacer un objetivo general del sistema”.

PASSI es una adición reciente a la lista de metodologías. Aunque no ha sido descrita en detalle, esta parece ser completa y suministra una herramienta de soporte.

4. FASE DE ESPECIFICACIÓN DEL SISTEMA

En el capitulo anterior, se definen los conceptos centrales para la construcción de agentes inteligentes y también se elige a Prometheus como la metodología más idónea para desarrollar el sistema de planificación del agarre. En el libro *Developing Intelligent Agent Systems* expuesto en [41], *se* presenta una guía práctica para el desarrollo de sistemas multi-agente bajo la metodología Prometheus. En los siguientes capítulos se siguen los pasos propuestos por esta guía para cumplir con cada una de las tres fases de desarrollo de esta metodología y así diseñar la arquitectura basada en agentes para la planificación del agarre estable. Particularmente, en este capítulo se desarrollan cada uno de los pasos que propone la metodología Prometheus en su primera fase de especificación del sistema. En esta monografía solo se expondrán los artefactos de diseño más importantes. Si embargo, en el anexo B se desarrolla en detalle todo el diseño de la arquitectura de planificación bajo la metodología Prometheus.

4.1 ESPECIFICACIÓN DE OBJETIVOS

El enfoque para la planificación del agarre abordado en el presente trabajo utiliza sensores de pre-contacto en su diseño y está basado en multi-agentes. Para identificar los objetivos de nuestro sistema de planificación, se hace una descripción breve del sistema y se destaca en negrita las palabras o frases importantes que serán usadas para identificar los principales objetivos a lograr:

El principal objetivo de la mano es el **agarre estable** de objetos. Para ayudar a que la prótesis robótica cumpla con este objetivo se propone que, a partir de la intención de movimiento que tiene el usuario se **convierta el valor que identifica el tipo de agarre en una postura de pre-agarre**. Después de alcanzar la postura de pre-agarre los dedos deben realizar los movimientos que conducen al agarre del objeto. Estos movimientos se realizan de tal manera que se **evite el desplazamiento del objeto por el contacto prematuro de algún dedo** y que **el tipo de agarre se adapte a la geometría del objeto**.

De esta descripción se extrae el siguiente conjunto de objetivos:

- **convertir el valor que identifica el tipo de agarre en una postura de pre-agarre.**
- **evitar el contacto prematuro de algún dedo con el objeto.**
- **adaptar el tipo de agarre a la geometría del objeto.**

En este punto nos debemos peguntar: ¿cómo es que se ayuda a una prótesis de mano robótica lograr un agarre estable aplicando los tres objetivos anteriores? La respuesta a esta pregunta es detallada en la siguiente sección.

4.1.1 Determinación del agarre estable

Debido al alcance del proyecto, solo se diseñó la arquitectura basada en multi-agentes para la planificación del agarre estable. Sin embargo, para efectos de validación, nosotros nos basamos en estudios científicos que sustentan los objetivos descritos anteriormente y hacemos uso del simulador *GraspIt!* para comprobar su validez en una mano robótica que tiene las mismas características que se proponen en la sección 2.2.3. En esta sección solo se detallan los estudios científicos; Las pruebas en *GraspIt!* se detallan en el capítulo 7 y en el anexo D.

A. Formar una postura de pre-agarre a partir del tipo de agarre.

La planificación de agarre automático para manos robóticas, es un problema complejo debido a la gran cantidad de posibles configuraciones (posturas) que puede tener una mano. Sin embargo, los humanos simplifican el problema escogiendo una postura apropiada para el objeto y la tarea que se desea desempeñar. [1]. Esta postura hace que la ejecución del agarre sea una labor eficiente aumentando así las probabilidades de realizar un agarre estable del objeto. En [1] se implementa un planificador que usa un subsistema de visión para obtener una versión simplificada del objeto, que consiste de un conjunto de formas primitivas: esferas, cilindros, conos y cajas. De acuerdo a la versión simplificada del objeto, el planificador genera un conjunto de posiciones iniciales de agarre. Cada posición inicial de agarre consiste de una postura de pre-agarre y de una posición y orientación de la palma de la mano. A partir de la postura de pre-agarre se ejecutan los diferentes movimientos que conllevan al agarre del objeto. La puesta en práctica de este planificador muestra como el formado de una postura de pre-agarre adecuada aumenta significativamente las probabilidades de un agarre estable.

En este trabajo se trata de dar una posible solución al problema de la planificación del agarre estable de un objeto. Para hacer de este sistema de planificación, un sistema verdaderamente eficiente no solo se debe considerar aspectos propios de la mano, sino que, también, se deben tener en cuenta los criterios de planificación aportados por el paciente. Estos criterios son identificados a través de las señales mioeléctricas que genera el paciente de las cuales se puede identificar el tipo de agarre que desea realizar el paciente sobre el objeto. En este trabajo se tiene en cuenta la interpretación que se le da a este tipo de señales para generar una correspondiente postura de pre-agarre

B. Evitar el contacto prematuro de algún dedo con el objeto

Es más fácil realizar el agarre de un objeto que está quieto y no de un objeto que se mueve sin saber en qué dirección. El movimiento del objeto hace que el coeficiente de rozamiento entre los dedos y el objeto sea vencido con menos fuerza, haciendo que el objeto se deslice más fácilmente. Este movimiento del objeto casi siempre es ocasionado por el contacto prematuro de algún dedo en la fase de pre-agarre. Por lo tanto, es de vital importancia que los dedos

contacten el objeto al mismo tiempo no solo para evitar un desplazamiento del mismo, sino también porque el contacto no simultáneo de los dedos produce una fuerza neta distinta de cero que posiblemente ocasione la inestabilidad del agarre [34]. Por el contrario, el contacto simultáneo le permite a la mano aplicar una fuerza neta igual a cero a lo largo de toda la dinámica transiente del agarre (fase de pre-agarre), y no solo justamente al final del estado de equilibrio [34]. Esto es válido para los robots quienes no tienen la suficiente inteligencia, habilidad y delicadeza para acercarse a un objeto y adaptarse a él, tal y como lo hacen los humanos. Por lo anterior, es que proponemos darle un nuevo sentido a la mano robótica, mediante los sensores de proximidad, los cuales permiten sensar la distancia que tiene cada dedo al objeto y por medio de esta distancia coordinarse para lograr contactar el objeto al mismo tiempo, es decir, producir un agarre en el cual el objeto conserva la misma posición en toda la fase de pre-agarre.

C. Adaptar el tipo de agarre a la geometría del objeto

Una buena adaptación de la postura de agarre a la forma y orientación del objeto es un criterio fundamental para lograr un agarre estable. [63]. Joshua Smith [53], Uno de los creadores de la tecnología de pre-contacto, menciona que el objetivo de esta tecnología es dar a los robots la habilidad de mejorar el agarre de objetos de diferente forma en ambientes humanos no estructurados. Nuestro sistema de planificación logra adaptar el tipo de agarre a la geometría del objeto gracias a dos criterios:

El primer criterio corresponde a un criterio de estabilidad que trata de conservar las normales de los contactos de forma co-planar, en orden de minimizar la cantidad de fricción necesaria para que las fuerzas de contacto estén en equilibrio y evitar el deslizamiento de los dedos por la superficie del objeto. Este criterio es ampliamente usado por diferentes autores que enfocan sus trabajos en la planificación del agarre. En el enfoque de planificación propuesto por [38] las normales de un conjunto de posturas guardadas se hacen corresponder con las normales de las superficies locales de un objeto. En los trabajos de; [11], [33] y [34] se proponen estrategias de agarre, que se basan principalmente, en la información parcial del objeto obtenida de los sensores pre-touch. Esta información parcial se usa para que la superficie de la punta de los dedos de una mano robótica contacte de forma paralela la superficie local del objeto. El resultado de la aplicación de esta estrategia, es un agarre con una mayor probabilidad de estabilidad en comparación con la obtenida en agarres para los cuales no se consideró una forma parcial del objeto (superficie local) en la fase de pre-agarre. En el anexo C.7 se realizan diferentes estudios para probar la veracidad de este criterio.

El segundo criterio, considera el movimiento de oposición del dedo pulgar.

La importancia del dedo pulgar no necesita ser sub-rayada, es el generador de la pinza pulgar-dedo. Sin él los movimientos de los dedos constituirían solo prensiones globales en vez de movimientos precisos [39]. El dedo pulgar, posee además un esqueleto especial, que constituye

su columna, de la cual hay que destacar la importancia de la articulación carpometacarpiana (CMC) que confiere al dedo la posibilidad de oponerse a los demás dedos de tal manera que la postura de agarre sea simétrica alrededor del objeto [39]. Esta capacidad de oposición se realiza en todo el movimiento del agarre y se lleva a cabo de diferentes maneras dependiendo del tamaño y forma del objeto. Esto con el fin de obtener un agarre más estable del mismo.

Debido a las ventajas que ofrece el posicionamiento co-planar de las normales de contacto y el movimiento de oposición del dedo pulgar, nuestro sistema de planificación trata de posicionar la superficie de la punta de cada dedo lo mas paralela posible a la superficie local del objeto y, además, de reconfigurar la articulación que realiza el movimiento de oposición del dedo pulgar en toda la fase de pre-agarre.

4.1.2 Identificación de los sub-objetivos

Después de identificar los objetivos, la siguiente tarea es obtener los sub-objetivos. Una técnica simple es responder a la pregunta "¿Cómo se puede conseguir el objetivo? Esta técnica se aplica para cada uno de los objetivos encontrados.

- ¿Cómo **convertir el valor del tipo de agarre a una postura de pre-agarre?**

Cada dedo tiene su propia base de datos, en la cual se guardan sus posiciones articulares, que conforman la postura de pre-agarre para cada tipo de agarre. Consecuentemente, cada dedo moverá sus articulaciones de forma incremental hasta alcanzar dicha posición. Por lo tanto, es preciso alcanzar los siguientes dos sub-objetivos:

- Obtener postura de pre-agarre de la base de datos.
- Alcanzar la postura de pre-agarre.

Para lograr el primer sub-objetivo, ¿Qué datos obtengo de la base de datos de posturas de pre-agarre?

- la posición de pre-agarre de las articulaciones que realizan los movimientos de flexión-extensión.
- la posición de pre-agarre de la articulación que realiza los movimientos de aducción-abducción.
- la posición de pre-agarre de la articulación que realiza el movimiento de oposición del pulgar.

Del segundo sub-objetivo surge la pregunta: ¿Cómo alcanzo la postura de pre-agarre?

Realizando movimientos de flexión-extensión y/o aducción-abducción hasta alcanzar la postura de pre-agarre obtenida de la base de datos. En resumen:

- Realizar movimientos de flexión-extensión hasta alcanzar postura de pre-agarre.
- Realizar movimientos de aducción-abducción hasta alcanzar postura de pre-agarre.
- Realizar movimientos de oposición del dedo pulgar.

Las bases de datos son consideradas como depósitos de datos a los cuales los agentes deben acceder. En este trabajo no se considera su diseño.

- ¿Cómo **evitar el desplazamiento del objeto por el contacto prematuro de algún dedo?**

Respuesta: logrando que todos los dedos toquen el objeto al mismo tiempo.

¿Cómo lograr que los dedos toquen el objeto al mismo tiempo?

Flexionando los dedos de tal manera que cada uno de ellos llegue a una distancia umbral[7] en relación con la superficie del objeto. En el resto de este trabajo a este umbral se le llama Set-Point. Luego que los dedos han alcanzado esta distancia se continúa con el proceso de flexión hasta contactar el objeto.

En resumen, los siguientes dos sub-objetivos dan respuesta a las preguntas anteriores:

- Flexionar los dedos hasta el Set-Point.
- Flexionar los dedos desde el Set-point hasta contactar el objeto.

- ¿Cómo **el tipo de agarre se adapta a la geometría del objeto**?

- logrando que la superficie de la punta de los dedos quede lo más paralela posible a la superficie local del objeto.
- Lograr que la postura de agarre sea simétrica alrededor del objeto.
- Flexionar los dedos hasta el Set-Point.

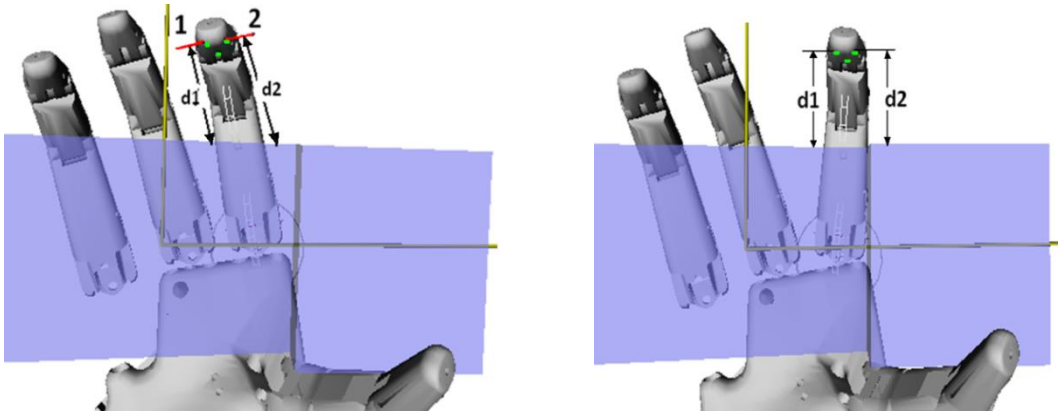
Cabe resaltar que los sub-objetivos que hemos encontrado no son secuenciales. Más adelante se consideraran estos aspectos.

Pero, ¿Cómo lograr que la superficie de la punta de los dedos quede lo más paralela posible a la superficie local del objeto?

Al igual que en [11], en este trabajo los sensores de la punta de los dedos son ubicados de tal manera que formen un triangulo (ver figura 4.1 a)). De esta manera se obtiene una orientación 3D de la superficie local del objeto. En la figura 4.1 se muestra una mano robótica que tiene las mismas características a las propuestas en la sección 2.2.3. Esta mano intenta agarrar un

[7] Por defecto esta distancia es igual a 1 cm, pero puede ser modificada si alguna de las puntas de los dedos antes de realizar la fase de pre-agarre a sensado una distancia menor. Más adelante se describe este proceso con más detalle.

teléfono de tal manera que la superficie de la punta del dedo índice quede lo mas paralela posible a la superficie local del objeto. En la figura 4.1 a) se muestra un ejemplo representativo de la posible configuración de los sensores ópticos en la punta del dedo índice (la gráfica no está a escala). Para que la superficie de la punta de este dedo logre quedar lo mas paralela posible al superficie local del objeto el dedo deber realizar un movimiento de aducción o abducción. Este movimiento depende de la distancia detectada por los sensores hasta la superficie del objeto. Por ejemplo, en la figura 4.1 a), el sensor 2 del dedo detecta mayor distancia hasta el objeto que el sensor 1, entonces el dedo índice debe realizar un movimiento de abducción hasta que las distancias de estos dos sensores sean aproximadamente iguales o hasta que los limites articulares ya no permitan mas movimiento (ver figura 4.1 b)). El tercer sensor, es usado para generar un región de traslape entre las señales emitidas de los tres sensores y así detectar, de una manera mas precisa, la distancia de la punta del dedo al objeto.



a) Distancia detectada desde el sensor 1 (d1) menor a la distancia detectada desde el sensor 2 (d2).

b) movimiento de abducción del dedo indice que logra que las distancias d1 y d2 sean aproximadamente iguales

.

Figura 4.1. Ejemplo representativo que muestra como la superficie de la punta del dedo índice logra quedar lo mas paralela posible a la superficie local del objeto.

En resumen, la superficie de la punta de los dedos queda lo más paralela posible a la superficie local del objeto:

- Realizando movimientos de aducción-abducción de los dedos en la fase de pre-agarre.

Y ¿como Lograr que la postura de agarre sea simétrica alrededor del objeto?

La importancia del dedo pulgar no necesita ser sub-rayada, es el generador de la pinza pulgar-dedo. Sin él los movimientos de los dedos constituirían solo prensiones globales en vez de movimientos precisos [39]. El dedo pulgar, posee además un esqueleto especial, se trata de la columna del dedo pulgar, de la cual hay que destacar la importancia de la articulación

carpometacarpiana (CMC) que confiere al dedo la posibilidad de oponerse a los demás dedos de tal manera que la postura de agarre sea simétrica alrededor del objeto.

En resumen, la simetría alrededor del objeto se logra:

- Realizando el movimiento de oposición del dedo pulgar en base a los movimientos de aducción-abducción del dedo anular.

4.1.3 Notación abreviada de los objetivos y sub-objetivos

Por razones de comodidad, hemos decidido renombrar los tres objetivos principales con un nombre más corto que los represente: El objetivo **Convertir el valor del tipo de agarre a una postura de pre-agarre,** ahora se llamará "configuración "; el objetivo "**El tipo de agarre se adapta a la geometría del objeto"** ahora se llamará "adaptación" y "**los dedos tocan el objeto al mismo tiempo"** se llamará "contacto".

Cada dedo tiene 3 grados de libertad (GDL): dos para movimientos de flexión-extensión y uno para aducción-abducción, en el caso del índice, medio y anular; y en el caso del pulgar, dos para flexión-extensión y uno para oposición. En total la mano robótica tiene 12 GDL. En este trabajo, los grados de libertad ubicados en cada una de las articulaciones serán resumidos. Al grado de libertad que se ubica en la articulación interfálangica proximal (PIP) se le denominará como GDL(1); al grado de libertad que se ubica en la articulación metacarpofalángica (MCP) y que es el encargado de la flexión-extensión se le denominará como GDL(2); y al grado de libertad que se ubica en esta última articulación pero que es el encargado de la aducción-abducción de los dedos se le denominará como GDL(3). Estas notaciones son iguales para todos los dedos, con la excepción de que si nos estamos refiriendo al dedo pulgar las notaciones GDL(1), GDL(2) y GDL(3) representan los grados de libertad ubicados en las articulaciones interfalángica (IP), metacarpofalángica (MCP) y carpometacarpiana (CMC) respectivamente. Adicionalmente se usa la notación: GDL(1y2) para referirse a los grados de libertad GDL(1) y GDL(2). De modo que, por ejemplo, la notación GDL(1y2)pul se refiere a los grados de libertad encargados de realizar la flexión-extensión del dedo pulgar y GDL(3)pul al grado de libertad que realiza el movimiento de oposición del dedo pulgar.

Otras abreviaturas que se serán usadas en el resto de este trabajo son las siguientes:

pul: pulgar.
ind: índice.
med: medio.
anu: anular.
s-p: set-point.
BD: Base de datos.
pos: postura de la mano robótica.

pre: pre-agarre
pos_pre: postura de pre-agarre
opo: oposición del pulgar con el anular.
||: paralelismo entre la superficie local del objeto y la superficie de la punta del dedo.

Con esta notación, a continuación se abrevia cada uno de los objetivos y sub-objetivos.

Objetivo: **Convertir el valor del tipo de agarre a una postura de pre-agarre**
Nombre: **CONFIGURACIÓN**

- Sub-objetivos para: obtener postura de pre-agarre de BD

  - obtener GDL(1y2)ind: Obtener la posición de pre-agarre de las articulaciones que realizan los movimientos de flexión-extensión del dedo índice.
  - obtener GDL(1y2)med: Idem para el dedo medio
  - obtener GDL(1y2)anu
  - obtener GDL(1y2)pul
  - obtener GDL(3)ind
  - obtener GDL(3)med
  - obtener GDL(3)anu
  - obtener GDL(3)pul

- Sub-objetivos para: alcanzar la postura de pre-agarre

  - mover GDL(1y2)ind a pos_pre: Realizar movimientos de flexión-extensión hasta alcanzar postura de pre-agarre del dedo índice.
  - mover GDL(1y2)med a pos_pre: Idem para el dedo medio.
  - mover GDL(1y2)anu a pos_pre
  - mover GDL(1y2)pul a pos_pre
  - mover GDL(3)ind a pos_pre
  - mover GDL(3)med a pos_pre
  - mover GDL(3)anu a pos_pre
  - mover GDL(3)pul a pos_pre

Objetivo: **Los dedos tocan el objeto al mismo tiempo**.
Nombre: **CONTACTO**

- Sub-objetivos para: flexionar los dedos hasta el set-point.

  - mover GDL(1y2)ind a s-p: Mover las articulaciones de flexión-extensión del índice hasta el set-point.
  - mover GDL(1y2)med a s-p: Idem para el dedo medio
  - moverGDL(1y2)anu a s-p.
  - moverGDL(1y2)pul a s-p.

- Sub-objetivos para: flexionar los dedos desde el set-point hasta contactar el objeto.

    - mover GDL(1y2)ind a contacto: Mover articulaciones de flexión-extensión hasta que el dedo índice contacte el objeto.
    - moverGDL(1y2)med a contacto.
    - mover GDL(1y2)anu a contacto.
    - mover GDL(1y2)pul a contacto.

Objetivo: **El tipo de agarre se adapta a la geometría del objeto**
Nombre: **ADAPTACIÓN.**

- Sub-objetivos para: lograr que la superficie de la punta de los dedos quede lo más paralela posible a la superficie local del objeto

    - mover GDL(3)ind a ||:mover la articulación encargada de la aducción-abducción del dedo índice en la fase de pre-agarre hasta obtener el máximo paralelismo.
    - mover GDL(3)med a ||: Idem para el dedo medio.
    - mover GDL(3)anu a ||.

- Sub-objetivos para: lograr que la postura de agarre sea simétrica alrededor del objeto

    - mover GDL(3)pul a opo: mover la articulación encargada de la oposición del dedo pulgar en base a los movimientos de aducción-abducción del dedo anular.

Tal como propone la metodología Prometheus, se debe hacer un diagrama que visualice la estructura de los objetivos y sub-objetivos formando una red (ver figura 4.2). Cada objetivo o sub-objetivo es representado por un óvalo y la dirección de las flechas conectan a los objetivos con sus respectivos sub-objetivos. Así por ejemplo, el objetivo ***contacto*** está relacionado por medio de flechas con sus correspondientes sub-objetivos ***flexionar hasta s-p; flexionar hasta contacto.*** Estos a su vez están conectados sus correspondientes sub-objetivos que se describen con la misma nomenclatura asignada anteriormente.

4.2 ESPECIFICACIÓN DE LAS FUNCIONALIDADES

Funcionalidad es un término que se usa para describir un comportamiento del sistema en particular. En una funcionalidad se incluye un grupo de objetivos relacionados, así como las percepciones, acciones y los datos relevantes. Una funcionalidad debe ser coherente en el sentido de poder ser descrita en una o dos frases y asignarle un nombre adecuado de modo que capte toda su esencia, además, permiten un diseño mixto: diseño descendente (*top-down*) y diseño ascendente (*bottom-up*). En el diseño descendente se pueden identificar nuevos

objetivos. Por otro lado, el diseño ascendente provee un mecanismo para determinar los tipos de agentes y sus responsabilidades. El comportamiento de la mano depende de las capacidades de movimiento de cada uno de los dedos y puede divirse en tres sub-comportamientos: flexión-extensión, aducción-abducción y oposición (pulgar). Cada uno de estos sub-comportamientos esta presente en cada dedo, y dependen del movimiento coordinado de todos ellos. Con excepcion de la oposición que solo pertenece al pulgar.

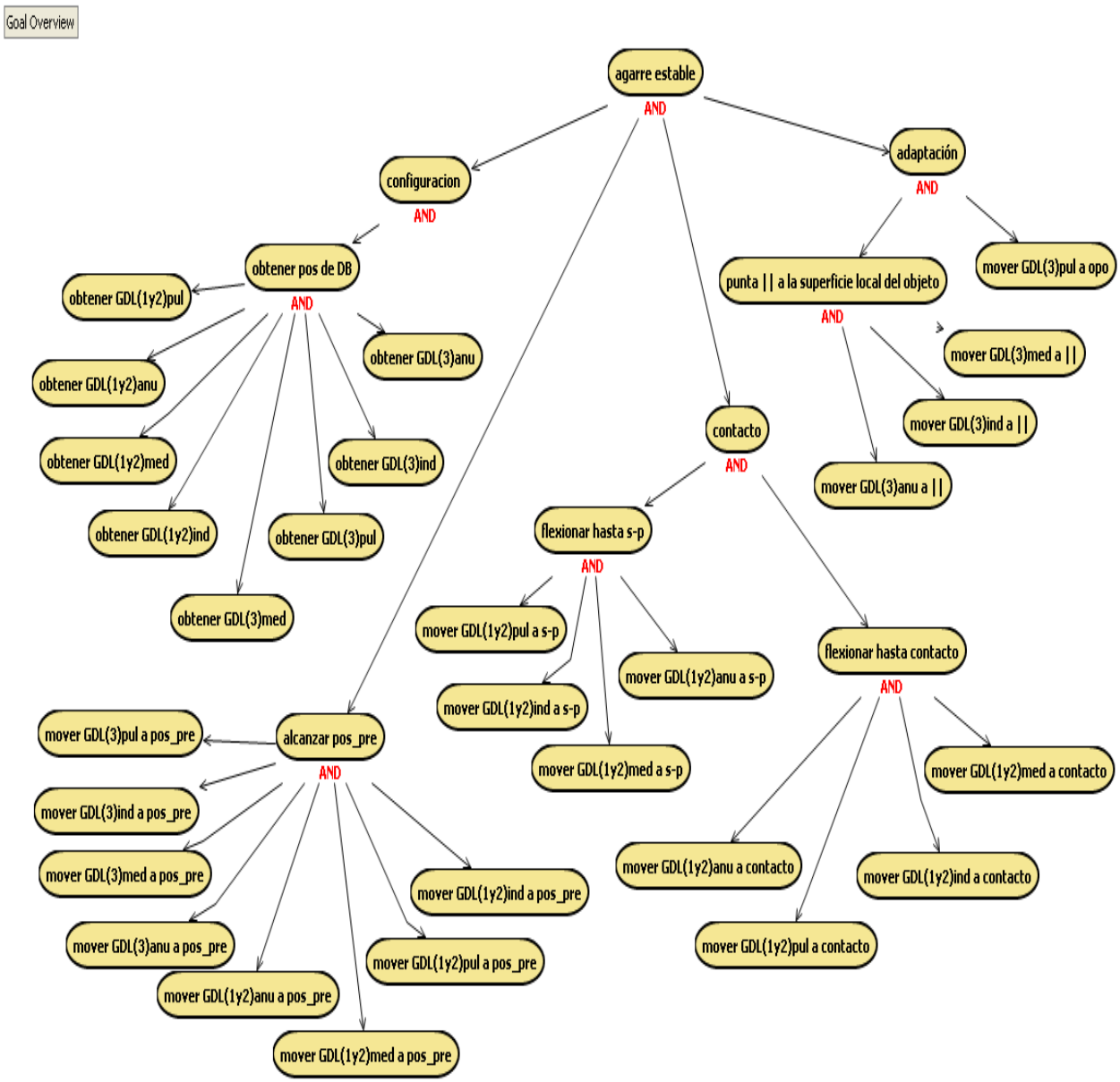


Figura 4.2. Diagrama de Objetivos para el sistema de planificación.

En nuestro sistema de planificación se decide agrupar objetivos que se realcionan entre sí y que describen una parte del comportamiento de la mano y sus acciones, formando una

funcionalidad. Por ejemplo, la funcionalidad llamada flexión-extensión del dedo indice (***flexión-extensión_ind***) incluye todos los objetivos o sub-objetivos que se se relacionen con las articulaciones de flexion-extensión del dedo indice, es decir con GDL(1) y GDL(2). Las funcionalidades que pertenecen a los demás dedos agrupan objetivos que se encargan del movimiento de las articulaciones y su nombre esta relacionado con el movimiento que representan.

Para cumplir con los objetivos es necesario realizar ciertas acciones, por lo tanto para cada funcionalidad se definen acciones que tiene que ver con el movimiento de los grados de liebertad. Por ejemplo: el movimiento de flexión-extensión se realiza por medio de los grados de libertad GDL(1) y GDL(2) de cada dedo, de aquí salen dos acciones que son denotadas como: mover GDL(1) y mover GDL(2).

Según recomienda la metodología, se debe realizar un diagrama que muestre cada una de las funcionalidades con sus respectivos objetivos y acciones. En la figura 4.3 se muestra como cada funcionalidad (rectángulos redondeados de color azul) es conectada a sus respectivos objetivos (óvalos de color amarillo) y acciones (rectángulo extendido de color verde).

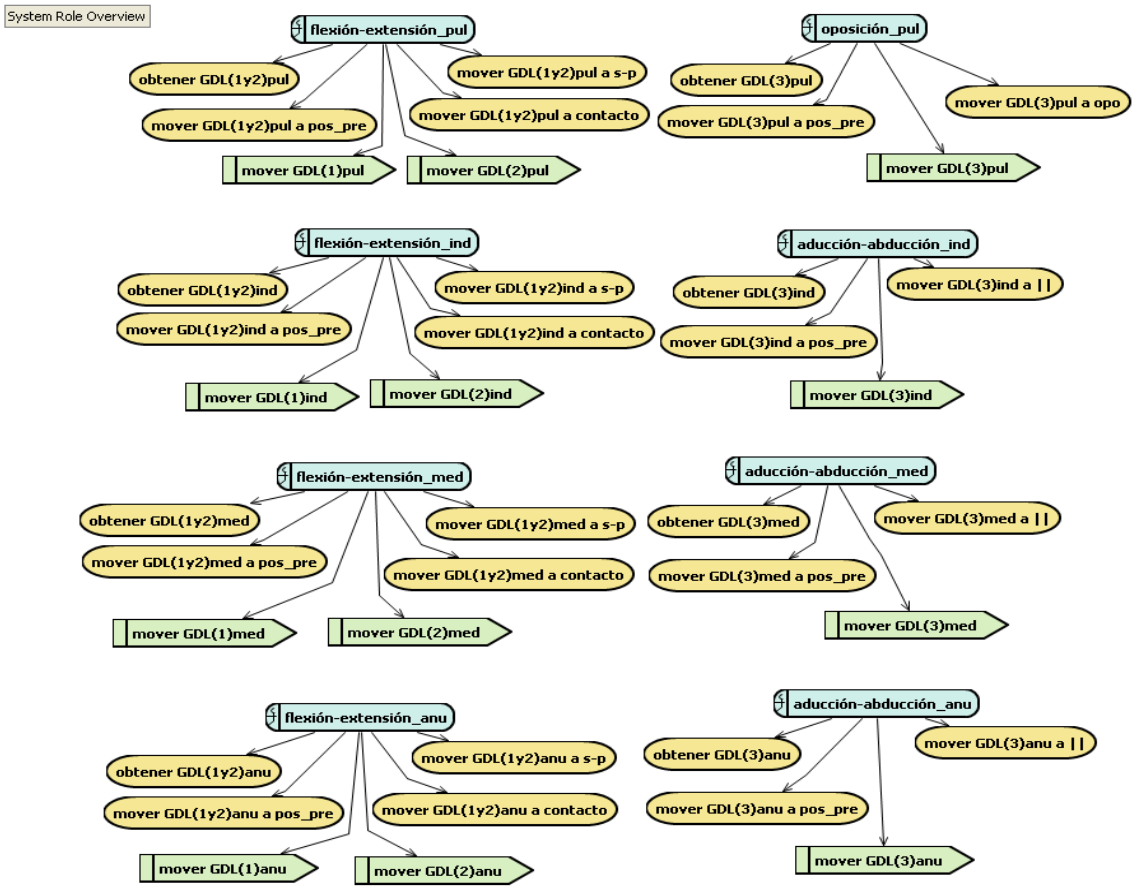


Figura 4.3. Diagrama de funcionalidades para el sistema de planificación del agarre.

Construidas las funcionalidades se desarrollan los *descriptores de funcionalidad*. En estos descriptores cada funcionalidad se describe con pocas palabras y se nombran sus objetivos y acciones. Los descriptores de funcionalidad también incluyen los llamados *accionadores*: los cuales contienen la información acerca de los eventos o situaciones que activarán alguna tarea dentro de la funcionalidad. Por último, en los descriptores se describen también los datos requeridos y producidos por la funcionalidad.

A continuación se muestran los descriptores para tres de las funcionalidades más representativas: **flexión-extensión_ind, aducción-abducción_ind y oposición_pul.** Para una lectura más agradable de este capítulo, los demás descriptores se encuentran en el Anexo B.1.3.

Descriptor de la funcionalidad Flexión-Extensión_ind

Nombre: **Flexión-Extensión_ind**
Descripción: Esta funcionalidad es la encargada de realizar y regular el movimiento de flexión-extensión del dedo índice. Con dos propósitos, alcanzar la postura de pre-agarre y que el dedo toque el objeto al mismo tiempo que los demás.
Objetivos: Esta funcionalidad logra los siguientes objetivos: obtener GDL(1y2)ind, mover GDL(1y2)ind a pos_pre, mover GDL(1y2)ind a s-p, mover GDL(1y2)ind a contacto.
Acciones: Mover a la posición deseada las articulaciones encargadas de flexión-extensión del índice (mover GDL(1)ind; mover GDL(2)ind).
Accionadores (trigger): *i*) Llegada del valor del tipo de agarre; *ii*) Distancia del dedo al objeto menor o igual al set-point.
Información usada: posición de las articulaciones de flexión-extensión que forman la postura de pre-agarre del dedo índice.
Información producida: Distancia del dedo índice al objeto**.**
Nota: La información producida por esta funcionalidad es de vital importancia ya que es utilizada por lo demás dedos para tocar el objeto al mismo tiempo. Como se detallará más adelante esta información es tomada por el dedo pulgar el cual, entre otras cosas, es el encargado de actualizar el set-point

Descriptor de la funcionalidad **Aducción-Abducción_ind**

Nombre: Aducción-Abducción_ind
Descripción: Esta funcionalidad es la encargada de realizar y regular el movimiento de aducción-abducción del dedo índice. Con tres propósitos: alcanzar la postura de pre-agarre; lograr que la superficie de su punta sea paralela a superficie local del objeto; y por último, evitar una colisión con el dedo medio.
Objetivos: Esta funcionalidad logra los siguientes objetivos: obtener GDL(3)ind; mover GDL(3)ind a pos_pre ; mover GDL(3)ind a ||
Accion: Mover a posición la articulación encargada de la aducción-abducción del índice (mover GDL(3)ind).

Accionadores (trigger): *i*) llegada del valor del tipo de agarre; *ii*) distancia del dedo al objeto menor o igual a una referencia específica (3.5 cm). Esta distancia corresponde al momento cuando entran a funcionar los sensores de proximidad ópticos. No confundir con el Set-point.
Información usada: *i)* posición de la articulación de aducción-abducción que forma la postura de pre-agarre del dedo índice; *ii*) posición de la articulación de aducción-abducción del dedo medio.
Información producida: posición de la articulación de aducción-abducción del dedo índice.
Nota: La información sobre la posición de la articulación de aducción-abducción es usada por los demás dedos para que no ocurra una colisión entre ellos al momento de cumplir con los objetivos de esta funcionalidad.

Descriptor de la funcionalidad **Oposición_pul**

Nombre: **Oposición_pul**
Descripción: Esta funcionalidad es la encargada de oponer el pulgar al dedo anular.
Objetivos: obtener GDL(3)pul; mover GDL(3)pul a pos_pre ; mover GDL(3)pul a opo.
Accion: Mover a posición la articulación encargada de la oposición del pulgar (mover GDL(3)pul).
Accionadores (trigger): *i*) Llegada del valor del tipo de agarre; *ii*) movimiento de la articulación de aducción-abducción del dedo anular.
Información usada: *i*) posición de la articulación de oposición para formar la postura de pre-agarre; *i*) posición de la articulación de aducción-abducción del dedo anular.
Información producida: no hay.

4.3 ESPECIFICACIÓN DE LOS ESCENARIOS

El núcleo del escenario consiste de una secuencia de pasos, los cuales normalmente son: cumplir un objetivo (OBJETIVO), realizar una acción (ACCION), recibir una percepción (PERCEPCION), o referirse a otro escenario (ESCENARIO). En el desarrollo de los escenarios se pueden identificar nuevos objetivos, que al igual que los antiguos, se les puede detallar el tipo de información usada y producida así como también a que funcionalidad pertenecen. La secuencia de pasos no es estrictamente ordenada y puede ocurrir que un paso o conjunto de pasos se ejecute pero con ciertas restricciones. Estas restricciones forman las alternativas, las cuales describen una secuencia de pasos alternos a la ejecución normal del sistema. De acuerdo con Prometheus se pueden capturar alternativas usando dos técnicas. La primera técnica es que con cada escenario se incluya una descripción de las posibles alternativas. La segunda técnica es construir otros escenarios que tengan en cuenta las alternativas encontradas.

En la planificación del agarre utilizando el enfoque pre-touch se han identificado 4 escenarios básicos de funcionamiento:

1. El escenario de **configuración**: en este escenario se describen los pasos necesarios para que cada dedo alcance su postura de pre-agarre de acuerdo al valor del tipo de agarre.

2. El escenario de **adaptación por cierre**: en este escenario se describen los pasos necesarios para que todos los dedos lleguen al set-point.

3. El escenario de **adaptación por alineación**: en este escenario se describen los pasos necesarios para que cada dedo posicione la superficie de su punta paralela a la superficie local del objeto, y que además, el dedo pulgar posicione su articulación de oposición de acuerdo los movimientos de aducción-abducción del dedo anular.

4. El escenario de **contacto:** este escenario, después de que todos los dedos hayan alcanzado el set-point, describe los pasos que se deben llevar a cabo para que todos los dedos contacten el objeto.

A continuación se muestran tablas que describen de manera detallada la secuencia de pasos en cada uno de los escenarios identificados. Lo primero que se hace es una descripción general del escenario. Luego se puntualiza el *accionador* que es un evento que activa el escenario. Seguidamente se detallan cada uno de los pasos y su tipo. Cabe recordar que los pasos pueden ser una *percepcion, acción u objetivo.* Si el paso es de tipo *objetivo* se especifica que datos produce y que datos usa de la base de datos. Adicionalmente, se especifica a que funcionalidad pertenece cada uno de los pasos.

Es preciso aclarar que los escenarios describen una secuencia de pasos que es igual para todos los dedos. Así que, por ejemplo, el escenario de *Configuración* es igual tanto para el dedo índice como para los demás dedos.

4.3.1 Descripción detallada del escenario de configuración

Este escenario, representado por la tabla 3, muestra la secuencia de pasos que debe seguir cualquier dedo para alcanzar su postura de pre-agarre de acuerdo con al valor del tipo de agarre. Lo primero que debe hacer cada dedo después de llegar el valor del tipo de agarre es percibir su estado, es decir, la posición actual de sus articulaciones (pasos 1-3). Al mismo tiempo debe acceder a su base de datos para extraer la posición de las articulaciones para formar su postura de pre-agarre (4-7). Estas posiciones son enviadas al controlador de posición el cual se encarga de que efectivamente las articulaciones alcancen las posiciones de pre-agarre (pasos 8-10). En la siguiente tabla se detalla el *escenario de Configuración:*

Tabla 3. Escenario de configuración

Nombre: Configuración

Descripción: En este escenario, de acuerdo al valor del tipo de agarre, cada dedo se encarga de alcanzar su postura de pre-agarre.
Accionador (*Trigger*): llegada del valor del tipo de agarre.
Pasos:

#	Tipo de paso	Paso	Funcionalidad	Datos: * Usados + Producidos
1	Percepción	Valor del tipo de agarre	Aducción-Abducción Flexión-Extensión y/o oposición	
2	Percepción	posición GDL(1y2)	Flexión-Extensión	
3	Percepción	posición GDL(3)	Aducción-Abducción	
4	Objetivo	obtener GDL(1y2)	Flexión-Extensión	***pos_pre.**
5	Objetivo	obtener GDL(3)	Aducción-Abducción y/o Oposición	***** pos_pre.**
6	Objetivo	mover GDL(1y2) a pos_pre	Flexión-Extensión	
7	Objetivo	mover GDL(3) a pos_pre	Aducción-Abducción y/o oposición	
8	Acción	mover GDL(1)	Flexión-Extensión	
9	Acción	mover GDL(2)	Flexión-Extensión	
10	Acción	mover GDL(3)	Aducción-Abducción	

Alternativas: no hay.

Notación usada:

pos_pre: posición de las articulaciones que describe la postura de pre-agarre que debe alcanza cada dedo.

4.3.2 Descripción detallada del escenario de adaptación por cierre

La finalidad de este escenario, representado por la tabla 4, es mostrar la secuencia de pasos que se deben realizar para lograr que el dedo alcance el set-point. Este escenario se inicia cuando termina el escenario Configuración, es decir, cuando todos los dedos han alcanzado su postura de pre-agarre (accionador). Después de alcanzar la postura de pre-agarre, cada dedo

percibe su estado actual, es decir, la posición de sus articulaciones y la distancia hasta el objeto (pasos 1-2). Si al alcanzar la postura de pre-agarre la distancia de falanges proximales y de la punta de los dedos al objeto son mayores al set-point por defecto (establecida en 1 cm) se inicia, por parte de cada dedo, un movimiento de flexión intentado alcanzar este valor. En caso contrario, los dedos que perciban una distancia desde su falange proximal o desde su punta menor al Set-point por defecto la envían al dedo pulgar. El pulgar compara las distancias recibidas, selecciona la menor y la envía de regreso a todos los dedos para que actualicen su set-point con este valor (paso 3). A continuación, cada dedo se flexiona hasta alcanzar este nuevo set-point (paso 4). Si al flexionar el dedo la distancia percibida por la falange proximal ya es aproximadamente igual al set-point entonces el movimiento realizado por GDL(2) debe detenerse, pero el movimiento de GDL(1) continua hasta que la punta del dedo alcance el set-point (paso 5). Si la punta del dedo alcanza el set-point antes que la falange proximal, entonces el movimiento de las articulaciones debe detenerse.

No debe extrañar que el dedo pulgar coordine los movimientos de los demás dedos. Se reconoce ampliamente la importancia del pulgar en la manipulación de objetos y su capacidad de oposición en los diferentes tipos de agarre humano. De hecho, podemos prescindir de cualquiera de los otros dedos pero nunca del pulgar. En [39] lo denominan el *agente del agarre*.

Tabla 4. Escenario adaptación por cierre

Nombre: adaptación por cierre

Descripción: En este escenario se describen los pasos necesarios para que todos los dedos lleguen al set-point.
Accionador (*Trigger*): Señal de que se ha alcanzado la Postura de pre-agarre.
Pasos:

#	**Tipo de paso**	**Paso**	**Funcionalidad**	**Datos:** *** Usados** **+ Producidos**
1	Percepción	posición GDL(1y2)	Flexión-Extensión	
2	Percepción	distancia	Flexión-Extensión, Aducción-Abducción y/o Oposición_pul	
3	Objetivo	actualizar s-p	Flexión-Extensión	***s-p, d_ind, d_med, d_anu, d_pul.** **+ s-p**
4	Objetivo	mover GDL(1y2) a s-p	Flexión-Extensión	*** s-p, d_ind, d_med ,d_anu, d_pul** **+ s-p**
5	Acción	mover GDL(1)	Flexión-extensión	

6	Acción	mover GDL(2)	Flexión-extensión	
7	Escenario	contacto		

Alternativas:

- La ejecución de todos los pasos de este escenario están limitados por el segundo paso. Si la distancia desde la falange proximal hasta el objeto ya es aproximadamente igual al set-point, entonces solo se ejecutan los pasos del 1 al 5 hasta que la punta del dedo alcance el set-point. Por otro lado si la distancia desde la punta del dedo al objeto es igual al set-point, entonces no se ejecuta ningún paso. La única manera en la cual se pueden ejecutar lo pasos del 1 al 6 es que ni la falange proximal ni la punta de los dedos hayan alcanzado el set-point.
  Después de que la punta de los dedos alcance el set-point se ejecuta el escenario de contacto (paso 7).

Notación usada:

La distancia de la cual se habla en este y en otros escenarios encapsula la información de distancia de la falange proximal y de la punta de los dedos en una sola percepción llamada *distancia*. Así por ejemplo:

d_ind: Distancia hasta el objeto desde la falange proximal y desde la punta del dedo índice.

4.3.3 Descripción detallada del escenario de adaptación por alineación

Este escenario, representado por la tabla 5, se inicia cuando los dedos han alcanzado la distancia a la cual los sensores ópticos ya pueden percibir la orientación de la superficie local del objeto (que se denominara en adelante como *pose*). De acuerdo a las características de los sensores ópticos esta distancia se ha seleccionado como 3.5 cm. En este escenario, lo primero que se hace es percibir la posición actual de las articulaciones de los dedos (paso 1), la pose (orientación de la superficie local del objeto, paso 2) y la distancia hasta el objeto desde la falange proximal y la punta de los dedos (paso 3). Según la pose percibida, los dedos intentaran llevar la superficie de su punta lo mas paralela posible a la superficie local del objeto (paso 4). Para lograrlo el dedo debe realizar movimientos de aducción-abducción. Esta clase de movimientos puede ocasionar una colisión entre dedos cercanos, y para evitar este suceso es necesario que los dedos se coordinen. De tal manera que los movimientos de aducción-abducción no solo dependen de la pose local sino también de la posición articular de los dedos vecinos, y de sus límites articulares. El dedo pulgar continuamente reconfigura su articulación de oposición de acuerdo a los movimientos de aducción-abducción del dedo anular (paso 4). Este escenario termina cuando la distancia de la falange proximal o de la punta de los dedos al objeto es igual o menor a un umbral (seleccionado en 0.5 cm). Este umbral es necesario para

prevenir que los movimientos de los dedos ocasionen que el objeto se desplace. El paso 5, se realiza cuando, según cálculos, el dedo ya sabe a dónde dirigirse.

Tabla 5. Escenario de adaptación por alineación

Nombre: adaptación por alineación

Descripción: En este escenario el movimiento de los dedos índice medio y anular conduce a que la superficie de sus puntas sean lo más paralela posible a la pose local del objeto y que el dedo pulgar se oponga la dedo anular.
Accionador (*Trigger*): Distancia de la punta del dedo al objeto en el rango 0.5cm<=distancia<=3.5cm.
Pasos:

#	**Tipo de paso**	**paso**	**Funcionalidad**	**Datos:** *** Usados** **+ Producidos**
1	Percepción	posición GDL(3)	Aducción-abducción Oposición_pul	
2	Percepción	pose: orientación de la superficie local del objeto	Aducción-abducción	
3	Percepción	distancia	Aducción-abducción Oposición	
4	Objetivo	moverGDL(3) a || y/o mover GDL(3)pul a opo.	Aducción-abducción Oposición_pul	***posición GDL(3)ind; posición GDL(3)med; y posición GDL(3)anu.** **+ posición GDL(3)ind; posición GDL(3)med; posición GDL(3)anu.**
5	Acción	Mover GDL(3)	Aducción-abducción Oposición_pul	

Alternativas: los pasos del 1 a 5 se repiten cíclicamente hasta que la distancia al objeto desde la punta del dedo o desde la falange proximal sea menor o igual a 0.5 cm.
El objetivo mover ***GDL(3)pul a opo*** (paso 4), solo le concierne al dedo pulgar y depende de los movimientos de aducción-abducción del dedo anular.

Notación usada:

posición GDL(3): Posición actual de la articulación en la cual se llevan a cabo los movimientos de aducción-abducción del GDL(3). En el caso del dedo pulgar es la posición actual de la articulación que realiza el movimiento de oposición.

4.3.4 Descripción detallada del escenario de contacto

Para dar inicio a este escenario, representado por la tabla 6, se supone que todos los dedos ya han alcanzado el set-point, si es así, entonces cada uno de ellos percibe su estado actual (pasos 1-2), y luego de manera autónoma, se flexiona hasta contactar el objeto (pasos 3-5). Si la falange proximal es la primera en contactar el objeto entonces el movimiento de GDL(2) se detiene, pero GDL(1) continua su movimiento hasta que la punta del dedo contacte el objeto. Por otro lado, si la punta del dedo es la primera en contactar el objeto, entonces las articulaciones no deben realizar ningún movimiento.

Tabla 6. Escenario de contacto

Nombre: Contacto

Descripcion: Luego que todos los dedos han alcanzado el set-point, este escenario describe los pasos que se deben llevar a cabo para que todos contacten el objeto.
Accionador (*Trigger*): Distancia de la punta de todos los dedos al objeto aproximadamente igual al set-point.
Pasos:

#	Tipo de paso	paso	Funcionalidad	Datos: * Usados + Producidos
1	Percepción	posición GDL(1y2)	Flexión-Extensión	
2	Percepción	distancia	Flexión-Extensión	
3	Objetivo	mover GDL(1y2) a contacto	Flexión-Extensión	*** d_ind;d_med;d_anu;d_pul**
4	Acción	mover GDL(1)	Flexión-Extensión	
5	Acción	mover GDL(2)	Flexión-Extensión	

Alternativas: el paso 5 se ejecuta únicamente si la distancia desde la falange proximal al objeto es diferente de cero. Para los demás casos los pasos 1 al 5 se ejecutan cíclicamente hasta que la distancia al objeto desde la punta de los dedos y desde la falange proximal sea igual a cero.

4.3.5 Ejecución de los escenarios para los diferentes tipos de agarre

No siempre se ejecutan todos los escenarios, depende del tipo de agarre que desea realizar el paciente. Es el caso del agarre tipo gancho que requiere solo ejecutar el *escenario de*

configuración, cuyo resultado es una configuración inicial de los dedos, pre-agarre. En el tipo de agarre lateral el objeto es agarrado con la parte lateral del dedo índice y la punta del dedo pulgar. Por lo tanto, para este tipo de agarre basta que los dedos, índice, medio y anular se queden quietos en una postura que permita el agarre y que el pulgar se flexione hasta contactar el objeto. Solo es necesario que los dedos ejecuten el escenario de configuración y el pulgar ejecute, adicionalmente, el escenario de contacto. Para este tipo de agarre el accionador del *escenario de Contacto* ya no es la llegada de la punta de todos los dedos al set-point sino la llegada de todos los dedos a su postura de pre-agarre. Para el caso del tipo de agarre de punta, los dedos índice y pulgar se oponen de tal manera que el objeto sea agarrado con sus puntas. Para este tipo de agarre no se ejecuta el *escenario de adaptación por alineación* ya que solo es necesaria una postura de pre-agarre que asegure que los dedos índice y pulgar al realizar el movimiento de flexión se opongan y agarren el objeto de forma exitosa. Los tipos de agarre esférico y cilíndrico pueden ser de poder o de precisión. Si el usuario desea un agarre de precisión entonces simplemente debe posicionar la mano de tal manera que cuando se flexionen los dedos solamente las puntas contacten el objeto. Por el contrario cuando se desea un agarre de poder el usuario debe acercar la mano de tal manera que el objeto sea agarrado con las falanges, con la punta de los dedos, y posiblemente con la palma de la mano. Aunque para los agarres tipo esférico y cilíndrico se pueden ejecutar todos los escenarios cabe anotar que para los agarres de poder los movimientos de aducción-abducción solo se pueden realizar si la falange proximal de los dedos no se encuentran a una distancia menor a 0.5 cm. Esta consideración se hace con el fin de que los dedos no desplacen el objeto al tratar de realizar estos movimientos. Por último, para el tipo de agarre palmar se ejecutan todos los escenarios ya que este tipo de agarre, generalmente, se hace con la punta de los dedos.

Hasta el momento suponemos que los contactos solamente se producen entre los dedos y el objeto. Pero puede suceder que el objeto que se desea agarrar sea relativamente más pequeño en comparación con el tamaño de la mano y en la fase de pre-agarre se produzcan contactos prematuros entre los dedos. Por ejemplo, si desea realizar el agarre cilíndrico de un objeto muy pequeño, los dedos (en la fase de pre-agarre), se flexionan hasta contactar el objeto. En este movimiento de flexión puede suceder que el dedo pulgar contacte primero el dedo anular antes que el objeto, produciendo así un agarre infructuoso. Debido a lo anterior, se hace necesario identificar como varía el movimiento de flexión de un dedo en comparación con la misma clase de movimiento de los otros dedos. Para identificar la relación entre los movimientos de los dedos se toma como referencia el trabajo realizado en [17]. En este trabajo se menciona que cualquier postura de una mano puede ser especificada completamente por medio de los valores de sus articulaciones. Si d es el número de grados de libertad (DOF) de la mano, entonces una postura posible P puede ser definida como:

$$P = [\theta_1 \ \theta_2 \ \ldots\ldots \ \theta_d \ ] \ \in \ \mathcal{R}^d \quad (1)$$

Donde θ_i es el valor del grado de libertad número *i* y d el número de grados de libertad de la mano.

Muchas de las posturas del agarre humano derivan de un conjunto relativamente pequeño de formas de pre-agarre, esto puede implicar que el rango de posturas usadas para llevar a cabo tareas de agarre comunes muestre un agrupamiento significativo en el espacio d-dimensional de los DOF's [17]. En [54] se comprueba esta hipótesis coleccionando una gran cantidad de datos que describen diferentes posturas de agarre que hicieron distintas personas cuando se les dijo que formaran sus manos como si fueran a agarrar un objeto conocido. EL análisis del componente principal de estos datos revela que los primeros dos componentes influyen en casi el 80% de la varianza. Sugiriendo que una muy buena caracterización de los datos guardados, puede ser obtenida usando una aproximación dimensionalmente más baja del espacio total de las articulaciones. En *GraspIt!* a los componentes principales de estas posturas se les conoce como Eigengrasps[8].

Cada Eigengrasp e_i es un vector d-dimensional. El movimiento a lo largo de una dirección Eigengrasp usualmente implicará movimiento a lo largo de todos o algunos grados de libertad de la mano. Un Eigengrasp puede ser definido como:

$$e_i = [e_{i,1}\ e_{i,2}\ \ldots\ldots e_{i,d}] \quad (2)$$

Donde e_i es el Eigengrasp número i, y $e_{i,j}$ el valor del grado de libertad j, del Eigengrasp i.

Si se escoge una base que comprenda b Eigengrasps, una postura de mano ubicada en el sub-espacio definido por esta base puede ser expresada como una función de las amplitudes a_i a lo largo de cada dirección Eigengrasp:

$$P = \sum_{i=1}^{b} a_i e_i \quad (3)$$

Por lo tanto, una postura es definida por el vector de amplitudes $a = [a_1 \ldots \ldots a_b] \in \mathcal{R}^b$.

En este trabajo se hace uso de los Eigengrasp simplemente para describir la relación de movimiento que debe existir entre todos los dedos y para cada uno de los tipos de agarre. Esta clase de relación no es tenida en cuenta para el diseño de la arquitectura basada en multi-agentes ya que como se expondrá más adelante esta es una característica que pertenece al subsistema de control, el cual es el encargado de realizar las respectivas acciones. En este trabajo los Eigengrasp pueden ser encontrados en el modelo de la mano robótica que tiene las características descritas en la sección 2.2.3. El modelo de la mano está contenido en el CD con el nombre: Robonaut_12DOF.xml

[8] El termino "Eigengrasp" es un termino creado por los creadores de *GraspIt!* [15]. En la literatura (particularmente en relación con la mano humana) el mismo concepto es referido como "sinergias de la mano".

4.4 DESCRIPCIÓN DE LA INTERFAZ

Los sistemas multi-agentes suelen estar en ambientes cambiantes y dinámicos. Por lo tanto se hace necesario que los agentes perciban información del ambiente (percepciones) para saber su estado y por medio de las acciones modificarlo. Las percepciones a menudo requieren cierto procesamiento para extraer la información que verdaderamente le interesa al sistema de agentes. Por ejemplo, el enfoque propuesto en este trabajo usa sensores ópticos. Estos sensores, por medio de la amplitud y fase de la luz que llega al receptor, determinan lo que verdaderamente le interesa a los agentes: la distancia de los dedos al objeto y la orientación de la superficie local del objeto. Otros sensores, como los encargados de determinar la posición articular de los dedos, contienen cantidades significativas de ruido, y se hace necesario algún tipo de procesamiento que filtre y limpie los datos. En general, este pre-procesamiento se hace fuera del paradigma de agentes ya que existen lenguajes o librerías más apropiadas para su implementación [41].

Del mismo modo, existen acciones que se realizan mejor fuera del paradigma de agentes. Por ejemplo, existe una cantidad significativa de programación de control de motores requerida para implementar efectivamente una acción. Por lo tanto las acciones que cada dedo realiza, como, por ejemplo, mover la articulación de flexión-extensión, son realizadas mejor por un sub-sistema de control, fuera del paradigma de agentes. Estas decisiones se toman a medida que se va desarrollando la metodología y se especifica de manera más detallada los requerimientos de interacción con el ambiente externo.

4.4.1 Acciones y percepciones

Las percepciones y acciones en el sistema de planificación del agarre que se detallan en la tabla 7 son el resultado del estudio de las interacciones del sistema con el ambiente externo.

Tabla 7. Percepciones y acciones realizadas en cada dedo

dedo	**percepciones**	**acciones**
todos	Valor del tipo de agarre	
pulgar	• d_pul • posición GDL(1y2)pul. • posición GDL(3)pul	• mover GDL(1)pul • mover GDL(2)pul. • mover GDL(3)pul.
índice	• d_ind. • posición GDL(1y2)ind. • posición GDL(3)ind.	• mover GDL(1)ind. • mover GDL(2)ind.

	• pose_ind.	• mover GDL(3)ind.
Medio	• d_med. • posición GDL(1y2)med. • posición GDL(3)med. • pose_med.	• mover GDL(1)med. • mover GDL(2)med. • mover GDL(3)med.
Anular	• d_anu. • posición GDL(1y2)anu. • posición GDL(3)anu. • pose_anu.	• mover GDL(1)anu • mover GDL(2)anu. • mover GDL(3)anu.

En la tabla 7 se muestra la notación usada para las percepciones y acciones realizadas por cada dedo. La combinación de una percepción y el dedo asociado construye la notación. Por ejemplo: **pose_med** es igual que decir: orientación de la superficie del objeto que esta enfrente de la punta del dedo medio; **posición GDL(1y2)med** es la posición actual de las articulaciones de flexión-extensión del dedo medio; **d_:** es la distancia vista por la falange proximal y la punta del dedo en consideración. **Posición:** significa la posición actual de las articulaciones que se colocan después de la abreviación. La razón de esta notación abreviada es para que los diagramas queden de una manera más ordenada y de un tamaño adecuado para un mejor entendimiento. La percepción: ***valor tipo de agarre*** es percibida por todos los dedos para que cada uno de ellos forme su correspondiente postura de pre-agarre.

4.4.2 Las bases de datos

A medida que se desarrollan los escenarios y las funcionalidades, también es importante identificar los datos que hacen parte de éstos. Por ejemplo, un dato producido por la funcionalidad flexión-extensión del dedo índice es la distancia de la punta del dedo al objeto (**d_ind**). Este dato además de ser importante para el dedo índice también es importante para los demás dedos ya que este dato contribuye a que todos los dedos toquen el objeto al mismo tiempo. La idea central de las bases de datos es que la información que tenga un dedo también sea conocida por los dedos que la necesiten. En este trabajo se ha considerado que cada dedo tenga su propia base de datos en la cual se deposite su propia información como: la posición de las articulaciones que forman su postura de pre-agarre, los imites articulares y la posición actual de su articulación de aducción-abducción.

Como resultado se obtienen 4 Bases de datos: **BD dedo índice; BD dedo medio; BD dedo anular; BD dedo pulgar.**

5. FASE DE DISEÑO DE LA ARQUITECTURA.

5.1 ESPECIFICACIÓN DE LOS TIPOS DE AGENTES

La decisión más importante que se debe tomar en el diseño de la arquitectura son los tipos de agentes. Los tipos de agentes se definen mediante la agrupación de funcionalidades. Esta agrupación se realiza considerando los escenarios y es evaluada de acuerdo con los criterios estándar de la ingeniería del software de *acoplamiento* y *cohesión* [41]. El acoplamiento es una propiedad de un grupo de componentes que mide que tanto depende un componente de otros. Un alto acoplamiento produce un sistema en el cual un cambio en un componente requiere cambios en un gran número de otros componentes, volviendo al sistema difícil de modificar. La cohesión es una propiedad de un solo componente. Un componente es cohesivo si todas sus partes están relacionadas. A menudo, la cohesión en un agente está basada en que los objetivos que persigue estén estrechamente relacionados. Un sistema de agentes debe ser lo mas desacoplado posible, pero cada agente debe poseer alta cohesión

5.1.1 Agrupación de funcionalidades

Una posible agrupación de funcionalidades coherente con el proceder de una mano robótica son los diferentes movimientos realizados por un dedo. Por ejemplo, la agrupación de los diferentes movimientos del dedo pulgar, con movimientos en sus articulaciones altamente co-relacionados, y en cierta medida desacoplados con relación al movimiento de los otros dedos. Esta apreciación da lugar a considerar a cada dedo como un agente. Una circunstancia que refuerza la idea de considerar a cada dedo como un agente viene del hecho de que, con bastante seguridad, todos los dedos contribuyen a un agarre estable persiguiendo objetivos propios. La cooperación que debe existir entre los dedos para producir un agarre estable no significa que un dedo dependa estrictamente del otro para lograr sus objetivos. Es por esta razón que si se pierde un dedo, el objetivo general de agarre estable se puede seguir cumpliendo: el dedo clave que es el pulgar y cualquier combinación de los otros dedos [39].

5.1.2 Diagrama de acoplamiento de datos

El agrupamiento de funcionalidades también debe considerar el acoplamiento de datos, es decir agrupar funcionalidades que usen los mismos datos. En este trabajo es mejor que las funcionalidades se agrupen alrededor de los datos que producen y no de los datos que usan. Si se hiciera la agrupación de funcionalidades de acuerdo con los mismos datos usados habría

funcionalidades en un mismo agente que no tienen una relación directa entre sí. Por ejemplo, las funcionalidades de aducción-abducción_ind y aducción-abducción_anu usan la base de datos del agente medio (***BD dedo medio)*** para extraer información que evite colisión entre los dedos anular e índice. Aunque estas funcionalidades usan los mismos datos, en realidad no existe una fuerte cohesión entre ellas ya que los objetivos que persiguen están únicamente relacionados con el dedo que las contienen. En la figura 5.1 se muestra la conexión de las funcionalidades de cada agente con sus respectivas bases de datos. Cada una de las funcionalidades (rectángulos redondeados de color azul) se conectan a la base de datos del agente que la contiene, de esta manera se extrae la posición de pre-agarre que debe alcanzar cada articulación, los limites articulares y se deposita la posición actual de GDL(3), este último dato solo es depositado por la funcionalidad de aducción-abducción. La funcionalidad de aducción-abducción de cada agente se conecta a la base de datos del dedo más cercano para conocer la posición de GDL(3) y así evitar una posible colisión. La funcionalidad de oposición del agente pulgar se conecta a la base de datos del agente anular para conocer la posición de GDL(3)anu y así logre oponerse a este dedo. Por último se debe recordar que el agente pulgar no produce ningún dato y es por esta razón que en la figura 5.1 cada funcionalidad del agente pulgar solo extrae información de la base de datos de este agente.

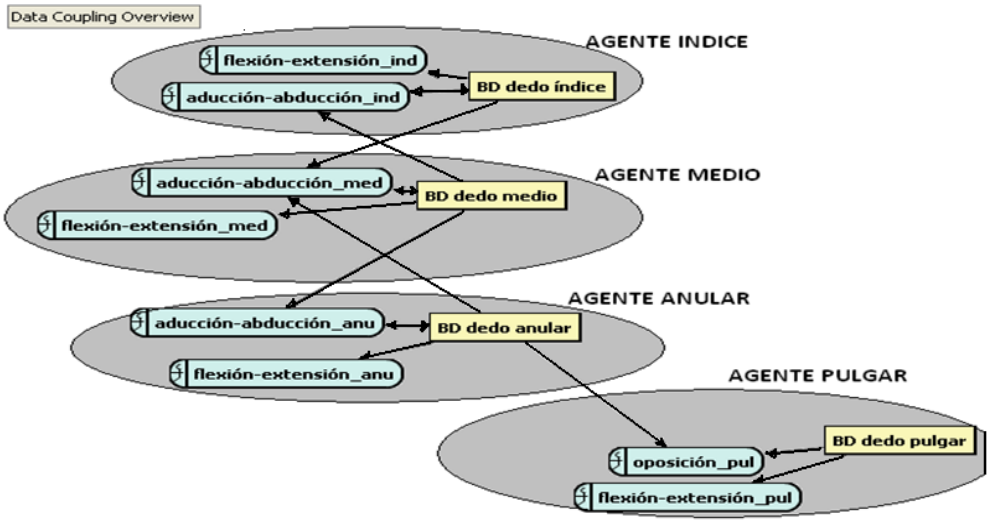


Figura 5.1. Diagrama de acoplamiento de datos.

5.1.3 Revisión del acoplamiento de los agentes - diagrama de relaciones

Los diagramas de relaciones se usan con el fin de evaluar los criterios de acoplamiento entre los agentes. En estos diagramas cada agente es enlazado con el agente que interactúa. Entre menos enlaces existan el sistema será más desacoplado. Una manera simple de analizar el acoplamiento, es determinando la densidad de los enlaces dentro del diagrama. Esta medida es una relación entre el acoplamiento real y el máximo acopamiento posible. El máximo

acoplamiento posible esta dado por la siguiente expresión: $\#agentes * \frac{\#agentes-1}{2}$ (la división por 2 es por que los enlaces son bidireccionales). El resultado de esta expresión es un número que divide el acoplamiento real (número de enlaces dentro del diagrama). Un diseño con baja densidad es menos acoplado y es por lo tanto preferible. Sin embargo, se deben considerar cuellos de botella y criterios de cohesión. En nuestro sistema de planificación solo existen 4 agentes, y estos son únicos, es decir, no existen dos o más agentes que representen un dedo. Esta unicidad hace que no se presenten los cuellos de botella ya que un agente interactúa solo con un agente de tipo distinto. Así, por ejemplo, el agente índice interactúa únicamente con un agente medio y su vez el agente medio interactúa únicamente con un solo agente índice.

En la figura 5.2 se observa el diagrama de relación que existe entre los agentes. En este diagrama la relación que existe entre cada uno de los agentes es bidireccional. Así por ejemplo, el agente medio necesita información de la posición de GDL(3) de los agentes índice y anular, pero estos a su vez necesitan información de GDL(3) del agente medio, esta información es necesaria para evitar una colisión. El agente pulgar se conecta a cada uno de los demás agentes para que sea enviado el s-p actualizado y las ordenes de movimiento y estos a su vez se comunican con el agente pulgar para que se puedan establecer parámetros de coordinación. En este diagrama el acoplamiento es de 5. Este número se obtiene contando los enlaces que existen entre los agentes.

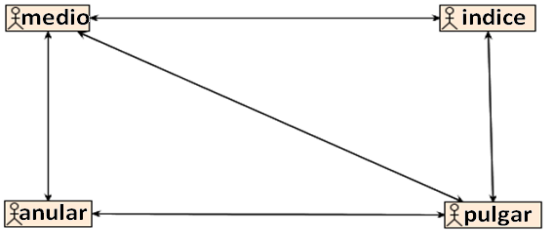


Figura 5.2. Diagrama de relaciones

5.1.4 Desarrollo de los descriptores de agente

Durante y después de tomar la decisión de cuales tipos de agentes formarán parte del sistema, hay algunas preguntas que necesitan ser resueltas:

- ¿Cuantos agentes de cada tipo habrán?
- ¿Cual es el tiempo de vida de cada agente?
- Inicialización del agente: ¿Qué se necesita para inicializar el agente?
- Desaparición del agente: ¿Qué debe hacer el agente antes de desaparecer?

Todas estas preguntas se responden con la ayuda de los descriptores de agente. Estos descriptores se basan en la funcionalidades y contienen información detallada sobre el agente; tal como su nombre, una descripción de lo que hace dentro del sistema y una lista de funcionalidades incorporadas en él. A continuación se presenta el descriptor para el agente pulgar, los descriptores de los otros agentes se pueden consultar en el Anexo B.2.2.

Descriptor del agente pulgar

Nombre: Pulgar
Descripción: Realiza los movimientos de Flexión-Extensión y oposición del dedo pulgar.
Cardinalidad: uno
Tiempo de vida: Desde la llegada del valor que identifica el tipo de agarre hasta el contacto con el objeto.
Inicialización: Lee de la *BD dedo pulgar* la posición de sus articulaciones para formar su postura de pre-agarre.
Desaparición: Al tomar contacto con el objeto, el agente pulgar debe mantener posición de sus articulaciones. En este trabajo solo se realiza de manera detallada la primera fase del agarre, es decir hasta el momento en que todos los dedos hayan tocado el objeto [29]. Para la segunda fase que corresponde a la aplicación de fuerzas se describe y se pone en práctica el enfoque de dedo virtual (ver capitulo 7). Por lo tanto, el agente puede continuar "vivo", cumpliendo ahora con otros objetivos, tal como los objetivos de fuerza.

Funcionalidades incluidas: Flexión-Extensión_pul y Oposición_pul.
Datos usados: *BD dedo pulgar, BD dedo índice, BD dedo medio y BD dedo anular.*
Datos producidos: no produce.
Objetivos:

- obtener GDL(1y2)pul.
- mover GDL(1y2)pul a pos_pre.
- mover GDL(1y2)pul a s-p.
- mover GDL(1y2)pul a contacto.
- obtener GDL(3)pul.
- mover GDL(3)pul a pos_pre.
- mover GDL(3)pul a opo.

Percepciones a las que reacciona:

- valor del tipo de agarre
- d_pul
- posición GDL(1y2)pul.
- posición GDL(3)pul

Acciones que realiza:

- mover GDL(1)pul.
- mover GDL(2)pul.
- mover GDL(3)pul.

Protocolos de interacción: todavía no definidos.

5.2 ESPECIFICACIÓN DE LAS INTERACCIONES

Una vez se han decidido los tipos de agentes, el siguiente paso del diseño de la arquitectura es capturar los aspectos dinámicos del sistema. Esto se logra mediante la especificación de las interacciones entre los agentes.

5.2.1 Diagramas de interacción a partir de los escenarios

Los diagramas de interacción muestran las interacciones entre los diferentes tipos de agentes y son desarrollados a partir de los escenarios creados en la fase de especificación del sistema. El desarrollo consiste en: (a) reemplazar cada funcionalidad con el agente que la contiene; (b) incluir la comunicación entre agentes donde sea necesario y luego, (c) expresar el resultado como un diagrama de interacción. A continuación se desarrollan los diagramas de interacción a partir de cada uno de los escenarios descritos en la fase de especificación del sistema de planificación del agarre (ver sección 4.3). En cada uno de los escenarios se reemplaza cada funcionalidad con el agente que la contiene (ver anexo B.2.3).

Para construir los diagramas de interacción se optó por usar el editor de diagramas *UML Poseidón*. En este editor, cada agente tiene una línea de vida (línea vertical punteada) de la cual salen flechas horizontales que determinan el orden de las interacciones. La información al interior de los símbolos (>…<) y (<…>) describen una percepción y una acción, respectivamente. Si la interacción no contiene estos símbolos es porque se trata de un mensaje. A continuación solo se muestra el diagrama de interacción creado a partir del escenario de adaptación por alineación. Los demás diagramas de interacción creados a partir de los otros escenarios se encuentran en el anexo B.2.3.

A. Diagrama de interacción a partir del escenario de configuración.

El resultado del escenario de configuración es una postura de pre-agarre de todos los agentes (dedos). Para alcanzar esta postura no es necesario que el agente interactué con los demás agentes sino con el ambiente a través de las percepciones: valor tipo de agarre, posición GDL(1y2) y GDL(3)). Así, según el escenario *de configuración*, el primer paso que se realiza es la percepción del valor tipo de agarre. Esta percepción es una señal que llega a todos los agentes y da inicio al funcionamiento del sistema, siendo entonces la primera interacción que debe ir en el diagrama; los pasos del 2 al 5 van después. Sin embargo, debe hacerse notar que la ejecución de los pasos del 2 al 5 no tienen por qué realizarse de modo estrictamente secuencial. Por ejemplo, el agente pulgar puede primero ejecutar el paso 5 y luego el 2 y el funcionamiento del sistema no cambia. Este tipo de comportamiento se describe como ejecución paralela. Luego de la ejecución de los pasos 2 al 5 los agentes deben ejecutar de

forma secuencial los pasos 6, 8 y 9, si el objetivo es mover GDL(1y2) a pos_pre; o los pasos 7 y 10 si el objetivo es mover GDL(3) a pos_pre.

B. Diagrama de interacción a partir del escenario de adaptación por cierre

Después de alcanzar su postura de pre-agarre, los agentes medio, índice y anular envían un mensaje al pulgar avisando que han alcanzado su postura de pre-agarre (en el diagrama: pos_pre_alcanzada). Si al alcanzar su postura de pre-agarre alguna de las distancias que percibe el agente, desde la falange proximal o desde la punta del dedo hasta el objeto, es menor al set-point por defecto (1 cm), entonces, en el mensaje pos_pre_alcanzada, se incluye simultáneamente esta información. El agente pulgar recibe todos los mensajes y mira si al interior de ellos existe información de distancia. Si es así entonces determina cual de las distancias es menor y la envía de regreso a todos los demás agentes (incluyendo el pulgar) para que actualicen su set-point. Esta distancia menor es enviada en un mensaje que ordena el movimiento de flexión de los dedos (orden_mover). Cuando el agente recibe la orden de movimiento empieza a flexionarse (mover GDL(1), y GDL(2)) hasta que alguna de las distancias que percibe el agente, desde la falange proximal o desde la punta del dedo hasta el objeto, sea aproximadamente igual al set-point actualizado. Cuando todos los agentes han alcanzado el set-point se da inicio al escenario de contacto. Según el proceso descrito, las percepciones que se describen en los pasos 1 y 2 se pueden llevar a cabo de forma paralela. Los pasos 3 y 4 se ejecutan de forma secuencial ya que el set-point debe ser actualizado antes de que los dedos intenten alcanzarlo. Las acciones descritas en los pasos 5 y 6 se ejecutan de forma paralela y después de haber llevado a cabo los pasos del 1 al 4.

C. Diagrama de interacción a partir del escenario de adaptación por alineación.

En la figura 5.3 se muestra el diagrama de interacción obtenido a partir del escenario de adaptación por alineación, este diagrama empieza cuando los dedos han alcanzado su postura de pre-agarre. Cuando esto ocurre los agentes perciben la posición del grado de libertad encargado de aducción-abducción (posición GDL(3)), la distancia del dedo hasta el objeto (d_) y la orientación de la superficie local del objeto (pose). Si la distancia detectada por la punta de los dedos es mayor a 0.5 cm y menor a 3.5 cm entonces los sensores ópticos ya pueden percibir la pose local. Esta pose se usa para que los agentes índice medio y anular, por medio de movimientos de aducción-abducción (mover GDL(3)), intenten llevar la superficie de su punta lo mas paralela posible a la superficie local del objeto. Antes de realizar esta clase de movimiento los agentes deben comunicarse entre ellos con el fin de evitar una colisión. Por ejemplo, el agente índice para colocar la superficie de su punta paralela a la superficie local del objeto debe realizar un movimiento de aducción-abducción. Este movimiento puede ocasionar una colisión con el agente medio. Para evitarlo, el agente índice, antes de realizar cualquier movimiento, solicita la posición de GDL(3) del agente medio (posición_GDL(3)_med?) y éste

inmediatamente se la envia (posición_GDL(3)_med). También, cuando el agente anular esta tratando de colocar la superficie de su punta lo mas paralela posible a la superficie local del objeto, envía un mensaje al agente pulgar (movimiento GDL(3)anu) donde informa sobre la posición actual de GDL(3) y de acuerdo a esta información el dedo pulgar realiza su movimiento de oposición (mover GDL(3)pul). Considerando el proceso descrito, los pasos 1, 2 y 3 se realizan en paralelo al igual que los pasos 4 y 5. Estos últimos se realizan en paralelo debido a que la única forma de posicionar la punta del dedo lo mas paralela posible a la superficie local del objeto es a través del grado de libertad encargado de realizar los movimientos de aducción-abducción (GDL(3)).

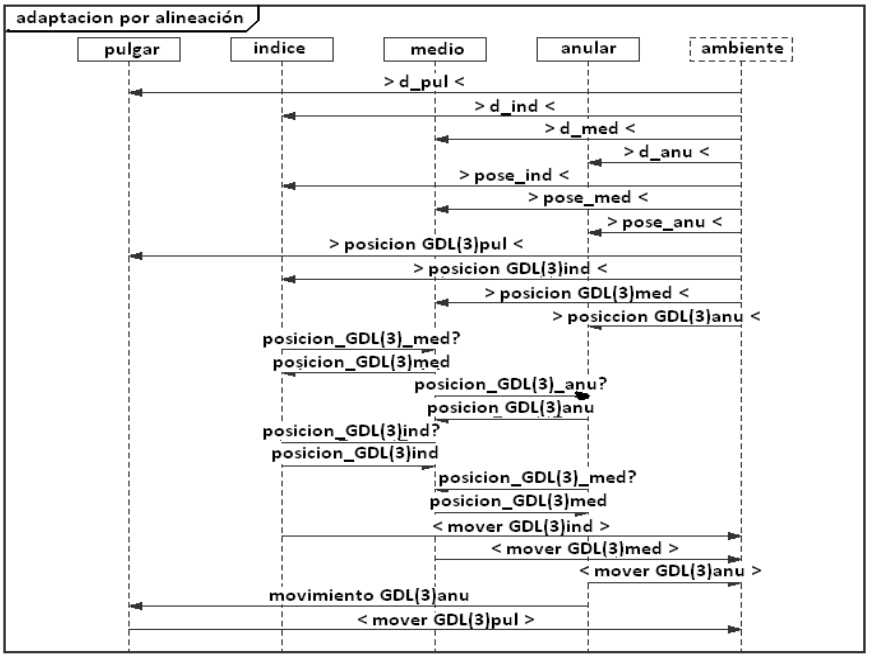


Figura 5.3. Diagrama de interacción de acuerdo con el escenario de adaptación por alineación.

D. Diagrama de interacción a partir del escenario de contacto

Cuando los agentes índice, medio y anular han alcanzado el set-point, envían un mensaje al agente pulgar avisando que dicha distancia ha sido alcanzada (_alcanza s-p). El agente pulgar recibe estos mensajes y si él también ha alcanzado el set-point, entonces envía de regreso un mensaje de orden de movimiento a todos los agentes (orden_mover). Cuando los agentes reciben este mensaje perciben la distancia hasta el objeto (d_) y su posición actual (posición GDL(1y2)). Estas percepciones se hacen con el fin de que los agentes continúen flexionándose (mover GDL(1) y GDL(2)) hasta detectar contacto. Según lo anterior, los pasos 1 y 2, descritos en el escenario, se hacen de forma paralela. El siguiente es el paso 3 que describe el objetivo de contactar el objeto. Para cumplir con este objetivo el agente debe mover sus articulaciones por lo que los pasos 4, 5 y 6 se deben realizar en forma paralela.

5.2.2 Desarrollo de protocolos de interacción a partir de los diagramas de interacción

En los escenarios se obtuvo solamente un conjunto representativo de diagramas de interacción, no el conjunto completo de interacciones que se deben cumplir. Para completar y precisar estas interacciones se pasará de los diagramas de interacción a los protocolos, en los cuales se definen únicamente las secuencias de interacción entre los agentes. Según se estableció en la sección anterior, en nuestro sistema de planificación se han identificado 4 diagramas de interacción. De estos diagramas se puede reconocer que los agentes interactúan básicamente cuando están realizando dos movimientos: de flexión y de aducción-abducción. En el movimiento de flexión los agentes interactúan para lograr contactar el objeto al mismo tiempo; en el movimiento de aducción-abducción los agentes interactúan para evitar una colisión entre ellos. Esto conlleva a producir dos protocolos de interacción: el protocolo para la **flexión** y el protocolo para la **aducción-abducción**. En estos protocolos los agentes cooperan y se coordinan entre sí para lograr una postura de agarre estable sobre el objeto.

Para desarrollar los protocolos se necesita incluir una notación especial que describan estructuras de control, tales como secuencia, selección, iteración, y otras, que pueden aparecer en el desarrollo de las interacciones. La notación especial usada en este trabajo es la *AgentUML*, recomendada por la metodología Prometheus [52].

De manera similar a los diagramas de interacción, las líneas verticales punteadas que están debajo de cada agente representan las líneas de vida (lifelines) para cada agente. En este caso, las flechas horizontales que salen de estas líneas de vida representan los mensajes que se trasmiten entre dos agentes. Estos mensajes están contenidos en lo que se denominan ***cajas***. Cada caja tiene un identificador que describe el tipo de caja. Una caja de tipo ***option (opt)*** tiene un *guard* (un texto entre corchetes) que indica una condición que debe ser verdadera para que el contenido de la caja pueda ser ejecutado. Una caja de tipo ***alternativa (alt)*** puede tener varias regiones (cada una con su propio *guard*), pero solamente la condición de un *guard* es verdadera y por lo tanto solo una de estas regiones es ejecutada. Cada región en las cajas se representa como una línea horizontal punteada. Una caja de tipo **par*allel (*par*)*** especifica que las regiones pueden ser ejecutadas en cualquier orden. Una caja de tipo ***Loop*** especifica cuantas veces se repite la región o regiones que contenga la caja. En este tipo de caja después del identificador (*loop*) se debe describir el número de repeticiones que se van a realizar al interior de la caja. Si no se especifica el número de repeticiones entonces quiere decir que el contenido de la caja se ejecuta de forma indefinida.

En el desarrollo de los protocolos el diagrama de interacción que resulta del escenario de configuración no será tenido en cuenta, ya que la función de los protocolos es describir las interacciones detalladas entre los agentes, y en este diagrama las interacciones se realizan únicamente con el ambiente. El agarre tipo gancho solo usa el escenario de configuración, y por lo tanto no usa ningún protocolo; solo basta que los dedos formen una correcta postura de pre-agarre. Para el tipo de agarre lateral los agentes usan únicamente el protocolo de flexión para

avisar al agente pulgar la llegada a su postura de pre-agarre. Para el tipo de agarre punta, los agentes índice y pulgar usan únicamente el protocolo para la flexión, el protocolo para la aducción-abducción no lo usan debido a que no es necesario que el agente índice realice movimientos de aducción-abducción, ya que antes ha formado una correcta postura de pre-agarre. Para los otros tipos de agarre los agentes hacen uso de ambos protocolos, para la flexión y para la aducción-abducción.

B. Protocolo para la flexión

En la figura 5.5 se muestra el protocolo para la flexión. El inicio del protocolo para la flexión es cuando los agentes envían un mensaje al agente pulgar avisando que su postura de pre-agarre ya ha sido alcanzada. Estos avisos se hacen de forma paralela es decir que puede tomar cualquier orden, por esta razón está contenido en una caja de tipo ***par.*** Cuando todos los agentes ya han alcanzado la postura de pre-agarre se verifica por medio de una caja detipo ***alt*** si el tipo de agarre es lateral si es así entonces este protocolo no se sigue ejecutando ya que para este tipo de agarre solo basta formar una correcta postura de pre-agarre. Si por el contrario, el tipo de agarre no es lateral entonces el agente pulgar envía en paralelo (caja de tipo ***par***) las ordenes de movimiento para que los agentes índice, medio y anular se flexionen hasta alcanzar el s-p, cuando estos agentes alcanzan el s-p envían en paralelo (***par***) un aviso que informa sobre su llegada a este punto. En ese momento que se describe por medio de una caja de tipo ***opt*** el agente pulgar revisa que el s-p actualizado sea diferente de cero, si es igual a cero entonces no envía la siguiente orden de movimiento en caso contrario envía la siguiente orden de movimiento que hace que los dedos se flexionen hasta contactar el objeto.

A. Protocolo para la aducción-abducción

En la figura 5.4 se muestra el protocolo para la aducción-abducción (ad/abducción). Lo primero que se hace es introducir todas las interacciones en una caja de tipo ***opt***. Con esto se garantiza que este protocolo se ejecuta si todos los agentes ya han alcanzado su postura de pre-agarre. Si es así, entonces las interacciones se introducen en una caja de tipo ***alt.*** En la cual se verifica la distancia de cada uno de los agentes al objeto. Si esta distancia es menor o igual a 0.5 cm, entonces no se deben realizar movimientos de aducción-abducción, ya que el objeto se desplazaría, si por el contrario la distancia es mayor a 0.5, estos movimientos son permitidos y las interacciones se introducen en una caja de tipo ***par***, en la cual se verifica si la distancia hasta el objeto desde cada uno de los agentes es menor o igual a 3.5 cm, si la distancia se encuentra en este rango entonces en paralelo los agentes solicitan información de GDL(3) del agente que se encuentra más cercano. Todas las interacciones anteriores están contenidas en una caja de tipo ***loop*** la cual hace que las interacciones se repitan hasta el instante en que los agentes perciban una distancia menor a 0.5cm. El movimiento de oposición del agente pulgar depende de los movimientos de aducción-abducción del agente anular, por lo tanto cuando el

agente anular se mueve envía la posición de GDL(3) al agente pulgar para que éste reconfigure GDL(3) y así logre oponerse al agente anular. Lo anterior puede verse como una interacción entre el agente anular y pulgar pero en realidad en un simple mensaje para el cual no existe replica, es decir, que el mensaje se trasmite en un solo sentido. Es por esta razón que no es necesario que el entre estos dos agentes exista alguna clase de protocolo de interacción para lograr la oposición.

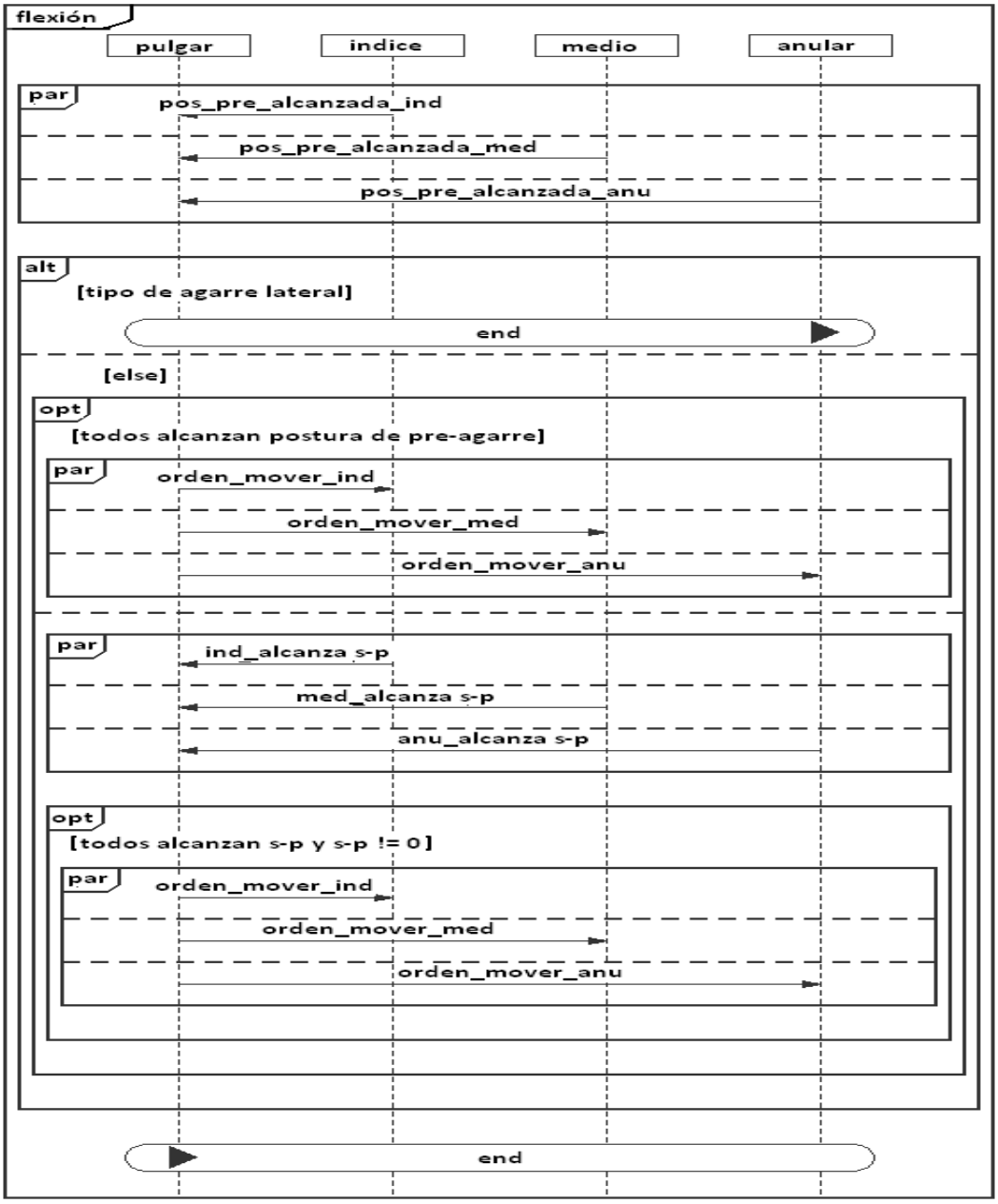


Figura 5.5. Protocolo para la flexión.

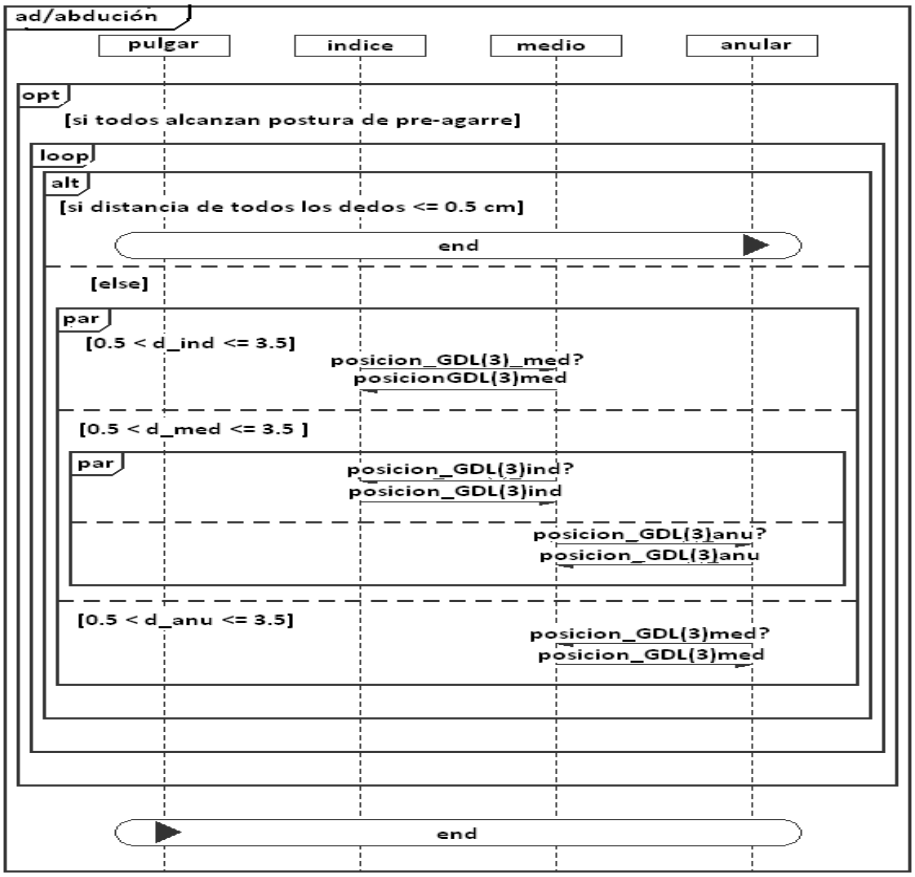


Figura 5.4. Protocolo para la aducción-abducción.

5.2.3 Desarrollo de los descriptores de protocolos y mensajes

Así como ocurre en el diseño final de cada fase del sistema, ahora se desarrollaran los descriptores de los protocolos y los mensajes.

A. Protocolos

- **Protocolo para la Flexión**
  Nombre: Flexión
  Descripción: Interacción entre los agentes en todo el proceso que involucra el movimiento de flexión de los dedos.
  Mensajes incluidos: pos_pre_alcanzada_ind; pos_pre_alcanzada_med; pos_pre_alcanzada_anu; orden_mover_ind; orden_mover_med; orden_mover_anu; ind_alcanza s-p; med_alcanza s-p; anu_alcanza s-p
  Escenarios: *Adaptación por cierre; Contacto.*

Agentes: Todos.
Notas:

- **Protocolo para la aducción-abducción**
  Nombre: ad/abducción.
  Descripción: Interacción entre los agentes en todo el proceso que involucra los movimientos de aducción-abducción de los dedos.
  Mensajes incluidos: posición_GDL(3)_med?; posición_GDL(3)_med; posición_GDL(3)_ind?; posición_GDL(3)_ind, posición_GDL(3)_anu?, posición_GDL(3)_anu;
  Escenarios: *Adaptación por alineación.*
  Agentes: Todos.
  Notas: los agentes solo reciben información sobre GDL(3), no de todas las posiciones articulares del dedo o dedos cercanos.

B. Mensajes

Debido a la gran cantidad de mensajes que se transmiten entre agentes, a continuación solo se mostrarán los descriptores de los mensajes que transmite y le son trasmitidos al agente medio. Los demás mensajes se describen en el Anexo B.2.4.

- **Postura de pre-agarre alcanzada por parte del agente medio**
  Nombre: pos_pre_alcanzada_med
  Descripción: En este mensaje se indica que la postura de pre-agarre del agente medio ya ha sido alcanzada.
  Del agente: medio
  Para el agente: pulgar
  Propósito: indicar que la postura de pre-agarre del agente medio ya ha sido alcanzada.
  Información llevada: distancia hasta el objeto desde la punta o desde la falange del dedo medio. Si esta información está contenida en el mensaje es porque la distancia del dedo al objeto es menor que el set-point por defecto (1 cm)

- **Orden de movimiento de flexión del agente medio**
  Nombre: orden_mover_med
  Descripción: En este mensaje se indica que el dedo medio puede seguir realizando sus movimientos de flexión.
  Del agente: pulgar
  Para el agente: medio
  Propósito: activar el movimiento de flexión del dedo medio.
  Información llevada: set-point actualizado por parte del agente pulgar. Este set-point es la distancia menor de todas las distancias enviadas en los mensajes pos_pre_alcanzada de cada agente.

- **El agente medio alcanza el set-point**
  Nombre: med_alcanza s-p
  Descripción: En este mensaje se indica que el agente medio alcanzo el set-point.
  Del agente: medio
  Para el agente: pulgar
  Propósito: indicar al agente pulgar que la distancia hasta el objeto es igual a la distancia actualizada o a la distancia establecida por defecto, es decir, igual al set-point.
  Información llevada: ninguna, solo indica.

- **Posición actual de GDL(3) del agente medio?**
  Nombre: posición_GDL(3)_med?
  Descripción: En este mensaje se solicita la posición actual de GDL(3) del agente medio.
  Desde los agentes: índice, anular
  Al agente: medio
  Propósito: que se le envíe al agente que lo solicite la posición de GDL(3) del agente medio.
  Información llevada: ninguna, solo es una solicitud.

- **Posición actual de GDL(3) del agente medio**
  Nombre: posición_GDL(3)_med
  Descripción: mensaje en el cual se envía la posición actual de GDL(3) del agente medio.
  Del agente: medio
  Para los agentes: anular, índice (según el agente que solicite)
  Propósito: enviar la posición actual de GDL(3) del agente medio.
  Información llevada: valor actual de GDL(3) del agente medio.

5.3 DISEÑO FINAL DE LA ARQUITECTURA

En el diseño final de la arquitectura se debe reunir toda la información generada a partir de los tipos agentes dentro del sistema y las interacciones que existen entre ellos. Una vez reunida esta información se necesita desarrollar una especificación completa de los datos que necesitan los agentes, la información del ambiente a la cual deben responder los agentes y las acciones que realizan cada uno de ellos. Toda esta información sirve para generar el *Diagrama General Del Sistema* y de esa manera capturar toda la arquitectura del sistema de agentes para la planificación del agarre.

5.3.1 Identificación de los límites del sistema de agentes

En la arquitectura de planificación del agarre que se propone en este trabajo se ha encontrado conveniente realizar el procesamiento de las señales en un subsistema que no es modelado

con agentes. En el caso de las percepciones, los sensores ópticos entregan señales que han sido procesadas para que el sistema multi-agente las entienda. Por ejemplo, Un sensor óptico básico consiste de un emisor, un foto-receptor y un circuito de procesamiento de señal. La luz del emisor es reflejada por una superficie cercana y la recibe el foto-receptor. La amplitud y fase de la luz varía en función de la distancia a la superficie y su orientación [11]. Este tipo de señales no es la que necesita el sistema de agentes, ya que a este sistema solo le interesa la distancia hasta el objeto y la orientación de la superficie local del objeto. Por lo tanto, el procesamiento de la amplitud y fase de la luz de todos los sensores ópticos se hace fuera del paradigma de agentes y se asume que este procesamiento ya ha sido realizado. Al igual que las percepciones, las acciones que los agentes realizan sobre el ambiente se asumen han sido modeladas fuera del paradigma de agentes.

En conclusión, el sistema general esta compuesto por la parte de sensado y control pero estos serán tomados como sub-sistemas que no son parte del modelado con agentes.

5.3.2 Desarrollo de los descriptores de las percepciones y acciones

Las percepciones son la información procesada que los agentes reciben del ambiente, mientras que las acciones representan el efecto que el agente puede tener sobre el ambiente externo. En este trabajo se ha considerado realizar únicamente los descriptores de las acciones y percepciones del dedo índice. Esta decisión se toma debido a que la información que contienen estos descriptores es básicamente la misma para todos los descriptores que pertenecen a los demás agentes, con excepción del agente pulgar quien no realiza movimientos de aducción-abducción sino de oposición.

A continuación se desarrollaran los descriptores de las acciones y el descriptor de una sola percepción. Los descriptores de las otras percepciones y acciones se encuentran en el Anexo B.1.5.

A. Percepciones

- **Valor del tipo de Agarre**

  Nombre: tipo de agarre
  Descripción: información que indica el tipo de agarre que desea realizar el paciente.
  Información llevada: Valor entero de un byte. A cada bit se le asigna un tipo de agarre [29].
  Actualización de la información: Actualiza el valor que representa el tipo de agarre que desea realizar el paciente.
  Recurso: electrodos.
  Procesamiento: el agente recibe esta percepción directamente y el análisis de su información se hace internamente.
  Agentes: pulgar, índice, medio y anular.

Frecuencia esperada: solo se espera que llegue una vez, por cada agarre.

B. Acciones

Así como las percepciones, las acciones también pueden ser complejas, requiriendo un rigoroso diseño y desarrollo fuera del paradigma de agentes, especialmente, cuando se usan actuadores físicos, como en el caso de una mano robótica.

Los descriptores que se muestran a continuación contienen toda la información necesaria de las acciones que realiza el agente índice sobre el ambiente. No se ha tenido en cuenta la acción mover GDL(2)ind. Ya que la información que contiene su descriptor es básicamente la misma a la contenida en el descriptor de la acción mover GDL(1)ind.

- **Mover GDL(1) para contribuir al proceso de flexión del dedo índice.**

  Nombre :mover GDL(1)ind.
  Descripción: mover GDL(1) para contribuir al proceso de flexión del dedo índice.
  Parámetros: Aceleración y velocidad angular de las articulaciones.
  Temporalidad: Duracional (puede tomar algunos ms).
  Detección de fallas: Una falla ocurre cuando la posición de GDL(1) no es alcanzada en un determinado tiempo o nunca es alcanzada.
  Cambio parcial: desplazamiento de la posición de GDL(1) hasta el instante en el cual ocurrió la falla.
  Efectos secundarios: reducción de la potencia de la batería.

- **Mover GDL(3) para permitir los movimientos de aducción-abducción del dedo índice.**

  Nombre : mover GDL(3)ind.
  Descripción: mover GDL(3) para permitir los movimientos de aducción-abducción del dedo índice.
  Parámetros: aceleración y velocidad angular de las articulaciones,
  Temporalidad: Duracional (puede tomar algunos ms)
  Detección de fallas: Una falla ocurre cuando la posición articular no es alcanzada en un determinado tiempo o nunca es alcanzada.
  Cambio parcial: desplazamiento de la articulación hasta el instante en el cual ocurrió la falla.
  Efectos secundarios: reducción de la potencia de la batería.

5.3.3 Definición de los objetos de datos compartidos (*Shared Data Objects*)

En esta sección se desarrollan los descriptores de las bases de datos que fueron encontradas en la fase de especificación del sistema. A continuación solo se desarrolla el descriptor de la

base de datos del agente índice. Los descriptores de las bases de datos de los agentes: anular, pulgar y medio contienen básicamente la misma información del descriptor de la base de datos del agente índice y pueden ser encontradas en el Anexo B.1.5.

Nombre: BD dedo índice.
Descripción: En esta base de datos se guardan tres tipos de datos: i) información sobre la posición que deben alcanzar GDL(1y2) y GDL(3) para formar la postura de pre-agarre del dedo índice. ii) ángulos máximos y mínimos de GDL(1y2) y GDL(3) (limites articulares). iii) posición actual de GDL(3).
Tipos de datos: flotante.
Campos o aspectos incluidos: los tres tipos de datos que se describieron anteriormente.
Producido por: agente índice.
Usado por: el agente medio y el agente índice.
Persistente: (*este ítem detalla si la información en la base de datos está siendo constantemente modificada o es persistente, es decir, constante):* Los 2 primeros tipos de datos que se guardan en la base de datos son persistentes pero el último no lo es.
Inicialización: ninguna.
Cuando se usa: Cuando llega el valor del tipo de agarre y cuando él o un agente cercano intenta realizar movimientos de aducción-abducción.

5.3.4 Diagrama de visión general del sistema

Los pasos anteriores han identificado todas las partes de la arquitectura del sistema. Existen dos artefactos para agrupar esta información: el diagrama de visión general del sistema y el diccionario de datos. El diagrama de visión general del sistema es indudablemente el artefacto más importante del proceso de diseño y es por lo tanto el que se desarrolla en este trabajo. En la figura 5.7 se desarrolla el diagrama de visión general del sistema. Los símbolos usados en este diagrama se muestran en la siguiente figura:

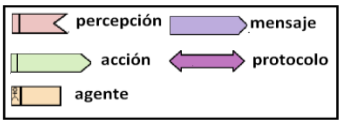


Figura 5.6. Símbolos gráficos utilizados en el diagrama de visión general del sistema. [41].

En el diagrama general se enlazan las percepciones y las acciones con sus respectivos agentes. Luego los protocolos quienes enlazan a los agentes que interactúan mediante ellos. Por ejemplo, para evitar una colisión con agentes cercanos el agente medio interactúa con los agentes índice y anular mediante el protocolo llamado ad/abducción, y viceversa. Así mismo, para contactar el objeto al mismo tiempo cada uno de los agentes se comunica por medio del protocolo de flexión con el agente pulgar. El mensaje (***movimiento GDL(3)anu***) que le envía el agente anular al agente pulgar es utilizado para que exista una oposición entre estos agentes.

Este mensaje no necesita de un protocolo debido a que el intercambio de información es en un solo sentido. Para realizar todo lo anterior existe la herramienta de Diseño De Prometheus (PDT) donde a medida que se van definiendo los agentes y los protocolos estos van apareciendo automáticamente en la plantilla de diseño y lo único que tiene que hacer el usuario es enlazarlos.

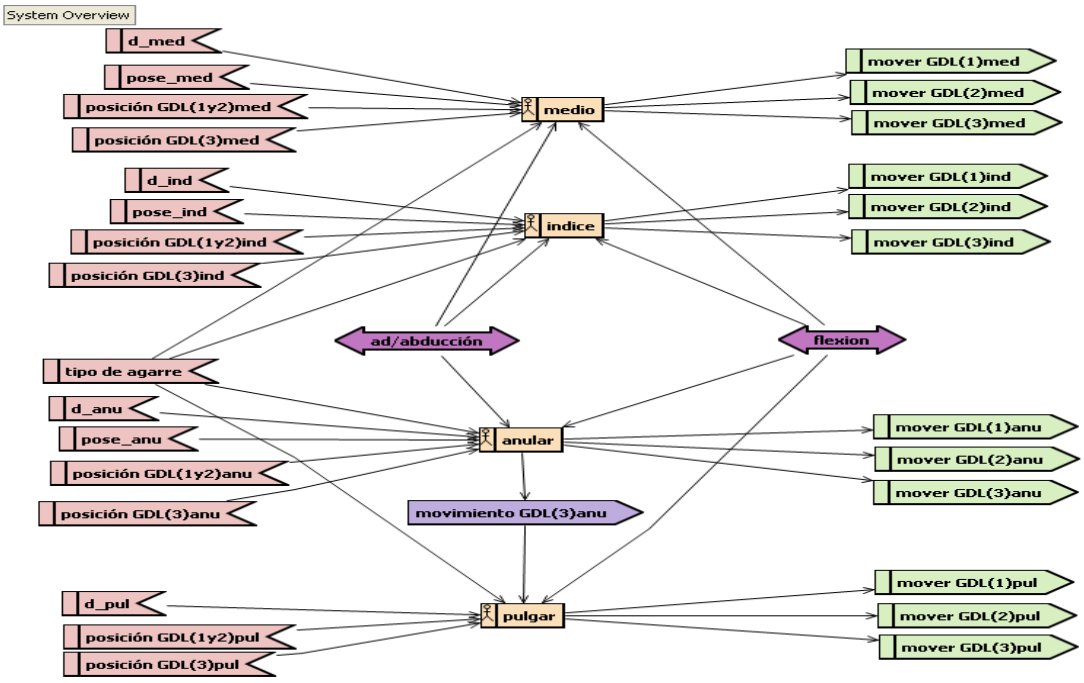


Figura 5.7. Diagrama de visión general del sistema para la planificación de agarre en una mano robótica.

En la figura 5.7 se observa como los agentes usan los protocolos de interacción para comunicarse entre sí y lograr cumplir con sus objetivos de diseño. Además, se observa la información que los agentes perciben del ambiente y las acciones que ejecutan. Las razones anteriores son suficientes para que la figura 5.7 sea considerada como el artefacto final de diseño que represente la arquitectura de planificación que se propone en este trabajo. Sin embargo, los múltiples enlaces y la falta de organización entre entidades, hacen que esta figura no sea la más idónea para representar la arquitectura de planificación.

En este trabajo se ha construido un diagrama simplificado en el cual se muestra la estructura completa del sistema y en donde cada una de las entidades y su enlaces son presentados de una manera mas ordenada y entendible. Este nuevo diagrama, mostrado en la figura 5.8, presenta a los agentes: ***pulgar, índice, medio y anular*** interactuando por medio de dos protocolos; el primer protocolo para flexión-extensión (*flex-ext*) es utilizado por todos los agentes para contactar el objeto al mismo tiempo; el segundo protocolo para aducción-abducción (*ad-abd*) es utilizado por los agentes para evitar una posible colisión entre ellos.

Adicionalmente, se cuenta con un mensaje que le trasmite el agente anular al agente pulgar con el fin de que siempre exista un espacio de posición entre el agente pulgar y los demás agentes.

En este diagrama también se muestran las percepciones y acciones que deben llevar a cabo los agentes para cumplir con sus objetivos de diseño. Las percepciones, encapsuladas en el bloque de ***sensado***, están separadas según la información que contengan. *El clasificador de agarre* se refiere a la interpretación que se debe hacer de las señales electromiográficas para identificar el tipo de agarre que desea realizar el paciente sobre el objeto. *El estimador de distancia y orientación*, entrega información relacionada con la orientación de la superficie local del objeto y la distancia desde la punta y falange proximal de cada dedo al objeto. *La posición articular*, entrega información sobre la posición actual de los grados de libertad (GDL) en cada uno de los agentes. Toda la información (*información procesada*) que le trasmite el bloque de sensado a cada uno de los agentes se asume ya ha sido procesada y por lo tanto su diseño no se considera en este trabajo.

Las acciones, representadas por el bloque de ***actuadores***, contiene toda la programación de control que se debe llevar a cabo para mover los motores que representan cada uno de los GDL de la mano robótica. El movimiento de estos motores (*acción mecánica*) se realiza de acuerdo a las órdenes de posición (*señales de control*) enviadas por cada uno de los agentes y hace que el bloque de sensado detecte nuevas señales (*realimentación de la mano*); tal como la distancia hasta el objeto, su orientación local y la posición articular de los dedos de la mano.

El usuario es muy importante en el proceso del agarre ya que gracias a su visión (*realimentación visual*) que éste genera las señales electromiográficas que permiten identificar el tipo de agarre y de posicionar la mano de tal manera que se pueda producir un agarre eficiente sobre el objeto.

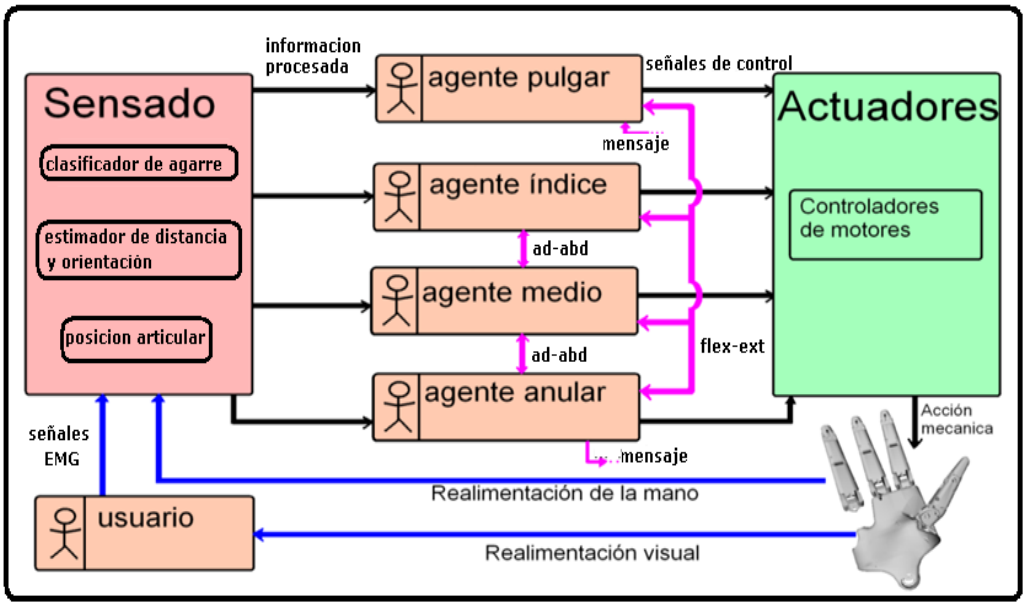


Figura 5.8. Arquitectura basada en agentes

6. FASE DE DISEÑO DETALLADO

El diseño detallado se enfoca en los siguientes aspectos:

- Construir el diagrama de visión general del agente en términos de sus capacidades. La capacidad de un agente es la agrupación de funcionalidades al interior del agente. Esta agrupación se basa en la estrecha relación que pueda existir entre algunas funcionalidades.
- Construir el diagrama general de las capacidades. Al interior de estos diagramas se encuentran los planes los cuales suministran los detalles de cómo lograr los objetivos y cómo reaccionar ante diferentes situaciones.
- Desarrollar las especificaciones de los procesos los cuales muestran de manera aún mas detallada la secuencia en la cual se transmiten los mensajes entre agentes y son creados a partir de los protocolos de interacción.
- Especificación del algoritmo dentro de cada plan, así como también de los datos asociados y la especificación detallada de los eventos. Esto es capturado en los descriptores de planes, datos y eventos.

Para generar cada uno de los diagramas se usa la simbología determinada por la herramienta de Prometheus (PDT). En este simbología los mensajes son de color morado; los planes de color azul; las acciones de color verde; las BD de color amarillo; las capacidades de color naranja. Las entidades más descoloridas ya habían sido creadas y representan la interacción del agente con su mundo externo.

Por razones de espacio, en esta monografía solamente se realiza el diseño detallado del agente índice. El diseño detallado correspondiente a los demás agentes se encuentra en el Anexo B.3

6.1 DIAGRAMA DE VISIÓN GENERAL DEL AGENTE

En la figura 6.1 se muestra el diagrama de visión general del agente índice. Las capacidades que se encuentran en el interior de este diagrama tienen el mismo nombre y cumplen con los mismos objetivos de las funcionalidades que fueron identificadas para el agente índice. Esta correspondencia de uno a uno entre estas entidades se debe a la alta cohesión que existe al interior de las funcionalidades y que no permite más particiones.

El diagrama de visión general del agente también están confinados los planes. Para el agente índice se identificaron los siguientes planes:

- Plan **actualizar s-p**: cuando llega el mensaje *orden_mover_ind*, este plan mira en su interior. Si existe un set-point diferente al establecido por defecto (1 cm), entonces lo actualiza y lo envía en el mensaje *s-p actualizado* a la capacidad *flexión-extensión_ind.*

- Plan **asignar pos_pre_ind:** dependiendo del tipo de agarre, este plan es el encargado de asignar de la *BD dedo índice* las posiciones de pre-agarre a las capacidades de *flexión-extensión_ind* y *aducción-abducción_ind* en los mensajes *pos_pre_GDL(1y2)ind* y *por_pre_GDL(3)ind,* respectivamente.

- Plan **pos_pre_ind_alcanzada**: Este plan es el encargado de enviar el mensaje *pos_pre_alcanzada_ind* al agente pulgar. En este mensaje se avisa que la postura de pre-agarre del agente índice ya ha sido alcanzada. El envío de este mensaje depende de si los grados de libertad comandados por las capacidades de *flexión-extensión_ind* y *aducción-abducción_ind* ya han alcanzado su posición de pre-agarre.

- Plan **respuesta a posición_ind**: Plan encargado de dar respuesta a las solicitudes de posición de GDL(3) del agente índice que esta confinada en *BD dedo índice*. Esta solicitud se hace por medio del mensaje *posición_GDL(3)_ind?* y se responde por medio el mensaje *posición_GDL(3)_ind.*

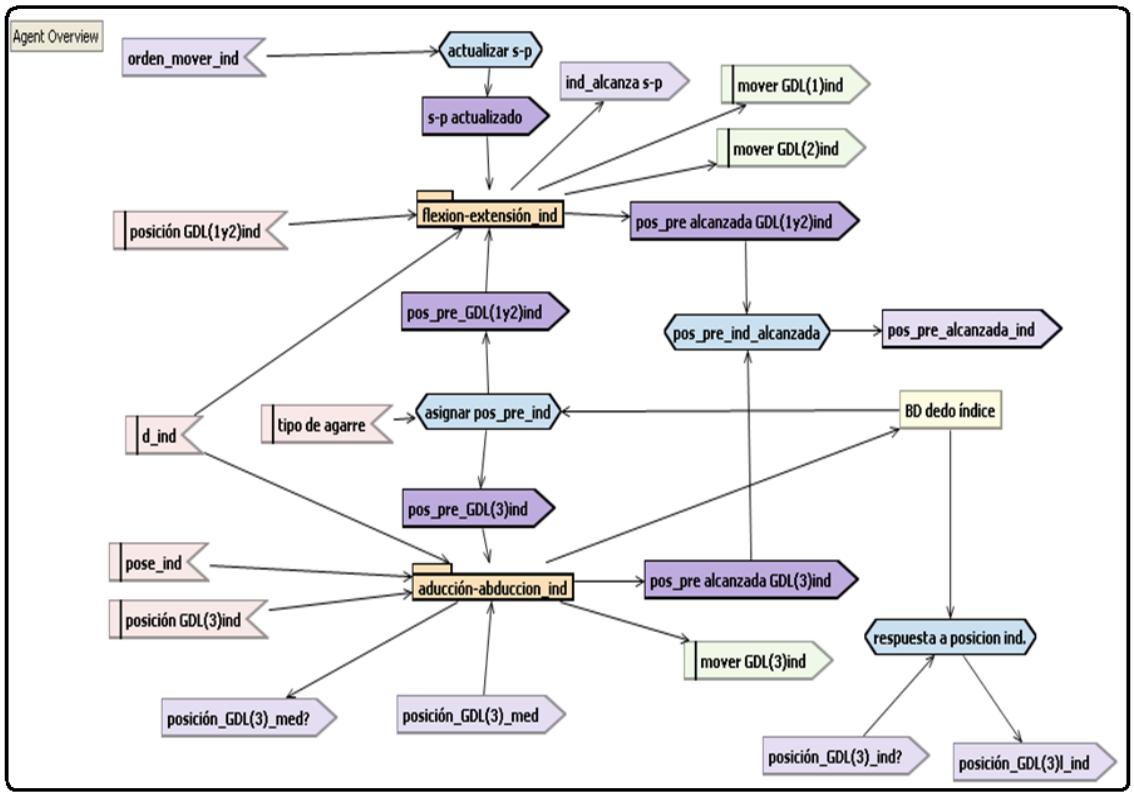


Figura 6.1. Diagrama de visión general del agente índice.

A partir de la figura 6.1 se construye un diagrama simplificado de la vista general del agente, el cual muestra la arquitectura interna de los agentes: índice medio y anular. En este nuevo diagrama, representado por la figura 6.2, se muestra la interfaz del agente con el ambiente (*percepciones y las señales de control*) y con otros agentes (*interfaz de comunicación*). Esta última interfaz es representada por los protocolos. En este diagrama también se muestran las capacidades de flexión-extensión y aducción-abducción que se encargan de responder a las percepciones y mensajes que entran al agente, y generar así, cambios en el ambiente tal como: el movimiento coordinado de los dedos. Cada una de estas capacidades usa o produce datos en la base de conocimiento. Esta base de conocimiento es la misma base de datos interna que tiene cada agente (ej. *BD dedo índice*).

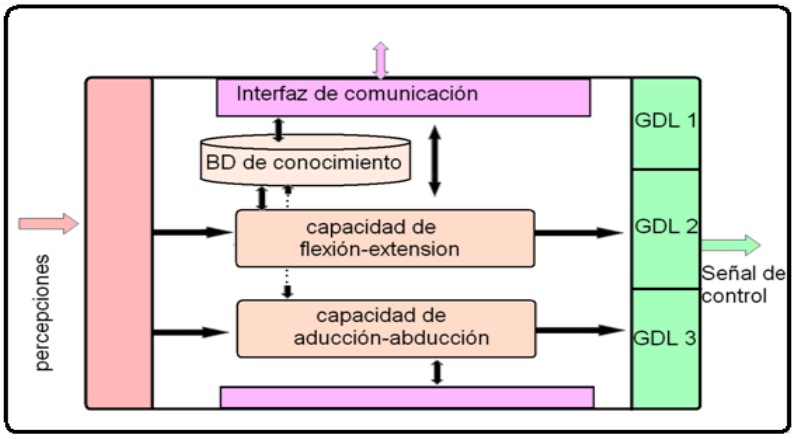


Figura 6.2. Arquitectura interna de los agentes (índice, medio o anular)

6.2 DIAGRAMAS DE VISIÓN GENERAL DE LAS CAPACIDADES

Los diagramas de visión general de las capacidades muestran la estructura interna de cada una de las capacidades de los agentes y al igual que en los diagramas anteriores en estos también se deben detallar los planes o mensajes internos que hacen que cada una de las capacidades cumpla con sus objetivos.

6.2.1 Capacidad de flexión-extensión_ind

En la figura 6.3 se muestra el diagrama general de la capacidad de flexión-extensión del agente índice. En este diagrama; la función del plan *llevar a posición_max_ind* es flexionar los dedos hasta que la punta o falange proximal de los agentes llegue a una distancia umbral antes de contactar el objeto, es decir, hasta el set-point (s-p). Este set-point viene contenido en el mensaje *s-p actualizado.* Cuando el plan logra alcanzar el s-p envía en paralelo dos mensajes;

el primero (*ind_alcanza s-p*) es un aviso hacia el exterior en donde informa de la llega al s-p; el segundo (*parar GDL(1y2)*) es enviado al plan *mantener GDL(1y2)ind* el cual se encarga de mantener la posición actual de GDL(1y2).

La función del plan *llevar a pos_pre_GDL(1y2)ind* es lograr que GDL(1y2) alcancen su correspondiente postura de pre-agarre, cuando lo anterior ocurre se envía el mensaje *parar GDL(1y2)ind* cuya función ya fue explicada.

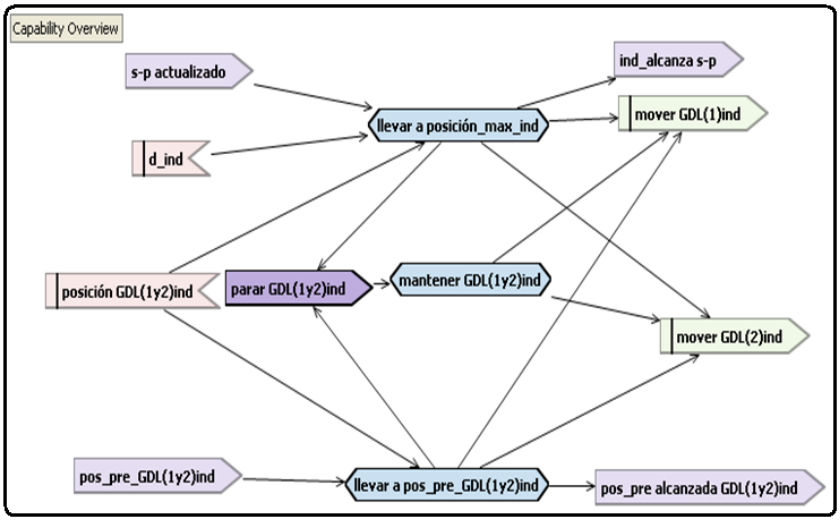


Figura 6.3. Diagrama de visión general de la capacidad de flexión-extensión_ind

6.2.2 Capacidad de aducción-abducción_ind

En la figura 6.4 se muestra el diagrama de visión general de la capacidad de aducción-abducción del agente índice. En este diagrama el plan *llevar GDL(3)ind a* || se encarga de posicionar la superficie de la punta del agente índice lo mas paralela posible a la superficie local del objeto. Para lograr este paralelismo el plan, por medio de los mensajes *posición_GDL(3)_med?* y *posición_GDL(3)_med,* se informa de la posición de GDL(3) del agente medio y así evitar una posible colisión del agente índice con el agente medio. Otra función de este plan es enviar el mensaje *actualización de GDL(3)ind* en el cual informa sobre la posición actual de GDL(3), este mensaje llega al plan *actualizar BD_ind* el cual se encarga de actualizar este nuevo valor de posición en *BD dedo índice,* la anterior es la base de datos interna del agente índice. Los otros planes que parecen en esta diagrama cumplen con las mismas funciones de los planes *pos_pre_GDL(1y2)ind* y *mantener GDL(1y2)ind contenidos en* la capacidad *de flexión-extensión_ind* la única diferencia es que en lugar de establecer acciones sobre GDL(1y2) las acciones son orientadas hacia GDL(3).

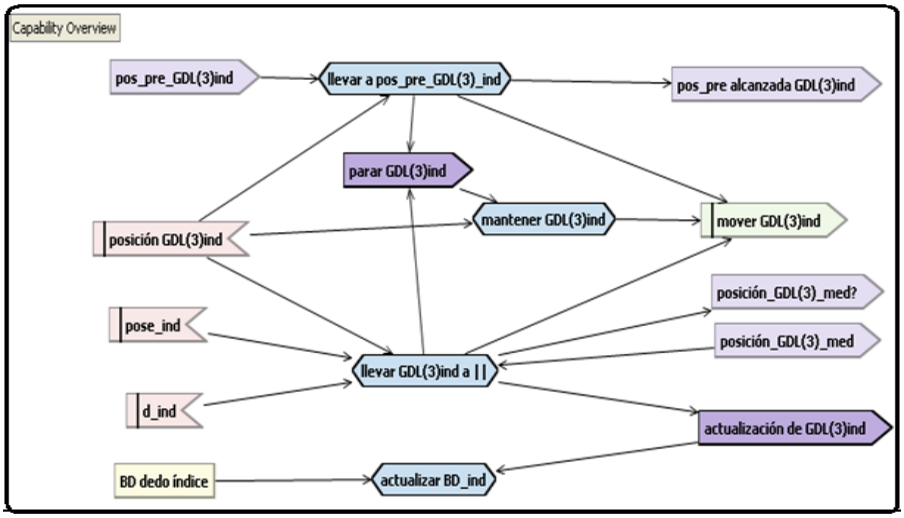


Figura 6.4 Diagrama de visión general de la capacidad de aducción-abducción_ind

6.3 PROCESOS

Durante la etapa de especificación del sistema, se especificaron los escenarios, los cuales son procesos de alto nivel. Estos procesos fueron refinados y especificados completamente usando los diagramas de interacción y los protocolos en la etapa del diseño de la arquitectura. En el diseño detallado, también se necesita de la ayuda de algún mecanismo para especificar más detalladamente este proceso. El mecanismo usado por la metodología Prometheus es una variante de los diagramas de actividad UML. Estos diagramas describen de manera aún más detallada la interacción entre los agentes por medio de mensajes y se desarrollan a partir de los escenarios, los diagramas de interacción y los protocolos.

En la figura 6.5 se ilustran algunas notaciones para la especificación de los procesos según la metodología Prometheus.

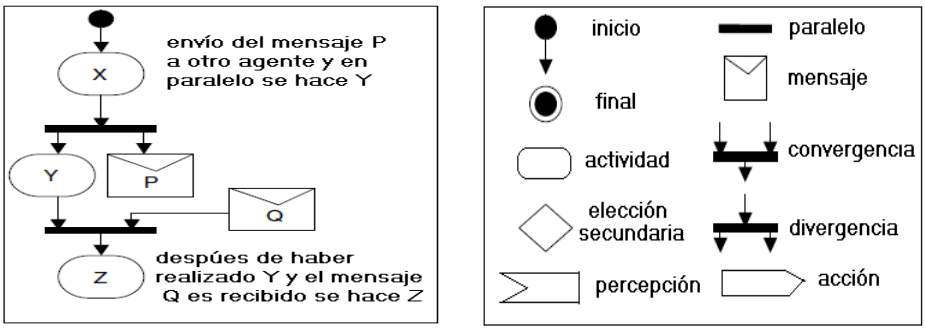


Figura 6.5 Ilustración de algunas notaciones para la especificación de los procesos.

6.3.1 Proceso de configuración del agente índice

La figura 6.6 muestra el proceso de configuración del agente índice. El proceso inicia cuando llega el mensaje *pos_pre_GDL(1y2)ind.* En este mensaje se contiene la posición que deben alcanzar GDL(1y2) para formar su correspondiente postura de pre-agarre. Una vez que este mensaje llega al agente, se intentan alcanzar estas posiciones de pre-agarre, (*llevar a pos_pre_GDL(1y2)ind*). En el momento en que la postura de pre-agarre ha sido alcanzada se deben mantener las posiciones de pre-agarre (*mantener GDL(1y2)ind*) y enviar un mensaje avisando que ya fueron alcanzadas (*pos_pre_alcanzada GDL(1y2)ind*). Para realizar las actividades anteriores, es necesario percibir del ambiente la posición actual de GDL(1y2), (*posición GDL(1y2)ind*) y realizar acciones sobre estos grados de libertad (*mover GDL(1)ind y mover GDL(2)ind*).

En el proceso descrito en la figura 6.6 solo se tienen en cuenta los grados de libertad GDL(1y2), pero este mismo proceso se realiza para GDL(3) y por lo tanto las actividades, mensajes percepciones y acciones van enfocadas hacia este único grado de libertad.

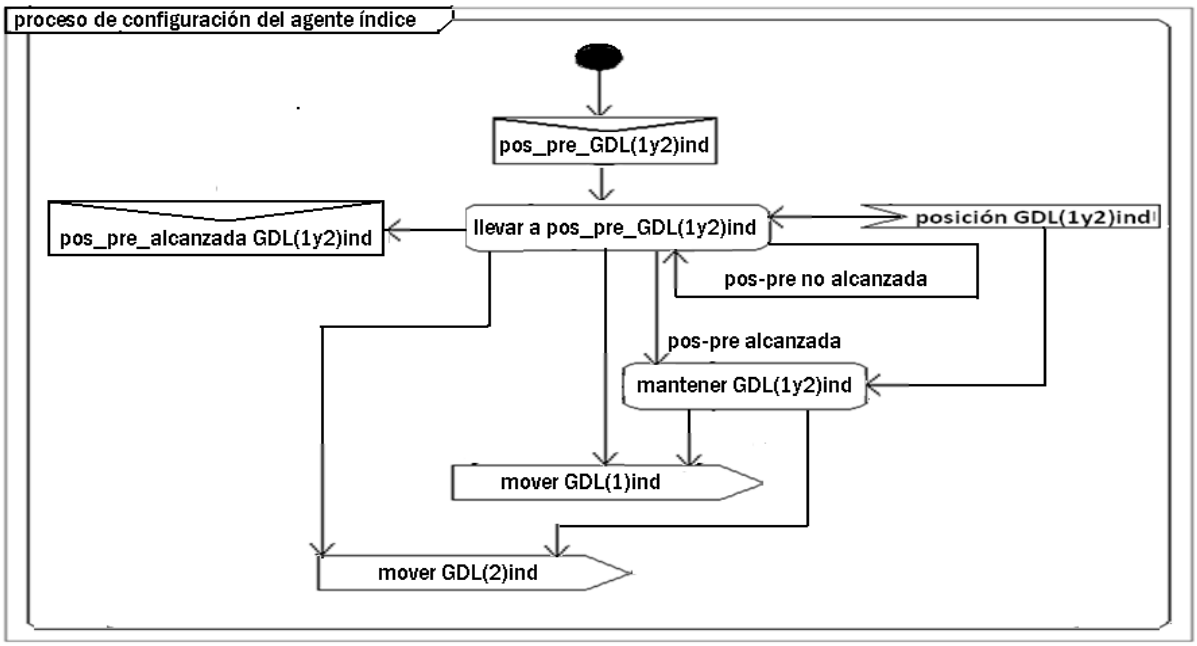


Figura 6.6 proceso de configuración del agente índice.

6.3.2 Proceso de flexión del agente índice

En la figura 6.7 se muestra el proceso de flexión-extensión del agente índice. El inicio de este proceso es la llegada del mensaje *orden_mover_ind* a la actividad *actualizar s-p* la cual es la encargada de actualizar el s-p o distancia a la cual debe llegar el agente con relación a la

superficie del objeto. Si en el mensaje *orden_mover_ind* existe alguna información de distancia, entonces esa distancia corresponde al nuevo s-p que debe alcanzar el agente. Por lo contrario si en el mensaje no existe ninguna información, entonces el s-p es el establecido en 1cm. Una vez actualizado el s-p, se realiza la actividad *llevar a posición_max.* Esta actividad es la encargada de llevar los grados de libertad encargados de la flexión-extensión del agente índice a sus posiciones máximas (mover GDL(1) y GDL(2)), es decir hasta las posiciones establecidas por los *Eigengrasp*. Esta actividad se realiza hasta que la distancia detectada por el agente sea aproximadamente igual al s-p, una vez ocurrido esto se deja de ejecutar la actividad *llevar a posición_max* para enviar el mensaje *ind_alcanza_s-p* y dar inicio a la actividad *mantener GDL(1y2).* En esta actividad los grados de libertad encargados de la flexión-extensión del agente índice (GDL(1y2)) se mantienen en la última posición que tenían antes de que se dar inicio a esta última actividad. Según el protocolo para la flexión la orden de movimiento se envía dos veces, por lo tanto cuando la actividad *actualizar s-p* recibe por segunda vez el mensaje *orden_mover_ind*, el set-point es actualizado a cero, es decir, la distancia de contacto con el objeto.

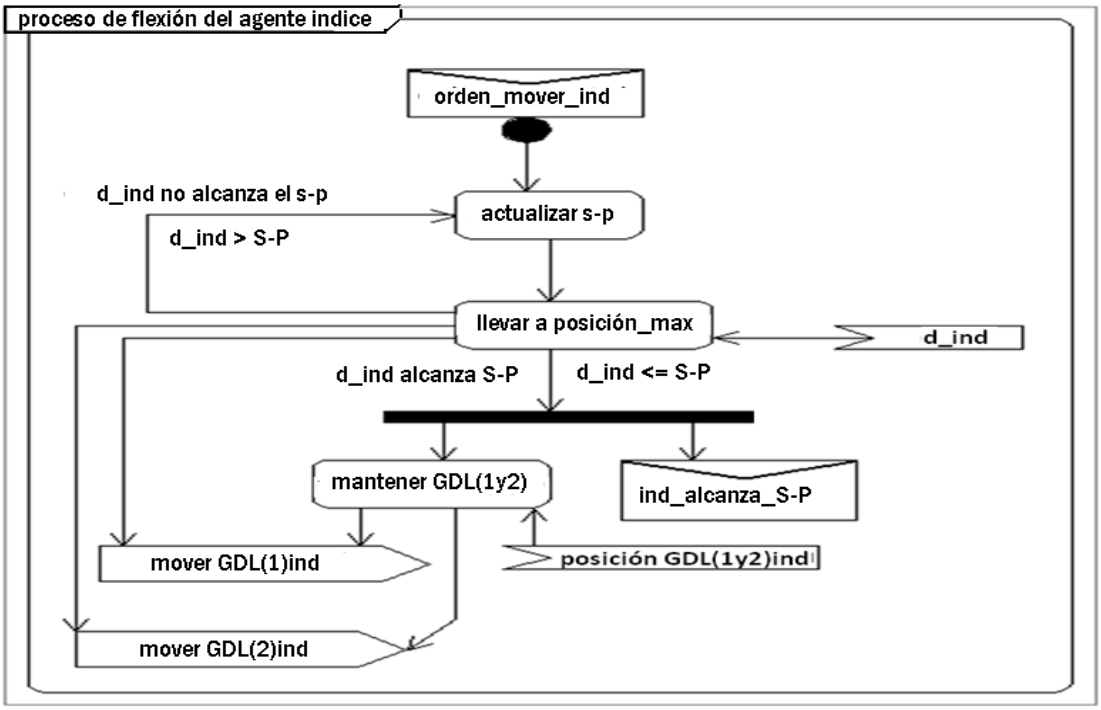


Figura 6.7 Proceso de flexión-extensión del agente índice.

6.3.3 Proceso de aducción-abducción del agente índice.

En la figura 6.8 se muestra el proceso de aducción-abducción del agente índice. El inicio de este proceso es la llegada del mensaje *pos_pre alcanzada GDL(3)ind* que avisa que la postura de pre-agarre del agente índice ya fue alcanzada. Después de la llegada de éste mensaje, se

envía el mensaje *posición GDL(3)med?* que solicita la posición de GDL(3) del agente medio. El agente medio se la envía en el mensaje *posición GDL(3)med* al agente índice. Cuando llega este último mensaje se supervisa que la distancia desde el agente índice al objeto está entre 0.5cm y 3.5 cm, si es así, entonces se realiza la actividad *llevar GDL(3)ind a ||,* que intentará llevar la superficie de la punta del dedo índice lo mas paralela posible a la superficie local del objeto. Si este paralelismo entre superficies todavía no se alcanza, entonces se repite todo el proceso anterior, y se asume que ya llegó el mensaje de inicio. Por el contrario si se logra alcanzar el paralelismo entonces se envía el mensaje *parar GDL(3)ind,* y así, dar paso a la actividad *mantener GDL(3)ind* cuya tarea es mantener la posición actual de GDL(3). Si la distancia desde el agente índice al objeto es mayor a 3.5 cm o menor a 0.5 cm, entonces no se realiza ningún movimiento y solo se debe mantener la posición actual de GDL(3). Las dos actividades; *llevar GDL(3)ind a || y mantener GDL(3)ind,* para llevar a cabo sus tareas deben realizar una acción y percibir del ambiente la información que necesitan tal como se muestra en la figura 6.8.

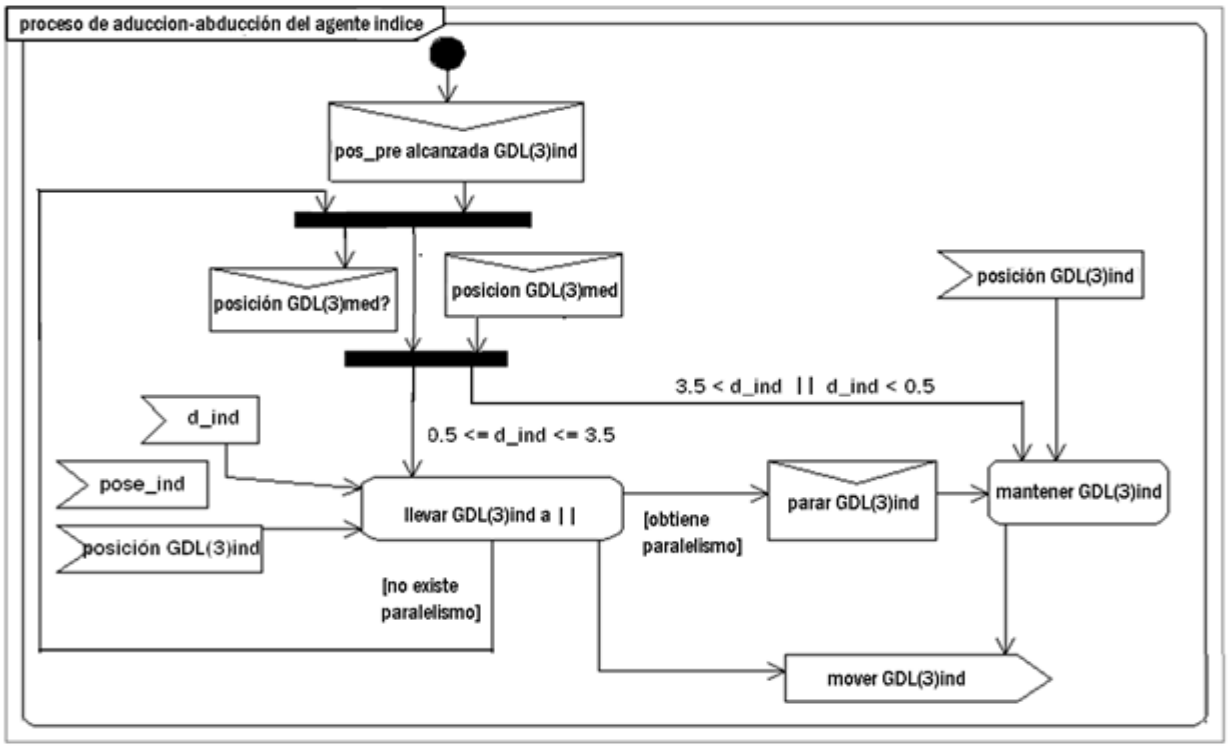


Figura 6.8 Proceso para la aducción-abducción del agente índice

7. CONTROL DE FUERZA EN EL AGARRE

Los objetivos generales que se plantearon en la sección 4.1 ayudan a que una prótesis de mano robótica pueda realizar un agarre estable de un objeto. La puesta en práctica de estos objetivos sin la aplicación de fuerza, arrojaría resultados en los cuales el objeto se desliza, fácilmente, al tratar de levantarlo. Esto no quiere decir que los objetivos planteados no sirvan para realizar un agarre estable, sino como se menciona en los estudios científicos (descritos en la sección 4.1.1), estos objetivos ayudan a aumentar las probabilidades de realizar un agarre estable. Por ejemplo, una de las razones de posicionar la superficie de la punta de los dedos paralela a la superficie local del objeto, es que al aplicar fuerza por parte de los dedos se incremente el área de contacto producida por la deformación de la superficie del dedo. Si no se aplica fuerza, entonces no se genera ninguna área de contacto y por consiguiente el enfoque de paralelismo no tiene sentido.

En la arquitectura de planificación del agarre que se propone en este trabajo no se implementa el control de fuerza, pero esto no quiere decir que se deja de lado ya que como se mencionó anteriormente sin ésta los objetos se deslizan al tratar de levantarlos. Por lo tanto, en este capítulo se estudian las posibilidades de implementar un enfoque de control de fuerza que permita, junto con los objetivos planteados en la sección 4.1, a los agentes levantar el objeto sin que éste se deslice y que, además, puedan resistir disturbios desde cualquier dirección.

La mayoría de los estudios de fuerza en el agarre solo trabajan con el tipo de agarre pinza en el cual solo dos dedos ejercen fuerza sobre un objeto. En este agarre, la fuerza puede ser descompuesta en dos componentes: la fuerza del agarre (componente horizontal) y la fuerza de carga (componente vertical). El criterio para que un agarre de pinza sea estable es simple: la suma de las fuerzas de carga debe ser igual al peso del objeto y la proporción de la fuerza de agarre a la fuerza de carga debe exceder un valor específico, el cual es determinado por el coeficiente de fricción entre el objeto y el dedo, además, se ha demostrado que las personas usan la información táctil para ajustar ligeramente la fuerza de agarre por encima del valor de la fuerza de carga [55]. Cuando la carga es predecible, las personas ajustan el valor de la fuerza de agarre anticipándose a mantener un margen de seguridad constante.

Ahora bien, lo anterior es solo para los tipos de agarre pinza, pero como llevar este criterio de estabilidad, aplicable a los diferentes tipos de agarre en los cuales el agarre se realiza con mas de dos dedos?

En [56] se menciona que la fuerza del dedo pulgar, generalmente va dirigida a oponerse a un *dedo virtual* que está ubicado entre los dedos: índice y medio, esto para el caso de un agarre que incluye tres dedos. Este dedo virtual junto con el pulgar produce un tipo de agarre que puede ser visto como una clase de pinza que realiza un determinado agarre. Este mismo criterio es usado por *Flanagan* y sus colegas [57]. En su trabajo hacen uso del concepto de dedo virtual

y proponen el control de un agarre de tres dedos. El agarre es simplificado como el equivalente a un agarre tipo pinza (pulgar e índice), en el cual el pulgar se opone a un solo dedo virtual en lugar de dos dedos reales independientes. En el ejemplo mostrado en la figura 7.1, dos dedos actúan sobre la misma superficie de un objeto; estos actúan de manera similar y pueden ser representados como un solo dedo virtual.

En [58] se define un espacio de oposición como "*el área dentro las coordenadas de la mano donde las fuerzas de oposición puede ser aplicadas entre la superficie de los dedos virtuales* para *lograr un agarre estable"*. En la figura 7.2 se muestran diferentes espacios de oposición (representados por flechas bidireccionales) generados por 3 diferentes tipos de agarre. En la figura 7.2 a) se muestra el espacio de oposición generado entre el dedo pulgar y la superficie de la punta de los otros dedos. En la figura 7.2 b) se muestra el espacio de oposición generado entre los dedos y la palma de la mano, por último en la figura 7.2 c) se muestra el espacio de oposición generado entre la parte lateral del dedo índice y la superficie palmar del dedo pulgar.

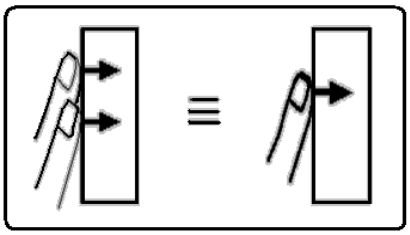


Figura 7.1. Dos dedos y su correspondiente dedo virtual [56]

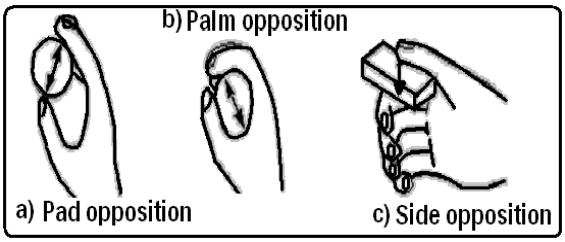


Figura 7.2. Espacios de oposición [58]

El concepto de dedo virtual sugiere que para lograr un agarre estable, es necesario cierto grado de acoplamiento en el control de fuerza ejercida por los dedos virtuales [59]. En [57] se demostró el acoplamiento de fuerzas entre el dedo pulgar y otros dos dedos. La demostración se basa en que para sostener un objeto en el aire, el pulgar ocupaba el 41.4% del de la fuerza vertical total y los otros dos dedos ocupaban, cada uno, el 29.3%. Siendo la fuerza tangencial horizontal igual a cero.

En nuestro enfoque hacemos uso del concepto de dedo virtual desde el momento en que definimos las posturas de pre-agarre ya que el formado de estas posturas incluye el espacio de oposición de acuerdo con los tipos de agarre, los cuales naturalmente contienen un espacio o varios espacios de oposición definidos. Nosotros solo escogemos el espacio de oposición que se puede generar por el contacto entre los dedos, ya que la oposición con la palma se da de manera natural al cerrar los dedos. El espacio de oposición definido por las posturas de pre-agarre es modificado una vez los dedos empiezan a flexionarse. Este proceso de flexión se realiza de tal manera que los dedos contacten simultáneamente el objeto. El contacto simultáneo, establecido en el enfoque de pre-contacto, nos permite representar a los dedos índice, medio y anular como un dedo virtual de una manera mucho más compatible con la hipótesis de dedos virtuales, ya que se permite aplicar fuerzas al mismo tiempo logrando que

estos tres dedos actúen como una unidad y el dedo pulgar como otra unidad, tal y como se hace para el tipo de agarre pinza.

Ahora bien, después de que todos los dedos han contactado el objeto es necesario que las fuerzas que se aplican por parte de cada uno de ellos sea lo más acoplada posible para, lograr así, levantar el objeto de manera estable. Para conseguir este acoplamiento, consideramos a los dedos: índice, medio y anular como un dedo virtual y nos basamos en los resultados obtenidos en [57]. Estos resultados muestran como el dedo pulgar aplica una fuerza aproximadamente igual a la suma de las fuerzas aplicadas por los otros dedos. Si algún dedo no participa en el agarre se debe redistribuir las fuerzas para tratar de conservar el equilibrio de fuerzas.

La cantidad de fuerza que se debe aplicar depende de la tarea que se desea realizar, el peso del objeto y su coeficiente de fricción. Estas tres características definen el valor máximo a configurar en los controladores de cada articulación para lograr un equilibrio de fuerzas horizontales y verticales que conllevan a un agarre estable. Dependiendo del coeficiente de fricción, se puede aumentar o disminuir la fuerza en los contactos de tal manera que se optimice la cantidad de fuerza necesaria para mantener la estabilidad. Existe evidencia de que las señales táctiles de los dedos obtenidas cuando una persona comienza el agarre informan acerca de las condiciones de fricción en las superficies de contacto [60]. Con sensores en una mano robótica que detecten estas condiciones de fricción se puede planificar el aumento o disminución de las fuerzas de los contactos sobre un valor que asegure un margen de estabilidad constante. Debido al alcance del proyecto solo se estudio el control de fuerza en maya abierta, donde no se tiene en cuenta grandes variaciones de peso del objeto o grandes coeficientes de rozamiento.

Se hicieron pruebas en *GraspIt!* para validar los objetivos planteados en la sección 4.1 y la aplicación de la hipótesis de dedo virtual en la aplicación de fuerzas. Para las pruebas se utilizó una mano robótica con las mismas características que fueron establecidas en la sección 2.2.3. Cada uno de los grados de libertad (GDL), fue representado por un motor cuya velocidad es definida por el espacio Eigengrasp. Para los dedos: índice, medio y anular los motores podían establecer un máximo esfuerzo de 1/3 en comparación con el esfuerzo establecido por los motores incrustados en el dedo pulgar. Para controlar los motores de cada dedo se utilizó un controlador PD.

La superficie de la mano robótica y del objeto fue configurada de goma (*rubber*) y los contactos generados como punto-punto. No se le colocaron contactos *soft-finger* debido a que en GraspIt! no se implementa de forma optimizada este tipo de contactos, específicamente en el modo dinámico.

En el inicio de cada prueba la mano robótica fue configurada en una postura tal que la distancia desde cada uno de los dedos al objeto fuera la misma y que la superficie de su punta fuera lo más paralela posible a la superficie local del objeto. Esto se realiza de manera manual ya que el planificador no está implementado. Luego de alcanzar estas condiciones iniciales se activa el

motor de dinámicas que controla la velocidad y fuerza en los dedos. Una vez que los dedos han contacto el objeto, se retira la base que lo sostiene, se toman las medidas de calidad y se observa si el agarre es estable o no.

Para estos experimentos utilizamos la siguiente definición de agarre estable: "*el agarre estable es un movimiento en el que la mano de un robot o de un humano continua sosteniendo un objeto de forma estable*". [61]. Para hacer aún más robusta cada una las pruebas, para cada agarre realizado se introdujo un disturbio. Este disturbio consistía en dejar caer un segundo objeto sobre el agarre actual. Si la mano continuaba agarrando de manera estable el objeto deseado, entonces el agarre se considera estable.

Las pruebas de agarre se realizaron para 5 objetos de diferente forma, para cada uno de ellos se hicieron 10 agarres distintos. En total se efectuaron 50 experimentos.

Para cada uno de los experimentos o pruebas se detalla si el agarre es estable, si es estable entonces se anotan las respectivas medidas de calidad. Los experimentos y resultados de las simulaciones se pueden encontrar en el Anexo D. A continuación solo se mostrara el promedio obtenido para cada una de las medidas de calidad y de agarre estable en los 50 experimentos y se hará un breve análisis de los resultados.

Probabilidad de que la mano agarre estable un objeto: 76%.

Calidad Épsilon promedio para agarres de tipo potencia: 0.1908

Calidad Volumen promedio para agarres de tipo potencia: 1.9227

Calidad Épsilon para el agarre de tipo precisión: 0.0061

Calidad Volumen para el agarre de tipo precisión: 0.0755

Masa promedio de los objetos agarrados: 308 gramos.

Los agarres de precisión son aquellos que produjeron contactos con las puntas de los dedos y los agarres de potencia son aquellos que se realizaron con las puntas, falanges y posiblemente con la palma de la mano.

En los resultados de calidad se puede ver como los agarres de potencia son más estables, mientras que los agarres de precisión tienden a ser inestables por la poca calidad del agarre que tienen. Algunas veces, durante los experimentos, las calidades fueron muy bajas pero igual la mano siguió agarrando el objeto en contra de la gravedad y ciertos disturbios que se le aplicaban. Lo realmente importante es que hemos obtenido un resultado favorable (una probabilidad del 76%). Se observa además que el diseño mecánico y la aplicación de fuerzas de la mano permiten redistribuir la carga a todos los dedos para conservar el equilibrio y lograr la estabilidad.

8. PARÁMETROS DE IMPLEMENTACIÓN DE LA ARQUITECTURA DE PLANIFICACIÓN.

El proceso de diseño de Prometheus resulta en diagramas y descriptores que describen un rango de entidades de diseño como: objetivos, funcionalidades, agentes, capacidades, planes, percepciones, acciones, mensajes, datos, protocolos y escenarios. Este proceso de diseño debe dar paso a la siguiente fase que corresponde a la implementación. Sin embargo, no todas estas entidades de diseño se llevan a la implementación. Por ejemplo, las funcionalidades son usadas para determinar los tipos de agentes pero ellas no corresponden a ninguna entidad en tiempo de ejecución. En términos generales, las entidades que se implementan son las que se producen en la fase de diseño detallado: agentes, capacidades, planes, mensajes y creencias; también como las acciones, percepciones y los objetivos.

En este capítulo se muestran las principales plataformas de agentes que permiten el mapeo directo de las entidades, determinadas por la metodología Prometheus, hacia la implementación, y se establecen los parámetros necesarios para conectar el software de simulación robótico *GraspIt!* con las plataformas de implementación de agentes.

8.1 PLATAFORMAS PARA LA IMPLMENTACIÓN DE AGENTES

Una plataforma de agentes es una arquitectura tecnológica que provee el ambiente en el cual los agentes pueden existir y operar activamente para lograr sus objetivos. La plataforma de agentes puede, adicionalmente, soportar aplicaciones y desarrollos basados en agentes autónomos [65].

En la actualidad existen varias plataformas de agentes, entre las cuales se encuentran: *JADE, JACK, JADEX, LS/TS, ADK, Cougaar, JAM* [65], entre otras. Cuando se considera cual plataforma de agentes se debe usar, es importante tener en cuenta que existen plataformas que se acoplan de una manera más adecuada a unas metodologías que a otras. Por ejemplo, existen plataformas que no contienen las características suficientes para implementar agentes basados en planes, creencias e intenciones como los diseñados con la metodología Prometheus. Es por esta razón que para implementar el sistema de agentes, propuesto en este trabajo, se ha considerado usar las plataformas: *JACK y JADEX* ya que estas plataformas consideran todas las características necesarias de implementación para la metodología Prometheus.

JACK: Es una plataforma basada en el modelo ***BDI*** (creencias-deseos-intenciones). Esta plataforma considera a un agente cuyos planes son accionados por eventos y los mensajes son vistos como un sub-tipo de evento. Un programa en *JACK* consiste de declaraciones de

entidades como: agentes, capacidades, planes, eventos y el conjunto de creencias (también denominadas como base de datos) [41]. Para estas declaraciones se utiliza un leguaje intuitivo que extiende el lenguaje de programación *Java* con ciertas palabras clave que son específicas hacia los agentes.

Muchos de los conceptos de Prometheus mapean directamente a *JACK*, es decir, que un agente que ha sido declarado en Prometheus resulta o se convierte directamente en un agente *JACK*. Por ejemplo, las capacidades de los agentes que habían sido diseñadas en la metodología Prometheus resultan con el mismo nombre y con las mismas características a las capacidades declaradas en *JACK* [41].

JADEX: Es una extensión ***BDI*** para la plataforma de agentes *JADE*, la cual es una plataforma de agentes de distribución libre desarrollada por *TILAB* [66]. *JADE* es ampliamente utilizada en la investigación, así como en proyectos comerciales y tiene usuarios que pertenecen a una comunidad de desarrolladores muy activos.

JADEX al igual que *JADE* permite la programación de agentes software inteligentes en el lenguaje de programación *Java* y permite a los desarrolladores usar conceptos BDI en el proceso de desarrollo de agentes. Además, *JADEX* está influenciada por el sistema comercial y de investigación *JACK*, lo cual permite pasar del diseño de nuestra arquitectura a la implementación de una manera más directa, tal y como sucede con la plataforma *JACK*.

8.2 CONEXIÓN DE *JACK* Y *JADEX* CON *GraspIt!*

Para la implementación de la arquitectura, que se propone en este trabajo, no solamente se debe considerar el uso de una plataforma de diseño de agentes, sino también el ambiente con el cual estos agentes van interactuar, es decir, las percepciones y las acciones. Las plataformas de implementación de sistemas de agentes asumen que las percepciones y acciones ya han sido procesadas y que a los agentes solo llega la información necesaria. Es por esta razón que se debe considerar el uso de un software, adicional, del cual se pueda extraer la información necesaria (percepciones) y ejecutar las respectivas acciones sobre el ambiente (mano robótica).

En el capítulo 1, se hace una análisis del agarre y se muestra como el software *GraspIt!* implementa adecuadamente todas las características esenciales de un agarre, tal como el espacio *Wrench* el agarre y dos medidas de calidad definidas sobre dicho espacio, además, en este software se pueden cargar diferentes modelos de manos robóticas lo cual permite extraer toda la información necesaria y ejecutar diferentes acciones como el movimiento y fuerza en las articulaciones. En el anexo C se muestra el estudio completo del software de simulación GraspIt!

En este trabajo se considera usar a *GraspIt!* como el software más idóneo para simular el ambiente con el cual los agentes van a interactuar.

Para que los agentes puedan tener control sobre una mano robótica y percibir del ambiente la información necesaria se debe establecer una conexión de la plataforma donde "viven (*JACK o JADEX*) y el software GraspIt! Afortunadamente, cuando *GraspIt!* inicia, este inicializa un servidor *TCP* que acepta conexiones con otros programas. Esto permite a programas externos emitir comandos y recolectar datos de simulación. Como ejemplo esta el Programa *MATLAB*, este puede comunicarse con *GraspIt!* haciendo uso de unas funciones implementadas como archivos *MEX* en leguaje *C*. Cada una de estas funciones abre una conexión con el servidor de *GraspIt!,* envía un comando de cambio o solicitud del estado del mundo, y cierra la conexión.

De la misma manera como se comunica *MATLAB* con *GraspIt!* se puede hacer que una plataforma de agentes se comunique con *GraspIt!.* Para esto se debe implementar un "cliente" que acceda al servidor de *GraspIt!* y envié los comandos necesarios para poder controlar la mano y percibir del ambiente al información necesaria. Una consideración que se debe tener en cuenta es que el lenguaje de comunicación entre la plataforma y *GraspIt!* debe ser el mismo. *JACK y JADEX* utilizan el lenguaje *Java* y el servidor de *GraspIt!* el lenguaje C. Así que se deben utilizar *sockets*[9] los cuales permiten conectar un programa *Java* con un programa en C de una manera sencilla y rápida.

En la figura 8.1 se muestra una vista abstracta de cómo debe ser la conexión de *JACK y GraspIt!* utilizando una arquitectura cliente-servidor. En esta figura se muestra a la plataforma *JACK* actuando como cliente y a *GraspIt!* como servidor. *JACK* se comunica con *GraspIt!* para extraer información del mundo de simulación y enviar señales de control a la mano robótica simulada en *GraspIt!* En esta figura también se muestra que *GraspIt!* tiene la capacidad de controlar una mano robótica real, permitiendo así, que una vez probado cualquier algoritmo en *GraspIt!* este pueda ser probado en un mano robótica real**.** Lo anterior es solo para la plataforma *JACK* pero la misma explicación puede ser orientada a la conexión de *JADEX* con *GraspIt!*

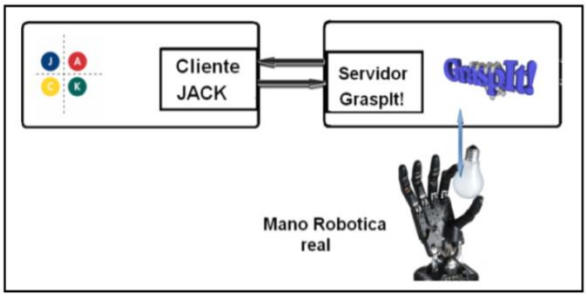


Figura 8.1 Conexión de la plataforma *JACK* con *GraspIt!*

Existe otra forma de conectar *JADEX* con *GraspIt!* mediante la utilización de un software llamado *MACSimJX* (*Multi Agent Control for Simulink with Jade eXtension*) [67]. Este software

[9] Un **socket** no es más que un "canal de comunicación" entre dos programas que corren sobre ordenadores distintos o incluso en el mismo ordenador.

comunica la plataforma *JADE* con el *SIMULINK* de *MATLAB*. *MACSimJX* permite que los agentes desarrollados en *JADE* controlen procesos en el *SIMULINK*. Luego, mediante *SIMULINK* y usando los comandos de interfaz (*MATLAB-Graspit!)* se puede acceder a *GraspIt!* La comunicación de *JADEX* con *GraspIt!,* pasando antes por *MATLAB*, da la posibilidad de que en *MATLAB* se desarrollen complejos cálculos matemáticos entre los cuales pueden estar los utilizados para la optimización de fuerzas en el agarre logran un agarre mas adecuado sobre el objeto.

En la figura 8.2 se muestra una vista abstracta de la conexión de la plataforma *JADEX*, con *MATLAB, GraspIt!* y una mano robótica real. *MATLAB* actúa como cliente y *GraspIt! y JADEX* como servidores. *GraspIt!* ejecuta las señales de control enviadas por *JADEX* (que antes han pasado por *MATLAB*) y envía la información necesaria del ambiente a *MATLAB* para que este les haga un pre-procesamiento y se las envíe a la plataforma *JADEX*, es decir al sistema de agentes. Al igual como se muestra en la figura 8.1, *GraspIt!* se comunica con una mano robótica real para enviar las señales correctas que han resultado en un agarre estable.

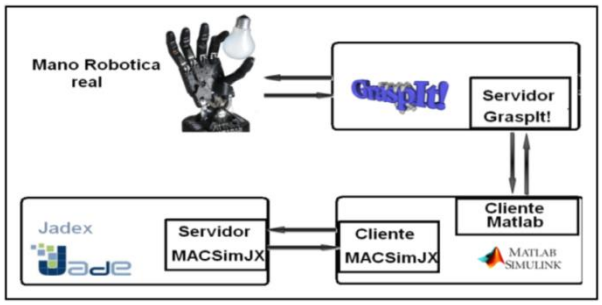


Figura 8.2. Conexión de *JADEX* con *GraspIt!,* pasando antes por *MATLAB*.

Una vez establecida la conexión, ya sea directamente entre *JADEX y GraspIt!* o antes pasando por *MATLAB*, se deben establecer los comandos que serán usados para que la información que se solicita sea efectivamente la información que se envíe, ya sea de cliente a servidor o de servidor a cliente. En *GraspIt!* se establecen comandos que son usados para extraer información del ambiente de simulación o para ejecutar acciones sobre la mano robótica. Estos comandos se nombran a continuación:

- *computeNewVelocities* – calcula la nueva velocidad de cada cuerpo dinámico.
- *getAverageContacts* - retorna la posición y la fuerza de contacto promedio para uno o varios cuerpos.
- *getBodyName* - retorna el nombre de uno o varios cuerpos.
- *getContacts* – retorna todas las posiciones y fuerzas de contacto de uno o varios cuerpos.
- *getDOFVals* – retorna las posiciones actuales de los grados de libertad de uno o varios robots.
- *getRobotName* – retorna el nombre de uno o varios robots.
- *moveDOFs* – mueve cinematicamente los DOFs de un robot a una posición deseada.
- *moveDynamicBodies* – mueve cada cuerpo dinámico secuencialmente o hasta que haya contacto.

- *Render* – instruye a GraspIt! para representar la escena.
- *setDOFForces* – configura las fuerzas que actúan en los grados de libertad de un robot o robots.

Con los comandos anteriores no es posible obtener detalladamente la información del ambiente (percepciones) que nuestro sistema de planificación necesita, pero esto no quiere decir que no se puedan implementar en *GraspIt!*

El bloque de sensado considerado en nuestra arquitectura de planificación (ver figura 5.8) contiene las siguientes percepciones: la posición articular de cada dedo, el tipo de agarre, la orientación de la superficie local del objeto que se encuentra en frente de la punta de cada dedo y la distancia desde la falange proximal y punta de cada dedo al objeto. Estas percepciones pueden ser fácilmente detectadas del mundo de simulación de *GraspIt!*

La posición articular de cada uno de los dedos de la mano robótica puede ser extraída por la plataforma de agentes por medio del comando: *getDOFVals* – el cual retorna las posiciones actuales de los grados de libertad de uno o varios robots del mundo de simulación de *GraspIt!*

El tipo de agarre es un dato que llega una sola vez al sistema, por lo que no es necesario que se implemente de forma directa en *GraspIt!* ya que simplemente se puede enviar manualmente o cargarlo antes en un comando de forma estática, es decir, que se le puede dar un nombre arbitrario a un comando en *GraspIt!* que contenga el tipo de agarre y cada vez que un cliente quiera saber el tipo de agarre simplemente debe enviar el nombre de este comando y automáticamente *GraspIt!* envía la información necesaria.

Las percepciones que corresponden a la orientación y distancia hasta el objeto no son consideradas en los comandos de *GraspIt!,* pero estas pueden ser perpetradas fácilmente gracias al sistema de detección de colisión y contacto implementado en *GraspIt!*

El sistema usado en *GraspIt!* para la detección de colisión es el PQP[10] (*Proximity Query Package*) [68]. En este sistema los objetos son modelados como un conjunto de triángulos. Una vez modelados los objetos el sistema PQP calcula la distancia mínima que existe entre los objetos, es decir, la distancia más corta que existe entre los triángulos de un objeto en relación con los triángulos de otro objeto. Las características anteriores hacen que el sistema PQP sea competente con las percepciones que se consideran en este trabajo, ya que para saber la orientación de la superficie local del objeto y la distancia que existe entre la punta y falange proximal de cada dedo, se deben insertar en el modelo de la mano robótica objetos 3D que simulen la posición y forma de los sensores ópticos. Entonces, la idea es que estos nuevos objetos 3D que simulan los sensores ópticos y el objeto que se desea agarrar sean enviados al sistema PQP. Este sistema se encargará de calcular la distancia mínima entre el objeto y cada

[10] PQP es una librería que implementa un sistema de detección de colisión, el cual utiliza volúmenes basados en barrido de esferas para determinar la distancia entre objetos, así como diversos tipos de volúmenes envolventes que pueden ser seleccionados por el usuario.

uno de los sensores u objetos 3D. Una vez calculada esta distancia el software *GraspIt!* debe enviarlas a la plataforma de agentes (*JACK o JADEX*) que haya sido escogida para el diseño del sistema de planificación del agarre. El envío de estas distancias se hace por medio de un comando que antes ha sido declarado. La declaración de este comando se hace de la misma manera como fueron declarados los comandos anteriores y su acceso o envío depende de las necesidades de información que tenga el cliente, en este caso la plataforma de agentes.

El bloque de actuadores que se considera en la arquitectura propuesta en este trabajo (ver figura 5.8), representa las acciones que deben realizar los agentes sobre el ambiente, estas acciones corresponden al cambio de posición y fuerza en cada una de las articulaciones de la mano robótica. Los comandos que se establecen en *GraspIt!* son suficientes para ejecutar estas acciones. Por ejemplo el comando: *moveDOFs* – mueve cinemáticamente los DOF`s de un robot a una posición deseada y el comando: *setDOFForces* – configura las fuerzas que actúan en los grados de libertad de uno o varios robots.

8.3 DATOS

Las creencias representan el conocimiento del agente acerca de su ambiente y de si mismo. Estas creencias se basan en datos. Estos datos pueden ser representados de varias maneras dependiendo de la plataforma de agentes que se usa. Por ejemplo:

- En *JACK* los datos son representados como objetos que pueden ser simplemente programados en Java. Los datos que son representados como base de datos relacionales pueden ser mapeados en los *beliefsets* de *JACK.* Los *beliefsets* son usados en *JACK* para mantener las creencias del agente acerca del mundo [69].

- En *JADEX* las creencias pueden ser cualquier objeto de Java. Ellas están guardadas en una base de creencia y puede ser accedida ya sea mediante un método de interfaz de la base de creencia o mediante solicitudes formuladas en el *lenguaje de expresión* (*Expression*) de *JADEX* que es muy similar a *OQL-Like Query Languaje* [70].

Para que un agente pueda acceder a su propia base de conocimiento cuenta con planes, que harán las veces de un mecanismo de inferencia de datos. Por ejemplo: dependiendo de la llegada del valor del tipo de agarre existe un plan encargado de sacar datos de la base de datos de acuerdo con el valor que le llegó en ese momento y luego pasar los datos a la correspondiente capacidad del agente (ver capitulo 6).

En resumen los datos pueden ser representados como creencias en la plataforma de implementación de agentes (*JACK o JADEX*) y se puede acceder a ellos mediante planes que seguramente harán uso de las herramientas de acceso de datos que ofrece la plataforma de implementación utilizada.

9. CONCLUSIONES Y TRABAJOS FUTUROS

9.1 CONCLUSIONES

Cuando se trata de diseñar un sistema que va interactuar con un ambiente desconocido o altamente dinámico, se deben considerar los altos márgenes de error que pueden producirse al tratar de poner en práctica dicho sistema. La planificación de agarre estable es uno de estos sistemas, ya que el objeto que se desea agarrar tiene características desconocidas y el ambiente cambia constantemente, por lo tanto en este trabajo los sistemas multi-agente mostraron ser una posible solución a los problemas descritos anteriormente ya que toda la información que se percibe del ambiente (objeto) se divide en entes más pequeños (agentes) los cuales, por medio de mecanismos de coordinación y cooperación, son los encargados no solo de manejar una parte de información, sino también de utilizarla con el fin de lograr unos objetivos específicos de diseño y contribuir así a una posible solución del problema de agarre estable.

En el diseño de sistemas multi-agente existen muchas metodologías, pero no todas ellas tienen una descripción detallada, ya que asumen que el diseñador tiene un alto conocimiento o experiencia en agentes. En este trabajo la metodología Prometheus mostró ser una metodología idónea en el diseño de sistemas multi-agente aplicados a la planificación del agarre en una mano robótica, ya que en su desarrollo, además de ser descrito detalladamente y de forma iterativa, se exponen ejemplos sobre sistemas robóticos y en éstos se especifica que debería y que no debería ser modelado con agentes, sin embargo, en esta metodología no es muy clara la relación que existe entre las capacidades del agente y los procesos realizados por estos. Cuando definimos los planes al interior de las capacidades, se observó que en el desarrollo de la metodología estos planes no tenían una relación muy clara con las actividades descritas en los procesos lo cual hizo más difícil identificar el inicio y/o final de los procesos y las entradas y salidas de los planes.

En la arquitectura mostrada en la figura 5.8 se observa que el diseño de los bloques de sensado y actuadores se hace fuera del paradigma de agentes. En este trabajo se comprobó que era más sencillo y versátil tratar a estos bloques como entes externos, ya que actualmente existen programas o librerías más apropiadas para el diseño y funcionamiento de estos bloques.

En la arquitectura, los dedos: índice, medio, anular y pulgar fueron considerados como agentes. En este trabajo se comprobó que la anatomía y movimiento de los dedos hacían que estos fueran en cierta medida desacoplados unos con otros. Este desacoplamiento hace que cada dedo tenga cierta autonomía para posicionarse de la manera más correcta en relación con el objeto. Si los agentes en lugar de dedos fueran considerados como una agrupación de dedos entonces al interior de cada agente deberían existir entes más pequeños que controlen

aspectos separados por cada dedo, logrando con ello una pérdida de cohesión al interior del agente.

El contacto simultáneo permite aplicar, de una manera más efectiva, la hipótesis de los dedos virtuales para el agarre. En este trabajo se combinaron los enfoques de contacto simultáneo y dedo virtual y se probó que estos son muy acoplados entre sí. Este acoplamiento se debe a que cuando los dedos se flexionan de tal manera que la distancia desde cada uno de ellos al objeto sea la misma, se logra que los dedos: índice, medio y anular sean considerados, efectivamente, un solo dedo virtual.

GraspIt! es un software creado para la investigación del agarre robótico y para el desarrollo de este trabajo mostró ser una herramienta esencial para comprobar los enfoques considerados en la arquitectura de planificación, sin embargo, este software tiene algunas fallas. En este trabajo la mano robótica está constituida de un material deformable, este material genera contactos *soft-finger,* al hacer simulaciones con este tipo de contactos GraspIt! generó problemas en los cálculos matemáticos en el modo dinámico.

9.2 TRABAJOS FUTUROS

En el capítulo 7, se propone un enfoque para la aplicación de fuerzas, que se basa en la hipótesis de dedo virtual. Queda para un trabajo futuro el control realimentado de fuerzas que sea capaz de optimizar la fuerza necesaria para realizar el agarre estable de diferentes objetos, y que además este enfoque de aplicación de fuerzas, pueda ser implementado en la arquitectura de planificación que se propone en este trabajo.

.

En este trabajo solo se consideró el diseño de una arquitectura basada en sistemas multi-agentes para la planificación de agarre estable pero queda para un trabajo futuro su implementación en el software de simulación *GraspIt!* ya que se podrían evidenciar posibles mejoras al enfoque propuesto.

En el desarrollo del proyecto se consideró tratar a los dedos como agentes, pero dadas las ventajas que éstos ofrecen, queda abierta la posibilidad de tratar al paciente como otro “agente” al cual se le informe (según la posición de la mano) si el agarre no será exitoso y que por lo tanto se debe re-posicionar la mano.

BIBLIOGRAFÍA

[1] A. Miller, S. Knoop, H. Christensen and P. Allen, “Automatic grasp planning using shape primitives,” in *Proceedings of the International Conference on Robotics and Automation IEEE*, 2003, pp. 1824–1829.

[2] Hanafusa, H., Asada, H, "Stable Prehension By a Robot Hand With Elastic Fingers," in *Proc. of 5th intern Syrup,* Brady et al Ed. Tokyo: Brady et al, 1977.

[3] N. Nguyen, “Constructing force-closure grasps”, *IEEE Int. Journal of Robotics Research*, vol 7, No.3, pp. 345-362, 1988.

[4] D. Berenson, R. Diankov, K. Nishiwaki, S. Kagami, and J. Kuffner, “Grasp planning in complex scenes,” in Proceedings of IEEE-RAS International Conference on Humanoid Robots (Humanoids), 2007.

[5] Y. Li and N. S. Pollard, "A Shape Matching Algorithm for Synthesizing Humanlike Enveloping Grasps," in *Proceedings of the 2005 IEEE-RAS International Conference on Humanoid Robots*, Tsukuba, Japan, 2005.

[6] C. Goldfeder, P. Allen, L. Claire, R. Pelossof, “Grasp Planning via Decomposition Trees,” In *IEEE Int. Conference on Robotics and Automation.* Rome: IEEE ICRA, 2007.

[7] A. Saxena, J. Driemeyer and A. Ng., “Robotic Grasping of Novel Objects using Vision,” investigation dissertation, Computer Science Department, Stanford University, Stanford, 2008.

[8] E. Torres, “Sensitive manipulation,” PhD. Thesis, Massachusetts institute of technology, Massachusetts, 2007.

[9] R. Platt, R. Grupen and A. Fagg, “Improving Grasp Skills Using Schema Structured Learning,” .In *Fifth International Conference on Development and Learning*, Bloomington, Indiana, 2006

[10] J. Coelho, and R.A. Grupen. “A control basis for learning multifingered Grasps,” *Journal of Robotic Systems*, pp. 545-557, 1997.

[11] K. Hsiao, P. Nangeroni, M. Huber, A. Saxena, and A. Y. Ng, “Reactive grasping using optical proximity sensors,” in *Conference on Robotics and Automation (ICRA)*, Kobe, Japan, 2009.

[12] M. Wooldridge and N. Jennings, *Intelligent agents: theory and practice.* Department of Computing Manchester Metropolitan University: Cambridge University Press, 1995.

[13] R. Suarez, M. Roa and J. Cornella, "Grasp quality measures," Universitat Politecnica de Catalunya, Institut d'Organitzacio i Control de Sistemes Industrials, Tech. Rep. IOC-DT-P, 2006.

[14] A. Vivas and E. Aguilar, “Modelado geométrico y dinámico de una prótesis de mano” en *III IEEE Colombian Workshop on Robotics and Automation*, Cartagena, Colombia, 2007.

[15] A. Miller, “Graspit!: A Versatile Simulator for Robotic Grasping," PhD. Thesis, Columbia University, Columbia (USA), 2001.

[16] J. Pang and J. Trinkle, “Stability Characterizations of Fixtured Rigid Bodies with Coulomb Friction,” in *IEEE International Conference on Robotics and Automation*, San Francisco, CA, 2000, pp. 361-368.

[17] M. Ciocarlie, C. Goldfeder, and P. Allen, "Dimensionality reduction for hand-independent dexterous robotic grasping," in *Proc. of the IEEE/RSJ Conf. on Intelligent Robots and Systems*, 2007, pp. 3270- 3275.

[18] N. Pollard, “Parallel Methods for Synthesizing Whole-Hand Grasps from Generalized Prototypes,” PhD thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Massachusetts, 1994.

[19] C. Ferrari and J. Canny, “Planning optimal grasps” In *Proc. of the 1992 IEEE Intl. Conf. on Robotics and Automation*, 1992, pp. 2290–2295.

[20] D. Lide, *Handbook of Chemistry and Physics*, 75th Ed. CRC Press, 1995. pp. 15 - 40 a 42. University of Liverpool, UK: John Wiley & Sons LTD, 2002.

[21] C. Barber, D. P. Dobkin and H. Huhdanpaa, “The Quick hull algorithm for convex hulls”, Univ. of Minnesota, Minneapolis, Tech. Rep. GCG53, 1993.

[22] D.G Kirkpatrick, B. Mishra and C.K Yap, “Quantitative Steinitz's theorems with applications to multi_fingered grasping”. In *Proceedings 20th Symposium on Theory of Computing*, Baltimore, Maryland (USA), pp. 341-351, 1990.

[23] Z. Li and S. Sastry. “Task-oriented optimal grasping by multi fingered robot hand,” IEEE Journal of Robotics and Automation, vol. 4, pp. 32–44, Febrero 1988.

[24] E. López, D. Sidobre y R. Alami, “Grasp planning for non-convex objects,” in *Proc. 36th International Symposium on Robotics (ISR*)., Tokyo, Japan, November 2005.

[25] C. Gaviria, J. Diaz y V. Mosquera, “Agarre estable de objetos con una prótesis de mano robótica,” Tech. Rep. Resultados de investigación, Proyecto Prótesis Robótica de Mano, Universidad del Cauca, Popayán, 2008.

[26] J. Napier. “The prehensile movements of the human hand,”.*Journal of Bone and Joint Surgery*, Vol.38B, No.4, pp 902–913, 1956.

[27] G. Schlesinger**,** *Der Mechanische Aufbau der Kunstlichen Glider*. Berlin: Ersatzglider und Arbeitshilfen, M. Borchardt et al.,1919.

[28] C. Peggy, “Grasp Types Recognition, A Cyber Glove Project,” [online]. Available: http://cg.cis.upenn.edu/hms/research/RIVET/graspTypeRecog.pdf. [Accessed Enero. 12, 2010].

[29] J. Cabrera, “Conjunto normalizado de Interfaces para los componentes de un simulador virtual de Prótesis de mano,” Tesis de maestría, Universidad del Cauca, Colombia, 2009.

[30] B. Norman, B. Cary, L. Bonnie, *Simulating humans: computer graphics, animation, and control*. Philadelphia: Department of Computer and Information Science, University of Pennsylvania, 1999.

[31] M. Zecca, S. Micera, M. Carrozza and P. Dario, “Control of Multifunctional Prosthetic Hands by Processing the Electromyographic Signal,” investigation dissertation, ARTS Lab, Scuola Superiore Sant’Anna, Pontedera, Italy, 2002.

[32] M. Huber and R. Grupen, “Learning to coordinate controllers – reinforcement learning on a control basis,”. in *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence*, Nagoya, Japan, 1997.

[33] R. Wistort and J. Smith, “Electric field servoing for robotic manipulation,” In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2008)*, Nice, France, 2008.

[34] B. Mayton, E. Garcia, L. Legrand, J. Smith, “Electric Field Pretouch: Towards Mobile Manipulation,”.in *Robotics Science and systems*, 2009.

[35] L. Overgaard, N. Bradley and K. Pradeep, “A Multi-Agent Framework For Grasping Using Visual Servoing and Collision Avoidance,” in *IEEE International Conference on Robotics and Automation,* Minneapolis, Minnesota, 1996.

[36] T. Matsui, T. Omata and Y. Kuniyoshi, “Multi-Agent Architecture for Controlling a Multi-Fingered Robot,” in *Proc. IEEE Conf. on Intelligent Robots and Systems*, 1992, pp. 182-186.

[37] F. Ruini, “Communication and Distributed Control in Multi-Agent Systems: Preliminary Model of Micro-unmanned Aerial Vehicle (MAV) Swarms,” investigation dissertation, Adaptive Behaviour and Cognition Research Group School of Computing, Communications and Electronics, University of Plymouth, Uk, 2008.

[38] Y. Li, I. Kao, “A Review of Modeling of Soft-Contact fingers and Stiffness Control for Dexterous Manipulation in Robotics,” In *Proc. IEEE Conference on Robotic and Automation,* Seoul, Korea, pp. 3055-3060, 2001.

[39] H. Rouviére y A. Delmas, “Anatomía Humana: Descriptiva, Topográfica y Funcional,” Ph.D. dissertation, faculté de médecine Necker-Enfants Malades, Université René Descartes, Francia, 2005, pp. 304-312.

[40] L. Ascari, U. Bertocchi, P. Corradi, C. Laschi and D. Paolo, “Bio-inspired grasp control in a robotic hand with massive sensorial input,” Biological Cybernetics (submitted), 2007

[41] L. Padgham and M. Winikoff, *Developing Intelligent Agent Systems: A practical guide*. Australia, RMIT University: John Wiley & Sons, Ltd, 2004.

[42] S. Franklin, A. Graesser, “*Is it an Agent, or Just a Program? A Taxonomy for Autonomous Agents,”* Third International Workshop on Agent Theories, Architectures, and Languages, Springer-Verlag, 1996

[43] M. Wooldridge, *An Introduction to MultiAgent Systems*. Department of Computer Science, *University of Liverpool, UK: John Wiley & Sons, LTD,2002.*

[44] M. E. Bratman, *Intentions, Plans, and Practical Reason*. Massachusets: Harvard University Press, Cambridge. MA, 1987.

[45] A. Rao and MP. Georgeff, “An abstract architecture for rational agents,” In *Proceedings of the 3rd International Conference on Principles of Knowledge Representation and Reasoning*, San Mateo, CA. Cambridge, 1992.

[46] J. Ferber, *Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence,* New York: Addison-Wesley, 1999.

[47] L. Qingguo and P. Shahram, *Distributed Manipulation Systems: A Multi-agent Systems Perspective.* Canada: Experimental Robotics Laboratory, School of Engineering Science, Simon Fraser University Burnaby, 2003.

[48] C. González, “*Contribuciones Al Diseño De Sistemas Tutores Inteligentes Usando Razonamiento Basado En Casos,”* Tesis Doctoral, Universidad De Vigo, Brasil, 2008.

[49] R. Ramón, L. Faraón y P. Mar, *Arquitecturas y comunicación entre agentes.* España: Universidad de Alicante, Dpto. de ciencia de la computación e inteligencia artificial, 2001.

[50] W. Brenner, R. Zarnekow, and H. Witting, *Intelligent Software agents: Foundations and applications*. Alemania: Springer-Verlag, 1998, pp. 245-257.

[51] F. Cerón y J. P. Torres, “Modelado mediante sistemas multi-agente del ámbito funcional PPC del modelo CIM de Siemens,”. Tesis de pre-grado, Universidad Del Cauca, Colombia, 2008.

[52] MP. Huget, J. Odell, N. Haugen, S. Cranefield, R. Levy and L. Padgham, “FIPA modeling: interaction diagrams,” *www.auml.org under,* working documents. [online]. Available: http://www.auml.org/auml/documents/ID-03-07-02.pdf. [Accessed Sept. 5, 2010].

[53] J. R. Smith, E. Garcia, R. Wistort, and G. Krishnamoorthy, “Electricfield imaging pretouch for robotic graspers,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2007*), San Diego, California, 2007.

[54] M. Santello, M. Flanders and J. Soechting. “Postural hand synergies for tool use*,” Journal of Neuroscience*, vol. 18, no. 23, pp. 10 105–10 115, 1998.

[55] RS. Johansson, KJ. Cole. “Grasp stability during manipulative actions,” Canadian Journal of physiology and pharmacology, pp. 24-511, 1994.

[56] M. Arbib, T. Iberall, and D. Lyons, *Coordinated control programs for control of the hands*, Eds., In A. W. Goodwin and I. Darian-Smith. Berlin: Springer-Verlag, 1985.

[57] J. Flanagan, M. Bursted, And R. Johansson, *Control of fingertip forces in multidigit manipulation*. Ontario, Canada: *Queen’s University,* 1999.

[58] T. Iberall, G. Bingham and M.A Arbib, “Opposition space as a structuring concept for the analysis of skilled hand movements,” In *Generation and Modulation of Action Pattern,* Berlin, 1986.

[59] M. Gentilucci, L. Caselli and C. Secchi, *Finger control in the tripod grasp*. Parma: Instituto de Fisiologia Humana, 2003.

[60] K. Magnus, J. Flanagan and R. Johansson, *Control Of Grasp Stability In Humans Under Different Frictional. Conditions During Multi-digit Manipulation.* Ontario, Canada: Department of Psychology, Queen's University, 1999.

[61] I. kazuto and M. Takashi, “Hierarchical Control Method for Manipulating/Grasping Tasks using Multi-fingered Robot Hand,” In *Conference on Intelligent Robots and Systems*, Las Vegas, Nevada , 2003.

[62] G. Vass, “Object manipulation planning for dextrous robot systems,*”* PhD. Thesis, Budapest university of technology and economics, Budapest, Hungary, 2005.

[63] C. Baysa and L. Üngör, “A Fuzzy Inference System For Preshape Determination Of Robot Hand Grasping,” investigation dissertation, Electrical-Electronics Engineering Department, Kayseri, Türkiye, 2007.

[64] J. Wen, H. Xing, X. Luo and J. Yan, "Multi-agent Based Distributed Control System for an Intelligent Robot," Services Computing IEEE International Conference on (SCC'04), pp. 633-637, 2004.

[65] R. Leszyna, *Evaluation of agent platforms*. Italy: Institute for the protection and security of the Citizen, Ispra, 2004.

[66] F. Bellifemine, G. Rimassa, and A. Poggi, “JADE - A FIPA-Compliant Agent Framework,” In *Proceedings of the 4th International Conference and Exhibition on the Practical Application of Intelligent Agents and Multi-Agents*, UK, 1999.

[67] R. Charles, *Decentralized Data Fusion Using Agents*. New York: University of New York, Department of Electronics, 2008.

[68] E. Larsen, S. Gottschalk, M. Lin and D. Manocha, “Fast proximity queries with swept sphere volumes,” Dept. of Computer Science, University of North Carolina, Tech. Rep TR99-018, 1999.

[69] JACK autonomous software, *JACK Intelligent Agents Manual,* Agent Oriented Software Pty. Ltd, 2009.

[70] A. Pokahr, L. Braubach, *JADEX: User Guide. Distributed Systems Group.* Germany: University of Hamburg, 2005.