UNIVERSIDAD AUTÓNOMA DE CIUDAD JUÁREZ

INSTITUTO DE INGENIERÍA Y TECNOLOGÍA
DEPARTAMENTO DE INGENIERÍA ELÉCTRICA Y COMPUTACIÓN

CONTROL VARIANTE EN EL TIEMPO Y TEORÍA τ-*Jerk* PARA GENERACIÓN DE TRAYECTORIAS DE MANIPULADOR REDUNDANTE CON OBSERVADOR VISUAL NEURONAL RECURRENTE BICAPA

TESIS QUE PRESENTA:

IVAN CARBAJAL CARLOS

COMO REQUISITO PARCIAL PARA OBTENER EL TÍTULO DE:

MAESTRO EN CÓMPUTO APLICADO

ASESORES
DR. EDGAR ALONSO MARTÍNEZ GARCÍA
DR. RAFAEL TORRES CÓRDOBA

CIUDAD JUÁREZ, CHIHUAHUA Diciembre del 2020

Página asignada para el

OFICIO DE AUTORIZACIÓN DE IMPRESIÓN, EMITIDO POR EL DEPARTAMENTO DE IEC Y FIRMADO POR LAS AUTORIDADES CORRESPONDIENTES.

Esta página deberá ser removida al momento de la impresión final del documento y reemplazarla por el oficio correspondiente.

Página asignada para el

ACTA DE EXAMEN (firmada por el claustro evaluador).

Esta página deberá ser removida al momento de la impresión final del documento, y reemplazarla por el oficio correspondiente

Dedicatorias y agradecimientos

Dedico este trabajo a mi hija Luna Isabel Carvajal por ser la fuente de inspiración y coraje para seguir creciendo personal y académicamente por que aunque se que no necesariamente seguirá mis pasos, deseo ser un ejemplo de que siempre debes tratar de ser la mejor version de ti por que la dedicación, esfuerzo y trabajo que pongas en algo vale la pena si lo hacer con pasión. A mi familia y amigos por el apoyo moral e incluso económico que se requiere para ser un estudiante tiempo completo. Especialmente a mi madre quien en los dias en que tuve que dedicar 14 horas del día a este proyecto, siempre procuro que comiera y que tuviera un espacio limpio donde poder trabajar. Agradezco a mis revisores de tesis el Dr. Rodriguez, Dr. Rodas, el Dr. Carrillo y al Dr. Elifalet por las palabras de apoyo e incluso criticas que me permitieron mejorar y construir este documento por que aunque se que no siempre atendí todos los consejo por falta de tiempo o por que decidí priorizar el plan de mis asesores, aun así siempre tomaba en cuenta todo lo que me dijeron y trate de ponerlo en practica. Y por supuesto a mis asesores el Dr. Edgar Alonzo Martinez y el Dr. Rafael Torres Cordova a quienes admiro profundamente por el trabajo que realizan. Aprecio los consejos que recibí de ellos tanto académicos como personales con ellos aprendí ética académica y el valor de aprender mas que solo terminar. Ademas aprendí que existe mas felicidad en la satisfacción de hacer las cosas bien aunque cuesten más, que hacer algo para quedar bien o aparentar.

Resumen

Se desarrolla un sistema de generación, seguimiento y control para un manipulador robótico con seis grados de libertad que permita realizar el proceso de manufactura. El sistema debe reconocer unas corbatas para ensamble de arnés de seguridad mediante una cámara RGB que utiliza Red Neural de Hopfield para procesar los datos de los tableros de ensamble para el reconocimiento de las zonas de ensamble.

Además se plantea el diseño cinemático de la estructura robótica para manipular y controlar el sistema. En este documento se realiza una análisis matemático del modelo cinemático que describe la estructura, articulaciones y eslabones, lo que permite la obtención de las coordenadas ”x” ”y” y ”z” del efector final. Se deduce y prueban dos métodos de control. El método tau jerk al que se le integra el método Newton-Rapshon para obtener la cinemática inversa. Además se utiliza método algebraico que es adaptativo y variante en el tiempo cuyo control no requiere matrices cuadradas, así como tampoco la integración de un método de cinemática inversa.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

En la época actual la investigación robótica a evolucionado debido a las necesidades humanas y como pueden ser resueltas mediante la robótica. Cada día la interacción entre humanos y robots se vuelve más común. Elena Garcia, et al. [1] menciona distintas áreas entre las que se encuentran los robots manipuladores los cuales han sido utilizado en procesos productivos desde la década de los 60 y hasta la fecha se sigue investigando para optimizar los cálculos y algoritmos computacionales que permitan el control del efector final.

Ahmed K. Noor [2] propone que los robots se vuelven máquinas autónomas, debido a que anteriormente los robots únicamente desarrollaban tareas para que lo que habían sido programados previamente. En la era digital las tareas que debían realizar resolvían problemas complejos, pero con la entrada de la era cognitiva ahora deben volverse totalmente autónomos, lo que implica que se deban considerar más grados de libertad. Además de sensores y actuadores que permitan realizar y controlar los movimientos que estos robots requieren y por lo tanto la solución es más compleja.

Un manipulador robótico también conocido como robot industrial o brazo robótico es dispositivo electromecánico programable que se utiliza para remplazar o asistir en tareas del ser humano. Un manipulador es una cadena de extremidades rígidas diseñadas para realizar una tarea con su actuador final. Los brazos robóticos pueden tener desde 1 hasta n grados de libertad dependiendo de la complejidad de la tarea a realizar. Los grados de libertad se definen según el tipo de articulación que tenga y los movimientos que se requieren pueden ser tanto lineales y rotacionales. Además, el robótico puede ser equipado con otros mecanismos como sensores y actuadores tener sistemas que permita la retroalimentación con el brazo robótico para realizar de mejor manera la tarea programada mediante un sistema de control.

En la actualidad se han modelados varios sistemas de control que permiten la manipulación de diferentes dispositivos. Desarrollar un sistema de estado estacionario involucra la aplicación de la teoría de control para estabilizar un sistema. Además, si los sistemas de lazo cerrado requieren del uso de sensores que permitan la retroalimentación al sistema, usualmente se utilizan modelos matemáticos para aplicar las teorías de control de manera efectiva. En los últimos años se ha popularizado el uso del control difuso debido a que no requieren modelación matemática compleja. En su lugar, se utilizan modelos basados en el conocimiento de un experto humano, utilizando inferencias y el uso de teoría difusa. Prabhas Ranjan Naik et al[3]. realizaron un estudio comparativo entre un sistema de control difuso y un sistema de control de modo deslizante para un robot de 2 grados de libertad En el estudio se encontró que el modo de control deslizante resulto el mejor entre ambos. Por lo tanto, aún cuando la lógica difusa es una buena herramienta, la abstracción matemática tiene mejores resultados y en problemas complejos la solución tal vez no sea la lógica difusa si no la asistencia de las matemáticas mediante la computación.

Existen diversos métodos en los que se puede aprovechar el procesador mediante el computo paralelo

como el cforest, M. Otte et al. [4] probaron que éste método es efectivo para el control de los manipuladores robóticos. Gracias a estos métodos es posible manipular los brazos robóticos a pesar de la complejidad de los modelos matemáticos e integración de diversos sensores que reducen los tiempos de cálculo.

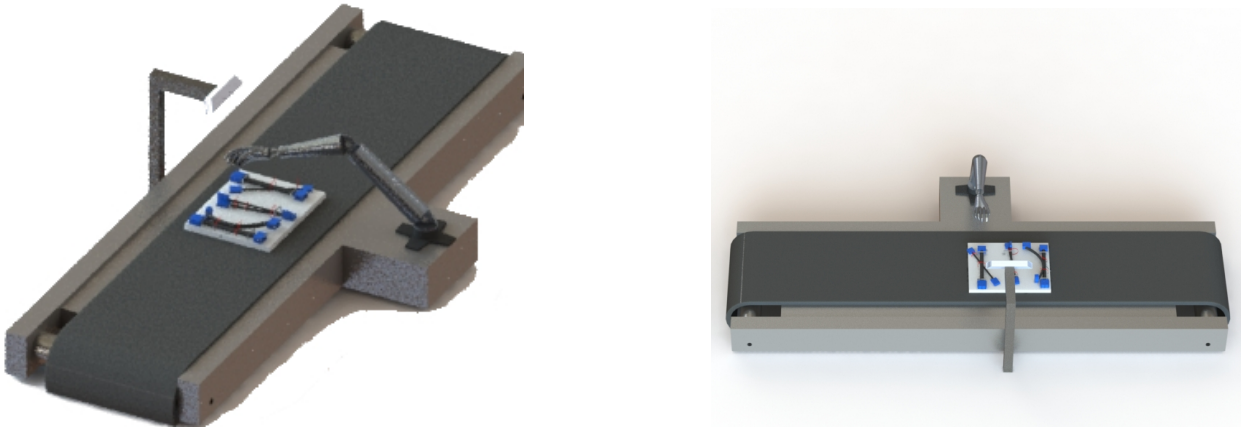
Sin embargo no es el único método utilizado. Los métodos utilizados para el seguimiento de trayectoria depende de diversos factores, los grados de libertad del robot, los sensores integrados y el número de brazos que se controlan . Tsuji, et al. [5] realizó un método capaz de controlar la trayectoria de varios brazos en tareas simples y complejas.

Los manipuladores robóticos permiten integrar diferentes características en las estaciones de trabajo a esto se le conoce como manufactura de robótica flexible [6]. Los manipuladores pueden realizar distintas tareas y aún más si se integran sensores que aumentes sus capacidades e incluso les permitan la autonomía pues actualmente muchos de estos sensores operan a çiegas"por eso se han integrado sistemas de visión que permitan tener una retroalimentación con el ambiente en que operan.

Los sistemas de visión no solo le permiten el reconocimiento del entorno a los robots si no que también les es posible reconocer objetos [7].Lo que permite realizar diferentes procesos en la industrial tales como clasificar objetos, soldar o manipularlos de diferentes maneras.

Uno de los retos para un robot es la generación y seguimiento de trayectorias, esto no solo implica la manipulación de la posición del robot. Además, es pueden existir obstáculos que se deban evadir. Por eso existen diversos métodos que permitan generar una trayectoria entre dos puntos como lo muestra Juan Carlos Restrepo [8] quien realizó un método de generación de trayectoria para basado en modelos geométricos, su método fue utilizado en Robocup un competencia internacional de robótica obteniendo buenos resultados.

Para el desarrollo de este proyecto se realiza un manipulador robótico realice el proceso de manufactura en una empresa maquiladora. Proceso que requiere generar y controlar la trayectoria del manipulador. Además debe ser capaz de detectar la zona donde esta el objetivo como se muestra en la figura 1.1. La detección de la zona es obtenida mediante un sensor de visión. La generación y control de la trayectoria es realizada a partir de un modelo matemático programado en c++ que esta conectado al manipulador robótico y al sensor de visión.



(a) Isometrica (b) Trimetrica

Figura 1.1: Foto de planta

1.1. Planteamiento del problema

En la época actual la robótica ha logrado grandes avances en la industria realizando tareas que anteriormente realizaban los seres humanos de manera mas rápida y eficiente. Hoy se utilizan distintos tipos de robots para resolver diferentes problemas. En específico los manipuladores son utilizados para mover o manejar materiales. Actualmente los manipuladores son programados para realizar una función de manera automática o para ser de manera remota. Sin embargo, con la llegada de la industria 4.0 [9] se prevé nuevos avances que requieren sistemas robóticos más complejos. Las nuevas tareas requieren que el manipulador siga trayectorias mas complejas y con mayor precisión . Para poder ajustarse a esta cuarta revolución industrial se han realizado avances como lo muestra Jonathan Kofman, et al. quienes desarrollaron un manipulador teleoperado de manera remota mediante un sistema de visión [10] permitiendo un control y generación de trayectoria en tiempo real gracias a la retroalimentación. Sin embargo, el manipulador requiere tener a un operador controlando permanentemente y no de manera autónoma.

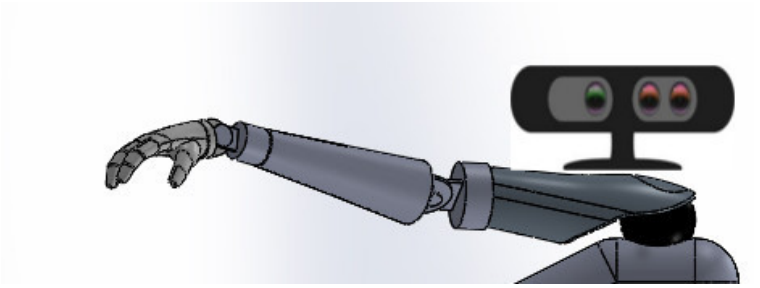


Figura 1.2: Manipulador Robótico

Un manipulador autónomo permite la realizar una tarea sin la necesidad de tener un operador manipulándolo ni tener una rutina previamente programada para conseguir el objetivo es necesario un sistema de visión que pueda reconocer la zona y los objetivos necesarios para realizar una tarea. Dándole visión al robot figura 1.2 tal como lo hace Yn. Yamamoto, et al. con un robot móvil que lleva un manipulador integrado y un sistema de visión que le permite detectar la zona donde debe insertar unos tornillos [11]. Este Robot es capaz de generar y controlar la trayectoria para un brazo robótico de 3 grados de libertad permitiendo realizar una tarea de insertar tornillos de forma autónoma.

Sin embargo, controlar la trayectoria de un manipulador requiere de un modelo matemático que se vuelve más complejo conforme aumentan los grados de libertad lo que también requiere un sistema de control adecuado para controlarlo. Por ejemplo, el control de modo deslizante es comparado con un control PD para un brazo robótico articulado donde el control de modo deslizante prueba reducir mejor el error [12].

Para manipular de manera efectiva un brazo con n grados de libertas que además incluye un sistema de visión es importante aprovechar el cómputo paralelo como lo describe H. Kasahara and S. Narita, [13] aún con los avances en procesadores que se ha tenido. Se requiere aprovechar el computo paralelo en especial si además de los brazos se requiere tener un sistema de visión tal como lo hace J. P. Urban et al. [14]

Un posible solución para generar y controlar la posición de un manipulador con seis grados de libertad mediante un sistema de visión puede ser el realizar un algoritmo a partir de un modelo matemático que permita aprovechar los recurso del cómputo paralelo para manipular el brazo robótico y el sistema de visión en tiempo real.

1.2. Justificación

Los manipuladores robóticos son usados comúnmente en la industria para realizar distintos procesos, por ejemplo moviendo objetos muy pesados para una persona. También pueden reemplazar tareas donde se pone en riesgo al ser humano o en tareas repetitivas. Cada vez se utilizan más en fabricas y con la entrada de la industria 4.0 [9] el trabajo colaborativo entre robots y humanos es cada vez más común tener manipuladores en fabricas realizando distintos procesos y los robots tradicionales pueden representar un riego. Por eso es necesario controlar las trayectorias e integrar sensores en los robots que permitan al robot no trabajar a çiegasz realizar este trabajo de forma segura permitiendo el trabajo colaborativo entre robots y humanos como lo describe S. Robla-Gómez, et al.[15].

Además, hoy en día los manipuladores se usan en distintas áreas de la sociedad. En la industria médica se han desarrollado con éxito varios manipuladores para asistir en cirugías [16] y lo que requiere tener un control preciso de su trayectoria y además poder detectar objetos que le permitan realizar el proceso de manera segura y precisa. Sin embargo, aún se puede seguir trabajando para crear nuevos métodos que permitan la manipulación del robot de manera precisa y que permitan aprovechar mejor los recursos de las computadoras actuales como el que muestra Huang Jiang et al. [17]

1.3. Objetivo general

Desarrollar un sistema que permita generar y controlar la trayectoria de un manipulador de 6 grados de libertad para realizar un proceso de manufactura de manera autónoma. Mediante la detección de su posición con un sistema de visión.

1.3.1. Objetivos particulares

1. Establecer el modelo de derivadas de orden superior de la geometría del movimiento del sistema físico de 6GDL de un manipulador, así como sus restricciones cinemáticas.
2. Desarrollar un sistema de reconocimiento visual con estimación métrica en 3D, e identificación con RNAs recurrentes multicapa, como modelo para generación de trayectorias para ensamble.
3. Deducir un modelo de controlabilidad adaptativo para seguimiento de trayectorias no lineales del manipulador para ensamblaje robusto.
4. Desarrollar el sistema de control, sensado y percepción en tiempo real mediante algoritmos de computación científica.

1.4. Metas

Por la naturaleza multidisciplinaria de los proyectos en robótica las metas a alcanzar se dividen en las distintas áreas de conocimiento que el proyecto requiere.

- Industria
  - Revisión de antecedentes de manipuladores en la industria
  - Conocer el proceso dónde se requiere el manipulador
  - Caracterizar el proceso.
- Robótica
  - Caracterizar Manipular robótico de la empresa

  - Obtener el modelo cinemático de las ecuaciones de posición.
  - Resolver cinemática inversa
  - Programar en c++ la manipulación del robot

- Observador Visual.
  - Seleccionar el mejor sensor de visión para la tarea requerida.
  - Caracterizar el sensor para reconocerlo y manipularlo.
  - Crear la interfaz que permita conectar el sensor con el programa
  - Obtener los datos para el reconocimiento de zona.
  - Entrenar el sistema para reconocer los objetos necesarios
  - Programar en c++ la detección de zona.

- Control
  - Seleccionar el sistema de control adecuado.
  - Modelo matemático del sistema de control.
  - Programar en C++

- Redes Neurales
  - Seleccionar la red neural mas óptima para el proceso.
  - Modelar la Red con el sistema para clasificar objetos.
  - Programar en C++

- Computación
  - Seleccionar las herramientas de computo adecuadas
  - Generar un sistema de tiempo real
  - Programar en c++

1.5. Metodología

En el desarrollo de la robótica se involucran conocimientos en mecánica, electrónica, control e informática por lo que es necesario conocer y tener una amplia literatura previa. No solo en área de robótica si no en el área en que sera aplicada por lo que Kose, Toshihiro et al [18] sugiere una metodología que permite identificar los artículos relevantes para una investigación robótica, haciendo la tarea más efectiva por lo que esta metodología fue utilizada para el desarrollo de este proyecto. Por lo tanto, ésta primera parte del proyecto se enfoca en obtener los conocimientos teóricos y antecedentes necesarios para el resolver le proyecto.

El siguiente paso es el resolver el proyecto que se plantea. En un estudio que se realizó con estudiantes de maestría. Tenían que resolver un problema con un manipulador robótico para una clase de robótico autónoma [19]. Este estudio demostró que el enfoque basado en proyecto tuvo mejores resultados.

Basado en este estudio se pretende resolver este proyecto, dividiéndolo en proyectos prácticos más pequeños que permitan dar solución al problema final. De esta manera al terminar cada pequeña tarea se integran para obtener el objetivo planteado en este documento.

Al final se deberán realizar pruebas y los ajustes necesarios para comprobar la funcionalidad de este brazo. Debido a que este proyecto sera implementado en una empresa maquiladora.

1.6. Alcances

El manipulador debe ser capaz de controlar la trayectoria utilizando los 6 grados de libertad de manera autónoma mediante la detección de zona proporcionada por el sensor de visión.

1.7. Delimitaciones

1. El modelo cinemático del manipulador se establece en 6GDL para modelar al de uso industria.
2. El sistema de ensamble se desarrolla en parte a nivel simulación mediante el uso de un motor de física.
3. Se utiliza un robot de laboratorio en lugar de los robots (Yaskawa) industriales para validación.

1.8. Impacto

En la sociedad actual actual los robots están presentes en distintas áreas de la vida cotidiana e incluso en la virtual [47] lo que indica que la sociedad está preparada para que la robótica sustituya a los seres humanos en diversas actividades. Sin embargo para conseguir remplazar las tareas humanas es necesario desarrollar robots autónomos con la capacidad de realizar tareas cada vez más complejas.

En el sector industrial los robots han incrementado sus aplicaciones y su uso es cada vez mas común. Los manipuladores son tan populares en aplicaciones industrial que en países de primer mundo como Estados Unidos se han realizado estrategias que permitan el desarrollo en inversión en robótica industrial [48]. Gracias a esto no solo se generan mejores procesos que permiten ahorrar costos o incrementar la producción. Si no que también mejorar la calidad y eficiencia del proceso.

En el área académica a pesar de que el modelo de ecuaciones que determinan la posición es un problema resuelto se siguen desarrollando distintos métodos que permitan reducir la complejidad de los algoritmos y permita reducir el tiempo de cálculo en la ejecución de programas. Como lo hace Yesid Veslin, et al. [20] quienes utilizan una meta-heurística que permite resolver el problema o por medio de un sistema de control de modo deslizante como lo muestra L. Huang et al. [17]. Además de aprovechar mejor las capacidades del procesador como lo muestran. Kasahara, et al. [13] y J. P. Urban, et al. [14].

Capítulo 2

Marco Teórico

2.1. Antecedentes

Hoy en día sean construido diferentes tipos de manipuladores. Yusuke Yamamoto, et al. construyeron un robot que es capaz de insertar tornillos, como se ve en la conferencia [11]. El robot consta de un manipulador y realiza a una tarea específica que consiste en insertar un tornillo, y apretarlo enroscarlo en la tuerca. Para esto utiliza un modelo matemático basadas en la posición del tornillo y la ubicación de la tomando en cuenta el diámetro de la misma. Además el robot cuenta con un localizador láser y unas cámaras que le permiten detectar la zona y el diámetro de la perforación. También cuenta con un sensor de presión que permite medir la fuerza conforme se inserta el tornillo evitando que se dañe o se desvié durante el proceso. Para la realización de pruebas se construyó un vehículo, con un brazo robótico (MITSUBISHI-PAlO), con una cámara atada a la pinza, el sensor de fuerza, el localizador láser para la localización de la posición y una cámara tipo estéreo para el reconocimiento de los objetos de trabajo. Se realizaron varias pruebas en donde se siguió una secuencia, la cual fue ejecutada correctamente por el robot. La retroalimentación visual fue exitosa por lo que el robot pudo realizar la secuencia que se le había programado.

Los sensores de visión han sido integrados en numerosas aplicaciones, por ejemplo M. Hu, et al. muestran como utilizar un sensor kinect dentro de un software que es capaz de reconocer los movimientos del cuerpo humano, para que otro usuario tenga que replicarlos, sirviendo como un instructor de baile, o incluso terapeuta físico, como lo vemos en su artículo ”Real-Time Human Movement Retrieval and Assessment With Kinect Sensor” [21] . Donde basándose en modelos matemáticos, extrajo los movimientos y los integro en un esqueleto, que puede programarse con una rutina de movimiento, para que un usuario tenga que imitarlas. Para realizar los experimente realizaron estadísticas de varios movimientos para medir su eficiencia y además realizaron experimentos con individuos de prueba quienes utilizaron el sistema y registrar resultados. Los resultados obtenidos fueron positivos en eficiencia y eficacia, además de una buena aceptación por el público que lo probó, aunque se puede mejorar el algoritmo de control de articulaciones para obtener un sistema más robusto, para trabajos futuros se pretende integrar una base de datos que penalice cuando se detecte una diferencia entre las posiciones.

Existen diferentes tipos de manipuladores industriales, los de tipo paralelo tienen movimientos lineales y usualmente se utilizan para ensamblar objetos. Este año en abril 2020 AUTORPENDIENTE et al. desarrollo un método para evitar singularidades y obtener la trayectoria óptima en cuanto a tiempo, complejidad y jerk. La cual además funciona con múltiples objetivos. Para el modelo cinemático los autores utilizaron el determinante del jacobiano a partir de la matriz de rotación de euler. Después parametrizaron el modelo y con ayuda del un algoritmo genético PSO. obtuvieron la solución para múltiples objetivos. El algoritmo de PSO por si mismo es capaz de obtener solución para múltiples objetivos lo cual fue una ventaja para los autores por que solo fue necesario parametrizar el modelo para obtener resultados sin necesidad de modificar

el algoritmo. Para seleccionar la mejor ruta utilizaron un comprensión difusa. Los autores obtuvieron buenos resultados en cuanto a precisión. Lograron ensamblar la pieza la cual era el objetivo del robot y relucieron el tiempo a una hora a diferencia de las 4 que tomaba con el método que tenia anteriormete. Como trabajo futuro el autor menciono que trabajarían en el análisis de el error, La relación con mi proyecto de tesis es con las generación de trayectorias para múltiples objetivos y como evitar las singularidades que el mismo sistema físico del manipulador presenta.

El reconocimiento de imágenes es parte fundamental de muchos sistemas robóticos que utilizan las cámaras como sensores de visión que permitan relacionarse con su entorno. Para poder realizar esto no solo deben ser capaces de obtener las imágenes además deben procesarlas e interpretar los datos. Una manera de realizar esto es mediante redes neurales, en particular la red hopfield es una red simple monocapa sin enmbargo se ha utilizado en multicapa para reconocer objetos como lo muestra S. S. Young [22]. Otros autores también han utilizado la red hopfield con multicapa donde se utiliza el mismo metodo para reconocer imagenes [23].

2.2. Robótica Industrial

La robótica es una rama de la ingeniería mecatrónica que se encarga de la construcción de artefactos capaces de remplazar una tarea humana llamados robots. De acuerdo con la Real Academia de la lengua española [24] la palabra robot viene de la palabra checa robota", que significa trabajo o prestación personal. Por lo tanto, los robots son asistentes programables diseñados para realizar tareas antes reservadas al ser humano. La robótica puede dividirse en dos categorías la robótica móvil y los manipuladores robóticos. Aunque también existen híbridos que combinan ambas categorías.

La robótica móvil tiene la capacidad de desplazarse a través de su entorno pueden incluir vehículos rodantes, aéreos o que se mueven dando pasos como los biológicamente inspirados. Los manipuladores por otra parte son dispositivos diseñados para mover o manipular materiales siguiendo una trayectoria programada. Son usados comúnmente en la industria para mover materiales muy pesados para el ser humano o para realizar una tarea repetitiva.

Los manipuladores o brazos robóticos son también llamados frecuentemente robots industriales debido a la amplia utilización de estos dispositivos en procesos de manufactura. El incremento de estos dispositivos requiere también mayor seguridad para poder trabajar en conjunto con los seres humanos. Esto hace que los robots industriales requieran sensores que permitan ver su entorno como lo muestran S. Robla-Gómez et al. [25] en un articulo done estudian este fenómeno.

Los manipuladores robóticos artefactos unidos en secuencia y comúnmente inspirados en brazos humanos consta de diferentes partes. El punto fijo o base donde se apoya el resto del manipulador, las partes rígidas son llamadas eslabones y definen el largo del manipulador y por lo tanto su espacio de trabajo. Las partes que unen a los eslabones son llamadas articulaciones y pueden ser de distintos tipos. Los manipuladores pueden tener tantos eslabones y articulaciones como sea necesario. Además, tienen un actuador final que suele ser una pinza o cualquier herramienta que sea necesaria para cumplir con su tarea como se muestra en la figura 2.1.

Las articulaciones pueden dividirse en dos según el movimiento que permiten al robot pueden ser movimiento linear ó rotacional. Además cada una permite diferentes GDL (grados de libertad) como se muestra en la figura 2.2

Se pueden realizar distintos tipos de configuraciones usando distintas articulaciones.

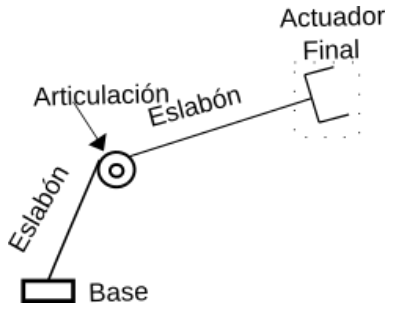


Figura 2.1: Manipulador Robotico

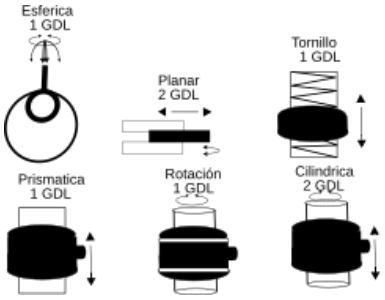


Figura 2.2: Tipos de articulaciones

2.3. Cinemática Directa

La cinemática es una rama de la mecánica encargada de estudiar el movimiento de un cuerpo sin tomar en cuenta las causas que lo producen en robótica se utiliza para generar las trayectorias del actuador final.

Una robot puede ser subactuado cuando tiene menos variables de control que de resultados, totalmente actuado cuando tiene las mismas variables de control y redundantemente actuados cuando son mas variables de control que de resultados. Los que resulta en matrices no cuadradas al momento de realizar sus cálculos. Estos manipuladores requieren entonces métodos de control que permitan estos cálculos. Además en algunas aplicaciones como un robot quirúrgico [30] se requiere de mucha presición.

El método mas popular para la solución cinemática de cadenas articuladas es el de Denavit-Hartenberg. El método se describe en los libros de robótica [31] se basa en la multiplicación de una matriz de transformación T para cada articulación.

$$A = \prod_{l=N}^{0} T \tag{2.1}$$

La matriz de transformación o también llamada matriz homogénea se compone de 4 submatrices la matriz de rotación, traslación, perspectiva y escala. Obteniendo una matriz 3x3 en el plano o 4x4 en el espacio. En el método Denavit-Hartenberg se utiliza de la siguiente manera.

$$T(3x3) = \begin{bmatrix} R_{2x2} & r_{2x1} \\ P_{1,2} & E_{1x1} \end{bmatrix}, T(4x4) = \begin{bmatrix} R_{3x3} & r_{3x1} \\ P_{1,3} & E_{1x1} \end{bmatrix} \tag{2.2}$$

La submatriz de rotación en el espacio 3x3 o 2x2 en el plano se compone de las coordenadas cartesianas

X, Y, Z para cada articulación en el manipulador.

$$R(2x2) = \begin{bmatrix} xx & xy \\ yx & yy \end{bmatrix}, R(3x3) = \begin{bmatrix} xx & xy & xz \\ yx & yy & yz \\ zx & zy & zz \end{bmatrix} \tag{2.3}$$

La matriz de rotación representa la posición cartesiana del eje del actuador respecto al eje original fig 2.3.

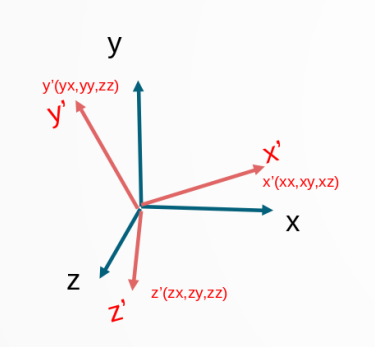


Figura 2.3: Rotación

Sin embargo, visualmente es difícil ver la orientación cuando es representada de esta manera. Sabemos por ejemplo que cuando los valores de las coordenadas x, y o z coinciden como por ejemplo.

$$R = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, R = \begin{bmatrix} 0 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix}, R = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix} \tag{2.4}$$

Aquí fácilmente podemos determinar que en la primera matriz el manipulador esta volteado 180 ^{o} sobre el eje x (-1 en el circulo unitario), así mismo en Y y Z donde hay un -1 nos dice que ha girado 180 grados respecto al eje de coordenadas original. En la coordenada correspondiente (X, Y Z). Pero esto se complica si la orientación del manipulador no esta dada en ángulos rectos o de 45 grados. Por lo que para determinar la orientación es preferible recurrir a los ángulos de Euler.

Los ángulos de Euler son los ángulos formados entre los dos ejes de coordenadas. A estos angulos se les conoce como α el angulo formado entre los ejes X, β el angulo formado entre los ejes Z y γ el angulo formado entre los ejes Z fig 2.4

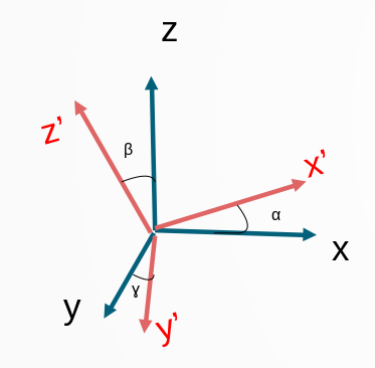


Figura 2.4: Angulos de Euler

Estos ángulos los podemos obtener partiendo de la matriz de rotación multiplicandola por el vector de posicion.

$$\begin{bmatrix}\alpha\\ \beta\\ \gamma\end{bmatrix} = \begin{bmatrix}xx & xy & xz\\ xy & yy & yz\\ xz & zy & zz\end{bmatrix} . \begin{bmatrix}X\\ Y\\ Z\end{bmatrix} \tag{2.5}$$

La submatriz de traslación determina comúnmente el largo del eslabon de cuya articulación se este analizando.

$$r(2x1) = \begin{bmatrix}x\\ y\end{bmatrix}, r(3x1) = \begin{bmatrix}x\\ y\\ z\end{bmatrix} \tag{2.6}$$

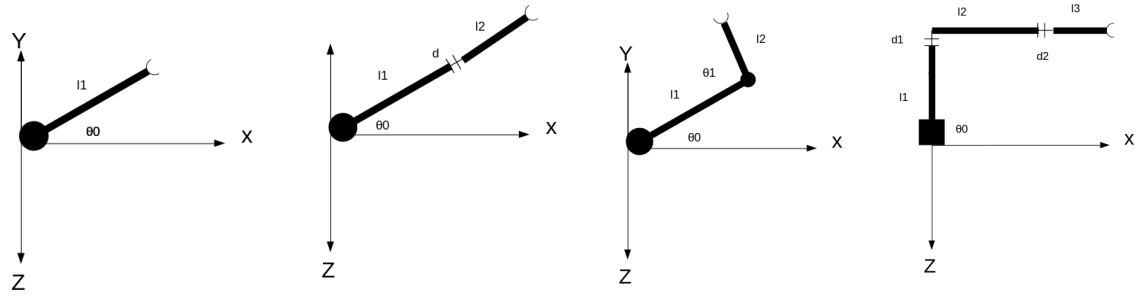
La submatriz de perspectiva en una matriz homogénea determina la deformación del objeto a analizar y la escala el tamaño del mismo para el análisis cinemático de rebotica no es necesario debido a que nuestro objeto es un solido en la vida real el cual tiene su tamaño y forma definido. Por lo tanto la matriz de perspectiva es de 0 (1x2 en el plano o 1x3 en el espacio) y la submatriz de escala es un escalar con valor de 1 (100 %). Aun cuando estas matrices no cambian permiten la formación de la matriz homogenea que es una matriz cuadrada y permite la multiplicación directa entre las matrices que se van a formar.

$$P(1x2) = \begin{bmatrix}0 & 0\end{bmatrix}, P(1x3) = \begin{bmatrix}0 & 0 & 0\end{bmatrix} \tag{2.7}$$

$$E = 1 \tag{2.8}$$

2.4. Denavit-Hartenberg

Para realizar el metodo Denavit-Hartenberg primero se crea un punto de referencia local Q_i con sus 3 ejes ortogonales (X,Y,Z) para cada articulación del robot Además se crea una tabla con sus parámetros el numero de la articulación a_iS los ángulos respecto x α articulaciones prismáticas d_l ángulos respecto a y θ . En la imagen 2.5 se muestran 3 ejemplos



(a) 1 (b) 2 (c) 3 (d) 4

Figura 2.5: Ejemplos Manipuladores

Ej	articulacion	a_i	α	d_i	θ
# 1	1	l1	0	0	θ_0
# 2	1	l1	0	0	θ_0
	2	0	0	d	0
# 3	1	l1	0	0	θ_1
	2	l2	0	0	θ_2
# 4	1	0	0	0	θ_0
	2	0	0	d1	0
	3	0	0	d2	0

Tabla 2.1: Tabla Denavit Hartenberg

La matriz de rotación se obtiene con la regla de la mano derecha para estos ejercicios los ejes de torsión giran sobre y de tal forma que nuestra matriz de rotación es.

$$R_y \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.9}$$

La matriz r_{Q_i} mediante la solución geométrica de cada articulación en el ejemplo 1.

$$r = \begin{bmatrix} l\cos\theta \\ l\sin\theta \\ 0 \end{bmatrix} \tag{2.10}$$

Y la matriz T Dado que no existe otra articulación la solución es el resultado final $x = l * \cos\theta$ y $x = l * \sin\theta$ para la posición.. Si tomamos el ejercicio 3 tenemos que $A = \prod_{l=N}^{0} T$ como tenemos 2 articulaciónes entonces $A = T_1 T_2$ por l tanto

$$T_1 = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & l\cos\theta_0 \\ \sin\theta & \cos\theta & 0 & l\sin\theta_0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, T_2 = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & l\cos\theta_1 \\ \sin\theta & \cos\theta & 0 & l\sin\theta_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.11}$$

$$r = \begin{bmatrix} \cos\theta_0\cos\theta_1 l_2 - \sin\theta_0\sin\theta_1 l_2 - l_1\cos\theta_0 \\ \sin\theta_0\cos\theta_1 l_2 - \cos\theta_0\sin\theta_1 l_2 - l_1\sin\theta_0 \\ 0 \end{bmatrix} \tag{2.12}$$

La matriz T obtenida al final también obtiene la matriz de rotación del sistema que es la orientación del actuador final del manipulador expresada con una matriz de 3x3. La cual podemos transformarla en ángulos de Euler o dejarla expresada como matriz según nos convenga.

2.5. Cinemática Inversa

La cinemática directa estudia el movimiento por tanto la cinemática inversa trata de determinar la posición original que causo ese movimiento. Para esto existen diferentes métodos en manipuladores robóticos se pueden utilizar métodos geométricos que utiliza el teorema del arcotangente para encontrar la solución sin embargo dado que las ecuaciones se forman por la sudatoria de ángulos dentro de funciones de seno y coseno. El numero máximo de grados de libertad que se pueden resolver con este método son 3. También pueden determinar con el desacoplando las articulaciones del robot. Sin embargo, también existen métodos iterativos que se basan en análisis computacional que permite encontrar el resultado pueden ser metaheuristicas, redes neurales o algún método matemático como Newton Rapson o basándose en el Determinante Jacobiano.

Uno de los métodos mas simples es el método geométrico, si partimos de la idea mas simple con un manipulador de 1 grado de libertad tenemos que la posición XY esta dada mediante el teorema de pitagoras como se ve en la figura 2.9.

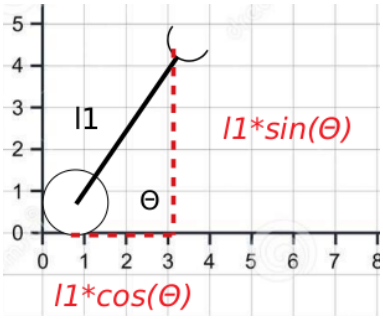
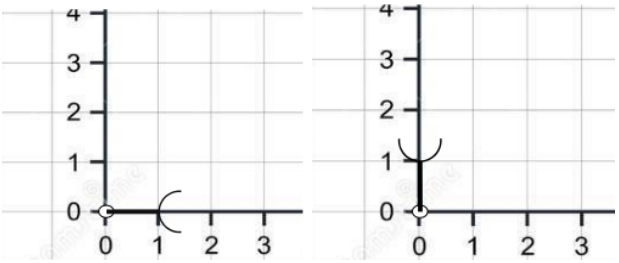


Figura 2.6: Cinematica 1 GDL

Por lo tanto tenemos que la tangente esta formada por la posición XY del manipulador esto permite obtener el angulo de dicho manipulador por medio del arcotangente de los catetos.

$$\phi = atan = \frac{Y}{X} \tag{2.13}$$

Podemos comprobar esto de manera muy simple sabiedo que en reposo su posicion en X es igual al largo del manipulador y a 90 grados es el total del largo del manipulador en el eje Y como se ve en la figura 2.7.



(a) Reposo (b) 90 ^{o}

Figura 2.7: Cinematica inversa

De esta manera puedo obtener el angulo al que debo mover el manipulador por ejemplo si deseo quiero mi manipulador a la posición (0.7,0.7) obtengo el angulo de 45 grados o $\frac{\pi}{4}$ como se ve en la figura

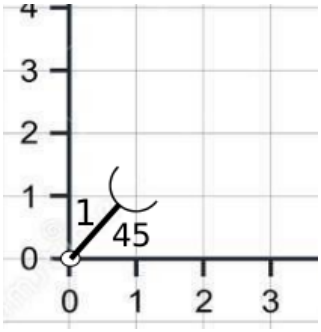


Figura 2.8: Cinemática Inversa 45 ^{o}

Esto solo funciona para 1 grado de libertad si tenemos por ejemplo un manipulador en el espacio de 3 grados de libertad tenemos que usar algunas leyes trigonométricas que nos permitan encontrar los ángulos en la figura se muestra un manipulador en donde para encontrar ϕ_0 podemos usar el arcotangente tal cual el ejemplo anterior.

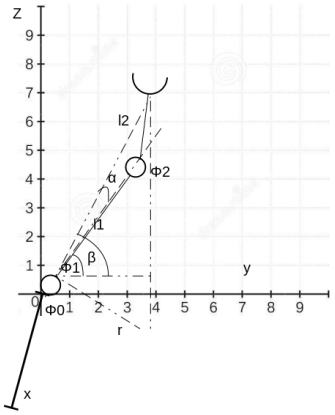


Figura 2.9: Cinemática Inversa 3 GDL

Podemos obtener ϕ_1 y ϕ_2 mediante el análisis trigonométrico obteniendo la resta de los angulo β y α

$$\phi_1 = \beta - \alpha \tag{2.14}$$

$$\beta = \frac{Pz}{r} = atan\frac{Pz}{\pm\sqrt{Px^2 + Py^2}} \tag{2.15}$$

$$\alpha = atan(\frac{l2\sin\phi_2}{l1 + l2\cos\phi_2}) \tag{2.16}$$

De manera similar podemos obtener ϕ_3 sin embargo este es el máximo numero de grados de libertad que podremos obtener mediante este método geometrico. Para obtener mas grados de libertad debemos utilizar otros métodos como puede ser por polinomios, desacoplamiento cinemático, por medio de la matriz homogénea o con análisis numérico.

2.6. Teoría General Tau

La teoría General de Tau, el artículo menciona que hasta la década de los 80's solo se habían concentrado en la información visual, pero que se podía generalizar para todos los sistemas de perceptivos. Cada sistema recogía un τ y existe una brecha (distancia, ángulos fuerza, etc). De esta manera se puede resumir la teoría general de tau.

- Brecha de acción: Cada movimiento implica un cierre entre cada acción una separación entre el estado actual y el objetivo.
- τ la medida de una brecha de acción: El tiempo del cierre de la brecha de acción al ritmo actual. Es un control que puede venir de cualquier percepcion sensorial.
- τ acoplamiento. Ésta asegura que el cierre entre dos τ terminen simultáneamente. La ecuación es : $\tau_x(t) = K_{x,y}\tau_y$ donde X , Y se mantienen en un radio constante $K_{x,y}$

- Detección a través del acoplamiento: Sirve para la especificación sensorial cuando la forma es: $S(t) = CX(t)$. La ecuación es: $\tau_x(t) = K_{x,s}\tau_y(s)$
- Guía de movimiento intrínseco τ_G: Se describió la fase de desaceleración en la brecha de acción cuando la b es constantemente guiada por la ecuación $\dot{\tau}_{(x)} = b$ que es apropiado cuando existe una desaceleración prolongada como en el ejemplo de un conductor y el frenado. Sin embargo, existen ocasiones en que el movimiento proviene de un estado de reposo. El cual tiene una desaceleración inmediata y vuelve al reposo. La teoría general $/tau$ dice que estos movimientos son intrínsecamente guiados por un acoplamiento siguiendo la ecuación. $\tau_x(t) = K_{x,G}\tau_y(G)$

En resumen, la teoría general de τ modela la manera en que se perciben y realizan acciones en el mundo de manera natural. Donde la acción se realizará basándose en la información sensorial y calculando la brecha de acción en un objetivo y su meta mediante el ritmo τ en que este llega meta.

2.6.1. Método Tau

De acuerdo con el artículo [49] la teoría Tau y la teoría general. Nació en Ittaca entre los años 1969 y 1970 en el laboratorio de Jimmy y Jackie Gibson's. La teoría General de tau es una reflexión de como percibimos y actuamos en el mundo de manera natural, lo que posteriormente inspiró a desarrollar la matemática de estas acciones y percepciones. La fórmula de $\dot{\tau}_{(x)} = b$, en un ejemplo simple donde el conductor de un vehículo se detiene en un obstáculo. Como en el artículo [50], b es la constante de desaceleración, si la $b <= \frac{1}{2}$ en vehículo se detendrá antes de llegar al obstáculo pero si $b > \frac{1}{2}$ ocurrirá una colisión.

2.6.2. Teoria General Tau

La teoría general de tau refiere a que no el Tau no solo describe la percepción visual si no que también describe la manera de se mueven los cuerpos biológicos. La estrategia de acoplamiento describe el movimiento de dos cuerpos por ejemplo el de un futbolista y el balón este acoplamiento se describe mediante la formula $\tau_q = k_{q,p}\tau_p$ donde k es una constante de acoplamiento y p y q es la diferencia del estado inicial y final de la brecha. Esta teoria ha sido utilizada durante varios años con buenos resultados [53].

2.6.3. Acoplamiento Tau

El acoplamiento hace referencia a la manera en que dos objetos se juntan durante un movimiento se puede calcular mediante la ecuación de acomplamiento de $\tau_q = k_{q,p}\tau_p$.

2.7. Visión Artificial

Los colores que un ser humano percibe se forman por la reflexión de la luz sobre los cuerpos quienes absorben parte de esa luz y reflejan el resto, la luz que se refleja es la que causa el color que vemos. Sin embargo, no podemos percibir toda todo tipo de luz solo la que se encuentra dentro de la longitud de onda que los ojos pueden captar lo que se conoce como espectro visible. Los colores primarios no están relacionados con la física del espectro de luz si no con la biología del ojo el cual contiene tres receptores que le dan forma a los colores primarios que conocemos, Rojo, verde y azul Red", "Green", "Blue.[en] ingles el resto de los colores es una mezcla entre los colores primarios.

Las imágenes son obtenidas mediante cámaras existen distintos tipos de cámara, para obtener imágenes a color, profundidades, calor, etc. Para una cama RGB se obtienen los datos de estas cámaras es necesaria su representación en píxeles, que son cuadros de un solo color que contiene. De tal forma que se obtiene una

matrices con todos los píxeles que contiene. El rango mas común de cada color es el de 8 bits (0-255), por lo que una imagen en escala a grises es un vector con rango [0-255]. Mientras que una imagen a color esta formada por los vectores RGB([0-255] ,[0-255], [0-255]). Por lo que si se gráfica se puede obtener un cubo que representa el color.

Cubo RGB El cubo RGB forma los colores RGB donde los ejes del plano son los colores RGB con un rango de [0-255], de tal forma que se representar un color en la imagen 2.10 se puede apreciar un punto dentro del cubo RGB que sera la coordenada[R,G,B].

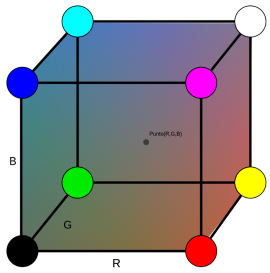


Figura 2.10: Cubo RGB

2.7.1. Imágenes Digitales

La visión artificial en un robot esta dada mediante diferentes sensores y cámaras que permitan la inspección y reconocimiento de objetos o del ambiente mismo del robot. A este conjunto de sensores y cámaras se le conoce como sistema de visión. Su funcionamiento básico es la adquisición y procesamiento de imágenes que son enviadas a un sistema que pueda interpretar órdenes a partir de estos datos.

En los manipuladores los sistemas de visión permiten un lazo cerrado con la posición en que se encuentra en efectos final del manipulador. También les permite reconocer objetos y realizar tareas que se les programen como la intersección de objetos en tiempo real. [35].

Uno de los sistemas de visión más cotidianos es el de microsoft el sensor kinect usado en videojuegos cuenta con tres tipos de cámaras distintas una cámara óptica, una cámara RGB y una cámara de profundidad las cuales en combinación logran captar una imagen del cuerpo humano y extraer un esqueleto del cual pueden extraerse los movimientos para poder utilizarlos como comandos en un programa.

2.7.2. Cámaras RGB-D

Los sensores RBG-D son dispositivos que sirven para capturar la luz del ambiente y transmite las imágenes en los colores primarios rojo, verde y azul RGB"por sus siglas en ingles y además captura profundidad mediante un sensor infrarrojo que permite capturar la profundidad "Deep"mediante un haz infrarroja que transmite para que rebote y regrese a un transmisor como su nombre lo indica esta luz esta por debajo de la luz roja en el prisma de colores por lo que se encuentra fuera del espectro visible del ser humano. Existen son muy utilizadas con distintos propósitos desde sensores de visión hasta videojuegos. Siendo, uno de los mas comunes el sensor de kinect de microsoft, al cual también se le han dado aplicaciones científicas gracias que son de código abierto.

A pesar de que el sensor kinect ha sido utilizado en diferentes proyectos de investigación y que sus características resultan eficientes en los proyectos además de tener un SDK libre para uso en desarrollo científico [36] Otros sensores más nuevos han surgido como el que presenta ASUS véase figura **??**. De

acuerdo con su página de internet [37] cuenta con cámara RGB, cámara de profundidad y dos micrófonos las cámaras de profundidad son VGA (640x480) : 30 fps, QVGA (320x240): 60 fps. Además se puede trabar en diferentes plataformas con C++/C# o Java. Estas características han permitido que también sea un buen recurso para el desarrollo de investigaciones.

2.8. Redes Neurales Artificiales

Las redes neuronales artificiales se desarrollaron en la década de 1940 Son modelos computacionales basadas en la manera en que las neuronas biológicas funcionan.Una de las principales aplicaciones de las ANN es en el campo de predicción. Estas ANNs Algunas aplicaciones de la predicción por redes neuronales son: predecir el clima en un período, el comportamiento del nivel de venta de mercancías o mantener el stock, la predicción de razones financieras. Se basan en el reconocimiento de patrones para un aprendizaje mediante una matriz denominada matriz de pesos el cual puede ser supervisado si se le asignan los patrones iniciales que deben aprender o no supervisados si la matriz de pesos se actualiza a si misma. Existen diferentes tipos de redes.

Perceptrón Simple

Es la red mas simple corresponde a una capa de neuronas con entrada y salida. Se compone de las entradas, la sumatoria de pesos y la función de activación.

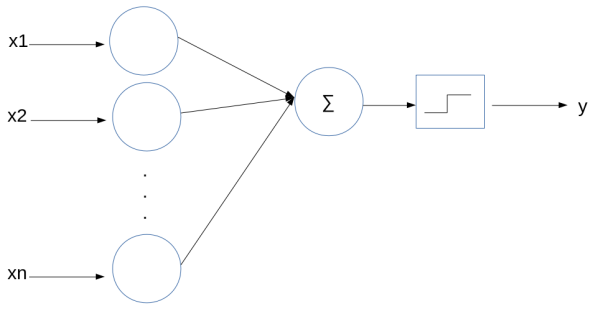


Figura 2.11: Perceptrón Simple

Redes Recurrentes

La red recurrente en donde la salida se retro-alimenta con si misma. Igual que en el perceptron simple requiere una sumatoria para los pesos y una función de activación.

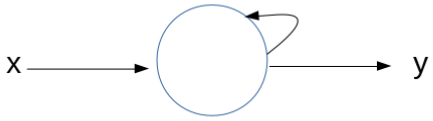


Figura 2.12: Redes Recurrentes

Multicapa

Una red multicapa de la capa con las neuronas de entrada, la capa oculta y una capa de salida.

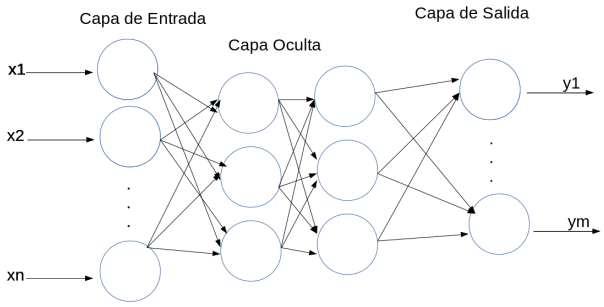


Figura 2.13: Multicapa

El Dr. Rircardo Rodriguez[32] afirma que en las ANN existe cierta incertidumbre. Para resolver esto utilizó métodos de predicción con unidad lineal LNU, La unidad neural cuadrática (QNU) y la unidad neural cúbica (CNU).

2.9. Red Hopfield

La red hopfield es una red recurrente diseñada por John Hopfield en la década de los 80. es una red recurrente mono capa con valores binarios que permite reconocer patrones2.14. En el cada neurona y cada salida están conectadas a través de pesos sinápticos.

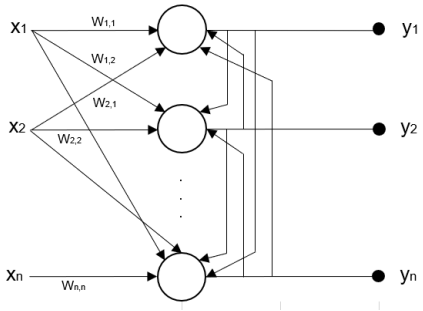


Figura 2.14: Red Neuronal Hopfield

2.9.1. Entrenamiento

La arquitectura de la red de Hopfield esta formada por una capa con todas las neuronas conectadas entre si, cuya informacion viaja de ida y vuelta. Las entradas $x_i(t)$ son binarias ó polares $(\{0,1\}o\{-1,1\})$, y las salidas que en principios llamaremos $y_i(t)$, también binarias ó polares.

El cálculo de los pesos consiste en determinar la matriz de pesos W por medio de multiplicación de matrices.

$$W_i j = \sum_{i=0}^{M-1} x_i^T \cdot x_j \tag{2.17}$$

Podemos utilizar una función ”step”para convertir los valores a binario de tal forma que tenemos dos únicos posibles valores poniendo un limite a los valores análogos.

$$xn_i = \begin{cases} 1 & \text{si } es \text{ mayor al limite} \\ 0 & \text{de lo contrario} \end{cases} \tag{2.18}$$

Capítulo 3

Análisis y Modelado Cinemático Redundante

La robótica es un área multidisciplinaria que requiere del estudio en integración de distintas ciencia según el proyecto que se este desarrollando. Para este proyecto se requiere generar y controlar la trayectoria y posición de un brazo robótico. Por lo que las ciencias que se estudiaran incluyen la cinemática del manipulador, la ingeniería de control y los sistemas de visión.

3.1. Diseño Mecánico

Si tomamos un manipulador simple con una articulación rotacional y lo colocamos sobre un eje cartesiano. Podemos definir su posición en x , y mediante el teorema de pitagóricas como se muestra en la figura 3.1. Donde el largo del eslabón l_1 es la hipotenusa y define el tamaño del manipulador mientras que la articulación define el tipo de movimiento que tendrá en este caso el manipulador puede rotar sobre el plano xy. Por lo que las ecuaciones de posición para este brazo seria $x = l_1 \cos(\phi_0)$, $y = l_1 \sin(\phi_0)$

Vemos que la ecuación es similar a un sistema de coordenadas polares donde el eslabón es el radio y cada articulación es el polo.

Si se siguen agregando más eslabones con articulaciones rotacionales de un grado de libertad sobre el mismo plano se deben sumar los eslabones tomando en cuenta los ángulos sobre los cuales van a rotar y los ángulos previos para determinar la posición del actuador final como se muestra en la figura 3.1. Por lo que las ecuaciones se definen por $x = l_1 \cos(\phi_0) + l_2 \cos(\phi_0 + \phi_1) + l_3 \cos(\phi_0 + \phi_1 + \phi_2)$, $y = l_1 \sin(\phi_0) + l_2 \sin(\phi_0 + \phi_1) + l_3 \sin(\phi_0 + \phi_1 + \phi_2)$

Para obtener un movimiento en un espacio tridimensional se puede utilizar una articulación esférica. Para realizarlo se hace una proyección sobre el plano trazando recto en el actuador final como se ve en la figura 3.1. Así se obtiene una L que permite calcular la posición del actuador final mediante $x = L \sin(\phi_0)$, $y = L \cos(\phi_0)$ y z se forma de los ángulos que definen la altura $z = l_1 \cos(\phi_1)$ donde la L se resolviendo de la trigonometría que involucra afecta el actuador final. Para este problema no se tienen otras articulaciones o eslabones así que la L será igual a $l_1 \cos(\phi_1)$ teniendo entonces $x = l_1 \cos(\phi_1) \sin(\phi_0)$, $y = l_1 \cos(\phi_1) \cos(\phi_0)$ y $z = l_1 \cos(\phi_1)$ si se tienen más articulaciones esféricas se continua de la misma manera trazando $L's$ sobre planos imaginarios en cada articulación esférica. Si se combinan con articulaciones rotacionales con en la figura 3.1. Solo es necesario modificar la L y la agregar las rotaciones que afecten la altura en z. $x = L \sin\phi_0 = (l_1 \cos(\phi_1) + l_2 \cos(\phi_1 + \phi_2) + l_3 \cos(\phi_1 + \phi_2 + \phi_3)) . \sin\phi_0$, $y = L \cos\phi_0 = (l_1 \cos(\phi_1) + l_2 \cos(\phi_1 + \phi_2) + l_3 \cos(\phi_1 + \phi_2 + \phi_3)) . \cos\phi_0$ y $z = (l_1 \sin(\phi_1) + l_2 \sin(\phi_1 + \phi_2) + l_3 \sin(\phi_1 + \phi_2 + \phi_3)$

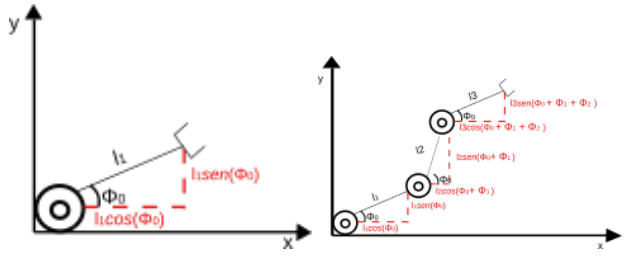
Para obtener torsión se puede girar alguna de las articulaciones de tal forma que queden paralela al es-

labón. Si se coloca en el actuador final no tiene afecta las ecuaciones de movimiento. Pero si se tienen otras articulaciones la ecuación se ve afectada por la torsión. Para darle solución se utiliza la matriz de rotación dependiendo del eje donde se requiera en la figura

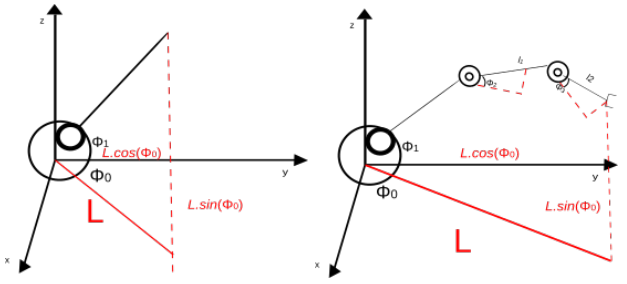
La ecuación si no se tomara en cuenta la torsión nos queda de la misma manera que la figura 3.1 pero tiene una torsión en $\phi 3$ que afecta al eslabón l_3 por lo tanto la ecuación para resolverlo es $P = Roty\phi_3 P_{l_3}$

$$\mathbf{P} = Roty\phi_3 \cdot \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} \cos\phi_3 & 0 & \sin\phi_3 \\ 0 & 1 & 0 \\ -\sin\phi_3 & 0 & \cos\phi_3 \end{bmatrix} \cdot \begin{bmatrix} l_3\cos(\phi_1+\phi_4+\phi_6))(\sin\phi_0) \\ l_3\cos(\phi_1+\phi_4+\phi_6))(\cos\phi_0) \\ l_3\sin(\phi_1+\phi_4+\phi_6) \end{bmatrix}$$

$$\mathbf{P} = \begin{bmatrix} l_3\cos(\phi_1+\phi_4+\phi_6))(\sin\phi_0)(\cos\phi_5) - l_3\sin(\phi_1+\phi_4+\phi_6)\sin\phi_5 \\ l_3\cos(\phi_1+\phi_4+\phi_C6))(\cos\phi_0) \\ l_3\cos(\phi_1+\phi_4+\phi_6))(\sin\phi_0)(\sin\phi_5) + l_3\sin(\phi_1+\phi_4+\phi_6)(\cos\phi_5) \end{bmatrix}$$



(a) 1 GDL (b) 3 GDL



(c) Esférica (d) Torsión

Figura 3.1: Manipuladores

El resultado se suma al eslabón que se ve afectado por la torsión. De tener mas articulaciones en torsión se debe seguir realizando los mismos pasos. Las ecuaciones para este manipulador quedan entonces de la siguiente manera.

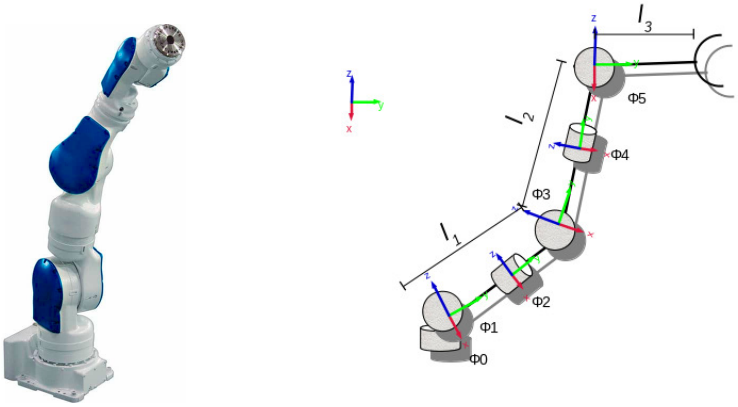
$x = (l_1\cos(\phi_1).\sin\phi_0 + l_2\cos(\phi_1+\phi_2).\sin\phi_0 + l_3\cos(\phi_1+\phi_4+\phi_6))(\sin\phi_0)(\cos\phi_5) - l_3\sin(\phi_1+\phi_4+\phi_6)\sin\phi_5.\sin\phi_0$

$y = (l_1\cos(\phi_1) + l_2\cos(\phi_1+\phi_2) + l_3\cos(\phi_1+\phi_2+\phi_3)).\cos\phi_0$

$z = (l_1\sin(\phi_1)\cos\phi_0 + l_2\sin(\phi_1+\phi_2)\cos\phi_0 + l_3\cos(\phi_1+\phi_4+\phi_6))(\sin\phi_0)(\sin\phi_5) + l_3\sin(\phi_1+\phi_4+\phi_6)(\cos\phi_5)$

3.2. Modelo Cinemático

El manipulador que se va a utilizar se muestra en la 5.1, es uno de los manipuladores mas comunes en la industrial debido a que asemeja a un brazo humano.



(a) Yaskawa SIA20D (b) Diagrama 6 DOF

Figura 3.2: Manipulador 6 DOF

Para obtener las ecuaciones de movimiento primero se encuentran las ecuaciones sin tomar en cuenta la torsión.

$$x = L\sin\phi_0 = (l_1\cos(\phi_1) + l_2\cos(\phi_1+\phi_2) + l_3\cos(\phi_1+\phi_2+\phi_3)).\sin\phi_0$$

$$y = L\cos\phi_0 = (l_1\cos(\phi_1) + l_2\cos(\phi_1+\phi_2) + l_3\cos(\phi_1+\phi_2+\phi_3)).\cos\phi_0$$

$$z = (l_1\sin(\phi_1) + l_2\sin(\phi_1+\phi_2) + l_3\sin(\phi_1+\phi_2+\phi_3)$$

Si se agrega torsión en $\phi 2$ y $\phi 4$ que afecta al eslabón l_2 y l_3 respectivamente, primero se obtiene la ecuación sin tomar en cuenta la torsión obteniendo las ecuaciones de posición sin torsión 3.1.

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} x = (l_1\cos(\phi_1) + l_2\cos(\phi_1+\phi_2) + l_3\cos(\phi_1+\phi_2+\phi_3)).\sin\phi_0 \\ y = (l_1\cos(\phi_1) + l_2\cos(\phi_1+\phi_2) + l_3\cos(\phi_1+\phi_2+\phi_3)).\cos\phi_0 \\ z = (l_1\sin(\phi_1) + l_2\sin(\phi_1+\phi_2) + l_3\sin(\phi_1+\phi_2+\phi_3) \end{bmatrix} \tag{3.1}$$

Los ángulos rotacionales que permiten la torsión ϕ_2 y ϕ_3 giran sobre el eje y, por lo tanto la matriz de rotación necesaria es la que muestra la ecuación 3.2.

$$R_y(\phi) = \begin{bmatrix} \cos\phi & 0 & \sin\phi \\ 0 & 1 & 0 \\ -\sin\phi & 0 & \cos\phi \end{bmatrix} \tag{3.2}$$

Para resolverlo, se multiplica la matriz de rotación en cada torsión solo para los eslabones que se ven afectados. Para la torsión del ángulo ϕ_4 que afecta al eslabón l_3.

$$\mathbf{p} = R_y(\phi_{32} * P_{l_2}$$

$$\mathbf{p} = R_y(\phi_4)\cdot\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} \cos\phi_4 & 0 & \sin\phi_4 \\ 0 & 1 & 0 \\ -\sin\phi_4 & 0 & \cos\phi_4 \end{bmatrix}\cdot\begin{bmatrix} l_3\cos(\phi_1+\phi_3+\phi_5))(\sin\phi_0) \\ l_3\cos(\phi_1+\phi_3+\phi_5)(\cos\phi_0) \\ l_3\sin(\phi_1+\phi_3+\phi_5) \end{bmatrix}$$

$$\mathbf{p} = \begin{bmatrix} l_3\cos(\phi_1+\phi_3+\phi_5))(\sin\phi_0)(\cos\phi_4) + l_3\sin(\phi_1+\phi_3+\phi_5)\sin\phi_4 \\ l_3\cos(\phi_1+\phi_3+\phi_C5))(\cos\phi_0) \\ -l_3\cos(\phi_1+\phi_3+\phi_5))(\sin\phi_0)(\sin\phi_4) + l_3\sin(\phi_1+\phi_3+\phi_5)(\cos\phi_4) \end{bmatrix}$$

El resultado se suma al eslabón que se ve afectado por la torsión en ϕ_3. teniendo. $\mathbf{p} = R_y(\phi)_2 \cdot \begin{bmatrix} x \\ y \\ z \end{bmatrix} =$

$$\begin{bmatrix} C_{\phi_2} & 0 & S_{\phi_2} \\ 0 & 1 & 0 \\ -S_{\phi_2} & 0 & C_{\phi_2} \end{bmatrix} \cdot \begin{bmatrix} l_2 C_{(\phi_1+\phi_2)}(S_{\phi_0}) + l_3 C_{(\phi_1+\phi_3+\phi_5)})(S_{\phi_0})(C_{\phi_4}) + l_3 S_{(\phi_1+\phi_3+\phi_5)} S_{\phi_4} \\ l_2 C_{(\phi_1+\phi_2)}(S_{\phi_0}) + l_3 C_{(\phi_1+\phi_3+\phi_C 5))}(C_{\phi_0}) \\ l_2 S_{(\phi_1+\phi_2)} + -l_3 C_{(\phi_1+\phi_3+\phi_5)})(S_{\phi_0})(S_{\phi_4}) + l_3 S_{(\phi_1+\phi_3+\phi_5)}(C_{\phi_4}) \end{bmatrix}$$

Por lo tanto la ecuaciones que describen el movimiento de este manipulador de 6 grados de libertad estan dadas por las ecuaciones 5.5, 5.6 y 5.7

$$\begin{aligned} x =& l_1 C_{(\phi_1}.S_{(\phi_0} + l_2(C_{(\phi_1+\phi_3}.S_{(\phi_0} C_{(\phi_2} + S_{\phi_1+\phi_3}.S_{(\phi_2)} + l_3 C_{\phi_1+\phi_3+\phi_5}.S_{(\phi_0} C_{(\phi_2} C_{(\phi_4} + S_{\phi_1+\phi_3} C_{(\phi_2} S_{(\phi_4} \\ & - C_{\phi_1+\phi_3+\phi_5} S_{(\phi_0} S_{(\phi_2} S_{(\phi_4} + S_{\phi_1+\phi_3+\phi_5} S_{(\phi 2} C_{(\phi_4)} \end{aligned} \tag{3.3}$$

$$y = (l_1 C_{\phi_1} + l_2 C_{\phi_1+\phi_3} + l_3 C_{\phi_1+\phi_3+\phi_5}) C_{\phi_0} \tag{3.4}$$

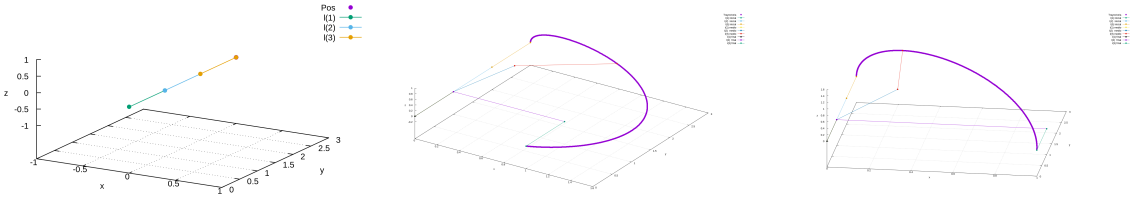
$$\begin{aligned} z =& l_1 S_1 + l_2(-C_{\phi_1+\phi_3} S_{\phi_0} S_{\phi_2} + S_{\phi_1+\phi_3} C_{\phi_2}) + l_3(-C_{\phi_1+\phi_3+\phi_5} S_{\phi_0} S_{\phi_2} C_{\phi_4} + \\ & S_{\phi_1+\phi_3+\phi_5} S_{\phi_2} S_{\phi_4} - C_{\phi_1+\phi_3+\phi_5} S_{\phi_0} C_{\phi_2} S_{\phi_4} + S_{\phi_1+\phi_3+\phi_5} C_{\phi_2} C_{\phi_4}) \end{aligned} \tag{3.5}$$

Para comprobar que las ecuaciones son correctas se genero la trayectoria con las siguientes 4 suposiciones

1. Dos rotaciones a 90 ° la posición final debe ser 180°
2. Dos rotaciones a 90 ° y -90 ° la pocision final debe ser 0 °
3. Dos rotaciones a 45 ° la posición final debe ser igual a 90 °
4. Rotar ϕ_2 y ϕ_5 ambos a la vez y ver la posición de la última pinza

En la simulación se realiza con los ángulos de rotación incrementando de 0 ° a 90 ° simultáneamente y los ángulos que se ven afectados por la rotación ϕ_4 y ϕ_6 0 ° a 90 °. Obteniendo que la posición del actuador final comienza a dar un giro y termina regresando con lo que la primera suposición de que dice que dos rotaciones a 90° la posición final debe ser 180° es cierta como se muestra en la figura 3.3.

La segunda suposición dice que dos rotaciones a 90° y -90° la posición final debe ser 0° lo cual es correcto respecto al origen vuelve a la posición inicial en x. 3.3



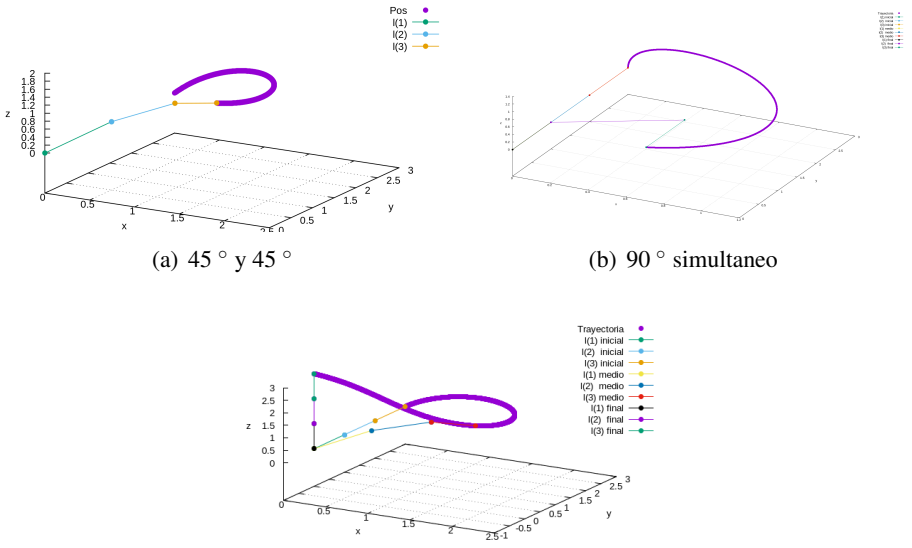
(a) Reposo (b) 0 ° a 90 ° (c) 90 ° a -90 °

Figura 3.3: Trayectorias de ejemplo

La tercera suposición dice que dos rotaciones a 45° y la posición final debe ser igual a 90° la cual es cierta termina siendo un angulo de 90° con respecto al origen. 3.4

La tercera afirmación pide rotar ambos a la vez, se todos los ángulos 90 grados de manera simultánea, mostrando el movimiento de los brazos y la trayectoria del actuador final como se ve en la figura 3.4

La ultima suposición es rotar ambos a la vez y ver la posición de la última pinza se ve en la figura 3.4



(a) 45 ° y 45 °

(b) 90 ° simultaneo

(c) Ambos a 90 °

Figura 3.4: Trayectorias de ejemplo 2

Esta es una manera de comprobar visualmente el comportamiento del manipulador. Hay que recordar que aqui el tamaño del eslabón es unitario por lo que todos miden 1. Un manipulador comúnmente tiene eslabones de diferentes tamaños. Ademas esta simulación esta basada unicamente en las ecuaciones matemáticas obtenidas 3.1 el que tenga aspecto de manipulador comprueba que las ecuaciones fueron aplicadas correctamente.

Capítulo 4

Generación de Trayectoria, Teoría τ-*Jerk*

La generación de trayectoria en un manipulador refiere al seguimiento de la posición del efector final para llegar a un punto de referencia deseado, tendríamos entonces podemos obtener la posición final restando la posición inicia o de reposo del manipulador a la posición de referencia deseada $P_{Ref} - P_i = P_f$ Las trayectorias pueden ser de punto a punto donde se indica únicamente el punto inicial y el punto final al que se debe llegar, también pueden utilizarse trayectorias continuas que el manipulador debe seguir, típicamente son lineas rectas o semicírculos que le permiten al manipulador llegar a su objetivo. Si tenemos por ejemplo una linea recta determinada por la ecuación y=mx +h. Tendríamos entonces que la posición x Px del manipulador debe ser igual a la función x de referencia y la posición y del manipulador Py debe ser igual a la y de la función de referencia. En el caso de un semicírculo las ecuaciones a seguir pueden ser determinadas por las coordenadas polares $x = R\cos\theta$ y $y = R\sin\theta$

Para una linea recta:

$$Px = \frac{y-h}{m} \tag{4.1}$$

$$Py = mx + h \tag{4.2}$$

Para un Semicírculo con $0 < \theta < 180$:

$$Px = R\cos\theta \tag{4.3}$$

$$Py = R\sin\theta \tag{4.4}$$

Sin embargo, los motores con los cuales se mueven estos manipuladores no pueden llegar de un punto a otro de manera brusca por lo que las trayectorias que se genera por lo que se utilizan valores discretos para las funciones continuas.Sin embargo existen infinitas posibilidades de lineas rectas o semicírculos para alcanzar un punto deseado dentro del área de trabajo del manipulador. Por lo que se buscan métodos que permitan encontrar trayectorias de un punto inicial a un punto final, actualmente distintos trabajos utilizan algoritmos genéticos para generar trayectorias en manipuladores o sistemas robóticas móviles [27], [28], [29]. En este trabajo se presento el método de control Tau-Jerk que aunque resulta menos eficiente que el control variante en el tiempo.

4.1. Tau-Jerk

Se conoce como Tau-Jerk cuando el acoplamiento se hace con su propia aceleración. Con la ecuación de acoplamiento de $\tau_q = k_{q,p}\tau_p$, podemos acoplar dos o mas objetivos, permitiendo que cierren sus brecha simultáneamente. En un manipulador, sabemos que su aceleración inicia y termina en cero por lo que podemos acoplar esta aceleración con la brecha de su posición al final del movimiento. De acuerdo con la teoría general de Tau, la brecha en tau se define como $\tau_q = \frac{x}{\dot{x}}$.

Partiendo entonces de la ecuación de acoplamiento $\tau_q = k_{q,p}\tau_p$, podemos acoplar la brecha x, con la brecha intrínseca generada por el Jerk (sobre aceleración) que se puede expresar de la siguiente manera.

$$x_j = \frac{1}{6} j T_d^3 - \frac{1}{6} j t^3 \tag{4.5}$$

$$\dot{x}_j = \frac{1}{2} j t^2 \tag{4.6}$$

$$\tau_j(t) = \frac{x}{\dot{x}} = \frac{1}{3}(t - \frac{T_d^3}{t^2}) \tag{4.7}$$

Por lo tanto $p_0 = T^3$ y el tiempo de la brecha p es la diferencia entre el tiempo T menos el tiempo actual t. $p = T^3 - t^3$, por lo que sustituyendo en la ecuación 4.8, tenemos que

$$x(t) = \frac{x_0}{T^{3/k_{p,q}}}(T^3 - t^3)^{1/k_{qp}} \tag{4.8}$$

Con la ecuación 4.8, podemos encontrar la posición del manipulador, se deben tener las ecuaciones que describan el movimiento del manipulador, los cuales juntas forman un vector P y para encontrar la distancia de su brecha. Tenemos que $\triangle p = pd - p(t)$. Por lo tanto, sustituyendo en la fórmula.

$$p(t) = \frac{\triangle p}{T^{3/k_{p,q}}}(T^3 - t^3)^{1/k_{qp}} \tag{4.9}$$

Si por el contrario se tiene como referencia los ángulos ϕ, entonces la brecha es la diferencia entre los ángulos $\triangle\phi = \phi d - \phi(t)$

$$\phi(t) = \frac{\triangle \phi}{T^{3/k_{p,q}}}(T^3 - t^3)^{1/k_{qp}} \tag{4.10}$$

Utilizando esta ecuacion podemos ajustar los valores escalares de la constante k para obtener el valor optimo para nuestra aplicación. En la figura 4.1 se utiliza un tiempo T = 2.0 y un valor inicial de x0 = 2.0.

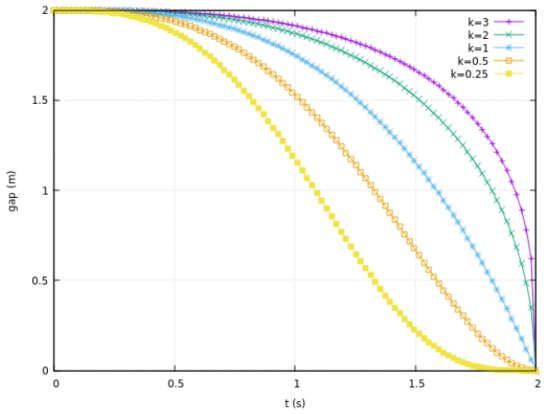


Figura 4.1: Ajuste de constante k

Sin embargo, si se requiere encontrar los ángulos a partir de una posición de referencia, es necesario un método para encontrar las raíces, debido a que el método Tau jerk no relaciona la posición con sus ángulos.Se puede recurrir a el método Newton-Rapshon.

Newton Rapshon

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \tag{4.11}$$

Para obtener el resultado de manera numérica podemos utilizar la regla de Cramer, que se define como $A.X = b$ donde la incógnita x_i es $\frac{|A_j|}{|A|}$, donde la Aj, es la matriz A cambiado la columna j por la columna con términos independientes b. Si lo generalizamos en función de los ángulos, ϕ obtenemos que:

$$\phi(n)_{t+1} = \phi(n)_t - \frac{det(\mathbf{A}j)}{det(\mathbf{A})} \tag{4.12}$$

Este método iteractivo calcula un error, en este documento ese error debe ser menor a 0,1 y es calculado por:

$$error = Pref - P$$

Este método permite encontrar las raíces de un sistema de ecuaciones no lineales, pero se utiliza el determinante para encontrarlas, por lo que es necesario que las matrices sean cuadradas en un manipulador para que este sistema sea cuadrado debe ser totalmente actuado, o utilizar otro método que permita obtener una matriz cuadrada como utilizar las ecuaciones que describen movimientos angulares en el actuador final.

El manipulador planteado en este documento es redundate su matriz no es cuadrada (3x6), sin embargo se puede utilizar la orientación para obtener una matriz cuadrada de (6x6) de tal forma que obtenemos las ecuaciones de Newton-Rapshon

$$x(\Phi) = xt + (\phi 0_t + 1 - \phi 0_t)\frac{\partial xt}{\partial \phi_0} + ... + (\phi 5_t + 1 - \phi 5_t)\frac{\partial xt}{\partial \phi_5} = 0$$

$$y(\Phi) = yt + (\phi 0_t + 1 - \phi 0_t)\frac{\partial yt}{\partial \phi_0} + ... + (\phi 5_t + 1 - \phi 5_t)\frac{\partial yt}{\partial \phi_5} = 0$$

$$z(\Phi) = zt + (\phi 0_t + 1 - \phi 0_t)\frac{\partial zt}{\partial \phi_0} + ... + (\phi 5_t + 1 - \phi 5_t)\frac{\partial zt}{\partial \phi_5} = 0$$

$$\alpha(\Phi) = \alpha t + (\phi 0_t + 1 - \phi 0_t)\frac{\partial t}{\partial \phi_0} + ... + (\phi 5_t + 1 - \phi 5_t)\frac{\partial \alpha t}{\partial \phi_5} = 0$$

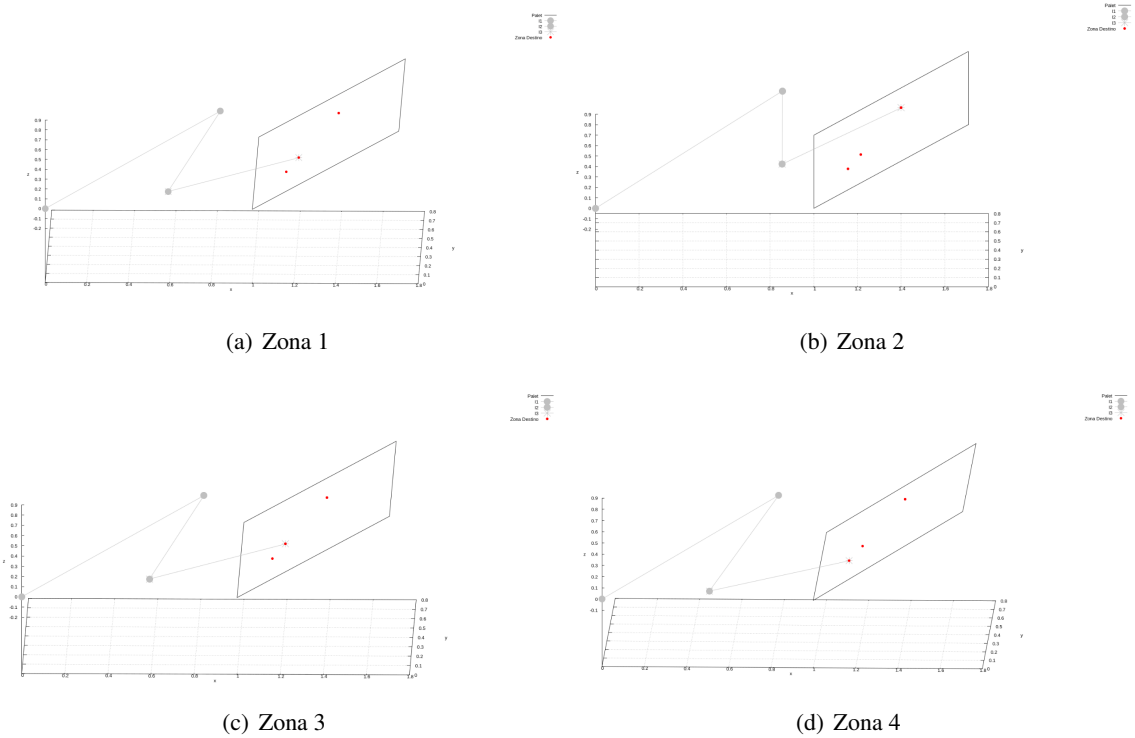
$$\beta(\Phi) = \beta t + (\phi 0_t + 1 - \phi 0_t)\frac{\partial \beta t}{\partial \phi_0} + ... + (\phi 5_t + 1 - \phi 5_t)\frac{\partial \beta t}{\partial \phi_5} = 0$$

$$\gamma(\Phi) = \gamma t + (\phi 0_t + 1 - \phi 0_t)\frac{\partial \gamma t}{\partial \phi_0} + ... + (\phi 5_t + 1 - \phi 5_t)\frac{\partial \gamma t}{\partial \phi_5} = 0$$

El método tau nos permite encontrar la trayectoria de un punto final y siendo un método biológicamente inspirado los puntos discretos forman una trayectoria suave. De tal forma que aunque no utilicemos el método para controlar las variables ene el tiempo, podemos utilizarlo para darle seguimiento a la posición del manipulador, mediante la ecuación recursiva 4.9. En donde $p(t)$ es cada punto de ensamble a donde debe llegar el manipulador En uestro ejemplo tenemos 7 puntos de ensamble por lo tanto tendríamos que para n =1 hasta 7 $p(t)_n = \frac{\triangle p}{T^{3/k_{p,q}}}(T^3 - t^3)^{1/k_{qp}}$

4.2. Procesamiento de Datos

Para obtener el los ángulos que se requieren para mover los motores es necesario el uso de un método de cinemática inversa y que además permita el control de manipulador de manera suave para esto utilizaremos el método algebraico de aproximaciones sucesivas que resulto mejor que el método tau jerk. Esto como ya se vio utilizando las ecuaciones 8.3. También se dibujo el tablero y los brazos del manipulador (l1,l2,l3). Se Muestran simulación en gnuplot de 4 zonas destino



(a) Zona 1 (b) Zona 2

(c) Zona 3 (d) Zona 4

Figura 4.2: Zonas Destino

Capítulo 5

Control Variante en el Tiempo Basado en Modelo

5.1. Control Variante en el Tiempo

La cinemática inversa se encarga de encontrar la ecuación en la que el manipulador deberá moverse para llegar a cierta posición. Exiten distintos métodos para encontrar la cinemática inversa. Sin embargo, debido a que las ecuaciones de movimiento implican la suma de ángulos dentro de funciones de senos y cosenos no es posible encontrar un solución analítica mediante métodos algebraicos. En el método geográfico nos permite encontrar analíticamente utilizando las funciones inversa de arctan y relaciones trigonométricas sin embargo solo es adecuado para manipuladores con pocos grados de libertad. Otros métodos utilizan matrices o desacoplamientos de los eslabones del robot. Sin embargo, también existen métodos numéricos que son adecuados para el uso de sistemas computacionales y suelen ser muy exactos.En un trabajo de M. Rolf y J. J. Steil logran que un robot aprenda y realizó la tarea de cinemática inversa dividiéndolo en 2 o más manipuladores virtuales. [26] Por la naturaleza del manipulador sabemos que su espacio de trabajo esta limitado por el largo de sus eslabones y sus articulaciones. Además tiene más de un camino para llegar a un punto. En este documento se realizara la comparación de dos métodos numéricos que resuelven la cinemática inversa. El primero es un método biológico llamado Tau-Jerk y el segundo un método Algebraico que controla la variable y realiza aproximaciones sucesivas al objetivo planteado.

La posición del efector final está dada en función de los ángulos ϕ de la siguiente manera $\vec{P} = f(\vec{\phi})$. Despejando esta ecuación tenemos que $\vec{\phi} = f^{-1}\vec{P}$. Sin embargo, dado que no existe una función inversa para obtener los ángulos, se utiliza el jacobiano. El jacobiano es una función que se utiliza para aproximar linealmente una función a un punto. Se obtiene el jacobiano mediante las derivadas parciales de primer orden, como lo muestra la ecuación 5.8

$$\mathbf{J}(\vec{\phi}) = \begin{bmatrix} \frac{\partial x}{\partial \phi_0} & \frac{\partial x}{\partial \phi_1} & \cdots & \frac{\partial x}{\partial \phi_n} \\ \frac{\partial y}{\partial \phi_0} & \frac{\partial y}{\partial \phi_1} & \cdots & \frac{\partial y}{\partial \phi_n} \\ \frac{\partial z}{\partial \phi_0} & \frac{\partial z}{\partial \phi_1} & \cdots & \frac{\partial z}{\partial \phi_n} \end{bmatrix} \tag{5.1}$$

Por lo tanto el jacobiano $\mathbf{J} = \frac{d\vec{P}}{d\phi}$ se aproximaría a $f(\vec{P})$. Teniendo entonces la ecuación

$$\vec{P}' = \mathbf{J}(\vec{\phi}).\vec{\phi}' \tag{5.2}$$

En caso de los manipuladores redundantes se puede utilizar el método de la pseudoinversa de Penrose para obtener la inversa del jacobiano. $A^+ = (A^T.A)^{-1}.A^T$

Si desarrollamos esta ecuación tenemos que:

$$\vec{P}' = \mathbf{J}(\vec{\phi}).\vec{\phi}'$$

$$\mathbf{J}(\vec{\phi})^{-1}.\vec{P}' = \mathbf{J}(\vec{\phi})^{-1}.\mathbf{J}(\vec{\phi}).\vec{\phi}'$$
$$\vec{\phi}' = \mathbf{J}(\vec{\phi})^{-1}.\vec{P}'$$

$$\int_{\phi_i}^{\phi_f} \frac{\vec{\phi}}{dt} = \mathbf{J}(\vec{\phi})^{-1} \int_{p_i}^{p_f} \frac{\vec{P}'}{dt}$$

$$\phi_f - \phi_i = J^1\vec{\phi}.(\vec{P}_f - P_i)$$

Donde P_i y es la posición inicial y P_f es la posición final. De la misma manera con ϕ_i y ϕ_f partiendo de esta ecuación podemos despejarla para obtener los ángulos o la posiciones, a partir de una referencia en ángulos o posición según se requiera. Además, la posición final sera tomada como la posición de referencia y la inicial como la posición actual del manipulador, si utilizamos ambas ecuaciones de manera que una llame a la otra podemos obtener una solución numérica como se muestra en las siguientes ecuaciones:

$$\vec{\phi}_{t+1} = \vec{\phi}_t + \mathbf{J}\vec{\phi}^{-1}.(\vec{P}_{ref} - \vec{P}_t) \tag{5.3}$$

$$\vec{P}_{t+1} = \vec{P}_t + \mathbf{J}\vec{\phi}^{-1}.(\vec{\phi}_{t+1} - \vec{\phi}_t) \tag{5.4}$$

Donde:
$\vec{P}_{ref}$ = Posición de referencia
$\vec{\phi}_{t+1}$ = Ángulos en la interacción siguiente
$\vec{\phi}_t$ = Ángulos Actuales
$\vec{P}_{t+1}$ = Posición Siguiente
$\vec{P}_t$ = Posición actual
$\mathbf{J}\vec{\phi}$ = Jacobiano de los ángulos ϕ

Si lo que se tienen son los ángulos de referencia, simplemente se invierten las ecuaciones. Para la reducción del error es dada por la diferencia de la posición y en este experimento cuando esta es menor a 0.1. Además, para realizar la simulación mas parecida a la realidad, se multiplica por una constante de atenuamiento, debido a que en la realidad un motor no puede tener cambios tan grandes. Esta constante es α y puede ser de 0 a 1, siendo 1 el valor mas alto, por lo que las ecuaciones quedan como $\vec{\phi}_{t+1} = \vec{\phi}_t + \alpha.\mathbf{J}\vec{\phi}^{-1}.(\vec{P}_{ref} - \vec{P}_t)$ y/o $\vec{P}_{t+1} = \vec{P}_t + \alpha.\mathbf{J}\vec{\phi}^{-1}.(\vec{\phi}_{t+1} - \vec{\phi}_t)$ sin ser necesario agregar el α en ambas ecuaciones, el error entonces se calcula con la posición de referencia y la actual.

Se realizaron ambos métodos pero el método de control que se utilizara es el método algebraico de aproximaciones sucesivas. Una ventaja de este método algebraico es que no requiere configurar parámetros de manera manual, este método nos da la ruta mas óptima para el objetivo planteado, debido a que controla y aproxima la función al punto más cercano. Tampoco es necesario el uso de un método externo, pues las ecuaciones de posición y movimiento angular se llaman una a la otra. Además, no es necesario tener una matriz cuadrada, gracias a la inversa de Penrose. El resultado se muestra en el figura 5.1. Podemos observar como el método reduce la brecha conforme se acerca al objetivo. Para este método, se necesita conocer las ecuaciones que describen el movimiento de un manipulador de 6 DOF son la siguientes.

$$\begin{aligned} x =& l_1 C_{(\phi_1}.S_{(\phi_0} + l_2 * (C_{(\phi_1+\phi_3}.S_{(\phi_0}C_{(\phi_2} + S_{\phi_1+\phi_3}.S_{(\phi_2)} + l_3 C_{\phi_1+\phi_3+\phi_5}.S_{(\phi_0}C_{(\phi_2}C_{(\phi_4} + \\ & S_{\phi_1+\phi_3}C_{(\phi_2}S_{(\phi_4} - C_{\phi_1+\phi_3+\phi_5}S_{(\phi_0}S_{(\phi_2}S_{(\phi_4} + S_{\phi_1+\phi_3+\phi_5}S_{(\phi 2}C_{(\phi_4)} \end{aligned} \tag{5.5}$$

$$y = (l_1 C_{\phi_1} + l_2 C_{\phi_1+\phi_3} + l_3 C_{\phi_1+\phi_3+\phi_5})C_{\phi_0} \tag{5.6}$$

$$\begin{aligned} z =& l_1 S\phi_1 + l_2(-C_{\phi_1+\phi_3}S_{\phi_0}S_{\phi_2} + S_{\phi_1+\phi_3}C_{\phi_2}) + l_3(-C_{\phi_1+\phi_3+\phi_5}S_{\phi_0}S_{\phi_2}C_{\phi_4} + \\ & S_{\phi_1+\phi_3+\phi_5}S_{\phi_2}S_{\phi_4} - C_{\phi_1+\phi_3+\phi_5}S_{\phi_0}C_{\phi_2}S_{\phi_4} + S_{\phi_1+\phi_3+\phi_5}C_{\phi_2}C_{\phi_4}) \end{aligned} \tag{5.7}$$

Además, se debe realizar el jacobiano, que es el resultado de la derivación parcial de la ecuaciones en cada ángulo ϕ, como se muestra a continuación:

$$\mathbf{J} = \begin{bmatrix} \frac{\partial x}{\partial \phi_0} & \frac{\partial x}{\partial \phi_1} & \frac{\partial x}{\partial \phi_2} & \frac{\partial x}{\partial \phi_3} & \frac{\partial x}{\partial \phi_4} & \frac{\partial x}{\partial \phi_5} \\ \frac{\partial y}{\partial \phi_0} & \frac{\partial y}{\partial \phi_1} & \frac{\partial y}{\partial \phi_2} & \frac{\partial y}{\partial \phi_3} & \frac{\partial y}{\partial \phi_4} & \frac{\partial y}{\partial \phi_5} \\ \frac{\partial z}{\partial \phi_0} & \frac{\partial z}{\partial \phi_1} & \frac{\partial z}{\partial \phi_2} & \frac{\partial z}{\partial \phi_3} & \frac{\partial z}{\partial \phi_4} & \frac{\partial z}{\partial \phi_5} \end{bmatrix} \tag{5.8}$$

$\frac{\partial x}{\partial \phi_0} = l_1 C(\phi 1) C(\phi_0) + l_2 C(\phi_1 + \phi_3) C(\phi_0) C(\phi_2) + l_3 (C(\phi_1 + \phi_3 + \phi_5) C(\phi_0) C(\phi_2) C(\phi_4) - C(\phi_1 + \phi_3 + \phi_5) C(\phi_0) S(\phi_2) S(\phi_4))$

$\frac{\partial x}{\partial \phi_1} = l_1 - S_{\phi_1} S_{\phi_0} + l_2(-S_{\phi_1+\phi_3} S_{\phi_0} C_{\phi_2} + C_{\phi_1+\phi_3} S_{\phi_2}) + l_3(-S_{\phi_1+\phi_3+\phi_5} S_{\phi_0} C_{\phi_2} C_{\phi_4} + C_{\phi_1+\phi_3} C_{\phi_2} S_{\phi_4} + S_{\phi_1+\phi_3+\phi_5} S_{\phi_0} S_{\phi_2} S_{\phi_4} + C_{\phi_1+\phi_3+\phi_5} S_{02} C_{\phi_4})$

$\frac{\partial x}{\partial \phi_2} = l_2(C_{\phi_1+\phi_3} S_{\phi_0} - S_{\phi_2} + S_{(\phi_1+\phi_3} C_{\phi_2}) + l_3(C_{\phi_1+\phi_3+\phi_5} S_{\phi_0} - S_{\phi_2} C_{\phi_4} + S_{\phi_1+\phi_3} - S_{\phi_2} S_{\phi_4} - C_{\phi_1+\phi_3+\phi_5} S_{\phi_0} C_{\phi_2} S_{\phi_4} + S_{\phi_1+\phi_3+\phi_5} C_{02} C_{\phi_4})$

$\frac{\partial x}{\partial \phi_3} = l_2(-S_{\phi_1+\phi_3} S_{\phi_0} C_{\phi_2} + C_{\phi_1+\phi_3} S_{\phi_2}) + l_3(-S_{\phi_1+\phi_3+\phi_5} S_{\phi_0} C_{\phi_2} C_{\phi_4} + C_{\phi_1+\phi_3} C_{\phi_2} S_{\phi_4} + S_{\phi_1+\phi_3+\phi_5} S_{\phi_0} S_{\phi_2} S_{\phi_4} + C_{\phi_1+\phi_3+\phi_5} S_{02} C_{\phi_4})$

$\frac{\partial x}{\partial \phi_4} = l_3(C_{\phi_1+\phi_3+\phi_5} S_{\phi_0} C_{\phi_2} - S_{\phi_4} + S_{\phi_1+\phi_3} C_{\phi_2} C_{\phi_4} - C_{\phi_1+\phi_3+\phi_5} S_{\phi_0} S_{\phi_2} C_{\phi_4} + S_{\phi_1+\phi_3+\phi_5} S_{02} - S_{\phi_4})$

$\frac{\partial x}{\partial \phi_5} = l_3(-S_{\phi_1+\phi_3+\phi_5} S_{\phi_0} C_{\phi_2} C_{\phi_4} + S_{\phi_1+\phi_3+\phi_5} S_{\phi_0} S_{\phi_2} S_{\phi_4} + C_{\phi_1+\phi_3+\phi_5} S_{02} C_{\phi_4})$

$\frac{\partial y}{\partial \phi_0} = (l_1 C_{\phi_1} + l_2 C_{\phi_1+\phi_3} + l_3 C_{\phi_1+\phi_3+\phi_5}) - S_{\phi_0}$

$\frac{\partial y}{\partial \phi_1} = (l_1 - S_{\phi_1} + l_2 - S_{\phi_1+\phi_3} + l_3 - S_{\phi_1+\phi_3+\phi_5}) C_{\phi_0}$

$\frac{\partial y}{\partial \phi_2} = 0$

$\frac{\partial y}{\partial \phi_3} = (-S_{\phi_1+\phi_3} + l_3 - S_{\phi_1+\phi_3+\phi_5}) C_{\phi_0}$

$\frac{\partial y}{\partial \phi_4} = 0$

$\frac{\partial y}{\partial \phi_5} = -S_{\phi_1+\phi_3+\phi_5} C_{\phi_0}$

$\frac{\partial z}{\partial \phi_0} = l_2(-C_{\phi_1+\phi_3} C_{\phi_0} S_{\phi_2} + l_3(-C_{\phi_1+\phi_3+\phi_5} C_{\phi_0} S_{\phi_2} C_{\phi_4} - C_{\phi_1+\phi_3+\phi_5} C_{\phi_0} C_{\phi_2} S_{\phi_4}$

$\frac{\partial z}{\partial \phi_1} = l_1 C_1 + l_2(S_{\phi_1+\phi_3} S_{\phi_0} S_{\phi_2} + C_{\phi_1+\phi_3} C_{\phi_2}) + l_3(S_{\phi_1+\phi_3+\phi_5} S_{\phi_0} S_{\phi_2} C_{\phi_4} + C_{\phi_1+\phi_3+\phi_5} S_{\phi_2} S_{\phi_4} + S_{\phi_1+\phi_3+\phi_5} S_{\phi_0} C_{\phi_2} S_{\phi_4} + C_{\phi_1+\phi_3+\phi_5} C_{\phi_2} C_{\phi_4})$

$\frac{\partial z}{\partial \phi_2} = l_2(-C_{\phi_1+\phi_3} S_{\phi_0} C_{\phi_2} + S_{\phi_1+\phi_3} - S_{\phi_2}) + l_3(-C_{\phi_1+\phi_3+\phi_5} S_{\phi_0} C_{\phi_2} C_{\phi_4} + S_{\phi_1+\phi_3+\phi_5} C_{\phi_2} S_{\phi_4} - C_{\phi_1+\phi_3+\phi_5} S_{\phi_0} - S_{\phi_2} S_{\phi_4} + S_{\phi_1+\phi_3+\phi_5} - S_{\phi_2} C_{\phi_4})$

$\frac{\partial z}{\partial \phi_3} = l_2(S_{\phi_1+\phi_3} S_{\phi_0} S_{\phi_2} + C_{\phi_1+\phi_3} C_{\phi_2}) + l_3(S_{\phi_1+\phi_3+\phi_5} S_{\phi_0} S_{\phi_2} C_{\phi_4} + C_{\phi_1+\phi_3+\phi_5} S_{\phi_2} S_{\phi_4} + S_{\phi_1+\phi_3+\phi_5} S_{\phi_0} C_{\phi_2} S_{\phi_4} + C_{\phi_1+\phi_3+\phi_5} C_{\phi_2} C_{\phi_4})$

$\frac{\partial z}{\partial \phi_4} = l_3(-C_{\phi_1+\phi_3+\phi_5} S_{\phi_0} S_{\phi_2} - S_{\phi_4} + S_{\phi_1+\phi_3+\phi_5} S_{\phi_2} C_{\phi_4} - C_{\phi_1+\phi_3+\phi_5} S_{\phi_0} C_{\phi_2} C_{\phi_4} + S_{\phi_1+\phi_3+\phi_5} C_{\phi_2} - S_{\phi_4})$

$\frac{\partial z}{\partial \phi_5} = l_3(S_{\phi_1+\phi_3+\phi_5} S_{\phi_0} S_{\phi_2} C_{\phi_4} + C_{\phi_1+\phi_3+\phi_5} S_{\phi_2} S_{\phi_4} + S_{\phi_1+\phi_3+\phi_5} S_{\phi_0} C_{\phi_2} S_{\phi_4} + C_{\phi_1+\phi_3+\phi_5} C_{\phi_2} C_{\phi_4})$

Una vez resueltas estas ecuaciones, solo se realiza el método numérico para obtener los resultados. El manipulador en la simulación se mueve a la posición (1,1,1).

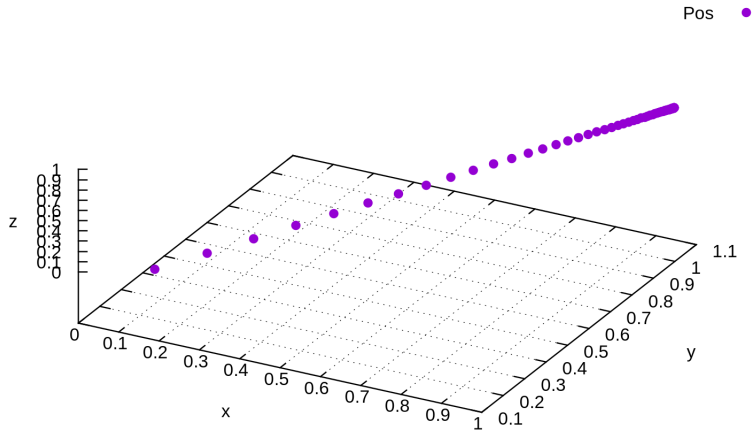


Figura 5.1: *Método de Aproximaciones Sucesivas.*

Observaciones

A partir de los resultados obtenidos se analizan las siguientes observaciones, en las que se determina que aunque el metodo Tau-Jerk es eficiente. Sin embargo, el manipulador que se esta probando en este documento, el método algebraico la mejor opción.

- Configuración de Parámetros El método por aproximaciones sucesivas, no requiere ajuste de parámetros lo que permite realizar cambios en el manipulador simplemente modificando las ecuaciones de movimiento. Obteniendo siempre la ruta mas óptima. Por otro lado, el método de Tau-Jerk requiere optimización de las variables de periodo de tiempo T y la constante k. Sin embargo, permite controlar el tiempo total del movimiento y el acoplamiento.

- Coordenadas y Desplazamiento Angular Ambos métodos permiten obtener la posición del manipulador en coordenadas cartesianas o en función de los ángulos ϕ. Sin embargo, el método de aproximaciones sucesivas utiliza ambos los ángulos y las posiciones en su algoritmo, lo que permiten intercambiar las ecuaciones para obtener los ángulos a partir de una posición dada o viceversa. En cambio, el método tau jerk utiliza las ecuaciones de manera independiente, puedes obtener una posición dando una posición de referencia o un ángulo dando un ángulo de referencia, por lo que para obtener los ángulos a partir de una posición dada es necesario utilizarlo algún método de cinemática inversa.

- Manipuladores Totalmente Actuados y Redundantes El método de aproximaciones sucesivas puede solucionar manipuladores totalmente actuados o redundantes, sin la necesidad de ajustar las ecuaciones de movimiento o el algoritmo. Para el caso de los manipuladores redundantes se utiliza la inversa de Penrose. Mientras que en el método de tau jerk, los manipuladores deben ser totalmente actuados para poder resolver el sistema de ecuaciones no lineales. En el método utilizado Newton-Rapshon, es necesario el determinante, por lo que si la matriz no es cuadrada, no puede obtener las raíces. Para solucionar este problema, se pueden modificar o agregar ecuaciones de movimiento para obtener el mismo numero de ecuaciones que de raíces.

Capítulo 6

Extracción Visual de Características

El sistema robotico propuesto debe reconocer su entorno para manipular unas corbatas en un proceso de manufactura. Se realizara mediante una cámara RGB que retroalimentara al sistema con la imagen del tablero de ensamble en vista de planta. Por lo tanto, se requiere un sistema que permita segmentar y procesar la imágenes obtenidas.

6.1. Segmentación RGB

Una imagen se puede representar como $I = I_R \cup I_G \cup I_B$ donde I es una matriz de píxeles de tamaño mxn. Para segmentar un canal por ejemplo el rojo se tendrían que discriminar los canales G y B y quedaría como $I_c = I_R \cup (I_G \cap I_B)$.z

En cuanto a la segmentación de colores, se debe conocer el color de la mezcla resultante por ejemplo el color blanco es la unión de los 3 canales cuando esos canales tienen un valor mayor que 200 $I_{blanco} = (I_R > 200) \cap (I_R > 200) \cap (I_R > 200)$ y de lo contrario el negro se encuentra en la esquina opuesta teniendo $I_{negro} = (I_R < 50) \cap (I_R < 50) \cap (I_R < 50)$. Quedando el color gris en la linea que conecta estos dos valores formando así la escala a grises. Para segmentar por ejemplo el cyan es una unión de azul y verde con rojo en un valor 0 quedando como $I = (I_G \cap I_B) \cup I_R$.

Dentro del procesamiento de imágenes es común también convertirla en valores binarios eligiendo la intensidad th con un rango máximo donde segmentar representado con th_{max}.

$$\text{Pixel(x,y)} = \begin{cases} 1 & \text{si } th > th_{max} \\ 0 & \text{de lo contrario} \end{cases}$$

6.2. Procesamiento de imagen

Los datos del sensor serán obtenidos mediante una cámara RGB. colocada encima del tablero hecho de delrin que contiene los cables y las corbatas que este manipulador debe cortar, este vídeo capturado en tiempo real obtendrá una imagen estática con opencv para procesarla posteriormente. Durante la simulación se creo una imagen en Solidworks con los componentes requeridos como se ve en la imagen 6.1.

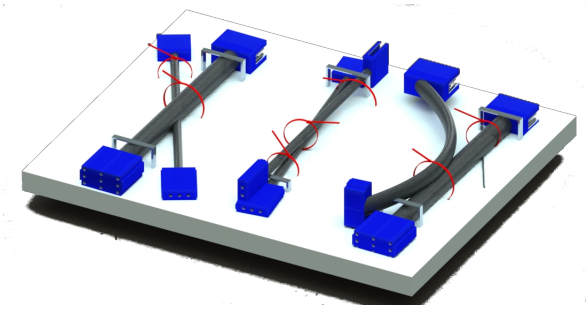


Figura 6.1: *tablero con elementos de ensamble*

La cámara se posiciona arriba del tablero para obtener una imagen 2D en la simulación como se muestra en la simulación de esta captura 6.2

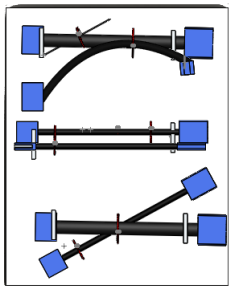


Figura 6.2: *Captura 2d*

Una vez que se obtiene la imagen de la cámara se guarda una imagen y se procesa, el primer proceso es convertirla de RGB a escala en grises de esta manera. Se puede realizar el siguiente proceso que es convertir los rangos de [0-255] en valores binarios 0,1. Para esta imagen por el fondo blanco se invierte [1,0] 6.3 se realiza la operación con la función de opencv cv::Threshold. El cual sigue la formula 6.1.

$$st(x,y) = \begin{cases} 1 & \text{si } st(x,y) > th_{max} \\ 0 & \text{si } st(x,y) <= th_{max} \end{cases} \tag{6.1}$$

Donde st es el un matriz de la imagen, (x ,y) son los elementos de la matriz, th_{max} es el valor máximo en la escala de grises (50 en ese ejercicio).

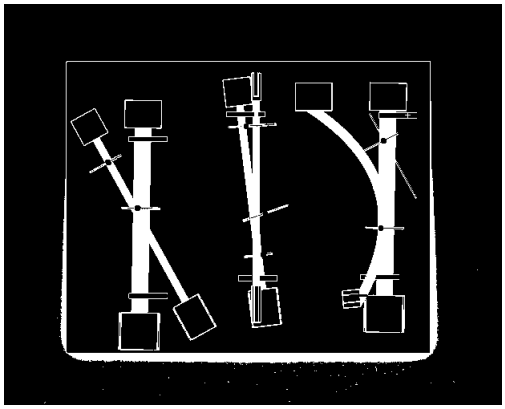


Figura 6.3: *Threshold Binario Inverso*

Después se puede utilizar la función contorno de opencv, esta función hace un barrido por toda la matriz

y busca los cambios de valores binarios en la imagen(0 o 1) que son continuos y hace todos los contornos posible para guardarlos en un vector. De esta manera podemos buscar el contorno mas grande para obtener la imagen del tablero solo. Opencv integra una herramienta de centros basado en las intensidades de los colores(momentos). Sin embargo, no es necesario por que nuestra imagen es cuadrada podemos localizar el centro sacando la mitad de las filas y columnas del vector mas grande. En la imagen 6.4 se dibujaron este contorno y centro sobre la imagen original y en verde para distinguirlo.

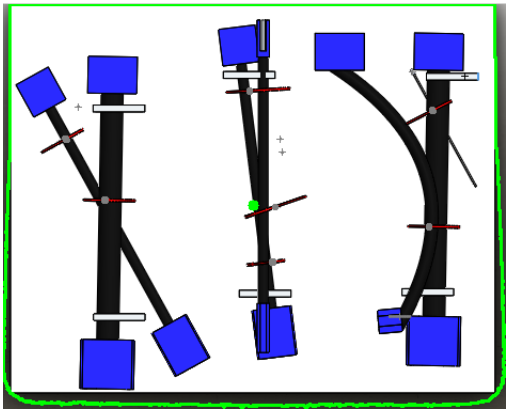


Figura 6.4: *Contorno y centro*

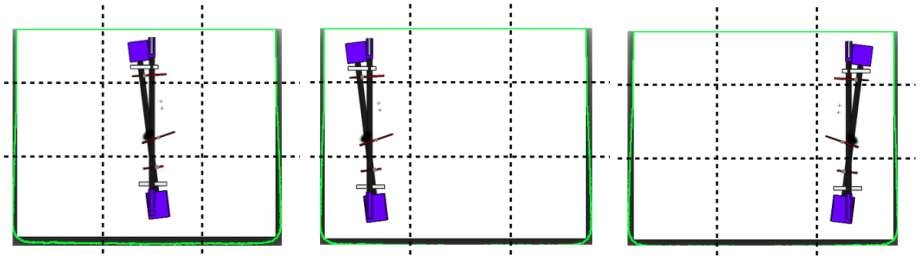
Este proceso sirve para encontrar el centro del tablero, Pues sera el punto de origen el manipulador. Despues de este proceso se utiliza la matriz original con los valores RGB para realizar en entrenamiento de la red hopfield

Capítulo 7

Red Recurrente Bicapa

Una vez que se obtiene los valores de la camara RGB el siguiente paso es que el manipulador aprenda a reconocer corbatas por medio de los colores por lo tanto se utiliza un red neural recurrente.

La red hopfield es una red monocapa recurrente que tiene salidas para obtener un patrón simple con el que podamos comprobar la efectividad de la red Hopfield tomaremos como ejemplo el tablero y lo dividiremos de manera simple en un espacio de 3 x 3. 7.2. Donde solo se encuentra un cable en linea recta con 3 corbatas como en la figura 7.1



(a) (b) (c)

Figura 7.1: Distribución simple de tablero

En este ejemplo existe 3 posibles maneras de acomodar los cables por lo tanto tenemos las entradas x1 , x2 y x3 respectivamente. Si reducimos los nueve espacios y los convertimos a su forma binaria mediante dos colore distintos obtenemos un patrón que podemos utilizar.Podemos representar el tablero como un matriz de 9(3x3) figura 7.2.

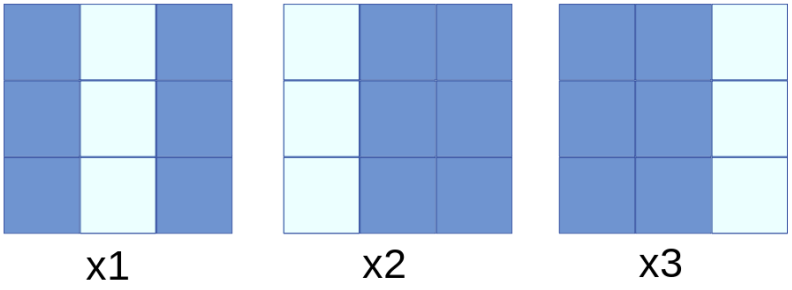


Figura 7.2: Patrones de Entrenamiento

El primer paso es convertirlo a binario mediante la funcion ”step”

$$xn_i = \begin{cases} 1 & \text{si } es \text{ threshold} \\ 0 & \text{de lo contrario} \end{cases} \tag{7.1}$$

$$x1 = [0,0,0,1,1,1,0,0,0] x2 = [1,1,1,0,0,0,0,0,0] x3 = [0,0,0,0,0,0,1,1,1] \tag{7.2}$$

Para el entrenamiento de pesos se multiplica por su transpuesta

$$x1^t.x1 = \begin{bmatrix}0\\0\\0\\1\\1\\1\\0\\0\\0\end{bmatrix} \cdot \begin{bmatrix}0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0\end{bmatrix} = \begin{bmatrix}0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0\\0&0&0&1&1&1&0&0&0\\0&0&0&1&1&1&0&0&0\\0&0&0&1&1&1&0&0&0\\0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0\end{bmatrix} \tag{7.3}$$

$$x2^t.x2 = \begin{bmatrix}1\\1\\1\\0\\0\\0\\0\\0\\0\end{bmatrix} \cdot \begin{bmatrix}0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0\end{bmatrix} = \begin{bmatrix}1&1&1&0&0&0&0&0&0\\1&1&1&0&0&0&0&0&0\\1&1&1&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0\end{bmatrix} \tag{7.4}$$

$$x3^t.x3 = \begin{bmatrix}0\\0\\0\\0\\0\\0\\1\\1\\1\end{bmatrix} \cdot \begin{bmatrix}0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0\end{bmatrix} = \begin{bmatrix}0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&1&1&1\\0&0&0&0&0&0&1&1&1\\0&0&0&0&0&0&1&1&1\end{bmatrix} \tag{7.5}$$

Se realiza la sumatoria de las xn

$$W = \begin{bmatrix}0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0\\0&0&0&1&1&1&0&0&0\\0&0&0&1&1&1&0&0&0\\0&0&0&1&1&1&0&0&0\\0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0\end{bmatrix} + \begin{bmatrix}1&1&1&0&0&0&0&0&0\\1&1&1&0&0&0&0&0&0\\1&1&1&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0\end{bmatrix} + \begin{bmatrix}0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&1&1&1\\0&0&0&0&0&0&1&1&1\\0&0&0&0&0&0&1&1&1\end{bmatrix} \tag{7.6}$$

Se realiza la sumatoria y luego se convierte la diagonal en 0.

$$W = \begin{bmatrix} 1&1&1&0&0&0&0&0&0 \\ 1&1&1&0&0&0&0&0&0 \\ 1&1&1&0&0&0&0&0&0 \\ 0&0&0&1&1&1&0&0&0 \\ 0&0&0&1&1&1&0&0&0 \\ 0&0&0&1&1&1&0&0&0 \\ 0&0&0&0&0&0&1&1&1 \\ 0&0&0&0&0&0&1&1&1 \\ 0&0&0&0&0&0&1&1&1 \end{bmatrix} = \begin{bmatrix} 0&1&1&0&0&0&0&0&0 \\ 1&0&1&0&0&0&0&0&0 \\ 1&1&0&0&0&0&0&0&0 \\ 0&0&0&0&1&1&0&0&0 \\ 0&0&0&1&0&1&0&0&0 \\ 0&0&0&1&1&0&0&0&0 \\ 0&0&0&0&0&0&0&1&1 \\ 0&0&0&0&0&0&1&0&1 \\ 0&0&0&0&0&0&1&1&0 \end{bmatrix} \tag{7.7}$$

Ahora la matriz de pesos ha sido entrenada, Para verificar que este bien se realiza un test para verificar que es correcta. con la formula $y_{t+1} = A \cdot W$ y converge cunado $y = y_{t+1}$.

Si tenemos un patrón A =[0,0,0,1,1,1,0,0,0] el patrón se multiplica por la matriz de pesos entrenada.

$$y_{t+1} = A \cdot W \begin{bmatrix} 0&0&0&1&1&1&0&0&0 \end{bmatrix} \cdot \begin{bmatrix} 0&1&1&0&0&0&0&0&0 \\ 1&0&1&0&0&0&0&0&0 \\ 1&1&0&0&0&0&0&0&0 \\ 0&0&0&0&1&1&0&0&0 \\ 0&0&0&1&0&1&0&0&0 \\ 0&0&0&1&1&0&0&0&0 \\ 0&0&0&0&0&0&0&1&1 \\ 0&0&0&0&0&0&1&0&1 \\ 0&0&0&0&0&0&1&1&0 \end{bmatrix} = \begin{bmatrix} 0&0&0&1&2&3&0&0&0 \end{bmatrix} \tag{7.8}$$

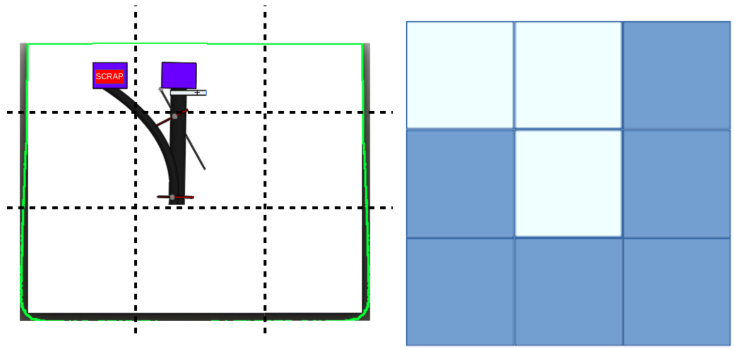
Ahora se realiza la función de activación

$$f(y_{t+1}) = \begin{cases} 1 & \text{si } es \text{ blanco} \\ 0 & \text{de lo contrario} \end{cases} \tag{7.9}$$

$$y = \begin{bmatrix} 0&0&0&1&1&1&0&0&0 \end{bmatrix}$$

Aquí desde la primera iteraccion $y = y_{t+1}$ ya que es uno de los patrones de entrenamiento. De no ser así se aplica la función de activación y se vuelve a multiplicar por la matriz de peso de manera recurrente hasta que $y = y_{t+1}$. Si la matriz converge y el resultado es igual a alguno de los patrones de entrada x1, x2 o x3 entonces significa que el patrón es una linea vertical de lo contrario no lo es.

Tomemos por ejemplo un patrón no entrenado para verificar que no sera reconocido como parte de la red. Supongamos ahora que tenemos un cable mal cortado con una etiqueta roja de scrap como en la figura 7.3el patrón seria B= [1 1 0 0 1 0 0 0 0] por lo tanto y=b.



(a) Scrap (b) Patrón

Figura 7.3: Patrón con pieza de scrap

$$y_{t+1} = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \end{bmatrix} \tag{7.10}$$

$$y = \begin{bmatrix} 1 & 2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, f(y) = \begin{bmatrix} 1 & 1 & 2 & 1 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}), y \neq y_{t+1} \tag{7.11}$$

$$y = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \end{bmatrix} \tag{7.12}$$

$$y = \begin{bmatrix} 2 & 2 & 2 & 0 & 1 & 2 & 0 & 0 & 0 \end{bmatrix} f(y) = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \end{bmatrix})$$

$$y \neq y_{t+1}$$

$$y_{t+1} = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \end{bmatrix}) \cdot W$$

$$\begin{bmatrix} 2 & 2 & 2 & 1 & 1 & 2 & 0 & 0 & 0 \end{bmatrix})$$

$$f(y) = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \end{bmatrix})$$

$$y_{t+1} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \end{bmatrix}) \cdot W$$

$$\begin{bmatrix} 2 & 2 & 2 & 1 & 2 & 3 & 0 & 0 & 0 \end{bmatrix})$$

$$f(y) = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \end{bmatrix})$$

$$y = y_{t+1}$$

Aquí la red converge pero el patrón y no coincide con ninguno en la red neural x1, x2 o x3 lo que significa que el patrón no es una linea recta y por lo tanto no es un cable para ensamble.

Por otro lado, los cables para el ensamble pueden venir de diferentes posiciones y maneras, por lo que un entrenamiento monocapa de patrones simple puede ser insuficiente, es necesaria una clasificación basada en los colores RGB y se propuso una segunda capa a la red Hopfield

7.0.1. Clasificación

Para clasificar con la red hopfield es posible determinar sus patrones de salida desde un inicio para así poder identificarlos a la salida. En el ejemplo anterior podríamos por ejemplo intentar clasificar las columnas. En la imagen 7.2 Se formaron imágenes con patrones de 3x3, en nuestro ejercicio tratábamos de encontrar lineas verticales si además de eso quisiera saber si esa linea vertical esta al principio en medio o al final puedo formar 3 clases

De acuerdo con el ejemplo tenemos 3 neuronas una para cada entrada x1, x2, x3 y 3 salidas las representaremos como ya, yb, yc que son las salidas de cada neurona en binario y juntas forman la y de la salida total y del sistema. Las clases se forman con la activaciones de estas neuronas siendo el máximo 2^n pero solo necesitamos 3 una por cada columna.

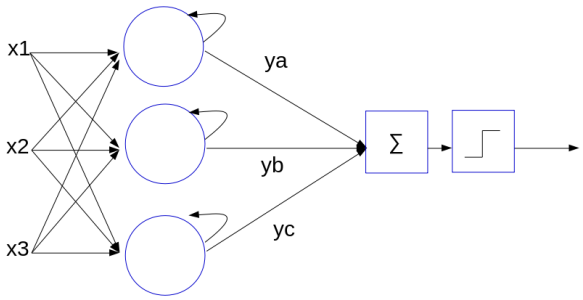


Figura 7.4: Red Hopfield 3 entradas

Clase 1 Columna 1

$$ya > 0, yb < 0, yc < 0$$

Clase 2 Columna 2

$$ya < 0, yb > 0, yc < 0$$

Clase 3 Columna 3

$$ya < 0, yb < 0, yc > 0$$

Este ejemplo tiene patrones muy simples por lo que aunque describe el proceso de clasificación, podemos utilizar otro que permita visualizar mejor la clasificación.

Supongamos que tengo un patrón de coordenadas x,y con 50 muestras y quiero conocer a que cuadrante del plano cartesiano pertenecen. Entonces las coordenadas x,y son las entradas X1, X2. y las clases serán los cuadrantes del plano cartesiano que son 4 como el numero máximo clases.

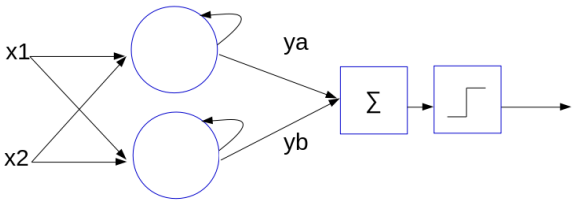


Figura 7.5: Red Hopfield 2 entradas

Los cálculos se realizan en un programa computacional, donde se utilizaran 50 valores aleatorios con rango de -10 a 10 para las entradas X1 y X2 de tal manera que obtenemos una gráfica de los datos antes del convertirlos a binario o polar como ya se había mencionado el proceso es el mismo, solo se utilizara polar por conveniencia así obtendremos salidas con números negativos y con eso se ajustaran a cada cuadrante sin necesidad de otro proceso.

$$f(yn_{t+1}) = \begin{cases} 1 & \text{si } es \text{ blanco} \\ -1 & \text{de lo contrario} \end{cases} \tag{7.13}$$

Clase 1 Primer Cuadrante

$$ya > 0, yb > 0$$

Clase 2 Segundo Cuadrante

$$ya > 0, yb < 0$$

Clase 3 Tercer Cuadrante

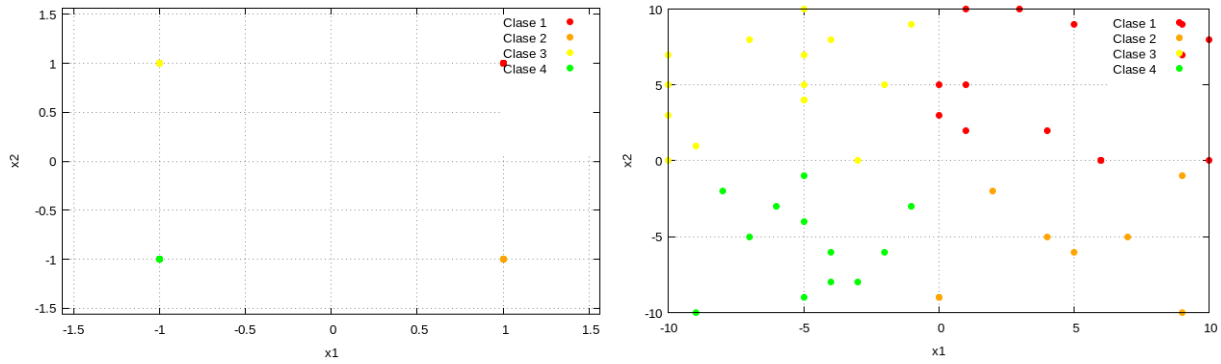
$$ya < 0, yb > 0$$

Clase 4 Cuarto Cuadrante

$$ya < 0, yb < 0$$

Después se realiza la función de activación se comienza con el entrenamiento, el entrenamiento de la matriz de pesos y los test para verificar el entrenamiento.

Al final obtenemos la red clasificada con las neuronas de Hopfield las cuales tienen un valor polares [-1.1] por lo que solo vemos las neuronas activadas en cada región de acuerdo con la activación de las neuronas ya y yb.



(a) Activación de Neuronas

(b) Valores Clasificados

Figura 7.6: Clasificación Hopfield

Sin embargo se plantea utilizar una red bicapa con los datos de salida de la red hopfield y volver a aplicar el método de esta forma tendríamos una red multicapa como se ve en la imagen 7.10

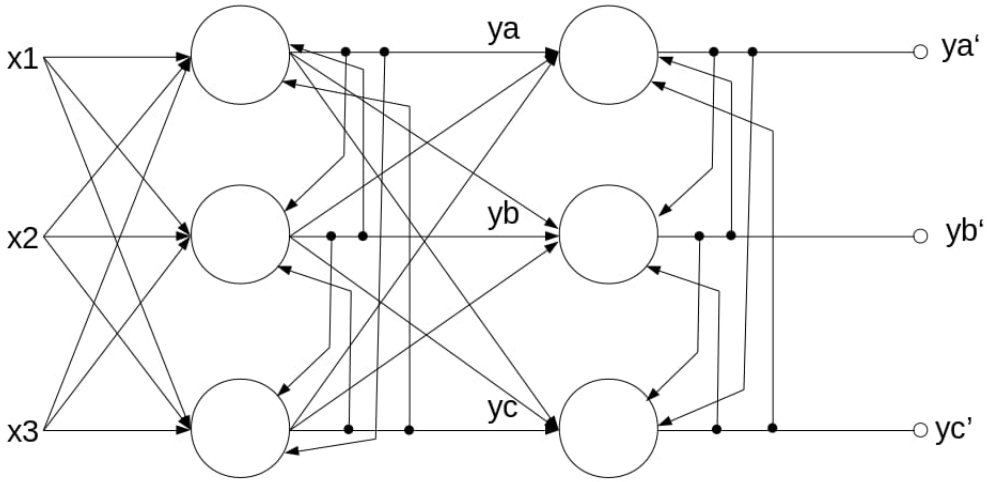


Figura 7.7: *Red Hopfield multicapa*

Las matrices de imagenes suelen tener valores muy grandes lo que vuelve poco realista tener un sistema de tiempo real en robots manipuladores, por ello lo primero que se hace es reducir la imagen para obtener matrices esta imagen se redujo al 50 % quedando (236 x 190) son 44840 pixeles que se forman por 3 valores RGB un total de 134520 datos para entrenamiento. Estos datos seran las entrada de nuestra red hopfield x1, x2 y x3 la imagen 7.8 lo representa como el cubo RGB en un sistema de coordenadas.

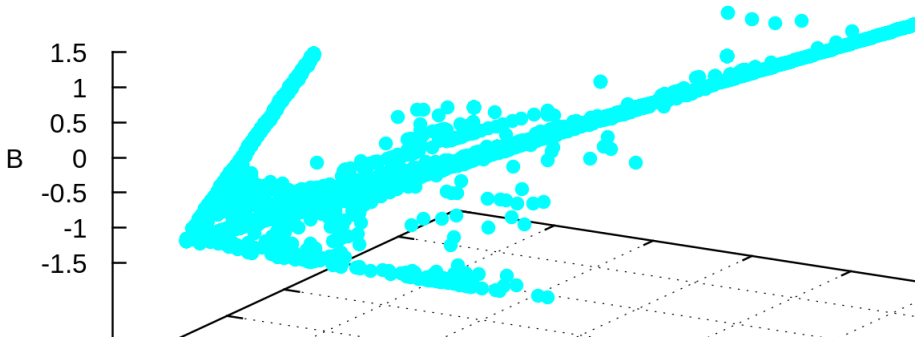


Figura 7.8: *Datos para entrenamiento*

7.1. Patrones de entrada

Una vez que se obtienen los datos de los colores RGB que se utilizaran para el entrenamiento y clasificación de la red lo primero que se tienen que hacer es convertirlos a binario pues las neuronas en Hopfield son binarias. Para eso dividiremos a la mitad a los valores máximos de RGB (254). Además los dividimos entre

100 para manejarlos en valores unitarios.

$$x_1 = \frac{(R-127)}{100} \tag{7.14}$$

$$x_2 = \frac{(G-127)}{100} \tag{7.15}$$

$$x_3 = \frac{(B-127)}{100} \tag{7.16}$$

Para procesar los datos lo primero que se tienen que hacer en la red hopfield es convertirlos a binario pues las neuronas en hopfield son binarias. Para eso dividiremos le restaremos la mitad a los valores RGB. y además los dividimos entre 100 para manejarlos en unidades.

Es claro que de esta manera perdemos una alta gama de colores intermedios pero tenemos 4 objetos a detectar por lo que necesitamos 4 y los valores que buscamos están en el canal R y lo podemos obtener mediante la segmentación $I_R \cup (I_G \cap I_B)$ donde I es la matriz de la imagen. También debemos obtener el blanco que representa el tablero con sus sujetadores y graciass a que ahora son binarios $I_{blanco} = (I_R > 0) \cap (I_R > 0) \cap (I_R > 0)$ y el negro que son los cables $I_{negroc} = (I_R < 1) \cap (I_R < 1) \cap (I_R < 1)$. Además tenemos el azul que son los conectores$I_B \cup (I_R \cap I_G)$. La imagen 7.9 muestra el cubo RGB ya clasificad0.

Clase 1 Rojo Corbatas

$$x_1 > 0 \cdot x_2 <= 0 \cdot x_3 <= 0 \tag{7.17}$$

Clase 2 Azul Conectores

$$x_1 <= 0 \cdot x_2 < 0 \cdot x_3 > 0 \tag{7.18}$$

Clase 3 Blanco tablero

$$x_1 > 0 \cdot x_2 > 0 \cdot x_3 > 0 \tag{7.19}$$

Clase 4 Negro Cables

$$x_1 <= 0 \cdot x_2 <= 0 \cdot x_3 <= 0 \tag{7.20}$$

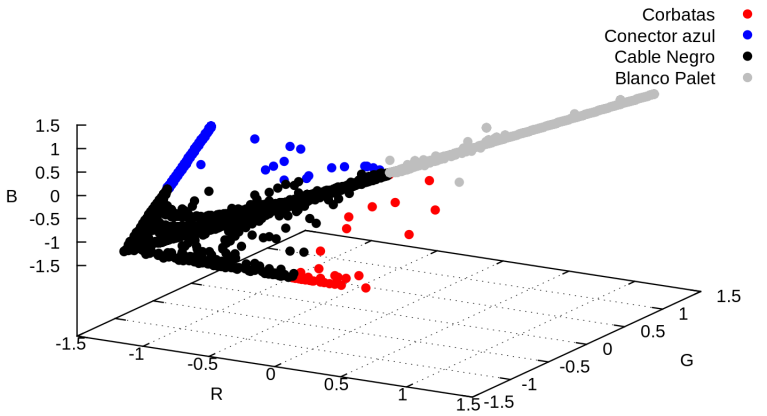


Figura 7.9: *Primer Clasificación en Hopfield*

Aquí ya podemos trabajar con los datos como se muestra es congruente la cantidad de puntos en cada color, los entrenaremos obteniendo la matriz de pesos para guardar los patrones.$W_i j = \sum_{i=0}^{M-1} x_i^T \cdot x_j$ sin embargo añadiremos otra capa para clasificar los valores la imagen 7.10muestra la red con sus entradas (x1,x2,x3) y salidas(ya, yb, yc).

Ahora tenemos entradas y salidas binarias con las que podemos segmentar mas los datos para evitar fallos obteniendo la siguiente grafica 7.10. Después del entrenamiento estos datos son los que se usaran para la zona destino (clase 4).

ya	yb	yc	Objeto
1	1	1	Sujetadores /tablero
1	1	0	Fallos
1	0	1	Fallos
1	0	0	Zona Destino
0	1	1	Fallos
0	1	0	Fallos
0	1	0	Conectores
0	0	0	Cables

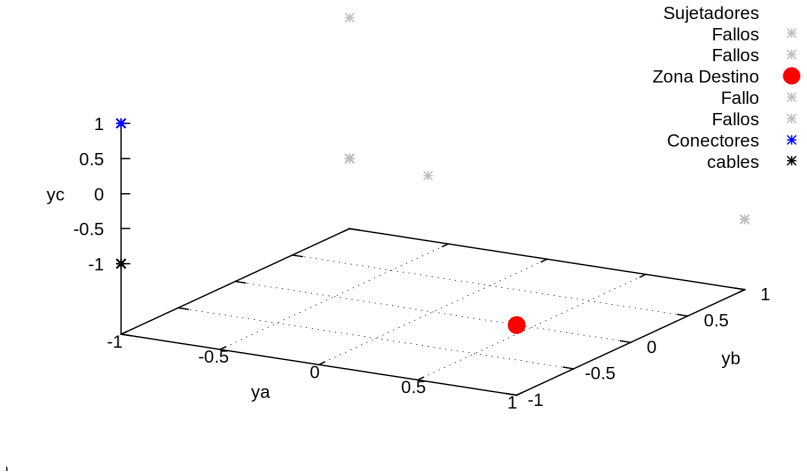


Figura 7.10: *Segunda Clasificación en Hopfield*

7.2. Patrones Clasificados de Salida

una vez que la red a reconocido los objetos se obtienen los valores de la matriz de datos para detectar los pixeles donde se encuentran las zonas que nos interesan en la imgagen 7.11 pueden verse de color amarillo los puntos de destino.

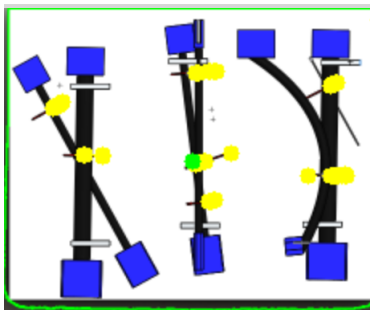


Figura 7.11: *Zona Destino*

Antes de mandar los datos se preprocesan debido a que lo que obtuvimos son las posiciones de la matriz. La cual esta en pixeles con una simple regla de 3 los convertimos a puntos sobre el tablero real el cual mide 500 mm x 400 mm. (0.5 m y 0.4 m) $x = (0.5/F) \cdot Zd_x$ y $y = (0.4/C) \cdot Zd_y$ donde x y son las indices de la matriz, F es el total de filas de la matriz, C numero de columnas de la matriz y p es el Zd es el punto de la zona destino.

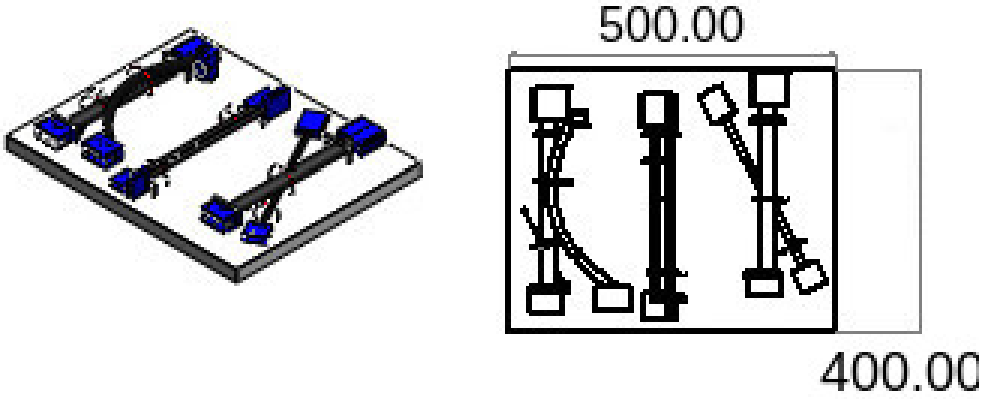


Figura 7.12: *Plano del tablero*

Capítulo 8

Simulación computacional

Una vez que se obtienen los datos del modelo perceptual se utiliza una memoria compartida por medio de la libreria boost::interprocess. Lo que hara es crear un archivo temporal donde se guardaran los datos en forma de matriz para ser ledos por otro proceso. De esta manera se aprovechan los hilos de los procesadores, utiliza un mutex de tal forma que la mientras un programa escribe queda un candado que no se libera hasta que termina entonces el otro programa puede leer estos datos 8.3.

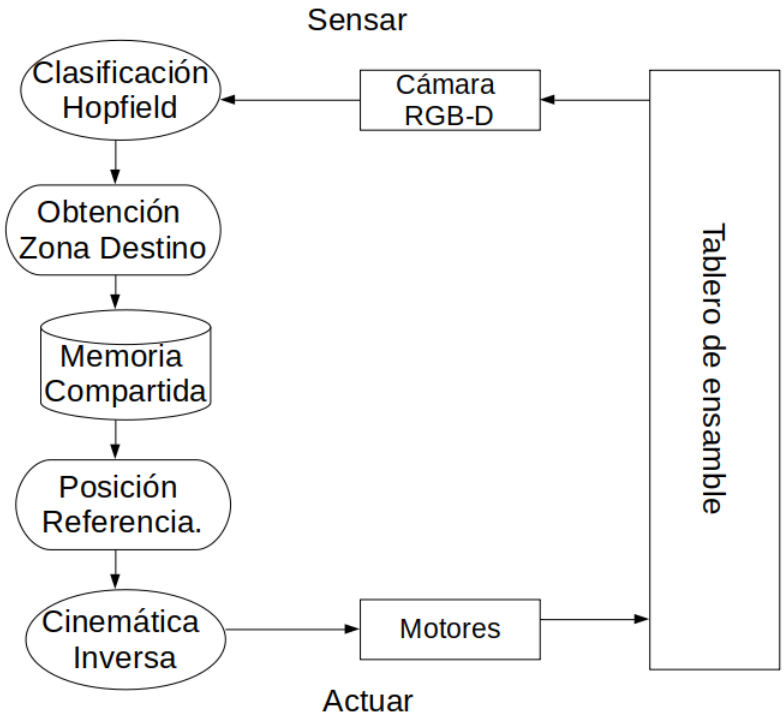


Figura 8.1: *Esquema de procesos*

Para poder probar el sistema robótico se pretende utilizar una herramienta de simulación robótica que permita simular variables físicas de manera realista. Existen diferentes simuladores para robótica regularmente están basados en motores de física que permiten simular la cinemática y dinámica del robot a fin de simular de la manera mas realista el comportamiento del robot. Se consideraron varios simuladores como Project Chrono, el cual utiliza irrlicht un motor 3D de tiempo real que se programa en c++. Además de aprovechar las librerías de chrono de boost, entre otros paquetes de simulación de física. También esta gazebo un simulador especializado en robótica y la evolución de playerstage que incluyen los controladores para varios

sistemas roboticos reales permitiendo probarlos después de la simulación. Por ultimo se considero ROS una paqueteria que une varias herramientas incluidas gazeboo. Además de tros paquetes que permiten diseñar, simular y probar los sistemas roboticos.

8.1. ROS Melodic

Robot Operative System (ROS) es una paquetería que une diferentes librerías y modulos optimizados para uso en robótica, existe una versión de ROS por cada versión LTS (long term support) de ubuntu. La versión de ROS que se utilizo fue la de Melodic en Ubuntu 18.04 para aprovechar MoveIt un paquete de ROS que se especializa en manipuladores roboticos.

ROS funciona mediante un espacio de trabajo llamado catkin que une todo el proyecto y funciona de manera similar a un espacio de trabajo de cmake, incluso la compilacion es a travez de cmake. Dentro de este workspace se crea un bash que permite reconocer los comandos de ros como roslaunch, roscd, rosrun. que hacen los mismo que sus analogos de la terminal de linux. Por ejemplo roscd y cd, en una terminal hacen lo mismo con la diferencia que usando roscd recorre solo las carpetas dentro del espacio de trabajo de catkin. Igualmente en lista solo esas carpetas. roslaunch correrá un archivo .launch con el nombre sin importa la carpeta donde se encuentre siempre que este dentro del espacio de trabajo. rosrun corre programas instalados en ros como ros graph. Ros une todos sus programas en nodos controlador por la librería ros.h que comunica todo el espacio de trabajo haciendo simple la tarea de comunicación y permitiendo ejecutar instrucciones por separado de cualquier nodo en ros.

8.2. MoveIt

Moveit es un paquete en ROS diseñado para trabajar con manipuladores roboticos, basicamente permite controlar el manipulador mediante la ejecución de ordenes, trayectorias etc. Sin necesidad de mucho conocimiento al respecto. Para la simulación se puede utilizar GAZEBOO o RVIZ (Ros Visualization). En un principio se penso en Gazebo incluso se diseño un manipulador simple que permitía probar el modelo planteado. Sin embargo, Rviz es nativo de ROS y esta pensado en la visualización de las tareas especificas que se requieren. En Rviz no es posible editar el modelo 3d ni las propiedades o fuerzas físicas, solo las del robot. Para utilizar un robot nuevo se debe diseñar el URDF and SRDF que incluye todas las definiciones de articulaciones y eslabones del robot. En rviz se definen únicamente los grupos (cuales son del manipulador y cuales del efector por ejemplo), y funciones agregadas como cámaras u otros sensores. Por fortuna ya se cuenta con modelos ya realizados para este proyecto se iba a utilizar un manipulador Yaskawa montoman el cual que viene para descargar en la pagina de MoveIt.

Para ejecutar las acciones se puede utilizar el mismo Rviz y hacerlo de manera visual con el ratón y ajustando parámetros en el mismo programa o bien puede recibir comandos mediante python o c++. Además existen dos maneras de programarlo de manera completa donde se programa en c++ o python incluyendo movimientos y preferencias de visualización Rviz (colores, objetos, trayectorias etc). O simplemente programar movimientos y trayectorias y abrir por separado Rviz con el manipulador y preferencias requeridas, uniéndose con un nodo de ros a esto se le llama Move group. Utilizar los move groups permite mantener abierto Rviz y programar diferentes trayectorias.

8.3. Modulo ANN

Para obtener las zonas destino primero se procesa la imagen, en este trabajo se utiliza una imagen png simulando la toma de una cámara real y mediante una red de hopfield se clasifican los objetos para obtener los puntos deseados.

Para esto se creo un nuevo modulo llamado ANN dentro del espacio de trabajo de ROS donde se encuentra instalado moveit. Esta red ANN utiliza Armadillo, openCV y varias librerias de boost los cuales se anexaron al cmake del nuevo modulo ANN junto a su package.xml para poder ejecutarse. Cuando obtiene las zonas destino crea el archivo .txt y se comunica mediante un nodo de ROS al move group Taujerk/Aprox para avisarle que ya esta actualizado el archivo .txt. Creando un lazo cerrado entre estos programas.el nodo de la red ANN se llama hopf y los move gruop taujerk/Aprox. Mediante rqt graph podemos ver los nodos que se ejecutan en ROS.

También se pueden ver los topics. Los topics son los nombres de los canales a través de los cuales los nodos intercambian mensajes

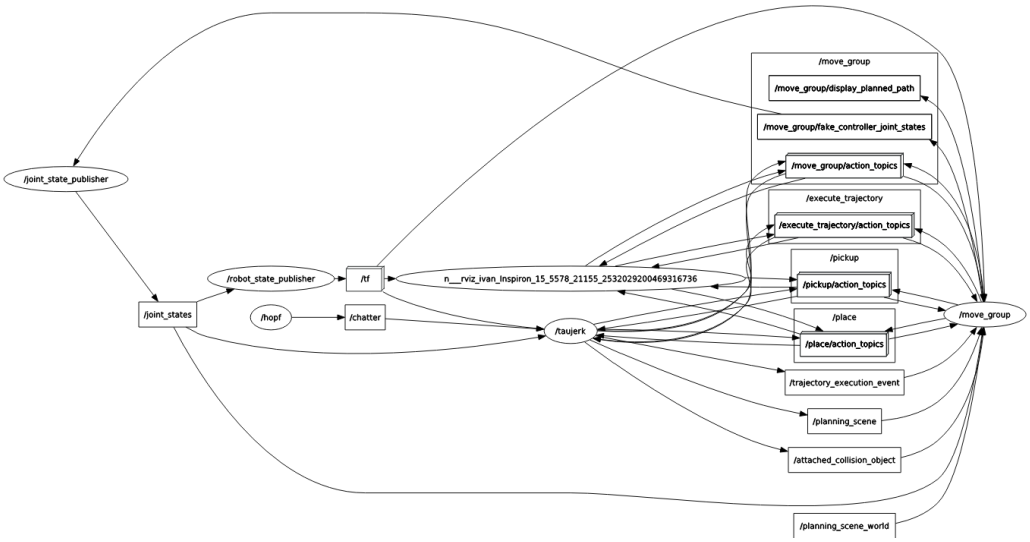


Figura 8.2: Nodos y topics con rqt graph

8.4. Modulo taujerk

Se realizo de nuestra parte fue el Move group de Taujerk en c++ para esto se crea el programa se ligan a cmake y al package.xml del paquete donde se encuentre. En este caso lo realize en el mismo paquete de Moveit tutorials aunque podria haber separado el programa y hacer un paquete aparte solo con el move group ya que todo esta ligado.

Para realizar el paquete de taujerk se obtienen las zonas destino a donde llegara el manipulador y posteriormente se realiza el control de posición con taujerk mediante la formula.

$$p(t) = \frac{\triangle p}{T^{3/k_{p,q}}}(T^3 - t^3)^{1/k_{qp}} \tag{8.1}$$

Cada nueva posición que se obtiene se manda al manipulador con RVIz y este se encarga de realizar la cinemática inversa y llegar a la posición deseada hasta alcanzar la zona destino deseada. Además se simula un obstáculo el cual debe ser evadido por el manipulador. Se utilizo una T = 20 por lo que fueron 20 iteraciones por cada zona destino pero solo 10 posiciones pues las primeras 10 son inalcanzables para el manipulador (cercanos a 0). Las zonas destino se obtiene mediante la comunicación con la red neural de hopfield.

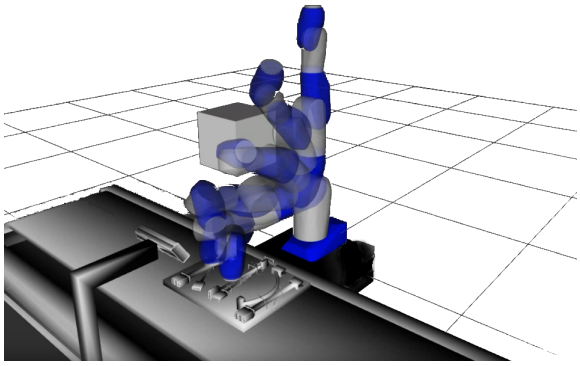


Figura 8.3: *Simulación con el modulo Tau-jerk*

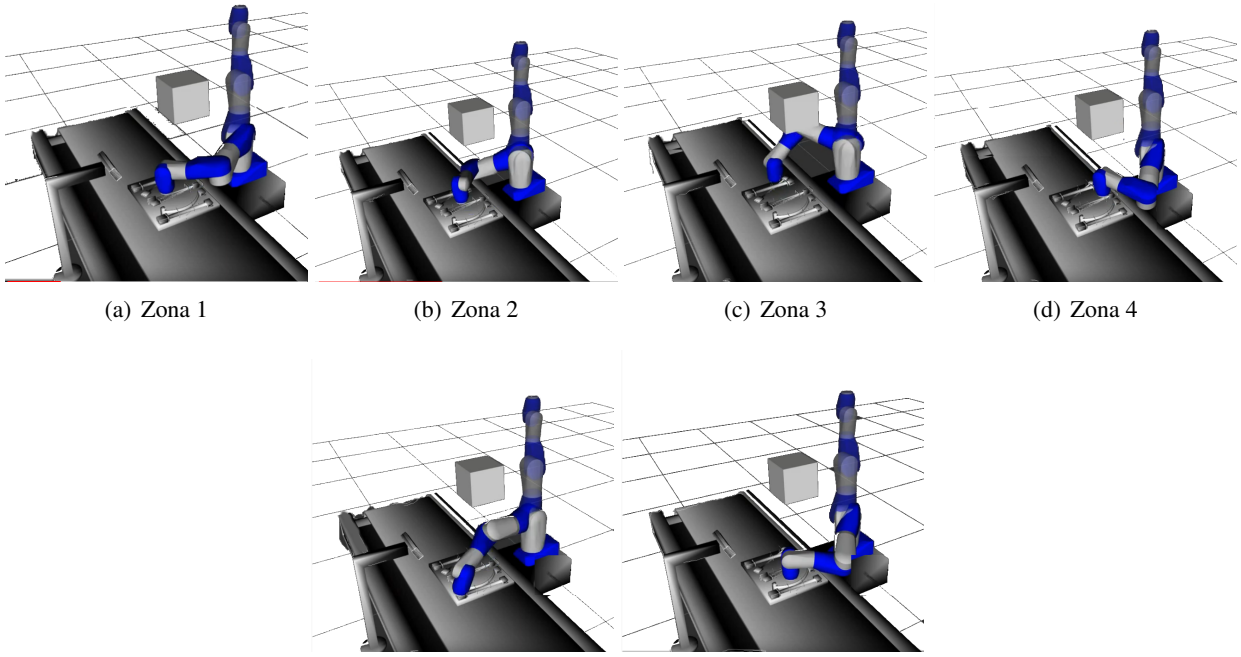
8.5. Control Variante En El Tiempo

Para programar los movegroups con el método algebraico de aproximaciones sucesivas este modulo recopila las zonas destino y utiliza la ecuaciones.

$$\vec{\phi}_{t+1} = \vec{\phi}_t + \mathbf{J}\vec{\phi}^{-1}.(\vec{P}_{ref} - \vec{P}_t) \tag{8.2}$$

$$\vec{P}_{t+1} = \vec{P}_t + \mathbf{J}\vec{\phi}^{-1}.(\vec{\phi}_{t+1} - \vec{\phi}_t) \tag{8.3}$$

Conforme se obtienen las posiciones angulares se van ingresando los ángulos de cada articulación en el robot permitiendo que la simulación en Rviz siga la trayectoria que se genera.



(a) Zona 1 (b) Zona 2 (c) Zona 3 (d) Zona 4

(e) Zona 5 (f) Zona 6

Figura 8.4: Simulación de ensamble en ROS

Capítulo 9

Conclusiones

Los proyectos robóticos integran distintas áreas de conocimiento para darle solución a un problema complejo, en este caso,se realizo la generación de trayectoria mediante la clasificación de zonas de destino en una red de hopfield. Obtenidas del procesamiento de imágenes RGB. La red hopfield es mono capa recurrente sin embargo, se realiza el proceso dos veces para utilizarla como si fuera multicapa. La red resulto ser capaz de detectar las corbatas por medio de sus colores. Además, se reducen los fallos gracias a su segunda capa aunque los resultados muestran algunos puntos repetidos se pueden incrementar las capas y la cantidad de neuronas para mejorar los resultados. También se pueden buscar soluciones de manufactura como utilizar corbatas blancas y marcadores rojos para marcar zonas especificas.

Debido a la complejidad de múltiples operaciones y la cantidad de datos en el procesado de imágenes se requirió el uso de computo paralelo, el cual fue implementado con una memoria compartida que se ejecuta en código c++ y las librerías de boost.

Se calcularon las posiciones del manipulador con el método algebraico de aproximaciones sucesivas. El seguimiento de las posiciones de manera lineal y control de las variables a través del tiempo siendo mejor opción que el método de TauJerk en manipuladores redundantes.

Para la simulación se utilizaron ambos métodos de control mediante el método de taujerk encuentran las posiciones y ROS se encarga de encontrar la cinemática inversa y la trayectoria que va a ejecutar. Mediante el sistema algebraico encontramos la posición angular del robot y ros simula la trayectoria propuesta.

Se desarrollo un algoritmo que permite generar trayectorias mediante zonas detectadas por una red hopfield, que clasifica diferentes objetos basado en el patrón de colores obtenidos por imágenes RGB. Se transmiten los datos mediante una memoria compartida y se consigue el seguimiento y control de trayectoria para un brazo robótico con 6 grados de libertad con el método de aproximaciones sucesivas, retro alimentado por la simulación de una cámara RGB.

Los resultados son simulaciones basadas en la abstracción matemática del problema en cuestión, y se consiguen probar teóricamente que una manipulador de 6 grados de libertad puede realizar una tarea en manufactura de manera autónoma mediante la retroalimentación de un sistema de visión.

Se pretende implementar el sistema de manera real para una empresa en Cd Juarez Chihuahua. En el modelo de hopfield se plantea el reconocimiento de las zonas destino por patrones formados por la figura del objeto y agregarlo en una tercera capa de la red neural. Además se pasaran los procesos a una tarjeta de prototipos como paralalle, cerebot o a un proceso en la nube.

Bibliografía

[1] E. Garcia, M. A. Jimenez, P. G. De Santos and M. Armada, "The evolution of robotics research," in IEEE Robotics & Automation Magazine, vol. 14, no. 1, pp. 90-103, March 2007. doi: 10.1109/MRA.2007.339608

[2] Noor, Ahmed K. "Potential of cognitive computing and cognitive systems." Open Engineering 5.1 (2015).pp. 75-80, DOI 10.1515/eng-2015-0008

[3] P. R. Naik, J. Samantaray, B. K. Roy and S. K. Pattanayak, "2-DOF robot manipulator control using fuzzy PD control with SimMechanics and sliding mode control: A comparative study," 2015 International Conference on Energy, Power and Environment: Towards Sustainable Growth (ICEPE), Shillong, 2015, pp. 1-6.doi: 10.1109/EPETSG.2015.7510101

[4] M. Otte and N. Correll, "C-FOREST: Parallel Shortest Path Planning With Superlinear Speedup," in IEEE Transactions on Robotics, vol. 29, no. 3, pp. 798-806, June 2013. doi: 10.1109/TRO.2013.2240176

[5] T. Tsuji, A. Jazidie and M. Kaneko, "Distributed trajectory generation for cooperative multi-arm robots via virtual force interactions," in IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), vol. 27, no. 5, pp. 862-867, Oct. 1997. doi: 10.1109/3477.623238

[6] A. Yalcin and T. O. Boucher, "Deadlock avoidance in flexible manufacturing systems using finite automata," in IEEE Transactions on Robotics and Automation, vol. 16, no. 4, pp. 424-429, Aug. 2000. doi: 10.1109/70.864237

[7] R. C. Luo and C. Kuo, "Intelligent Seven-DoF Robot With Dynamic Obstacle Avoidance and 3-D Object Recognition for Industrial Cyber–Physical Systems in Manufacturing Automation," in Proceedings of the IEEE, vol. 104, no. 5, pp. 1102-1113, May 2016. doi: 10.1109/JPROC.2015.2508598

[8] J. C. Restrepo, J. Villegas, A. Arias, S. Serna and C. Madrigal, "Trajectory generation for a robotic in a robocup test scenery using Kalman filter and B-spline curves," 2012 XVII Symposium of Image, Signal Processing, and Artificial Vision (STSIVA), Antioquia, 2012, pp. 110-115, doi: 10.1109/STSIVA.2012.6340566.

[9] L. Bassi, "Industry 4.0: Hope, hype or revolution?," 2017 IEEE 3rd International Forum on Research and Technologies for Society and Industry (RTSI), Modena, 2017, pp. 1-6. doi: 10.1109/RTSI.2017.8065927

[10] J. Kofman, Xianghai Wu, T. J. Luu and S. Verma, "Teleoperation of a robot manipulator using a vision-based human-robot interface," in IEEE Transactions on Industrial Electronics, vol. 52, no. 5, pp. 1206-1219, Oct. 2005. doi: 10.1109/TIE.2005.855696

[11] Y. Yamamoto, N. Maekawa, M. Hida, X. Yang, K. Aoyama, T. Kataoka, Y. He, K. Tatsuno, "Task performance tests on inserting the bolts by an experimental system for power distribution line maintenance - grope action under compliance control," Proc. 2012 Int. Symp.

[12] G. P. Incremona, G. De Felici, A. Ferrara and E. Bassi, "A Supervisory Sliding Mode Control Approach for Cooperative Robotic System of Systems," in IEEE Systems Journal, vol. 9, no. 1, pp. 263-272, March 2015. doi: 10.1109/JSYST.2013.2286509

[13] H. Kasahara and S. Narita, "Parallel processing of robot-arm control computation on a multimicroprocessor system," in IEEE Journal on Robotics and Automation, vol. 1, no. 2, pp. 104-113, June 1985. doi: 10.1109/JRA.1985.1087004

[14] J. P. Urban, J. L. Buessler and H. Kihl, "Parallel neural processing for the visual servoing of a robot arm" 1996 IEEE International Conference on Systems, Man and Cybernetics. Information Intelligence and Systems (Cat. No.96CH35929), Beijing, China, 1996, pp. 1806-1811 vol.3. doi: 10.1109/ICSMC.1996.565382

[15] S. Robla-Gómez, V. M. Becerra, J. R. Llata, E. González-Sarabia, C. Torre-Ferrero and J. Pérez-Oria, "Working Together: A Review on Safe Human-Robot Collaboration in Industrial Environments," in IEEE Access, vol. 5, pp. 26754-26773, 2017

[16] Burgner-Kahrs, D. C. Rucker and H. Choset, "Continuum Robots for Medical Applications: A Survey," in IEEE Transactions on Robotics, vol. 31, no. 6, pp. 1261-1280, Dec. 2015. doi: 10.1109/TRO.2015.2489500

[17] L. Huang and R. Jiang, "A new method of inverse kinematics solution for industrial 7DOF robot," Proceedings of the 32nd Chinese Control Conference, Xi'an, 2013, pp. 6063-6065.

[18] T. Kose and I. Sakata, "Analysis of Technology Convergence in Robotics and Technological Portfolios among Robot-Related Organizations," 2018 Portland International Conference on Management of Engineering and Technology (PICMET), Honolulu, HI, 2018, pp. 1-12. doi: 10.23919/PICMET.2018.8481796

[19] S. Michieletto and E. Pagello, "Competitions and industrial tasks as a way to learn basic concepts in robotics," 2018 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC), Torres Vedras, 2018, pp. 173-178. doi: 10.1109/ICARSC.2018.8374179

[20] E. Yesid Veslin, M. Suell Dutra, O. Lengerke, E. A. Carreno and M. Judtih Morales Tavera, "A Hybrid Solution for the Inverse Kinematic on a Seven DOF Robotic Manipulator," in IEEE Latin America Transactions, vol. 12, no. 2, pp. 212-218, March 2014. doi: 10.1109/TLA.2014.6749540

[21] M. Hu,C. Chen, W.Cheng, C. Chang, J. Lai, J. W, Real-Time human movement retrieval and assessment with kinect sensor, IEEE transactions of cybernetics, 45,4, pp. 743-753, Abril 2015.

[22] S. S. Young, P. D. Scott and N. M. Nasrabadi, "Object recognition using multilayer Hopfield neural network," ,Industrial Robot, abril 2020, Manuscrit id: IR-04-2020-0079.

[23] E. Salari and S. Zhang, "Integrated recurrent neural network for image resolution enhancement from multiple image frames," in IEE Proceedings - Vision, Image and Signal Processing, vol. 150, no. 5, pp. 299-, 22 Oct. 2003, doi: 10.1049/ip-vis:20030524.

[24] Real Academica de la lengua española, (17 mayo 2019),www.rae.com, Disponible [En linea]: `http://dle.rae.es/srv/search?m=30&w=robot`

[25] S. Robla-Gómez, V. M. Becerra, J. R. Llata, E. González-Sarabia, C. Torre-Ferrero and J. Pérez-Oria, "Working Together: A Review on Safe Human-Robot Collaboration in Industrial Environments," in IEEE Access, vol. 5, pp. 26754-26773, 2017, doi: 10.1109/ACCESS.2017.2773127.

[26] M. Rolf and J. J. Steil, "Efficient Exploratory Learning of Inverse Kinematics on a Bionic Elephant Trunk," in IEEE Transactions on Neural Networks and Learning Systems, vol. 25, no. 6, pp. 1147-1160, June 2014. doi: 10.1109/TNNLS.2013.2287890

[27] Capi, Genci, et al. "Optimal trajectory generation for a prismatic joint biped robot using genetic algorithms." Robotics and autonomous systems 38.2 (2002): 119-128.

[28] Kubota, Naoyuki, et al. "Trajectory generation for redundant manipulator using virus evolutionary genetic algorithm." Proceedings of International Conference on Robotics and Automation. Vol. 1. IEEE, 1997.

[29] Arakawa, Takemasa, and Toshio Fukuda. "Natural motion trajectory generation of biped locomotion robot using genetic algorithm through energy optimization." 1996 IEEE International Conference on Systems, Man and Cybernetics. Information Intelligence and Systems (Cat. No. 96CH35929). Vol. 2. IEEE, 1996.

[30] H. Su, C. Yang, G. Ferrigno and E. De Momi, "Improved Human–Robot Collaborative Control of Redundant Robot for Teleoperated Minimally Invasive Surgery," in IEEE Robotics and Automation Letters, vol. 4, no. 2, pp. 1447-1453, April 2019, doi: 10.1109/LRA.2019.2897145.

[31] Barrientos, A.; Peñín, L.F.; Balaguer, C. & Aracil, R.Fundamentos de Robótica 2 Ed.McGraw-Hill, 2007

[32] Rodriguez-Jorge, Ricardo, et al. "Weight Adaptation Stability of Linear and Higher-Order Neural Units for Prediction Applications." International Conference on Multimedia and Network Information System. Springer, Cham, 2018.

[33] K. Lee, H. Kim, S. Lee, S. Choo, S. Lee and K. Nam, "High precision hand-eye self-calibration for industrial robots," 2018 International Conference on Electronics, Information, and Communication (ICEIC), Honolulu, HI, 2018, pp. 1-2. doi: 10.23919/ELINFOCOM.2018.8330661

[34] E. M. Jafarov, M. N. A. Parlakci and Y. Istefanopulos, "A new variable structure PID-controller design for robot manipulators," in IEEE Transactions on Control Systems Technology, vol. 13, no. 1, pp. 122-130, Jan. 2005.

[35] Dongkyoung Chwa, Junho Kang and Jin Young Choi, "Online trajectory planning of robot arms for interception of fast maneuvering object under torque and velocity constraints," in IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans, vol. 35, no. 6, pp. 831-843, Nov. 2005. doi: 10.1109/TSMCA.2005.851340

[36] N. M. DiFilippo and M. K. Jouaneh, "Characterization of Different Microsoft Kinect Sensor Models," in IEEE Sensors Journal, vol. 15, no. 8, pp. 4554-4564, Aug. 2015. doi: 10.1109/JSEN.2015.2422611

[37] Asus, (17 mayo 2019),www.asus.com, Disponible [En linea]: `https://www.asus.com/us/3D-Sensor/Xtion_PRO_LIVE/overview/`

[38] S. Yeh, Y. Chiou, H. Chang, W. Hsu, S. Liu and F. Tsai, "Performance improvement of offline phase for indoor positioning systems using Asus Xtion and smartphone sensors," in Journal of Communications and Networks, vol. 18, no. 5, pp. 837-845, October 2016. doi: 10.1109/JCN.2016.000112

[39] V. Ovsyak, O. Ovsyak, D. Bui and J. Petruszka, "Research methodology of mathematical support of computing machines and systems," 2017 12th International Scientific and Technical Conference on Computer Sciences and Information Technologies (CSIT), Lviv, 2017, pp. 44-47. doi: 10.1109/STC-CSIT.2017.8098733

[40] X. Yang, S. Deb, S. Fong, X. He and Y. Zhao, "From Swarm Intelligence to Metaheuristics: Nature-Inspired Optimization Algorithms," in Computer, vol. 49, no. 9, pp. 52-59, Sept. 2016. doi: 10.1109/MC.2016.292

[41] McCarthy, John. Programs with common sense. RLE and MIT computation center, 1960.

[42] M. Deepan Raj, I. Gogul, M. Thangaraja and V. S. Kumar, "Static gesture recognition based precise positioning of 5-DOF robotic arm using FPGA," 2017 Trends in Industrial Measurement and Automation (TIMA), Chennai, 2017, pp. 1-6. doi: 10.1109/TIMA.2017.8064804

[43] P. Siagian and K. Shinoda, "Web based monitoring and control of robotic arm using Raspberry Pi," 2015 International Conference on Science in Information Technology (ICSITech), Yogyakarta, 2015, pp. 192-196. doi: 10.1109/ICSITech.2015.7407802

[44] B. C. Purba, M. Diva Pasha and Y. Bandung, "Design and Implementation of WebRTC-Based Video Conference System in Odroid Board," 2018 12th International Conference on Telecommunication Systems, Services, and Applications (TSSA), Yogyakarta, Indonesia, 2018, pp. 1-6. doi: 10.1109/TSSA.2018.8708747

[45] hardkernel, (17 mayo 2019),www.hardkernel.com, Disponible [En linea]: `https://www.hardkernel.com/shop/odroid-n2-with-4gbyte-ram/`

[46] Gentoo, (17 mayo 2019),www.gentoo.org, Disponible [En linea]: `https://www.gentoo.org/`

[47] D. Prattichizzo, "Robotics in second life," in IEEE Robotics & Automation Magazine, vol. 16, no. 1, pp. 99-102, March 2009. doi: 10.1109/MRA.2009.932131

[48] H. I. Christensen, "Formulation of a U.S. National Strategy for Robotics [Industrial Activities]," in IEEE Robotics & Automation Magazine, vol. 19, no. 2, pp. 10-14, June 2012. doi: 10.1109/MRA.2012.2193931

[49] David N Lee. General tau theory: evolution to date.Perception, 38(6):837, 2009.

[50] David N Lee. A theory of visual control of braking based on information about time-to-collision.Perception, 5(4):437–459, 1976.

[51] Zhen Zhang and Xu Yang. Bio-inspired motion planning for reaching movement of amanipulator based on intrinsic tau jerk guidance.Advances in Manufacturing, pages1–11

[52] Pendientes, "An Optimal Singularity-Free Motion Planning Method for a 6-DOF Parallel Manipulator" in IEEE Transactions on Image Processing", vol. 6, no. 3, pp. 357-372, March 1997, doi: 10.1109/83.557336.

[53] Lee, David N. "General Tau Theory: evolution to date." Perception 38.6 (2009): 837.

[54] Alberto B., *Congestión de Tráfico y Como enfrentarlo*, CEPAL, Chile, 2015.

[55] Self-Driving Uber Car Kills Pedestrian in Arizona, Where Robots Roam, (March 19 2018), By Daisuke W., The New York Times, Avaliable [Online]: `https://www.nytimes.com/2018/03/19/technology/uber-driverless-fatality.html`