INSTITUTO POLITECNICO NACIONAL

UNIDAD PROFESIONAL INTERDISCIPLINARIA EN INGENIERIA Y TECNOLOGIAS AVANZADAS

Diseño de un esquema de control servo visual para robots 3D basado en potenciales artificiales para la intercepción de múltiples objetos móviles

Tesis
Que para obtener el grado de:
Maestro en Tecnología Avanzada

Presenta:

Ing. Julio Alberto Mendoza Mendoza

Directores de Tesis
Dr. Gabriel Sepúlveda Cervantes
Dr. Emmanuel Carlos Deán León

México D.F Julio del 2011

SIP-13

INSTITUTO POLITÉCNICO NACIONAL
SECRETARIA DE INVESTIGACIÓN Y POSGRADO

ACTA DE REGISTRO DE TEMA DE TESIS
Y DESIGNACIÓN DE DIRECTOR DE TESIS

México, D.F. a 30 de Octubre del 2009

El Colegio de Profesores de Estudios de Posgrado e Investigación de U.P.I.I.T.A. en su sesión Extraordinaria No. 16 celebrada el día 28 del mes de octubre conoció la solicitud presentada por el(la) alumno(a):

Mendoza	Mendoza	Julio Alberto
Apellido paterno	Apellido materno	Nombre (s)

Con registro: A 0 9 0 0 1 5

Aspirante de: MAESTRIA EN TECNOLOGIA AVANZADA

1.- Se designa al aspirante el tema de tesis titulado:
"Diseño de un esquema de control servo visual para robots 3D basado en potenciales artificiales para la intercepción de múltiples objetos móviles"

De manera general el tema abarcará los siguientes aspectos:
Se describe una metodología teórico – experimental para interceptar objetos móviles utilizando algoritmos de control servo visual en el dominio de pares y fuerzas con base en campos de potenciales artificiales aplicados a robots tridimensionales empleado observadores de estado para el seguimiento

2.- Se designa como Director de Tesis al C. Profesor: Dr. Emmanuel Carlos Dean Leon y Codirector Dr. Gabriel Sepúlveda Cervantes.

3.- El trabajo de investigación base para el desarrollo de la tesis será elaborado por el alumno en: IPN
que cuenta con los recursos e infraestructura necesarios.

4.- El interesado deberá asistir a los seminarios desarrollados en el área de adscripción del trabajo desde la fecha en que se suscribe la presente hasta la aceptación de la tesis por la Comisión Revisora correspondiente:

El Director de Tesis

p.a.
Dr. Emmanuel Carlos Dean Leon

El Codirector de Tesis

Dr. Gabriel Sepúlveda Cervantes

El Aspirante

Julio Alberto Mendoza Mendoza

El Presidente del Colegio

M. en C. Griselda Sánchez Ortega

S. E. P.
INSTITUTO POLITECNICO NACIONAL
UNIDAD PROFESIONAL INTERDISCIPLINARIA EN INGENIERIA Y TECNOLOGIAS AVANZADAS.
SECCION DE ESTUDIOS DE POSGRADO E INVESTIGACION

SIP-14

INSTITUTO POLITÉCNICO NACIONAL

SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

ACTA DE REVISIÓN DE TESIS

En la Ciudad de México, D.F. siendo las 17:00hrs horas del día 23 del mes de junio del 2011 se reunieron los miembros de la Comisión Revisora de Tesis, designada por el Colegio de Profesores de Estudios de Posgrado e Investigación de SEPI-UPIITA para examinar la tesis titulada:

Diseño de un esquema de control servo visual para robots 3D basado en potenciales artificiales para la intercepción de múltiples objetos móviles.

Presentada por el alumno:

MENDOZA	MENDOZA	JULIO ALBERTO
Apellido paterno	Apellido materno	Nombre(s)

Con registro:

A	0	9	0	0	1	5

aspirante de:

MAESTRÍA EN TECNOLOGÍA AVANZADA

Después de intercambiar opiniones, los miembros de la Comisión manifestaron ***APROBAR LA DEFENSA DE LA TESIS***, en virtud de que satisface los requisitos señalados por las disposiciones reglamentarias vigentes.

LA COMISIÓN REVISORA

Director(a) de tesis

DR. GABRIEL SEPÚLVEDA CERVANTES

1er VOCAL

DR. PRIMO ALBERTO CALVA CHAVARRÍA
PRESIDENTE

DRA. LAURA NADONE GARAY JIMÉNEZ
SECRETARIO

DR. JOSÉ LUIS CASAS ESPÍNOLA
2° VOCAL

DRA. SARA GUADALUPE CRUZ Y CRUZ
VOCAL

PRESIDENTE DEL COLEGIO DE PROFESORES

M. EN C. GRISELDA SÁNCHEZ OTERO

S. E. P.
INSTITUTO POLITECNICO NACIONAL
UNIDAD PROFESIONAL INTERDISCIPLINARIA
EN INGENIERIA Y TECNOLOGIAS AVANZADAS.
SECCION DE ESTUDIOS DE
POSGRADO E INVESTIGACION

Agradecimientos

A las ciencias

A las Artes

A la nación Mexicana y el CONACYT

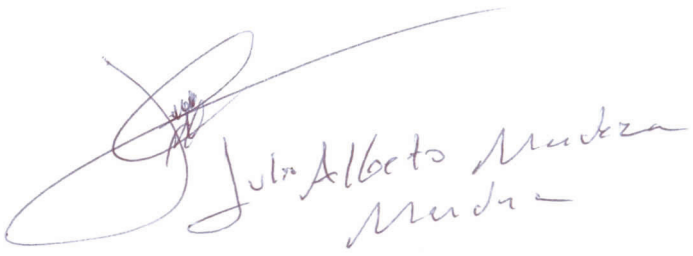
Al Dr. Boy y la Dra Sara

A MI ABUE

CARTA DE SESION DE DERECHOS

En la ciudad de México el día 29 del mes de Julio de 2011, el que suscribe Julio Alberto Mendoza Mendoza alumno del Programa de Maestría en Tecnología Avanzada del Instituto Politécnico Nacional con número de registro A090015, adscrito a la Unidad Profesional Interdisciplinaria en Ingeniería y Tecnologías Avanzadas manifiesta que es autor intelectual del presente trabajo de tesis bajo la dirección del Dr. Gabriel Sepúlveda Cervantes y el Dr. Carlos Emmanuel Dean León y cede los derechos del trabajo titulado: "Diseño de un esquema de control servo visual para robots 3D basado en potenciales artificiales para la intercepción de múltiples objetos móviles", al Instituto Politécnico Nacional para su difusión, con fines académicos y de investigación.

Los usuarios de la información no deben reproducir el contenido textual, graficas o datos sin el permiso expreso del autor o directores del trabajo. Este puede ser obtenido escribiendo a: jamendozam0300@ipn.mx, si el permiso se otorga, el usuario deberá dar el agradecimiento correspondiente y citar la fuente del mismo.



Resumen

Se describe y desarrolla una metodología teórico-experimental para interceptar/esquivar objetos móviles utilizando algoritmos de control servo visual aplicados a robots manipuladores 3D

La técnica descrita se realiza en el dominio de pares y fuerzas cartesianos y se basa en visión estereoscópica para la detección, filtro de Kalman para la predicción y campos de potenciales artificiales para la evasión y/o captura.

Intercepción:

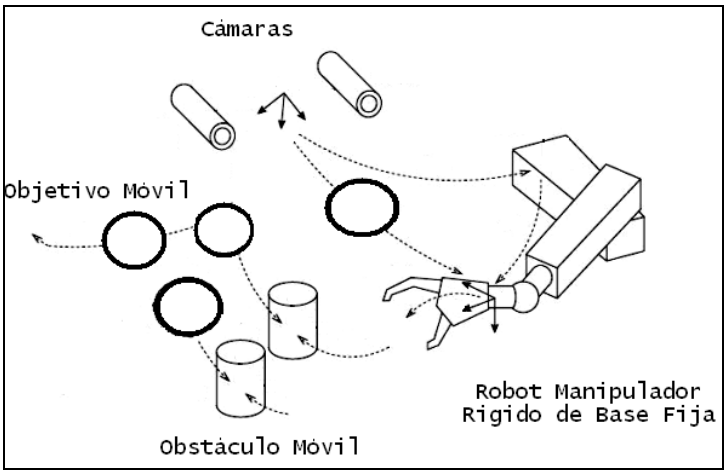
Ubicación (Visión Estero)	→	Predicción (Filtro de Kalman)	→	Captura (Potenciales Artificiales)

Evasión:

Ubicación (Visión Estereo)	→	Acción Evasiva (Potencial Artificial)

Palabras clave

Servovisual, Potenciales Artificiales, Robot Manipulador, Visión Estereo, Filtro de Kalman, Evasión, Intersección, Objetos Móviles



Abstract

It is described and developed an experimental-theoretical methodology for avoiding/intercepting movil objects by using servovisual control algorithms in 3D robotic manipulators

The technique takes place in the cartesian force domain and is based on stereovision for detection, kalman filter for prediction and artificial potencial fields for evasion and capture

Interception:

Tracking → Prediction → Capture
(Stereovision) (Kalman Filter) (Artificial Potential Fields)

Evasion:

Tracking → Elusive action
(Stereovison) (Artificial Potential Fields)

Keywords

Servovisual, Artificial potencial fields, Robotic Manipulator, Stereovision, Kalman Filter, Evasion, Capture, Movil Objects

Indice General

Índice de Figuras

Índice de Ecuaciones

Ecuación 37: Ecuación de Fuerzas asociada a la técnica del potencial artificial de flujo laminar

Ecuaciones 38 a 40: Matrices de transformación homogénea básicas y forma de obtener algunas mas que se necesiten

Ecuación 41: Definición del Jacobiano

Ecuaciones 42: Desacople del Jacobiano.

Ecuaciones 43 a 46: Generación del Jacobiano del robot a partir de las matrices de transformación homogénea, ejemplo efector final y Cm 22

Ecuaciones 47 a 48: Jacobiano y Cinemática Directa del Efector final

Ecuación 49: desplazamiento virtual.
Ecuación 50 a 52: trabajo virtual y deduccion del jacobiano transpuesto.

Ecuación 53: Lagrangiano del sistema.

Ecuación 54 a 55: Deduccion de la ecuación dinamica de robots.

Ecuación 56: Matriz de inercias

Ecuación 57: Matriz de Coriolis

Ecuación 58: Propiedad de antisimetria

Ecuación 59 a 61: Obtención del Vector de pares gravitacionales

Ecuación 62: Ecuación dinamica de robots manipuladores con friccion

Ecuación 63: Torque real de un robot para realizar tareas de intercepción y evasión de objetos

Ecuación 64: Torque simplificado del robot para realizar tareas de intercepción y evasión de objetos

Ecuación 65: propiedades de las funciones potenciales a utilizarse para la demostración de estabilidad

Ecuación 66: Familia de controladores por potenciales artificiales

Ecuación 67: Función potencial del diferencial de error

Ecuación 68: Campo de fuerzas de (67)

Índice de Tablas

Glosario:

Control Servovisual: Es aquel que utiliza información extraída de un sensor visual para cerrar el lazo de control en un sistema actuado mecánicamente.

Grado de Libertad: La cantidad mínima de estados para describir completamente la cinemática y dinámica de un sistema.

Estados: Variables propias de la dinámica y cinemática de un sistema (por ejemplo: posición, velocidad, aceleración, corriente, voltaje, etc).

Parámetros: Variables propias del sistema y en general invariantes ante su dinámica o cinemática (por ejemplo masa, longitudes, inercias, viscosidad, resistencia eléctrica, color, etc).

Espacio cartesiano: Espacio Euclidiano de dimensión 3, tiene 6 grados de libertad simples (3 traslaciones y 3 rotaciones).

Espacio articular: Espacio de dimensión n, tiene tantos grados de libertad como articulaciones tenga el robot, el mapeo de este espacio al espacio cartesiano se realiza con una transformación Jacobiana.

Espacio cámara: Espacio Euclidiano ante ciertas condiciones de corrección focal de dimensión 2 (posición vertical y horizontal), se mide directamente en píxeles o se convierte a unidades dimensionales estandarizadas (metros, pulgadas, etc).

Cinemática Directa: Mapeo del espacio articular al cartesiano.

Cinemática Inversa: Mapeo del espacio cartesiano al articular.

Observador: Cualquier ente capaz de realizar mediciones matemáticas de magnitudes físicas en un sistema para obtener información sobre el estado físico de dicho sistema (puede definirse como un sensor matemático o de reglas lógicas o de ambos casos).

Predictor: Observador capaz de predecir mediciones.

Controlador: Cualquier ente capaz de llevar a un sistema de una condición inicial a una deseada.

Lazo cerrado: Sistema retroalimentado cuya salida argumento de entrada para el controlador que la regula.

Segundo método de Lyapunov: Técnica para determinar si el modelo matemático de las ecuaciones diferenciales asociadas al sistema y su controlador en lazo cerrado tienen una solución estable (evita la resolución de las ecuaciones diferenciales).

Control por Campos Potenciales Artificiales: controlador basado en un campo virtual de fuerzas, generalmente cartesianas para llevar un sistema (en este caso, un robot manipulador o una partícula) de una condición inicial a una de operación en un espacio por lo general cartesiano, a pesar de que las fuerzas son mecánicas, se emplean modelos hidráulicos, térmicos, de interacción de cargas electromagnéticas, de conducción eléctrica, de resortes o de atracción gravitatoria por mencionar algunos.

Simbología y abreviaturas recurrentes:

n	Numero de grados de libertad del espacio articular.
u, v	Ejes de posición horizontal y vertical de una cámara.
q	Grado de libertad genérico o vector de grados de libertad de un robot.
qp	Derivada respecto al tiempo de q.
qd	Valores deseados de q.
x	Vector de posición cartesiano (x,y,z).
xr(q)	Vector de posición cartesiano asociado al robot.
xd	Vector de posición cartesiano deseado
GDL	Grado de libertad.
τ	Vector de Torques y Fuerzas articulares.
F	Vector de Fuerzas Cartesianas.
C.I.	Cinemática inversa.
C.D.	Cinemática directa.

Capitulo 1: Introducción

1.1 Introducción

Existen muchas aplicaciones Fig. 1 donde es deseable que un brazo manipulador capture objetos ya sean móviles o estáticos, ejemplos de ello son la asistencia deportiva (básicamente atrapar pelotas) o de una forma mucho mas usual y menos compleja dadas las condiciones dinámicas de los objetivos, en las líneas de producción, en el proceso, transporte y empaquetado de materia prima, herramientas o producto.

Por otra parte, un brazo manipulador en general se utiliza restringiendo su área de trabajo de otros objetos que no sean los prescindibles para la tareas que este realice incluyendo a otros robots o al operario, sin embargo existen condiciones que por limitantes de espacio o por necesidad, como es el caso de la interacción robot-humano o la existencia de paredes, requieren la inclusión de estos elementos

Como puede notarse la existencia tanto de objetivos como de obstáculos ya sean móviles o estáticos motiva el desarrollo de métodos para la interacción del manipulador con mencionados cuerpos.

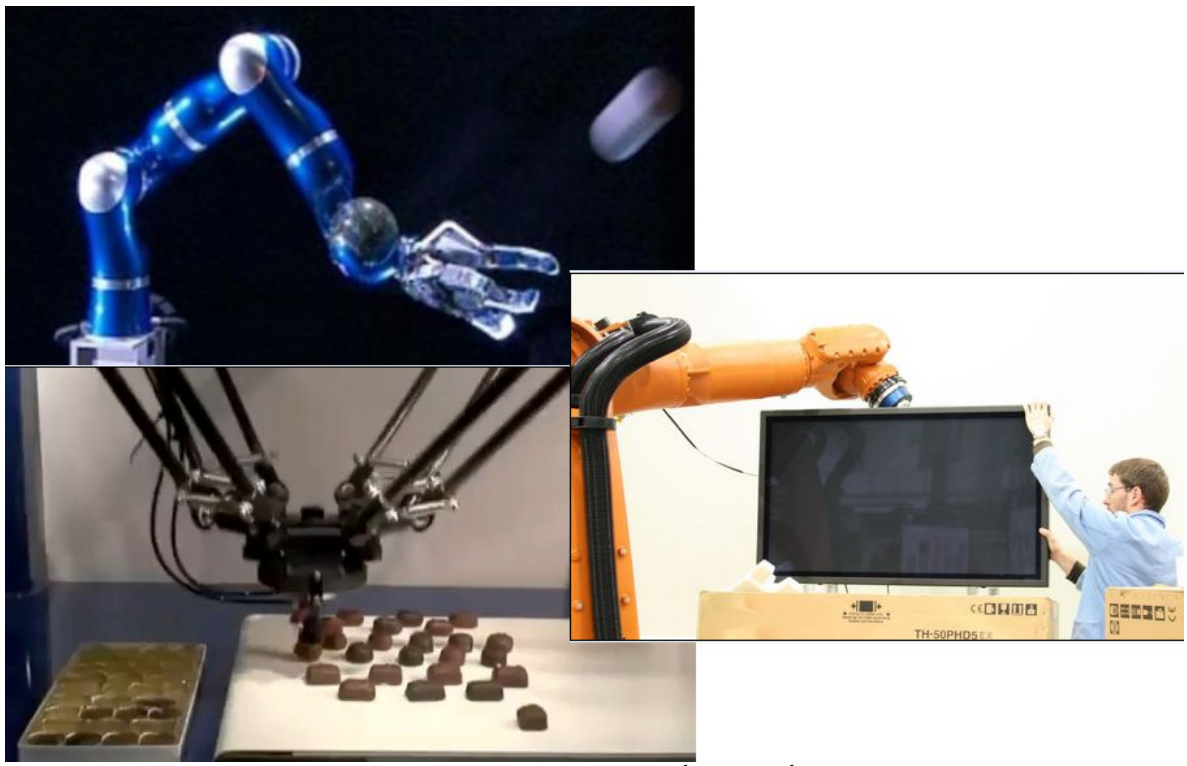


Figura 1. Tareas usuales de intercepción y evasión de objetos

1.2 Objetivo

Siendo la finalidad de esta tesis una resultante de 2 tareas: captura y elusión, se ilustra el proceso humano para realizarlas, el objetivo es adaptar ese proceso a un brazo manipulador robótico tridimensional auxiliándose de las herramientas matemáticas descritas (aunque se presenta de una forma secuencial, los procedimientos pueden ser paralelos).

Captura
Ubicación → Predicción → Captura

Herramientas matemáticas
Visión Estereo → Filtro de Kalman en modo Predictor → Potencial Artificial Atractor

Evasión
Ubicación → Monitoreo → Acción Evasiva

Herramientas matemáticas

Modo básico
Visión Estereo → Potencial Artificial Repulsor

Modo contra ruido
Visión Estereo → Filtro de Kalman en modo Seguidor → Potencial Artificial Repulsor

El procedimiento cuyo diagrama de bloques se ilustra en la Fig. 2 involucra las siguientes etapas:

a) Se definen todos los posibles obstáculos y objetivos diferenciando unos de otros por color y/o forma.
b) Aleatoriamente o por distancia al efector final (mano del robot, única parte con posibilidad de colisión con los blancos), se selecciona un objetivo y en tanto este no se atrape los demás objetos son considerados como obstáculos.
c) Una vez atrapado el objeto este desaparece visual o mecánicamente de la escena (por ejemplo marcándolo o retirándolo).
d) Se repiten b) y c) hasta que todos los objetivos son alcanzados

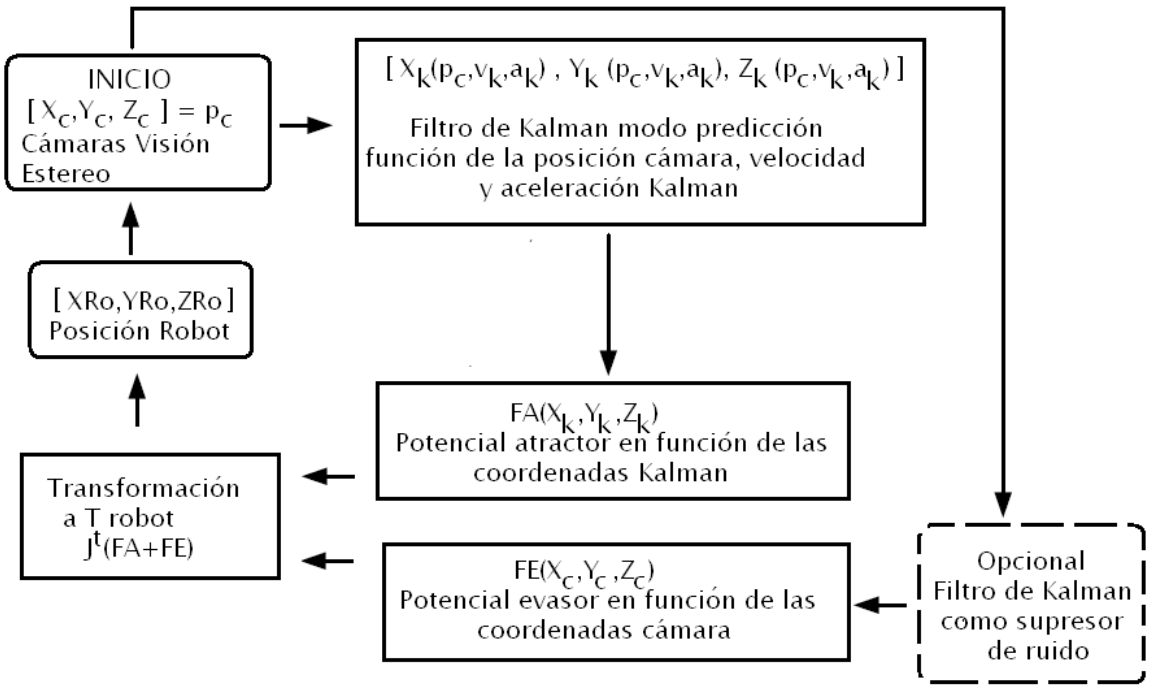


Figura 2. Diagrama de Bloques de la Tarea conjunta, x, y, z describen posiciones cartesianas, los subíndices c, Ro, k indican que corresponden a las cámaras, el robot y el filtro de Kalman respectivamente, p es el vector de posición cámara FA y FE son fuerzas de evasión y repulsión T son los pares y fuerzas del robot relacionados con las anteriores mediante el jacobiano transpuesto

1.3 Justificación

Uno de los fundamentos de la interacción robot –medio es la capacidad de la maquina para realizar la tarea programada sin afectar la integridad del operador, materia prima, otros robots o las herramientas que se encuentren dentro de su espacio de trabajo, por tanto es necesario el diseño de sistemas de evasión-intercepción de objetos, pudiendo ser partículas y/o volúmenes.

La mayoría de las tareas de intercepción y evasión de objetos son de amplia difusión y uso en robótica móvil, el principal problema de adaptarlas a un brazo manipulador consiste en que gran parte de los algoritmos de planeación de movimiento requieren ser validados por medio de las cinemáticas directa e inversa Fig. 4 lo cual puede presentar múltiples soluciones, soluciones extensas y en el peor de los casos ninguna (condiciones de operación conocidas como singularidades).

Lo anterior se debe a que los móviles, a excepción de aquellos con remolque, son considerados partículas para su interacción con el espacio de tarea, espacio cartesiano, por otra parte como se ilustra en la Fig. 3 los robots manipuladores al ser cadenas cinemáticas controladas por diversas articulaciones (no necesariamente todas como es el caso de los robots subactuados) son entidades interdependientes multiespaciales que requieren transformaciones cinemáticas para ser operables en el mundo cartesiano (por ejemplo un robot de 8 grados de libertad en el espacio tridimensional o un péndulo triple actuando en el plano)

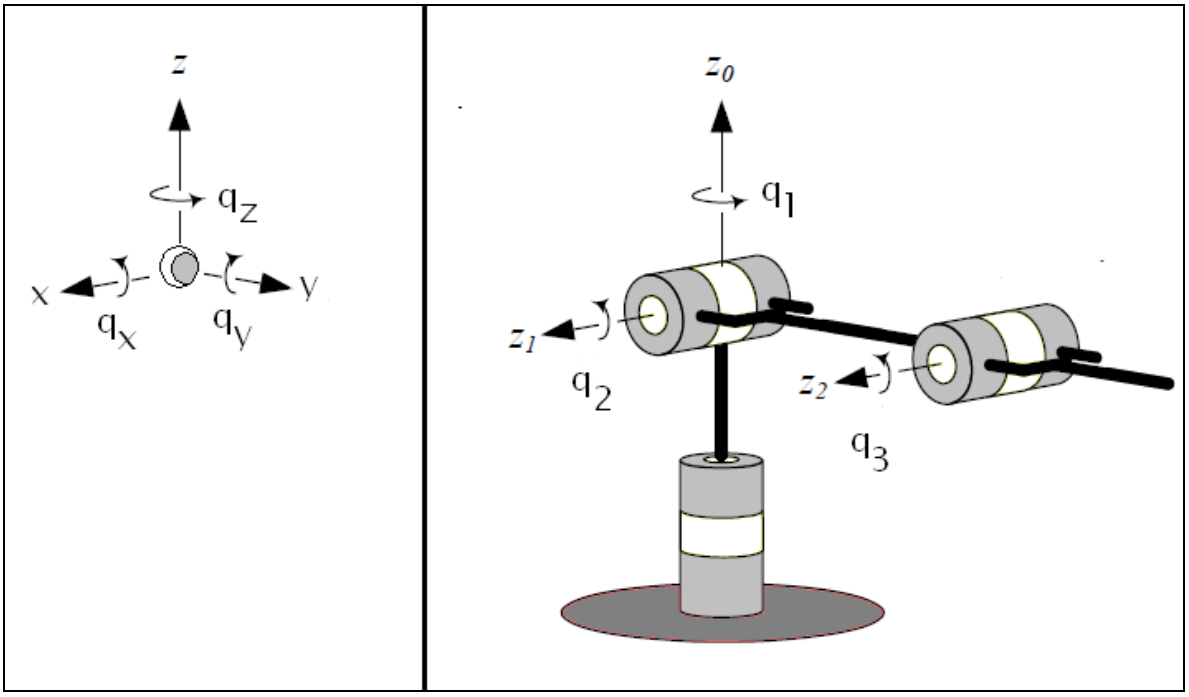


Figura 3. Espacio partícula - Espacio robot

Izquierda: partícula o robot móvil, Derecha: robot manipulador; En este caso todas las q son grados de libertad rotacional con respecto a cada uno de sus marcos coordenados [x,y,z], en el caso de la partícula la traslación en estos ejes es libre, en el caso del brazo manipulador esta restringida a su geometría, donde cada restricción espacial modifica la movilidad del robot

Así pues, para aplicaciones de intercepción y evasión de objetos móviles, donde la ejecución en tiempo real es un factor de prioridad, es importante contar con modelos de rastreo y captura igualmente rápidos (planeacion *online* punto a punto) y que en su evolución no presenten condiciones de inoperabilidad para el robot, por ejemplo árboles de decisiones, celdas neurodifusas, modelos de primitivas básicas de imagen, modelos de bordes y en el caso de este y otros trabajos, el uso de potenciales artificiales que básicamente generan fuerzas virtuales de atracción en dirección al punto objetivo y de repulsión respecto a los obstáculos intermedios, estas fuerzas se

basan en modelos matemáticos de fenómenos físicos de cohesión-evasión como la transferencia de calor, el flujo en tuberías, el comportamiento eléctrico o magnético de cargas o los modelos de atracción gravitatoria entre otros, al ser estos modelos en el espacio fuerza, no es requerida una transformación de cinemática inversa al espacio movimiento y por tanto los operadores de transformación no presentan condiciones de cancelación.

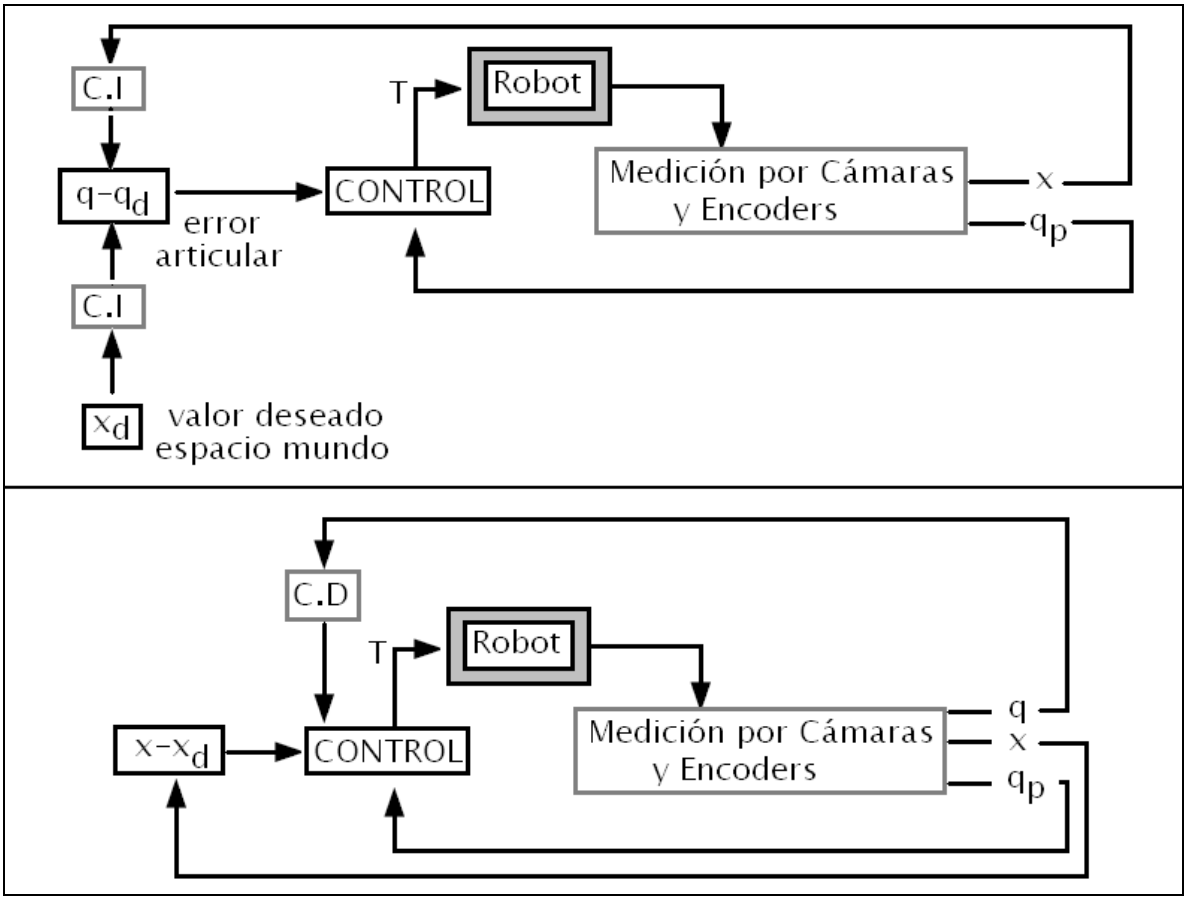


Figura 4. Control servovisual basado en posición:

Superior: Ciclo de cinemática inversa propio de la mayoría de los controladores de brazos manipuladores en los espacios de movimiento, Inferior: ciclo exclusivo de cinemática directa propio del control Fuerza en este caso potenciales artificiales C.I y C.D se refieren a la cinemática inversa y directa, los subíndices d indican valor deseado, p indica primer derivada, x y q son vectores cartesianos y articulares respectivamente

1.4 Antecedentes

Un primer análogo de este trabajo es aquel relacionado al modelo misil inteligente - blanco móvil, aunado a gran cantidad de investigaciones en robótica móvil, de lo que se da referencia en [12] y [42].

De entre las investigaciones mencionadas, la que motivo esta tesis es la de Mora y Tornero [1], en ella los autores diseñaron una metodología para la evasión de objetos móviles utilizando potenciales artificiales y filtro de Kalman, su tesis se limitó a robots móviles en 2D, las variantes de la presente investigación son aplicables a condiciones de operación mas complejas y extensibles a brazos manipuladores de base fija.

Arimoto [43] y Kelly [36] establecen la formalidad para su validación matemática y en [40], [41] esta la información indispensable para poner en operación al filtro de Kalman

Trabajos análogos a este tienen extensas publicaciones por parte de Pomares y Torres [34], la diferencia con esta tesis radica en las técnicas de evasión de obstáculos

Por otra parte, información referente a los 4 cimientos de este texto es decir: Robótica, Visión Estereo, Filtro de Kalman y Campos Potenciales Artificiales, exceptuando los libros, solo se indican los autores recomendados pues su obra en el área correspondiente es vasta e influye de alguna forma al desarrollo de esta tesis, algunos de sus trabajos mas importantes se incluyen en la bibliografía:

Robótica:

Bases: Spong, Robot Modeling and Control; Reza N. Jazar, Theory of Applied Robotics; Oussama Kathib, Handbook of Robotics
Controladores: Artículos de Arimoto, Hebertt Sira-Ramirez, Vicente Parra, Rafael Kelly, Fernando Torres y Jorge Pomares

Simuladores: Peter Corke

Innovaciones:
Artículos de Fernando Torres y Jorge Pomares, Vicente Parra Vega, Rafael Kelly

Visión Estereo (Stereopsis, Estereoscopia), Control servovisual:

Bases: Gang Xu, Zhengyou Zhang, Epipolar Geometry in Stereo Motion and Object Recognition; Fernando Torres, Jorge Pomares, Robots y Sistemas Sensoriales

Robots Móviles: Mora, M. C, Tornero, J, Fernando Torres, Jorge Pomares

Robots Manipuladores: J, Fernando Torres, Jorge Pomares

Simuladores: Mariottini, Giorgio Panin, OPENCV

Innovaciones: Emmanuel Dean, Peter Corke

Filtro de Kalman:

Bases: Kalman; Eli Brookner, Tracking and Kalman Filtering Made Easy; Mohinder S. Grewal, Kalman Filtering: Theory and Practice Using MATLAB

Robótica en general: Pomares, Torres, Roland Siegwart, Mora, M. C, Tornero

Simuladores: Houman Zarrinkoub, Brett Shoelson, Yi Cao

Innovaciones: Prakhar Agarwal, Yi Cao

Campos Potenciales Artificiales:

Bases: Kathib; Latombe Robot Motion Planning

Aplicaciones a la Robótica en general: Mora, M. C, Tornero, Torres y Pomares

Evasión de mínimos locales: Michael Goodrich, Xingjian Jing, Yuechao Wang, Kim, J.-O. Khosla, P.K, Latombe

1.5 Consideraciones

Para el desarrollo de este trabajo se considera:

a) Los algoritmos y metodologías descritas a lo largo de la tesis son aplicables a cualquier robot tridimensional de base fija con cadena cinemática abierta como se muestra en las presentes simulaciones

b) Se hacen suposiciones de cuerpo rígido, es decir con eslabones y articulaciones indeformables respecto a su estática y dinámica, además los cuerpos mencionados tienen masa constante

c) El control servovisual se realiza en el espacio cartesiano o bien el espacio imagen, siempre y cuando exista la transformación de cinemática directa que permita interactuar con el robot, esta tesis se desarrolla exclusivamente en el espacio cartesiano

e) El tiempo de muestreo se estandariza a unidades de ms, la magnitud dependerá del tiempo de captura de las cámaras (u otro sensor espacial, por ejemplo un radar)

d) Las tareas a realizar serán de posicionamiento, no de orientación Fig. 5 y serán en el espacio coordenado cartesiano

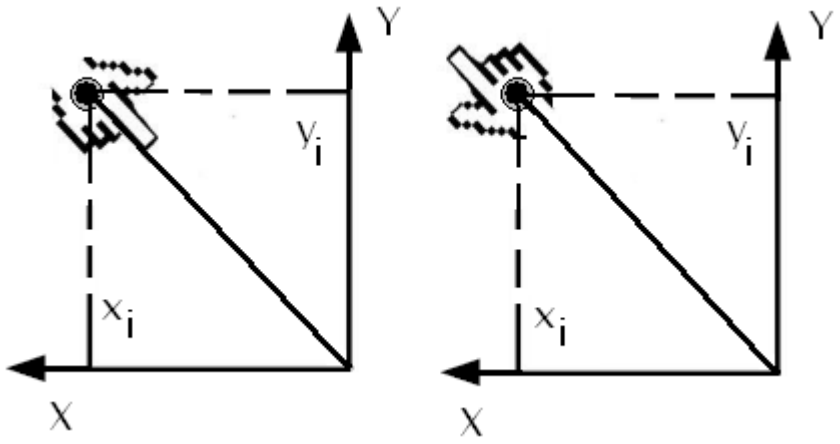


Figura 5. Diferencia entre tareas de posicionamiento y de orientación, la mano representativa de un robot móvil o del efector final de un brazo manipulador, en ambos casos tiene la misma posición (x_i,y_i), pero su orientación es distinta, en algunas tareas como lo es la colocación de tornillos no basta con que un desarmador llegue a una posición sino que este tenga cierta orientación sobre la ranura

1.6 Contribuciones de la tesis

1. Desarrollo de dos técnicas para evitar mínimos locales en la planeacion de trayectorias por potenciales artificiales, además se propone una mas con posible control de tiempo y posición basado en la ecuación de flujo de Navier-Stokes.

2. Desarrollo de una metodología para la aplicación conjunta del filtro de Kalman y las técnicas de potenciales artificiales en cualquier manipulador 3D de base fija y cadena cinemática abierta, así como robots móviles cuyo espacio articular sea igual al cartesiano.

3. Un Programa y metodología para calcular el regresor lineal de un sistema como el mencionado en caso de utilizar controladores que así lo requieran

4. Una Estrategia para la simulación de los algoritmos utilizando Matlab® y las herramientas de Peter Corke [2] y Mariottini [3], ambas son entornos gráficos para desarrollar y en el caso de esta tesis visualizar algoritmos de control y control servovisual Fig. 6.

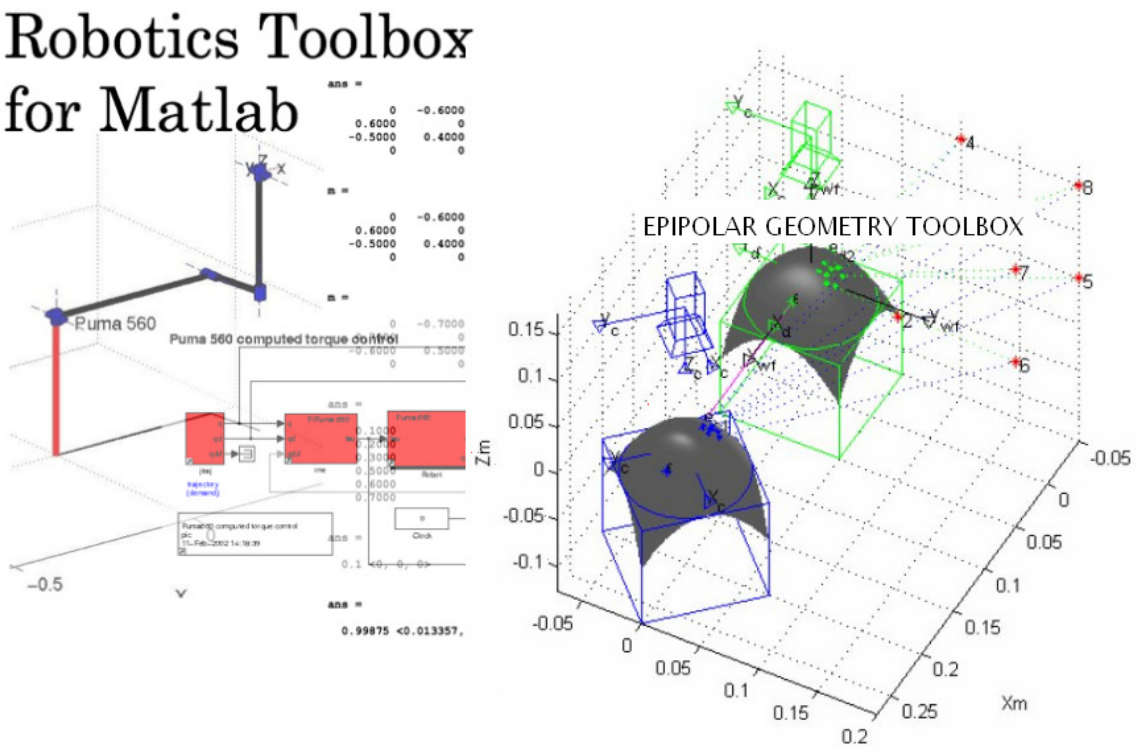


Figura 6. Entornos gráficos para la simulación de robots y control servovisual de Corke y Mariottini

1.7 Materiales y métodos (esquema de la tesis)

Tal como fueron descritas las tareas, los capítulos 2 3 y 4 abordan las técnicas necesarias para que la computadora ubique objetos en el espacio, el capitulo 5 tiene 2 vertientes evasión e intercepción, en el capitulo 6 se muestra su adecuación a robots manipuladores, un ideograma del texto subsiguiente se ilustra a continuación, al final de cada capitulo se ofrece una discusión sobre los puntos relevantes concernientes a su contenido y en el desarrollo de los mismos los resultados, finalmente se presentan las conclusiones y bibliografía.

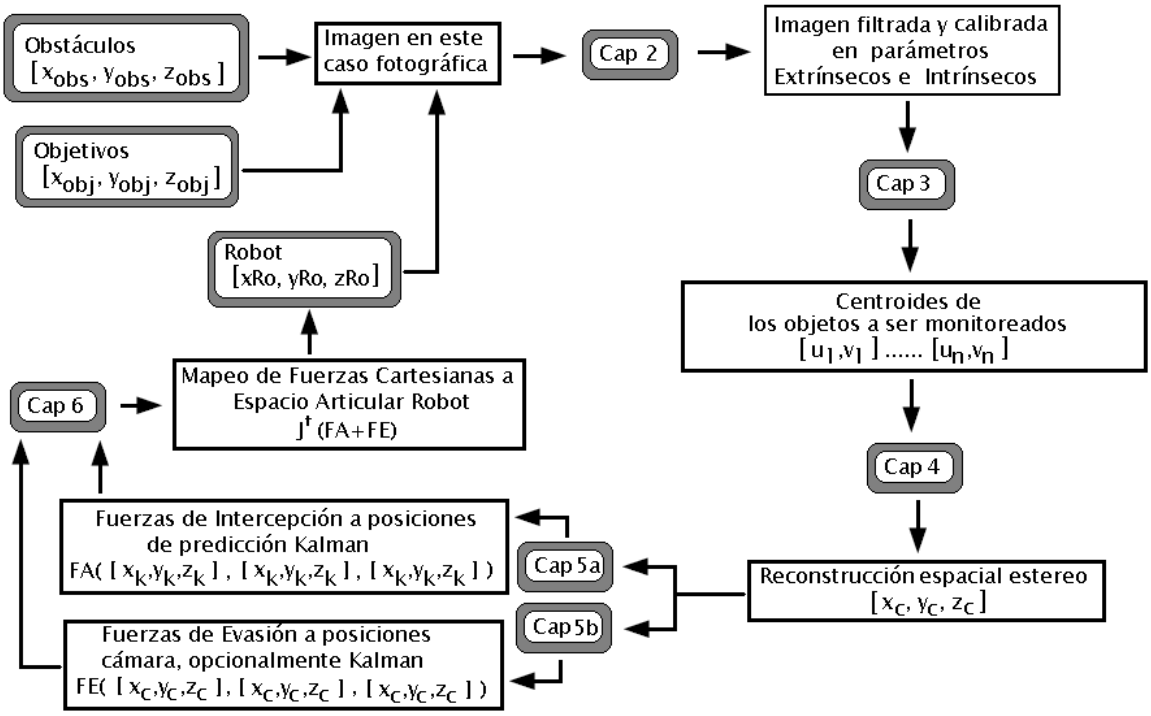


Figura 7. Diagrama de bloques de la tesis respecto al procedimiento descrito en la sección 1.2, cada capitulo se representa como una caja negra junto con sus respectivas entradas y salidas, se usa para los casos correspondientes la misma notación que en la Fig. 3, los subíndices obs y obj referencian a obstáculos y objetivos respectivamente; u,v indican posición de puntos específicos en cada imagen

Capitulo 2: Calibración de cámaras

El presente capitulo[1] considera que el control se realiza en el espacio cartesiano y no en el de imagen

2.1 Parámetros intrínsecos

Los parámetros intrínsecos también conocidos como parámetros internos de la cámara, Fig. 9, son propios de la lente y son útiles para encontrar la distancia focal, el punto principal, el centro óptico, el escalamiento o simplemente para corregir distorsión óptica y ruido de la lente y obtener píxeles cuadrados en vez de convexos o cóncavos, así como eliminar ruido impulsivo, gaussiano o multiplicativo lo cual modificaría la interpretación del mundo cartesiano y/o de los objetos a monitorear. La técnica de calibración mas usual es la de Tsai descrita en Pomares [7] y los filtros mas comunes para eliminación de ruido son las mascaras de media, de mediana y la de gauss Fig. 8.

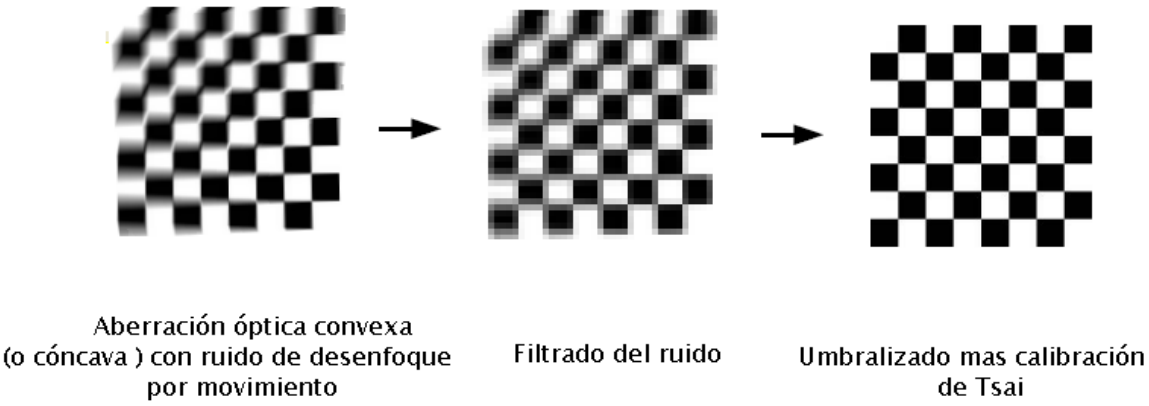


Figura 8. Calibración de Tsai y reducción de ruido

[1] *La teoría vista desde este capitulo hasta el 4to puede implementarse con librerías desarrolladas por el usuario para tal propósito o bien, con OPENCV [4] u OPENTL [5] por mencionar algunas librerías con algoritmos preprogramados, mas información de los capítulos mencionados puede hallarse en Xu [6] Pomares [7] y Trucco [8]*

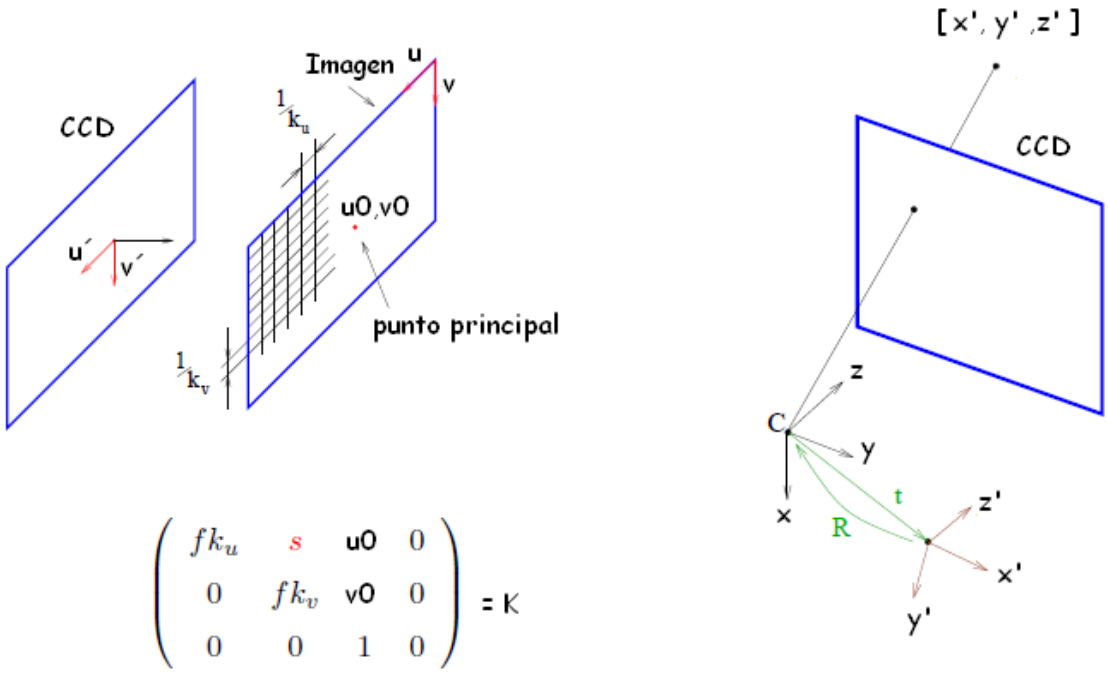


Figura 9. Parámetros intrínsecos y extrínsecos, modificado de Baumela [44], u,v son los valores pixel luego de la calibración intrínseca K (izquierda), estos permiten la transformación píxel - longitud métrica, la matriz de calibración intrínseca consta de escalamientos (k), las coordenadas del punto u,v origen y el sesgo o distorsión angular de cámara (s), a la derecha se muestran los parámetros extrínsecos que asocian los sistemas de referencia cámara con los del objeto a monitorear, es decir no dependen de la cámara sino de su pose, en este caso solo se ilustran traslación y rotación.

2.2 Parámetros extrínsecos

Luego de obtener una imagen corregida y cuadrada, prosigue la parte relacionada a indicar la configuración de las cámaras respecto al mundo, es decir como habrán de colocarse en pose (posición y orientación) respecto a un punto origen y si su configuración será cámara en mano o fija (externa al cuerpo) Fig. 10

Para el caso de este trabajo la configuración es de cámara fija, por tal la ecuación de parámetros extrínsecos e intrínsecos es (1):

$$s\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K_{3x4}\begin{bmatrix} R_{3x3} & t_{3x1} \\ 0 \quad 0 \quad 0 & 1 \end{bmatrix}\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \qquad (1)$$

$$s \propto z$$

A esta ecuación resultante de la corrección intrínseca y la proyección extrínseca se le denomina modelo proyectivo cámara, (u,v) es algún punto de real de los objetos en 3D proyectados en la cámara K es la matriz de parámetros intrínsecos, R y t son parámetros extrínsecos de la pose de la cámara (rotación y traslación), s es un factor de alejamiento o escala proporcional a la profundidad del objeto, el resultado es una proyección bidimensional o punto cámara.

En el caso cámara en mano la fila [0 0 0 1] adopta otros valores y en general la matriz de transformación homogénea se vuelve:

$$\begin{bmatrix} R_{3x3} & t_{3x1} \\ P_{1x3} & E_{1x1} \end{bmatrix} \qquad (2)$$

Donde P y E son en ese orden el vector de perspectiva y el factor de escalamiento.

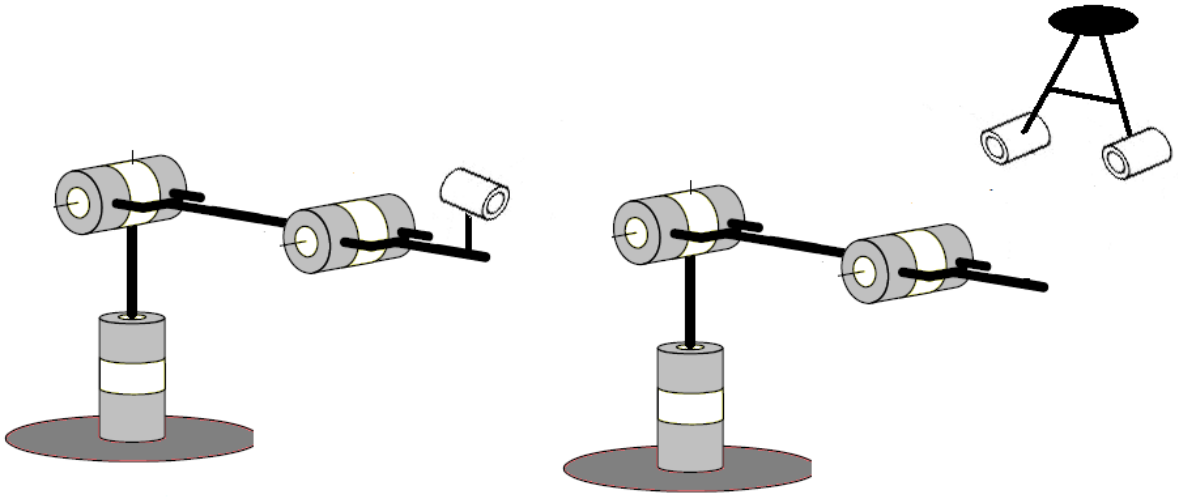


Figura 10. Izquierda configuración cámara en mano, derecha cámara fija

2.3 Discusión

El argumento de entrada para el procedimiento descrito en este capitulo es la imagen sin procesar, la salida es simplemente la misma imagen sin ruido, corregida en deformación y calibrada en pose, los principales problemas en este punto son la reducción de ruido el cual se ve afectado por la iluminación, el desenfoque o el simple movimiento del objeto o la cámara, además de obtener una adecuada relación píxel-metro.

Como se menciono en líneas previas, la configuración usada es la de cámara fija (es decir que no esta sobre el cuerpo del robot).

La calibración se realiza como sigue (usando el toolbox de Mariottini):

1. Se establecen los marcos mundo (negro) y el o los de cámara (verde).

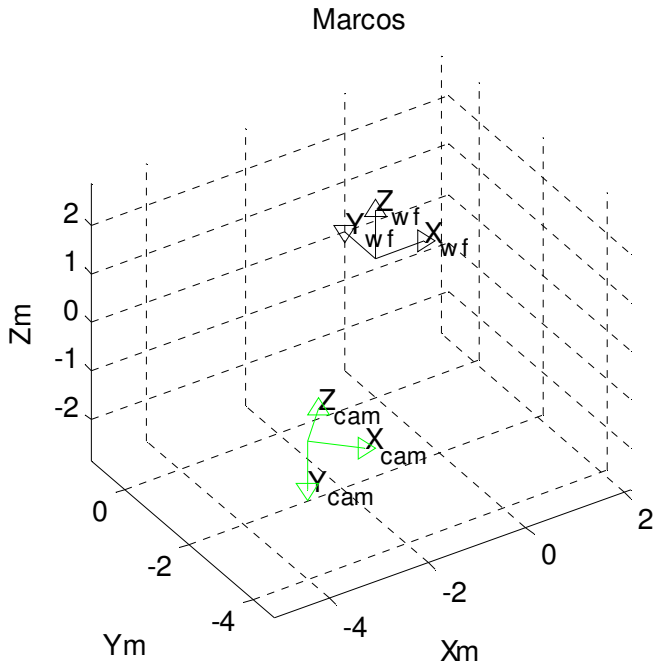


Figura 11. Marcos de mundo y cámara

2. Se colocan las camaras y puntos a monitorear.

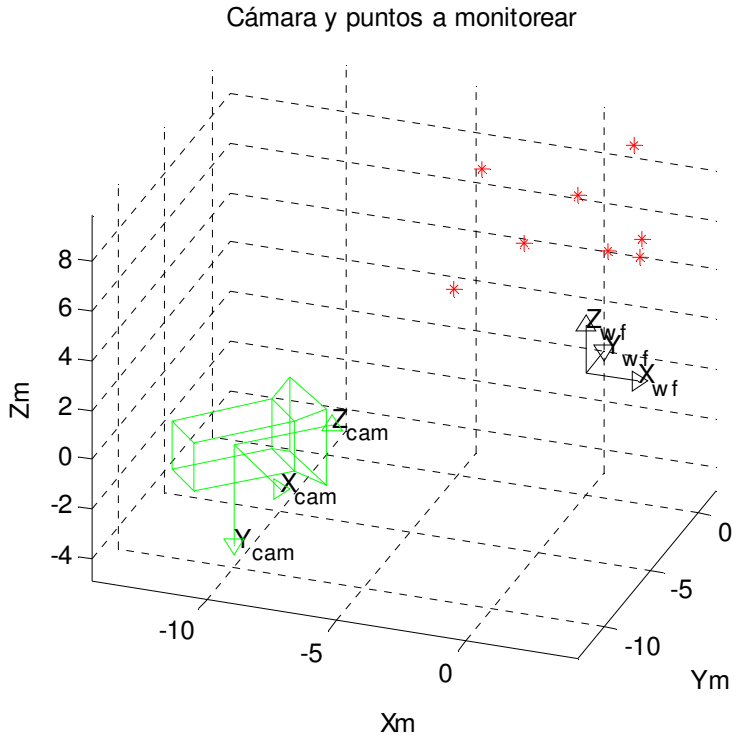


Figura 12. Definición de cámara y puntos a monitorear

3. Se definen y encuentran los parámetros extrínsecos e intrínsecos con ello se obtienen la proyección imagen.

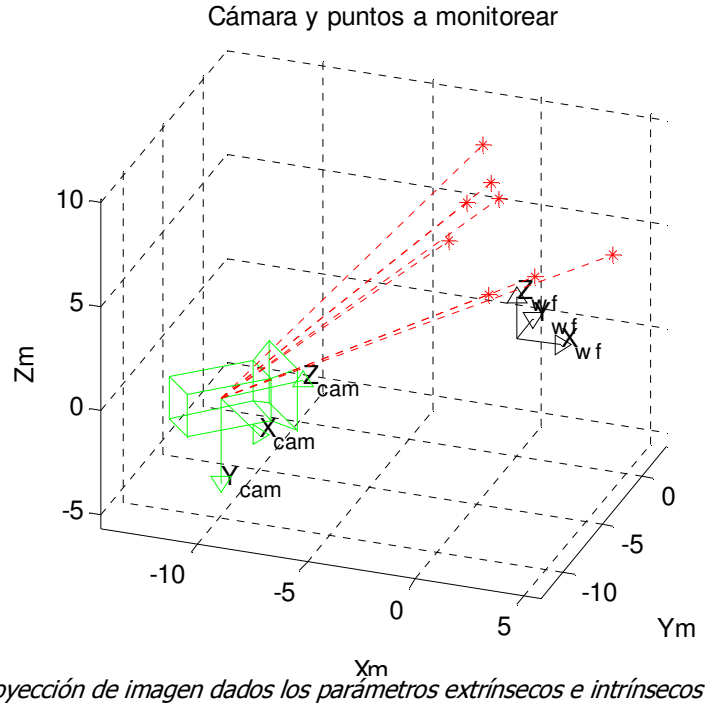


Figura 13. Proyección de imagen dados los parámetros extrínsecos e intrínsecos (vista isometrica)

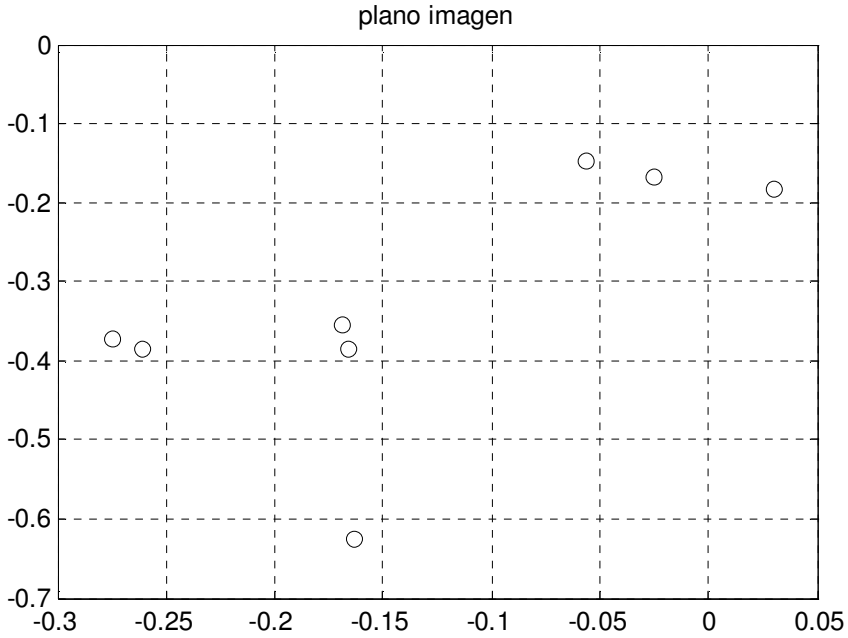


Figura 14. Proyección de imagen dados los parámetros extrínsecos e intrínsecos (vista fotográfica)

Capitulo 3: Detección de objetos

3.1 Metodología sugerida

De la imagen descrita en el capitulo anterior solo interesa aislar el referente al centroide de objetos en especifico, no toda la fotografia (lease el capitulo 7 para ver los resultados), se presenta a continuación una metodología sugerida para hacerlo, básicamente consta de seleccionar al conjunto de los objetos de interés, filtrar el ruido de selección, separar los objetos destacados como objetos individuales y finalmente de cada entidad obtener su posición representativa (centroide)

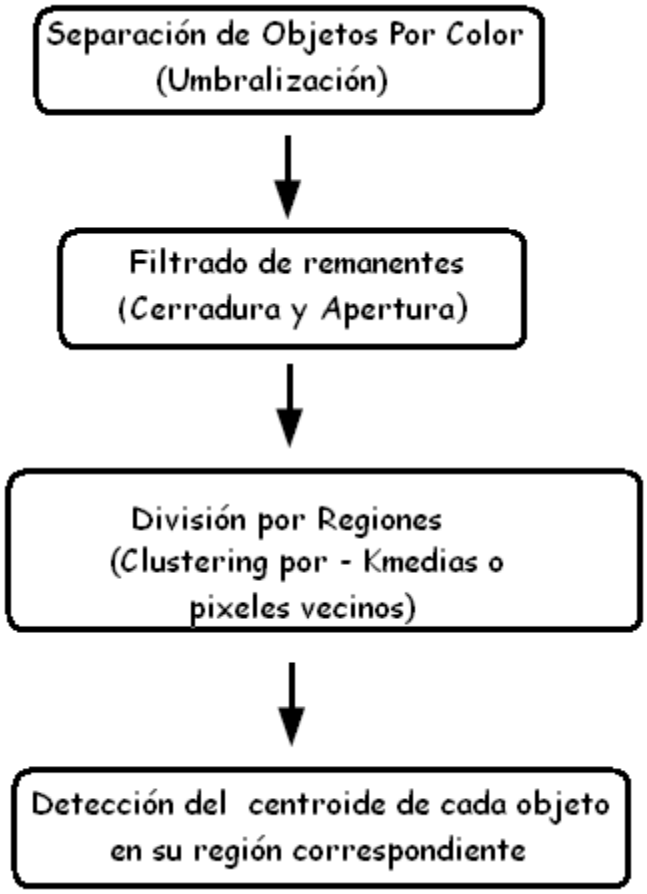


Figura 15. Metodología sugerida para la detección de objetos en operaciones de seguimiento

3.2 Separación por color y filtrado de remanentes

En primer lugar es conveniente separar todos los objetivos y obstáculos así como marcas visuales del robot del resto de la imagen, asignando colores brillantes a los objetos mencionados. Un método alternativo es la detección de formas, pero dado que facilita los métodos de esta tesis emplear esferas, se recurre a utilizar umbralizado y alguna técnica de suavización de imagen para tener bordes adecuados y no pixeleados del objeto a observar por ejemplo erosionado combinado con dilatación, en operaciones de cerradura y apertura Fig. 16.

Figura 16. Umbralizado, apertura y cerradura (Ejemplificación exagerada)

3.3 Segmentación

A continuación se subdivide la imagen en regiones. Algunas técnicas existentes para este propósito son la asociación de píxeles vecinos (conexos) o las k-medias, la teoría asociada es extensa, pero puede hallarse en Pomares[7], Trucco[8] o Bradsky[4] por citar algunos referentes.

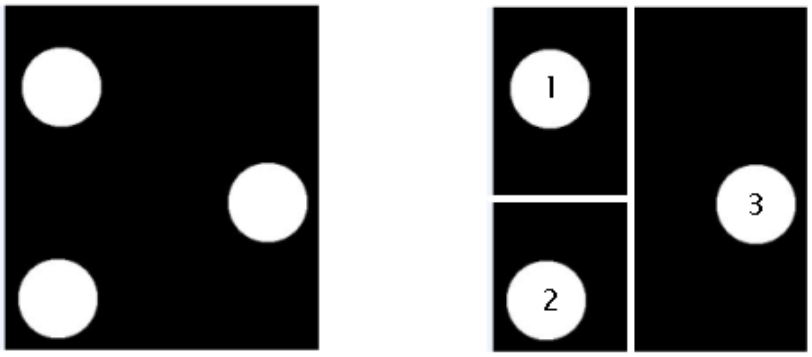


Figura 17. Segmentación

3.4 Detección del centroide

Una vez separados los objetos, se localizan sus centroides para operar con un solo referente coordenado de su posición (u,v)

Dado que cada región tendrá un valor mínimo y un máximo (ya sea en píxeles o en su correspondencia métrica) en ejes vertical y horizontal, la posición del centroide del *n*-esimo objeto estará definida por el promedio de los valores máximos en cada dirección [25] [26], es decir:

$$u_n = \frac{\max arriba_n + \max abajo_n}{2}$$

$$v_n = \frac{\max der_n + \max izq_n}{2} \qquad (3)$$

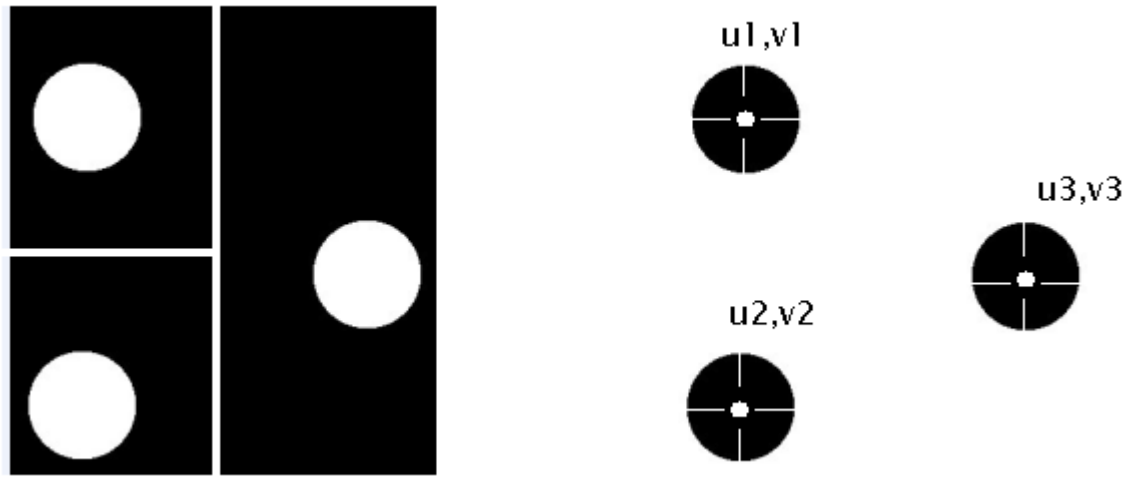


Figura 18. Ubicación del centroide

En este caso la geometría de los objetos a monitorear permite tal simplicidad, no obstante, para objetos de formas diversas se utiliza un algoritmo análogo al calculo de los centros de masa donde el equivalente de las masas son las intensidades de píxel. Este algoritmo es invariante ante rotaciones, traslaciones y escalamiento

Definiendo los momentos de área (o momentos geométricos) de orden p,q

$$M_{p,q} = \int\int x^p y^q f(x,y)dxdy \quad (4)$$

Cuyo caso discreto es:

$$M_{p,q} = \sum_x \sum_y x^p y^q f(x,y) \quad (5)$$

Donde f es el valor de color correspondiente a cada píxel (coordenada de información de la imagen), $M_{0,0}$ por ejemplo equivaldría a el área del objeto

Se determina al centroide de cada objeto como sigue:

$$u = M_{1,0}/M_{0,0}$$
$$v = M_{1,0}/M_{0,0} \quad (6)$$

3.5 Discusión

El argumento de entrada para el proceso descrito en este capitulo es la imagen corregida del capitulo 2, la salida son puntos específicos de la imagen (u,v), una imagen sin control de iluminación conduce a que un circulo deje de serlo y por tanto el centroide obtenido no arroja información posicional adecuada, otro problema consiste en el tiempo de disparo de la cámara, ya que si es lento en comparación al móvil, pueden obtenerse objetos "fantasma" Fig. 19, por esta razón se deben utilizar cámaras de captura progresiva en lugar de entrelazada Pomares[7] y monitorear objetos de movimiento lento respecto al sensor.

Figura 19. Objeto "fantasma"

Capitulo 4: Conversión 2D-3D

Se cuenta con dos imágenes del mismo evento y los objetos de interés aislados, a continuación se reconstruye su posición espacial, toda conversión 2D a 3D involucra al menos dos capturadores o un capturador divido en dos tiempos o regiones puesto que un solo sensor ofrece solamente una descripción planar (2 parámetros) y es requerida una descripción espacial (3 parámetros), a toda conversión se le asocia una matriz de transformación cuya ecuación depende de la geometría de captura

4.1 Técnica estereo - introducción

Es la empleada en el presente trabajo, es de las mas utilizadas en control de robots debido a su rango visual y esto se debe a su parecido con algunas configuraciones biológicas, también se emplea en humanoides y robots móviles, los ejes focales actúan en el mismo plano (coplanar) o convergen hacia un mismo punto con la correspondiente variación angular igual en magnitud y opuesta en sentido (es decir una cámara rotada a *n* grados respecto al plano de trabajo en sentido horario y la otra a *n* grados en antihorario, esta es llamada configuración estero convergente)

En ambos casos, se posee información posicional referente al plano de colocación, la profundidad se obtiene con una analogía matemática del proceso natural de hacer bizcos (disparidad), objeto lejano disparidad menor (puntos de proyección tendientes a paralelos), objeto cercano disparidad mayor (puntos de proyección tendientes a colineales) Fig. 20

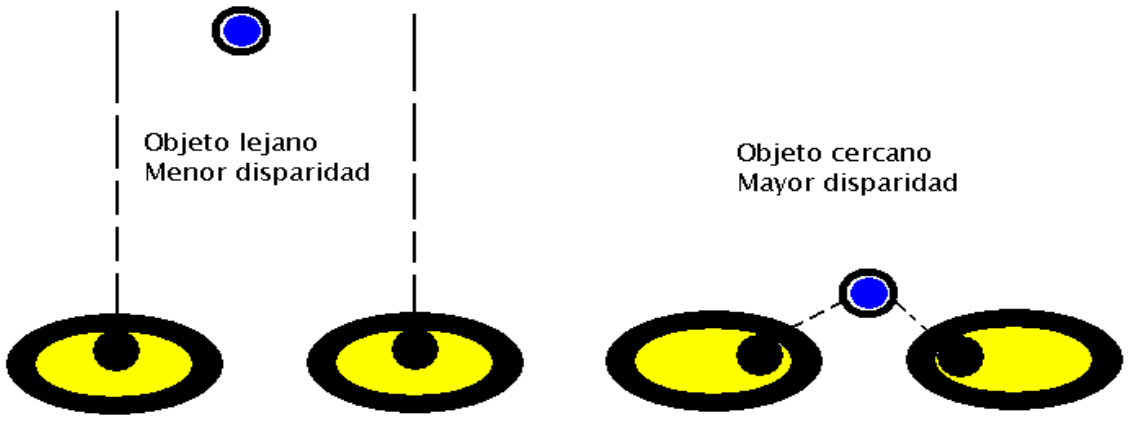


Figura 20. Analogía ocular de la disparidad (configuración estereo), la profundidad del objeto queda determinada como una función de la disparidad

4.2 Estereo coplanar

De acuerdo a la Fig. 21 y por relación de triángulos, se obtienen las presentes ecuaciones (7) (las alturas de las cámaras (v_i), (v_d) deben ser iguales), además se elige un mismo tipo de cámaras para garantizar en lo posible que f es decir la distancia focal en cada cámara sea equivalente

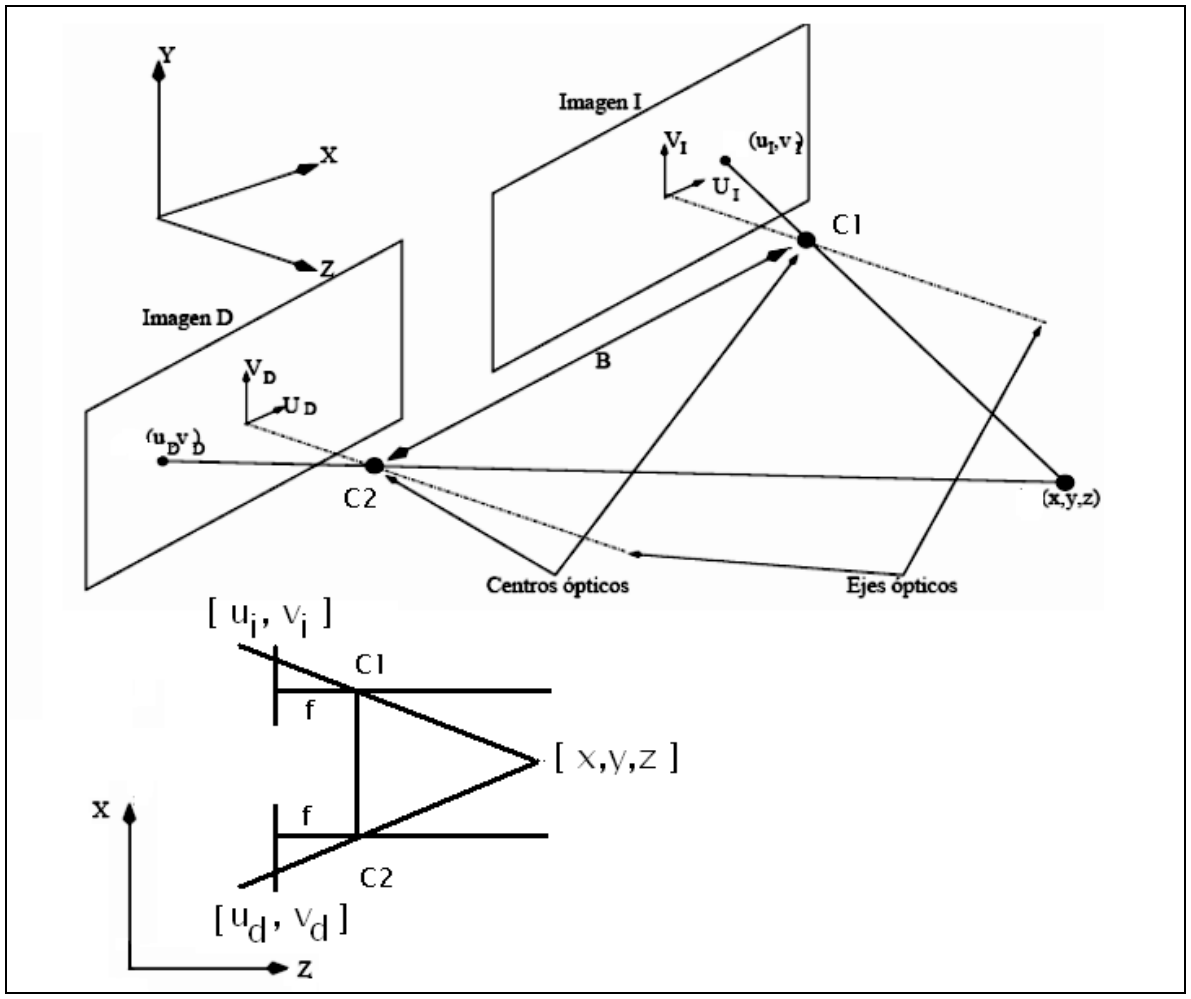


Figura 21. Configuración Estereo Coplanar, superior vista isométrica, inferior vista (x, z), los subíndices i, d indican izquierda y derecha respectivamente, C se refiere a cada cámara, B es la distancia entre centros ópticos y f la distancia focal.

$$X = \frac{u_i B}{u_i - u_d}$$

$$Y = \frac{v_i B}{u_i - u_d} \qquad (7)$$

$$Z = \frac{fB}{u_i - u_d}$$

En el sistema de ecuaciones descrito $u_i - u_d$ es la disparidad.

4.3 Estereo convergente

Con el propósito de aumentar el rango visual y reducir los puntos ciegos de la visión estereo coplanar Fig. 22, es decir aquellos que presenta un objeto no transparente en el plano opuesto de captura y también aquellos donde una cámara pierde información total del objeto (muy a la izquierda o muy a la derecha), existe la versión convergente Fig. 23, esta se ve limitada por la distancia de convergencia a costa de una mejor visibilidad.

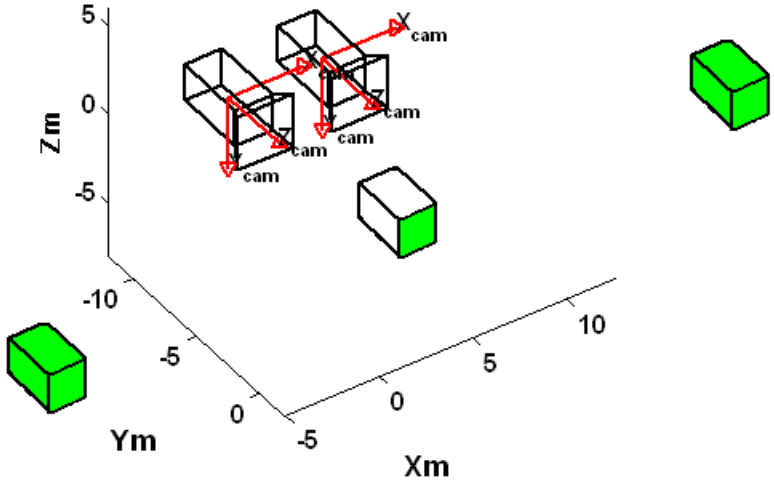


Figura 22. Puntos ciegos en la configuración estereo coplanar

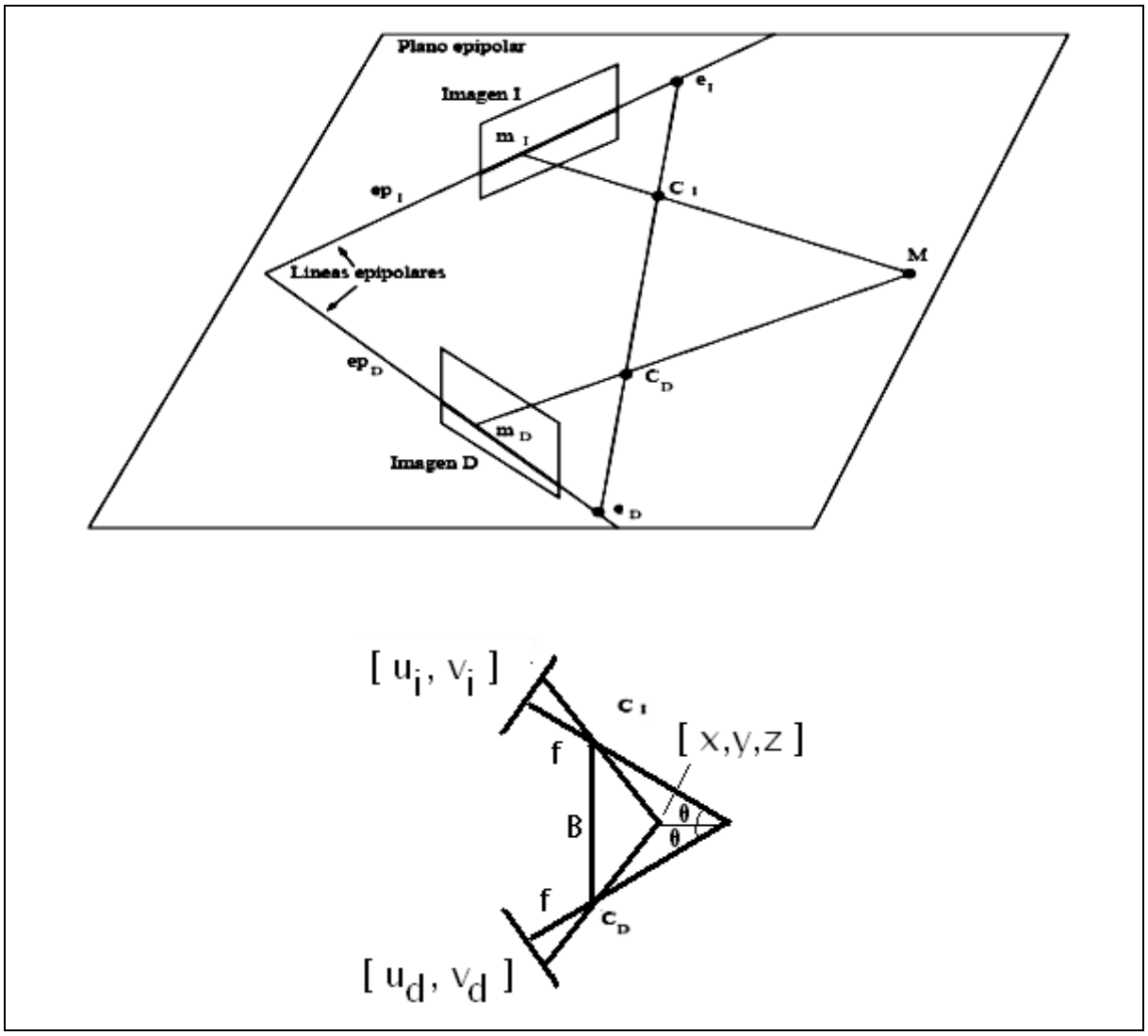


Figura 23. Configuración estereo de ejes convergentes vista isométrica y superior.

Sin embargo la solución del sistema de ecuaciones (8) generado a partir de tal modificación implica procesos de rectificado para llevarlo a un sistema virtual de ejes paralelos y eso conlleva técnicas de geometría epipolar [6], [7]. Por tal razón, para fines de este trabajo se ha optado por emplear la configuración coplanar.

$$u_i = \frac{f\left(X-\frac{B}{2}\right)\cos\theta+Z\sin\theta}{\left(X-\frac{B}{2}\right)(-\sin\theta)+Z\cos\theta} \qquad v_i = \frac{fY}{\left(X-\frac{B}{2}\right)(-\sin\theta)+Z\cos\theta}$$

$$u_d = \frac{f\left(X+\frac{B}{2}\right)\cos\theta+Z\sin\theta}{\left(X+\frac{B}{2}\right)(-\sin\theta)+Z\cos\theta} \qquad v_d = \frac{fY}{\left(X+\frac{B}{2}\right)(-\sin\theta)+Z\cos\theta} \quad (8)$$

4.4 Estereo coplanar no calibrada

El inconveniente de la calibración de imágenes radica en que por cada modificación en la pose de las cámaras se debe rehacer el proceso de calibración parametrica descrito en líneas previas, sin embargo, existen técnicas que permiten trabajar directamente en el espacio imagen sin tener que calibrar los sensores, una muy usual es la mencionada en Yakimovsky-Cunningham [57], de amplio uso en la NASA [55] y la universidad de Oxford[56] esta se basa en la obtención de 4 vectores (CAHV Fig. 24) donde C es un vector de dimensión 3 que indica el centro focal de cada cámara, A es un vector unitario normal al plano de la cámara y V, H son vectores de referencia vertical y horizontal con origen en el punto C, todos son igualmente tridimensionales.

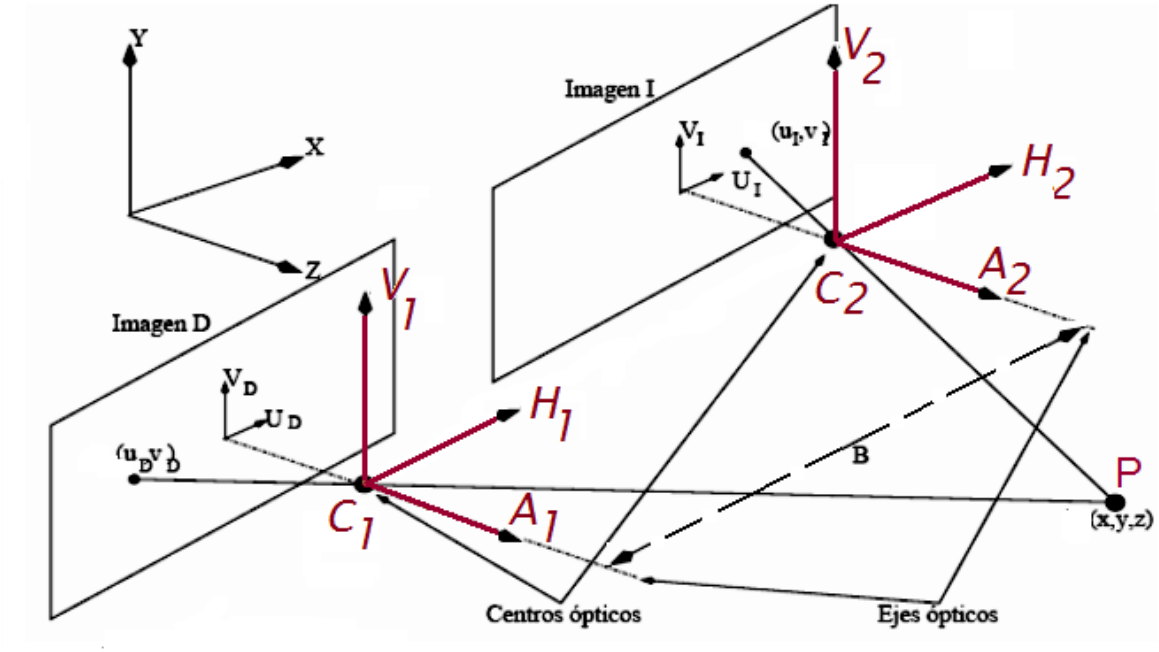


Figura 24. Configuración Estereo Coplanar no calibrada

Para utilizar visión no calibrada se necesita un mapeo de espacios (cámara-mundo) conocido como jacobiano visual, el cual tambien proporciona condiciones de oclusión-monitoreo, a continuación se detallan las ecuaciones para su obtención:

Dado un punto espacial a monitorear P=(x,y,z), se obtiene por cada cámara su proyección imagen Fig. 24:

$$u = \frac{(P-C)\cdot H}{(P-C)\cdot A} \qquad v = \frac{(P-C)\cdot V}{(P-C)\cdot A} \quad (9)$$

Reescribiendo estas ecuaciones y notando que $(u(A)-H),(v(A)-V)$ son perpendiculares a (P-C) podemos definir r el cual es un vector unitario proporcional a la profundidad (u, v guardan relación con x, y; r lo hace con z).

$$(P-C)\cdot(u(A)-H)=0$$
$$(P-C)\cdot(v(A)-V)=0$$

$$r=(u(A)-H)\times(v(A)-V) \quad (10)$$

Ahora bien, dado que las cámaras no están calibradas, cada cámara tiene su correspondiente sistema u,v,r (en las configuraciones precalibradas se forza a r a ser el mismo vector), por tanto se considera que la imagen P se forma en un punto promedio Fig. 25.

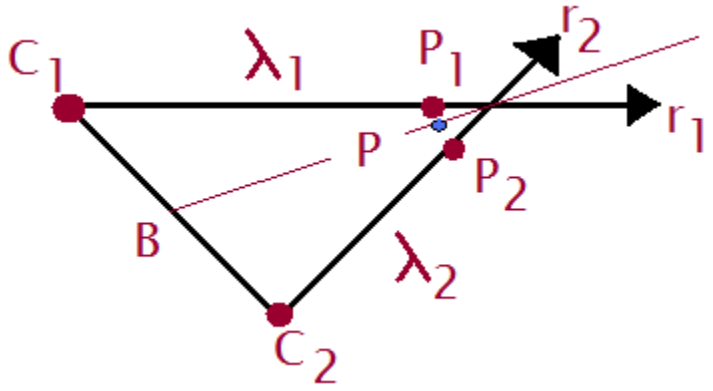


Figura 25. Vista superior y obtención de P como un promedio

$$P=f(u_1,v_1,u_2,v_2)=\frac{P_1+P_2}{2}=\frac{C_1+C_2+\lambda_1 r_1+\lambda_2 r_2}{2} \quad (11)$$

Donde

$$P_1=C_1+\lambda_1 r_1 \;\; ; \;\; P_2=C_2+\lambda_2 r_2$$
$$\lambda_1=\frac{b\cdot r_1-(b\cdot r_2)(r_1\cdot r_2)}{1-(r1\cdot r2)^2}$$
$$\lambda_2=(r_1\cdot r_2)\lambda_1-b\cdot r_2$$

Esto debido a que:

$$(P_2 - P_1) \cdot r_1 = (C_2 + \lambda_2 r_2 - C_1 - \lambda_1 r_1) \cdot r_1 = 0$$
$$(P_2 - P_1) \cdot r_2 = (C_2 + \lambda_2 r_2 - C_1 - \lambda_1 r_1) \cdot r_2 = 0$$

Con base en las ecuaciones presentadas, se obtiene el Jacobiano que relaciona espacio imagen con espacio cartesiano

$$J_i = \frac{\partial f(x,y,z)}{\partial(u_1,v_1,u_2,v_2)} = \frac{\lambda_1 \nabla r_1 + \lambda_2 \nabla r_2 + r_1 \nabla \lambda_1 + r_2 \nabla \lambda_2}{2} \quad (12)$$

Donde:

$$\nabla r_1 = \begin{bmatrix} \frac{\partial r_{1x}}{\partial u_1} & \frac{\partial r_{1x}}{\partial v_1} & 0 & 0 \\ \frac{\partial r_{1y}}{\partial u_1} & \frac{\partial r_{1y}}{\partial v_1} & 0 & 0 \\ \frac{\partial r_{1z}}{\partial u_1} & \frac{\partial r_{1z}}{\partial v_1} & 0 & 0 \end{bmatrix} \quad \nabla r_2 = \begin{bmatrix} 0 & 0 & \frac{\partial r_{2x}}{\partial u_2} & \frac{\partial r_{2x}}{\partial v_2} \\ 0 & 0 & \frac{\partial r_{2y}}{\partial u_2} & \frac{\partial r_{2y}}{\partial v_2} \\ 0 & 0 & \frac{\partial r_{2z}}{\partial u_2} & \frac{\partial r_{2z}}{\partial v_2} \end{bmatrix}$$

$$\nabla \lambda_2 = (r_2^T \nabla r_1 + r_1^T \nabla r_2)\lambda_1 + (r_1^T r_2)\nabla \lambda_1 - b^T \nabla r_2$$
$$\nabla \lambda_1 = \frac{\nabla u(x) v(x) - \nabla v(x) u(x)}{v(x)^2}$$

Con:

$$u(x) = b \cdot r_1 - (b \cdot r_2)(r_1 \cdot r_2)$$
$$v(x) = 1 - (r_1 \cdot r_2)^2$$
$$\nabla u(x) = b^T \nabla r_1 - (b^T \nabla r_2)(r_1 \cdot r_2) - (b \cdot r_2)(r_2^T \nabla r_1 + r_1^T \nabla r_2)$$
$$\nabla v(x) = -2(r_1 \cdot r_2)(r_2^T \nabla r_1 + r_1^T \nabla r_2)$$

4.5 Discusión

Teniendo como entrada los argumentos (u,v) de dos imágenes se obtiene una posición cámara (x_c, y_c, z_c) cuya igualdad con la posición real (x_r, y_r, z_r) esta en función de las consideraciones mostradas en los capítulos 2 3 y 4, las técnicas descritas requieren que las cámaras operen exactamente a la misma altura para tener un referente geométrico llamado plano epipolar, de lo contrario existirán errores de correspondencia de píxeles entre cámaras y en consecuencia errores de transformación/medición, para la presente tesis basta un modelo de ejes coplanares por la simpleza de sus ecuaciones asociadas respecto a la configuración convergente, aunque esto restringe al control servovisual al espacio cartesiano.

Por otra parte, el modelo estero no calibrado, cuyo control se realiza en el espacio imagen requiere un procesador que soporte la realización de las ecuaciones descritas en el capitulo 4.4 y aunque prescinde del proceso de calibración dotando de mayor flexibilidad al sistema estéreo de captura, también necesita el conocimiento de los parámetros CAHV, lo cual puede verse como una “calibracion” puramente intrínseca.

Para poner en practica el algoritmo estereo coplanar no calibrado con la herramienta de Mariottini basta colocar ambas cámaras en la disposición descrita y una vez obtenidos los valores imagen hallar la transformación 3D con las ecuaciones referidas.

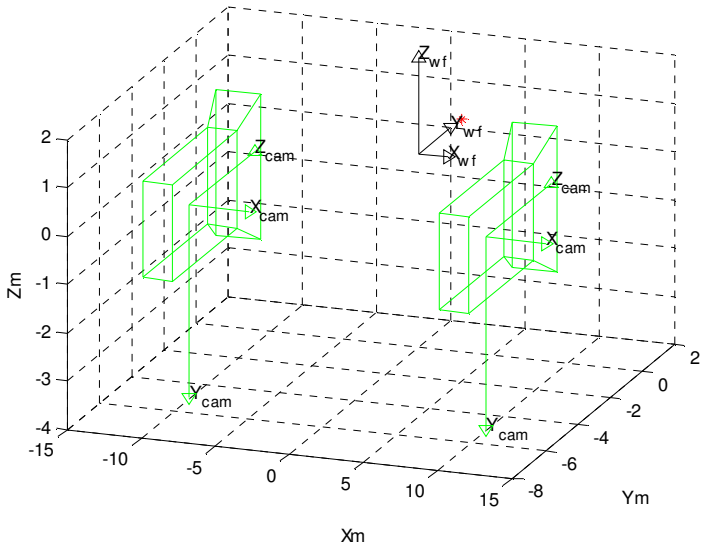


Figura 26. Simulación configuración estereo calculando la posición de un punto móvil.

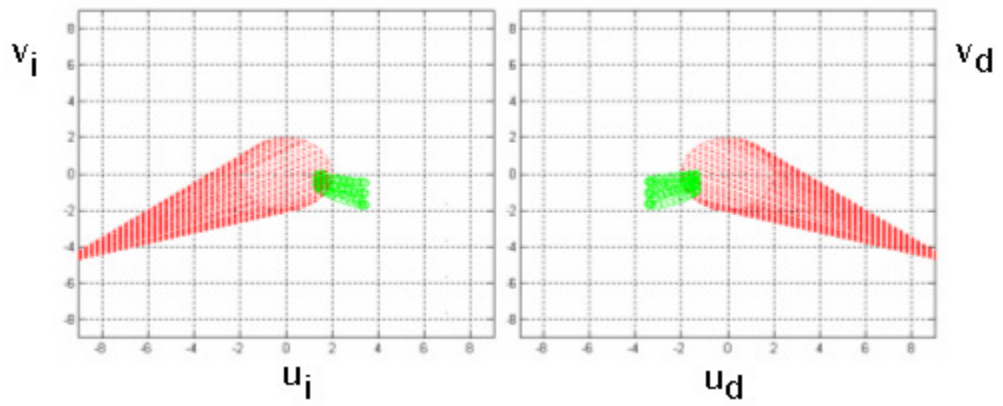


Figura 27. Imágenes cámara izquierda, cámara derecha, las líneas indican el plano epipolar

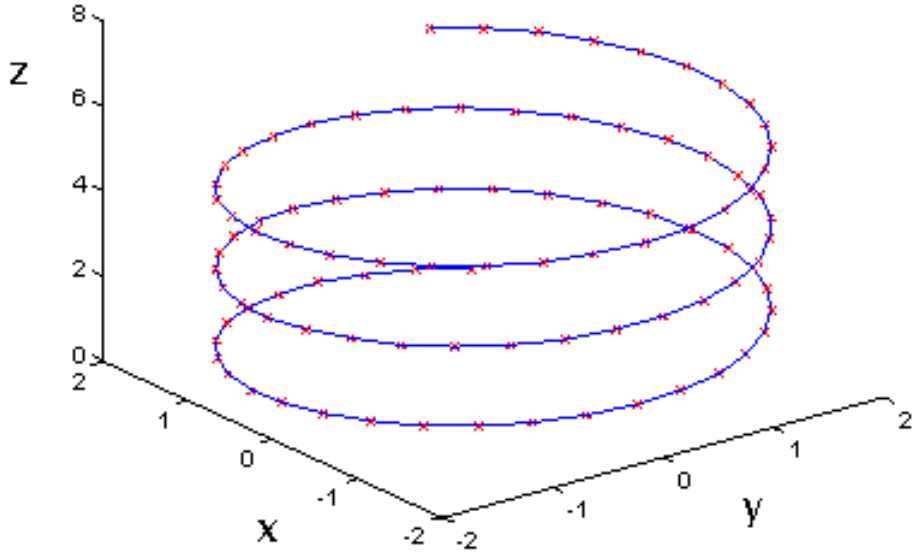


Figura 28. Trayectoria espacial cartesiana real (línea en cruz) y trayectoria espacial reconstruida (línea continua)

Capitulo 5a: Intercepción de objetos

Conociendo la posición de los objetos con las técnicas descritas en el capitulo previo, la siguiente tarea es interceptarlos. La intercepción se subdivide en:

1. Predecir el movimiento del objeto a capturar.
2. Posteriormente generar la trayectoria de captura.

5a.1 Predicción

En lo referente a la intercepción de objetivos, no es suficiente saber la posición del objeto, sino tener una trayectoria de su movimiento futuro, Fig. 29, pues el tener la posición actual solo serviría para realizar seguimiento pero no captura, existen 2 tipos de predicción, la determinista y la probabilística.

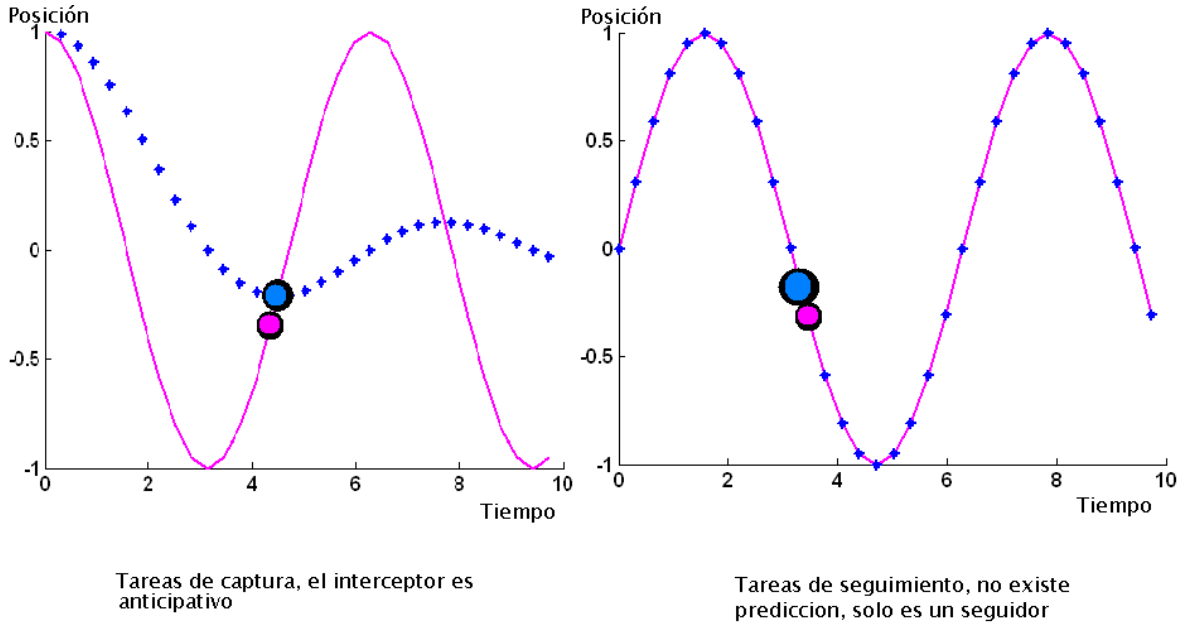


Figura 29. Tareas de Seguimiento vs Tareas de Captura, en ambos casos, la trayectoria del robot se representa con una línea punteada, la continua se refiere al objetivo.

5a.1.1 Determinista

Es la predicción de movimiento más simple y esta restringida al conocimiento a priori de las funciones que caracterizan el móvil, es útil en tareas repetitivas y predefinidas por ejemplo la toma de objetos sobre una banda transportadora o en tiro parabólico

ver Flores Campos [10], sin embargo se ve limitada si el objeto repentinamente frena o se acelera pues el evento calculado de llegada no sucede, la premisa es:

Dado un cuerpo o multiples objetos que se muevan describiendo trayectorias dependientes del tiempo *x=f(t)* previamente conocidas, con velocidad *v=x'* y aceleración *a=x''*, además de ciertos parámetros dinámicos (masas, inercias, friccion etc), se determina su posición futura evaluando en las ecuaciones un tiempo de deseado, realizado esto, se encuentra la ecuación polinomial x=p(t) tal que se calculen los coeficientes necesarios del polinomio de trayectoria a ese punto en ese instante, como puede verse, su mayor ventaja es la asignación de tiempos de captura.

5a.1.2 Determinista-probabilistica (filtro de kalman)

El filtro de Kalman es un algoritmo que sirve para identificar el estado oculto de un sistema lineal o linealizado, aunque presenta una parte determinista semejante a la técnica anterior, se diferencia por tener un proceso paralelo de generación probabilística de ganancias - P, K - y aun mas, tiene una etapa de corrección que actualiza los valores de predicción en función de los valores reales medidos, la mayor desventaja del filtro de Kalman es que no se tiene control sobre el tiempo de predicción (es decir predice, pero el usuario no tiene la capacidad de indicar cuando), una forma de solventar esto, como se verá más adelante es emplear derivación fraccional. Su versatilidad es que puede emplearse en la intercepción de múltiples objetos sin conocer las funciones que describen sus movimientos.

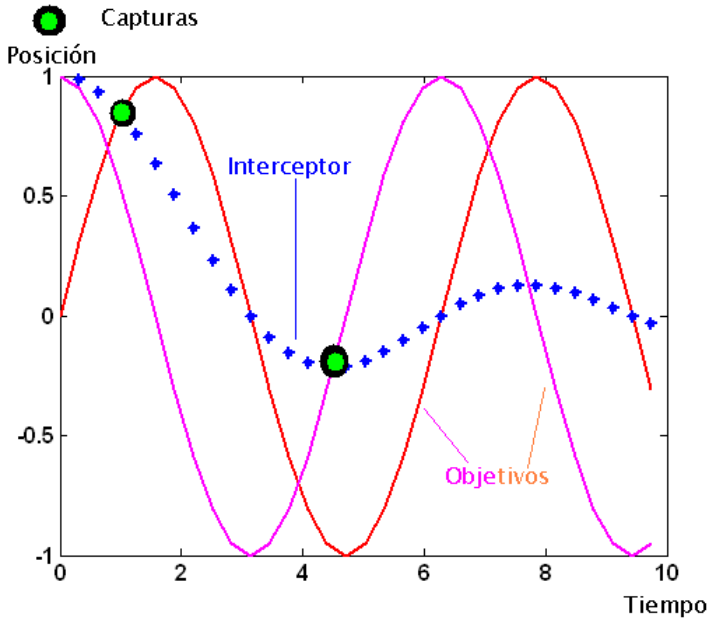


Figura 30. Trayectorias de múltiples objetos y trayectoria única de intercepción

Se exhibe a continuación el algoritmo lo cual se explica subsecuentemente

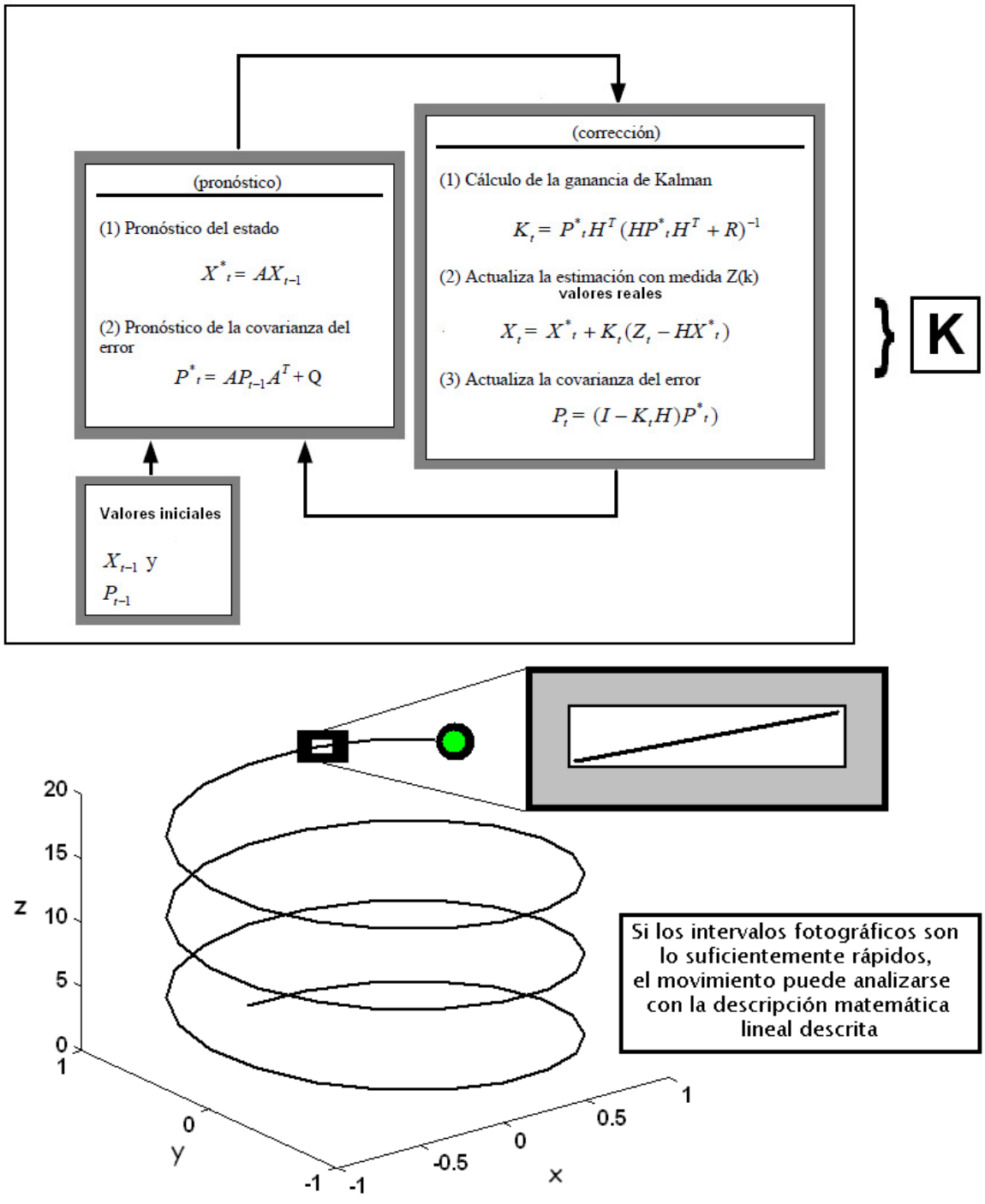


Figura 31. Superior, algoritmo del filtro lineal de Kalman, inferior, justificación del filtro lineal de Kalman en esta tesis, el subíndice t indica valor en tiempo actual, t-1 valor previo, X es el vector de estados del sistema, A es la matriz de transición de estados asociada al sistema, H es el vector de observación de estados, Z son los estados medidos P es el pronostico de la covarianza de error Q asociada al sistema, R es la covarianza de error asociada al observador, K es la ganancia de Kalman.

A continuación se expone como utilizar el filtro de Kalman para la intercepción de partículas, ver Fig. 31.

1) Modelar la cinematica del sistema: si la velocidad de los objetos a seguir es lenta, respecto al sistema de vision, se adopta un modelo de comportamiento lineal Fig. 31 , el objeto a monitorear es poseedor de masa y sobre este actua una fuerza, por tanto su dinamia es modelada por la segunda ley de Newton cuyas ecuaciónes diferenciales son de segundo orden con respecto al tiempo y entonces cada grado de libertad necesita ser modelado por al menos dos ecuaciones (en el espacio 3D hay 6GDL para una partícula).

El modelo adopta una forma simple por la suposición de linealidad y la geometría esférica de los móviles lo cual permite considerarlos puntuales e invariantes a rotacion, en caso opuesto, un movil es modelado por 12 ecuaciones lineales (es decir 6 para describir posición en x y z, adicional a otras 6 para la orientación en los ejes mencionados), a continuación se presentan 2 casos, el primero es con aceleración constante y el siguiente con aceleración lineal variable.

Caso aceleración constante

$$x = x_0 + v_{0x}t + a_x\frac{t^2}{2} \quad (13) \qquad v_x = v_{0x} + a_x t \quad (14)$$

$$y = y_0 + v_{0y}t + a_y\frac{t^2}{2} \quad (15) \qquad v_y = v_{0y} + a_y t \quad (16)$$

$$z = z_0 + v_{0z}t + a_z\frac{t^2}{2} \quad (17) \qquad v_z = v_{0z} + a_z t \quad (18)$$

Caso aceleración variable

$$x = x_0 + v_{0x}t + a_{ox}\frac{t^2}{2} + b_x\frac{t^3}{6} \quad (19) \qquad v_x = v_{0x} + a_{0x}t + b_x\frac{t^2}{2} \quad (20)$$

$$y = y_0 + v_{0y}t + a_{oy}\frac{t^2}{2} + b_y\frac{t^3}{6} \quad (21) \qquad v_y = v_{0y} + a_{0y}t + b_y\frac{t^2}{2} \quad (22)$$

$$z = z_0 + v_{0z}t + a_{oz}\frac{t^2}{2} + b_z\frac{t^3}{6} \quad (23) \qquad v_z = v_{0z} + a_{0z}t + b_z\frac{t^2}{2} \quad (24)$$

$$a_x = a_{0x} + bt \quad (25) \qquad a_y = a_{0y} + bt \quad (26) \qquad a_z = a_{0z} + bt \quad (27)$$

Donde x, y, z asi como los subíndices asociados indican movimiento en los ejes mencionados, v son las correspondientes velocidades, a las aceleraciones, b es la derivada temporal de la aceleración correspondiente, t es el tiempo y los subíndices 0 indican condiciones iniciales.

2) Establecer la representación de estados del sistema: El filtro de Kalman es un observador de estados con capacidad de predicción (estados son la mínima cantidad de parámetros que describen completamente la evolución de un sistema, en este caso 3 posiciones y 3 velocidades). Al igual que la mayoría de los controladores y observadores, su implementación requiere llevar al sistema a la representación matricial de espacio-estados [13], [14], en el caso de las ecuaciones de la partícula, esta se obtiene de la reescritura matricial de (13) a (18), esta es:

$$X_t = AX_{t-1} + BU \; ; \quad Z = HX \quad (28)$$

$$\begin{pmatrix} x \\ y \\ z \\ v_x \\ v_y \\ v_z \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & t & 0 & 0 \\ 0 & 1 & 0 & 0 & t & 0 \\ 0 & 0 & 1 & 0 & 0 & t \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_0 \\ y_0 \\ z_0 \\ v_{0x} \\ v_{0y} \\ v_{0z} \end{pmatrix} + \begin{pmatrix} \frac{t^2}{2} & 0 & 0 \\ 0 & \frac{t^2}{2} & 0 \\ 0 & 0 & \frac{t^2}{2} \\ t & 0 & 0 \\ 0 & t & 0 \\ 0 & 0 & t \end{pmatrix} \begin{pmatrix} a_x \\ a_y \\ a_z \end{pmatrix}$$

$$Z = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ v_x \\ v_y \\ v_z \end{pmatrix} \quad (29)$$

Donde X es el vector de estados en los tiempos actual t y previo t-1, A es la matriz de transición de estados de la partícula a observarse, H es el vector de observación de estados, Z son los estado medidos, B es la matriz de trancision de entrada y U el vector de entrada (en este caso aceleraciones constantes dependientes de una fuerza externa)

3) dotar al filtro de Kalman de condiciones iniciales de operación y predicción por ejemplo:

$$x_0 = 0 \qquad y_0 = 0 \qquad z_0 = 0$$

$$P_0 = 0_{nxn}$$

dado que A_{nxn} donde n es el numero de elementos del vector de estados.

4) entrenar heuristicamente, es decir a prueba y error las ganancias R y Q del filtro, aunque en [11] y [12] se sugieren formas para establecerlas.

5) ejecutarlo en paralelo al sistema de visión.

El diagrama de bloques para el filtro lineal de Kalman es el siguiente:

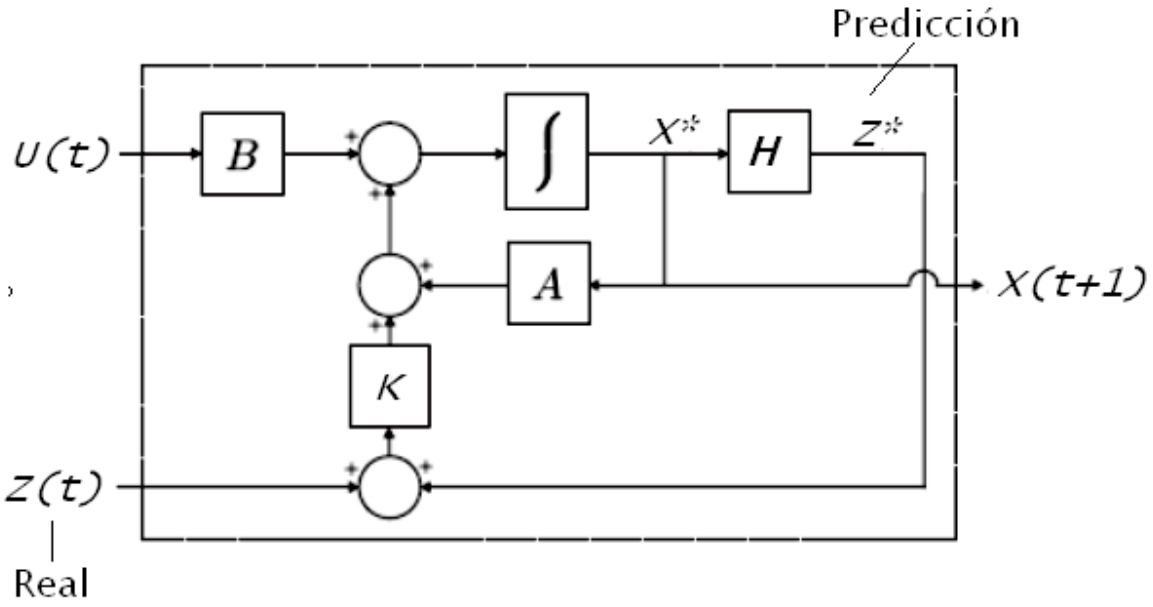


Figura 32 Representación de bloques de un observador genérico, en este caso K es el filtro de Kalman, modificado de [13].

En caso de usar sensores de captura lentos y dado que el filtro de Kalman en este caso es un predictor lineal ocurre el error por linealizacion mostrado en la Fig. 33

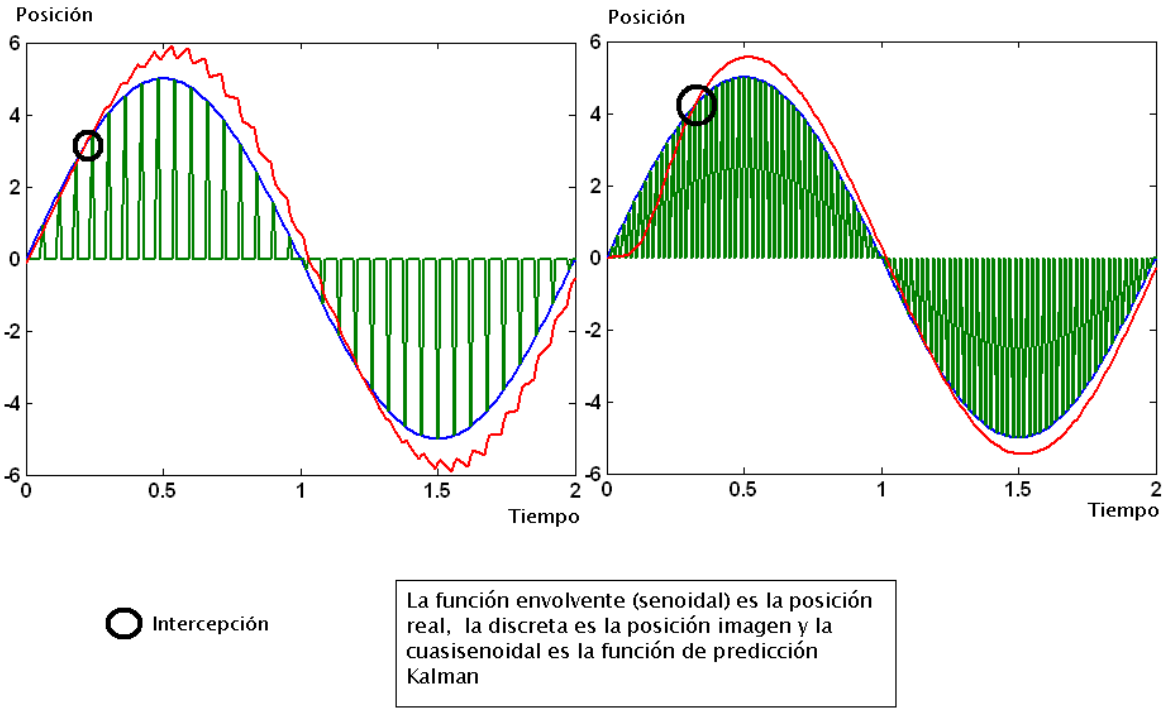


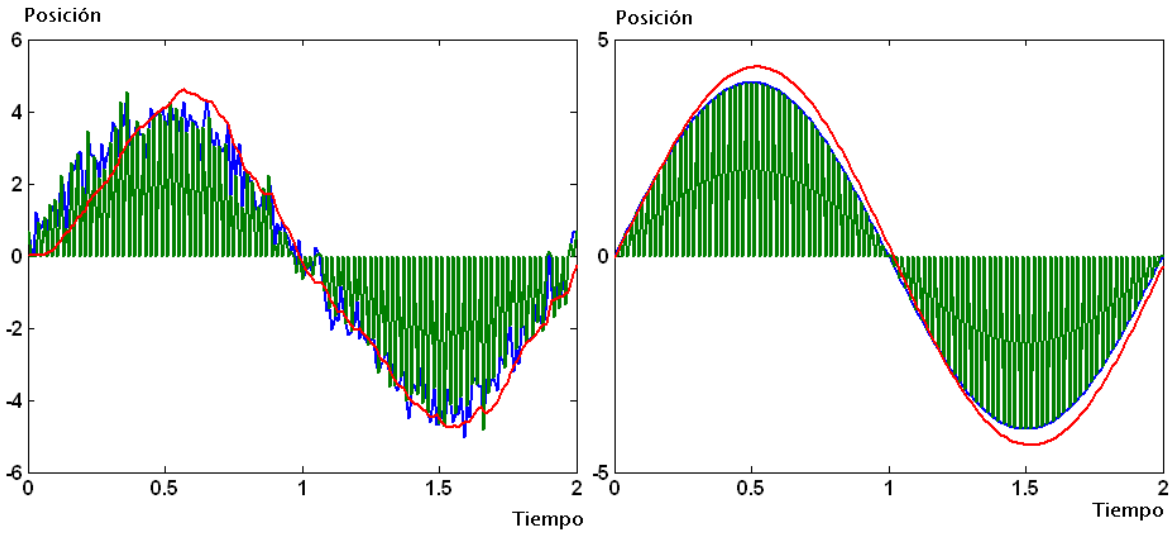
Figura 33 Izquierda señal de sensado lenta respecto al sistema (kalman con frecuencia lenta), derecha sensor de captura rapida respecto al sistema (kalman aplicado con frecuencia rápida)

Una sugerencia es que la frecuencia de actualización Kalman sea paralela e igual a la del tiempo de medición de un sensor rápido (es decir coordinar cada predicción de kalman con cada imagen).

Un método distinto al sugerido es usar el filtro no lineal o extendido de kalman (EKF) ver Apéndice A.1.

Un dato importante es que como el filtro de Kalman es una linealizacion, el grado de términos incluidos lo cual define la matriz A (30), lo vuelven o un seguidor o un predictor según sea el caso Fig. 35, Fig. 36

$$A = \begin{pmatrix} 1 & 0 & 0 & t & 0 & 0 \\ 0 & 1 & 0 & 0 & t & 0 \\ 0 & 0 & 1 & 0 & 0 & t \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \quad A = \begin{pmatrix} 1 & 0 & 0 & t & 0 & 0 & \frac{t^2}{2} & 0 & 0 \\ 0 & 1 & 0 & 0 & t & 0 & 0 & \frac{t^2}{2} & 0 \\ 0 & 0 & 1 & 0 & 0 & t & 0 & 0 & \frac{t^2}{2} \\ 0 & 0 & 0 & 1 & 0 & 0 & t & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & t & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & t \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{30}$$



Caso sensor de imagen con ruido (ultrasonidos)

Caso sensor de imagen estandar (cámaras)

La envolvente es la señal real, la discreta es la del sensor y la señal Kalman se aproxima a la envolvente y la filtra

Figura 34 Filtro de Kalman como seguidor y supresor de ruido

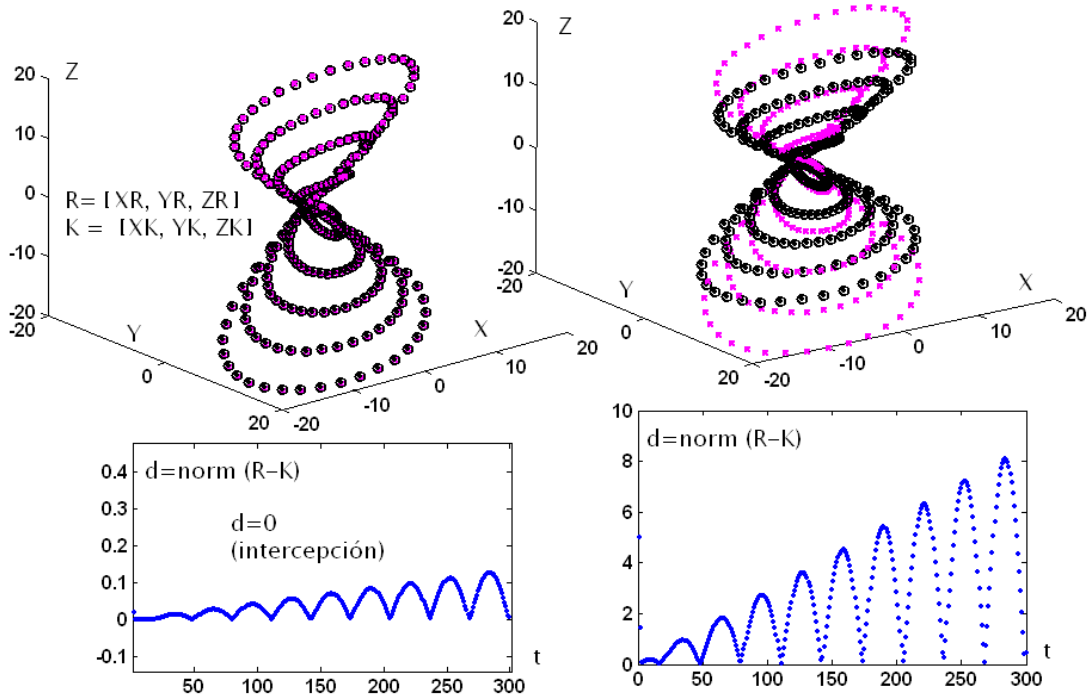


Figura 35 Predicción por Filtro de Kalman en 3D la intercepción ocurre en las desaceleraciones, Izquierda Matriz A con múltiples términos actuando como seguidor; Derecha Matriz A con menos términos: actuando como predictor, los círculos huecos son la predicción Kalman, los rellenos son la posición espacial real de la partícula, se incluyen las graficas de distancia entre la posición Kalman y la real

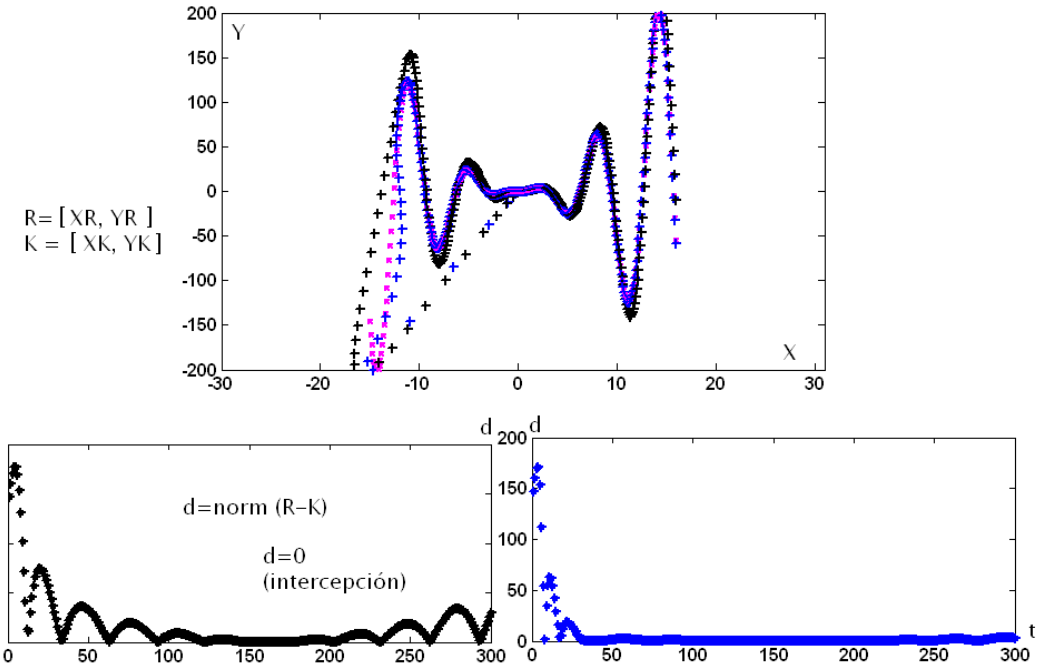


Figura 36 Filtro de Kalman sensible a velocidad contra filtro de kalman sensible a aceleración caso 2D, las señales Kalman en ambos casos proviene del origen, la señal real inicia en el punto (-15,-150)

Como puede verse en la Fig. 35, Fig. 36 y ecuación (30) el orden de la derivada a emplearse equivale a un termino de linealizacion Taylor, y pasa drásticamente de ser observador de predicción a observador de seguimiento con la adición de un solo termino derivativo, luego entonces, una técnica propuesta para regular el tiempo de predicción es el uso de derivadas fraccionarias [47].

5a.2 Captura

Una vez que se posee información de la trayectoria de intercepción (trayectoria futura generada por el filtro de Kalman), se requiere desplazar el móvil en el recorrido mencionado (por ejemplo un misil, un vehículo o un robot manipulador), existen muchas técnicas para llevarlo a cabo, las hay en control analítico (por ejemplo: PID, control por modos deslizantes, controles adaptables etc) y/o difuso (por ejemplo: árboles de decisiones, celdas neurodifusas, modelos de primitivas básicas de imagen, modelos de bordes, redes neuronales etc),

En esta tesis, el controlador empleado es el de potenciales artificiales [2] y es así por dos motivos.

1. Esta técnica de control al aplicarse en robots manipuladores no se realiza en el espacio articular, por lo tanto no suceden condiciones denominadas singularidades ya que no requiere la cinemática inversa del sistema para su puesta en marcha Fig. 4, esto se profundiza en el siguiente capitulo.

2. Este controlador permite generar perfiles de Fuerza basados en fenómenos físicos de cohesión-evasión y teoría de campos vectoriales, esto lo vuelve idóneo para interactuar con múltiples objetos pues cada cuerpo contribuye a la totalidad del movimiento tal como lo haría un sistema de resortes o de partículas eléctricas por ejemplo.

[2] *El criterio de estabilidad de Lyapunov referente a este controlador, se exhibe en el capitulo 6*

5a.2.1 Control por potenciales artificiales

Un controlador por potenciales artificiales es a grandes rasgos un control PD (proporcional-diferencial) múltiple cuya función error se elige de tal forma de dotar al sistema interceptor con un comportamiento especifico de atracción-repulsión, opera en el espacio de trabajo (x,y,z) por lo que la función de error a elegirse es de base X-Xd, se dice que es múltiple por que cada obstáculo y objetivo agregan su propio termino de error a la parte proporcional del control, antes de ahondar en su descripción se presentan algunos conceptos:

1. Campo Vectorial:

Un campo vectorial f es una construcción del cálculo vectorial que asocia un vector a cada punto en el espacio euclideano, por ejemplo, dado un punto tridimensional (x,y,z), se tiene un perfil de fuerza también tridimensional (Fx,Fy,Fz) que apunta de (x,y,z) hacia algún otro punto espacial (x0,y0,z0) .

$$f : \mathrm{R}^n \rightarrow \mathrm{R}^n \quad (31)$$

2. Tipos de Campo Vectorial:

Como puede verse en la Fig. 37 una clasificación es relativa al perfil de movimiento que generan: los hay convergentes y divergentes, rotacionales y laminares de tipo paralelo o perpendicular, por mencionar algunos de los más usados en robótica , su definición matemática formal puede hallarse en [30], sus correspondientes graficas representan líneas de fuerza

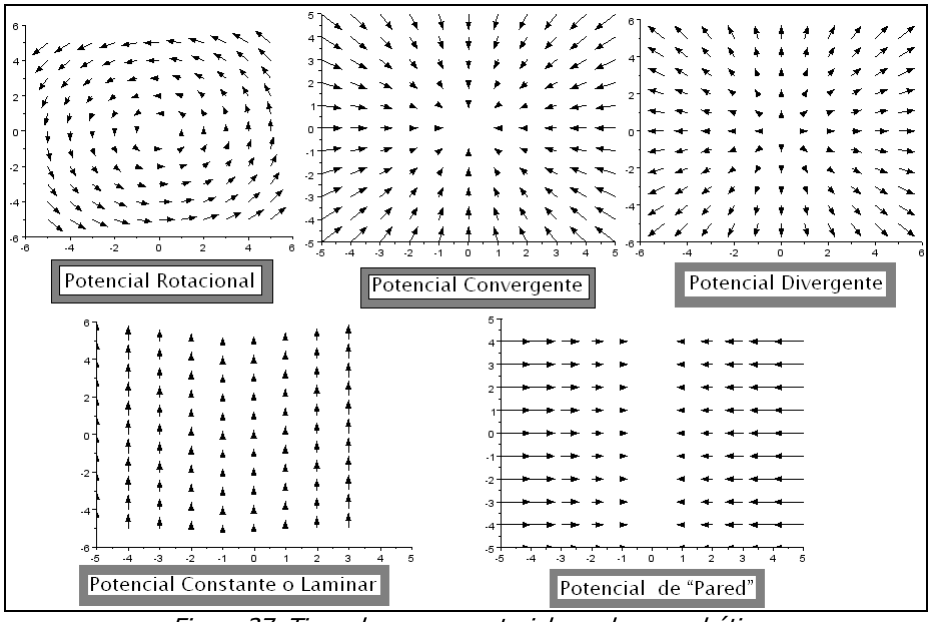


Figura 37. Tipos de campo vectorial usuales en robótica

3. Selección de Campos de Fuerza variantes en Magnitud, Dirección y Sentido:

Los campos vectoriales tienen múltiples aplicaciones, en particular se usan para modelar la intensidad y dirección de un campo de fuerzas, esto puede hacerse de forma analítica siempre que exista una función escalara derivable, en caso opuesto como en los de tipo rotacional, el calculo se realiza de forma numérica [30], mencionada función es la energía potencial de la fuerza atractiva o repulsiva.

De acuerdo a la tarea a realizarse existen variaciones en un mismo perfil de campo vectorial Fig. 38, Fig. 39, Fig. 40.

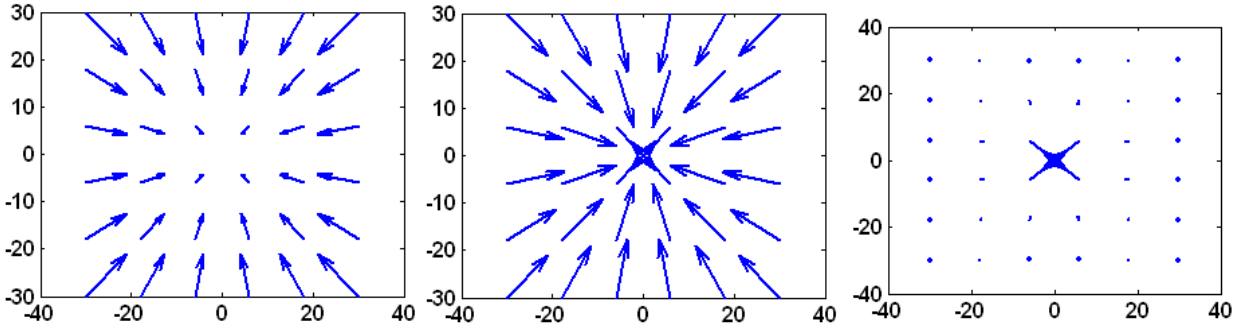


Figura 38. Campo vectorial variante en magnitud, Izquierda: magnitud decreciente a 0, Resorte Central: magnitud constante al centro, Resorte Normalizado Derecha magnitud creciente al centro Carga Eléctrica Atractiva

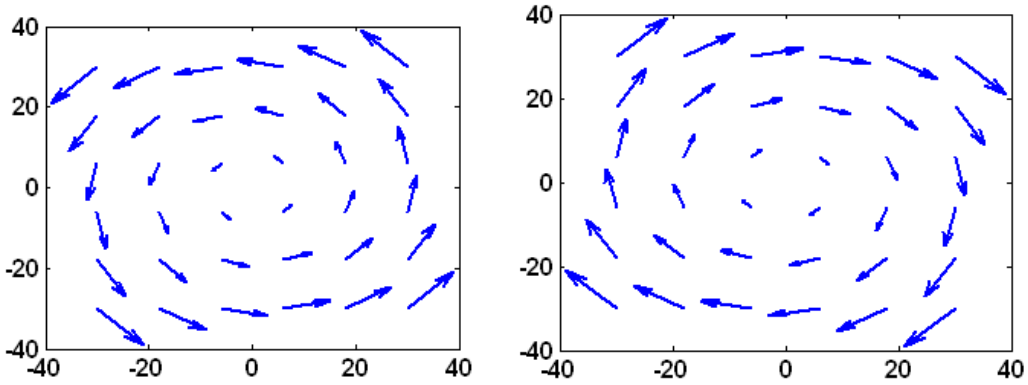


Figura 39. Campo vectorial variante en sentido, Izquierda: giro a contrarreloj, Derecha giro en dirección horaria

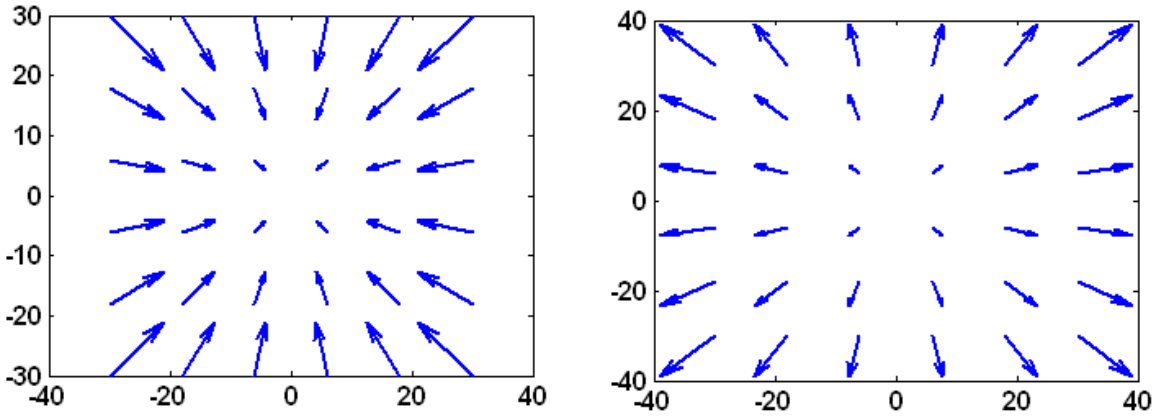


Figura 40. Campo vectorial variante en dirección,
Izquierda: campo vectorial atractor, Derecha: campo vectorial repulsor

Un ejemplo es el siguiente: se desea realizar una tarea de intercepción de objetos

La primer aseveración es una trayectoria corta, pensando en la línea recta, se escoge e impone un primer perfil de potencial: no rotacional

De las posibilidades mencionadas resulta evidente que si deseamos llegar a algún punto específico debe ser de tipo convergente (atractor)

Se desea llegar de un modo tal que en el instante de captura ya no existan fuerzas de movilidad (en caso opuesto se generaría chattering), de esa forma se concluye que el campo vectorial idóneo es aquel de tipo irrotacional, convergente y decreciente en magnitud al punto de llegada (origen)

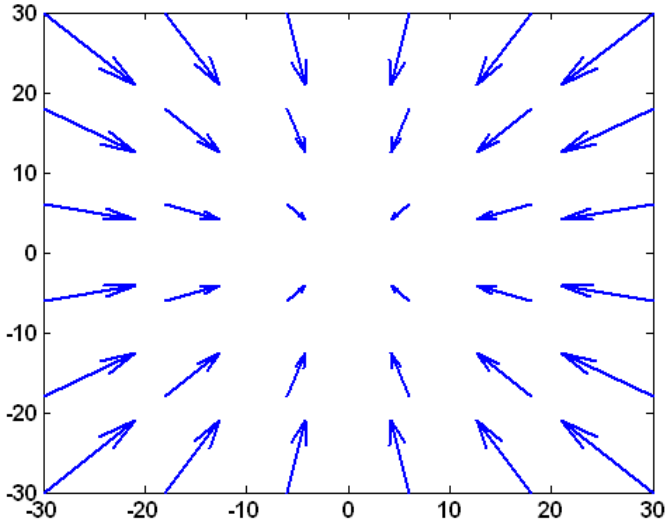


Figura 41. Campo vectorial para tareas de intercepción

Una función potencial que al derivarse genera este tipo de fuerzas es:

$$f = -\frac{k}{2}(x^2 + y^2 + z^2)$$
$$\nabla f(x, y, z) = < -kx, -ky, -kz > \quad (32)$$

La cual está asociada al comportamiento de un resorte lineal en el espacio 3D y es el potencial atractor utilizado en este trabajo Fig. 41, k es un factor multiplicativo que indica la magnitud de la fuerza.

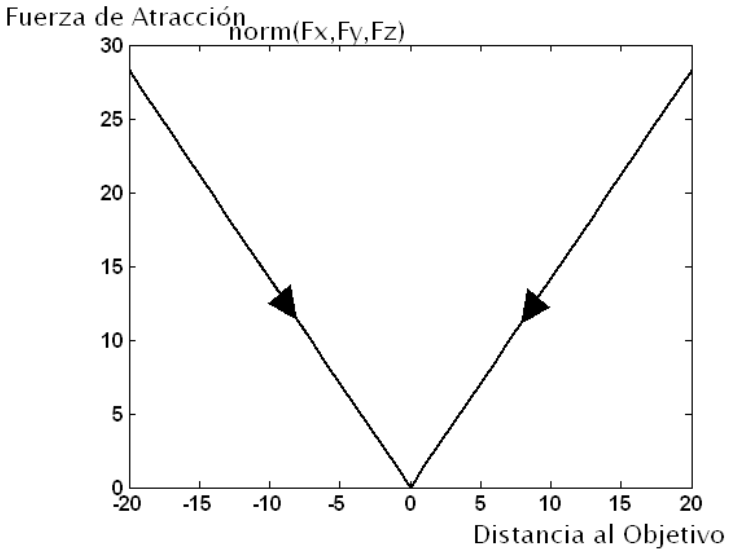


Figura 42. Perfil de Fuerza-Distancia asociado al campo vectorial de intercepción utilizado

5a.3 Discusión

El mayor problema para aplicar el Filtro de Kalman lineal en la predicción del movimiento es que si los sensores no son rápidos, el sistema a monitorear deja de ser lineal y debe emplearse el Filtro Extendido de Kalman, cuyos operadores no actúan sobre funciones conocidas sino en trayectorias punto a punto y por tanto se requiere el uso de métodos numéricos complicados y en consecuencia tiempo de procesamiento computacional, aunado al hecho de que la predicción ocurre pero no se tiene control sobre el tiempo de predicción, en el caso de esta tesis es suficiente el filtro lineal de Kalman dada la velocidad de captura y procesamiento de imágenes.

El problema principal para definir fuerzas de potencial artificial es que la metodología consiste en: dado el perfil geométrico de las líneas de flujo, hallar el campo vectorial que las genera, es decir: dada una familia de soluciones encontrar la ecuación diferencial que las genera esto, en la mayoría de las ocasiones no presenta soluciones analíticas o el proceso para encontrarlas requiere algoritmos numéricos extensos y por tanto tiempo de procesamiento.

El proceso de predicción captura precisa de los valores de reconstrucción espacial de las cámaras $[X_c, Y_c, Z_c]$, luego se obtienen valores de predicción, en este caso filtro de Kalman $[X_k, Y_k, Z_k]$ y finalmente Fuerzas de trayectoria en función a los valores de predicción, en este caso por Potenciales artificiales de Atracción $[FA_x(X_k, Y_k, Z_k), FA_y(X_k, Y_k, Z_k), FA_z(X_k, Y_k, Z_k)]$

Capitulo 5b: Evasión de objetos

5b.1 Monitoreo

Para evadir objetos no es tan relevante el predecir la posición sino tener certeza de ella en su estado presente, de hecho no se evade al obstáculo, se evade a un obstáculo aumentado dimensionalmente (distancia de seguridad), la cual se propone debido a la inercia propia del interceptor (puede verse como una “predicción” indirecta)

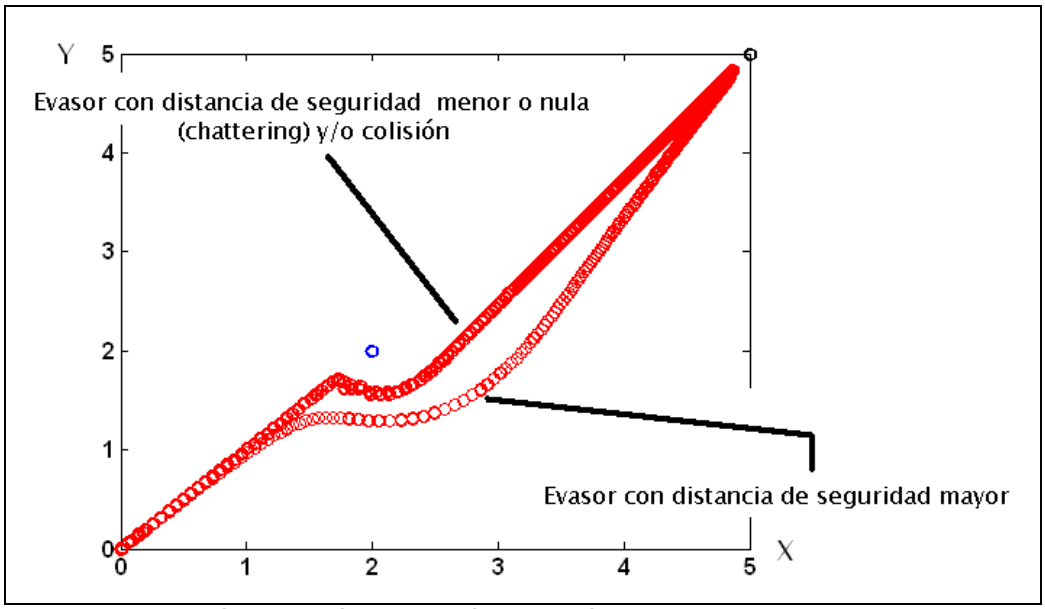


Figura 43. Evasión del obstáculo, Evasión del obstáculo aumentado (distancia de seguridad)

Una opción es usar el filtro de kalman no como predictor sino como un filtro seguidor de la señal de medición de la posición objeto en combinación con un extrapolador para generar mayor resolución temporal del sensor y reducir el ruido (principio de operación del zoom digital), esto se logra imponiendo una matriz A de alta ganancia lo cual equivale a seguir la señal en lugar de predecirla (esto se hace por ejemplo en radares que al igual que las cámaras miden posición pero conllevan altas señales de ruido) ver Fig. 34.

5b.2 Acción evasiva y potenciales artificiales libres de mínimos locales

Una vez conocida la posición del obstáculo es necesaria la acción evasiva, para ello se procede a elegir un campo potencial siguiendo la metodología del capitulo previo.

Existen dos tipos de campos evasores, los divergentes y los rotacionales repulsores, para mostrar por que ocurren los mínimos locales se elige en primer instancia un campo vectorial divergente no rotacional.

El siguiente punto es la magnitud de la fuerza, en esta tarea basta con generar una fuerza mayor entre menor sea la distancia al objeto, es decir uno con magnitud creciente al centro.

Una función generadora de este tipo de campos es la de cargas electrostáticas de repulsión Fig. 44, nuevamente k es un factor de amplitud de fuerza.

$$f = -k(x^2 + y^2 + z^2)^{-\frac{1}{2}}$$

$$\nabla f(x,y,z) = < \frac{kx}{\left(x^2+y^2+z^2\right)^{\frac{3}{2}}}, \frac{ky}{\left(x^2+y^2+z^2\right)^{\frac{3}{2}}}, \frac{kz}{\left(x^2+y^2+z^2\right)^{\frac{3}{2}}} > \quad (33)$$

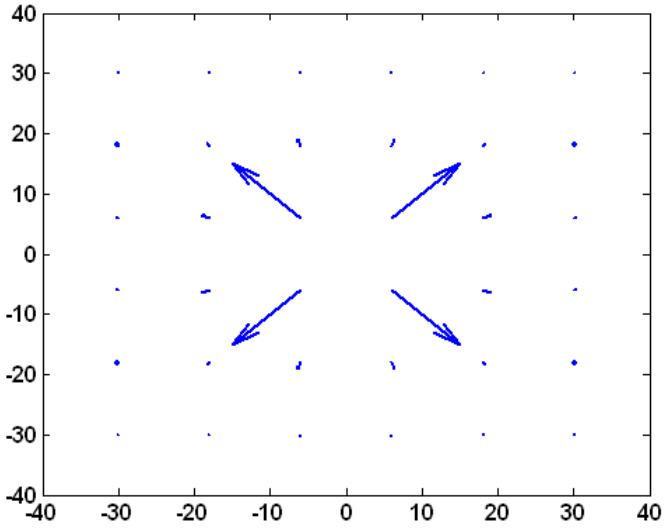


Figura 44. Campo vectorial para tareas de evasión

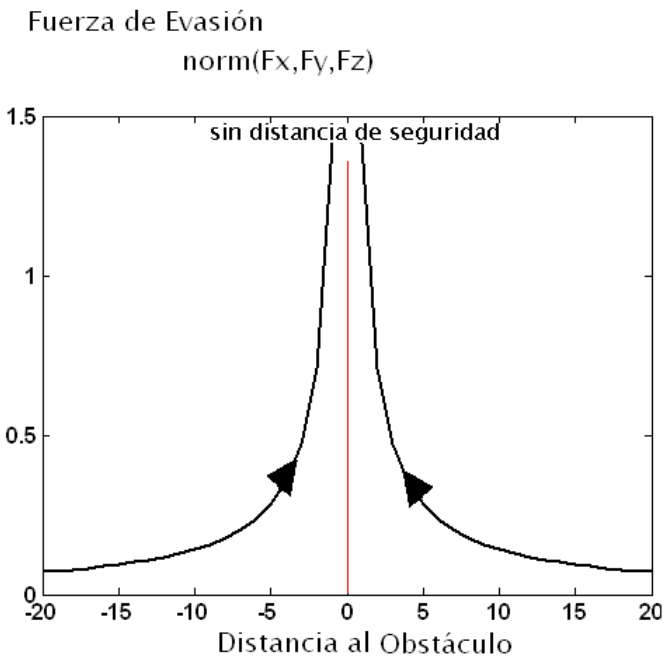


Figura 45. Perfil de Fuerza-Distancia asociado al campo vectorial de evasión del fenómeno de repulsión de cargas electrostáticas

Como se ve en la Fig. 45, el primer problema de esta función es que existe un crecimiento abrupto de la fuerza de repulsión (división entre cero) al aproximarse el interceptor al obstáculo, una forma de sortear esta indeterminación es agregando la mencionada distancia de seguridad, Kathib[18][39], lo cual limita la fuerza de evasión:

$$\rho = \sqrt{(x - x_{ob})^2 + (y - y_{ob})^2 + (z - z_{ob})^2}$$

$$f = \begin{cases} \frac{k}{2}\left(\frac{1}{\rho} - \frac{1}{dseg}\right)^2 & \text{si } \rho \le dseg \\ 0 & \text{si } \rho > dseg \end{cases}$$

$$\nabla f(x, y, z) = \begin{cases} \frac{k}{2}\left(\frac{1}{\rho} - \frac{1}{dseg}\right)\left(\frac{1}{\rho^3}\right) < x - x_{ob}, y - y_{ob}, z - z_{ob} > & \text{si } \rho \le dseg \\ 0 & \text{si } \rho > dseg \end{cases} \quad (34)$$

Donde dseg es la mencionada distancia de seguridad

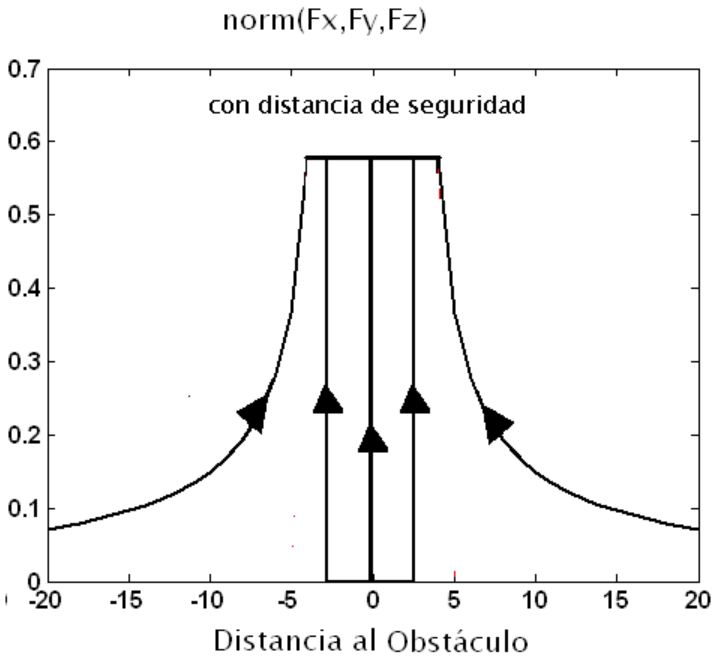


Figura 46. Perfil de Fuerza-Distancia asociado al campo vectorial de evasión del fenómeno de repulsión de cargas electrostáticas modificado.

El segundo problema es que al interactuar con las fuerzas de atracción asociadas a los objetivos, ocurren mínimos locales por efectos de cancelación de Fuerzas.

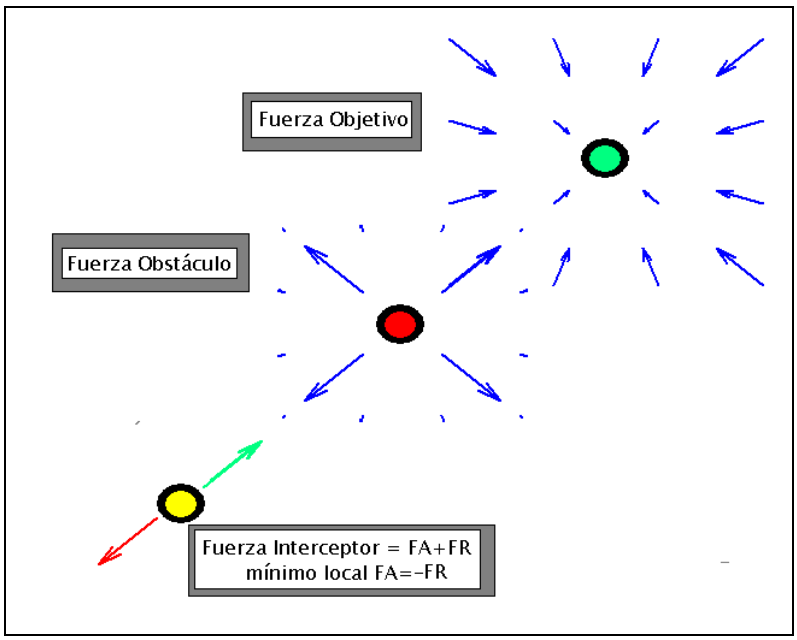


Figura 47. Mínimos locales

Un método para evitar mínimos locales consiste en emplear fuerzas tangenciales (campos rotacionales), el problema es que al no contar con fuerzas normales de divergencia es propicio un choque, esto hace pensar entonces en un campo rotacional-divergente (fuerza con componente normal y tangencial) Fig. 48, la forma de lograr esto es multiplicar una matriz de rotación (cuyo ángulo indica la proporción rotacional y la repelente) con el vector de campo de fuerzas repulsor, un ejemplo es el siguiente campo de fuerzas mitad repulsor, mitad rotacional (ángulo de 45º)

Usando el potencial definido en la ecuación (30) caso 2D

$$\nabla f(x,y)_{ROT} = R\nabla f(x,y)$$

Con R igual a una matriz de rotación

$$R = \begin{bmatrix} \cos\frac{pi}{4} & -\sin\frac{pi}{4} \\ \sin\frac{pi}{4} & \cos\frac{pi}{4} \end{bmatrix} \quad (35)$$

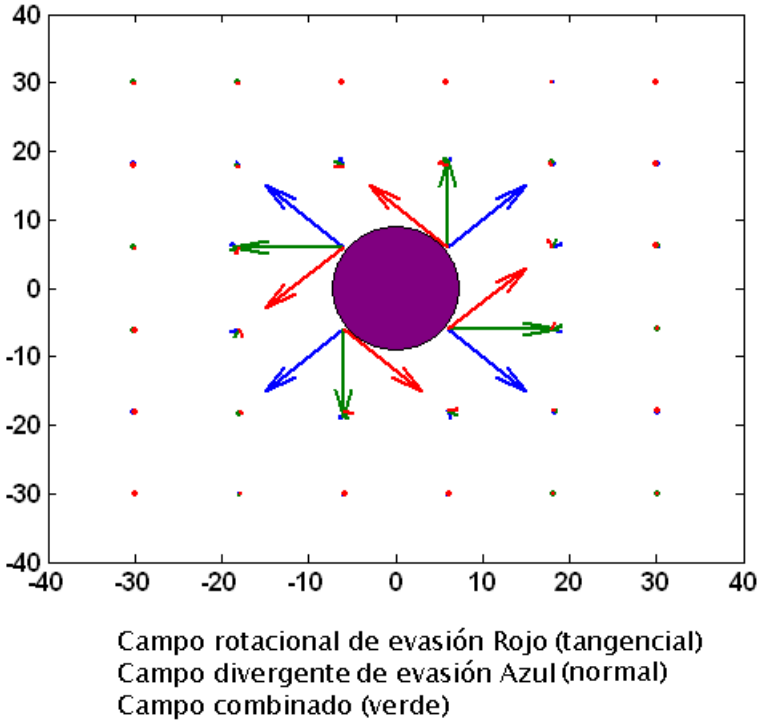


Figura 48. Campo vectorial para tareas de evasión modificado, parte rotacional, parte divergente.

Si se cambia el modelo x-y a un modelo polar, podemos advertir que nos encontramos ante un mismo campo divergente, de hecho cuando la condición del referente angular

medido respecto a la componente vertical es 0º nos encontramos frente al campo divergente cartesiano, el modelo generalizado para este campo divergente-rotacional es:

$$\nabla f(x,y)_{ROT} = R\nabla f(x,y)$$
$$R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \quad (36)$$

Como puede notarse se puede hacer que el componente rotacional aumente conforme se acerca el objeto teniéndose un campo repulsor inteligente.

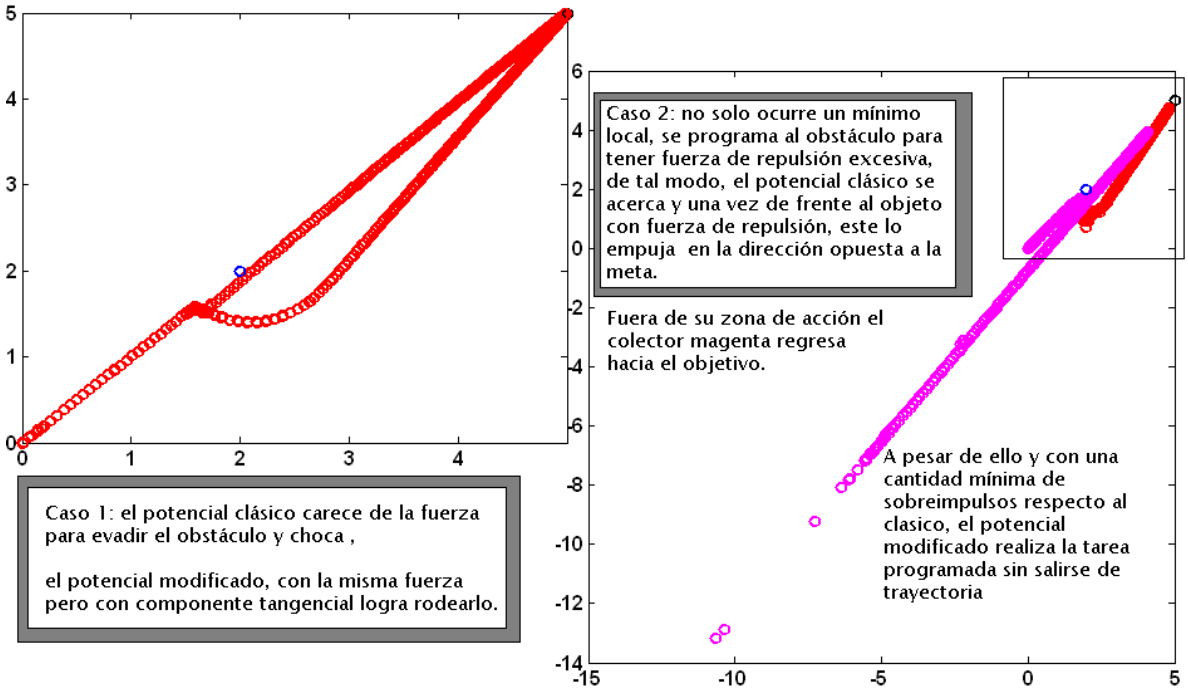


Figura 49. Desempeño y salida del mínimo local.

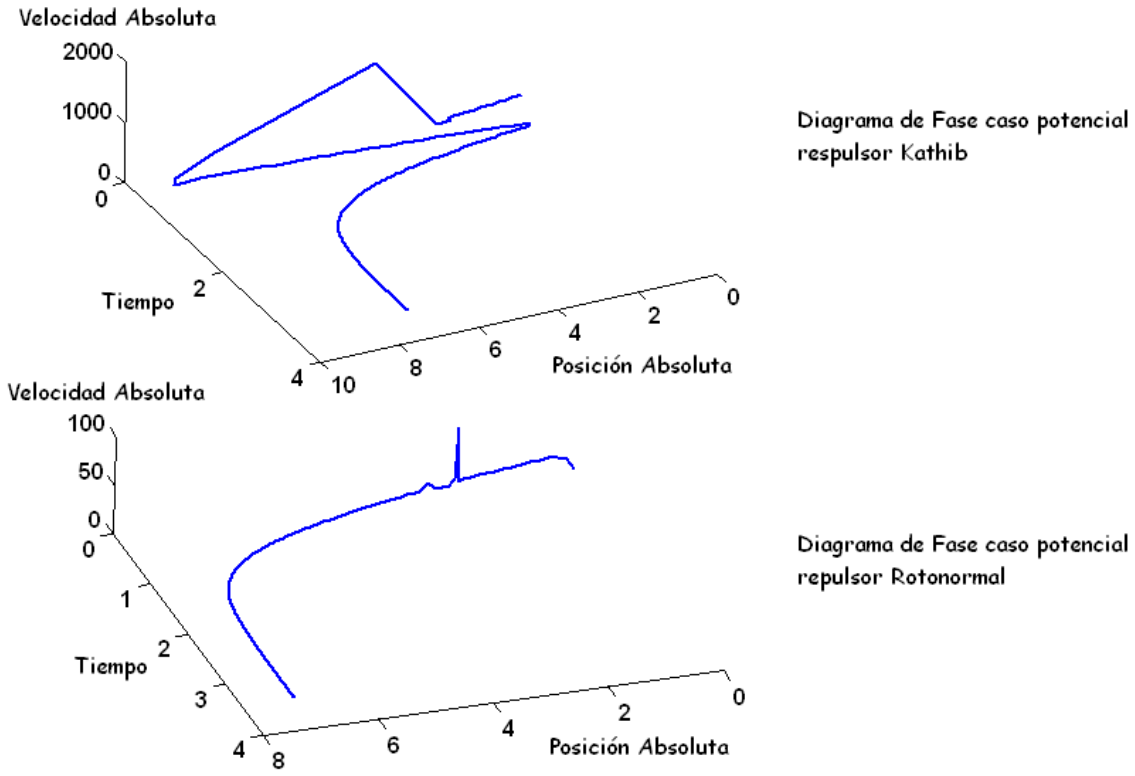


Figura 50. Diagramas de Fase de la Fig. 49 respecto a la interacción con ambos tipos de repulsores.

De tal manera, cuando las fuerzas normales se cancelan (mínimo local), las componentes tangenciales permiten salir de el rodeándolo Fig. 49, Fig. 50.

Un segundo método para evitar mínimos locales, es usar el comportamiento ideal del flujo laminar Fig. 51, es decir: aunada a la presencia del potencial objetivo (atractor) y los evasivos (obstáculos), se introduce al sistema de fuerzas virtuales una tercer fuerza que describe un potencial constante (laminar) en dirección al punto atractor (se obtiene el vector normal entre el interceptor y el objetivo y se envía una fuerza constante en tal dirección), esto equivaldría a colocar múltiples objetivos rodeando al atractor principal, al haber un obstáculo, el campo de fuerza de tal impedimento en combinación con el potencial laminar generaría una trayectoria divergente rodeando la geometría del cuerpo a evadir, por otra parte un atractor lo suficientemente fuerte equivaldría a introducir una cascada que hace converger todos los flujos en su periferia:

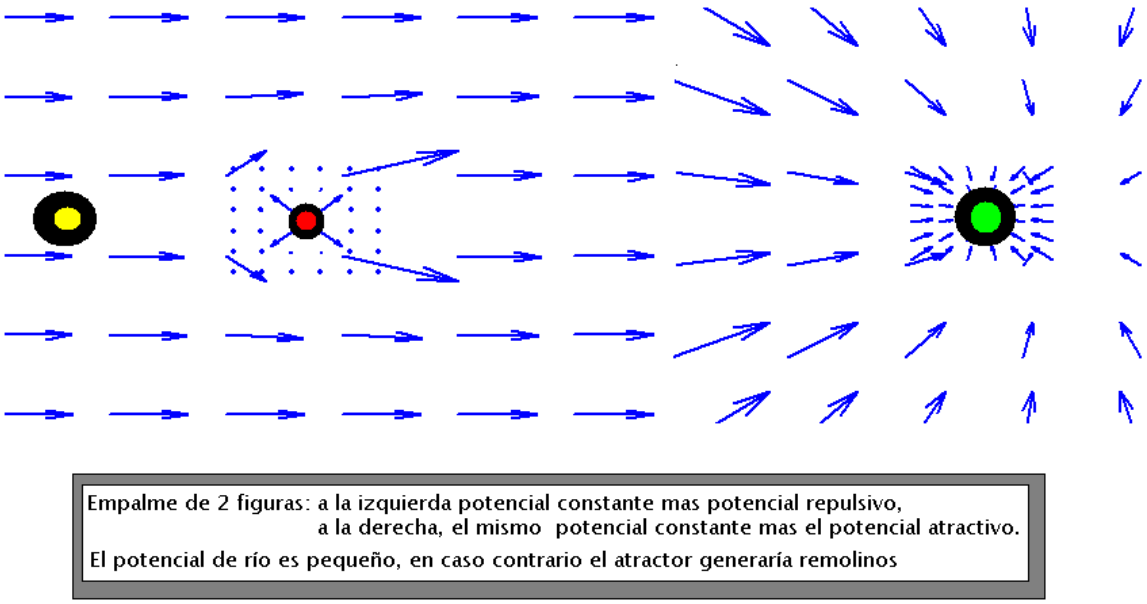


Figura 51. Diseño del potencial de flujo laminar ("río de fuerzas")

$$\nabla f(x,y,z)_{ATRACCION} + \nabla f(x,y,z)_{REPULSION} + \nabla f(x,y,z)_{CONSTANTE}$$

$$\nabla f(x,y,z)_{CONSTANTE} = -K\frac{e}{norm(e)}$$

$$e =< x - x_{objetivo}, y - y_{objetivo}, z - z_{objetivo} > \quad (37)$$

Como es de esperarse, la única forma de que existan mínimos locales es que los obstáculos bloqueen el "cause" del "río" Fig. 52 es decir la zona de acción del flujo laminar, pero esta posibilidad es mínima ya que el potencial laminar tiene la particularidad de cambiar su ruta en cada cambio posicional ya sea del atractor o del capturador (una tubería o un río de dirección dinámica respecto al vector normal interceptor-meta)

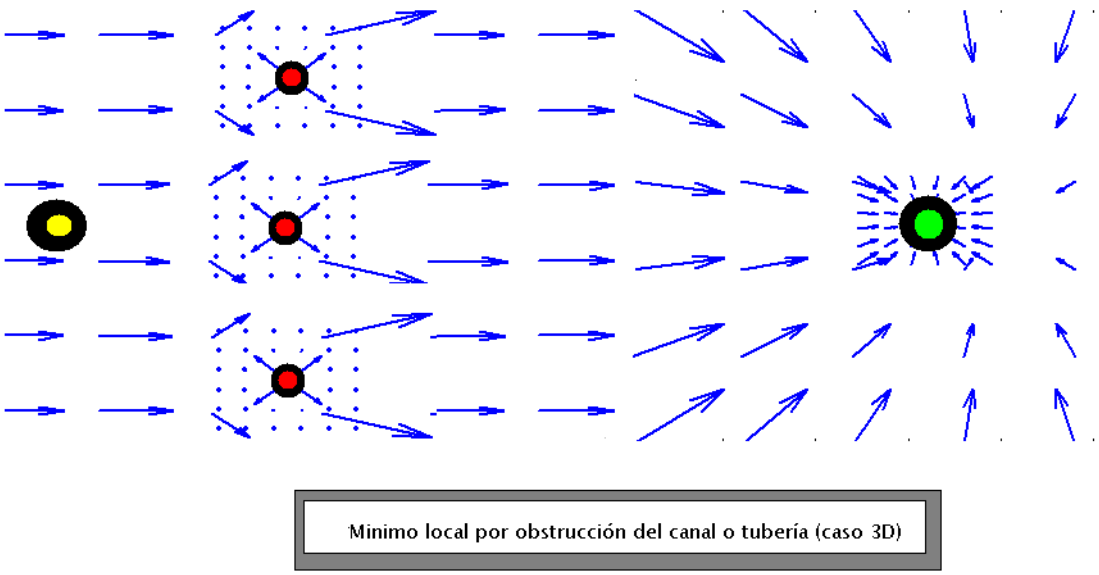


Figura 52. Posible cancelación de fuerzas ("muro de obstáculos"), aunque como el canal no tiene restricciones físicas (dureza), el "fluido" rodearía los obstáculos y una vez fuera proseguiría al objetivo

Ahora bien, no es que dejen de existir mínimos locales, lo que sucede es que visualizando la tarea de intercepción-evasión con una analogía geográfica Fig. 53 , el efector partiendo desde la cima llegara a la meseta u objetivo, los obstáculos equivaldrían a colocar cerros en el trayecto y un mínimo local resultaría en un valle ocurriendo antes de llegar al destino principal, pues bien, este al ser un potencial de flujo de líquidos, aguardaría en la meseta del mínimo local hasta desbordarse de ella y continuar su trayecto al atractor.

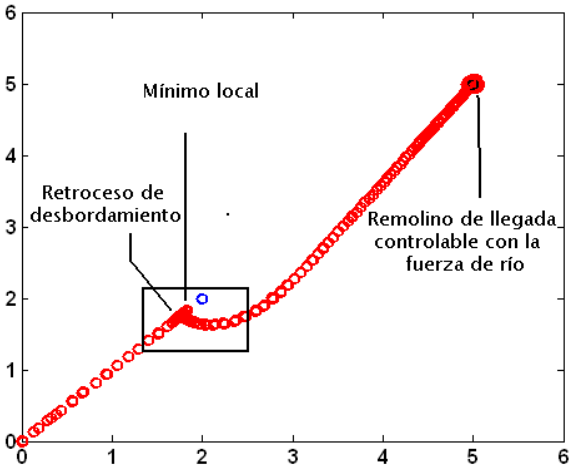


Figura 53. Analogía geográfica del campo potencial artificial con comportamiento fluidico y ubicación de un mínimo local, desbordamiento del mínimo local y continuación hacia el punto destino izquierda, derecha ejemplo visto en una tarea 2D

Un inconveniente de esta generación de fuerzas artificiales es el posible efecto remolino Fig. 54, ocurrido al presentarse una gran fuerza de río, el hecho es que el atractor al ser un sumidero, genera fuerzas que en oposición al caudal ocasiona efectos “vortex”, por lo cual es practico un río con poca fuerza y creciente en tal magnitud ante la presencia de un obstáculo

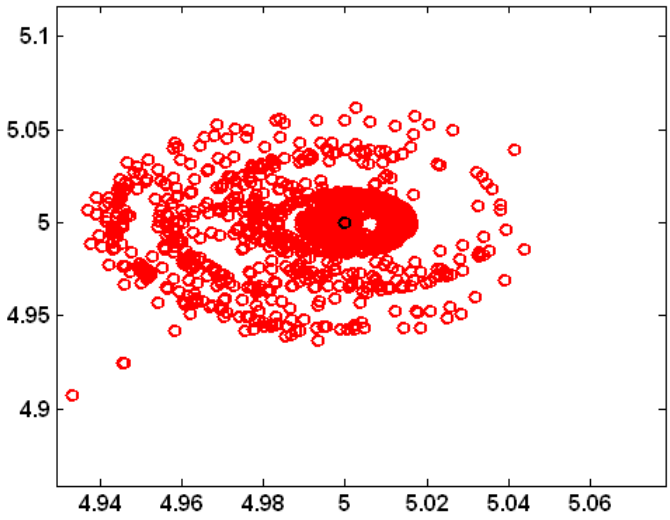


Figura 54. Remolino asociado al potencial de flujo laminar

5b.3 Discusión

En el presente capitulo se presentaron dos formas de solventar los llamados “mínimos locales”, de ambos métodos el utilizado en este trabajo es el rotonormal, este se utilizo por su carencia de chattering (vortex de llegada presente en el de rio).

También se muestra la realización de la tarea conjunta evasión-intercepción, en todos los casos se presupone que el interceptor contiene un torque tal que pueda compensarse su inercia Fig. 55 la cual si es excesiva conduciría a un choque, una propuesta a este punto es utilizar un control de fuerza (no de movimiento, sino de cantidad de fuerza aplicada en el interceptor) y esto puede ser posible en el potencial de flujo laminar modelando la trayectoria con la ecuación de Navier-Stokes lo cual permite añadir términos de viscosidad variable al sistema “hidrodinámico”, esto, debido a que la ecuación referida implica no solo términos gradiente (en este caso fuerzas), sino también términos laplaciano (en este caso derivada de fuerzas respecto al tiempo).

En el siguiente capitulo se presenta la forma de incorporar las técnicas descritas a un manipulador robótico de base fija y cadena cinemática abierta

El proceso de monitoreo y evasión requiere de los valores de reconstrucción espacial de las cámaras $[X_c, Y_c, Z_c]$, o de los valores del filtro de Kalman en su modo de seguidor-reductor de ruido $[X_k, Y_k, Z_k]$, esto con la finalidad de obtener fuerzas de evasión por Potenciales artificiales $[FE_x(X_c, Y_c, Z_c), FE_y(X_c, Y_c, Z_c), FE_z(X_c, Y_c, Z_c)]$

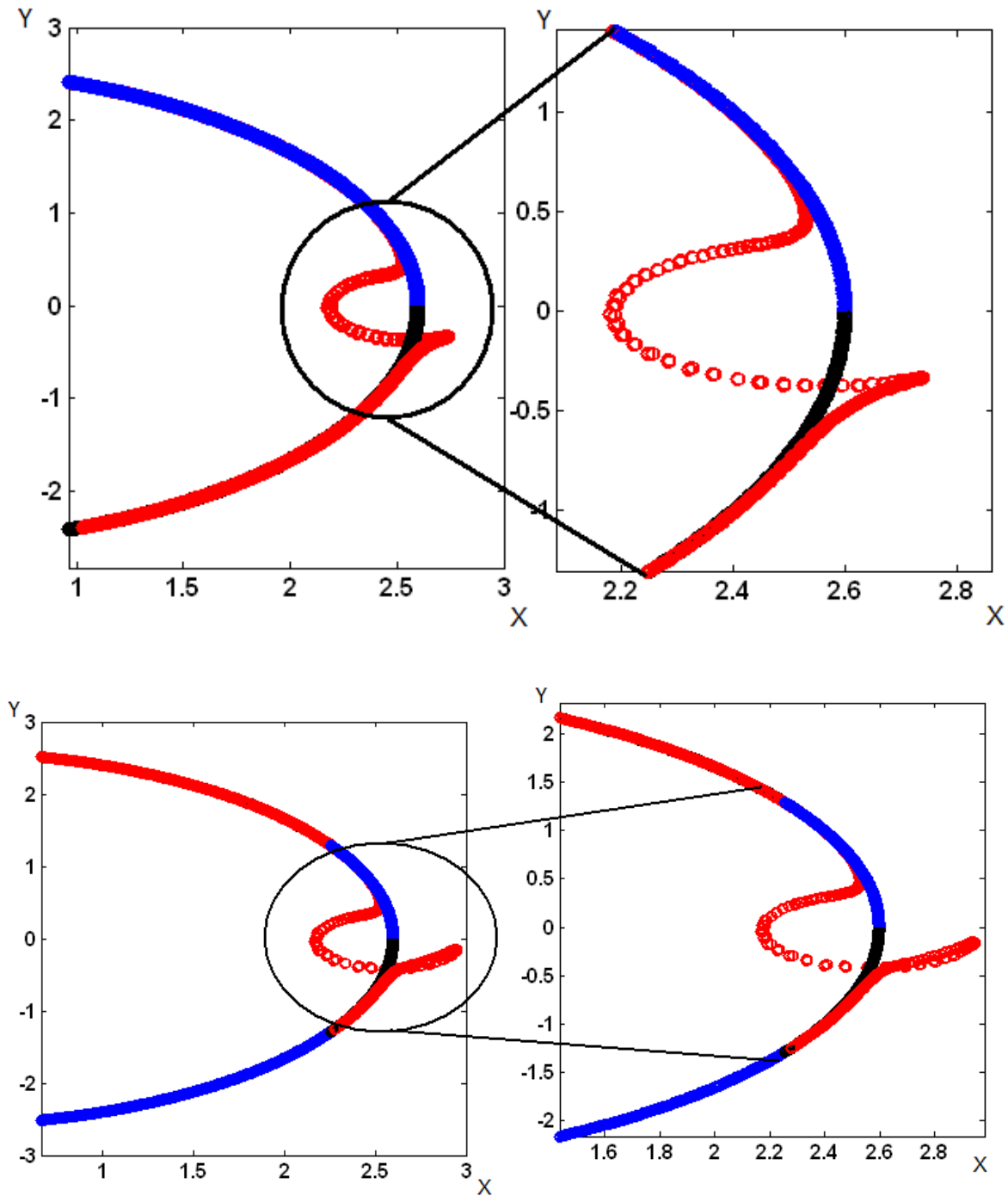


Figura 55. Efectos inerciales del capturador Superior compensación inercial en viscosidad, Inferior inercia del móvil no compensada (patinamiento).

Capitulo 6: Cinemática y dinámica del interceptor (ejemplo práctico)

Hasta el momento las técnicas descritas en capítulos previos son funcionales para partículas (por ejemplo misiles o robots móviles sin remolque) o robots cartesianos (por ejemplo el Novint Falcon que aunque es una plataforma delta, su control es directamente en el espacio Euclideano de dimensión 3) en la Fig. 56 , se muestra un ejemplo de evasión-intercepción extensible a 3D modificando las dimensiones de las matrices y vectores correspondientes, se trata de un objeto persiguiendo a su objetivo y eludiendo múltiples partículas en dirección de movimiento opuesto al atractor, el problema con una partícula es que es un ente de movilidad cartesiana y un robot requiere ecuaciones de transformación a su espacio articular (menor, igual o mayor en grados de libertad), por lo cual antes de ahondar en esta matriz de transformación espacio robot-espacio cartesiano, es necesaria la teoría elemental de robots manipuladores, esto se exhibe a continuación con un caso practico de simulación

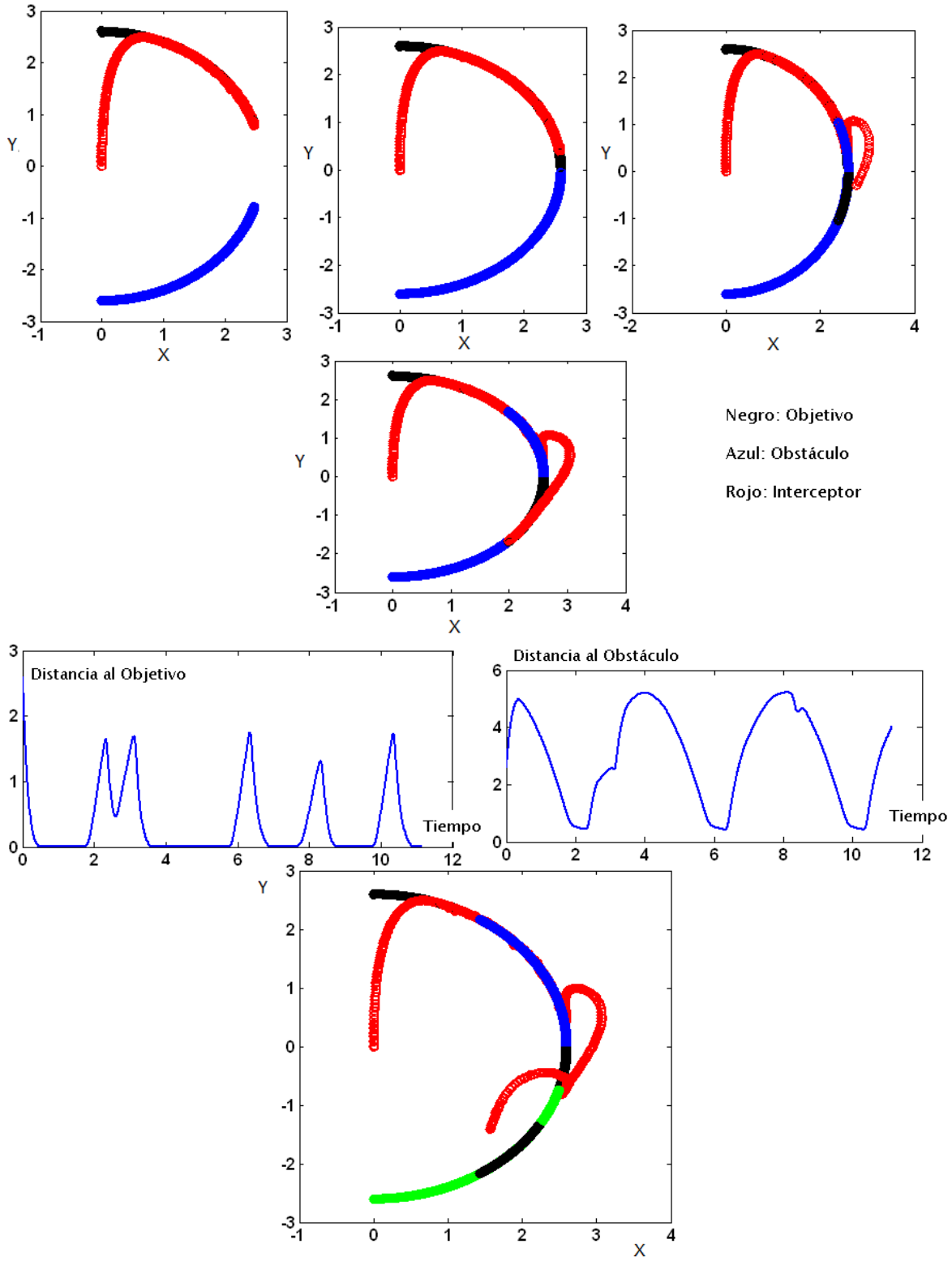


Figura 56. Partícula en tareas de intercepción-evasión de múltiples objetos móviles,, Superior: Evolución de la Evasión-Intercepción; Intermedia: distancia del interceptor al Objetivo y al Obstáculo; Inferior: Un segundo Obstáculo

Cabe aclarar que se muestra paso a paso el estudio y simulación de un robot manipulador con el procedimiento requerido y sin la formalidad que sustenta cada ecuación, esto debido a que no es objetivo de esta tesis ser un texto sobre robótica, sin embargo, la formalidad necesaria para cada procedimiento puede hallarse en [17] y [18], de hecho, la metodología expuesta y las herramientas computacionales utilizadas están basadas en Spong[17] aunque existen alternativas como en [19] y [20].

Se escoge la siguiente configuración de robot por ser de uso común en la industria y sector educativo:

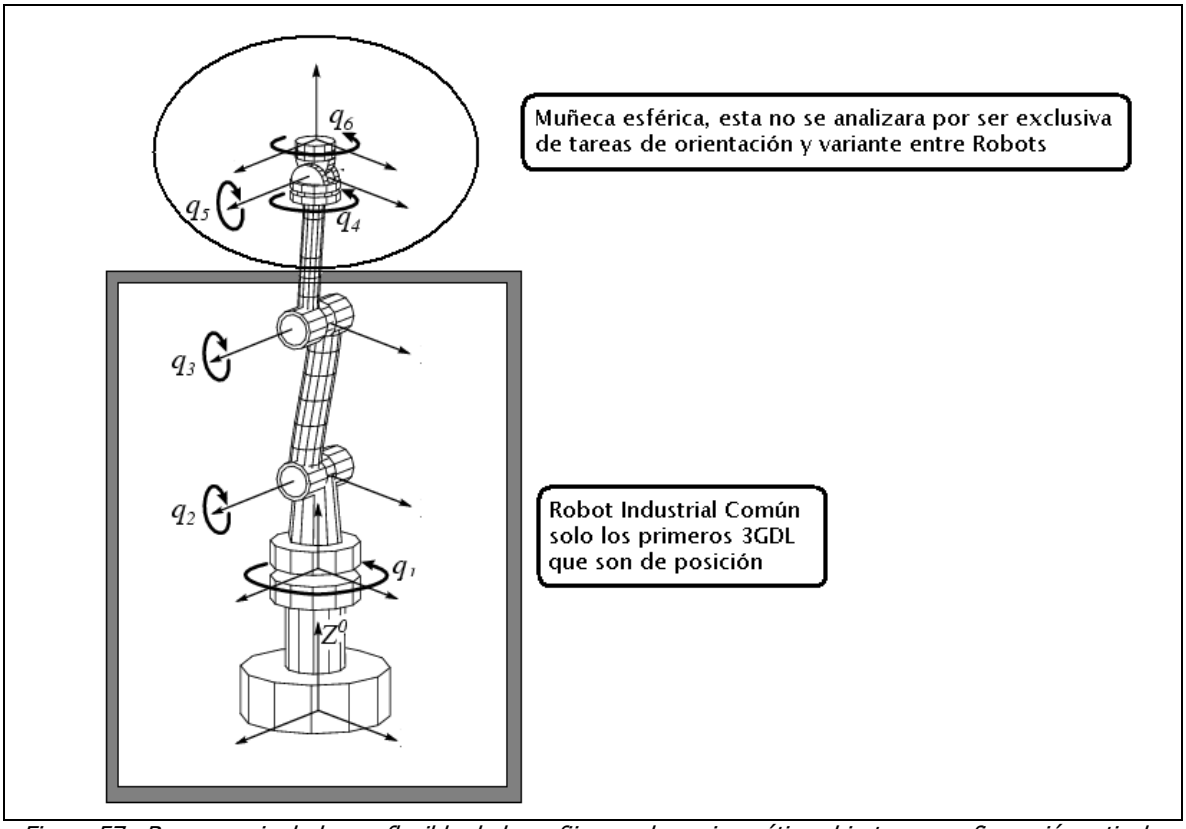


Figura 57. Brazo manipulador no flexible de base fija y cadena cinemática abierta en configuración articular

Tal disposición es llamada articular antropomorfa tiene 3 GDL de rotación y es asociada al movimiento simplificado (cadera-hombro-codo), gran parte de los robots manipuladores la presentan para realizar tareas de posicionamiento 3D (ubicación en un punto del espacio sin importar la orientación).

6.1 Cinemática y Dinámica de Robots Manipuladores

Cinemática

1. Establecer un diagrama de articulaciones y eslabones

Cada articulación representa a un actuador, en este caso 3 motores de efecto rotacional, cada eslabón representa las distancias básicas de los cuerpos que interconectan las articulaciones

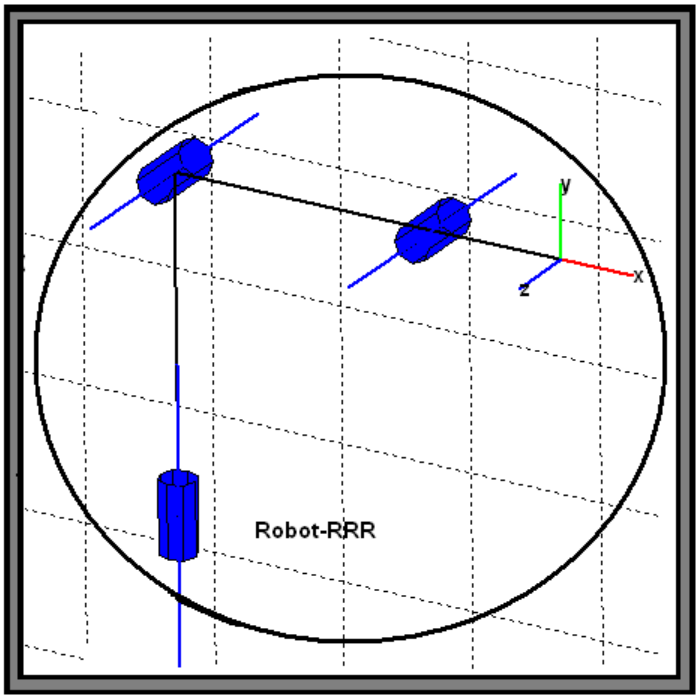


Figura 58. Diagrama de eslabones y articulaciones asociado al robot a simularse

2. Establecer una configuración de ceros (configuración desde la cual las variables articulares son referidas a su valor cero) y sobre esta geometría colocar marcos coordenados conforme a las reglas de [7]

Esto es llamado procedimiento estándar de Denavit-Hartenberg y es útil pues reduce a 4 ecuaciones los 6 posibles movimientos de cada articulación o punto del robot a monitorear/controlar (como pueden ser centros de masa), es conveniente generar cinemáticas de articulación y de centros de masa para el calculo de la dinámica y jacobianos que sean necesarios

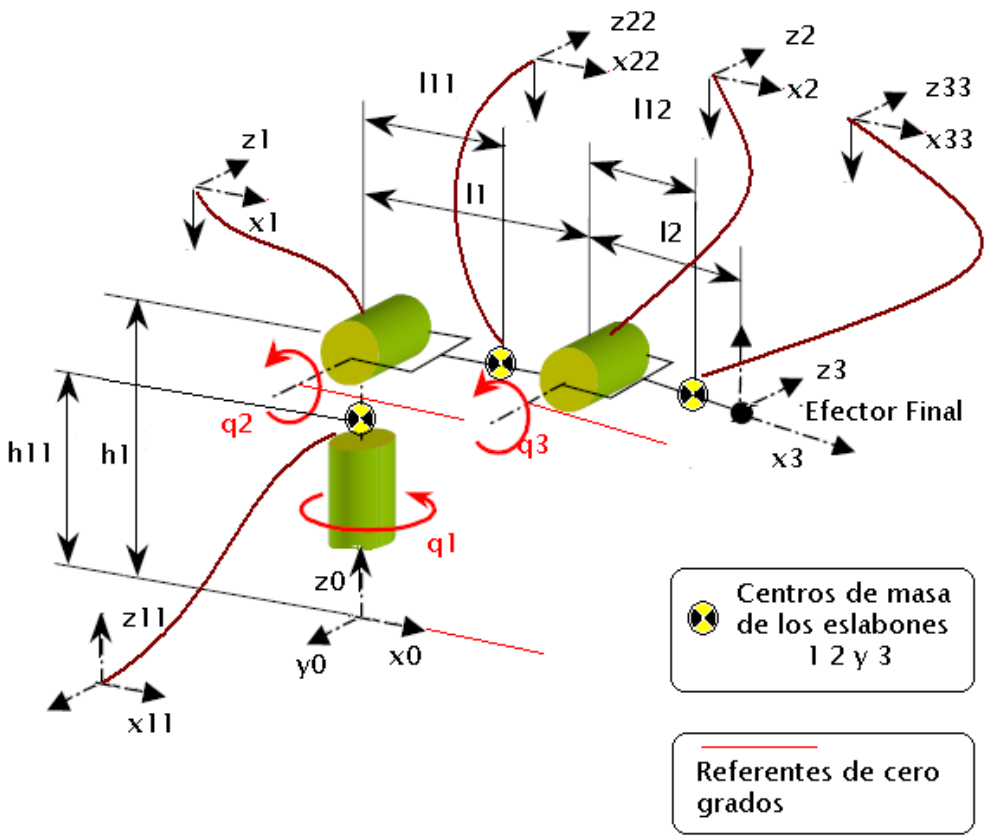


Figura 59. Posición cero robot, marcos coordenados [x,y,z] y distancias de referencia h,l

3. Con base a la geometría de espacios generada en el punto 2 de este proceso y de acuerdo a [7], se crea la matriz de Denavit Hartenberg (parametrizacion de 4 elementos: dos traslaciones y dos rotaciones que actúa sobre ejes definidos perpendiculares)

i	θ_i	d_i	a_i	α_i
1	q_1	h_1	0	$\frac{\pi}{2}$
2	q_2	0	l_1	0
3	q_3	0	l_2	0
Cm11	q_1	h_{11}	0	0
Cm22	q_2	0	l_{11}	0
Cm33	q_3	0	l_{22}	0

Tabla 1: tabla de parámetros DH del robot articular descrito

4. Con base a [7] y en si a la metodología DH estándar, se construyen las matrices de transformación homogénea, estas dan la información entre las transformaciones espaciales sucesivas requeridas para ir de un marco coordenado del robot a otro

$$T_{i-1}^{i} = \begin{bmatrix} \cos\theta_i & -\sin\theta_i\cos\alpha_i & \sin\theta_i\sin\alpha_i & \alpha_i\cos\theta_i \\ \sin\theta_i & \cos\theta_i\cos\alpha_i & -\cos\theta_i\cos\alpha_i & \alpha_i\sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Donde T es la matriz de transformación entre el marco coordenado i e i-1, en el caso del n-esimo eslabón

$$T_0^n = (T_0^1)(T_1^2).....(T_{n-1}^n) \quad (38)$$

En el caso de centros de masa *Cmnn* por ejemplo el del eslabón 2

$$T_0^{Cm22} = (T_0^1)(T_1^{Cm22}) \quad (39)$$

En general

$$T_0^{Cmnn} = (T_0^{n-1})(T_{n-1}^{Cmnn}) \quad (40)$$

Por otra parte, se define un Jacobiano como la matriz de transformación entre espacios vectoriales diferenciales, en el caso de un brazo manipulador estos espacios son el cartesiano y el articular

$$\begin{bmatrix} \frac{dx(q_1,q_2....,q_n)}{dt} \\ \frac{dy(q_1,q_2....,q_n)}{dt} \\ \frac{dz(q_1,q_2....,q_n)}{dt} \\ \frac{dq_x(q_1,q_2....,q_n)}{dt} \\ \frac{dq_y(q_1,q_2....,q_n)}{dt} \\ \frac{dq_z(q_1,q_2....,q_n)}{dt} \end{bmatrix} = \begin{bmatrix} \frac{\partial x(q_1,q_2....,q_n)}{\partial q_1} & \cdot & \cdot & \cdot & \cdot & \frac{\partial x(q_1,q_2....,q_n)}{\partial q_n} \\ \frac{\partial y(q_1,q_2....,q_n)}{\partial q_1} & \cdot & & & & \cdot \\ \frac{\partial z(q_1,q_2....,q_n)}{\partial q_1} & & \cdot & & & \cdot \\ \frac{\partial q_x(q_1,q_2....,q_n)}{\partial q_1} & & & \cdot & & \cdot \\ \frac{\partial q_y(q_1,q_2....,q_n)}{\partial q_1} & & & & \cdot & \cdot \\ \frac{\partial q_z(q_1,q_2....,q_n)}{\partial q_1} & \cdot & \cdot & \cdot & \cdot & \frac{\partial q_z(q_1,q_2....,q_n)}{\partial q_n} \end{bmatrix} \begin{bmatrix} \frac{dq_1}{dt} \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \frac{dq_n}{dt} \end{bmatrix}$$

Al rescribir lo anterior se obtiene la definición matemática del Jacobiano

$$\frac{dX}{dt} = J(q)\frac{dq}{dt} \quad (41)$$

5. Una matriz homogénea contiene información tanto de rotación como de traslación, esto se describe en la siguiente figura y teniendo eso en mente se procede a encontrar el jacobiano del robot de una forma alterna y simple de programar con respecto a la ya mencionada en la (41), la parte referente a velocidades lineales de tal jacobiano es la transformación requerida y mencionada al inicio del capitulo pues su transpuesta relaciona fuerzas cartesianas con torques articulares ver Spong[7]

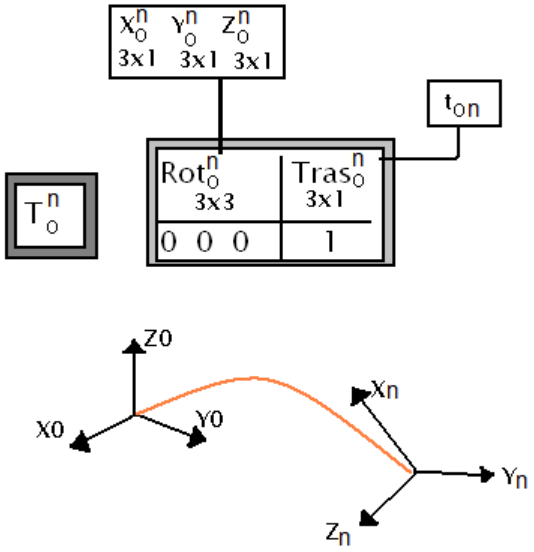


Figura 60. Desglose de una matriz homogénea requerido para esta parte del procedimiento

$$J_{(6xn)} = \begin{bmatrix} J_{v_{(3xn)}} \\ J_{w_{(3xn)}} \end{bmatrix} = \left[J_{0_{(6x1)}}, J_{1_{(6x1)}}, J_{m_{(6x1)}} \ldots\ldots, J_{n-1_{(6x1)}} \right] \quad (42)$$

El jacobiano obtenible solo mediante esta metodología puede subdividirse en 2 bloques: lineal y angular, asimismo en n bloques individuales que representan la acción de cada eslabón a la totalidad del manipulador

Caso Revoluta o Rotacional

$$J_m = \begin{bmatrix} z^m_{0(3x1)} \times (t_{0n} - t_{0m}) \\ z^m_{0(3x1)} \end{bmatrix} \quad (43)$$

Esto se debe a que además de tener un termino exclusivamente rotacional parte inferior del vector, las revolutas generan movimiento lineal obtenible respecto al brazo de palanca parte superior del vector

Caso Articulación Lineal o Prismática

$$J_m = \begin{bmatrix} z^m_{0(3x1)} \\ 0_{(3x1)} \end{bmatrix} \quad (44)$$

Las articulaciones lineales solo generan movimiento lineal por tanto su parte rotacional es 0

Ejemplos para el brazo manipulador empleado se muestran a continuación:

$$J_{Cm22} == \begin{bmatrix} z_0 \times t_{022} & z^1_0 \times (t_{022} - t_{01}) & 0_{3x1} \\ z_0 & z^1_0 & 0_{3x1} \end{bmatrix} \quad (45)$$

$$J_E = J_3 = \begin{bmatrix} z_0 \times t_{03} & z^1_0 \times (t_{03} - t_{01}) & z^2_0 \times (t_{03} - t_{02}) \\ z_0 & z^1_0 & z^2_0 \end{bmatrix} \quad (46)$$

Se indica el Jacobiano del efector final, y su cinemática directa ya que estas se utilizan en la simulación

$$J_E = \begin{bmatrix} -\sin(q_1)(l_2\cos(q_2+q_3)+l_1\cos(q_2)) & -\cos(q_1)(l_2\sin(q_2+q_3)+l_1\sin(q_2)) & -l_2\sin(q_2+q_3)\cos(q_1) \\ \cos(q_1)(l_2\cos(q_2+q_3)+l_1\cos(q_2)) & -\sin(q_1)(l_2\sin(q_2+q_3)+l_1\sin(q_2)) & -l_2\sin(q_2+q_3)\sin(q_1) \\ 0 & l_2\cos(q_2+q_3)+l_1\cos(q_2) & l_2\cos(q_2+q_3) \\ 0 & \sin(q_1) & \sin(q_1) \\ 0 & -\cos(q_1) & -\cos(q_1) \\ 1 & 0 & 0 \end{bmatrix} \quad (47)$$

$$X_R(q) = t_{03} = \begin{bmatrix} \cos(q_1)(l_2\cos(q_2+q_3)+l_1\cos(q_2)) \\ \sin(q_1)(l_2\cos(q_2+q_3)+l_1\cos(q_2)) \\ h_1 + l_2\sin(q_2+q_3)+l_1\sin(q_2) \end{bmatrix} \quad (48)$$

Dada la ecuación (41), el desplazamiento virtual resulta:

$$\delta X = J(q)\delta Q \quad (49)$$

El trabajo virtual asociado es:

$$\delta w = F^T\delta X - \tau^T\delta Q \quad (50)$$

Al reemplazar (49) queda

$$\delta w = F^T J(q)\delta Q - \tau^T\delta Q$$

Cuando el manipulador se encuentra en equilibrio

$$F^T J(q)\delta Q - \tau^T\delta Q = 0 \quad (51)$$

De esta forma el equivalente de (41) en el espacio fuerza es:

$$T_{(nx1)} = J_{v\ (nx3)}^{T} F_{(3x1)} \quad (52)$$

En este momento ya se tiene la matriz de transformación espacio mundo- espacio robot, sin embargo, como el manipulador cuenta con longitudes, momentos inerciales y masas por mencionar algunos parámetros propios del sistema, Tal torque solo tiene un efecto proporcional y es necesario saber su dinámica gravitatoria así como una acción diferencial para implementarlas como parte del controlador y que no afecte las tareas principales abordadas en capítulos previos [13], [21], [22]

Dinámica

Conocida la cinemática del robot, lo siguiente es obtener su dinámica (ya sea para usarse como parte del controlador o en este caso únicamente para simular al sistema)

Aunque existen metodologías Newtonianas y Hamiltonianas, la compatible con las técnicas descritas en el presente texto es la de Euler-Lagrange, pues permite obtener la dinámica de un sistema dadas sus energías cinéticas y potenciales, energías que

están en función de la velocidad y posición del sistema lo cual es justamente lo que se hubo calculado con la cinemática y permite definir los parámetros dinámicos del robot como se explica a continuación:

Se define el Lagrangiano del sistema como:

$$L = K - U \quad (53)$$

En la cual K es la energía cinética total del sistema y U es la potencial,

A partir de esta ecuación se obtiene la expresión de Fuerzas y Pares del sistema de la siguiente forma:

$$\tau = \frac{d}{dt}\left(\frac{\partial L}{\partial(\frac{dq}{dt})}\right) - \frac{\partial L}{\partial q} \quad (54)$$

Remplazando L

$$\tau = \frac{d}{dt}\left(\frac{\partial K}{\partial(\frac{dq}{dt})}\right) - \frac{\partial K}{\partial q} + \frac{\partial U}{\partial q}$$

Redefiniendo se obtiene la ecuación de brazos manipuladores rígidos

$$\frac{\partial U}{\partial q} = G$$

$$\frac{d}{dt}\left(\frac{\partial K}{\partial(\frac{dq}{dt})}\right) - \frac{\partial K}{\partial q} = M(q)\frac{d^2q}{dt^2} + C(\frac{dq}{dt}, q)\frac{dq}{dt}$$

$$M(q)\frac{d^2q}{dt^2} + C(\frac{dq}{dt}, q)\frac{dq}{dt} + G(q) = \tau \quad (55)$$

Una forma más simple computacionalmente hablando de obtener mencionado modelo es como sigue:

6. Obtener M, C y G que modelan la ecuación general de movimiento del robot

Dos razones motivan a este punto, la primera como se dijo es simular el comportamiento de un robot, la segunda es que al ser los potenciales artificiales un controlador PD, este para operar en 3D en forma adecuada necesita un compensador gravitatorio [17], [18], [19], [23] el cual esta determinado por la adición del vector de pares gravitacionales G :

$$M(q) = \sum_{i=1}^{n} (m_i J_{vi}^T J_{vi} + J_{wi}^T R_i I_i R_i^T J_{wi}) \quad (56)$$

En este caso $n = 3$

Donde *m* es la masa individual de cada eslabón e *I* es su correspondiente tensor de Inercia (3x3), R es la matriz de rotación de la correspondiente matriz homogénea.

La matriz de Coriolis por otra parte, se obtiene elemento a elemento, con la metodología de Christoffel y a partir de M, C también es de dimensión nxn

$$C_{(k,j)} = \sum_{i=1}^{n} \frac{1}{2} \left\{ \frac{\partial M_{(k,j)}}{\partial q_i} + \frac{\partial M_{(k,i)}}{\partial q_j} - \frac{\partial M_{(i,j)}}{\partial q_k} \right\} \frac{dq_i}{dt} \quad (57)$$

Existen otras formas de obtener tal matriz, pero esta es la que permite demostrar la estabilidad de los controladores ya que cumple con la llamada propiedad de antisimetria

$$N = \frac{dM}{dt} - 2C$$

$$N_{(j,k)} = -N_{(k,j)} \quad (58)$$

Donde N es una matriz antisimetrica

Por ultimo, el vector de pares gravitacionales:

A partir del vector de gravedad:

$$V_g = \begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix} \quad (59)$$

se obtiene la energía potencial del sistema .

$$P = V_g^T \sum_{i=1}^{n} t_{0Cmii} m_i \quad (60)$$

Derivando, se encuentra G el cual es el vector de fuerzas y pares gravitacionales.

$$G(q_1, q_2.., q_n) = \nabla P_q \qquad (61)$$

Control

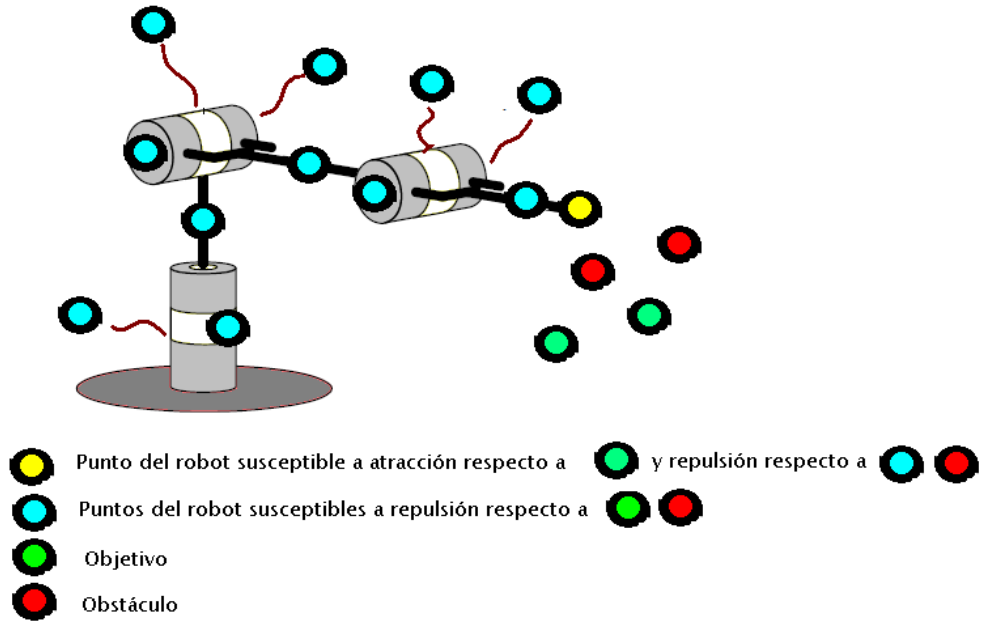
Se define la ecuación del manipulador asociada al sistema de la siguiente forma

$$M(q)\frac{d^2q}{dt^2} + C(\frac{dq}{dt}, q)\frac{dq}{dt} + G(q) = T - B\frac{dq}{dt} \quad (62)$$

7. Se establece el controlador para intercepción y evasión cuyo diseño se ilustra en Fig. 61.

$$\tau = J_{vE}^T F_{at} + J_{vE}^T \sum_{o=1}^{s} Fo_{rep} + \sum_{l=1}^{r} J_{vl}^T \sum_{o=1}^{s} Fo_{rep} - K_d \frac{de}{dt} + G \qquad (63)$$

F_{at} Es la fuerza de atracción a la partícula a interceptar en turno (una a una) Fo_{rep} Es la Fuerza de repulsión de cada obstáculo, se utilizan tantos Jacobianos como puntos de control en el robot sean requeridos, el del efector final siempre es requerido, G es el vector de pares gravitacionales Kd es la ganancia diferencial actuando como amortiguador sobre el error de velocidad articular



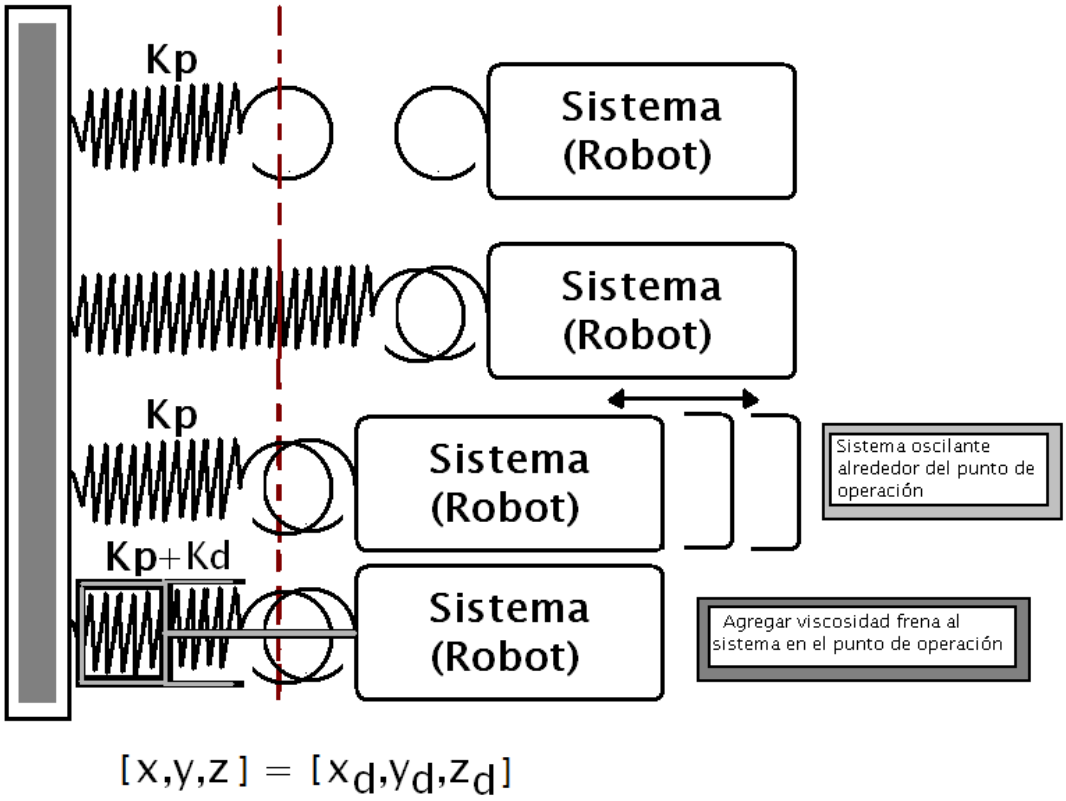


Figura 61. Zonas de interacción del robot manipulador con los objetivos y obstáculos, analogía del controlador con un resorte y un amortiguador, al inicio sin conexión, después el resorte llevando al sistema a la condición deseada, finalmente y debido al comportamiento puramente oscilante del resorte se añade un amortiguador

Se recomienda que para un robot se utilicen como puntos de control al menos uno por centro de masas, uno por articulación y el del efector final.

Ahora bien esto no evita colisiones, pues para evitar un choque con una articulación hacen falta mas puntos monitores de su geometría, de hecho una tesis que pudiese surgir de esta afirmación implicaría hallar la distribución de masa o carga de cualquier punto de un volumen dado y sujeto a rotaciones y traslaciones, para conocer en forma veraz su área de influencia respecto a una partícula ver Fig. 62, la ecuación (63) se transformaría en una sumatoria de integrales volumétricas y sus parámetros de integración quedarían definidos en términos de la pose propia de cada eslabón (orientación y posición)

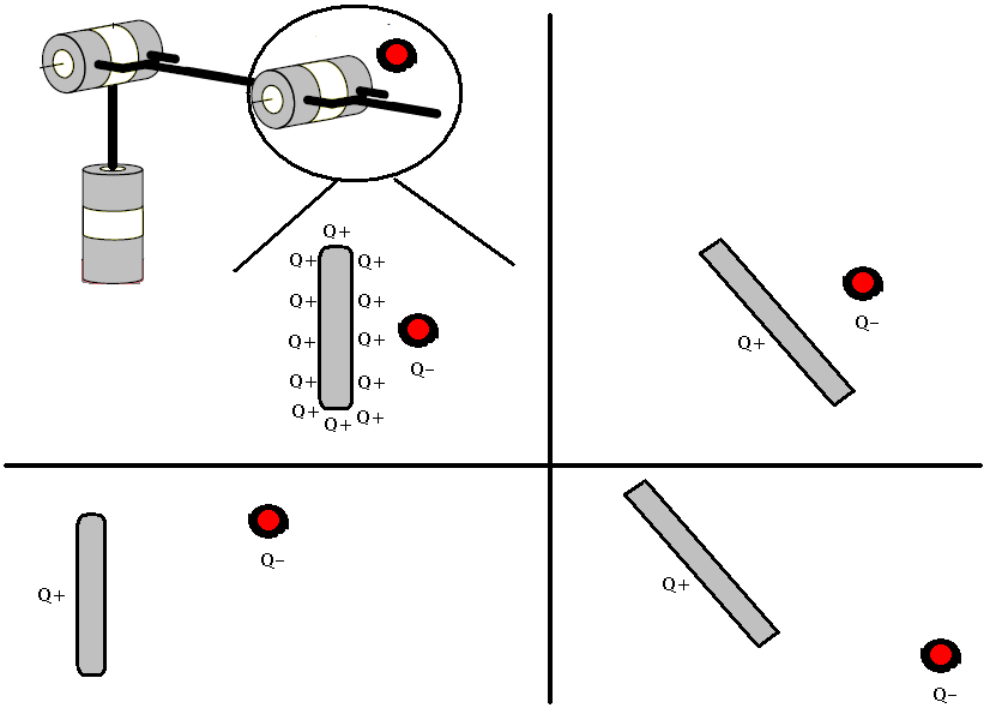


Figura 62. Distribución volumétrica de cargas o masas para generar fuerzas de atracción-repulsión en una partícula; superior izquierda: caso estándar (simple o base); superior derecha: caso volumen rotado; inferior izquierda: caso volumen trasladado; inferior derecha: caso volumen trasladado y rotado, el volumen en su modo primigenio es un prisma que describe longitud, anchura y espesor máximos por cada eslabón del robot

6.2 Demostración por Lyapunov de la estabilidad del controlador

Para la presente demostración por Lyapunov, se considera que:

$$J_{vE}^{T} F_{at} \gg J_{vE}^{T} \sum_{o=1}^{s} Fo_{rep} + \sum_{l=1}^{r} J_{vl}^{T} \sum_{o=1}^{s} Fo_{rep}$$

Es decir que la fuerza de atracción es predominante sobre todas las de repulsión o equivalentemente dicho que los obstáculos son perturbaciones diseñables del sistema, una forma de lograr esto es estableciendo potenciales artificiales sin mínimos locales, la existencia de mínimos locales se debe a que las cimas procedentes de los obstáculos son irrotacionales, la salida de estas funciones se logra cuando las funciones de repulsión (es decir los valles locales) giran Fig. 63, por otra parte, se dice que los potenciales repulsores son perturbaciones porque introducen energía al sistema en lugar de disiparla (en su diseño, por si mismos no cumplen con los métodos de Lyapunov), existen funciones de energía potencial artificial como la mostrada por Ren y McIsaac[48], donde se puede observar en los diagramas de energía que ocurre precisamente una introducción de energía al sistema debida al obstáculo, sin embargo el atractor esta diseñado para que el sistema supere literalmente este 'bache' Fig. 64 y nuevamente se estabilice, ese comportamiento es precisamente el que permiten los potenciales rotonormales propuestos en esta tesis

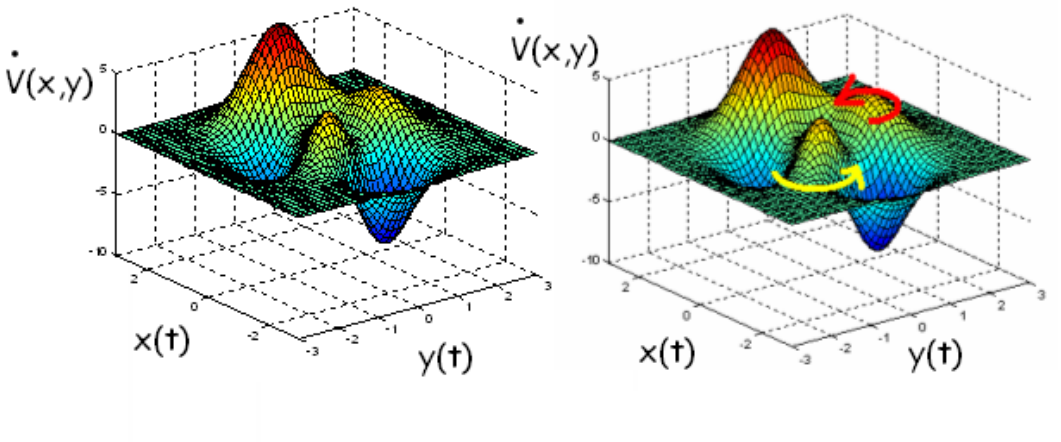


Figura 63. Analogía geográfica de los campos potenciales artificiales con y sin mínimos locales vistos desde el punto de vista del segundo método de Lyapunov

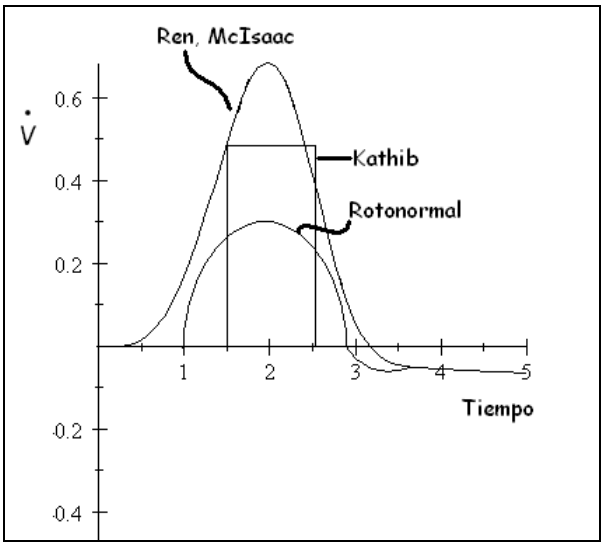


Figura 64. Analogía de los repulsores con perturbaciones (baches diseñables)

Tomando en cuenta estas consideraciones, la ecuación (63) se simplifica a:

$$\tau = J_v^T F_{at} - K_d \frac{de}{dt} + G(q) \quad (64)$$

La demostración que prosigue, presupone que los campos potenciales artificiales satisfacen las siguientes propiedades, ello se profundiza e ilustra en el desarrollo

Para ello se emplea la siguiente notación $Y_R = X_R - X_d$ donde X_R es la posición del efector final definida por su cinemática directa, en este caso es (48) y X_d es la posición cartesiana deseada

1 El punto mínimo de la función potencial artificial ocurre en $Y = 0$

2 $\nabla_q f(0) = 0$

3 $(\frac{dq}{dt})^T \nabla_q f(\frac{dq}{dt}) > 0 \quad \forall \frac{dq}{dt} \neq 0$ (65)

A primer vista la ecuación (64) resulta semejante al modelo propuesto por Arimoto

[43] y Kelly [36]

$$\tau = -\nabla_q U_a(Y_R(q)) - \nabla_q f(\frac{dq}{dt}) + G(q) \qquad (66)$$

Sin embargo es necesario establecer una igualdad entre las ecuaciones referidas, para tal propósito sea:

$$f(\frac{dq}{dt}) = \frac{k_1}{2}\left(\frac{dq_1}{dt}\right)^2 + \frac{k_2}{2}\left(\frac{dq_2}{dt}\right)^2 + \ldots \frac{k_n}{2}\left(\frac{dq_n}{dt}\right)^2 \qquad (67)$$

Una función generadora de un campo potencial normal respecto al vector $\frac{dq}{dt}$, encontrando su gradiente:

$$\nabla_{\frac{dq}{dt}} f(\frac{dq}{dt}) = \left(k_1 \frac{dq_1}{dt}, k_2 \frac{dq_2}{dt} \ldots, k_n \frac{dq_n}{dt}\right) \qquad (68)$$

Y estableciendo un comportamiento atractor tal que el campo resulte convergente a su valor global mínimo

$$-\nabla_{\frac{dq}{dt}} f(\frac{dq}{dt}) = -\left(k_1 \frac{dq_1}{dt}, k_2 \frac{dq_2}{dt} \ldots, k_n \frac{dq_n}{dt}\right)$$

$$-\nabla_{\frac{dq}{dt}} f(\frac{dq}{dt}) = -\begin{bmatrix} k_1 & 0 & 0 \\ 0 & k_2 & 0 \\ 0 & 0 & k_n \end{bmatrix}_{nxn} \begin{bmatrix} \frac{dq_1}{dt} \\ \frac{dq_2}{dt} \\ \frac{dq_n}{dt} \end{bmatrix}_{nx1} = -K_d \frac{dq}{dt} \qquad (69)$$

Solo resta encontrar la igualdad en el termino referente al jacobiano transpuesto, para ello sea

$$Y_R = \begin{bmatrix} x(q) - x_d \\ y(q) - y_d \\ z(q) - z_d \end{bmatrix} \qquad (70)$$

Sea U_a una función generadora de campos potenciales atractivos respecto a Y_R

$$U_a(Y_R) = U_a(Y_R(q)) \qquad (71)$$

Se encuentran las fuerzas asociadas $\nabla_q U_a(Y_R(q))$ lo que por regla de la cadena

equivale a

$$\nabla_q U_a(Y_R(q)) \ = [\nabla_q Y_R(q)]^T \nabla_{Y_R} U_a(Y_R) \quad (72)$$

De nueva cuenta se impone un comportamiento atractor

$$-\nabla_q U_a(Y_R(q)) \ = -[\nabla_q Y_R(q)]^T \nabla_{Y_R} U_a(Y_R)$$

El factor $-\nabla_{Y_R} U_a(Y_R)$ es el campo de fuerzas predominantemente atractivas F descrito en capítulos anteriores, ello supone que $\nabla_q Y_R(q) = J_V$

Esto se demuestra a continuación:

El jacobiano de velocidad lineal o analítico se define como

$$\frac{dY}{dt} = J_v(q)\frac{dq}{dt} \quad (73)$$

$$J_v(q) = \begin{bmatrix} \frac{\partial x}{\partial q_1} & \frac{\partial x}{\partial q_2} & \frac{\partial x}{\partial q_n} \\ \frac{\partial y}{\partial q_1} & \frac{\partial y}{\partial q_2} & \frac{\partial y}{\partial q_n} \\ \frac{\partial z}{\partial q_1} & \frac{\partial z}{\partial q_2} & \frac{\partial z}{\partial q_n} \end{bmatrix} = \begin{bmatrix} \nabla_q x(q) \\ \nabla_q y(q) \\ \nabla_q z(q) \end{bmatrix} = \nabla_q Y_R(q) \quad (74)$$

De esta forma queda demostrado $J_v^T F = -\nabla_q U_a(Y_R(q))$ y en consecuencia

$$\tau = J_v^T F - K_d \frac{de}{dt} + G(q) = -\nabla_q U_a(Y_R(q)) \ - \nabla_q f(\frac{dq}{dt}) + G(q) \quad (75)$$

Una vez establecida la igualdad se desarrolla el calculo de estabilidad de Lyapunov propuesto por Kelly, para ello se usa la ecuación del brazo manipulador (62)

$$M(q)\frac{d^2q}{dt^2} + C(\frac{dq}{dt}, q)\frac{dq}{dt} + G(q) = \tau - B\frac{dq}{dt}$$

Expresada en la siguiente forma

$$M\frac{d^2q}{dt^2} = \tau - B\frac{dq}{dt} - C\frac{dq}{dt} - G \quad (76)$$

Reemplazando τ de *la ecuación (66)*

$$M\frac{d^2q}{dt^2} = -\nabla_q U_a(Y_R(q)) - \nabla_q f(\frac{dq}{dt}) + G - B\frac{dq}{dt} - C\frac{dq}{dt} - G$$

$$M\frac{d^2q}{dt^2} = -[\nabla_q Y_R(q)]^T \nabla_{Y_R} U_a(Y_R) - \nabla_q f(\frac{dq}{dt}) - B\frac{dq}{dt} - C\frac{dq}{dt} \qquad (77)$$

Usando la función candidata de Lyapunov

$$V(q_d - q, \frac{dq}{dt}) = \frac{1}{2}(\frac{dq}{dt})^T M(q)\frac{dq}{dt} + U_a(Y_R(q)) - U_a(Y(q_d)) \qquad (78)$$

La cual cumple con el primer criterio de Lyapunov dado que el termino $\frac{1}{2}(\frac{dq}{dt})^T M(q)\frac{dq}{dt}$ implica un factor cuadrático respecto a $\frac{dq}{dt}$ y la matriz M es simétrica y definida positiva.

El termino $U_a(y(q)) - U_a(y(q_d))$ implica ser definido positivo debido a que es una diferencia de potencial referida al mínimo valor de $U_a(Y(q_d))$ es decir cero, donde $Y(q_d)$ es el valor deseado donde deberá llegar el robot, esto restringe el uso a potenciales artificiales con un mínimo global como se muestra en la Fig. 65.

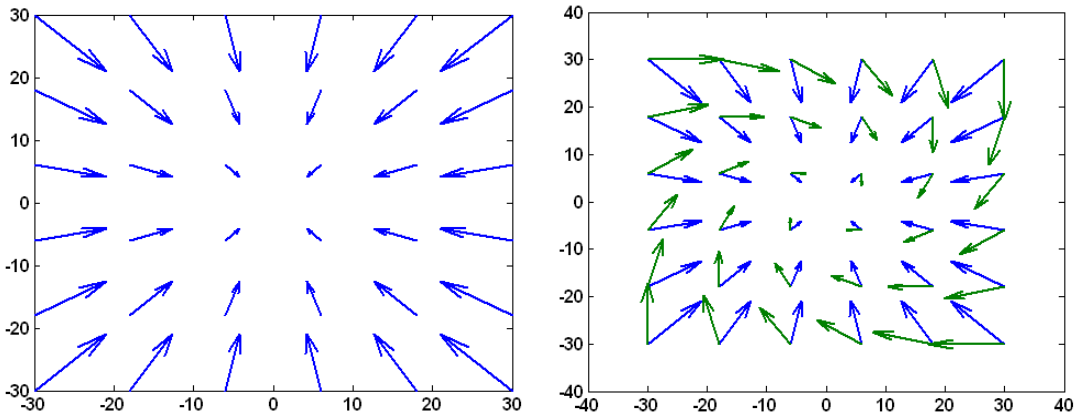


Figura 65. Ejemplos de campos potenciales artificiales que cumplen con los criterios de Estabilidad de Lyapunov

Tambien puede demostrarse por sustitución que $V(0,0) = 0$ y $V(\infty,\infty) = \infty$ y esto implica el uso de potenciales artificiales cuyos campos decrezcan hacia el objetivo, Fig. 65.

Luego de esto se deriva la función de Lyapunov propuesta

$$\dot{V}(q_d - q, \frac{dq}{dt}) = (\frac{dq}{dt})^T M(q) \frac{d^2q}{dt^2} + \frac{1}{2}(\frac{dq}{dt})^T \frac{d(M(q))}{dt} \frac{dq}{dt} + (\frac{dq}{dt})^T \nabla_q U_a(Y_R(q)) \quad (79)$$

Sustituyendo $M(q)\frac{d^2q}{dt^2}$

$$\dot{V}(q_d - q, \frac{dq}{dt}) = -(\frac{dq}{dt})^T \nabla_q U_a(Y_R(q)) - (\frac{dq}{dt})^T \nabla_q f(\frac{dq}{dt}) - (\frac{dq}{dt})^T B \frac{dq}{dt} - (\frac{dq}{dt})^T C \frac{dq}{dt}$$

$$+ \frac{1}{2}(\frac{dq}{dt})^T \frac{d(M(q))}{dt} \frac{dq}{dt} + (\frac{dq}{dt})^T \nabla_q U_a(Y_R(q)) \quad (80)$$

Simplificando

$$\dot{V}(q_d - q, \frac{dq}{dt}) = -(\frac{dq}{dt})^T \nabla_q f(\frac{dq}{dt}) - (\frac{dq}{dt})^T B \frac{dq}{dt} - (\frac{dq}{dt})^T C \frac{dq}{dt} + \frac{1}{2}(\frac{dq}{dt})^T \dot{M} \frac{dq}{dt} \quad (81)$$

Los términos $\frac{1}{2}(\frac{dq}{dt})^T \dot{M} \frac{dq}{dt} - (\frac{dq}{dt})^T C \frac{dq}{dt}$ por la forma en que se definió el calculo de C permiten la llamada propiedad de antisimetria, es decir:

$$\frac{1}{2}(\frac{dq}{dt})^T \dot{M} \frac{dq}{dt} - (\frac{dq}{dt})^T C \frac{dq}{dt} = 0 \quad (82)$$

Sustituyendo este resultado y $-(\frac{dq}{dt})^T \nabla_q f(\frac{dq}{dt})$

$$\dot{V}(q_d - q, \frac{dq}{dt}) = -(\frac{dq}{dt})^T K_d \frac{dq}{dt} - (\frac{dq}{dt})^T B \frac{dq}{dt} \quad (83)$$

Lo cual es definido negativo mientras K_d sea positiva y simetrica, la consecuencia de este resultado por el llamado teorema de Lasalle es que $q \to q_d$, $\frac{dq}{dt} \to 0$ y por dependencia $Y_R \to 0$ y $X_R = X_d$

6.3 Simulador

1 El bloque de Peter Corke es un visualizador 3D animado del robot y sus movimientos, sus entradas son la tabla DH y actualiza el display acorde a los parámetros variantes

2 Se colocan las cámaras virtuales en configuración estereo, usando el toolbox de Mariottini, este permite incluir parámetros extrínsecos e intrínsecos, así como variantes en la forma de la lente

3 Se establecen las marcas visuales usando el mismo toolbox, estas se colocan en los objetivos, los obstáculos y en sitios estratégicos del manipulador como centros de masa y articulaciones usando para ello los vectores de traslación correspondientes: t_{0n}

4 Para que la salida sea la aceleración de q, esta debe obtenerse a partir del modelo asociado al robot correspondiente, esto se ilustra en la Fig. 68.

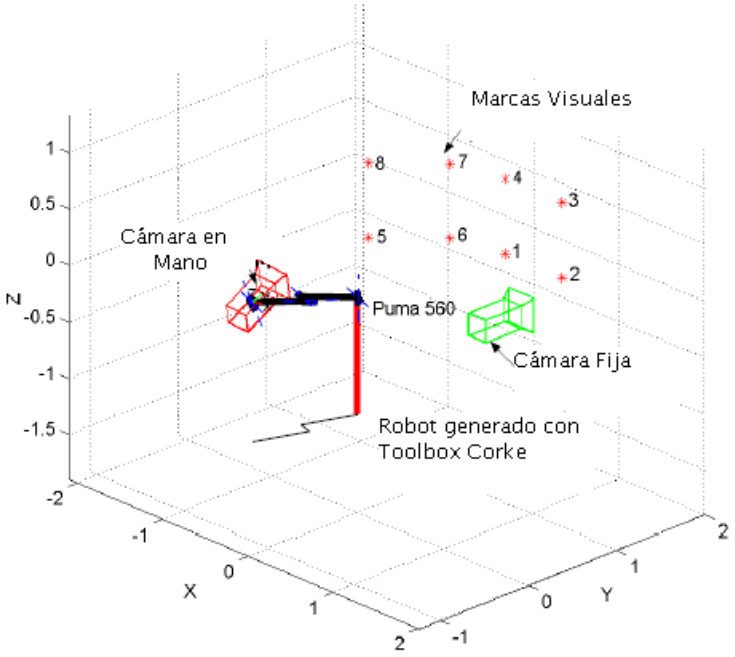


Figura 66. Toolbox de Corke Interactuando con el de Mariottini, tomado del manual del Epipolar Geometry Toolbox

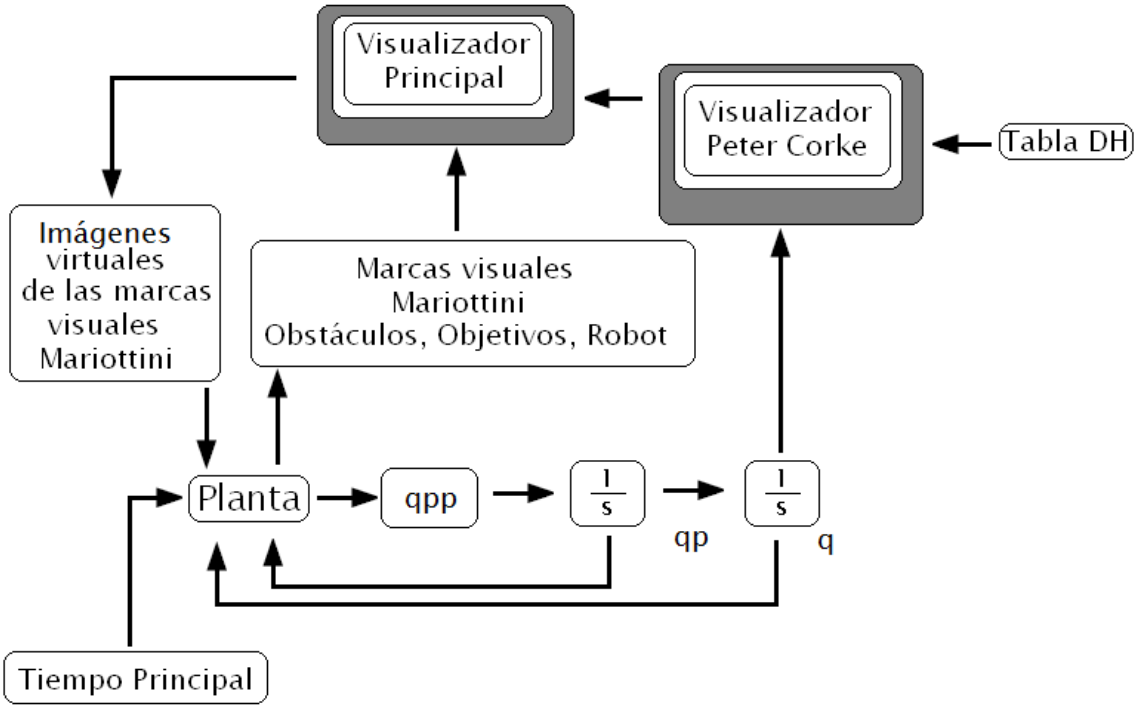


Figura 67. Descripción principal del simulador

La Fig. 67 se describe como sigue:

La planta es el referente genérico de un brazo manipulador rígido de cadena cinemática abierta, la información de salida de la planta es la aceleración articular que se obtiene al despejar la ecuación (76), esta salida se integra en dos ocasiones definiendo el usuario las condiciones iniciales de integración y obteniéndose con ello velocidad y posición articulares, el visualizador de Corke se actualiza con las variaciones de las posiciones articulares. Tanto posiciones como velocidades articulares regresan a la planta para retroalimentar el controlador basado en cinemática directa.

Por otra parte, el controlservovisual estéreo se realiza con las cámaras de Mariottini, adicional a que mencionada herramienta cuenta con sus propias marcas visuales colocadas en los obstáculos y objetivos, esto también se retroalimenta a la planta pues representa la posición de atractores y repulsores.

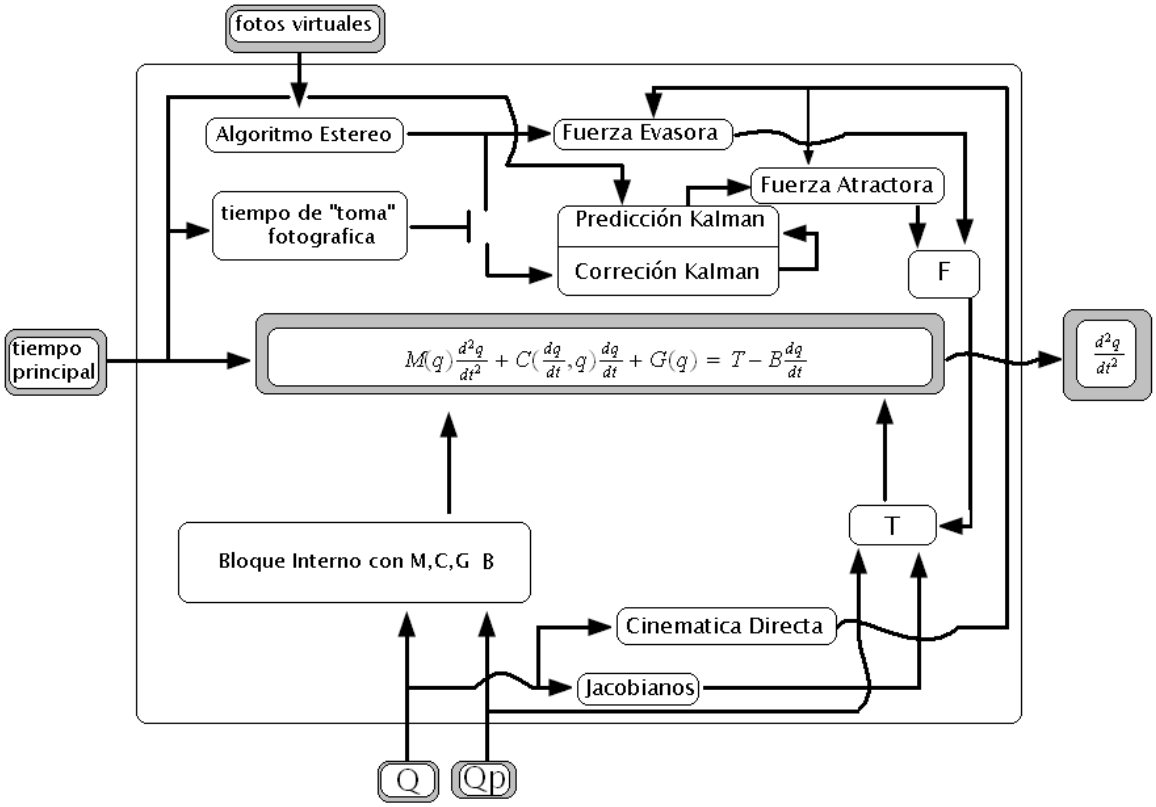


Figura 68. Descripción de la planta

La planta de la Fig. 68 se describe a continuación:

Para obtenerse la salida, es decir la aceleración articular del brazo manipulador, se necesita la ecuación (76) y ello implica definir M, G, C y B en línea pues son dependientes de la retroalimentación de posiciones y velocidades articulares.

La parte relativa al control de la ecuación (76) la define T y este al ser de base jacobiano transpuesto requisita la cinemática directa del sistema para generar mencionada matriz de transformación y multiplicarlo a las fuerzas de atracción y repulsión dependientes de las posiciones de las marcas visuales obtenidas mediante la herramienta de Mariottini en configuración estéreo (con un pos procesamiento mediante el filtro de Kalman en el caso de los atractores).

Como se explico en secciones previas, el controlador requisita una etapa de amortiguamiento (diferencial) y por tanto el controlador también depende de las velocidades articulares retroalimentadas.

Capitulo 7: Consideraciones de implementación

7.1 Parte Visual

1) Hardware

La elección del Hardware de trabajo depende de la siguiente formula:

$$(EV)(EH)(Ca)(bpp)(fps)(cams)=Mbps \quad (84)$$

Donde EV, EH son la resolucion vertical y horizontal respectivamente, Ca es el numero de canales (RGB por ejemplo maneja 3), bpp son los bits que tiene cada pixel, fps son la cantidad de imágenes tomadas en cada segundo, cams es el numero de cámaras simultaneas y Mbps son la cantidad de megabits por segundo .

Por ejemplo, considerese la siguiente transmisión de datos (resolución minima ante pixeleado) en visión estéreo (2 camaras):

$$640(480)(3)(8)(30)(2)=442.368Mbps$$

Lo cual excede al USB 1 y al USB 2 pues aunque el bus USB 2 indica tener 480Mbps, en general las tarjetas madre lo subdividen en 4 conectores con 120Mbps para cada uno, no se puede modificar el tiempo de fps, pues la velocidad máxima del brazo manipulador Mitsubishi usado en esta tesis es de 1m/s, lo cual guarda proporción con el brazo humano y por tanto se requisita de una velocidad de monitoreo visual semejante a los 24 fps del ojo, la única forma de reducir la cantidad de Mbps radica en la resolución, disminuyendo esta a la mitad se tiene:

$$320(240)(3)(8)(30)(2)=110.592Mbps$$

Lo cual seria pareciera ser suficiente de no estar conectados dispositivos periféricos como raton y mouse (necesarios evidentemente para inicializar, pausar o parar las tareas a programar)

Por ultimo la 4ta parte de la resolución:

$$160(120)(3)(8)(30)(2) = 27.648Mbps$$

La cual como puede verse en la Fig. 69 es permitida en la computadora de prueba, debe mencionarse que ambas cámaras deben tener la misma resolución de captura, pues la conversión de espacios requiere una estandarización en pixeles

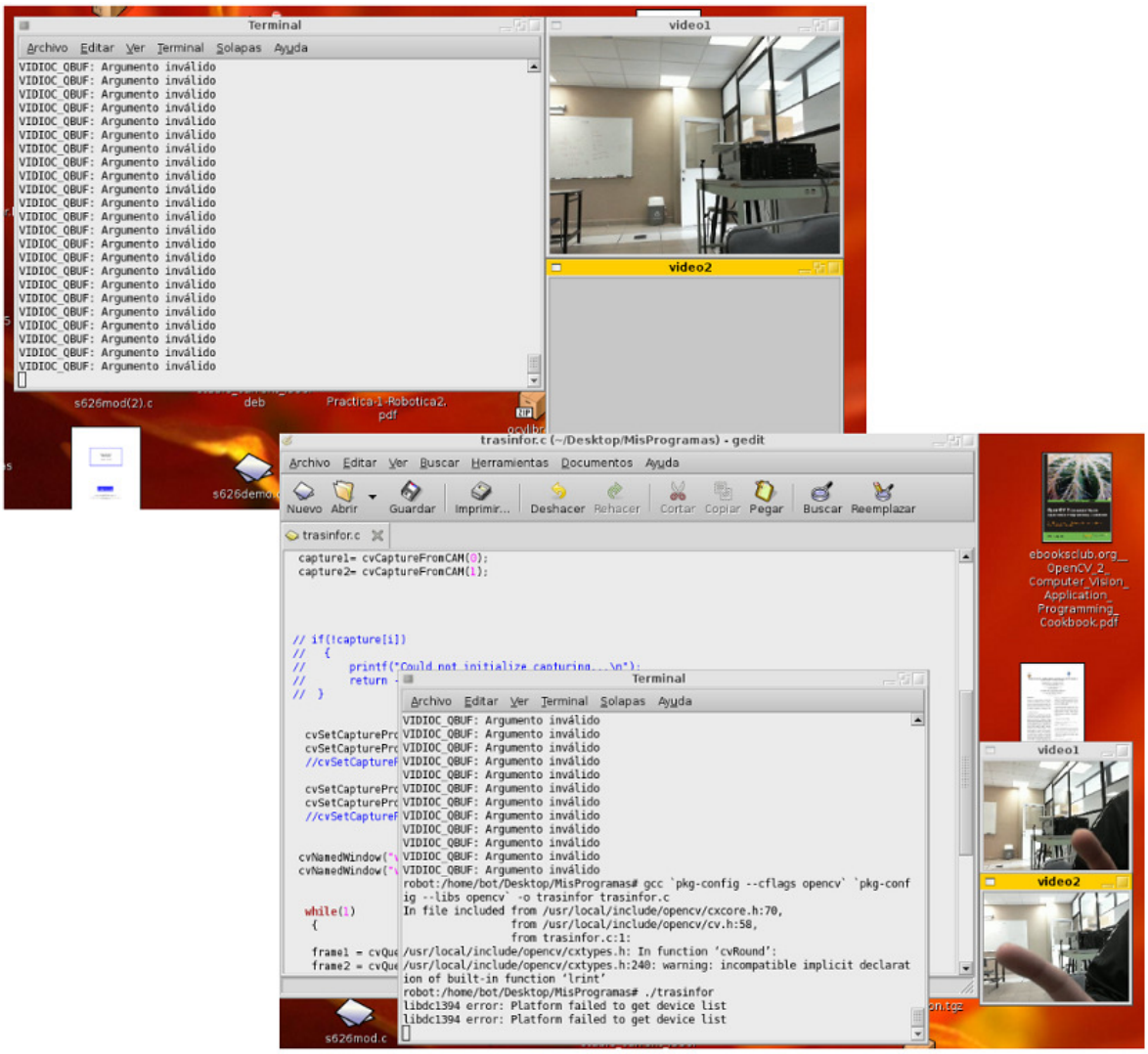


Figura 69. Resolucion permitida en USB 2.0

Existen 2 alternativas para solventar el problema del bus de datos:

1. Tener un CPU cuya tarjeta madre cuente con 2 buses USB independientes

2. Contar con cámaras Firewire (al menos del tipo Firewire 800) cuyo ancho de banda es de 786.5 Mbps

Aunado a lo descrito, el software a utilizarse requisita al menos un procesador de 2Ghz y 4Gigabytes de Memoria Ram

2) Software

Por otra parte, se necesita realizar la tarea en tiempos predefinidos y sin exceder de ellos, es decir cada señal de control (calculado en tesis anterioes con la constante de tiempo del manipulador que se vera mas adelante) implica enviar un pulso de potencia a los motores del robot cada milisegundo sin excederse de este tiempo o adelantarse, por tal razón se requiere flexibilidad de modificación al tiempo real del sistema (tiempo real es la ejecución de una tarea en un tiempo preacotado como lo demanda esta tesis y no necesariamente involucra “rapidez”)

El sistema operativo mas flexible para este tipo de tareas es Linux, que además de ser software libre en la mayoría de sus distribuciones, puede obtenerse como es el caso de Debian parches de asignación de prioridades en tiempo real (RTAI –Real Time Application Interface-) [50]

El lenguaje de programación utilizado por su compatibilidad con los procesos RTAI es ANSI C, una librería de algoritmos de visión artificial compatible con mencionado sistema operativo y también mencionado lenguaje de programación es OPENCV.

A continuación se muestran los resultados obtenidos en OPENCV del procedimiento descrito en el capitulo 3 de esta tesis, los códigos se incluyen en el correspondiente apéndice y están basados en [4], [51], [52] y [53].

Deteccion de móviles por colores:

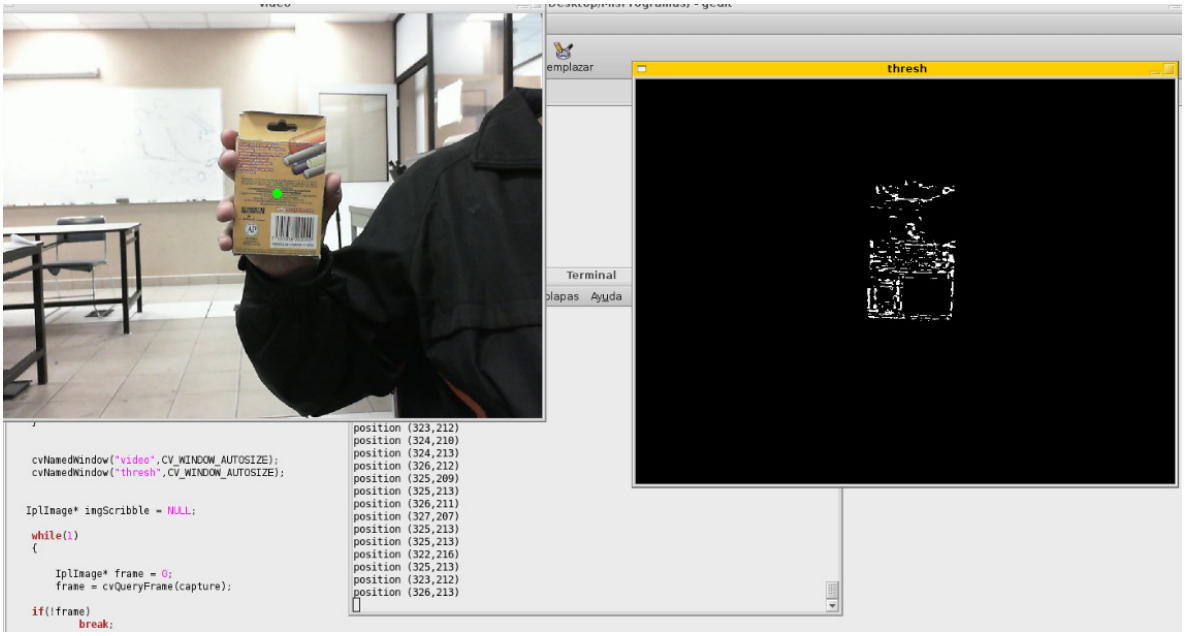


Figura 70. Detector de moviles amarillos

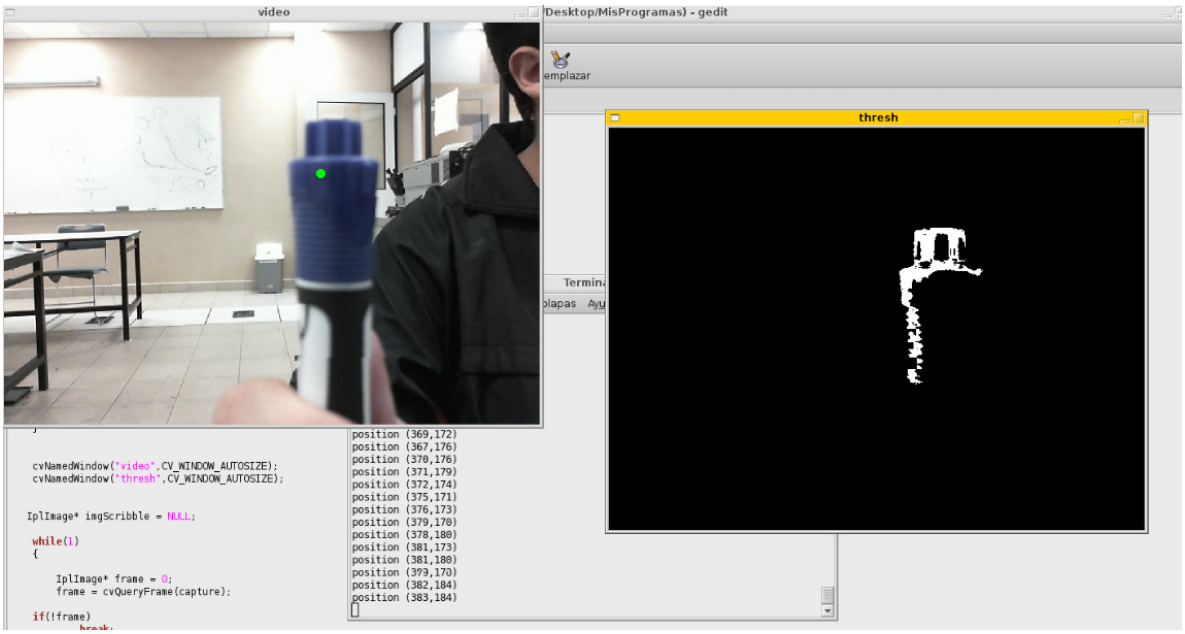


Figura 71. Detector de moviles azules

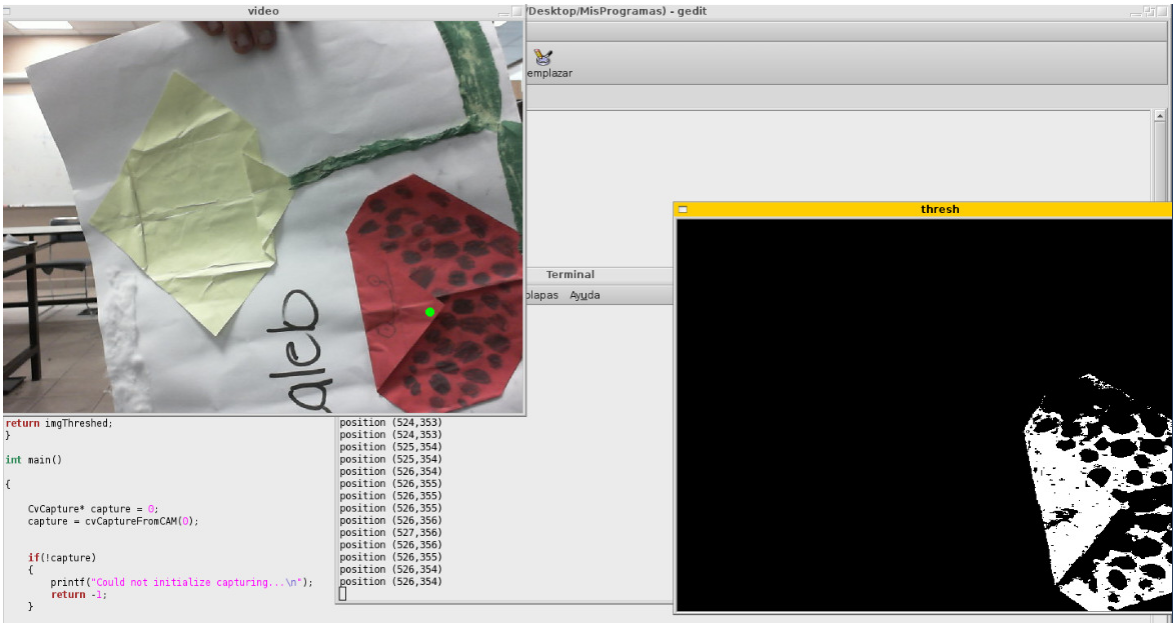


Figura 72. Detector de moviles rojos

Deteccion de móviles por figuras (circulos):

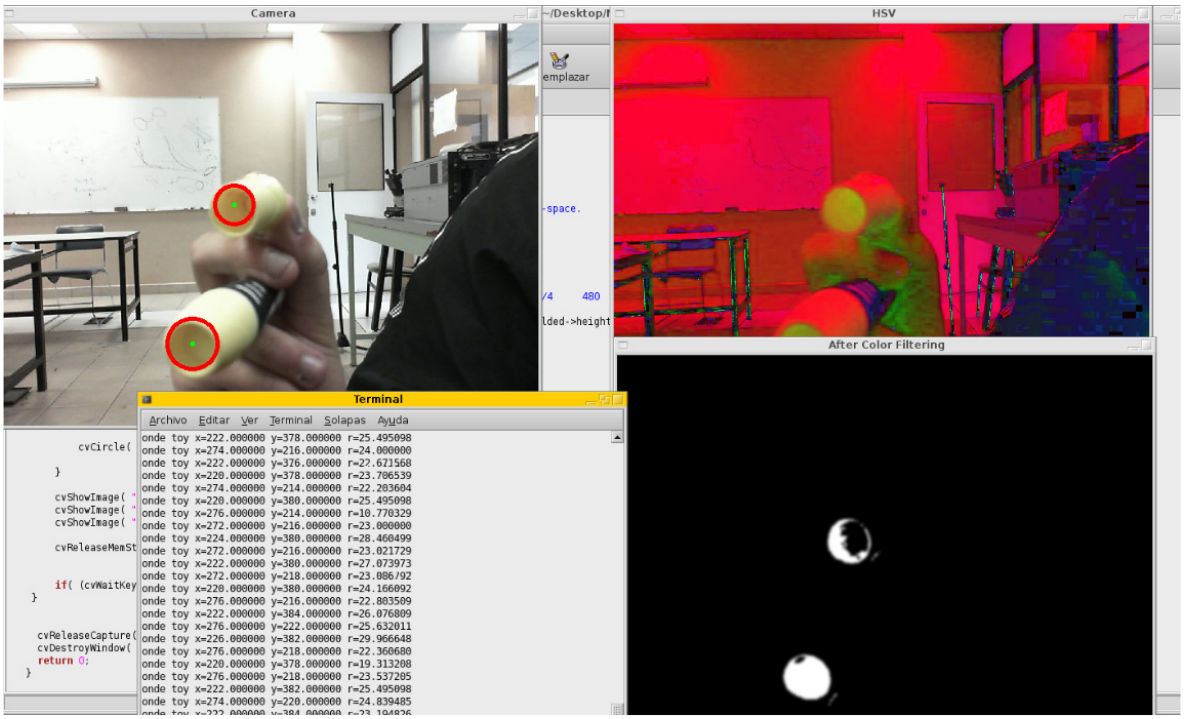


Figura 73. Detector de multiples moviles circulares

Como los colores son sensibles a cambios de iluminación y la detección de figuras arroja multiples resultados, se realizan ambas, es decir, solo se detecta el movimiento del centroide de los objetos que presenten una gama de color predefinida y que además tengan forma circular, el algoritmo utilizado permite detectar móviles rapidos respecto a la velocidad del brazo manipulador y los cuadros por segundo que captura la cámara.

Finalmente se presenta el resultado a 2 camaras simultaneas

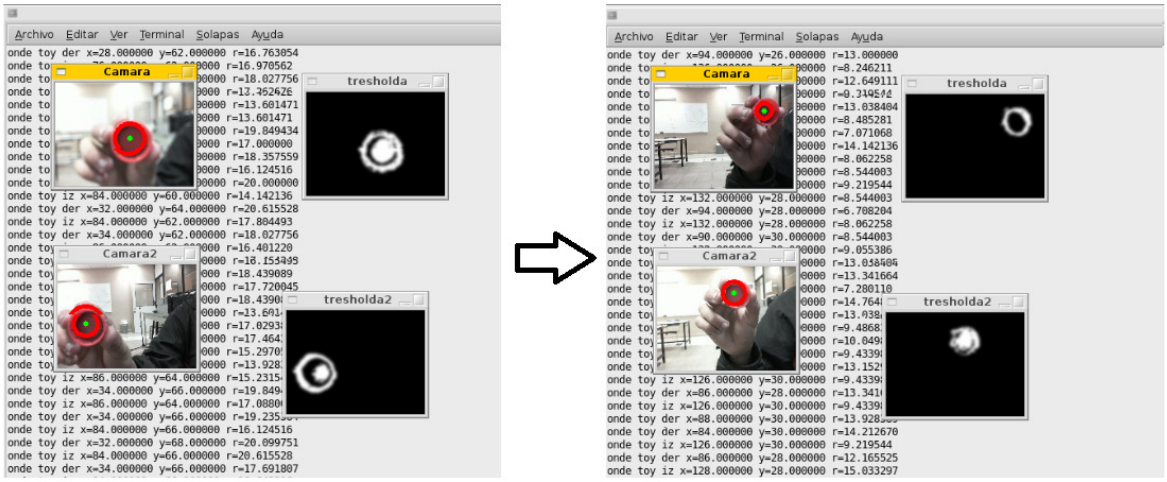


Figura 74. Detector de multiples moviles circulares en 2 camaras

Ahora bien, los presentes resultados solo son validos para la detección del centroide de los móviles y a pesar de que se realiza a 2 camaras, aun no se realiza visión estéreo, para ello se requiere seguir el algoritmo de calibración intrinseco presentado en [4] y [54], asi como la transformación extrínseca que otorgue la referencia al marco principal del brazo manipulador.

El procedimiento consiste en lo siguiente:

Se imprime un tablero de ajedrez con dimensiones conocidas en sus cuadros, estas dimensiones otorgaran la relación pixel-metrica para determinar profundidad en unidades dimensionales cartesianas estandarizadas (cm, mm, m, in, ft, etc) unidades con las que trabaja el brazo manipulador en la configuración no calibrada del control servovisual.

Se sigue el procedimiento de [4], la idea de lo que esto realiza se ilustra en la siguiente figura, Fig. 75 para ello se generan imágenes en ambas cámaras del tablero de ajedrez visto en diversidad de posiciones y orientaciones, Fig. 76

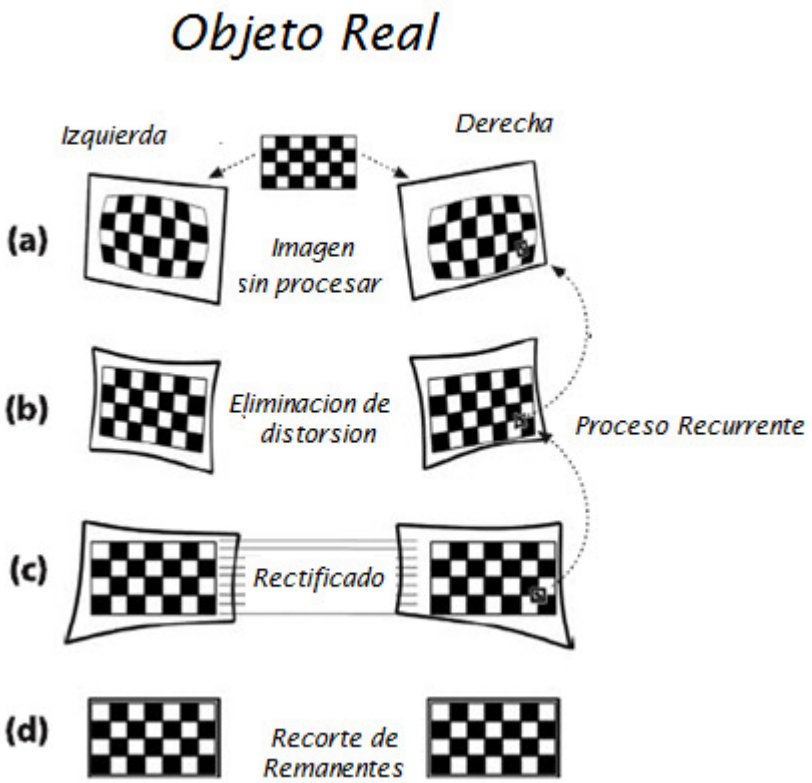


Figura 75. Calibracion estereo

Como ya se explico no se realiza visión estéreo (al menos no la implementacion) por la indisponibilidad del hardware que brinde la resolución requerida (el pixeleado implícito a baja resolución conlleva una pésima calibracion) Fig. 75

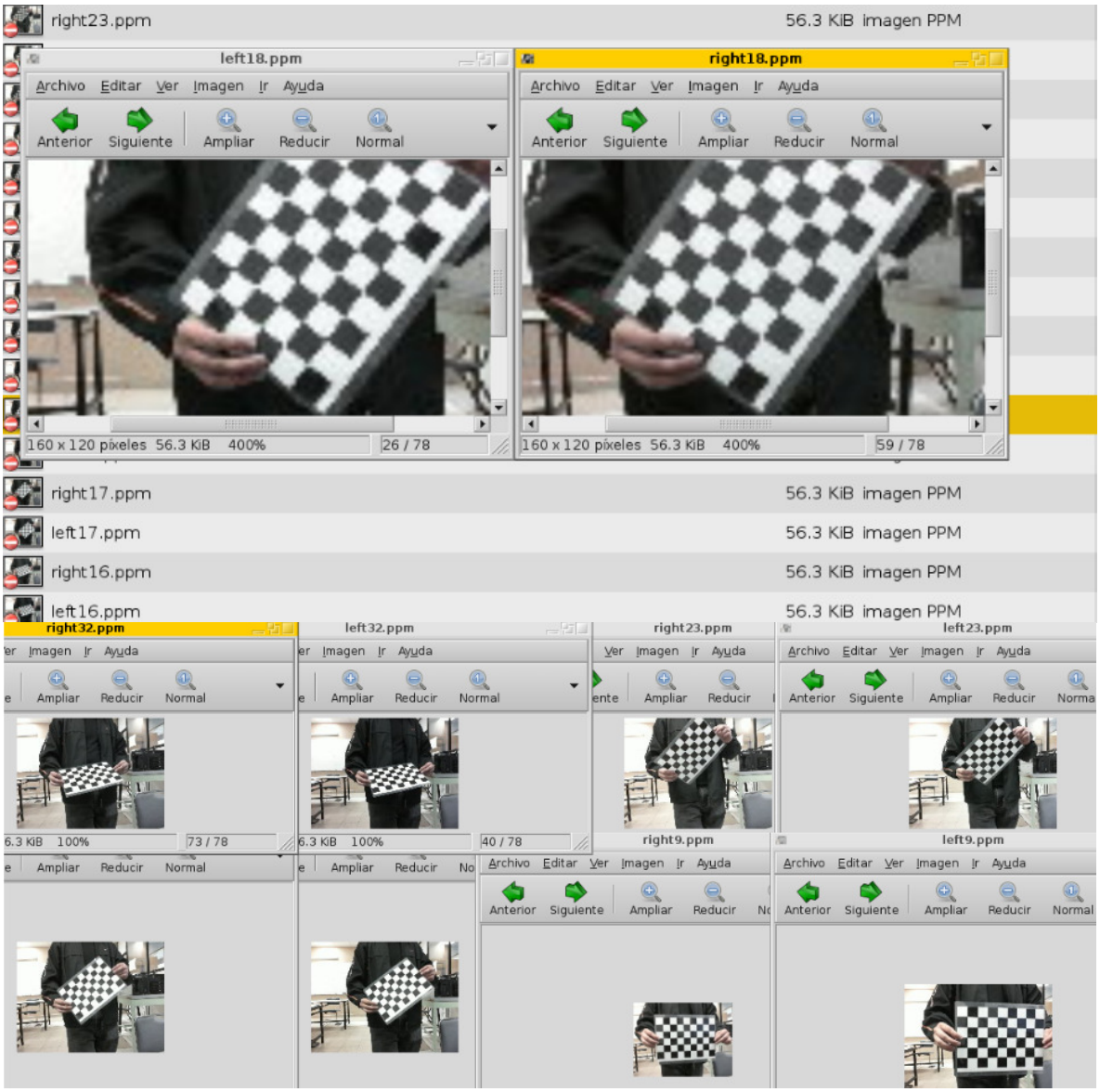


Figura 76. Tomas multiples de camaras izquierda y derecha para la calibracion

Continuando con la explicación, una vez que se dispone de multiples y simultaneas tomas de las cámaras derecha e izquierda del tablero, se realiza el algoritmo de calibración mencionado en [4], el resultado de aplicarlo, además de una corrección de imagen es la llamada transformación de Bouguet (Q), la cual permite obtener la transformación espacial de cámaras a unidades cartesianas

referidas y la minima magnitud sera aproximadamente la longitud de los cuadrados que forman el tablero:

Dados los centroides izquierdo y derecho de cada objeto de la imagen calibrada (u_i, v_i), (u_d, v_d), se obtienen las siguientes relaciones espaciales

$$x = \frac{u_i Q(0,0)+Q(0,3)}{w}$$
$$y = \frac{v_i Q(1,1)+Q(1,3)}{w}$$
$$z = \frac{Q(2,3)}{w} \quad (85)$$

Con:

$$d = u_d - u_i; \quad w = dQ(3,2) + Q(3,3)$$

7.2 Parte Robótica

1) Sobre la sintonización de ganancias

A pesar de la incertidumbre heurística a la cual se somete el controlador presentado en esta tesis, es decir:

Sintonizacion de las ganancias kalman + Sintonizacion de las ganancias de potenciales artificiales

Basta mencionar que los metodos de Lyapunov garantizan la realizan de la tarea de intercepción deseada, la condición como se mostro en el capitulo correspondiente, es que dichas ganancias cumplan con la definición de función definida negativa para el segundo método y definida positiva para el primero, su efecto únicamente se ve reflejado en la forma en que la energía es disipada, pudiendo generar comportamientos sobreamortiguados, subamortiguados, o críticamente amortiguados según corresponda la elección de estas, sin embargo, lo mas conveniente es una respuesta sin sobreimpulso y con tiempo de llegada inmediato, como una forma de automatizar esta selección y hacerla cercana al ideal mencionado, se recurre a la sintonización por inteligencia artificial (redes neuronales), como se menciona en [58] y [59]

Mas alla de esta aclaración, se hace notar que la elección de ganancias por prueba y error, dentro de las acotaciones indicadas por el sistema solo influye en el comportamiento de respuesta (modo de llegar), mas no en el destino (valores deseados de operacion), esto se ilustra en la siguiente imagen, en la cual existen 3 posibles formas de llegar al punto deseado acorde a la sintonización de las ganancias:

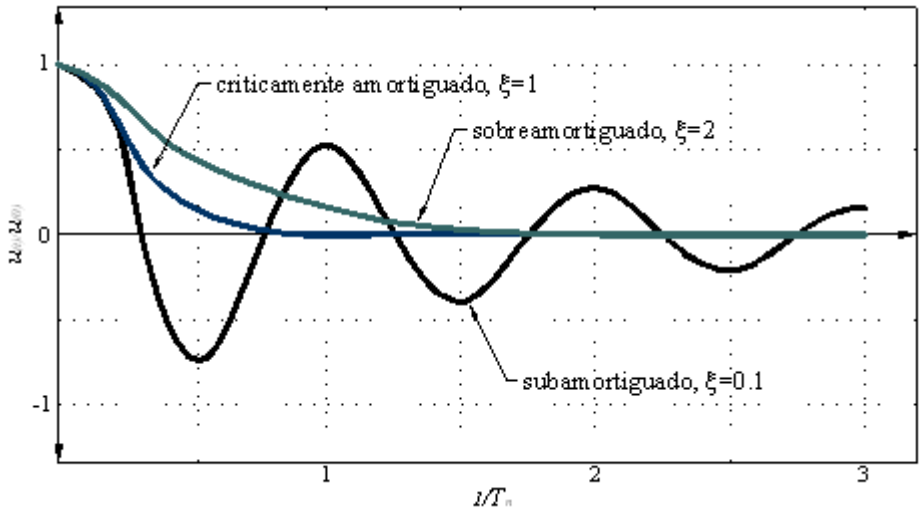


Figura 77. Arribo al punto de operación acorde a la selección de ganancias

2) Sobre la frecuencia del brazo manipulador (constante de tiempo)

Determinar la frecuencia natural de un mecanismo nos ayuda a evitar la llamada zona de resonancia, es decir establecer una zona de operación de velocidades.

Si se parte de la premisa que un robot es análogo al oscilador armonico amortiguado y de hecho la ecuación de un brazo manipulador es la forma generalizada de mencionado caso particular

$$M(q)\frac{d^2q}{dt^2} + C(q,\frac{dq}{dt})\frac{dq}{dt} + G(q) = \tau \qquad \rightarrow \qquad M\frac{d^2x}{dt^2} + B\frac{dx}{dt} + Kx = F \tag{86}$$

De igualmodo se puede hacer una relación para obtener la frecuencia natural del robot

$$\omega \sim \sqrt{\frac{G(q)}{M(q)}} \qquad \rightarrow \qquad \omega = \sqrt{\frac{K}{M}} \tag{87}$$

Sin embargo puede notarse que en el caso de un manipulador, tal frecuencia natural es una función de la posición y por tanto existe un valor para cada configuración que adopte el robot, para evitar de esta forma actuar en un valor no constante, lo que se hace en robotica es calcular la constante de tiempo (relacionada con la frecuencia natural) y esta indica la minima cantidad de tiempo en que debe ser enviado un voltaje a los motores para que estos lleguen satisfactoriamente a su posición sin ocasionar golpeteo (para el robot Mitsubishi que se utiliza en esta tesis, el diseñador de la unidad de arquitectura abierta determino que es de 1ms pues el fabricante indica en sus especificaciones que el manipulador posee velocidad lineal compuesta máxima de 1000mm/s), el procedimiento descrito en [60] es básicamente la caracterización simultanea de todos los motores ante condiciones de carga plena y de velocidad máxima (sin carga).

De esta forma tenemos 2 procesos concurrentes: el control del robot que se realiza cada milisegundo y la adquisición de imágenes cada 33ms, durante esos 33ms se debe garantizar el procesamiento de imágenes y la modificación pertinente a la ley de control, esto se logra nuevamente con el parche RTAI

En este momento se debe pensar que los algoritmos matematicos debiesen ser de tipo discreto, pero es la velocidad de envio y adquisición de datos considerablemente superior a la accion mecanica del robot que este puede operar satisfactoriamente con los modelos continuos de control.

3) Sobre el control del robot

En la Fig. 78 se ilustra el diagrama de control servovisual utilizado en la practica, todo parte del programa hecho en ANSI C, el cual contiene los algoritmos de control ver Fig. 79, sus entradas son los centroides de los objetos y la posición real del robot preprocesada por la tarjeta de adquisición de datos, su salida son los valores de torque (equivalentes de voltaje) para enviar a la tarjeta de adquisición de datos, esta los reenvia a la etapa de potencia (unidad controladora de arquitectura abierta es decir flexible de programar) donde se amplifican y se envian a los motores del brazo manipulador, el cual mediante sus encoders y las camaras cierra el lazo de control enviando información de su posición real.

Con respecto a la conversión de torque de control a voltaje de control, esta se ilustra en la Fig. 80 para el caso particular de la tarjeta Sensoray s626 (esta figura es una profundizacion del bloque: Control del Robot de la ilustración anterior):

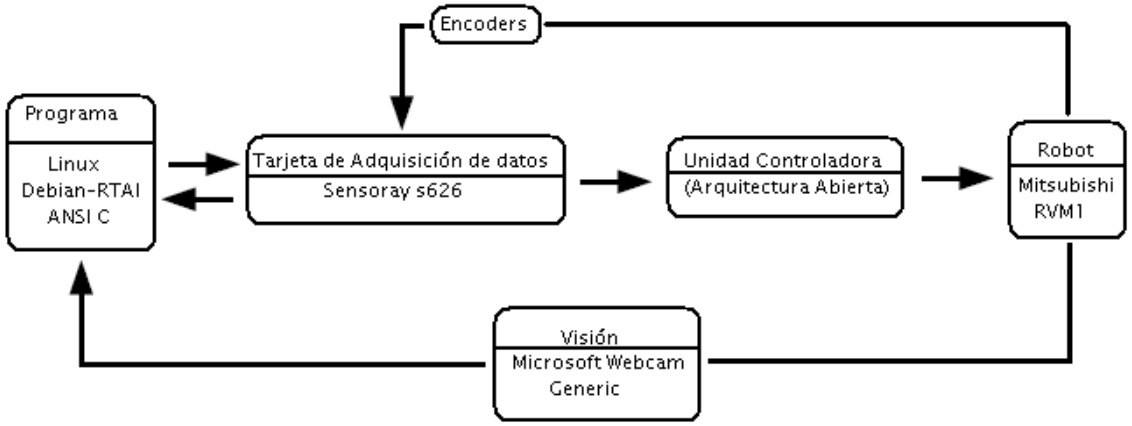


Figura 78. Diagrama de control servovisual caso practico

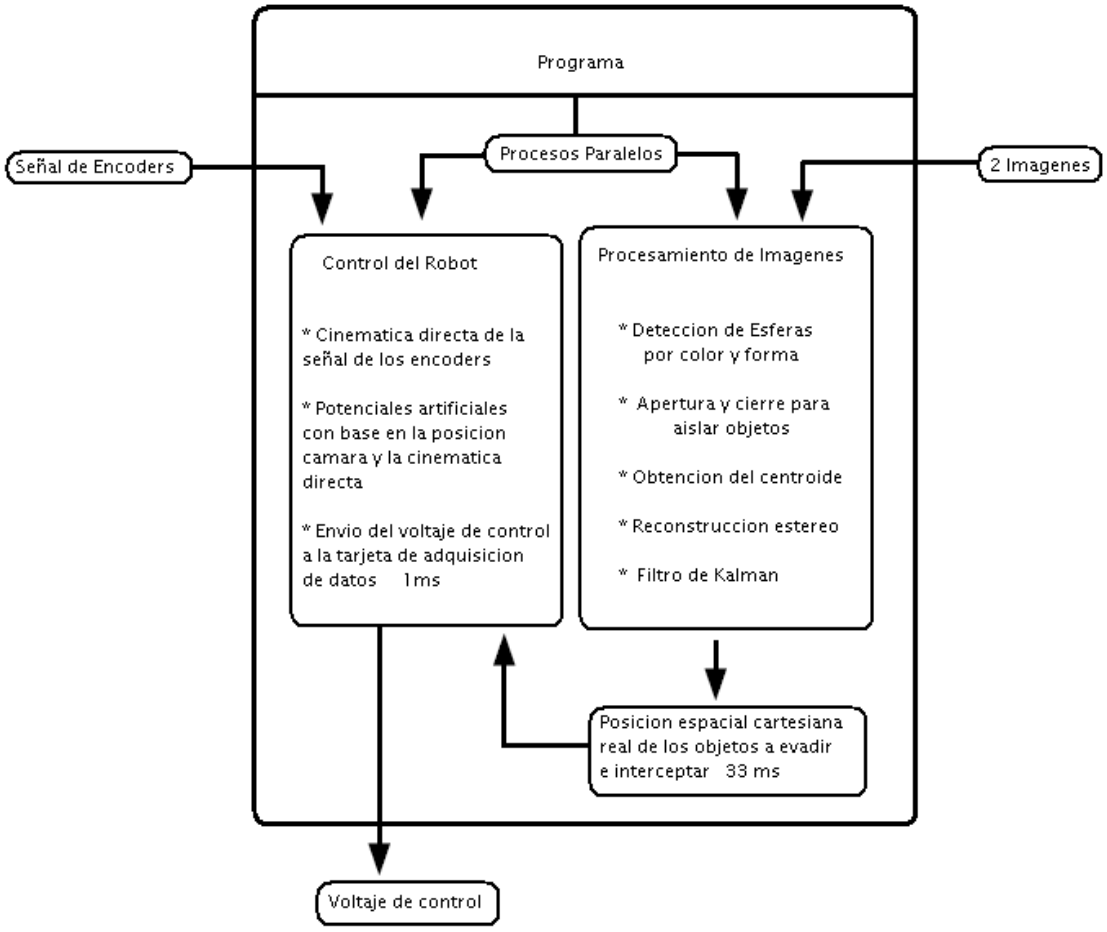


Figura 79. Diagrama de bloques del programa

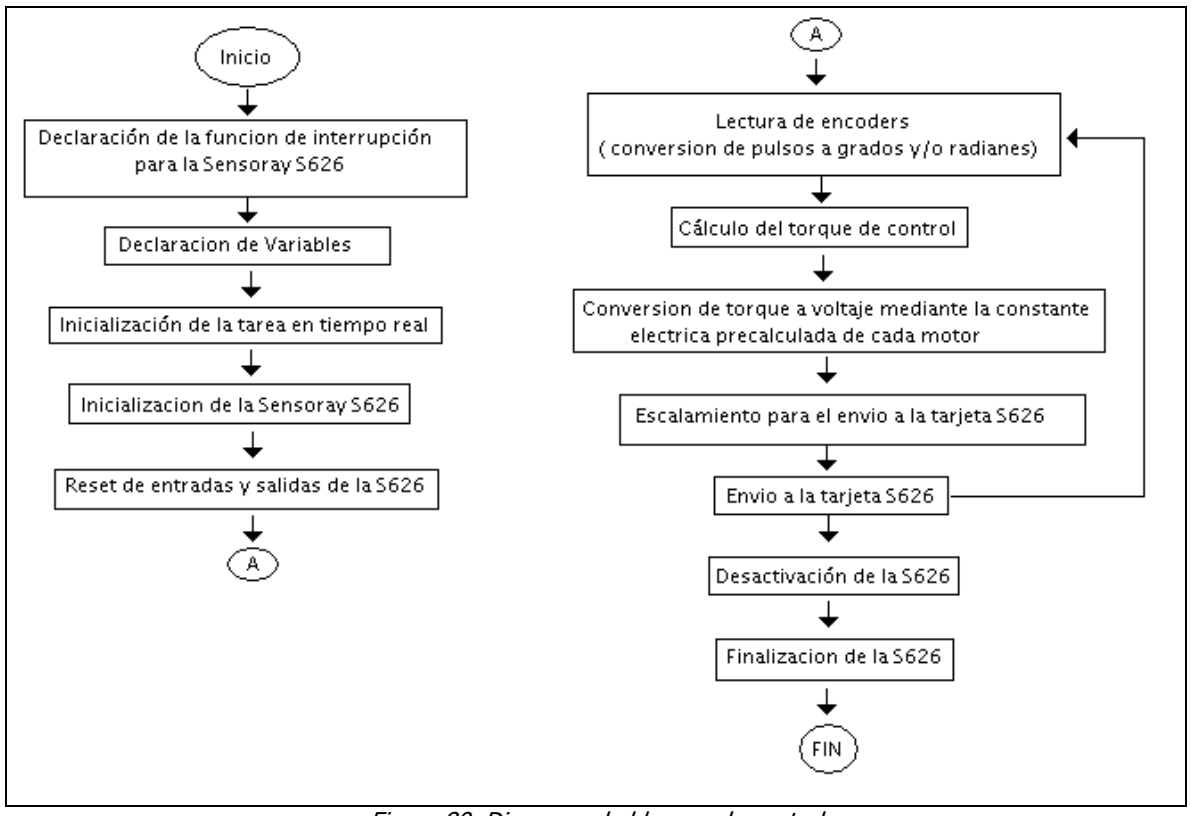


Figura 80. Diagrama de bloques de control

A continuación, se incluyen las graficas experimentales de un control articular PID de regulación realizado en base al manual de la tesis: *"Plataforma de arquitectura abierta basada en el robot Mitsubishi RV-M1 para realizar tareas en tiempo real"* [61], en este control solo se lleva a las articulaciones de un valor actual a uno deseado, la señal de error esta en mili-radianes

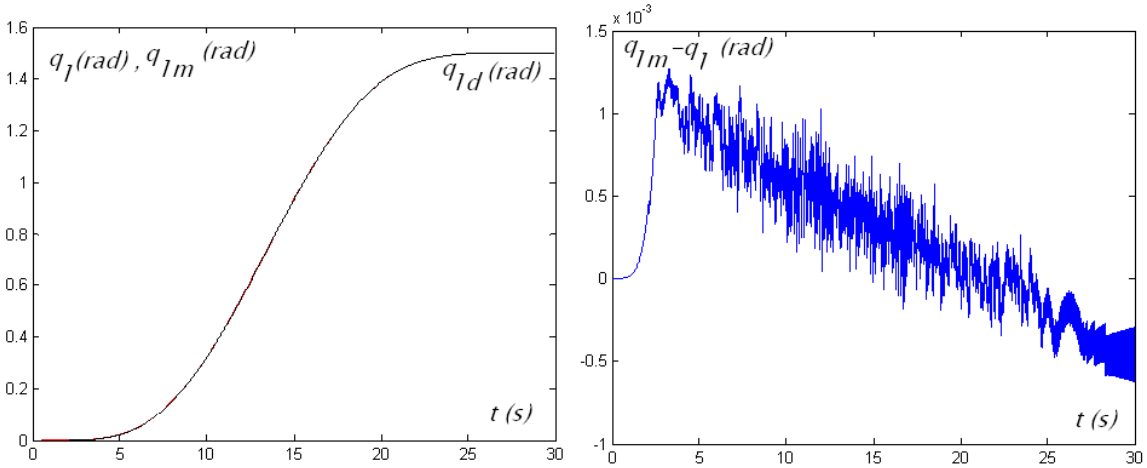


Figura 81. Control de posicionamiento real de la articulación q1

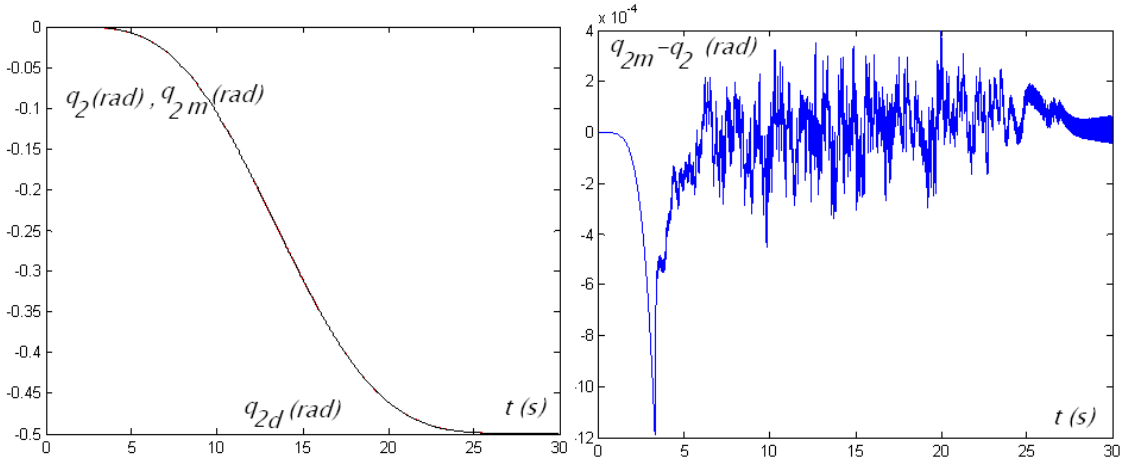


Figura 82. Control de posicionamiento real de la articulación q2

4) Programas requeridos: se presenta a modo de guia una lista de programas necesarios para realizar este tipo de tareas:

1: Programas de integración y derivación numerica, estos son utilizados en la ley de control y en el predictor de Kalman.

2: Programa de operaciones de matrices y vectores, esto se utiliza en todo el proceso y son: producto de matrices, producto de vectores, calculo de inversas y transpuestas, calculo de determinantes, escalamientos y rotaciones.

3: Programa de escritura de datos a archivos e imágenes, el primero se usa para guardar en archivo los datos obtenidos y después utilizarlo por generadores graficos, el segundo es util para la calibración de camaras

3: Programa de deteccion de centroides: este es usado para detectar el movimiento de objetivos y obstáculos

4: Programa de calibración de camaras: este es usado para hacer la calibración estereo de parámetros intrinsecos y extrinsecos

5: Programas de conversión 2D-3D y filtro de Kalman

6: Programa generador de procesos paralelos en RTAI

7: Programa para determinar la Cinematica directa (e inversa si se usa otro algoritmo de control)

8: Programa de control.

9: Simulador

Los mas importantes se incluyen en el apendice correspondiente.

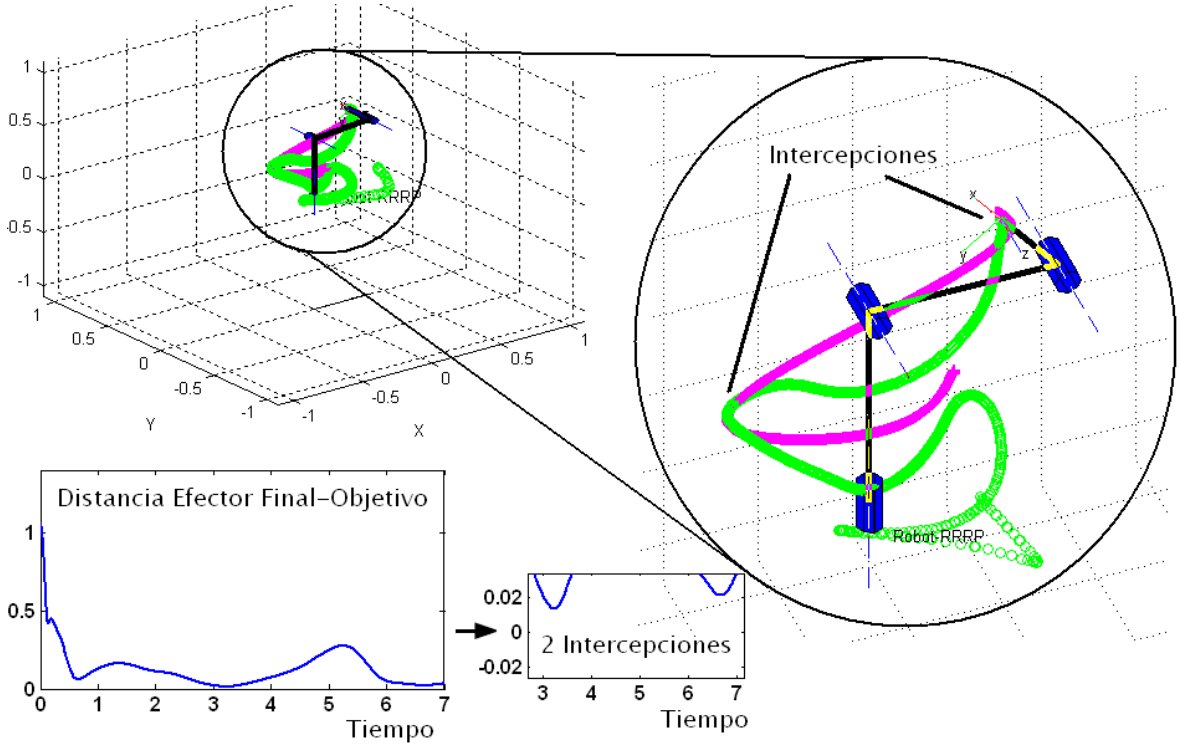


Figura 83. Intercepción de un móvil, en magenta la partícula móvil, en verde y más larga, la trayectoria del efector final

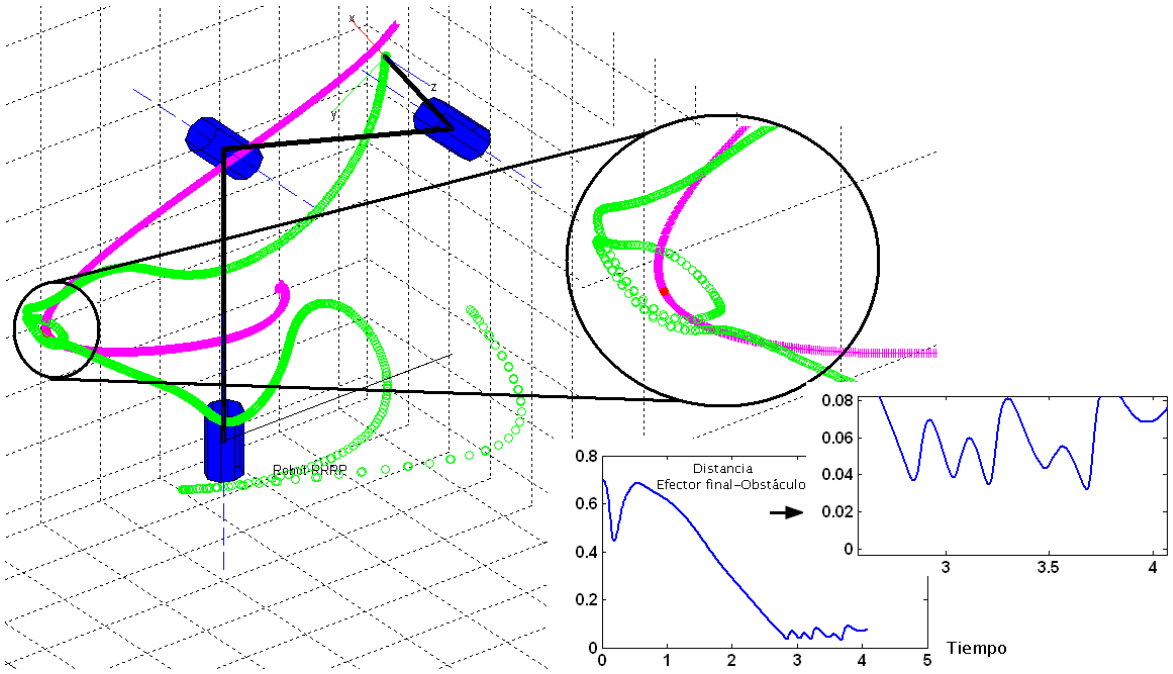


Figura 84. Intercepción y evasión de móviles, se agrega un obstáculo de trayectoria opuesta al objetivo, ver zoom punto rojo

Se describe a continuación el experimento de simulación:

El brazo manipulador partiendo de su posición de reposo intercepta al móvil de trayectoria arbitraria, dada la trayectoria programada ocurren dos intercepciones en el transcurso de la simulación, estas se indican en la figura y adicionalmente se agrega una grafica que permite la visualización de la distancia entre el efector final y el objetivo.

En la siguiente grafica, se agrega un obstáculo que sigue una trayectoria opuesta al objetivo (es decir que ambos están en colisión directa), con ello se forza al manipulador a que en una de las zonas de intercepción evada al obstáculo móvil, lo cual se logra con éxito y se exhibe en la grafica que muestra la distancia entre el obstáculo y el efector final.

Los algoritmos empleados para la simulación fueron:

1. Visión estéreo coplanar monitoreando las posiciones espaciales de objetivo y obstáculo

2. Filtro de Kalman sensible a cambios de aceleración para predecir el movimiento del objetivo

2. Potencial atractor de resorte para generar la trayectoria hacia el objetivo dada su predicción de movimiento por Kalman

3. Potencial repulsor rotonormal para generar la trayectoria de evasión al obstáculo dada su posición por visión estéreo.

Conclusiones y discusión general

Se efectuaron las tareas de evasión e intercepción de objetos múltiples utilizando un simulador de robots tridimensionales (debido al inconveniente de las camaras, no se realizo la implementacion fisica), el mayor problema es que este proceso al ser una sumatoria de observadores y controladores requiere la sintonización heurística (por prueba y error) de cada una de las ganancias de control y predicción.

Adicional a este hecho, puede demostrarse que el controlador propuesto tiene el mismo error de operación de un PD+G (proporcional diferencial con compensación gravitatoria), por tanto para su mejor funcionamiento requiere la incorporación de un termino integrativo Rafael Kelly[49]

Se plantearon y probaron 2 posibles técnicas para eludir mínimos locales en la aplicación de los campos potenciales artificiales, la que demuestra mayor versatilidad y cumple en su diseño con los criterios de estabilidad Lyapunov presentados, es la basada en potenciales de evasión con componentes rotonormales (parte tangencial, parte divergente)

El filtro de Kalman demostró ser eficiente ante tiempos de captura fotográfica rápida, aun así es deseable incorporarle un control de tiempo de predicción así como un generador numérico de matrices para su extensión a tiempos de captura fotográfica lenta y en consecuencia movimiento no lineal

La metodología descrita a lo largo de esta tesis aplica a todo robot móvil cuyo movimiento pueda ser descrito en coordenadas cartesianas y desde luego a brazos manipuladores de base fija y cadena cinemática abierta en 2D y 3D.

Para tareas de interacción con objetos reales se necesita un control hibrido posición-fuerza, pues el efector final acumula inercias en su movimiento (sobre todo en la evasión de los obstaculos) y esto puede causar que llegue con fuerza excesiva al objeto a atrapar pudiendo destruirlo o rebotarlo en la colision

Apéndices

A.1 Trabajos a futuro

1 Evasión de Volúmenes (método de detección de colisiones entre elipsoides)

Como se indica en la Fig. 62 existe la necesidad de evitar colisiones entre volúmenes y no solo en puntos predefinidos, esto es porque cada eslabón del manipulador robótico es una entidad tridimensional.

Ahora bien, como los campos potenciales artificiales de repulsión en la definición de sus ecuaciones presentan una condicional de activación por distancias, una propuesta es envolver cada objeto sujeto a estas fuerzas (eslabones del manipulador, objetivos y obstáculos) en elipsoides como se indica en la siguiente figura

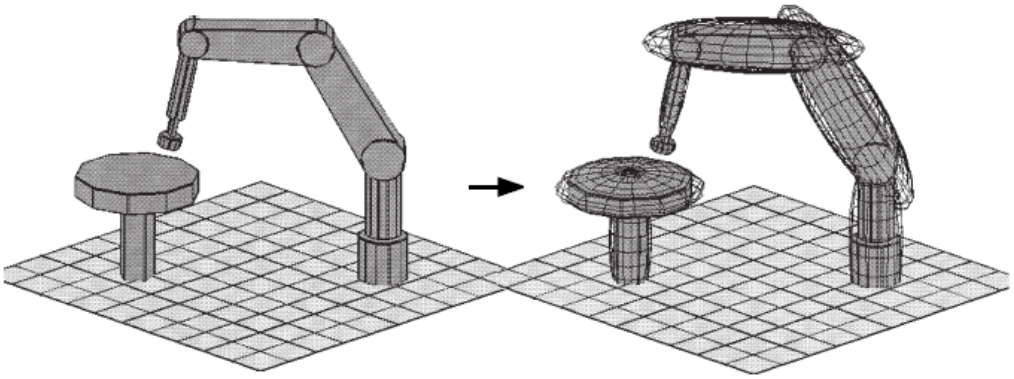


Figura 85. Elipsoides para detección y evasión de colisiones volumétricas, tomado de Boyd [45]

En la literatura el problema de evasión de colisiones entre los elipsoides radica en conocer la distancia mínima entre pares de elipsoides sujeta a 2 restricciones

(multiplicadores de Lagrange), esas restricciones están referidas a la geometría y pose de mencionados cuerpos, dicho problema descrito por Boyd [45] tiene como inconveniente La resolución en línea de un par de polinomios variantes en coeficientes y sujetos a operaciones de traslación y rotación de orden 5 para después evaluar y hallar analíticamente la distancia.

El resultado de esto se resume en la siguiente metodología programable:

Dada la matriz homogénea que describe posición y orientación del centro geométrico de un elipsoide

$$RTA_n = \begin{bmatrix} & trasa_{0nx} \\ Rot_{3x3}^{0n} & trasa_{0ny} \\ & trasa_{0nz} \\ 0_{1x3} & 1_{1x1} \end{bmatrix} \quad (88)$$

El sistema coordenado

$$X = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \quad (89)$$

y las dimensiones del elipsoide cuyos radios son $r_{\alpha n} \quad r_{\beta n} \quad r_{\zeta n}$

$$A_n = \begin{bmatrix} \frac{1}{(r_{\alpha n})^2} & 0 & 0 & 0 \\ 0 & \frac{1}{(r_{\beta n})^2} & 0 & 0 \\ 0 & 0 & \frac{1}{(r_{\varsigma n})^2} & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix} \quad (90)$$

se transforman los ejes

$$A_n^{'} = (RTA_n)^T (A)(RTA_n) \quad (91)$$

Dados 2 elipsoides cuyas ecuaciones son

$$\varepsilon_1(a_1, A_1) = X^T A_1^{'} X \qquad (92)$$

$$\varepsilon_2(a_2, A_2) = X^T A_2^{'} X \qquad (93)$$

el problema es encontrar la distancia mínima entre ellos

$$\min \quad \| \varepsilon_1(a_1, A_1) - \varepsilon_2(a_2, A_2) \| \qquad (94)$$

sujeta a las restricciones de Lagrange

$$\mu_1 = f(\varepsilon_1, \varepsilon_2) \qquad (95)$$

$$\mu_2 = g(\varepsilon_1, \varepsilon_2) \qquad (96)$$

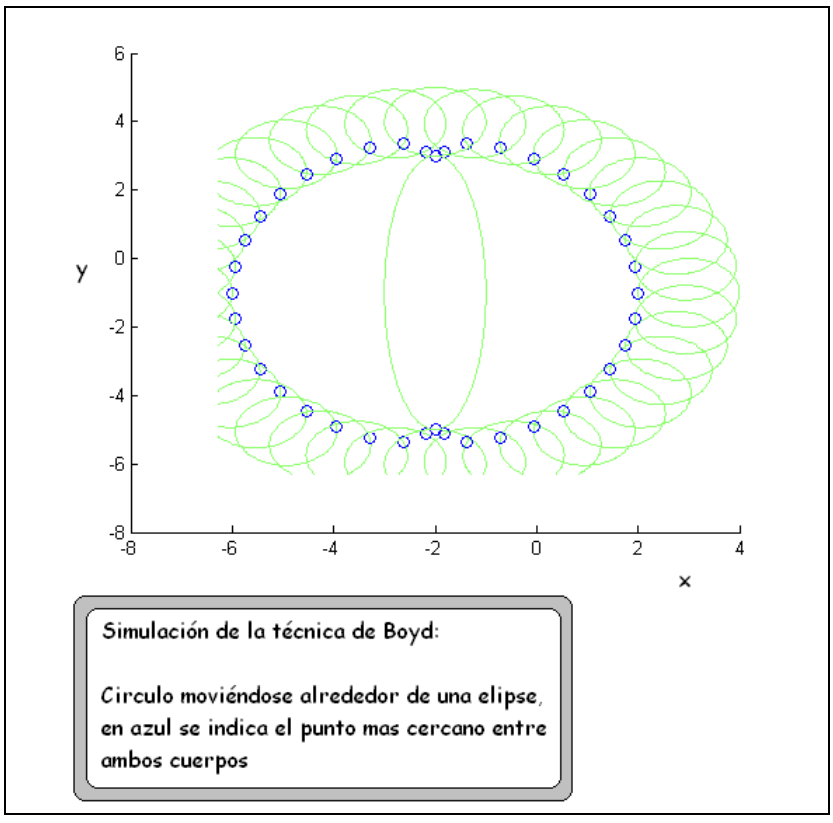


Figura 86. Simulación del método de Boyd [45]

Una técnica alterna es la propuesta por Wang[46], donde se combinan ambas ecuaciones en un modelo matemático mas simple y se indica no el valor de distancia mínima entre cuerpos sino la ocurrencia de contactos e invasión entre ellos con base al signo de las raíces lo cual puede ser condición suficiente para activar las fuerzas de repulsión.

Se genera el polinomio dependiente de λ

$$p(\lambda) = \det(\lambda A_1^{'} - A_2^{'}) \qquad (97)$$

se resuelve $p(\lambda) = 0$

Ocurren los siguientes casos:

1: Si no hay raíces negativas ambos elipsoides se interceptan en mas de un punto (invasión).
2: Si hay 2 raíces negativas iguales se interceptan solo en un punto (contacto).
3: Si hay 2 raíces negativas distintas no hay contacto ni invasión.

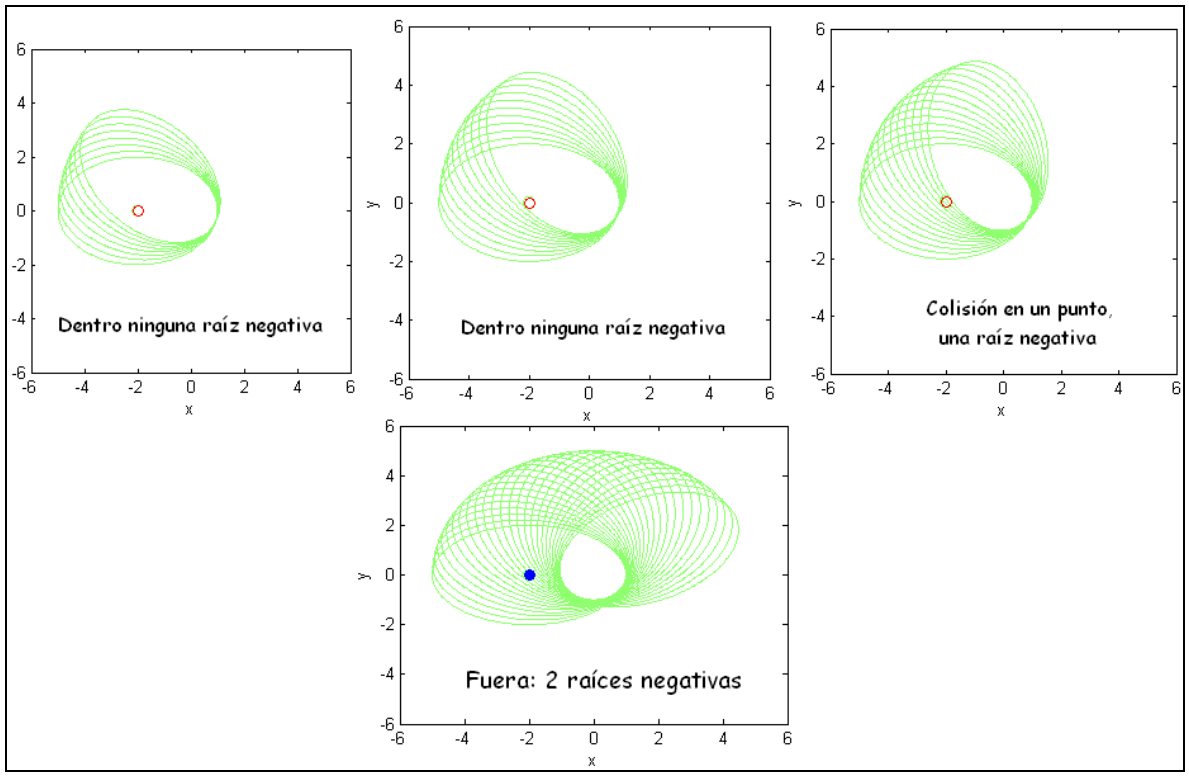


Figura 87. Simulación del método de Wang

De esta forma las publicaciones referidas demuestran la evasión total de volúmenes subdividiéndoles en elipsoides cuyos parámetros de control son los centros geométricos de cada cuerpo sujetos a traslación y rotación obtenibles de la matriz de transformación homogénea asociada, así como sus distancias máximas (longitud, espesor y anchura) tal que al adicionar un valor delta a cada dimensión cubran en totalidad al objeto

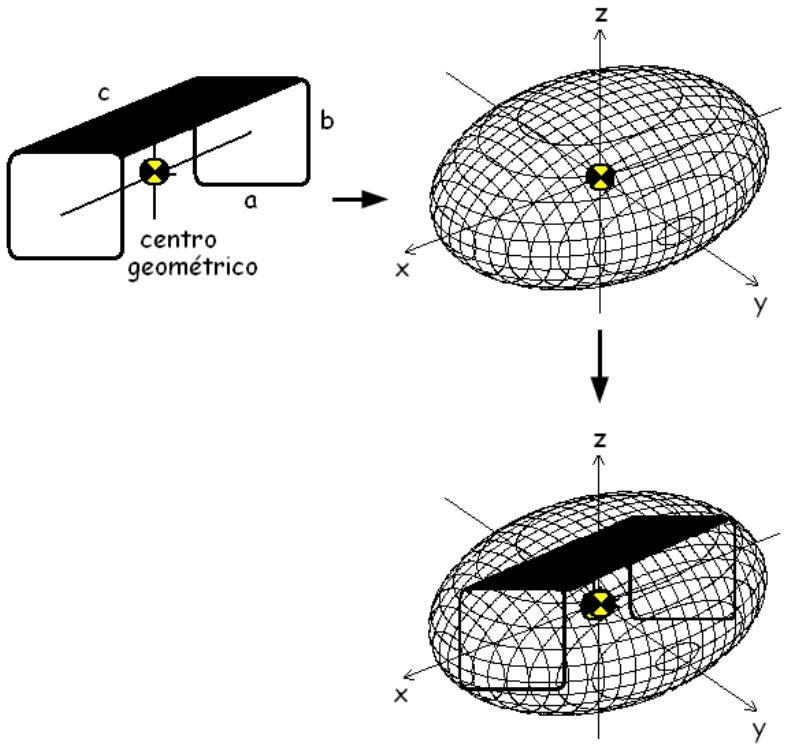


Figura 88. Dimensiones del objeto y del elipsoide

a continuación se presenta un caso simple, un eslabón de dimensiones a-delta,b-delta interactuando con un obstáculo (circunferencia representada como una elipse de radios iguales), para fines didácticos, los eventos se realizan en una geometría bidimensional

La principal ventaja de estos métodos frente a otros que conllevan polígonos es que solo requiere conocer el centro del elipsoide (centro geométrico del eslabón), la ubicación del mismo (dada por la matriz de Transformación homogénea

correspondiente) y los radios (que se obtienen con las dimensiones máximas del eslabón)

2 Predicción con Filtro Extendido de Kalman y regulación del tiempo de predicción

El filtro de Kalman extendido linealiza la función alrededor de la trayectoria del móvil y obtiene una matriz A variante, la ventaja es que aumenta la sensibilidad del sensor (aunado al hecho de que también elimina ruido) el problema es que se requiere conocer las funciones analíticas que describen ese sistema y en caso opuesto aplicar métodos numéricos para calcular el Jacobiano, esta técnica no lineal [15] se describe a continuación

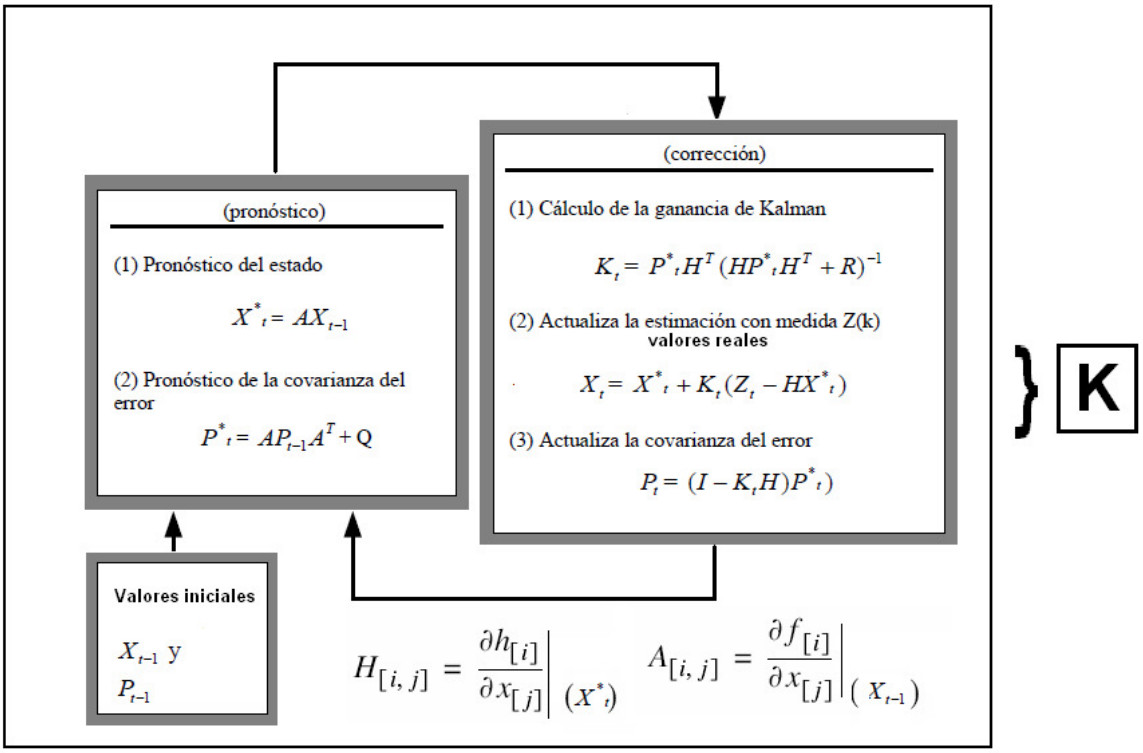


Figura 89. Algoritmo del Filtro de Kalman extendido

Para tener control sobre el tiempo de predicción del filtro de Kalman se plantea utilizar derivadas fraccionarias.

3 Diseño de un controlador de fuerza ejercida base potenciales artificiales

Teniendo la función del campo de fuerzas (atractiva, repulsiva y generadora de flujo constante respectivamente)

$$\nabla f(x, y, z)$$

Y definiendo a sus componentes x, y, z como:

$$F_x \qquad F_y \qquad F_z$$

$$\begin{aligned} -F_x + F_{dx} &= \nabla^2(F_x) - \left(\tfrac{d(F_x)}{dt} + (\nabla f(x, y, z) \cdot \nabla F_x)\right) \\ -F_y + F_{dy} &= \nabla^2(F_y) - \left(\tfrac{d(F_y)}{dt} + (\nabla f(x, y, z) \cdot \nabla F_y)\right) \\ -F_z + F_{dz} &= \nabla^2(F_y) - \left(\tfrac{d(F_z)}{dt} + (\nabla f(x, y, z) \cdot \nabla F_z)\right) \end{aligned} \quad (98)$$

4 Otros

- Control directamente en el espacio Cámara, no calibrado

- Diseño de potenciales atractivos y repulsivos mediante métodos numéricos y/o funciones armónicas

- Incorporar un termino integral al control por jacobiano transpuesto

- Autorregulación de las ganancias en el filtro de Kalman y el controlador

A.2 Programas

A continuación se incluyen los códigos fundamentales de las simulaciones, el software empleado es Matlab ® y Simulink ®, para ejecutarlos se debe contar con los toolboox de Peter Corke y de Mariottinni, a su posterior instalación en el directorio raíz, deben indicarse los siguientes directorios en el PATH de Matlab®

```
addpath 'c:\robot'

addpath 'c:\robot\simulink'

addpath 'c:\eg'
```

Codigo Principal

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Qpp es la salida del Diagrama de bloques de las

%% Figuras [56, 57]

%% u define todas las entradas al sistema

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


function Qpp=Dinamica_RRRkalest(u)


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Posiciones Articulares

%% Velocidades Articulares

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


q1=u(1);

q2=u(2);

q3=u(3);
```

```
qp1=u(4);

qp2=u(5);

qp3=u(6);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Parámetros Cinemáticos (longitudes)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


l1=u(7);%h1

l2=u(8);%h11

l3=u(9);%l1

l4=u(10);%l2

l5=u(11);%l11

l6=u(12);%l21


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Parámetros Dinámicos (masas y tensores de inercia)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


m1=u(13);

m2=u(14);

m3=u(15);


I111=u(16);

I112=u(17);

I113=u(18);

I122=u(19);
```

```
I123=u(20);
I133=u(21);

I211=u(22);
I212=u(23);
I213=u(24);
I222=u(25);
I223=u(26);
I233=u(27);

I311=u(28);
I312=u(29);
I313=u(30);
I322=u(31);
I323=u(32);
I333=u(33);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Gravedad y Tiempo
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

g=u(34);
t=u(35);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Matriz de fricción viscosa
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
Beta(1,1)=u(36);

Beta(2,2)=u(37);

Beta(3,3)=u(38);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Vector de Posición Articular y Vector de Velocidad Articular

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


Q=[q1; q2; q3];

Qp=[qp1; qp2; qp3];


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Velocidad Articular deseada al llegar a cada punto

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


Qpd=[0; 0; 0];


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Error de Velocidad Angular

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


ep=Qp-Qpd;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Matriz de masas e inercias

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
M(1,1)=I133 + cos(q2 + q3)*(I322*cos(q2 + q3) + I312*sin(q2 + q3)) + sin(q2 + q3)*(I312*cos(q2 + q3) + I311*sin(q2 + q3)) + m3*(l6*cos(q2 + q3) + l3*cos(q2))^2 + cos(q2)*(I222*cos(q2) + I212*sin(q2)) + sin(q2)*(I212*cos(q2) + I211*sin(q2)) + m2*((l5^2*cos(2*q2))/2 + l5^2/2);

M(1,2)=I323*cos(q2 + q3) + I313*sin(q2 + q3) + I223*cos(q2) + I213*sin(q2);

M(1,3)=I323*cos(q2 + q3) + I313*sin(q2 + q3);

M(2,1)=I323*cos(q2 + q3) + I313*sin(q2 + q3) + I223*cos(q2) + I213*sin(q2);

M(2,2)=m3*l3^2 + 2*m3*cos(q3)*l3*l6 + m2*l5^2 + m3*l6^2 + I233 + I333;

M(2,3)=m3*l6^2 + l3*m3*cos(q3)*l6 + I333;

M(3,1)=I323*cos(q2 + q3) + I313*sin(q2 + q3);

M(3,2)=m3*l6^2 + l3*m3*cos(q3)*l6 + I333;

M(3,3)=m3*l6^2 + I333;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Matriz de efectos Centripetos y Coriolis

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


C(1,1)=I212*qp2*cos(2*q2) + (I211*qp2*sin(2*q2))/2 - (I222*qp2*sin(2*q2))/2 + I312*qp2*cos(2*q2 + 2*q3) + I312*qp3*cos(2*q2 + 2*q3) + (I311*qp2*sin(2*q2 + 2*q3))/2 + (I311*qp3*sin(2*q2 + 2*q3))/2 - (I322*qp2*sin(2*q2 + 2*q3))/2 - (I322*qp3*sin(2*q2 + 2*q3))/2 - (l5^2*m2*qp2*sin(2*q2))/2 - (l3^2*m3*qp2*sin(2*q2))/2 - (l6^2*m3*qp2*sin(2*q2 + 2*q3))/2 - (l6^2*m3*qp3*sin(2*q2 + 2*q3))/2 - l3*l6*m3*qp2*sin(2*q2 + q3) - (l3*l6*m3*qp3*sin(2*q2 + q3))/2 - (l3*l6*m3*qp3*sin(q3))/2;

C(1,2)=I212*qp1*cos(2*q2) - m3*qp1*sin(2*q2 + q3)*l3*l6 - (m2*qp1*sin(2*q2)*l5^2)/2 - (m3*qp1*sin(2*q2 + 2*q3)*l6^2)/2 - (m3*qp1*sin(2*q2)*l3^2)/2 + (I211*qp1*sin(2*q2))/2 - (I222*qp1*sin(2*q2))/2 + I312*qp1*cos(2*q2 + 2*q3) + (I311*qp1*sin(2*q2 + 2*q3))/2 - (I322*qp1*sin(2*q2 + 2*q3))/2 + I313*qp2*cos(q2 + q3) + I313*qp3*cos(q2 + q3) - I323*qp2*sin(q2 + q3) - I323*qp3*sin(q2 + q3) + I213*qp2*cos(q2) - I223*qp2*sin(q2);

C(1,3)=I312*qp1*cos(2*q2 + 2*q3) + (I311*qp1*sin(2*q2 + 2*q3))/2 - (I322*qp1*sin(2*q2 + 2*q3))/2 + I313*qp2*cos(q2 + q3) + I313*qp3*cos(q2 + q3) - I323*qp2*sin(q2 + q3) - I323*qp3*sin(q2 + q3) - (l6^2*m3*qp1*sin(2*q2 + 2*q3))/2 - (l3*l6*m3*qp1*sin(2*q2 + q3))/2 - (l3*l6*m3*qp1*sin(q3))/2;

C(2,1)=(qp1*(m3*sin(2*q2)*l3^2 + 2*m3*sin(2*q2 + q3)*l3*l6 + m2*sin(2*q2)*l5^2 + m3*sin(2*q2 + 2*q3)*l6^2 - 2*I312*cos(2*q2 + 2*q3) - I311*sin(2*q2 + 2*q3) + I322*sin(2*q2 + 2*q3) - 2*I212*cos(2*q2) - I211*sin(2*q2) + I222*sin(2*q2)))/2;
```

```
C(2,2)=-l3*l6*m3*qp3*sin(q3);

C(2,3)=-l3*l6*m3*sin(q3)*(qp2 + qp3);

C(3,1)=(qp1*(l322*sin(2*q2 + 2*q3) - l311*sin(2*q2 + 2*q3) - 2*l312*cos(2*q2 + 2*q3) + l6^2*m3*sin(2*q2 +
2*q3) + l3*l6*m3*sin(q3) + l3*l6*m3*sin(2*q2 + q3)))/2;

C(3,2)=l3*l6*m3*qp2*sin(q3);

C(3,3)=0;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Vector de pares gravitacionales

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


G(1,1)=0;

G(2,1)=g*m3*(l6*cos(q2 + q3) + l3*cos(q2)) + g*l5*m2*cos(q2);

G(3,1)=g*l6*m3*cos(q2 + q3);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Matriz de Ganancia Diferencial

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


Kd=[5,0,0;0,5,0;0,0,5];


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%% CONTROLADOR POR JACOBIANO TRANSPUESTO CON FUERZA POTENCIAL

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% trayectorias reales de los obstáculos y objetivos
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% if (t<10)
  x=0.3*cos(t);
  y=0.3*exp(-t*t);
  z=t/10;

  xobs=0.3*cos(-t);
  yobs=0.3*exp(-t*t);
  zobs=-(t-5)/10;

% elseif((t>=10)&(t<19))
%    x=0.3*cos(t-6);
%    y=0.3*exp(-(t-6)*(t-6));
%    z=(t-6)/10;
%
%    xobs=0.3*cos(3);
%    yobs=0.3*exp(-3*3);
%    zobs=3/10;
%
% else (t>=19)
%    z=0.5;
%    y=0;
%    x=t/50;
%
```

```
%     xobs=0.3*cos(3);
%     yobs=0.3*exp(-3*3);
%     zobs=3/10;
% end

xrep1=xobs;
xrep2=yobs;
xrep3=zobs;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Corrección marcos coordenados
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

xd1=-y;
xd2=x;
xd3=z;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Posición real de Obstáculos
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

posObs=[xrep1;xrep2;xrep3];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Posición real de Objetivos
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
posO=[xd1;xd2;xd3];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Posición real del Efector Final acorde a la Cinemática Directa
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

posRo=[ cos(q1)*(l4*cos(q2 + q3) + l3*cos(q2));
sin(q1)*(l4*cos(q2 + q3) + l3*cos(q2));
l1 + l4*sin(q2 + q3) + l3*sin(q2)];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Jacobiano Lineal del Efector Final
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Jaco(1,1)= -sin(q1)*(l4*cos(q2+q3)+l3*cos(q2));
Jaco(1,2)= -cos(q1)*(l4*sin(q2+q3)+l3*sin(q2));
Jaco(1,3)= -l4*sin(q2+q3)*cos(q1);
Jaco(2,1)= cos(q1)*(l4*cos(q2+q3)+l3*cos(q2));
Jaco(2,2)= -sin(q1)*(l4*sin(q2+q3)+l3*sin(q2));
Jaco(2,3)= -l4*sin(q2+q3)*sin(q1);
Jaco(3,1)= 0;
Jaco(3,2)= l4*cos(q2+q3)+l3*cos(q2);
Jaco(3,3)= l4*cos(q2+q3);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Jacobiano Transpuesto
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
Jacobot=transpose(Jaco);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Parámetros Intrínsecos y Extrínsecos de las Cámaras en configuración Estereo
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Rd=rotoz(0)*rotoy(0)*rotox(0);
Rd2=rotoz(0)*rotoy(0)*rotox(0);
td=[-1,-5,0]';
td2=[1,-5,0]';
Hd=f_Rt2H(Rd,td);
Hd2=f_Rt2H(Rd2,td2);
Kdd=eye(3);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% DIBUJO AUXILIAR DE LAS CAMARAS
% f_3Dframe(Hd,'k',0.2,'_{cam}');
% f_3Dcamera(Hd,'r',0.2); %3D pin-hole camera
% f_3Dframe(Hd2,'k',0.2,'_{cam}');
% f_3Dcamera(Hd2,'r',0.2); %3D pin-hole camera
% SOLO SIRVE PARA SU COLOCACION AL EJECUTARLO ES MEJOR
% DEJAR ESTA PARTE COMO COMENTARIO
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% COLOCACION DE MARCAS VISUALES EN  Obstáculos, Objetivos y Efector final del Robot
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
P3=[posObs(1,1);posObs(2,1);posObs(3,1)];
P2=[posRo(2,1);-posRo(1,1);posRo(3,1)];
P=[x;y;z];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% PROYECCION BIDIMENSIONAL FOTOGRAFICA DE CADA MARCA
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% cámara izquierda
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

[ud,vd]=f_perspproj(P,Hd,Kd,1);
[ud2,vd2]=f_perspproj(P2,Hd,Kd,1);
[udobs,vdobs]=f_perspproj(P3,Hd,Kd,1);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% cámara derecha
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

[u2d,v2d]=f_perspproj(P,Hd2,Kd,1);
[u2d2,v2d2]=f_perspproj(P2,Hd2,Kd,1);
[u2d2obs,v2d2obs]=f_perspproj(P3,Hd2,Kd,1);

posizqu1=ud;
posizqv1=vd;
posizqu2=ud2;
```

```
posizqv2=vd2;
posizqu3=udobs;
posizqv3=vdobs;

posderu1=u2d;
posderv1=v2d;
posderu2=u2d2;
posderv2=v2d2;
posderu3=u2d2obs;
posderv3=v2d2obs;

uizq=[posizqu1;posizqu2;posizqu3];
vizq=[posizqv1;posizqv2;posizqv3];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Total de posiciones cámara izquierda
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

posizq=[uizq;vizq];

uder=[posderu1;posderu2;posderu3];
vder=[posderv1;posderv2;posderv3];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Total de posiciones cámara derecha
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
posder=[uder;vder];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Total de posiciones reales
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

xobjeto=[x;posRo(2,1);posObs(1,1)];
yobjeto=[y;-posRo(1,1);posObs(2,1)];
zobjeto=[z;posRo(3,1);posObs(3,1)];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% RECONSTRUCCION ESTEREO
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

f=1;
Beta=abs(td(1)-td2(1));
xobreco=(uizq*Beta)./(uizq-uder);
yobreco=(vizq*Beta)./(uizq-uder);
zobreco=(f*Beta)./(uizq-uder);
xcama=xobreco-1;
ycama=zobreco-5;
zcama=-yobreco;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Total de posiciones Estereo
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
 posOcam=[-ycama(1,1);xcama(1,1);zcama(1,1)];
 posRocam=[-ycama(2,1);xcama(2,1);zcama(2,1)];
 posObcam=[-ycama(3,1);xcama(3,1);zcama(3,1)];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% FILTRO DE KALMAN aplicado a la posicion del objetivo en
%% reconstruccion Estereo el disparador fotografico se implementa
%% en la funcion kalman
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
posOcamK = kalman01c(posOcam);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Distancias y errores cartesianos entre el Efector Final así como los Obstáculos y Objetivos
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

erpoc=(posOcamK-posRo);
erpobs=(posObcam-posRo);
disobs=dist(transpose(posRo),posObcam);
posRoaux=transpose(posRo);
tancia=dist(posRoaux,posOcamK);
sale=[xobjeto(1,1);
    posOcamK(1,1);
    yobjeto(1,1);
    posOcamK(2,1);
    zobjeto(1,1);
    posOcamK(3,1)];
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Campo POTENCIAL ARTIFICIAL  de Fuerzas de Atracción
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

mRo=2;
m1c=200;
Fatra=(mRo*m1c*erpoc);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Campo de Fuerzas de Repulsión
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

segu=1;
if (disobs<=segu)
Frepu=(0.008*((1/disobs)-(1/segu)))*(1/(disobs*disobs))*(erpobs/disobs);
else
Frepu=zeros(3,1);
End

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Matrices de rotación para las Fuerzas de Repulsión 3D
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

a=pi/4;
Rz=[cos(a),-sin(a),0;sin(a),cos(a),0;0,0,1];
Rx=[1,0,0;0,cos(a),-sin(a);0,sin(a),cos(a)];
Ry=[cos(a),0,sin(a);0,1,0;-sin(a),0,cos(a)];
```

```
Frerepu=Rx*Ry*Rz*Frepu;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Fuerza Cartesiana Total
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
F=Fatra+Frerepu;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Despliegue Grafico de las Trayectorias del Efector Final, Objetivo y obstáculo
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

plot3(x,y,z,'+m');
hold on
plot3(posRo(2,1),-posRo(1,1),posRo(3,1),'og');
plot3(posObcam(2,1),-posObcam(1,1),posObcam(3,1),'or');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% LEY DE CONTROL
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Tao=(Jacobot*F)-(Kd*ep)+G;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% despeje y salida de Qpp
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Qpp=[inv(M)*(Tao-C*Qp-G-Beta*Qp)];
```

Código Para la obtención de las ecuaciones de M, C y G a partir de las matrices de transformación homogénea

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Ejemplo elaborado para un Robot de 6GDL
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear all
clc

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Definición de todas las variables simbólicas
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
syms q1 q2 q3 q4 q5 q6 h1 l1 l2 l3 h11 l11 l22 l33 l55 m1 m2 m3 m4 m5 m6;
syms I111 I112 I113 I122 I123 I133 I211 I212 I213 I222 I223 I233;
syms I311 I312 I313 I322 I323 I333 I411 I412 I413 I422 I423 I433;
syms I511 I512 I513 I522 I523 I533 I611 I612 I613 I622 I623 I633;
syms q1p q2p q3p q4p q5p q6p;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Tensores de Inercia
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
I1=[
I111    I112    I113
I112    I122    I123
I113    I123    I133
];
```

I2=[

I211 I212 I213

I212 I222 I223

I213 I223 I233

];

I3=[

I311 I312 I313

I312 I322 I323

I313 I323 I333

];

I4=[

I411 I412 I413

I412 I422 I423

I413 I423 I433

];

I5=[

I511 I512 I513

I512 I522 I523

I513 I523 I533

];

I6=[

I611 I612 I613

I612 I622 I623

```
I613    I623    I633
];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Matrices de Transformacion
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
T01=[
cos(q1)  0        sin(q1)   0
sin(q1)  0        -cos(q1)  0
0        1        0         h1
0        0        0         1
];

T12=[
cos(q2)  -sin(q2)  0        l1*cos(q2)
sin(q2)  cos(q2)   0        l1*sin(q2)
0        0         1        0
0        0         0   1
];

T23=[
cos((pi/2)+q3)    0        sin((pi/2)+q3)    0
sin((pi/2)+q3)    0        -cos((pi/2)+q3)   0
0        1        0        0
0        0        0        1
];

T34=[
```

```
cos(q4)  0        -sin(q4)  0
sin(q4)  0        cos(q4)   0
0        -1       0         l2
0        0        0         1
];

T45=[
cos(q5)  0        sin(q5)    0
sin(q5)  0        -cos(q5)   0
0        1        0          0
0        0        0          1
];

T56=[
cos(q6)  -sin(q6)  0         0
sin(q6)  cos(q6)             0         0
0        0         1         l3
0        0         0         1
]

T02=T01*T12;
T03=T02*T23;
T04=T03*T34;
T05=T04*T45;
T06=T05*T56;
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Vectores de traslación obtenidos de las anteriores
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
t06=T06(1:3,4);
t01=T01(1:3,4);
t02=T02(1:3,4);
t03=T03(1:3,4);
t04=T04(1:3,4);
t05=T05(1:3,4);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Ejes principales de movimiento obtenidos de la parte rotacional también a partir de las matrices de
%% transformación
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
z0=[0;0;1];
z1=T01(1:3,3);
z2=T02(1:3,3);
z3=T03(1:3,3);
z4=T04(1:3,3);
z5=T05(1:3,3);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Jacobiano del Efector final
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
JaEf=[
cross(z0,t06) cross(z1,(t06-t01)) cross(z2,(t06-t02)) cross(z3,(t06-t03)) cross(z4,(t06-t04))  cross(z5,(t06-t05))
z0                  z1                          z2                    z3                    z4                    z5
];

% Comprobación del jacobiano analítico
% Compro1=simple((diff(t06,q1))-(JaEf(1:3,1)));
% Compro2=simple((diff(t06,q2))-(JaEf(1:3,2)));
% Compro3=simple((diff(t06,q3))-(JaEf(1:3,3)));
% Compro4=simple((diff(t06,q4))-(JaEf(1:3,4)));
% Compro5=simple((diff(t06,q5))-(JaEf(1:3,5)));
% Compro6=simple((diff(t06,q6))-(JaEf(1:3,6)));


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Matrices de transformación a los centros de masa
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

T011=[
cos(q1)  -sin(q1)  0         0
sin(q1)   cos(q1)  0         0
0          0         1         h11
0          0         0         1
];

T122=[
cos(q2)  -sin(q2)  0         l11*cos(q2)
```

```
sin(q2)  cos(q2)  0        l11*sin(q2)
0        0        1        0
0        0        0        1
];

T233=[
cos(q3)  -sin(q3)  0        l22*cos(q3)
sin(q3)  cos(q3)  0        l22*sin(q3)
0        0        1        0
0        0        0        1
];

T344=[
cos(q4)  -sin(q4)  0        0
sin(q4)  cos(q4)  0        0
0        0        1        l33
0        0        0        1
];

T455=[
cos(q5)  -sin(q5)  0        0
sin(q5)  cos(q5)  0        0
0        0        1        0
0        0        0        1
];
```

```
T566=[
cos(q6)  -sin(q6)  0        0
sin(q6)   cos(q6)  0        0
0         0        1        l55
0         0        0        1
];

T022=T01*T122;
T033=T02*T233;
T044=T03*T344;
T055=T04*T455;
T066=T05*T566;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% vectores de traslación  y matrices de rotación de las anteriores
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
t011=T011(1:3,4);
t022=T022(1:3,4);
t033=T033(1:3,4);
t044=T044(1:3,4);
t055=T055(1:3,4);
t066=T066(1:3,4);

R011=T011(1:3,1:3);
R022=T022(1:3,1:3);
R033=T033(1:3,1:3);
R044=T044(1:3,1:3);
```

```
R055=T055(1:3,1:3);
R066=T066(1:3,1:3);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% vector de ceros
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
V0=zeros(3,1);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Calculo de cada Jacobiano de centro de masas requerido para definir M
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Jm1=[
cross(z0,t011)   V0   V0   V0   V0   V0
z0               V0   V0   V0   V0   V0
];

Jm2=[
cross(z0,t022)   cross(z1,(t022-t01))   V0   V0   V0   V0
z0               z1                     V0   V0   V0   V0
];

Jm3=[
cross(z0,t033)   cross(z1,(t033-t01))   cross(z2,(t033-t02))   V0   V0   V0
z0               z1                     z2                     V0   V0   V0
];

Jm4=[
```

```
cross(z0,t044)    cross(z1,(t044-t01))        cross(z2,(t044-t02))        cross(z3,(t044-t03))        V0        V0

z0                z1                          z2                          z3                          V0   V0

];


Jm5=[

cross(z0,t055)    cross(z1,(t055-t01))        cross(z2,(t055-t02))        cross(z3,(t055-t03))
        cross(z4,(t055-t04))        V0

z0                z1                          z2                          z3                          z4                    V0

];


Jm6=[

cross(z0,t066)    cross(z1,(t066-t01))        cross(z2,(t066-t02))        cross(z3,(t066-t03))
        cross(z4,(t066-t04))        cross(z5,(t066-t05))

z0                z1                          z2                          z3                          z4                    z5

];


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% División de los anteriores en lineales y rotacionales

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


Jv1=Jm1(1:3,1:6);

Jw1=Jm1(4:6,1:6);

Jv2=Jm2(1:3,1:6);

Jw2=Jm2(4:6,1:6);

Jv3=Jm3(1:3,1:6);

Jw3=Jm3(4:6,1:6);

Jv4=Jm4(1:3,1:6);
```

```
Jw4=Jm4(4:6,1:6);

Jv5=Jm5(1:3,1:6);

Jw5=Jm5(4:6,1:6);

Jv6=Jm6(1:3,1:6);

Jw6=Jm6(4:6,1:6);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Calculo de M
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


M1=(m1*((transpose(Jv1))*Jv1))+((transpose(Jw1))*(R011)*(I1)*(transpose(R011))*(Jw1));

M2=(m2*((transpose(Jv2))*Jv2))+((transpose(Jw2))*(R022)*(I2)*(transpose(R022))*(Jw2));

M3=(m3*((transpose(Jv3))*Jv3))+((transpose(Jw3))*(R033)*(I3)*(transpose(R033))*(Jw3));

M4=(m4*((transpose(Jv4))*Jv4))+((transpose(Jw4))*(R044)*(I4)*(transpose(R044))*(Jw4));

M5=(m5*((transpose(Jv5))*Jv5))+((transpose(Jw5))*(R055)*(I5)*(transpose(R055))*(Jw5));

M6=(m6*((transpose(Jv6))*Jv6))+((transpose(Jw6))*(R066)*(I6)*(transpose(R066))*(Jw6));

M=M1+M2+M3+M4+M6+M6;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Calculo de C a partir de M por Christoffel
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Ka=0;

Jo=0;

un=0;

dos=0;

tres=0;

cuatro=0;
```

```
cinco=0;

seis=0;


Qvec=[q1,q2,q3,q4,q5,q6];


    for Ka = 1:6,

      for Jo = 1:6,

          un=(diff(M(Ka,Jo),q1))+(diff(M(Ka,1),Qvec(1,Jo)))-(diff(M(1,Jo),Qvec(1,Ka)));

          dos=(diff(M(Ka,Jo),q2))+(diff(M(Ka,2),Qvec(1,Jo)))-(diff(M(2,Jo),Qvec(1,Ka)));

          tres=(diff(M(Ka,Jo),q3))+(diff(M(Ka,3),Qvec(1,Jo)))-(diff(M(3,Jo),Qvec(1,Ka)));

          cuatro=(diff(M(Ka,Jo),q4))+(diff(M(Ka,4),Qvec(1,Jo)))-(diff(M(4,Jo),Qvec(1,Ka)));

          cinco=(diff(M(Ka,Jo),q5))+(diff(M(Ka,5),Qvec(1,Jo)))-(diff(M(5,Jo),Qvec(1,Ka)));

          seis=(diff(M(Ka,Jo),q6))+(diff(M(Ka,6),Qvec(1,Jo)))-(diff(M(6,Jo),Qvec(1,Ka)));

          C(Ka,Jo) =
(1/2)*(((un)*(q1p))+((dos)*(q2p))+((tres)*(q3p))+((cuatro)*(q4p))+((cinco)*(q5p))+((seis)*(q6p)));

      end

    end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% calculo de G

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


syms g

grav=[0;0;-g];

Poten=(m1*(transpose(grav))*t011)+(m2*(transpose(grav))*t022)+(m3*(transpose(grav))*t033)+(m4*(transpos
e(grav))*t044)+(m5*(transpose(grav))*t055)+(m6*(transpose(grav))*t066);

G=[diff(Poten,q1);diff(Poten,q2);diff(Poten,q3);diff(Poten,q4);diff(Poten,q5);diff(Poten,q6)];
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% %% Comprobación de C por propiedad de antisimetria
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% dH=(diff(M,q1)*q1p)+(diff(M,q2)*q2p)+(diff(M,q3)*q3p)+(diff(M,q4)*q4p)+(diff(M,q5)*q5p)+(diff(M,q6)*q6p);
% % % N(1,1)=dH(1,1)-(2*C(1,1));
% % % N(1,2)=dH(1,2)-(2*C(1,2));
% % % N(1,3)=dH(1,3)-(2*C(1,3));
% % % N(1,4)=dH(1,4)-(2*C(1,4));
% % % N(1,5)=dH(1,5)-(2*C(1,5));
% % % N(1,6)=dH(1,6)-(2*C(1,6));
% % % N(2,1)=dH(2,1)-(2*C(2,1));
% % % N(2,2)=dH(2,2)-(2*C(2,2));
% % % N(2,3)=dH(2,3)-(2*C(2,3));
% % % N(2,4)=dH(2,4)-(2*C(2,4));
% % % N(2,5)=dH(2,5)-(2*C(2,5));
% % % N(2,6)=dH(2,6)-(2*C(2,6));
% % % N(3,1)=dH(3,1)-(2*C(3,1));
% % % N(3,2)=dH(3,2)-(2*C(3,2));
% % % N(3,3)=dH(3,3)-(2*C(3,3));
% % % N(3,4)=dH(3,4)-(2*C(3,4));
% % % N(3,5)=dH(3,5)-(2*C(3,5));
% % % N(3,6)=dH(3,6)-(2*C(3,6));
% % % N(4,1)=dH(4,1)-(2*C(4,1));
% % % N(4,2)=dH(4,2)-(2*C(4,2));
% % % N(4,3)=dH(4,3)-(2*C(4,3));
```

```
% % % N(4,4)=dH(4,4)-(2*C(4,4));

% % % N(4,5)=dH(4,5)-(2*C(4,5));

% % % N(4,6)=dH(4,6)-(2*C(4,6));

% % % N(5,1)=dH(5,1)-(2*C(5,1));

% % % N(5,2)=dH(5,2)-(2*C(5,2));

% % % N(5,3)=dH(5,3)-(2*C(5,3));

% % % N(5,4)=dH(5,4)-(2*C(5,4));

% % % N(5,5)=dH(5,5)-(2*C(5,5));

% % % N(5,6)=dH(5,6)-(2*C(5,6));

% % % N(6,1)=dH(6,1)-(2*C(6,1));

% % % N(6,2)=dH(6,2)-(2*C(6,2));

% % % N(6,3)=dH(6,3)-(2*C(6,3));

% % % N(6,4)=dH(6,4)-(2*C(6,4));

% % % N(6,5)=dH(6,5)-(2*C(6,5));

% % % N(6,6)=dH(6,6)-(2*C(6,6));


% % % sumu1=((veitor(1,1)*N(1,1)) + (veitor(1,2)*N(2,1)) + (veitor(1,3)*N(3,1)) + (veitor(1,4)*N(4,1)) +
(veitor(1,5)*N(5,1)) + (veitor(1,6)*N(6,1)))*veitor(1,1);

% % % sumu2=((veitor(1,1)*N(1,2)) + (veitor(1,2)*N(2,2)) + (veitor(1,3)*N(3,2)) + (veitor(1,4)*N(4,2)) +
(veitor(1,5)*N(5,2)) + (veitor(1,6)*N(6,2)))*veitor(1,2);

% % % sumu3=((veitor(1,1)*N(1,3)) + (veitor(1,2)*N(2,3)) + (veitor(1,3)*N(3,3)) + (veitor(1,4)*N(4,3)) +
(veitor(1,5)*N(5,3)) + (veitor(1,6)*N(6,3)))*veitor(1,3);

% % % sumu4=((veitor(1,1)*N(1,4)) + (veitor(1,2)*N(2,4)) + (veitor(1,3)*N(3,4)) + (veitor(1,4)*N(4,4)) +
(veitor(1,5)*N(5,4)) + (veitor(1,6)*N(6,4)))*veitor(1,4);

% % % sumu5=((veitor(1,1)*N(1,5)) + (veitor(1,2)*N(2,5)) + (veitor(1,3)*N(3,5)) + (veitor(1,4)*N(4,5)) +
(veitor(1,5)*N(5,5)) + (veitor(1,6)*N(6,5)))*veitor(1,5);

% % % sumu6=((veitor(1,1)*N(1,6)) + (veitor(1,2)*N(2,6)) + (veitor(1,3)*N(3,6)) + (veitor(1,4)*N(4,6)) +
(veitor(1,5)*N(5,6)) + (veitor(1,6)*N(6,6)))*veitor(1,6);

% % % sumotototа=sumu1+sumu2+sumu3+sumu4+sumu5+sumu6;
```

Código Para la obtención del modelo grafico del robot usando el Toolbox de Peter Corke (ejemplo automatizado por el Dr Emmanuel Dean)

```
%Este archivo fue generado automÃ¡ticamente por la funciÃ³n--Generador_Robot--
clear L
clear Daux
clear Dr
clear sigma
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Tabla DH requerida por el bloque de Simulink
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
sigma(1,1)=0;
sigma(2,1)=0;
sigma(3,1)=0;
Daux(1,1)=1.570796e+00;%alfa
Daux(1,2)=0;%a
Daux(1,4)=5.000000e-01;%D
Daux(1,3)=0;%theta
Daux(2,1)=0;%alfa
Daux(2,2)=4.000000e-01;%a
Daux(2,4)=0;%D
Daux(2,3)=0;%theta
Daux(3,1)=0;%alfa
Daux(3,2)=2.000000e-01;%a
Daux(3,4)=0;%D
Daux(3,3)=0;%theta
fprintf(1,'  alfa     a      theta     d     sigma\n');
Dr=[Daux,sigma]
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% definición de cada eslabón a partir de la matriz anterior
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

L{1}=(link(Dr(1,:),'standard'));
L{2}=(link(Dr(2,:),'standard'));
L{3}=(link(Dr(3,:),'standard'));
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% coordenadas marco base
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
base=[0 1 0 0;-1 0 0 0;0 0 1 0;0 0 0 1];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% definición del robot, esto genera los datos que requiere el bloque de simulink  roblocks/plot
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear Robot_RRRF
Robot_RRRF=robot(L,'Robot-RRRP','Fabricante_RRR');
Robot_RRRF.base=base;
Robot_RRRF.plotopt={'nobase'};
% Robot_RRRF.plotopt={'nobase' 'workspace' [-1.5 1.5 -1.5 1.5 -0.1 1.5] 'mag' 13};
clear L
clear Dr
clear Daux
clear sigma
```

Código del filtro de Kalman basado en las referencias [40] y [41]:

```
function y = kalman01c(z)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%  un dt grande equivale a proporcionar mas terminos derivativos
%%  Matriz A de la partícula a rastrear y predecir
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
dt=200;

A=[ 1 0 0 dt 0 0;
   0 1 0 0 dt 0;
   0 0 1 0 0 dt;
   0 0 0 1 0 0;
   0 0 0 0 1 0 ;
   0 0 0 0 0 1 ];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Matriz de medición x y z
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
H = [ 1 0 0 0 0 0 ; 0 1 0 0 0 0;0 0 1 0 0 0];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Matrices Q y R
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Q = eye(6);
R = 1000 * eye(3);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% condiciones iniciales
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
persistent x_est p_est
if isempty(x_est)
    x_est = zeros(6, 1);
    p_est = zeros(6, 6);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% covarianza y estados predecidos
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
x_prd = A * x_est;
p_prd = A * p_est * A' + Q;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% ganancia Kalman
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
S = H * p_prd' * H' + R;
B = H * p_prd';
klm_gain = (S \ B)';
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% estados corregidos con base a la adquisición de datos del sensor z
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
x_est = x_prd + klm_gain * (z - H * x_prd);
p_est = p_prd - klm_gain * H * p_prd;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% variables de salida predichas
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
y = H * x_est;
```

Diseño de los potenciales artificiales (aplicación a robots móviles)

```
function xpp=particula(u)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Entradas al sistema realimentación por integradores de posición y velocidades; tiempo
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
x=u(1);
y=u(2);
xp=u(3);
yp=u(4);
t=u(5);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Parámetros del sistema
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
m=0.1;
g=9.81;
G=[0;g];
C=[30,0;0,30];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Ganancias de atracción, repulsión y diferencial
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Kpa=[600,0;0,600];
Kpr=[20,0;0,20];
Kd=[20,0;0,20];
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% posicion particula
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
r=[x;y];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% posición deseada y posición obstáculos
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% if t<=10
    rd=[2.6*sin(pi*t/4);2.6*cos(pi*t/4)];
% else
%     rd=[2.6*sin(pi*t/4)+0.2;2.6*cos(pi*t/4)];
% end

   %rob=[2;1.6];
    rob=[2.6*sin(pi*t/4);-2.6*cos(pi*t/4)];
    rob2=[2.6*sin(pi*t/8);-2.6*cos(pi*t/8)];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% vector de velocidad del interceptor
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
rp=[xp;yp];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Kalman de la posición deseada móvil
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
rdk = kalman01se(rd);
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% grafica de las partículas y sus trayectorias
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
plot(rdk(1),rdk(2),'ko')
hold on
plot(rob(1),rob(2),'bo')
plot(rob2(1),rob2(2),'go')
plot(x,y,'ro')

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% errores de posición y distancias a obstáculos
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
e=rdk-r;
eo=r-rob;
eo2=r-rob2;
dis=dist(transpose(r),rob);
dis2=dist(transpose(r),rob2);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% campo potencial artificial de río
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%MagFrio=50;
%vecno=-e/norm(e);
%Frio=MagFrio*vecno;
%normeo=norm(eo);
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% fuerza de repulsión de Kathib
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
segu=10;
if (dis<=segu)
Frepu=(Kpr*((1/dis)-(1/segu)))*(1/(dis*dis))*(eo/dis);
else
Frepu=zeros(2,1);
end

if (dis2<=segu)
Frepu2=(Kpr*((1/dis2)-(1/segu)))*(1/(dis2*dis2))*(eo2/dis2);
else
Frepu2=zeros(2,1);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% repulsion total
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Frep=Frepu+Frepu2;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% fuerza rotonormal basada en la fuerza de kathib
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Frerepu(1,1)=(1/sqrt(2))*(Frep(1)-Frep(2));
Frerepu(2,1)=(1/sqrt(2))*(Frep(1)+Frep(2));
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% fuerza de atraccion
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Fatra=-1*(Kpa*(e))+Frio;
Fatra=-1*(Kpa*(e));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% ley de control
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Tao=(Fatra+Frerepu)-(Kd*rp)+G;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% salida
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
xpp=[1/m*(Tao-C*rp-G)];
```

Programa auxiliar para el cálculo del regresor lineal

Para entender este programa se propone el siguiente ejemplo ilustrativo, sea:

$$\tau = \begin{bmatrix} bgl_2m_1 - gl_1m_1(a+c) \\ egl_2m_1 - gl_3m_2(d+f) \end{bmatrix} \quad (99)$$

Donde . $m_1 \quad m_2 \quad l_1 \quad l_2 \quad l_3 \quad g$ son parámetros conocidos.

Se desea encontrar el llamado regresor lineal (de utilidad en ciertos controladores) que es una factorizacion de la siguiente forma

$$\tau_{(nx1)} = Y(q, \frac{dq}{dt}, \frac{d^2q}{dt^2})_{(nxr)}\theta_{(rx1)} \quad (100)$$

Donde $\theta_{(rx1)}$ es un vector de parámetros conocidos y $Y(q, \frac{dq}{dt}, \frac{d^2q}{dt^2})_{(nxr)}$ es una matriz llamada de estados

1. se expanden las ecuaciones de Torque (es recomendable usar el comando simplify de matlab)

$$\tau = \begin{bmatrix} bgl_2m_1 - gl_1m_1(a+c) \\ egl_2m_1 - gl_3m_2(d+f) \end{bmatrix} = \begin{bmatrix} bgl_2m_1 - agl_1m_1 - cgl_1m_1 \\ gl_2m_1e - dgl_3m_2 - fgl_3m_2 \end{bmatrix} \quad (101)$$

En las ecuaciones de robótica son comunes este tipo de términos
$\alpha \sin(q_1 + q_2 - q_3) + \beta \cos(q_3 + q_1 - q_2)\ldots.$

Para que ello no afecte los siguientes pasos de la metodología basada en operaciones de cadenas y caracteres se genera una función que reemplaza todo signo dentro de las funciones tal que la expresión anterior sea vista como:
$\alpha \sin(q_1 ?q_2 !q_3) + \beta \cos(q_3 ?q_1 !q_2)\ldots.$

2. Después de resolver el inconveniente anterior, la ecuación vectorial se trata como entidades individuales y sobre cada una se realizan los pasos subsecuentes.

$$\tau_1 = -agl_1m_1 - cgl_1m_1 + bgl_2m_1 \quad (102)$$

$$\tau_2 = gl_2m_1e - dgl_3m_2 - fgl_3m_2 \quad (103)$$

3. La ecuación debe iniciar sin signo, eso se logra intercambiando cualquier termino positivo.

$$\tau_1 = bgl_2m_1 - agl_1m_1 - cgl_1m_1 \quad (104)$$

4. Se sacan los signos y se reemplazan por espacios se almacenan en un vector de igual dimensión a la cantidad de signos mas uno.

bgl_2m_1	agl_1m_1	cgl_1m_1
Sig(1)	Sig(2)	Sig(3)
+	−	−

Tabla 2: Vector de signos

5. Cada espacio se interpreta como un nuevo elemento del vector de terminos que tiene la misma dimensión que el de signos

term(1)	term(2)	term(3)
bgl_2m_1	agl_1m_1	cgl_1m_1

Tabla 3: Vector de términos

6. Se genera una tabla de ocurrencias que relaciona cada termino con el correspondiente parámetro predefinido por el usuario

	term(1)	term(2)	term(3)
	bgl_2m_1	agl_1m_1	cgl_1m_1
m_1	1	1	1
l_2	1	0	0
g	1	1	1
l_1	0	1	1
l_3	0	0	0

Tabla 4: Tabla de ocurrencias términos vs parámetros simplificados

7 se concatena hacia abajo lo que se encuentre indicado a 1 y se almacena en una variable lo que arroja esta concatenación la cual es del mismo tamaño que los términos

term(1)	term(2)	term(3)
bgl_2m_1	agl_1m_1	cgl_1m_1
terparam(1)	terparam(2)	terparam(3)
m_1l_2g	m_1gl_1	m_1gl_1

Tabla 5: Concatenamiento descendente de Tabla 4

8 se hace una comparación de cadenas en terparam y se genera un nuevo termino en param cada vez que se encuentra una cadena distinta, el contenido será esa cadena asimismo al dar afirmativo dicho termino genera un termb el cual es term pero sin param, también se concatena termb(n) con sig(n)

	termb(1)	termb(2)	termb(3)
param(1)	$+b$		
m_1l_2g			
param(1)		$-a$	$-c$
m_1gl_1			

Tabla 6: simplificación parametrica de Tabla 5

9 se genera una tabla de ocurrencias de param respecto a term

	termb(1)	termb(2)	termb(3)
param(1) $m_1 l_2 g$	1	0	0
param(1) $m_1 g l_1$	0	1	1

Tabla 7: Tabla de ocurrencias terminos simplificados vs parámetros comunes

10 se genera un vector llamado estado de la misma dimensión en columnas que las filas de param y su resultado es la concatenación lateral de ocurrencias de la tabla anterior

estado(1)	estado(2)
b	$-a-c$

Tabla 8: Concatenación lateral de Tabla 7 y estados

Se comprueba

$$\tau_1 = \begin{bmatrix} b & -a-c \end{bmatrix} \begin{bmatrix} m_1 l_2 g \\ m_1 g l_1 \end{bmatrix} = b g l_2 m_1 - g l_1 m_1 (a+c) \quad (105)$$

11 como lo anterior se realiza ecuación a ecuación y el sistema tiene n ecuaciones es momento de juntar los datos

$$\tau_1 = \begin{bmatrix} b & -a-c \end{bmatrix} \begin{bmatrix} m_1 l_2 g \\ m_1 g l_1 \end{bmatrix} \quad (106)$$

$$\tau_2 = \begin{bmatrix} e & -d-f \end{bmatrix} \begin{bmatrix} m_1 l_2 g \\ m_2 g l_3 \end{bmatrix} \quad (107)$$

12 se concatenan los vectores de parámetros individuales y se genera un global con los elementos no repetidos

$$\begin{bmatrix} m_1 l_2 g \\ m_1 g l_1 \\ m_1 l_2 g \\ m_2 g l_3 \end{bmatrix} \rightarrow \begin{bmatrix} m_1 l_2 g \\ m_1 g l_1 \\ m_2 g l_3 \end{bmatrix}$$

Tabla 9: Simplificación a parámetros globales

13 se buscan todos los parámetros distintos y su ocurrencia en cada ecuación llenándose la siguiente tabla

$$\begin{bmatrix} & \tau_1 & \tau_2 \\ m_1 l_2 g & b & e \\ m_1 g l_1 & -a-c & 0 \\ m_2 g l_3 & 0 & -d-f \end{bmatrix}$$

Tabla 10: Obtención de la matriz global de estados

14 Finalmente se comprueba

$$\tau_{(nx1)} = \left(\begin{bmatrix} m_1 l_2 g & m_1 g l_1 & m_2 g l_3 \end{bmatrix} \begin{bmatrix} b & e \\ -a-c & 0 \\ 0 & -d-f \end{bmatrix} \right)^T = \begin{bmatrix} b g l_2 m_1 - g l_1 m_1 (a+c) \\ g l_2 m_1 e - g l_3 m_2 (d+f) \end{bmatrix} \quad (108)$$

$$Y(q, \frac{dq}{dt}, \frac{d^2 q}{dt^2})_{(nxr)} = \begin{bmatrix} b & e \\ -a-c & 0 \\ 0 & -d-f \end{bmatrix}^T \quad (109)$$

$$\theta_{(rx1)} = \begin{bmatrix} m_1 l_2 g & m_1 g l_1 & m_2 g l_3 \end{bmatrix}^T \quad (110)$$

Teniendo M, C y G de programas anteriores se encuentra el torque de regresor de la siguiente forma:

```
syms qpp1r qpp2r qpp3r qp1r qp2r qp3r;
Qppr=[qpp1r;qpp2r;qpp3r];
Qpr=[qp1r;qp2r;qp3r];
TAOr=(M*Qppr)+(C*Qpr)+G;
TAOr=simplify(TAOr);
```

Para no saturar la memoria, es conveniente guardar cada termino de TAOr como un txt habiendo n archivos uno por cada grado de libertad, posteriormente se modifican los terminos internos asociados a funciones seno y coseno que se describieron líneas arriba

```
clc
clear all;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Se abre el archivo que contiene al TAOr correspondiente, esto se hace para cada txt
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    fabre = fopen('RUTA Y NOMBRE DEL ARCHIVO', 'r');
    cadenaprin=fread(fabre, '*char');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% se almacena el contenido de TAOr en una variable
%% cadenaprin='cos(a+b+c-d+e-g)*sin(r-w+u)+r+sin(d-e)-cos(a+b)'
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    cadenaprin=transpose(cadenaprin)
```

```
        fclose(fabre);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% se establecen los identificadores de paréntesis y los de signo
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
ideop=findstr(cadenaprin,'(');
idecl=findstr(cadenaprin,')');
tamop=size(ideop);
tamop=tamop(2);
tamca=size(cadenaprin);
tamca=tamca(2);

idemas=findstr(cadenaprin,'+');
idemen=findstr(cadenaprin,'-');
tammas=size(idemas);
tammas=tammas(2);
tammen=size(idemen);
tammen=tammen(2);
principio=0;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%  se cicla el proceso para todos los términos
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

fin=0;
for ve=1:tamop
   principio=ideop(ve);
   fin=idecl(ve);
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% si existen simbolos + dentro de (), se reemplazan por  ?
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

   for cambiamas=1:tammas
      posidemas=idemas(cambiamas);
      if((idemas(cambiamas)>principio)&(idemas(cambiamas)<fin))
         cadenaprin(posidemas)='?';
      end
   end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% si existen símbolos - dentro de (), se reemplazan por  ¡
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

   for cambiamen=1:tammen
      posidemen=idemen(cambiamen);
      if((idemen(cambiamen)>principio)&(idemen(cambiamen)<fin))
         cadenaprin(posidemen)='¡';
      end
   end

end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% se escribe un archivo con el que se trabajara en el algoritmo del calculo de regresor
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
    cadeni=strcat('NUEVO NOMBRE DE ARCHIVO CORREGIDO'.txt');

    fid = fopen(cadeni, 'w');

    fwrite(fid, cadenaprin, 'char');

    fclose(fid);
```

Una vez contando con los nuevos txt, se implementa el programa del cálculo del regresor:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% programa que encuentra el regresor de parámetros

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear all;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% % se carga la matriz T

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    chp=int2str(p);

    chq=int2str(q);

    cadeno=strcat('NOMBRE CON RUTA','.txt');

    fabre = fopen(cadeno, 'r');

    leido=fread(fabre, '*char');

    leido=transpose(leido);

    fclose(fabre);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Se genera el vector vació que contendrá los signos
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
leecarac=leido;
refer=size(leecarac);
tamvacia=refer(1,2);
vacio=ones(1,tamvacia);
vacio=num2str(vacio);
vacio=strrep(vacio,'1  ','*');
vacio=strrep(vacio,'1','*');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% identificador de - y +
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

imen = findstr('-',leecarac);
imas = findstr('+',leecarac);
banderas=[imas,imen];
banderas=sort(banderas);
terminos=size(banderas);
terminos=(terminos(1,2))+1;

tanmas=size(imas);
tanmas=tanmas(1,2);
tanmen=size(imen);
tanmen=tanmen(1,2);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% rellenar con + en la posición correspondiente al termino donde se  localizo tal signo
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
for ma=1:tanmas;
   signoposicion=imas(ma);
   vacio(1,signoposicion)='+';
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% rellenar con - en la posición correspondiente al termino donde se localizo tal signo
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for me=1:tanmen;
   signoposicion=imen(me);
   vacio(1,signoposicion)='-';
end

tomasigno='+p'; %el primero es mas por default
for variadorsize=1:tamvacia;
   if(vacio(variadorsize)=='*')
      tomasigno=tomasigno;
   else
      tomasigno=strcat(tomasigno,vacio(variadorsize),'p');
   end
end
[elsi] =  strread(tomasigno,'%s','delimiter','p');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% definición de los parámetros madre, se recomienda el uso de d en vez de  l para evitar problemas
%%interpretativos (Bugg)
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%buscadato='I111,I112,I113,I122,I123,I133,I211,I212,I213,I222,I223,I233,I311,I312,I313,I322,I323,I333';
%buscadato='m2,m3,m4,m5,m6,I1,I2,I4,I5,I6,I7,g';
buscadato='m1,m2,m3,d1^2,d2^2,d3^2,d4^2,d5^2,d6^2,I1,I2,I3,I4,I5,I6,g';
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% se genera una tabla de ocurrencias que relaciona cada termino con el correspondiente  parámetro
%%predefinido por el usuario
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[vdedatos] = strread(buscadato,'%s','delimiter',',');
contenedordatos=size(vdedatos);
contenedordatos=contenedordatos(1,1);

for datou=1:contenedordatos;
   cvdedatos=char(vdedatos(datou,1));
   imda{1,datou}=findstr(cvdedatos,leecarac);
end

leecaraccon = strrep(leecarac,'-','+');

for datou=1:contenedordatos;
   cvdedatos=char(vdedatos(datou,1));
   leecarac=strrep(leecarac,cvdedatos,'');
end

leecaracsin = strrep(leecarac,'-','+');
```

```
[b] = strread(leecaracsin,'%s','delimiter','+');
[a] = strread(leecaraccon,'%s','delimiter','+');

b=strcat(elsi,b,'w');
b=strrep(b,'**','*');
b=strrep(b,'*w','');
b=strrep(b,'w','');
sirve=size(a);


for iterala=1:sirve(1,1);
   aa=char(a(iterala,1));

   for datou=1:contenedordatos;
      cvdedatos=char(vdedatos(datou,1));
      identi(datou,iterala)=isempty(findstr(cvdedatos,aa));
      if (identi(datou,iterala)==1)
         identii{datou,iterala}='*';
      else
         identii{datou,iterala}=cvdedatos;
      end
   end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Concatenamiento hacia abajo
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
for numelem=1:terminos
   concateidentificados(1,numelem)=identii(1,numelem);
   for datis=2:contenedordatos;
      concateidentificados(1,numelem)=strcat(concateidentificados(1,numelem),identii(datis,numelem));
   end
end

for numelem=1:terminos
concateidentificados(1,numelem)=strrep(concateidentificados(1,numelem),'*',');
end

concateidentificados=transpose(concateidentificados);

b=strcat(b,'|',concateidentificados,'t|');
combina=identi;
combina=transpose(combina);

    for numelem=1:terminos
       for numparam=1:contenedordatos
          if (combina(numelem,numparam)==0)
             combi(numelem,numparam)=1;
          else
             combi(numelem,numparam)=0;
          end
       end
    end
```

```
%
     matrizcombinacion=mat2str(combi);
     matrizcombinacion=strrep(matrizcombinacion,'[','');
     matrizcombinacion=strrep(matrizcombinacion,']','');




    rem=matrizcombinacion;


    aumenta=0;
    while(isempty(rem)== 0)
        aumenta=aumenta+1;
        [token,rem] = strtok(rem,';');
        tokensigno=bin2dec(token);
        nombreparam(1,aumenta)=tokensigno;

    end

    nombreparam=transpose(nombreparam);
    listaparam=sort(nombreparam);
    %%caution
    contador2=1;
    vectordeparam(1)=min(nombreparam);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% concatenamiento lateral
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
for numelem=2:terminos

        if (listaparam(numelem,1)==listaparam(numelem-1,1))
            contador2=contador2;
        else
            contador2=contador2+1;
            vectordeparam(contador2)=listaparam(numelem,1);
        end

end

tamparametrico=size(vectordeparam);
dimenparam=tamparametrico(1,2);
vectorestado=cell(1,dimenparam);

for auxiparam=1:dimenparam
   for auxiparam2=1:terminos
      if(nombreparam(auxiparam2,1)==vectordeparam(1,auxiparam))
         vectorestado(1,auxiparam)=strcat(vectorestado(1,auxiparam),'_',b(auxiparam2,1));
      else
         vectorestado(1,auxiparam)=vectorestado(1,auxiparam);
      end
   end
end
```

%%%

%% Generación de tantos txts como parámetros se hallan encontrado, todos contienen información %%concerniente a sus estados paso 10 del procedimiento descrito

%%%

```
numdetxt=size(vectorestado);

numdetxt=numdetxt(1,2);

for selectortxt=1:numdetxt

        nombredeltxt=char(vectorestado(1,selectortxt));

        [token2,rem2] = strtok(char(vectorestado(1,selectortxt)),'|');

        %token2

        [token2,rem2] = strtok(rem2,'|');

        %token2

        cadenaprin=strcat(token2,'taur31','.txt');

        nombredeltxt=strrep(nombredeltxt,'_','');

        nombredeltxt=strrep(nombredeltxt,token2,'');

        nombredeltxt=strrep(nombredeltxt,'||','');

        nombredeltxt=strrep(nombredeltxt,'**','*');

        fides = fopen(cadenaprin, 'w');

        fwrite(fides, nombredeltxt, 'char');

        fclose(fides);


end
```

A continuación se incluyen los códigos de procesamiento de imágenes elaborados en ANSI C y OpenCV, los de control del manipulador son modificaciones de [61] y no se incluyen por motivos de derechos de terceros.

Programa auxiliar para operaciones de matrices:

```
#include <stdio.h>
#include "cv.h"
#include "highgui.h"

////////////////////////////////////////////////////////////////////
////////// funcion para imprimir matrices tipo OPENCV ////////////
////////////////////////////////////////////////////////////////////

void print_matrix(double a[][3]);

inline void cvDoubleMatPrint( const CvMat* mat )
{
    int i, j;
    for( i = 0; i < mat->rows; i++ )
    {
        for( j = 0; j < mat->cols; j++ )
        {
            printf( "%f ",cvmGet( mat, i, j ) );
        }
        printf( "\n" );

    }
}


int main()
{

////////////////////////////////////////////////////////////////////////
///////////////// definicion de matrices o vectores ////////////////////
////////////////////////////////////////////////////////////////////////

  double dA[] = {
    1.0, 2.0, 5.0,
    1.7, 0.8, 0.3,
    0.6, 1.0, 4.0,
  };
  CvMat A = cvMat(3, 3, CV_64FC1, dA );

  double dB[] = {
    1.0, 0.0, 0.0,
    0.0, 1.0, 0.0,
    0.0, 0.0, 1.0,
  };

  CvMat B = cvMat( 3, 3, CV_64FC1, dB );
```

```
CvMat *C = cvCreateMat( 3, 3, CV_64FC1);


cvMatMul( &A, &B, C );

cvDoubleMatPrint(C);

printf("multiplicacion cuadrada\n");


double dAA[] = {
  1.0, 2.0, 3.0,
  3.0, 5.0, 1.0,
};
CvMat AA = cvMat(2, 3, CV_64FC1, dAA );

double dBB[] = {
  1.0,
  1.0,
  1.0,
};
CvMat BB = cvMat( 3, 1, CV_64FC1, dBB );

CvMat *CC = cvCreateMat( 2, 1, CV_64FC1);

////////////////////////////////////////////////////////////////////
////// multiplicacion, suma, inversa, transpuesta y determinante ////
////////////////////////////////////////////////////////////////////

cvMatMul( &AA, &BB, CC );

cvDoubleMatPrint(CC);

printf("multiplicacion no cuadrada\n");


CvMat *D = cvCreateMat( 3, 3, CV_64FC1);
cvAdd( &A, &B, D, 0 );
cvDoubleMatPrint(D);
printf("suma\n");

CvMat *E = cvCreateMat( 3, 3, CV_64FC1);
cvInv(&B,E,CV_LU);
cvDoubleMatPrint(E);
printf("inversa\n");


CvMat *F = cvCreateMat( 3, 2, CV_64FC1);
cvT(&AA, F);
cvDoubleMatPrint(F);
printf("transpuesta\n");

double determi=0;
float determi2=0;

determi=cvDet(&B);
```

```
  determi2=(float)determi;
  printf("%f\n",determi2);

return (0);
}
```

Programa auxiliar para guardado de imágenes con 2 cámaras simultaneas:

```
#include <stdarg.h>

#include "cv.h"
#include "highgui.h"
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>



int main()

{

/////////////////////////////////////////////////////////////////
//////////////////// Definicion de las camaras //////////////////
/////////////////////////////////////////////////////////////////

    CvCapture* capture = 0;
    IplImage* frame = 0;
    capture = cvCaptureFromCAM(0);

    CvCapture *capture2 = 0;
    IplImage *frame2 = 0;
    capture2= cvCaptureFromCAM(1);

/////////////////////////////////////////////////////////////////
////////////////// ajuste de tamaño de imagen ///////////////////
/////////////////////////////////////////////////////////////////

   cvSetCaptureProperty( capture, CV_CAP_PROP_FRAME_WIDTH, 640/4 );
   cvSetCaptureProperty( capture, CV_CAP_PROP_FRAME_HEIGHT, 480/4 );


   cvSetCaptureProperty( capture2, CV_CAP_PROP_FRAME_WIDTH, 640/4 );
   cvSetCaptureProperty( capture2, CV_CAP_PROP_FRAME_HEIGHT, 480/4 );

    cvNamedWindow("video",CV_WINDOW_AUTOSIZE);
    cvNamedWindow("video2",CV_WINDOW_AUTOSIZE);

/////////////////////////////////////////////////////////////////
/////// rutina para la generacion automatica del nombre /////////
/////////////////////////////////////////////////////////////////

    int nuima=0;
    const char base[] = "left";
    const char base2[] = "right";
```

```
    const char extencion[] = ".ppm";
    char figu [ FILENAME_MAX ];
    char figu2 [ FILENAME_MAX ];

////////////////////////////////////////////////////////////////////////
////////////////// captura y guardado de imagenes //////////////////////
////////////////////////////////////////////////////////////////////////

    while(1)
    {

        frame = cvQueryFrame(capture);
        frame2 = cvQueryFrame(capture2);



     cvShowImage("video", frame);
     cvShowImage("video2", frame2);
     nuima=nuima+1;
     sprintf(figu, "%s%d", base, nuima);
     sprintf(figu, "%s%s", figu,extencion);
     sprintf(figu2, "%s%d", base2, nuima);
     sprintf(figu2, "%s%s", figu2,extencion);


     cvSaveImage(figu, frame, 0);
     cvSaveImage(figu2, frame2, 0);
     int c = cvWaitKey(1000);
        if(c!=-1)
        {
            break;
        }


     }


    cvReleaseCapture(&capture);
    return 0;
}
```

Programa para el monitoreo del centroide de circunferencias a alta velocidad por color y forma en 2 camaras:

```
#include "cv.h"
#include "highgui.h"
#include "cxcore.h"



#include <stdio.h>
#include <math.h>
```

```
int main()

{

////////////////////////////////////////////////////////////////////////
////////////// declaracion de las camaras y resolucion /////////////////
////////////////////////////////////////////////////////////////////////

    CvSize size = cvSize(640/4,480/4);
    CvCapture* capture = cvCaptureFromCAM( 0 );
    CvCapture* capture2 = cvCaptureFromCAM( 1 );

    cvSetCaptureProperty( capture, CV_CAP_PROP_FRAME_WIDTH, 640/4 );
    cvSetCaptureProperty( capture, CV_CAP_PROP_FRAME_HEIGHT, 480/4 );


    cvSetCaptureProperty( capture2, CV_CAP_PROP_FRAME_WIDTH, 640/4 );
    cvSetCaptureProperty( capture2, CV_CAP_PROP_FRAME_HEIGHT, 480/4 );

    cvNamedWindow( "Camara", CV_WINDOW_AUTOSIZE );
    cvNamedWindow( "tresholda", CV_WINDOW_AUTOSIZE );

    cvNamedWindow( "Camara2", CV_WINDOW_AUTOSIZE );
    cvNamedWindow( "tresholda2", CV_WINDOW_AUTOSIZE );


////////////////////////////////////////////////////////////////////////
///////////////// selección de color para umbralizacion ////////////////
////////////////////////////////////////////////////////////////////////

    CvScalar hsv_min = cvScalar(150, 100, 100, 0);
    CvScalar hsv_max = cvScalar(200, 255, 255, 0);

    IplImage*  hsv_frame    = cvCreateImage(size, IPL_DEPTH_8U, 3);
    IplImage*  threshold   = cvCreateImage(size, IPL_DEPTH_8U, 1);

    IplImage*  hsv_frame2    = cvCreateImage(size, IPL_DEPTH_8U, 3);
    IplImage*  threshold2   = cvCreateImage(size, IPL_DEPTH_8U, 1);

        IplImage* frame = 0;
        IplImage* frame2 = 0;

    while( 1 )
    {

////////////////////////////////////////////////////////////////////////
///////////////// captura y umbralizacion //////////////////////////////
////////////////////////////////////////////////////////////////////////

      frame= cvQueryFrame( capture );
      frame2= cvQueryFrame( capture2 );


        cvCvtColor(frame, hsv_frame, CV_BGR2HSV);
        cvInRangeS(hsv_frame, hsv_min, hsv_max, threshold);
```

```
        cvCvtColor(frame2, hsv_frame2, CV_BGR2HSV);
        cvInRangeS(hsv_frame2, hsv_min, hsv_max, threshold2);



        CvMemStorage* storage = cvCreateMemStorage(0);
        CvMemStorage* storage2 = cvCreateMemStorage(0);

////////////////////////////////////////////////////////////////////////
///  filtrado de las imagenes, erosion, dilatacion, apertura o cierre ///
///  segun convenga                                                   ///
////////////////////////////////////////////////////////////////////////

        cvSmooth( threshold, threshold, CV_GAUSSIAN, 9, 9, 0, 0 );
        cvSmooth( threshold2, threshold2, CV_GAUSSIAN, 9, 9, 0, 0 );


////////////////////////////////////////////////////////////////////////
////////////// detector de circulos de hough ///////////////////////////
////////////////////////////////////////////////////////////////////////

        CvSeq* circles = cvHoughCircles(threshold, storage,
CV_HOUGH_GRADIENT, 2, threshold->height/4, 100, 50, 10, 400);
        CvSeq* circles2 = cvHoughCircles(threshold2, storage2,
CV_HOUGH_GRADIENT, 2, threshold2->height/4, 100, 50, 10, 400);

        int quo=1;
        int quo2=1;
        quo=circles->total;
        quo2=circles2->total;
      int i=0;
        int i2=0;
        float* p=0;
        float* p2=0;

////////////////////////////////////////////////////////////////////////
///////////////// ubicacion de los puntos centrales ////////////////////
////////////////////////////////////////////////////////////////////////

        for (i = 0; i < quo ; i++)
        {
            float* p = (float*)cvGetSeqElem( circles, i );

            printf("onde toy iz x=%f y=%f r=%f\n\r",p[0],p[1],p[2]);

cvCircle( frame, cvPoint(cvRound(p[0]),cvRound(p[1])),3, CV_RGB(0,255,0),
-1, 8, 0 );
cvCircle( frame, cvPoint(cvRound(p[0]),cvRound(p[1])),cvRound(p[2]),
CV_RGB(255,0,0), 3, 8, 0 );


        }

        for (i2 = 0; i2 < quo2; i2++)
        {
```

```
            float* p2 = (float*)cvGetSeqElem( circles2, i2 );
            printf("onde toy der x=%f y=%f r=%f \n\r",p2[0],p2[1],p2[2]);
cvCircle( frame2, cvPoint(cvRound(p2[0]),cvRound(p2[1])),3,
CV_RGB(0,255,0), -1, 8, 0 );
cvCircle( frame2, cvPoint(cvRound(p2[0]),cvRound(p2[1])),cvRound(p2[2]),
CV_RGB(255,0,0), 3, 8, 0 );
        }


////////////////////////////////////////////////////////////////////////
////////////////////// despliegue en pantalla ///////////////////////////
////////////////////////////////////////////////////////////////////////

        cvShowImage( "Camara", frame );
        cvShowImage( "tresholda", threshold );
        cvShowImage( "Camara2", frame2 );
        cvShowImage( "tresholda2", threshold2 );

        cvReleaseMemStorage(&storage);
        cvReleaseMemStorage(&storage2);


        if( (cvWaitKey(10) & 255) == 27 ) break;

    }


     cvReleaseCapture( &capture );
     cvDestroyWindow( "mywindow" );
     cvReleaseCapture( &capture2 );
     cvDestroyWindow( "mywindow2" );
     return 0;
   }
```

Bibliografía

[1] Mora, M. C, Tornero, J. Planificación de movimientos mediante la propagación de campos potenciales artificiales, 8º congreso iberoamericano de ingeniería mecánica

[2] P.I. Corke, "A Robotics Toolbox for MATLAB", IEEE Robotics and Automation Magazine, Volume 3(1), March 1996, pp. 24-32.

http://petercorke.com/Robotics_Toolbox.html

[3] G.L. Mariottini and D. Prattichizzo, "EGT: a Toolbox for Multiple View Geometry and Visual Servoing", IEEE Robotics and Automation Magazine, Volume (3), December 2005.

http://egt.dii.unisi.it/

[4] Gary Bradski, Adrian Kaehler, Learning OpenCV: Computer Vision with the OpenCV Library, 2008, O'Reilly Media, ISBN-10: 0596516134.

http://opencv.willowgarage.com/wiki/

[5] Giorgio Panin ,Model-based Visual Tracking: The OpenTL Framework, 2011, Wiley, ISBN-10: 0470876131

http://www.opentl.org/

[6] Gang Xu, Zhengyou Zhang, Epipolar Geometry in Stereo Motion and Object Recognition, Springer, 1996, ISBN-10: 9048147433

[7] Fernando Torres, Jorge Pomares, Robots y Sistemas Sensoriales, Prentice Hall, 2002, ISBN-10: 8420535745

[8] Emanuele Trucco, Alessandro Verri ,Introductory Techniques for 3-D Computer Vision, Prentice Hall, 1998, ISBN-10: 0132611082

[9] T.Sahin, E. Zergeroglu, “Adaptative Visual Servo Control of Robot Manipulator via Composite Camera Inputs”, Robot Motion and Control Recent Developments, Springer, 2004

[10] J. Alejandro Flores Campos, L. Héctor Hernández Gómez, Metodología de Intercepción Multiple de Trayectorias Bidimensionales Aplicada a un Sistema de Manufactura Flexible, Cientifica ESIME-IPN vol 9 num 002, pp 87-98.

[11] Enrique Pérez Fernández, Nuevo Método de Detección y Análisis en Tiempo Real de Eventos en la Tensión de Suministro de Energía Eléctrica Empleando un Modelo Combinado Wavelets-Filtro de Kalman Extendido, Universidad de Cantabria Departamento de Electrónica y Computadores, Marzo 2006

[12] Eli Brookner, Tracking and Kalman Filtering Made Easy, Wiley, 1998, ISBN-10: 9780471184072

[13] Hebertt Sira-Ramirez, Richard Marquez, Control de Sistemas No Lineales, Prentice Hall, 2005, ISBN-10: 8420544493

[14] Katsuhiko Ogata, Modern Control Engineering, Prentice Hall, 1996, ISBN-10: 0132273071

[15] Mohinder S. Grewal, Kalman Filtering: Theory and Practice Using MATLAB, 2008, ISBN-10: 9780470173664

[16] Gabriel Sepulveda Cervantes, “Estación Háptica para Deformación, Corte y Sutura de Organos Deformables con Propiedades Superficiales ”, CINVESTAV, Marzo 2009

[17] Mark W. Spong, Robot Modeling and Control, Wiley, 1989, ISBN-10: 047161243X

[18] Bruno Siciliano, Oussama Kathib, Handbook of Robotics, Springer, 2008, ISBN-10: 9783540239574

[19] John J. Craig, Introduction to Robotics: Mechanics and Control, Prentice Hall, 2004, ISBN-10: 9780201543612

[20] Reza N. Jazar, Theory of Applied Robotics, Springer, 2010, ISBN-10: 1441917497

[21] Yoshiyuki Tanaka, Toshio Tsuji, “Dynamic control of redundant manipulators using the artificial potential field aproach with time scaling ”, Artif Life Robotics, 1999, pp 79-85.

[22] Paraskevas Dunias, Autonomous Robots Using Artificial Potential Fields, 1996, Techsnique Universiteit Eindhoven, ISBN 90-386-0200-6

[23] R.Kelly, V. Santibáñez, Control of Robot Manipulators in Joint Space, Springer, ISBN-10: 1852339942

[24] Diego Aracena Pizarro, Pedro Campos, Comparación de Tecnicas de Calibración de Camaras Digitales, Universidad de Tarapaca, Revista Facultad de Ingenieria vol 13, numero 001 pp 55-67

[25] Marco Antonio Moreno Armendáriz, “Visión artificial estereo con aplicación al control de un brazo de robot”, CINVESTAV

[26] Emmanuel Carlos Dean Leon, “Seguimiento Dinámico de Robots Manipuladores Utilizando Retroalimentación Visual No Calibrada”, CINVESTAV

[27] Jiménez Camacho, E, Medición de distancias por medio de procesamiento de imágenes y triangulación, haciendo uso de cámaras de video, Universidad de las Americas Puebla

[28] Dávila Hernández Gustavo, Esquivel Ruiz Julio, Fierros Álvarez Javier, “Implementación de un sistema mecatronico para el robot Mitsubishi RV-M1 para realizar tareas de intercepción múltiple de objetos con trayectorias planares” UPIITA

[29] Larson, Hostetler Edwards, Calculo Vol 2, McGraw Hill, 2000, ISBN-10: 9701027566

[30] Purcell, Varberg, Rigdon, Calculo, Prentice Hall, 2001, ISBN-10: 9702601320

[31] Antonio Nieves, Federico C. Domínguez, Métodos Numéricos, CECSA, 2002, ISBN-10: 9702402581

[32] Dennis G. Zill, Differential Equations with Boundary-Value Problems, Thomson International, 2002, ISBN-10: 9706861211

[33] Prakhar Agarwal, David Federman et al, Target Tracking: Implementing the Kalman Filter

[34] Pomares, R. Gutiérrez, Seguimiento de trayectorias 3D mediante control visual basado en imagen, Universidad de Alicante

[35] www.wikipedia.org

[36] Rafael Kelly, Eloisa Garcia, On transpose Jacobian-Based Regulators Using Unit Quaternions: an energy shaping approach, CICESE

[37] Domingo Mery, Visión por computador, Universidad Católica de Chile

[38] Carlos Soria, Flavio Roberti, Control Servovisual de un Robot Manipulador tipo SCARA basado en Pasividad, Instituto de Automática Universidad Nacional de San Juan

[39] Jean-Claude Latombe, Robot Motion Planning, 1990, Springer, ISBN-10: 9780792391296

[40]http://www.mathworks.com/matlabcentral/fileexchange/14243-2d-target-tracking-using-kalman-filter

[41]http://www.mathworks.com/matlabcentral/fileexchange/26862-kalman-filtering-demo-in-matlab-with-automatic-matlab-to-c-code-generation

[42] Roland Siegwart, Illah Nourbakhsh, Inroduction to Autonomous Mobile Robots, The MIT press, 2004 ISBN-10: 026219502X

[43] Takegaki, M. and Arimoto, S, "A new feedback for dynamic control of manipulators", 1981, Transactions of the ASME; Journal of Dynamic Systems, Measurement and Control, 102, pp.119–125.

[44] Luis Baumela, José Miguel Buenaposada, Visión por computador, Universidad Politécnica de Madrid

[45] Elon Rimon, Sthepen Boyd, Obstacle Collision Detection Using Best Ellipsoid Fit, Stanford University, 1996

[46] Wang, Kim , An algebraic condition for the separation of two ellipsoids, University of Hong Kong, 2001

[47] Blas M. Vinagre, Introducción al control fraccionario, Universidad de Extremadura España, 2006

[48] Jing Ren, Kenneth A. McIsaac, A Hybrid-Systems Approach to Potential Field Navigation for a Multi-Robot Team, University of Western Ontario, 2003

[49] Rafael Kelly, Ilse Cervantes, On transpose Jacobian Control for monocular fixed-camera 3D Direct Visual Servoing, CICESE

[50] http://wiki.linuxcnc.org/emcinfo.pl?Debian_Lenny_Compile_RTAI

[51] Robert Laganiere, OpenCV 2 Computer Vision Application Programming Cookbook, 2011, ISBN-10: 1849513244

[52] http://www.aishack.in/2010/07/tracking-colored-objects-in-opencv/

[53] http://www.lirtex.com/robotics/fast-object-tracking-robot-computer-vision/

[54] http://blog.martinperis.com/2011/01/opencv-stereo-camera-calibration.html

[55] Gabe Sibley, Stereo Observation Models, University of Southern California, 2003

[56] Nicholas John Hollinghurst, Uncalibrated Stereo and Hand-Eye Coordination, Trinity Hall, 1997

[57] Y. Yakimovsky and R. Cunningham, A system for extracting three-dimensional measurements from a stereo pair of tv cameras. In Computer Graphics and Image Processing, volume 7, pages 1995–2010, 1978

[58] Fernando Reyes Cortez, Robotica, Control de Robots Manipuladores, Alfaomega, 2011, ISBN: 9786077071907

[59] Fernando Reyes Cortez, Teorema para el diseño de reguladores auto-sintonizables para robots manipuladores, BUAP, 2002

[60] Ryuichi Hara, Atsuo Nagayama, Time Constant Setting Method for a Track Program of a Robot, FANUC USA Patent 5751130, 1998

[61] Altamirano Gómez Gerardo Esteban, Plataforma de arquitectura abierta basada en el robot Mitsubishi RV-M1para realizar tareas en tiempo real, UPIITA, 2008