UNIVERSIDAD AUTÓNOMA CHAPINGO

Posgrado de Ingeniería Agrícola y Uso Integral del Agua

Diseño y construcción de un sistema de corte de hojas para la propagación in vitro◇

TESIS

que como requisito parcial para obtener el grado de:

Doctor en Ingeniería

presenta

M.C José Guillermo Cebada Reyes

UNIVERSIDAD AUTONOMA CHAPINGO MEXICO
DIRECCION GENERAL ACADEMICA
DEPTO. DE SERVICIOS ESCOLARES
OFICINA DE EXAMENES PROFESIONALES

8 de abril de 2016

Chapingo, Estado de México

◇ Tesis apoyada por CONACyT y la DGIP

POSGRADO IALIIA

Diseño y construcción de un sistema automatizado para corte de tejido vegetal

Tesis realizada por José Guillermo Cebada Reyes bajo la dirección del comité asesor indicado, aprobada por el mismo y aceptada como requisito parcial para obtener el grado de *Doctor en Ingeniería Agrícola y Uso Integral del Agua opción en Biosistemas*

Director: Dr. Federico Felix Hahn Schlam

Asesor: Dr. Eugenio Romantchik Kriuchkova,

Asesor: Dr. Agustín Ruiz García

Asesor: Dr. José Luis Rodríguez de la O

Lector Externo: Dr. Antonio Michua Camarillo

AGRADECIMIENTOS

A la Universidad Autónoma Chapingo (UACh), a la Dirección General de Investigación y al Posgrado en Ingeniería Agrícola y Uso Integral del Agua (IAUIA), así como al Consejo Nacional de Ciencia y Tecnología (CONACYT) por el apoyo económico y la formación que me brindaron para realizar mis estudios de doctorado.

Al Dr. Federico Hahn Sachlam por brindarme un reto hermoso para formación como doctor y brindarme sus sabios consejos durante mis estudios de posgrado y a lo largo de las diferentes fases de esta investigación.

Al Dr. Eugenio Romantchik, por abrirme las puertas de este posgrado y ser un gran apoyo para mi persona, en las etapas criticas de esta tesis y hacer que diera lo mejor de mi.

Al Dr. Irineo L. López Cruz por su guía, enseñanzas y paciencia durante mis estudios de posgrado y a lo largo del desarrollo de esta investigación.

A los Dr. José Luis Rodríguez de la O, Dra Petra Wierderhold, Dr. Antonio Michua y M.C. Pablo Sánchez por abrirme las puertas de sus laboratorios y centros de investigación para el desarrollo de esta tesis.

A los Dr. Mauricio Carrillo García y Dr. Agustin Ruíz García por el apoyo brindado durante mis estudios de doctorado así como su invaluable amistad.

A Mayra Luna, Rosa María Mendoza, y mis amigos de posgrado por su apoyo durante mi estancia en esta Universidad.

M.C. José Guillermo Cebada Reyes

DATOS BIOGRÁFICOS

M.C. José Guillermo Cebada Reyes nació en la ciudad de Puebla, Pue. México, el 17 de octubre de 1983. En 2009, bajo la dirección del M.C Pablo Sánchez Sánchez y del Dr. Antonio Michua Camarillo obtuvo su licenciatura en Ciencias de la Electrónica en la Facultad de Ciencias de la Electrónica de la Benemérita Universidad Autónoma de Puebla (BUAP), con la tesis Diseño y desarrollo de una plataforma gráfica de simulación para un robot de dos grados de libertad. Bajo la asesoría del Dr. Fernando Reyes Córtes y del M.C Pablo Sánchez Sánchez obtuve el grado de maestría en ciencias en Electrónica con especialidad en Automatización en la BUAP con la tesis Control Cartesiano: Teoría y Práctica. Ha participado en 5 congresos nacionales y 2 internacionales. Actualmente, es estudiante de doctorado en el posgrado de Ingeniería Agrícola y Uso Integral del Agua de la Universidad Autónoma Chapingo. Bajo la dirección del Dr. Federico Hahn Schlam y el Dr. Eugenio Romantchik Kiurchkova, en este trabajo de investigación se desarrollo, una metodología para la implementación de un robot de arquitectura abierta dedicada al corte de hojas para la micro-propagación vegetal.

RESUMEN

Diseño y construcción de un sistema de corte de hojas para la propagación in vitro

José Guillermo Cebada Reyes[a]
Federico Hahn Schlam[b]

En la propagación a través de hojas se considera la porción apical de la planta, es decir, la porción que contiene las hojas jóvenes de un tamaño promedio de 5 a 6 centímetros de longitud, para posteriormente tomar de 5 a 7 trozos de $2cm^2$ cada uno según la especie. Por lo tanto el objetivo principal de esta tesis es desarrollar un robot cartesiano de tres grados de libertad, dedicada a la disección de hojas, cuya electrónica fue desarrollada usando tecnología Arduino. La interfaz gráfica de usuario fue desarrollado con el software GUIDE-MATLAB creando ventanas interactivas que permite al usuario manipular e interpretar con facilidad el comportamiento del robot. Por otra parte se evaluó dos herramientas de corte, un cortador vertical y un sacabocados, encontrando que la herramienta tipo sacabocados genera mejores cortes sin presentar desagarres, por el contrario el cortador vertical presenta desgarros de $1mm$. Sin embargo en este proceso se encontraron problemas de sintonización en la estructura de control PD+, los cuales se solucionaron empleando algoritmos evolutivos y bio-inspirados. Encontrando que las ganancias ajustadas con estos algoritmos, mejoran el comportamiento del sistema. Los algoritmos Cuckoo, JDE y DE son buenos métodos para el sintonizado de ganancias. Este ajuste fue implementado en el controlador para el posicionamiento, visión y seguimiento de trayectoria. Como sistema de identificación de hojas se implemento el algoritmo de Bhargav. Este se basa en la manipulación de colores RGB, que para el caso de hojas de (*Philodendron spp*, se fijo el espectro a R=1, G=1 y B=0.45. Para asignar un centroide sobre las hojas. Y con la herramienta de corte obtuvimos secciones de $1cm$, $2cm$ de diámetro y $4cm$ de diámetro, sobre hojas de Helecho (*Philodendron spp*. Dando pauta como trabajo futuro la evaluación de estos cortes en laboratorio.

Palabras clave: hojas, modelo dinámico, robot manipulador, optimización global, visión artificial, control $PD+$.

[a]Tesista
[b]Director de tesis

SUMMARY

Design and construction of an automated system for cutting plant tissue

José Guillermo Cebada Reyes[a]
Federico Hahn Schlam[b]

In in vitro culture through leaves, the apical portion of the plant is considered, that is, the portion containing young leaves averaging 5 to 6 centimeters in length, to subsequently take 5 to 7 pieces of 2 cm^2 each per species. To accomplish this objective, a three degrees of freedom Cartesian robot was developed using Arduino electronic technology. The graphical user interface was developed with MATLAB GUIDE-creating interactive windows software that allows the user to easily manipulate and interpret the behavior of the robot.Moreover two cutting tools, a vertical cutter and punch was evaluated, finding that type punch tool generates best cuts without presenting desagarres, on the other hand presents vertical cutter lacerations of $1mm$. In this process, however, tuning problems were found in the PD+ control structure, which we solved using evolutionary and bio-inspired algorithms. The Cuckoo Search, JDE and DE algorithms are excellent methods for gains tuned. This adjustment was implemented in the controller, for positioning, visual servoing and trajectory approach. As identification system leaves Bhargav algorithm is implemented. This is based on the manipulation of RGB colors, which in the case of leaves (emph Philodendron spp), the spectrum is set to R = 1, G = 1 and B = 0.45. To allocate a centroid on the leaves. Moreover, with the cutting tool, we obtained pieces of 2 and $4cm$ in diameter, on fern leaves.

Key words: Cutting, leaves, dynamic model, Robot manipulator, global optimization, artificial vision, $PD+$ control.

[a]Doctoral student
[b]Principal advisor

Índice general

Índice de figuras

Índice de cuadros

Simbología

Símbolo	Parámetro
q	Posición expresada en coordenadas articulares
$\dot{q}$	Velocidad expresada en coordenadas articulares
$\ddot{q}$	Aceleración expresada en coordenadas articulares
$\tilde{q}$	Error de posición expresado en coordenadas articulares
q_d	Posición deseada expresada en coordenadas articulares
x	Posición expresada en coordenadas cartesianas
$\dot{x}$	Velocidad expresada en coordenadas cartesianas
$\ddot{x}$	Aceleración expresada en coordenadas cartesianas
$\tilde{x}$	Error de posición expresado en coordenadas cartesianas
x_d	Posición deseada expresada en coordenadas cartesianas
$J(q)$	Matriz Jacobiana
$\mathcal{K}(q, \dot{q})$	Energía cinética
$\mathcal{U}(q)$	Energía potencial
$\mathcal{L}(q, \dot{q})$	Lagrangiano

Símbolo	**Parámetro**
g	gravedad
$M(q)$	Matriz de masas e inercias expresada en coordenadas articulares
$C(q, \dot{q})$	Matriz de Coriolis y fuerza centrípeta en coordenadas articulares
$g(q)$	Par gravitacional expresado en coordenadas articulares
τ	Par aplicado expresado en coordenadas articulares
M_x	Matriz de masas e inercias expresada en coordenadas cartesianas
C_x	Matriz de Coriolis y fuerza centrípeta en coordenadas cartesianas
g_x	Par gravitacional expresado en coordenadas cartesianas
f_x	Fuerza lineal
m	masa
v	velocidad

Capítulo 1

Introducción

En la agricultura las innovaciones tecnológicas son de suma importancia para mejorar el rendimiento de los cultivos, su tiempo de producción y su adaptabilidad al clima y al suelo. Al utilizar tecnología se miden y cuantifican variables que permiten conocer y entender el comportamiento de los cultivos ante cambios de alguna variable [1]. El mejoramiento genético convencional o clásico (selección e hibridación) ha permitido obtener nuevas especies de plantas o híbridos con características agronómicas más favorables.

La propagación *in vitro* se basa en la habilidad de regenerar plantas completas a partir de la diferenciación y re-diferenciación de células vegetales, generalmente provenientes de plantas maduras. La regeneración es potenciada por dos vías morfo génicas, *la organogénesis* y la *embriogénesis somática* [1,3–7]. Estas respuestas se pueden obtener a través de:

- Cultivo de secciones de hoja.
- Cultivo de yemas axilares.
- Cultivo de meristemos.

La propagación a tráves de *secciones de hojas* tiene por objetivo el mejoramiento genético de algunas especies vegetales; por ejemplo en el café se busca la tolerancia del patógeno *H. vastatrix*. Esta es la enfermedad más destructiva del cafeto, debido a que esta enfermedad provoca la caída prematura de las hojas, propiciando la reducción de la capacidad fotosintética así como el debilitamiento de árboles enfermos [2].

El procedimiento, en general, consiste en seleccionar hojas jóvenes libres de patógenos y seccionarlos en tamaños no mayores de $2cm^2$ e inducirlos en un *medio de cultivo*, generando las condiciones artificiales para la obtención de un individuo nuevo [8].

Este es un proceso repetitivo, por lo que se han buscado nuevas tecnologías para la producción masiva de tejido vegetal. En este sentido, ha cobrado importancia las disciplinas de la mecatrónica y la robótica para potencializar y elevar la eficiencia de este proceso. Por ejemplo, se ha comprobado en la industria que estas máquinas hacen más eficiente los trabajos repetitivos, reduciendo costos en personal y tiempo de producción.

1.1. Antecedentes de la robótica en la agricultura

En cultivos hortícolas se han implementado máquinas que brindan una solución mecánica única capaz de realizar exactamente la misma tarea una y otra vez. Ejemplos son la mecanización de alta tecnología o robots que sustituyen el trabajo humano [9–11]. El primer robot desarrollado para tareas agrícolas que se tiene registrado, fue el desarrollado por Kawamura en 1982 en la Universidad de Kyoto, figura (1.1, cuya función fue la de recolectar tomates [12].

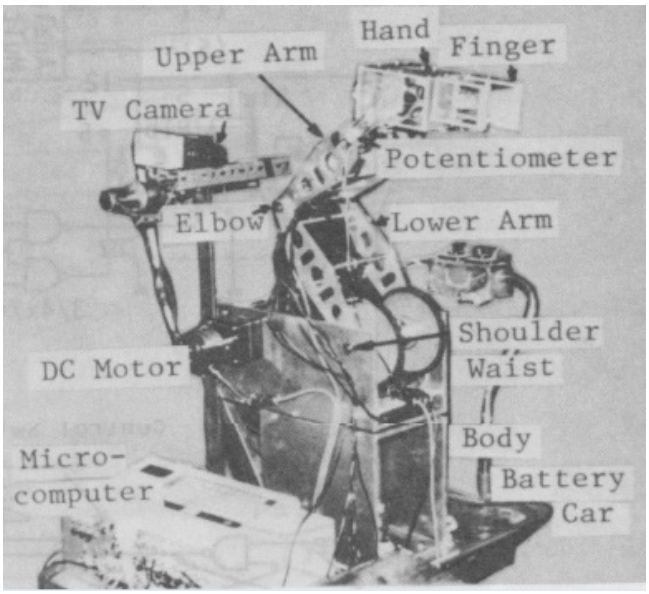


Figura 1.1: Primer robot dedicado a la recolección agrícola de Kawamura.

Posteriormente, Nishiura en 1996 desarrolla un robot dedicado a la injertación de plantas, debido que este es un proceso delicado que requiere un alto grado de habilidad física y mentalmente exigente. Este robot alcanzó una tasa de éxito del 97 % y 10 veces más rápido que los trabajadores humanos. Esta máquina es la única comercialmente disponible y puede ser utilizado para el injerto de pepino, melón de agua, melón, tomate y berenjena a una capacidad de 800 plantas por hora [9, 10, 12].

En el campo de la biotecnología vegetal, el cultivo *in vitro* consiste en cuatro etapas los cuales son:

- Fase de esterilización, los instrumentos y el material vegetal son desinfectados
- Corte del tejido vegetal.
- Propagación, se establecen los cortes en un medio de cultivo.
- Aclimatación, las plántulas son llevadas a invernadero.

En este sentido, se ha buscado la automatización de este proceso. En 1988 se diseña el primer esquema automatizado para este proceso y se hizo a través de la búsqueda de birreactores para la propagación masiva de tejido vegetal reportado en *Nature*, tomando en cuenta los trabajos realizados en la propagación de *Begonia* a través de medio líquido, figura (1.2). Este sistema mecatrónico, tiene la característica de tener un bioprocesador, que separa tejido por tamaño y distribuye la masa de tejido embriogénico u organogénico en envases de cultivo [13].

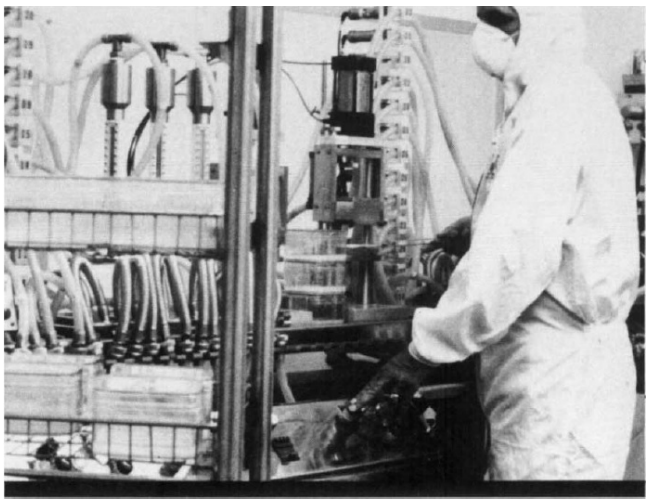


Figura 1.2: Biorreactor desarrollado por Levin.

En 1995 en el *CIRAD* de Mompellier se mejora el proceso, a través de la implementación de bombas de aire y recipientes gemelos para el intercambio de medio de cultivo e intercambio gaseoso [14]. En 1998 Kondo, a través de dos robots tipo PUMA recrean la manipulación de hojas de té para la propagación in vitro [12]. En 2011 Ashraf, Kondo y Shiigi desarrollan una máquina de visión para detección y manipulación de yemas axilares de tomate [15].

1.2. Definición y morfología de un robot

La definición más aceptada en el campo de la robótica es la propuesta por la Asociación de Industrias Robóticas (RIA), en 1981:

Un robot es un manipulador multifuncional reprogramable, capaz de mover material, piezas, herramientas, o dispositivos especializados, según trayectorias variables, programadas para realizar tareas diversas [11,16,17].

El robot manipulador está formado por un conjunto de eslabones interrelacionados por medio de articulaciones. Dichas articulaciones confieren la capacidad de realizar diversos movimientos independientes con respecto a los ejes x, y, z. Las articulaciones de un robot son de diferentes tipos, los cuales son:

- *Prismática*: permite desplazamientos lineales, por tanto tiene un grado de libertad de traslación.
- *Rotación*: permite girar respecto de un eje cartesiano, por tanto tiene un grado de libertad de rotación.
- *Cilíndrica*: es una combinación de las dos anteriores por lo tanto tiene dos grados de libertad.
- *Planar*: este tiene dos grados de libertad de traslación y uno de rotación.
- *Universal*: posee dos grados de libertad rotacional.
- *Esférica*: posee tres grados de libertad rotacional.
- *Helicoidal*: este tiene un grado de libertad de traslación y uno de rotación, relacionados por el paso de la rosca.

Para adentrarnos al conocimiento de las configuraciones básicas de los robots manipuladores, es preciso tocar el tema que se refiere al volumen de trabajo.

El volumen de trabajo de un robot se refiere únicamente al espacio dentro del cual puede desplazarse su extremo final [16, 17]. De esta forma definiremos las configuraciones básicas de los robots los cuales se nombra a continuación:

Cuadro 1.1: Configuraciones básicas de los robots manipuladores

Cartesiano	Presenta volúmenes de trabajo regulares, es decir, genera una figura cúbica. Posee tres movimientos lineales, los cuales corresponden a los ejes x, y, z
Cilíndrico	Presenta un volumen de trabajo parecido a un cilindro. Puede realizar dos movimientos lineales y uno rotacional.
Polar	Su volumen de trabajo es una esfera. Este robot tiene tres movimientos, un lineal, un rotacional y un angular.
S.C.A.R.A	Su volumen de trabajo es irregular. Este robot puede hacer movimientos horizontales y un movimiento lineal
Configuración angular (brazo articulado)	Su volumen de trabajo es irregular. Puede realizar tres movimientos un rotacional y dos angulares.

1.3. Hipótesis

Típicamente la tarea de corte de secciones de hoja como tejido vegetal, se realiza sobre una superficie plana dentro de una caja de Petri; involucrando dos herramientas de corte y otra de manipulación, es decir, el laboratorista sujeta las hojas usando pinzas de precisión, con una mano y con la otra realiza el corte con un bisturí. Este proceso se realiza bajo una campana de flujo laminar a una temperatura de $35°C$ lo cual representa un ambiente duro de trabajo. Para solucionar este tipo de problema se planteó un sistema conformado por un robot cartesiano el cual seccionara hojas en un plano cartesiano de dos dimensiones (x, y) con la ayuda de un sistema de visión.

1.4. Objetivos

El objetivo principal de esta investigación es diseñar un sistema de corte que permita obtener secciones de hojas para la propagación *in vitro*.
De este objetivo se desprende los siguientes objetivos particulares:

- Diseñar el sistema mecánico, el sistema de control electrónico y la interface que permita la ejecución de la tarea encomendada.
- Plantear la estructura de control del robot de corte y modelar el sistema.
- Implementar un sistema de visión que permita la ubicación de la herramienta de corte en el objetivo.
- Sintonización de la estructura de control, para obtener una buena respuesta del robot cortador en el esquema de regulación de posición y movimiento.
- Realizar cortes básicas sobre una hoja con diferentes herramientas de corte.

1.5. Estructura de la tesis

En este capítulo se han presentado: la introducción, hipótesis y el objetivo. El resto del documento está organizado de la siguiente forma.

En el capítulo 2 se presenta el diseño y construcción de todas las etapas del sistema, las herramientas a utilizar para seccionar hojas, así como el lazo de control electrónico. Por su parte el capítulo 3 expone el modelo matemático del robot, el cual se dividió en el estudio del controlador y la deducción del modelo dinámico. Al término del capítulo se expresa el modelo completo del sistema en términos de la transformación de *Arimoto* así como las consideraciones para llegar a este.

Otro punto importante que se toca en este capítulo es la definición de un trayectoria circular y la implementación de polinomios de quinto orden para estimar velocidades deseadas.

En el capítulo 4 se explica el modelo de visión que se implementó en el robot de corte y explica las transformaciones de los pixeles para trabajar en coordenadas reales, asi como las limitantes del algoritmo de detección. En el capítulo 5 se presentan los diferentes métodos de sintonización del controldor PD+, siendo este un problema importante desde 1987 para encontrar una metodología de sintonía para sistemas no lineales.

El capítulo 6 se presentan los resultados mas relevantes en este trabajo ya que en este se aprecia la aplicación de cuatro estrategias de control, siendo *posición*, *movimiento*, *optimización* y *visión*,explicados en seis experimentos diferentes. Donde el primer experimento es la evaluación del modelo dinámico del robot a diferentes sintonias. El experimento 2 explica la sintonización del controlador PD+ sin y con acciones acotadas a través del algoritmo de Cuckoo Search en dos puntos de prueba. El experimento 3 se relaciona con la búsqueda del método de evolución diferencial más eficiente para la sintonía del control PD+ de acciones saturadas. El experimento 4, se evalúa las diferentes herramientas de corte y los experimentos 5 y 6 se muestra la integración del sistema de visión al robot para la localización y disección de hojas.

En el capítulo 7 se presentan las conclusiones dados los objetivos alcanzados y posibles opciones de trabajo para futuros investigadores. Por último, en los apéndices A, B, C y D, se muestra los programas, demostraciones, la interfaz gráfica y las publicaciones aceptadas en esta tesis.

Capítulo 2

Diseño del robot de corte

En la actualidad, las actividades agrícolas tienden a la automatización de múltiples procesos utilizando una gran variedad de herramientas. Un robot es una herramienta clave capaz de realizar tareas repetitivas de forma rápida y precisa, ayudando a los seres humanos a reducir el síndrome *Repetitive Strain Injury* (RSI) [9,10]. En la propagación a través de hojas se considera la porción apical de la planta, es decir, la porción que contiene las hojas jóvenes de un tamaño promedio de 5 a 6 centímetros de longitud, para posteriormente tomar de 5 a 7 trozos de $2cm^2$ cada uno según la especie [19,21]. En este sentido, un robot cartesiano de tres grados de libertad es una herramienta desarrollada para la realización de esta tarea con el objetivo de hacer repetitivo este proceso.

2.1. Elementos del robot

Toda máquina está constituida por *mecanismos*, estos son un conjunto de elementos mecánicos que hacen una función determinada en un sistema para el desarrollo de una tarea encomendada. El mecanismo implementado en este sistema se llama *mecanismos de movimiento lineal*, estos mecanismos están conformados por cremalleras-piñón, husillo lineal, biela-manivela, corredera de movimiento osci-

lante, etcétera [19].

El robot cortador consta de tres eslabones, cada eje de los cuales posee dos mecanismos diferentes para su movimiento. Los eslabones son montados en baleros lineales que permiten el desplazamiento de los eslabones de manera uniforme y con poca fricción colocados en cada una de las masas del robot. En los ejes y, z, correspondientes a los eslabones 2 y 3 se usó un sistema cremallera-piñón unido a los motores, este mecanismo convierte el movimiento circular de un piñón en uno lineal continuo por parte de la cremallera, que es una barra rígida dentada, proporcionando una transmisión suave con una precisión de paso $2.5mm$ [21]. El eje x empleó un husillo de bolas el cual es un actuador lineal mecánico que convierte el movimiento de rotación en movimiento lineal con pocas pérdidas por fricción; su eje roscado proporciona un camino de rodadura helicoidal a unos rodamientos de bolas que actúan como un tornillo de precisión de paso $1.15mm$ [23, 24].

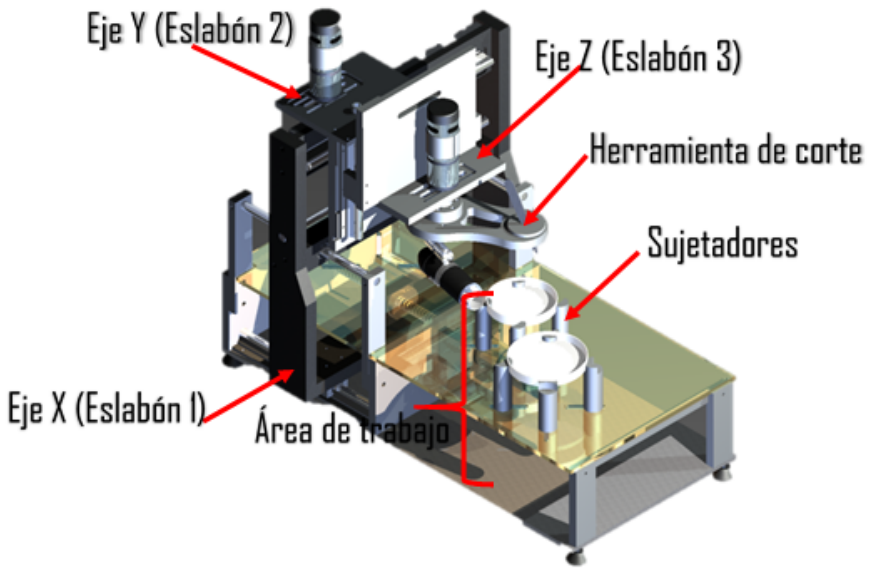


Figura 2.1: Componentes del robot cortador.

El área de trabajo del robot posee un juego de sujetadores de cajas de Petri modificadas, donde mantendrán las hojas a cortar en su interior, además de tener una lámpara UV para la descontaminación de las plántulas.

Las dimensiones y pesos de este sistema son presentadas en el siguiente Cuadro 2.1:

Cuadro 2.1: Características del robot de corte

Característica	Eslabón 1	Eslabón 2	Eslabón 3
Largo	40cm	20cm	15cm
Ancho	37cm	20cm	20cm
Masa	3.8kg	2.7kg	2.3kg

Para identificar el valor de los parámetros físicos del robot, se pesaron los eslabones con una báscula digital modelo 2008C con una resolución de 100gr, figura (2.2) las dimensiones del robot se midieron con un flexómetro.

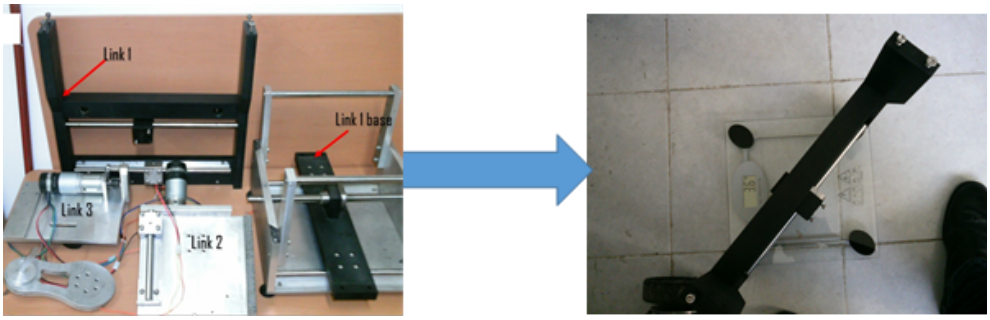


Figura 2.2: Forma de pesado de los eslabones del robot.

2.2. Características de los motores del robot de corte

Los *motorreductores* implementados en el robot son de la marca *EGM* 49 de 24 voltios; este dispositivo es un motor de corriente directa dotado de un encoder, permitiéndole ser llevado a posiciones angulares específicas al enviar una señal codificada. [17]. Cada motor posee un reductor de 49 : 1, un encoder incremental

magnético, además de poseer un supresor de ruido para elevar la eficiencia en aplicaciones con retro alimentación [28].

Cuadro (2.2):

Cuadro 2.2: Característica del motor EGM 49

Característica	Valor nominal
Voltaje de alimentación	$24V$
Velocidad máxima	$122rpm$
Corriente máxima	$2.1A$
Torque máxima (Nm)	$1.568Nm$
Potencia	$34.7W$
No. Pulsos	980

El consumo de corriente de cada motor que conforma el robot cartesiano a medida que este se le exija mover más peso va incrementándose.

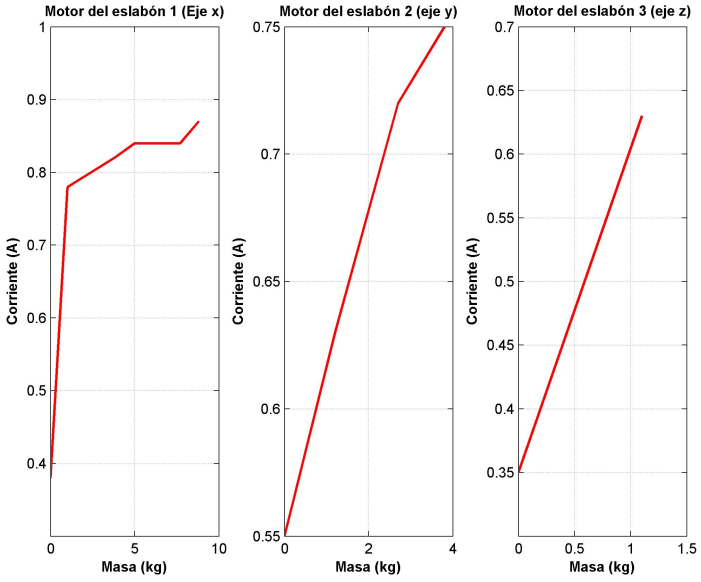


Figura 2.3: Corriente consumido por cada motor.

En la figura (2.3) se puede apreciar el consumo de corriente de cada motor se incrementará cundo se le exija mover un peso más grande. La Fig. 2.3a, muestra como la corriente del motor 1 del eje x va incrementándose conforme se aumenta

el peso, hasta llegar a un consumo $0.87A$. Esto se debe a que el motor mueve el peso de los tres eslabones $8.8kg$. La figura 2.3b, muestra el consumo de corriente ejercida por el motor 2. El consumo de corriente total del motor 2 eje y es de 0.75 A y carga un peso de $3.8kg$ correspondiente a los eslabones 2 y 3. En la figura. 2.3c, 3, se puede apreciar que el consumo de la corriente para este eslabón es de $0.63A$ para un peso de $1.1kg$.

El *encoder* es un transductor rotativo que transforma un movimiento angular en una serie de impulsos digitales; y para este robot se utilizaron encoders incrementales. Los encoders *incrementales* generan un número exactamente definido de impulsos por revolución. Éstos indican la medida de la distancia angular y lineal recorrida. Esto se logra proporcionado dos formas de onda cuadrada y desfasadas entres sí noventa grados, identificadas como canal A y canal B [19].

La unidad de medida para definir la precisión de un encoder es el grado eléctrico, éste determina la división de un impulso generado por el encoder: En efecto, los 360° eléctricos corresponden a la rotación mecánica del eje del motor, necesaria para realizar un ciclo o impulso de la señal de salida. Para saber a cuántos grados mecánicos corresponden trescientos sesenta grados eléctricos, se aplica la fórmula siguiente [25].

$$360^\circ electricos = \frac{360^\circ}{No.Pulsos} \tag{2.1}$$

Por lo tanto, aplicando (2.1) obtenemos la precisión del encoder:

$$360^\circ electricos = \frac{360^\circ}{980} = 0.367^\circ \tag{2.2}$$

El valor anterior nos indica la precisión por cada impulso del encoder.

Para obtener el paso mecánico para cada motor, utilizamos el valor de la ecuación

(2.2) y lo dividimos por la reducción mecánica del motor.

$$\theta_{reducido} = \frac{0.367}{49} = 0.0074^{\circ} \tag{2.3}$$

Posteriormente se considera el radio de cada elemento mecánico siendo para el piñón $r_p = 22.6mm$ y para el balero lineal $r_b = 15.8mm$, por lo tanto utilizando el valor de (2.3) el paso para cada motor es:

$$Paso_x = r_b\theta_{reducido} = 0.000116m \tag{2.4}$$

$$Paso_{y,z} = r_p\theta_{reducido} = 0.000167m \tag{2.5}$$

2.3. Sistema electrónico del robot de corte

El sistema electrónico que controla el prototipo se diseñó usando una plataforma de hardware libre, basada en una placa con un micro controlador Atmel y un entorno de desarrollo de la marca *Arduino*. Esta flexibilidad le permite ser compatible con MATLAB, LABVIEW, PHYTON, JAVA entre otros software de Ingeniería Asistida por Computadora CAE [24, 25].

La tarjeta utilizada para este fin fue el *Arduino Uno* Se basa en un micro controlador Atmel ATmega320 de 8 bits a $16Mhz$ que funciona a $5V$, $32KB$ de memoria flash ($0,5KB$ reservados para el bootloader), $2KB$ de SRAM y $1KB$ de EEPROM. Las salidas pueden trabajar a voltajes entre 5 y $12V$. Contiene 14 pines digitales, 6 de ellos se pueden emplear como PWM. Posee 6 pines analógicos que pueden trabajar con intensidades de corriente de hasta 40mA. Además de tener 2 pines de interrupciones, los cuales corresponden a los puertos 2 y 3 de la tarjeta arduino. [26, 28].

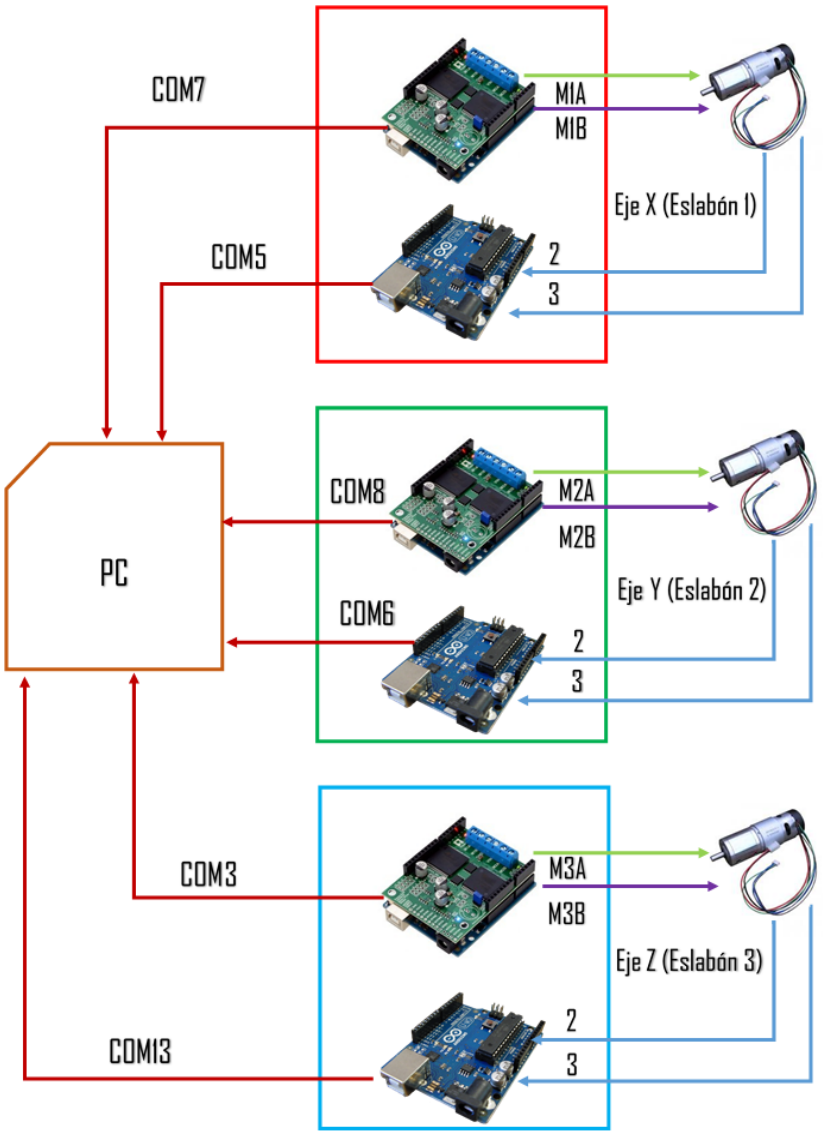


Figura 2.4: Diagrama de conexión para control de los motores del robot de corte.

En la figura (3.1) se puede apreciar la conexión de cada motor en su respectivo puerto de salida y entrada, donde los puertos de salida para el motor uno es $M1A$ para el positivo y $M1B$ para el negativo del motor uno, para el motor dos análogamente será $M2A$ y $M2B$ y de la misma manera el motor tres es $M3A$ y $M3B$; las entradas de las señales enviadas por los encoders se realizará con tres tarjetas arduinos independientes cuyos puertos serán el 2 y 3, conformando tres lazos de control.

La prioridad que dará la computadora será el lazo de control para el motor 1 correspondiente al eje x, si este no se ejecuta no se ejecutará los demás lazos de control, esta prioridad lo asignará la computadora de acuerdo al número de puerto que asigna para cada tarjeta arduino, siendo la $COM7$ el primer puerto que dará prioridad la computadora y así sucesivamente. Es importante señalar que este número dependerá del modelo y de la marca de la computadora donde se conectarán los arduinos. Por lo tanto para la computadora *Toshiba Satellite* C55 los puertos para cada tarjeta son $COM7$, $COM5$, $COM8$, $COM6$, $COM3$ y $COM13$ todos tipo USB [58].

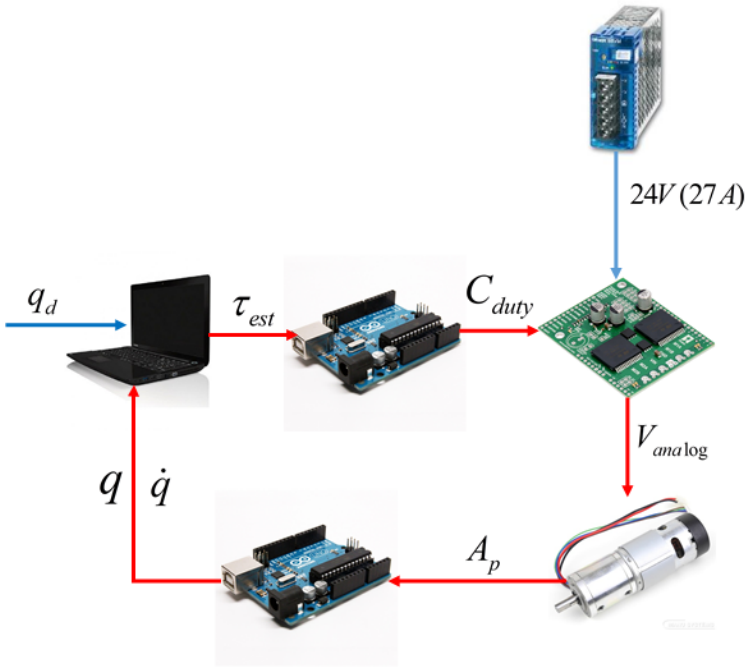


Figura 2.5: Lazo de control electrónico.

El lazo de control para cada uno de los motores está conformado por una tarjeta Arduino de lectura y otra de escritura acoplada a una tarjeta DC VNH5019, cuya función es controlar la velocidad y sentido de los motores a través de la modulación por ancho de pulso (PWM) dependiente de un par estimado (τ_{est}). Está modulación es una señal cuadrada donde su amplitud máxima es de $5V$, de

frecuencia constante y un *Duty Cycle* variable (C_{duty}), este último es un ciclo de trabajo que modula el ancho de pulso del PWM, es decir, controla la cantidad de potencia proveída a la carga. Dicha modulación es interpretada por una tarjeta DCVNH la cual contiene dispositivos MOSFET para controlar el motor usando una fuente de alimentación de 24 V. Este dispositivo activa los motores para entregar pulsos (A_p) que serán almacenados y procesados por la tarjeta de lectura Arduino para mandarlos de retorno a la computadora y este estime el error de posición del motor $\tilde{q}$, figura (2.5). Es decir el par aplicado, ecuación (2.6) será una función dependiente de las ganancias proporcional K_p y derivativa K_v, además de los errores de posición $\tilde{q}_d$ y velocidad $\dot{\tilde{q}}_d$, así como de la dinámica del robot τ_m [16,17].

$$\tau_{est} = f(K_p, K_v, \tilde{q}_d, \dot{\tilde{q}}_d, \tau_m) \tag{2.6}$$

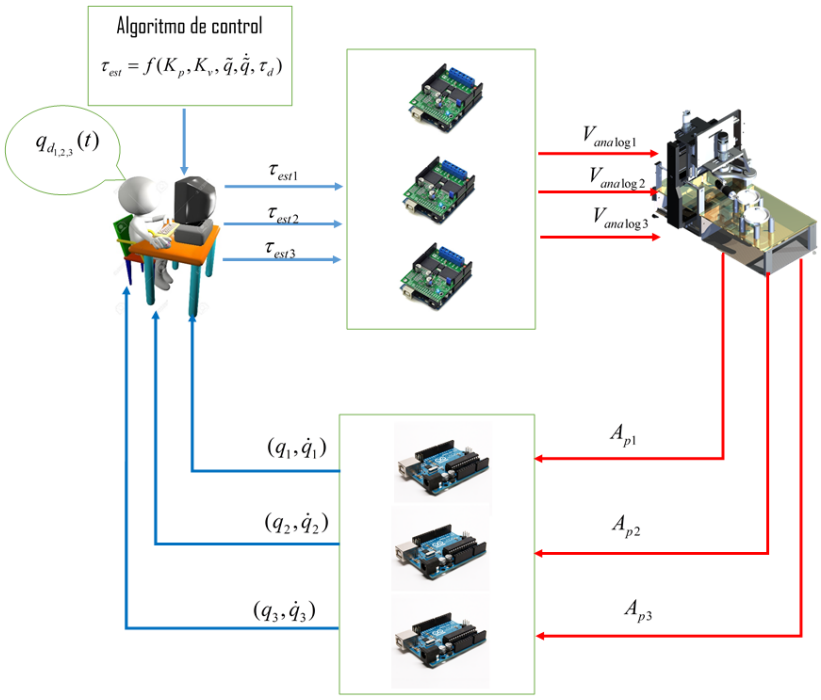


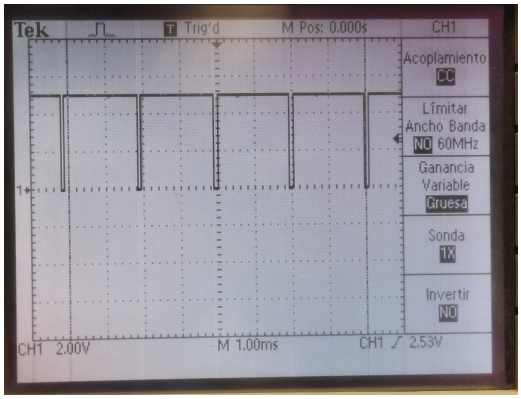
Figura 2.6: Funcionamiento del robot.

El funcionamiento general del robot cartesiano, figura (2.6), consiste en que cada posición deseada q_d es introducida por el *usuario*, como dato, de tal manera que el *algoritmo de control* cumpla el objetivo de determinar una función vectorial τ_{est} de tal forma que las posiciones y velocidades asociadas a las articulaciones del robot sigan con exactitud a las posiciones deseadas y velocidades deseadas estimadas [16, 17].

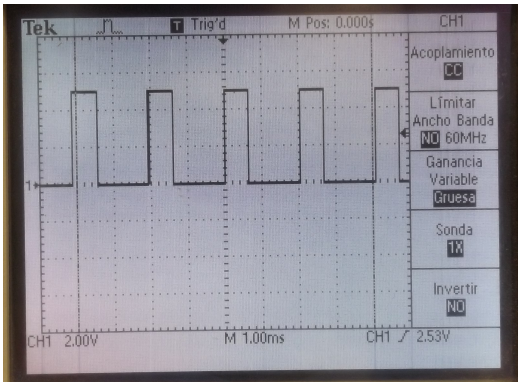
Para convertir el par estimado (τ_{est}) a *Ciclo de trabajo* (C_{duty}), se considera el torque máximo del motor obtenido de las características físicas del motor y la resolución del ADC de la tarjeta Arduino uno. Por cada valor (τ_{est}) estimado será asociada una palabra digital de 0 a 255, que corresponde a una palabra digital de 8 bits que la tarjeta Arduino transformara a un Duty cycle, ecuación (2.7), este ciclo de trabajo es interpretado por el DC VNH5019 y entrega un valor de voltaje de la fuente externa para el accionamiento de los motores del sistema cartesiano. Donde C_{duty} está definido como:

$$C_{duty} = \frac{(\tau_{est})(255)}{\tau_{max}} \tag{2.7}$$

En la figura (2.7) se puede apreciar la variación del ancho de pulso variable, siendo del 99 % para obtener un torque de $1,5Nm$ en el motor x (figura 2.7(a)); en la figura 2.7(b) el ciclo de trabajo es del 55 % y proporciona un par de $0.7Nm$. El mapeo que hace posible esta operación se realiza a tráves de la ecuación (2.7) y entregado a la tarjeta arduino con la ayuda del comando *analogWrite(C_{duty})*. La frecuencia del ciclo de trabajo es de $5kHz$ tal como se aprecia en la medición del osciloscopio.



(a) ***Comportamiento del PWM al 99 % de la capacidad***



(b) ***Comportamiento del PWM al 55 % de la capacidad***

Figura 2.7: Comportamiento del lazo de control eléctronico

En la figura (2.10), se aprecia la variación de voltaje entregado por el puente H en función del par estimado. El comportamiento de los 3 motores fue similar por lo que solo se muestra uno.

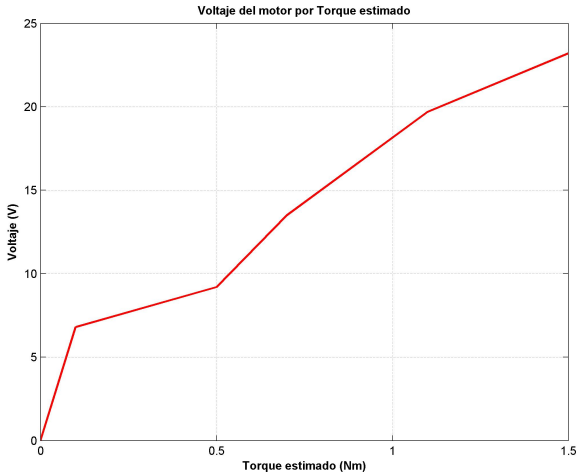


Figura 2.8: Comportamiento del motor de 24 Volts .

La lectura de datos de los encoders se hace a través de los puertos de interrupciones de la tarjeta Arduino [26], ya que estos periféricos permiten detectar eventos de forma asíncrona con independencia de las líneas de código que se estén ejecutando; es decir, detectará los cambios de flancos del encoder e irá acumulando los

pulsos, de tal manera que:

$$q = \frac{(A_p)(360^\circ)}{N_{pt}} \tag{2.8}$$

donde q es la posición real medida en grados, A_p es un número entero que representa la acumulación de pulsos y N_{pt} es el número de pulsos totales que entrega el encoder por cada vuelta del eje del motor. Estos datos son guardados en un vector y a través de una derivada de Euler, se estima la velocidad, ecuación (2.9).

$$\dot{q} = \frac{q(t+h) - q(t)}{h} \tag{2.9}$$

donde h es el tiempo de muestro.

2.4. Características de la GUI del sistema automatizado

La interfaz gráfica de usuario (*GUI*), se desarrolló usando la herramienta GUIDE-MATLAB, la cual permite ejecutar comandos de manera gráfica sin desaprovechar la potencia de la plataforma de MATLAB. Como se puede observar en la figura (2.9, al iniciar la *GUI* solicita la inicialización de las tarjetas Arduino, posteriormente abre un menú principal, donde el usuario tiene la opción de seleccionar *control de posición* para calibrar el sistema o definir una secuencia de puntos para establecer trayectorias llamado *control de movimiento*. Seleccionada la opción, inicia el modo gráfico y entra en operación el lazo de control hasta un tiempo final, terminando el bucle, y se cierra el programa [29].

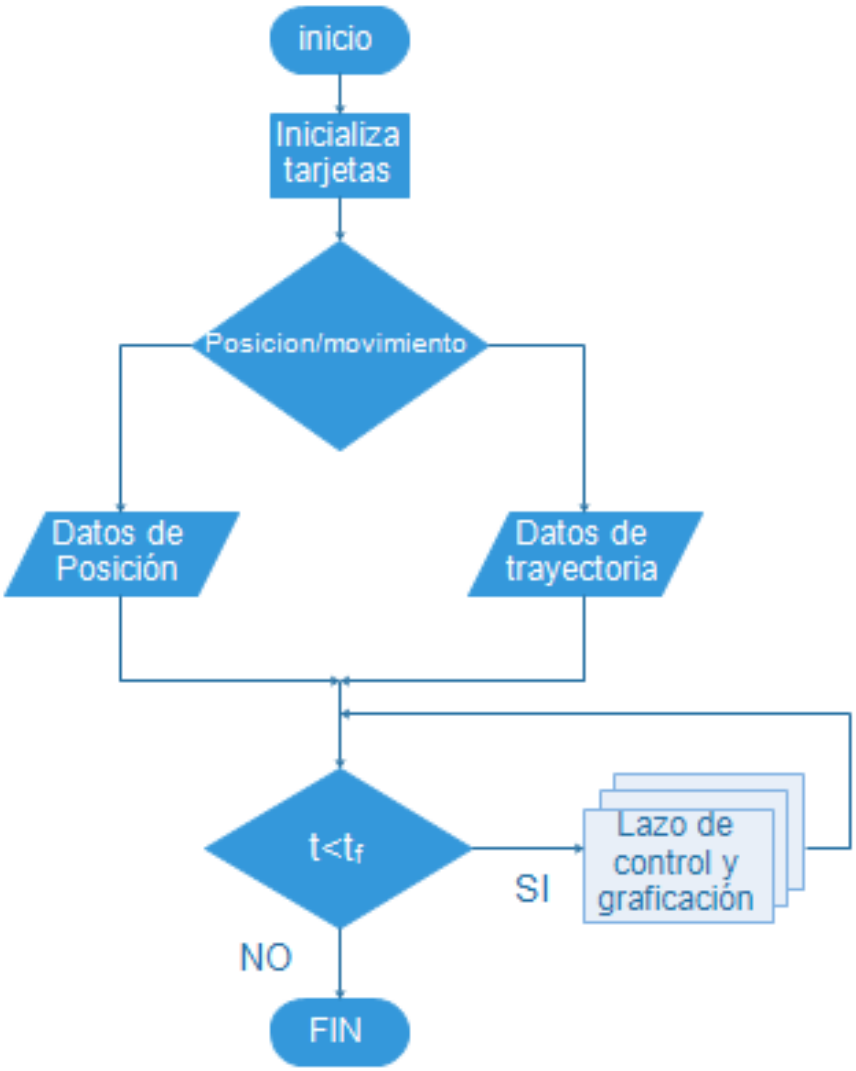


Figura 2.9: Diagrama de flujo de la interfaz principal.

La diferencia entre control de *posición* y de *movimiento* radica en el establecimiento de datos de entrada, es decir, en control de posición se establecen un punto para cada eslabón en la cual debe arribar el efector final, este movimiento describe rectas de manera aleatoria. En el caso de control de movimiento los datos de entrada son vectores de puntos para cada eslabón con la finalidad de desarrollar trayectorias cerradas de seguimiento, es decir, se define una forma o figura en $2D$ que el robot tratará de describir.

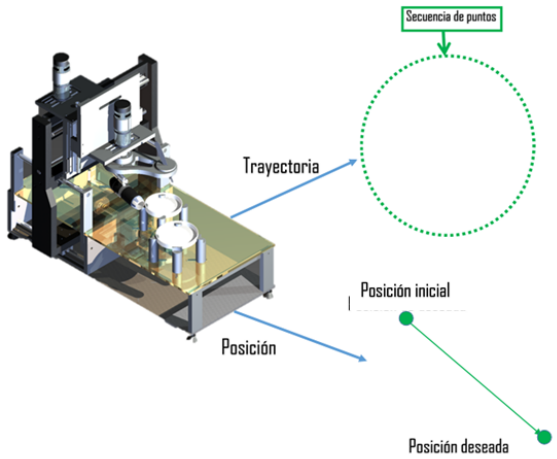


Figura 2.10: Ejemplificación de posición y trayectoria.

La inicilización de las tarjetas *Arduino* consiste principalmente en la verificación de conexión de cada tarjeta al interfaz gráfico de usuario, para realizar esta tarea, *Matlab*® y la marca *Arduino* desarrollaron un controlador que se descarga de manera gratuito en la página de internet `es.mathworks.com`, tecleando en su buscador *controlador Arduino* para *Matlab*® se obtiene el driver así como algunos ejemplos ilustrativos para su utilización.

2.4.1. Rutina de Asignación de datos al lazo de control

En el siguiente diagrama de flujo, figura 2.11, describe el primer subprograma. Se puede apreciar que los datos enviados por el usuario y la tarjeta *Arduino* de lectura, son utilizados para estimar la posición y velocidad del robot utilizando las ecuaciones (2.8) y (2.9) para obtener los errores de velocidad y posición, ecuaciones (2.10) y (2.11).

$$\tilde{q} = q_d - q \tag{2.10}$$

$$\dot{\tilde{q}} = \dot{q}_d - \dot{q} \tag{2.11}$$

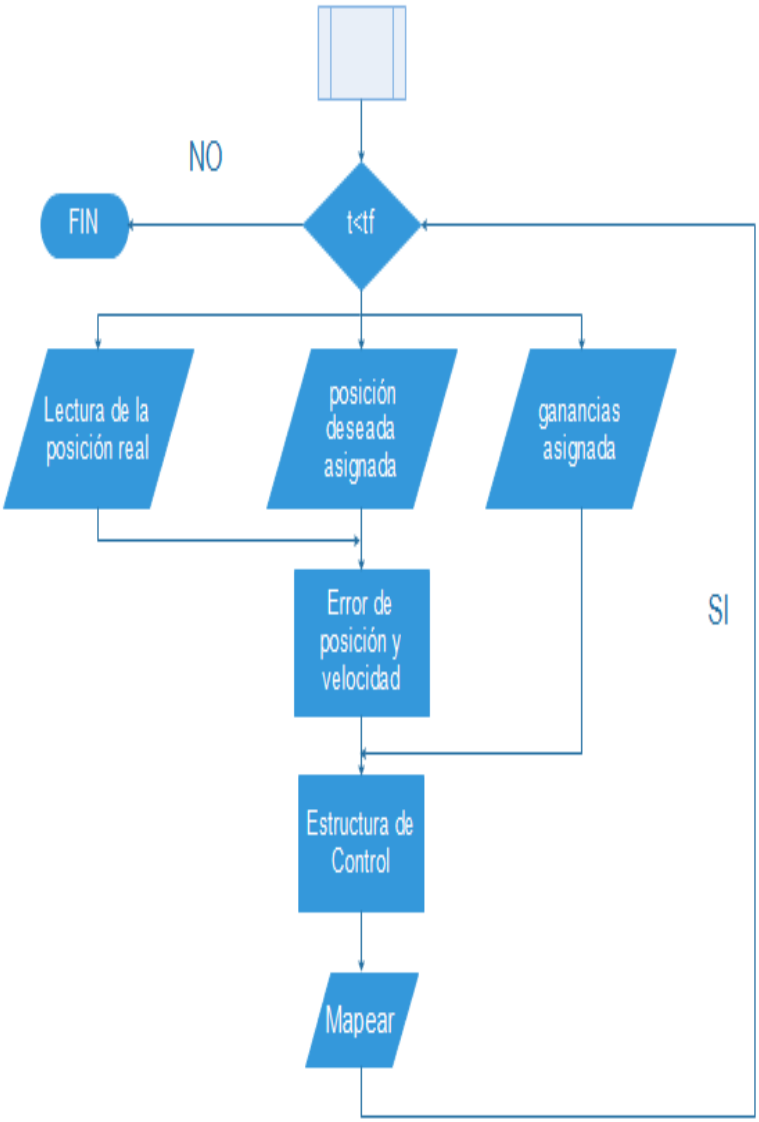


Figura 2.11: Rutina de asignación de datos de la GUI.

Para lograr la tarea de asignación de datos dentro de la rutina principal de la interfaz gráfica de usuario, se diseñó dos subprogramas. El primer subprograma llamado *Rutina de asignación de datos* permite el cálculo de un torque estimado a través de una *estructura de control*. Que básicamente es una ecuación matemática que depende de las ganancias asiganadas por el usuario, el error de posición y el error de velocidad, ecuación (2.6), para su posterior mapeo mediante la ecuación (2.8)y accionamiento de los motores, figura (2.12).

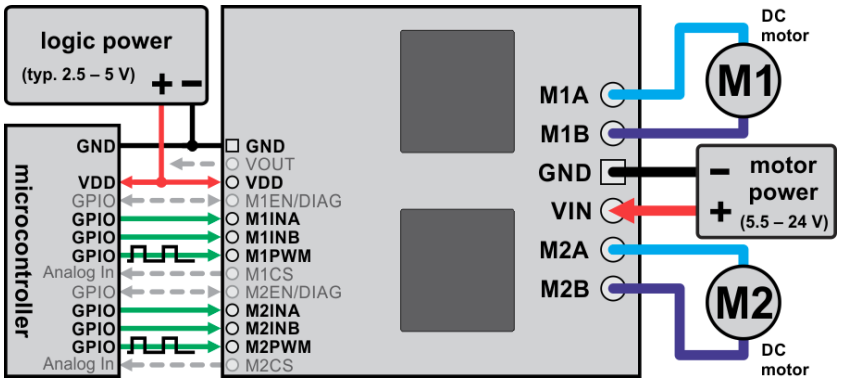


Figura 2.12: Esquema del DC VNH5019.

Posteriormente son enviados a la estructura de control, (definido en el capítulo 3), para estimar un torque que será mapeado como un , *Ciclo de trabajo* a través de la ecuación (2.7) que será enviados de la computadora a los puertos al convertidor análogo-digital del Arduino que son los puertos 9 y 10 que conecta a la tarjeta DC VNH5019 para que interprete la cantidad de voltaje que le proporcionara a los motores del robot, figura (2.12).

2.4.2. Rutina de graficación

El segundo subprograma,figura (2.13) ,utiliza los datos enviados por el encoder del robot y los datos estimados por el primer sub programas para ser visualizados por el modo gráfico de GUIDE-MATLAB a través del comando *encoderRead(puerto de entrada)*. Para el graficado, se establecen vectores, los cuales van almacenando los datos entregados por las tarjetas de lectura y las operaciones de estimación. Los vectores son llamados a través de la rutina *plot* el cual toma el punto anterior y el punto siguiente y lo une con una línea, y visualizarlos a través de un *axes* y entregados través de un archivo tipo *.txt* [27].

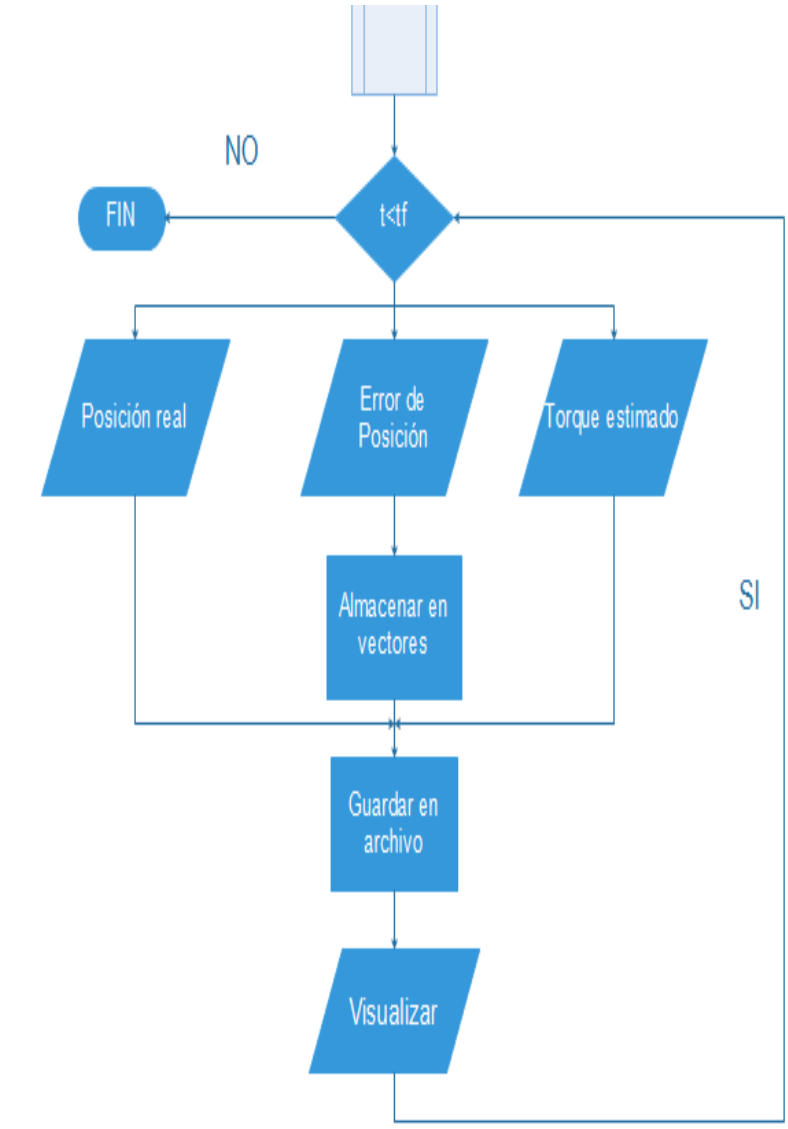


Figura 2.13: Rutina de graficación de datos de la GUI.

2.5. Herramientas de corte

2.5.1. Cortador vertical

Este tipo de herramienta de corte se utiliza en los procesos de fresado. Dependiendo del tipo de trabajo y material que se tenga que maquinar, se debe utilizar el tipo de cortador con el número de gavilanes (filos de corte) adecuado. El criterio

para elegir el tipo de cortador depende de la dureza del material, entre más duro sea el material, él número de gavilanes será mayor, entre más suave será menor. En el caso para seccionar hojas se utilizará un cortador de $2mm$ de dos gavilanes de acero inoxidable con una velocidad de rotación de $3000rpm$.

Figura 2.14: Cortador vertical.

Esta herramienta estará montada en el eslabón 3 correspondiente al eje z y trabajará con el esquema de control de trayectorias. Es decir, se asignará una secuencia de puntos que el robot pueda seguir en los ejes (x, y), y de manera controlada la herramienta baje para desgarrar el tejido hasta diseccionar el pedazo deseado.

2.5.2. Sacabocados

Los sacabocados se utilizan para cortar secciones redondas en el interior de diferentes materiales, los usos y aplicaciones específicos. Los materiales que son capaces de cortar son espuma de poliuretano, hule, cartón, cuero, piel y material para juntas. Están fabricados de acero alto carbón, dureza de 53 a $66HRC$, figura (2.15).

Esta herramienta como la anterior estará montada en el eslabón 3 correspondiente al eje z y trabajará con el esquema de control de posición. Es decir, se asignará un punto de arribo para que el robot pueda dejar caer de manera controlada la herramienta para generar una perforación sobre el tejido a clonar.

Figura 2.15: Cortador tipo sacabocados.

Capítulo 3

Control y descripción matemática

Antes de diseñar o implementar un esquema de control [65], es importante tomar en cuenta:

- Estudio del sistema que será controlado y decidir qué tipos de sensores y actuadores deben ser utilizados.
- Modelar el sistema que será controlado.
- Simplificar el modelo si es necesario.
- Analizar el resultado del modelo.
- Decidir sobre las especificaciones de desempeño.
- Decidir el tipo de controlador a utilizar.
- Diseñar el controlador y dar a conocer sus especificaciones.
- Simular el sistema a controlar.
- Repetir el paso 1.
- Sintonizar el controlador.

Los modelos dinámicos son abstracciones de la realidad en este sentido el comportamiento del robot puede determinarse mediante un modelo matemático que describa su dinámica; Este se obtiene a través de la aplicación de las ecuaciones de movimiento de *Euler-Lagrange*. El modelo matemático servirá para la aplicación de controladores de movimiento o trayectorias y la utilización de los métodos de optimización global para hallar ganancias que permitirán ajustar del comportamiento del sistema [16–19].

3.1. Control de movimiento

El *Control de movimiento* o seguimiento de trayectorias, consiste en determinar una función vectorial τ de tal forma que las posiciones y velocidades asociadas a las articulaciones del robot sigan con exactitud a las posiciones y velocidades deseadas, respectivamente [16, 17].

En otras palabras, el *objetivo de control de movimiento* consiste en encontrar τ tal que:

$$\lim_{t\to\infty} \begin{bmatrix} \tilde{q} \\ \dot{\tilde{q}} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{3.1}$$

En la figura (3.1) se muestra el diagrama a bloques del control de trayectorias del robot cortador. Obsérvese que las variables que definen el problema de control tales como el error de posición y error de velocidad son procesadas por la estructura matemática de control, cual requiere el conocimiento completo de la dinámica del robot. Esto brindara robustez, exactitud y desempeño en este tipo de estructuras de control, entre las que destacan *el control PD+* y el *control par calculado*.

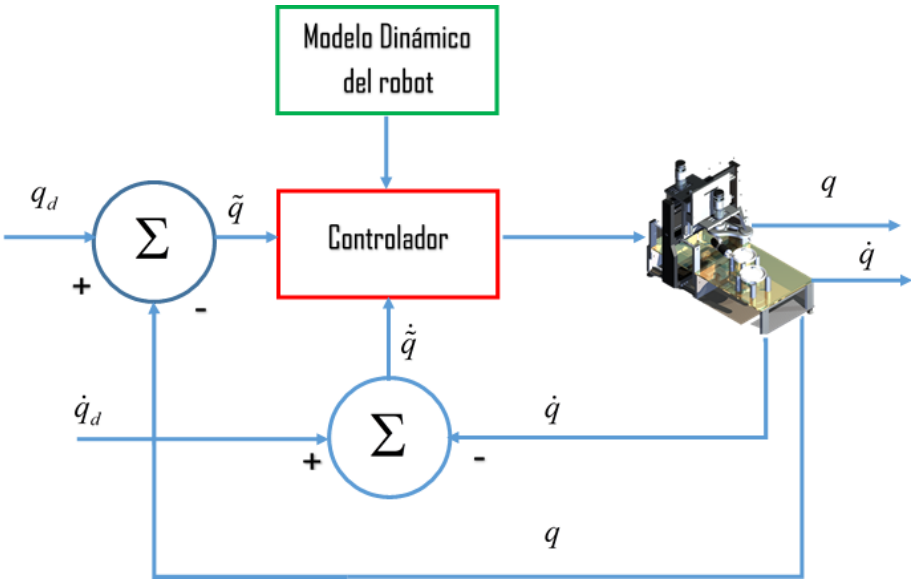


Figura 3.1: Lazo de control de movimiento.

3.2. Controlador de movimiento PD+

El control de movimiento o trayectorias tiene la función de controlar el error de posición $\tilde{q}$ y el error de velocidad $\dot{\tilde{q}}$. Estas variables a controlar son procesadas por el controlador, que básicamente es una estructura matemática la cual requiere el conocimiento de la dinámica del robot. La primera estructura de control diseñada fue propuesta por Paden y Panja. en 1987. Esta estructura de control para fines de robótica, incluye control proporcional del error de posición, control proporcional del error de velocidad más la dinámica completa del robot manipulador, brindando robustez, esta estructura de control es conocida como PD+ [16, 17, 30, 31], denotada por τ_{pd+}.

Originalmente esta estructura de control tiene la forma:

$$\tau_{pd+} = K_p\tilde{q} + K_v\dot{\tilde{q}} + \tau_m \tag{3.2}$$

donde K_p, $K_v \in \mathbb{R}^{n\times n}$ son matrices de ganancias simétricas definidas positivas, $\tilde{q} = q_d - q \in \mathbb{R}^{n\times 1}$ es el error de posición, $\dot{\tilde{q}} = \dot{q_d} - \dot{q} \in \mathbb{R}^{n\times 1}$ es el error de la velocidad y $\tau_m \in \mathbb{R}^{n\times 1}$ es el *modelo dinámico* del sistema.

Sin embargo este tipo de controlador tiene la desventaja de presentar el problema de saturación. La saturación es un fenómeno no lineal que es muy frecuente en control de robots manipuladores; causando que los servomotores operen en sus límites físicos, conocido como región de saturación. Este problema se presenta cuando los algoritmos de control demandan señales grandes de control, como es el caso de este tipo de estructura, ecuación (3.2) [16]. Para resolver este problema surge una familia de este tipo de controladores llamados de *acciones acotadas*.

3.2.1. Control PD+ con acciones acotadas.

Típicamente para proponer variantes de este tipo de controlador se realiza a través del método de moldeo de energías (energy shamping), el cual considera el modelo dinámico con ausencia de fricciones y otras perturbaciones [16, 17]. Una variante de este controlador con buen desempeño es la de acciones acotadas que se caracteriza por mantener la acción de control dentro de los límites de saturación del servo amplificador, es decir el torque estimado no saldrá de los límites físicos del torque real entregado por el servo amplificador para cada eslabón, ecuación (3.3), este efecto de acotamiento lo entrega la función tangente hiperbólica o aproximaciones trigonométricas de esta función:

$$|\tau_i| \leq \tau_i^{max} \tag{3.3}$$

Esta estructura de control para fines de robótica, se mantiene en los límites la acción de control proporcional del error de posición y el control proporcional del error de velocidad [16, 17], donde:

$$\tau_{pd+} = K_p \psi_{\nabla U_p} + K_v \psi_{\nabla U_v} + \tau_m$$

$$\psi_{\nabla U_p} = \begin{bmatrix} \tanh(\tilde{q_1}) & \tanh(\tilde{q_2}) & \dots & \tanh(\tilde{q_n}) \end{bmatrix} \tag{3.4}$$

$$\psi_{\nabla U_v} = \begin{bmatrix} \tanh(\dot{\tilde{q_1}}) & \tanh(\dot{\tilde{q_2}}) & \dots & \tanh(\dot{\tilde{q_n}}) \end{bmatrix}$$

En la estructura de este esquema de control se involucra la trayectoria de seguimiento, velocidad y aceleración deseada. Observe que el esquema de control $PD+$ de acciones acotadas, ecuación (3.4), cuando la posición deseada q_d no es variante en el tiempo, es decir es constante, la velocidad deseada $\dot{q}_d$ y la aceleración deseada $\ddot{q}_d$ son vectores cero. Por otro lado, el error de velocidad, cumple que $\dot{\tilde{q}} = -\dot{q}$ [16, 17], es decir tiene la ventaja de convertirse en un esquema de *control de posición* o *regulación*.

3.2.2. Control de posición.

El esquema de control de posición consiste en mover el extremo final del robot manipulador desde cualquier posición inicial hacia una posición deseada. Esto significa que la $i-$ésima articulación del robot, deberá moverse hacia la respectiva $i-$ésima posición deseada [16, 17]. Matemáticamente esto es:

$$\lim_{t \to \infty} \begin{bmatrix} \tilde{q} \\ \dot{q} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{3.5}$$

El algoritmo de control (3.4) puede ser formulado de la siguiente manera:

$$\tau_{pd} = K_p \tanh(\tilde{q}) - K_v \tanh(\dot{q}) \tag{3.6}$$

donde K_p, $K_v \in \mathbb{R}^{n\times n}$ son matrices de ganancias simétricas definidas positivas, $\tilde{q} = q_d - q \in \mathbb{R}^{n\times 1}$ es el error de posiciona una referencia fija, $\dot{q} \in \mathbb{R}^{n\times 1}$ es el vector de la velocidad y $\tau \in \mathbb{R}^{n\times 1}$ es la fuerza generalizada o par aplicado [11,12,28,29].

3.3. Modelo dinámico

El *modelo dinámico* de un robot permite explicar todos los fenómenos físicos que se encuentran en su estructura mecánica. La mecánica analítica representa la herramienta sólida de las ciencias exactas para formular modelos matemáticos que estudia la relación que existe entre las fuerzas que actúan sobre un cuerpo y el movimiento que en él se origina.

Para calcular τ_m de la ecuación (3.4) utilizamos las ecuaciones de movimiento de *Euler-Lagrange*. Esto nos permitirá más adelante hacer un análisis y mejoras en la estructura de control del robot de corte.

3.3.1. Método de Euler-Lagrange

Los robots manipuladores son sistemas mecánicos articulados con eslabones conectados entre sí a través de uniones o articulaciones. Las articulaciones son básicamente de tres tipos: rotacionales, angulares y traslacionales. Las posiciones correspondientes a cada articulación del robot, que se miden por medio de sensores colocados en los actuadores localizados generalmente justo en las articulaciones, las posiciones se agrupan en el vector de posición articular q [16, 17]. En consecuencia, para un robot con n grados de libertad, el vector de posición de q tendrá n elementos:

$$q = \begin{bmatrix} q_1 \\ q_2 \\ \vdots \\ q_n \end{bmatrix} \tag{3.7}$$

Ahora bien, *el modelo dinámico de un robot manipulador* es aquel que describe el comportamiento de un sistema a un estímulo específico, ya sea este un estímulo interno o externo [16,17,19]; esta representación matemática se obtiene aplicando leyes físicas para interpretar la dinámica del robot, cuya representación matemática es:

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) + f(\tau,\dot{q}) = \tau \tag{3.8}$$

donde:

- $q, \dot{q}, \ddot{q} \in \mathbb{R}^{n\times 1}$ son el vector de posición, velocidad y aceleración articular del robot, respectivamente.
- $M(q) \in \mathbb{R}^{n\times n}$ es la matriz de inercias.
- $C(q,\dot{q}) \in \mathbb{R}^{n\times n}$ es la matriz de Coriolis y fuerza centrípeta
- $g(q) \in \mathbb{R}^{n\times 1}$ es el par gravitacional
- $f(\tau,\dot{q}) \in \mathbb{R}^{n\times 1}$ es el vector de fricción
- $\tau \in \mathbb{R}^{n\times 1}$ es el par aplicado.

Un método ampliamente utilizado para obtener el modelo dinámico de un robot manipulador es el método de Euler-Lagrange, el cual utiliza la diferencia entre la energía cinética $\mathcal{K}$ y la energía potencial $\mathcal{U}$ que está definida por: [11–13],

$$\mathcal{L}(q, \dot{q}) = \mathcal{K}(q, \dot{q}) - \mathcal{U}(q) \tag{3.9}$$

donde $\mathcal{L}(q, \dot{q})$ es el lagrangiano.

Las ecuaciones de movimiento de Lagrange para un manipulador de n grados de libertad, están dadas por:

$$\frac{d}{dt}\left[\frac{\partial \mathcal{L}(q, \dot{q})}{\partial \dot{q}}\right] - \frac{\partial \mathcal{L}(q, \dot{q})}{\partial \dot{q}} = \tau \tag{3.10}$$

o de forma equivalente

$$\frac{d}{dt}\left[\frac{\partial \mathcal{L}(q, \dot{q})}{\partial \dot{q}_i}\right] - \frac{\partial \mathcal{L}(q, \dot{q})}{\partial \dot{q}_i} = \tau_i \qquad i = 1, ..., n \tag{3.11}$$

donde τ_i son las fuerzas generalizadas o pares ejercidos externamente (por actuadores) en cada articulación así como fuerzas no conservativas.

El uso de las ecuaciones de Euler-Lagrange para obtener el modelo dinámico de manipuladores se reduce a seis pasos [11]:

- Cálculo de la cinemática directa.
- Cálculo de los vectores de velocidad.
- Cálculo de la rapidez del robot.
- Cálculo de la energía cinética $\mathcal{K}(q, \dot{q})$ y potencial $\mathcal{U}(q)$.
- Obtención del Lagrangiano.
- Aplicación de las ecuaciones de movimiento de Euler-Lagrange.

3.3.2. Desarrollo del modelo dinámico

Paso 1. Cinemática directa de un robot: describe la relación entre la posición articular q y la posición espacial (x, y, z) del efector final del robot [11–13].

$$x = f(q) \tag{3.12}$$

Para el caso particular de cortar tejido vegetal para la propagación basada en la porción apical de la planta, se construyó un robot manipulador tipo cartesiano de tres grados de liberta, con atención a los investigadores, para realizar de manera repetitiva el trabajo. En base a la figura 3.2 que representa el movimiento espacial llamado cinemática directa del robot se obtiene:

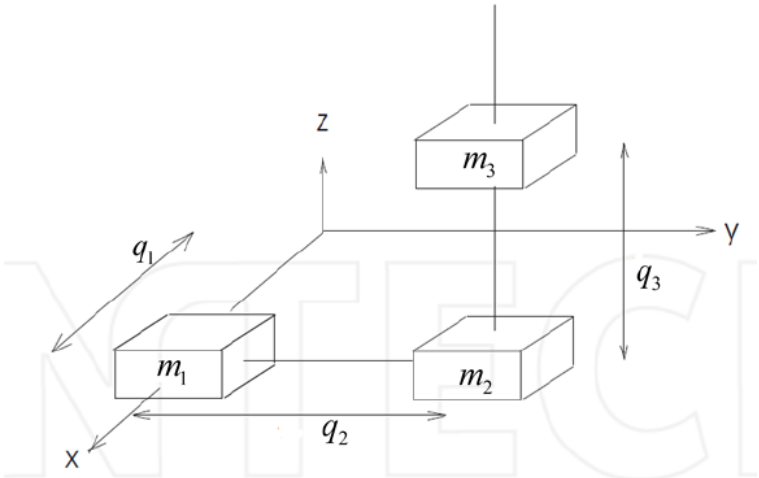


Figura 3.2: Cinématica directa del robot.

Aplicando análisis geométrico y trigonometría encontramos la cinemática directa para cada eslabón expresadas en el plano (x,y,z). Por lo tanto para el eslabón 1 esta representado como:

$$\begin{aligned} x_1 &= q_1, \\ y_1 &= 0 \\ z_1 &= 0 \end{aligned} \tag{3.13}$$

para el eslabón 2

$$\begin{aligned} x_2 &= q_1, \\ y_2 &= q_2 \\ z_2 &= 0 \end{aligned} \tag{3.14}$$

finalmente para el eslabón 3

$$\begin{aligned} x_3 &= q_1, \\ y_3 &= q_2 \\ z_3 &= q_3 \end{aligned} \tag{3.15}$$

Paso 2. Vectores de velocidad: Se obtiene derivando la cinemática directa.

$$v = \frac{d}{dt}\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} \tag{3.16}$$

Paso 3. Rapidez del sistema: se calcula aplicando la norma del vector $v^2 = \|v\| = v^T v$ para cada eslabón, por lo tanto:

Eslabón 1

$$v_1^2 = \dot{q_1}^2 \tag{3.17}$$

Eslabón 2

$$v_2^2 = \dot{q_1}^2 + \dot{q_2}^2 \tag{3.18}$$

Eslabón 3

$$v_3^2 = \dot{q_1}^2 + \dot{q_2}^2 + \dot{q_3}^2 \tag{3.19}$$

Paso 4. Obtención de la energía cinética y potencial. La *Energía cinética* está definida como el trabajo necesario para acelerar un partícula de una masa dada desde su posición de equilibrio hasta una velocidad deseada, definida como:

$$\mathcal{K}(q, \dot{q}) = \frac{1}{2}mv^2 + \frac{1}{2}I\dot{q}^2 \tag{3.20}$$

donde:

- m es la masa de la partícula.
- v es la velocidad de la partícula.
- I es el momentum de inercia de la partícula.
- q es la distancia angular de la partícula.

La *Energía potencial gravitatoria* es la energía que una partícula tiene debido a su posición en el espacio, no a su movimiento [9].
Cuya ecuación es:

$$\mathcal{U}(q) = m\,g\,h \tag{3.21}$$

donde:

- m es la masa de la partícula.
- g es la fuerza de gravitatoria.
- h es la altura o la componente y.

Las *energías cinéticas* de cada eslabón del robot cartesiano no presenta momento de inercia, por lo tanto queda representado de la siguiente forma:
Eslabón 1:

$$\mathcal{K}_1(q,\dot{q}) = \frac{1}{2} m_1 \dot{q_1}^2 \tag{3.22}$$

Eslabón 2:

$$\mathcal{K}_2(q,\dot{q}) = \frac{1}{2} m_2 [\dot{q_1}^2 + \dot{q_2}^2] \tag{3.23}$$

Eslabón 3:

$$\mathcal{K}_3(q,\dot{q}) = \frac{1}{2} m_3 [\dot{q_1}^2 + \dot{q_2}^2 + \dot{q_3}^2] \tag{3.24}$$

Ahora bien, usando la ecuación (3.21) y la componente en z la energía potencial esta dado definido como:

$$\mathcal{U}_3(q) = m_3 g q_3 \tag{3.25}$$

Paso 5. Lagrangiano: El lagrangiano es la diferencia entre la energía cinética y la energía potencial, ecuación (3.9), que para el caso del robot cartesiano está definido como:

$$\mathcal{L}(q,\dot{q}) = \tfrac{m_1+m_2+m_3}{2} q_1^2 + \tfrac{m_2+m_3}{2} q_2^2 + \tfrac{m_3}{2} q_3^2 - m_3 g q_3 \tag{3.26}$$

Paso 6. Ecuación de movimiento de Euler-Lagrange: Una vez hallado el lagrangiano (3.26), se aplica la ecuación (3.10) y se obtiene cada componente:

$$\frac{d}{dt}\left[\frac{\partial \mathcal{L}(q,\dot{q})}{\partial \dot{q}_i}\right] = \begin{bmatrix} m_1+m_2+m_3 & 0 & 0 \\ 0 & m_2+m_3 & 0 \\ 0 & 0 & m_3 \end{bmatrix} \begin{bmatrix} \ddot{q}_1 \\ \ddot{q}_2 \\ \ddot{q}_3 \end{bmatrix} \tag{3.27}$$

Finalmente de la ecuación el termino $\frac{\partial \mathcal{L}(q,\dot{q})}{\partial q_i}$:

$$\frac{\partial \mathcal{L}(q,\dot{q})}{\partial q_i} = \begin{bmatrix} 0 \\ 0 \\ m_3 \end{bmatrix} g \tag{3.28}$$

donde $i = 1, 2, 3$

Los terminos (3.27) y (3.28) de la ecuación (3.10) nos permite conocer el modelo dinámico del robot cartesiano de corte, definido como:

$$\begin{bmatrix} \tau_1 \\ \tau_2 \\ \tau_3 \end{bmatrix} = \begin{bmatrix} m_1 + m_2 + m_3 & 0 & 0 \\ 0 & m_2 + m_3 & 0 \\ 0 & 0 & m_3 \end{bmatrix} \begin{bmatrix} \ddot{q}_1 \\ \ddot{q}_2 \\ \ddot{q}_3 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ m_3 \end{bmatrix} g \tag{3.29}$$

Es importante resaltar que los robots cartesianos no poseen la matriz de Coriolis y fuerza centrípeta [11, 12, 17, 18].

3.4. Mapeo del espacio de trabajo del modelo dinámico

Las coordenadas cartesianas facilitan la interpretación del planteamiento de problemas y su implementación. En contraste, las coordenadas articulares o generalizadas puede resultar mucho más complicadas. Para realizar esta transformación se utiliza las ecuaciones de mapeo de *Arimoto*, esta metodología se basa en el Jacobiano transpuesto de cualquier robot manipulador [16, 32].

Estas relaciones matemáticas que permiten el cambio de espacio de trabajo son:

$$
\begin{aligned}
f_x &= J^{-T}(q)\tau \\
M_x &= J^{-T}(q)M(q)J^{-1}(q) \\
C_x &= J^{-T}(q)C(q,\dot{q})J^{-1}(q) - M_x\dot{J}(q)J^{-1}(q) \\
g_x &= J^{-T}(q)g(q)
\end{aligned}
\tag{3.30}
$$

donde f_x representa la fuerza lineal, M_x es la matriz de masas e inercias expresada en coordenadas cartesianas, C_x es matriz de Coriolis y fuerzas centrípetas cartesiana y $g(x)$ es el vector de par gravitacional expresado en coordenadas cartesianas [16,32].

Por lo tanto el modelo dinámico de un robot manipulador de n grados de libertad, ecuación (3.8), en coordenadas cartesianas queda expresada como:

$$M_x\ddot{\chi} + C_x\dot{\chi} + g_x = f_x \tag{3.31}$$

3.4.1. Mapeo de las coordenadas de trabajo

Si la trayectoria deseada se encuentra en coordenadas cartesianas $x_d(t)$, entonces las posiciones articulares en coordenadas generalizadas $q_d(t)$ se obtiene directamente de la cinemática inversa [11,30], y viceversa [16,19,32]. Estas relaciones de posición, velocidad y la aceleración se mapea de la siguiente manera:

Cuadro 3.1: Ecuaciones de mapeo

Coordenadas articulares	Coordenadas cartesianas
$q = f^{-1}(x)$	$x = f(q)$
$\dot{q} = J^{-1}(q)\dot{x}$	$\dot{x} = J(q)\dot{q}$
$\ddot{q} = J^{-1}(q)\ddot{x} - J^{-1}(q)\dot{J}(q)J^{-1}(q)\dot{x}$	$\ddot{x} = J(q)\ddot{q} + \dot{J}(q)\dot{q}$

3.4.2. Jacobiano del robot

El Jacobiano es una matriz que se puede ver como la versión vectorial de la derivada de una función escalar. El Jacobiano es importante en el mapeo de coordenadas generalizadas q a coordenadas cartesianas x para lograr la ejecución del control de movimiento [19]. Por lo tanto el Jacobiano para el problema de robótica esta descrito como:

$$J(q) = \begin{bmatrix} \frac{\partial f_1(q)}{\partial q_1} & \frac{\partial f_1(q)}{\partial q_2} & \cdots & \frac{\partial f_1(q)}{\partial q_n} \\ \frac{\partial f_2(q)}{\partial q_1} & \frac{\partial f_1(q)}{\partial q_2} & \cdots & \frac{\partial f_1(q)}{\partial q_n} \\ \vdots & & & \\ \frac{\partial f_n(q)}{\partial q_1} & \frac{\partial f_n(q)}{\partial q_2} & \cdots & \frac{\partial f_n(q)}{\partial q_n} \end{bmatrix} \tag{3.32}$$

Para hallar el Jacobiano del robot cortador, aplicamos la ecuación (3.32) en el término matemático (3.15) y tenemos:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = \begin{bmatrix} \frac{\partial x}{\partial q_1} & \frac{\partial x}{\partial q_2} & \frac{\partial x}{\partial q_3} \\ \frac{\partial y}{\partial q_1} & \frac{\partial y}{\partial q_2} & \frac{\partial y}{\partial q_3} \\ \frac{\partial z}{\partial q_1} & \frac{\partial x}{\partial q_2} & \frac{\partial x}{\partial q_3} \end{bmatrix} \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{bmatrix} \tag{3.33}$$

donde $J(q)$ es:

$$J(q) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.34}$$

por lo tanto el término de mapeo para la velocidad del robot es:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{bmatrix} \tag{3.35}$$

Como se puede observar en la ecuación (3.34)la Jacobiano del robot cortador es matriz

Identidad. Se sabe que $I^{-1} = I$ y la derivada de la matriz identidad es la matriz nula $\frac{dI}{dt} = N$. Por lo tanto la relación de aceleración (3.30) para el robot cartesiano es:

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \ddot{q}_1 \\ \ddot{q}_2 \\ \ddot{q}_3 \end{bmatrix} \tag{3.36}$$

3.5. Modelo dinámico en espacio cartesiano del robot

A lo largo de esta sección abordamos el modelado de un robot cartesiano de tres grados de libertad cuya función es cortar secciones de hojas. Para definirlo en términos de las coordenadas conocidas (coordenadas cartesianas) es necesario ocupar las transformaciones de *Arimoto*, relaciones matemáticas (3.30), a continuación se abordara paso a paso esta transformación:

Paso 1. Para obtener la matriz de masas e inercias, se procede:

$$M_x = \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{J(q)^{-T}} \underbrace{\begin{bmatrix} m_1+m_2+m_3 & 0 & 0 \\ 0 & m_2+m_3 & 0 \\ 0 & 0 & m_3 \end{bmatrix}}_{M(q)} \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{J(q)^{-1}} \tag{3.37}$$

resolviendo se tiene:

$$M_x = \begin{bmatrix} m_1+m_2+m_3 & 0 & 0 \\ 0 & m_2+m_3 & 0 \\ 0 & 0 & m_3 \end{bmatrix} \tag{3.38}$$

Por otra parte en la ecuación (3.29), no presenta fenómeno de Coriolis.

Paso 2. Se obtiene el vector de pares gravitacionales cartesianas:

$$g_x = \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{J(q)^{-T}} \underbrace{\begin{bmatrix} 0 \\ 0 \\ m_3 \end{bmatrix}}_{g(q)} g \tag{3.39}$$

obteniendo:

$$g_x = \begin{bmatrix} 0 \\ 0 \\ m_3 \end{bmatrix} g \tag{3.40}$$

Paso 3. Finalmente se obtiene el vector de fuerzas lineales:

$$f_x = \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{J(q)^{-T}} \underbrace{\begin{bmatrix} \tau_{m1} \\ \tau_{m2} \\ \tau_{m3} \end{bmatrix}}_{\tau_m} \tag{3.41}$$

lo que da:

$$f_x = \begin{bmatrix} \tau_{m1} \\ \tau_{m2} \\ \tau_{m3} \end{bmatrix} \tag{3.42}$$

En conjunto el modelo dinámico del robot cartesiano expresado en fuerzas lineal es:

$$\begin{bmatrix} f_{x1} \\ f_{x2} \\ f_{x3} \end{bmatrix} = \underbrace{\begin{bmatrix} m_1 + m_2 + m_3 & 0 & 0 \\ 0 & m_2 + m_3 & 0 \\ 0 & 0 & m_3 \end{bmatrix}}_{M_x} \underbrace{\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix}}_{\ddot{x}} + \underbrace{\begin{bmatrix} 0 \\ 0 \\ m_3 \end{bmatrix} g}_{g_x} \tag{3.43}$$

por lo tanto el controlador en espacio de trabajo cartesiano es:

$$f_{pd+} = K_p \psi_{\nabla u_p} + K_v \psi_{\nabla u_v} + f_{xm}$$

$$\psi_{\nabla u_p} = \begin{bmatrix} \tanh(\tilde{\chi_1}) & \tanh(\tilde{\chi_2}) & \dots & \tanh(\tilde{\chi_n}) \end{bmatrix} \tag{3.44}$$

$$\psi_{\nabla u_v} = \begin{bmatrix} \tanh(\dot{\tilde{\chi}}_1) & \tanh(\dot{\tilde{\chi}}_2) & \dots & \tanh(\dot{\tilde{\chi}}_n) \end{bmatrix}$$

Este análisis matemático se puede apreciar que el modelo *articuar* y el modelo *lineal*, para este tipo de robot *tiene una relación uno a uno*.

3.6. Trayectoria de prueba y polinomios de quinto orden

La trayectoria de entrada debe seleccionarse para exhibir un perfil de movimiento sin cambios abruptos en la posición, velocidad y aceleración de principio a fin del movimiento, así como generar un bajo contenido de fricción, de tal manera que la dinámica del robot sea la que predomine, además de prevenir que los servo actuadores entren en zona de saturación [33]. La trayectoria que servirá para obtener secciones de hojas, será una circular de $2cm^2$ [21]. Cada sección de ese tamaño sirve para clonar la planta madre, es decir, propaga de tejido vegetal con fines de bio preservación, como se muestra en la figura (3.3).

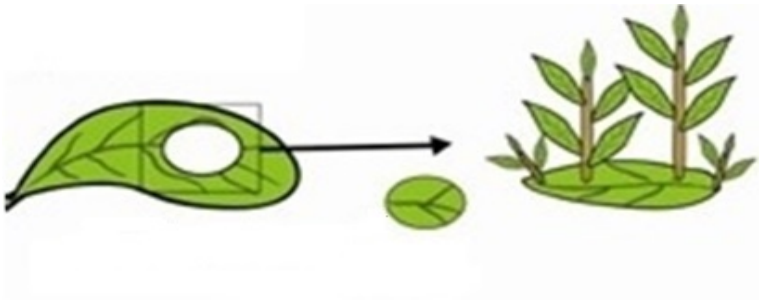


Figura 3.3: Secciones circulares de hoja.

Esta trayectoria está definida, para el caso de control de movimiento del cortador vertical como:

$$\begin{bmatrix} x \\ \\ y \end{bmatrix} = \begin{bmatrix} x_c + r\sin(t\omega) \\ \\ y_c + r\cos(t\omega) \end{bmatrix} \tag{3.45}$$

donde x_c y y_c representan las coordenadas del centro del círculo, r es el radio del círculo, ω es el periodo en el que se realizara el trazado del círculo, t es la evolución del tiempo.

Mientras que para el caso del cortador de tipo sacabocados se utilizará el control punto a punto definido matemáticamente en la ecuación (2017).

Para obtener unas trayectorias suaves en el espacio cartesiano del robot cortador de un punto inicial a un punto final deseado se logra a través de polinomios de quinto, esto permite que el efector final comience y termine con velocidades y aceleración nula. De esta manera se realiza en un tiempo finito [33–35]. Por lo tanto la posición deseada en espacio cartesiano para el i esimo elemento $x_d(t)$, en el caso del movimiento del eje x se define como:

$$x_{di}(t) = a_{x0} + a_{x1}t + a_{x2}t^2 + a_{x3}t^3 + a_{x4}t^4 + a_{x5}t^5 \tag{3.46}$$

Derivando está ecuación con respecto al tiempo se puede obtener la velocidad $\dot{x_d}(t)$ y la aceleración deseada $\ddot{x_d}(t)$:

$$\dot{x}_{di}(t) = a_{x1} + 2a_{x2}t + 3a_{x3}t^2 + 4a_{x4}t^3 + 5a_{x5}t^4 \tag{3.47}$$

$$\ddot{x}_{di}(t) = 2a_{x2} + 6a_{x3}t + 12a_{x4}t^2 + 20a_{x5}t^3 \tag{3.48}$$

Los coeficientes $a_{x0}, a_{x1}, a_{x2}, a_{x3}, a_{x4}, a_{x5}$ se calculan para satisfacer las posiciones inicial y final deseadas (para $t = 0$ y $t = t_f$, respectivamente), que de acuerdo a [31–33], son:

$$
\begin{aligned}
a_{x0} &= x_i \\
a_{x1} &= 0 \\
a_{x2} &= 0 \\
a_{x3} &= \tfrac{10}{t^3}(x_f - x_i) \\
a_{x4} &= \tfrac{15}{t^4}(x_f - x_i) \\
a_{x5} &= \tfrac{6}{t^5}(x_f - x_i)
\end{aligned}
\tag{3.49}
$$

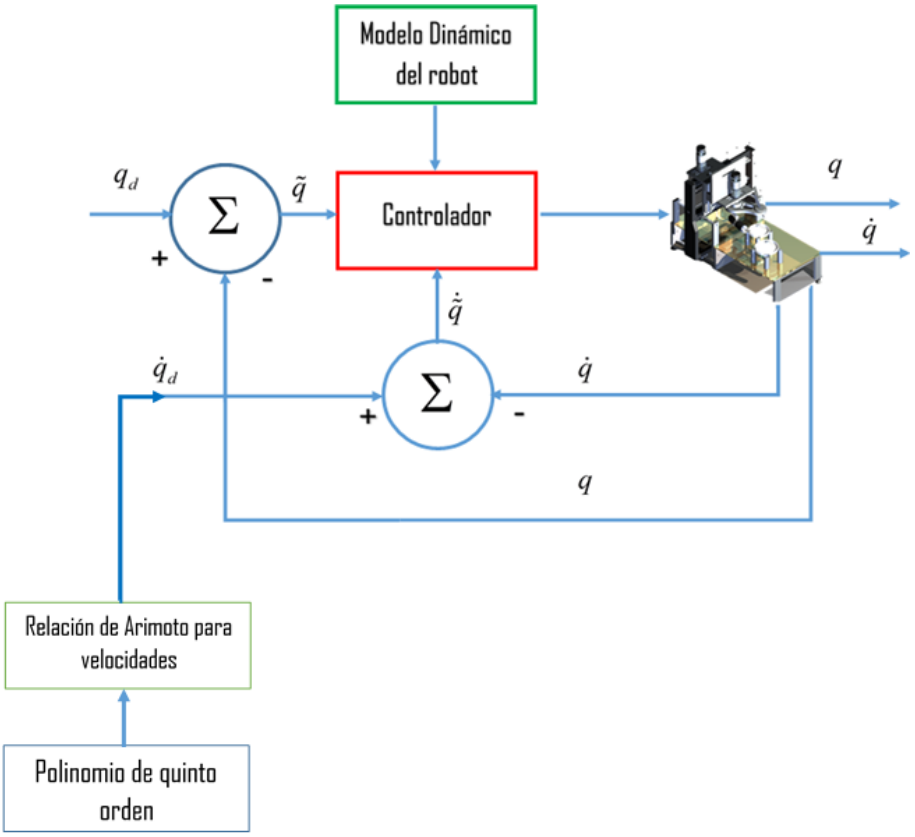


Figura 3.4: Lazo de control de movimiento con polinomio de quinto orden.

donde x_f es la posición final deseada y x_i es la posición inicial deseada. Los cálculos para $y_{di}, \dot{y}_{di}, \ddot{y}_{di}$, se realiza de manera similar. Estas ecuaciones estiman la velocidad deseada $\dot{q}_d$ a través de las relaciones de Arimoto, figura (3.4).

Capítulo 4

Visual servoing

El control Visual (Visual servoing) se refiere al empleo de un sistema de visión para controlar el movimiento de un robot. Los datos se adquieren desde una cámara fijada directamente en el robot manipulador o en un robot móvil. El movimiento es inducido por la cámara, la cual puede estar fija en el lugar de trabajo con una configuración estacionaria [59].

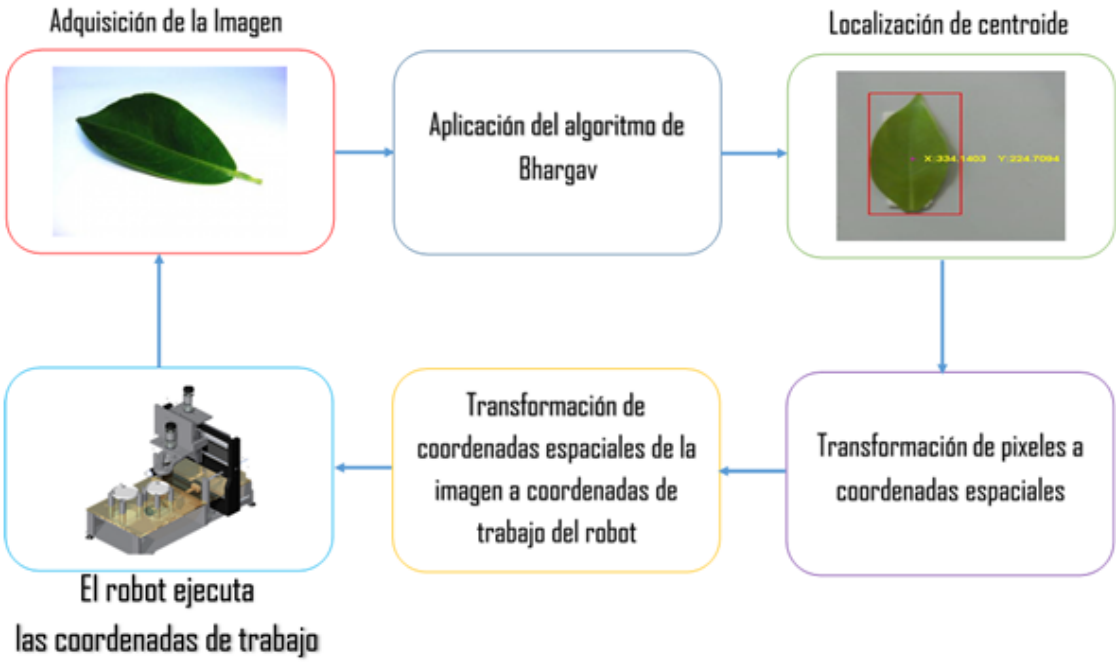


Figura 4.1: Etapas de funcionamiento del sistema de visión del robot.

Una vez definido los componentes del sistema de visión se identifican los eventos que involucraran el desarrollo de la tarea, en la figura (4.1), el primer evento es la adquisición de la imagen que desea analizar. El segundo evento es el procesamiento para hallar el centroide de la(s) hoja(s) a cortar. El tercer evento transforma los pixeles de la imagen a coordenadas espaciales a través de la ecuación (4.9)de los objetos detectados por el algoritmo de Bhargav. El cuarto evento transforma los centroides de los objetos detectados a puntos de desplazamiento para el robot a través de las ecuaciones (4.10) y (4.11) y obtener dos vectores de puntos de trabajo para el robot $x \in \mathbb{R}^n$ y $y \in \mathbb{R}^n$ y enviar estos puntos a las ecuación (3.45) para efectuar control de movimiento o bien control de posición con el PD+ (3.44).

4.1. Arquitectura del control visual

Cuando se introduce un sistema de visión como lazo de retroalimentación se busca incrementar la flexibilidad y capacidad de interacción de este con el medio ambiente. Es común que en aplicaciones donde se involucra control visual, la complejidad en el análisis de la imagen, incrementa el costo de la tarea. Por esta razón necesario simplificar la tarea mediante sistemas mecánicos y de iluminación. [60].

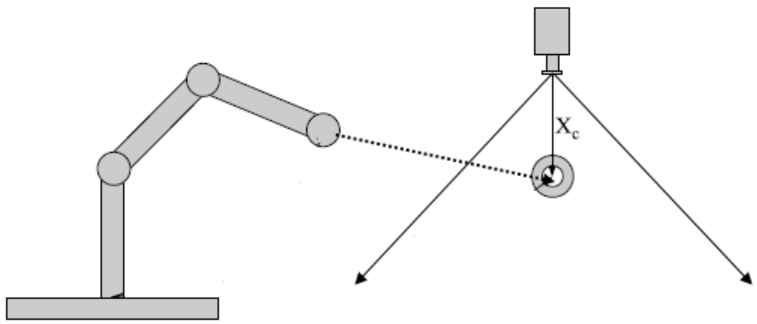


Figura 4.2: Configuración cámara fija para la retroalimentación visual del robot.

El control visual ha sido incorporado a los robots para dotarlos con la capacidad de in-

teractuar con su entorno, por lo que se dispone de dos configuraciones básicas cámara robot [34]: *cámara en mano* y *cámara fija*. Para detectar plántulas en el robot de corte se opta por la configuración de *cámara fija* el cual tiene la característica de que la cámara se encuentra fija en el espacio de trabajo; por lo que la cámara capturará el espacio de trabajo del robot y/o al robot, figura (4.2).

Esta estructura de control visual usa la estrategia de visión de *mirar y mover estático*, el cual consiste en extraer los rasgos visuales de una imagen inicial del espacio de trabajo y la consecuencia es la acción de control del robot. El robot ejecuta la tarea considerando que el espacio de trabajo no se ha modificado. Como se puede observar en la figura (4.3), esta estrategia de control por visión tiene la característica de que el lazo de control para el robot y la extracción de imágenes son dos tareas secuenciales.

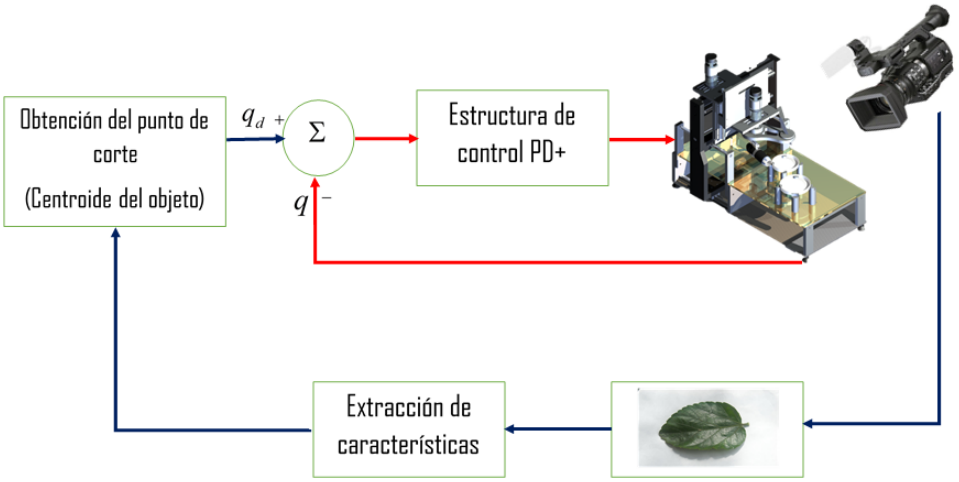


Figura 4.3: Estrategia mirar y mover.

En esta configuración la adquisición de imágenes posee *la prioridad* como evento, para proporcionarles las coordenadas (x_c, y_c) a la ecuación (3.45) descrita en el capítulo 3, a través de la detección de centroides. Se designa el punto de partida en una trayectoria circular e iniciando como *segundo evento* el control de movimiento para el cortador vertical. También puede proporcionar una referencia deseada x_d para ejecutar el control de posición para designar un punto de golpe para el cortador tipo sacabocados.

4.2. Elementos del lazo de control visual

Para desarrollar el lazo de control vía retro alimentación visual *mirar y mover* es necesario considerar la iluminación del área de trabajo, las características de la cámara, la orientación de la cámara y el algoritmo de detección de objetos.

4.2.1. Iluminación

La iluminación juega un papel importante en la visión artificial, pues simplifica el análisis y la interpretación de la escena captada.

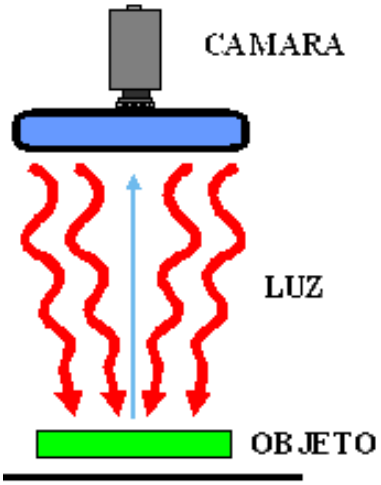


Figura 4.4: Orientación de la luz.

Para lograr el mejoramiento de la captación de plántulas se utilizó la iluminación tipo *direccional*, figura (4.4), el cual consiste en aplicar una iluminación orientada al objeto usando un haz de luz blanca orientada a un ángulo de orientación Θ [61]. Para el caso de detección de plántulas, se utilizó una orientación perpendicular es decir $\Theta = 90^{\circ}$ tanto para la *iluminación* como la *orientación de la cámara*. Esto se debe a que las hojas dentro del área de trabajo están sobrepuestas a una base blanca. El tipo de luz utilizada para el experimento es de halogeno blanco de $600 lumen$ a $60 Watts$.

4.2.2. Características de la cámara

La cámara capta la información luminosa de la escena, y la transmite a la computadora como una señal analógica o digital. Actualmente estas cámaras son de estado sólido y utilizan *dispositivos de carga acoplada*, CCD, como elementos de sensado [61]. La cámara utilizada para la detección de plántulas es la cámara *Logitech C270*, cuyas características [63] son:

Cuadro 4.1: Características de la camara Logitech C270

Característica	Valor
Distancia focal	$4mm$
Tamaño del pixel	$2.8 \times 2.8\mu m$
Resolución de la imágen	$640 \times 480 pixeles$

La cámara esta fija a una distancia del objetivo de $39cm$ en paralelo con el área de trabajo.

4.2.3. Algoritmo de detección de objetos

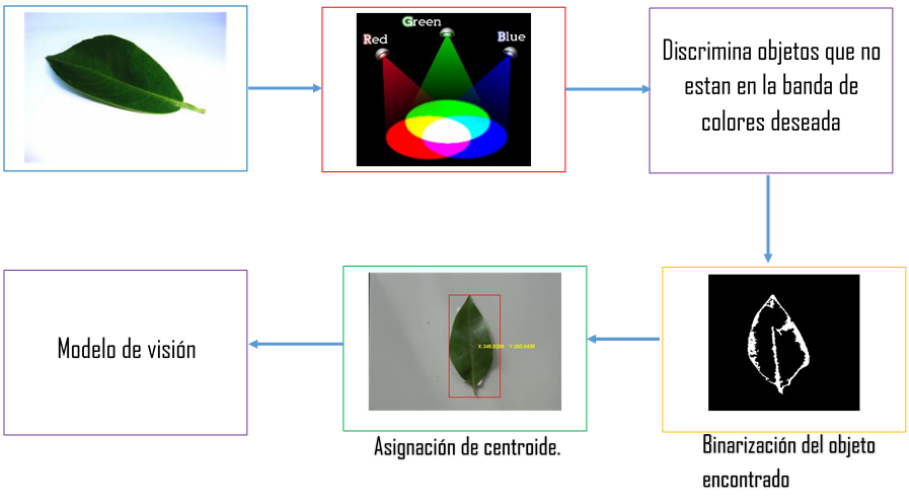


Figura 4.5: Funcionamiento del algoritmo de Bhargav .

En 2010 Bhargav Anand desarrolló un detector de objetos que se basa en la manipulación del plano de colores *RGB*. Para captar de colores como el rojo, amarillo y diferentes tona-

lidades de verdes, azul entre otros. Donde la manipulación del plano de colores se realiza a través de la siguiente ecuación (4.1):

$$W = (R_{max}/R_{ajus}) + (G_{max}/G_{ajus}) + (B/B_{ajus}) \tag{4.1}$$

donde R_{max} es el valor máximo del rojo, G_{max} es el valor máximo del verde y B_{max} es el valor máximo del azul, cuyo valor máximo es uno; R_{ajus}, G_{ajus} y B_{ajus} son los factores de atenuación para los planos rojo, verde y azul, estos valores son asignados por el usuario y varían de $[0.1, 1]$ [62]

Una vez detectados estos objetos el algoritmo asigna un *centroide* que es un punto que define el centro geométrico del objeto a localizar [62]. Si se considera que los pixeles del objeto tiene un peso unitario, el centro de masas, o centroide, será aquel punto (x_{pix}, y_{pix}) del objeto para el cual hay la misma masa arriba, abajo, a la izquierda y a la derecha. Y matemáticamente está definido como:

$$CM_x = NPix_x/NPix_{obj} \tag{4.2}$$

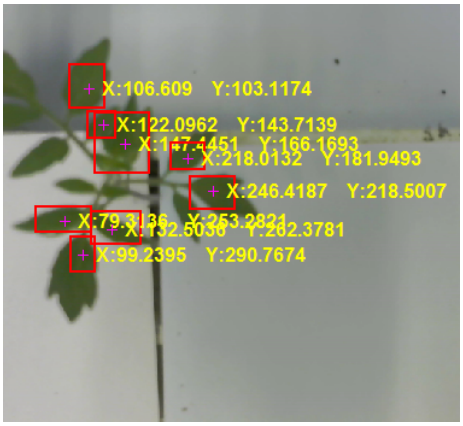
$$CM_y = NPix_y/NPix_{obj} \tag{4.3}$$

donde CM_x es el punto centroide en x de la imagen y CM_y es el punto centroide en y de la imagen, $NPix_x$ es el número total de pixeles en la referencia x, $NPix_y$ es el número total de pixeles en el eje y; mientras que $NPix_{obj}$ son los pixeles detectados por el objeto.

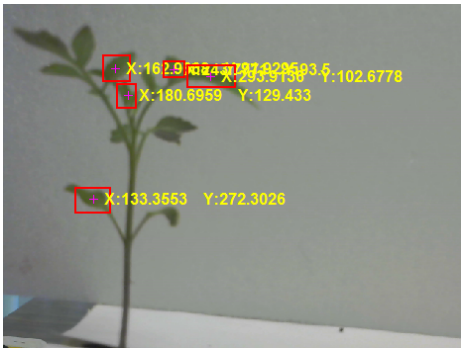
El detector toma una foto, mientras se manipula el plano de colores a tráves de unos slides (barras de deslizamiento), hasta llegar a la banda de color deseada, se filtra, figura (4.5). Se binariza, con el objetivo de separar las regiones u objetos de interés en una imagen del resto la escena es decir se discriminan todos los objetos que no estan dentro de esa banda de color para posteriormente invocar el comando *regionprops(bw, 'BoundingBox', 'Centroid');* asigna un centroide al objeto con esas características de colores[1].

[1] En `mathworks.com/matlabcentral/fileexchange/28757-tracking-red-color-objects`

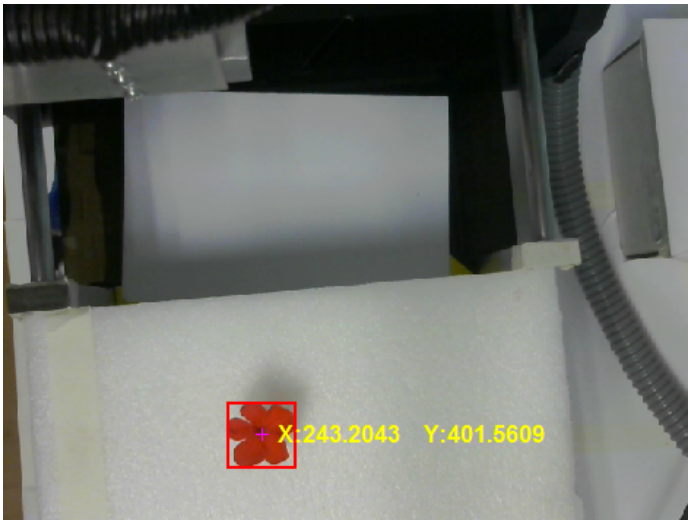
En la figura (4.6(c)), se aprecia que el algoritmo de Bhargav para un plántula de varias hojas contiguas, con una separación de $0.5cm$, es detectada como un solo un solo objeto, este una desventaja para detectar múltiples objetos en distancias cortas, otra desventaja que tiene el algoritmo de Bhargav es para cuerpos complejos los centroides tienden a no distinguirse por , figura (4.6(a)) y (4.6(b),) en cada parte del cuerpo principal del objeto que se esta analizando.



(a) ***Plantula vista vertical***



(b) ***Plantula vista horizontal***



(c) ***Plantula roja***

Figura 4.6: Respuesta del algoritmo de Bhargav a cuerpos complejos

En la siguiente figura (4.7), se muestra el funcionamiento del algoritmo de Bhargav, se puede apreciar como detecta tres hojas con una separación $8cm$ y le asigna un centroide diferente a cada objeto. Este sentido para este algoritmo solo funciona para objetos bien contrastados y definidos con una buena distancia; que para el caso de este problema es suficiente su funcionamiento.

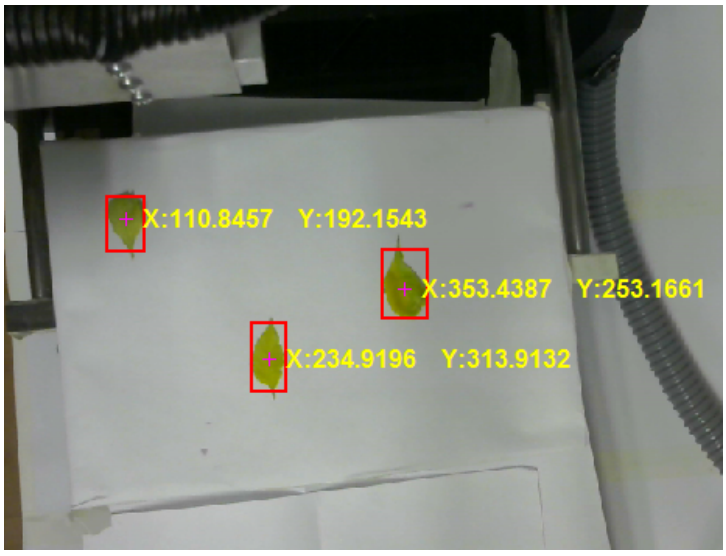


Figura 4.7: Detección del centroide en tres hojas.

4.3. Modelo del sistema de visión

Una cámara digital forma una representación discreta del plano del objeto que incide sobre un arreglo de fotosensores. El número de fotosensores en el arreglo *CCD* no coincide con el número de elementos en la imagen digital debido al procesamiento análogo y la digitalización. Así mismo, las coordenadas del centro del plano imagen tampoco concuerdan con las coordenadas del centro de la imagen digital. El modelo geométrico de la cámara se encuentra después de hacer una transformación de un sistema de coordenadas 2-D en el que se encuentra el objeto, al sistema de coordenadas del plano imagen, considerando la orientación y posición de la cámara con respecto del sistema en 3-D. [60, 61].

El sistema de coordenadas, (figura 4.8), de la escena O por lo regular no está alineado con

el sistema de coordenadas de la cámara que tiene su origen en el centro de proyección O_i, lo que requiere transformar al punto $P(x_m, y_m)$ en el sistema de coordenadas de la escena a el mismo punto P en el sistema de coordenadas $\acute{P}(u, v)$ de la cámara (parámetros extrínsecos).

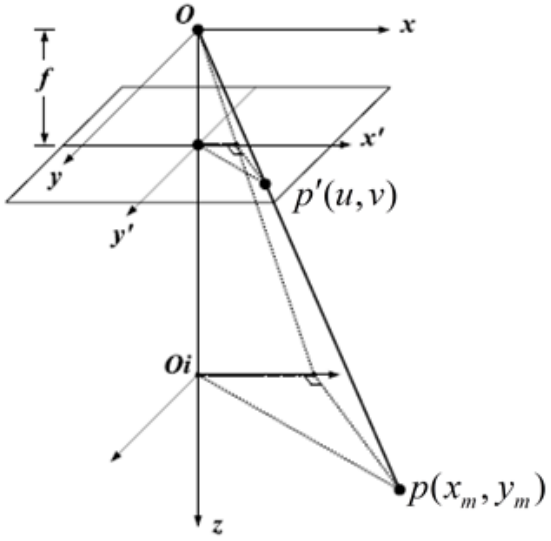


Figura 4.8: Proyección de perspectiva.

En este caso si la cámara se encuentra perpendicular a la escena que se está analizando la ecuación que relaciona pixeles a coordenadas de la escena [60] está definido como:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \frac{f}{z} \begin{bmatrix} \alpha_x & 0 \\ 0 & \alpha_y \end{bmatrix} \begin{bmatrix} x_m \\ y_m \end{bmatrix} + \begin{bmatrix} u_0 \\ v_0 \end{bmatrix} \tag{4.4}$$

donde $u, v \in \mathbb{R}^n$ son las coordenadas en pixeles del objeto localizado; $x, y \in \mathbb{R}^n$ son las coordenadas del espacio de trabajo de la escena; α_x, α_y es el factor de escala de la imagen; z es la distancia de la cámara a la escena y f es la distancia focal de la cámara. Sustituyendo los parámetros intrínsecos descritos en el cuadro (4.1) obtenemos el factor de escala.

$$\alpha_x = \frac{1}{p_x} \tag{4.5}$$

$$\alpha_y = \frac{1}{p_y} \tag{4.6}$$

Se puede apreciar que las ecuaciones (4.5) y (4.6) están en función del tamaño del pixel (p_x, p_y) en (x, y) y de acuerdo a la cuadro (4.1) de la cámara es del mismo tamaño el sensor lo que el factor de escala horizontal y vertical es el mismo $\alpha_x = \alpha(y) = \alpha$, por lo que la ecuación (4.4) se puede escribir como:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \frac{f\alpha}{z} \begin{bmatrix} x_m \\ y_m \end{bmatrix} \tag{4.7}$$

Sustituyendo los valores de las características de la cámara y tomando en cuenta que la cámara está a $39cm$ de la escena para que no colisione el robot con la estructura de la cámara, se tiene:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \frac{(4^{-3})(2.8^{-6})^{-1}}{0.39} \begin{bmatrix} x_m \\ y_m \end{bmatrix} \tag{4.8}$$

Por lo tanto, para un punto en la superficie del área de trabajo del robot existe un punto en el plano imagen el cual debe multiplicarse por un factor de conversión constante.

$$\begin{bmatrix} u \\ v \end{bmatrix} = K \begin{bmatrix} x_m \\ y_m \end{bmatrix} = 3663.00 \begin{bmatrix} x_m \\ y_m \end{bmatrix} \tag{4.9}$$

Este modelo relaciona las coordenadas del espacio real de la escena con las coordenadas de la imagen.

Para transformación las coordenadas espaciales de la imagen,(ecuación 4.9) a coordenadas cartesianas de trabajo del robot, (cuadro 3.1), es necesario tomar en cuenta el desplazamiento máximo de cada eje de robot y el tamaño máximo espacial de la imagen [64].
La transformación de coordenadas para el eje x correspondiente al eslabón 1 se realiza a través de:

$$x_{real} = \frac{(x_{image} - x_{minimage})}{(x_{maximage} - x_{minimage})} D_{maxX} + x_{mar} \tag{4.10}$$

donde $x_{real} \in \mathbb{R}^n$ es un vector que contiene las coordenadas centroides detectados para ser ejecutados en el robot; x_{image}, es el punto centroide de la imagen entregado en coordenadas cartesianas para la referencia en x entregado por el algoritmo de Bhargav; $x_{minimage}$ es la coordenada cartesiana mínima de la escena; $x_{maximage}$ es la coordenada cartesiana máxima de la escena; D_{maxX} es el desplazamiento total que realiza el robot con su efector final y x_{mar} es un margen de ajuste.

Para el eje y correspondiente al eslabón 2 se realiza a través de:

$$y_{real} = \frac{y_{image} - y_{minimage})}{(y_{maximage} - y_{minimage})} D_{maxY} + y_{mar} \tag{4.11}$$

$y_{real} \in \mathbb{R}^n$ es un vector que contendra todas las coordenadas centroides detectados para ser ejecutados en el robot; y_{image}, es el punto centroide de la imagen entregado en coordenadas cartesianas para la referencia en y entregado por el algoritmo de Bhargav; $y_{minimage}$ es la coordenada cartesiana mínima de la escena; $y_{maximage}$ es la coordenada cartesiana máxima de la escena; D_{maxY} es el desplazamiento total que realiza el robot con su efector final y y_{mar} es un margen de ajuste.

El margen de ajuste (x_{mar}, y_{mar}) dependerá del tamaño del fector final, cuadro (4.2).

Cuadro 4.2: Margen de ajuste del robot

Herramienta	x_{mar}	y_{mar}
Cortador Vertical	$5cm$	$7cm$
Sacabocados	$13cm$	$5cm$

La siguiente figura (4.9), se muestra el movimiento de la robot sobre las hojas, la figura (4.9 a) muestra la escena inicial de trabajo que mira la cámara, la figura (4.9 b) se muestra como el robot se ubica en la primera hoja en la coordenada $(11, 8)cm$, para posteriormente ubicarse en la segunda hoja con la coordenada $(15, 11)cm$, ejecutada la posición, pasa a la

siguiente ubicada en $(11, 13)$ correspondiente a la tercera hoja, posteriormente encuentra la cuarta hoja que se ubica en la $(10, 19)$.

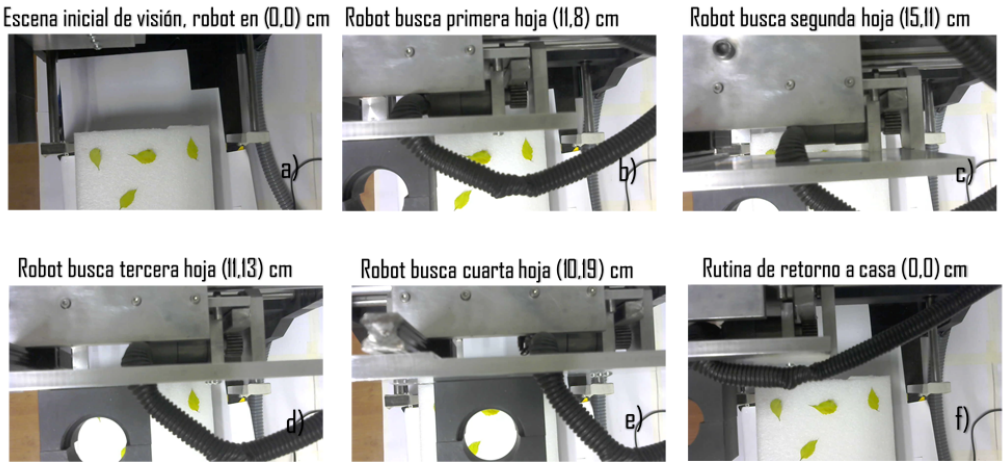


Figura 4.9: Movimiento del robot a través de la escena.

Capítulo 5

Sintonización del controlador

La sintonización de ganancias y parámetros en las estructuras de control para robots manipuladores, es un procedimiento totalmente empírico, es decir, hallar estos valores numéricos es a prueba y error para el ajuste del comportamiento de estos sistemas [11]. Para el caso de la estructura de control $PD+$, su sintonización se realiza a través de las relaciones matemáticas (5.1) cayendo en múltiples soluciones para ajustar las ganancias del controlador $PD+$.

$$\begin{aligned} K_p &\leq (0.8)(\tau_{max}) \\ K_v &\leq (0.25)(K_p \end{aligned} \tag{5.1}$$

Sin embargo, al considerar el modelo dinámico del sistema, en la estructura de control (5.1) aumenta el grado de dificultad en la sintonía de estas estructuras de control [16].

El ajuste de parámetros y ganancias de la estructura de control $PD+$ se realizó a través de los *métodos de optimización global*, entre los que destacan *los algoritmos evolutivos* y *bio-inspirados*. Estos permiten a través de una herramienta computacional, la optimización de un problema específico [36], destacando dos características importantes la intensificación y la diversificación [37]. La intensificación busca las mejores soluciones y selecciona los mejores candidatos actuales o prospectos. La diversificación se asegura de que el algo-

ritmo puede explorar el espacio de búsqueda de una manera más eficiente, a menudo aleatorio [36].

5.1. Métodos de optimización global

Los algoritmos evolutivos y bio-inspirados son dos métodos totalmente diferentes en esencia pero que tienen como fin la optimización global de problemas multimodales, es decir aquellos problemas donde se pueden hallar más de una solución [37].

Estos métodos de optimización global interactúan con el modelo dinámico del robot, ecuación (3.29), los límites máximos y mínimos de las ganancias, relación matemática (5.1). Trabajan con la *función objetivo* que será la ecuación que será optimizada dadas las limitaciones o restricciones determinadas y con variables que necesitan ser minimizadas o maximizadas usando técnicas de programación lineal o no lineales [42–46]. Todo ello, para obtener finalmente los parámetros ajustados, figura (5.1).

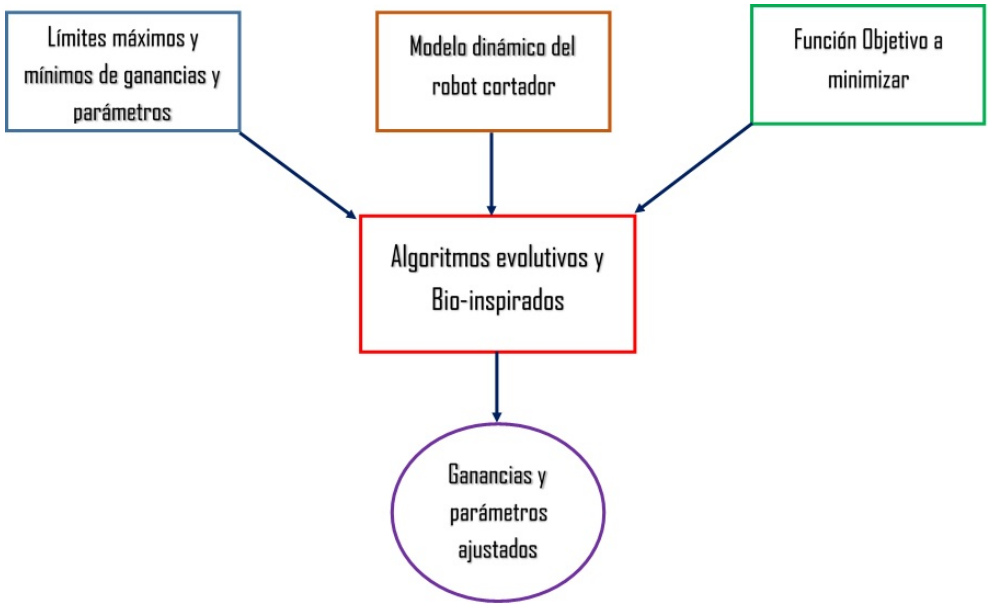


Figura 5.1: Diagrama de implementación de los métodos de optimización global.

Cada algoritmo de optimización global varia el criterio de búsqueda ya sea son *evolutivos* o *bio-inspirados*. Sin embargo computacionalmente tienen el mismo esquema de búsqueda

general, el cual parte en el establecimiento de los límites de búsqueda de los parámetros y estos límites varían de acuerdo al problema que se intenta optimizar. Por ejemplo en el caso de la sintonización de la ecuación (3.4) se toma ecuación (5.1) para hallar el límite máximo de las ganancias del controlador $PD+$, así como los parámetros medidos que se hallan en el cuadro (2.1). Una vez que se decide el método de búsqueda para generar nuevos parámetros ajustados esto se repite hasta obtener los parámetros finales ajustados, figura (5.2).

La función objetivo que se utiliza para minimizar el problema de búsqueda de ganancias es la norma $\mathcal{L}_2$, que más adelante se describirá.

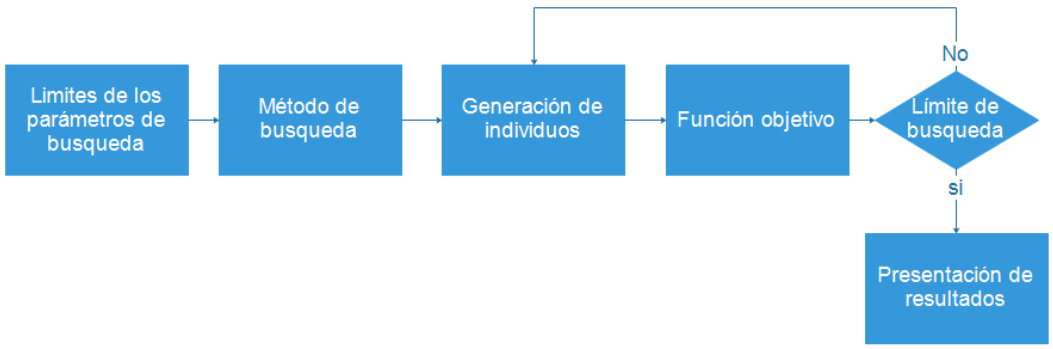


Figura 5.2: Diagrama de flujo del funcionamiento de los métodos de búsqueda global.

5.2. Método de Cuckoo Search

Este método de búsqueda global es del tipo *bio-inspirado* el cual se basan en emplear analogías con sistemas naturales, y tienen como característica ser auto adaptativo, auto-organizado y ser capaz de auto aprender.

Este método consiste en encontrar los valores de K_p y K_v de la ecuación (3.4) a través de un método de optimización global meta heurística, basado en el comportamiento de la ave Cuckoo. Este comportamiento se puede resumir en tres reglas idealizadas: 1) cada cuco

pone un huevo a la vez y lo deposita en un nido elegido al azar; 2) los mejores nidos con huevos de alta calidad serán transferidos a las próximas generaciones; 3) el número de nidos hospederos es fijo y el huevo puesto por un cuco es descubierto por el ave hospedera con una probabilidad $P_a \in [0, 1]$. En este caso el hospedero puede desechar el huevo o abandonar el nido y construir uno nuevo. Para simplificar, esta última suposición puede ser aproximada por la fracción de los nidos que son reemplazados (nuevas soluciones aleatorias) [36–40]. Es decir cada huevo en el nido representa una solución y cada huevo puesto por un cuco es una nueva solución al problema en cuestión, esto es:

$$x_i^{t+1} = x_i^t + \alpha \bigoplus Levy(\lambda) \tag{5.2}$$

donde α es el tamaño de paso y debe estar relacionado con las escalas del problema en cuestión. En la mayoría de los casos se puede utilizar $\alpha = 1$. La ecuación (5.2) es esencialmente la ecuación estocástica de recorrido aleatorio. En general, un recorrido aleatorio es una cadena de *Markov* cuya próxima posición depende de la ubicación actual (primer término en la ecuación) y la probabilidad de transición (segundo término). El símbolo $\bigoplus$ significa producto con respecto a las entradas. Este recorrido aleatorio a través del vuelo *Lévy* es más eficiente en la exploración del espacio de búsqueda. El vuelo Lévy representa un recorrido aleatorio y es la longitud de paso obtenida mediante una distribución de Lévy

$$\begin{aligned} &Levy \sim u = t^{-\lambda} \\ &-(1 < \lambda < 3) \end{aligned} \tag{5.3}$$

donde la ecuación (5.3) tiene media y varianza infinitas. En este caso, los pasos consecutivos de un cuco forman un proceso de caminata aleatoria que obedece a una distribución de ley de potencias con longitud de paso de doble cola.

Este algoritmo se utilizó para encontrar las ganancias K_p y K_v de la ecuación (3.4), donde el tamaño de la escala es de 9 parámetros de búsqueda, se utilizó el modelo dinámico descrito en el capítulo 3 y sirvió para sintonizar el controlador (3.2) y (3.4) ésta sintonización se encuentra descrita en el experimento 2 y 3 del capítulo de resultados.

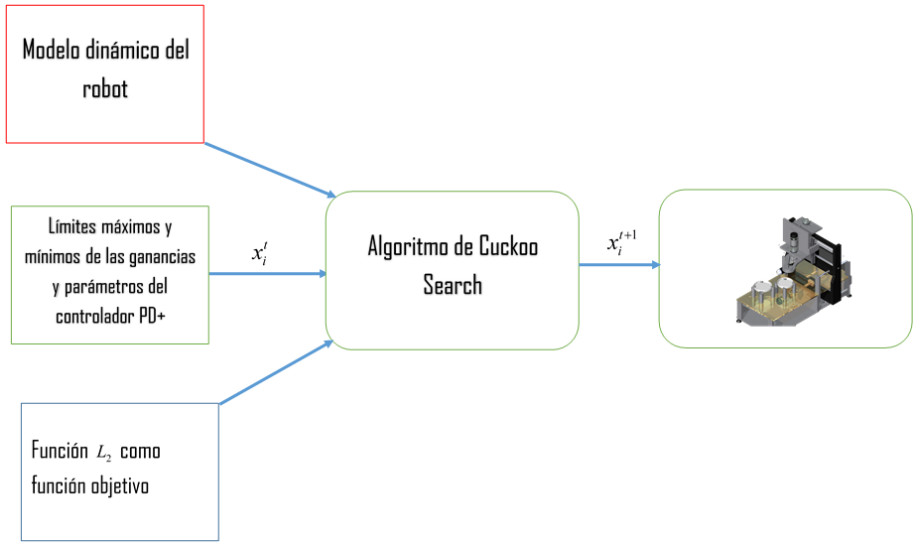


Figura 5.3: Elementos que conforma la búsqueda con algoritmo de Cuckoo.

En la figura (5.3) se puede apreciar que el vector de entrada x_i^t está constituido por los parámetros que se desean ajustar del robot a través del modelo dinámico del robot, para entregar un vector de salida x_i^{t+1} que representa los parámetros ajustados por el algoritmo de Cuckoo Search. El programa detallado se encuentra en el apéndice A sección del 1 al 3.

5.3. Algoritmos evolutivos

Los *algoritmos evolutivos* que son métodos de optimización y búsqueda de soluciones basados en los postulados de la evolución biológica. En ellos se mantiene un conjunto de entidades que representan posibles soluciones, las cuales se mezclan, y compiten entre sí, de tal manera que las más aptas son capaces de prevalecer a lo largo del tiempo, evolucionando hacia mejores soluciones [37, 38]. En este trabajo de tesis se utilizaron para hallar

que método de esta familia de algoritmos fue el mejor para sintonizar las ganancias K_p y K_v así como los parámetros que conforman el controlador $PD+$, ecuación (3.4), cuyo experimento se describe en el capítulo de resultados.

5.3.1. Implementación de la familia de los algoritmos evolutivos

En la figura (5.4) se puede apreciar que el vector de entrada $V_{i,G}$ que caracterisa a los algoritmos de evolución diferencial y son los parámetros que se desean ajustar del robot a través del modelo dinámico del mismo, para entregar un vector de salida $U_{i,G}$ que representa los parámetros ajustados por cada familia de estos algoritmos evolutivos y que son implementados en el robot. El programa detallado se encuentra en el apéndice A sección del 1 al 2 y del 4 al 7, y la utilización se halla en el capítulo de resultados experimento 4.

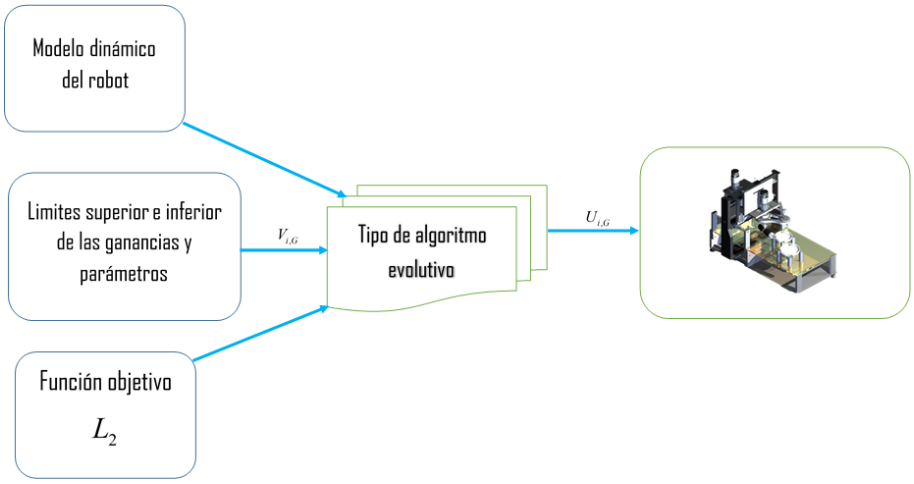


Figura 5.4: Elementos que conforma la búsqueda la familia de algoritmos evolutivos.

A continuación se describe cada algoritmo de esta familia.

5.3.2. Método de Evolución Diferencial DE clásico

Este algoritmo considerado como el algoritmo de evolución diferencial estándar desde su presentación en 1997 por Storn y Price [47–50]. Este método de búsqueda utiliza una población de vectores. Como regla, se asume una distribución uniforme para las decisiones

aleatorias a tomar. Por lo que tiene la característica es de ser un operador simultáneo de cruzamiento y mutación. El esquema de este algoritmo implica un vector $\vec{v}_i$ de prueba tal que:

$$\begin{aligned} &\vec{v}_{i,G+1} = x_{r1,G} + F(x_{r2,G} - x_{r3,G}) \\ &i = 1, 2, \ldots NP \end{aligned} \tag{5.4}$$

La función de la ecuación (5.4) es generar un nuevo vector de parámetros $\vec{v}_i$ a través de la suma de la diferencia de pesos $(x_{r2,G} - x_{r3,G})$ entre dos vectores miembros de la población NP, con un tercer vector miembro $x_{r1,G}$ o miembro principal, a esto se le llama mutación. $F \in [0, 1]$ es un factor mutación real y constante que controla la amplificación en la variación diferencial.

El cruzamiento combina el vector de poblaciones mutadas $v_{ji,G+1}$ con el vector de poblaciones elegidas para generar un nuevo vector de poblaciones $u_{ji,G+1}$ de tal manera que la ecuación (5.5) indica que si la aptitud del vector resultante o población mutada es menor que del miembro de la población elegida entonces el nuevo vector reemplaza al vector con el cual fue comparado, de lo contrario mantiene la población elegida, por cada generación G :

$$u_{ji,G+1} = \begin{cases} v_{ji,G+1} \\ if(randb(j) \leq CR) or j = rnbr(i) \\ \\ x_{ji,G} \\ if(randb(j) > CR) or j \neq rnbr(i) \end{cases} \tag{5.5}$$

Donde $randb(j) \in [0, 1]$, es la j-esima evaluación de números aleatorios generados uniformemente, $randb(j) \in [1, 2 \ldots D]$ es un índice elegido al azar, $CR \in [0, 1]$ es la probabilidad de cruzamiento, un parámetro que aumenta la diversidad de los individuos de la población.

5.3.3. Método de SaDE

El control de parámetros en Evolución Diferencial (DE) y la elección adecuada de la estrategia de aprendizaje utilizada en el operador de mutación, son altamente dependientes del problema en consideración. Para esta tarea específica, generalmente se necesita invertir una enorme cantidad de tiempo para probar varias estrategias de aprendizaje y calibrar los parámetros correspondientes. Este dilema motivó a *Quin* y *Suganthan* a desarrollar un algoritmo auto adaptativo basado en DE [50–52].

SaDE, está basada en *learning period* o como denominaremos *Periodos de Aprendizaje*. Esta técnica está enfocada en la auto adaptación de la estrategia de aprendizaje, sin embargo también se adaptan los parámetros de recombinación y mutación. Los autores de la técnica definen como parámetros críticos de la ED a: CR, F y NP (operador de cruza, operador de mutación y tamaño de población sus trabajos se centran en adaptar CR y F.
Las Estrategias de aprendizaje utilizadas en SaDE son:

Estrategia 1. $DE/ran/1/bin$.

$$\begin{aligned} &\vec{v}_{i,G+1} = x_{r1,G} + F(x_{r2,G} - x_{r3,G}) \\ &i = 1, 2, \ldots NP \end{aligned} \tag{5.6}$$

Estrategia 2. $DE/current - to - best/2/bin$.

$$\begin{aligned} &\vec{v}_{i,G+1} = x_{i,G} + F(x_{best,G} - x_{i,G}) + F(x_{r1,G} - x_{r2,G}) \\ &i = 1, 2, \ldots NP \end{aligned} \tag{5.7}$$

Estrategia 3. $DE/rand/2/bin$.

$$\begin{aligned} &\vec{v}_{i,G+1} = x_{r1,G} + F(x_{r2,G} - x_{r3,G}) + F(x_{r4,G} - x_{r5,G}) \\ &i = 1, 2, \ldots NP \end{aligned} \tag{5.8}$$

Estrategia 4. $DE/current - to - rand/1$.

$$\begin{aligned} U_{i,G} &= x_{r1,G} + K(x_{r3,G} - x_{i,G}) + F(x_{r1,G} - x_{r2,G}) \\ i &= 1, 2, \ldots NP \end{aligned} \tag{5.9}$$

K, es el coeficiente de combinación en un rango de $[0.5, 1.5]$.

La probabilidad de aplicar cada estrategia de aprendizaje en la población inicial se fija en $p_1 = 0.5$ y $p_2 = 1 - p_1$ para la primera versión y en la segunda se extiende esta probabilidad a p_i donde $i = 1, 2, 3, 4$ iniciando $p_i = 0.25$.

5.3.4. Método de JDE

Este algoritmo fue desarrollado por Brest y se basa en el método de búsqueda clásica DE1 ecuaciones (5.4) y (5.5). Esta estrategia es la más utilizada en la práctica y muestra resultados competitivos frente al SaDE [53, 54]. El mecanismo auto adaptativo de control de parámetros es utilizado para ajustar los valores de los parámetros F y CR durante la ejecución del algoritmo, donde el factor de mutación y cruzamiento esta dado como:

$$F_{i,G+1} = \begin{cases} 0.1 + 0.9 rand_1 \\ if(rand_2 \leq \tau_1) \\ F_{i,G} \\ otherwise \end{cases} \tag{5.10}$$

$$CR_{i,G+1} = \begin{cases} rand_3 \\ if(rand_3 \leq \tau_2) \\ CR_{i,G} \\ otherwise \end{cases} \tag{5.11}$$

Donde $rand_j$, $j = 1, 2, 3, 4$ son valores aleatorios distribuidos entre $[0, 1]$ y $\tau_1 = \tau_2 = 0.1$ representa la probabilidad de ajuste en los parámetros de control del algoritmo.

Esta técnica pretende la auto adaptación de los parámetros de recombinación y mutación, sin embargo no utiliza los operadores de recombinación y mutación propios de la ED si no

que propone operadores adicionales para el cálculo del control de parámetros del nuevo vector padre, con una fuerte carga de elementos aleatorios.

5.3.5. Método de JADE

Esta variante de DE fue desarrollado por Zang y Sanderson y tiene la característica de auto adaptar los parámetros de control y hacer uso de un archivo en el que se almacenan individuos que son reemplazados en la selección. Estos individuos se utilizan como candidatos a individuos aleatorios en el operador de mutación, introduciendo mayor diversidad en generación del vector mutado ha sido extendido por varios autores con la intención de mejorar su robustez, esto es, su capacidad de resolver problemas con diferentes características [56, 57]. Dentro de estas extensiones están el auto adaptación de parámetros y estrategias evolutivas que influyen en el comportamiento del algoritmo. El vector de prueba o vector de mutación esta dado como:

$$\begin{aligned} &\vec{v}_{i,G+1} = x_{r0,G} + F_i(x^p_{best,G} - x_{i,G}) + F_i(x_{r1,G} - \tilde{x}_{r2,G}) \\ &i = 1, 2, \ldots NP \end{aligned} \tag{5.12}$$

$x_{r0,G}$ y $x_{r1,G}$ son dos vectores aleatoriamente seleccionados de una población P , $\tilde{x}_{r2,G}$ es un vector elegido de $P \bigcup A$. Donde A es la población de individuos almacenados que son reemplazados en la selección. El operador de cruzamiento es igual al DE1 ecuación (5.5). El operador de selección está definido como:

$$x_{i,G+1} = \begin{cases} U_{i,G}, f(U_{i,G}) < f(x_{i,G}) or f(U_{i,G} = f(x_{best,G}) \\ \\ x_{i,G} \\ otherwise \end{cases} \tag{5.13}$$

5.4. Función objetivo para sintonización del controlador

La norma $\mathcal{L}_2$ es el conjunto de todas las funciones continuas $f : \mathbb{R}_+ \to \mathbb{R}^n$ tal que:

$$\begin{aligned} \|f\|_{\mathcal{L}_2} &= \sqrt{\int_0^\infty f^T(t) f(t) dt} \\ \|f\|_{\mathcal{L}_2} &= \sqrt{\int_0^\infty \|f(t)\| dt} < \infty \end{aligned} \tag{5.14}$$

entonces la integral del cuadrado de la norma euclidiana es medible y está acotada superiormente. La norma $\|f\|_{\mathcal{L}_2}$ es ampliamente empleada para evaluar el desempeño de algoritmos de control de robots manipuladores. La interpretación de la norma $\|f\|_{\mathcal{L}_2}$ y el desempeño del esquema de control a evaluar es inversamente proporcional al desempeño [16].

Esta expresión matemática se le conoce normalmente *índice de desempeño*, que coloquialmente se refiere a la exactitud que debe tener la respuesta de un robot. Un alto desempeño del esquema de control significa que la exactitud en el error de posicionamiento debe ser idealmente cero, entonces la respuesta del robot es de calidad, es decir, presenta una curva con perfil suave, picos o sobre impulsos atenuados, rápido estado transitorio, no hay vibración mecánica, ni oscilaciones [17].

El índice de desempeño de un algoritmo de control de un robot manipulador se define como:

$$\mathcal{L}_2 = \sqrt{\frac{1}{T} \int_0^T \|\tilde{q}(t)\|^2 dt} \tag{5.15}$$

donde T representa el tiempo de simulación o de experimentación.

El mejor comportamiento de desempeño del controlador corresponde a la norma $\mathcal{L}_2$ más pequeña. Un valor alto en el índice de desempeño representa un bajo rendimiento del robot manipulador.

Por la naturaleza de la ecuación (5.15), cumple el objetivo de función objetivo a minimizar para lograr un ajuste global de ganancias y parámetros del controlador $PD+$, ecuación (3.4).

Capítulo 6

Resultados experimentales

En este capítulo se mostrara seis experimentos que permitieron validar la matemática desarrollada y verificar el funcionamiento del robot, para finalmente desarrollar un corte circular sobre una hoja.

6.1. Experimento 1: Simulación del modelo dinámico

Sea el modelo dinámico del sistema cartesiano:

$$\begin{bmatrix} \tau_1 \\ \tau_2 \\ \tau_3 \end{bmatrix} = \begin{bmatrix} m_1+m_2+m_3 & 0 & 0 \\ 0 & m_2+m_3 & 0 \\ 0 & 0 & m_3 \end{bmatrix} \begin{bmatrix} \ddot{q}_1 \\ \ddot{q}_2 \\ \ddot{q}_3 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ m_3 \end{bmatrix} g \tag{6.1}$$

y el controlador PD+ como regulador de posición, definido como:

$$\tau_{pd} = K_p \tanh(\tilde{q}) - K_v \tanh(\dot{q}) + g(q) \tag{6.2}$$

Para verificar el comportmiento del modelo dinámico se establecio un *punto arbitrario* para los tres eslabones siendo $q_{d1} = 2100°$; $q_{d2} = 700°$ y $q_{d3} = 210°$. Posteriormente se implementaron los datos del cuadro (2.1) para alimentar el modelo dinámico; los valores de ganancias utilizadas son los presentadas en los cuadros (6.1) y (6.2) para sintonización

del controlador del robot cartesiano (robot cortador)

Cuadro 6.1: Ganancias utilizadas para el modelo dinámico del robot

Parámetro	Ganancias por huristica	Ganancias de Cuckoo	Ganancias de DE Clasico
K_{p1}	$1.15\frac{Nm}{rad}$	$0.847\frac{NmS}{rad}$	$0.798\frac{Nm}{rad}$
K_{p2}	$1.35\frac{Nm}{rad}$	$0.807\frac{NmS}{rad}$	$1.033\frac{Nm}{rad}$
K_{p3}	$1.2544\frac{Nm}{rad}$	$0.940\frac{NmS}{rad}$	$0.975\frac{Nm}{rad}$
K_{v1}	$0.25\frac{NmS}{rad}$	$0.258\frac{NmS}{rad}$	$0.310\frac{Nm}{rad}$
K_{v2}	$0.35\frac{NmS}{rad}$	$0.312\frac{NmS}{rad}$	$0.309\frac{Nm}{rad}$
K_{v3}	$0.3136\frac{NmS}{rad}$	$0.186\frac{NmS}{rad}$	$0.148\frac{Nm}{rad}$

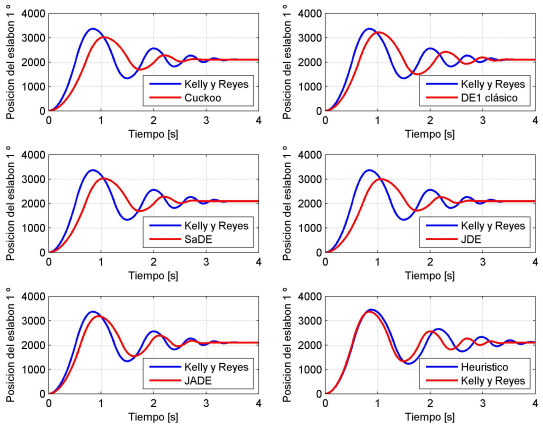
Cuadro 6.2: Segunda cuadro de ganancias utilizadas para modelo dinámico del robot

Parámetro	Ganancias JDE	Ganancias SaDE	JADE
K_{p1}	$0.780\frac{Nm}{rad}$	$0.972\frac{Nm}{rad}$	$0.788\frac{Nm}{rad}$
K_{p2}	$0.776\frac{Nm}{rad}$	$0.828\frac{Nm}{rad}$	$0.946\frac{Nm}{rad}$
K_{p3}	$0.582\frac{Nm}{rad}$	$0.791\frac{Nm}{rad}$	$0.825\frac{Nm}{rad}$
K_{v1}	$0.312\frac{NmS}{rad}$	$0.312\frac{Nm}{rad}$	$0.308\frac{Nm}{rad}$
K_{v2}	$0.295\frac{NmS}{rad}$	$0.309\frac{Nm}{rad}$	$0.281\frac{Nm}{rad}$
K_{v3}	$0.297\frac{NmS}{rad}$	$0.291\frac{Nm}{rad}$	$0.192\frac{Nm}{rad}$

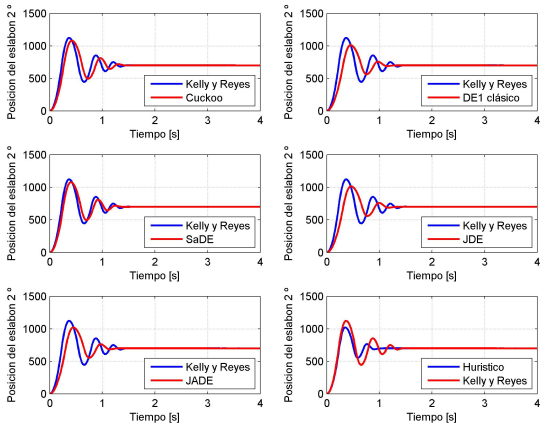
Las ecuaciones (6.1) y (6.2) se combinan para obtener la ecuación de lazo cerrado (6.3), que define el problema de control de posición:

$$\frac{d}{dt}\begin{bmatrix}\tilde{q}\\ \dot{q}\end{bmatrix} = \begin{bmatrix}-\dot{q}\\ M(q)^{-1}[K_p \tanh(\tilde{q}) - K_v \tanh(\dot{q})]\end{bmatrix} \tag{6.3}$$

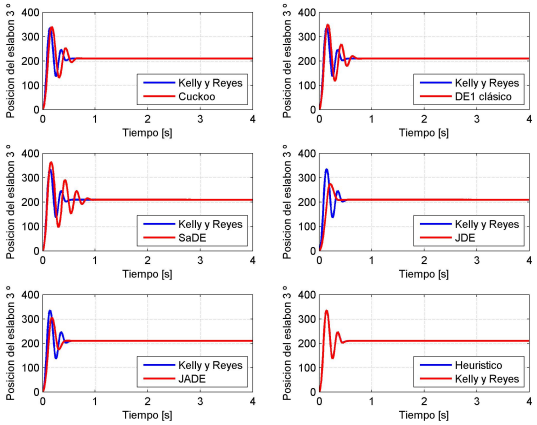
Posteriormente se programa en *Matlab*® la ecuación (6.3) y se resuelve a través del metodo número de integración Runge-Kutta 4. En la figura (6.1), se puede observar el comportamiento del *modelo dinámico* con los diferentes métodos de sintonización que se abordaran más adelante en este capítulo con detalle. Podemos apreciar que para el eslabón 1, (Fig. 6.1(a)), la simulación arroja que los métodos de *Cucko*, $DE1$ y *JDE* se estabilizan el sistema a los 2.5 segundos; En esta Prueba los peores métodos de sintonía fue el límite propuesto por Reyes y Kelly y el heurístico.



(a) ***Comportamiento del eslabón 1 a diferentes métodos de sintonización***



(b) ***Comportamiento del eslabón 2 a diferentes métodos de sintonización***



(c) ***Comportamiento del eslabón 3 a diferentes métodos de sintonización***

Figura 6.1: Comportamiento modelo dinámico del robot a diferentes métodos de sintonización

Para el eslabón 2, (figura 6.1(b)), se puede apreciar que el mejor tiempo de asentamiento fue para *JADE* con 1s y con un sobre impulso de 300° sobre la referencia deseada, seguido de *DE*1 y *JDE* con 1.1s y con sobre impulso de 307° y 305°. Finalmente el metodo *JDE* sobresale de los demas métodos al estabilizar el eslabón 3, (Fig. 6.1(c)), en 0.3s con un sobre impulso máximo de 60°.

6.2. Experimento 2: Implementación del algoritmo de Cuckoo Search

El siguiente experimento consistió en evaluar el funcionamiento del algoritmo de Cuckoo Seacrh, para ello se fijó un punto para las coordenadas (x, y),en este caso al centro de la caja de Petri correspondiente en la región media del espacio de trabajo del robot y z , para el primer experimento que consiste en evaluar el controlador *PD+ clásico* se fija a un punto arbitrario, denominado punto de prueba A. En el segundo experimento se fija dos puntos de prueba para evaluar la presición y el controlador *PD+ de acciones acotadas*, figura (6.2).

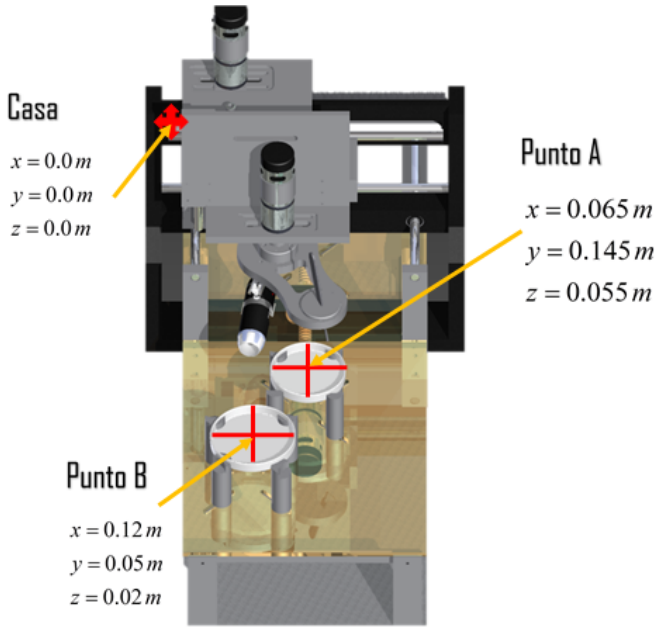


Figura 6.2: Puntos de prueba para la sintonía del algoritmo de Cuckoo Search.

Posteriormente se programa los elementos que se presentan en el esquema (5.3), para minimizar el índice de desempeño de tal manera que $\lim_{t\to\infty} \mathcal{L}_2 \to 0$ para obtener la media de nueve parámetros que conforma la estructura de control (3.2) y (3.4).

Para el funcionamiento del algoritmo de *Cuckoo Search* se consideró 20 generaciones y 25 nidos o posibles soluciones para cada generación, haciendo 10 ciclos de búsqueda con una tasa de descubrimiento $P_a = 0.25$. Este algoritmo fue programado en una computadora Toshiba AMD 8 de 12 GHz con 12 Gb en RAM.

6.2.1. Sintonización del controlador PD+ clásico

Este controlador originalmente fue propuesta en 1987 por Paden y Panja, ecuación (3.2), este controlador tiene la característica de no tener acotada su salida de control es decir su curva de deslizamiento crece en forma de recta para posiciones negativas y positivas, figura (6.3), por lo que sus ganancias al ser calculadas son al orden del 10 % del torque máximo del motor debido que este tipo de controlador tiende a saturarse de manera rapida, es decir, sobre pasa los límites reales de torque del motor.

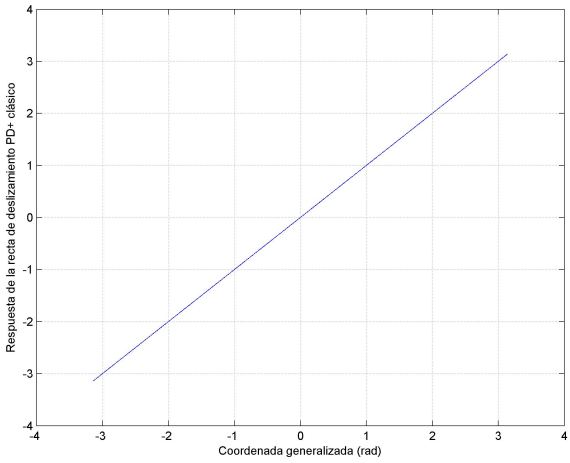


Figura 6.3: Curva de deslizamiento del PD+ clásico.

Los límites de las ganancias son obtenidas a través de la relación matemática (5.1), para obtener la siguiente cuadro (6.3).

Cuadro 6.3: Límites de los parámetros

Parámetro	Limite inferior	Limite Superior
K_{p1}	$0N$	$1.25N$
K_{p2}	$0N$	$1.25N$
K_{p3}	$0N$	$1.25N$
K_{v1}	$0Ns^{-1}$	$1.25Ns^{-1}$
K_{v2}	$0Ns^{-1}$	$1.25Ns^{-1}$
K_{v3}	$0Ns^{-1}$	$1.25Ns^{-1}$

Con estos elementos se obtuvo los mejores valores, cuadro (6.4), (ganancias y masas) para cada ciclo, entregados por el algoritmo de *Cuckoo Search*, con un valor de $\mathcal{L}_2 = 0.0949 \pm 0.0149$ este índice nos indica que, entre más cercano a cero se traduce a un mejor rendimiento en el esquemas de control (3.2).

Cuadro 6.4: Valores de ganancias optimizados

Parámetro	Media $\pm\sigma$
K_{p1}	$0.0338 \pm 0.3163N$
K_{p2}	$0.0449 \pm 0.1546N$
K_{p3}	$0.0744 \pm 0.2361N$
K_{v1}	$0.0105 \pm 0.1286Ns^{-1}$
K_{v2}	$0.0215 \pm 0.3136Ns^{-1}$
K_{v3}	$0.0288 \pm 0.1416Ns^{-1}$

Estos valores optimizados fueron implementados en la interfaz en la sección del controlador del robot cortador, obteniendo la siguiente respuesta:

Como se puede observar en la figura (6.4),la sintonización por el algoritmo de Cuckoo mejora la respuesta del robot manipulador como son: atenuando los sobre impulsos, y reduciendo el tiempo de establecimiento fue mucho bajo. El tiempo de establecimiento para el eslabón 1 fue de $t_s = 1.25s$; para el eslabón 2, $t = 0.49s$; y para el eslabón 3, $t = 0.33s$. También mejora en el posicionamiento del efector final, gráfica en negro, corrigiendo la sobre estimación de la sintonización, grafica en rojo, de *Kelly* y *Reyes*.

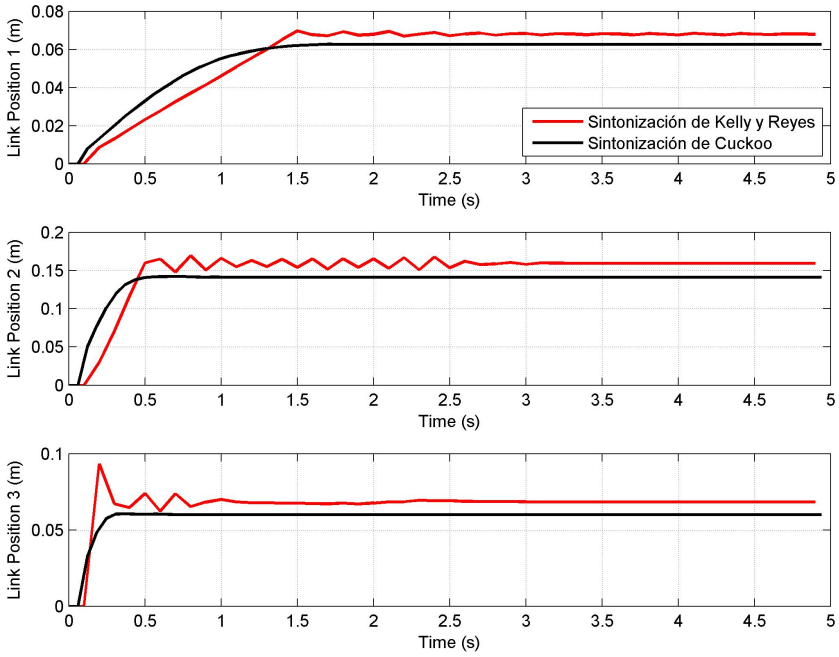


Figura 6.4: Respuesta de la sintonización del robot por el método de Kelly y Reyes vs Sintonización de Cuckoo.

Esto se ve evidente en el siguiente cuadro (6.5). La diferencia entre el eslabón 1 y la deseada, es decir el error de posición, a través de la sintonización de Cuckoo es de $2mm$, para el eslabón 2, es de $5mm$ y del eslabón 3 es de $5mm$.

Cuadro 6.5: Posiciones adquiridas por ambas sintonizaciones

Eslabón	Kelly y Reyes	Cuckoo	Pos. deseada
1	$0.070m$	$0.063m$	$0.065m$
2	$0.15m$	$0.14m$	$0.145m$
3	$0.068m$	$0.060m$	$0.055m$

6.2.2. Sintonización del PD+ de acciones acotadas

Este controlador fue propuesto originalmente por *Cai* y *Song* y reformulado más tarde por *Kelly, R.*, *Santibañez, V.* y *Reyes, F.*, este controlador tiene la característica de tener en su curva de deslizamiento un acotamiento entre 1 y -1, figura (6.5), para conservar dentro

de los límites permitidos el torque máximo del motor, evitando así un daño.

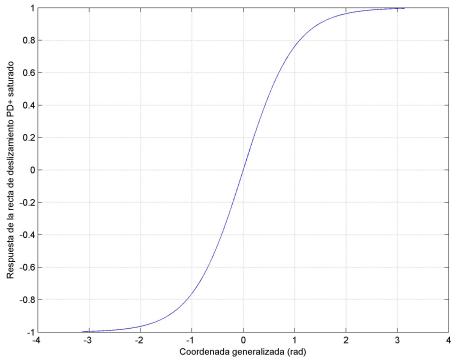


Figura 6.5: Curva de deslizamiento del PD+ saturado.

Para evaluar la precisión del robot cortador sintonizando la máquina con la sintonización máxima tolerada para los motores y la sintonía de *Cuckoo Search*. Para determinar el desempeño del controlador y la precisión mecánica a través de las ecuaciones (6.4), (6.5) y (6.6). Para ello, se fijaron dos puntos de prueba lo cuales son: el punto central de la caja de Petri en la región media del espacio de trabajo del robot como punto A; y el punto B, ubicada en el centro de la caja de Petri en región más alejada del espacio de trabajo del robot, figura (6.2).

$$\bar{\bar{x}} = S_x \bar{\bar{q}}_1 \tag{6.4}$$

$$\bar{\bar{y}} = S_y \bar{\bar{q}}_2 \tag{6.5}$$

$$\bar{\bar{z}} = S_z \bar{\bar{q}}_3 \tag{6.6}$$

Esto quiere decir, la precisión de los elementos mecánicos utilizados para cada eje son las siguientes: de S_y y S_z tiene un valor de 2.5 mm, mientras en S_x es de 1.15 mm.

Para utilizar la sintonización de *Cuckoo Seacrh*, fijamos un punto de *prueba articular arbitrario* de $q_d = 360^\circ$ para los tres motores y minimizando la función objetivo (5.15) a cero

para obtener la media de las ganancias optimizadas.

A continuación, en el siguiente cuadro (6.6) se muestran las ganancias máximas culculadas a tráves de la ley de sintonización de *Kelly* y *Reyes* además las de Cuckoo:

Cuadro 6.6: Ganancias experimentales implementadas en el robot

Parámetro	Ganancias Kelly y Reyes	Ganancias de Cuckoo
K_{p1}	$1.254N$	$0.847N$
K_{p2}	$1.254N$	$0.807N$
K_{p3}	$1.254N$	$0.940N$
K_{v1}	$0.313Ns^{-1}$	$0.258Ns^{-1}$
K_{v2}	$0.313Ns^{-1}$	$0.312Ns^{-1}$
K_{v3}	$0.313Ns^{-1}$	$0.186Ns^{-1}$

La figura (6.6), nos indica las posiciones promedio en las que llegaron los tres eslabones para ambas posiciones con los dos diferentes tipos de sintonización.

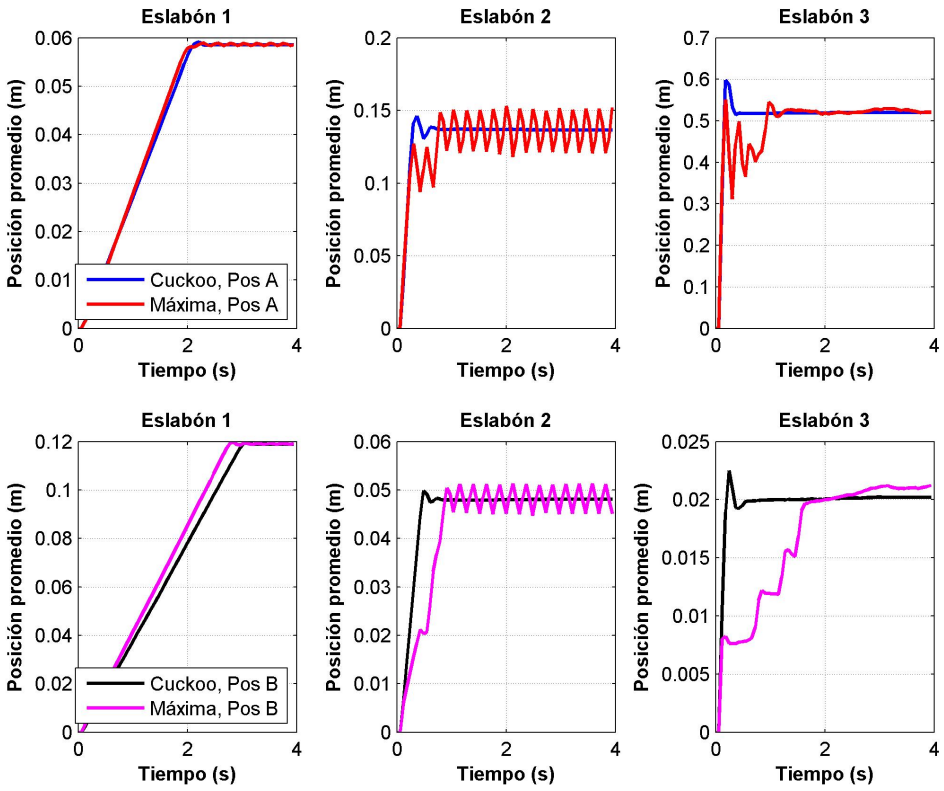


Figura 6.6: Comportamiento de los puntos de prueba.

Se observa que para el eslabón 1 no hay mucha diferencia en trabajar con la sintonización máxima y con la de *Cuckoo Search*, sin embargo para el eslabón 2 no se estabiliza, trabajando con la capacidad máxima del motor, para ambos casos. Mientras con el ajuste de *Cuckoo Search* se logra una estabilización por debajo de los 0.75 segundos. Para el eslabón 3 los máximos sobre impulsos trabajando a la máxima capacidad son altos, y para el caso del punto B su tiempo de estabilización fue mayor, además de sufrir un desplazamiento erroneo evidenciándose en un barrido en la cremallera-piñon evitando que se posicionara adecuadamente el eslabón 3 y reflejandose en el índice de desempeño. Sin embargo, ajustando con el algoritmo de *Cuckoo Search* las ganancias del controlador, la respuesta en estado transitorio mejora disminuyendo los sobre impulsos y el tiempo de asentamiento. El siguiente cuadro (6.7), se muestra la precisión mecánica para los dos puntos, se puede apreciar que la precisión mecánica se eleva con la sintonía de *Cuckoo Search* ya que en el punto A se obtuvo un promedio de $1.7mm$ para el punto B de prueba se obtuvo un promedio de $2.2mm$, esto revela que trabajar en la región máxima de los motores hace decaer el desempeño del robot.

Cuadro 6.7: Presición por punto

Iteración	Punto A por K y R	Punto A por CS	Punto B por K y R	Punto B por CS
1	$0.0037m$	$0.0019m$	$0.0037m$	$0.0021m$
2	$0.0035m$	$0.0019m$	$0.0014m$	$0.0020m$
3	$0.0037m$	$0.0018m$	$0.0051m$	$0.0026m$
4	$0.0042m$	$0.0015m$	$0.0039m$	$0.0021m$
5	$0.0036m$	$0.0014m$	$0.0036m$	$0.0023m$
$E\bar{M}$	$0.0044m$	$0.0017m$	$0.0035m$	$0.0022m$

6.3. Experimento 3: Búsqueda del mejor método evolutivo para sintonización

El experimento consistió en la evaluación del desempeño de cada algoritmo de evolución diferencial implementando los parámetros obtenidos por cada método en el robot de cor-

te, con la finalidad de saber cual es el mejor método evolutivo de sintonización. El ajuste se realiza fijando el mismo un punto de prueba de la figura (6.7).

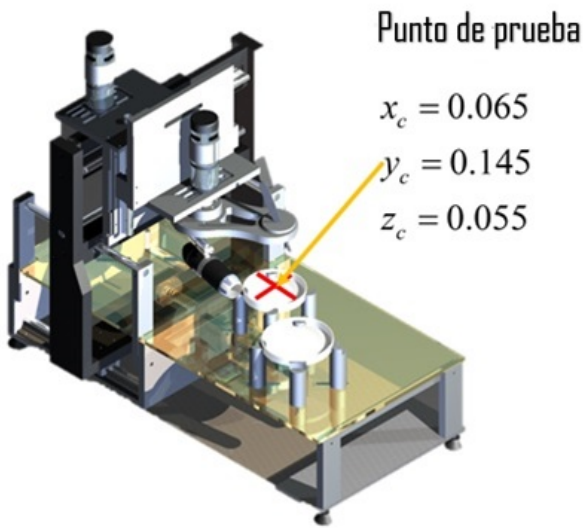


Figura 6.7: Punto de prueba para sintonía de la familia DE.

Se programaron y ejecutaron los algoritmos DE1, SaDE, JDE y JADE, para minimizar la ecuación (5.15) llevando a cabo un total de 10 optimizaciones para obtener el promedio de los parámetros ajustados presentados en el cuadro (6.8).

En el cuadro (6.9) se puede apreciar que el promedio de los errores de posición llamado índice de desempeño en el robot cortador, con el método *DE1* es mucho menor que los demás métodos de ajuste de parámetros. Como se puede apreciar $DE1$ presenta un valor de índice promedio de $0.23rad$. Superando claramente a los 4 métodos de ajuste de parámetros PD+.

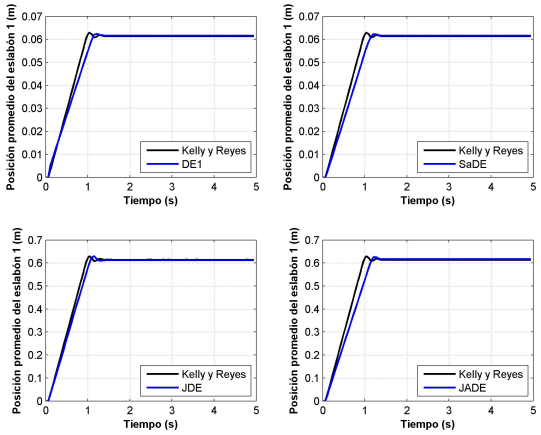
Estas ganancias se implementaron en el robot experimental y se repitió 5 veces la aproximación al punto de prueba. Esto permitió obtener el índice de desempeño experimental promedio del controlador ajustado por cada método.

Cuadro 6.8: Ganancias obtenidas por método

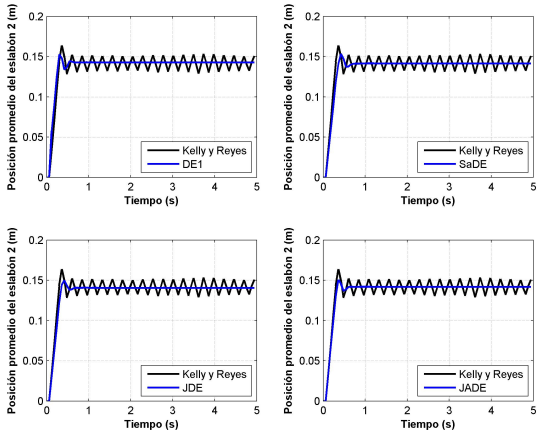
Parámetro	Clásico	DE1	SaDe	JDE	JADE
K_{p1}	$1.254N$	$0.798 \pm 0.0923N$	$0.780 \pm 0.318N$	$0.972 \pm 0.401N$	$0.788 \pm 0.060N$
K_{p2}	$1.254N$	$1.033 \pm 0.1801N$	$0.776 \pm 0.331N$	$0.828 \pm 0.438N$	$0.946 \pm 0.211N$
K_{p3}	$1.254N$	$0.975 \pm 0.1711N$	$0.582 \pm 0.120N$	$0.791 \pm 0.262N$	$0.825 \pm 0.245N$
K_{v1}	$0.313Ns^{-1}$	$0.310 \pm 0.007Ns^{-1}$	$0.312 \pm 0.041Ns^{-1}$	$0.312 \pm 0.009Ns^{-1}$	$0.308 \pm 0.001Ns^{-1}$
K_{v2}	$0.313Ns^{-1}$	$0.309 \pm 0.035Ns^{-1}$	$0.295 \pm 0.390Ns^{-1}$	$0.309 \pm 0.011Ns^{-1}$	$0.281 \pm 0.038Ns^{-1}$
K_{v3}	$0.313Ns^{-1}$	$0.148 \pm 0.057Ns^{-1}$	$0.297 \pm 0.031Ns^{-1}$	$0.291 \pm 0.067Ns^{-1}$	$0.192 \pm 0.074Ns^{-1}$

Cuadro 6.9: Promedio de errores e índice desempeño

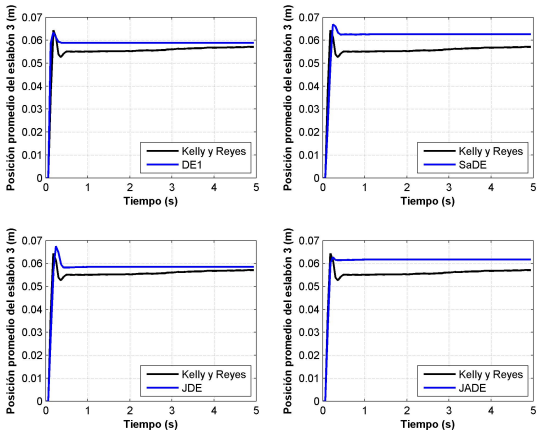
Metodo	$\bar{\tilde{q}_1}rad$	$\bar{\tilde{q}_2}rad$	$\bar{\tilde{q}_3}rad$	$\bar{\mathcal{L}}_2 rad$
Clasico	0.08	0.65	0.14	0.38
DE1	0.09	0.11	0.27	0.23
SaDE	0.13	0.17	0.59	0.32
JDE	0.18	0.17	0.36	0.29
JADE	0.21	0.17	0.51	0.32



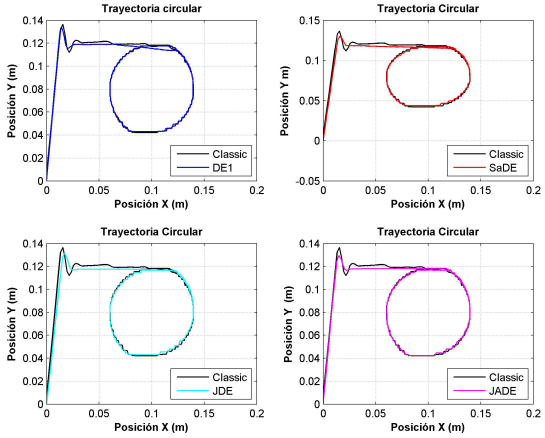
(a) ***Comportamiento del eslabón 1 a diferentes métodos de sintonización***



(b) ***Comportamiento del eslabón 2 a diferentes métodos de sintonización***



(c) ***Comportamiento del eslabón 3 a diferentes métodos de sintonización***



(d) ***Comportamiento de los eslabones a diferentes métodos de sintonización en trayectoria cerrada***

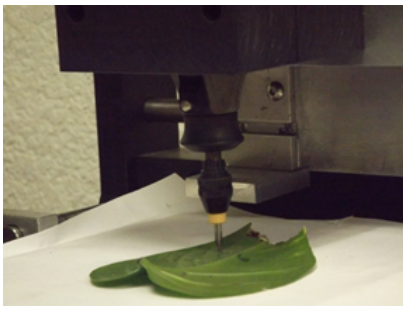
Figura 6.8: Comportamiento del robot a diferentes métodos de sintonización

En la figuras (6.8(a)), (6.8(b)), (6.8(c)) y (6.8(d)) nos indican las posiciones promedio en las que llegaron los tres eslabones para la posición de prueba con los 4 diferentes tipos de sintonización. Se observa que para el eslabón 1 no hay mucha diferencia en trabajar con la sintonización clásica y con los métodos de sintonización de evolución diferencial. Para el eslabón 2 no presenta estabilización en el estado estacionario, con la sintonización propuesta por Kelly y Reyes [16, 17], mientras con el ajuste de los algoritmos de evolución diferencial se logra una estabilización por debajo de los $0.5s$. Para el eslabón 3 los máximos sobre impulsos trabajando con la sintonización de [16, 17], son altos y con el ajuste de los algoritmos de evolución diferencial mejora la respuesta en estado transitorio, disminuyendo los sobre impulsos y el tiempo de asentamiento fue menor a 0.31 segundos. Sin embargo como se corrobora en el cuadro (6.9) el método $DE1$ fue mejor que los otros tres métodos de sintonización $DE1$ ubica el efector final a $0.053m$, muy cercano a la posición deseada del punto de prueba, seguido de JDE que lo ubica en $0.052m$. JADE y SaDE presenta una sobre estimación ubicando el eslabón 3 a $0.061m$ para SaDE y $0.058m$ para JADE. El tiempo de aproximación del eslabón 1 con el ajuste de $DE1$ y JDE es de 1.14 segundos.

En la figura (6.8(d)) que el método $DE1$, supera el desempeño y definición en el trazado de una trayectoria circular con respecto a los métodos adaptivos (SaDE, JDE y JADE). Ya que se puede apreciar mejor definición en la trayectoria circular, atenuación en los sobre impulsos en el eje y, el peor método para el desarrollo del proceso de seguimiento de trayectoria fue el método SaDE ya que se pierde definición en la trayectoria circular. En los cuatro métodos llegan al mismo tiempo en el punto donde empezará la trayectoria de corte, siendo el eje x el más lento con un tiempo de llegada de 2.3 segundos, mientras que el eje y llega a los 0.75 segundos.

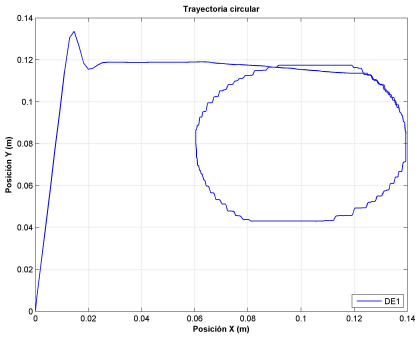
6.4. Experimento 4: Evaluación de las herramientas de corte

Una vez obtenido el mejor método de ajuste de parámetros para la sintonización del controlador (3.4), se procedio en la evaluación de cada herramienta de corte. El primer cortador que se evaluo fue el cortador vertical de $2mm$, (figura 6.9(a)), con las ganancias obtenidas con el método $DE1$ y con la trayectoria circular definida en la ecuación (3.45). Se realizó seguimiento de trayectoria circular sobre la hoja de Filodendro (*Philodendron spp*) obteniendo secciones de un centímetro y cuatro centímetros de diámetro, figura (6.9(b)). Se puede observar un pequeños desgarres de un milimetro en las hojas.



(a) ***Cortador vertical***

(b) ***Secciones de hojas***



(c) ***Trayectoria seguida por el robot***

Figura 6.9: Herramienta de corte vertical y su trayectoria

La segunda herramienta de corte evaluada fue un sacabocados de $10.92mm$ de diámetro montado en el robot cortador, figura (6.11(a)), para obtener secciones pequeñas de $2cm^2$. Utilizando las ganancias obtenidas con el método de $DE1$, cuadro (6.8),y seis posiciones diferentes sobre la hoja de Filodendro (*Philodendron spp*), sienda estas:

Cuadro 6.10: Posiciones de los cortes

Parámetro	Eslabón 1	Eslabón 2	Eslabón 3
Posición 1	$2cm$	$7cm$	$6cm$
Posición 2	$6cm$	$21cm$	$6cm$
Posición 3	$15cm$	$3cm$	$6cm$
Posición 4	$11cm$	$5cm$	$6cm$
Posición 5	$12cm$	$5cm$	$6cm$
Posición 6	$9cm$	$4cm$	$6cm$

En la figura (6.10) se observa la trayectoria recorrido en los tres eslabones que constituye el robot de corte, se observa que con la sintonización de $DE1$ se atenua los sobre impulsos en el eslabón 2 correspondiente al eje y, en el eslabón 3 correspondiente al eje z recorre el mismo trayecto de $6cm$, con $4mm$ en algunas repeticiones.

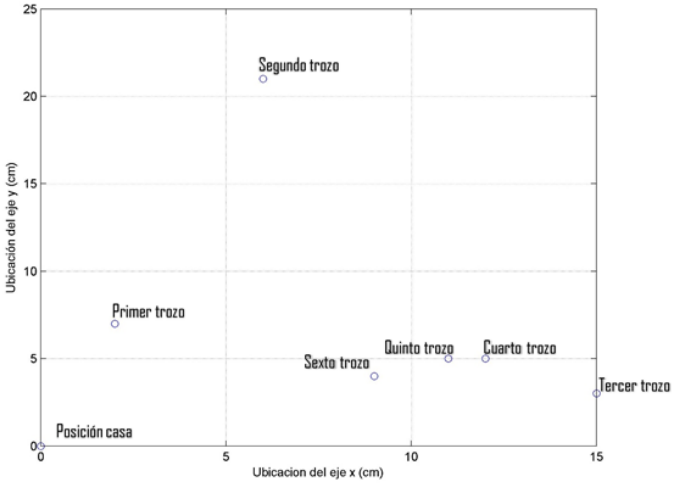
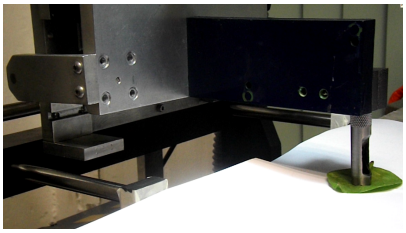
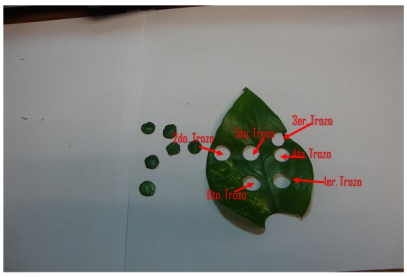


Figura 6.10: Comportamiento de los puntos de prueba.

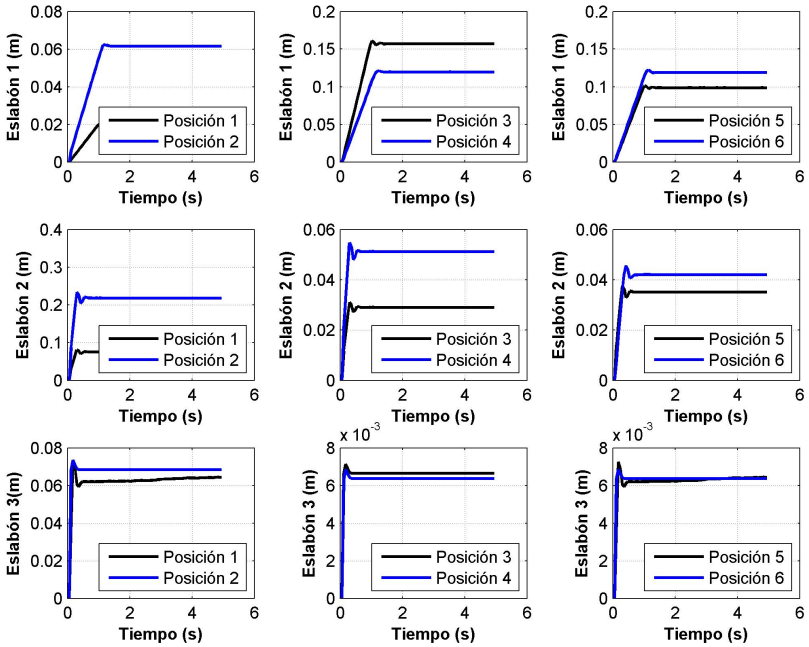
En la figura (6.11(a)) se puede apreciar el uso del cortador tipo sacabocados y los cortes realizados por la herramienta,(figura 6.11(b)), los cuales tienen un diámetro de 1cm y los bordes se aprecian mas finas que el anterior cortador. Finalmente en la figura (6.11(c)), se puede apreciar los puntos que siguió alrededdor de la hoja por el cortador, notese que el punto más alejado es el punto que coincide en el borde más alejado de la hoja.



(a) ***Cortador sacabocados***



(b) ***Secciones de hojas***



(c) ***Trayectoria seguida por el robot***

Figura 6.11: Herramienta sacabocados y su trayectoria

Finalmente como se puede apreciar en la figura (6.11(b)) y la figura (6.9(b)) que los cortes realizados por ambas herramientas tienen la misma área de $1.57cm^2$, sin embargo, el cortador tipo sacabocados es mas limpia que del cortador vertical y que este ultimo presenta desgarres de $1mm$.

6.5. Experimento 5: Localización de hojas

El siguiente experimento es la integración del sistema de visión para el reconocimiento de hojas, para realizar esta práctica se tomaron cuatro hojas de la misma tonalidad y se establecio los valores $R = 1$, $G = 1$ y $B = 0.45$, para calibrar el algoritmo de Bhargav con la finalidad de localizar el espectro de de color que le corresponde a las hojas y asignarle un centroide; como se muestra en la figura (6.12) referenciado en pixeles el algoritmo hallo cuatro objetos.

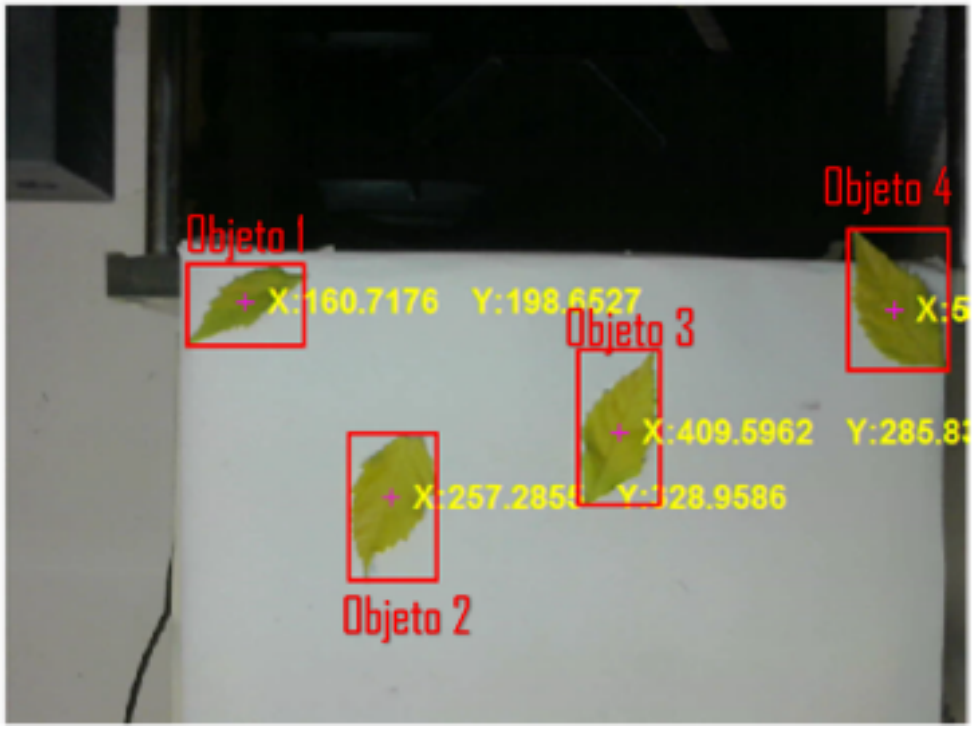


Figura 6.12: Hojas localizadas con el algoritmo de Bhargav.

Posteriormente estas referencias en pixeles son transformadas en posiciones deseadas pa-

ra cada objeto, a tráves de las ecuaciones (4.9), (4.10) y (4.11) siendo estas las siguientes:

Cuadro 6.11: Posiciones de las hojas

Posiciones	x_{pix}	y_{pix}	x_{real}	y_{real}
Posición 1	$160pix$	$198pix$	$0.04m$	$0.054m$
Posición 2	$257pix$	$328pix$	$0.070m$	$0.089m$
Posición 3	$409pix$	$285pix$	$0.11m$	$0.077m$
Posición 4	$502pix$	$210pix$	$0.13m$	$0.057m$

Para la implementación de las posiciones deseadas por el algoritmo de Bhargav, x_{real} y y_{real} son almacenados en un vector del tamaña del vector es decir $x_{real} \in \mathbb{R}^4$ y $y_{real} \in \mathbb{R}^4$, para alimentar el esquema (4.3), es importante considerar un fondo blanco para el contraste de colores. Utilizando las ganancias $DE1$ del cuadro (6.8) para el controlador $PD+$, ecuación (3.4), se obtuvo el siguiente comportmiento:

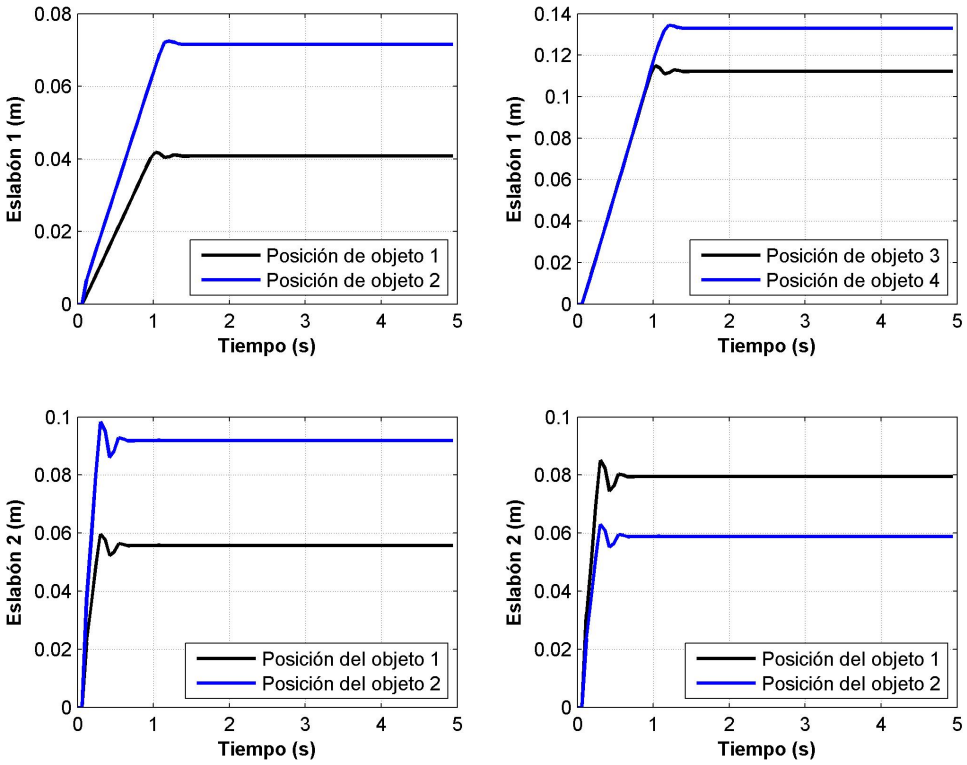


Figura 6.13: Posiciones de cada hoja localizada por el algoritmo de Bhargav.

En la figura (6.13) se puede apreciar el arrivo del eslabón 1 correspondiente al eje x a cada punto, como se puede apreciar a tráves de la sintonización de $DE1$ llevo la herramienta de manera suave a los cuatro objetos hallados. De la misma manera se puede observar que el eslabón 2 lleva el objeto a cada punto calculado por el modelo de visión presentado en las ecuaciones (4.9),(4.10) y (4.11). El tiempo para llegar a los objetos uno y dos respectivamente por el eslabón 1s para el el objeto uno y de 1.2 segundos para el objeto dos. Para los objetos 3 y 4 son alcanzados en periodos de tiempo de 1.25s.

En la siguiente figura (6.14), se puede apreciar los puntos que siguió el robot sobre la superficie de trabajo, sí sigue del punto al primer punto del objeto 1, despues al punto del objeto 2 y así de manera consecutiva se aprecia que forma un cuadro.

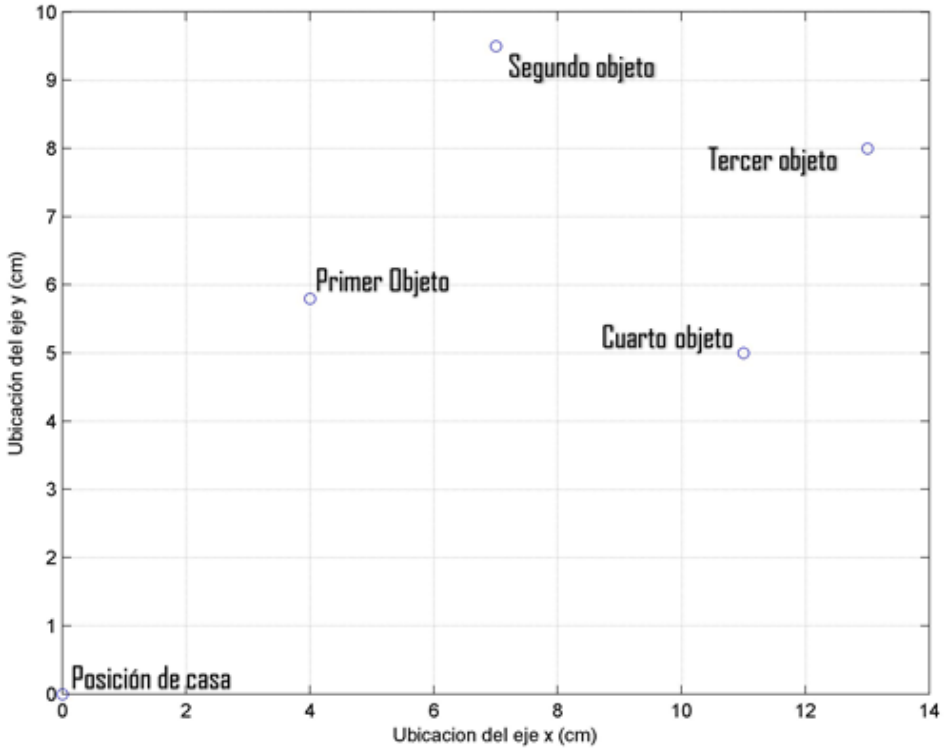


Figura 6.14: Posicion del efector final en el espacio de trabajo.

6.6. Experimento 6: Arribo del efector final con visión

El siguiente experimento es la ubicación de las dos herramienta de corte efectivas, el cortador vertical y el sacabocados, utilizando dos hojas de Filodendro (*Philodendron spp*) para ubicar el centroide del objeto a tráves el algoritmo de Bhargav. Estas referencias una vez mas se transforman de pixeles, figura (6.15), a posiciones deseadas para cada hoja detectada, a tráves de las ecuaciones (4.9), (4.10) y (4.11) siendo estas las siguientes:

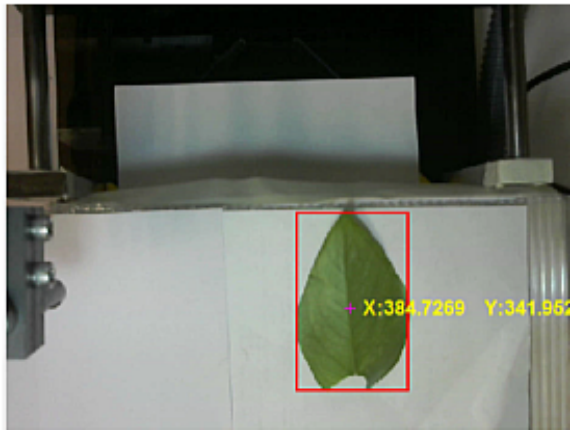
Cuadro 6.12: Posiciones de las hojas a cortar

Posiciones	x_{pix}	y_{pix}	x_{real}	y_{real}
Posición 1	$384pix$	$363pix$	$0.10m$	$0.09m$
Posición 2	$341pix$	$159pix$	$0.090m$	$0.043m$

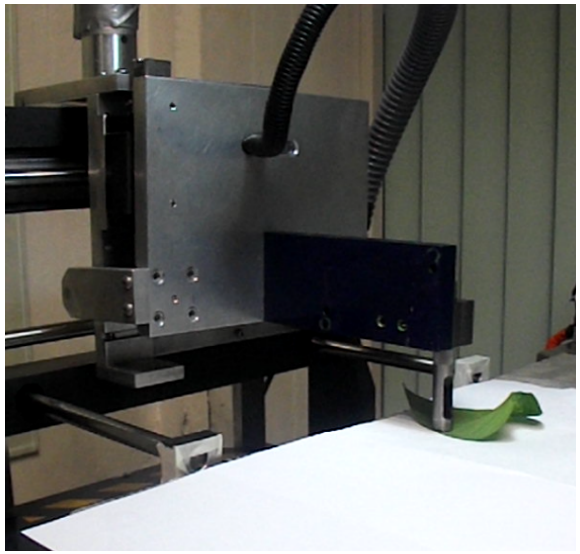
Una vez más las posiciones deseadas por el algoritmo de Bhargav, x_{real} y y_{real} son almacenados en un vector del tamaña del vector es decir $x_{real} \in \mathbb{R}^2$ y $y_{real} \in \mathbb{R}^2$, para alimentar el esquema (4.3), es importante considerar un fondo blanco para el contraste de colores. Utilizando las ganancias $DE1$ del cuadro (6.8) para el controlador $PD+$, ecuación (3.4).

En la figura (6.15(d)), se puede apreciar el arrivo del eslabón 1 correspondiente al eje x y del eslabón 2 correspondiente al eje y a tráves de la sintonización de $DE1$ llevo la herramienta de manera suave a la hoja, (figura 6.15(c)), al punto calculado por el modelo de visión presentado en las ecuaciones (4.9),(4.10) y (4.11). Sin embargo al realizar el corte este sale desviado $1cm$ del centroide original, figura (6.15(b)).

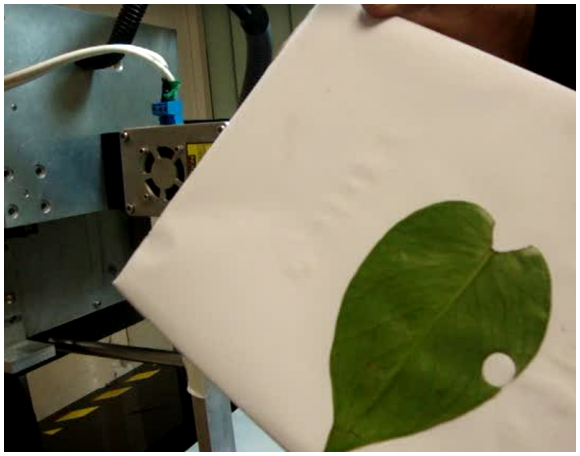
En la figura (6.16(d))se observa el posicionamiento de los eslabones uno y dos al centroide calculado por el modelo de visión los cuales son a los $0.09m$ para el eslabón 1 y 0.04 para el eslabón 2. Se puede apreciar en la figura (6.16(c)), aunque no hay mucha desviación al punto de arrivoe la sección de hoja es irregular con este cortador, por lo que se recomienda el uso del sacabocados y afinar solo el sistema de visión o bien otro cortador que permita hacer cortes sin desgarrar de manera irregular.



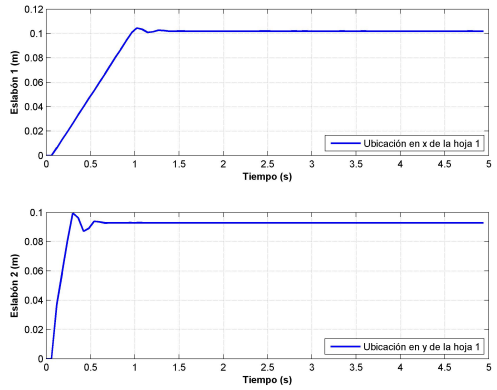
(a) ***Centroide de la primera hoja de Filodendro para el cortador sacabocados***



(b) ***Posicionamiento del sacabocados sobre la hoja***

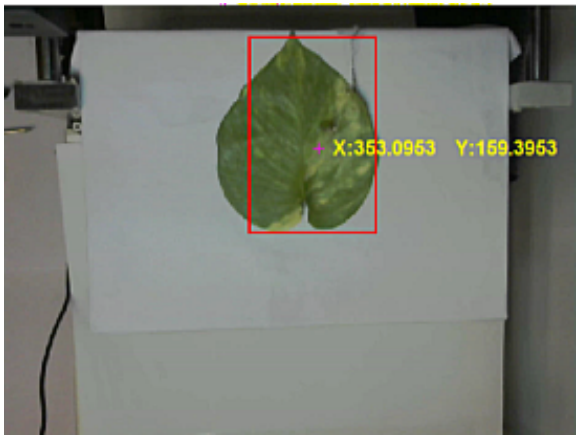


(c) ***Seccion de hoja adquirida por el sacabocados***



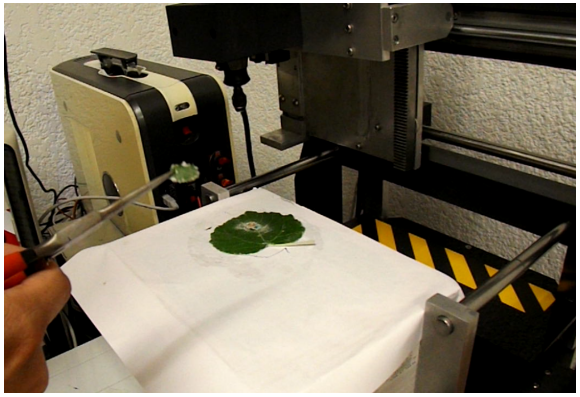
(d) ***Respuesta de los eslabones*** x, y

Figura 6.15: Proceso para detección y corte del Filodendro (*Philodendron spp*)con el sacabocados

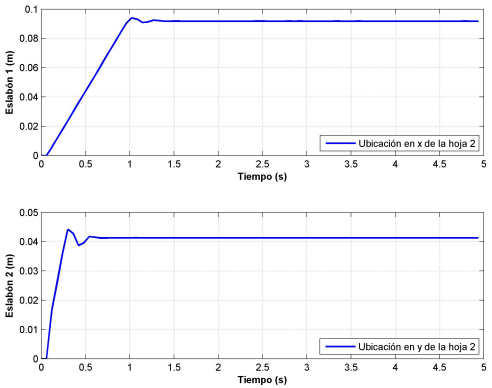


(a) ***Centroide de la segunda hoja de Filodendro para el cortador vertical***

(b) ***Posicionamiento del cortador vertical sobre la hoja***



(c) ***Seccion de hoja adquirida por el cortador vertical***



(d) ***Respuesta de los eslabones*** x, y

Figura 6.16: Proceso para detección y corte del Filodendro (*Philodendron spp*)con el cortador vertical

Conclusiones

En este tema de tesis se presenta el diseño y control de un sistema de corte de hojas para la propagación de tejido vegetal basado en una configuración cartesiana. Se analizaron dos formas de sintonización a dos puntos diferentes de trabajo, donde el controlador puede ejercer. Se encontró que la sintonización a través del algoritmo de Cuckoo Search es una herramienta útil para la sintonización de sistemas dinámicos no lineales.

Para el caso del robot cartesiano cortador se hallaron, con Cuckoo Search, valores que aumentaron el desempeño del controlador, reduciendo las vibraciones mecánicas y mejorando la respuesta del estado transitorio para los tres eslabones, lo que asegura mayor estabilidad el momento de llevar las herramientas de corte a la zona de trabajo. Estos resultados obtenidos también validan el desarrollo mecánico, la instrumentación electrónica que se realizó a través de Arduino y la interfaz desarrollada usando $GUIDE - MATLAB$. A través de pruebas de repetitividad bajo el esquema de control de posición obteniendo que la sintonización realizado por el algoritmo de Cuckoo search mejora el desempeño del robot en tiempo de estabilización y reducción de máximo sobre impulsos. No es recomendable trabajar al límite máximo permitido de sintonización ya que decae el desempeño mecánico, esto físicamente pone en riesgo los actuadores al presentar vibraciones mecánicas.

Para el caso del problema de control de movimiento o seguimiento de trayectoria se observó que para $DE1$ y JDE presento un índice de desempeño de 0.23 y 0.29 radianes esto quiere decir que el efector final del robot manipulador se aproximó con mayor precisión al punto de inicio de la trayectoria de corte. Sin embargo el método de $DE1$ supera a los mé-

todos adaptivos para el ajuste de parámetros para el controlador PD+ en el seguimiento de trayectoria logrando secciones de hojas de $1cm$ y $4cm$ de diámetro.

En la evaluación de las herramientas nos arrojó que el sacabocados brinda un corte más limpio con respecto al cortador vertical aunque ambos fueron capaces de obtener secciones de hoja de $1cm^2$.

La implementación de visión permitió añadirle al sistema más autonomía ya que permite detectar objetos, sin embargo, para objetos complejos el algoritmo de Bhargav no tiene un adecuado funcionamiento, pero por otra parte abre las posibilidad de hacer un estudio más profundo de técnicas de visión para para su mejoramiento o bien proponer uno nuevo.

Finalmente se recomienda hacer una base que permita la sujeción y re orientación de la hoja para obtener diferentes cortes y medidas para la evaluación de estas secciones de hoja en laboratorio y determinar si esta herramienta es adecuada como auxiliar en el corte de tejido para propagación vegetal. Además de una evaluación de más herramientas de corte para el perfeccionamiento de la máquina.

Apéndice A

Programas para adaptación de parámetros

A.1. Programa del modelo dinámico

```
function dx=FRDMPcorte(t,x)


%Esta es la simulacion de un robot cartesiano de cuatro grados de libertad


global m1 m2 m3  qt1 qt2 qt3
global kv1 kv2 kv3  kd1 kd2 kd3
%Vector de errores de posicion
global qd1 qd2 qd3 qdp1 qdp2 qdp3
global qdpp1 qdpp2 qdpp3


%vector de posiciones
q=[x(1);x(2);x(3)];
```

```
%vector de velocidades


qp=[x(4);x(5);x(6)];


%error de posicion


qt1=qd1-q(1);

qt2=qd2-q(2);

qt3=qd3-q(3);


%MATRIZ DE MASAS E INERCIAS


m11=m1+m2+m3;

m12=0;

m13=0;


m21=0;

m22=m2+m3;

m23=0;


m31=0;

m32=0;

m33=m3;

Ga=0.000000000066742;

mtb=0.510;

l3=0.15;


M=[m11 m12 m13; m21 m22 m23; m31 m32 m33];
```

```
%PAR GRAVITACIONAL


g1=0;

g2=0;

g3=(m3*mtb/l3)*Ga;

G=[g1;g2;g3];



%ERROR DE VELOCIDAD

qt1p=qdp1-qp(1);

qt2p=qdp2-qp(2);

qt3p=qdp3-qp(3);


%TORQUES APLICADOS para posicionamiento


       func1=m11*qdpp1+m12*qdpp2+m13*qdpp3+kd1*tanh(qt1)+kv1*tanh(qt1p);

       func2=m21*qdpp1+m22*qdpp2+m23*qdpp3+kd2*tanh(qt2)+kv2*tanh(qt2p);

       func3=m31*qdpp1+m32*qdpp2+m33*qdpp3+kd3*tanh(qt3)+kv3*tanh(qt3p);



       Tau1=g1+func1;

       Tau2=g2+func2;

       Tau3=g3+func3;

       TAU=[Tau1 ; Tau2; Tau3];

%vector de aceleracion articular

q2p=inv(M)*(TAU-G);


%vector de salida
```

```
dx=[qp(1);qp(2);qp(3);q2p(1);q2p(2);q2p(3)];


end
```

A.2. Programa de la Función Objetivo

```
function L2 = ajustegainp(xp)
% modelo desacoplado del robot de corte, sistema de
% posicionamiento+orientacion


global kv1 kv2 kv3 kd1 kd2 kd3
global m1 m2 m3


global qt1 qt2 qt3 ts tfi



%ganancias para el modelo de posicion
kd1=xp(1)*(16/1.568);
kv1=xp(2)*(16/1.568);
kd2=xp(3)*(16/1.568);
kv2=xp(4)*(16/1.568);
kd3=xp(5)*(16/1.568);
kv3=xp(6)*(16/1.568);


%Ganancias para el modelo de orientacion


m1=xp(7);
m2=xp(8);
```

```
m3=xp(9);



opciones=odeset('RelTol',1e-3); %ordinary differential setting
%relative tolerance, comando para definir la tolerancia
[t,x]=ode23('FRDMPcorte',ts,[0; 0; 0; 0; 0; 0],opciones); %el tiempo puede
%ser sustituido por t solamente ya que ha sido definido con anterioridad.



%funcion objetivo a calibrar
Qt=[qt1 qt2 qt3];

integral=trapz(abs(Qt));

L2=sqrt((1/tfi)*integral);
```

A.3. Programa del algoritmo de Cuckoo Search

```
clear all
close all
itermaxc=10;
Solu=0;
Pindex=0;
tic;
for fil=1:itermaxc
global kv1 kv2 kv3 kd1 kd2 kd3
global m1 m2 m3
global qd1 qd2 qd3 t ts tfi
```

```
global qdp1 qdp2 qdp3

global qdpp1 qdpp2 qdpp3


h=0.00025;

ti=0;

tfi=20;


ts=ti:h:tfi;


%Cordenadas articulares para posicionamiento

qd1=pi*2;

qd2=pi*2;

qd3=pi*2;


qdp1=0;

qdp2=0;

qdp3=0;


qdpp1=0;

qdpp2=0;

qdpp3=0;


fobj = 'ajustegain'; %  name of the objective function

n = 25; % Number of nests (or different solutions)

pa = 0.25; % Discovery rate of alien eggs/solutions

Tol = 1e-6; % Tolerance

% Lower and Upper bound of the control variable.

Lb=[0.0484, 0.01205, 0.0944, 0.02361, 0.245, 0.0621];

Ub=[1.25,0.313, 1.25, 0.313, 1.25, 0.313];
```

```
itermax = 20; % Maximum number of generations
nopt = 1; % number of re-runs

mejorsol = [];
mejorfeval = [];
numevalfun = [];
for k=1:nopt
    letrero = sprintf('Evaluacion %d de %d', k,nopt);
    disp(letrero);
    [bestnest,fmin,nfeval] = cuckoo_search(fobj,Lb,Ub,n,pa,Tol,itermax);
    mejorsol = [mejorsol; bestnest];
    mejorfeval = [mejorfeval; fmin];
    numevalfun=[numevalfun;nfeval];
end

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

nol=length(bestnest)

for col=1:1:nol
    Solu(fil,col)=bestnest(col);
end



Pindex(1,fil)=fmin;

disp('Valores estimados: ')
%ganancias para el modelo de posicion
kd1=bestnest(1);
```

```
kv1=bestnest(2);

kd2=bestnest(3);

kv2=bestnest(4);

kd3=bestnest(5);

kv3=bestnest(6);
```

A.4. Programa del algoritmo de DE1

```
% *********************************************************
% Parametros del algoritmo Evolucion Diferencial
fname = 'ajustegain';
CR = 0.5; % Crossing over probability
F = 0.5;  % Differential variation factor
n = 8;   % Number of variables to be optimized
NP = n*5; % Population size
strategy = 7; % Selected strategy:7=rand/1/bin,11=rand/2/bin,12=current-to-rand/1/bin
Accuracy = 1e-5; % The population converges when the difference between
                 % the worst solution and the best solution is less than
                 % Accuracy
itermax = 150; % Maximum number of generations
xmin = [0.0484, 0.01205, 0.0944, 0.02361, 0.245, 0.0621];
xmax = [1.25,0.313, 1.25, 0.313, 1.25, 0.313];
% *********************************************************
%nopt = 2; % Numero de corridas

[BestIndividual,BestValue,FunEvals] = ...
        deoptc(fname,Accuracy,n,xmin,xmax,NP,F,CR,itermax,strategy);
```

```
disp('Valores estimados: ')
KP1 = BestIndividual(1)
KV1 = BestIndividual(2)
KP2 = BestIndividual(3)
KV2 = BestIndividual(4)
KP3 = BestIndividual(5)
KV3 = BestIndividual(6)
KPO1 = BestIndividual(7)
KVO1 = BestIndividual(8)
```

A.5. Programa del algoritmo de SaDE

```
clear all
close all


clear all
close all
clc
iter=10;
Solu=0;
Pindex=0;
tic;
global kv1 kv2 kv3 kd1 kd2 kd3
global m1 m2 m3 tfi
global qd1 qd2 qd3 t ts
global qdp1 qdp2 qdp3 jj
global qdpp1 qdpp2 qdpp3


for fil=1:iter
```

```
r3=(0.02269)/2;

r2=(0.02269)/2;

r1=(0.01588)/2;



  %Punto de Prueba enb coordenadas cartesianas

    xc=0.20;

    yc=0.15;

    zc=0.05;


%Cordenadas articulares para posicionamiento

qd1=xc/r1;

qd2=yc/r2;

qd3=zc/r3;


qdp1=0;

qdp2=0;

qdp3=0;


qdpp1=0;

qdpp2=0;

qdpp3=0;


h=0.01;

ti=0;

tfi=10;


ts=ti:h:tfi;
```

```
fil
fname='ajustegainp';  %   name of the objective function
 % User defined routine according to the particular problem
% This routine calls a Simulink model and it call a C-Mex s-function
%%                   %User would desire to modify these three parameters
%%                   CR, F, NP in order to tune algorithms
CR=01;              % Crossing over probability
F=0.9;               % Differential variation factor
NP=50;               % Population size
Accuracy=1e-8;       % The population converges when the difference
% between the worst solution and the best solution is less than Accuracy
n=9;          % Number of variables to be optimized. Piece-wise constant approximation
% for the controls is assumed
Kp=1;
xmin = [0.0484, 0.01205, 0.0944, 0.02361, 0.245, 0.0621, 0.1, 0.1, 0.1];
xmax = [1.25,0.313, 1.25, 0.313, 1.25, 0.313, 0.4, 0.3, 0.2];
itermax=150;         % Maximum number of generations
strategy=7;         % Selected strategy: DE/rand/1/bin
Pf=1;
VTR=1e-6;
Fl=0.1;
Fu=0.9;
Tau1=0.1;
Tau2=0.2;


[BestIndividual,BestValue,FunEvals]=
sadeopt(fname,Accuracy,n,xmin,xmax,NP,F,CR,itermax,strategy,Kp,VTR,Fl,Fu,Tau1,Tau2);
```

```
 nol=length(BestIndividual);

    for col=1:1:nol
    Solu(fil,col)=BestIndividual(col);
end


Pindex(1,fil)=BestValue




end

 averageTime = toc

disp('Valores estimados: ')
Prom=median(Solu)
desvstd=std(Solu)

disp('Indice de desempeño estimados: ')

Promin=median(Pindex)
desvstdin=std(Pindex)

fid=fopen('Promedios.txt','w');
fprintf(fid,'%i \n',Prom)

               kd1=Prom(1);
```

```
kv1=Prom(2);
kd2=Prom(3);
kv2=Prom(4);
kd3=Prom(5);
kv3=Prom(6);


 %Parametros del robot de 2 grados de libertad
m1=Prom(7);
m2=Prom(8);
m3=Prom(9);
```

A.6. Programa del algoritmo de JDE

```
clear all
close all

clear all
close all
clc
iter=10;
Solu=0;
Pindex=0;
tic;

global kv1 kv2 kv3 kd1 kd2 kd3
global m1 m2 m3 tfi
global qd1 qd2 qd3 t ts
global qdp1 qdp2 qdp3
```

```
global qdpp1 qdpp2 qdpp3


for fil=1:iter
r3=(0.02269)/2;
r2=(0.02269)/2;
r1=(0.01588)/2;


  %Punto de Prueba enb coordenadas cartesianas
    xc=0.20;
    yc=0.15;
    zc=0.05;


%Cordenadas articulares para posicionamiento
qd1=xc/r1;
qd2=yc/r2;
qd3=zc/r3;


qdp1=0;
qdp2=0;
qdp3=0;


qdpp1=0;
qdpp2=0;
qdpp3=0;


h=0.01;
ti=0;
tfi=10;
```

```
ts=ti:h:tfi;



fname='ajustegainp';   % User defined routine according to the particular problem
% This routine calls a Simulink model and it call a C-Mex s-function
%%                  %User would desire to modify these three parameters
%%                  CR, F, NP in order to tune algorithms
CR=0.9;             % Crossing over probability
F=0.9;              % Differential variation factor
NP=50;              % Population size
Accuracy=1e-8;      % The population converges when the difference
% between the worst solution and the best solution is less than Accuracy
            % Number of variables to be optimized. Piece-wise constant approximation
           % for the controls is assumed
n=9;      % Number of variables to be optimized. Piece-wise constant approximati
        % for the controls is assumed
Kp=1;
xmin = [0.0484, 0.01205, 0.0944, 0.02361, 0.245, 0.0621, 0.1, 0.1, 0.1];
xmax = [1.25,0.313, 1.25, 0.313, 1.25, 0.313, 0.4, 0.3, 0.2];
itermax=150;        % Maximum number of generations
strategy=7;        % Selected strategy: DE/rand/1/bin
Pf=1;
VTR=1e-6;
Fl=0.1;
Fu=0.9;
Tau1=0.1;
Tau2=0.1;

[BestIndividual,BestValue,FunEvals]=
```

```
jdeopt(fname,Accuracy,n,xmin,xmax,NP,F,CR,itermax,strategy,Kp,VTR,Fl,Fu,Tau1,Tau2);
nol=length(BestIndividual);

    for col=1:1:nol
    Solu(fil,col)=BestIndividual(col);
end



Pindex(1,fil)=BestValue;



end

 averageTime = toc
disp('Valores estimados: ')
Prom=median(Solu)
desvstd=std(Solu)

disp('Indice de desempeño estimados: ')

Promin=median(Pindex)
desvstdin=std(Pindex)

fid=fopen('Promedios.txt','w');
fprintf(fid,'%i \n',Prom)

              kd1=Prom(1)*(16/1.568);
              kv1=Prom(2)*(16/1.568);
              kd2=Prom(3)*(16/1.568);
```

```
            kv2=Prom(4)*(16/1.568);
            kd3=Prom(5)*(16/1.568);
            kv3=Prom(6)*(16/1.568);

    %Parametros del robot de 2 grados de libertad
   m1=Prom(7);
   m2=Prom(8);
   m3=Prom(9);
```

A.7. Programa del algoritmo de JADE

```
clc
clear all
format long g

% ************************************************************************
% Define the dimension of the problem
n = 9;
popsize = n*2;
% lu: define the upper and lower bounds of the variables
lu = [0.0484, 0.01205, 0.0944, 0.02361, 0.245, 0.0621, 0.1, 0.1, 0.1;...
   1.25,0.313, 1.25, 0.313, 1.25, 0.313, 0.4, 0.3, 0.2];
NumMaxTime = 150;
% ************************************************************************


global qd1 qd2 qd3 ts
global qdp1 qdp2 qdp3 tfi
```

```
global qdpp1 qdpp2 qdpp3


% Record the best results

outcome = [];

nEvalFObj = [];

BestIndividual = [];


% Main body which was developed by the authors

time = 1;


% The total number of runs

totalTime = 10;

tic


    while time <= totalTime


r3=(0.02269)/2;

r2=(0.02269)/2;

r1=(0.01588)/2;



  %Punto de Prueba enb coordenadas cartesianas

    xc=0.20;

    yc=0.15;

    zc=0.05;


%Cordenadas articulares para posicionamiento

qd1=xc/r1;

qd2=yc/r2;
```

```
qd3=zc/r3;


qdp1=0;

qdp2=0;

qdp3=0;


qdpp1=0;

qdpp2=0;

qdpp3=0;


h=0.01;

ti=0;

tfi=10;


ts=ti:h:tfi;

rand('seed', sum(10000 * clock));

% Initialize the main population

 popold = repmat(lu(1, :),

 popsize, 1) + rand(popsize, n) .* (repmat(lu(2, :) - lu(1, :), popsize, 1));

valParents = ajustegainpJADE(popold);

 %AQUI SE PONE EL NOMBRE DEL ARCHIVO DE LA FUNCION OBJETIVO


 c = 1/10;

 p = 0.05;

 CRm = 0.7; % 0.5

 Fm = 0.7; % 0.5


 Afactor = 1;
```

```
 archive.NP = Afactor * popsize; % the maximum size of the archive
 archive.pop = zeros(0, n); % the solutions stored in te archive
archive.funvalues = zeros(0, 1); % the function value of the archived solutions

%% the values and indices of the best solutions
[valBest, indBest] = sort(valParents, 'ascend');
BestIndividualTemporal = zeros(n,1);

iter = 1;
FES = 0;
nEval = 0; % Numero de evaluaciones de la funcion objetivo
while iter <= NumMaxTime
% while FES < n * 10000 %& min(fit)>error_value(problem)

pop = popold; % the old population becomes the current population

if FES > 1 && ~isempty(goodCR) && sum(goodF) >
 0 % If goodF and goodCR are empty, pause the update
CRm = (1 - c) * CRm + c * mean(goodCR);
Fm = (1 - c) * Fm + c * sum(goodF .^ 2) / sum(goodF); % Lehmer mean
end

% Generate CR according to a normal distribution with mean CRm, and std 0.1
% Generate F according to a cauchy distribution with location parameter Fm,
[F, CR] = randFCR(popsize, CRm, 0.1, Fm, 0.1);

r0 = [1 : popsize];
popAll = [pop; archive.pop];
[r1, r2] = gnR1R2(popsize, size(popAll, 1), r0);
```

```
% Find the p-best solutions
pNP = max(round(p * popsize), 2); % choose at least two best solutions
randindex = ceil(rand(1, popsize) * pNP); % select from [1, 2, 3, ..., pNP]
randindex = max(1, randindex); % to avoid the problem that rand = 0 and thus ceil(rand) =
pbest = pop(indBest(randindex), :); % randomly choose one of the top 100p% solutions

% == == == == == == == == == == == == == == == Mutation == =
vi = pop + F(:, ones(1, n)) .* (pbest - pop + pop(r1, :) - popAll(r2, :));

vi = boundConstraint(vi, pop, lu);

% == == == == = Crossover == == == == =
mask = rand(popsize, n) > CR(:, ones(1, n));
% mask is used to indicate which elements of ui comes from the parent
rows = (1 : popsize)'; cols = floor(rand(popsize, 1) * n)+1;
% choose one position where the element of ui doesn't come from the parent
jrand = sub2ind([popsize n], rows, cols); mask(jrand) = false;
ui = vi; ui(mask) = pop(mask);

valOffspring = ajustegainpJADE(ui);
%AQUI SE PONE EL ARCHIVO DE LA UNCION OBJETIVO
FES = FES + popsize;
nEval = nEval + size(ui,1);

% == == == == == == == == == == == == == == == Selection == == ==
% I == 1: the parent is better; I == 2: the offspring is better
[valParents, I] = min([valParents, valOffspring], [], 2);
popold = pop;
```

```
archive = updateArchive(archive, popold(I == 2, :), valParents(I == 2));


popold(I == 2, :) = ui(I == 2, :);


goodCR = CR(I == 2);
goodF = F(I == 2);


[valBest indBest] = sort(valParents, 'ascend');
% Find best individual
BestIndividualTemporal = popold(indBest(1),:);


% Find best solution TEMPORAL
LabelForUser = sprintf('Generation: %d  %.16e',iter,min(valBest));
disp(LabelForUser);
 iter = iter + 1;
  end


        outcome = [outcome min(valParents)];
        nEvalFObj = [nEvalFObj nEval];
        BestIndividual = [BestIndividual; BestIndividualTemporal];
        LabelForUser = sprintf('Time: %d',time);
        disp(LabelForUser);
        time = time + 1;


    end
toc


format long e
```

```
disp('Numero de evaluaciones de funcion objetivo: ')
nEvalFObj'
disp('Valores funcion objetivo:')
outcome'
disp('Media funcion objetivo: ')
mean(outcome)'
disp('Desviacion estandar funcion objetivo: ')
std(outcome)'

disp('Media parametros estimados: ')
mean(BestIndividual)'
disp('Desviacion estandar parametros estimados: ')
std(BestIndividual)'
```

Apéndice B

Demostración de la estabilidad del controlador PD+

Para fines de demostración la estructura de control($C.1$) :

$$\tau_{pd+} = K_p \psi_{\nabla U_p} + K_v \psi_{\nabla U_v} + \tau_m$$

$$\psi_{\nabla U_p} = \begin{bmatrix} \tanh(\tilde{q_1}) & \tanh(\tilde{q_2}) & \dots & \tanh(\tilde{q_n}) \end{bmatrix} \tag{B.1}$$

$$\psi_{\nabla U_v} = \begin{bmatrix} \tanh(\dot{\tilde{q}_1}) & \tanh(\dot{\tilde{q}_2}) & \dots & \tanh(\dot{\tilde{q}_n}) \end{bmatrix}$$

se considera que, cuando la posición deseada q_d no es variante en el tiempo, es decir es constante, la velocidad deseada $\dot{q}_d$ y la aceleración deseada $\ddot{q}_d$ son vectores cero. Por otro lado, el error de velocidad, cumple que $\dot{\tilde{q}} = -\dot{q}$ se obtiene la el controlador de posición con compensación de gravedad denominado tanh fue propuesto originalmente por *Cai* y *Song* y reformulado más tarde por *Kelly, R.*, *Santibañez, V.* y *Reyes, F.*, este controlador pertenece a la familia de los *controladores de acciones saturados* cuya característica es que convergen a cierta cota a medida que el argumento crece indefinidamente, estas funciones son radialmente acotadas, cuya representación matemática es:

$$\tau = \quad K_p \tanh(\tilde{q}) - K_v \tanh(\dot{q}) + g(q) \tag{B.2}$$

donde:

$$\tanh(\tilde{q}) = \begin{bmatrix} \tanh(\tilde{q}_1) \\ \tanh(\tilde{q}_2) \\ \vdots \\ \tanh(\tilde{q}_n) \end{bmatrix} \tag{B.3}$$

Sustituimos la ecuación (C.2) en (3.29):

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} = \quad K_p \tanh(\tilde{q}) - K_v \tanh(\dot{q}) \tag{B.4}$$

obtenemos la ecuación del sistema en lazo cerrado:

$$\frac{d}{dt}\begin{bmatrix} \tilde{q} \\ \dot{q} \end{bmatrix} = \begin{bmatrix} -\dot{q} \\ M(q)^{-1}[K_p \tanh(\tilde{q}) - K_v \tanh(\dot{q}) - C(q,\dot{q})\dot{q}] \end{bmatrix} \tag{B.5}$$

cuya función candidata de Lyapunov es:

$$V(\dot{q},\tilde{q}) = \quad \frac{\dot{q}^T M(q)\dot{q}}{2} + \begin{bmatrix} \sqrt{\ln(\cosh(\tilde{q}_1))} \\ \sqrt{\ln(\cosh(\tilde{q}_2))} \\ \vdots \\ \sqrt{\ln(\cosh(\tilde{q}_n))} \end{bmatrix}^T K_p \begin{bmatrix} \sqrt{\ln(\cosh(\tilde{q}_1))} \\ \sqrt{\ln(\cosh(\tilde{q}_2))} \\ \vdots \\ \sqrt{\ln(\cosh(\tilde{q}_n))} \end{bmatrix} \tag{B.6}$$

al derivarla con respecto al tiempo $\dot{V}(\dot{q},\tilde{q})$, queda:

$$\dot{V}(\dot{q},\tilde{q}) = \dot{q}^T M(q)\ddot{q} + \frac{\dot{q}^T \dot{M}(q)\dot{q}}{2} + \begin{bmatrix} \sqrt{\ln(\cosh(\tilde{q}_1))} \\ \sqrt{\ln(\cosh(\tilde{q}_2))} \\ \vdots \\ \sqrt{\ln(\cosh(\tilde{q}_n))} \end{bmatrix}^T K_p \begin{bmatrix} \frac{\tanh(\tilde{q})}{\sqrt{\ln(\cosh(\tilde{q}))}} \end{bmatrix} \dot{\tilde{q}} \quad \text{(B.7)}$$

donde $\left[\frac{\tanh(\tilde{q})}{\sqrt{\ln(\cosh(\tilde{q}))}}\right] \in \mathbb{R}^{n\times n}$ y $K_p \in \mathbb{R}^{n\times n}$ y tiene la forma:

$$\left[\frac{\tanh(\tilde{q})}{\sqrt{\ln(\cosh(\tilde{q}))}}\right] = \begin{bmatrix} \left[\frac{\tanh(\tilde{q_1})}{\sqrt{\ln(\cosh(\tilde{q_1}))}}\right] & 0 & 0 & 0 \\ 0 & \left[\frac{\tanh(\tilde{q_2})}{\sqrt{\ln(\cosh(\tilde{q_2}))}}\right] & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \left[\frac{\tanh(\tilde{q_n})}{\sqrt{\ln(\cosh(\tilde{q_n}))}}\right] \end{bmatrix} \quad \text{(B.8)}$$

sustituimos $M(q)\ddot{q}$ de la ecuación de lazo cerrado $(C.4)$ se obtiene:

$$\dot{V}(\dot{q},\tilde{q}) = -\dot{q}^T K_v \begin{bmatrix} \tanh(\dot{q}_1) \\ \tanh(\dot{q}_2) \\ \vdots \\ \tanh(\dot{q}_n) \end{bmatrix} + \frac{1}{2}\dot{q}^T[\dot{M}(q) - 2C(q,\dot{q})]\dot{q} \quad \text{(B.9)}$$

aplicamos la propiedad anti simetria se simplifica a:

$$\dot{V}(\dot{q},\tilde{q}) = -\dot{q}^T K_v \begin{bmatrix} \tanh(\dot{q}_1) \\ \tanh(\dot{q}_2) \\ \vdots \\ \tanh(\dot{q}_n) \end{bmatrix} \leq 0 \quad \text{(B.10)}$$

la función $V(\dot{q},\tilde{q})$ es entonces una función de Lyapunov dado que:

$$\dot{V}(\dot{q},\tilde{q}) \leq 0 \quad \text{(B.11)}$$

obteniendo estabilidad asintótica global con el empleo del *teorema de LaSalle*:

$$\dot{V}(\dot{q}, \tilde{q}) < 0 \tag{B.12}$$

Apéndice C

Descripción de la barra de herramientas

MATLAB-GUIDE se basa en la *programación orientada a objetos* (POO). La POO permite realizar grandes programas mediante la unión de elementos más simples, que pueden ser diseñados y comprobados de manera independiente al programa que va a usarlos. Muchos de estos elementos podrán ser reutilizados en otros programas. A estas piezas, módulos o componentes, que interactúan entre sí cuando se ejecuta un programa, se les denomina *objetos*. Estos objetos contienen tanto datos como las funciones que actúan sobre esos datos. Cada uno de estos objetos corresponde a algún elemento que debe utilizar el programa. Durante la ejecución del programa, los objetos interactúan pasándose *mensajes* y *respuestas*. Es fundamental darse cuenta de que un objeto no necesita conocer el funcionamiento interno de los demás objetos para poder interactuar con ellos (igual que el hombre no necesita conocer cómo funciona por dentro un televisor o una computadora para poder utilizarlos), sino que le es suficiente con saber la forma en que debe enviarle sus mensajes y cómo va a recibir la respuesta [18].

Por lo tanto MATLAB-GUIDE tiene la característica de manejar: que:

- *Objetos* Es un encapsulamiento genérico de datos y de los procedimientos para manipularlos.

- *Mensajes:* Se da cuando un programa orientado a objetos se ejecuta, los objetos están recibiendo, interpretando y respondiendo a mensajes de otros objetos, lo que origina cambios en el estado del objeto.

- *Métodos:* Es cuando se implementa en una clase y determina como tiene que actuar el objeto cuando recibe un mensaje.

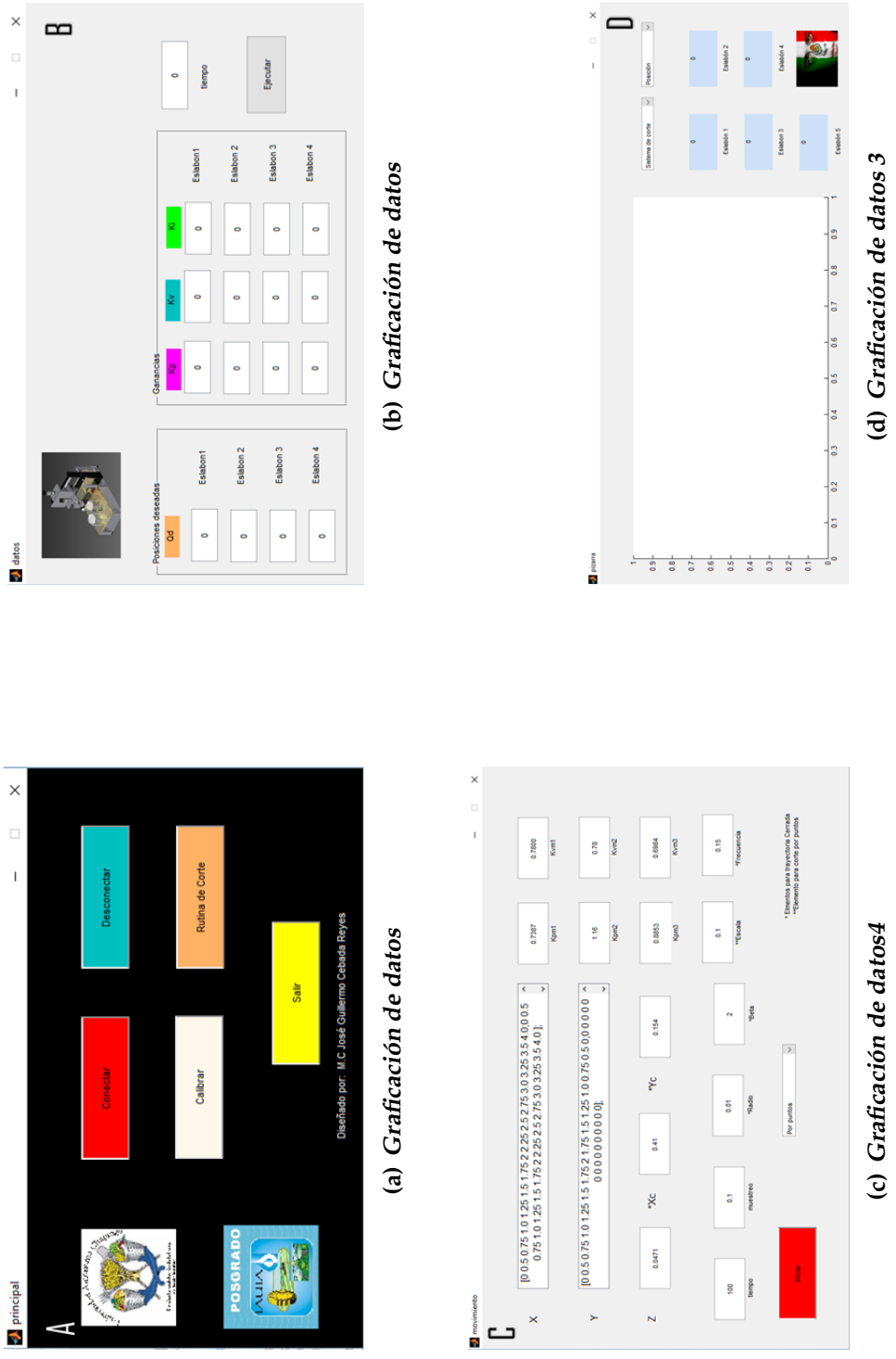
- *Clases:* Se puede considerar como una plantilla para crear objetos de esa clase o tipo; esta describe los métodos y los atributos que definen las características comunes a todos los objetos de esa clase.

La interfaz grafica de usuario es un conjunto de métodos para lograr interactividad entre el usuario y el control del robot cortador. La *GUI* del robot cortador lo conforma cuatro barras de herramientas: La *barra principal*, la *barra de calibración*,la *barra de rutina de movimiento* y la *barra graficadora*. En la figura (C.1), A, se aprecia la *barra de herramientas principal*. Cuya funcion es la de inicializar y trabajar. La forma en la que va interactuar la barra de herramientas con el usuario es a través de banderas y variables tipo puntero, a traves de sus cinco botones, los cuales llamaran por referencia a las clases *Calibrar* y el *Rutina de movimiento*, estas variables accederán a todas los *métodos* y *propiedades* de la interfaz.

- En la figura (C.1), B, se presenta la *barra de calibración*. Tiene la función de calibrar el robot cortador en la posición de casa, así como pruebas de posicionamiento de repetitividad a un punto de prueba para ambas maquinas. Cuenta con un formulario para ubicar los eslabones en cualquier punto del espacio de trabajo del sistema automatizado.

- En la *barra de asignación de trayectoria*,figura (C.1), C, posee un formulario para asignar puntos de corte o bien trabajar con trayectorias circulares para la disección de hojas.

- Finalmente la figura (C.1), D, representa la *barra graficadora*. Permite observar el comportamiento de la dinámica del robot cortador.



(a) ***Graficación de datos***

(b) ***Graficación de datos***

(c) ***Graficación de datos4***

(d) ***Graficación de datos 3***

Figura C.1: Elementos de la interfaz gráfica de usuario

Apéndice D

Publicaciones Aceptadas

- PD+ controller tuning with tje Cuckoo Search algorithm for a leaf cutter robot (Aceptado en Acta Horticulturae). *Autores: J.G Cebada, I. L. López y F. Hahn*
- Mechanical Precision evaluation of a robot for plant tissue propagation with CS tuning (Aceptado en Transaction on Latin America, IEEE). *Autores : J. G Cebada, F. Hahn y A. Ruiz.*

Bibliografía

[1] F.Camarena, J. Chura and R. Blas, *Mejoramiento genético y biotecnológico de plantas* (AgroSaber, UNALM/AGROBANCO, pp 210-228)

[2] H. M. Robles, *Coffee producers in Mexico: problems and excercise budget* (Mexican Rural Develompment Research Reports, Vol.14, pp 6-14,February 2011)

[3] F.Aureoles, J. Rodriguez, J. Legaria and J. Sahagun , *PROPAGACION in vitro DEL MAGUEY BRUTO (Agave inaequidens Koch), UNA ESPECIE AMENAZADA DE INTERES ECONOMICO* (Rev. Chapingo Horticultura, Vol.14, No. 3, December 2008, pp 263-269)

[4] M.K. Razdan, *Introduction to plant tissue culture* (2nd Ed., USA: Science Publishers, Inc., 2003, pp 234-237)

[5] J.P. Mather and P.E. Roberts, *Introduction to cell and tissue culture* (1er ed., USA: Plenum Press., 1998, pp 23-155)

[6] Y. L. Chin, *The use of thidiazuron in tissue culture.* (Plant in vitro cellular and developmental biology, Vol. 29, No. 2, pp 92-96)

[7] J.S Garcia del Rio, *De Stephen Hales a la Biologia molecular* (1er. ed., Spain: Publicacion de la Universidad de Valencia, 2003, pp. 22-23)

[8] P. Lopez Gomez, L. Iracheta Donjuan, M. Castellanos JuÃ¡rez, I. Mendez Lopez, A. Sandoval Esquivez, J.F. Aguirre Medina, M.C. Ojeda Zacarias and A. Gutierrez DÃez, *Influence of explant and culture medium on somatic embryogenesis of coffee leaves* (Rev. Fitotecnia Mexicana, Vol. 33, No.3, pp. 205-213, February 2010)

[9] E.J. Pekkeriet E.J. and E.J van Enten, *Current Developments of High-Tech Robotic and Mechatronic Systems in Horticulture and Challenges for the Future* (Acta. Hort 893, pp 85-94, June 2009)

[10] C. Wunter Bac C and E.J van Enten, *Harvesting Robots for High-value Crops: State of the art review and Challenges Ahead* (Journal Field Robotics, Vol 31, No.6, pp 888-914, April 2014)

[11] M. Spong, S. Hutchinson and M. Vidyasagar, *Robot Modeling and Control* (USA: Wiley, 2005, pp 45-125)

[12] N. Kondo N. and K.C Ting, *Robotics for plant production* (Artificial Intelligence Review, Vol. 12, pp 227-243, June 1998)

[13] R. Levin R., V. Gaba V, B. Tal, S. Hirsh. and D. Denola, *Automated Plant Tissue Culture for Mass Propagation* (Nature, Vol 6, pp 1035-1040, June 1988)

[14] M. Berthouly, M. Dufour, D. Alvard, D. Carasco, I. Alemanno and C. Teisson, *Coffee micropropagation in a liquid medium using the temporary inmersion technique* (Procedings of the 16th ASIC coloquium, Kyoto, 1995, pp 514-519)

[15] M. A. Ashraf, N. Kondo and T. Shiigi, *Use the machine vision to sort tomatoe seedling for grafting robot* (Engineering in Agriculture, Environment and Food, Vol. 4, No. 4, pp 119-125, 2011)

[16] F. Reyes, *Robotics: Control of robot manipulators* (1er ed., Mexico DF: Alfa Omega, 2011, pp 251-470)

[17] R. Kelly and V. Santibanez, *Motion control for robot manipulators* (Spain, Madrid: Pearson Educacion, 2003, pp 5-269)

[18] J. G. Cebada Reyes, *Diseno y desarrollo de una plataforma grafica de simulacion para un robot manipulador de dos grados de libertad* (Ing. thesis, Dept. Electron. Eng., BUAP, Puebla, Mexico, 2009)

[19] J. G. Cebada Reyes, *Control cartesiano para robots manipuladores: Teoria y Practica* (McS. thesis, Dept. Electron. Eng., BUAP, Puebla, Mexico, 2011)

[20] T. M. Khan, M. Arshad and M. A. Choudhry, *Modeling and Control of Cartesian Robot Manipulator* (IEEE 9th International Multitopic Conference INMIC, Karachi, Pakistan, 24-25 December, 2005.)

[21] C. De Los Santos Briones and T. S. M. Hernandez Sotomayor, *Coffee biotechnology* (Brazilian Journal of Plant Physiology, Vol 18, No.1, pp. 217-227, March 2006.)

[22] R. L. Mott and V. Gonzalez *Diseño de elemento de maquina* (Pearson educacion, 2006)

[23] CPO CP Rancks and Pinions Catalog, (CPO 5, 2014)

[24] THK Linear motion guides, (THK Co. LTD D-8090S, 2014.)

[25] Eltra, *Manual de encoders absolutos e incrementales* (Argentina, 2000.)

[26] M. Margolis, *Arduino Cook book* (1er ed., Sebastopol CA. USA: O'Really Media Inc, 2011, pp 6-15.)

[27] P. Francis, *Arduino Essentials* (U.K, Birmingham, Packt Publishing, 2015, pp 7-27)

[28] Pololu Dual VNH 5019 Motor Drive Shield Users Guide (Pololu Co.UK, 2014.)

[29] Matlab, *Creating Graphical User Interfaces R2014b* (The mathworks Inc, US, 2014)

[30] B. Paden and R. Panja, *Globally asymptotically stable PD+ controller for robots manipulators* (International Journal of Control, Vol 47, No.6, pp 1697-1712, June 1987)

[31] R. Kelly, V. Santibañez and F. Reyes, *On saturated-proportional derivative feedback with adaptive gravity compensation of robot manipulator* (Int. Journal of Adaptive Control and Signal Processing, Vol 10, No.4, pp 465-479, December 1998)

[32] P. Sanchez and F. Reyes, *Cartesian Control for Robot Manipulators, Robot Manipulators Trends and Development* (Agustin Jimenez and Basil M Al Hadithi (Ed.), InTech, 2010).

[33] M. A Villegas Aguilar, *Control Cartesiano de una mano mecÃ¡nica de dos dedos* (M.S. thesis, Dept. Electron. Eng., UNAM, Mexico DF, 2006).

[34] P. Corke, *Robotics, Vision and Control, fundamentals Algorithms in MATLAB* (Heidlberg, Berlin: Springer, 2011, pp 59-67).

[35] S. Gomez Moya, *Planeacion de trayectorias para el control de fuerzas sobre superficies esfericas empleando cuarteniones unitarios* (M.S. thesis, Dept. Electron. Eng., UNAM, Mexico DF, 2015).

[36] C. Pinar and B. Erkan. , *A conceptual comparison of the Cuckoo-search, particle swarm optimization, differential evolution and artificial bee colony algorithms* (Artificial Intelligence Review, Vol. 39, No. 4, pp 315-346, July 2014.)

[37] X.S Yang. , *Cuckoo Search and Firefly Algorithm Theory and Applications* (Switzerland: Springer, 2014, pp 347-348.)

[38] A. Hossein Gandomi, X.S. Yang, and A. Hossein Alavi, *Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems* (Engineering with Computers, Vol 29, No.1, pp 17-35, July 2011.)

[39] R. Sethi, S. Panda and B.P. Sahoo, *Cuckoo Search Algorithm Based Optimal Tuning of PID Structured TCSC Controller* (Computational Intelligence in data mining, Vol 1, No 31 of the series Smart Innovation, Systems and Technologies, pp 251-263, December 2014.)

[40] R. L. Salgado Reyes, R. A. Trujillo Rasua and Y.Y. Tarancon *Hybrid Cuckoo Search applied to the Inverse Additive Singular Values Problem* (Conferencia CientÃfica Internacional UCIENCIA, La Habana, Cuba, 24-26 april 2014)

[41] B. Adriane, and S. Serapião, *Cuckoo Search for Solving Economic Dispatch Load Problem* (Intellig. Control Autom, Vol 21, No.8, pp 1995-2004, May 2012.)

[42] G. Zwe Lee, *A particle Swarm optimization approach for optimum design of PID controller in AVR system* (IEEE Transactions on Energy Conversion, Vol. 29, No.3, pp 1523-1531, June 2004.)

[43] I.M Willjuice and S. Baskar, *Evolutionary algorithms based design of multivariable PID controller* (Expert Systems with Applications Vol.36, No. 5, pp 9159-9167, July 2009.)

[44] S.K Smit. and A.E Eiben, *Comparing Parameter Tuning Methods for Evolutionary Algorithms* (IEEE Congress on Evolutionary Computation CEC, Trondheim, Norway, 18-21 May, 2009.)

[45] M. Saad, J. Hishamuddin and Z.M.D. Intan, *PID Controller Tuning Using Evolutionary Algorithms* (WSEAS Trans.on Systems and Control, Vol 7, No.4, pp 139-149, Octuber 2012.)

[46] M.J Neath, A.K. Swain, U.K. Madawala and D.J. Thrimawithana, *An Optimal PID Controller for a Bidirectional Inductive Power Transfer System Using Multiobjective Genetic Algorithm* (IEEE Trans. on Power Electronics, Vol 29, No.3, pp 1523-1531, May 2013.)

[47] I.L López Cruz, L.G Van Willingenburg, and E. Van Straten, *Efficient Differential Evolution algorithms for multimodal optimal control problem* (Applied Soft Computing, Vol 3, pp 97-122, September 2003.)

[48] E.C Trejo Zuñiga, I.L. López Cruz and A. Ruíz García, *Parameter estimation for crop growth model using evolutionary and bio-inspired algorithms* (Applied Soft Computing, Vol 23, pp 474-482, October 2014.)

[49] M. Arslan, M. Cunkas and S. Tahir, *Determination of induction motor parameters with differential evolution algorithm* (Neural Computing and Applications, Vol 21, No.8, pp 1995-2004, May 2012.)

[50] L.V Santana, *An Algorithm Based on Differential Evolution Problem Solving Multiobjective* (M.S. thesis, Dept. Electron. Eng., CINVESTAV, México DF, 2004.)

[51] P. Novoa, C. Cruz, and D. Pelta, *A comparative study on self-adaptive differential evolution in dynamic environments* (Rev. Cubana de Ciencias Inf., Vol. 8, No.4, pp 86-99, December 2004.)

[52] A. K. Qin, V. L. Huang, and P. N. Suganthan , *Differential Evolution Algorithm With Strategy Adaptation for Global Numerical Optimization* (IEEE Trans. on evolutionary computation, Vol. 13, No.2, pp 398-416, April 2009.)

[53] J. Zhang and A.C Sanderson , *Adaptive Differential Evolutio* (Heidelberg, Berlin: Springer, 2009, pp 39-82.)

[54] J. Brest, S Greiner, B. Borko, M. Mernik, and Z. Viljem, *Self-Adapting Control Parameters in Differential Evolution: A Comparative Study on Numerical Benchmark Problems* (IEEE Trans. on evolutionary computation, Vol. 10, No. 6, pp 646-657, December 2006)

[55] J. Zhang and A. C. Sanderson, *JADE, Adaptive Differential Evolution with Optional External Archive* (IEEE Trans. on evolutionary computation, Vol. 13, No. 5, pp 945-958, October 2009.)

[56] A.E. Eiben and J.E. Smith, *Introduction to Evolutionary Computing* (Heidelberg, Berlin: Springer, 2015, pp 26-257.)

[57] X. Wang and S. Zhao , *Differential Evolution Algorithm with Self-Adaptive Population Resizing Mechanism* (Journal Mathematical Problems in Engineering, Vol. 2013, pp 1-14, January 2013.)

[58] L. Durán , *El gran libro del PC interno* (México DF: Alfa Omega,2007, pp 697-699.)

[59] R. González, L. Hernández, E. Izaguirre and E. Rubio, *Estrategia de control para robots manipuladores con realimentación visual y plataforma electro-neumática de 3gdl* (Revista de Ingeniería Mecánica, Vol. 14. No. 3, pp 245-257, December 2011.)

[60] C. Pérez, *Control visual difuso de un sistema no-lineal* (M.S. thesis, Dept. Electron. Eng., CIC-IPN, México DF, 2009.)

[61] A. de la Escalera, *Visión por computador: Fundamentos y Métodos* (Madrid, Sapin: Prentice Hall, 2001, pp 12-14.)

[62] A. Bhargav Anand, (Amrita School of Eng., Bangalore, 2010), *Tracking red color objects using matlab* (Recuperado: https://www.mathworks.com/matlabcentral/profile/authors/2310475-a-bhargav-anand)

[63] Logitech Camera, *Logitech C270 data sheet*(USA, 2014, pp. 1-20)

[64] J. H. Sossa, *Visión Artificial* (Madrid, Sapin: Ra-Ma, 2013, pp 136-188.)

[65] J. Doyle, *Feedback control theory* (U.K: Mcmillian Publishing Co., 1990, pp 1.)