ESCUELA POLITÉCNICA NACIONAL
Comisión de Investigación y Extensión

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

DEPARTAMENTO DE AUTOMATIZACIÓN Y CONTROL INDUSTRIAL

PROGRAMA INDIVIDUAL DE MAESTRÍA EN SISTEMAS DE CONTROL

DESARROLLO DE ALGORITMOS DE INTELIGENCIA ARTIFICIAL APLICADOS A SISTEMAS DISTRIBUIDOS ROBOTIZADOS

TESIS PREVIA A LA OBTENCIÓN DEL TÍTULO DE MAGÍSTER EN SISTEMAS DE CONTROL

Ing. PAULO CÉSAR LEICA ARTEAGA

SUPERVISOR: Dr. LUIS CORRALES

Quito, Febrero 2008

Dr. Luis Corrales : luisco55@yahoo.com
Ing. Paulo Leica: paulocesarl@hotmail.com

DECLARACIÓN

Yo, Paulo César Leica Arteaga, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentada para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedo mis derechos de propiedad intelectual correspondientes a este trabajo, a la Escuela Politécnica Nacional, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.

Ing. Paulo Leica Arteaga

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por el Ing. Paulo César Leica Arteaga, bajo mi supervisión.

Dr. Luis Corrales
DIRECTOR DEL PROYECTO

AGRADECIMIENTO

A mi padre celestial que con su voluntad me permitió culminar este trabajo. Al Dr. Luis Corrales por su dedicación al proyecto, al Dr. Robin Alvarez por su gran aporte a esta investigación, al Dr. Rafael Fierro por su gran contribución y colaboración a esta tesis y al Ing. Patricio Burbano por todo el apoyo mostrado en el desarrollo de este trabajo.

DEDICATORIA

A mi madre Maria Luisa por su incondicional ayuda en el día a día, a mi padre Cesar Augusto por la fortaleza y rectitud que me transmitió, a mis hermanas Rebeca y Pilar por su cariño y apoyo. A mis amigos por los ánimos brindados cada momento.

A mi enamorada Noemí, que con su compresión y paciencia siendo un pilar primordial para el desarrollo y conclusión de este trabajo.

CONTENIDO

CAPÍTULO 1
ESTADO DEL ARTE

CAPÍTULO 2
MATERIALES Y MÉTODOS

CAPÍTULO 3
PRUEBAS Y RESULTADOS

CAPÍTULO 4

CONCLUSIONES Y RECOMENDACIONES

RESUMEN

En este trabajo se pretende copiar el movimiento de un ser humano de tomar un objeto con una mano y pasarlo a la otra, reubicando dicho objeto. Ésta tarea que aparentemente es sencilla para un ser humano se vuelve compleja para un sistema robótico, ya que se debe remplazar los ojos por cámaras, los brazos por manipuladores robóticos y además otorgarle al sistema, la capacidad que tiene el ser humano para decidir cual objeto desea tomar.

Para lo cual se realizó un algoritmo de posicionamiento de dos robots manipuladores de tres grados de libertad los mismos que puedan tomar objetos y trasladarlos del un manipulador al otro. Para lograr esto se obtuvo las relaciones matemáticas que permitan posicionar el extremo del brazo robótico sobre un objeto determinado, siendo necesario obtener las coordenadas del objeto dentro del plano real, para lo cual se empleo dos cámaras. Una cámara obtendrá la imagen de un determinado número de objetos de diferentes formas (esferas, paralelepípedos, cubos, pirámides), que serán clasificados mediante un algoritmo basado en redes neuronales, permitiendo detectar y posicionar el objeto seleccionado en dos dimensiones ($2D$), mediante una segunda cámara se podrá obtener la profundidad del objeto quedando el objeto plenamente posicionado dentro del plano real.

Para validar estos algoritmos, el sistema se simuló en Realidad Virtual empleando el programa computacional Matlab (The Mathworks), para el mismo efecto se probaron los algoritmos en un brazo real de tres grados de libertad.

Para la implementación del sistema se pretende que cada manipulador sea gobernado por su propio computador mediante un Sistema Distribuido para lo cual fue necesario crear una red LAN que permita transferir información entre los dos manipuladores. La parte final del proyecto será estudiar la aplicación de los algoritmos desarrollados hacia un sistema de ensamblaje industrial.

PRESENTACIÓN

En este trabajo se pretende unir tres áreas como la robótica, inteligencia artificial y visión artificial, con la finalidad de desarrollar un sistema de selección y reubicación de objetos. Para lo cual se empezó por investigar las diferentes técnicas de visión artificial aplicables a este proyecto.

De igual manera se investigo sobre algoritmos de redes neuronales específicamente el de perceptrón multicapa, el mismo que empleará para la clasificación de objetos y fue entrenado mediante el algoritmo de retropropagación.

Se culminará esta fase preliminar, desarrollando los algoritmos de posicionamiento para los dos manipuladores de tres grados de libertad para su posterior simulación en el programa computacional Matlab(The Mathworks). Y finalmente se diseñarán pruebas que verifiquen el comportamiento de los algoritmos desarrollados y los validen o rechacen.

CAPÍTULO 1

ESTADO DEL ARTE

CAPÍTULO 1

ESTADO DEL ARTE

La robótica y la visión artificial, son hoy en día un campo de gran interés. Investigaciones en estas áreas prometen grandes avances en muchos aspectos. Así, mucho esfuerzo se está dedicando para desarrollar aplicaciones que combinan la robótica y la visión artificial, para aprovechar los beneficios de la combinación de estas dos tecnologías.

Otra disciplina que hoy por hoy se encuentra relacionada a la Robótica es la Inteligencia Artificial, esto se debe a la búsqueda de un robot que pueda remplazar a un ser humano en trabajos peligrosos, de difícil acceso o repetitivos. Las técnicas de inteligencia artificial intentan conseguir que estas máquinas trabajen sin necesidad de intervención humana constante.

Un trabajo repetitivo como la clasificación y selección de piezas, producen efectos negativos en un ser humano: cansancio, fatiga, falta de confiabilidad y repetibilidad, factores que se deterioran con el paso del tiempo. Por lo mencionado, en el presente trabajo se pretende desarrollar un sistema robotizado que tenga como objetivo global remplazar al ser humano en actividades peligrosas, monótonas o para mejorar los procesos de producción.

Así, una tarea tan simple para el ser humano como la de trasladar un objeto de una mano a otra, se vuelve compleja para un robot. Esto implica, simular una actividad biológica como reconocer el entorno, procesar información, toma de decisión, identificar, posicionar, capturar y trasladar el objeto. Todo esto involucra un desarrollo matemático complejo y una alta carga computacional.

1.1 ESTADO DEL ARTE SOBRE TÉCNICAS DE VISIÓN ARTIFICIAL

Uno de los sentidos mediante el cual se introduce la mayor parte de información al cerebro, es la visión y es así que en muchos campos tecnológicos, en especial en la robótica, se ha tratado de reproducir esta habilidad del ser humano.

Así como el ser humano se familiariza con el ambiente que lo rodea, de igual forma un robot debe hacerlo en su ambiente, para lo cual éste podría servirse de la visión artificial. Este entorno es tridimensional desde el punto de vista espacial y además es dinámico por que está sometido a cambios. Estos cambios están determinados por la variación de la posición y orientación de los objetos, debido a la propia manipulación de los robots o a la movilidad de los objetos.

La ubicación de los objetos y robots en un entorno de trabajo, pueden variar en el tiempo, por lo que es necesario dotarles de capacidades de sensorización similares a las del ser humano. La sensorización visual es una capacidad que permitirá a los robots conocer su localización respecto a otros elementos, y así proceder a manipularlos.

En la visión aplicada a la robótica se presentan dos problemas principales: localización y reconocimiento. La localización consiste en la determinación de las posiciones de los elementos que hay presentes en el entorno, al que se le conoce como escena. Una vez que ha sido registrada una vista de la misma se podría pasar a la fase de reconocimiento. El reconocimiento hace referencia a identificar las características de los elementos, y en particular de los que pueden ser objeto de una posible manipulación. Estas características permitirán diferenciar al objeto que está siendo analizado.

Realizando una comparación entre sistema de percepción visual biológico y sistema de percepción visual artificial, se puede concluir que un sistema de visión artificial, para ser considerado como tal en robótica debe constar de los siguientes elementos:

- Un sistema de entrada de información equivalente al ojo biológico. Este sistema puede ser una cámara de vídeo o su equivalente. Este sistema captura la información en forma de imágenes que más adelante serán procesadas.
- Un sistema de almacenamiento y procesamiento equivalente al cerebro, él cual puede ser implementado por cualquier dispositivo microprocesado o computarizado.
- Un sistema de salida que permita mostrar la información visual que esta siendo capturada y procesada en cada momento.
- El sistema de visión artificial deberá aprovechar al máximo el aporte de cada una de estas etapas para aproximarse al trabajo que el ser humano realiza de modo inconsciente.

1.1.1 TÉCNICAS DE VISIÓN ARTIFICIAL

El proceso de formación de imágenes es un problema de tipo geométrico, y se puede representar mediante un modelo geométrico denominado modelo de cámara.

1.1.1.1 Modelo de una cámara

El modelo de una cámara es una representación geométrica de cómo se proyectan objetos tridimensionales en imágenes. La proyección de un punto de la superficie de un objeto tridimensional sobre un plano de imagen bidimensional se puede describir como una proyección central o paralela. Como se observa en la Figura 1.1. En el modelo geométrico de una cámara aparecen distintos sistemas de coordenadas, como son: coordenadas de mundo, coordenadas de cámara y coordenadas de la imagen. [1]

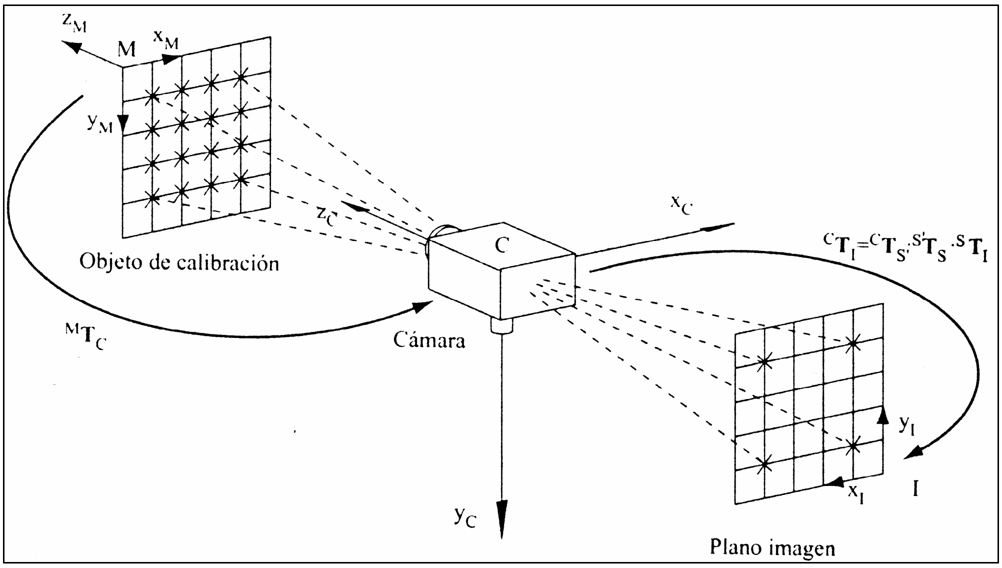


Figura 1.1. Sistema de coordenadas del mundo M

1.1.1.2 Calibración

1.1.1.2.1 Parámetros Intrínsecos y Extrínsecos

El proceso consiste en obtener información tridimensional del objeto a partir de la información bidimensional que se tiene almacenada en la imagen digital, como consecuencia del proceso de formación de imágenes.

Los parámetros intrínsecos describen la geometría y óptica del conjunto cámara y sistema de adquisiciones de imágenes, mientras que los parámetros extrínsecos son aquellos que definen la orientación y posición de la cámara respecto a un sistema de coordenadas conocido al que se llama sistema del mundo.

1.1.1.2.2 Proceso de Calibración

El objeto de utilizar visión artificial en robótica es facilitar información sobre posiciones y orientaciones de objetos en el entorno de trabajo. Los parámetros intrínsecos y extrínsecos permiten obtener información tridimensional de estos objetos a partir de información bidimensional. Así se puede entender la calibración como el proceso que permite establecer la relación entre coordenadas tridimensionales de los objetos en el entorno con sus correspondientes proyecciones bidimensionales en la imagen, y viceversa. La calibración es una técnica que facilita el reconocimiento de posiciones de objetos en el entorno de trabajo para que puedan ser manipulados por un brazo robótico.

El procedimiento que se sigue para calcular los parámetros intrínsecos y extrínsecos consiste en:

a. Determinar las posiciones tridimensionales de los puntos de un objeto conocido (puntos de referencia).
b. Determinar el valor de sus proyecciones sobre la imagen.
c. Buscar la correspondencia entre los puntos tridimensionales y sus proyecciones en la imagen, evaluando la correspondencia mediante una función de error.

Principalmente se pueden destacar dos tipos de calibración [1]; la coplanar y otra no coplanar. La tipo coplanar suele usar un plano con un patrón impreso al que se llama rejilla de calibración; y el no coplanar suele utilizar un cubo en el que las distintas caras tienen el mismo patrón impreso.

1.1.2 PREPROCESAMIENTO

El objeto del preprocesamiento de imágenes es la mejora de la apariencia de la imagen original y transformarla para conseguir una manera más apropiada de interpretar la información que en ella se refleja; dicho de otra forma, una representación más adecuada para su posterior análisis por un computador.

A continuación se describen dos técnicas de preprocesamiento: las que buscan eliminar características no deseadas y las que buscan resaltar determinadas características de la imagen. Cada una de estas se pueden clasificar según el dominio de trabajo, en técnicas espaciales o frecuenciales [1]. Es importante realizar un preprocesamiento de las imágenes captadas para que el proceso de extracción de características, a partir del cual se va a llevar a cabo el reconocimiento y localización de objetos en la escena, se haga con la mayor robustez posible.

1.1.2.1 Reducción de Ruido

De las técnicas de preprocesamiento que busca eliminar características no deseadas, las más importantes a destacar son las encargadas de la reducción de ruido. El proceso de adquisición y digitalización de una imagen conlleva una introducción de ruido. El ruido se clasifica en varios tipos [1] entre los que cabe destacar:

- Ruido Gaussiano: este se debe a las diferencias de ganancias en el sensor, ruido en los digitalizadores y en el proceso de transmisión. Este tipo de ruido produce pequeñas variaciones en la imagen. Se puede comprobar tomando dos imágenes de la misma escena en distintos momentos de tiempo y restándolas.

- Ruido impulsional: este se debe a la pérdida de señal o a la saturación de la imagen. Los píxeles con ruido toman un valor muy alto o muy bajo de luminancias.

- Ruido frecuencial: el ruido se presenta en forma de patrones en la imagen. La imagen con ruido resulta de sumar la imagen ideal más el patrón de ruido que produce algún elemento durante la captura.

A continuación se presentan algunas técnicas más usadas para eliminar ese tipo de datos no deseados (ruido):

1.1.2.1.1 Filtros Lineales

Filtrar una imagen consiste en generar una nueva que resulta de convolucionar la imagen de la que se quiere eliminar el ruido y una máscara que refleja las características del filtro que se aplica. Es importante tener en cuenta que este tipo de filtro produce un efecto de suavizado, así como también el proceso de filtrado también puede eliminar información de interés.

- Filtro de media: en este caso el filtro de media busca generar una nueva imagen donde la intensidad para cada píxel en esa nueva imagen vendrá dada por el promedio de los píxeles vecinos.

- Filtro de Gauss: el filtro de gauss se basa en una distribución de tipo Gaussiano, en el que la desviación típica de dicha distribución determina el nivel de suavizado. Así, desviaciones típicas grandes proporcionan mayor suavizado y por tanto mayor eliminación de ruido.

- Promedio de imágenes: esta técnica consiste en promediar píxeles de diferentes imágenes de una misma escena, en las que la única diferencia es la presencia de ruido en ambas. [1]

$$I(x,y)=\frac{1}{n}\sum_{k=1}^{n} I_k\,(x,y)=\frac{1}{n}\sum_{k=1}^{n} I\acute{}_k\,(x,y)+\eta\,(x,y) \qquad (1.1)$$

donde $I_k(x,y)$ representa a cada una de las imágenes reales a promediar; $I\acute{}_k\,(x,y)$ representa a esas imágenes en el supuesto de que no hubiere ruido, $\eta(x,y)$ representa el ruido en cada una de esas imágenes, y n es el número de imágenes con ruido a promediar. A mayor valor de n más ruido se elimina. Esta técnica solo es aplicable cuando la escena no varia, es decir, los objetos no se han movido, las condiciones de luz tampoco son diferentes, etc.

1.1.2.1.2 Filtros no Lineales

Los filtros que se han presentado hasta ahora presentan un importante inconveniente. Debido al afán de eliminar el ruido se produce un efecto de difuminado sobre características de interés en la imagen, como pueden ser los bordes, las esquinas, etc. Una alternativa a estos tipos de filtro son los conocidos como filtros no lineales, siendo uno de ellos el de mediana. El Filtro de mediana se basa en el reemplazo de los niveles de gris de cada píxel por la mediana de los niveles de gris de los píxeles vecinos. El único inconveniente de este filtro es que, tras su aplicación, se pierden detalles como líneas delgadas o puntos aislados en la imagen.

En una secuencia de n píxeles ordenados por valor de niveles de gris, cuyos valores de gris son i_1, i_2, …, i_n, la mediana viene dada por el elemento $i_{n-1}/2$ [1]. Si en lugar de considerar una secuencia, se considera una máscara cuadrada

$n \times m$, se elige la mediana de entre todos los valores de la máscara. La aplicación del filtro consiste en colocar la máscara sobre la imagen y sustituir el valor de nivel de gris del centro de la máscara por el de la mediana de entre todos los valores de la misma.

1.1.2.1.3 Realce de Bordes

Como técnicas de preprocesamiento que buscan resaltar determinadas características es importante mencionar las destinadas al realce de bordes. Estas técnicas son útiles cuando se quiere realzar bordes para mejorar la detección de contornos de un objeto en la escena, de modo que se facilite y mejore la localización y reconocimiento.

El realce de bordes es el efecto contrario al suavizado que se utiliza para la eliminación de ruidos. De esto se deduce que, si se busca realzar una imagen que tienen una importante componente de ruido, la imagen resultante tendrá más visible esa componente de ruido. Por ejemplo el filtro Max-Min, se trata de una técnica iterativa que consiste en comparar valores máximos y mínimos con respecto al valor de un píxel central en un pequeño entorno de vecindad. El valor del píxel central se reemplaza por cualquiera de los valores extremos en ese entorno de vecindad.

1.1.3 DETECCIÓN

Luego de haber realizado un preprocesamiento previo en la imagen capturada, es necesario detectar los objetos. La metodología para detectarlos consiste en la extracción de características que definen los posibles objetos a detectar. Las características más comunes utilizadas para detectar objetos son bordes, esquinas, texturas, color, etc.

1.1.3.1 Detección de Bordes

La detección de bordes es una técnica de gran utilidad en muchas aplicaciones de visión artificial. Esta detección facilitará la realización de tareas de segmentación y análisis de la escena, a partir de las cuales se realizará la localización y reconocimiento de los distintos objetos en una imagen para su posterior localización en el mundo.

El objetivo de las técnicas de detección de bordes es la localización de los puntos en los que se produce una variación de luminancias. Los métodos de detección de bordes se basan en cuatro tipos de transformaciones: técnicas basadas en máscaras con distintas orientaciones, aproximaciones por gradiente, aproximación Laplaciana y técnicas que ajustan valores de luminancia con modelos parametrizados de los bordes.

1.1.3.1.1 Técnicas basadas en Máscaras Orientadas

Estas consisten en realizar una correlación de la imagen con un conjunto de patrones que abarcan todas las orientaciones posibles, por ejemplo el Operador de Kirsch [1], que consiste en aplicar 8 máscaras a cada punto en la imagen, y así, de este modo, determinar la dirección y magnitud del gradiente. Existen otras máscaras propuestas, como la de Prewit o Robinson [1], pero la aplicación de este tipo de técnicas supone un costo computacional excesivo.

1.1.3.1.2 Técnicas basadas en el Operador Gradiente

Estas técnicas consisten en buscar picos que representan un cambio brusco de intensidades, estos cambios vienen dados por la primera derivada. El vector gradiente representa la máxima variación de luminancia, proporcionando una estimación de la magnitud y el sentido de la variación. En imágenes digitales se utiliza la aproximación del gradiente, de tal manera que realmente se esté trabajando con un gradiente discreto y no con el concepto de gradiente continuo (derivada). En visión artificial se define el gradiente discreto, como la diferencia entre los niveles de gris en una determinada imagen [1]:

$$\nabla I(x,y) = \left|G_x, G_y\right| = \left|I(x,y) - I(x-1,y), I(x,y) - I(x,y-1)\right| \tag{1.2}$$

El gradiente G de una imagen I se puede obtener como la convolución ($\otimes$) de la imagen con dos máscaras que representan la respuesta impulso del gradiente:

$$G_x = I(x,y) \otimes M_x(x,y) \tag{1.3}$$

$$G_y = I(x,y) \otimes M_y(x,y) \tag{1.4}$$

La magnitud del vector gradiente se obtiene a partir de las ecuaciones:

$$|\nabla I(x,y)| = \sqrt{G_X^2 + G_y^2} \tag{1.5}$$

$$|\nabla I(x,y)| = \arctan\left(\frac{G_y}{G_x}\right) \tag{1.6}$$

Debido al costo computacional de la Ecuación 1.5, la magnitud del vector gradiente se la aproxima por:

$$|\nabla I(x,y)| = |G_x| + |G_y| \tag{1.7}$$

Existe un conjunto de operadores de tipo gradiente que son más usados que los mencionados como clásicos, estos operadores son los de Roberts, Prewitt, Sobel, etc [1]. Su mayor versatilidad de uso se debe principalmente a que son menos sensibles al ruido que el operador gradiente clásico.

- Operador de Roberts: El operador de Roberts ofrece unas buenas prestaciones en cuanto a localización. El inconveniente es su sensibilidad al ruido y por lo tanto proporciona una mala detección en presencia de ruido.

- Operador de Prewitt: El operador de Prewitt mejora frente al de Roberts la detección y presencia de ruido, gracias al mayor tamaño de la máscara.

- Operador de Sobel: El operador de Sobel se distingue del operador de Prewitt en que cada píxel del entorno de vecindad es ponderado de acuerdo con la distancia de éste al central. El operador de Sobel mejora la detección en los bordes diagonales respecto al de Prewitt, mientras que éste último hace una mejor detección en los bordes horizontales y verticales.

- Operador de Frei-Chen: El operador de Frei-Chen se creó con la finalidad de proporcionar un operador que funcione del mismo modo tanto para los bordes horizontales y verticales como para los diagonales.

1.1.3.1.3 Técnicas basadas en el Operador Laplaciano

Esta técnica se basa en encontrar lo que se denomina “paso por cero” o “zero-crossing”; un paso por cero no es más que un paso de respuesta positiva a negativa, y viene dado por la segunda derivada, en una imagen, los cambios de luminancia se representan por un máximo en la primera derivada a lo largo de dicha dirección, y por un paso por cero en la segunda derivada en cualquier dirección.

En el caso de imágenes digitales es necesario utilizar una aproximación Laplaciana del mismo modo a como se hizo con el gradiente, de tal manera que realmente se esté trabajando con una Laplaciana discreta y no con el concepto de Laplaciana continua (segunda derivada). En visión artificial se puede definir la Laplaciana discreta como la diferencia de pendientes a lo largo de cada uno de los ejes x_1 e y_1, como se muestra a continuación [1]:

$$\nabla^2 \ I(x,y) = \left|G_{xx}, G_{yy}\right| \tag{1.8}$$

$$= \left|G_x(x+1,y) - G_x(x,y), G_y(x,y+1) - G_y(x,y)\right| \tag{1.9}$$

$$= \left|I(x+1,y) - 2I(x,y) + I(x-1,y), I(x,y+1) - 2I(x,y) + I(x,y-1)\right| \tag{1.10}$$

La Laplaciana L de una imagen I se puede obtener como la convolución de la imagen con una cierta máscara M .

$$L(x,y) = I(x,y) \otimes M(x,y) \tag{1.11}$$

1.1.3.1.4 Técnicas basadas en Derivadas Gaussianas: Operador de Canny.

El detector de Canny [1] se fundamenta en un proceso de optimización. Este proceso busca optimizar tres parámetros clave: la localización, la detección, y ausencia de ambiguedad; es decir, se busca minimizar la distancia entre los píxeles señalados como bordes y los bordes reales. Además, es necesario detectar todos los bordes y únicamente ellos. Y por último, debe identificarse una única respuesta para cada borde. El detector de Canny es una combinación de estos tres factores, donde se busca minimizar el producto de la localización y la detección, sujetos a la condición de la ausencia de ambiguedad.

1.1.3.2 Detección de Esquinas

La detección de esquinas tiene especial importancia en visión artificial cuando ésta se destina a labores de análisis de una escena, correspondencia en visión estéreo y análisis del movimiento. Todas ellas importantes tareas para localización en ámbitos de robots y robótica móvil.

En general, las técnicas de detección se basan en un cálculo posterior a una imagen de bordes u operan sobre la imagen en niveles de gris directamente. Generalmente cuando se habla de detección de esquinas se hace referencia a la localización de imágenes de un determinado punto característico perteneciente a bordes de objetos en dicha imagen. La detección de esquinas básicamente se puede entender, por lo tanto, como la obtención de coordenadas píxeles de esos puntos.

A la hora de evaluar un buen detector de esquinas se deben tener en cuenta cuatro factores según Z. Zheng y H. Wang [1]: detección, localización, estabilidad y complejidad. En referencia al factor de detección, no deben ser detectados como esquinas puntos de ruido en la imagen. Respecto a la localización, las esquinas deben detectarse lo más fielmente posible a su posición real en la imagen. Estabilidad, se dice que un detector es estable cuando las posiciones de las esquinas detectadas no varían cuando se toman varias imágenes de una misma escena. El algoritmo de detección será lo más sencillo posible, de su sencillez dependerá el tiempo de ejecución y de procesamiento. A continuación se describen dos grandes grupos de detectores de esquinas:

a. Los que se basan en la extracción de bordes y a partir de estos bordes buscan los puntos de máxima curvatura. Y los que a partir de la extracción de bordes realizan una aproximación poligonal y buscan las intersecciones de las líneas segmento que aparecen.
b. Los que trabajan directamente con una imagen escala de grises y extraen las esquinas directamente a partir del mapa de luminancias, entre estos se tiene: Beaudet, Kitchen y Rosendfeld, Noble, etc. [1]

1.1.3.3 Segmentación

La segmentación es un proceso en el cual se divide la imagen en regiones con significado, diferenciando los objetos que aparecen y separándolos del entorno en el que se encuentran. La segmentación es uno de los procesos más importantes en la localización de objetos en un entorno.

1.1.3.3.1 Técnicas basadas en Frontera: Transformada de Hough

Este método permite detectar ciertas formas geométricas en una imagen. Las formas geométricas más usadas que detecta la transformada de Hough son las líneas rectas y las líneas curvas (cónica). Su ventaja es su robustez ante el ruido, y sus inconvenientes su costo computacional y la necesidad de utilizarla junto con detectores de bordes.

1.1.3.3.2 Uso de transformada de Hough para detectar líneas rectas.

Para el cálculo de la transformada de Hough de líneas rectas, se toma la ecuación de la recta en coordenadas polares [1]:

$$\rho = x\cos\theta + y sen\theta \quad (1.12)$$

donde ρ es la distancia de la recta al origen y θ es el ángulo que forma la normal con el eje x.

1.1.3.3.3 Uso de transformada de Hough para detectar cónicas.

Para calcular la transformada de Hough de cónicas, se toma la ecuación de una cónica [1]:

$$r^2 = (x-a)^2 + (y-b)^2 \tag{1.13}$$

donde r es el radio que varía entre $0-\alpha$ y (a,b) son las coordenadas del centro.

1.1.3.4 Técnicas basadas en valores de Intensidad: Umbralización.

El objeto es distinguir en una imagen regiones de interés, así como eliminar todas aquellas otras que no son importantes. Se entiende por proceso de umbralización aquel por el cual se determina un nivel de luminancia, al que se llama umbral U .[1]

$$I(x,y) > U \tag{1.14}$$

donde I es el nivel de luminancia. En la práctica en una imagen aparecen varios objetos, en estos casos es imprescindible trabajar con más de un umbral, de modo que los distintos rangos en los valores de luminancia determinen las cotas superiores e inferiores para los que un píxel pertenezca a uno, u otro objeto o fondo.

$$U_i < I(x,y) < U_2 \tag{1.15}$$

$$I(x,y) \geq U_2 \tag{1.16}$$

$$I(x,y) < U_1 \tag{1.17}$$

Una manera de fijar el número de umbrales requeridos y como determinarlos, consiste en calcular los mínimos entre dos máximos en el histograma de la imagen.

1.1.3.4.1 Métodos de umbral óptimo.

Este método se basa en considerar el histograma como la suma de una serie de funciones Gaussianas solapadas, que no son más que funciones de densidad de probabilidad. Cada una de ellas representa los valores de luminancia de los píxeles de un objeto. Así, habrá una función Gaussiana por cada objeto en la imagen y otra para el fondo de la imagen. El valor de los umbrales se calcula como el mínimo valor de la suma ponderada de cada una de las varianzas de las Gaussianas presentes en el histograma.

En el caso particular de que el histograma está formado por suma de dos Gaussianas, inicialmente se parte de una aproximación del umbral al que se llama inicial. Para calcular el umbral óptimo buscado, basta recorrer los diferentes niveles de luminancias en el histograma y calcular para cada nivel la varianza [1]:

$$\sigma^2 = p_1\sigma_1^{\ 2} + p_2\sigma_2^{\ 2} \qquad (1.18)$$

Donde los coeficientes p_1, p_2 son las probabilidades de cada uno de los objetos en la imagen y el valor máximo de luminancias en la misma:

$$p_1 = \sum_{k=1}^{U\ inicial} P(k) \qquad (1.19)$$

$$p_2 = \sum_{k=1+U\ inicial}^{1} P(k) \qquad (1.20)$$

donde las medidas de ambas Gaussianas vienen dadas por las ecuaciones:

$$u_1 = \sum_{k=1}^{U\ inicial} k.\frac{P(k)}{p_1} \qquad (1.21)$$

$$u_2 = \sum_{k=1+U\ inicial}^{1} k\,\frac{P(k)}{p_2} \qquad (1.22)$$

y las varianzas por:

$$\sigma_1^{\ 2} = \sum_{k=1}^{U\ inicial} (k - u_1)^2 \frac{P(k)}{p_1} \tag{1.23}$$

$$\sigma_2^{\ 2} = \sum_{k=1+U\ inicial}^{1} (k - u_2)^2 \frac{P(k)}{p_2} \tag{1.24}$$

En este caso se suponen 2 Gaussianas, pero el método en general es n Gaussianas ($n-1$ objetos sobre el mismo fondo), al cual se le conoce como selección del umbral óptimo o método de Otsu [1].

1.1.3.4.2 Métodos basados en Píxeles de Borde.

Estos métodos proporcionan un mecanismo de segmentación menos dependiente del tamaño de los objetos representados en la imagen. Estos métodos se basan en calcular los posibles umbrales a partir de los bordes conocidos de los objetos en la imagen; de aquí se deduce que a priori es necesario realizar una detección de bordes.

Empleando el método de gradiente y Laplaciana [1] para detección de bordes, se puede formar tres niveles de luminancia, como se describen a continuación:

$$\begin{array}{ll} 0 & si\ \nabla I < U \\ - & si\ \nabla I \geq U_y\ \nabla^2 I > 0 \\ + & si\ \nabla I \geq U_y \nabla^2 I < 0 \end{array} \tag{1.25}$$

donde $0,+,-$ representan los tres nuevos niveles de luminancia. Así, todos los píxeles que no pertenecen al borde se etiquetan con el nivel 0; los píxeles situados en el nivel más bajo de luminancia se etiquetan con $(+)$, y los píxeles con el valor alto de luminancia se etiquetan con $(-)$. Después de umbralizar, es necesario someter a la imagen a un proceso de etiquetado que identifique qué píxeles pertenecen a cada objeto.

1.1.3.5 Técnicas basadas en Regiones.

Se agrupan como técnicas basadas en regiones aquellas que realizan la búsqueda de regiones en una imagen sin necesidad de utilizar mecanismos adicionales como la determinación de umbrales o detección de bordes en la imagen. Las dos técnicas más conocidas son: a) Crecimiento de Regiones y b) División y unión de regiones. [1]

1.1.4 DESCRIPTORES Y REPRESENTACIÓN DE CARACTERÍSTICAS.

Se presentan técnicas basadas en la extracción de características de las regiones detectadas mediante segmentación. La extracción de estas características permitirá describir el objeto para un posterior reconocimiento del mismo. A la hora de construir un modelo se busca que las características y descriptores cumplan en la medida de lo posible ciertos requisitos. Para poder identificar un objeto de entre un conjunto distinto es necesario que:

a. Los objetos tengan descripciones distintas, en definitiva que tengan características únicas y permitan diferenciar objetos similares sin ser iguales.
b. La descripción del objeto tiene que ser lo más completa que se pueda y no presentar ambiguedades.
c. Los descriptores usados sean invariantes. Se entiende por invariantes que no modifiquen su valor de descripción.

1.1.4.1 Área Perímetro y comparación.

El perímetro “ p ” en una región se define como el número de píxeles de borde de esa región. Y el área de una región, como el número de píxeles de borde más el número de píxeles interiores. Las características del área y perímetro varían en función del tamaño de la región representada. Así, dos objetos exactamente iguales pero con diferentes tamaños aparecerán con distinto tamaño y área. De igual modo, un mismo objeto puede variar su área y su perímetro en la imagen en función de la focal de la cámara. Para solventar este problema se introduce el concepto de compactación. El grado de compactación de una región en la imagen viene determinado por [1]:

$$c = \frac{p^2}{A} \tag{1.26}$$

1.1.4.2 Elongación.

La elongación ayuda a conocer información referente a cual es la forma del objeto, es decir, cuál es la relación entre longitud y anchura de éste. Se calcula como el cociente entre el máximo y el mínimo diámetro del objeto.

1.1.4.3 Convexidad.

La convexidad es un descriptor que da información sobre la región más pequeña que contiene el objeto de modo que dos puntos cualesquiera de éste puedan unirse mediante una línea recta, asegurando que además todos los puntos de la recta estarán dentro de la región. De ella se puede extraer propiedades conocidas que permitan caracterizar las área, perímetro, elongación, etc.

1.1.4.4 Centro de gravedad o centroide.

El centro de gravedad o centroide de un objeto $(\bar{x}, \bar{y})$ se puede definir en visión como el punto más representativo del objeto en la imagen. Este punto permitirá tener una aproximación de la localización (x, y) del objeto en la imagen.

$$\bar{x} = \frac{\sum_{x=0}^{m-1} \sum_{y=0}^{n-1} x.i(x, y)}{\sum_{x=0}^{m-1} \sum_{y=0}^{n-1} i(x, y)} \tag{1.27}$$

$$\bar{y} = \frac{\sum_{x=0}^{m-1} \sum_{y=0}^{n-1} y.i(x, y)}{\sum_{x=0}^{m-1} \sum_{y=0}^{n-1} i(x, y)} \tag{1.28}$$

donde $i(x, y)$ es el valor de intensidad de cada píxel (x, y) en la imagen I .

1.1.4.5 Momentos de inercia centrales.

El momento de inercia es una característica que permite determinar el ángulo de orientación de un objeto en la imagen. El ángulo que se calcula a partir del momento de inercia no refleja la orientación real del objeto en la imagen, pero sí permitirá una importante aproximación de la dirección que ocupa el objeto. Los momentos de inercia para calcular la orientación son los momentos centrales (los momentos centrales son invariantes a la posición). A partir de ellos se puede calcular el ángulo del eje principal del objeto cuya dirección proporciona el ángulo de orientación. [1]

$$m_{20}\sum_{x=0}^{m-1}\ \sum_{y=0}^{n-1}(x-\bar{x})^2.i(x,y) \tag{1.29}$$

$$m_{11}\sum_{x=0}^{m-1}\ \sum_{y=0}^{n-1}(x-\bar{x})(\_\,\bar{y}).i(x,y) \tag{1.30}$$

$$m_{02}\sum_{x=0}^{m-1}\ \sum_{y=0}^{n-1}(y-\bar{y})^2\,.i(x,y) \tag{1.31}$$

Donde m es el momento, los subíndices de m determinan el orden del momento e $I(x,y)$ es el nivel de luminancia del píxel situado en la coordenada (x,y) de la imagen I de tamaño m x n. El ángulo del eje principal del objeto que proporciona su orientación es:

$$\theta = -0.5.\arctan\left(\frac{2m_{11}}{m_{20}-m_{02}}\right) \tag{1.32}$$

1.1.4.6 Momentos invariantes.

Los momentos invariantes son un conjunto de momentos a partir de los cuales se pretende reconocer un objeto, independiente de cual sea su posición y orientación en la imagen.

1.1.4.7 Código de cadena.

Los códigos de cadena se utilizan para representar el contorno de un objeto, mediante unos segmentos determinados. Así, dado un objeto, se va asociando un código a cada una de las partes del contorno de éste que se va recorriendo. El procedimiento más usado es utilizar una rejilla que determine la longitud y dirección de cada parte del contorno del objeto, y a partir de esta rejilla recorrerlo en sentido horario.

1.1.4.8 Descriptores de Fourier y signaturas.

Se denominan descriptores de Fourier a aquellos conjuntos de puntos del espacio frecuencial que se han obtenido como consecuencia de aplicar la transformada discreta de Fourier a los puntos de un contorno. Otros tipos de descriptores son las signaturas de objetos. La signatura se define como la curvatura que representa la distancia de cada punto del contorno al centroide o centro de masas.

Las signaturas son descriptores invariantes ante traslación, aunque cambie la posición del objeto su signatura no varía. Sin embargo, éstas no son invariantes ante rotación o escala. Para conseguir que las signaturas sean invariantes también ante rotación y escala se procede de la siguiente manera:

a. Si se divide la función entre la distancia máxima al centroide, se obtiene invarianza ante escalamiento.
b. Si se comienza la representación eligiendo siempre como punto de comienzo el más alejado al centroide.

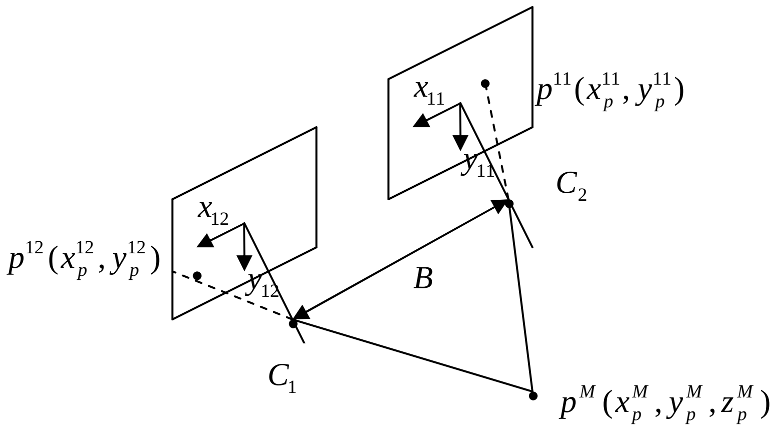
Las signaturas son sensibles a la posición del centroide y cualquier error en el cálculo de éste supone variaciones en la representación.

1.1.5 VISIÓN ESTÉREO.

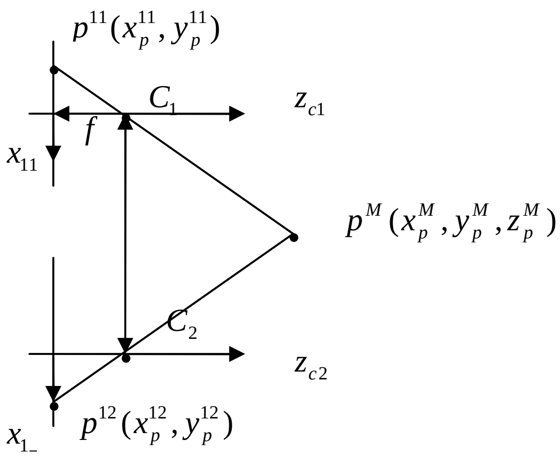
El principal objetivo de la visión estéreo cuando se trabaja en robótica es el cálculo de profundidades. El sistema estéreo de dos cámaras es conocido como sistema binocular. Se pueden tener en cuenta dos casos útiles en función de cómo se encuentren orientados los ejes de ambas cámaras: Disposición convergente o Disposición alineada.

1.1.5.1 Disposición alineada.

El modelo de ejes alineados [1] es un sistema de cámaras paralelas tal como indica en la Figura 1.2.



a)



b)

Figura 1.2. a) Esquema de cámaras con ejes alineados.
b) Ejes alineados vista superior

Sea un punto $p^M\left(x_p{}^M, y_p{}^M, z_p{}^M\right)$ en coordenadas del mundo (Figura 1.2). $p^{I1}\left(x_p{}^{I1}, y_p{}^{I1}\right)$ y $p^{I2}\left(x_p{}^{I2}, y_p{}^{I2}\right)$ las proyecciones de este punto p^M sobre cada una de las dos cámaras del par estereoscópico, y conocida la distancia B que separa los centros ópticos de ambas cámaras y sus respectivas focales (f_1, f_2), se cumple las siguientes relaciones [1]:

$$x_p{}^M = \frac{x_p{}^M . z_p{}^M}{f_1} \qquad y_p{}^M = \frac{y_p{}^{I1} . z_p{}^M}{f_1} \tag{1.33}$$

$$x_p{}^M = \frac{x_p{}^{I2} . z_p{}^M}{f_2} + B \qquad y_p{}^M = \frac{y_p{}^{I2} . z_p{}^M}{f_2} \tag{1.34}$$

De estas dos relaciones, igualando las coordenadas en x, suponiendo que la focal de ambas cámaras es la misma, se puede obtener la profundidad del punto p^M sustituyendo en las ecuaciones 1.33 y 1.34. En el valor de profundidad se puede obtener de la posición del punto p^M en el espacio:

$$\begin{aligned} x_p{}^M &= \frac{x_p{}^{I1} . B}{x_p{}^{I1} - x_p{}^{I2}} \\ y_p{}^M &= \frac{y_p{}^{I1} . B}{x_p{}^{I1} - x_p{}^{I2}} \\ z_p{}^M &= \frac{f . B}{x_p{}^{I1} - x_p{}^{I2}} \end{aligned} \tag{1.35}$$

El cálculo de la focal y de la distancia B tiene que obtenerse a partir de un proceso de calibración. Es importante destacar que en este modelo se cumple $y_p{}^{I1} = y_p{}^{I2}$. Y que la diferencia $x_p{}^{I1} - x_p{}^{I2}$, determina la disparidad entre ambas cámaras.

Si se observa la relación entre profundidad $z_p{}^M$ y disparidad $x_p{}^{I1} - x_p{}^{I2}$, se comprueba que profundidad y disparidad son conceptos inversamente proporcionales. De aquí se deduce que es más preciso obtener distancias de objetos cercanos a las cámaras que de objetos alejados.

1.1.5.2 Disposición convergente.

El modelo de ejes convergentes [1] es un sistema de cámaras en el cual los ejes ópticos de ambas cámaras se cruzan en el espacio. Sea un punto $p^M\left(x_p{}^M, y_p{}^M, z_p{}^M\right)$ en coordenadas del mundo y $p^{I1}\left(x_p{}^{I1}, y_p{}^{I1}\right) \wedge p^{I2}\left(x_p{}^{I2}, y_p{}^{I2}\right)$ las proyecciones de ese punto p^M sobre cada una de las dos cámaras del par estereoscópico. Conocida la distancia B que separa los centros ópticos de ambas cámaras y la focal f, se puede obtener un sistema lineal de ecuaciones para calcular las coordenadas del punto p^M en el mundo $\left(x_p{}^M, y_p{}^M, z_p{}^M\right)$. Al proceso que consiste en resolver este sistema de ecuaciones se le llama proceso de triangulación.

En el caso de ejes convergentes, la disparidad tendrá dos componentes, ya que ahora se cumple $y_p{}^{I1} \neq y_p{}^{I2}$, y la distancia vendrá definida por $d = \left(x_p{}^{I1} - x_p{}^{I2}, y_p{}^{I1} - y_p{}^{I2}\right)$. En un sistema de ejes convergentes pueden aparecer puntos en el espacio que tienen la misma disparidad. A la región del espacio que agrupa todos los puntos de misma disparidad se le llama región de Horopter [1].

En un sistema de ejes alineados, la región de Horopter es un plano paralelo a la línea base y conforme los ejes hacen converger, la región de Horopter se va curvando.

1.2 ESTADO DEL ARTE SOBRE MANIPULADORES INDUSTRIALES

En la actualidad el uso de robots manipuladores se ha generalizado. Estos son empleados en un sin número de aplicaciones: ensamblaje de circuitos impresos, cirugías, destacándose su aplicación en la industria automotriz: en aplicaciones como soldadura de puntos, pintura spray, manipulación de partes de carrocería, chasis y motor.

Un robot industrial es un manipulador programable multifuncional diseñado para mover materiales, piezas, herramientas o dispositivos especiales, mediante movimientos variados, que son programados para la ejecución de distintas tareas. Estos se caracterizan por tener un funcionamiento repetitivo, son precisos, rápidos y de alta repetibilidad, con percepción limitada.

1.2.1 MORFOLOGIA DE UN ROBOT MANIPULADOR

Un robot está formado por los siguientes elementos [2]: estructura mecánica, transmisiones, actuadores, sensores, elementos terminales y controlador. Aunque los elementos empleados en los robots no son exclusivos de estos, las altas prestaciones que se exigen a los robots han motivado que en ellos se empleen elementos con características específicas. La constitución física de la mayor parte de los robots industriales guarda cierta similitud con la anatomía de las extremidades superiores del cuerpo humano, por lo que, en ocasiones, para hacer referencia a los distintos elementos que componen el robot, se usan términos como cintura, hombro, brazo, codo, muñeca, etc.

Estructura mecánica de un robot son los: enlaces (grados de libertad), tipos de articulaciones, configuraciones básicas, elementos finales, volumen de trabajo, transmisores y reductores, actuadores (Eléctricos, Hidráulicos, Neumáticos) y sensores (Elementos Terminales o actuadores finales).

1.2.1.1 Estructura mecánica de un robot.

Un robot manipulador está típicamente formado por una serie de elementos (segmentos, eslabones o links) unidos mediante articulaciones que permiten un movimiento relativo entre cada dos eslabones consecutivos y este movimiento es producido por los actuadores.

El movimiento de la articulación puede ser: de desplazamiento, de giro y combinación de ambos. Cada uno de los movimientos independientes que puede realizar cada articulación con respecto a la anterior se denomina grado de libertad. El número de GDL (Grados de Libertad) del robot viene dado por la suma de los GDL de las articulaciones que lo componen. Los grados de libertad equivalen al número de parámetros independientes que fijan la ubicación del elemento terminal.

1.2.1.1.1 Transmisores y Reductores.

Los transmisores son elementos encargados de transmitir el movimiento desde los actuadores hasta las articulaciones y los reductores o engranajes son elementos encargados de adaptar el par y la velocidad de la salida del actuador a los valores adecuados para el movimiento de los elementos del robot.

1.2.1.1.2 Actuadores.

Los actuadores generan el movimiento de los elementos del robot. La mayoría de los actuadores simples controlan únicamente 1 GDL (izq-der, arriba-abajo). Estos pueden clasificarse en [2]:

- Actuadores Eléctricos: Un sistema eléctrico se pude considerar seccionador cuando producen determinada acción: conmutadores de tipo ON-OFF selenoides, o motores eléctricos en los que una corriente es usada para producir un movimiento de rotación continua, en robótica interesa aquellos accionadores cuya acción produce un movimiento capaz de mover una articulación. Según el tipo de corriente y el modo de funcionamiento, los motores eléctricos se pueden clasificar en tres categorías: motores de corriente alterna, motores de corriente continua y motores paso a paso.

- Actuadores Hidráulicos: Ejercen presiones aplicando el principio de la prensa hidráulica de Pascal, fluido que circula por tuberías a presión, útil para levantar grandes cargas. Se controlan con servo válvulas (Motor eléctrico de baja velocidad y alto torque) que controlan el flujo que circula.

- Actuadores Neumáticos: Son aquellos movidos por fluido compresibles; generalmente aire. Este suele mover pistones lineales que controlan válvulas neumáticas, los mismos que son muy seguros y robustos. Sin embargo, tienen poca exactitud en la posición final, son difíciles de controlar el Aire es demasiado compresible.

1.2.1.1.3 Sensores.

Los sensores internos son aquellos que dan información como posición, orientación, velocidad, aceleración del robot, mientras que los sensores externos dan información sobre el entorno del robot (proximidad, tacto, fuerza, visión).

1.2.1.1.4 Elementos terminales o actuadores finales.

Para las aplicaciones industriales, las capacidades del robot básico deben aumentarse por medio de dispositivos adicionales. En robótica, el término de actuador final se utiliza para describir la mano o herramienta que está unida a la muñeca. El actuador final representa la herramienta especial que permite al robot de uso general realizar una aplicación particular, y debe diseñarse específicamente para dicha aplicación.

Los actuadores finales pueden dividirse en dos categorías [2]: pinzas y herramientas. Las pinzas se utilizan para tomar un objeto, normalmente la pieza de trabajo, y sujetarlo durante el ciclo de trabajo del robot. Hay una diversidad de métodos de sujeción que pueden utilizarse, además de los métodos mecánicos obvios de agarre de la pieza entre dos o más dedos. Estos métodos suplementarios incluyen el empleo de imanes, ganchos y cucharas.

Una herramienta se utiliza como actuador final en aplicaciones en donde se exija al robot realizar alguna operación sobre la pieza de trabajo. Estas aplicaciones incluyen la soldadura por puntos, la soldadura por arco, la pintura por pulverización y las operaciones de taladro. En cada caso, la herramienta particular está unida a la muñeca del robot para realizar la operación.

1.2.1.2 Descripción de la posición y orientación.

Un robot, para llevar a cabo cualquier tarea, al igual que si se realiza manualmente necesita localizarse (posicionarse y orientarse adecuadamente en el espacio), para lo cual se requiere determinar un sistema de referencia el cual permita describir la posición y orientación del extremo del robot (pinza o herramienta).

Con este propósito Forest [3] introduce el concepto de coordenadas homogéneas y la matriz de transformación homogénea que son empleadas en robótica para determinar conjuntamente en una sola matriz la posición y orientación de un objeto respecto de un sistema de referencia. De allí la matriz de transformación homogénea es una matriz de 4×4 compuesta por cuatro submatrices.

$$T = \left[\begin{array}{c|c} \frac{rotación}{perspectiva} & \frac{traslación}{escalado} \end{array} \right] = \left[\begin{array}{c|c} 3x3 & \begin{array}{c} 3 \\ x \\ 1 \end{array} \\ \hline 1x3 & 1x1 \end{array} \right] \qquad (1.36)$$

Los vectores de perspectiva y escalado no tienen sentido en robótica ya que se trabaja con posiciones y orientaciones reales en el espacio tridimensional de objetos reales, así el vector perspectiva toma el valor de cero y de la misma forma el escalado toma el valor de uno.

Las tres primeras columnas de la matriz de transformación homogénea representan la dirección de los ejes principales de un sistema asociado a un objeto O, referenciados con respecto a un sistema de referencia M, mientras que la cuarta columna representa la posición del sistema asociado al objeto O, respecto al sistema de referencia M.

De esta manera quedaría definida la matriz de transformación que permite localizar un sistema O respecto de otro M. Sin embargo, en otras ocasiones tal vez sea necesario conocer la localización de M respecto a O, lo que corresponde a determinar la matriz de transformación inversa.

1.2.1.2.1 Traslación.

Se puede considerar tres traslaciones básicas sobre cada uno de los ejes principales de un sistema de referencia tal como se muestra en Figura 1.3.

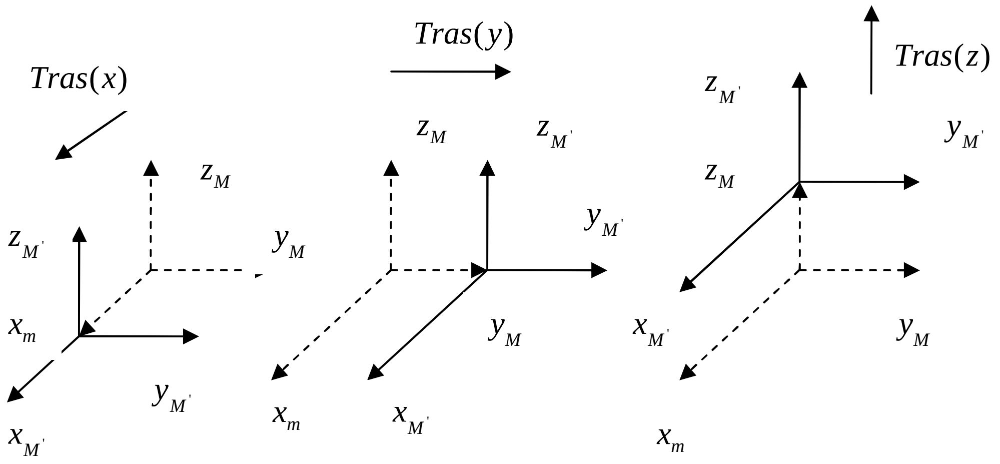


Figura 1.3. Traslaciones Básicas

La matriz de transformación homogénea resultante estará compuesta por una submatriz de rotación que será la identidad, ya que no se ha producido ninguna rotación respecto de la posición inicial, y un vector de posición final cuyas componentes son las magnitudes de las traslaciones efectuadas sobre cada una de los ejes principales. [3]

$$Tras(p) = Tras(x, y, z) = \left[\begin{array}{ccc|c} 1 & 0 & 0 & x \\ 0 & 1 & 0 & y \\ 0 & 0 & 1 & z \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \qquad (1.37)$$

1.2.1.2.2 Rotación.

Se puede obtener tres rotaciones básicas considerando tres giros con respecto a cada uno de los ejes principales de un sistema de referencia. Considerando la rotación respecto al eje x determinado por el ángulo α, rotación respecto al eje y determinado por β y rotación respecto a z, determinado por y como se muestra en las siguientes Figuras 1.4, 1.5, 1.6:

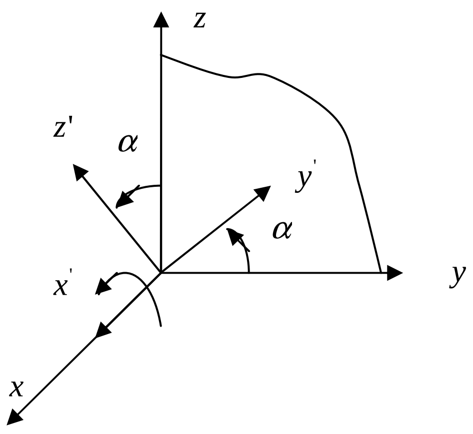


Figura 1.4. Giro Respecto al Eje x

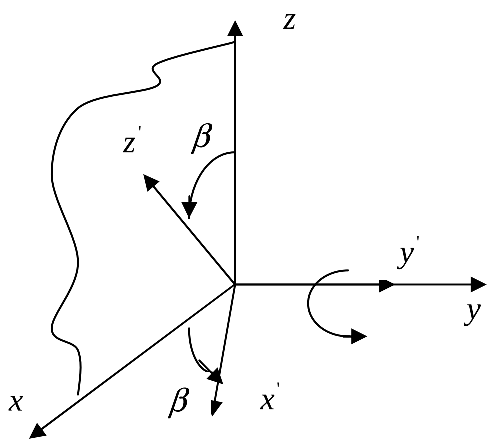


Figura 1.5. Giro Respecto al Eje y

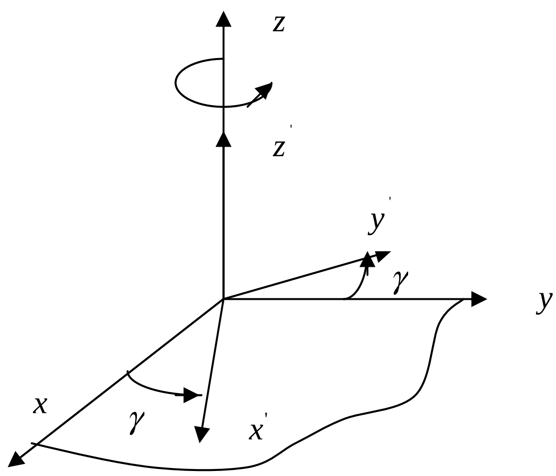


Figura 1.6. Giro Respecto al Eje z

Las matrices que definen las rotaciones son [3]:

$$Rot(x,\alpha)=\left[\begin{array}{ccc|c} 1 & 0 & 0 & 0 \\ 0 & \cos(\alpha) & -sen(\alpha) & 0 \\ 0 & sen(\alpha) & \cos(\alpha) & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \tag{1.38}$$

$$Rot(y,\beta)=\left[\begin{array}{ccc|c} \cos(\beta) & 0 & sen(\beta) & 0 \\ 0 & 1 & 0 & 0 \\ -sen(\beta) & 0 & \cos(\beta) & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \tag{1.39}$$

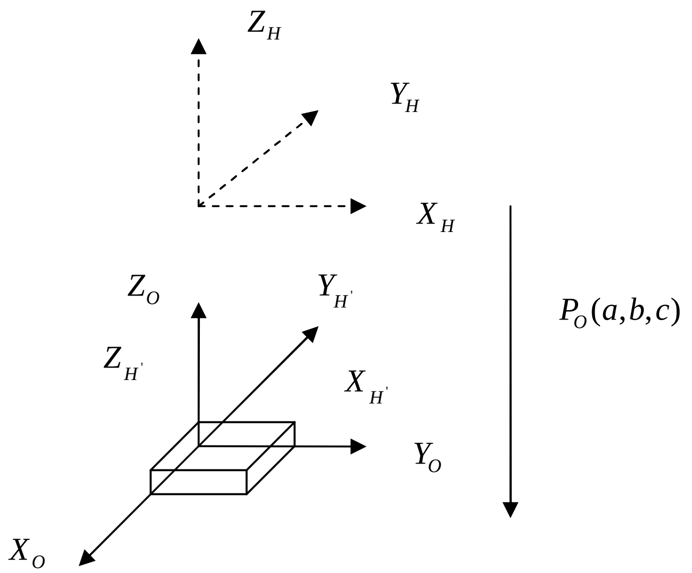
$$Rot(z,\gamma)=\left[\begin{array}{ccc|c} \cos(\gamma) & -sen(\gamma) & 0 & 0 \\ sen(\gamma) & \cos(\gamma) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \tag{1.40}$$

1.2.1.2.3 Composición de Transformaciones.

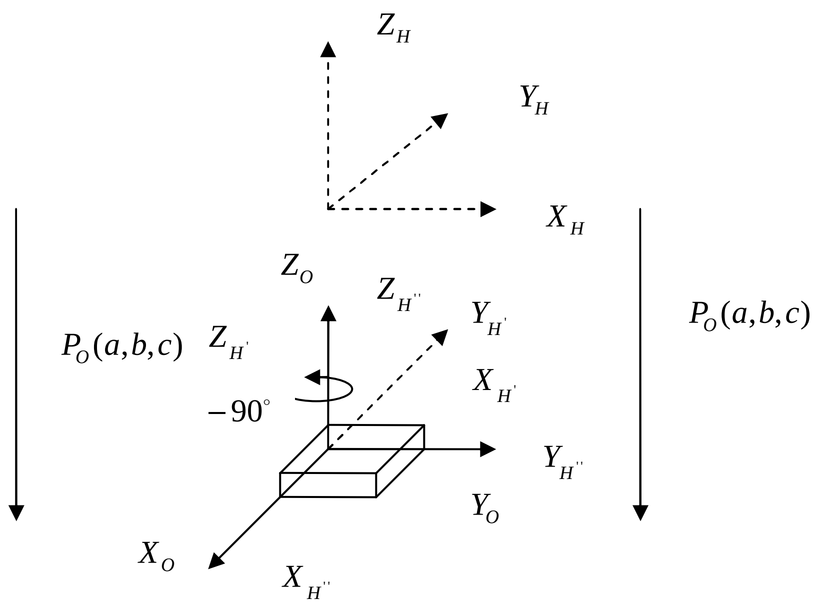
Es habitual descomponer una transformación total en una serie de transformaciones básicas de traslación si lo que cambia es la posición del objeto con respecto a un sistema de referencia y/o de rotación, si lo que se produce es un giro del objeto con respecto al sistema de referencia. La composición de transformaciones, al estar representada por matrices, supone que el orden en que se aplica cada una de las transformaciones básicas que la componen es relevante, puesto que el producto entre matrices no es conmutativo. Es así que no es lo mismo trasladar y girar con respecto a un sistema de referencia, que girar y trasladar con respecto al mismo sistema de referencia.

Por ejemplo, si se pretende trasladar al objeto y posteriormente rotarlo, como se observa en la Figura 1.7, primero se realizaría una traslación del sistema H asociado al extremo del robot (pinza). Esta traslación queda determinada mediante la ecuación [3]:

$$p^{H} = {}^{H}Tras_{H'}(a,b,c).p^{H'} \qquad (1.41)$$



a)



b)

Figura 1.7. a) Posición de un Sistema H respecto de un Objeto.
b) Orientación de un Sistema H respecto de un Objeto.

La segunda operación es un giro del extremo del robot (pinza), pero ahora aplicada no sobre la posición inicial que ocupaba si no la que tiene en ese momento. Por tanto, entre los sistemas H', posición intermedia, y el H'', posición final, se puede considerar una relación similar a [3]:

$$p^{H} = {}^{H}Tras_{p'}(a,b,c)\,.\,{}^{H''}Rot_{H'}(z,-90).p^{H''} \qquad (1.42)$$

1.2.1.3 Cinemática.

Estudiar la cinemática del manipulador es necesario pues es la base para desarrollar la simulación, para poder observar la trayectoria del manipulador y evaluar si los resultados de los algoritmos son correctos. Para la evaluación se puede resolver dos problemas: Cinemática directa e inversa. La cinemática directa determina la posición y orientación del extremo del robot según unos valores conocidos de las variables articulares.

Se considera la cinemática inversa, para que determine que valores tienen que tomar las variables articulares para que el extremo del robot se encuentre en una posición y orientación dada. Sin embargo, se debe partir del análisis matemático de la cinemática directa.

1.2.1.3.1 Cinemática Directa.

Para desarrollar la cinemática del manipulador se utilizan matrices de transformación homogénea. En la Figura 1.8 se muestra una imagen del robot el cual cuenta con seis grados de libertad.

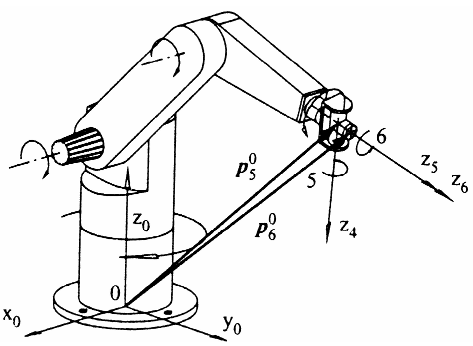


Figura 1.8. Gráfica del Robot

Las matrices de transformación homogénea para la cinemática directa son representadas por la letra T, con un superíndice que representa el eje de coordenadas inicial y un subíndice que representa el eje de coordenadas final. Los términos q_i representan el ángulo en radianes de cada eslabón, partiendo desde la base y hasta el último eslabón. Los L_i representan la longitud de cada eslabón, de igual forma desde la base y hasta el último eslabón tal como se muestra en las Figuras 1.9 y 1.10.

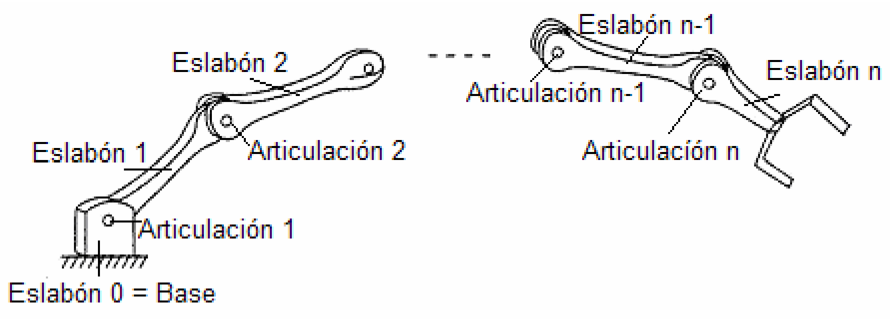


Figura 1.9. Numeración de eslabones y articulaciones

Para poder resolver el problema cinemático directo, se establece un método sistemático que puede dividirse en tres fases: a) Definición de los parámetros Denavit Hartemberg [1], b) Asignación de Sistemas de Referencia y c) Transformaciones Homogéneas.

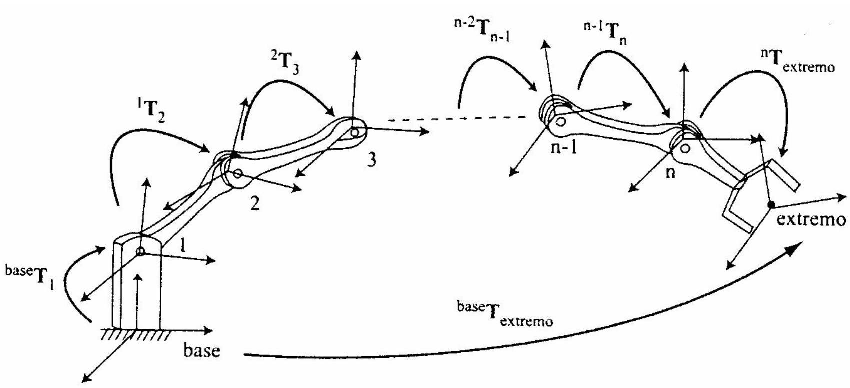


Figura 1.10. Transformaciones de los distintos sistemas del Robot

Para determinar los parámetros de Denavit Hartemberg se definen los siguientes parámetros: a_i , distancia entre los ejes i e $i+1$ de las articulaciones a lo largo de la normal común: α_i , Angulo de existencia entre los ejes i e $i+1$ de las articulaciones si estos se cortasen en los puntos de corte de la línea normal común: d_i , Distancia entre las intersecciones de las normales comunes al eje de la articulación i, medida a lo largo de dicho eje: θ_i , Angulo que existiría entre las líneas normales comunes al eje de la articulación i si se cortasen en el mismo punto del eje de la articulación.

El análisis de asignación de Denavit Hartemberg, asignación de sistemas de referencia y de transformaciones homogéneas, se detallarán de mejor manera en el capítulo 2.

1.2.1.3.2 Cinemática Inversa.

Como se ha visto, en la cinemática directa se resuelve el problema de determinar la posición y orientación del extremo del robot según los valores conocidos de las variables articulares. Sin embargo, quizás el problema que resulta más interesante, desde un punto de vista práctico, es determinar qué valores deben tomar las variables articulares para que el extremo del robot se encuentre en una posición y orientación dada.

Para resolver el problema cinemático inverso se puede optar por dos vías: solución numérica o solución cerrada. La primera se desecha por que resulta más lenta y costosa computacionalmente, mientras que la segunda hace referencia a soluciones algebraicas (establecer un sistema de ecuaciones) y solución geométrica (descomponer al robot en varios planos geométricos resolviéndolo por trigonometría).

1.2.1.3.3 Cinemática de Movimiento.

Hasta ahora se a estudiado la cinemática desde un punto de vista estático; es decir, tan solo se han establecido las relaciones entre el espacio articular y el cartesiano considerando una posición estática, en ausencia del movimiento del robot. Sin embargo, también interesa conocer las relaciones existentes que determinan con que velocidad lineal y angular se mueve el extremo del robot, para lo cual se puede emplear la matriz Jacobiana [4].

La matriz Jacobiana es una matriz de derivadas que es un conjunto de m funciones que dependen de n variables independientes que en forma matricial se puede expresar como sigue:

$$\begin{bmatrix} \dot{x}^{0}_{extremo} \\ \dot{y}^{0}_{extremo} \\ \dot{z}^{0}_{extremo} \\ \dot{\alpha}^{0}_{extremo} \\ \dot{\beta}^{0}_{extremo} \\ \dot{\gamma}^{0}_{extremo} \end{bmatrix} = J(q).\begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ . \\ . \\ . \\ \dot{q}_n \end{bmatrix} \tag{1.43}$$

O lo que es igual:

$$\begin{bmatrix} v^{0}_{extremo} \\ w^{0}_{extremo} \end{bmatrix} = \begin{bmatrix} Jv(q) \\ Jw(q) \end{bmatrix}.\dot{q} = J(q).\dot{q} \tag{1.44}$$

Al igual que ocurre en la cinemática de posición, quizá resulte más interesante la relación inversa, determinar las velocidades articulares para que el extremo del robot se mueva a una velocidad lineal y angular determinada. Considerando esta perspectiva se emplean las siguientes expresiones [4]:

$$\begin{aligned} q_1 &= f_1(x^{0}_{extremo}, y^{0}_{extremo}, z^{0}_{extremo}, \alpha^{0}_{extremo}, \beta^{0}_{extremo}, \gamma^{0}_{extremo}) \\ q_2 &= f_1(x^{0}_{extremo}, y^{0}_{extremo}, z^{0}_{extremo}, \alpha^{0}_{extremo}, \beta^{0}_{extremo}, \gamma^{0}_{extremo}) \\ . & \\ q_n &= f_1(x^{0}_{extremo}, y^{0}_{extremo}, z^{0}_{extremo}, \alpha^{0}_{extremo}, \beta^{0}_{extremo}, \gamma^{0}_{extremo}) \end{aligned} \tag{1.45}$$

Por lo que derivando respecto al tiempo y expresado en forma matricial se tiene:

$$\dot{q} = J^{-1}(x^0_{extremo}, y^0_{extremo}, z^0_{extremo}, \alpha^0_{extremo}, \beta^0_{extremo}, \gamma^0_{extremo}).\begin{bmatrix} v^0_{extremo} \\ w^0_{extremo} \end{bmatrix}$$

1.3 ESTADO DEL ARTE SOBRE REDES NEURONALES DE PERCEPTRON MULTICAPA.

1.3.1 INTRODUCCION

La inteligencia artificial estudia como lograr que las máquinas realicen tareas que, por el momento, son realizadas mejor por los seres humanos. Actualmente, el mayor esfuerzo en la búsqueda de la inteligencia artificial se centra en el desarrollo de sistemas de procesamientos de datos que sean capaces de imitar a la inteligencia humana, realizando tareas que requieran aprendizaje, solución de problemas y toma de decisiones. Así, la finalidad de la inteligencia artificial consiste en crear teorías y modelos que muestren la organización y funcionamiento de la inteligencia.

Una de las aplicaciones de la inteligencia artificial es en el campo de la robótica, las mismas que van tomadas de la mano ya que la una se encarga de la parte mecánica, y la otra de la parte analítica. La robótica es el diseño, fabricación y utilización de máquinas automáticas programables con el fin de realizar tareas repetitivas como el ensamble de automóviles, aparatos, etc. y otras actividades, en cambio la inteligencia artificial es la parte analítica o la parte que determina la acción de los robots. Cabe destacar que el desarrollo de estas tecnologías no pretende reemplazar al ser humano sino que tratan de mejorar el estilo de vida del mismo.

Uno de los temas importantes en el desarrollo de la inteligencia artificial es el aprendizaje. La capacidad de aprendizaje adaptivo es una de las características más atractivas de las redes neuronales, las mismas que forman parte de la inteligencia artificial. Las redes neuronales pueden aprender de diferentes patrones mediante ejemplos y entrenamiento, no es necesario que se elabore modelos a priori, ni se necesita especificar funciones de distribuciones de probabilidad.

La presente aplicación se basará específicamente en la red “backpropagation” cuyas aplicaciones más importantes son reconocimiento de caracteres, reconocimiento de patrones, reconocimiento de lenguaje hablado, clasificación de información, etc.

1.3.2 TOPOLOGÍA DE LAS REDES NEURONALES

La topología o arquitectura de las redes neuronales consiste en la organización y disposición de las neuronas en la red formando capas o agrupaciones de neuronas más o menos alejadas de la entrada y salida de la red. Las redes neuronales en términos topológicos se clasifican en: redes con una sola capa o nivel de neuronas y las redes con múltiples capas (2,3, etc). [5]

1.3.2.1 Redes monocapa

En este tipo de redes se establecen conexiones laterales entre las neuronas que pertenecen a la única capa que constituye la red, también pueden existir conexiones autorrecurrentes (salida de una neurona conectada a su entrada) pero en algunos modelos no son utilizadas.

1.3.2.2 Redes multicapa

Las redes multicapa son aquellas que disponen de conjuntos de neuronas agrupadas en varios niveles o capas (2,3, etc). En estos casos, una forma para distinguir la capa a la que pertenece una neurona, consistiría en fijarse en el origen de las señales que recibe a la entrada y el destino de la señal de salida.

Normalmente, todas las neuronas de una capa reciben señales de entrada de otra capa anterior, más cercana a la salida de la red. A estas conexiones se las denomina conexiones hacia delante (feedforward). Sin embargo, en un gran número de estas redes también existe la posibilidad de conectar las salidas de las neuronas de capas posteriores a las entradas de las capas anteriores, a estas conexiones se las denomina conexiones hacia atrás (feedback). [5]

Estas dos posibilidades permiten distinguir entre dos tipos de redes con múltiples capas: las redes con conexiones hacia delante o redes feedforward, y las redes que disponen de conexiones tanto hacía adelante como hacia atrás o redes feedforward/feedback. [5]

1.3.2.3 Redes con conexiones hacia delante (feedforward)

Este grupo de redes se caracterizan por arquitecturas en niveles y conexiones estrictamente hacia delante entre las neuronas. Entre este grupo se tiene las siguientes: Perceptron, Adaline, Madaline, Linear Adaptive Memory, Drive-Reinforce-Ment y Back-Propagation. [5]

1.3.2.4 Redes con conexiones hacia delante y hacia atrás (feedforward/feedback)

Esta red permite circulación de información tanto hacia delante (forward) como hacia atrás (backward) cuando la red esta en funcionamiento. Esto es posible por cuanto las neuronas tienen conexiones feedforward y conexiones feedback. Estas redes suelen ser bicapa (dos capas), por lo que existen dos conjuntos de pesos: conexiones feedforward de la primera capa hacia la segunda y las conexiones feedback de la segunda a la primera. Los valores de los pesos de estos dos tipos de conexiones son diferentes.

La estructura bicapa es usada para realizar una asociación de una información o patrón de entrada con otra información o patrón de salida en la segunda capa. Esto se conoce como heteroasociación pero también es utilizada para la clasificación de parámetros.

Algunas redes de este tipo se basan en la resonancia lo cual implica que la información en la primera y segunda capa interactúa entre sí hasta alcanzar un estado estable, lo cual permite un mejor acceso a la información almacenada en la red.

Existen dos modelos de red feedforward/feedback más conocidos: la red ART (Adaptive Resonante Theory) y la red BAM (Bidirectional Associative Memory) [5]. En este grupo de redes existen algunas que tienen conexiones laterales entre neuronas de la misma capa, estas conexiones se diseñan: como conexiones excitadoras permitiendo la cooperación entre las neuronas correspondientes o como inhibidoras, estableciéndose una competición entre las neuronas correspondientes; entre estas se tiene la red Cabam. [5]

Existe un tipo de red feedforward/feedback multicapa denominada Neocognitron [5], en las que las neuronas se disponen en planos superpuestos, lo cual permite que puedan eliminarse las variaciones geométricas o distorsiones que presenten las informaciones o patrones de entrada a la red.

1.3.3 MECANISMOS DE APRENDIZAJE

El aprendizaje es el proceso por el cual una red neuronal modifica sus pesos en respuesta a una información de entrada. Los cambios que se producen durante el proceso de aprendizaje se reducen a la destrucción, modificación y creación de conexiones entre las neuronas.

En los modelos de redes neuronales artificiales, la creación de una nueva conexión implica que el peso de la misma pasa a tener un valor distinto de cero, así mismo una conexión se destruye cuando su peso pasa a ser cero.

Durante el proceso de aprendizaje, los pesos de las conexiones de la red sufren modificaciones, por tanto se puede afirmar que este proceso ha terminado cuando los valores de los pesos permanecen estables ($dw_{ij} / dt = 0$).

Un criterio para diferenciar las redes de aprendizaje se basan en considerar si la red puede aprender durante su funcionamiento habitual (ON-LINE) o si el aprendizaje supone la desconexión de la red (OFF-LINE).

Otro criterio para la clasificación de las redes neuronales es de acuerdo al tipo de aprendizaje, se divide en: redes neuronales con aprendizaje supervisado y redes neuronales con aprendizaje no supervisado.

1.3.3.1 Redes con aprendizaje supervisado

El aprendizaje supervisado se realiza mediante un entrenamiento controlado por un agente externo que determina la respuesta que deberá generar la red a partir de una entrada determinada.

El supervisor controla la salida de la red y en el caso de que ésta no coincida con la deseada, se procederá a modificar los pesos de las conexiones, con el fin de conseguir que la salida obtenida se aproxime a la deseada. Existen tres formas de desarrollar un aprendizaje supervisado: aprendizaje por corrección de error, aprendizaje por refuerzo y aprendizaje estocástico. En el presente trabajo se ampliará el aprendizaje por corrección de error.

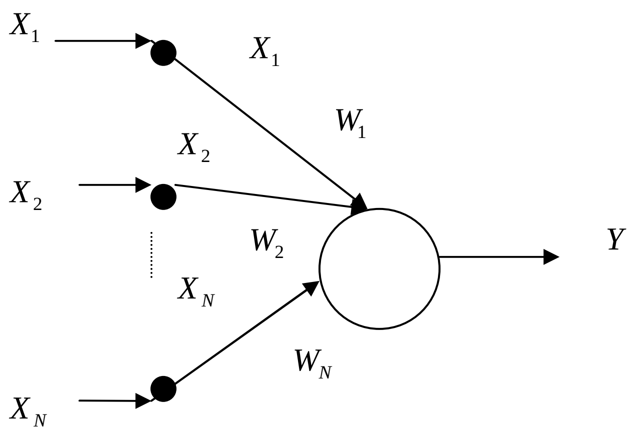
El aprendizaje por corrección de error consiste en ajustar los pesos de las conexiones de la red en función de la diferencia de los valores deseados y los obtenidos en la salida de la red; es decir, en función del error cometido en la salida

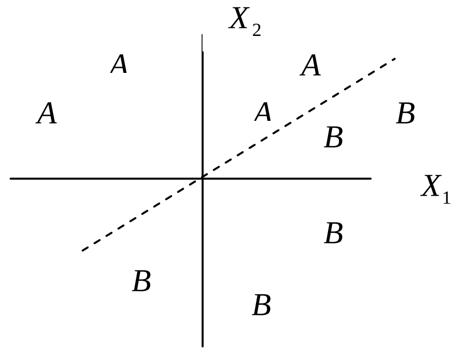
1.3.3.2 Redes con aprendizaje no supervisado

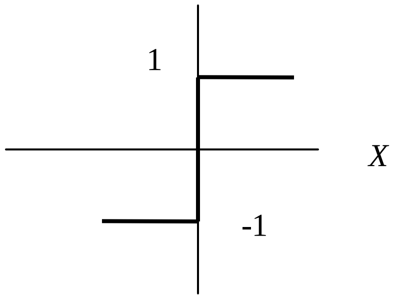
Las redes con aprendizaje no supervisado no requieren influencia externa para ajustar los pesos de las conexiones entre sus neuronas. La red no recibe información del entorno que le indique si la salida generada en respuesta a una determinada entrada es o no correcta, estas redes son capaces de auto organizarse. Existe dos tipos de aprendizaje no supervisado: aprendizaje hebbiano, aprendizaje competitivo y cooperativo.

1.3.4 RED NEURONAL DE PERCEPTRON MULTICAPA

Éste permite reconocer patrones sencillos; así, un Perceptron, formado por varias neuronas lineales para recibir las entradas a la red y una neurona de salida, es capaz de decidir cuando una entrada presentada a la red pertenece a una de las dos clases que es capaz de reconocer (Figura 1.11). [6]







El perceptron puede clasificar un vector analógico de entrada en dos clases A y B

$$X_2 = \frac{-W_1}{W_2} X_1 + \frac{\theta}{W_2}$$

$$Y = f\left[\sum_{i=1}^{N} W_i X_i - \theta\right]$$

Figura 1.11. El Perceptron

1.3.4.1 Red backpropagation

Esta red consiste en su capacidad de auto adaptar los pesos de las neuronas de las capas intermedias para aprender la relación que existe entre un conjunto de patrones dados y sus salidas correspondientes.

La regla delta ha sido extendida a redes con capas intermedias (regla delta generalizada) con conexión hacia delante, y cuyas células tienen funciones de activación continuas, dando lugar al algoritmo de retropropagación. Estas funciones continuas son derivables y no decrecientes.

El método que sigue la regla delta generalizada para ajustar los pesos es exactamente el mismo que el de la regla delta utilizada en el Perceptron y

Adaline; es decir, los pesos se actualizan en forma proporcional a la delta (δ=salida deseada – salida obtenida).

Dado una neurona (U_i) y la salida que produce (y_i) (Figura 1.12), el cambio que se produce en el peso de la conexión que une la salida de dicha neurona con la unidad U_j para un patrón de aprendizaje P determinado es [6]:

$$\Delta w_{ij}(t+1) = \alpha.\delta_{pj}\, y_{pi} \qquad (1.47)$$

En donde el subíndice p se refiere al patrón de aprendizaje y α es la constante o taza de aprendizaje.

En una red Backpropagation existe una capa de entrada con n neuronas y una capa de salida con m neuronas y al menos una capa oculta de neuronas internas.

Cada neurona de una capa (excepto las de entrada) recibe entradas de todas las neuronas de la capa anterior y envía su salida a todas las neuronas de la capa posterior (excepto la salida). No hay conexiones hacia atrás (feedback) ni laterales entre neuronas de la misma capa Figura 1.12.

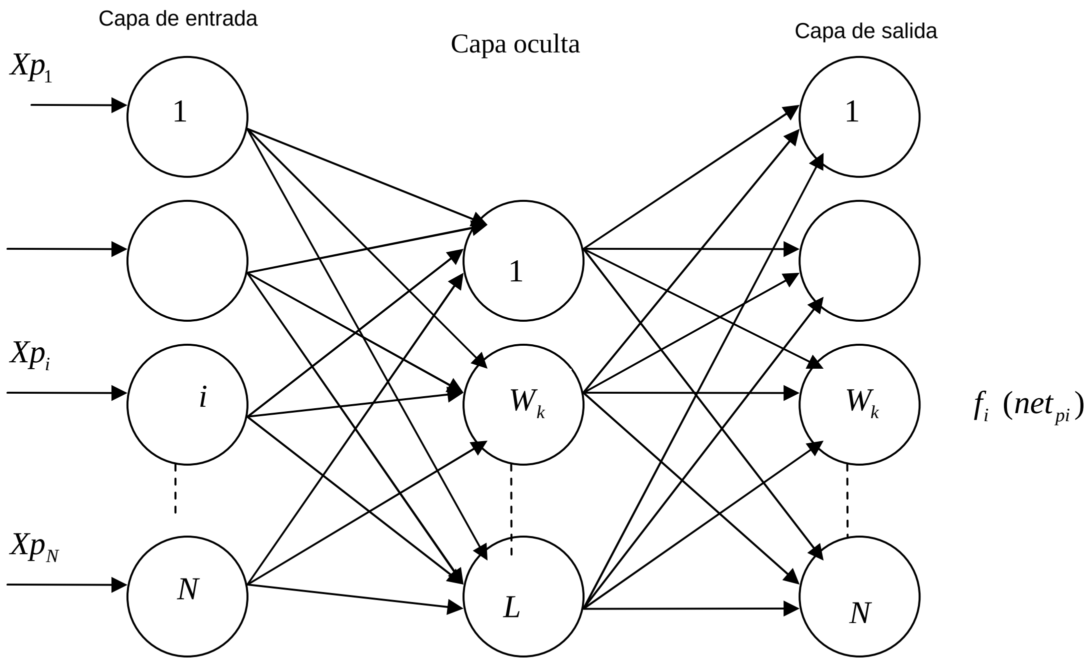


Figura 1.12. Modelo de arquitectura de una red Backpropagation

A diferencia de la regla delta en el caso del Perceptron, la técnica Backpropagation o generalización de la regla delta, requiere el uso de neuronas cuya función de activación sea continua, y por tanto diferenciable. Generalmente la función utilizada será de tipo sigmoidal.

A continuación se presenta a modo de síntesis, los pasos y formulas a utilizar para aplicar el algoritmo de entrenamiento [6]:

<u>Paso 1</u>.- Inicializar los pesos de la red con valores pequeños aleatorios.

<u>Paso 2</u>.- Presentar un patrón de entrada x_p y especificar la salida deseada que debe generar la red $d_1, d_2, ..., d_M$.

<u>Paso 3</u>.- Calcular las entradas netas para las neuronas ocultas procedentes de las neuronas de entrada:

$$net_{pj}^{h} = \sum_{i=1}^{N} w_{ji}^{h} x_{pi} + \theta_{j}^{h} \qquad (1.48)$$

El subíndice p, se refiere al p-ésimo vector de entrenamiento y j la enésima neurona oculta. El término θ puede ser opcional, pues actúa como una entrada más.

Se calcula las salidas de las neuronas ocultas:

$$y_{pi} = f_{j}^{h}(net_{pj}^{h}) \qquad (1.49)$$

Se realizan los mismos cálculos para obtener las salidas de las neuronas de salida:

$$\begin{aligned} net_{pk}^{0} &= \sum_{j=1}^{L} w_{kj}^{o} y_{pj} + \theta_{k}^{o} \\ y_{pk} &= f_{k}^{o}(net_{pk}^{o}) \end{aligned} \qquad (1.50)$$

<u>Paso 4</u>.- Calcular los términos de error para todas las neuronas:

$$\delta_{pj}^{h} = x_{pi}(1 - x_{pi})\sum_{k}\delta_{pk}^{o}\; w_{kj}^{o} \qquad (1.51)$$

Paso 5.- Actualizar los pesos. Para ello se utiliza el algoritmo recursivo, comenzando por las neuronas de salida y trabajando hacia atrás hasta llegar a la capa de entrada ajustando los pesos. Para los pesos de la neurona de la capa de salida se tiene:

$$\Delta w_{kj}^{o}(t+1) = \alpha\,\delta_{pk}^{o}\; y_{pj} \qquad (1.52)$$

Y para los pesos de las neuronas de la capa oculta:

$$\Delta w_{ji}^{h}(t+1) = \alpha\,\delta_{pj}^{o}\; x_{pj} \qquad (1.53)$$

Paso 6.- Repetir el proceso hasta que el término de error resulta aceptablemente pequeño para cada uno de los patrones aprendidos, en base a la siguiente fórmula:

$$E_{p} = \frac{1}{2}\sum_{k=1}^{M}\delta_{pk}^{2} \qquad (1.54)$$

1.3.4.1.1 Consideraciones sobre el Algoritmo de Aprendizaje

Uno de los problemas que presenta este algoritmo de entrenamiento de redes multicapa es que busca minimizar la función de error, pudiendo caer en un mínimo local o en un punto estacionario (Figura 1.13), con lo cual no se llega a encontrar el mínimo global de la función de error; sin embargo, puede ser suficiente encontrar un error mínimo preestablecido. [6]

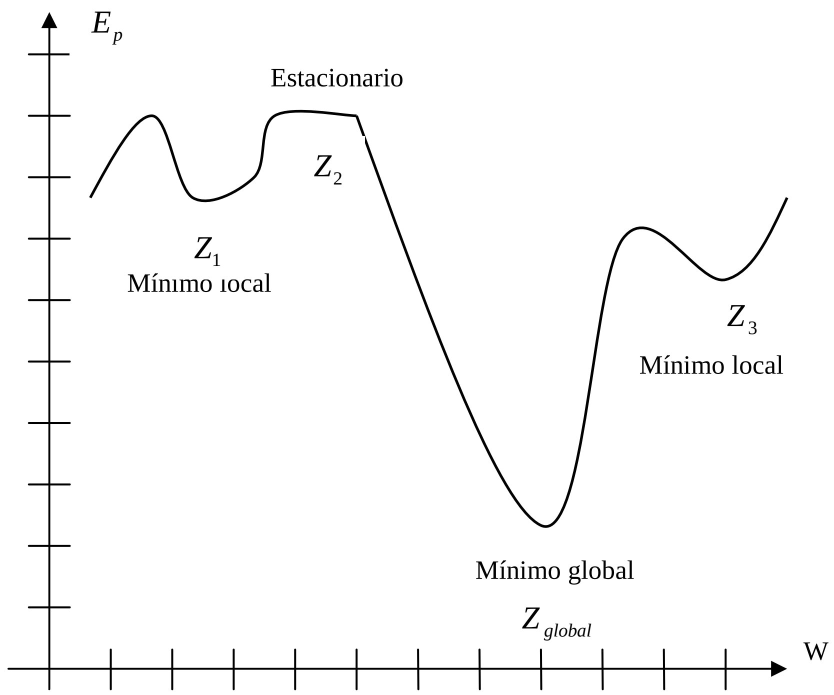


Figura 1.13. Ejemplo representativo de una forma de la superficie de error, donde W representa los posibles valores de matriz de pesos de la red

El elegir un incremento de paso adecuado influye en la velocidad con la que converge el algoritmo. Esta velocidad se controla a través de la constante de proporcionalidad o taza de aprendizaje (α). Si esta constante es muy grande, los cambios de peso son muy grandes, con el riesgo de saltar el mínimo y estar oscilando alrededor de él, pero sin poderlo alcanzar.

Respecto al número de capas de la red, en general tres capas son suficientes (entrada-oculta-salida). Sin embargo, hay veces que un problema es más difícil de resolver con más de una capa oculta. El tamaño de las capas, tanto de entrada como de salida, suele venir determinado por la naturaleza de la aplicación.

El número de neuronas ocultas interviene en la eficacia de aprendizaje y de generalización de la red y no hay ninguna regla que indique el número óptimo, simplemente se trata de tener el menor número posible de neuronas en la capa oculta, por que cada una de ellas supone una mayor carga de procesamiento en el caso de una simulación en software. Para la inicialización de los pesos se parte de un punto cualquiera del espacio, inicializando los pesos con valores pequeños aleatorios cualesquiera (por ejemplo 0.5).

1.4 ESTADO DEL ARTE SOBRE SISTEMAS DISTRIBUIDOS.

La computación desde sus inicios ha sufrido muchos cambios, desde los grandes computadores que permitían realizar tareas en forma limitada y de uso un tanto exclusivo de organizaciones muy selectas, hasta los actuales computadores ya sean personales o portátiles, que tienen las mismas e incluso mayores capacidades que los primeros y que están cada vez más introducidos en el quehacer cotidiano de una persona.

El desarrollo de las redes de área local y de las comunicaciones que permitieron conectar computadores con posibilidad de transferencia de datos a alta velocidad. Es en este contexto que aparece el concepto de "Sistemas Distribuidos". Sistemas cuyos componentes hardware y software, que están en computadores conectados en red, se comunican y coordinan sus acciones mediante el paso de mensajes, para el logro de un objetivo.

1.4.1 CARACTERÍSTICAS DE LOS SISTEMAS DISTRIBUIDOS

Concurrencia: esta característica de los sistemas distribuidos permite que los recursos disponibles en la red puedan ser utilizados simultáneamente por los usuarios y/o agentes que interactúan en la red.

Carencia de reloj global: las coordinaciones para la transferencia de mensajes entre los diferentes componentes para la realización de una tarea, no tienen una temporización general, esta más bien distribuida a los componentes.

Fallos independientes de los componentes: cada componente del sistema puede fallar independientemente, con lo cual los demás pueden continuar ejecutando sus acciones. Esto permite el logro de las tareas con mayor efectividad, pues el sistema en su conjunto continua trabajando.

1.4.2 EVOLUCIÓN DE SISTEMA DISTRIBUIDO

El procesamiento central, es uno de los primeros modelos de computadores interconectados, llamados centralizados, donde todo el procesamiento de la organización se llevaba a cabo en una sola computadora, normalmente un Mainframe, y los usuarios empleaban sencillos computadores personales. Los problemas de este modelo se presentan cuando la carga de procesamiento aumentaba: se tenía que cambiar el hardware del Mainframe, lo cual es más costoso que añadir más computadores personales clientes o servidores que aumenten las capacidades.

El otro problema que surgió son las modernas interfases gráficas de usuario, las cuales podían conllevar a un gran aumento de tráfico en los medios de comunicación y, por consiguiente, podían colapsar.

Otro modelo que entró a competir con el anterior es el grupo de servidores, también un tanto centralizado, son un grupo de computadores actuando como servidores, normalmente de archivos o de impresión, para un número de minicomputadores que hacen el procesamiento conectados a una red de área local. Los problemas de este modelo podrían generar una saturación de los medios de comunicación entre los servidores y los minicomputadores, por ejemplo cuando se solicitan archivos grandes por varios clientes a la vez, podían disminuir en gran medida la velocidad de transmisión de información.

La Computación Cliente Servidor, este modelo, que predomina en la actualidad, permite descentralizar el procesamiento y recursos, sobre todo, de cada uno de los servicios y de la visualización de la interfaz gráfica de usuario. Esto hace que ciertos servidores estén dedicados solo a una aplicación determinada y por lo tanto ejecutarla en forma eficiente.

1.4.3 CLIENTE-SERVIDOR

Sistema donde el cliente es una máquina que solicita un determinado servicio un servidor que es la máquina que lo proporciona. Los servicios pueden ser:

ejecución de un determinado programa, acceso a un determinado banco de información, acceso a un dispositivo de hardware.

Es elemento primordial la presencia de un medio físico de comunicación entre las máquinas, y dependerá de la naturaleza de este medio la viabilidad del sistema.

1.4.3.1 Clasificación de los sistemas cliente servidor.

A continuación se muestra la clasificación de los sistemas cliente/servidor de acuerdo al nivel de abstracción del servicio que ofrecen:

Representación distribuida: la interacción con el usuario se realiza en el servidor, el cliente hace de pasarela entre el usuario y el servidor (Figura 1.14).

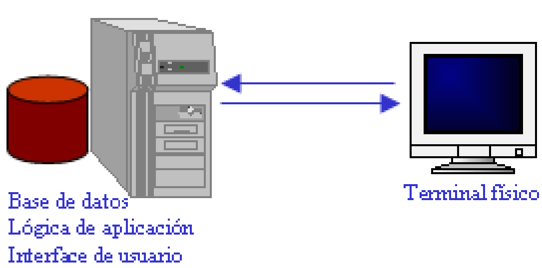


Figura 1.14. Representación distribuida

Representación Remota: la lógica de la aplicación y la base de datos se encuentran en el servidor. El cliente recibe y formatea los datos para interactuar con el usuario como se muestra en la Figura 1.15.

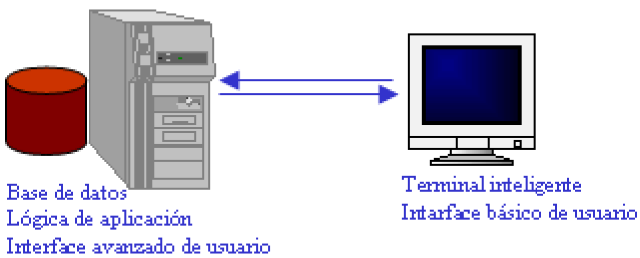


Figura 1.15. Representación Remota

Lógica Distribuida: el cliente se encarga de la interacción con el usuario y de algunas funciones triviales de la aplicación. Por ejemplo controles de rango de campos, campos obligatorios, etc. Mientras que el resto de la aplicación, junto con la base de datos, están en el servidor (Figura 1.16).

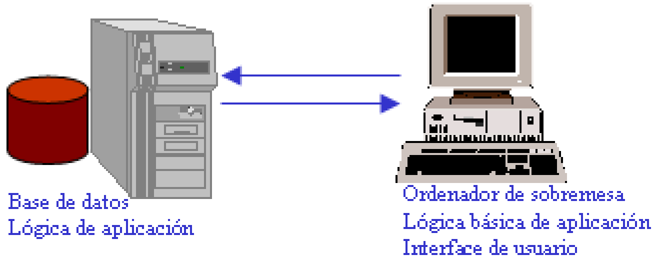


Figura 1.16. Lógica Distribuida

Gestión Remota de Datos: el cliente realiza la interacción con el usuario y ejecuta la aplicación y el servidor es quien maneja los datos (Figura 1.17).

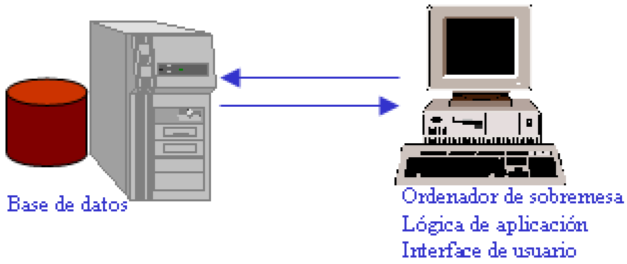


Figura 1.17. Gestión Remota de Datos

Base de Datos Distribuidas: el cliente realiza la interacción con el usuario, ejecuta la aplicación, debe conocer la topología de la red, así como la disposición y ubicación de los datos. Se delega parte de la gestión de la base de datos al cliente (Figura 1.18).

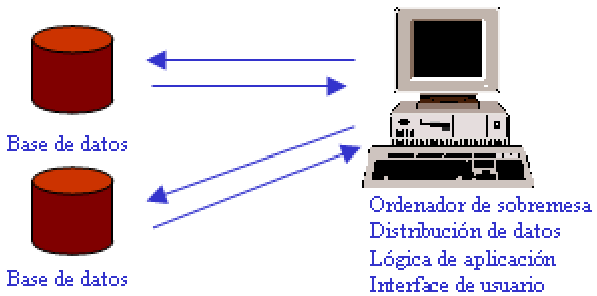


Figura 1.18. Base de Datos Distribuidas

Cliente servidor a tres niveles: el cliente se encarga de la interacción con el usuario, el servidor de la lógica de aplicación y la base de datos puede estar en otro servidor (Figura 1.19).

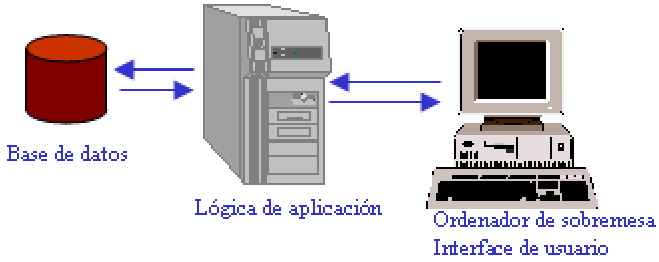


Figura 1.19. Cliente servidor a tres niveles

1.4.4 VENTAJAS DE LOS SISTEMAS DISTRIBUIDOS

Con respecto a sistemas centralizados, una de las ventajas de los sistemas distribuidos es la economía, pues es mucho más barato añadir servidores y clientes cuando se requiere aumentar la potencia de procesamiento. El trabajo en conjunto; por ejemplo: en una fábrica de ensamblado los robots tienen sus CPUs diferentes y realizan acciones en conjunto, dirigidos por un sistema distribuido. Tienen una mayor confiabilidad, al estar distribuida la carga de trabajo en muchas máquinas, la falla de una de ellas no afecta a las demás, el sistema sobrevive como un todo. Además poseen capacidad de crecimiento incremental, es decir, se puede añadir procesadores al sistema incrementando su potencia en forma gradual según sus necesidades.

Con respecto a PCs independientes, se pueden compartir recursos, como programas y periféricos, ejemplo: impresoras, dispositivos de almacenamiento masivo, etc. Al compartir recursos, satisfacen las necesidades de muchos usuarios a la vez, ejemplo: sistemas de reservas de aerolíneas. Se logra una mejor comunicación entre las personas, ejemplo; el correo electrónico. Tienen mayor flexibilidad, la carga de trabajo se puede distribuir entre diferentes ordenadores.

1.5 ÁREAS DE APLICACIÓN DE TECNOLOGÍAS EN EL PAÍS

Actualmente existen muchos tipos de robots industriales, los cuales han permitido al hombre facilitar sus labores e incrementar su productividad; esto es, al realizar trabajos repetitivos que requieran precisión o realizar labores que ponen en peligro la vida o las dos cosas al mismo tiempo. Entre estas áreas se tiene la telecirugía, es una especialidad que se origina a partir de la robótica, en la cual se combinan la telecomunicación y los robots para realizar cirugías a distancia.

El Hospital de los Valles y el Metropolitano trabaja en proyectos, denominados quirófano avanzado, que permitirá exponer casos a hospitales y médicos de otros países, obtener segundas opiniones y ser parte de teleconferencias en diferentes temas.

En Ecuador se realizó la primera experiencia en cuanto a la robótica en noviembre del 2005. El doctor Manolo Cortez realizó por primera vez una cirugía con la asistencia de un brazo robótico. La intervención vesicular se concretó en el Hospital Metropolitano, pero la experiencia no se ha repetido. Esto debido a la falta de recursos económicos. En esa ocasión el brazo fue prestado por una institución privada, el precio del brazo robótico es de aproximadamente $250.000.

En los casos de cáncer sucede lo mismo. La radiocirugía robótica consigue extenderse con éxito al tratamiento de tumores extracraneales. Este método ya se está utilizando en el tratamiento de tumores de columna, pulmón, páncreas y próstata, especialmente en los países desarrollados. El sistema conocido como CyberKnife monta un acelerador lineal en un brazo robótico capaz de llevar sus movimientos con la respiración del paciente, solo hay 60 aparatos en todo el mundo.

La radiocirugía esterotáxica Gammaknife, que hace unas décadas supuso la revolución en el tratamiento no invasivo de tumores intracraneales, ha evolucionado hasta lo que hoy se conoce como radiocirugía robótica. El único exponente es el sistema CyberKnife, los pacientes tratados en todo el mundo con GammaKnife animaron a los investigadores a intentar superar el reto de extender los beneficios de la radiocirugía esterotáxica a otros órganos.

Se espera que, un día, sus tentáculos delgados puedan ser usados para controlar operaciones delicadas en los rincones del cerebro humano. Actualmente en el país esta tecnología no esta en apogeo, sin embargo en un futuro no muy lejano esta podrá ser implementada en diferentes sectores industriales tales como: automotriz, química, alimentos u otras industrias.

1.5.1 INDUSTRIA AUTOMOTRIZ

El sector automovilístico es el primer consumidor de robots y de sistemas de automatización. La automatización en este sector está orientada a maximizar la productividad, la calidad y la seguridad. La carga de herramientas pesadas y la permanencia en entornos peligrosos han sido sustituidas por trabajos de supervisión y mantenimiento. De hecho, es uno de los sectores con menor grado de siniestralidad.

Uno de los sectores en los que más se han volcado tradicionalmente los fabricantes de robots y automatismos ha sido en el proceso de fabricación de automóviles. Alrededor del 25% de todos los robots instalados en el sector de automoción se dedica a la soldadura de carrocerías y diversas piezas, bien sea por puntos o por arco.

Otra aplicación importante es el ensamblado de subconjuntos, tales como motores, lunas, depósitos, ruedas, embellecedores externos, etc. Se estima que al menos el 20% de todas las aplicaciones de ensamblado está robotizado en la industria del automóvil.

La pintura de las carrocerías está totalmente robotizada en la mayoría de las fábricas a nivel mundial Figura 1.20, esta aplicación es muy similar a otras: sellado de juntas y uniones, aplicación del pegamento para las lunas, corte de piezas de plástico y metálicas, bien sea por láser o por chorro de agua.

Figura 1.20. Brazo Robótico de pintura para automoción

Por otro lado, las aplicaciones de transporte y manipulado también están robotizadas casi en su totalidad. Las aplicaciones más típicas son paletización de producto en almacenes y almacenes intermedios, y la alimentación de máquinas (prensas de chapa, máquinas-herramientas, autoclaves, etc.). La actual tendencia en producción es sustituir las máquinas especializadas por sistemas de propósito general, siendo en algunos casos robots industriales.

1.5.2 INDUSTRIA QUÍMICA

La industria química es la parte más representativa de la industria de control de procesos, cuyas variables físicas son casi todas continuas. Los sectores que abarca, van desde el petroquímico hasta el cementero, pasando por la agroquímica, polímeros, farmaquímica y desalinización. No obstante, sea cual sea el sector, el proceso básicamente consiste en la manipulación de materias primas, la reacción química propiamente dicha, la separación primaria de los productos, la separación posterior de productos líquidos o sólidos y la purificación del producto final.

Debido al número elevado de variables físicas que hay que controlar y supervisar en un proceso químico, los sistemas de control son de arquitectura distribuida. Esto implica una arquitectura de hardware en la que cada computador se encargue de adquirir, a través de sensores, la información necesaria para cerrar el bucle de control local y de comunicarse con otros computadores.

1.5.3 INDUSTRIA DE ALIMENTOS

La industria de alimentos emplea una cantidad importante de mano de obra en operaciones bastante repetitivas. La introducción de sistemas automatizados con un alto grado de flexibilidad y una continua adaptación a la demanda (que actualmente se centra en productos frescos) son requisitos básicos para la actual industria de alimentación.

En la industria cárnica (Figura 1.21), los sistemas automáticos de despiece más avanzados utilizan tecnología de visión $3D$ con iluminación estructurada. El sector de lácteos y bebidas cuenta con un alto nivel de automatización mediante sistemas rígidos. Su productividad es muy alta, pero en los últimos años crece la necesidad de un mayor nivel de flexibilidad. Se desea utilizar al máximo las instalaciones, sin efectuar paradas innecesarias y producir más productos distintos en la misma factoría.

Figura 1.21. Robot de despiece cárnico

1.5.4 OTRAS INDUSTRIAS

En la **industria cerámica**, las principales aplicaciones de la automatización de la producción se centran en la automatización de máquinas y procesos. Una de las áreas de automatización prioritarias es el transporte y almacenamiento de piezas delicadas, para lo que se necesitan equipos de paletización termoventilados basados en robots de pórtico con ruedas, que permiten el llenado de las cajas, el etiquetado, la aplicación de flejes y el paletizado final. Para el control de calidad de la cerámica se emplea la visión artificial.

En la **industria textil**, la fabricación cuenta ya con un alto nivel de automatización pero la utilización de robots en esta industria está muy limitada, centrándose casi exclusivamente en aplicaciones de paletizado y manipulado. Una de las aplicaciones robóticas más novedosas es el manipulado de telas para su posterior cosido automático. La principal dificultad está en desarrollar pinzas capaces de manipular telas. Existen prototipos de pinzas que generan un chorro de aire en una dirección determinada, levantando la tela para posteriormente cogerla; este problema de manipular y colocar la tela en posición correcta es sumamente complejo. La automatización de la **industria de plásticos** pasa por mejorar sus numerosos equipos. Las modernas máquinas de fabricación de piezas plásticas se pueden dividir en: fabricación de moldes, granuladoras, secadoras, dosificadores, máquinas de inyección de gran velocidad, moldeadoras de grandes piezas, manipuladores para carga y descarga de máquinas, etc.

1.6 OBJETIVOS DEL PRESENTE TRABAJO

El objetivo global es:

1.- Desarrollar algoritmos de control basados en inteligencia y visión artificial, con la finalidad de controlar un manipulador que pueda tomar una pieza entre un conjunto de ellas y lo pase a otro manipulador.

Los objetivos específicos son:

2.- Realizar la adquisición y procesamiento de imágenes en el programa computacional Matlab (The Mathworks) con la finalidad de identificar y posicionar los objetos en el plano real.

3.- Analizar e implementar una red neuronal de perceptrón multicapa el mismo que será entrenado con el algoritmo de retropropagación, para la clasificación de objetos.

4.- Simular el algoritmo de posicionamiento de los dos brazos robóticos en el programa computacional Matlab (The Mathworks).

5.- Finalmente se buscara áreas de aplicación para esta tecnología en nuestro país.

CAPÍTULO 2

MATERIALES Y MÉTODOS

CAPÍTULO 2

MATERIALES Y MÉTODOS

2.1 INTRODUCCIÓN

En el presente capítulo se describen los algoritmos desarrollados de procesamiento de imágenes y la simulación del sistema distribuido. Se partirá describiendo el sistema distribuido robotizado y se detallará las interfaces gráficas creadas, posteriormente se explicarán los algoritmos desarrollados y finalmente se expondrá el diseño de la tarjeta de control para operar un manipulador de tres grados de libertad.

Los algoritmos fueron implementados en el programa computacional Matlab, a excepción del programa del microcontrolador de la tarjeta de control que fue realizado en el programa MPLAB.

2.2 SISTEMA DISTRIBUIDO ROBOTIZADO

En la Figura 2.1 se muestra un diagrama de bloques del sistema distribuido, mismo que está conformado por dos puestos de trabajo, que de aquí en adelante se los denominará Host local y Host remoto los mismos que se encuentran formando una red LAN.

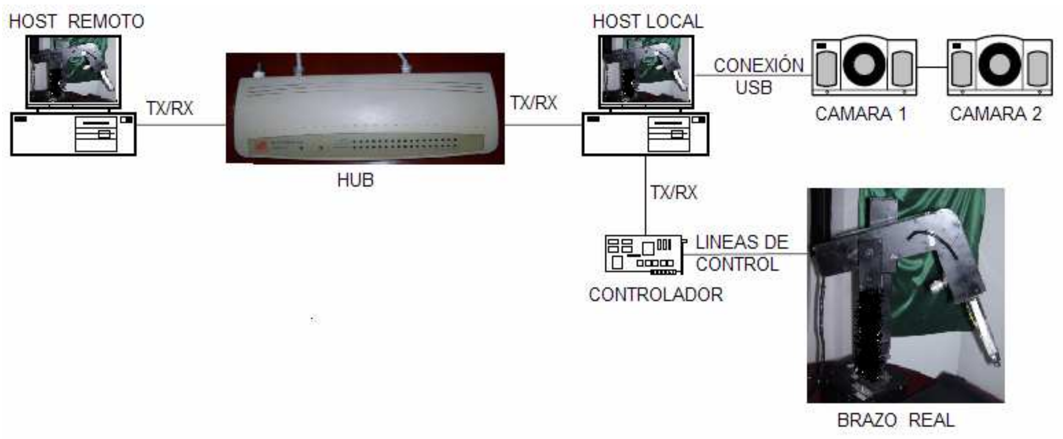


Figura 2.1. Sistema distribuido implementado

En el Host local se realiza la adquisición de las imágenes mediante dos cámaras web, también contiene el algoritmo de procesamiento de imágenes, clasificación y posicionamiento de los objetos, visualiza la simulación del sistema robotizado y por último, tiene conectado un controlador permitiendo operar un manipulador real de tres grados de libertad.

El Host remoto permite visualizar la simulación del sistema distribuido y recibe datos desde el Host local, los datos recibidos determinan la nueva ubicación del objeto dentro de la simulación. La conexión de los dos host para la transferencia de datos se lo realiza mediante un HUB.

2.2.1 SIMULACIÓN DEL SISTEMA DISTRIBUIDO

Para la simulación del sistema distribuido robotizado, se empleo el toolbox de Realidad Virtual de Matlab. Esta herramienta permite al observador movilizarse a través del entorno virtual y observar a detalle y desde cualquier posición el proceso que se encuentra simulando.

Virtual Reality Tollbox permite conectar un mundo virtual con un modelo realizado en Matlab. Esta herramienta está basada en el lenguaje VRML (Virtual Reality Modeling Language), para construir objetos y mundos virtuales interactuando en 3D. Cada archivo VRML, implícitamente establece un sistema de coordenadas para todos los objetos definidos e incluidos en el archivo.

Un archivo VRML puede ser creado y editado mediante programación estructurada (C, C++, archivos.m). Hoy en día se encuentran programas computacionales en entornos gráficos con mayor potencial para la creación de interfaces virtuales, como el V-Real Builder 2.0 que es una poderosa herramienta para la creación de ambientes en tres dimensiones, con la ventaja que se pueden diseñar los objetos virtuales en forma gráfica. Este programa se encuentra instalado junto a Matlab para versiones superiores a 6.0.

Para enlazar el Matlab con el entorno virtual creado en V-Realm Builder 2.0, se puede hacerlo mediante simulink o un archivo de texto (*.m). En el presente trabajo se utiliza el archivo editor de Matlab (*.m), debido a que todos los algoritmos desarrollados se encuentran en este tipo de archivo.

2.2.1.1 Diseño de un objeto virtual.

Para diseñar el entorno virtual se debe abrir el programa V-Real Builder 2.0, que se encuentra en "MATLAB-raiz\toolbox\vr\vrealm\program\vrbuik2.exe", desplegando la siguiente pantalla (Figura 2.2):

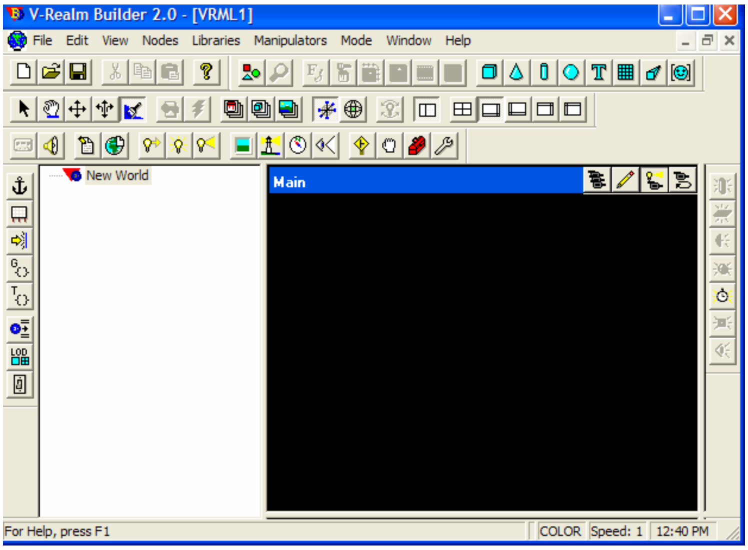


Figura 2.2. Pantalla de inicio de V-Real-Builder 2.0

Una vez abierto el programa, en la barra de herramientas superior se da un click en "Insert Extrusion" para insertan las figuras (cilindros, cubos, y formas irregulares) que conformaran las articulaciones de los manipuladores. Para fines didácticos se presentará la construcción de la pinza del brazo, entendiéndose que se procederá de igual forma para el resto de partes del manipulador, como se muestra en la Figura 2.3.

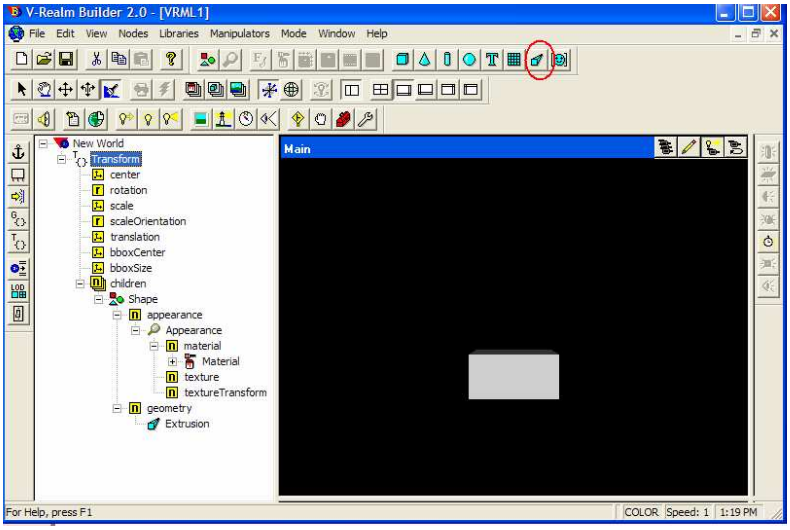


Figura 2.3. Insertar un objeto

Para cambiar las características del objeto, se trabaja en el árbol de nodos (Figura 2.4).

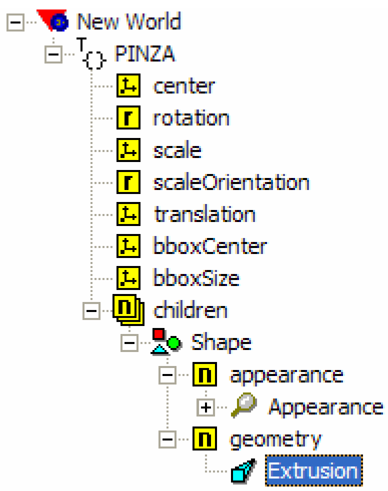


Figura 2.4. Árbol de nodos

Dando doble click en la opción "Extrusión", se despliega un editor de características del objeto como se muestra en la Figura 2.5 en el cual se puede cambiar la forma y tamaño del objeto.

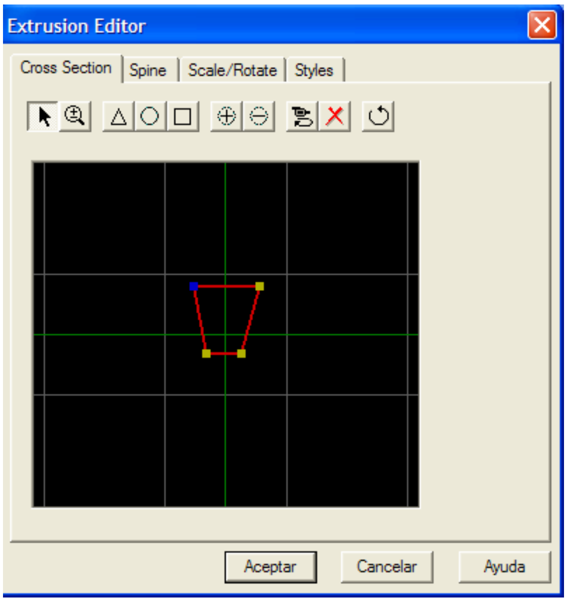


Figura 2.5. Editor del objeto

El Cross Section determina la forma del área transversal de la figura. El Spine establece la forma de la cara superior. El Scale/Rotate fija las escalas de los ejes X y Y. El style configura la textura y caras del objeto, ver Figura 2.6.

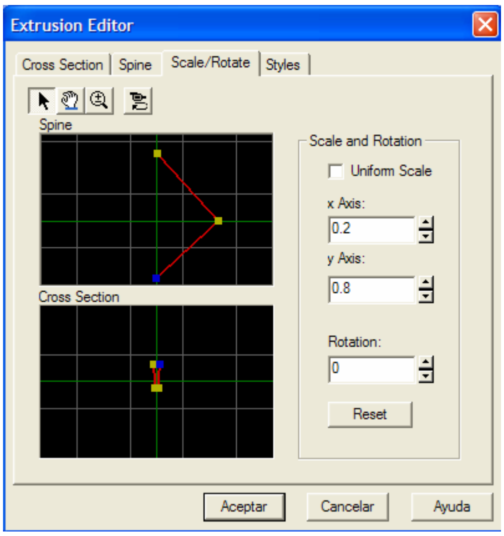


Figura 2.6. Editor de Escala y Rotación

Una vez configurados estos parámetros se obtiene la parte derecha de la pinza del manipulador, como se muestra en la Figura 2.7.

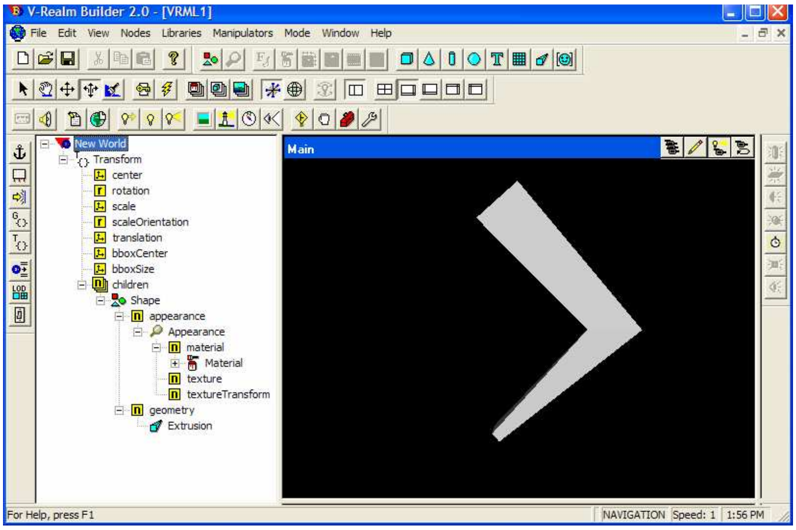


Figura 2.7. Parte derecha de la pinza del robot

En el árbol de nodos se puede observar todos los componentes del mundo virtual del proyecto, así como también sus respectivas propiedades. Entre las propiedades mas útiles y comunes para todos los objetos se tiene:

Center.- Representa la posición (x, y, z) de un punto del objeto, sobre el cual se establece la traslación y rotación del mismo.

Rotation.- Establece la rotación absoluta del cuerpo con respecto al sistema de coordenadas globales.

Scale.- Constituye el factor de escalado del objeto en cada uno de los ejes.

Traslation.- Establece la traslación del cuerpo con respecto al sistema de coordenadas globales.

Children.- Este nodo contiene toda la información de la apariencia del objeto.

Appearance.- Este nodo contiene las características de color y textura del objeto.

Una vez que se insertan todos los objetos dentro del mundo virtual es necesario unirlos para formar una sola pieza (Por ejemplo: base, antebrazo, brazo, etc), para lo cual se ayuda de las cuatro vistas que contiene el V-Realm Builder 2.0: vista principal, superior, derecha y frontal (Figura 2.8).

Una vez construido el entorno virtual, se guarda el documento como un archivo "*.wrl", tanto en el Host local (Host_local.wrl) y en el Host remoto (Host_remoto.wrl).

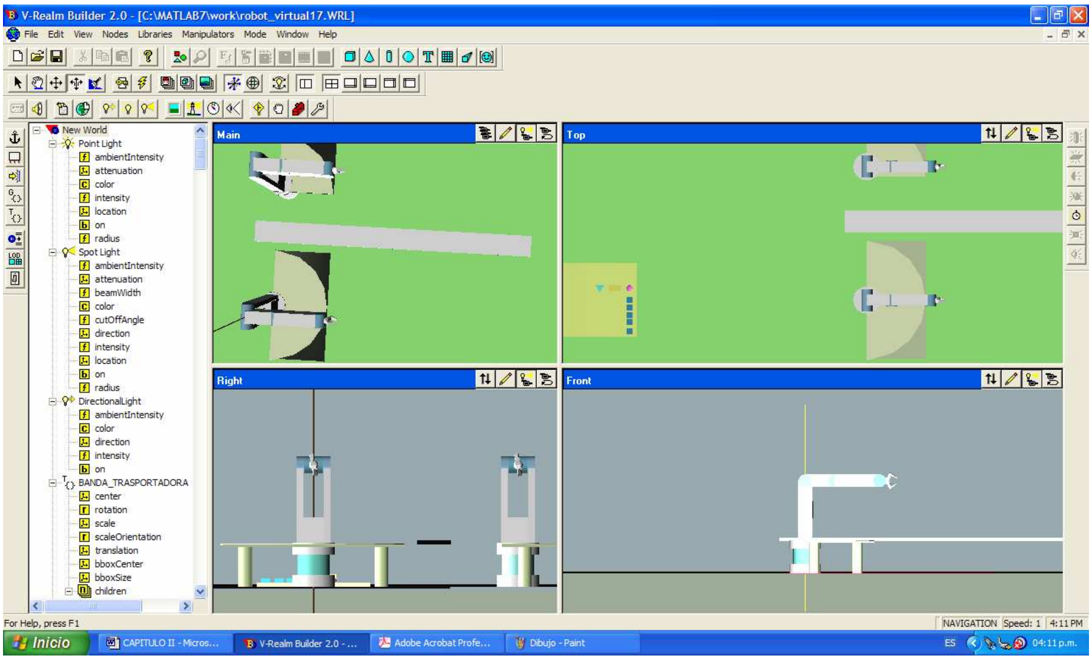


Figura 2.8. Vistas de trabajo del V-Real-Builder 2.0

2.2.1.2 Enlace entre el mundo virtual y un archivo "*.m"

Para poder realizar un enlace entre el mundo virtual y un archivo "*.m", se debe seguir los siguientes pasos:

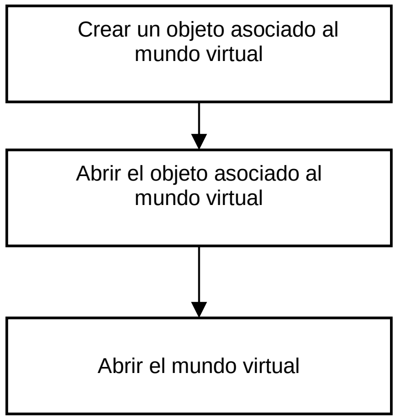


Figura 2.9. Pasos para enlazar el mundo virtual

Para crear un objeto asociado al mundo virtual se debe conocer el nombre exacto del objeto que se encuentra dentro del V-Real-Builder 2.0. En la Figura 2.4, se puede observar que el nombre del objeto es "PINZA", este se utiliza para enlazar el objeto al archivo "*.m". Para enlazar el objeto "PINZA" al archivo "*.m" se emplea la instrucción ***vrworl***.

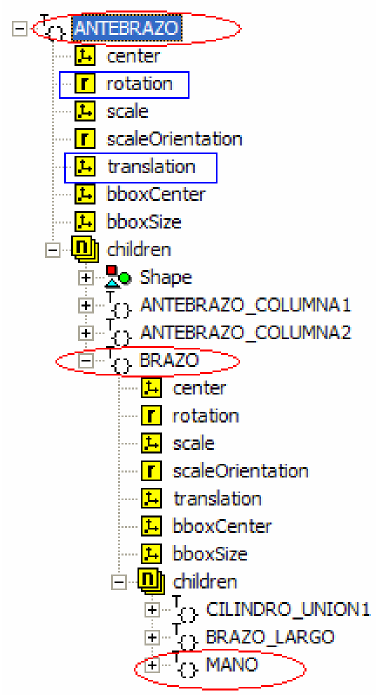


Figura 2.10. Nombre de los objetos creados

En el presente caso los nombres de los objetos creados en el V-Real-Builder 2.0, son: antebrazo (articulación 1), brazo (articulación 2), mano (articulación 3) y pinza, como se muestra en la Figura 2.10

Una vez asociado el objeto al archivo "*.m", estos se trabajan como cualquier variable dentro de este archivo. Con la diferencia que necesitan las instrucciones de **"translation**" y "**rotation**" que son propias del V-Real-Builder 2.0 (Figura 2.10). Estas instrucciones se trabajan como cualquier instrucción de Matlab, tomando en cuenta que los nombres de los objetos enlazados sean asignados como variables ***globales***.

Una vez configurado los objetos como variables dentro del archivo "*.m" se puede abrir el entorno virtual (Figura 2.11) empleando el comando "***view***".

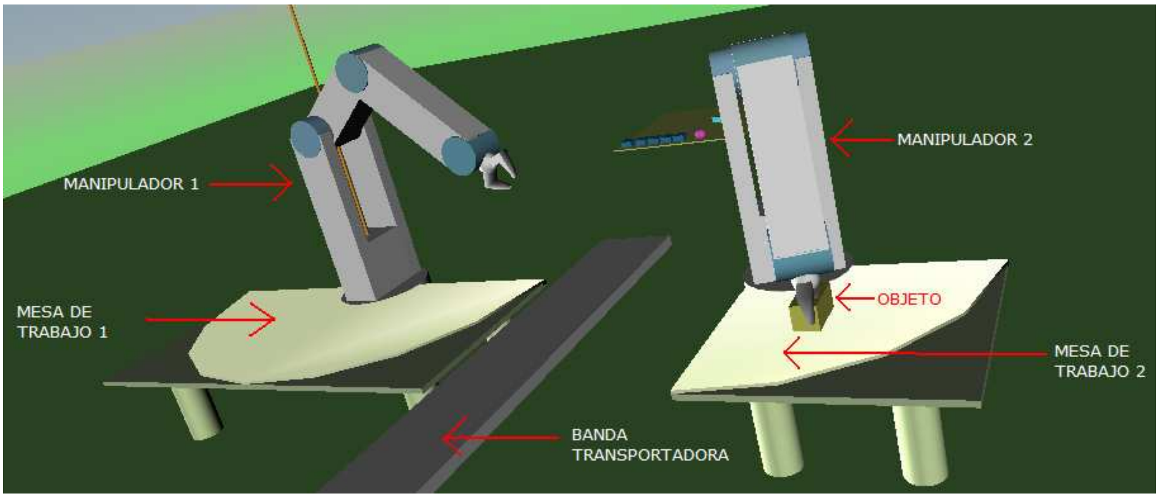


Figura 2.11. Entorno Virtual del Sistema Distribuido

2.3 DESCRIPCIÓN DEL HOST LOCAL

Como se mencionó, el Host local se encarga de la adquisición y procesamiento de la imagen, clasificación y posicionamiento de los objetos, y de la simulación del sistema robotizado. En la Figura 2.12 se muestran los pasos a seguir para la operación del Host.

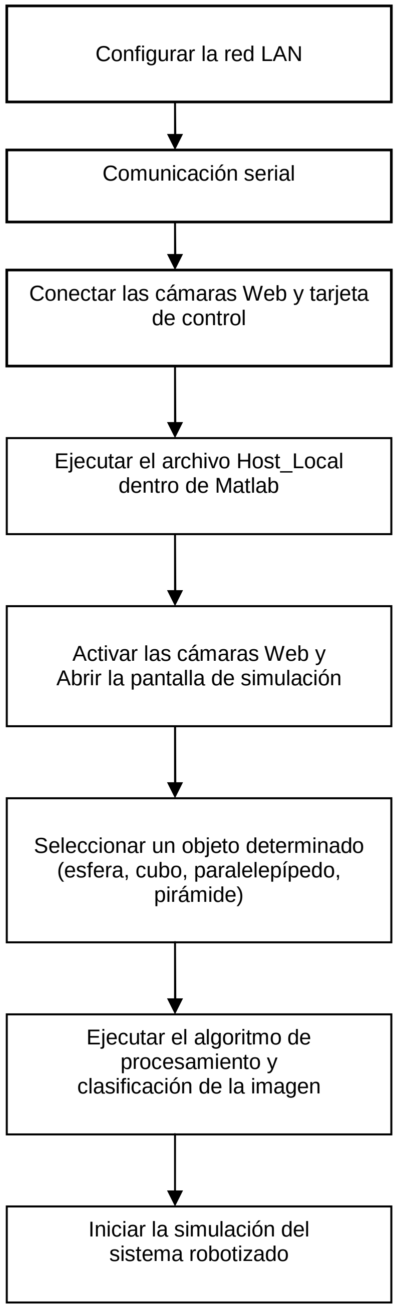


Figura 2.12. Pasos para trabajar con el Host local

2.3.1 CONFIGURACIÓN DE LA RED LAN

En la Figura 2.13 se presenta el diagrama de bloques de los pasos a seguir para la configuración de la red LAN. En el Host local y en el Host remoto se crea un objeto UDP (user datagram protocol) en el cual se define un puerto de enlace que debe estar contenido entre 1 y 65535, el valor por defecto para UDP en Matlab es de 9090, el mismo que se lo seteo en 4114 para comprobar que se puede asignar cualquier valor comprendido en el rango mencionado. Luego, se establece una dirección IP de clase C, en nuestro caso se seleccionó 192.168.77.183 para el Host local y 192.168.77.182 para el Host remoto. Una vez asignado el puerto de enlace y una IP para cada Host, se abre el puerto de enlace para la transferencia de datos.

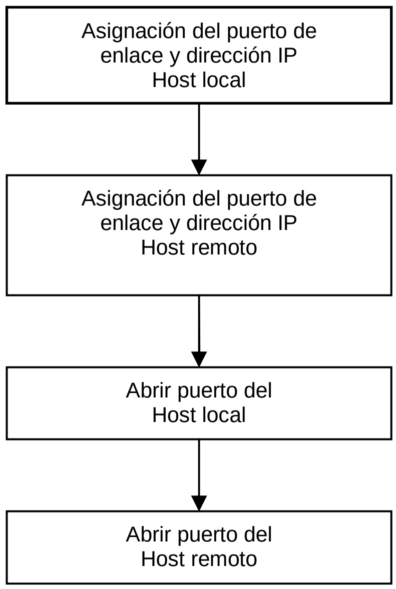


Figura 2.13. Pasos para la configuración de la red LAN

2.3.2 CONFIGURACIÓN DE LA COMUNICACIÓN SERIAL

La transferencia de datos entre el Host local y el controlador (tarjeta de control para el brazo real) se la realizó mediante comunicación serial, para lo cual, se determina el pórtico serial (en nuestro caso el COM4) por el cual se va a transmitir los datos. Se configuran los parámetros tales como: velocidad de transmisión de 9600 bps, trama de 8 bits, sin paridad y con bit de parada. Además se debe abrir el puerto de comunicación serial para permitir la transferencia de datos.

2.3.3 CONECTAR LAS CÁMARAS Y TARJETA DE CONTROL

Las cámaras implementadas en este trabajo son las WebCAM GF 112, las mismas que son conectadas a cualquier puerto USB previa instalación del driver.

2.3.4 EJECUTAR EL ARCHIVO HOST_LOCAL DENTRO DE MATLAB

Una vez ejecutado el archivo Host_Local de Matlab, se despliega la siguiente pantalla (Figura 2.14), en la que se muestra el HMI del sistema.

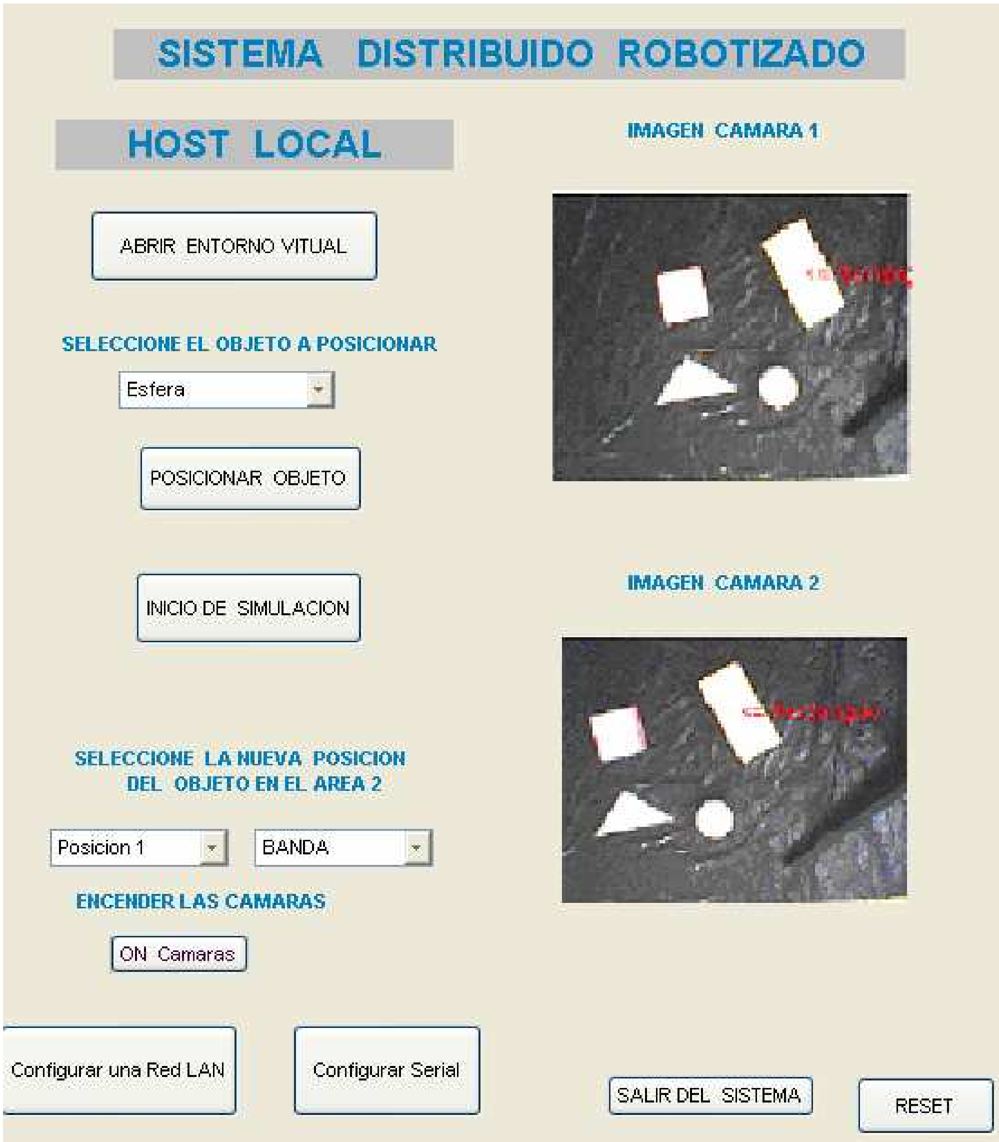


Figura 2.14. Interfaz Gráfica del Host Local

On cámaras.- Este botón permite abrir el puerto de las cámaras y activar las mismas para la obtención de las imágenes, se debe mencionar que debido a las características de las cámaras es necesario esperar unos 8 segundos, para evitar tomar imágenes oscuras o desenfocadas, este proceso se realiza una sola vez, al inicio.

Imagen cámara 1 y cámara 2.- Estas muestran las imágenes capturadas desde las cámaras web estándar.

Selección el Objeto a Posicionar.- En esta pestaña se selecciona el tipo de objeto que se desea detectar (esfera, paralelepípedo, cubo ó pirámide).

Posicionar Objeto.- Este botón permite posicionar el objeto seleccionado, el mismo que será visualizado en la imagen cámara 1 y cámara 2.

Abrir entorno virtual.- Permite abrir la simulación del sistema distribuido.

Inicio de simulación.- Permite iniciar la simulación de la realidad virtual que fue abierta previamente por el botón simulación ON.

Configurar Serial.- Configura la comunicación serial entre el computador y la tarjeta de control, esto se lo realiza una sola vez.

Configuración de una red LAN.- Configura los valores del puerto de enlace y las direcciones IP de este Host.

Seleccione la nueva posición del objeto.- En éstas pestañas se selecciona si el segundo manipulador coloca el objeto en la mesa de trabajo (pestaña izquierda) o en la banda trasportadora (pestaña derecha).

Salir del Sistema.- Este botón, cierra todas las pantallas del sistema.

Reset.- Reinicia todas las variables y ubica a los manipuladores en la posición inicial.

2.3.5 ACTIVAR LAS CÁMARAS Y ABRIR LA PANTALLA DE SIMULACIÓN

Una vez desplegada la pantalla del Host local, se presiona el botón ***ON Cámaras*** para activar las cámaras y de esta manera poder capturar las imágenes para su posterior procesamiento.

2.3.6 SELECCIÓN DE UN OBJETO DETERMINADO

El presente trabajo contempla la selección de cuatro figuras entre las cuales se toma: esfera, cubo, paralelepípedo y pirámide.

2.3.7 EJECUTAR EL ALGORITMO DE PROCESAMIENTO Y CLASIFICACIÓN DE LA IMAGEN

Los algoritmos a ejecutar son de procesamiento de imágenes, detección y clasificación de objetos. La clasificación de objetos se lo realizó utilizando una red neuronal de perceptron multicapa entrenada con el algoritmo de back propagation con capa oculta. Estos algoritmos se detallarán en los apartados 2.5 y 2.6, respectivamente.

2.3.8 INICIAR LA SIMULACIÓN DEL SISTEMA ROBOTIZADO

Para iniciar y ejecutar la simulación es necesario determinar la traslación y rotación que sufre cada una de las partes del manipulador. Para este efecto es necesario realizar un análisis matemático. En el presente trabajo se emplea la cinemática inversa, para lo cual se requiere determinar que valores tienen que tomar las variables articulares para que el extremo del brazo robótico se encuentre en una posición y orientación dada. Esto se desglosará en el apartado 2.8.

2.4 EJECUTAR EL ARCHIVO HOST_REMOTO DENTRO DE MATLAB

En la Figura 2.15 se presenta la interfaz gráfica del Host_Remoto, la misma que se detalla a continuación:

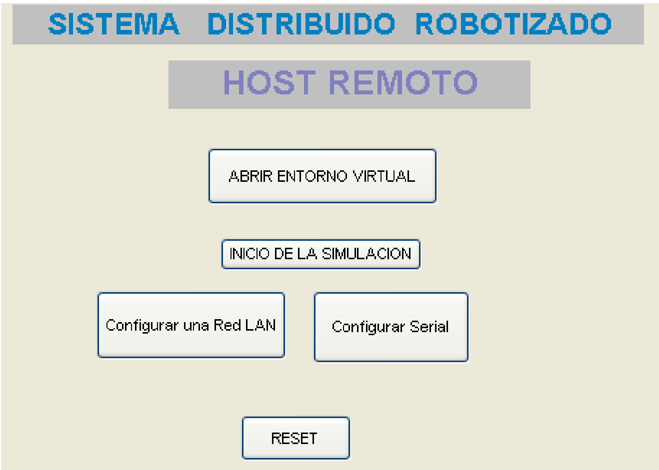


Figura 2.15. Interfaz Gráfica del Host_Remoto

Abrir entorno virtual.- Permite abrir la simulación del sistema distribuido.

Inicio de simulación.- Permite iniciar la simulación de la realidad virtual que fue abierta previamente por el botón simulación ON.

Configurar Serial.- Configura la comunicación serial entre el computador y la tarjeta de control. Esto se lo realiza una sola vez.

Configuración de una red LAN.- Configura los valores del puerto de enlace y las direcciones IP de este Host.

Reset.- Reinicia todas las variables y ubica a los manipuladores en la posición inicial.

2.5 ALGORITMO DE PROCESAMIENTO DE IMÁGENES

Se trabajará con un entorno no estructurado, para lo cual se necesita reconocer, localizar y posicionar los objetos dentro de un entorno real. Para comprender de mejor manera la implementación de este algoritmo, en la Figura 2.16 se muestra un diagrama de flujo del programa.

2.5.1 OBTENCIÓN DE LA IMAGEN

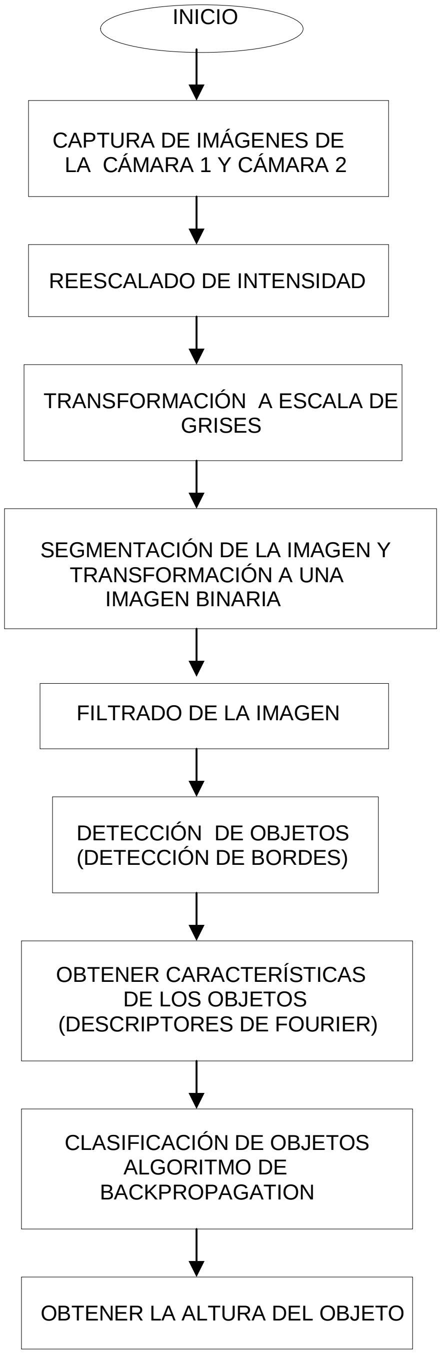


Figura 2.16. Diagrama de flujo del algoritmo de procesamiento de imágenes

Un proceso clave para que el procesamiento de imágenes no sea tan complicado es la obtención de una imagen de buena calidad, para los fines que convenga a la aplicación.

En la obtención de la imagen es necesario que el contraste entre el objeto y el fondo sea muy bueno, debe tener una buena iluminación y evitar sombras sobre cada uno de los objetos

La obtención de la imagen de la Figura 2.17 se realizó con una cámara web estándar. Entre mejores características posea la cámara y mejor sistema de iluminación se tenga, se obtiene mejor calidad en las imágenes y por consiguiente el procesamiento se simplifica.

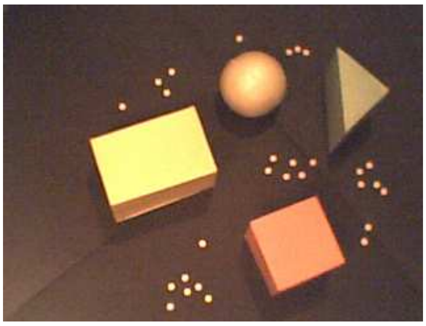


Figura 2.17. Imagen capturada con los objetos

El comando empleado para capturar las imágenes de las cámaras es ***videoinput,*** en el cual se define un número de identificador para cada cámara. En el presente caso se identificará con 1 la cámara uno y con 2 para la cámara dos. Además se define el formato de video y el adaptador de dispositivo que en este caso es el VFW (video para Windows).

2.5.2 REESCALADO DE INTENSIDAD

Al realizar la captura y procesamiento de la imagen se observó que la componente de brillo afecta directamente a la extracción de las características de los objetos, por lo cual fue conveniente reescalar la intensidad. En este caso se empleo la función ***imadjust.*** Esta herramienta permite transformaciones de intensidad. Esta función depende únicamente de la intensidad de cada pixel y no de la posición, por lo cual para realizar el escalado de grises, se asigna un valor a gama, que para el caso presente es un valor de 5. Este fue tomado en base a pruebas realizadas, tratando de obtener la menor variación en las características de los objetos.

2.5.2.1 Transformación a escala de grises

Mediante el comando ***rgb2gray,*** se convierte la imagen RGB o de mapa de colores a una imagen de escala de grises (Figura 2.18), debido a que en la aplicación presente solo se pretende identificar la forma del objeto y no el color del mismo.

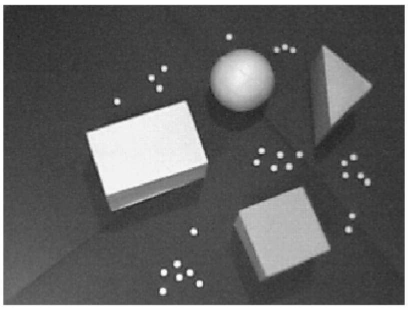


Figura 2.18. Imagen en escala de grises

2.5.2.2 Segmentación de la imagen y transformación a una imagen binaria

La segmentación es un proceso en el cual se divide la imagen en regiones, diferenciando los objetos que aparecen y separándolos del entorno en el que se encuentran. En el presente trabajo se empleará el método de selección del umbral óptimo o método de Otsu [1], el cual se encuentra implementado en Matlab mediante la instrucción ***graythresh***. Una vez obtenida la segmentación de la imagen se procede a transformarla a una imagen binaria, mediante el comando ***im2bw*** obteniendo una imagen de intensidad normalizada de valores ceros o unos, como se muestra en la Figura 2.19.

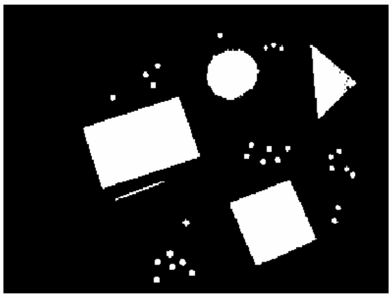


Figura 2.19. Imagen binaria

2.5.2.3 Filtrado de la imagen usando operaciones morfológicas

Se genera un arreglo de conectividad entre los diferentes objetos que contienen N número de píxeles, esto quiere decir que se genera una matriz que concentra un grupo de píxeles vecinos para formar un objeto, y así sucesivamente con todos los objetos de la imagen. Sin embargo, hay objetos que por su número reducido de píxeles se pueden considerar despreciables (ruido), y para eliminar estos objetos de la imagen binaria obtenida se emplea la operación morfológica abierta que está implementada en Matlab en el comando ***bwareaopen.*** Para el caso presente se ha establecido un valor umbral de 300 pixeles, es decir cualquier objeto que contenga un número igual o menor a 300 pixeles se eliminan de la imagen (Figura 2.20). Este valor fue tomado en base a pruebas realizadas. Con esta técnica no se produce un efecto de difuminado sobre características de interés en la imagen como pueden ser los bordes, las esquinas, etc.

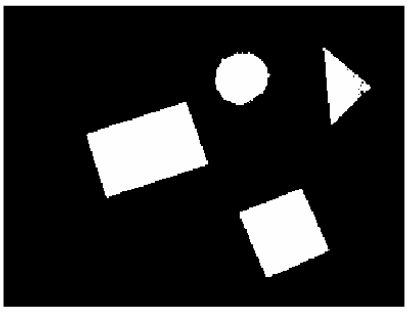


Figura 2.20. Imagen filtrada

2.5.3 DETECCIÓN DE OBJETOS

Para la detección de objetos se empleó la técnica de detección de bordes, que es la localización de los puntos en los que se produce una variación de iluminación. En este trabajo se emplea el comando ***bwboundaries,*** el mismo que permite determinar el contorno de los objetos existentes en la imagen. Esta instrucción está basada en la concatenación del número de píxeles vecinos que para el caso presente se tomo el valor por defecto, que es de 8 píxeles vecinos.

Una vez conocidos los bordes de los objetos, implícitamente se tiene como dato los valores X y Y correspondientes al perímetro de cada figura. Una vez obtenidos los valores X y Y de cada objeto se puede determinar el centroide de los mismos, con lo cual queda plenamente identificada la posición del objeto en $2D$.

2.5.4 CLASIFICACIÓN DE LOS OBJETOS

Una vez determinados, posicionados y caracterizados los objetos de la imagen, se llega a la tarea de identificar y clasificar los mismos. Los algoritmos implementados en este trabajo son, de Área-Perímetro, de umbral óptimo de brillo, método de matrices, Método basado en el espectro de Fourier y algoritmo bakprogation. Estos se explican de mejor manera en el apartado 2.6.

2.5.5 DETERMINACIÓN EN 3D DE LOS OBJETOS (VISIÓN ESTÉREO)

En este trabajo se consideró que todos los objetos se encuentran perpendiculares al lente de la cámara, sin embargo bajo esta suposición existe una variación respecto al plano real, en pruebas realizadas se observó que la variación depende de la ubicación de las cámaras, por lo cual se realizó una compensación en la determinación de la distancias para obtener mejores resultados. Para la determinación de la distancia de un objeto se aplicará el método de modelos alineados explicado en el Capítulo I.

Las cámaras se encuentran ubicadas a una altura h sobre la mesa de trabajo. Para la determinación de la altura de un objeto es necesario conocer el cálculo de la focal y de la distancia de los centros de las imágenes de cada cámara; sin embargo, la distancia focal (f) de la cámara no fue posible determinarla, debido a que es una característica propia de cada cámara, por lo cual, la calibración de estos parámetros se lo efectuó en base a pruebas. La distancia de separación entre los centros ópticos de las cámaras (B) se tomo con una regla. La técnica de modelos alineados se caracteriza por mantener constante el valor de la coordenada Y de cada objeto dentro de sus respectivas imágenes, esto quiere decir que para poder aplicar este modelo se debe mantener a una misma altura las dos cámaras. Como se observa en la Figura 2.21 y Figura 2.22, el valor en píxeles del eje Y permanece casi constante: 372.27 píxeles para la cámara 1 y 372,52 píxeles para la cámara 2.

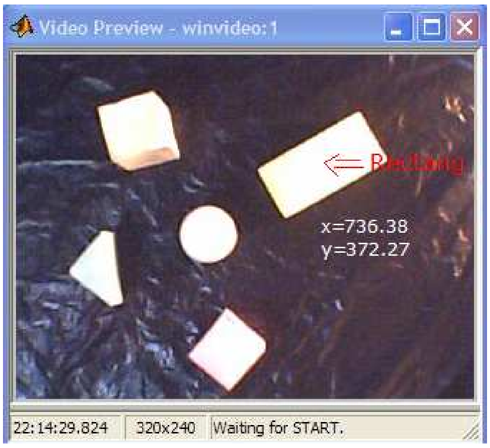


Figura 2.21. Centroide del objeto cámara 1

Figura 2.22. Centroide del objeto cámara 2

En base a la teoría vista en el Capítulo I, una vez determinados los parámetros de las cámaras, solo resta determinar los valores de x de los centroides de cada figura en las dos imágenes, para determinar la altura del objeto (Ecuación 1.35).

2.6 MÉTODOS DE CLASIFICACIÓN DE OBJETOS

2.6.1 MÉTODO BASADO EN EL ÁREA-PERÍMETRO

Este método consiste en calcular el cociente entre el perímetro y el área de cada objeto, dependiendo del valor de esta métrica (Perímetro/Área) se pueden clasificar los objetos. Sin embargo, al tener dos objetos con perímetros y áreas similares, pero de diferente forma el algoritmo clasifica estos objetos en un solo grupo, razón por lo cual una vez experimentada no se consideró esta metodología en el presente trabajo.

2.6.2 MÉTODO BASADO EN UMBRAL ÓPTIMO DE BRILLO DEL OBJETO

Este método consiste en obtener una función discreta (histograma) que representa la frecuencia de los valores de luminancias que aparecen en la imagen. Para determinar el valor del umbral máximo del objeto, se recorre el histograma de 0 a 255, calculando el pico máximo de intensidad. En la Figura 2.23, se observa el histograma de una imagen que contiene un cubo, en la misma que se observa un pico máximo de intensidad de brillo del cubo y del fondo de la imagen.

Este método genera buenos resultados siempre que la imagen analizada contenga un número constante de objetos. Si el número de figuras varía en la imagen, los picos máximos de cada objeto también varían. Razón por la cual una vez experimentada esta técnica no se lo implementó en este trabajo.

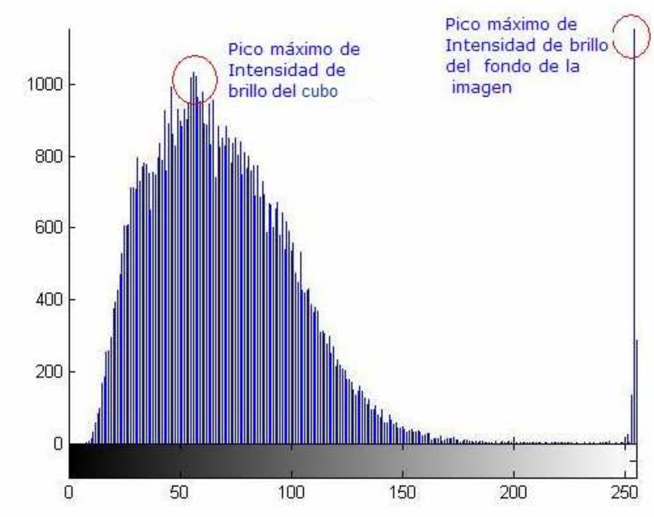


Figura 2.23. Histograma para un cubo

2.6.3 MÉTODO BASADO EN MATRICES

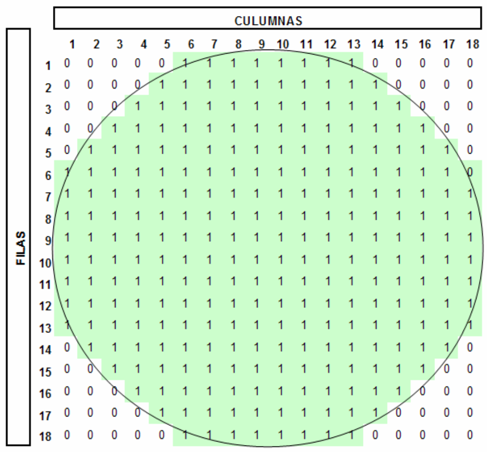


Figura 2.24. Matriz patrón para la esfera

Este método consiste en obtener una matriz binaria por cada objeto de la imagen y compararla con una matriz binaria patrón. En la Figura 2.24, se muestra la matriz patrón de una esfera vista en $2D$. Este algoritmo presentó excelentes resultados, sin embargo la carga computacional que este requería, ocasionaba un tiempo de ejecución muy elevado en el orden de los 6 segundos para un computador Pentium III. En vista de lo cual se busco otra alternativa, con menor carga computacional.

2.6.4 MÉTODO BASADO EN TRASFORMADA DE FOURIER

Mediante técnicas de procesamiento de imágenes se ha extraído el objeto seleccionado (Figura 2.25), a la misma que se procede a aplicar la transformada de Fourier para obtener los coeficientes de Fourier, con el fin de obtener las características más relevantes de cada objeto.

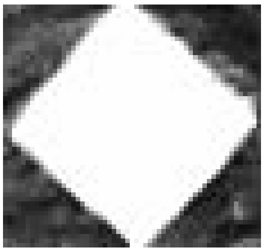


Figura 2.25. Cubo seleccionado rotado 45º

Se calculó la transformada de Fourier de cada objeto de la imagen, a diferentes valores de intensidad y ángulo de rotación, con el fin de obtener matrices de $m_i \times n_j$ que representa los coeficientes de Fourier de cada uno de los objetos. Así, al aplicar la transformada de Fourier para un cubo (Figura 2.25) se obtiene la forma de onda de la Figura 2.26, y de esta forma se procede con cada uno de los objetos contenidos en la imagen. Las Figuras 2.27, 2.28 y 2.29, presentan la transformada de Fourier en $2D$ del cubo, pirámide y paralelepípedo.

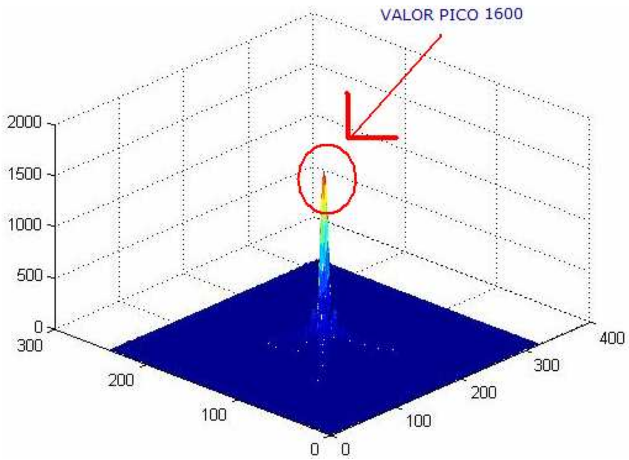


Figura 2.26. Representación gráfica de Fourier del cubo en $2D$

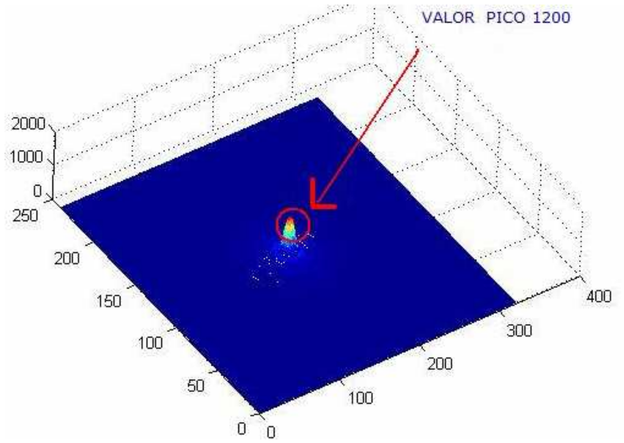


Figura 2.27. Representación gráfica de Fourier de la esfera en $2D$

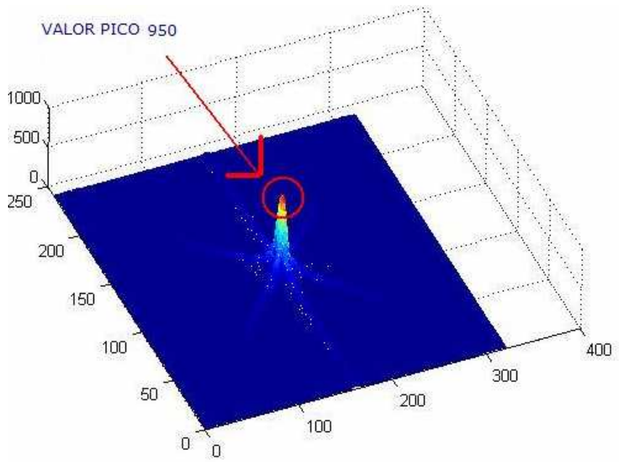


Figura 2.28. Representación gráfica de Fourier de la pirámide en $2D$

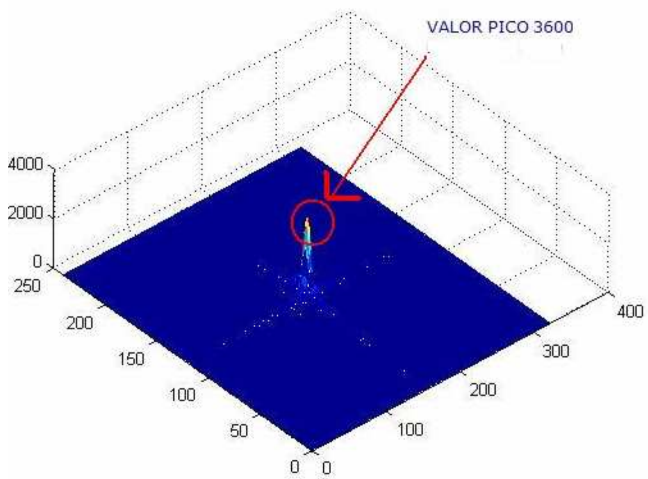


Figura 2.29. Representación gráfica de Fourier de un paralelepípedo en $2D$

Como se puede observar, la transformada de Fourier es diferente para cada objeto. Por lo que en este trabajo se implementó la Transformada de Fourier como una muy buena herramienta para la clasificación de objetos.

Tratando de no crear un algoritmo con una fuerte carga computacional, se optó por encontrar un solo valor que caracterice a toda la matriz de Fourier, para lo cual se realizó la sumatoria de todas las filas que conformar dicha matriz, obteniéndose un vector $1 \times n$ que contiene las características de cada figura. En base a pruebas realizadas se observó que el valor máximo de estos vectores, permiten caracterizar a cada figura; es decir, se obtiene un valor máximo para cada objeto de la imagen, con lo cual quedan plenamente caracterizadas cada una de las figuras. Una vez extraída esta característica de los objetos que se encuentran contenidas en la imagen, se puede realizar la clasificación de los mismos.

2.6.5 MÉTODO BASADO EN TRASFORMADA DE FOURIER Y CLASIFICACIÓN MEDIANTE UNA RED NEURONAL

Este método fue implementado en el presente trabajo y consiste en obtener las características de los objetos en base a la transformada de Fourier (descrito en el apartado 2.6.4) y clasificar los mismos en base a una Red Neuronal MLP entrenada con el algoritmo de Back Propagation, que se describe de mejor manera en al apartado 2.7.

2.7 RED NEURONAL MLP Y EL ALGORITMO BACK PROPAGATION

2.7.1 RED NEURONAL MLP

La red neuronal está constituida por una capa de entrada, una oculta y una de salida tal como se ilustra en la Figura 2.30.

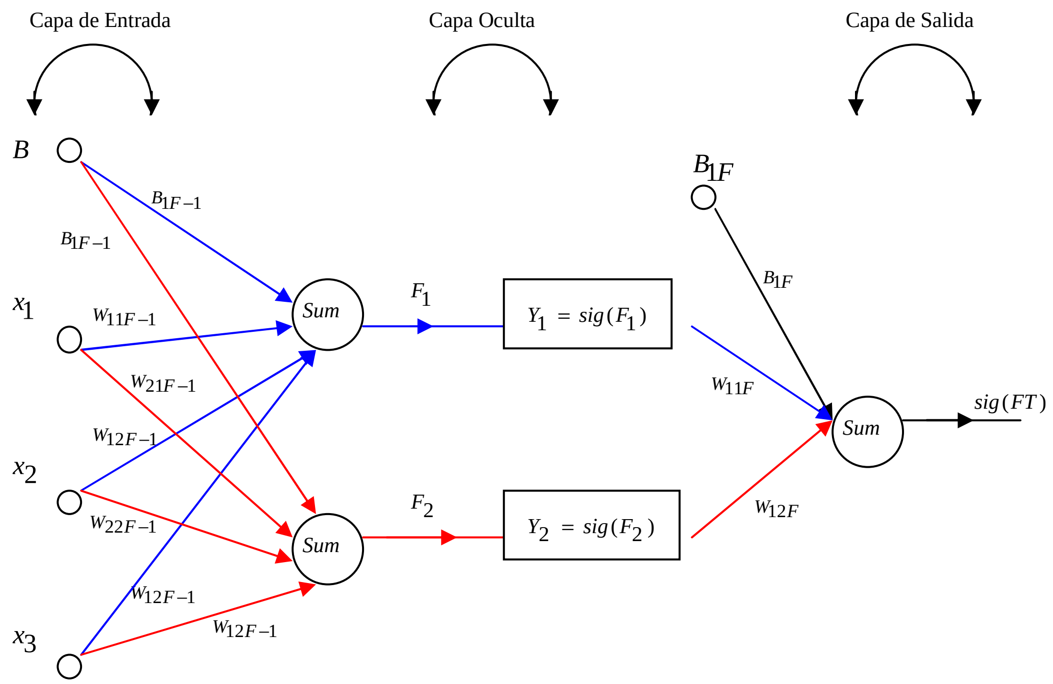


Figura 2.30. Estructura de la red neuronal implementada

La especificación de los valores de entrada se realiza con un conjunto de vectores con entradas reales (Transformada de Fourier). El algoritmo implementado consta de cuatro entradas, tres entradas variables $(X_1, X_2\ y\ X_3)$ y una entrada B fija $(B = 1)$. Las entradas de la red $X_1, X_2\ y\ X_3$, corresponden a la transformada de Fourier calculada a 0 grados, 45 grados y 90 grados respectivamente.

2.7.2 EL ALGORITMO BACK PROPAGATION

La función de transferencia "sig" que se considera en el algoritmo es la función sigmoidal, esta función, además de ser diferenciable, tiene la particularidad de que su derivada se puede expresar en términos de sí misma. Esto servirá para simplificar los cálculos en el algoritmo de aprendizaje aquí descrito. Para comprender de mejor manera el algoritmo, en la Figura 2.31 se presenta un diagrama de bloques de la red implementada.

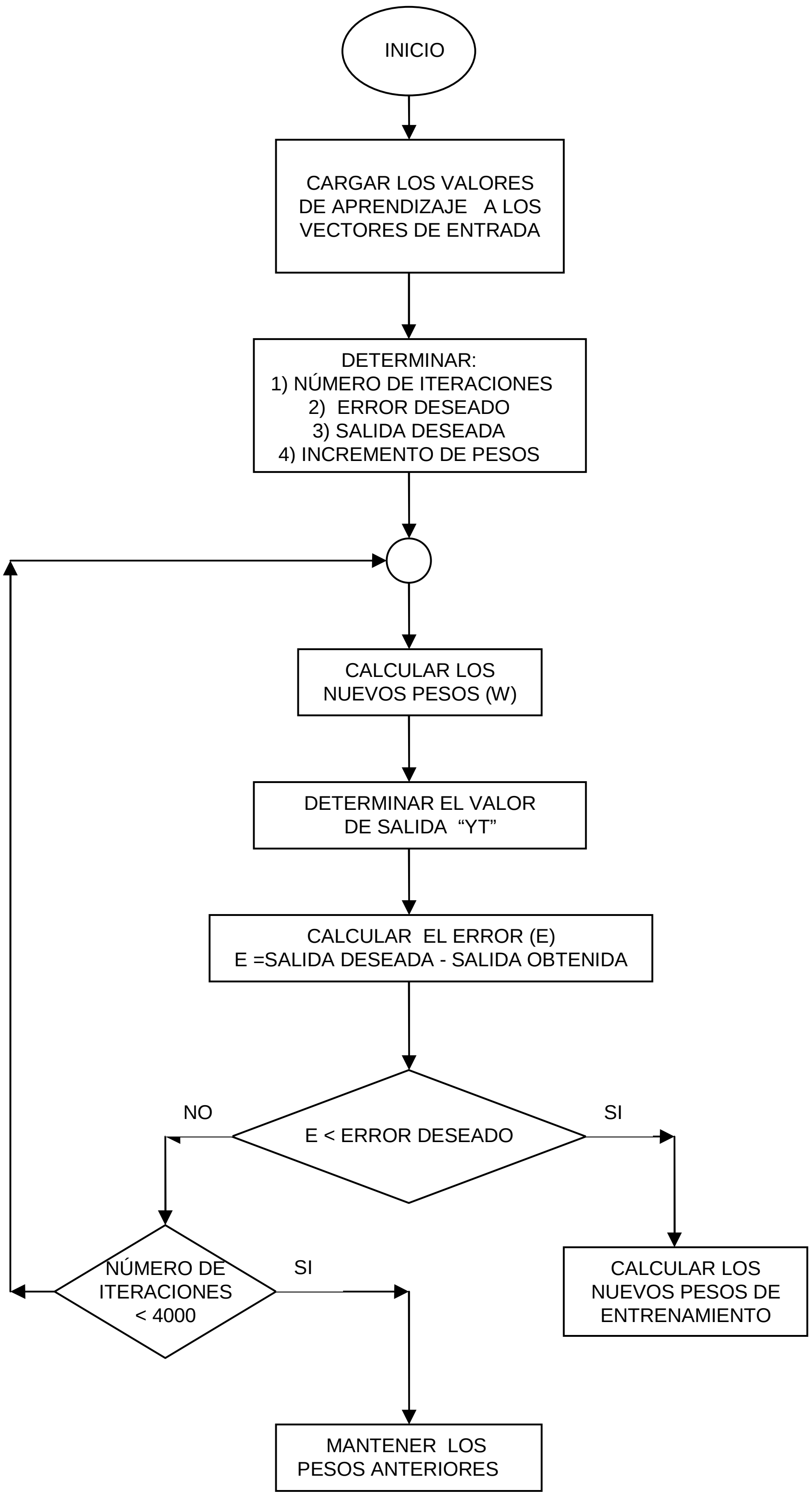


Figura 2.31. Diagrama de bloques del algoritmo Back Propagation

En la fase de aprendizaje se calcula la salida de la red Y_T, en base a los vectores de entrenamiento (patrones), los cuales se obtienen mediante el algoritmo de procesamiento de imágenes. La dimensión de estos vectores viene determinada por las características del proceso. En el caso presente estos vectores normalizados son de 1×9. Siguiendo el diagrama mostrado en la Figura 2.30, se obtienen las siguientes relaciones:

$$F_1 = B_{1F-1} + x_1.w_{11F-1} + x_2.w_{12F-1} + x_3.w_{13F-1} \tag{2.1}$$

$$F_2 = B_{2F-1} + x_1.w_{21F-1} + x_2.w_{22F-1} + x_3.w_{23F-1} \tag{2.2}$$

$$Y_1 = sig(F_1) \tag{2.3}$$

$$Y_2 = sig(F_2) \tag{2.4}$$

$$F_T = y_1.w_{11F} + y_2.w_{12F} + B_{1F} \tag{2.5}$$

$$Y_T = sig(F_T) \tag{2.6}$$

Una vez definida la salida de la red (Y_T), el siguiente paso es compararla con la salida deseada "t" y calcular el error en base a la siguiente fórmula:

$$E(X, W_F, W_{F-1}) = \frac{1}{2}\sum_{i=1}^{m}(Y_{T_i} - t_i)^2 \tag{2.7}$$

Se compara el error calculado con un error deseado, si el error obtenido es mayor al error deseado la red continua ajustando los pesos de cada neurona para reducir el error. Este proceso se repite varias veces hasta encontrar un error deseado; sin embargo, en algunos casos el algoritmo puede divergir sin encontrar una solución, por lo cual es necesario limitar el número de iteraciones. Para el caso presente los valores del error deseado y el número de iteraciones se los tomó de 0.0005 y 4000, respectivamente, en base a pruebas realizadas. Concluida la fase de aprendizaje con los nuevos pesos de la red, se pasa a la fase de operación en la cual la red queda lista para clasificar los objetos de la imagen.

2.7.2.1 Ajuste de los pesos de salida

Por las características anteriormente mencionadas, se puede considerar una sola neurona de salida y realizar el análisis como si estuviese aislada. Esto se puede apreciar en la Figura 2.32. [5]

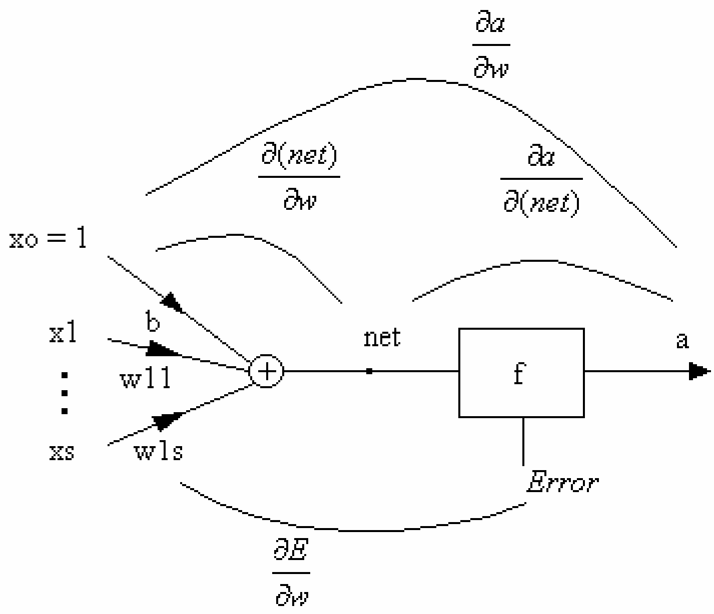


Figura 2.32. Estructura de la red neuronal

Lo que se busca es saber cómo varía el error con las variaciones de los pesos de las entradas, o lo que es lo mismo:

$$\frac{\partial E}{\partial w} \tag{2.8}$$

Entendiendo por E el error cometido, es decir, la diferencia entre la salida obtenida y la deseada. Para llegar a conocer esto se debe seguir el siguiente desarrollo:

$$\frac{\partial E}{\partial w} = \frac{\partial E}{\partial a} \cdot \frac{\partial a}{\partial w} \tag{2.9}$$

La primera de las dos partes en que se divide el problema es simplemente la variación del error con el valor de salida "a", y conociendo la definición del error (error cuadrático medio) es trivial llegar al resultado final:

$$\frac{\partial E}{\partial a} = -2 \cdot (t - a) \tag{2.10}$$

Siendo t la salida deseada y a la obtenida. Por otra parte el segundo problema se puede dividir en otros dos [5]:

$$\frac{\partial a}{\partial w} = \frac{\partial a}{\partial (net)} \cdot \frac{\partial (net)}{\partial w} \tag{2.11}$$

Y teniendo en cuenta que:

$$net = w \cdot x \tag{2.12}$$

Obtenemos:

$$\frac{\partial (net)}{\partial w} = \frac{\partial \left[\sum w \cdot x\right]}{\partial w} = x \tag{2.13}$$

$$\frac{\partial a}{\partial (net)} = \frac{\partial f(net)}{\partial (net)} = f'(net) \tag{2.14}$$

De lo que concluimos que f ha de ser derivable. Y uniendo todos los pasos anteriores llegamos a la expresión final [5]:

$$\Delta w = w(k+1) - w(k) = -\frac{\partial E}{\partial w} = 2 \cdot (t - a) \cdot f' \cdot x \tag{2.15}$$

Si f es la función sigmoidal [5]:

$$f(x) = \frac{1}{1 - e^{-x}} >>>> f'(x) = \frac{e^{-x}}{(1 + e^{-x})^2} \tag{2.16}$$

Y al reemplazar en la expresión que se había calculado anteriormente (haremos la sustitución $a = f(x)$:

$$\Delta w = 2 \cdot (t - a) \cdot a \cdot (1 - a) \cdot x \tag{2.17}$$

Si f es la función $\tan h$ [5]:

$$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} >>>> f'(x) = x(1 - f^{2}(x)) \tag{2.18}$$

$$\Delta w = 2 \cdot (t - a) \cdot x \cdot (1 - a^{2}) \cdot x \tag{2.19}$$

Si f fuera la función lineal $f = x$ [5]:

$$f(x) = x >>>> f'(x) = 1 \tag{2.20}$$

$$\Delta w = 2 \cdot (t - a) \cdot x \tag{2.21}$$

2.7.2.2 Ajuste de los pesos de la capa oculta

Para comenzar a entender en qué consiste el problema que nos ocupa vamos a estudiar la dependencia del error de salida de una neurona [5] de una capa de los pesos de la capa anterior. Para ello extraemos el siguiente gráfico (Figura 2.33).

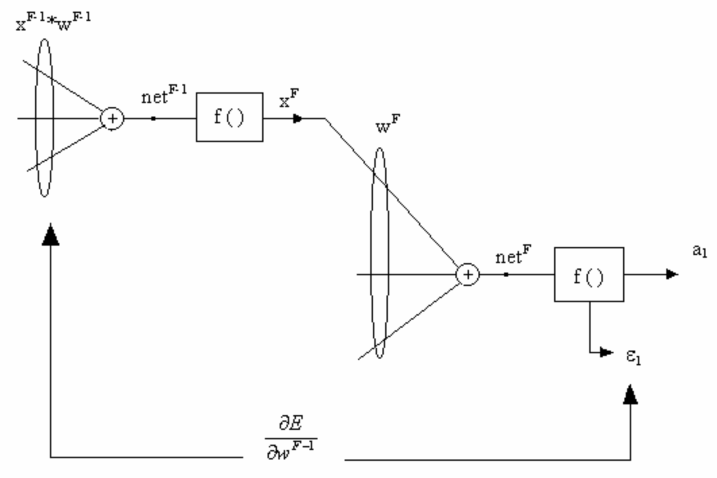


Figura 2.33. Relación capa de entrada vs capa de salida

Se busca la variación del error con los pesos de la capa anterior, separaremos el problema en pequeños pasos que simplifiquen la operación. Esto se puede ver a continuación:

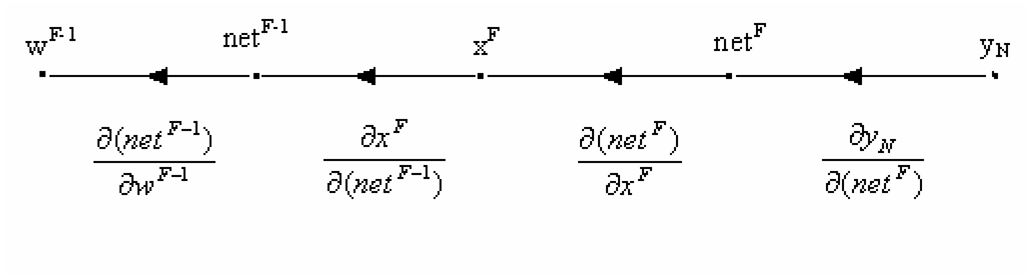


Figura 2.34. Relación capa de entrada vs capa de salida

Por lo que el problema consiste en hallar:

$$\frac{\partial E}{\partial w^{F-1}} = \frac{\partial E}{\partial y_N} \cdot \frac{\partial y_N}{\partial(net^F)} \cdot \frac{\partial(net^F)}{\partial x^F} \cdot \frac{\partial x^F}{\partial(net^{F-1})} \cdot \frac{\partial(net^{F-1})}{\partial w^{F-1}} \tag{2.22}$$

De la definición de error cuadrático:

$$\frac{\partial E}{\partial y_N} = -2 \cdot (y_d - y_n) \tag{2.23}$$

Directamente de la Figura 2.32:

$$\frac{\partial y_N}{\partial(net^F)} = f'(net) \tag{2.24}$$

Teniendo en cuenta que:

$$net^F = \sum x^F \cdot w^F \tag{2.25}$$

$$\frac{\partial(net^F)}{\partial x^F} = w^F \tag{2.26}$$

Una vez más se puede apreciar a partir de la Figura 2.30:

$$\frac{\partial x^F}{\partial (net^{F-1})} = f'(net^{F-1}) \tag{2.27}$$

Por último, y de manera análoga:

$$\frac{\partial (net^{F-1})}{\partial w^{F-1}} = x^{F-1} \tag{2.28}$$

Como conclusión tenemos que la regla para el aprendizaje para los pasos de la penúltima capa es [5]:

$$\frac{\partial E}{\partial w^{F-1}} = -2 \cdot (y_d - y_n) \cdot f'(net^F) \cdot w^F \cdot f'(net^{F-1}) \cdot x^{F-1} \tag{2.29}$$

De esta manera se puede calcular los pesos de la siguiente fase a partir de los de la fase anterior con la variación contraria al error, es decir:

$$w(k+1) - w(k) = -\frac{\partial E}{\partial w^{F-1}} \tag{2.30}$$

Y por tanto [5]:

$$w(k+1) = w(k) + 2 \cdot (y_d - y_n) \cdot f'(net^F) \cdot w^F \cdot f'(net^{F-1}) \cdot x^{F-1} \tag{2.31}$$

Para el resto de las capas se aplicaría la misma regla recursivamente hasta la entrada, momento en el cual se vuelve a aplicar otra entrada de entrenamiento y nuevamente se propaga el error hacia atrás. Para este trabajo, la red tiene: 1 capa oculta de 2 neuronas logísticas $(f = sigmoidal)$, 1 neurona logística de salida, pesos iniciales (randómicos), entradas de entrenamiento y una salida deseada (t).

2.8 ALGORITMO DE POSICIONAMIENTO DEL BRAZO ROBÓTICO

Para los cálculos en la cinemática inversa se requiere determinar que valores tienen que tomar las variables articulares para que el extremo del brazo robótico se encuentre en una posición y orientación dada. Teniendo en cuenta la matriz de transformación homogénea total que representa la posición y orientación del sistema de referencia del extremo del robot con respecto al sistema de la base, se desarrolla la cinemática inversa para un robot cuya cadena cinemática está compuesta por tres grados de libertad, se dispondrá de tres ecuaciones con tres incógnitas. Para el desarrollo de la cinemática inversa se parte del dato de la matriz de transformación homogénea ${}^{\circ}T_3$ que representa la posición y orientación. La matriz de transformación ${}^{\circ}T_3$ es la matriz correspondiente a los tres eslabones del Robot. Para determinar la cinemática inversa del Robot es necesario calcular los parámetros de Denavit Hartemberg [1], como se muestran en la Tabla 2.1.

Eje	a_{i-1}	α_{i-1}	d_i	θ_i
1	0	0	0	q1
2	0	90	L1	q2
3	L2	0	0	-q3
4	L3	0	0	0

Tabla 2.1. Parámetros de Denavit-Hartemberg

De los parámetros de Denavit Hartemberg y del desarrollo matemático en base a la fisonomía del robot se tienen las siguientes relaciones, donde q_1 es el ángulo de la articulación uno, q_2 es el ángulo de la articulación dos y q_3 es el ángulo de la articulación tres; L_{g1}, L_{g2} y L_{g3} representan la longitud de cada una de las articulaciones que para el caso presente es 35, 17 y 37 cm, respectivamente, las variables x, y, z son las coordenadas del extremo del robot. A continuación se presentan las ecuaciones que permiten calcular el valor de las variables articulares $(q1, q2, q3)$ en función de las coordenadas del extremo del robot (x, y, z):

$$q_1 = a\tan\left(\frac{y}{x+0.01}\right) \tag{2.32}$$

$$a = 4*L_{g2}{}^{\wedge 2}(x^{\wedge 2}+y^{\wedge 2}+z^{\wedge 2}-2*L_{g1}*z+L_{g1}{}^{\wedge 2}) \tag{2.33}$$

$$b = 4*L_{g2}*(z-L_{g1})*(x^{\wedge 2}+y^{\wedge 2}+z^{\wedge 2}-2*L_{g1}*z+L_{g1}{}^{\wedge 2}+L_{g2}{}^{\wedge 2}-L_{g3}{}^{\wedge 2}) \tag{2.34}$$

$$c = (x^{\wedge 2}+y^{\wedge 2}+z^{\wedge 2}-2*L_{g1}*z+L_{g1}{}^{\wedge 2}+L_{g2}{}^{\wedge 2}-L_{g3}{}^{\wedge 2})^{\wedge 2}-4*L_{g2}{}^{\wedge 2}*(x^{\wedge 2}+y^{\wedge 2}) \tag{2.35}$$

$$q_2 = a\sin\left(\frac{-b+sqrt(b^{\wedge 2}-4*a*c)}{2*a+0.01}\right) \tag{2.36}$$

$$q_3 = a\cos\left(\frac{x^{\wedge 2}+y^{\wedge 2}+z^{\wedge 2}+L_{g1}{}^{\wedge 2}-L_{g1}{}^{\wedge 2}-L_{g2}{}^{\wedge 2}-2*L_{g3}*z}{2*L_{g2}*L_{g3}+0.001}\right) \tag{2.37}$$

Una vez definido los ángulos q_1, q_2 y q_3 que corresponden a las articulaciones del robot, es necesario determinar el origen del eje de coordenadas de la articulación 2 $(dxo_{L3}, dyo_{L3}, dz_{L3})$, y el origen del eje de coordenadas $(dxo_{L4}, dyo_{L4}, dz_{L4})$ de la articulación 3. Para lo cual partiendo del análisis de la cinemática directa se obtienen las siguientes relaciones matemáticas:

$$P_{XYZ03} = T_{TRAS}\, T_{ROTX} T_{ROTY} T_{ROTZ} P_{REF\_1} \tag{2.38}$$

$$dxo_{L3} = Vector_X\, P_{XYZ03} \tag{2.39}$$

$$dyo_{L3} = Vector_Y P_{XYZ03} \tag{2.40}$$

$$dzo_{L3} = Vector_Z P_{XYZ03} \tag{2.41}$$

$$P_{XYZ04} = T_{TRAS}\, T_{ROTX} T_{ROTY} T_{ROTZ} P_{REF\_1} \tag{2.42}$$

$$dxo_{L4} = Vector_X P_{XYZ04} \tag{2.43}$$

$$dyo_{L4} = Vector_Y P_{XYZ04} \tag{2.44}$$

$$dzo_{L4} = Vector_Z P_{XYZ04} \tag{2.45}$$

Donde P_{XYZ03} y P_{XYZ04} son los vectores que contienen el origen del sistema de referencia de la articulación 2 y la articulación 3 respectivamente. Los vectores $Vector_X, Vector_Y, Vector_Z$ son vectores que permiten extraer el origen de coordenadas $(dxo_{L3}, dyo_{L3}, dzo_{L3})$ y $(dxo_{L4}, dyo_{L4}, dzo_{L4})$ de las articulaciones 2 y 3, para poder realizar la simulación en el entorno virtual. T_{TRAS} es la matriz de traslación y T_{ROTX} , T_{ROTY} y T_{ROTZ} son las matrices de rotación. P_{REF_1}, es un punto cualquiera de un sistema de referencia que se va ha trasladar a un nuevo sistema de referencia.

2.9 CONTROLADOR

El controlador contiene una ***tarjeta de control*** y una ***tarjeta de potencia*** que permiten operar un manipulador real de tres grados de libertad (Figura 2.35). El brazo real consta de cuatro motores DC (tres para las articulaciones y uno para la pinza) los cuales pueden trabajar en los dos sentidos de giro.

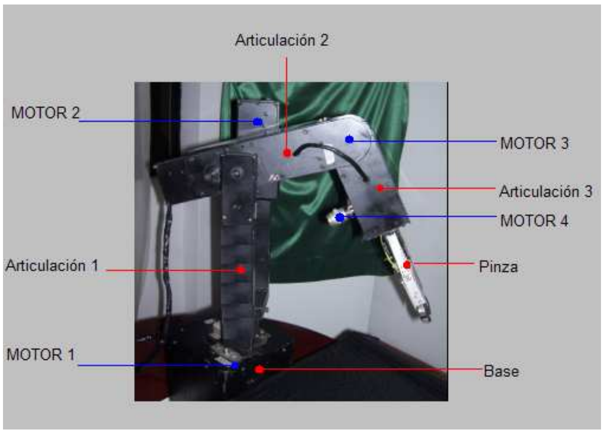


Figura 2.35. Brazo real de tres grados de libertad

El Host Local envía la nueva posición del brazo al controlador, los datos son recibidos por la tarjeta de control, la misma que entrega las señales correspondientes a la tarjeta de potencia para posicionar la pinza del brazo real sobre el objeto seleccionado. La ***tarjeta de control*** consta principalmente de un MAX232, un PIC 16F877A y borneras (8 líneas de control).

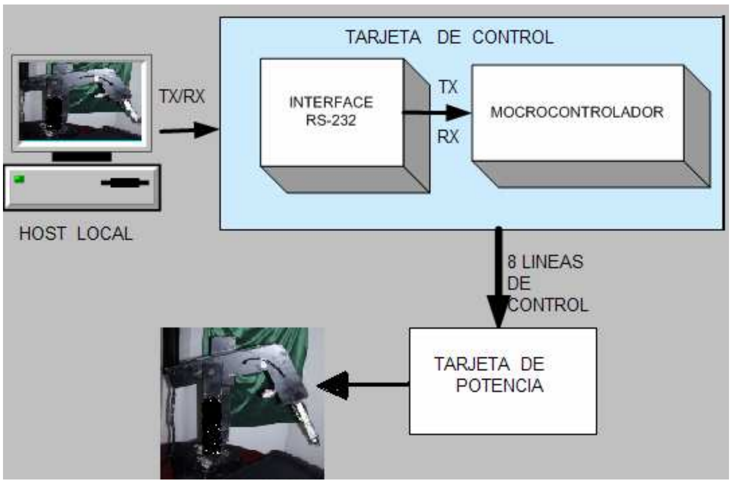
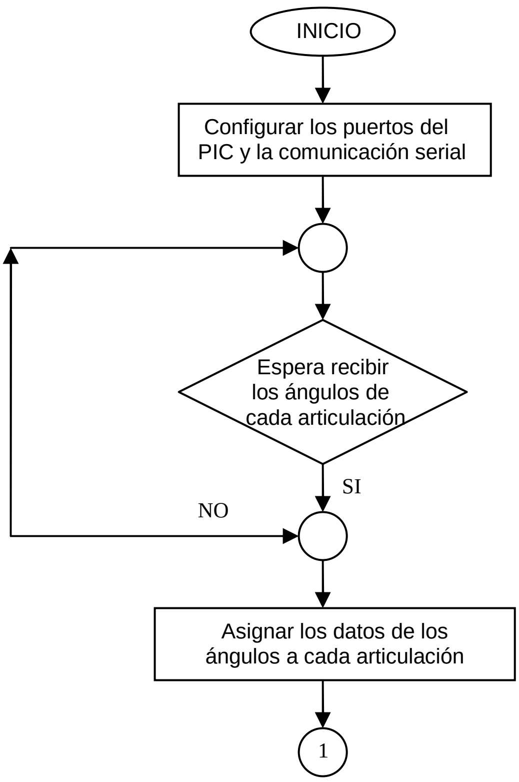


Figura 2.36. Diagrama de bloques del controlador

La ***tarjeta de potencia*** consta de 4 puentes H, y permite mover al brazo en base a las señales de control, en la Figura 2.36, se muestra un diagrama de bloques del controlador. En la Figura 2.37, se muestra un diagrama de flujo del programa del microcontrolador que posiciona el brazo sobre un objeto.



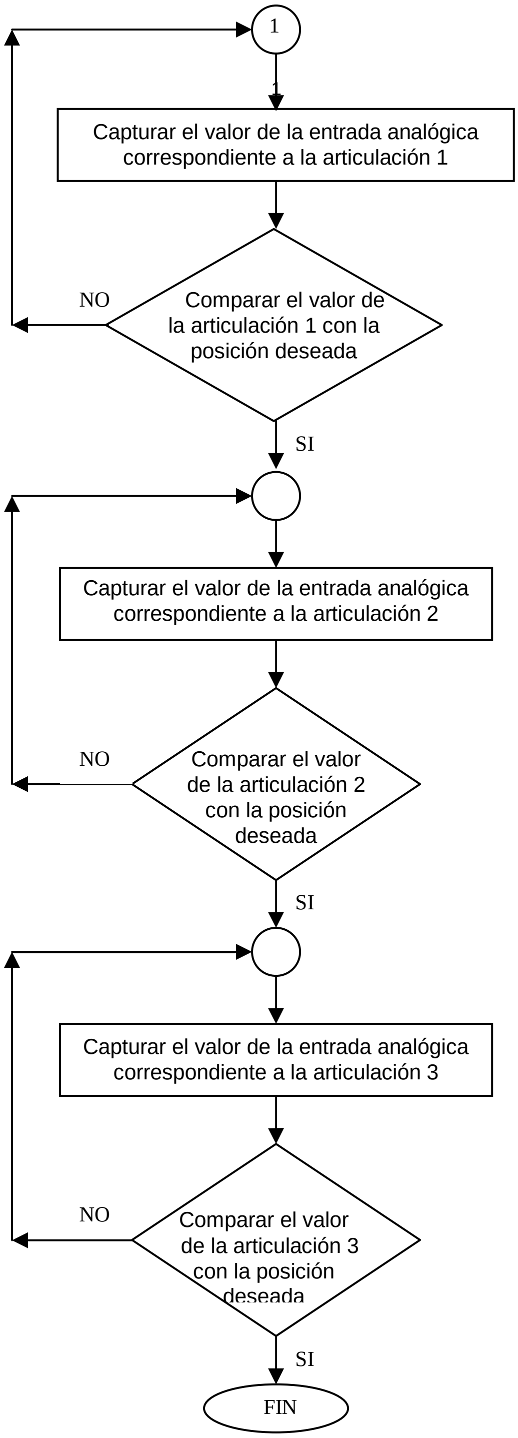


Figura 2.37. Diagrama de flujo del programa del PIC

En el presente capitulo, se desglosó los algoritmos implementados en este trabajo. Cabe destacar que se implementaron varios algoritmos de clasificación comparándolos entre ellos para determinar el más apropiado. En el siguiente capítulo, se presentarán las pruebas y resultados obtenidos, para validar los algoritmos desarrollados.

CAPÍTULO 3

PRUEBAS Y RESULTADOS

CAPÍTULO 3

PRUEBAS Y RESULTADOS

3.1 INTRODUCCIÓN

En este capítulo se presentan las pruebas y resultados obtenidos de la simulación, con la finalidad de establecer el margen de error que posee la simulación respecto a lo real.

Para este efecto, se realizaron las siguientes pruebas:

a. Pruebas de posicionamiento del manipulador.
b. Pruebas de posicionamiento de objetos.
c. Pruebas de clasificación de objetos.
d. Pruebas de comunicación de la red LAN.

3.2 PRUEBAS DE POSICIONAMIENTO DEL MANIPULADOR

En este apartado se realizaron las siguientes pruebas:

a. Pruebas de posicionamiento en la simulación del archivo.m
b. Pruebas de posicionamiento en la simulación del VRML.

3.2.1 PRUEBAS DE POSICIONAMIENTO EN LA SIMULACIÓN DEL ARCHIVO.M

El objetivo de realizar esta prueba es determinar el margen de error que tiene la simulación en archivo.m, respecto a cálculos matemáticos. En la Figura 3.1, se muestra la simulación de dos manipuladores para una posición cualquiera, en la cual se observa el valor de las coordenadas del origen de $L3$ para el manipulador_1. De igual forma en la Figura 3.2, se muestra las coordenadas de la pinza del manipulador 1; de ésta forma se tomaron los valores para diferentes posiciones.

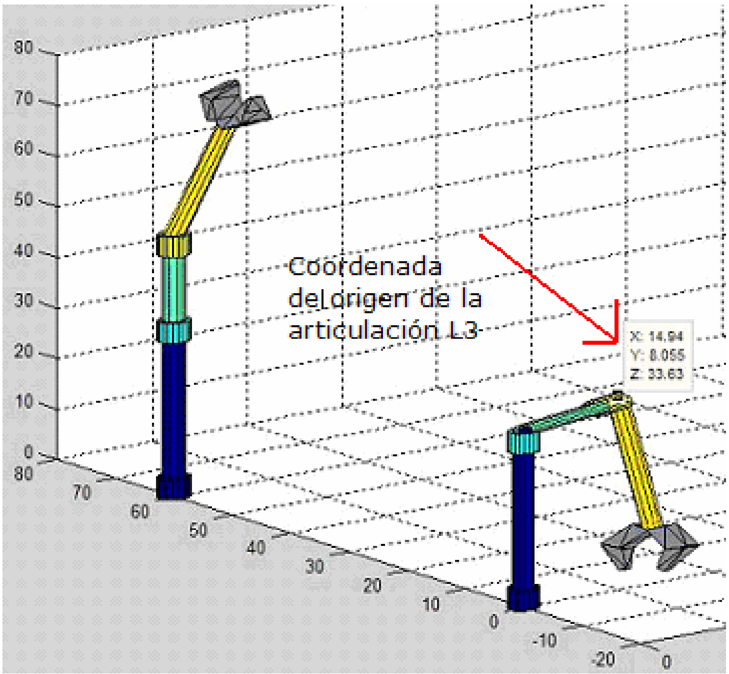


Figura 3.1. Coordenada del origen de la articulación $L3$

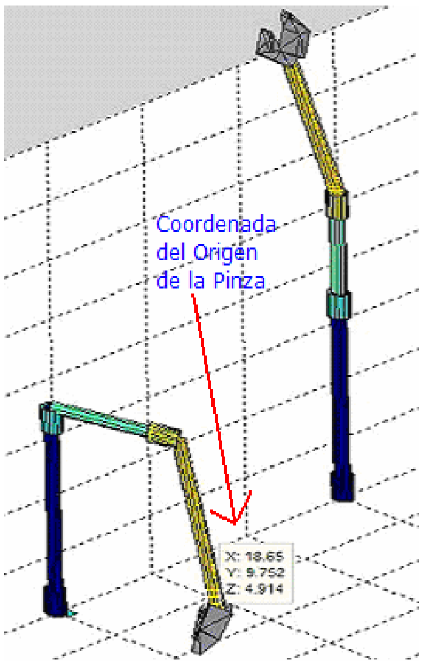


Figura 3.2. Coordenada del origen de la pinza

En la Tabla 3.1 y Tabla 3.2, se muestran los datos de la posición del manipulador obtenidos a partir de cálculos matemáticos y mediante la simulación. Los datos

$(dxo_{L3}, dyo_{L3}, dzo_{L3})$ representan las coordenadas del origen de la articulación L_3, mientras que $(X_{FINAL}, Y_{FINAL}, Z_{FINAL})$ representan las coordenadas del origen de la pinza.

POSICIONES	Cálculos matemáticos	Datos Simulación	Error (%)	Cálculos matemáticos	Datos Simulación	Error (%)	Cálculos matemáticos	Datos Simulación	Error (%)
	dxo_{L3}	dxo_{L3}		dyo_{L3}	dyo_{L3}		dzo_{L3}	dzo_{L3}	
POSICIÓN 1	12.34	12.4	0.49	11.52	11.65	1.12	32.9	34.22	3.86
POSICIÓN 2	12.16	12.42	2.14	11.7	11.69	0.09	32.91	34.42	4.39
POSICIÓN 3	13.68	14.56	6.43	5.53	5.56	0.54	26.56	27.28	2.64
POSICIÓN 4	14.74	14.94	1.36	8.13	8.1	0.37	32.61	33.63	3.03
POSICIÓN 5	14.76	15.05	1.96	6	6.31	4.91	29.08	30.14	3.52

Tabla 3.1. Datos de los cálculos matemáticos y datos de la simulación de las coordenadas articulación L_3

POSICIONES	Cálculos matemáticos	Datos Simulación	Error (%)	Cálculos matemáticos	Datos Simulación	Error (%)	Cálculos matemáticos	Datos Simulación	Error (%)
	X_{FINAL}	X_{FINAL}		Y_{FINAL}	Y_{FINAL}		Z_{FINAL}	Z_{FINAL}	
POSICIÓN 1	12.83	12.38	3.63	11.98	12.51	4.24	5	5	0
POSICIÓN 2	10.58	9.98	6.01	10.18	9.77	4.20	5	5.1	1.96
POSICIÓN 3	30.24	30.07	0.57	12.23	12.61	3.01	5	4.58	9.17
POSICIÓN 4	18.83	18.65	0.97	10.39	9.75	6.56	5	4.91	1.83
POSICIÓN 5	28	28.77	2.68	11.38	10.9	4.40	5	5.32	6.02

Tabla 3.2. Datos de los cálculos matemáticos y datos de la simulación de las coordenadas de la pinza

Se puede observar de la Tabla 3.1 y 3.2, que los errores generados en la simulación, puede deberse a que para trasladar el brazo en dirección del eje Z, se realiza una compensación (alineamiento al eje Z), debido a que no existe una función en Matlab que realice esta operación (traslación y rotación al mismo tiempo). Esta compensación genero una alta carga computacional y no muy precisa, por lo que se optó por buscar otra alternativa de simulación.

3.2.2 PRUEBAS DE POSICIONAMIENTO EN LA SIMULACIÓN DEL VRML

Esta prueba se realizó con la finalidad de determinar el margen de error del análisis matemático respecto a lo entregado por la simulación en VRML. Para determina una coordenada dentro del VRML, es suficiente trasladarse por el mundo virtual y observar en la parte inferior derecha la posición del observador, como se observa en las Figura 3.3.

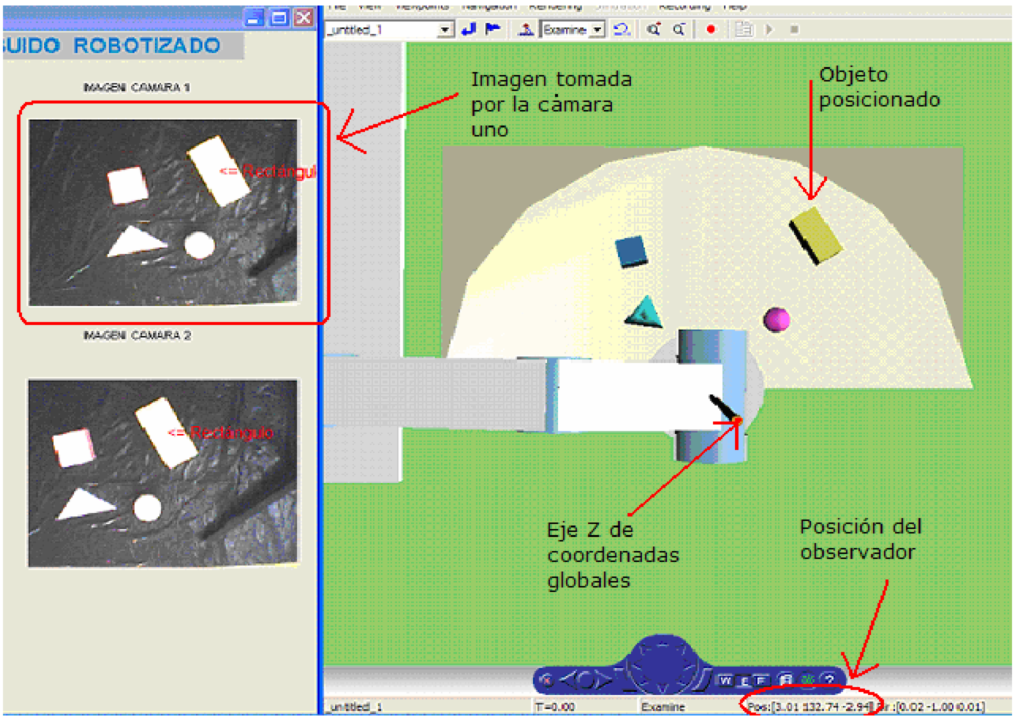


Figura 3.3. Vista superior del entorno virtual

A continuación se presenta el mismo manipulador de la Figura 3.3, observado desde otro punto de vista.

Figura 3.4. Vista frontal del entorno virtual

En la Tabla 3.3 y 3.4, se muestran los datos de la posición del manipulador 1 obtenidos a partir de cálculos matemáticos y mediante VRML.

POSICIONES	Cálculos matemáticos dxo_{L3}	Datos Simulación dxo_{L3}	Error %	Cálculos Matemáticos dyo_{L3}	Datos Simulación dyo_{L3}	Error %	Cálculos matemáticos dzo_{L3}	Datos Simulación dzo_{L3}	Error %
POSICIÓN 1	12.34	12.13	1.7	11.52	11.67	1.3	32.9	32.82	0.2
POSICIÓN 2	12.16	12.22	0.5	11.7	11.57	1.1	32.91	32.12	0.6
POSICIÓN 3	13.68	13.56	0.9	5.53	5.65	2.2	26.56	26.31	0.9
POSICIÓN 4	14.74	14.51	1.6	8.13	8.20	0.9	32.61	31.73	2.7
POSICIÓN 5	14.76	14.56	1.4	6	6.14	2.3	29.08	28.67	1.4

Tabla 3.3. Posicionamiento del objeto en VRML

POSICIONES	Cálculos matemát. X_{FINAL}	Datos Simulación X_{FINAL}	Error %	Cálculos matemát. Y_{FINAL}	Datos Simulación Y_{FINAL}	Error %	Cálculos matemát. Z_{FINAL}	Datos Simulación Z_{FINAL}	Error %
POSICIÓN 1	12.83	12.47	2.8	11.98	11.66	2.7	5	5.24	4.8
POSICIÓN 2	10.58	10.83	2.4	10.18	10.33	1.5	5	4.87	2.6
POSICIÓN 3	30.24	30.52	0.9	12.23	12.45	1.8	5	5.1	2.0
POSICIÓN 4	18.83	18.47	1.9	10.39	10.62	2.2	5	2.91	2.6
POSICIÓN 5	28	28.29	1.0	11.38	11.73	3.1	5	3.32	1.4

Tabla 3.4. Posicionamiento del objeto en VRML

Como se puede observar en la Tabla 3.3 y 3.4, los errores obtenidos en VRML son menores con respecto a los datos obtenidos mediante la simulación por "archivo.m". Dentro del entorno VRML la posición donde se encuentra el observador es proporcionada por el simulador, sin embargo, la posición depende de donde quiere situarse el observador, por lo que si se colocará una regla dentro del entorno virtual para que el observador se guié y se posicione de forma mas precisa, los datos obtenidos tendrían un margen menor de error. A continuación se presenta el proceso que realiza el sistema distribuido para mover un objeto (paralelepípedo) de una posición a otra el mismo que se encuentra simulado en VRML:

Paso1.- El manipulador 1 toma el objeto que se encuentra en la posición 1 (Figura 3.5).

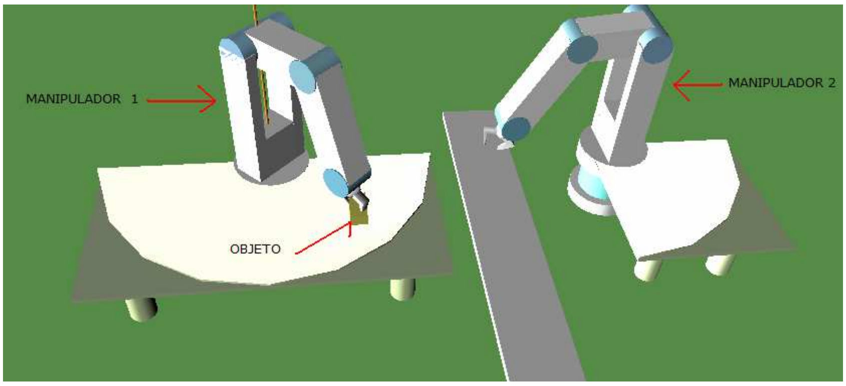


Figura 3.5. Manipulador 1 toma el objeto

Paso 2.- El manipulador 1 traslada el objeto al manipulador 2 (Figura 3.6).

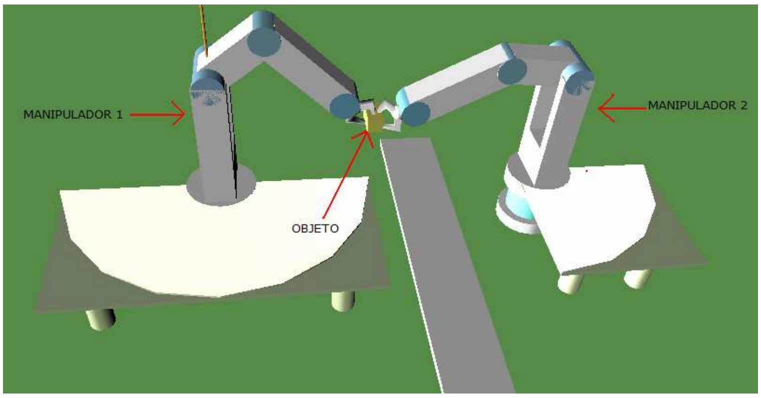


Figura 3.6. Objeto trasladado al manipulador 2

Paso 3.- El manipulador 2 ubica el objeto en la mesa de trabajo 2 (Figura 3.7).

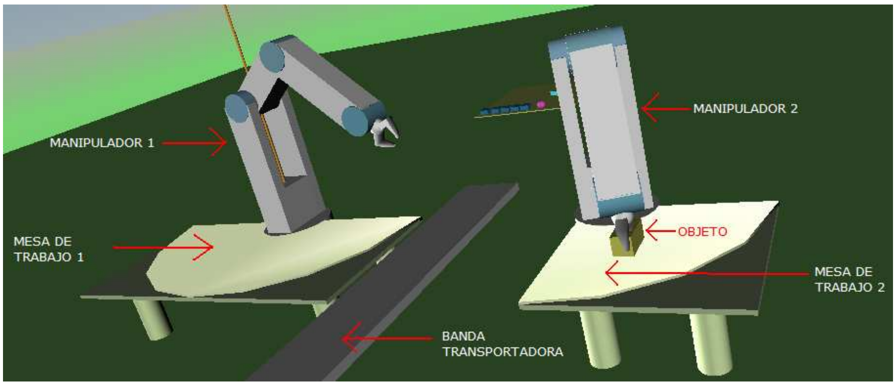


Figura 3.7. Objeto trasladado a la nueva posición

Paso 4.- Una vez que el objeto se encuentra en la nueva posición, el manipulador 2 se ubica a la espera de otro objeto (Figura 3.8).

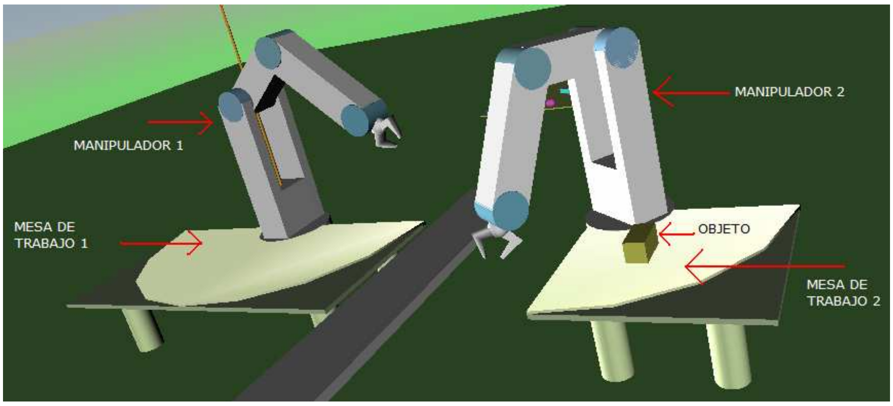


Figura 3.8. Nueva posición del manipulador 2

A continuación se presenta el posicionamiento del objeto en la banda transportadora (Figura 3.9).

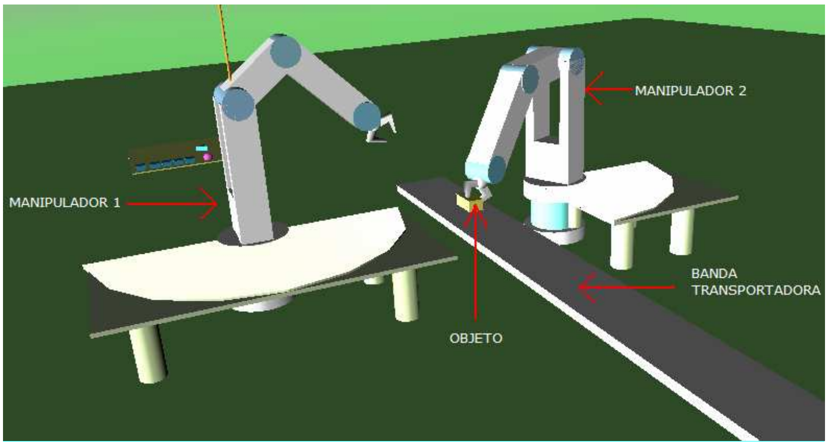


Figura 3.9. Manipulador 2 coloca el objeto en la banda transportadora

3.3 PRUEBAS DE POSICIONAMIENTO DE OBJETOS

Esta prueba permite establecer el margen de error existente entre la posición de un objeto en plano real y la ubicación de este objeto mediante el algoritmo de posicionamiento. Los datos de X y Y del plano real fueron determinados mediante una regla, mientras que los valores del algoritmo se presentan junto al objeto, como se muestra a continuación.

***<u>Posición 1</u>*.-** Cubo trasladado a la posición 1.

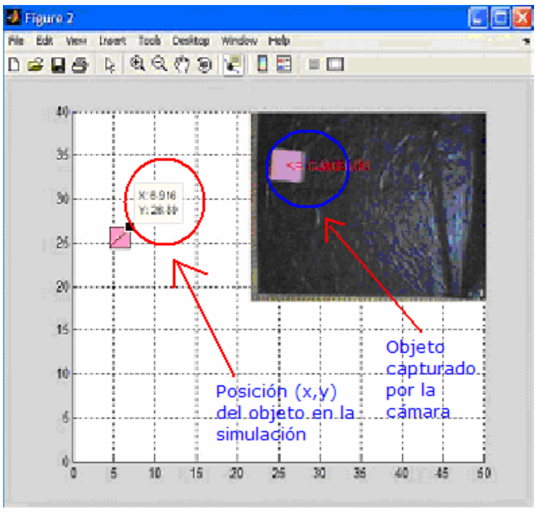


Figura 3.10. Coordenadas X y Y del algoritmo de procesamiento de imágenes

***Posición 2*.-** Cubo trasladado a la nueva posición.

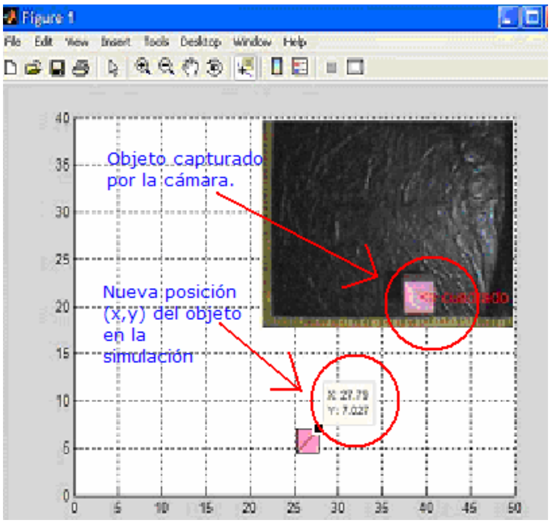


Figura 3.11. Coordenadas X y Y del algoritmo de procesamiento de imágenes

	COORDENADA X							
	X1	**X2**	**X3**	**X4**	**X5**	**X6**	**X7**	**X8**
DATO DE PROCESAMIENTO DE IMAGEN	7.55	6.82	7.92	8.85	15.62	16.92	16.82	18.9
DATO REAL	7.32	6.42	8.2	8.7	16.2	17.5	17.8	18.8
Error (%)	3.14	6.23	3.41	1.72	3.58	3.31	5.51	0.53
	X9	**X10**	**X11**	**X12**	**X13**	**X14**	**X15**	**X16**
DATO DE PROCESAMIENTO DE IMAGEN	27.79	27.98	28.09	27.61	37.04	37.61	36.38	35.44
DATO REAL	28	28	27.6	27.1	37.2	37.8	36.2	35.2
Error (%)	0.75	0.07	1.78	1.88	0.43	0.50	0.50	0.68
	COORDENADA Y							
	Y1	**Y2**	**Y3**	**Y4**	**Y5**	**Y6**	**Y7**	**Y8**
DATO DE PROCESAMIENTO DE IMAGEN	7.6	17.88	26.89	31.02	7.1	15.99	25.69	32.38
DATO REAL	8.2	17.3	25.5	29.3	7.2	15.5	24.5	30.8
Error(%)	7.32	3.35	5.45	5.87	1.39	3.16	4.86	5.13
	Y9	**Y10**	**Y11**	**Y12**	**Y13**	**Y14**	**Y15**	**Y16**
DATO DE PROCESAMIENTO DE IMAGEN	7.03	16.45	26.88	31.75	7.39	16.88	26.97	30.17
DATO REAL	7.1	16.1	25.6	30.3	7.3	15.8	25.6	29.2
Error (%)	0.99	2.17	5.00	4.79	1.23	6.84	5.35	3.32

Tabla 3.5. Posicionamiento del objeto a partir del plano real

En la Tabla 3.5, se muestran los datos X y Y, obtenidos del algoritmo de procesamiento de imágenes y de la medición realizada con una escuadra en el plano real. Se observa que el margen de error generado en la simulación, no supera el 8% en el eje Y, y en la coordenada X presenta un error no superior al 7%. También se puede observar que para valores pequeños de X los datos tienen un mayor margen de error, esto se debe a que se realiza una compensación en el eje Z, para subir o bajar las articulaciones del robot. En la Figura 3.12 se muestran cinco objetos (cilindro, paralelepípedo, triángulo, cubo, rectángulo), los mismos que contienen alturas diferentes. Empleando el algoritmo de posicionamiento de imágenes se obtuvieron las coordenadas de cada objeto mostrados en la Tabla 3.6.

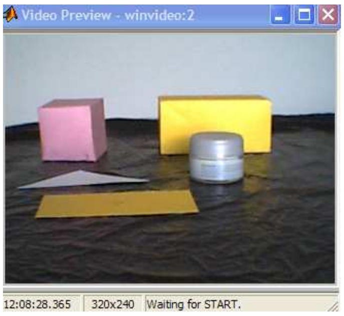


Figura 3.12. Plano real de objetos

En la Tabla 3.6, se muestran los datos X, Y y Z, obtenidos del algoritmo de procesamiento de imágenes, donde Z representa la altura del objeto tomado como referencia la mesa de trabajo (altura cero). Los datos mostrados corresponden a diferentes posiciones de los objetos indicados.

	Triángulo	Cubo	Cilindro	Rectángulo	Paralelepípedo
x	32.44	25.33	35.91	20.31	14.86
y	15.88	25.26	24.26	8.44	21.8
z	0.096	5.16	2.39	0.1	4.82
Z real	0.1	5	2.5	0.1	5
error %	4	3.2	4.4	0	3.6
	Triángulo	Cubo	Cilindro	Rectángulo	Paralelepípedo
x	29.95	16.89	18.68	31.24	21.76
y	16.86	26.6	16.67	26.04	9.88
z	0.1	4.7	2.4	0.09	4.76
Z real	0.1	5	2.5	0.1	5
error %	0	6	4	4	4.8

Tabla 3.6. Datos de la altura de los objetos

Se observa en la Tabla 3.6, que el algoritmo de visión estéreo, para determinación de la altura, no supera el error del 6%. Se debe tomar en cuenta que entre más cercanos se encuentran los objetos de la cámara, se obtendrá mejores resultados, debido a la técnica de visión estéreo empleada.

3.4 PRUEBAS DE CLASIFICACIÓN DE OBJETOS

Se probaron diferentes algoritmos de clasificación de objetos, los cuales se desglosan a continuación:

a. Algoritmo basado en la relación área-perímetro.
b. Algoritmo basado en el máximo del histograma de intensidad.
c. Algoritmo basado en la comparación de matrices.
d. Algoritmo basado en Transformada de Fourier.
e. Algoritmo basado en red MLP con el algoritmo de backpropagation (con una capa oculta).

3.4.1 ALGORITMO BASADO EN EL ÁREA Y PERÍMETRO

En la Tabla 3.7, se presentan los valores tomados de la métrica (relación entre el área y perímetro) de los objetos (Pirámide, paralelepípedo, esfera y cubo), manteniendo constante la luminosidad y orientación de las figuras.

Prueba	Pirámide	Paralelepípedo	Esfera	Cubo	Descripción
1	0.55	0.72	1.3	0.92	SIN ROTAR
2	0.69	0.75	1.1	0.96	SIN ROTAR
3	0.66	0.71	0.9	0.94	SIN ROTAR
4	0.69	0.69	0.98	0.91	SIN ROTAR
5	0.72	0.74	0.99	0.86	SIN ROTAR
6	0.67	0.76	1.0	0.87	SIN ROTAR
7	0.65	0.70	1.1	0.90	SIN ROTAR
8	0.64	0.73	1.11	0.86	SIN ROTAR
9	0.58	0.68	1.3	0.94	SIN ROTAR
10	0.60	0.71	1.25	0.92	SIN ROTAR
11	0.62	0.75	0.96	0.91	SIN ROTAR
12	0.7	0.67	0.92	0.90	SIN ROTAR
13	0.59	0.69	0.99	0.89	SIN ROTAR
14	0.58	0.72	1.1	0.94	SIN ROTAR
15	0.52	0.79	0.97	0.95	SIN ROTAR
16	0.65	0.69	0.98	0.86	SIN ROTAR
17	0.73	0.72	0.96	0.82	SIN ROTAR

Prueba	Pirámide	Paralelepípedo	Esfera	Cubo	Descripción
18	0.64	0.71	0.99	0.84	SIN ROTAR
19	0.69	0.68	0.94	0.87	SIN ROTAR
20	0.68	0.80	0.93	0.86	SIN ROTAR

Tabla 3.7. Métrica de los objetos

Bajo las condiciones de esta prueba, se esperaba que la métrica de cada objeto permanezca constante, sin embargo en la Tabla 3.7, se observa que esto no sucede, por lo cual se optó por buscar otro algoritmo de clasificación.

3.4.2 ALGORITMO BASADO EN UMBRALIZACIÓN

En la Tabla 3.8, se presenta datos del umbral de la imagen a diferentes valores de iluminación, manteniendo constante la orientación de las figuras. De los datos presentados, se observa que para una misma intensidad de luz se puede obtener diferentes umbrales, lo cual no permite generar un algoritmo robusto de clasificación, razón por lo cual se optó por seleccionar otro algoritmo más fiable.

Luz	Umbral	Luz	Umbral	Luz	Umbral	Luz	Umbral
1110	39	327	29	67	16	1416	48
1148	42	338	29	69	16	1315	48
1280	44	347	34	209	44	169	38
1291	44	350	34	220	44	156	38
1375	36	339	29	220	44	126	32
1412	37	344	34	223	46	116	28
1448	35	8	26	228	44	109	25
1466	37	8	26	239	44	100	23
1783	37	8	25	355	44	89	21
1800	37	8	23	370	44	270	34
1857	37	8	24	688	50	295	34
1916	37	70	16	563	37	300	34
350	32	68	16	480	34		
340	34	66	16	424	37		

Tabla 3.8. Valores de la Transformada de Fourier vs. Umbral

3.4.3 ALGORITMO BASADO EN MATRICES

Este algoritmo está basado en rotar la imagen un cierto número de grados e ir obteniendo por cada ángulo una matriz binaria de cada objeto, la cual se la compara (operación AND lógica) con una matriz patrón, el resultado es una matriz de error, que representa cuan diferente es un objeto respecto al patrón. Así el objeto que contenga el menor error será el seleccionado. En la Tabla 3.9 se puede observar que el menor valor es para el cubo, además se observan los errores de los demás objetos por cada ángulo de rotación.

Matriz de error para identificación del cubo									
Ángulo (grados)	Paralelepípedo	Cubo	Pirámide	Esfera	Ángulo (grados)	Paralelepípedo	Cubo	Pirámide	Esfera
1	3823	392	3077	502	46	2123	1700	1703	546
2	3761	395	3082	496	47	2178	1594	1706	534
3	3854	483	3077	502	48	2226	1706	1764	554
4	3779	573	2949	494	49	2382	1596	1767	540
5	3788	574	2947	502	50	2441	1593	1832	690
6	3720	665	2945	484	51	2547	1593	1888	544
7	3814	660	2820	546	52	2659	1594	1853	556
8	3744	758	2824	548	53	2627	1594	1847	676
9	3783	856	2694	546	54	2737	1596	1850	548
10	3636	761	2689	542	55	2795	1480	1830	554
11	3688	856	2567	546	56	2906	1486	1824	690
12	3684	955	2568	550	57	3025	1483	1883	552
13	3742	960	2442	546	58	2988	1485	1860	548
14	3684	957	2442	546	59	3107	1377	1850	554
15	3566	1060	2323	548	60	3174	1374	1908	684
16	3635	1163	2202	542	61	3131	1372	1817	684
17	3633	1057	2205	540	62	3301	1373	1832	674
18	3545	1163	2205	544	63	3313	1372	1847	542
19	3504	1266	2085	422	64	3332	1266	1842	548
20	3607	1262	2087	488	65	3440	1267	1816	554
21	3447	1373	1968	548	66	3396	1162	1837	544
22	3397	1374	1861	546	67	3522	1161	1852	548
23	3352	1265	1856	544	68	3426	1159	1838	688
24	3300	1370	1746	546	69	3549	1163	1778	682
25	3253	1373	1633	552	70	3538	1059	1797	684
26	3144	1483	1538	552	71	3676	959	1850	674
27	3098	1484	1541	548	72	3574	956	1834	682
28	3002	1483	1421	548	73	3699	858	1745	542
29	2951	1486	1320	558	74	3652	858	1759	546
30	2951	1482	1332	548	75	3677	857	1809	542
31	2799	1588	1321	544	76	3746	862	1820	686
32	2755	1593	1319	682	77	3783	669	1869	682
33	2656	1596	1379	542	78	3697	666	1813	682
34	2567	1591	1434	540	79	3743	666	1823	680
35	2472	1708	1438	550	80	3820	571	1872	548
36	2426	1709	1489	540	81	3795	575	1918	678
37	2334	1704	1551	542	82	3789	571	1923	680
38	2255	1704	1597	550	83	3771	394	1931	548
39	2122	1592	1606	544	84	3834	392	1940	550
40	2078	1596	1619	544	85	3823	306	1985	680

Ángulo (grados)	Paralelepípedo	Cubo	Pirámide	Esfera	Ángulo (grados)	Paralelepípedo	Cubo	Pirámide	Esfera
41	1954	1704	1621	552	86	3745	304	1953	548
42	1964	1705	1672	544	87	3732	270	1993	542
43	1922	1707	1734	552	88	3898	267	2038	550
44	1925	1592	1642	538	89	3979	328	2038	540
45	2026	1707	1699	550	90	3981	333	1988	546
Error mínimo 267									

Tabla 3.9. Matriz de error para seleccionar el cubo

De los datos de la Tabla 3.9, se observa que el menor error se encuentra en la columna del cubo, con lo cual se establece que éste es el objeto más parecido al patrón. Como se trabajo para el cubo se procede para cada objeto dentro de la imagen. En el caso específico de la esfera no es necesaria su rotación, sin embargo se la rotó hasta 10º para comprobar la robustez del algoritmo, estos resultados se muestran en la Tabla 3.10.

Matriz de error para identificación de la esfera				
Ángulo	**Paralelepípedo**	**Cubo**	**Pirámide**	**Esfera**
1	2531	356	2165	65
2	2623	363	2178	58
3	2736	379	2181	65
4	2737	415	2133	61
5	2730	400	2143	65
6	2724	441	2147	54
7	2792	428	2048	75
8	2780	468	2060	72
9	2867	512	1956	69
10	2782	463	1967	71
Error mínimo 54				

Tabla 3.10. Matriz de error para seleccionar la esfera

Se observa en Tabla 3.10, que el error mínimo corresponde a la columna de la esfera. Y de igual forma se procede con el resto de figuras.

Matriz de error para identificación del Paralelepípedo									
ángulo	**Paralelepípedo**	**Cubo**	**Pirámide**	**Esfera**	**ángulo**	**Paralelepípedo**	**Cubo**	**Pirámide**	**Esfera**
1	3488	1094	2475	547	37	274	1865	1671	568
2	3188	1095	2470	550	38	240	1855	1645	588
3	3140	1174	2473	549	39	189	1810	1618	547
4	3047	1225	2436	537	40	186	1794	1518	573
5	2994	1200	2438	533	41	**148**	1901	1492	535
6	2902	1279	2438	526	42	158	1882	1460	565
7	2804	1254	2402	600	43	159	1858	1378	535

ángulo	Paralelepípedo	Cubo	Pirámide	Esfera	ángulo	Paralelepípedo	Cubo	Pirámide	Esfera
8	2704	1304	2402	601	44	162	1806	1196	554
9	2606	1377	2352	596	45	220	1912	1158	530
10	2400	1325	2355	588	46	233	1865	1122	560
11	2303	1399	2317	588	47	246	1802	1095	528
12	2271	1437	2314	584	48	314	1885	1014	566
13	2136	1400	2271	586	49	398	1812	985	539
14	2045	1437	2263	584	50	505	1767	946	561
15	1892	1506	2227	583	51	625	1811	928	549
16	1768	1537	2190	586	52	742	1766	880	523
17	1744	1499	2177	585	53	731	1800	858	550
18	1522	1529	2219	593	54	842	1752	845	545
19	1454	1585	2185	515	55	951	1708	733	530
20	1431	1619	2169	520	56	1080	1740	713	561
21	1237	1639	2129	541	57	1156	1681	707	551
22	1182	1664	2093	544	58	1288	1709	692	535
23	1054	1614	2088	551	59	1260	1615	670	522
24	1003	1626	2040	556	60	1376	1638	672	562
25	957	1647	2003	569	61	1516	1656	599	550
26	851	1697	1961	579	62	1592	1593	598	539
27	812	1706	1952	587	63	1600	1610	605	544
28	688	1723	1919	537	64	1680	1551	590	537
29	658	1736	1882	550	65	1812	1562	562	532
30	640	1748	1864	563	66	1959	1462	565	529
31	535	1742	1851	583	67	2020	1467	576	521
32	519	1747	1841	581	68	2032	1477	558	572
33	438	1750	1781	548	69	2092	1483	538	571
34	395	1751	1758	565	70	2241	1411	551	568
35	324	1871	1734	586	71	2306	1383	594	573
36	316	1870	1715	545	72	2310	1384	609	573
73	2374	1313	555	530	107	4024	934	1461	549
74	2532	1309	569	536	108	4047	944	1488	534
75	2624	1306	619	544	109	4063	1008	1537	517
76	2760	1303	630	537	110	4120	1068	1564	589
77	2860	1201	644	545	111	4012	965	1601	560
78	2805	1188	588	557	112	4018	1028	1618	547
79	2913	1176	598	570	113	4035	1132	1665	535
80	3034	1145	612	533	114	4043	1021	1697	527
81	3181	1133	627	537	115	4051	1081	1716	518
82	3121	1113	630	554	116	3927	1117	1682	491
83	3210	1012	644	523	117	3939	1179	1725	484
84	3306	992	651	540	118	3937	1107	1770	484
85	3436	992	669	546	119	3946	1173	1810	478
86	3345	968	671	521	120	3951	1237	1779	549
87	3430	934	693	544	121	3826	1126	1839	549
88	3528	907	722	506	122	3820	1188	1823	511
89	3618	844	716	529	123	3826	1226	1894	545
90	3614	745	708	526	124	3778	1177	1961	544
91	3610	757	723	525	125	3776	1214	1945	542
92	3679	818	723	550	126	3770	1288	2008	544
93	3762	788	767	520	127	3766	1208	2027	541

ángulo	Paralelepípedo	Cubo	Pirámide	Esfera	ángulo	Paralelepípedo	Cubo	Pirámide	Esfera
94	3802	819	774	553	128	3714	1273	2093	541
95	3747	783	841	520	129	3573	1308	2081	554
96	3778	848	853	558	130	3568	1265	2093	473
97	3849	809	862	526	131	3519	1302	2093	477
98	3885	843	962	567	132	3624	1257	2161	479
99	3964	809	1010	540	133	3613	1301	2186	483
100	3851	851	1042	578	134	3600	1376	2181	491
101	3927	812	1092	556	135	3702	1306	2212	510
102	3953	846	1242	516	136	3695	1354	2203	523
103	4021	916	1311	578	137	3799	1309	2201	531
104	4044	873	1348	538	138	3793	1368	2328	535
105	3936	908	1371	524	139	3897	1315	2324	471
106	4007	943	1432	586	140	3874	1367	2362	439
Error Mínimo 148									

Tabla 3.11. Matriz de error para seleccionar el paralelepípedo

El error mínimo de la Tabla 3.11, pertenece a la columna del paralelepípedo. A continuación se presenta la Tabla 3.12, en la cual se muestra que el valor mínimo corresponde a la pirámide.

Matriz de error para la Identificación de la Pirámide									
ángulo	Paralelepípedo	Cubo	Pirámide	Esfera	ángulo	Paralelepípedo	Cubo	Pirámide	Esfera
1	3652	746	1306	565	23	4390	1207	400	516
2	3915	691	1311	544	24	4309	1223	319	515
3	4078	711	1292	563	25	4220	1248	236	514
4	4043	730	1215	549	26	4170	1258	196	468
5	4126	775	1199	573	27	4084	1291	207	464
6	4087	802	1189	556	28	4015	1262	157	502
7	4249	851	1114	545	29	3923	1281	155	495
8	4211	884	1098	518	30	3911	1313	215	494
9	4329	924	1025	539	31	3780	1340	284	480
10	4328	865	1006	517	32	3692	1377	342	504
11	4441	908	968	533	33	3608	1348	588	505
12	4435	958	949	505	34	3497	1381	685	490
13	4533	1017	872	523	35	3413	1364	729	477
14	4478	1026	852	489	36	2426	1709	1489	270
15	4458	1011	775	500	37	2334	1704	1551	271
16	4538	1071	693	513	38	2255	1704	1597	275
17	4534	1086	676	482	39	2122	1592	1606	272
18	4513	1081	622	490	40	2078	1596	1619	272
19	4444	1148	574	509	41	1954	1704	1621	276
20	4569	1162	552	546	42	1964	1705	1672	272
21	4456	1172	470	504	43	1922	1707	1734	276
22	4377	1189	395	509	44	1925	1592	1642	269
Error Mínimo 155									

Tabla 3.12. Matriz de error para seleccionar la Pirámide

En la Figura 3.13 se observa la rotación de 45 grados de la imagen, obteniendo la matriz de error para la identificación de la pirámide, para el método de comparación de matrices.

Figura 3.13. Rotación de 45 grados de la figura

3.4.4 ALGORITMO BASADO EN TRANSFORMADA DE FOURIER

En la Tabla 3.13, se presentan la Transformada de Fourier de los objetos (Pirámide, paralelepípedo, esfera y cubo), manteniendo constante la luminosidad así como la orientación de las figuras.

NUM	Luz(Luxes)	Pirámide	Paralelepípedo	Esfera	Cubo	Descripción
1	Fuera escala	986	2839	1335	1040	orientación constante
2	Fuera escala	989	2839	1332	1036	orientación constante
3	Fuera escala	986	2838	1333	1034	orientación constante
4	Fuera escala	993	2846	1340	1048	orientación constante
5	Fuera escala	988	2840	1336	1042	orientación constante
6	Fuera escala	965	2825	1320	1029	orientación constante
7	Fuera escala	958	2812	1308	1019	orientación constante
8	Fuera escala	957	2813	1308	1018	orientación constante
9	Fuera escala	972	2834	1343	1037	orientación constante
Fuera de escala: Valor de luminosidad superior a 2000 luxes.						

Tabla 3.13. Objetos con orientación constante

De la Tabla 3.13, se puede concluir que la transformada de Fourier es un excelente método de clasificación para el presente trabajo, siempre y cuando se mantenga un nivel de iluminación optimo. En la Tabla 3.14, se presentan las Transformadas de Fourier de los objetos, cambiando la orientación de la pirámide y manteniendo constante la luminosidad como la orientación del resto de figuras (cubo, paralelepípedo y esfera).

NUM	Luz(Luxes)	Pirámide	Paralelepípedo	Esfera	Cubo	Descripción
1	Fuera escala	990	2818	1312	1028	gira 90 Pirámide
2	Fuera escala	819	2852	1360	1062	gira 90 Pirámide
3	Fuera escala	754	2804	1305	1031	gira 90 Pirámide
4	Fuera escala	771	2818	1317	1035	gira 90 Pirámide
5	Fuera escala	755	2811	1310	1034	gira 90 Pirámide
6	Fuera escala	749	2803	1307	1031	gira 90 Pirámide
10	Fuera escala	990	2818	1312	1028	gira 90 Pirámide
11	Fuera escala	819	2852	1360	1062	gira 90 Pirámide
12	Fuera escala	754	2804	1305	1031	gira 90 Pirámide
13	Fuera escala	771	2818	1317	1035	gira 90 Pirámide
14	Fuera escala	755	2811	1310	1034	gira 90 Pirámide
15	Fuera escala	749	2803	1307	1031	gira 90 Pirámide
16	Fuera escala	911	2830	1325	1043	gira 135 Pirámide
17	Fuera escala	904	2823	1319	1040	gira 135 Pirámide
18	Fuera escala	903	2822	1316	1040	gira 135 Pirámide
19	1110	904	2818	1319	1038	gira 135 Pirámide
20	1148	906	2825	1321	1041	gira 270 Pirámide
21	1280	926	2838	1336	1053	gira 270 Pirámide
22	1291	878	2855	1339	1061	gira 315 Pirámide

Tabla 3.14. Transformada de Fourier con rotación de la pirámide

En la Tabla 3.15, se presentan la Transformada de Fourier variando aleatoria mente la intensidad de luz y la orientación de los objetos. Esta prueba fue realizada para verificar cuanto cambio los valores de la trasformada ante cambios de luz y rotación del objeto. De lo observado se puede concluir que la transformada varía considerablemente con la intensidad de luz, pero mantiene valores aproximados con respecto a la rotación del objeto.

NUM	Luz(candela)	Pirámide	Paralelepípedo	Esfera	Cubo	Descripción
1	c140	819	2983	1386	1035	giros aleatorios de la figuras
2	1416	892	2694	1388	1059	giros aleatorios de la figuras
3	1315	803	2736	1382	1077	giros aleatorios de la figuras
4	169	758	2672	1338	1044	giros aleatorios de la figuras
5	156	750	2664	1333	1036	giros aleatorios de la figuras
6	126	663	2711	1288	1034	giros aleatorios de la figuras
7	116	772	2695	1338	1009	giros aleatorios de la figuras
8	109	761	2675	1329	998	giros aleatorios de la figuras
9	100	619	2635	1292	969	giros aleatorios de la figuras
10	89	599	2565	1250	935	giros aleatorios de la figuras
11	270	793	3267	1437	978	giros aleatorios de la figuras
12	295	662	2652	1261	936	giros aleatorios de la figuras
13	300	729	2584	1175	917	giros aleatorios de la figuras

Tabla 3.15. Transformada de Fourier cambiando aleatoriamente la orientación y luminosidad de los objetos

En la Tabla 3.16, se presentan la Transformada de Fourier de los objetos, cambiando la orientación del paralelepípedo y manteniendo constante la luminosidad así como la orientación del resto de figuras (cubo, pirámide y esfera). Sin embargo las variaciones de luz mostradas no fueron controladas, esta prueba se realizó con luz natural (sol).

NUM	Luz(Luxes)	Pirámide	Paralelepípedo	Esfera	Cubo	Descripción
1	1375	833	2670	1307	1037	gira 45º Paralelepípedo
2	1412	834	2674	1310	1036	gira 45º Paralelepípedo
3	1448	765	2633	1308	1036	gira 90º Paralelepípedo
4	1466	770	2640	1310	1037	gira 90º Paralelepípedo
5	1783	864	2720	1313	1040	gira 135º Paralelepípedo
6	1800	870	2730	1317	1039	gira 135º Paralelepípedo
7	1857	865	2899	1325	1041	gira 90 Paralelepípedo
8	1916	868	2905	1329	1048	gira 90 Paralelepípedo

Tabla 3.16. Transformada de Fourier con rotación del Paralelepípedo

En la Tabla 3.17, se presentan la Transformada de Fourier de los objetos, cambiando la orientación del cubo y manteniendo constante la luminosidad así como la orientación del resto de figuras (paralelepípedo, pirámide y esfera).

NUM	Luz(candela)	Pirámide	Paralelepípedo	Esfera	Cubo	Descripción
1	c205	867	2913	1339	1050	gira cubo 45
2	c207	884	2928	1359	1063	gira cubo 45
3	c216	870	2910	1358	1044	gira cubo 90
4	c219	881	2919	1362	1046	gira cubo 90

Tabla 3.17. Transformada de Fourier con rotación del cubo

En la Tabla 3.18, se presentan los valores tomados de la Transformada de Fourier de los objetos, cambiando la cantidad de luminosidad y manteniendo constante la orientación de todos los objetos.

NUM	Luz(candela)	Pirámide	Paralelepípedo	Esfera	Cubo	Descripción
1	350	610	2545	1248	964	CAMBIANDO LA CANTIDAD DE LUZ
2	340	606	2546	1246	963	CAMBIANDO LA CANTIDAD DE LUZ
3	327	599	2526	1229	948	CAMBIANDO LA CANTIDAD DE LUZ
4	338	603	2542	1246	957	CAMBIANDO LA CANTIDAD DE LUZ
5	347	607	2545	1247	965	CAMBIANDO LA CANTIDAD DE LUZ
6	350	607	2548	1247	961	CAMBIANDO LA CANTIDAD DE LUZ
7	339	605	2536	1241	955	CAMBIANDO LA CANTIDAD DE LUZ
8	344	607	2536	1240	960	CAMBIANDO LA CANTIDAD DE LUZ
9	8	489	2515	1215	888	CAMBIANDO LA CANTIDAD DE LUZ
10	8	479	2503	1209	889	CAMBIANDO LA CANTIDAD DE LUZ
11	8	441	2488	1191	879	CAMBIANDO LA CANTIDAD DE LUZ
12	8	520	2537	1225	897	CAMBIANDO LA CANTIDAD DE LUZ
13	8	548	2560	1238	919	CAMBIANDO LA CANTIDAD DE LUZ
14	70	563	2554	1206	900	CAMBIANDO LA CANTIDAD DE LUZ
15	68	569	2536	1188	905	CAMBIANDO LA CANTIDAD DE LUZ
16	66	563	2516	1182	908	CAMBIANDO LA CANTIDAD DE LUZ
17	67	561	2517	1190	908	CAMBIANDO LA CANTIDAD DE LUZ
18	69	563	2533	1200	912	CAMBIANDO LA CANTIDAD DE LUZ
19	209	871	2910	1366	1058	CAMBIANDO LA CANTIDAD DE LUZ
20	220	878	2920	1372	1059	CAMBIANDO LA CANTIDAD DE LUZ
21	220	879	2922	1371	1064	CAMBIANDO LA CANTIDAD DE LUZ
22	223	880	2928	1375	1068	CAMBIANDO LA CANTIDAD DE LUZ
23	228	871	2910	1366	1057	CAMBIANDO LA CANTIDAD DE LUZ
24	239	879	2923	1371	1065	CAMBIANDO LA CANTIDAD DE LUZ
25	355	868	2905	1362	1056	CAMBIANDO LA CANTIDAD DE LUZ
26	370	873	2913	1366	1059	CAMBIANDO LA CANTIDAD DE LUZ
27	688	880	2936	1379	1072	CAMBIANDO LA CANTIDAD DE LUZ
28	563	848	2883	1351	1037	CAMBIANDO LA CANTIDAD DE LUZ
29	480	843	2880	1347	1035	CAMBIANDO LA CANTIDAD DE LUZ
30	424	841	2876	1347	1033	CAMBIANDO LA CANTIDAD DE LUZ
31	c262	899	2948	1381	1061	CAMBIANDO LA CANTIDAD DE LUZ
32	c260	900	2954	1383	1056	CAMBIANDO LA CANTIDAD DE LUZ
33	c246	892	2938	1378	1050	CAMBIANDO LA CANTIDAD DE LUZ
34	c235	888	2926	1372	1043	CAMBIANDO LA CANTIDAD DE LUZ
35	c215	879	2915	1358	1037	CAMBIANDO LA CANTIDAD DE LUZ
36	c205	892	2938	1374	1044	CAMBIANDO LA CANTIDAD DE LUZ

NUM	Luz(candela)	Pirámide	Paralelepípedo	Esfera	Cubo	Descripción
37	c195	885	2926	1365	1035	CAMBIANDO LA CANTIDAD DE LUZ
38	c210	895	2940	1374	1041	CAMBIANDO LA CANTIDAD DE LUZ
39	c192	888	2923	1362	1035	CAMBIANDO LA CANTIDAD DE LUZ
40	c178	891	2927	1365	1036	CAMBIANDO LA CANTIDAD DE LUZ
41	c167	895	2937	1371	1039	CAMBIANDO LA CANTIDAD DE LUZ

Tabla 3.18. Transformada de Fourier variando la intensidad de luz

De los valores mostrados en las Tablas 3.13 a la 3.18, se observa que la Transformada de Fourier da excelentes resultados para este trabajo, a niveles de luz constante o muy buena iluminación, sin embargo para entornos con poca o ninguna luminosidad presenta grandes variaciones en éstos descriptores. Por lo cual se opto por buscar otro algoritmo de clasificación.

3.4.5 ALGORITMO BASADO EN TRANSFORMADA DE FOURIER Y REDES NEURONALES

En la siguiente tabla se presentan la Transformada de Fourier obtenidos por el algoritmo backpropagation, para diferentes objetos:

	Cubo									
Ángulo	**Fourier 1**	**Fourier 2**	**Fourier 3**	**Fourier 4**	**Fourier 5**	**Fourier 6**	**Fourier 7**	**Fourier 8**	**Fourier 9**	**Fourier 10**
0	1307	1305,45	1312,09	1312,6	1312,62	1307,81	1305,83	1316,9	1307,93	1310,78
45	1303,18	1300,37	1300,79	1302,58	1302,38	1293,77	1303,63	1303,61	1296,25	1300,02
90	1316,01	1317,04	1315	1311,75	1318,77	1305,6	1318,16	1317,11	1310,98	1314,45
	Paralelepípedo									
Ángulo	**Fourier 1**	**Fourier 2**	**Fourier 3**	**Fourier 4**	**Fourier 5**	**Fourier 6**	**Fourier 7**	**Fourier 8**	**Fourier 9**	**Fourier 10**
0	3063,66	31,24,69	3096,06	3101,58	3158,75	3059,64	3079,56	3100,89	3069,61	3062,8
45	3096,29	3067,38	3076,15	3119	3098,25	3074,59	3094,22	3087,48	3078,81	3110,5
90	3097,39	3088,54	3121,82	3097,7	3102,24	3096,62	3100	3105,48	3115,22	3110,2
	Pirámide									
Ángulo	**Fourier 1**	**Fourier 2**	**Fourier 3**	**Fourier 4**	**Fourier 5**	**Fourier 6**	**Fourier 7**	**Fourier 8**	**Fourier 9**	**Fourier 10**
0	881,76	892,85	892,17	863,43	896,83	883,81	862,04	883,95	864,07	851,72
45	858,43	869,51	906,53	880,75	891,8	842,48	878,25	877,79	862,42	871,97
90	851,78	842,66	889,26	865,45	887,77	856,72	860,93	837,6	857,1	871,1
	Esfera									
Ángulo	**Fourier 1**	**Fourier 2**	**Fourier 3**	**Fourier 4**	**Fourier 5**	**Fourier 6**	**Fourier 7**	**Fourier 8**	**Fourier 9**	**Fourier 10**
0	1471,7	1475,16	1468,21	1464,75	1471,45	1460,31	1458,02	1471,18	1478,06	1468,46

Tabla 3.19. Transformada de Fourier obtenidos con algoritmo de Backpropagation

En base a los resultados mostrados en la Tabla 3.19, se observa que los valores obtenidos por la red neuronal a diferentes ángulos de rotación y a variaciones de luz, presentan mejores resultados que el algoritmo de Área-perímetro y el de umbral optimo de brillo. El algoritmo de comparación de matrices presta también excelentes resultados, sin embrago la carga computacional que este requiere es muy alta, por lo cual se decidió trabajar mediante MRL con algoritmo de entrenamiento backpropagation.

También se realizaron pruebas de comunicación entre los dos computadores, la misma que no presento errores en transferencia de archivos. Mediante esta prueba se pretende comprobar el funcionamiento de la simulación en el Host remoto, se enviaron datos tales como: posicionamiento del objeto, orientación del objeto, tipo de objeto, nueva posición del objeto.

En base a las pruebas efectuadas se puede concluir que el sistema es confiable, y ha cumplido con los alcances establecidos en este trabajo.

CAPÍTULO 4

CONCLUSIONES Y RECOMENDACIONES

CAPÍTULO 4
CONCLUSIONES Y RECOMENDACIONES

En este Capítulo se obtienen las conclusiones finales extraídas del análisis de los resultados de las pruebas. También se enuncian recomendaciones luego de la experiencia adquirida al realizar este proyecto.

4.1 CONCLUSIONES

- La simulación del sistema distribuido robotizado, se la realizó en Matlab (The Mathworks); en archivo.m y en el toolbox VRML, obteniéndose excelentes resultados en este último.

- EL desarrollo de la simulación en el archivo.m, involucra una mayor complejidad para su programación, implicando una alta carga computacional, lo cual limitó el tiempo de ejecución de la simulación.

- El toolbox de Realidad Virtual de Matlab resultó ser una herramienta poderosa, sin complejidad de programación y con buen tiempo de ejecución.

- Un proceso clave para que el procesamiento de imágenes no sea tan complicado, es la obtención de una imagen de buena calidad, para los fines que convenga a la aplicación.

- Para el posicionamiento del manipulador se consideró la cinemática inversa, que determina que valores tiene que tomar las variables articulares para que el extremo del robot se encuentre en una posición y orientación dada. Los resultados obtenidos probaron que este método genero buenos resultados.

- Para el presente trabajo se aplicó el modelo de ejes alineados para la determinación de la distancia de un objeto, teniendo en cuenta las características físicas de la cámara para la calibración de la distancia en un plano real.

- Como se puede observar, en base a las técnicas de procesamiento de imágenes se ha podido identificar y posicionar un objeto en tres dimensiones dentro de un plano real.

- La implementación del sistema distribuido permitió compartir información entre los dos puestos de trabajo desarrollados, observándose que al quedar deshabilitado uno de los puestos de trabajo el otro continúo operando, lo cual constituye una ventaja muy importante, en especial en el sector industrial.

- Para realizar la clasificación de objetos, en este trabajo se implementó varios algoritmos; área-perímetro, umbral óptimo, comparación de matrices, transformada de Fourier y backpropagation. Siendo este último el que presento mejores resultados.

- El desarrollo de la simulación en 3D ha sido de gran interés, debido a que para poder programar un robot industrial normalmente es necesario el sistema físico y el software de programación del mismo, que normalmente son de un costo no muy accesible, especialmente para instituciones de educación pública media y superior.

4.2 RECOMENDACIONES

- Para que la simulación de cualquier entorno virtual se aproxime a la realidad, se recomienda obtener un buen modelo matemático e incluir la mayor cantidad de variables posibles.

- En la obtención de la imagen para esta aplicación, se recomienda que el contraste entre el objeto y el fondo sea muy bueno. Además se debe iluminar el ambiente de modo que se evite sombras sobre cada uno de los objetos.

- Se recomienda profundizar en el estudio de este trabajo para dar soluciones específicas a los diferentes problemas que se presentan en el sector industrial.

- Sería de gran importancia extender el trabajo realizado hacia un análisis cinemática y dinámico, y realizar otras aplicaciones tales como seguimiento de trayectorias mínimas.

- Otra área en la que podría extenderse el trabajo es en el procesamiento de imágenes en tiempo real, empleando herramientas como OpenCV y sistemas operativos que permitan este tipo de control.

BIBLIOGRAFÍA

[1] F.Torres, J.Pomares, P.Gil y S.Puente, “Robots y Sistemas Sensoriales”, Madrid, 2002.

[2] A.Barrientos, L.F.Peñín, C.Balaguer y R.Aracil, “Fundamentos de Robótica”, 2ªed, Madrid, McGraw-Hill, 2007.

[3] A.O.Baturone, “Robótica Manipuladores y robots móviles”, Barcelona, Marcombo, 2007.

[4] J.J.Craig, “Robótica”, 3ªed, México, Pearson Educación, 2006.

[5] B.M.Brío y A.S.Molina, “Redes Neuronales y Sistemas Difusos”, 2ª ed. Madrid, Alfaomega, 2002.

[6] J.R.Hilera y V.J.Martínez, “Redes Neuronales Artificiales: Fundamentos Modelos y Aplicaciones”, Madrid, Alfaomega, 2000.

[7] D.Karnopp, D.Margolis and R.Rosenberg, “System Dynamics:Modeling and Simulation of Mechatronic Systems”, 3ªed, 2005.

[8] A. Zilouchian and M.Jamshidi, “Intelligent Control Systems Using Soft Computing Methodologies”,United States 2001.

[9] N.C.Braga, “Robotics, Mechatronics, and Artificial Intelligence: Experimental Circuit Blocks for Designers”, Newnes,2001.

[10] E.G.Bonilla, “Reconocimiento de Caracteres Mediante Redes Neuronales con Matlab”, Tesis (Ingeniero en Electrónica y Control), Quito, Ecuador, Escuela Politécnica Nacional, 2005.

[11] M.A.Hidalgo y G.F.Tene, “Sistema Inteligente para el Control de un Brazo Robótica Utilizando Señales Electroniograma”, Tesis ((Ingeniero en Electrónica y Control), Quito, Ecuador, Escuela Politécnica Nacional, 2005.

[12] H.P.Sanchez y M.J.Villacres, “Diseño y Construcción y control de un Brazo Articulado”, Tesis ((Ingeniero en Electrónica y Control), Quito, Ecuador, Escuela Politécnica Nacional, 1999.

[13] G.J.García, “Control Visual de Robots Manipuladores Aplicaciones a Entornos industriales”, Alicante, 2004.

[14] N.C.Braga, “Robotics, Mechatronics, and Artificial Intelligence: Experimental Circuit Blocks for Designers”, Newnes,2001.

[15] J. A.Soto, J.E.Vargas y J.C.Pedraza, “Generación de trayectorias para un robot manipulador utilizando procesamiento de imágenes y splines”, 2005.

[16] T.Dong, R.Tong, L.Zhang and J.Dong, “Approach to Cooperative Assembly Task Planning for Multiple Manipulators”, IEEE, 2003.

[17] M.Ferch, M.Hochsmann and J.Zhang,“Learning cooperative assembly with the graph representation of a state=action space”, IEEE, 2002.

[18] F.Martín, R.Gonzalez, P.Barrera, J.Cañas and Matellán, “Visual Based Localization for a Legged Robot”, IEEE, 2004.

[19] W.Sandoval y I.Salcedo, “Matlab en aplicaciones de robótica móvil”, 2005.

[20] Grupo de Robotica de la Facultad de Ciencias de la Electrónica BUAP, “Interfaz del Brazo Robot Puma 200con la PC”, 2006.

[21] M. Donado, D.García, A.Gonzalez, “Sistema Localizador de Objetos Basados en Visión Esteréo”. 2004.

[22] E.Guillermo, J.Larriva,J.Trelles y O.Vele, “Diseño, Construcción y Prueba de un Robot Humanoide”, 2002.

[23] A.Barrientos, “Uso de Matlab para el Análisis Cinemático de un Robot”, 2004.

[24] J.R.Mendoza, “Diseño del control de un robot de dos grados de libertad para aplicaciones de seguimientos de objetos”, Tonantzintia, 2003.

[25] W.E.Sanz, “Plataforma Virtual Interactiva para la modelación y simulación de robots bajo el ambiente de programación de Matlab”, Caracas, 2003.

[26] Jornadas de Control Visual,”Robo Tenis: Diseño, Simulación, Análisis Cinemática y Dinámico de un robot paralelo para Control Visual de altas prestaciones”, Alicante-Elche, 2003.

[27] http://ohm.utp.edu.co/paginas/docencia/neuronales/Capitulo2

[28] http://www.jvuletich.org/Squeak/PhotoSqueak/PhotoSqueak1

[29] http://neuralnets.web.cern.ch/NeuronalNets/nnwlnHepHard.html#sand

[30] www.INTELITEK.com