UNIVERSIDAD NACIONAL DE INGENIERÍA
FACULTAD DE INGENIERÍA MECÁNICA
ESPECIALIDAD DE INGENIERÍA MECATRÓNICA

ALGORITMOS DE APRENDIZAJE EN VISIÓN ARTIFICIAL PARA LA CAPTURA DE OBJETOS EN UN ESPACIO TRIDIMENSIONAL

TESIS
PARA OPTAR EL TÍTULO PROFESIONAL DE
INGENIERO MECATRÓNICO

KEVIN GIANFRANCO MAZA RIVERA

PROMOCIÓN 2009-II
LIMA - PERÚ
2011

CONTENIDO

CAPÍTULO III

CAPÍTULO IV

Índice de figuras

Índice de tablas

PRÓLOGO

La presente tesis aborda el tema de Robot Grasping desde un enfoque de la Inteligencia Artificial y la Visión por Computador. Este trabajo se basa en las características de la imagen de un objeto para inferir su punto de agarre sin realizar una reconstrucción tridimensional total de este, experimentando con tres algoritmos de aprendizaje y realizando pruebas de reconstrucción tridimensional.

El interés en desarrollar este tema surgió del deseo de dotar a un brazo robótico de la autonomía suficiente para que manipule objetos en un ambiente tridimensional. Para pensar en cómo desarrollar esta autonomía, como en muchos de los trabajos realizados en robótica, se observó primeramente cómo los humanos realizamos esta actividad.

Los algoritmos de aprendizaje entrenados durante el desarrollo de la presente tesis clasifican las características bidimensionales de las imágenes de objetos de uso doméstico con la finalidad de inferir un punto apropiado para su manipulación. Estos se integran a otros algoritmos que calculan las coordenadas tridimensionales de dicho punto.

Las experimentaciones de los algoritmos de aprendizaje y de obtención de coordenadas tridimensionales se realizan con objetos reales y en un espacio de trabajo real para demostrar que su aplicación es viable y útil en el control de cualquier brazo robótico.

En el primer capítulo I se introducirá el tema, se presentará la justificación, planteamiento del problema, los objetivos, así como también los alcances y las limitaciones del trabajo.

En el capítulo II se expondrán los fundamentos principales de la Visión por Computador, empezando por los fundamentos más básicos y luego explicando aquellos que se usaron en el desarrollo del tema. Las ecuaciones en este capítulo se presentarán de una forma general ya que son las bases sobre las cuales se hace el desarrollo del tema.

El capítulo III empieza describiendo los fundamentos de los algoritmos de aprendizaje usados, así como también los algoritmos de entrenamiento y de obtención de coordenadas tridimensionales. Las ecuaciones y los cálculos que se muestran en este capítulo son lo más cercano a lo desarrollado en este tema. Se presentarán también unos ejemplos de los algoritmos de reconstrucción tridimensional para facilitar su comprensión.

En el último capítulo se muestra un brazo robótico a manera de ejemplo para explicar su integración con el sistema de visión a desarrollar. Después, se expondrán los resultados obtenidos del entrenamiento y aplicación de los algoritmos de aprendizaje al problema, así como también los resultados de la reconstrucción tridimensional.

En la parte final se expondrán las conclusiones a las que se llegó después de desarrollar la tesis. En el apéndice se han colocado los programas que se usaron para las etapas de entrenamiento, evaluación, calibración de cámaras y reconstrucción tridimensional.

CAPÍTULO I

INTRODUCCIÓN

1.1. Antecedentes

La actividad de coger y manipular objetos es compleja y especializada debido a que ha sido perfeccionada por mucho tiempo como parte de nuestro aprendizaje y evolución. Es una actividad que nos diferencia de otras especies animales aunque hay algunos tipos de primates que también han desarrollado esta habilidad como los chimpancés, macacos, etc. Como consecuencia en la actualidad la mayoría de objetos con los que interactuamos están diseñados para ser manipulados ergonómicamente.

Trabajos como *The Neuroscience of Grasping* de Castiello U. [1] han identificado las principales áreas cognitivas que se activan para realizar un agarre basándose en el estudio del comportamiento de humanos y primates. Además se han realizado diversas pruebas experimentales para analizar cómo los humanos cogemos un objeto, las cuales han sido aplicadas y comparadas en otras especies como los macacos para encontrar qué áreas cerebrales se ponen en funcionamiento durante esta actividad, y para saber si estos usan las mismas regiones del cerebro que los humanos o si cada especie agarra al mismo objeto desde un punto distinto.

El punto del cual es cogido un objeto y la configuración de la mano para realizar la captura depende básicamente de las propiedades físicas de este (la

forma, el tamaño, la textura, la fragilidad y el peso) y de una apropiada postura para realizar el agarre. Esto puede hacer la diferencia en coger un objeto con toda la palma de la mano, solo con los dedos o con poca presión ejercida. Además puede existir más de una posibilidad de agarre para un solo objeto en la misma posición.

Es evidente que los humanos usamos la visión como una herramienta primordial para coger un objeto. La visión nos permite identificar el objeto, su posición, su ubicación, su forma y el punto más idóneo para ser capturado. Estas actividades son casi instantáneas, después viene el direccionamiento del brazo y la mano para alcanzar el objeto. Luego hay una apertura progresiva de esta hasta alcanzar el punto de donde se va a coger el objeto. La apertura de la mano está determinada por el tamaño del punto de donde se va a coger al objeto. Finalmente, cuando la mano alcanza el punto se cierra ejerciendo presión sobre el área o punto de agarre. A partir de este punto vienen otras actividades reactivas que permiten que el agarre se concrete y sea exitoso. La presión ejercida es regulada evitando deslizamientos o giros que produzcan que se pierda el contacto o agarre del objeto.

Los primeros trabajos en manipulación robótica fueron desarrollados con modelos en 2 dimensiones enfocándose en el diseño y control visual paso a paso de brazos robóticos hasta que capturen el objeto deseado. A esto se le denominó "Control Visual" (Visual Servoing). D. Kragic y H.Christensen [2] en su trabajo "Survey on Visual Servoing for Manipulation" desarrollaron un modelo que usa la data obtenida de un seguimiento visual a un brazo y a un objeto como entrada de retroalimentación para el lazo de control del brazo robótico. En el trabajo "Learning Visual Features to Predict Hand Orientations", Justus Piater [3] utiliza las características visuales de formas básicas (cuadrados, triángulos y círculos) en dos dimensiones para orientar el efector final de un brazo robótico, [2], [3].

1.2. Estado del arte

Como consecuencia del desarrollo de la inteligencia artificial (Machine Learning), el procesamiento de imágenes y las técnicas de reconstrucción tridimensional aparecieron técnicas como Whole Body Grasping y otras que tratan

de reconstruir el objeto tridimensionalmente usando escáneres láser o la estereoscopía.

Por ejemplo, K. Hsiao y T. Lozano-Pérez [4] en "Imitation Learning of Whole Body Grasps", desarrollaron un proyecto cuyo objetivo principal fue crear un sistema robótico que aprendiera a realizar agarres usando cualquier parte del cuerpo del robot con la demostración de ejemplos de agarres exitosos y fallidos.

En "Efficient Grasping from RGBD Image: Learning using a new Rectangle Representation" Y. Jiang, S. Moseson, y A. Saxena [5] utilizan una representación rectangular (ver figura 1-1) que se basa en las características visuales entre el fondo y el objeto para determinar un punto de agarre. A. Saxena, J. Driemeyer y Andrew Y. Ng [6] en "Learning 3-D Object Orientation from Images" propone el uso algoritmos de aprendizaje para estimar la orientación de objetos en un espacio tridimensional usando solo una imagen de este y usando una nueva representación para las orientaciones.

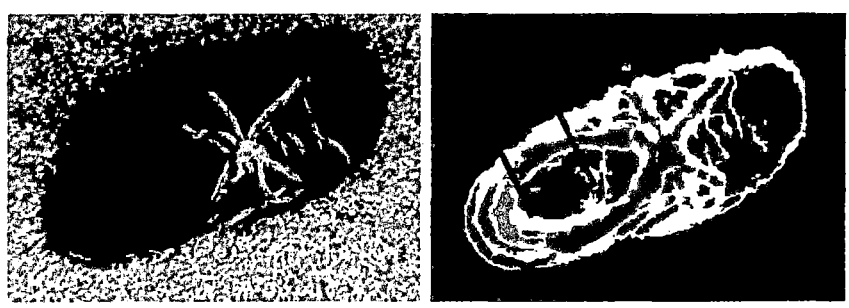


Figura 1-1. Imágenes de un objeto y su representación rectangular.

La información de escáneres 3D o imágenes estereoscópicas es adicionada a la información obtenida de la imagen del objeto. M. Quigley, S. Batra y S. Gould [7] usan un escáner tridimensional de alta resolución para mejorar el performance en la detección y apertura de puertas. A. Saxena y Lawson L.S. Wong [8] proponen un enfoque que estima la estabilidad de diferentes configuraciones para agarrar un objeto con la información obtenida de un sensor de profundidad (Swissranger depth sensor).

Como herramientas adicionales y para que el agarre se concrete fácilmente (o con mayor éxito) se usan sensores táctiles y de proximidad. K. Hsiao, P. Nangeroni y M. Huber [9] proponen un sistema para mejorar la manipulación de objetos usando sensores de proximidad en los dedos del manipulador (ver figura 1-2).

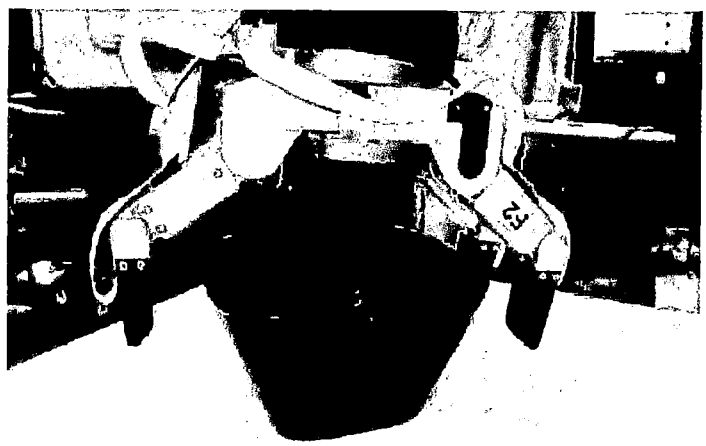


Figura 1-2. Manipulador Barrett Hand equipado con sensores ópticos de proximidad en los tres dedos.

Finalmente, E. Chinellato [10] presenta una investigación interdisciplinaria en la que une la robótica y la neurociencia. Define e implementa un modelo funcional de las áreas del cerebro involucradas en las acciones de agarre basadas en visión. Lo interesante de este enfoque es que propone la integración de las dos vías de procesamiento visual del cerebro humano (flujo dorsal y ventral).

1.3. Justificación

Los seres humanos podemos saber de dónde manipular un objeto usando una sola vista, por lo tanto, la selección del punto de agarre se puede obtener de las características visuales de una imagen del objeto que se desea manipular. Es por eso que se decide trabajar con las características bidimensionales de la imagen de un objeto y sólo realizar la reconstrucción tridimensional del punto de agarre para obtener sus coordenadas tridimensionales.

Encontrar un algoritmo de aprendizaje que nos relacione satisfactoriamente las características visuales (2D) de un objeto con su punto de agarre sin necesidad de reconstruir tridimensionalmente el objeto evitaría usar algoritmos más complejos y emplear tiempos de aprendizaje y cálculo aún mayores. Para este propósito usaremos una data generada digitalmente lo cual nos permitirá contar con suficiente cantidad de data en poco tiempo reduciendo las perturbaciones y lo laborioso que resulta realizar el etiquetamiento cuando se usa data real.

Los algoritmos de aprendizaje pueden presentar resultados diferentes para un mismo problema; algunos pueden resolver muy bien un problema específico pero tener resultados diferentes en otro. Estos resultados dependen de la complejidad del problema, del tipo de datos de entrenamiento y sobre todo, de los fundamentos de funcionamiento de cada algoritmo de aprendizaje. Por esta razón se plantea la experimentación de tres algoritmos de aprendizaje para saber cuál es más óptimo para esta aplicación.

1.4. Planteamiento del problema

Se tiene un sistema de visión con el cual se quiere obtener información visual para que un brazo robótico pueda capturar objetos en un ambiente tridimensional. Para que esto sea posible se necesitará principalmente las coordenadas tridimensionales del punto de agarre del objeto y la dirección para que el efector se posicione en ese punto.

Para obtener las coordenadas tridimensionales del punto de agarre se necesitarán por lo menos dos imágenes de ese punto tomadas de dos posiciones diferentes. Para la obtención de estas dos imágenes se ha propuesto usar un par estéreo formado con dos cámaras web de características similares.

Finalmente, para inferir el punto de agarre se va a realizar el entrenamiento de tres algoritmos de aprendizaje: Regresión Logística, Redes Perceptrón Multicapa y Redes con Función de Base Radial, para que relacionen las características de la imagen del objeto con su punto de agarre. El entrenamiento se realizará con data sintética de imágenes de tres clases de objetos en diferentes posiciones, colores y

textura, así como también de sus respectivos puntos de agarre. La finalidad es seleccionar el algoritmo con el que se puedan obtener los mejores resultados para este problema.

1.5. Objetivos

El objetivo principal es:

Encontrar un algoritmo de aprendizaje óptimo para la selección de un adecuado punto de agarre de un objeto, evaluando su desempeño con la data con la cual ha sido entrenado y con imágenes capturadas de un ambiente real.

Los objetivos específicos son:

- Desarrollar un algoritmo de reconstrucción tridimensional que nos permita calcular las coordenadas tridimensionales del punto de agarre inferido.
- Conocer el error que se produce en el cálculo de coordenadas usando los algoritmos de triangulación y correlación implementados.

1.6. Alcances

Las primeras pruebas se realizarán con imágenes sintéticas. La inferencia de imágenes de objetos reales se realizará sobre un fondo de color verde. Se probarán 22 objetos pertenecientes a las tres clases de objetos del entrenamiento y se captarán varias imágenes en diferentes posiciones y distancias. No hubo iluminación enfocada al objeto pues sólo se trabajó con la iluminación de la habitación en la noche y la luz solar en el día.

Por último, el cálculo de coordenadas tridimensionales se realizará con dos cámaras web soportadas por un trípode, las cuales capturarán imágenes del objeto. Además, se harán desplazamientos tratando de variar las distancias relativas entre las cámaras y el objeto; y lo que se va a calcular son las coordenadas tridimensionales del punto de agarre de este con respecto al sistema de

coordenadas usado en la calibración. No se realizará la inferencia de la dirección con la cual debe posicionarse el efector en ese punto.

1.7. Limitaciones

El fondo y el objeto son de color uniforme para incrementar el contraste entre ambos. Las pruebas sólo se realizarán con los objetos pertenecientes a los casos para los cuales se ha realizado el entrenamiento. No se presentará más de un objeto en la escena y se asume que se conoce la clase a la que pertenece el objeto del cual se va a inferir el punto de agarre para seleccionar el algoritmo entrenado. En las pruebas con objetos reales no se usarán objetos con perturbaciones como figuras o letras a colores.

CAPÍTULO II

FUNDAMENTOS DE VISIÓN POR COMPUTADOR

2.1. Visión por computador

Desde la perspectiva general, la visión por computador es la capacidad de la máquina para ver el mundo que le rodea, más precisamente para deducir la estructura y las propiedades del mundo tridimensional a partir de una o más imágenes bidimensionales.

En visión por computador, la escena tridimensional es vista por una, dos o más cámaras para producir imágenes monocromáticas o en color. Las imágenes adquiridas pueden ser segmentadas para obtener de ellas características de interés tales como bordes, textura, regiones, etc. Posteriormente, a partir de las características se obtienen las propiedades subyacentes mediante el correspondiente proceso de descripción, tras lo cual se consigue la estructura de la escena tridimensional requerida por la aplicación de interés. Las imágenes de intensidad están íntimamente ligadas al concepto de luminosidad.

El proceso de visión por computador puede subdividirse en seis técnicas principales:

- Captura o adquisición: Es el proceso a través del cual se obtiene una imagen digital utilizando un dispositivo de captura como una cámara digital.

- Pre procesamiento: Incluye técnicas tales como la reducción de ruido, realce del contraste, realce de ciertos detalles o características de la imagen.
- Segmentación: Es el proceso que divide una imagen en objetos que sean de nuestro interés.
- Descripción: Es el proceso que obtiene características convenientes para diferenciar un tipo de objeto de otro, como: la forma, el tamaño, área. etc.
- Reconocimiento: Es el proceso que identifica esos objetos.
- Interpretación: Es el proceso que asocia un significado a un conjunto de objetos reconocidos (llaves, tornillos, herramientas, etc.) y trata de emular la cognición.

En la figura 2-1 se muestra el diagrama de bloques que describe las etapas del proceso general de la visión por computador también llamada visión artificial.

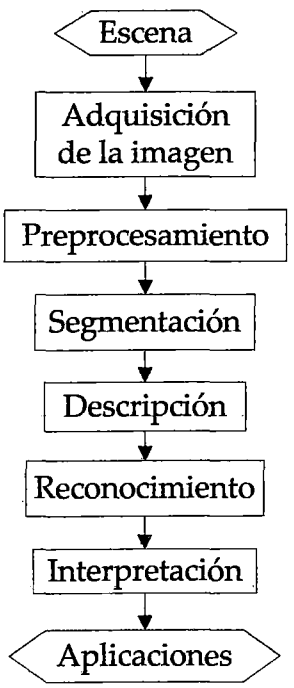


Figura 2-1. Diagrama general de la visión por computador.

2.2. Fundamentos de las imágenes

A continuación se presentarán los fundamentos de las imágenes de forma resumida ya que son los conceptos básicos para desarrollar los siguientes temas. En esta sección se explicará la formación de una imagen a partir de una escena tridimensional, y las características y representación de la data de una imagen.

2.2.1. Imagen

Una imagen es una representación visual que manifiesta la apariencia de una escena real. El concepto mayoritario al respecto corresponde al de la apariencia visual, por lo que el término suele entenderse como sinónimo de representación visual; sin embargo, hay que considerar también la existencia de imágenes auditivas, olfativas, táctiles, etcétera.

2.2.2. Formación de las imágenes

La luz de la escena pasa a través de un solo punto y proyecta una imagen invertida en el lado opuesto. En el ojo humano el brillo de la luz actúa similarmente. El funcionamiento de las cámaras analógicas y digitales muy parecido a diferencia que estas usan una combinación de lentes convergentes y divergentes para la formación de la imagen. En la figura 2-2 se presenta el esquema del funcionamiento de una cámara Pinhole para explicar cómo se forma la imagen a partir de las proyecciones de una escena tridimensional.

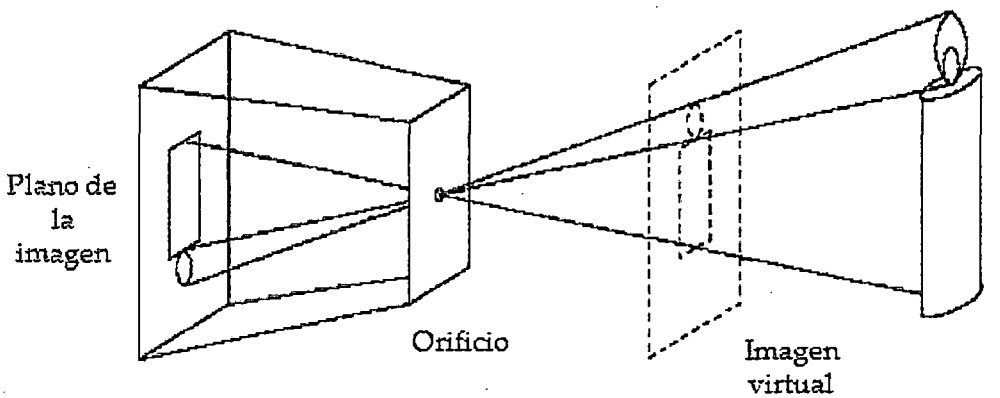


Figura 2-2. Funcionamiento de una cámara Pinhole.

2.2.3. Funcionamiento de la cámara digital

Una cámara digital es un dispositivo que utiliza un mecanismo para proyectar imágenes y un sensor electrónico para capturar esas proyecciones. La trayectoria que sigue la cámara para formar la imagen digital es la siguiente: La luz pasa a través de un conjunto de lentes convergentes y divergentes (objetivo) hasta llegar al sensor de imagen, denominado CCD formado por múltiples receptores fotosensibles denominados fotodiodos. La luz incidente en este sensor genera una pequeña señal eléctrica a cada receptor, que posteriormente, esta señal se transformará en datos digitales por el conversor ADC, como una serie de data binaria. En la figura 2-3 se muestran los componentes de una cámara digital que hacen posible la captura de una imagen.

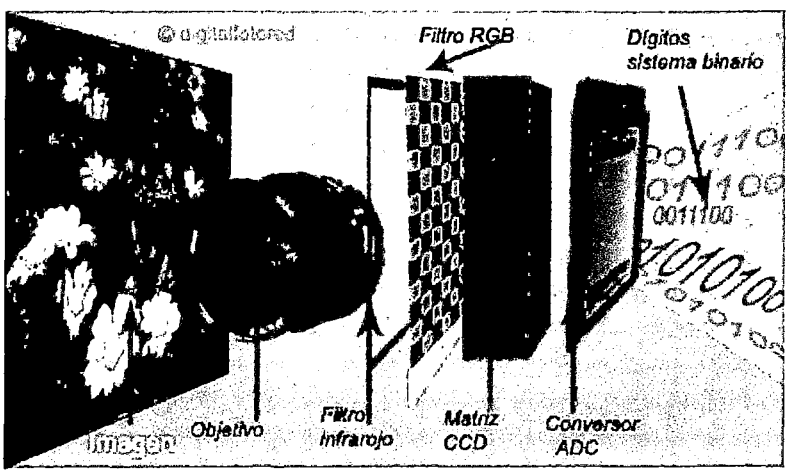


Figura 2-3. Componentes de una cámara digital.

2.2.4. Pixel

Un píxel o pixel, plural píxeles (acrónimo del inglés picture element, "elemento de imagen") es la menor unidad homogénea en color que forma parte de una imagen digital, ya sea esta una fotografía, un fotograma de vídeo o un gráfico.

2.2.5. Representación de imágenes digitales

Las imágenes digitales se forman como una sucesión de píxeles. La sucesión marca la coherencia de la información presentada, siendo su conjunto

una matriz coherente de información para el uso digital. El área donde se proyectan estas matrices, suelen ser rectangulares. La representación del píxel en pantalla, al punto de ser accesible a la vista por unidad, forma un área homogénea en cuanto a la variación del color y densidad por pulgada, siendo esta variación nula, y definiendo cada punto en base a la densidad, en lo referente al área.

En las imágenes de mapa de bits o en los dispositivos gráficos cada píxel se codifica mediante un conjunto de bits de longitud determinada (la llamada profundidad de color); por ejemplo, puede codificarse un píxel con un byte (8 bits), de manera que cada píxel admite 256 variaciones (2^8 variaciones con repetición de 2 valores posibles en un bit tomados de 8 en 8). En las imágenes de color verdadero, se suelen usar tres bytes para definir un color; es decir, en total podemos representar un total de 2^{24} colores, que suman 16.777.216 opciones de color (32 bits son los mismos colores que 24 bits, pero tiene 8 bits más para transparencia). En la figura 2-4 se presenta una imagen en escala de grises y la matriz que representa los datos de dicha imagen.

$$I_{(u,v)} = \begin{bmatrix} g_{11} & g_{12} & \cdots & g_{1n} \\ g_{21} & g_{22} & \cdots & g_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ g_{m1} & g_{m2} & \cdots & g_{mn} \end{bmatrix};$$

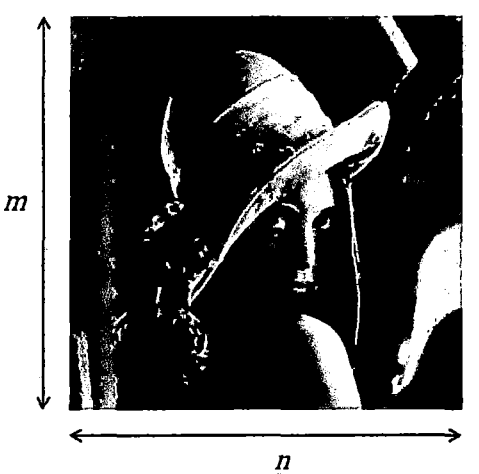
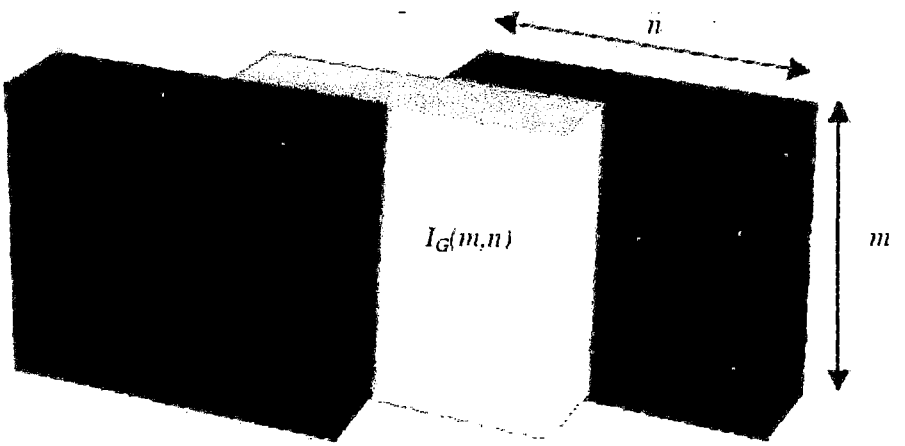


Figura 2-4. Representación de una imagen en escala de grises.

Para poder transformar la información numérica que almacena un píxel en un color, hemos de conocer, además de la profundidad y brillo del color (el tamaño en bits del píxel), el modelo de color que estamos usando. Por ejemplo, el modelo de color RGB (*Red-Green-Blue*) permite crear un color compuesto por los tres colores primarios según el sistema de mezcla aditiva. De esta forma, en función de la cantidad de cada uno de ellos que usemos veremos un resultado u otro. Por ejemplo, el color violeta se obtiene mezclando el rojo y el azul. Las distintas tonalidades del violeta se obtienen variando la proporción en que intervienen ambas componentes. En el modelo RGB es frecuente que se usen 8 bits para representar la proporción de cada una de las tres componentes primarias. De esta forma, cuando una de las componentes vale 0, significa que esta no interviene en la mezcla y cuando vale 255 ($2^8 - 1$) significa que interviene aportando el máximo de ese tono. La mayor parte de los dispositivos que se usan con un ordenador (monitor, escáner, etc.) usan el modelo RGB. En la figura 2-5 se muestra la conformación de datos de una imagen en formato RGB a partir de tres matrices de dimensiones iguales.



$$I_{m,n,1} = \begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ r_{21} & r_{22} & \cdots & r_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ r_{m1} & r_{m2} & \cdots & r_{mn} \end{bmatrix}; \; I_{m,n,2} = \begin{bmatrix} g_{11} & g_{12} & \cdots & g_{1n} \\ g_{21} & g_{22} & \cdots & g_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ g_{m1} & g_{m2} & \cdots & g_{mn} \end{bmatrix}; \; I_{m,n,3} = \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1n} \\ b_{21} & b_{22} & \cdots & b_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ b_{m1} & b_{m2} & \cdots & b_{mn} \end{bmatrix}$$

Figura 2-5. Representación de una imagen a color RGB.

2.2.6. Espacios de color: RGB, YCrCb y HSV

Los diferentes formatos de color son normalmente inventados por razones prácticas y aplicativas, por lo tanto hay una gran variedad de ellos. Por ejemplo algunas propiedades pueden ser representadas mejor en un formato que en otro. Esto puede ser útil para algunas aplicaciones que requieran énfasis en ciertas propiedades particulares.

a. Espacio de color RGB

La descripción RGB (del inglés Red, Green, Blue; "rojo, verde, azul") de un color hace referencia a la composición del color en términos de la intensidad de los colores primarios con que se forma: el rojo, el verde y el azul. Es un espacio de color basado en la síntesis aditiva, con el que es posible representar un color mediante la mezcla por adición de los tres colores luz primarios. El espacio de color RGB no define por sí mismo lo que significa exactamente rojo, verde o azul, por lo que los mismos valores RGB pueden mostrar colores notablemente diferentes en diferentes dispositivos que usen este espacio de color. Aunque utilicen un mismo espacio de color, sus espacios de color pueden variar considerablemente. En la figura 2-6 se muestra el modelo aditivo del espacio de color RGB.

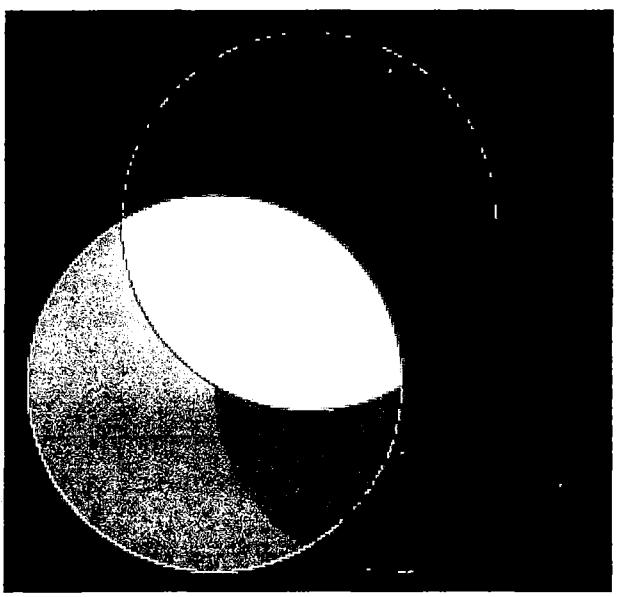


Figura 2-6. Modelo aditivo de colores rojo, verde, azul.

Para indicar con qué proporción mezclamos cada color, se asigna un valor a cada uno de los colores primarios, de manera, por ejemplo, que el valor 0 significa que no interviene en la mezcla y, a medida que ese valor aumenta, se entiende que aporta más intensidad a la mezcla. Aunque el intervalo de valores podría ser cualquiera (valores reales entre 0 y 1, valores enteros entre 0 y 37, etc.), es frecuente que cada color primario se codifique con un byte (8 bits). Así, de manera usual, la intensidad de cada una de las componentes se mide según una escala que va del 0 al 255.

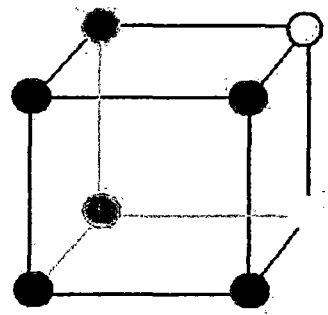


Figura 2-7. Cubo RGB.

En la figura 2-7 se muestra el cubo RGB donde cada arista representa un color. Por lo tanto, el rojo se obtiene con (255,0,0), el verde con (0,255,0) y el azul con (0,0,255), obteniendo, en cada caso un color resultante monocromático. La ausencia de color —lo que nosotros conocemos como color negro— se obtiene cuando las tres componentes son 0, (0,0,0). La combinación de dos colores a nivel 255 con un tercero en nivel 0 da lugar a tres colores intermedios. De esta forma el amarillo es (255,255,0), el cyan (0,255,255) y el magenta (255,0,255). Obviamente, el color blanco se forma con los tres colores primarios a su máximo nivel (255,255,255).

El conjunto de todos los colores se puede representar en forma de cubo. Cada color es un punto de la superficie o del interior de éste. La escala de grises estaría situada en la diagonal que une al color blanco con el negro.

b. Espacio de color YCrCb

El espacio de color YCbCr, a veces escrito YC_BC_R, es un modelo de representación de color usado en sistemas de video y fotografía digital. Y es la componente que representa la luminosidad y C_B y C_R son las componentes que representan la crominancia (color) respecto al azul y al rojo respectivamente. Este modelo se desarrolló para permitir la transmisión de información a color en televisores a color y a la vez garantizar que los televisores blanco y negro existentes continuaran mostrando una imagen en tonos de gris.

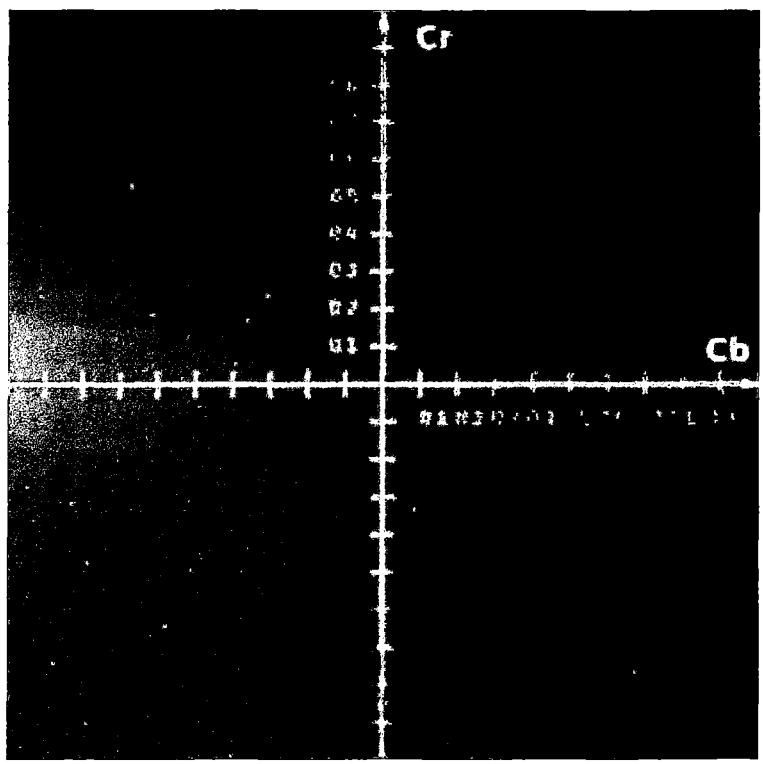


Figura 2-8. El plano Cb-Cr a una luminancia constante Y=0,5.

En la figura 2-8 se muestra el plano Cb-Cr para una luminancia constante de 0,5. A continuación se muestran las relaciones entre los valores de la representación RGB y la representación YCbCr:

- Y = 0.299R + 0.587 G + 0.114 B
- Cr = -0.147R - 0.289 G + 0.436B = 0.492(B - Y)
- Cb = 0.615R -0.515G -0.100B = 0.877(R-Y)

El espacio de color no es absoluto, es una forma de representar la información de una imagen. El color mostrado depende de los valores de RGB usados para mostrar la señal. Por lo tanto los valores expresados como YCbCr solo se pueden conocer si se usa el estándar RGB o un perfil ICC.

c. Espacio de color HSV

El espacio no lineal de color HSV (Hue, Saturation y Value - matiz, saturación y valor o brillo) es una representación en coordenadas cilíndricas de los puntos de una imagen en el espacio de color RGB. En esta representación la geometría RGB ha sido reacomodada para que sea perceptualmente más relevante que la representación cartesiana. Fue desarrollada en 1970 para aplicaciones en Computer Graphics y es usada como selector de color en herramientas de programas de edición de imagen, análisis de imágenes y visión por computadora.

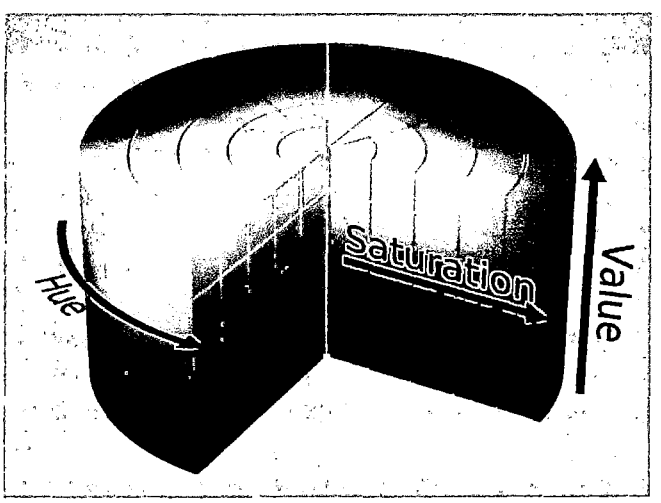


Figura 2-9. Representación cilindrica HSV.

En la figura 2-9 se puede apreciar la representación cilíndrica, el ángulo alrededor del eje vertical central corresponde al matiz (Hue), la distancia del eje corresponde a la saturación (Saturation), y la distancia a lo largo del eje corresponde al brillo (Value).

2.3. Procesamiento digital de Imágenes

El procesamiento digital de imágenes es el conjunto de técnicas que se aplican a las imágenes digitales con el objetivo de mejorar la calidad o facilitar la búsqueda de información. Dentro de estas técnicas están consideradas las operaciones morfológicas, la segmentación, el proceso de filtrado, etc. Por ejemplo en la figura 2-10 se muestra cuatro imágenes. La primera imagen (a) es la imagen sin la aplicación de ningún filtro y las tres restantes la imagen aplicando tres filtros espaciales (b, c y d).

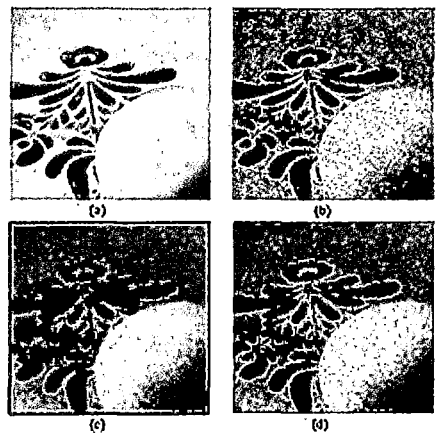


Figura 2-10. Ejemplos de filtrado sobre una imagen.

2.3.1. Operaciones morfológicas básicas

Una de las operaciones más utilizadas en visión sobre imágenes previamente binarizadas son las operaciones morfológicas. Las operaciones morfológicas son operaciones realizadas sobre imágenes binarias basadas en formas. Estas operaciones toman como entrada una imagen binaria regresando como resultado otra imagen binaria. El valor de cada pixel de la imagen binaria resultante está basado en el valor del correspondiente pixel de la imagen original binaria y sus vecinos. Entonces eligiendo apropiadamente la forma de los vecinos a considerar, puede construirse operaciones morfológicas sensibles a una forma en particular.

Las principales operaciones morfológicas son la dilatación y la erosión. La operación de dilatación adiciona píxeles en las fronteras de los objetos, mientras que la erosión los remueve. En ambas operaciones como se mencionó se utiliza una rejilla que determina qué vecinos del elemento central de la rejilla serán tomados en cuenta para la determinación del pixel resultante. La rejilla es un arreglo cuadricular que contiene unos y ceros, en los lugares que contiene uno serán los vecinos de la imagen original con respecto al pixel central, los cuales serán tomados en consideración para determinar el pixel de la imagen resultante, mientras los lugares que contengan cero no serán tomados en cuenta.

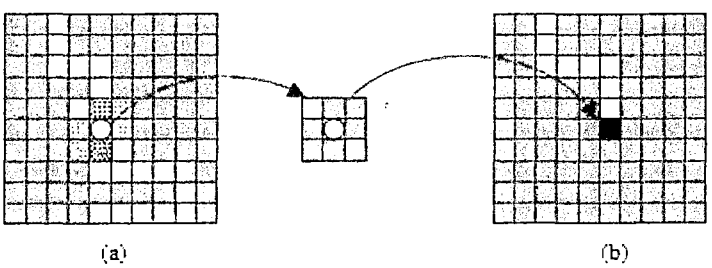


Figura 2-11. a) Imagen original y b) Imagen resultante.

Una vez determinado el tamaño de la rejilla y su configuración. Se aplica la operación morfológica. En la figura 2-11 se puede apreciar la imagen original (a) y la imagen resultante (b) de aplicación de una rejilla de 3x3. En el caso de la dilatación, si alguno de los pixeles de la rejilla configurados como unos coincide con al menos uno de la imagen original el resultado es uno. Por el contrario en la erosión todos los pixeles de la rejilla configurados como unos deben coincidir con todos los de la imagen si esto no sucede el pixel es cero.

2.3.2. Operaciones basadas en objetos

En una imagen binaria, puede definirse un objeto como un conjunto de pixeles conectados con valor 1. Por ejemplo en la figura 2-12 se presenta una imagen binaria conteniendo un objeto cuadrado de 4x4. El resto de la imagen puede ser considerado como fondo.

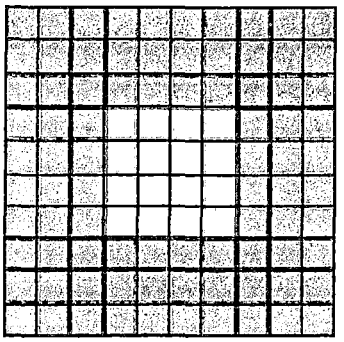


Figura 2-12. Imagen binaria conteniendo un objeto.

Para muchas operaciones la distinción de objetos depende de la convención utilizada de conectividad, que es la forma en la que se considera si dos pixeles tiene relación como para considerar que forman parte del mismo objeto. La conectividad puede ser de dos tipos, de conexión 4 o de conexión 8. La figura 2-13 esquematiza ambas conectividades.

Figura 2-13. a) Conectividad conexión 8 y b) conexión 4.

En la conectividad conexión 8 se dice que el pixel rojo pertenece al mismo objeto si existe un pixel de valor uno en las posiciones 1, 2, 3, 4, 5, 6, 7 y 8. Por su parte la conectividad conexión 4 relaciona solo a los pixeles 2, 4, 5 y 7.

Para las operaciones que consideran conectividad como un parámetro es importante tomar en cuenta que esta determina fuertemente el resultado final del procesamiento puesto que puede dar origen a objetos nuevos en donde que si se hubiera elegido otra conectividad no existieran.

2.3.3. Segmentación de imágenes digitales

La segmentación es el proceso de dividir una imagen en varias partes (grupos de pixeles) u objetos. El objetivo de la segmentación es simplificar y/o cambiar la representación de una imagen en otra más significativa y más fácil de analizar. La segmentación se usa tanto para localizar objetos como para encontrar los límites de estos dentro de una imagen. Más precisamente, la segmentación de la imagen es el proceso de asignación de una etiqueta a cada pixel de la imagen de forma que los píxeles que compartan la misma etiqueta también tendrán ciertas características visuales similares.

- **Imágenes binarias y segmentación por umbral**

Una imagen binaria es una imagen en la cual cada pixel puede tener solo un valor 0 ó 1. Como es lógico de suponer una imagen en estas condiciones es mucho más fácil de encontrar y distinguir características estructurales. En visión por computador el trabajo de imágenes binarias es muy importante ya sea para realizar segmentación por intensidad de la imagen, para generar algoritmos de reconstrucción o reconocer estructuras.

La forma más común de generar imágenes binarias es mediante la utilización del valor umbral de una imagen a escala de grises; es decir se elige un valor límite o un intervalo a partir del cual todos los valores que estén por debajo serán codificados a cero y el resto a 1.

- **Segmentación de imágenes en el espacio de color HSV**

La segmentación empleada tanto con RGB como HSV se fundamenta en la detección de umbrales del mapeo de componentes RGB y/o HSV. La umbralización de imágenes de objetos muchas veces es compleja por la falta de conocimiento a priori del número de objetos a detectar, por la influencia de elementos no deseados como sombras, brillos, la complejidad de colores, texturas y tamaños de los objetos y la posibilidad de solapamiento de estos, así como las variaciones en el fondo. Esto hace que la elección de un número bajo

de umbrales influya en la segmentación, se detectan más regiones de las que interesa. Tanto una como otra, se puede deber a dos motivos principalmente, la aparición de sombras y brillos por un lado, y por otro la similitud de los colores que tienen los objetos presentes en la escena, principalmente.

2.3.4. Proceso de filtrado

Es el conjunto de técnicas englobadas dentro del pre procesamiento de imágenes cuyo objetivo fundamental es obtener, a partir de una imagen origen, otra final cuyo resultado sea más adecuado para una aplicación específica mejorando ciertas características de la misma que posibilite efectuar operaciones del procesado sobre ella.

Los principales objetivos que se persiguen con la aplicación de filtros son:

- Suavizar la imagen: reducir la cantidad de variaciones de intensidad entre píxeles vecinos.
- Eliminar ruido: eliminar aquellos píxeles cuyo nivel de intensidad es muy diferente al de sus vecinos y cuyo origen puede estar tanto en el proceso de adquisición de la imagen como en el de transmisión.
- Realzar bordes: destacar los bordes que se localizan en una imagen.
- Detectar bordes: detectar los píxeles donde se produce un cambio brusco en la función intensidad.

Por tanto, se consideran los filtros como operaciones que se aplican a los píxeles de una imagen digital para optimizarla, enfatizar cierta información o conseguir un efecto especial en ella. El proceso de filtrado puede llevarse a cabo sobre los dominios de frecuencia y/o espacio.

- **Filtrado en el dominio del espacio**

El filtraje espacial es una de las operaciones comunes en la visión por computador ya sea para realizar efectos de eliminación de ruido o bien detección de bordes. En ambos casos la determinación de los pixeles de la nueva imagen

depende del pixel de imagen original y sus vecinos. De esta forma es necesario configurar una matriz (máscara o ventana) que considere cuales vecinos y en qué forma influirán en la determinación de el nuevo pixel. En la figura 2-14 se muestra una máscara de 3x3 que va a ser aplicada a una imagen. Esta máscara tiene un elemento central (i,j) y los ocho elementos restantes que forman su vecindad.

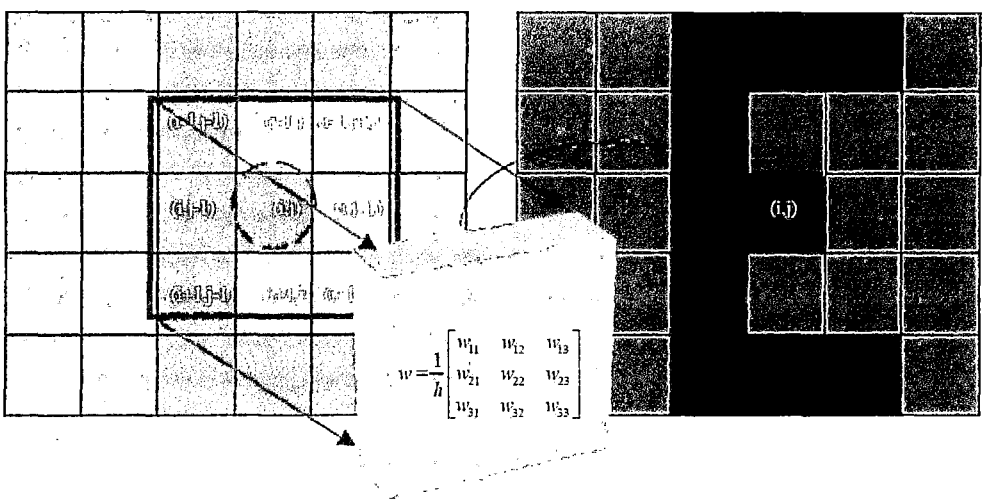


Figura 2-14. Filtrado espacial por una máscara 3x3.

Las operaciones de filtrado se llevan a cabo directamente sobre los píxeles de la imagen. En este proceso se relaciona, para todos y cada uno de los puntos de la imagen, un conjunto de píxeles próximos al píxel objetivo con la finalidad de obtener una información útil, dependiente del tipo de filtro aplicado, que permita actuar sobre el píxel concreto en que se está llevando a cabo el proceso de filtrado para, de este modo, obtener mejoras sobre la imagen y/o datos que podrían ser utilizados en futuras acciones o procesos de trabajo sobre ella. Los filtros en el dominio del espacio pueden clasificarse en filtros lineales (filtros basados en kernels o máscaras de convolución) y los filtros no lineales.

El concepto de *kernel* se entiende como una matriz de coeficientes donde el entorno del punto *(x,y)* que se considera en la imagen para obtener *g(x,y)* está determinado por el tamaño y forma del kernel seleccionado. Aunque la forma y tamaño de esta matriz es variable y queda a elección de cada usuario, es común el uso de kernels cuadrados *nxn*. Dependiendo de la implementación, en los

límites de la imagen se aplica un tratamiento especial (se asume un marco exterior de ceros o se repiten los valores del borde) o no se aplica ninguno. Es por ello, que el tipo de filtrado queda establecido por el contenido de dicho kernel utilizado.

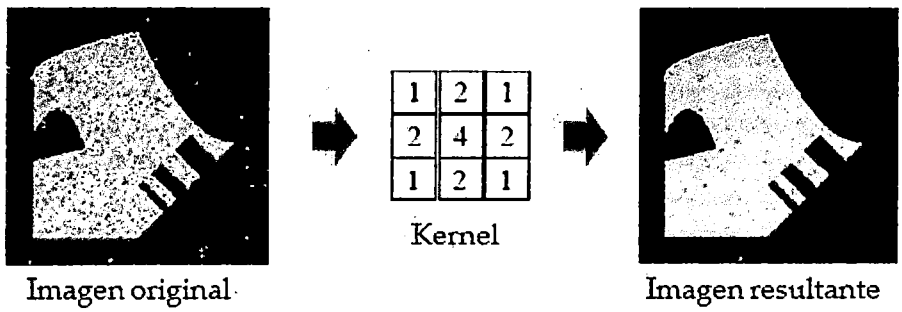


Figura 2-15. Aplicación de un filtro mediana.

La figura 2-15 muestra a aplicación de un filtro mediana ponderada (kernel) a una imagen con un ruido aleatorio (imagen original) y la imagen resultante donde se aprecia la disminución del ruido. Para realizar un filtrado en el dominio del espacio se realiza una convolución del kernel sobre la imagen. Para ello se sigue el Teorema de Convolución en el espacio: $g(x,y) = h(x,y) * f(x,y)$.

1. Cada píxel de la nueva imagen se obtiene mediante el sumatorio de la multiplicación del kernel por los píxeles contiguos: $g(x,y) = \sum\sum f(i,j) \cdot w(i,j)$.
2. Generalmente se divide sobre cierto valor constante para normalizar, que suele obtenerse de la suma de los valores del kernel empleado.

- **Filtros de bordes orientados (R. Nevatia y K. Babu 1980)**

Los filtros de bordes orientados que a continuación se presentan fueron desarrollados por R. Nevatia y K. Babu [11] y fueron publicados en Linear Feature Extraction and Description Computer Graphics and Image Processing 1980. Los filtros de bordes orientados tienen por objetivo el mejoramiento de la imagen y optimizar el proceso de interpretación. Se aplica filtros de bordes orientados para determinar la orientación dominante en una imagen y es

obtenida mediante una convolución de pequeñas máscaras similares a un borde. Los filtros orientados requieren de un parámetro de entrada denominado perfil de orientación. Un filtro orientado es un filtro rotado a partir de la combinación lineal de un conjunto de filtros base a orientaciones específicas. Filtros orientados usualmente se aplican en los siguientes 6 ángulos (0, 30, 60, 90, 120,150).

$$NB_1 = \begin{bmatrix} -100,-100,0,100,100 \\ -100,-100,0,100,100 \\ -100,-100,0,100,100 \\ -100,-100,0,100,100 \\ -100,-100,0,100,100 \end{bmatrix}$$

$$NB_2 = \begin{bmatrix} -100,32,100,100,100 \\ -100,-78,92,100,100 \\ -100,-100,0,100,100 \\ -100,-100,-92,78,100 \\ -100,-100,-100,-32,100 \end{bmatrix}$$

$$NB_5 = \begin{bmatrix} 100 & 100 & 100 & 100 & 100 \\ 100 & 100 & 100 & 78 & -32 \\ 100 & 92 & 0 & -92 & -100 \\ 32 & -78 & -100 & -100 & -100 \\ -100 & -100 & -100 & -100 & -100 \end{bmatrix}$$

Partiendo los filtros NB_1 (Nevatia y Babu), NB_2 y NB_5 podemos encontrar los siguientes filtros mediante operaciones matemáticas simples.

$$NB_3 = -NB_2'$$
$$NB_4 = -NB_1'$$
$$NB_6 = NB_5'$$

En la figura 2-16 se muestra la imagen de los kernel de los filtros orientados de 0, 30 y 60 grados con los cuales se realizará la convolución. Como se puede apreciar tienen una inclinación aproximada a los ángulos que representan.

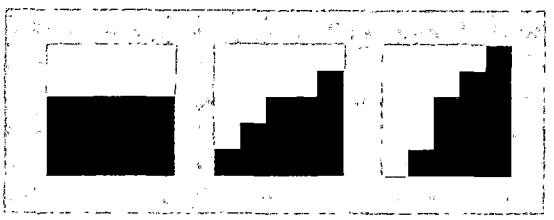


Figura 2-16. Filtros orientados a 0, 30 y 60 grados.

2.3.5. Textura

La definición de textura no es única o exacta pero todos los investigadores coinciden en que es la variación de intensidad entre pixeles cercanos. Es una propiedad homogénea a una escala espacial mayor que la resolución de la imagen. La IEEE en el estándar 610.4 [12] define la textura como un atributo que presenta la distribución espacial de los niveles de gris de una determinada región. El análisis de textura ayuda al reconocimiento y clasificación de objetos. Se aplica a imágenes satelitales, imágenes médicas, etc.

El termino textura se refiere, en general, a la repetición de elementos básicos llamados texels. Cada texel contiene varios pixeles, cuya ubicación puede ser periódica, cuasi periódica o aleatoria. Las texturas naturales son generalmente aleatorias, mientras que las artificiales son a menudo deterministas o periódicas. Una textura puede ser gruesa, fina, suave, granulada, rugosa, regular, irregular, lineal, etc. En la figura 2-17 podemos apreciar la imagen de una textura con dirección vertical y una imagen sin dirección. En la figura 2-18 tenemos dos ejemplos diferentes de textura; textura suave y textura rugosa.

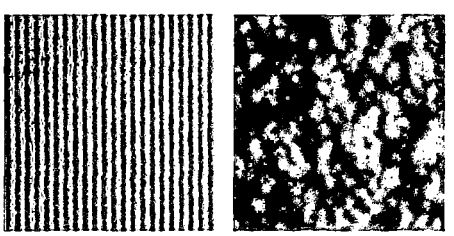


Figura 2-17. Textura direccional y textura no direccional.

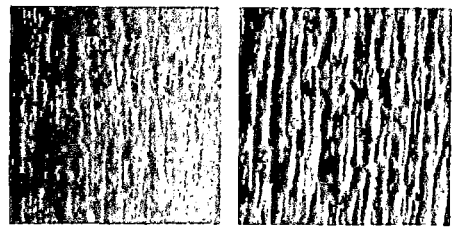


Figura 2-18. Textura suave y textura rugosa.

Como se explicó anteriormente la textura también se puede presentar de formas gruesas o finas (ver figura 2-19) o finalmente de formas regulares e irregulares (ver figura 2-20).

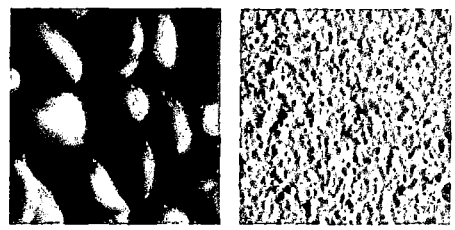


Figura 2-19. Textura gruesa y textura fina.

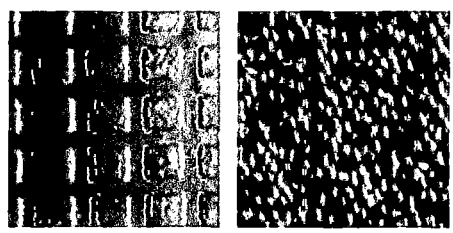


Figura 2-20. Textura regular y textura irregular.

En el análisis de la textura existen diversos métodos como los métodos estadísticos, geométricos y los métodos basados en el cálculo de la energía. En estos últimos la imagen se trata como si fuera una señal y se aplican filtros a la imagen extrayendo características por cada pixel y tienen varias aplicaciones en la segmentación y clasificación de imágenes. En esta oportunidad trabajaremos con el cálculo de la energía mediante el método de Laws.

- **Cálculo de la energía mediante el método de Laws**

La técnica de análisis de textura mediante la medida de la energía de textura fue desarrollada por K. Laws [13], (Laws, 1979) (Laws, 1980) y ha sido usada en muchas aplicaciones del análisis de imágenes como clasificación y segmentación. La medida de la energía de la textura en imágenes bidimensionales es calculada aplicando la convolución con filtros. En esta técnica tres filtros básicos son usados (Level, Edge y Spot).

$$L_3 = [1\ 2\ 1];\ E_3 = [-1\ 0\ 1];\ S_3 = [-1\ \ 2\ \ -1];$$

En este caso la longitud de los filtros es tres, pero también se usan extensiones de dimensiones a partir de estos filtros básicos. La extensión se puede realizar mediante la convolución de un par de filtros básicos. Por ejemplo, los filtros de longitud cinco se pueden obtener mediante la convolución de una de filtros de longitud tres.

Convolucionando estas características:

$$L_5 = [1, 4, 6, 4, 1] = L_3 * L_3;\ Level$$
$$S_5 = [-1, 0, 2, 0, -1] = L_3 * S_3;\ Spot$$
$$R_5 = [1, -4, 6, -4\ 1] = S_3 * S_3;\ Ripple$$
$$E_5 = [-1, -2, 0, 2, 1] = L_3 * E_3; Edge$$
$$W_5 = [1, -2, 0, 2, -1] = E_3 * S_3; Wave$$

Las características con las cuales están relacionados los filtros son:

- $Level$ es un filtro que calcula la media ponderada, proporciona información del nivel de gris promedio en el vecindario.
- $Edge$ es un filtro de realce de bordes.
- $Spot$ es un filtro que realza ciertas formas en la dimensión de los niveles de gris.
- $Wave$ es un filtro de ondulación.
- $Ripple$ es un filtro de rugosidad.

La suma de todos los elementos de los filtros es cero excepto el filtro L_5. Muchas aplicaciones usan los filtros de Laws con tamaños 3 y 5 para extraer los valores de la energía de la textura. Cada uno de esos filtros unidimensionales es usado para general filtros bidimensionales mediante su combinación. Como se muestra el siguiente ejemplo:

$$L_3'L_3 = \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

Todos los filtros de dos dimensiones (3 x 3) que se puede generar son los siguientes:

$$\begin{matrix} L_3'L_3 & L_3'E_3 & L_3'S_3 \\ E_3'L_3 & E_3'E_3 & E_3'S_3 \\ S_3'L_3 & S_3'E_3 & S_3'S_3 \end{matrix}$$

De forma similar se pueden crear filtros bidimensionales de dimensiones 5 x 5 o 7 x 7, etc.

$$E_5' \cdot L_5 = E_5L_5$$

$$\begin{bmatrix} -1 \\ -2 \\ 0 \\ 2 \\ 1 \end{bmatrix} * [1\ 4\ 6\ 4\ 1] = \begin{bmatrix} -1 & -4 & -6 & -4 & -1 \\ -2 & -8 & -12 & -8 & -1 \\ 0 & 0 & 0 & 0 & 0 \\ 2 & 8 & 12 & 8 & 2 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$$

Todas las combinaciones posibles de filtros bidimensionales de 5x5 se muestran a continuación. La figura 2-21 muestra las imágenes de algunos de los filtros aplicados en la extracción de características.

$$\begin{matrix} L_5'L_5 & E_5'L_5 & S_5'L_5 & W_5'L_5 & R_5'L_5 \\ L_5'E_5 & E_5'E_5 & S_5'E_5 & W_5'E_5 & R_5'E_5 \\ L_5'S_5 & E_5'S_5 & S_5'S_5 & W_5'S_5 & R_5'S_5 \\ L_5'W_5 & E_5'W_5 & S_5'W_5 & W_5'W_5 & R_5'W_5 \\ L_5'R_5 & E_5'R_5 & S_5'R_5 & W_5'R_5 & R_5'R_5 \end{matrix}$$

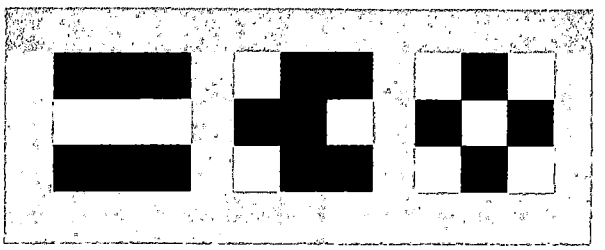


Figura 2-21. Filtros bidimensionales aplicados.

- **Aplicación de la convolución**

Una vez que los filtros bidimensionales se han obtenido, se realiza la convolución con la imagen que se desea analizar. La convolución de la imagen $I_{(i,j)}$ y el filtro $F_{(i,j)}$ de tamaño $2t+1$ por $2t+1$ es expresada en la siguiente ecuación:

$$R_{(i,j)} = F_{(i,j)} * I_{(i,j)} = \sum_{k=-t}^{t} \sum_{l=-t}^{t} F_{(k,l)} I_{(i+k,j+l)} \quad \text{Ec (2.1)}$$

Donde la ecuación Ec (2.1) denota una convolución en dos dimensiones. Finalmente el cálculo de la energía se realiza elevando al cuadrado cada elemento de la matriz $R_{(i,j)}$ resultado de la convolución. Por último también se pueden crear filtros tridimensionales a partir de los filtros unidimensionales (Suzuki and Yaginuma, 2007) [14].

2.4. Calibración de Cámaras

En el marco de los sistemas de visión por computadora, la calibración se refiere al establecimiento del modelo de la cámara. El problema es importante pues la determinación de los parámetros de funcionamiento de la cámara permite resolver una gran cantidad de problemas entre los que destacan el mapeo de punto de la escena de la imagen, la determinación de medidas en un sistema de referencia global y otros. Para su mejor estudio los parámetros de la cámara se dividen en intrínsecos y extrínsecos. Los parámetros intrínsecos relacionan las medidas en el sistema de referencia de la escena (milímetros) con la posición de los puntos en la imagen expresados en pixeles. Los parámetros extrínsecos relacionan los sistemas de referencia de la escena y la cámara.

2.4.1. Parámetros geométricos de las cámaras

Las coordenadas (x, y, z) de un punto P en una escena observada mediante una cámara de orificio están relacionadas con las coordenadas en la imagen (x', y') por la ecuación de perspectiva. En realidad, esa ecuación es sólo valida cuando todas las distancias son medidas en el cuadro de referencia de la cámara y las coordenadas de la imagen tienen su propio origen en el punto principal donde el eje de simetría atraviesa su retina. En la práctica, el sistema de coordenadas del espacio y de la cámara están relacionados por un conjunto de parámetros físicos, tales como la distancia focal de los lentes, el tamaño de los pixeles, la posición del punto principal, y la posición y orientación de la cámara.

A continuación identificaremos esos parámetros. Nosotros distinguiremos los parámetros intrínsecos como aquellos que relacionan el sistema de coordenadas de la cámara con un sistema de coordenadas idealizado y a los parámetros extrínsecos como aquellos que relacionan el sistema de coordenadas de la cámara con un sistema de coordenadas fijo en el espacio; especificando su posición y orientación en el espacio.

- **Parámetros intrínsecos**

Podemos asociar una cámara con dos planos diferentes de imagen: El primero es un plano normalizado ubicado a una unidad de distancia del orificio. Le atribuimos a ese plano su propio sistema de coordenadas con un origen ubicado en $\hat{C}$ donde el eje óptico lo atraviesa. La ecuación perspectiva de proyección puede ser escrita en ese sistema de coordenadas normalizadas como:

$$\begin{cases} \hat{u} = \frac{x}{z} \\ \hat{v} = \frac{y}{z} \end{cases} \leftrightarrow \hat{p} = \frac{1}{z}(Id \quad 0)\binom{P}{1} \qquad \text{Ec (2.2)}$$

Donde $\hat{p} \stackrel{\text{def}}{=} (\hat{u} \quad \hat{v} \quad 1)^T$ es el vector de coordenadas homogéneas de la proyección $\hat{p}$ del punto P en el plano normalizado.

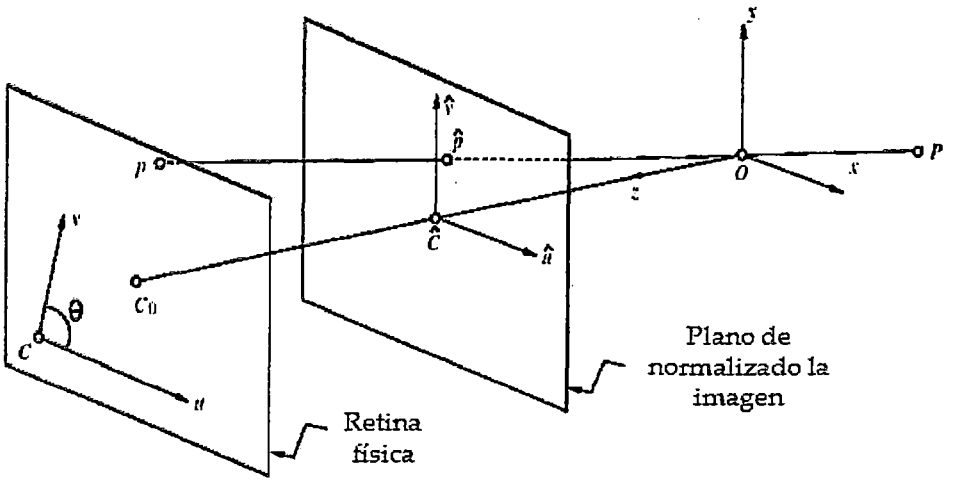


Figura 2-22. Plano físico y plano normalizado.

La retina física de la cámara es en general diferente (ver Figura 2-22), está localizada a una distancia $f \neq 1$ del orificio de la cámara, y las coordenadas en la imagen (u, v) del punto p están usualmente expresadas en unidades de pixeles. Además los pixeles son normalmente rectangulares en lugar de ser cuadrados, entonces la cámara tiene dos parámetros de escala dimensional k y l.

$$\begin{cases} u = kf\frac{x}{z} \\ v = lf\frac{y}{z} \end{cases} \qquad \text{Ec (2.3)}$$

Ahora hablaremos de las unidades: f es una distancia, expresada por ejemplo en metros, y a un pixel tendrá dimensiones $\frac{1}{k} \times \frac{1}{l}$, donde k y l esta expresados en pixel $\times m^{-1}$. Los parámetros k, l y f no son independientes, puede ser reemplazados por $\alpha = kf$ y $\beta = lf$ expresados en unidades de pixeles.

Ahora en general, el origen actual del sistema de coordenadas de la cámara está en una esquina C de la retina (por ejemplo en el caso representado en la figura 1, la esquina inferior izquierda o algunas veces la esquina superior izquierda, donde las coordenadas de la imagen son los índices de fila y columna del pixel) y no es su centro, y el centro de la matriz CCD usualmente no coincide

con el punto principal C_o. Esto agrega dos parámetros u_o y v_o que definen la posición (en unidades de pixel) de C_o en el sistema de coordenadas de la retina. Si reemplazamos estos parámetros en la Ec (2.3) tenemos:

$$\begin{cases} u = \alpha\frac{x}{z} + u_0 \\ v = \beta\frac{x}{z} + v_0 \end{cases} \qquad \text{Ec (2.4)}$$

Finalmente, el sistema de coordenadas de la cámara puede estar ligeramente inclinado probablemente por problemas de manufactura, entonces el ángulo θ entre los dos ejes de la imagen puede ser un ligeramente diferente de 90 grados. En ese caso, es fácil de reemplazar en la Ec (2.4) llegando a:

$$\begin{cases} u = \alpha\frac{x}{z} - \alpha\cot\theta\frac{y}{z} + u_0 \\ v = \frac{\beta}{\sin\theta}\frac{y}{z} + v_0 \end{cases} \qquad \text{Ec (2.5)}$$

Combinando las ecuaciones (2.2) y (2.5) nos permite escribir el cambio de coordenadas entre el cuadro físico de la imagen y el cuadro normalizado como una transformación planar:

$$p = K\hat{p}, \text{ donde } p = \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} \text{ y } K \stackrel{\text{def}}{=} \begin{pmatrix} \alpha & -\alpha\cot\theta & u_0 \\ 0 & \frac{\beta}{\sin\theta} & v_0 \\ 0 & 0 & 1 \end{pmatrix}.$$

Poniendo junto, obtenemos:

$$p = \frac{1}{z}MP, \text{ donde } M \stackrel{\text{def}}{=} (K \quad 0) \qquad \text{Ec (2.7)}$$

Y P denota ahora el vector de coordenadas homogéneas de P en el sistema de coordenadas de la cámara: Las coordenadas homogéneas nos han permitido representar la proyección de la perspectiva con una matriz M (3x4).

- **Parámetros extrínsecos**

Consideremos ahora el caso donde el cuadro de la cámara (C) es distinto de cuadro del espacio (W).

$$ {}^{C}_{\cdot}P = ({}^{C}_{W}R \quad {}^{C}_{\cdot}O_W)\begin{pmatrix} {}^{W}_{\cdot}P \\ 1 \end{pmatrix} \qquad \text{Ec (2.8)} $$

Substituyendo en la ecuación (2.7) resulta

$$ p = \frac{1}{z}MP, \text{ donde } M = K(R \quad t) \qquad \text{Ec (2.9)} $$

$R = {}^{C}_{W}R$ es una matriz de rotación, $t = {}^{C}_{\cdot}O_W$ es un vector de traslación y P denota el vector de coordenadas homogéneas de P en el marco W.

Generalmente escribimos la ecuación de proyección de la perspectiva como $zp = MP$, lo que es más abusando un poco de la notación como $p = MP$. La matriz M está definida por 11 coeficientes libres. De los cuales 5 son intrínsecos (α, β, u_0, v_0 y θ) y los 6 restantes son extrínsecos (los tres ángulos que definen R y las tres coordenadas de t), esto coincide con el número de coeficientes independientes de M.

La matriz M puede ser reescrita explícitamente como una función de los parámetros intrínsecos y extrínsecos de la cámara.

$$ M = \begin{pmatrix} \alpha r_1^T - \alpha \cot\theta\, r_2^T + u_0 r_3^T & \alpha t_x - \alpha \cot\theta\, t_y + u_0 t_z \\ \frac{\beta}{\sin\theta} r_2^T + v_0 r_3^T & \frac{\beta}{\sin\theta} t_y + v_0 t_z \\ r_3^T & t_z \end{pmatrix} \qquad \text{Ec (2.10)} $$

donde r_1^T, r_2^T, y r_3^T denotan las tres filas de la matriz R y t_x, t_y y t_z son las coordenadas del vector t en el cuadro añadido a la cámara. Si R es escrita como el producto de tres rotaciones elementales, los vectores $r_i (i = 1,2,3)$ por supuesto que pueden ser escritos explícitamente en términos de los tres ángulos correspondientes.

2.4.2. Métodos de calibración

Hay varios métodos para estimar los parámetros intrínsecos y extrínsecos de una cámara. En esta oportunidad nos enfocaremos en el método de

calibración geométrica. Específicamente supongamos que una cámara observa n características geométricas tales como puntos o líneas con posiciones y dimensiones conocidas en un sistema de coordenadas fijo que pertenece a un espacio. En la figura 2-23 se puede apreciar el sistema de coordenadas y los puntos que van a ser usados para los cálculos de calibración de un cámara.

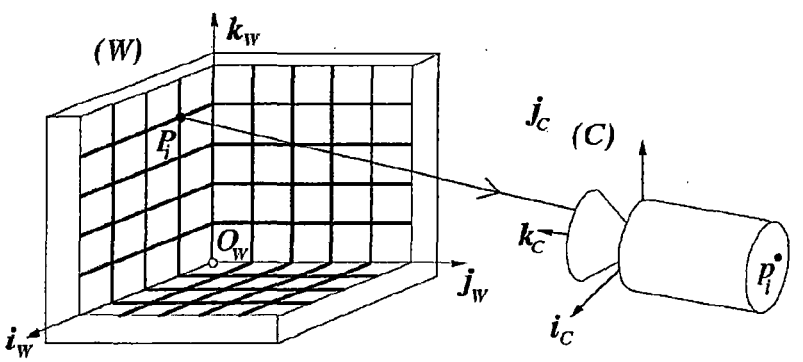


Figura 2-23. Esquema de calibración.

Trataremos el problema de calcular la matriz de proyección de la perspectiva M asociada con la cámara en ese sistema de coordenadas, calcular los parámetros intrínsecos y extrínsecos de cámara contenidos en esa matriz. Una vez que la cámara ha sido calibrada, es posible asociar cualquier punto de una imagen con un rayo que pasa a través del centro óptico de cámara para encontrar medidas tridimensionales de imágenes digitales.

- **Un enfoque lineal al problema de calibración de cámaras**

Asumamos primero que no hay inclinación y la matriz M es no singular. Si los cuatro vectores $P_i(i = 1,2,\dots,n)$ y $m_j^T(j = 1,2,3)$ denotan respectivamente los vectores de coordenadas homogéneas de los puntos P_i y las filas de la matriz M, nosotros podemos expresar la posición de la imagen de cada punto como:

$$\begin{cases} u_i = \dfrac{m_1 \cdot P_i}{m_3 \cdot P_i} \\ v_i = \dfrac{m_2 \cdot P_i}{m_3 \cdot P_i} \end{cases} \leftrightarrow \begin{cases} (m_1 - u_i \cdot m_3) \cdot P_i = 0 \\ (m_2 - v_i \cdot m_3) \cdot P_i = 0 \end{cases}$$

Colectando esas restricciones para todos los puntos produce un sistema homogéneo de $2n$ ecuaciones lineales en los doce coeficientes de la matriz M, es decir,

$$\mathcal{P}m = 0, \text{ donde } \mathcal{P} \stackrel{\text{def}}{=} \begin{cases} P_1^T & 0^T & -u_1 P_1^T \\ 0^T & P_1^T & -v_1 P_1^T \\ \cdots & \cdots & \cdots \\ P_n^T & 0^T & -u_n P_n^T \\ 0^T & P_n^T & -v_n P_n^T \end{cases} \text{ y } m \stackrel{\text{def}}{=} \begin{pmatrix} m_1 \\ m_2 \\ m_3 \end{pmatrix} = 0 \qquad \text{Ec (2.12).}$$

Cuando $n \geq 6$, el sistema de ecuaciones (2.12) es sobre restringido, no hay un vector diferente de cero $m \epsilon \mathrm{R}^{12}$ que satisfaga exactamente esas ecuaciones. Por otro lado, el vector cero es siempre una solución. La literatura de mínimos cuadrados provee métodos para calcular el valor del vector unitario m que minimice $|\mathcal{P}m|^2$. En particular, estimar el vector m (por consiguiente la matriz M) se reduce a calcular los autovectores y autovalores de la matriz $\mathcal{P}^T\mathcal{P}$ de 12×12.

- **Técnica de mínimos cuadrados**

Consideremos un sistema lineal de n ecuaciones con p incógnitas:

$$\begin{cases} u_{11} * x_1 + u_{12} * x_2 + \cdots + u_{1q} * x_p = b_1 \\ u_{21} * x_1 + u_{22} * x_2 + \cdots + u_{2q} * x_p = b_2 \\ \cdots \\ \cdots \\ \cdots \\ u_{n1} * x_1 + u_{n2} * x_2 + \cdots + u_{np} * x_p = y_n \end{cases} \leftrightarrow U * x = b \qquad \text{Ec (2.13)}$$

Denotaremos A la matriz $n \times p$ con coeficientes a_{ij}, $x = (x_1, \ldots, x_p)^T$ y $b = (b_1, \ldots, b_n)^T$. Sabemos del algebra lineal:

1. Cuando $n < p$, existe un espacio vectorial $(p - n)$ dimensional de vectores x que son las soluciones.
2. Cuando $n = p$, hay una única solución.
3. Cuando $n > p$ no hay solución.

Consideraremos el caso sobre restringido $n < p$. Ya que no hay una solución exacta en este caso, nos conformaremos con encontrar el vector x que minimiza la medida de error.

$$E \stackrel{\text{def}}{=} \sum_{i=1}^{n} (a_{i1}x_1 + \cdots + a_{ip}x_p - b_i)^2 = |Ax - b|^2$$

E es proporcional a la media cuadrática del error asociada con las ecuaciones, de ahí el nombre de método de mínimos cuadrados a la técnica que minimiza E.

Ahora, podemos escribir $E = |e^T e|$, donde $e \stackrel{\text{def}}{=} Ax - b$. Para encontrar el vector x que minimiza E, escribiremos que la derivada de la medida del error con respecto a las coordenadas $x_i (i = 1,2, \ldots, p)$ de x debe ser cero, es decir,

$$\frac{\partial E}{\partial x_i} = 2\frac{\partial e}{\partial x_i} \cdot e = 0, \text{para } i = 1, \ldots, p.$$

Pero si los vectores $c_i (i = 1, \ldots, p)$ denotan las columnas de A, tenemos

$$\frac{\partial e}{\partial x_i} = \frac{\partial}{\partial x_i}\begin{bmatrix} (c_1 & \cdots & c_p) & \begin{matrix} x_1 \\ \cdots \\ x_p \end{matrix} & -b \end{bmatrix} = \frac{\partial}{\partial x_i}(x_1 c_1 + \cdots + x_p c_p - b) = c_i$$

En particular, la restricción $\partial E / \partial x_i = 0$ implica que $c_i^T (Ax - b)$, y agrupando las restricciones asociadas con las coordenadas p de x resulta:

$$0 = \begin{pmatrix} c_1^T \\ \cdots \\ c_p^T \end{pmatrix} (Ax - b) = A^T (Ax - b) \leftrightarrow A^T A x = b.$$

Las ecuaciones en este sistema lineal son llamadas ecuaciones normales. Cuando A tiene un rango máximo p, la matriz $A^T A$ es fácil de ser invertida, y la solución del problema de los mínimos cuadrados puede ser escrito como:

$$x = A^{\dagger}b \text{ donde } A^{\dagger} \stackrel{\text{def}}{=} [(A^t A)^{-1}A^T].$$

La matriz $A^{\dagger}$ de $p \times p$ es llamada seudoinversa de A. Esta coincide con A^{-1} cuando la matriz A es cuadrada y no singular. Los problemas de mínimos cuadrados pueden ser resueltos sin calcular explícitamente la seudoinversa.

Se deduce que el vector unitario x que minimiza el error de mínimos cuadrados E es el autovector e_i asociado al menor autovalor de $A^T A$ y el menor valor correspondiente de E es λ_1.

Una vez que la matriz de proyección M ha sido estimada, esta puede ser usada para obtener los parámetros intrínsecos y extrínsecos de la siguiente forma. Si escribimos $M = (A \quad b)$, tenemos:

$$\rho \begin{pmatrix} a_1^T \\ a_2^T \\ a_3^T \end{pmatrix} = \begin{pmatrix} \alpha \cdot r_1^T + \alpha \cdot \cot(\theta) \cdot r_2^T + u_o r_3^T \\ \frac{\beta}{sen(\theta)} r_2^T + v_o \cdot r_3^T \\ r_3^T \end{pmatrix} \qquad \text{Ec (2.14)}$$

En particular, considerando el hecho de que las filas de la matriz de rotación tienen una unidad de longitud y son perpendiculares entre sí, obtenemos inmediatamente:

$$\begin{cases} \rho = \varepsilon / |a_3| \\ r_3 = \rho \cdot a_3 \\ u_o = \rho^2 (a_1 \cdot a_3) \\ v_o = \rho^2 (a_2 \cdot a_3) \end{cases} \qquad \text{Ec (2.15)}$$

donde $\varepsilon = \pm 1$.

Adicionalmente, tenemos:

$$\begin{cases} \rho^2 (a_1 \times a_3) = -\alpha r_2 - \alpha \cot\theta \, r_1 \\ \rho^2 (a_2 \times a_3) = \frac{\beta}{\sin\theta} r_1 \end{cases} \quad \text{y} \quad \begin{cases} \rho^2 |a_1 \times a_3| = \frac{|\alpha|}{\sin\theta} \\ \rho^2 |a_2 \times a_3| = \frac{|\beta|}{\sin\theta} \end{cases} \qquad \text{Ec (2.16)}$$

debido a que θ siempre tiene valores cercanos a $\frac{\pi}{2}$ con un seno positivo, se deduce que:

$$\begin{cases} \cos\theta = -\varepsilon_u\varepsilon_v \frac{(a_1\times a_3)\cdot(a_2\times a_3)}{|a_1\times a_3||a_2\times a_3|} \\ \alpha = \varepsilon_u\rho^2|a_1\times a_3|\,sin\,\theta \\ \beta = \varepsilon_v\rho^2|a_2\times a_3|\,sin\,\theta \end{cases} \qquad \text{Ec (2.17)}$$

donde $\varepsilon_u = {}^{\alpha}/_{|\alpha|}$ y $\varepsilon_v = {}^{\beta}/_{|\beta|}$.

Ahora podemos calcular r_1 y r_2 de la ecuación 11:

$$\begin{cases} r_1 = \frac{\rho^2\sin\theta}{\beta}(a_2\times a_3) = \frac{1}{|a_2\times a_3|}(a_2\times a_3) \\ r_2 = r_3\times r_1 \end{cases} \qquad \text{Ec (2.18)}$$

teniendo en cuenta que hay cuatro formas posibles de la matriz dependiendo de los valores de ε_u y ε_v.

Finalmente los parámetros de traslación son obtenidos de la siguiente forma:

$$\rho\begin{pmatrix} \alpha t_x - \alpha\cot\theta\, t_y + u_0 t_z \\ \frac{\beta}{\sin\theta}t_y + v_0 t_y \\ t_z \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix}$$

2.5. Geometría de Múltiples Vistas

A pesar de la riqueza de información contenida en una fotografía, la profundidad de un punto de una escena a lo largo de su correspondiente rayo de proyección no es directamente accesible. Con por lo menos dos imágenes, por otro lado, la profundidad puede ser medida a través de la triangulación. Esta es obviamente la razón por la cual la mayoría de animales tienen dos ojos o mueven su cabeza cuando están buscando algo. Primeramente debemos entender como

varias vistas de una misma escena restringen o definen su estructura tridimensional así como las correspondientes configuraciones de cámara.

2.5.1. Geometría epipolar para dos vistas

Consideramos las imágenes p y p' de un punto P observado por dos cámaras con dos centros ópticos O y O'. Estos cinco puntos pertenecen a un plano epipolar definido por la intersección de los rayos OP y $O'P$ (ver figura 2-24). En particular, el punto p' descansa en la línea l' donde este plano y la retina $\prod'$ de la segunda cámara interceptan. La línea l' es la línea epipolar asociada con el punto p, y pasa a través del punto e' donde la línea base une los centros ópticos O y O' intercepta al plano $\prod'$. También, el punto p descansa en la línea l epipolar asociada con el punto p', y esa línea pasa a través de la intersección e de la línea base con el plano $\prod$.

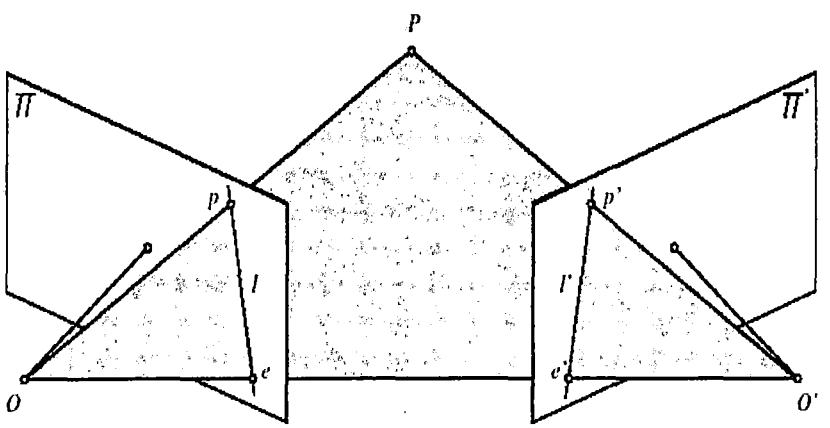


Figura 2-24. Geometría epipolar de dos vistas.

Los puntos e y e' son llamados epipolos en las dos cámaras. El epipolo e' es la imagen (virtual) del centro óptico O de la primera cámara en la imagen observada por la segunda cámara, y viceversa. Como se indicó antes, si p y p' son imágenes del mismo punto, entonces p' debe pertenecer a la línea epipolar asociada con p. Esta restricción epipolar juega un rol importante en la visión estéreo y el análisis de movimiento.

Si asumimos por ejemplo que conocemos los parámetros intrínsecos y extrínsecos de las dos cámaras de un equipo estéreo. Las coordenadas del punto p completamente determina el rayo que une O y p, y su plano epipolar asociado $OO'p$ y la línea epipolar. Las posibles correspondencias al punto p están restringidas a pertenecer a la asociada línea epipolar l', en lugar de toda la imagen (ver figura 2-25).

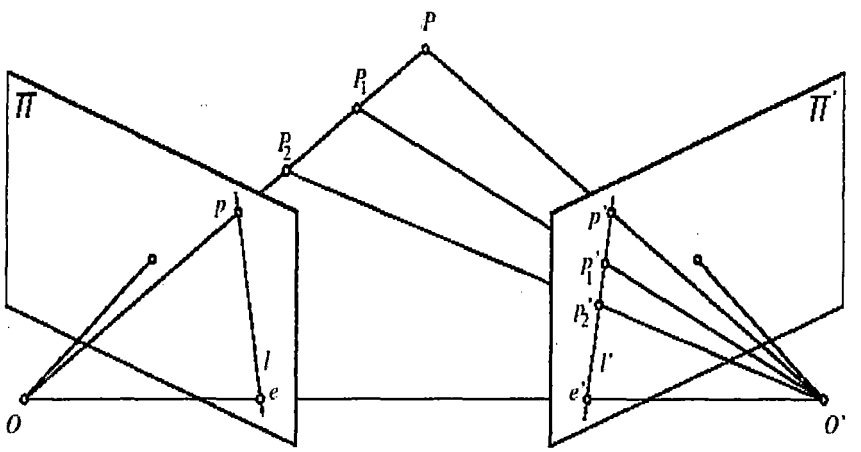


Figura 2-25. Restricción epipolar dado un sistema estéreo calibrado.

2.6. Visión Estéreo

Fusionar dos imágenes captadas por dos ojos y la diferencia (disparidad) entre ellas nos da una buena sensación de profundidad. La visión estéreo involucra dos procesos, la fusión binocular de las características observadas por dos ojos, y la reconstrucción tridimensional de estas características. Este último es relativamente simple: la pre imagen de correspondencia de puntos puede ser encontrada por la intersección de los rayos que pasan a través de esos puntos y los centros de pupila correspondiente (pinhole).

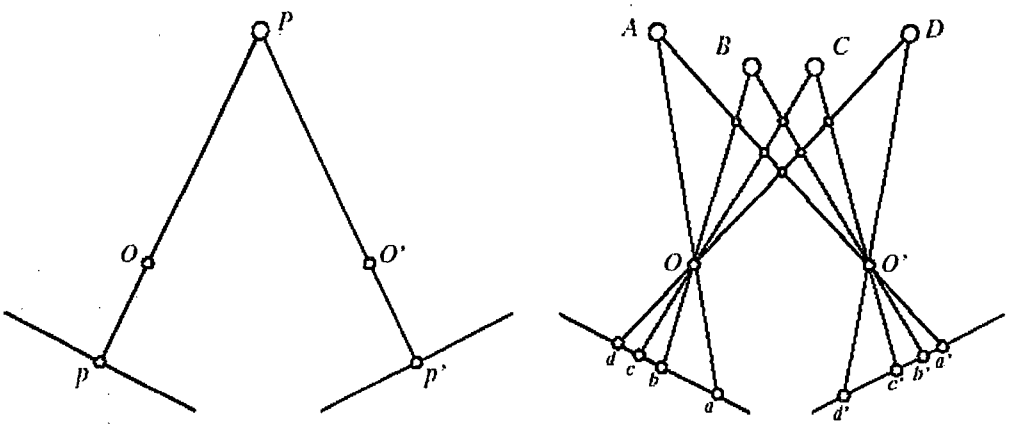


Figura 2-26. El problema de la fusión binocular.

En el diagrama izquierdo mostrado en el figura 2-26 no hay ambigüedad; por lo tanto la reconstrucción se vuelve simple. En el caso más usual (diagrama derecho), cualquiera de los cuatro puntos de la izquierda podrían relacionarse con cualquiera de los cuatro puntos de la derecha. Solo cuatro correspondencias son correctas.

2.6.1. Reconstrucción tridimensional: Triangulación

La triangulación en la visión por computador tiene diferentes aplicaciones siendo una de ellas la reconstrucción tridimensional de un punto a partir de dos o más imágenes. Hay varias formas de realizar una triangulación pero su fundamento se basa en geometría epipolar anteriormente explicada.

Después de lo estudiado anteriormente estamos en disposición de poder abordar la reconstrucción de la escena. La calidad de la reconstrucción que se obtendrá dependerá de la información que tengamos sobre el sistema de captura de las imágenes. Más en concreto, dependerá de nuestro conocimiento sobre los parámetros del sistema estéreo. Identificaremos tres casos:

a) Cuando se conocen tanto los parámetros intrínsecos como extrínsecos de la cámara.
b) Cuando solo se conocen los parámetros intrínsecos de la cámara

c) Cuando no se conocen ningún tipo de parámetros, es decir, solo conocemos el conjunto de puntos en correspondencia.

Nos enfocaremos en el primer caso en cual es posible reconstruir la escena de forma única usando triangulación sobre los puntos en correspondencia.

- **Conociendo los parámetros intrínsecos y extrínsecos**

Este es el caso más simple. Al conocer todos los parámetros del sistema la reconstrucción es directa. Como se muestra en la figura siguiente que muestra la geometría epipolar. El punto P es proyectado en un par de puntos en correspondencia $\hat{p}$ y $\hat{p}'$ se sitúa en la intersección de los dos rayos definidos por los centros de proyección O y O' y sus respectivos puntos de proyección $\hat{p}$ y $\hat{p}'$.

Aquí asumimos que los parámetros intrínsecos de cada cámara son conocidos, entonces $p = \hat{p}$. Claramente la restricción epipolar implica que los tres vectores $\overrightarrow{Op}$, $\overrightarrow{O'p'}$, y $\overrightarrow{OO'}$ son coplanares. Equivalentemente, uno de ellos debe pertenecer al plano abarcado por los otros dos.

$$\overrightarrow{Op} \cdot \left[\overrightarrow{OO'} \times \overrightarrow{O'p'}\right] = 0 \rightarrow \hat{p} \cdot [t \times R\hat{p}'] = 0 \qquad \text{Ec (2.19)}$$

Podemos reescribir la ecuación en el marco de coordenadas asociado a la primera cámara.

$$\hat{p} \cdot [t_{\times}]R\hat{p}' = \hat{p}^T \varepsilon \hat{p}' = 0 \qquad \text{Ec (2.20)}$$

donde $\hat{p} = (u, v, 1)^T$ es el vector de coordenadas homogéneas en la primera imagen de $\hat{p}$ y $\hat{p}' = (u', v', 1)^T$ es el vector de coordenadas homogéneas en la segunda imagen de $\hat{p}'$, t es el vector de coordenadas de traslación del punto O a O'. R es la matriz de rotación relacionada a ambas cámaras.

Las ecuaciones de dichos rayos pueden ser calculadas, y por tanto su intersección, pero el efecto del ruido sobre las coordenadas de los puntos de

proyección conducirá a que los rayos no se corten en el espacio como se muestra en la figura 2-27; por tanto su intersección deberá ser estimada como el punto del espacio de menor distancia a ambos rayos. La estimación de este punto es el núcleo del algoritmo de triangulación.

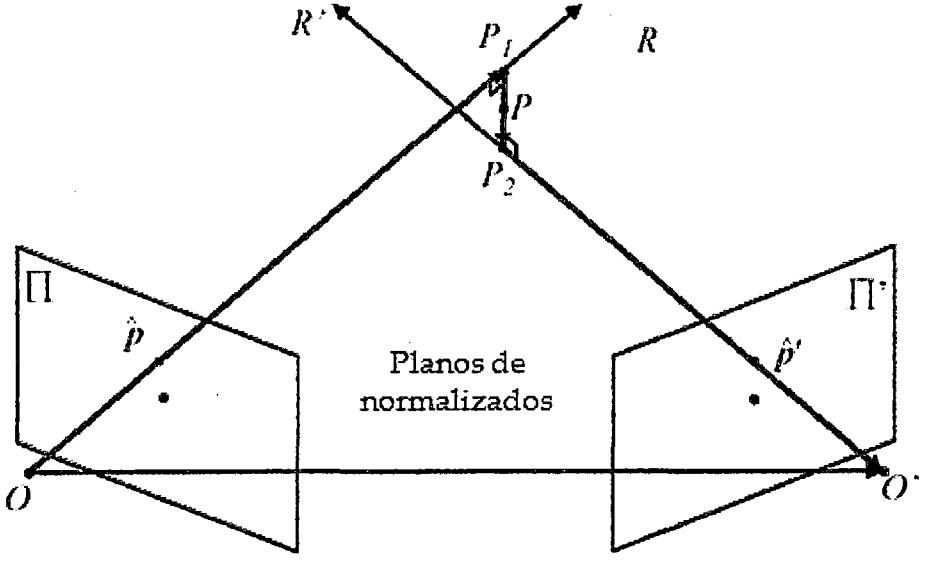


Figura 2-27. Triangulación de puntos.

Del modelo anterior se deduce las siguientes ecuaciones:

$$\overrightarrow{OP_1} + \overrightarrow{P_1P_2} + \overrightarrow{P_2O'} = \overrightarrow{OO'}$$

$$a\hat{p} + c(\hat{p} \times R\hat{p}') + bR\hat{p}' = t$$

$$aK^{-1}p + c(K^{-1}p \times RK'^{-1}p') + bRK'^{-1}p' = t$$

$$P = aK^{-1}p + \frac{c(K^{-1}p \times RK'^{-1}p')}{2}$$

Sea $a\hat{p}$, $(a \in R)$ un rayo a través de O y $\hat{p}$. Sea $bR\hat{p}'$, $(b \in R)$ un rayo a través de O' y $\hat{p}'$ expresado en el sistema de referencia izquierdo. Sea $c(\hat{p} \times R\hat{p}')$ un vector ortogonal a $a\hat{p}$ y $bR\hat{p}'$. El problema es determinar el punto medio del segmento $\overrightarrow{P_1P_2}$. La solución a este problema es muy simple ya que los puntos

extremos del segmento, digamos que $a\hat{p}$ y $t - bR\hat{p}'$ pueden calcularse resolviendo el sistema lineal de ecuaciones para a, b y c.

$$aK^{-1}p + c(K^{-1}p \times RK'^{-1}p') + bRK'^{-1}p' = t \qquad \text{Ec (2.21)}$$

Es importante señalar que el determinante del sistema anterior es siempre distinto cero salvo cuando los rayos son paralelos. También es posible calcular la reconstrucción directamente sobre las imágenes rectificadas en lugar de tener que usar las imágenes originales.

2.6.2. Correlación

Los métodos de correlación encuentran la correspondencia de pixeles entre dos imágenes comparando la intensidad en la vecindad de potenciales elementos correspondientes, y es una de las primeras técnicas propuestas para resolver el problema de la fusión binocular. Más precisamente, consideremos un par estéreo rectificado y un punto (u, v) en la primera imagen. Asociamos con el tamaño de la ventana centrado $p = (2m + 1) \times (2n + 1)$ en (u, v) el vector $w(u, v) \in \mathbb{R}^p$ obtenido escaneando los valores de la ventana una fila a la vez. Ahora dado el potencial elemento correspondiente $(u + d, v)$ en la segunda imagen, podemos construir un segundo vector $w'(u + d, v)$ y definir la correspondiente función de correlación normalizada como:

$$C(d) = \frac{1}{|w - \bar{w}|}\frac{1}{|w' - \bar{w}'|}(w - \bar{w}) \cdot (w' - \bar{w}'),$$

Donde los índices u, v y d han sido omitidos para una buena concisión y $\bar{a}$ denota el vector cuyas coordenadas son iguales a la media de las coordenadas de a (ver figura 2-28).

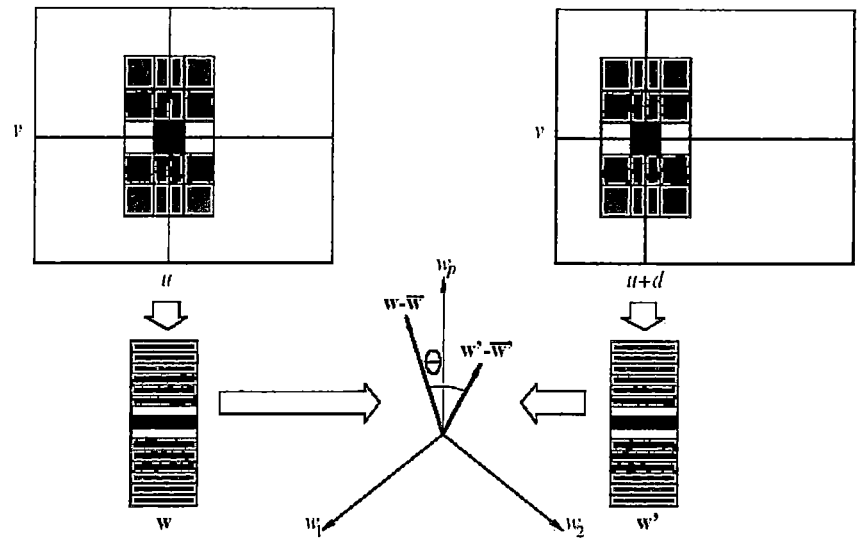


Figura 2-28. Correlación de dos ventanas a lo largo de las líneas epipolares.

La posición de la segunda ventana está separada de la primera por un desplazamiento d. Las dos ventanas están codificadas por dos vectores w y w' en $\mathbb{R}^{15}$, y la correlación mide el coseno del ángulo θ entre los vectores $w - \overline{w}$ y $w' - \overline{w}'$ obtenido substrayendo de los componentes de w y w' el promedio de intensidad en las correspondientes ventanas.

La función de correlación normalizada C tiene rango de -1 a +1, y este alcanza su máximo valor cuando los brillos (luminosidad) de las dos ventanas son relacionados por una transformación $I' = \lambda I + u$ con algunas constantes λ y u para $\lambda > 0$. En otras palabras, el máximo de esa función corresponde un elemento de la imagen separado por un desplazamiento constante y un factor de escala positivo, y el par estéreo correspondiente puede ser encontrado buscando el máximo de la función C sobre el predeterminado rango de disparidades.

En este punto, haremos algunas aclaraciones acerca de métodos de correspondencia basados en correlación. Primeramente, es fácil de mostrar que la maximización de la función de correlación es equivalente a la minimización de la norma de la diferencia entre los vectores $(1/|w - \overline{w}|)(w - \overline{w})$ y $(1/|w' - \overline{w}'|)(w' - \overline{w}')$, o lo que es equivalente la suma de diferencias al

cuadrado entre los valores de los pixeles de la ventana normalizada a ser comparada. Como segundo punto, a pesar de que el cálculo de la función de correlación normalizada por cada pixel de la una imagen para algunos rangos de disparidad tiene un costo computacional excesivo, puede ser implementado usando técnicas recursivas. Finalmente, el mayor problema con las técnicas basadas en la correlación para establecer correspondencias estéreo es que estas implícitamente asumen que la superficie observada es localmente paralela a los dos planos de las imágenes. El escorzo de dos superficies no paralelas es diferente para dos cámaras, tal y como se muestra en la figura 2-29.

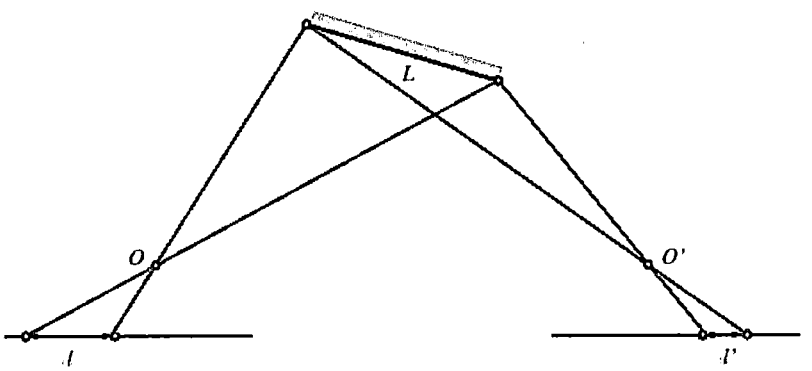


Figura 2-29. El escorzo de dos superficies no paralelas.

CAPÍTULO III

ALGORITMOS INTELIGENTES PARA VISIÓN ARTIFICIAL

A continuación se muestran los fundamentos teóricos de tres algoritmos de aprendizaje que se usaron para predecir o inferir la presencia de un punto de agarre en la imagen de un objeto. Estos algoritmos, aunque se basan en conceptos un tanto diferentes, tienen la característica de lograr un aprendizaje a través de una data de entrenamiento y presentar buenos resultados cuando se presentan nuevos datos. Estos dos algoritmos han sido ampliamente usados para resolver diversos problemas de clasificación y modelamiento de datos.

En el siguiente capítulo se compararán estos algoritmos de acuerdo resultado en las pruebas realizadas tiempos de entrenamiento a través de los cálculos y resultados donde se compararán directamente con datos de entrenamiento y datos reales evaluando principalmente su capacidad de generalización.

3.1. Modelos Lineales Generalizados

Los modelos lineales generalizados (GLM, Generalized Linear Models) tienen como objetivo describir el efecto de una o más variables explicativas o independientes sobre una o más variables respuesta o dependientes (ver figura 3-1).

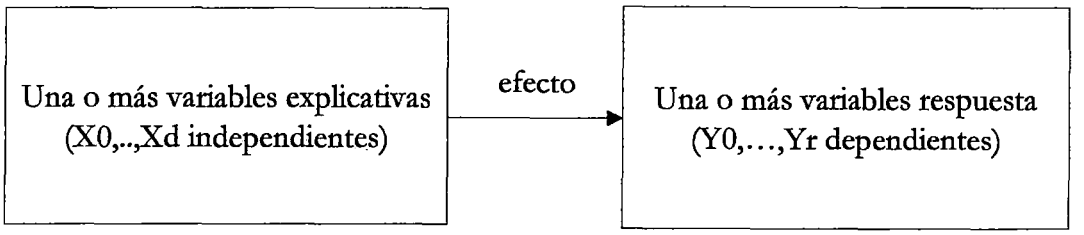


Figura 3-1. Diagrama de variables.

Las componentes del vector Y son variables con distribución proveniente de una familia exponencial. Las variables $X_0, \ldots, X_d$ originan un predictor lineal η dado por $\eta = \beta_0 + \beta_1 X_1 + \ldots + \beta_d X_d$, o en forma matricial $\eta = A\beta$.

El predictor lineal y la variable dependiente están relacionados por una función de enlace o link g, siendo g monótona y diferenciable. Podemos escribir el modelo en cualquiera de sus 3 formas:

$$g(\mu) = \beta_0 + \beta_1 X_1 + \ldots + \beta_d X_d$$

$$g(\mu_i) = \beta_1 X_{i1} + \beta_2 X_{i2} + \ldots + \beta_{ip} X_{ip} = \eta_i; \ \forall \ i = 1, \ldots, n$$

$$g(\mu) = g\begin{pmatrix} X_{11} & \cdots & X_{1p} \\ X_{21} & \cdots & X_{2p} \\ \cdots & \cdots & \cdots \\ X_{n1} & \cdots & X_{np} \end{pmatrix}\begin{pmatrix} \beta_1 \\ \cdots \\ \beta_p \end{pmatrix} = A\beta = \eta$$

Familia exponencial: una variable aleatoria Y tiene distribución proveniente de una familia exponencial si su función de probabilidad puntual o de densidad es de la forma:

$$f(y, \theta) = e^{a(y)b(\theta)+c(\theta)+d(y)}$$

Si $a(y) = y$ se dice que f está dada en su forma canónica. Ejemplos: Poisson, Normal y Binomial.

Un GLM queda especificado mediante tres componentes:

- Componente aleatoria: Es la distribución de probabilidad de la variable respuesta Y que pertenece a la familia exponencial natural.
- Componente sistemática: Es una función lineal de las variables explicativas que se usa como predictor lineal.
- Función de enlace o ligadura: Es la función g que describe la relación funcional entre la componente sistemática y el valor esperado de la componente aleatoria.

Componente Aleatoria	Función de enlace	Componente Sistemática	Modelo
Normal	Identidad	Continuo	Regresión
Normal	Identidad	Categórico	Análisis de varianza
Normal	Identidad	Mixto	Análisis de covarianza
Binomial	Logit	Mixto	Regresión logística
Poisson	Log	Mixto	Log lineal
Multinomial	Logit generalizado	Mixto	Respuesta multinomial

Tabla 3-1. Tipos de modelos lineales generalizados.

La tabla 3-1 describe los diferentes tipos de modelos lineales generalizados y sus correspondientes componentes y funciones. En la siguiente sección trataremos el tema de la regresión logística con mayor detalle.

3.1.1. Regresión logística

La regresión logística es un caso particular de los modelos lineales generalizados y es usada para la predicción de la probabilidad de ocurrencia de un evento modelando la data en la curva de una función logística. Esto es un modelo lineal generalizado para la regresión binomial. Como muchas formas de análisis por regresión, esta hace uso de varias variables predictivas que pueden ser numéricas o categóricas. Por ejemplo la probabilidad de que una persona tenga un paro cardiaco en periodo específico de tiempo puede ser predicho conociendo su edad, sexo y el índice de masa del cuerpo. La regresión es usada extensivamente en campos como la medicina y las ciencias sociales así como también en el marketing para predecir el comportamiento de los consumidores.

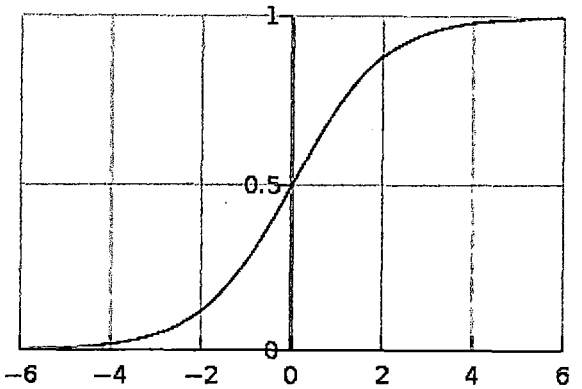


Figura 3-2. Función Logística con z en el eje horizontal y $f(z)$ en el eje vertical.

En la figura 3-2 se muestra la gráfica de la función logística para un intervalo de -6 a 6. Se observar de la gráfica que esta función toma valores entre cero y uno. La explicación de la regresión logística comienza con la explicación de la función logística:

$$f(z) = \frac{e^z}{e^z + 1} = \frac{1}{1 + e^{-z}}$$

La entrada es z y la salida es $f(z)$. La función logística es útil porque esta puede tomar como entrada cualquier valor desde el infinito negativo hasta el infinito positivo sin embargo la salida es acotada entre los valores 0 y 1. La variable z representa la exposición de un conjunto de variables independientes, mientras que $f(z)$ representa la probabilidad de un evento particular dado el conjunto de variables. La variable z es una medida de la contribución total de todas las variables independientes usadas en el modelo.

La variable z es usualmente definida como:

$$z = \beta_0 + \beta_1 * x_1 + \beta_2 * x_2 + \beta_3 * x_3 \ldots + \beta_k * x_k;$$

Donde β_0 es llamado el intercepto y β_1, β_2, β_3,..., β_k son llamados los coeficientes de regresión de x_1, x_2, x_3, ..., x_k respectivamente. El intercepto es el valor de z cuando el valor de todas la variables independientes es cero. Cada uno de los coeficientes de regresión describe el tamaño de contribución de las

variables independientes. Un positivo coeficiente de regresión significa que dicha variable incrementa la probabilidad del evento, si es negativo significa que la variable disminuye la probabilidad del evento. Regresión logística es una forma útil de describir la relación entre la una o más variables independientes y una variable binaria de respuesta y es expresada como probabilidad (0 ó 1).

- **La eficiencia depende del tamaño de la muestra**

La regresión logística tiende a sistemáticamente sobreestimar las razones de probabilidades o los coeficientes beta en pequeños y moderados número de muestras (muestras < 500 aproximadamente). Si se incrementa las muestras la magnitud de sobreestimación disminuye y la estimación de las razones de probabilidades asintóticamente alcanza el valor verdadero de la población. Sin embargo se ha concluido que esa sobreestimación puede en un solo estudio no tener relevancia en la interpretación de los resultados siempre que esta sea mucho menor que el error estándar de estimación.

Se recomienda un mínimo de 10 eventos por variable independiente. Por ejemplo, en un estudio donde el deceso de un persona es el evento de interés, y hubieron 50 decesos de un total de 100 pacientes, el número de variables independientes que puede soportar el modelo es 50/10=5.

- **Especificaciones matemáticas**

La regresión logística analiza data distribuida binomialmente de la forma.

$$Y_i \sim B(n_i, p_i), para\ i = 1,2, \dots, m;$$

Donde el número de intentos de Bernoulli n_i son conocidos y la probabilidad de exito pi es desconocida. Un ejemplo de esta distribución es la fracción de semillas (p_i) que germinan después n_i de ser sembradas.

El modelo propone que para cada intento hay un conjunto de variables explicativas que quizás influya en la probabilidad final. Esas variables

explicativas pueden ser consideradas como un vector X_i de k dimensiones y el modelo tiene la forma

$$p_i = E\left(\frac{Y_i}{n_i}, X_i\right).$$

La función ***logit*** de las probabilidades binomiales desconocidas son modeladas como una función lineal de X_i.

$$logit(p_i) = \ln\left(\frac{p_i}{1-p_i}\right) = \beta_0 + \beta_1 * x_{1,i} + \beta_2 * x_{2,i} + \beta_3 * x_{3,i} \ldots + \beta_k * x_{k,i};$$

Nótese que un elemento particular de X_i ser establecido como 1 para todos los valores i y producir una intercepción en el modelo. Los parámetros desconocidos son usualmente estimados con el principio de máxima verosimilitud usando un método común a todos los modelos lineales generalizados. Las estimaciones de máxima verosimilitud pueden ser calculadas numéricamente usando mínimos cuadrados ponderados iterativamente.

La interpretación de los parámetros estimados β_j es la de un efecto aditivo en el logaritmo de las probabilidades para un cambio unitario en la variable explicativa j.

El modelo tiene una formulación equivalente.

$$p = \frac{1}{1 + e^{-(\beta_0 + \beta_1 * x_{1,i} + \beta_2 * x_{2,i} + \beta_3 * x_{3,i} \ldots + \beta_k * x_{k,i})}}$$

Esta forma funcional es comúnmente llamada perceptron de una sola capa o red neuronal de una sola capa. La red neuronal de una sola capa calcula una salida continua en lugar de una función Step. La derivada de p_i con respecto a $X = x_1, \ldots, x_k$ es calculada en forma general de:

$$y = \frac{1}{1 + e^{-f(x)}}$$

Donde *f*(*X*) es una función analítica en *X*. Con esta elección, la neurona de una sola capa es idéntica al modelo de regresión logística. Esta función tiene una derivada continua la cual permite ser usada en algoritmos como Backpropagation. Esta función es preferida porque su derivada es fácil de calcular:

$$y' = y * (1 - y) * \frac{df}{dX}$$

- **Estimación de coeficientes por el método de máxima verosimilitud**

Si la función de distribución de Y pertenece a una familia de distribuciones H conocida, un método alternativo para estimar el vector de parámetros desconocidos de β es el método de máxima verosimilitud.

$$\min_{\beta \in \mathbb{R}} S(\beta) = (Y - X\beta)' V^{-1} (Y - X\beta) \qquad \text{Ec (3.1)}$$

Dado un vector de observaciones $y' = (y_1, \dots, y_n)$ la función de verosimilitud cuantifica la posibilidad (verosimilitud) de que un vector $\beta \epsilon R^p$ haya generado el vector de respuestas observado.

La función de verosimilitud está dada por la función de densidad conjunta de las variables aleatorias independientes $Y_1, \dots Y_n$ reduciéndose la expresión a:

$$L(\beta) = h(y_1, \dots, y_n) = h(y_1, \beta) h(y_2, \beta) \dots h(y_k, \beta) = \prod_{i=1}^{n} h(y_i; \beta) \qquad \text{Ec (3.2)}$$

El estimador de máxima verosimilitud de β es el vector b que maximiza $L(\beta)$ en el espacio paramétrico esto $\Omega = \{(\beta, \sigma^2) : \beta \epsilon R^p, \sigma^2 > 0\}$ esto es $L(b) \geq L(\beta)\ \forall\ \beta \in \Omega$.

Para obtener el estimador máximo verosímil necesitamos resolver el problema de maximizar $L(\beta)$ para $\beta \epsilon R^p$. Dado que la función logaritmo es monótona, aplicando logaritmo a la ecuación (3.2) se tiene:

$$I(\beta) = \log L(\beta) = \sum_{i=1}^{n} \log h(y_i; \beta)$$

En consecuencia, si la función $I(\beta)$ es continua y derivable, maximizar $L(\beta)$ o $I(\beta)$ son procesos equivalentes.

Para ilustrar, obtendremos el estimador de máxima verosimilitud para el caso del modelo lineal general, bajo la suposición que los errores se distribuyen normalmente con vector de medias cero y matriz de covarianza V. La función de verosimilitud será:

$$L = 2\pi^{-n/2}|V|^{-1/2}\exp\{-\frac{1}{2}(Y - X\beta)'V^{-1}(Y - X\beta)\}$$

Y su función de soporte es:

$$I(\beta) = -\frac{n}{2}\log\frac{2\pi}{2} - \frac{1}{2}\log|V| - \frac{1}{2}(Y - X\beta)'V^{-1}(Y - X\beta)$$

$I(\beta)$ es una función continua y derivable, por lo tanto, derivando e igualando a cero la expresión anterior se obtiene:

$$X'V^{-1}Xb = X'V^{-1}Y.$$

La solución de este sistema de ecuaciones nos conduce al estimado M.V. de β:

$$b = (X'V^{-1}X)^{-1}X'V^{-1}Y \qquad \text{Ec (3.3)}$$

3.2. Redes Neuronales Artificiales

Una de las ramas más destacadas del campo de la inteligencia artificial es la que corresponde a las Redes Neuronales Artificiales entendiendo como tales

aquellas redes en las existen procesadores de información de cuyas interacciones locales depende el comportamiento del conjunto del sistema.

Las redes neuronales artificiales tratan de emular el comportamiento del cerebro humano, caracterizado por el aprendizaje a través de la experiencia y la extracción de conocimiento genérico a partir de un conjunto de datos. Estos sistemas imitan esquemáticamente la estructura neuronal del cerebro, bien mediante un programa ordenador (simulación), bien mediante su modelo a través de estructuras de procesamiento con cierta capacidad del cálculo paralelo (emulación), o bien mediante la construcción física de sistemas cuya arquitectura se aproxima a la estructura de la red neuronal biológica (implementación).

3.2.1. Fundamentos de las redes neuronales artificiales

La teoría y modelado de redes neuronales artificiales está inspirada en la estructura y funcionamiento de los sistemas nerviosos, donde la neurona es el elemento fundamental. Existen neuronas de diferentes formas, tamaños y longitudes. Estos atributos son importantes para determinar la función y utilidad de la neurona.

- **Elementos de una red neuronal artificial**

Las redes neuronales artificiales son los modelos que intentan reproducir el comportamiento del cerebro. Como tal modelo, realiza una simplificación, averiguando cuáles son los elementos relevantes del sistema, bien porque la cantidad de información es excesiva o bien porque es redundante. Una elección adecuada de sus características, más que una estructura conveniente, es procedimiento convencionalmente utilizado para reconstruir redes capaces de realizar una determinada tarea.

Cualquier modelo de red neuronal consta de dispositivos elementales de proceso: las neuronas. A partir de ellas, se puede generar representaciones específicas, de tal forma que un estado conjunto de ellas puede significar una letra, un número o cualquier otro objeto. En el siguiente apartado se realiza la

idealización del funcionamiento neurobiológico de una neurona, que sirve de base de las redes neuronales artificiales (RNA). Generalmente se puede encontrar tres tipos de neuronas.

1. Aquellas que reciben estímulos externos, relacionadas con el aparato sensorial, que tomarán la información de entrada.
2. Dicha información se transmite a ciertos elementos internos que se ocupan de su procesado. Es en las sinapsis y neuronas correspondientes a este segundo nivel donde se genera cualquier tipo de representación interna de la información. Puesto que no tiene relación directa con la información de entrada ni con la salida, estos elementos se denominan unidades ocultas.
3. Una vez finalizado el periodo de procesado, la información llega a las unidades de salida, cuya misión es dar la respuesta al sistema.

La neurona artificial pretende mimetizar las características más importantes de las neuronas biológicas. Cada neurona *i*-ésima está caracterizada en cualquier instante por un valor numérico denominado *valor o estado de activación* $a_i(t)$; asociado a cada unidad, existe una *función de salida,* f_i, que transforma el estado actual de activación en una señal de salida y_i. Dicha señal es enviada a través de los canales unidireccionales a otras unidades de la red; en estos canales la señal modifica de acuerdo con la sinapsis (el peso, w_{ji}) asociada a cada uno de ellos según una determinada regla. Las señales moduladas que han llegado a la unidad *j*-ésima se combinan entre ellas, generando así la entrada total, Net_j.

$$Net_j = \sum_i y_i w_{ji}$$

Una función de activación, F, determina el nuevo estado de activación de la neurona $a_j(t+1)$, teniendo en cuenta la entrada total calculada y el anterior estado de activación $a_j(t)$.

La dinámica que rige la actualización de los estados de las unidades (evolución de la red neuronal) puede ser de dos tipos: modo asíncrono y modo síncrono. En el primer caso, las neuronas evalúan su estado continuamente, según les va llegando la información, y lo hacen de forma independiente. En el segundo caso síncrono, la información también llega de forma continua, pero los cambios se realizan simultáneamente, como si existiera un reloj interno que decidiera cuándo deben cambiar su estado. Los sistemas biológicos quedan probablemente entre ambas posibilidades.

En los siguientes apartados se describirán los conceptos básicos de los distintos componentes de un modelo de red neuronal artificial: tipos de unidades de proceso o neuronas, su estado o activación, su función de salida y de activación, conexión con otras neuronas y concepto de aprendizaje de una red neuronal.

a) Unidad de proceso: La neurona artificial

Si se tienen N unidades (neuronas), podemos ordenarlas arbitrariamente y designar la *j*-ésima unidad como U_j. Su trabajo es simple y único, y consiste en recibir las entradas de las células vecinas y calcular un valor de salida, el cual es enviado a todas las células restantes.

En cualquier sistema que se esté modelando, es útil caracterizar tres tipos de unidades: entradas, salidas y ocultas. Las unidades de entrada reciben las señales desde el entorno; estas entradas (que son a la vez entradas a la red) pueden ser señales provenientes de sensores o de otros sectores del sistema. Las unidades de salida envían la señal fuera del sistema (salida de la red); estas señales pueden controlar directamente potencias u otros sistemas. Las unidades ocultas son cuyas entradas y salidas se encuentran dentro del sistema; es decir, no tienen contacto con el exterior.

Se conoce como capa o nivel a un conjunto de neuronas cuyas entradas provenientes de la misma fuente (puede ser otra capa de neuronas)

y cuyas salidas se dirigen al mismo destino (que puede ser otra capa de neuronas).

b) Estado de activación

Adicionalmente al conjunto de unidades, la representación necesita los estados del sistema en un tiempo t. Esto se especifica por un vector de *N* números reales $A(t)$, que representa el *estado de activación* del conjunto de unidades de procesamiento. Cada elemento del vector representa la activación de una unidad en el tiempo t. La activación de una unidad U_i en el tiempo t se designa por $a_i(t)$; es decir:

$$A(t) = (a_1(t), a_2(t), \dots, a_i(t), \dots, a_N(t))$$

El proceso que realiza la red se ve como la evolución de un patrón de activación en el conjunto de unidades que lo componen a través del tiempo.

Todas las neuronas que componen la red se hallan en cierto estado. En una visión simplificada, podemos decir que hay dos posibles estados, *reposo* y *excitado*, a los que denominaremos globalmente *estados de activación*, y cada uno de los cuales se le asigna un valor. Los valores de activación pueden ser continuos o discretos. Además, pueden ser limitados o ilimitados. Si son discretos, suelen tomar un conjunto pequeño de valores o bien valores binarios. En notación binaria, un estado activo se indicaría por un 1, y se caracteriza por la emisión de un impulso por parte de la neurona (potencial de activación), mientras que un estado pasivo se indicaría por un 0, y significaría que la neurona está en reposo. En otros modelos se considera un conjunto continuo de estados de activación, en lugar de solo dos estados, en cuyo caso se les asigna un valor entre [0,1] o en el intervalo [-1,1], generalmente siguiendo una función de sigmoidal.

Finalmente, es necesario saber qué criterios o reglas siguen las neuronas para alcanzar tales estados de activación. En principio, esto va a depender de dos factores: a) Por un lado, puesto que las propiedades

macroscópicas de las redes neuronales no son producto de actuación de elementos individuales, sino del conjunto como un todo, es necesario tener idea del mecanismo de iteración entre las neuronas. El estado de activación estará fuertemente influenciado por tales iteraciones, ya que el efecto que producirá una neurona sobre otra será proporcional a la fuerza, peso o magnitud de la conexión entre ambas. b) Por otro lado, la señal que envía cada una de las neuronas a sus vecinas dependerá de su propio estado de activación.

c) Función de salida o de transferencia

Entre las unidades o neuronas que forman una red neuronal artificial existe un conjunto de conexiones que unen unas con otras. Cada unidad transmite señales a aquellas que están conectadas con su salida. Asociada con cada unidad U_i hay una función de salida $f_i(a_i(t))$ que transforma el estado actual de activación $a_i(t)$ en una señal de salida $y_i(t)$ es decir:

$$y_i(t) = f_i(a_i(t))$$

El vector que contiene las salidas de todas las neuronas en un instante t es:

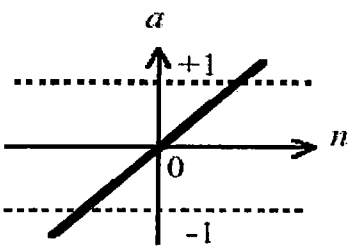
$$Y(t) = (f_1(a_1(t)), f_2(a_2(t)), \dots, f_i(a_i(t)), \dots f_N(a_N(t)))$$

En algunos modelos, esta salida es igual al nivel de activación de la unidad, en cuyo caso la función f_i, es la función identidad, $f_i(a_i(t)) = a_i(t)$. A menudo, f_i es de tipo sigmoidal, y suele ser la misma para todas las unidades.

Las funciones de transferencia típicas que se usan en redes neuronales multicapa son:

La función ***lineal o identidad***, que devuelve directamente el valor de activación de la neurona. Este tipo de función se utiliza en redes de baja

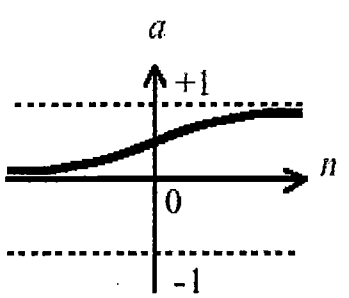
complejidad, como el modelo Adaline. Su gráfica se puede apreciar en la figura 3-3.



$a = purelin(n)$

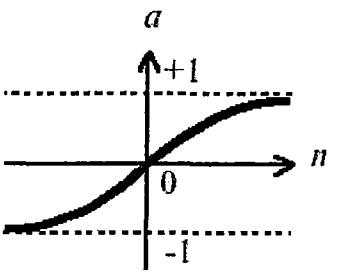
Figura 3-3. Gráfica de la función lineal.

La función ***sigmoidea***, definida en un determinado intervalo monotónico con límites superiores e inferiores. Entre las funciones sigmoideas de transferencia más aplicadas destacan la función sigmoide o logística (ver figura 3-4), la función tangente hiperbólica (ver figura 3-5), y la función sigmoide modificada. Las funciones sigmoideas se caracterizan por presentar una derivada siempre positiva e igual a cero en sus límites asintóticos, que toma su valor máximo cuando $x = 0$. Así, estas funciones admiten la aplicación de las reglas de aprendizaje típicas de la función escalón, con la ventaja adicional que la derivada se encuentra definida en todo el intervalo, lo que permite emplear algoritmos de entrenamiento más avanzados.



$a = logsig(n)$

Figura 3-4. Gráfica de la función sigmoide o logística.



$$a = tansig(n)$$

Figura 3-5. Gráfica de la función tangente hiperbólica.

d) Conexiones entre neuronas

Las conexiones que unen a las neuronas que forman una RNA tienen asociado un peso, que es el que hace que la red adquiera conocimiento. Consideremos y_i como el valor de la salida de una neurona en un instante dado. Una neurona recibe un conjunto de señales que le dan la información del estado de activación de todas las neuronas con las que se encuentra conectada. Cada conexión (sinapsis) entre la neurona i y la neurona j está ponderada por un peso w_{ji}. Normalmente, como simplificación, se considera que el efecto de cada señal es aditivo, de tal forma que la entrada neta que recibe una neurona (potencial postsináptico) net_j es la suma del producto de cada señal individual por el valor de la sinapsis que conecta ambas neuronas:

$$net_j = \sum_{i}^{N} w_{ji} y_i$$

Esta regla muestra el procedimiento a seguir para combinar los valores de entrada a una unidad con los pesos de las conexiones que llegan a esa unidad y es conocida como la regla de propagación.

Suele identificarse una matriz W con todos los pesos w_{ji} que reflejan la influencia sobre la neurona j tiene la neurona i. W es el conjunto de elementos positivos, negativos o nulos. Si w_{ji} es positivo, indica que la interacción entre las neuronas i y j es excitadora; es decir siempre que la

neurona i esté activada, la neurona j recibirá una señal que tenderá a activarla. Si w_{ji} es negativo la sinapsis es inhibidora. En este caso, si i está activada, enviará una señal a j, que tenderá a desactivar a ésta. Finalmente si $w_{ji} = 0$, se supone que no hay conexión entre ambas.

e) Funciones o reglas de activación

Así como es necesaria una regla de que combine las entradas a una neurona con los pesos de las conexiones, también se requiere una regla que combine las entradas con el estado actual de la neurona para producir un nuevo estado de activación. Esta función F produce un nuevo estado de activación en una neurona a partir del estado (a_i) que existía y la combinación de las entradas con los pesos de las conexiones (net_i).

Dado el estado de activación $a_i(t)$ de la unidad U_i y la entrada total que llega a ella, Net_i, el estado de activación siguiente, $a_i(t+1)$, se obtiene aplicando una función F llamada función de activación.

$$a_i(t+1) = F(a_i(t), Net_i)$$

En la mayoría de casos, F es la función identidad, por lo que el estado de activación de una neurona en $t+1$ coincidirá con el Net de la misma en t. En este caso, el parámetro que se le pasa a la función de salida, f, de la neurona será directamente el Net. El estado de activación no se tiene en cuenta. Según esto la salida de una neurona $i(y_i)$ quedará según la expresión:

$$y_i(t+1) = f(Net_i) = f(\sum_{j=1}^{N} w_{ij} y_j(t))$$

Por tanto y en lo consecutivo, consideraremos únicamente la función f, que denominaremos indistintamente de transferencia o activación. Además, normalmente la función de activación no está centrada en el origen del eje que representa al valor de entrada neta, sino que existe cierto desplazamiento debido a las características internas de la propia neurona y que no es igual en

todas ellas. Este valor se denota como θ_i y representa el umbral de activación de la neurona i.

$$y_i(t+1) = f(Net_i - \theta_i) = f(\sum_{j=1}^{N} w_{ij} y_j(t) - \theta_i)$$

f) Regla de aprendizaje

Existen muchas definiciones del concepto general de aprendizaje, una de ellas podría ser: *La modificación del comportamiento inducido por la interacción con el entorno y como resultado de experiencias conducente al establecimiento de nuevos modelos de respuesta a estímulos externos.* Esta definición fue enunciada muchos años antes de que surgieran las redes neuronales, sin embargo puede ser aplicada también a los procesos de aprendizaje de estos sistemas.

Biológicamente, se suele aceptar que la información memorizada en el cerebro está más relacionada con los valores sinápticos de las conexiones entre las neuronas que con ellas mismas; es decir, el conocimiento se encuentra en las sinapsis. En el caso de las redes neuronales artificiales, se puede considerar que el conocimiento se encuentra representado en los pesos de las conexiones entre neuronas. Todo proceso de aprendizaje implica cierto número de cambios en estas conexiones. En realidad, puede decirse que se aprende modificando los valores de los pesos de la red.

Al igual que el funcionamiento de una red depende del número de neuronas de la que disponga y cómo estén conectadas entre sí, cada modelo dispone de su o sus propias técnicas de aprendizaje.

- **Estructura de una red neuronal artificial**

Una vez presentados en el apartado anterior los componentes más importantes de una red neuronal centrados, sobre todo, en las características de

cada nodo de la red, veamos ahora como está organizada dicha red en función de:

- Número de niveles o capas.
- Formas de conexión entre neuronas.

a) Niveles de capas o neuronas

La distribución de neuronas dentro de la red se realiza formando niveles o capas de un número determinado de neuronas cada una. A partir de su situación dentro de la red, se puede distinguir tres tipos de capas:

- De entrada: Es la capa que recibe directamente la información proveniente de las fuentes externas a la red.
- Ocultas: Son internas a la red y no tiene contacto directo con el entorno exterior. El número de niveles ocultos puede estar entre cero y un número no muy elevado. Las neuronas de las capas ocultas pueden estar interconectadas de distintas maneras, lo que determina, junto con su número, las distintas topologías de las redes neuronales.
- De salida: Transfieren información de la red hacia el exterior.

En la figura 3-6 se muestra el esquema de la estructura de una posible red multicapa en la que cada nodo o neurona únicamente está conectada con neuronas de un nivel superior. Nótese que hay muchas más conexiones que nodos. En este sentido, se dice que una red es totalmente conectada si todas las salidas desde un nivel llegan a todos y cada uno de los nodos del nivel siguiente.

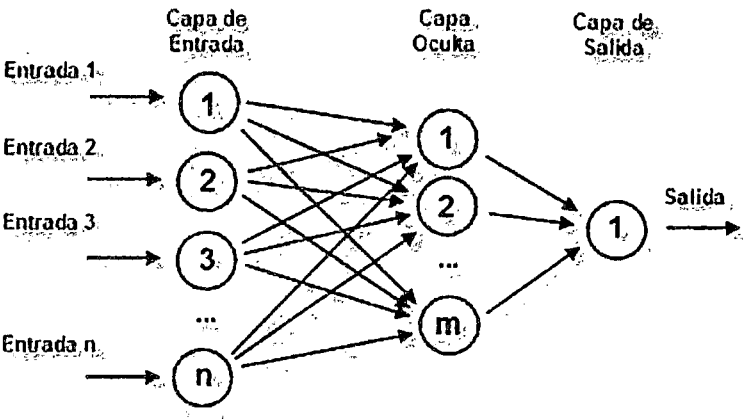


Figura 3-6. Estructura de una red multinivel con conexiones hacia adelante.

b) Formas de conexión entre neuronas

La conectividad entre nodos de una red neuronal está relacionada con la forma en que las salidas de las neuronas están canalizadas para convertirse en entradas de otras neuronas. La señal de salida de un nodo puede ser una entrada de otro elemento de proceso, o incluso ser una entrada de sí mismo (conexión auto recurrente).

Cuando ninguna salida de las neuronas es entrada de neuronas del mismo nivel o de niveles precedentes, la red se describe como de propagación hacia adelante (ver figura 3-2). Cuando las salidas pueden ser conectadas como entradas de neuronas de niveles previos o del mismo nivel, incluyéndose ellas mismas, la red es de propagación hacia atrás. Las redes de propagación hacia atrás que tienen lazos cerrados son sistemas recurrentes.

3.2.2. Características de las redes neuronales artificiales

Existen cuatro aspectos que caracterizan una red neuronal: su topología, el mecanismo de aprendizaje, tipo de asociación realizada entre la información de entrada y salida, y por último, la forma de representación de estas informaciones.

- **Topología de las redes neuronales**

Como se adelanto en el tema anterior, 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 o salida de la red. En este sentido, los parámetros fundamentales de la red son: el número de capas de, el número de neuronas por capa, el grado de conectividad y el tipo de conexiones entre neuronas.

Cuando se realiza una clasificación de las redes en términos topológicos, se suele distinguir entre las redes con una sola capa o nivel de neuronas y las redes con múltiples capas. En los siguientes apartados se analizarán ambos tipos de redes.

- **Redes neuronales multicapa**

Las redes multicapa son aquellas que disponen de conjunto de neuronas agrupadas en varios (2, 3, etc.) niveles o capas. En estos casos, una forma para distinguir la capa a la que pertenece una neurona, consistirá 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 entrada de la red, envían señales de salida a una capa posterior, más cercana a la salida de la red. A estas conexiones se les denomina conexiones hacia adelante o 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 de entradas de las capas anteriores, a estas conexiones se les denomina conexiones hacia atrás o feedback.

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

En las redes *feedforward*, todas las señales se propagan hacia adelante a través de las capas de la red. No existen conexiones hacía atrás y normalmente tampoco auto-recurrentes, ni laterales, excepto en el caso de los modelos de red propuesto por Kohonen denominados Learning Vector Quantizer, en las que existe unas conexiones explicitas muy particulares entre las neuronas de la capa de salida. Las redes feedforward más conocidas son: Perceptron, Adaline, Madaline, Linear Adaptive Memory y Backpropagation. Todas ellas son especialmente útiles en aplicaciones como clasificación de patrones.

- **Mecanismo 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 sistemas biológicos existe una continua creación y destrucción de conexiones. 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. De la misma forma, una conexión se destruye cuando su peso pasa a ser cero.

Durante el proceso de aprendizaje, los pesos de las conexiones de la red sufre modificaciones, por tanto se puede afirmar que este proceso ha terminado (la red ha aprendido) cuando los valores de los pesos permanecen estables.

Un aspecto importante respecto al aprendizaje en las redes neuronales es el conocer cómo se modifican los valores de los pesos; es decir, cuáles son los criterios que se siguen para cambiar el valor asignado a las conexiones cuando se pretende que la red aprenda una nueva información.

Estos criterios determinan lo que se conoce como la regla de aprendizaje de la red. De forma general, se suelen considerar dos tipos de reglas: las que responden a lo que habitualmente se conoce como aprendizaje supervisado y las correspondientes a un aprendizaje no supervisado.

Es por ello por lo que una de las clasificaciones que se realizan de las redes neuronales obedece al tipo de aprendizaje utilizado por dichas redes. Así se puede distinguir las redes neuronales con aprendizaje supervisado y las redes neuronales con aprendizaje no supervisado. La diferencia fundamental entre ambos tipos estriba en la existencia o de un agente externo (supervisor) que controle el proceso de aprendizaje de la red.

- **Redes con aprendizaje supervisado**

El aprendizaje supervisado se caracteriza porque el proceso de aprendizaje se realiza mediante un entrenamiento controlado por un agente externo (supervisor, maestro) que determina la respuesta que debería generar la red a partir de una entrada determinada. El supervisor comprueba 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. Dentro del aprendizaje supervisado se considera el aprendizaje por corrección de error, por refuerzo y estocástico.

- **Aprendizaje por corrección de error**

Consiste en ajustar los pesos de las conexiones de la red en función de la diferencia entre los valores deseados y los obtenidos en la salida de la red; es decir, en función del error obtenido en la salida.

Una regla simple o algoritmo simple de aprendizaje por corrección de error podría ser la siguiente:

$$\Delta w_{ji} = \alpha y_i (d_j - y_j)$$

Siendo:

Δw_{ji}: Variación en el peso de la conexión entre las neuronas i y j.

y_i: Valor de la salida de la neurona i.

d_j: Valor de salida deseado para la neurona j.

y_j: Valor de salida obtenido en la neurona j.

α: Factor de aprendizaje $(0 < \alpha \leq 1)$ que regula la velocidad de aprendizaje.

Un ejemplo de este tipo de algoritmo lo constituye la regla de aprendizaje del Perceptron, utilizada en el entrenamiento de la red del mismo nombre que desarrolló Rosemblatt en 1958. Sin embargo, existen otros algoritmos más evolucionados que éste, que presenta algunas limitaciones, como el no considerar la magnitud del error global cometido durante el proceso completo de aprendizaje de la red, considerando únicamente los errores de individuales (locales) correspondientes al aprendizaje de cada información por separado.

Un algoritmo muy conocido que mejora el del Perceptron y permite un aprendizaje más rápido y campo de aplicación es el propuesto por Widrow y Hoff en 1960, denominado regla delta del mínimo error cuadrado (LMS Error: Least Mean Squared Error), también conocida como regla de Widrow – Hoff, que se aplicó en las redes desarrolladas por los mismos autores, conocidas como Adaline (Adaptive Linear Element), con una única neurona de salida y Madaline (Multiple Adaline), con varias neuronas de salida.

Widrow y Hoff definieron una función que permitía cuantificar el error global cometido en cualquier momento durante el proceso de entrenamiento de la red, lo cual es importante, ya que cuanta más información se tenga sobre el error cometido, más rápido se puede aprender.

Este error medio se expresa de la siguiente forma:

$$Error_{global} = \frac{1}{2P} \sum_{k=1}^{P} \sum_{j=1}^{N} (y_j^{(k)} - d_j^{(k)})^2$$

Siendo:

N: Número de neuronas de salida (en el caso de ADALINE N=1).

P: Número de informaciones que debe aprender la red.

$\frac{1}{2}\sum_{j=1}^{N}(y_j^{(k)} - d_j^{(k)})^2$: Error cometido en el aprendizaje de la información *k*-ésima.

Por tanto, de lo que se trata es de encontrar unos pesos para las conexiones de la red que minimicen esta función de error. Para ello, el ajuste de los pesos de las conexiones de la red se puede hacer de forma proporcional a la variación relativa del error que se obtiene al variar el peso correspondiente:

$$\Delta w_{ji} = k\frac{\partial Error_{global}}{\partial w_{ji}}$$

Mediante este procedimiento, se llega a obtener un conjunto de pesos con los que se consigue minimizar el error medio.

Otro algoritmo de aprendizaje por corrección de error lo constituye la regla delta generalizada o algoritmo de retro propagación de error (error backpropagation), también conocida como regla LMS multicapa. Se trata de una generalización de la regla delta para poder aplicarla a redes con conexiones hacia adelante (feedforward) con capas o niveles internos u ocultos de neuronas que no tiene relación con el exterior. Son redes con capa de entrada, capas ocultas y capa de salida.

3.2.3. El algoritmo Backpropagation

En 1986, Rumelhart, Hinton y Williams, basándose en los trabajos de otros investigadores formalizaron un método para que una red neuronal aprendiera la asociación que existe entre los patrones de entrada a la misma y las clases correspondientes, utilizando más niveles de neuronas que las utilizó Rosemblatt para desarrollar el perceptron. Este método, conocido en general como backpropagation (propagación del error hacia atrás), está basado en la generación de la regla delta y, a pesar de sus propias limitaciones, ha ampliado de forma considerable el rango de aplicaciones de las redes neuronales.

El algoritmo de propagación hacia atrás, o retro propagación, es una regla de aprendizaje, que puede aplicar en modelos de redes con más de dos capas. Una característica importante de este algoritmo es la representación interna del conocimiento que es capaz de organizar en la capa intermedia para conseguir cualquier correspondencia entre la entrada y la salida de la red. Ya se ha mostrado que en muchos casos, como la resolución del problema XOR, es imposible encontrar los pesos adecuados para establecer la correspondencia entre la entrada y la salida mediante una red sin capas intermedias. Con una capa de neuronas ocultas, si es posible establecer dicha correspondencia.

De forma simplificada, el funcionamiento del algoritmo backpropagation consiste en un aprendizaje de un conjunto predefinido de entrada – salidas dados como ejemplo, empleando un ciclo propagación – adaptación de dos fases: primero se aplica un patrón de entrada como estímulo para la primera capa de las neuronas de la red, se va propagando a través de todas las capas superiores hasta generar una salida, se compara el resultado obtenido en las neuronas de salida con la salida que se desea obtener y se calcula un valor de error para cada neurona de salida. A continuación, estos errores de transmiten hacia atrás, partiendo de la capa de salida, hacia todas las neuronas de la capa intermedia que contribuyan directamente a la salida, recibiendo un porcentaje de error aproximado a la participación de la neurona intermedia en la salida original. Este proceso se repite, capa por capa, hasta que todas las neuronas de la red hayan recibido un error que describa su aportación relativa al error total. Basándose en el valor del error recibido, se reajustan los pesos de conexión de cada neurona, de manera que en la siguiente vez que se presente el mismo patrón, la salida esté más cercana a la deseada; es decir, el error disminuya.

La importancia del algoritmo backpropagation en una red multicapa consiste en su capacidad de autoadaptar los pesos de las neuronas de las capas intermedias para aprender la relación que existe entre un conjunto de patrones dados como ejemplo y sus salidas correspondientes. Para poder aplicar esa misma relación, después del entrenamiento, a nuevos vectores de entrada con ruido o incompletos, dando una salida activa si la nueva entrada es parecida a las presentadas durante el aprendizaje. Esta característica es importante, que se

exige a los sistemas de aprendizaje, es la capacidad de generalización, entendida como la facilidad de dar salidas satisfactorias a entradas que el sistema no ha visto nunca en su fase de entrenamiento. La red debe encontrar una representación interna que le permita generar salidas deseadas cuando se le dan las entradas de entrenamiento, y que pueda aplicar, además, a entradas no presentadas durante la etapa de aprendizaje para clasificarlas según las características que compartan con los ejemplos de entrenamiento.

- **La regla delta generalizada**

La regla propuesta por Widrow en 1960 ha sido extendida a redes con capas intermedias (regla delta generalizada) con conexiones hacia adelante y cuyas neuronas tienen funciones de activación continuas (lineales o sigmoidales), dando lugar a un algoritmo de retropropagación. Estas funciones continuas son no decrecientes y derivables. La función sigmoidal pertenece a este tipo de funciones, a diferencia de la función escalón que se utiliza en el perceptron, ya esta última no es derivable en el punto que se encuentra la discontinuidad.

Este algoritmo utiliza también una función o superficie de error asociada a la red, buscando el estado estable de mínima energía o de mínimo error a través del camino descendente de la superficie de error. Por ello, realimenta el error del sistema para realizar la modificación de los pesos en un valor proporcional al gradiente decreciente de dicha función de error.

a) Funcionamiento del algoritmo

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 de forma proporcional a la delta, o diferencia entre la salida deseada y la obtenida (δ=salida deseada – salida obtenida).

Dada una neurona (unidad U_i) y la salida que produce, y_i, el cambio que se produce en el peso de la conexión que une la salida de dicha neurona con la unidad $U_j(w_{ji})$ para un patrón de aprendizaje p determinado es:

$$\Delta w_{ji}(t+1) = \alpha \cdot \delta_{pj} y_{pi}.$$

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

El punto en el que difieren la regla delta generalizada de la regla delta es en el valor concreto de δ_{pj}. Por otro lado, en las redes multicapa, a diferencia de las redes sin capa oculta, en principio no se puede conocer la salida deseada de las neuronas de las capas ocultas para poder determinar los pesos en función del error cometido. Sin embargo, inicialmente sí podemos conocer la salida deseada de las neuronas de salida. Según esto, si consideramos la unidad U_j de salida, entonces definimos:

$$\delta_{pj} = (d_{pj} - y_{pj}) \cdot f'(net_j)$$

donde d_{pj} es la salida deseada de la neurona j para el patrón p y net_j es la entrada que recibe la neurona j.

Esta fórmula es como la de la regla delta, excepto en lo que se refiere a la derivada de la función de transferencia. Este término representa la modificación que hay que realizar en la entrada que recibe la neurona j. En el caso de que dicha neurona no sea de salida, el error que se produce estará en función del error que se cometa en las neuronas que reciban como entrada la salida de dicha neurona. Esto es lo que se denomina procedimiento de propagación de error hacia atrás.

Según esto, en el caso de que U_j no sea una neurona de salida, el error que se produce está en función del error que se comete en las neuronas que reciben como entrada la salida de U_j:

$$\delta_{pj} = (\sum_{k} \delta_{pk} w_{kj}) \cdot f'(net_j)$$

donde el rango de k cubre todas aquellas neuronas a las que está conectada la salida de U_j. De esta forma, el error que se produce en una neurona oculta es la suma de errores que se producen en las neuronas a las que está conectada la salida de ésta, multiplicando cada uno de ellos por el peso de la conexión.

b) Adición de un momento en la regla delta generalizada

El método de retropropagación del error, también conocido como del gradiente descendente, requiere un importante número de cálculos para lograr el ajuste de los pesos de la red. En la implementación del algoritmo, se toma una amplitud de paso que viene dada por la tasa de aprendizaje α. A mayor tasa de aprendizaje, mayor es la modificación de los pesos en cada iteración, con lo que el aprendizaje será más rápido, pero, por otro lado, puede dar lugar a oscilaciones. Rumelhart, Hinton y Williams sugirieron que para filtrar estas oscilaciones se añada un término (momento), β, de manera que dicha expresión quede:

$$\Delta w_{ji}(t+1) = \alpha \cdot \delta_{pj} y_{pi} + \beta \Delta w_{ji}(t)$$

donde β es una constante (momento) que determina el efecto en $t+1$ del cambio de los pesos en el instante t.

Con este momento se consigue la convergencia de la red menor número de iteraciones, ya que si en t el incremento de un peso era positivo y en $t+1$ también, entonces el descenso por la superficie de error en $t+1$ es mayor. Sin embargo, si en t el incremento era positivo y en $t+1$ es negativo, el paso que se da en $t+1$ es más pequeño, lo cual es adecuado, ya que eso significa que se ha pasado por un mínimo y que los pasos deben ser menores para poder alcanzarlo.

Resumiendo el algoritmo backpropagation queda finalmente:

$$w_{ji}(t+1) = w_{ji}(t) + [\Delta w_{ji}(t+1)]$$
$$w_{ji}(t+1) = w_{ji}(t) + [\alpha \delta_{pj} y_{pi} + \beta \Delta w_{ji}(t)]$$

donde:

$$\delta_{pj} = (d_{pj} - y_{pj}) \cdot f'(net_j)$$

si U_j es una neurona de salida y

$$\delta_{pj} = \left(\sum_k \delta_{pk} w_{kj} \right) \cdot f'(net_j)$$

si U_j no es una neurona de salida.

- **Estructura y aprendizaje de la red multicapa**

En una red multicapa 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 las de salida). No hay conexiones hacia atrás ni laterales entre neuronas de la misma capa.

Como se comentó anteriormente, la aplicación del algoritmo backpropagation tiene dos fases, una hacia adelante y otra hacia atrás. Durante la primera fase el patrón de entrada es presentado a la red y propagado a través de las capas hasta llegar a la capa de salida. Obtenido los valores de salida de la red, se inicia la segunda fase, comparándose estos valores con la salida esperada para obtener el error. Se ajustan los pesos de la última capa proporcionalmente al error. Se pasa a la capa anterior con una retropropagación del error, ajustando convenientemente los pesos y continuando con este proceso

hasta llegar a la primera capa. De esta manera se han modificado los pesos de las conexiones de la red para ejemplo o patrón de aprendizaje del problema, del que conocíamos su valor de entrada y la salida deseada que debería generar la red ante dicho patrón.

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 del tipo sigmoidal.

A continuación se presentan a modo de síntesis, los pasos y fórmulas a utilizar para aplicar el algoritmo de entrenamiento.

Paso 1

Inicializar los pesos de la red con valores aleatorios.

Paso 2

Presentar un patrón de entrada, X_p: x_{p1}, x_{p2}, x_{pN}, y especificar la salida deseada que debe generar la red: d_1, d_2,..., d_M (si la red se utiliza como un clasificador, todas la salidas serán cero, salvo una, que será la de la clase a la que pertenece el patrón de entrada).

Paso 3

Calcular la salida actual de la red, para ello presentamos las entradas a la red y vamos calculando la salida que presenta cada capa hasta llegar a la capa de salida, ésta será la salida de la red y_1, y_2,..., y_M. Los pasos son los siguientes:

- Se calculan las entradas netas para las neuronas ocultas procedentes de las neuronas de entrada.

Para una neurona j oculta:

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

en donde el índice se refiere a magnitudes de la capa oculta (hidden); el subíndice p, al p-ésimo vector de entrenamiento, y j a la j-esima neurona oculta. El término θ puede ser opcional, pues actúa como una entrada más.

- Se calculan las salidas de las neuronas ocultas:

$$y_{pj} = f_{j}^{h}(net_{pj}^{h})$$

- Se realizan los mismos cálculos para obtener las salidas de las neuronas de la salida.

$$net_{pk}^{o} = \sum_{j=1}^{L} w_{kj}^{o} y_{pj} + \theta_{k}^{o}$$

$$y_{pk} = f_{k}^{o}(net_{pk}^{o})$$

Paso 4

Calcular los términos de error para todas las neuronas.

Si la neurona k es una neurona de la capa de salida, el valor de la delta es:

$$\delta_{pk}^{o} = (d_{pk} - y_{pk}) \cdot f_{k}^{o'}(net_{pk}^{o})$$

La función f, como se citó anteriormente, debe cumplir el requisito de ser derivable, lo que implica la imposibilidad de utilizar una función escalón. En general, disponemos de dos formas de función de salida que nos pueden

servir; de función lineal de salida $f_k(net_{jk}) = net_{jk}$ y la función sigmoidal representada en la figura y definida por la expresión:

$$f_k(net_{jk}) = \frac{1}{1 + e^{-net_{jk}}}$$

La selección de la función de salida depende de la forma en que se decida representar los datos de salida: si se desea que las neuronas de salida sean binarias, se utiliza la función sigmoidal, puesto que esta función es casi biestable y, además, derivable. En otros casos es tan aplicable una función como otra.

Para la función lineal, tenemos: $f_k^{o\prime} = 1$, mientras que la derivada de una función sigmoidal es:

$$f_k^{o\prime} = f_k^o(1 - f_k^o) = y_{pk}(1 - y_{pk})$$

por lo que los términos de error para las neuronas de salida quedan:

$$\delta_{pk}^o = (d_{pk} - y_{pk})$$

para la salida lineal, y

$$\delta_{pk}^o = (d_{pk} - y_{pk})y_{pk}(1 - y_{pk})$$

para la salida sigmoidal.

Si la neurona j no es la salida, entonces la derivada parcial del error no puede ser evaluada directamente. Por lo tanto, se obtiene el desarrollo a partir de valores que son conocidos y otros pueden ser evaluados.

La expresión obtenida en este caso es:

$$\delta_{pj}^{\square} = f_j^h(net_{pj}^k) \sum_k \delta_{\mathrm{pk}}^o w_{kj}^o$$

donde observamos que el error en las capas ocultas depende de todos los términos de error de la capa de salida. De aquí surge el término de propagación hacia atrás. En particular, para la función sigmoidal:

$$\delta_{pj}^k = x_{pi}(1 - x_{pi}) \sum_k \delta_{pk}^o w_{kj}^o$$

donde k se refiere a todas las neuronas de la capa superior a la de la neurona j. Así, el error que se produce en una neurona oculta es proporcional a la suma de los errores conocidos que se producen en las neuronas a las que está conectada la salida de ésta, multiplicado cada uno de ellos por el peso de la conexión. Los umbrales internos de las neuronas se adaptan de forma similar, considerando que están conectados con los pesos desde entradas auxiliares de valor constante.

Paso 5

Actualización de los pesos.

Para ello, utilizaremos el algoritmo recursivo, considerado por las neuronas de salida y trabajando hacia atrás hasta llegar a la capa de entrada, ajustando los pesos de la forma siguiente:

Para los pesos de las neuronas de la capa de salida:

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

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

$$w_{ji}^h(t+1) = w_{ji}^h(t) + \Delta w_{ji}^h(t+1)$$

$$\Delta w_{ji}^{h}(t+1) = \alpha \delta_{pj}^{h} y_{pi}$$

En ambos casos para acelerar el proceso de aprendizaje, se puede añadir un término momento de valor: $\beta(w_{kj}^{o}(t) - w_{kj}^{o}(t-1))$ en el caso de la neurona de salida, y $\beta(w_{ji}^{h}(t) - w_{ji}^{h}(t-1))$ cuando se trata de una neurona oculta.

Paso 6

El proceso se repite hasta que el termino de error.

$$E_p = \frac{1}{2}\sum_{k=1}^{M} \delta_{pk}^{2}$$

Resulta aceptablemente pequeño para cada uno de los patrones aprendidos.

- **Consideraciones sobre el algoritmo de aprendizaje**

El algoritmo de backpropagation encuentra un valor mínimo de error (local o global) mediante la aplicación de pasos descendentes (gradiente descendente). Cada punto de la superficie de la función error corresponde a un conjunto de valores de los pesos de la red. Con el gradiente descendente, siempre que se realiza un cambio en todos los pesos de la red, se asegura el descenso por la superficie del error hasta encontrar el valle más cercano, lo que puede hacerse que el proceso de aprendizaje se detenga en un mínimo local de error.

Por tanto 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 algún punto estacionario, con lo cual no se llega a encontrar el mínimo global de la función del error. Sin embargo, ha de

tenerse en cuenta que no tiene porque alcanzarse el mínimo global en todas las aplicaciones, sino que puede ser suficiente con un error mínimo preestablecido.

a) Control de convergencia

En las técnicas de gradiente decreciente es conveniente avanzar por la superficie de error con incrementos pequeños de los pesos. Esto se debe a que tenemos una información local de la superficie y no se sabe lo lejos o cerca que se está del punto mínimo. Con incrementos grandes, se corre el riesgo de pasar por encima del punto mínimo sin conseguir estacionarse en él. Con incrementos pequeños, aunque se tarde más en llegar, se evita que ocurra esto.

El elegir un incremento paso adecuado influye en la velocidad con la que converge el algoritmo. Esta velocidad se controla a través de la constante de proporcionalidad o tasa de aprendizaje α. Normalmente de ser un número pequeño (del orden de 0,05 a 0,25), para asegurar que la red llegue a asentarse en una solución, Un valor pequeño de α significa que la red tendrá que hacer un gran número de iteraciones. Si esa constante es muy grande, los cambios de pesos son muy grandes, avanzando muy rápidamente por la superficie de error, con el riesgo de saltar el mínimo y estar oscilando alrededor de él, pero sin poder alcanzarlo.

Lo habitual es aumentar el valor de a medida que disminuye el error de la red durante la fase de aprendizaje. Así, aceleramos la convergencia, aunque sin llegar nunca a valores de α demasiados grandes, que hicieran que la red oscile alejándose demasiado del valor mínimo. Otra forma de incrementar la velocidad de convergencia consiste en añadir, como se ha comentado en apartados anteriores, un término de momento consistente en sumar la fracción del anterior cambio cuando se calcula el valor del cambio de peso actual. Este término adicional tiende a mantener los cambios de peso en la misma dirección.

Un último aspecto a tener en cuenta es la posibilidad de convergencia hacía alguno de los mínimos locales que pueden existir en la superficie de error del espacio de pesos. En el desarrollo matemático que se ha realizado para llegar al algoritmo de retropropagación, no se asegura en ningún momento que el mínimo que se encuentre sea global. Una vez que la red se asienta en un mínimo, sea local o global, cesa el aprendizaje, aunque el error siga siendo demasiado alto, si se ha alcanzado un mínimo local. En todo caso, si la solución es admisible desde el punto de vista del error, no importa si el mínimo es local o global o si se ha detenido en algún momento previo a alcanzar un verdadero mínimo.

En la práctica, si una red deja de aprender antes de llegar a una solución aceptable, se realiza un cambio en el número de neuronas ocultas o en los parámetros de aprendizaje o, simplemente, se vuelve a empezar con un conjunto distinto de pesos originales y se suele resolver el problema.

b) Dimensionamiento de la red: Número de neuronas ocultas

No se pueden dar reglas concretas para determinar el número de neuronas o el número de capas de una red para resolver un problema concreto. Lo mismo ocurre a la hora de seleccionar el conjunto de vectores de entrenamiento. En todos estos casos, lo único que se puede hacer es dar unas ideas generales deducidas de la experiencia de numerosos autores.

Respecto al número de capas de la red, en general tres capas son suficientes (entrada-oculta-salida). Sin embargo, hay veces en que un problema es más difícil de resolver (la red aprende más deprisa) con más de un capa oculta. El tamaño de las capas, tanto de entrada como de salida, suele venir determinado por la naturaleza de la aplicación. En cambio, decidir cuántas neuronas debe tener la capa oculta no suele ser tan evidente.

El número de neuronas ocultas interviene en la eficacia del aprendizaje y de generalización de la red. No hay ninguna regla que indique el número óptimo, en cada problema se debe ensayar con distintos números de

neuronas para organizar la representación interna y escoger el mejor. La idea más utilizada, sobre todo en los sistemas simulados, consiste en tener el menor número posible de neuronas en la capa oculta, porque cada una de ellas supone una mayor carga de procesamiento en el caso de una simulación en software. En un sistema implementado en hardware, este problema no es crucial; sin embargo, sí habrá que tener presente el problema de comunicación entre los distintos elementos de proceso.

Es posible eliminar neuronas ocultas si la red converge sin problemas, determinando el número final en función del rendimiento global del sistema. Si la red no converge, es posible que sea necesario aumentar este número. Por otro lado, examinando los valores de los pesos de las neuronas ocultas periódicamente en la fase de aprendizaje, se pueden detectar aquellas cuyos pesos cambian muy poco durante el aprendizaje respecto a sus valores iníciales, y deducir por tanto el número de neuronas que apenas participan en el proceso de aprendizaje.

c) Inicialización y cambio de pesos

Sería ideal, para una rápida adaptación del sistema, inicializar los pesos con una combinación de valores W muy cercano al punto de mínimo error buscado. Pero es imposible, porque no se conoce a priori dónde está el punto mínimo. Así, se parte de un punto cualquiera del espacio, inicializando los pesos con los valores pequeños y aleatorios cualesquiera, al igual que los términos umbral, que aparecen en las ecuaciones de entrada neta a cada neurona. Este valor umbral se suele tratar como un peso más que está conectado a una neurona ficticia de salida siempre 1. La utilización de los términos umbral es opcional, pues en caso de utilizarse, es tratado exactamente igual que un peso más y participa como tal en el proceso de aprendizaje.

La expresión de la entrada neta a cada neurona se podrá escribir de la forma:

$$net_{pk} = \sum_{j=1}^{L} w_{kj} x_{pj} + \theta_k$$

para una neurona de salida *k*. Si consideramos $\theta_k \equiv w^o_{k(L+1)}$; $x_{p(L+1)} \equiv 1$, se puede escribir la siguiente expresión:

$$net_{pk} = \sum_{j=1}^{L+1} w_{kj} x_{pj}$$

o, lo que es lo mismo, si consideramos $\theta_k \equiv w^o_{k0}$; $x_{p0} \equiv 1$, podemos tomarla como:

$$net_{pk} = \sum_{j=0}^{L} w_{kj} x_{pj}$$

La modificación de los pesos puede realzarse cada vez que un patrón ha sido presentado, o bien después de haber acumulado los cambios de los pesos en un número de iteraciones. El momento adecuado para cambiar los pesos depende de cada problema concreto.

3.2.4. Redes con Función de Base Radial

Las redes con función de base radial (RBF, Radial Basis Function) son redes neuronales multicapa con conexiones hacia adelante, se caracterizan porque están formadas por una única capa oculta y cada neurona de esta capa posee un carácter local. Este carácter local viene dado por el uso de las llamadas funciones de base radial, generalmente la función gaussiana, como función de activación. Se crearon con el objetivo de funcionar en tiempo real y son una de las redes más utilizadas en el diagnóstico de defectos.

- **Arquitectura de una RBF**

La arquitectura de una RBF tiene tres capas (ver figura 3-7). La capa oculta aplica una transformación no lineal a los datos de entrada hacia la capa

oculta. La capa de salida aplica una transformación lineal de los datos de la capa oculta hacia la salida. Las unidades ocultas usan funciones de base radial. Las funciones base normalmente toman la forma $\emptyset_j(\|\vec{x} - \vec{u}_j\|)$. La función depende de la distancia (usualmente considerada euclidiana) entre el vector de entrada $\vec{x}$ y el vector $\vec{u}_j$.

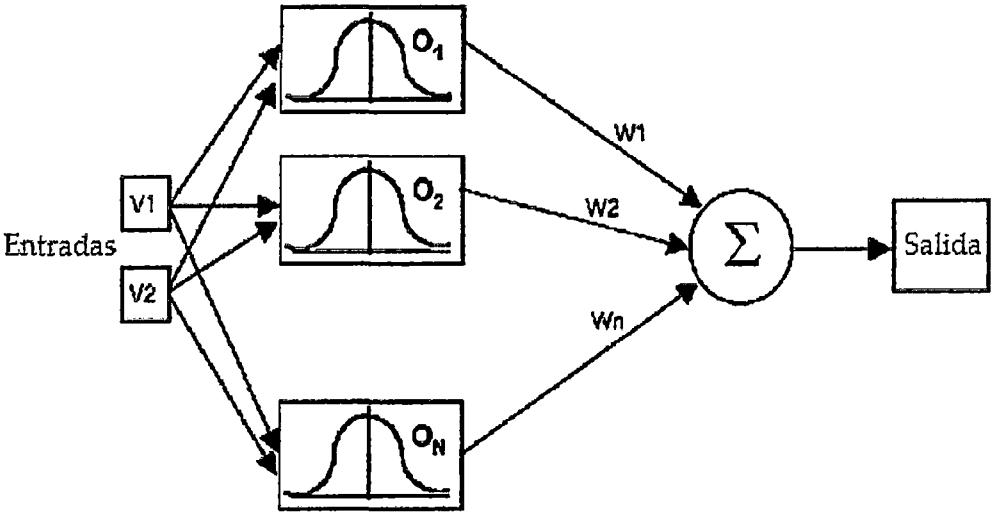


Figura 3-7. Arquitectura de una red RBF.

Las distancia euclidiana:

$$\|\vec{x} - \vec{u}_j\|^2 = \sum_i (x_i - u_{ij})^2$$

En las redes RBF la activación de una unidad oculta está determinada por la distancia entre la entrada del vector y el vector prototipo. La función es aproximada como una combinación lineal de funciones base.

$$z_k(\vec{x}) = \sum_{j=1}^{M} w_{kj}\, \emptyset_j(\vec{x}) + w_{k0} = \sum_{j=0}^{M} w_{kj}\, \emptyset_j(\vec{x})$$

Donde $\emptyset_j(\vec{x})$ es una función de activación de base radial, que suele adoptar tres formas distintas, la gaussiana, la inversa cuadrática y la multicuadrática.

- **Funciones de base radial**

Las funciones de base radial son una clase especial de funciones. Sus propiedades características son que su respuesta disminuye (o se incrementa) monotónicamente cuando se aleja del punto central. El centro, la escala de la distancia y la forma precisa de la función radial son parámetros del modelo y son todos fijos si el modelo es lineal.

Una función típica es la función radial Gaussiana la cual, en caso de una entrada escalar, es:

$$h(x) = exp\,(-\frac{(x-c)^2}{r^2}).$$

Sus parámetros son el centro c y su radio r. La figura 3-8 ilustra una función radial Gaussiana con centro igual a cero y radio igual a uno.

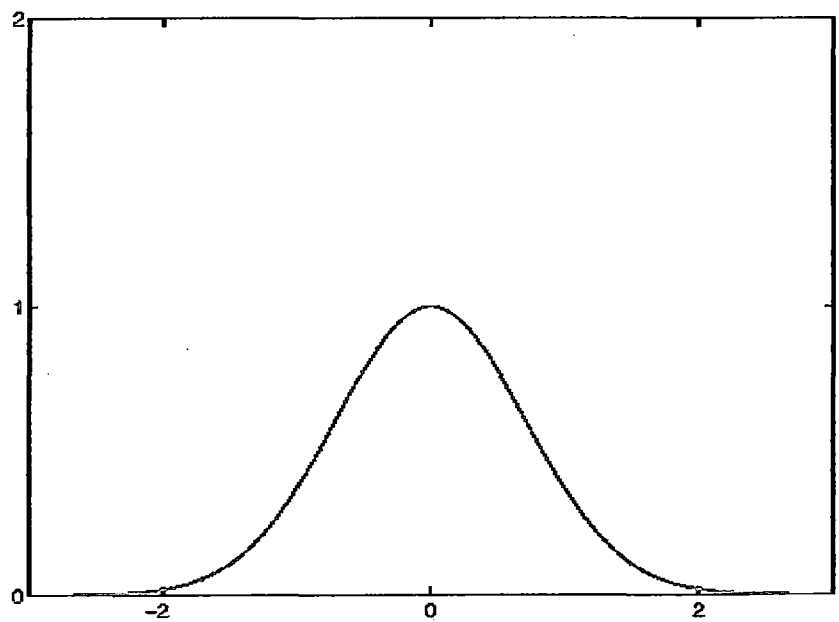


Figura 3-8. Gráfica de una función radial Gaussiana.

Una función radial Gaussiana disminuye monotónicamente cuando se aleja de su centro (ver figura 3-8). En contraste, la función radial Multicuadrática la cual, en el caso de una entrada escalar, es:

$$h(x) = \frac{\sqrt{r^2 + (x - c)^2}}{r},$$

se incrementa monotónicamente cuando se aleja de su centro (ver figura 3-9).

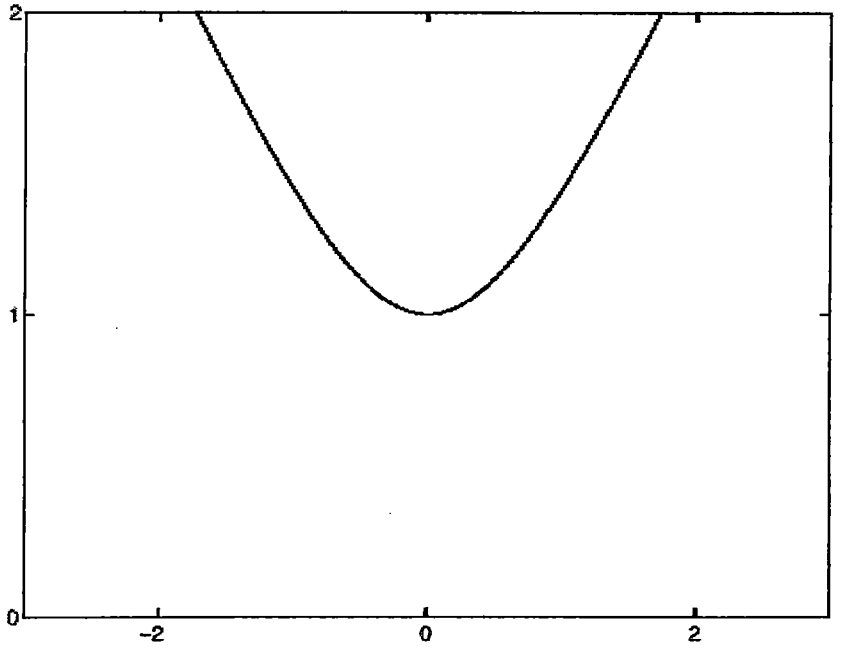


Figura 3-9. Gráfica de una función radial Multicuadrática.

Las funciones Gaussianas son locales, dan una respuesta significativa solo dentro del vecindario cercano a su centro. Por esta razón son más usadas que las funciones Multicuadráticas ya que estas últimas tienen una respuesta más global. Las funciones Gaussianas son más biológicamente plausibles porque su respuesta es finita.

Regresando al contexto de las redes RBF, el uso de la función gaussiana como función de activación de la capa oculta viene dado por:

$$\emptyset_j(\vec{x}) = e^{-\frac{\|\vec{x}-\vec{u}_j\|^2}{2\sigma_j^2}}$$

La función gaussiana es una función base localizada con la propiedad de $\emptyset(r) \to 0$ cuando $|r| \to \infty$. Donde $\vec{u}_j$ determina el centro de la función base $\emptyset_j$; σ_j es la amplitud del parámetro que controla como se expande la curva. La función gaussiana no está normalizada, ya que todos los factores generales pueden ser absorbidos en los pesos.

Usualmente se localiza la función base con la propiedad $\emptyset(r) \to 0$ cuando $|r| \to \infty$. También hay otras localizaciones de la función base.

$$\emptyset(r) = (r^2 + \sigma^2)^{-\alpha}, \qquad \alpha > 0. \ (r = \|\vec{x} - \vec{u}_j\|)$$

Una neurona en capa oculta es más sensible a los datos cerca de su centro. Esta sensibilidad puede ser afinada ajustando el parámetro σ. Si el parámetro σ es alto implica menos sensibilidad. Dado un vector de entrada, típicamente solo pocas unidades ocultas tendrán significantes activaciones. La capa oculta aplica transformaciones no lineales de la entrada hacia la capa oculta. Las redes neuronales con función de base radial son capaces de aproximar funciones, con pocas restricciones en la forma de la funciones base. Normalmente no se considera topologías de redes RBF que tiene más de una capa oculta.

La función gaussiana de la RBF puede ser generalizada para permitir matrices arbitrarias de covarianza $\sum_j$.

$$\emptyset_j(\vec{x}) = exp(-\frac{1}{2}(\vec{x} - \vec{u}_j{}^t) \sum_j^{-1} (\vec{x} - \vec{u}_j))$$

Asumimos que $\vec{x}$ es d dimensional. Las matrices son $\sum_j$ simetricas, cada función base tiene $d(d+3)/2$ parámetros independientes ajustables, en comparación con los $d+1$ parámetros independientes de la función regular base Gaussiana.

En la práctica se busca un balance adecuado entre usar un pequeño número de funciones base con muchos parámetros ajustables y un gran número de funciones pero menos flexibles.

Sea la función de error cuadrático:

$$E = \sum_{p=1}^{n} E_p \, , donde \; E_p = \frac{1}{2} \sum_{k=1}^{c} (t_{kp} - z_{kp})^2$$

donde t_{kp} es el valor objetivo para la salida unitaria k cuando a la red se le presenta el vector de entrada $\vec{x}_p$.

$$y_{jp} = e^{\frac{\|\vec{x}_p - \vec{u}_j\|^2}{2\sigma_j^2}} \quad z_{kp} = \sum_{j=0}^{M} w_{kj} y_{jp}$$

Donde M: Número de unidades ocultas.

- **Carácter local de las redes RBF**

Las funciones de base radial se caracterizan porque poseen un nivel máximo de activación para valores de entrada cercanos a cero y dicho nivel decrece a medida que la variable se aleja de dicho punto.

Se dice que las RBF son redes de carácter local, ya que, dado un patrón de entrada a la red, solo aquellas neuronas ocultas cuyos centros estén en la vecindad de dicho patrón se van a activar; el resto de neuronas permanecerán inactivas o con un menor nivel de activación.

En la figura 3-10 se observa la activación de dos neuronas vecinas para dos patrones de entrada, P(n)= [0,15 0,45].

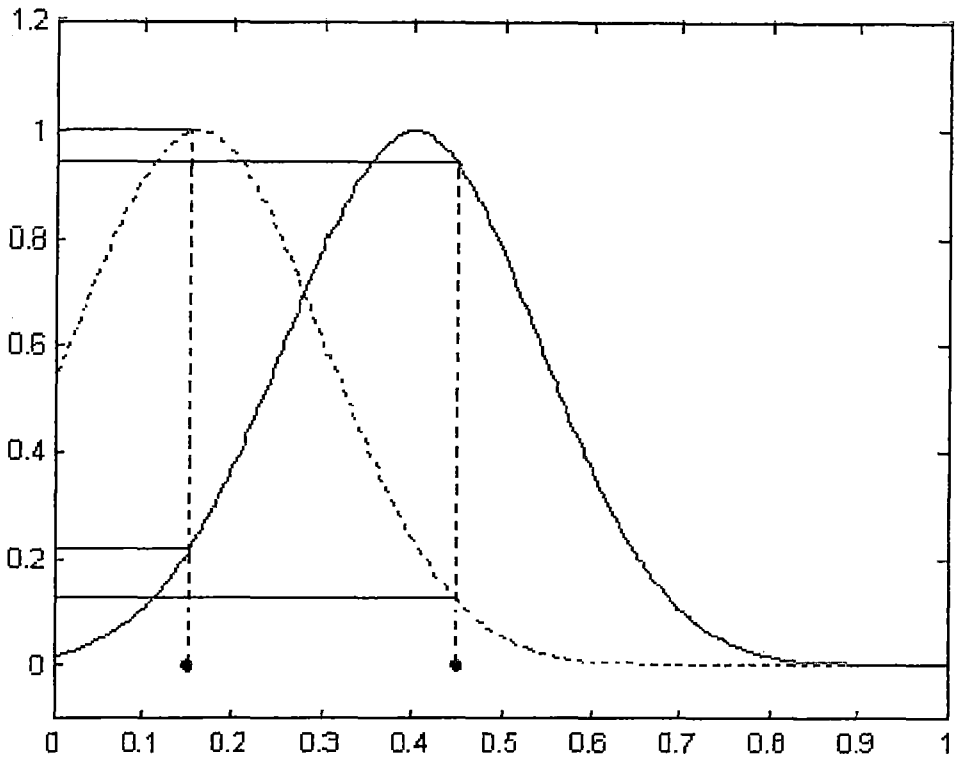


Figura 3-10. Activación de dos neuronas de la capa oculta de una red RBF.

Como puede observarse en la figura 3-10. Para un valor de P(n) dado, existen dos valores de activación. Para el caso de la neurona representada con la curva discontinua, para un valor de 0,15 de entrada, presenta una activación de 1 (el valor máximo), mientras que la neurona vecina también se activa, pero en este caso, alcanza un valor de 0,22. La diferencia de activaciones de las dos neuronas para los dos valores de entrada, se detalla con el color azul (para la neurona de la izquierda), y rojo (para la neurona de la derecha). Cuanto más alejado esté el patrón de entrada del centro de la función de base radial de la neurona, la activación será de menor valor.

- **Entrenamiento de las RBF**

Los parámetros de la función base deben ser escogidos intuitivamente para formar una representación de la densidad de probabilidad de los datos de entrada. Los centros de la función base $\vec{u}_j$ pueden ser considerados como

prototipos en los vectores de entrada. Esto permite que los parámetros que gobiernan la función base sean determinados usando entrenamiento no supervisado la cual solo depende de los datos de entrada del set de entrenamiento y que se ignore cualquier información sobre salida.

Esto conduce a un procedimiento de entrenamiento de dos etapas para las redes RBF. En la primera etapa los datos de entrada $\{\vec{x}_p\}$ son únicamente usados para determinar los parámetros de la función base ($\vec{u}_j$ y σ_j para la función base gaussiana). Por ejemplo:

- Aleatoriamente se selecciona un set de datos de entrenamiento como centro de la función base.
- Se identifica grupos de datos de entrenamiento usando métodos no supervisados como el algoritmo *K-means* y se pone una función base centrada en cada grupo. Se escoge σ_j para que sea múltiplo de la distancia entre los centros de las funciones base.

En segunda etapa, la función base se mantiene fija y los pesos de la segunda capa son optimizados.

$$z_k(\vec{x}) = \sum_{j=0}^{M} w_{kj} \emptyset_j(\vec{x})$$

Dado que la función base es considerada fija, la red es equivalente a una red neuronal de una sola capa. Consideramos la sumatoria de la función de error cuadrático como:

$$E = \frac{1}{2} \sum_p \sum_k \left(t_{kp} - z_k(\vec{x}_p)\right)^2$$

Dado que la función de error es una función cuadrática de los pesos, su minimizado puede ser encontrado en términos de la solución de un conjunto de ecuaciones lineales.

Estableciendo la gradiente de E con respecto a w_{kj} cero resulta:

$$(\emptyset^t \emptyset) W^t = \emptyset^t T$$

donde $(W)_{kj} = w_{kj}$, $(\emptyset)_{pj} = \emptyset_j(\vec{x}_p)$ y $(T)_{pk} = t_{kp}$.

Dada la seudoinversa $\emptyset^! = (\emptyset^t \emptyset)^{-1} \emptyset^t$, la solución formal para los pesos es dada por:

$$W^t = \emptyset^! T$$

En la práctica las ecuaciones son resueltas usando un valor singular de descomposición, para evitar problemas debidos a la posibilidad de que $\emptyset^t \emptyset$ sea singular o cercanamente singular. Los pesos de la segunda capa pueden ser hallados muy rápidamente usando técnicas de algebra lineal.

La posibilidad de elegir parámetros adecuados para las unidades ocultas, sin tener que realizar una completa optimización no lineal de la red, es una de las principales ventajas de la redes RBF en comparación con las redes neuronales multicapa.

Hay muchas aplicaciones potenciales para las redes neuronales donde sin etiquetar los datos de entrada es tedioso pero los casos donde los datos son etiquetados son pocos. Por ejemplo puede ser fácil colectar ejemplos de datos pero el etiquetado de los datos con variables objetivos puede tomar tiempo a un humano experto. Con tales aplicaciones, el entrenamiento de dos etapas para las redes RBF puede ser particularmente ventajoso ya que la determinación de la representación no lineal dada la primera capa de la red puede ser hecha usando una gran cantidad de datos no etiquetados, dejando un pequeño número de parámetros en la segunda capa para ser determinados usando los datos etiquetados.

En cada etapa del proceso entrenamiento, podemos asegurar que el número de datos es grande comparado con el número de parámetros a ser determinado, según sea necesario para una buena generalización.

El uso de técnicas no supervisadas para determinar los parámetros de la función base no necesita ser óptima para el problema de aprendizaje supervisado, debido a que no toma en cuenta las etiquetas de los objetivos asociados con la data. Técnicas no supervisadas pueden ser usadas para inicializar los parámetros de la función base antes de ejecutar el algoritmo de la gradiente descendente para incrementar la velocidad del proceso de entrenamiento.

A continuación se presentan dos ejemplos del uso de los dos algoritmos de entrenamiento para dos casos diferentes de data. En ambos casos de contrastará los resultados obtenidos.

Ejemplo 1: Se tiene el vector de datos P y la salida deseada T. Se va realizar el entrenamiento usando los dos algoritmos de aprendizaje, la red Backpropagation (BPN) y la red con función de base radial (RBF). Este es un problema de clasificación donde los datos de entrada (P) tendrán que ser clasificados de acuerdo al vector T.

```
P = [0.5 1 1 1.5 1 3; 1 2 5 3 3 3];
T = [0 0 0 1 1 1];
```

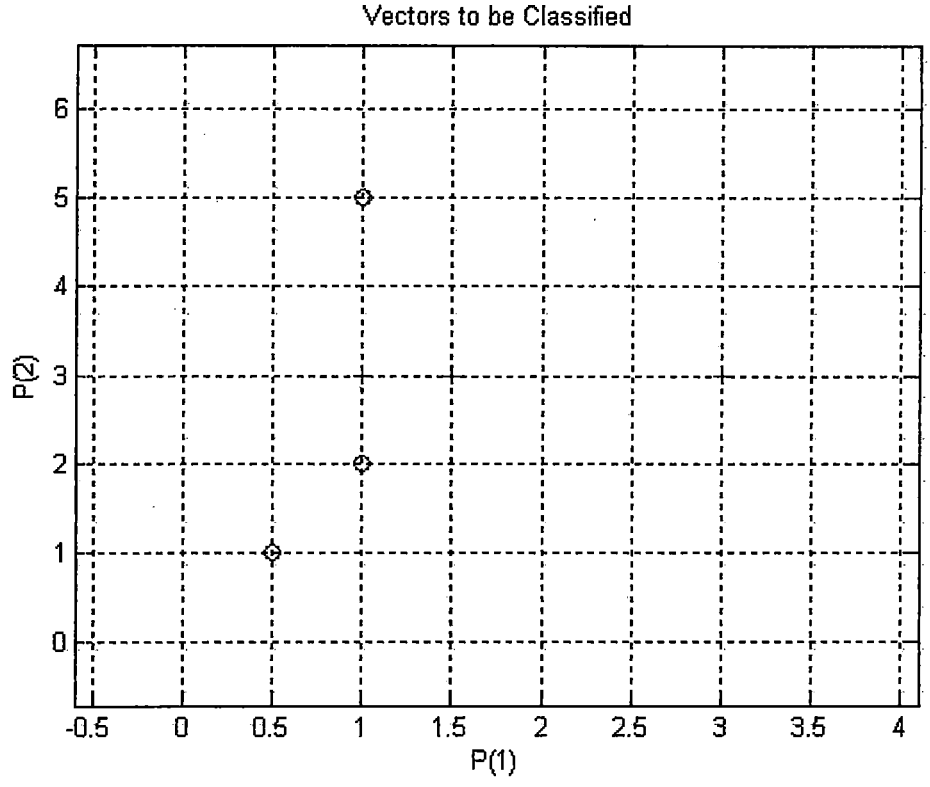


Figura 3-11. Datos a clasificar.

La figura 3-11 muestra la clasificación de los datos (P) que se desea alcanzar. El símbolo cruz indica que se debe clasificar como uno y el símbolo del círculo que se debe clasificar como cero.

La topología de la red Backpropagation entrenada tenía dos neuronas en la primera capa oculta, una neurona en la segunda capa oculta y una en la capa de salida. Las funciones de transferencia en las capas ocultas fueron ***tansig*** y en la capa de salida ***purelin***. Adicionalmente el MSE a alcanzar fue 0.001.

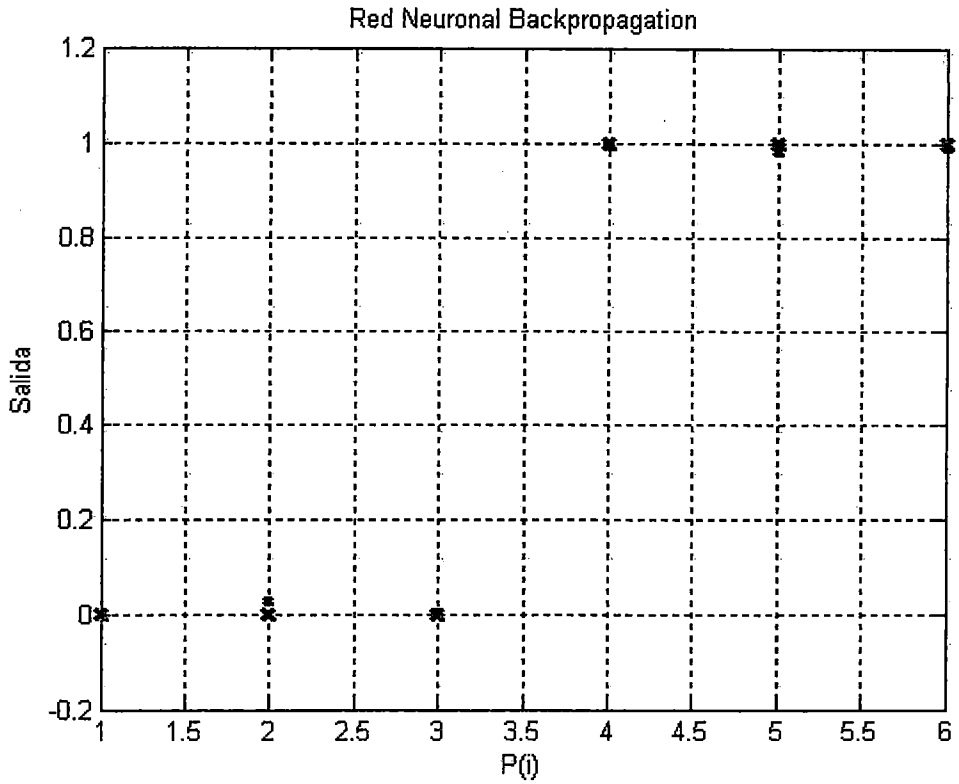


Figura 3-12. Simulación de la red Backpropagation.

Después del entrenamiento probamos los pesos obtenidos y se obtuvo la gráfica que se muestra en la figura 3-12. Los símbolos de color azul indican el valor de salida deseado y los símbolos de color rojo los valores obtenidos de la simulación después del entrenamiento de la red.

Para el entrenamiento de la red RBF está tiene 5 neuronas en la capa oculta y una neurona en la capa de salida. En el MSE a alcanzar fue 0.01 y el valor del spread 1. La figura 3-13 muestra la simulación de los pesos obtenidos en el entrenamiento de la red RBF.

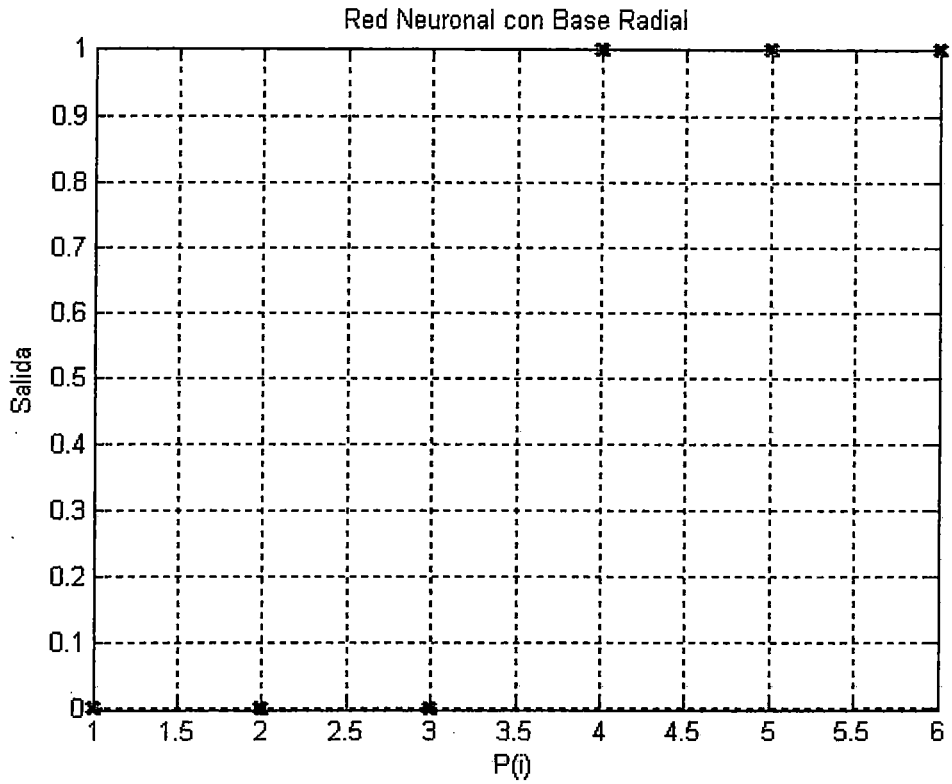


Figura 3-13. Simulación de la red RBF.

De las gráficas anteriormente mostradas (ver figuras 3-12 y 3-13) podemos observar que las redes RBF tienen valores más cercanos a los valores deseados (T). De la figura 3-14 podemos apreciar que la red neuronal con base radial converge más rápidamente alcanzando el error deseado en la quinta época mientras que la red Backpropagation alcanza un valor de convergencia aproximadamente a las 18 épocas.

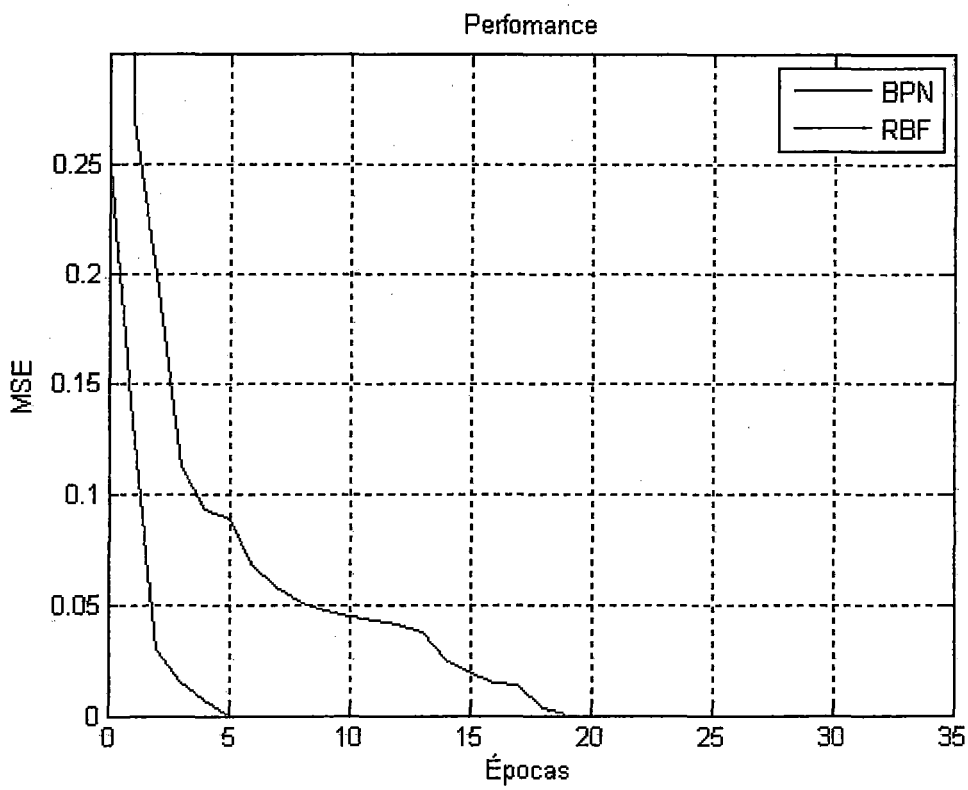


Figura 3-14. Performance de los dos algoritmos de aprendizaje.

Ejemplo 2: El siguiente ejemplo que se presenta no es un problema de clasificación debido a que la salida deseada (T) toma más de un valor. Este problema es más parecido a un problema de ajuste de funciones. Tenemos los siguientes datos:

P = [0 1 2 3 4 5 6 7 8 9 10];
T = [0 1 2 3 4 3 2 1 2 3 4].

En este caso la red neuronal Backpropagation usada tiene 5 neuronas en la primera capa oculta, una neurona en la segunda capa y una neurona en la capa de salida, con funciones de transferencia ***tansig*** para las neuronas de la capa oculta y ***purelin*** para las neuronas de la capa de salida. En el caso de la red RBF entrenada tenía 7 neuronas en la capa oculta y una en la salida. Para ambos casos el error a alcanza fue 0.01.

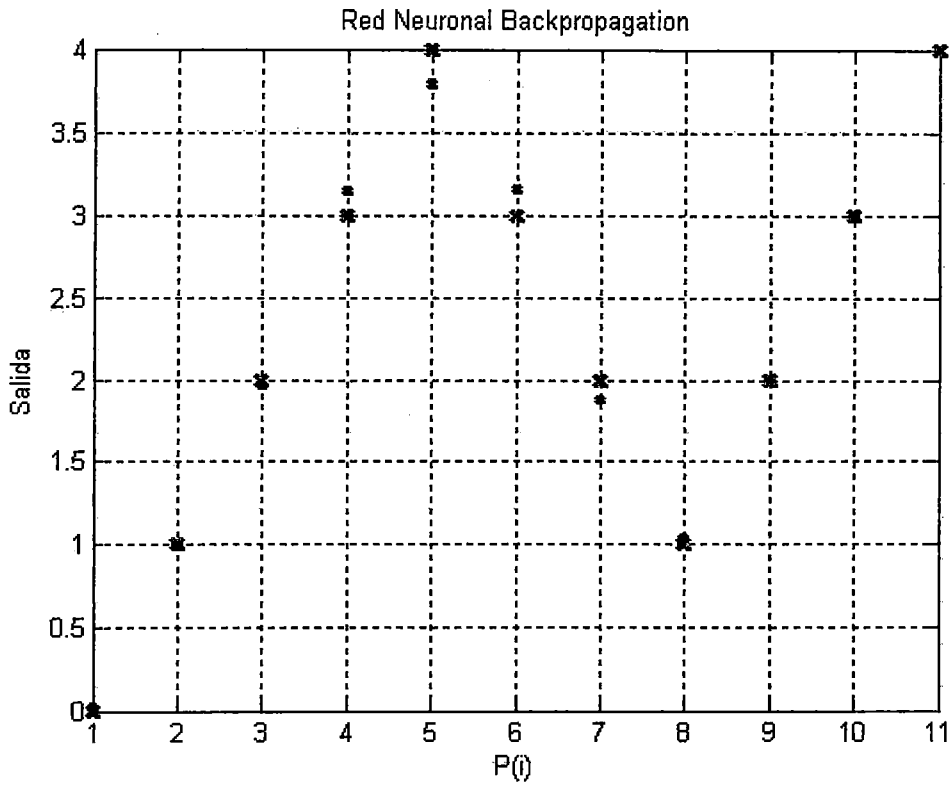


Figura 3-15. Simulación de la red Backpropagation.

La figura 3-15 muestra la simulación de la red Backpropagation entrenada. Los símbolos azules representan la salida deseada y los rojos la salida simulada con los pesos obtenidos después del entrenamiento.

La red RBF entrenada tenía 7 neuronas en la capa oculta y una en la salida. El error a alcanzar era 0.01 y con un valor de spread igual a 1. En la figura 3-16 se muestra la simulación de la red RBF después del entrenamiento.

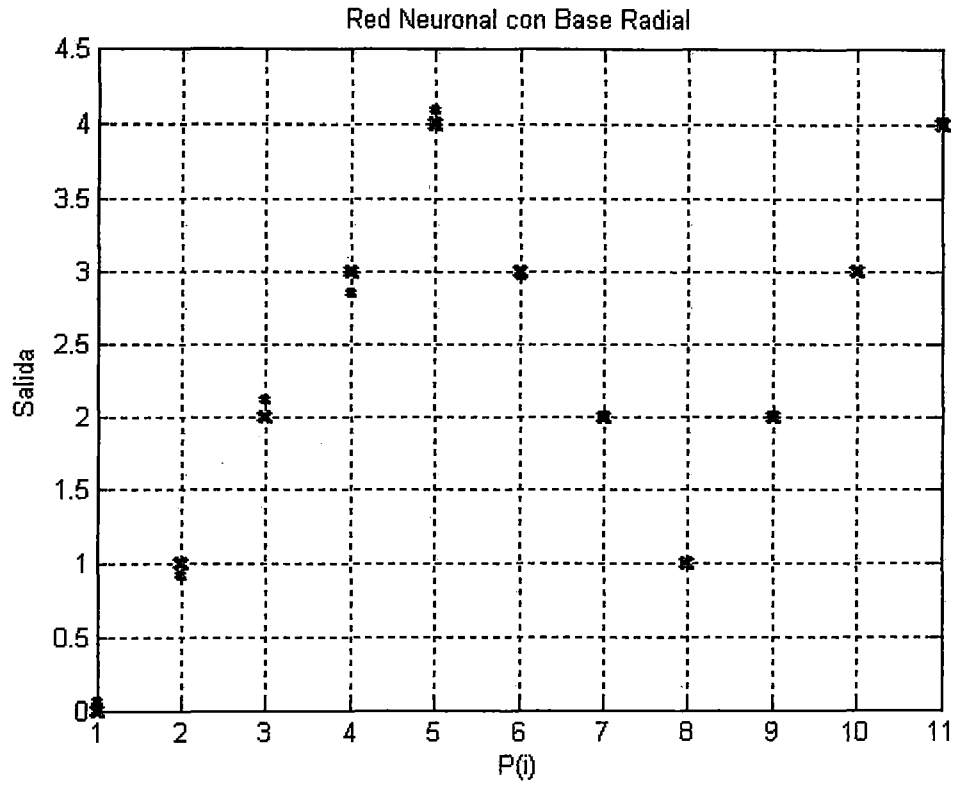


Figura 3-16. Simulación de la red RBF.

Se puede apreciar que los valores obtenidos simulando la red RBF son más próximos a los valores deseados que los valores obtenidos con la simulación de la red Backpropagation. En la figura 3-17 se muestra la gráfica de la performance de ambos algoritmos durante el entrenamiento. De la gráfica se aprecia que la red RBF converge más rápidamente que la red Backpropagation.

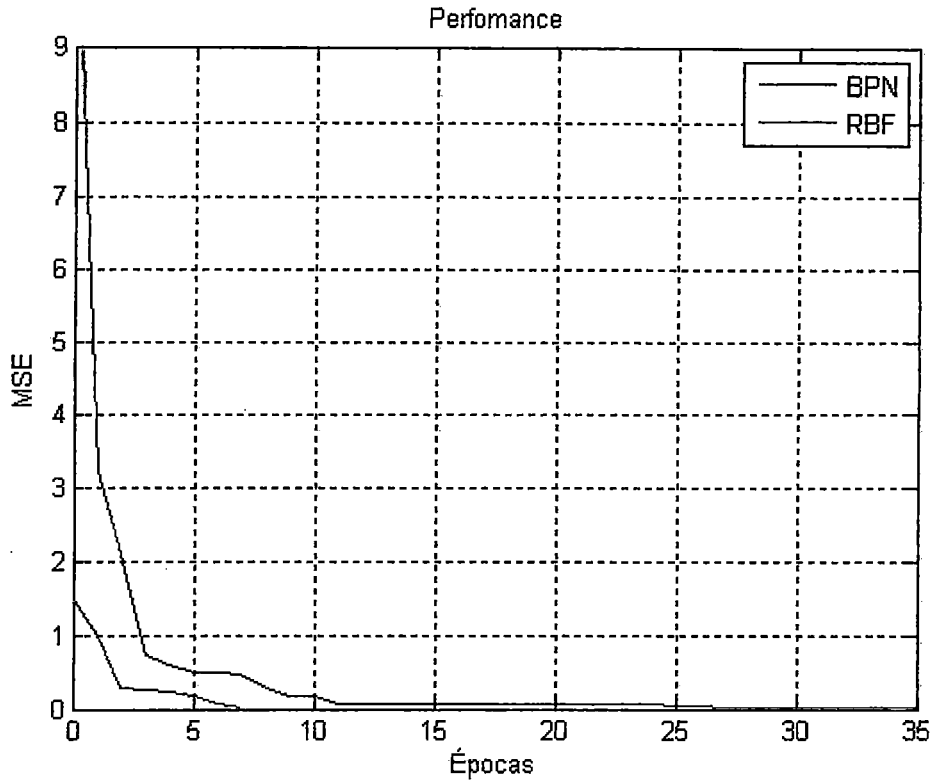


Figura 3-17. Performance de los dos algoritmos de aprendizaje.

De los resultados obtenidos en ambos ejemplos se puede decir para estos casos las redes RBF han tenido mejores resultados, tanto en la simulación como en la rapidez convergencia, que las redes Backpropagation. Pero esto solo es un primer alcance ya que en el siguiente capítulo también se hará el uso de estos dos algoritmos.

3.3. Algoritmo de Cálculo de Coordenadas Tridimensionales

El algoritmo nos permitirá calcular las coordenadas tridimensionales del punto de agarre de un objeto. Conocer esta información es útil para guiar un brazo robótico y ubicar el centro del efector final muy cerca al punto de agarre; y efectuar la captura del objeto sí se conociera la dirección o se estableciera ciertas reglas para estimarla en casos concretos. En la figura 3-18 se presenta el diagrama de flujo de todo el proceso para la obtención de las coordenadas tridimensionales del punto de agarre de un objeto.

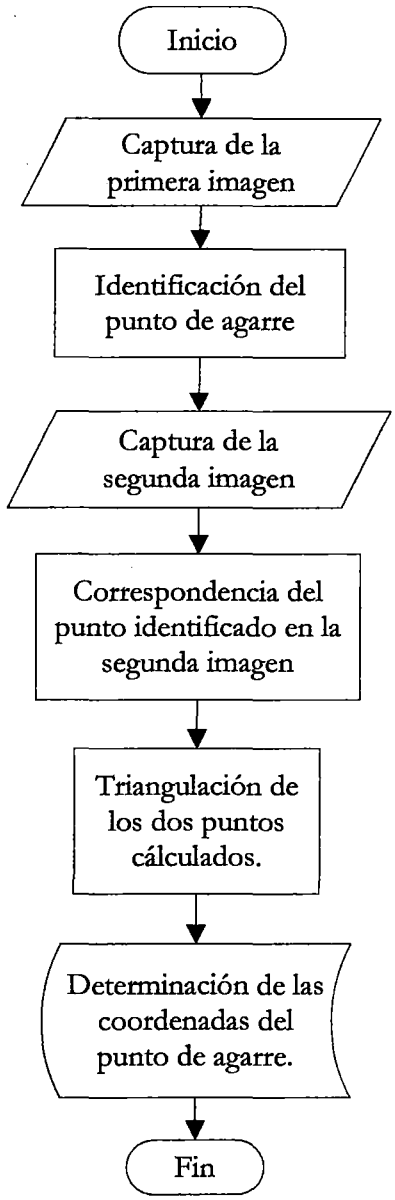


Figura 3-18. Diagrama del proceso de cálculo de coordenadas tridimensionales.

La identificación del punto de agarre en la imagen captada es en realidad la identificación de la proyección del punto de agarre que se encuentra en el espacio tridimensional. Capturada la primera imagen se procede a identificar el punto de agarre y sus coordenadas en la imagen (u_i, v_i). Después se captura una segunda imagen en una posición diferente pero cercana a la posición de donde se capturó la primera imagen, se realizará la operación de correlación para determinar las coordenadas correspondientes (u_d, v_d) del punto ya identificado en la segunda imagen. Finalmente con las coordenadas (u_i, v_i) y (u_d, v_d) en ambas imágenes se

realizará la operación de triangulación para el cálculo de las coordenadas tridimensionales del punto de agarre.

Como se indica en el diagrama de flujo, un proceso importante es lograr identificar el punto de agarre del objeto en la imagen que se ha capturado con la cámara. Para este propósito utilizaremos un algoritmo de aprendizaje previamente entrenado con data sintética. Este método no se enfocará en la identificación de la clase del objeto (tasa, libro o plato.) sino que se asumirá que se conoce con anterioridad. El entrenamiento se realizará para cada clase de objeto en forma independiente, obteniendo parámetros (coeficientes o pesos sinápticos) diferentes para clase. En otras palabras el algoritmo podrá inferir puntos de agarre conociendo previamente de objeto se trata. En la siguiente parte trataremos el algoritmo de entrenamiento y sus etapas.

3.4. Algoritmo de Entrenamiento

Para poder inferir posibles puntos de agarre en las imágenes capturadas se realizará el entrenamiento de tres algoritmos de aprendizaje basados en Regresión Logística, Redes Multicapa Perceptrón y Redes Neuronales con Función de Base Radial. El entrenamiento de estos algoritmos implica procesos previos como la extracción de datos y la formación de vectores de entrenamiento, tal como se muestra en el diagrama de flujo de la figura 3-19.

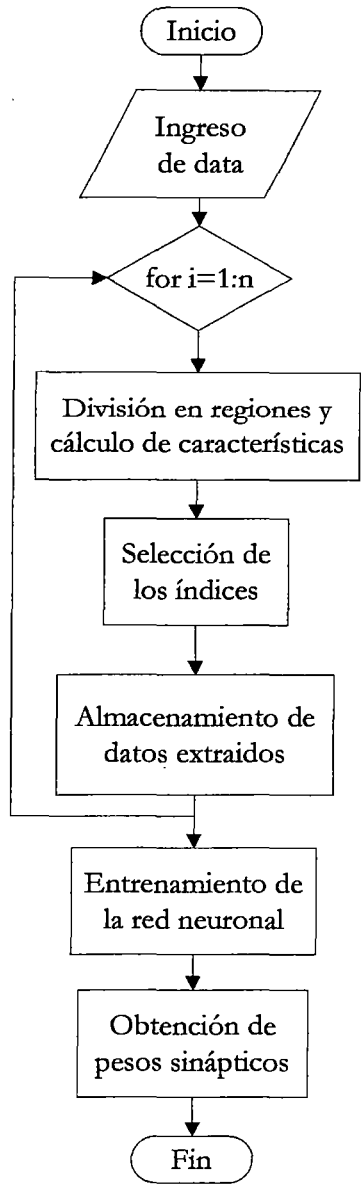


Figura 3-19. Diagrama de flujo del proceso entrenamiento.

3.4.1. Data sintética usada

Se trabajará con data sintética generada en el software libre PovRay [15]. Usar data sintética reduce el tiempo el recolectar data real y el ruido o perturbaciones que se puedan presentar. De esta manera se puede disponer de gran cantidad de data. La data sintética básicamente está compuesta por imágenes de tasas, libros y platos en diferentes posiciones y las imágenes correspondientes a los puntos de agarre de cada objeto. Todas las imágenes

están en el formato *.png (Portable Network Graphics). Esta data está disponible en [16] y es uso libre para realizar trabajos en visión por computador. En la figura 3-20 se muestra la imagen sintética de un libro usada en el entrenamiento.

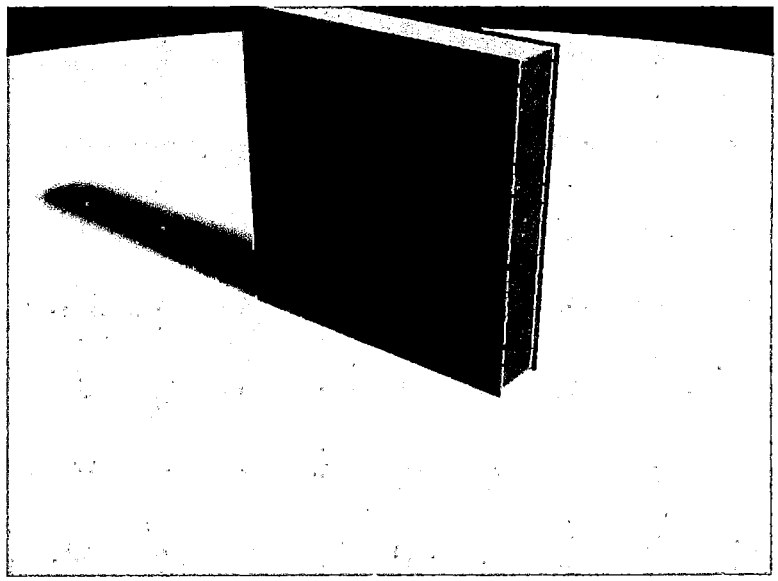


Figura 3-20. Imagen sintética de un libro.

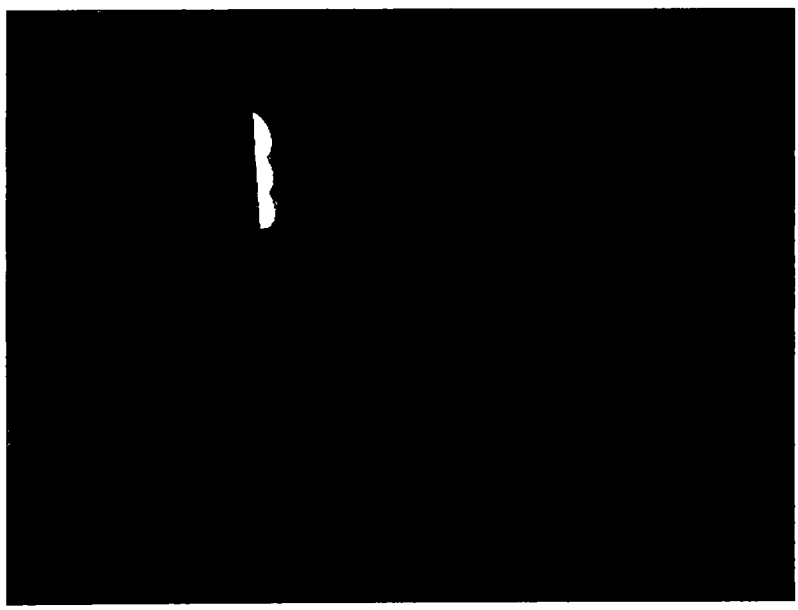


Figura 3-21. Imagen del punto de agarre del libro.

En la figura 3-21 se muestra la imagen del punto de agarre del objeto que se muestra en la figura 3-20. El punto de agarre está indicado por un área de

color blanco formada por pixeles de valor 255 en el entrenamiento será considerada el objetivo o target. La figura 3-22 muestra la imagen del libro y su correspondiente punto de agarre.

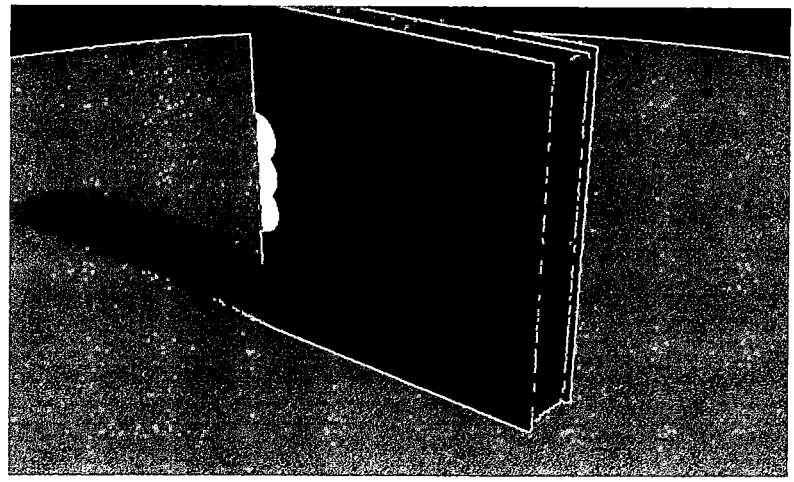


Figura 3-22. Imagen del objeto interceptada con su punto de agarre.

3.4.2. División de las imágenes en regiones

La imagen del objeto se dividirá en regiones 10x10 pixeles (ver figura 3-23) y cada región tendrá un vector de características con el cual se podrá inferir si dicha región pertenece o no a un punto de agarre.

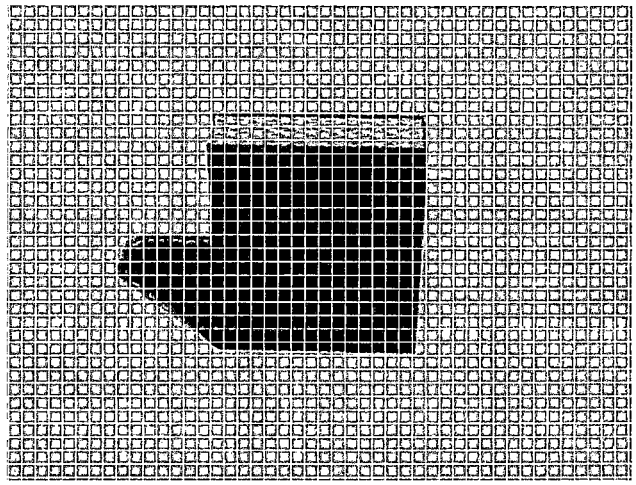


Figura 3-23. Imagen del objeto dividida en regiones de 10x10 pixeles.

La imagen del punto de agarre también dividirá en regiones de la misma dimensión y cada región tomará un valor de 1 si tiene al menos un pixel de valor 255 (color blanco que es color con la que se identifica el área del punto de agarre) y tomará cero en caso contrario. De esta forma se asignará el valor de 0 ó 1 a las regiones de acuerdo a la pertenencia al punto de agarre.

3.4.3. Cálculo de características de cada región

Para empezar a calcular el vector de características de cada región se procede a transformar la imagen del espacio de color RGB al espacio de color YCbCr (donde Y es la intensidad de canal, Cr y Cb son los canales de color). Primero se calcula la información de la textura al canal de intensidad (canal Y) aplicando 9 filtros texturas (Laws) mediante una convolución discreta con la matriz correspondiente al canal Y. Se aplicaron los nueve filtros que se muestran a continuación usando la operación de convolución.

$$\begin{matrix} L'_3L_3 & L'_3E_3 & L'_3S_3 \\ E'_3L_3 & E'_3E_3 & E'_3S_3 \\ S'_3L_3 & S'_3E_3 & S'_3S_3 \end{matrix}$$

Operaciones de convolución:

$$H_{(:,:,1)} = L'_3L_3 * Y$$
$$H_{(:,:,2)} = L'_3E_3 * Y$$
$$H_{(:,:,3)} = L'_3S_3 * Y$$
$$H_{(:,:,4)} = E'_3L_3 * Y$$
$$H_{(:,:,5)} = E'_3E_3 * Y$$
$$H_{(:,:,6)} = E'_3S_3 * Y$$
$$H_{(:,:,7)} = S'_3L_3 * Y$$
$$H_{(:,:,8)} = S'_3E_3 * Y$$
$$H_{(:,:,9)} = S'_3S_3 * Y$$

La información relacionada a la orientación de los bordes se obtiene aplicando 6 filtros de bordes orientados también al canal de intensidad Y. Se

aplican 6 filtros de bordes orientados en diferentes ángulos (0°, 30°, 60°, 90°, 120° y 150°) también con una convolución con la matriz Y.

$$H_{(:,:,10)} = \mathrm{NB}_1 * Y$$
$$H_{(:,:,11)} = NB_2 * Y$$
$$H_{(:,:,12)} = NB_3 * Y$$
$$H_{(:,:,13)} = NB_4 * Y$$
$$H_{(:,:,14)} = NB_5 * Y$$
$$H_{(:,:,15)} = NB_6 * Y$$

Al final la energía se calcula elevando al cuadrado cada uno de los elementos de la matriz H de dimensiones 480x640x15.

$$E_H = H.^{\wedge 2}$$

En la tabla 3-2 se muestra el total de filtros que aplicarán a las imágenes sintéticas para extraer las características. En la figura 3-24 se muestra la aplicación de los cuatro primeros filtros a la imagen sintética de libro mostrada anteriormente (ver figura 3-20).

Características por cada pixel	Cantidad
Filtros de texturas (Laws)	9
Filtros de bordes orientados	6
Total	15

Tabla 3-2. Filtros aplicados.

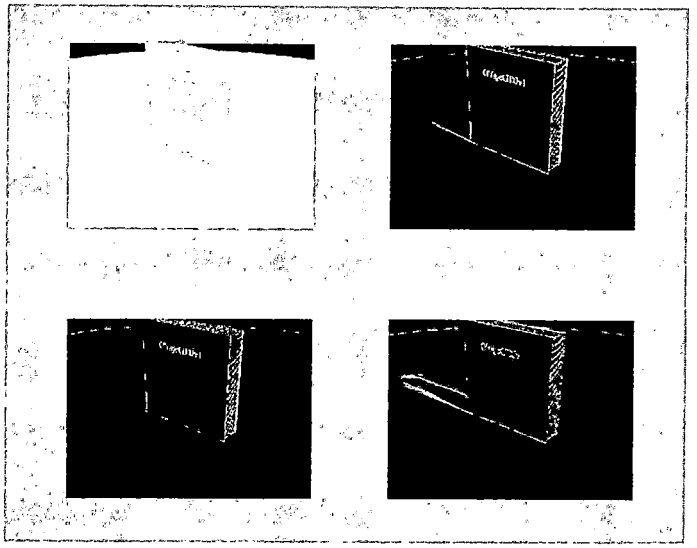


Figura 3-24. Matrices resultantes de la aplicación de algunos filtros.

Después de aplicar los 15 filtros para calcular las características al canal Y de la imagen del objeto y obtener la matriz E_H de dimensiones (480x640x15), se toma cada una de las matrices de las 15 matrices $H_{(:,:,i)}$ de dimensiones 480x640 y se dividen también en regiones de 10 x 10 elementos. Cada región tomará el valor numérico igual a la suma de todos sus elementos. Estas serán las primeras 15 características de cada región. Debido a que las imágenes presentan diferentes características en diferentes escalas se aplicarán los filtros anteriormente mencionados a la imagen del objeto pero a escalas diferentes, en este caso 1/3 y 1/9. De esta forma de obtendrán 30 características más para cada región generando en total un vector de 45 valores por cada región.

3.4.4. Adición de las características de las regiones vecinas

Al vector de características de cada región calculado anteriormente se le agregarán las características de sus regiones vecinas (solo las primeras 15 características calculadas) aumentando la dimensión de su vector de características dependiendo de la cantidad de regiones vecinas que se consideren. Las regiones vecinas se pueden escoger de acuerdo a las siguientes formas, con las cuales se ha experimentado en el entrenamiento.

- **La primera forma**

Se toman 24 regiones alrededor de la región central (ver figura 3-25) para tomar las características adicionales a las características ya calculadas.

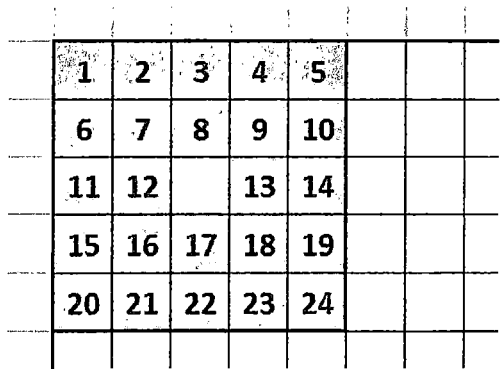


Figura 3-25. Primera forma de tomar las regiones adicionales

- **La segunda forma**

La segunda forma de tomar características adicionales tiene la siguiente forma (ver figura 3-26). El número de regiones evaluadas alrededor es 16.

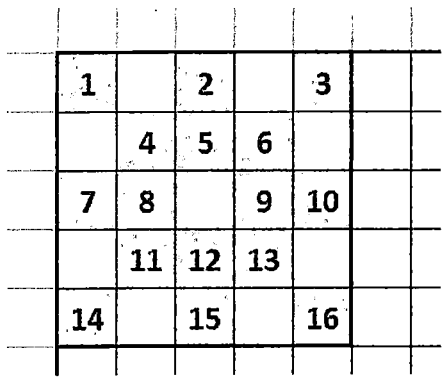


Figura 3-26. Segunda forma.

- **La tercera forma**

En la tercera forma se consideran las regiones en forma intercalada. Se evalúan 12 regiones alrededor, tal como se muestra en la figura 3-27.

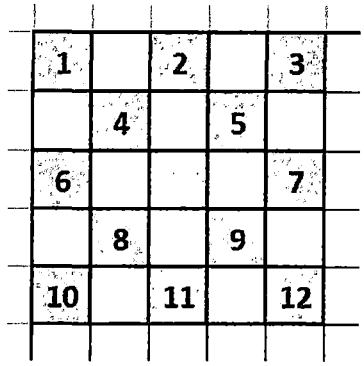


Figura 3-27. Tercera forma.

- **La cuarta forma**

Se toman regiones alrededor pero en direcciones 45, 135, 225 y 315 grados. Se evalúan 8 regiones (ver figura 3-28).

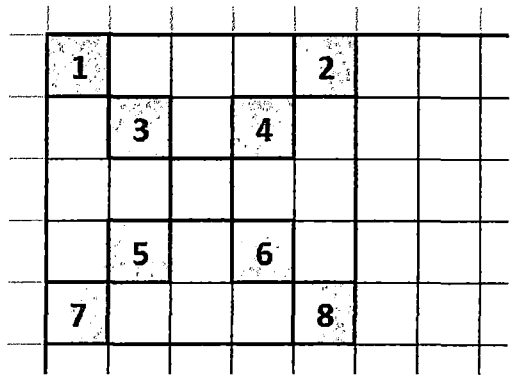


Figura 3-28. Cuarta forma.

La primera diferencia entre las todas las formas de tomar la vecindad está en la cantidad de información con la que se va a contar para el entrenamiento e inferencia. Probar las diferentes formas nos llevará a encontrar cual es la forma que nos entrega suficiente información para el entrenamiento analizando con imágenes del entrenamiento y en base a algunos indicadores. El objetivo es no contar con un exceso de información, claro está que esto depende también del tipo de algoritmo de aprendizaje que se usa.

Después del entrenamiento, en el proceso de inferencia también se calculará el mismo vector de características (la región y su vecindad). De esta

forma la inferencia punto de agarre no depende solo de las características de una región en análisis sino también de las regiones vecinas. Se calculará el vector de características a todas las regiones de la imagen del objeto. Después se seleccionarán para el entrenamiento solo las regiones del objeto y punto de agarre y sus respectivas características. Analizará más adelante hay una influencia de la selección de la vecindad en el entrenamiento

- **Total de características calculadas**

El proceso de cálculo de características de cada región ha ido incrementando el número de características en cada etapa. La tabla 3-3 indica el número total de características por región calculadas para el entrenamiento e inferencia usando la tercera forma de vecindad.

Características para cada forma	Forma 1	Forma 2	Forma 3	Forma 4
Región centrada en tres escalas	15*3	15*3	15*3	15*3
Regiones alrededor	15*24	15*16	15*12	15*8
Total	405	285	225	165

Tabla 3-3. Características en total para cada forma.

Ahora se cuenta con suficiente data para el entrenamiento. En la siguiente etapa seleccionaremos los índices de las regiones con los que se va a trabajar en el entrenamiento.

3.4.5. Selección de índices del objeto e índices del punto de agarre

Realizando una segmentación mediante umbral se procederá a identificar los índices de las regiones que pertenecen al objeto. Los índices de las regiones del punto de agarre los obtenemos de la imagen del punto de agarre que está binarizada. Estos índices nos servirán para el entrenamiento del algoritmo de aprendizaje. Solo se conservarán los índices de las regiones del objeto y del punto de agarre, y sus características. Los índices de las regiones del resto de la imagen no son de nuestro interés.

La segmentación del área del objeto consiste en transformar la imagen del objeto al formato HSV. Se calcula el promedio de los valores de los índices del canal H. Los índices mayores al valor promedio tomarán valor 1 y el resto de índices se le asignará valor 0. Esto da por resultado una imagen binaría (ver Figura 3-29) donde los pixeles de color blanco (valor 1) son los que pertenecen al objeto.

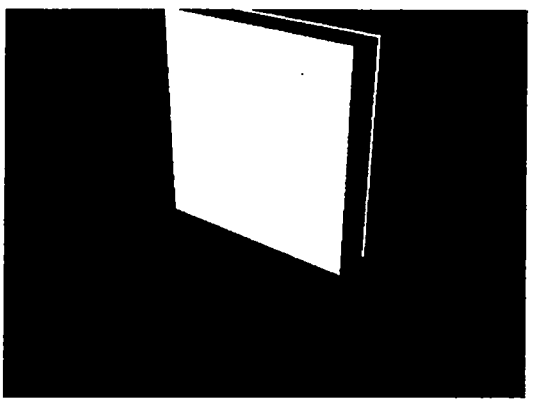


Figura 3-29. Imagen segmentada de un objeto.

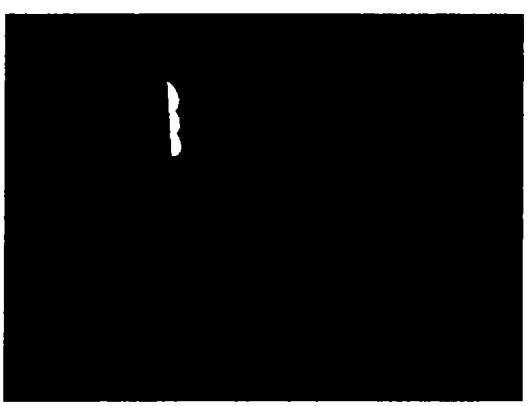


Figura 3-30. Imagen binaria del punto de agarre del libro.

Ahora los índices de las regiones que pertenezcan al objeto y no pertenezcan al punto de agarre serán llamados índices negativos (valor 0) y los que pertenezcan sólo al punto de agarre (ver Figura 3-30) serán llamados índices positivos (valor 1). La cantidad de índices positivos y negativos varía de acuerdo a la imagen del objeto.

Se almacenarán las características de las regiones positivas y negativas (ver figura 3-31). Las características de las regiones positivas y negativas serán los datos de entrada y los valores 0 ó 1 serán la salida deseada o target (ver figura 3-32).

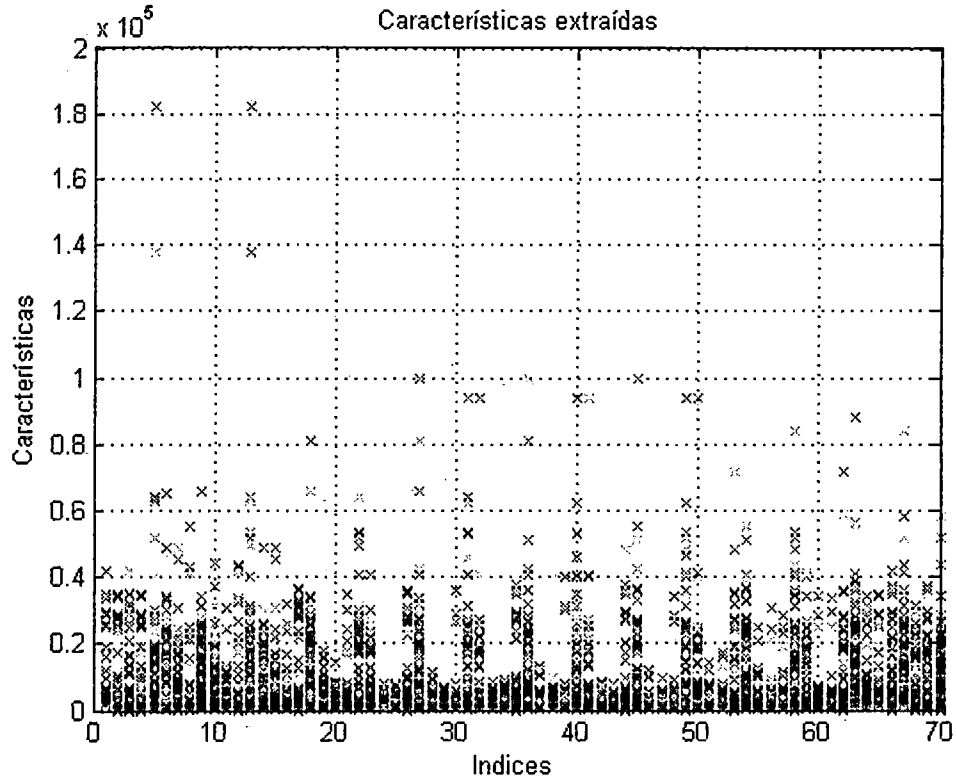


Figura 3-31. Vector de características de la imagen de un objeto.

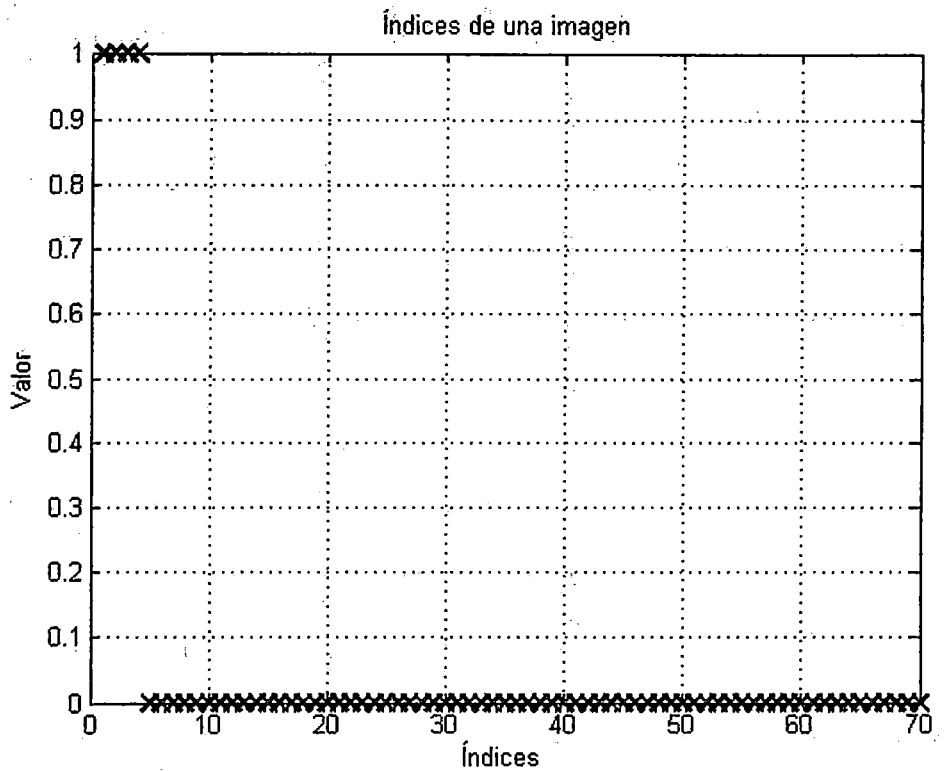


Figura 3-32. Vector de los índices de positivos y negativos.

Al entrenar con varias imágenes (modo batch) se realizará un reordenamiento de todos los datos colocando primero a los índices positivos (valor 1 o punto de agarre) y en segundo lugar a los índices negativos. Esto se realizará antes del entrenamiento de cualquier plataforma de aprendizaje.

3.4.6. Entrenamiento

Una vez que contamos con la data (índices positivos, negativos y sus características) entrenaremos los algoritmos de aprendizaje para que clasifique si un índice pertenece o no a un punto de agarre de acuerdo a sus características extraídas. Lo que en una forma general es relacionar una imagen de un objeto con su punto de agarre.

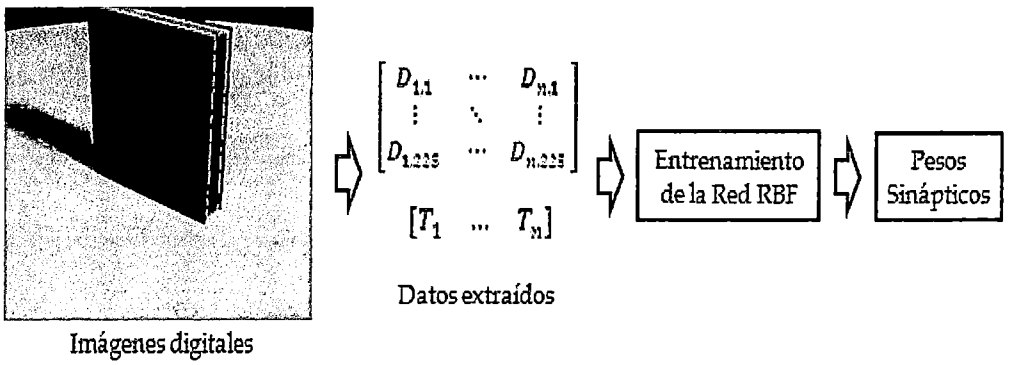


Figura 3-33. Esquema de los procesos de extracción de datos y de entrenamiento.

La figura 3-33 muestra de forma esquemática y resumida los procesos que se sigue para lograr el aprendizaje para una red RBF. Este concluye con la obtención de pesos sinápticos de la red. En este trabajo se usarán tres algoritmos de aprendizaje, Regresión Logística, Redes Multicapa Perceptrón y Redes con Función de Base Radial. Dependiendo del algoritmo aprendizaje se ejecutará los diferentes algoritmos de entrenamiento y establecerán sus respectivos parámetros de diseño.

3.5. Algoritmo de Reconstrucción Tridimensional

El algoritmos de reconstrucción tridimensional implica una serie de sub algoritmos previos necesarios para pode realizar la reconstrucción tridimensional. El primero es el algoritmo de calibración de cámaras digitales.

3.5.1. Algoritmo de calibración de cámaras digitales

A continuación se realizará la descripción del algoritmo de calibración para la obtención de los parámetros intrínsecos y extrínsecos de ambas cámaras. Estos parámetros nos servirán para realizar la reconstrucción tridimensional del punto de agarre o cualquier punto que haya sido proyectado en dos o más imágenes. En la figura 3-34 se muestra el diagrama de flujo del proceso de calibración.

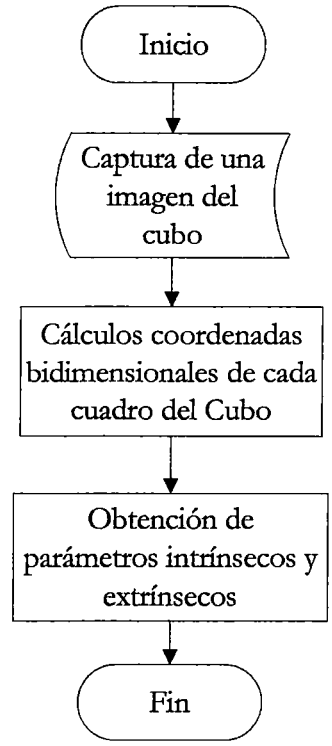


Figura 3-34. Diagrama de flujo del proceso de calibración.

En el proceso de calibración se usará un cubo como una herramienta ya que sus aristas están distribuidas en forma similar a los ejes de coordenadas X, Y y Z, se conoce sus medidas y además cada cara tiene una cuadricula de ajedrez (ver figura 3-35) para facilitar la identificación de suficientes puntos en cada plano XY, YZ y ZX.

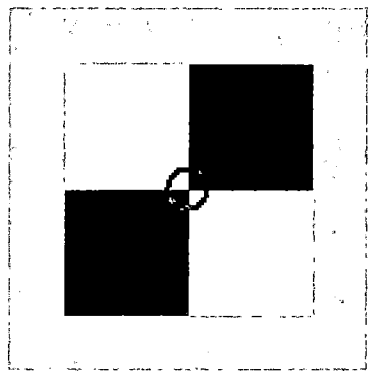


Figura 3-35. Punto a identificar.

- **Datos del cubo**

El cubo tiene 12.5 cm de lado y cada cuadrado de la cuadricula ubicada en sus caras tiene 2.5 cm de lado. Consideraremos cada lado del cuadrado como una unidad de medida (sabiendo anticipadamente que es 2.5 cm). Al momento de los cálculos se multiplicará por este factor para que todas las medidas estén en centímetros. El sistema de coordenadas referencial se fijará a un vértice del cubo y los ejes X, Y y Z se alinearán a sus respectivas aristas (ver figura 3-36).

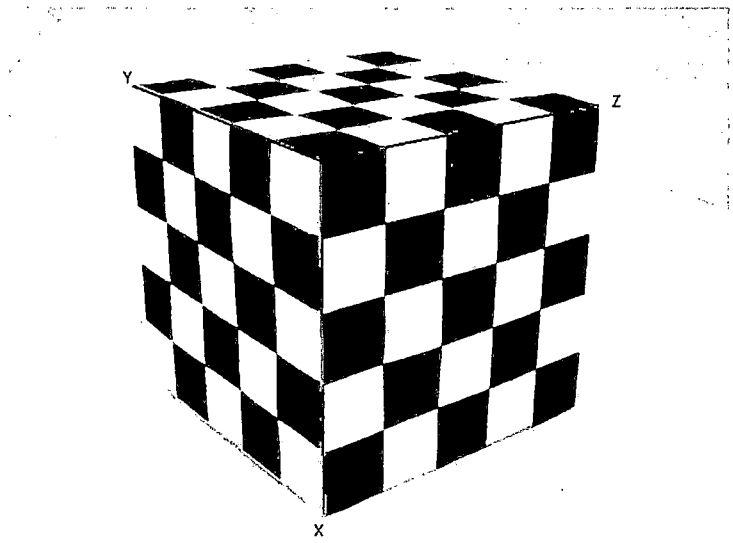


Figura 3-36. El cubo y el sistema de coordenadas referencial.

Debido a la construcción del cubo sabemos las coordenadas tridimensionales de cada vértice de cada cuadrado perteneciente a cada plano (XY, YZ y ZX) con respecto a sistema de coordenadas que se ha elegido como referencia. En la figura 3-37 se ha graficado los puntos tridimensionales de los planos XY y YZ considerados en la calibración de las cámaras.

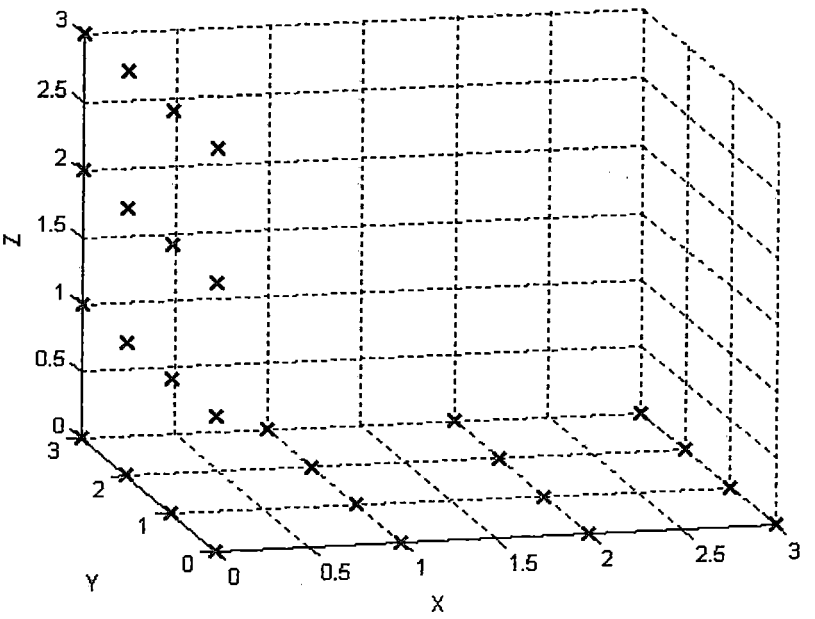


Figura 3-37. Puntos del cubo en el espacio tridimensional.

De la imagen capturada obtenemos las coordenadas (u, v) de las proyecciones de los puntos tridimensionales del cubo en el plano de la imagen. Estas coordenadas están expresadas en pixeles. En la figura 3-38 se muestra como se toman las coordenadas en una imagen.

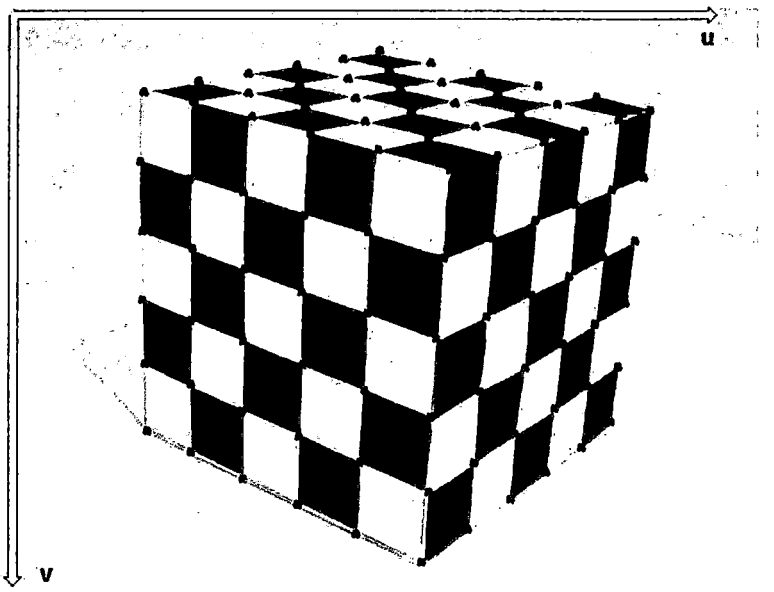


Figura 3-38. Puntos y ejes de coordenadas en una imagen.

Conociendo las coordenadas tridimensionales de todos los puntos visibles del cubo con respecto al sistema de coordenadas que se ha elegido para realizar la calibración (sistema de coordenadas referencial) y las coordenadas (u, v) en la imagen capturada evaluaremos las siguientes ecuaciones para obtener los parámetros de la cámara.

Recogiendo n puntos (u, v) y conociendo sus coordenadas tridimensionales con respecto al sistema de coordenadas referencial, tenemos entonces un sistema lineal homogéneo de $2n$ ecuaciones en los 12 coeficientes de M.

$$\rho = \begin{pmatrix} P_1^T & 0^T & -u_1 P_1^T \\ 0^T & P_1^T & -v_1 P_1^T \\ \cdots & \cdots & \cdots \\ P_n^T & 0^T & -u_n P_n^T \\ 0^T & P_n^T & -v_n P_n^T \end{pmatrix} \text{ y } m = \begin{pmatrix} m_1 \\ m_2 \\ m_3 \end{pmatrix}$$

- **Estimación de parámetros intrínsecos y extrínsecos**

Una vez que M ha sido hallada, los parámetros de la cámara pueden ser derivados de la siguiente forma:

Escribiendo $M = (Ab)$ denotando con las filas de A y con las filas de la matriz de rotación R,

$$\rho(A\ b) = K(R\ t) \leftrightarrow \rho \begin{pmatrix} a_1^T \\ a_2^T \\ a_3^T \end{pmatrix} = \begin{pmatrix} \alpha \cdot r_1^T + \alpha \cdot \cot(\theta) \cdot r_2^T + u_o r_3^T \\ \frac{\beta}{sen(\theta)} r_2^T + v_o \cdot r_3^T \\ r_3^T \end{pmatrix}$$

Las columnas r_1^T, r_2^T y r_3^T de la matriz de rotación son vectores unitarios perpendiculares.

$$\begin{cases} \rho = \varepsilon/|a_3| \\ r_3 = \rho \cdot a_3 \\ u_o = \rho^2(a_1 \cdot a_3) \\ v_o = \rho^2(a_2 \cdot a_3) \end{cases} \quad \varepsilon = \pm 1$$

Asumiendo que θ tiene un valor cercano a $\pi/2$.

$$\rho^2(a_1 \times a_3) = -\alpha r_2 - \alpha \cot\theta \, r_1 \rightarrow \rho^2|a_1 \times a_3| = \frac{|\alpha|}{\sin\theta}$$

$$\rho^2(a_2 \times a_3) = \frac{\beta}{\sin\theta} r_1 \rightarrow \rho^2|a_2 \times a_3| = \frac{|\beta|}{\sin\theta}$$

Utilizando una vez más el hecho de que r_1^T, r_2^T y r_3^T son vectores unitarios perpendiculares.

$$\begin{cases} \cos\theta = -\dfrac{(a_1 \times a_3) \cdot (a_2 \times a_3)}{|a_1 \times a_3||a_2 \times a_3|} \\ \alpha = \rho^2|a_1 \times a_3| \sin\theta \\ \beta = \rho^2|a_2 \times a_3| \sin\theta \end{cases}$$

$$\begin{cases} r_1 = \dfrac{\rho^2 \sin\theta}{\beta}(a_2 \times a_3) = \dfrac{1}{|a_2 \times a_3|}(a_2 \times a_3) \\ r_2 = r_3 \times r_1 \end{cases}$$

La ambigüedad es resuelta llamando y el signo de es generalmente conocido con anterioridad.

Ejemplo 3-3: A continuación se que hemos capturado la imagen izquierda y derechas de una serie de puntos del espacio tridimensional con respecto al cubo. Conocemos las coordenadas tridimensionales de esos punto, por lo tanto procederemos a encontrar la matrices M que contienes los parámetros intrínsecos y extrínsecos de ambas cámaras.

Los puntos capturados para la imagen izquierda.

(u_i, v_i) = [351 255;302 281;245 314;182 350;368 310;313 345;251 385;439 338;386 377;325 422;516 368;468 412;409 463; ...];

Los puntos capturados para la imagen derecha.

(u_d, v_d) = [268 242;208 266;142 295;66 326;264 299;196 331;119 366;325 333;256 368;178 409;393 370;325 411;247 457; ...];

Con los datos mostrados anteriormente se calculan las siguientes matrices.

Para la cámara izquierda:

$$M_i = 10^3 \cdot \begin{pmatrix} 0.8695 & -0.6070 & 0.1457 & -4.5078 \\ -0.1574 & -0.1568 & 0.9797 & -3.2543 \\ 0.0007 & 0.0005 & 0.0004 & -0.0128 \end{pmatrix}.$$

Para la cámara derecha:

$$M_d = 10^3 \cdot \begin{pmatrix} 0.9140 & -0.5091 & 0.1259 & -4.5078 \\ -0.1491 & -0.1862 & 0.9797 & -3.0851 \\ 0.0007 & 0.0006 & 0.0004 & -0.0128 \end{pmatrix}.$$

3.5.2. Algoritmo de correlación

Para nuestro caso usaremos la correlación para la búsqueda del punto inferido en la segunda imagen, también conocido como ***template matching***. Para esto tenemos que seleccionar una ventana o plantilla con la cual se realizará la búsqueda en la segunda imagen, el tamaño de la ventana es importante ya que si la venta tiene características únicas se podrá encontrar más fácilmente la correspondencia. De la primera imagen tenemos las coordenadas del punto inferido y lo que se hace es recortar una ventana de tamaño 200 x 200 pixeles tomando como centro dichas coordenadas. La correlación se realizará entre la ventana ya recortada y la segunda imagen mediante la aplicación de la siguiente formula:

$$\gamma(u,v) = \frac{\sum_{x,y}[f(x,y) - \bar{f}_{u,v}][t(x-u,y-v) - \bar{t}]}{\{\sum_{x,y}[f(x,y) - \bar{f}_{u,v}]^2 \sum_{x,\square}[t(x-u,y-v) - \bar{t}]^2\}^{0.5}}$$

Donde:

f: Es la imagen

$\bar{t}$: Es la media de la plantilla o ventana

$\bar{f}_{u,v}$: Es la media de $f(x,y)$ en la región dentro de la plantilla o ventana.

$\gamma(u,v)$: Es matriz de coeficientes de correlación

Al conocer la posición de la ventana o plantilla en la segunda imagen y al conocer la ubicación del punto con respecto a la ventana podemos conocer las coordenadas del punto en la segunda imagen.

Ejemplo 3-4: Tenemos la imagen izquierda de un objeto del cual hemos identificado el punto de agarre. Queremos conocer cuál es la ubicación de este punto del objeto en la segunda imagen (imagen derecha). Para eso vamos a seleccionar la ventana que contenga al punto inferido. La ventana tendrá dimensiones de 200 x 200 pixeles y el punto inferido estará en el centro. Esta ventana será buscada en la segunda imagen tal como se muestra en la figura 3-39.

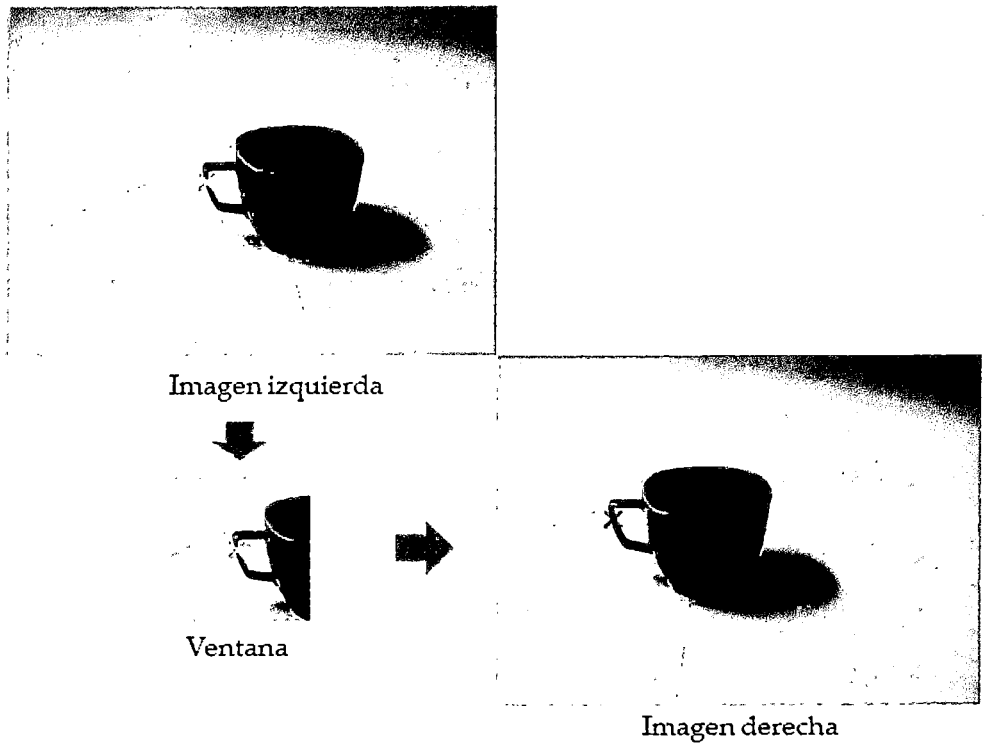


Figura 3-39. Esquema de la correlación entre dos imágenes.

Una vez encontrada la ventana más similar y sabiendo las coordenadas relativas del punto 1 con respecto a la ventana podemos conocer las coordenadas en la segunda imagen. Para este ejemplo el punto en la imagen izquierda es $p_i = (261, 241)$ y el punto encontrado en la imagen derecha es $p_d = (153, 226)$.

3.5.3. Algoritmo de triangulación de puntos

Para realizar el cálculo de coordenadas tridimensionales de un punto en el espacio necesitamos las coordenadas en por lo menos dos imágenes capturadas de este punto y los parámetros de las cámaras con las cuales fueron tomadas. Asumimos que los parámetros intrínsecos y extrínsecos de ambas cámaras ya han sido calculados por lo tanto solo quedaría conocer las coordenadas de dicho punto en ambas imágenes para realizar la triangulación. La figura 3-40 muestra el diagrama de flujo que sigue para el cálculo de coordenadas tridimensionales.

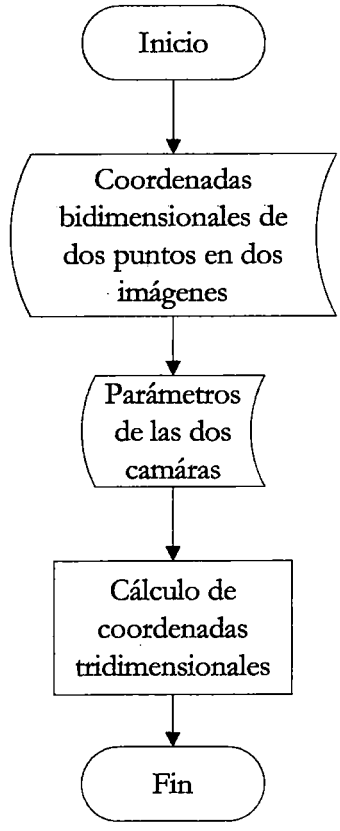


Figura 3-40. Diagrama de flujo de la triangulación.

En la figura 3-41 se muestra dos imágenes (izquierda y derecha) de un mismo objeto y el sistema de coordenadas con el cual se hizo la calibración y se obtuvo los parámetros de las cámaras.

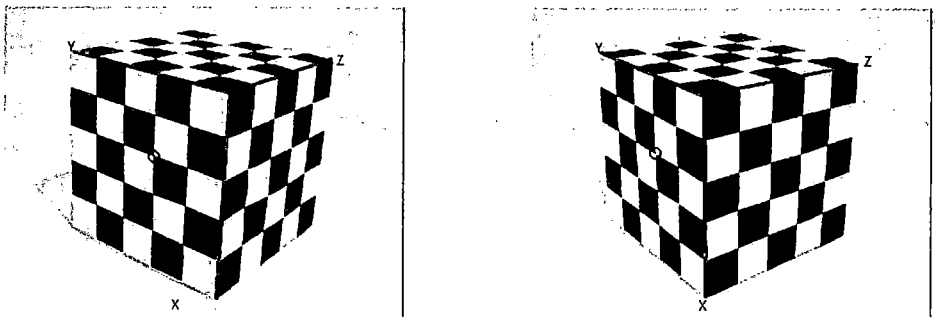


Figura 3-41. Imagen izquierda y derecha del cubo.

Por ejemplo queremos saber las coordenadas tridimensionales del punto marcado con azul en ambas imágenes. Sean p_i y p_d (marcados con el círculo azul) los puntos en ambas imágenes que corresponden a la proyección de un mismo punto del espacio tridimensional en dos imágenes o vistas, para nuestro caso la imagen izquierda y la imagen derecha. Estos puntos tienen coordenadas (u_i, v_i) y (u_d, v_d) con respecto a plano normalizado de cada imagen, izquierda y derecha respectivamente. Formaremos las siguientes matrices:

Para la cámara izquierda $[p_i] = \begin{pmatrix} 0 & -1 & v_i \\ 1 & 0 & -u_i \\ -v_i & u_i & 0 \end{pmatrix}$

Para la cámara derecha $[p_d] = \begin{pmatrix} 0 & -1 & v_d \\ 1 & 0 & -u_d \\ -v_d & u_d & 0 \end{pmatrix}$

De acuerdo a la teoría se tiene que cumplir que:

$$[p_i] \cdot M_i \cdot P_i = 0$$
$$[p_d] \cdot M_d \cdot P_d = 0$$

Agrupando las matrices:

$$\begin{pmatrix} [p_i] \cdot M_i \cdot P_i \\ [p_d] \cdot M_d \cdot P_d \end{pmatrix} (P) = 0$$

Para calcular el punto P que representa las coordenadas tridimensionales se minimiza:

$$P^T \cdot M^T \cdot M \cdot P$$

Donde:

$$M = \begin{pmatrix} [p_i] \cdot M_i \\ [p_d] \cdot M_d \end{pmatrix}$$

Se calcula el auto vector Av correspondiente al menor auto valor de $M^T \cdot M$.

$$Av = (au_1 \quad au_2 \quad au_3 \quad au_4)^T$$

El punto $P_{(x,y,z)}$ se calcula de:

$$P_{(x,y,z)} = \frac{(au_1 \quad au_2 \quad au_3)^T}{au_4}$$

Ejemplo 3-5: Con las coordenadas (u_i, v_i) y (u_d, v_d) de un mismo punto del objeto en ambas imágenes y las matrices M_i y M_d que contienen los parámetros intrínsecos y extrínsecos de cada cámara tenemos datos suficientes para conocer las coordenadas tridimensionales de ese punto.

Datos anteriormente calculados: $p_i = (261, 241)$ y $p_d = (153, 226)$.

Formamos la matriz para la cámara izquierda $[p_i] = \begin{pmatrix} 0 & -1 & 241 \\ 1 & 0 & -261 \\ -241 & 261 & 0 \end{pmatrix}$

Formamos la matriz para la cámara derecha $[p_d] = \begin{pmatrix} 0 & -1 & 226 \\ 1 & 0 & -153 \\ -226 & 153 & 0 \end{pmatrix}$

Para calcular el punto P que representa las coordenadas tridimensionales se minimiza:

$$P^T \cdot M^T \cdot M \cdot P$$

Donde:

$$M = \begin{pmatrix} [p_i] \cdot M_i \\ [p_d] \cdot M_d \end{pmatrix}$$

Se calcula el auto vector Av correspondiente al menor auto valor de $M^T \cdot M$.

$$Av = (0.8034 \quad 0.1220 \quad 0.4175 \quad 0.4067)$$

El punto $P_{(x,y,z)}$ se calcula de:

$$P_{(x,y,z)} = \frac{(Au_1 \quad Au_2 \quad Au_3)^T}{Au_4}$$

$$P_{(x,y,z)} = (9.8780 \quad 1.4996 \quad 5.1337)$$

CAPÍTULO IV

ANÁLISIS DE RESULTADOS

Empezamos este capítulo estableciendo los conceptos básicos para incorporar el sistema de visión a un brazo robótico de 5 grados de libertad de aplicación industrial. Esto lo haremos a manera de ejemplo porque el sistema de visión se puede incorporar a diferentes tipos de brazos robóticos.

Posteriormente, analizaremos los resultados del entrenamiento usando diferentes formas de tomar los datos de regiones vecinas y tres tipos de algoritmos de aprendizaje supervisado (Regresión Logística, Redes Multicapa Perceptrón y Redes con Función de Base Radial). Primero se evaluará el aprendizaje de los algoritmos con la data usada en el entrenamiento, luego con imágenes sintéticas y finalmente con imágenes de objetos reales.

Al final se revisarán los resultados de la triangulación. Las primeras pruebas se realizaron triangulando ciertos puntos dentro del espacio tridimensional y después triangulando puntos de agarre de objetos que se usaron en las pruebas de inferencia. En ambos casos se trata de evaluar la medida del error en el cálculo de las coordenadas tridimensionales.

4.1. Incorporación del Sistema de Visión a un Brazo Robótico

Al momento de incorporar un sistema de visión para la obtención de coordenadas tridimensionales a un brazo robótico debemos tener claro ciertos

conceptos adicionales que dependen también de este. Para ello, usaremos como ejemplo un brazo robótico que existe en el mercado, el Mitsubishi Movemaster RV-M1 (ver figura 4-1), con el cual se pueden desarrollar diferentes aplicaciones.

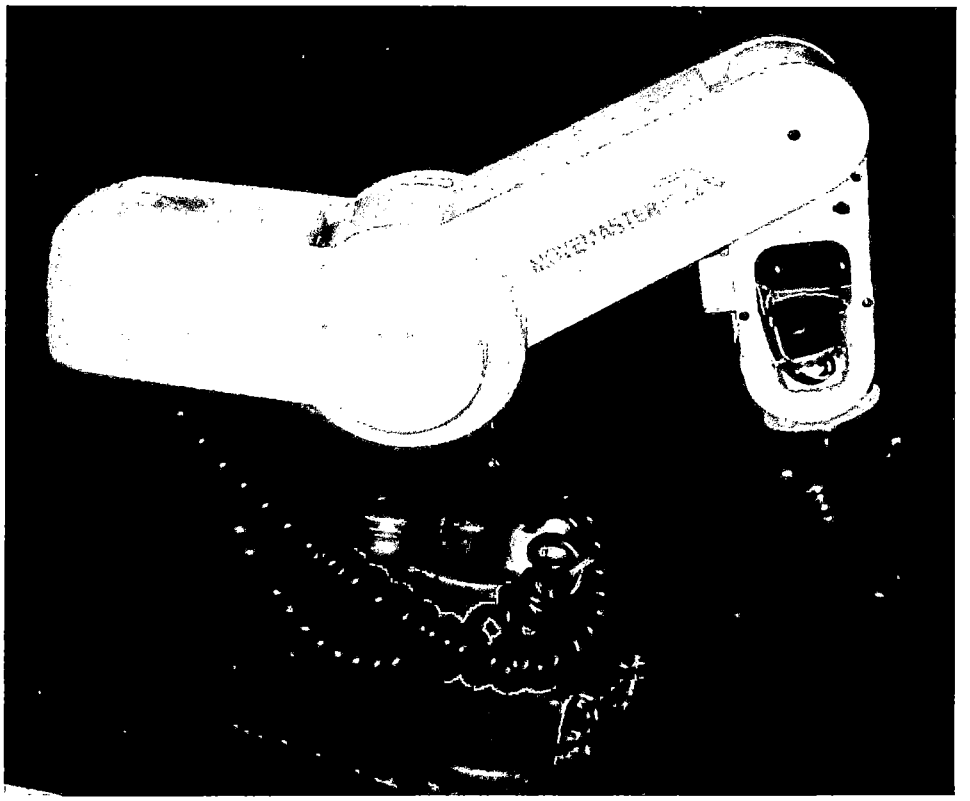


Figura 4-1. Brazo robótico Movemaster RV-M1 5GDL.

4.1.1. Características del brazo robótico

El brazo robótico que se muestra en la figura 4-1 tiene 5 grados de libertad (sin contar los grados de libertad del efector final) y es controlado por servo motores DC. En la tabla 4-1 se muestra el cuadro de especificaciones proporcionadas por el fabricante.

MOVEMASTER RV - M1		
Construcción		Vertical, articulada
Grados de libertad		5 (no incluye el efector final)
Sistema eléctrico de impulso		Motores servo DC
Alcance		250 + 160 mm
Rango de operación	Rotación de cintura	300° (max. 120°/s)
	Rotación de hombro	130° (max. 72°/s)
	Rotación de codo	110° (max. 109°/s)
	Rotación de muñeca 1 (Pitch)	±90° (max. 100°/s)
	Rotación de muñeca 2 (Roll)	±180° (max. 163°/s)
Velocidad máxima de trayectoria		1,000 mm/s
Capacidad de carga		1,2 kg (no incluye el efector)
Repetitividad de posición		±0,3 mm
Posición de instalación		Horizontal
Temperatura de trabajo		5° C - 40° C
Peso		19 Kg

Tabla 4-1. Cuadro de especificaciones del brazo robótico.

En la figura 4-2 se muestra esquemáticamente el espacio de trabajo del brazo robótico desde una vista de planta y una vista lateral.

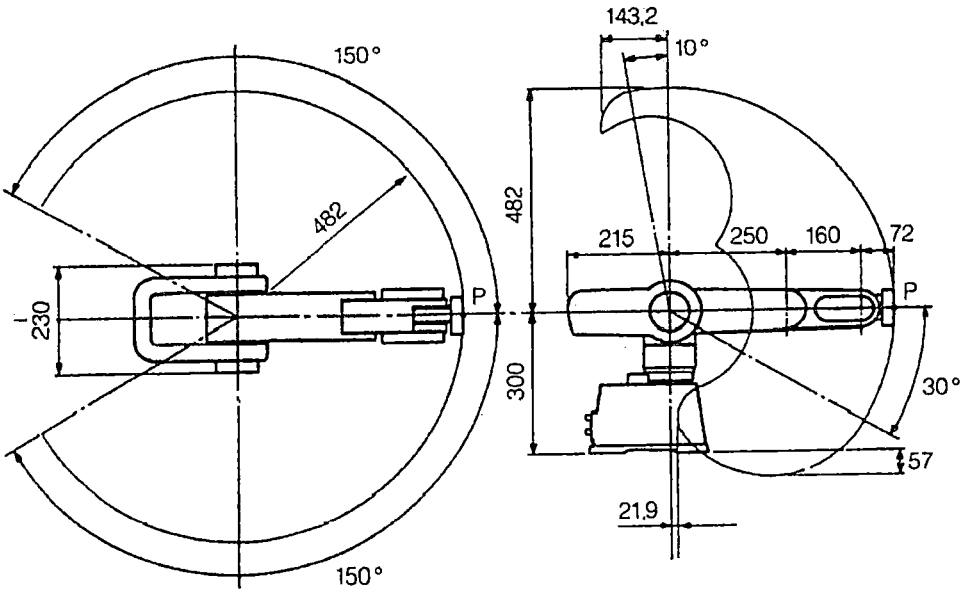


Figura 4-2. Espacio de trabajo del brazo robótico Movemaster.

4.1.2. Información para la captura tridimensional

La información que el brazo robótico necesita para que pueda coger un objeto es básicamente: la posición, la dirección y la trayectoria que lleve al efector del brazo robótico de la posición donde se encuentra hacia la posición deseada. Aunque hay otras herramientas como la incorporación de sensores (haptic sensors) que podrían aumentar su eficiencia y efectuar labores más complicadas, este trabajo solo se enfoca en los aspectos básicos y en hacer ciertas consideraciones para explicar la incorporación del sistema de visión.

- **Sistema de coordenadas del brazo robótico**

El sistema de coordenadas del brazo robótico es el sistema desde el cual se referencia cualquier movimiento realizado por este. Al calcular las coordenadas del punto de agarre del objeto a manipular se tendrá que realizar una traslación y rotación con respecto a ese sistema, ya que estas están calculadas con respecto al sistema de coordenadas usado en la calibración. No obstante, estos cálculos se explicarán más adelante.

- **La posición del efector final**

Para que el brazo robótico logre capturar al objeto del cual se ha inferido su punto de agarre debe hacer coincidir el centro del efector final con las coordenadas del punto de agarre obtenidas mediante la triangulación. Pero para que esta operación esté completamente definida se necesita determinar la dirección y trayectoria para situar el efector en esta posición. En la figura 4-3 se muestra el efector final del brazo robótico y el punto de donde se va a manipular el objeto mostrado.

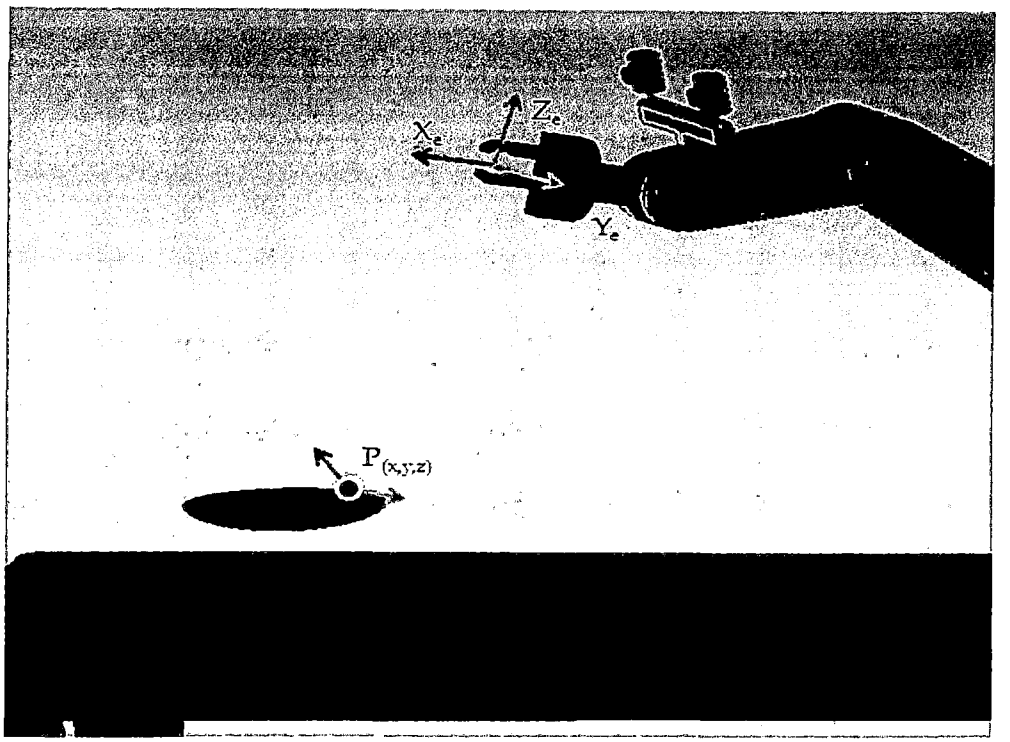


Figura 4-3. El efector final y el punto de agarre.

- **La dirección**

Una vez que se calculan las coordenadas tridimensionales de un punto de agarre pueden existir diferentes posibilidades de orientar el efector final. En la figura 4-3 se muestra una dirección posible del efector final para realizar la captura del objeto. Estas posibilidades pueden estar restringidas por las características físicas y visuales del objeto tratando siempre que se produzca un máximo contacto y reduciendo el deslizamiento.

Este trabajo no abarca la inferencia de la dirección del efector final cuando esté situado en el punto de agarre del objeto, por lo que, si se deseara implementar completamente se podría seguir más de un camino. El menos complejo es restringir para ciertos casos y para ellos establecer ciertas reglas heurísticas para poder obtener la dirección como las que se muestran a continuación. Mucho más complejo sería desarrollar algoritmos para estimar la orientación mediante imágenes o usando sensores o/y un escáner laser. Por ejemplo para el caso del libro, si consideramos que siempre está en posición de

cara vertical frontal se podría estimar la dirección Y_e considerándola también vertical. El efector puede aproximarse por la izquierda si el centroide del área segmentada del libro está ubicado a la derecha del punto inferido y por la derecha en el caso contrario.

Si conocemos la distribución tridimensional del objeto que se desea manipular, se puede conocer también la orientación del punto de agarre inferido realizando cálculos adicionales. La distribución tridimensional del objeto también se puede estimar a partir de las características visuales de este, ya sea por cálculos o por entrenamiento previo en base a data sintética, data real o una combinación de ambas [6].

- **La trayectoria**

La trayectoria puede ser encontrada en base a una serie de restricciones que dependen de la presencia de obstáculos como objetos o superficies y las características del brazo robótico como el espacio de trabajo (ver figura 4-2) y los grados de libertad (ver figura 4-4). Por lo tanto, se podrán emplear trayectorias simples como una trayectoria rectilínea cuando no haya la presencia de obstáculos o tener una trayectoria compleja debido a las restricciones cuando se está trabajando en ambiente complejo.

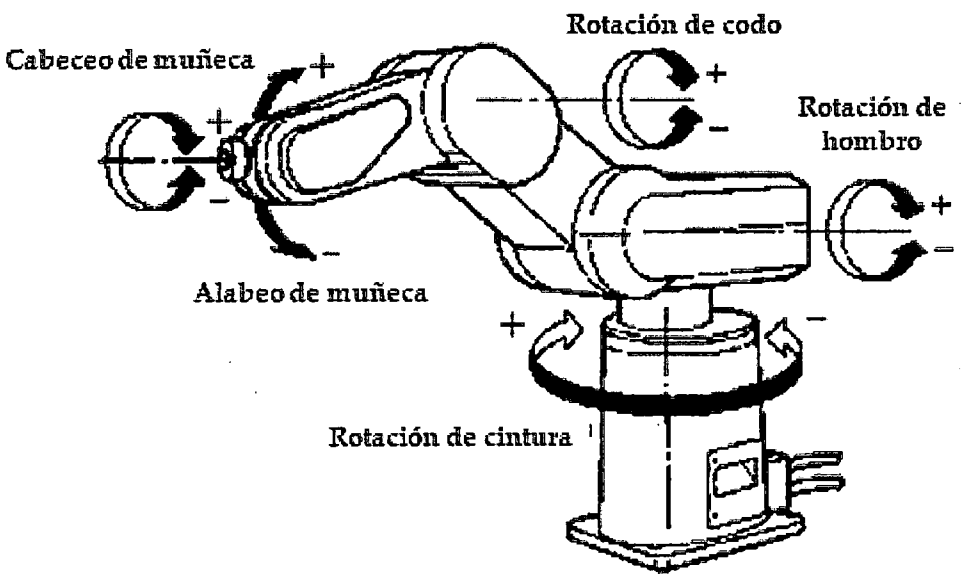


Figura 4-4. Grados de libertad del brazo robótico.

4.1.3. Posiciones posibles del par estéreo

El sistema de visión estéreo para el cálculo de coordenadas y el brazo robótico pueden trabajar en tres posibles posiciones. En la primera posición (ver figura 4-5) el par estéreo está fijo y por lo tanto su campo de visión también es fijo. En este caso la transformación de coordenadas con respecto al sistema de coordenadas del robot es más directa.

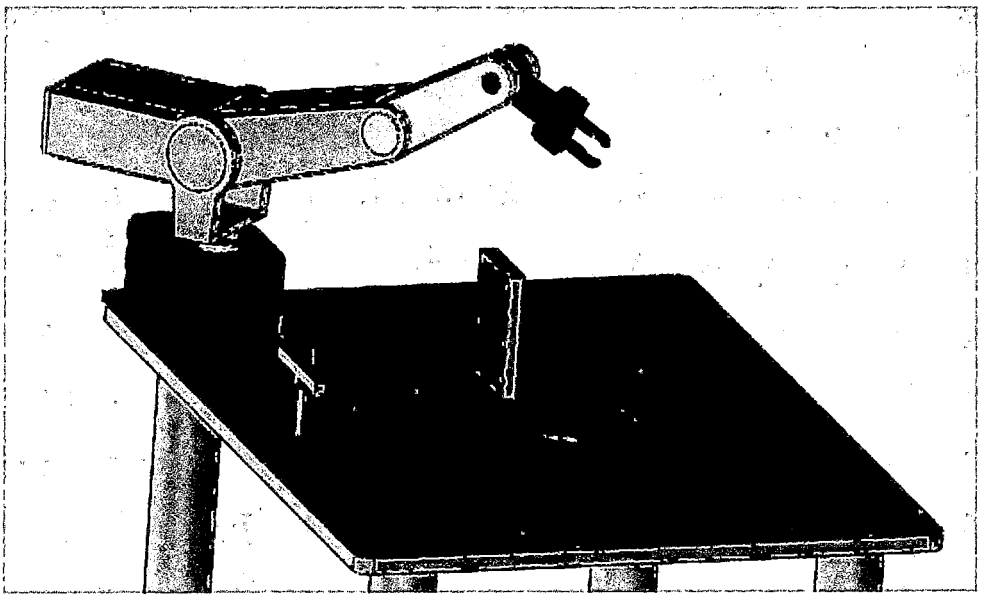


Figura 4-5. Primera posición, el par estéreo fijo.

En la siguiente posición (ver figura 4-6) el sistema estéreo está fijo al eslabón que está conectado a la base, por tanto solo puede rotar dentro del espacio de trabajo del robot ampliando el área de visión del par estéreo.

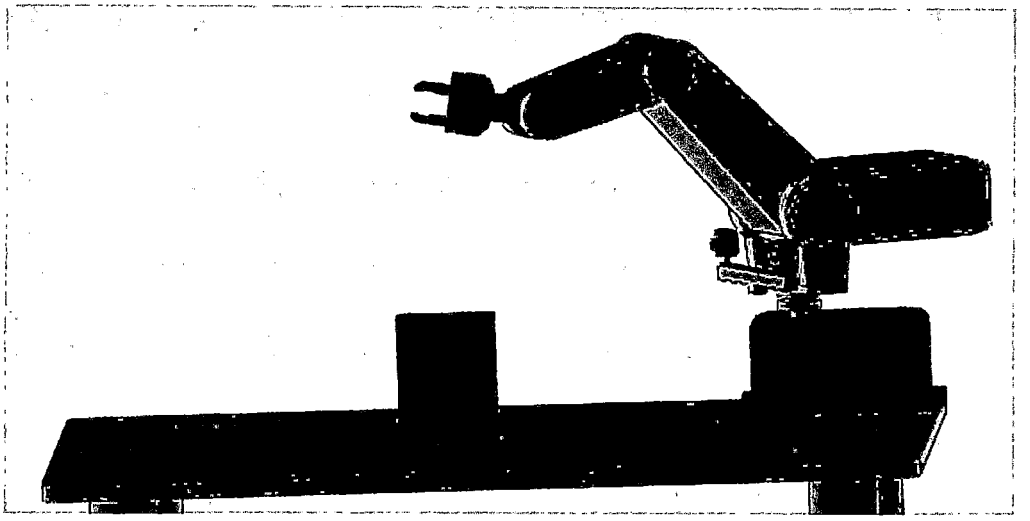


Figura 4-6. Segunda posición, el par estéreo puede rotar.

En la última posición (ver figura 4-7) el par estéreo se ubica sobre el eslabón conocido como antebrazo (forearm) del brazo robótico, teniendo más posibilidades de desplazamiento y logrando un campo de visión que varía de acuerdo a los movimientos del brazo robótico. En esta configuración se tiene que realizar dos traslaciones y dos rotaciones de coordenadas y esta será usada como ejemplo debido a que es la más completa.

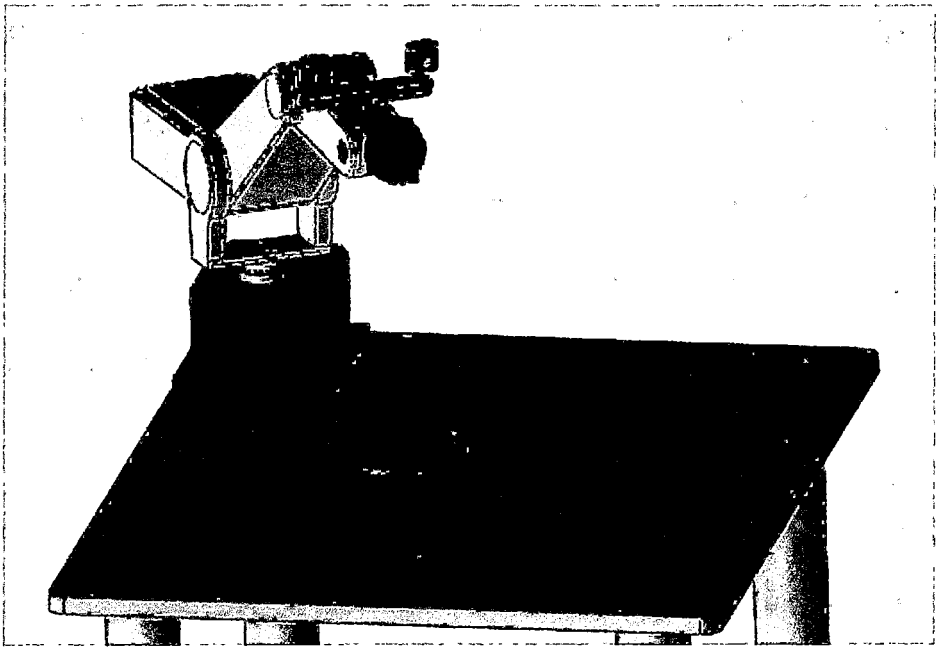


Figura 4-7. El par estéreo ubicado en un eslabón del brazo robótico.

4.1.4. Sistemas de coordenadas tridimensionales

Los algoritmos de inferencia del punto de agarre, correlación y triangulación conforman el sistema de visión que nos va a entregar las coordenadas tridimensionales del objeto que se desea capturar. Debemos aclarar primero que el sistema siempre calculará las coordenadas con respecto al sistema de coordenadas de calibración, después se puede realizar transformaciones de estas para que esta información sea útil para un brazo robótico.

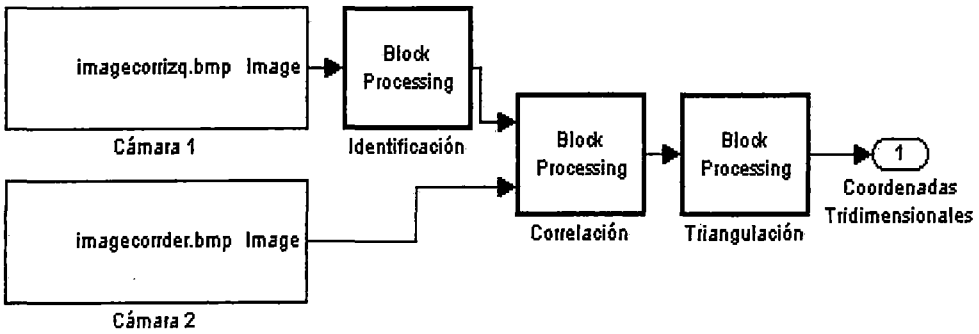


Figura 4-8. Diagrama de bloques del sistema de visión.

En la figura 4-8 se muestra el diagrama de bloques del sistema de visión. En este sistema las entradas son dos imágenes del objeto que se desea capturar, las cuales pasan por los procesos de identificación, correlación y triangulación donde finalmente se calculan coordenadas tridimensionales. En la figura 4-9 nos muestra el sistema de coordenadas usado en la calibración (X_1, Y_1, Z_1), el sistema de coordenadas del eslabón donde se ubica el par estéreo y el sistema de coordenadas de robot.

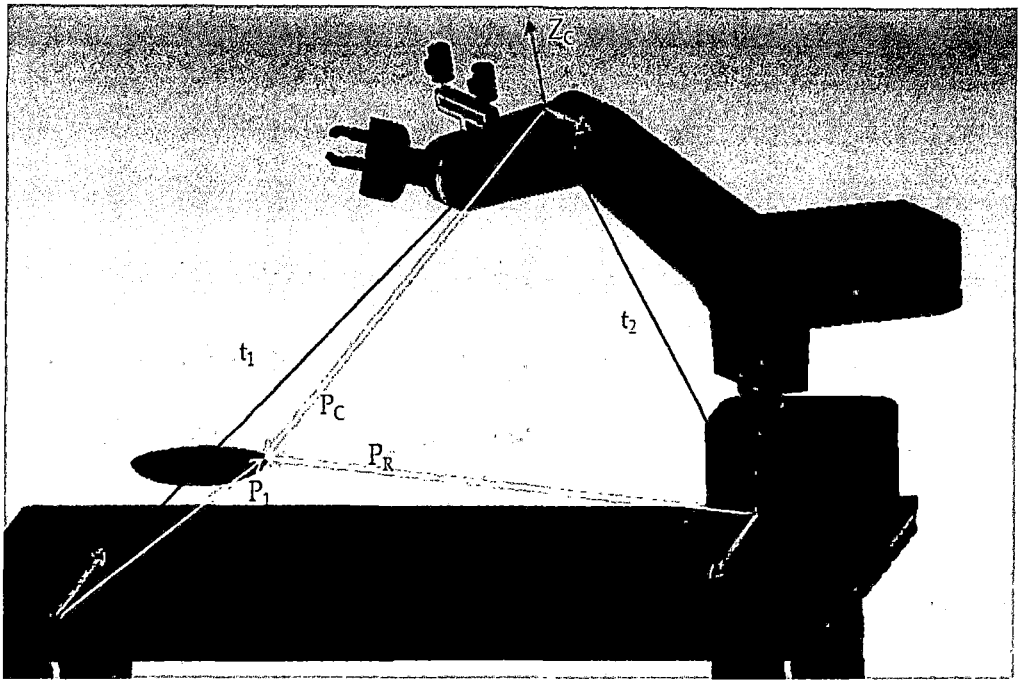


Figura 4-9. Los tres sistemas de coordenadas que se utilizan.

Como se mencionó anteriormente, el sistema de visión calcula las coordenadas respecto al sistema de coordenadas que se usó en la calibración (X_1, Y_1, Z_1). Luego, es necesario hacer la transformación de coordenadas con respecto a un punto fijo del eslabón (X_C, Y_C, Z_C) donde se ubica el par estéreo usando las matrices R_1 y t_1. Estas matrices de rotación (R_1) y traslación (t_1) son halladas en el proceso de calibración cuando el par estéreo está montado sobre el eslabón, y además, permanecerán constantes.

Luego de obtener las coordenadas respecto al eslabón donde se ubica el par estéreo, podemos hacer la transformación de estas con respecto al sistema de coordenadas del brazo robótico (X_R, Y_R, Z_R) conociendo R_2 y t_2. Estas matrices (R_2 y t_2) se conocen ya que al controlar el brazo robótico se sabe todo el desplazamiento y rotación de cada uno de los eslabones.

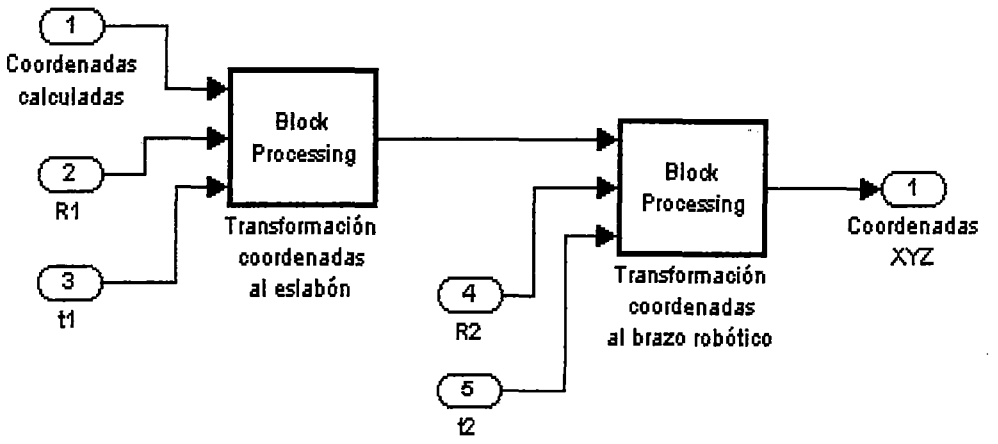


Figura 4-10. Diagrama de bloques de la transformación de coordenadas.

En la figura 4-10 se muestra el diagrama de bloques para calcular las coordenadas con respecto al sistema de coordenadas del brazo robótico. Cada bloque aplica la ecuación: ${}^{B}P = {}^{B}_{A}R \cdot {}^{A}P + {}^{B}_{A}t$. Después que se obtienen las coordenadas tridimensionales con respecto al sistema de coordenadas del robot podemos realizar un planeamiento de trayectoria que controle al brazo robótico para situar al efector final en ese punto y capturar el objeto (ver figura 4-11).

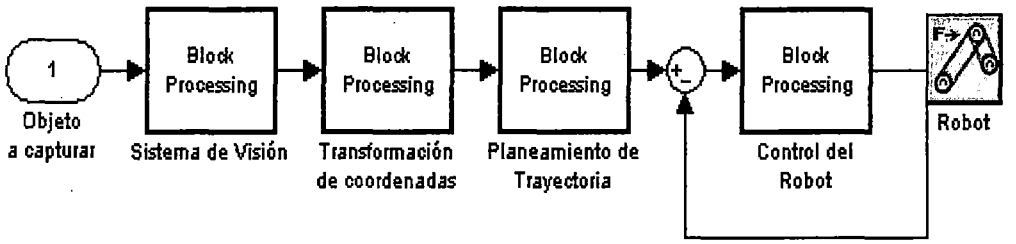


Figura 4-11. Diagrama de bloques del sistema de visión y el control de robot.

4.2. Entrenamiento y Pruebas con Regresión Logística

Este algoritmo de aprendizaje supervisado fue entrenado para tres clases de objetos con cuatro formas diferentes de tomar los datos de las regiones vecinas. Se formó cuatro sets de entrenamiento para cada clase de objeto de acuerdo a las cuatro formas de tomar las características, y se entrenó el algoritmo de aprendizaje

con el fin de evaluar cuál es la forma que mejores resultados nos da usando el menor número de características de regiones vecinas.

4.2.1. Inferencia de datos de tasas

Para esta clase de objeto se extrajo data de 350 imágenes obteniéndose aproximadamente 20´000 índices entre positivos y negativos. Luego de entrenar el algoritmo, se procedió a evaluar los coeficientes hallados con los índices de las imágenes usados en el entrenamiento (eje x) obteniéndose la gráfica 4-12. La salida (eje y) es el valor obtenido de dicha evaluación o inferencia.

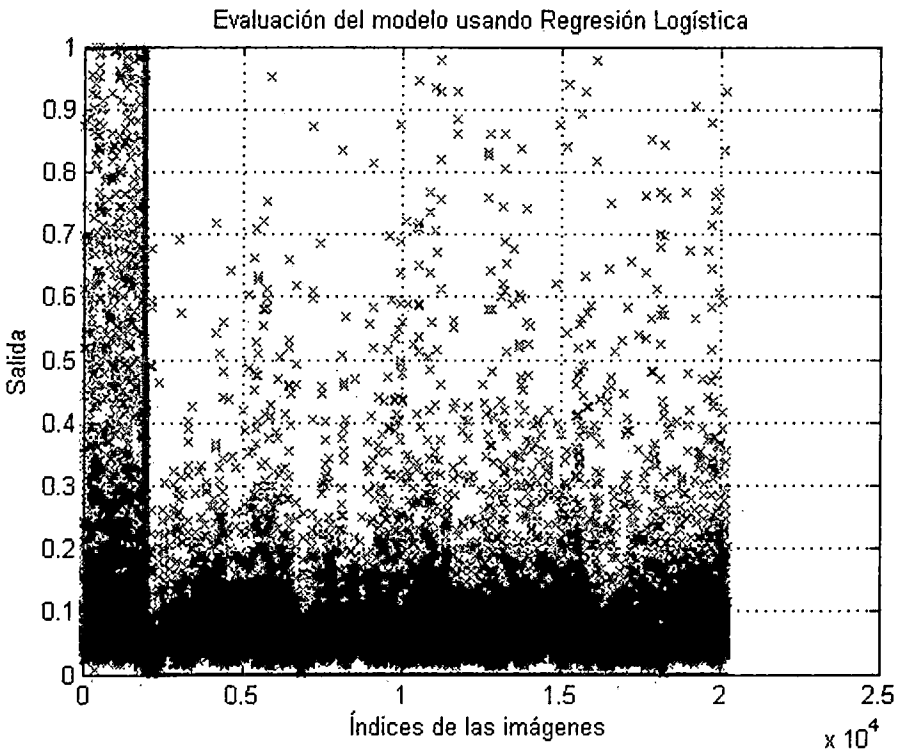


Figura 4-12. Gráfico de la inferencia de datos del caso: Tasa.

En la figura 4-12 se puede apreciar que hay varios índices positivos que tienen valores inferidos cercanos a cero, el resto está disperso entre cero y uno. La mayoría de índices negativos tienen valores cercanos a cero aunque algunos alcanzan valores cercanos a uno. Se sabe que la función ***Logit*** entrega valores entre un rango de cero y uno. Si revisamos el etiquetamiento de la data para

este caso (ver figura 4-13) nos damos cuenta que hay índices negativos y positivos que tienen características similares debido a que se en encuentran en la frontera donde termina el etiquetamiento de la data.

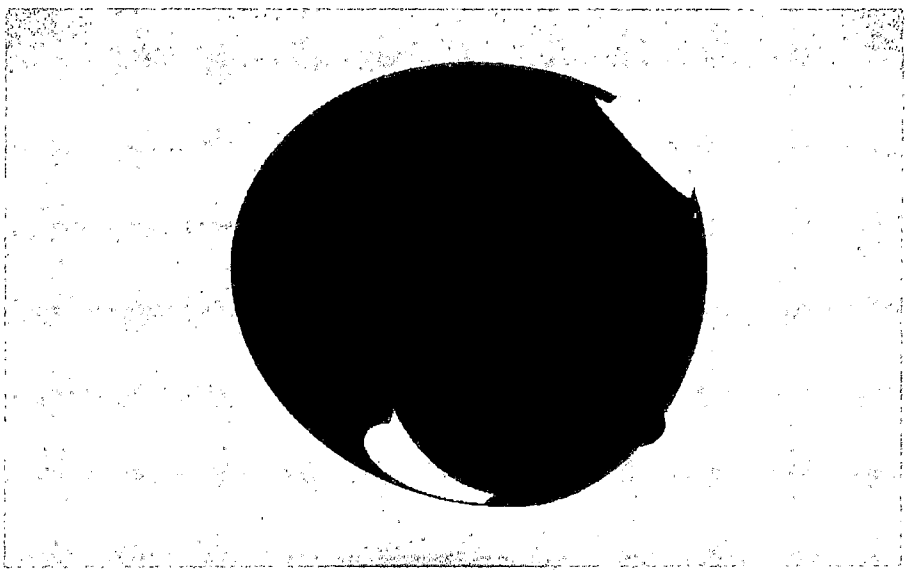


Figura 4-13. Imagen sintética de una tasa etiquetada.

4.2.2. Inferencia de datos de libros

Para esta clase de objeto se trabajó con 250 imágenes y se entrenó también para las cuatro formas de vecindad. En la figura 4-14 se grafica la inferencia de datos usando la tercera forma de tomar características de regiones vecinas.

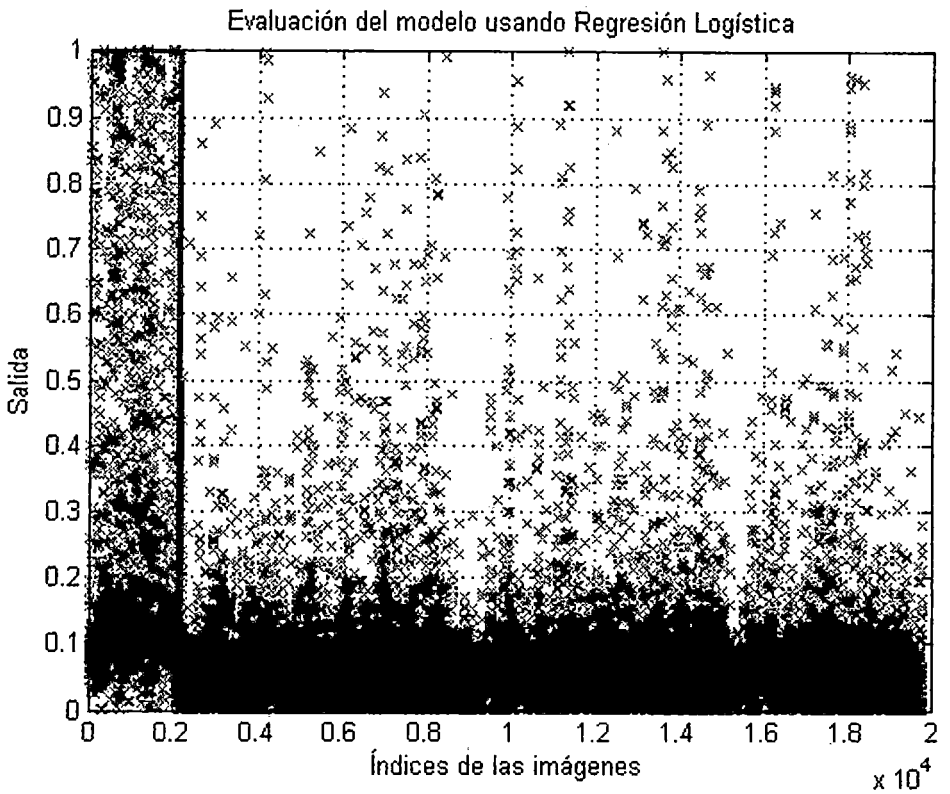


Figura 4-14. Gráfico de la inferencia de datos del caso: Libro.

En la figura 4-14 se puede apreciar que hay menos índices positivos que tienen valores cercanos a cero y ahora los valores de inferencia están más cercanos a uno. En este caso como en el anterior, la mayoría de los índices negativos tienen valores cercanos a cero. Finalmente, el porcentaje entre índices positivos y negativos es casi el mismo para el caso libro y el caso tasa.

4.2.3. Inferencia de datos de platos

En este caso se trabajó con 250 imágenes de platos y se volvió a entrenar usando las cuatro formas de tomar la vecindad. En la figura 4-15 se muestra la inferencia de datos usando la tercera forma; también se puede apreciar que en este caso el porcentaje de índices positivos es mayor que en los casos anteriores. Por otra parte, en el etiquetamiento de los platos se diferencian claramente dos zonas: el borde y la parte restante del plato (ver figura 4-16).

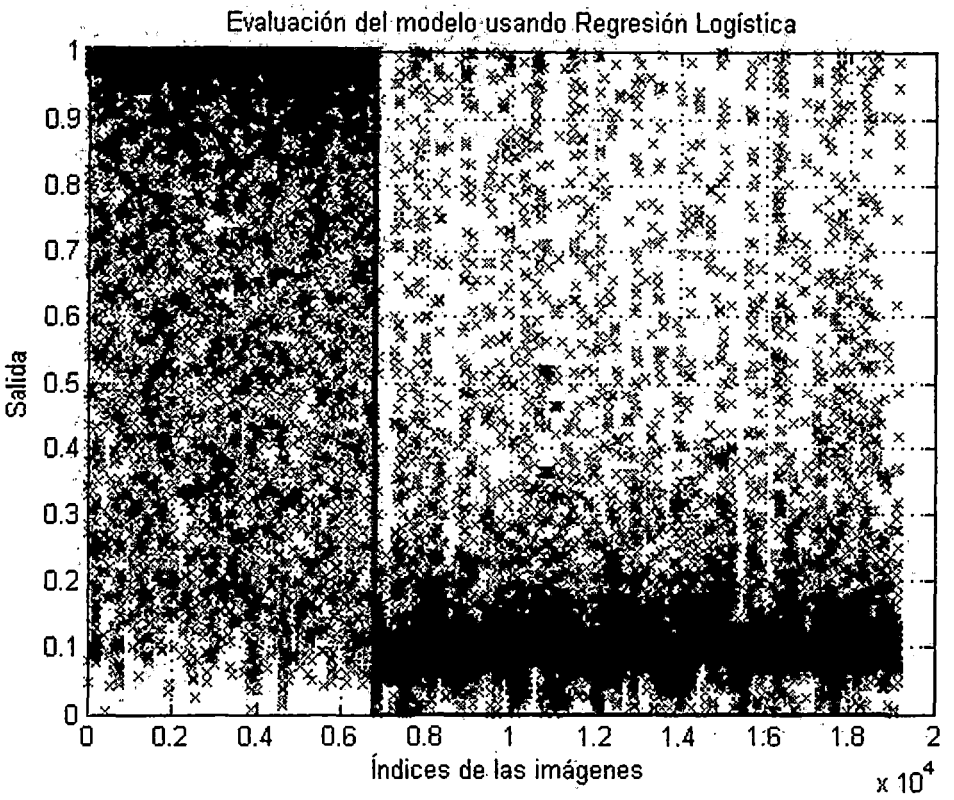


Figura 4-15. Gráfico de la inferencia de datos del caso: Plato.

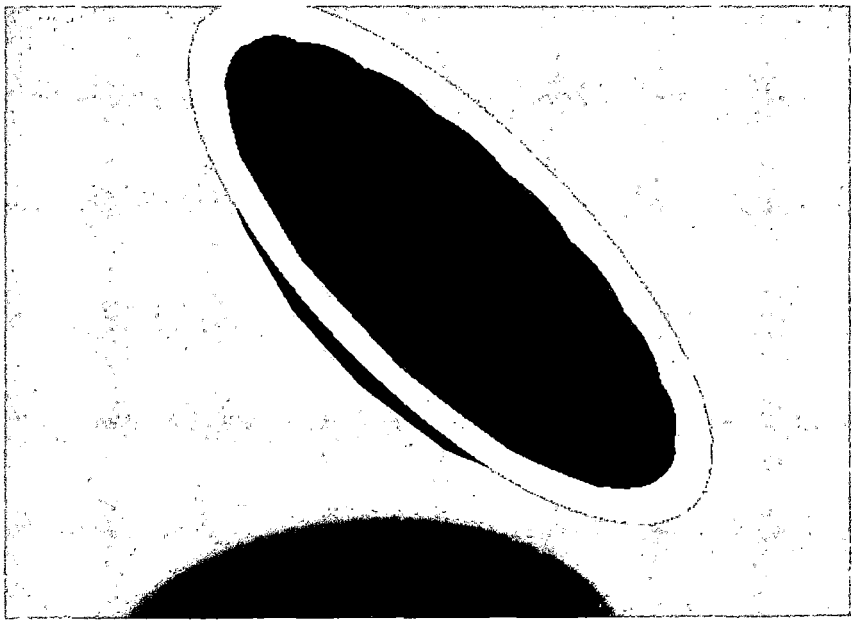


Figura 4-16. Imagen sintética de un plato etiquetado.

En este caso se aprecia que el algoritmo ha logrado clasificar mejor los datos de acuerdo a la salida deseada. Esto se debe al etiquetamiento de la data (ver figura 4-16) y a la forma del objeto. En el caso plato es más fácil obtener un aprendizaje y poder clasificar en índices positivos y negativos debido a las dos zonas mencionadas anteriormente. Si bien hay índices positivos con valores cercanos a cero no son la mayoría.

Los gráficos y los resultados de la inferencia de imágenes sintéticas dependen del objeto, de la forma de etiquetado, de las características tomadas de la imagen del objeto (filtros de textura y bordes) y también del algoritmo de aprendizaje con el que se está trabajando. Cuando se trabaje con Redes Multicapa Perceptrón se notará una mejora en las gráficas obtenidas y en la inferencia de imágenes sintéticas. Lo que se interpreta como que las redes neuronales Backpropagation encuentran mejores pesos o parámetros que logran clasificar un poco mejor los datos.

4.2.4. Comparación de formas de vecindad

Después de entrenar los algoritmos de aprendizaje compararemos los resultados probando los coeficientes obtenidos con 100 imágenes sintéticas de cada clase de objeto para verificar la correcta inferencia de puntos de agarre en cada imagen y poder escoger la forma que con menor cantidad de características nos permita una buena clasificación.

Debido a que la Regresión Logística es usada para predecir la probabilidad de ocurrencia de un evento, en nuestro caso la presencia de un punto de agarre, en la inferencia de imágenes se considerará como uno o índice positivo a los valores mayores al 90% del máximo valor inferido y como negativos en el caso contrario. De esta forma se eligen los más probables, los mayores valores inferidos. Además, se sabe que por cada objeto hay varios índices positivos y por lo menos, se requiere identificar uno parar realizar la triangulación.

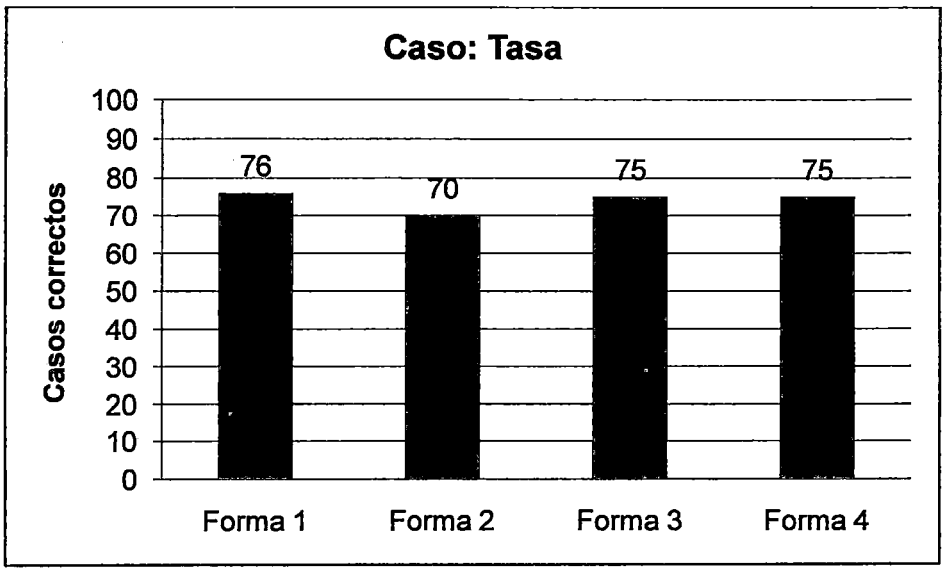


Figura 4-17. Resultados obtenidos con imágenes sintéticas de tasas.

En la figura 4-17 se muestran los resultados de la inferencia imágenes sintéticas de tasas para las cuatro formas de tomar la vecindad. Se puede apreciar que las formas 1, 3 y 4 tienen resultados cercanos. En la figura 4-18 se exponen los resultados para el caso libro. Los resultados de las formas 2, 3 y 4 son muy cercanos.

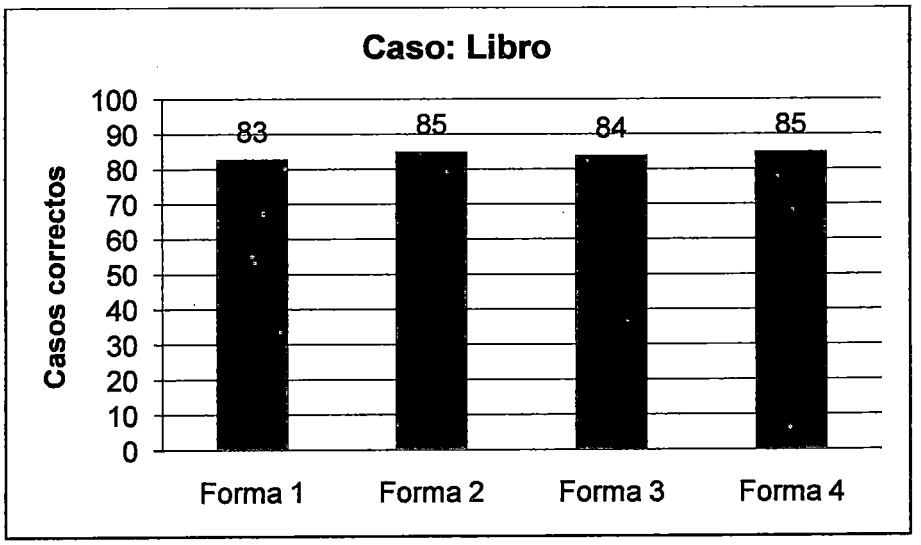


Figura 4-18. Resultados obtenidos con imágenes sintéticas de libros.

En la figura 4-19 se presentan los resultados de la inferencia de 100 imágenes sintéticas de platos. Como se puede apreciar, la tercera forma tiene el mayor número de imágenes correctamente inferidas.

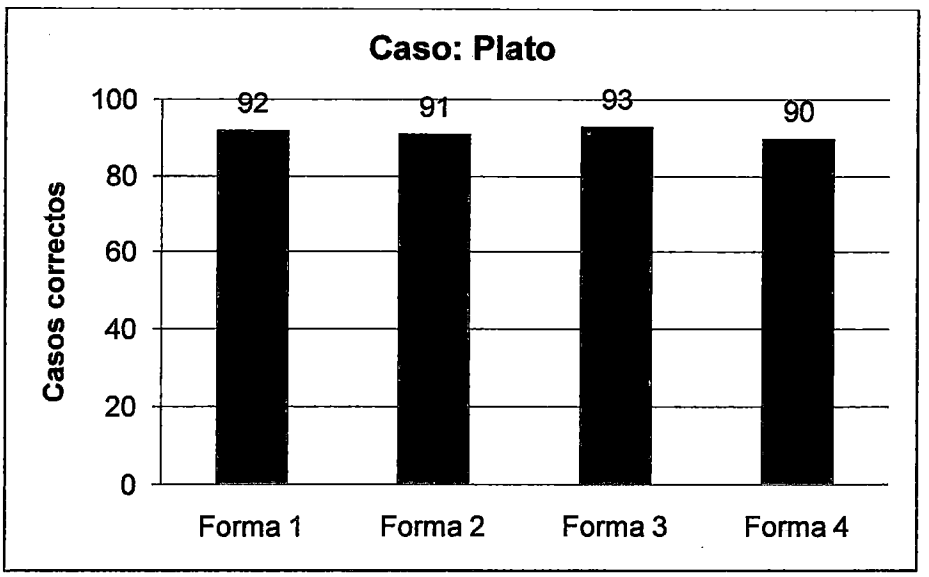


Figura 4-19. Resultados obtenidos con imágenes sintéticas de platos.

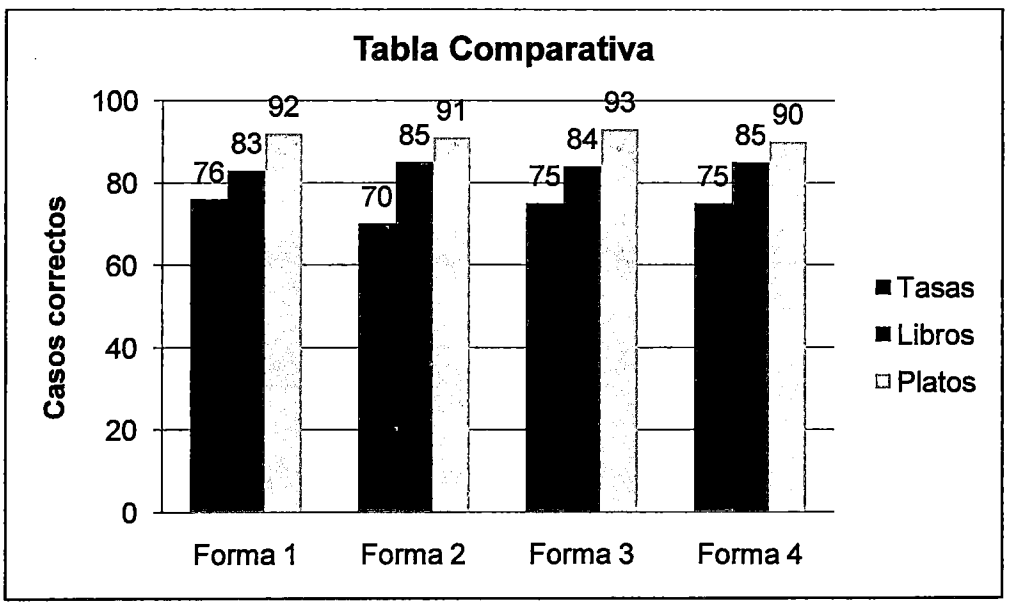


Figura 4-20. Tabla resumen para los tres objetos.

De la figura 4-20 se puede apreciar que en los tres casos con las 4 formas se obtienen resultados cercanos pero si queremos ser más exactos y si

sumamos los resultados obtenidos, la tercera forma tiene mayor cantidad de imágenes correctamente inferidas. Por lo tanto, si deseamos usar el menor número características tendríamos la posibilidad de usar la tercera como primera opción y cuarta forma como segunda opción. La tercera y la cuarta forma emplean 225 y 156 características en total respectivamente.

4.3. Entrenamiento y Pruebas con Redes Perceptrón Multicapa

Después de determinar con cual forma vecindad se puede trabajar se procederá a realizar el entrenamiento usando redes Perceptrón Multicapa (MLP). El entrenamiento de las redes MLP debido a que es un proceso iterativo de actualización de pesos, y que también depende de la topología usada y algoritmo de entrenamiento, podría llegar a ocupar mucha memoria sobre todo si la cantidad de patrones a clasificar y los sets de entrenamiento son muy grandes. Probablemente en estos casos el programa MATLAB no llegue a concluir los cálculos y presente un mensaje "OUT OF MEMORY" debido a que se ha excedido la capacidad de memoria asignada para estos cálculos.

Por este motivo, en el entrenamiento de las redes MLP solo se experimentó con las dos últimas formas de vecindad (tercera y cuarta) cuyos resultados ya habían sido comprobados con el primer algoritmo de aprendizaje (Regresión Logística). Estas son las que emplean la cantidad suficiente de características o patrones a clasificar, por lo tanto, emplean menor cantidad de memoria para el entrenamiento. De esta forma podemos experimentar diferentes topologías sin exceder el límite de memoria que es lo que sucede cuando se trabaja con la primera o segunda forma de vecindad.

Al momento de entrenar una red Perceptrón Multicapa hay varios parámetros de diseño que debemos definir, pues estos parámetros van a influenciar en el proceso de obtención del modelo y en los resultados que se puedan obtener de la aplicación de este algoritmo. Primeramente debemos tener claro que estamos frente a un problema de clasificación por lo tanto debemos seleccionar los parámetros para que nuestra red logre aprender de un set de entrenamiento y pueda clasificar correctamente nuevos datos.

Las primeras preguntas a responder son: ¿Cuántas capas ocultas y neuronas por capa se necesitan? ¿Qué funciones de activación se deben usar? Debido a que no existen reglas específicas ya que depende también del tipo de problema con el que se trata, se empezó a probar diferentes tipos de topologías con diferentes funciones de activación revisando constantemente el error alcanzado y las gráficas obtenidas con los datos de entrada. De esta manera se encontró algunas topologías que lograban clasificar mejor los datos de entrenamiento.

Después de las primeras pruebas con las redes MLP se llegó a la conclusión que para este problema de clasificación se tiene que usar funciones ***tansig*** en la capa oculta y funciones ***tansig*** o ***purelin*** en la capa de salida.

También se tiene que elegir un algoritmo de entrenamiento para determinar los pesos de nuestra red. El siguiente paso fue experimentar varios algoritmos de entrenamiento con nuestro problema de clasificación evaluando la rapidez de convergencia y MSE alcanzado.

4.3.1. Algoritmos de entrenamiento

Dentro del algoritmo ***Backpropagation*** hay muchas variaciones pero lo que tienen todas en común es que usan la gradiente de la función de performance para ajustar los pesos y bias. En el ***Toolbox*** de Matlab hay varios de estos algoritmos de entrenamiento que se pueden usar para el entrenamiento de una red MLP. Los que se probaron son:

- Traincgf: Conjugate gradient backpropagation with Fletcher - Reeves updates algorithm.
- Traincgp: Conjugate gradient backpropagation with Polak - Ribiére updates algorithm.
- Traincgb: Conjugate gradient backpropagation with Powell - Beale updates algorithm.
- Traingda: Gradient descent with adaptive learning rate backpropagation algorithm.
- Traingdx: Gradient descent with momentum and adaptive learning rate backpropagation.

- Trainscg: Scaled conjugated gradient backpropagation algorithm.
- Trainrp: Resilient backpropagation algorithm.
- Trainlm: Levenberg Marquart backpropagation algorithm.

Para experimentar con estos algoritmos de entrenamiento se usó dos topologías. La primera red está compuesta por 10 neuronas en la capa oculta y 1 neurona en la capa de salida. Las funciones de activación que usaron fueron ***tansig*** en la capa oculta y ***tansig*** en la capa de salida. La segunda red neuronal está compuesta por 10 neuronas en la capa oculta y 1 neurona en la capa de salida pero con funciones ***tansig*** en la capa oculta y ***purelin*** en la salida. Con estas dos redes se experimentó los diferentes algoritmos de entrenamiento usando los datos extraídos de imágenes sintéticas de libros.

En la figura 4-21 se muestra la performance de siete algoritmos de entrenamiento para la segunda red y el mismo set entrenamiento. Cada algoritmo de entrenamiento tiene una evolución diferente de la función de performance y también diferente forma de convergencia. Todos estos algoritmos de entrenamiento trabajan con MSE (Mean Square Error).

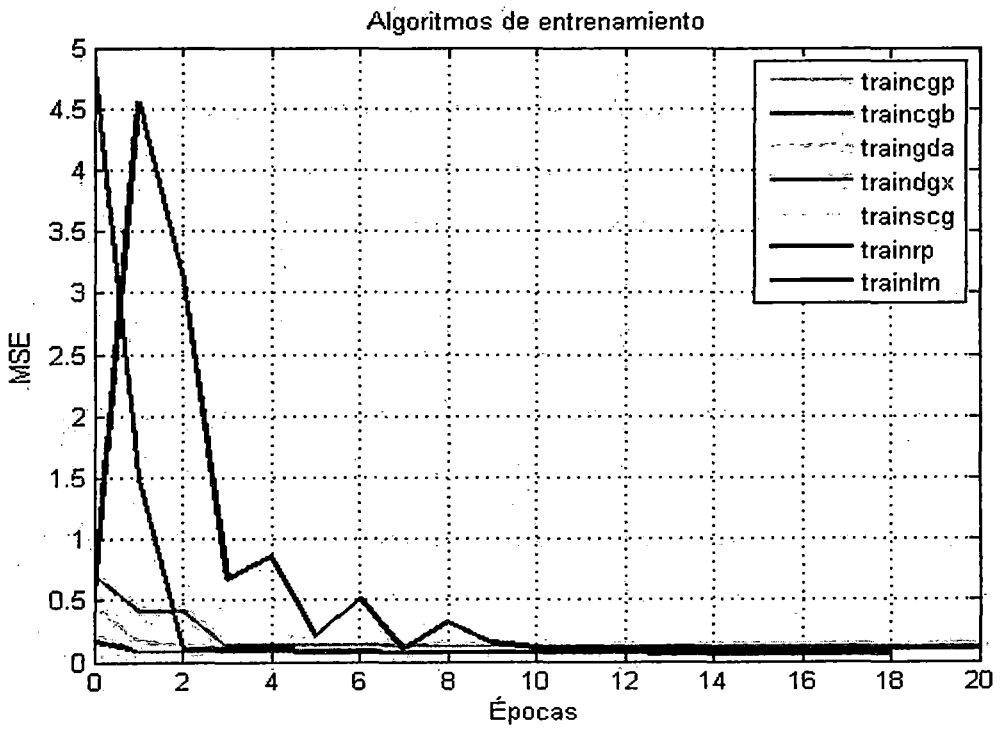


Figura 4-21. Gráfico de la performance de los siete algoritmos de entrenamiento.

En la figura 4-21 se puede apreciar que el algoritmo que alcanza un error (MSE) más bajo es el ***trainlm***, los algoritmos ***traincgp***, ***traingc*** y ***trainscg*** convergen rápidamente y también alcanzan un error cercano al error mínimo alcanzado. El algoritmo ***trainrp*** demora un poco en converger y los algoritmos ***traingda*** y ***traingdx*** convergen regularmente rápido; sin embargo, los errores alcanzados son los mayores de todos. Para las siguientes pruebas se trabajará con el algoritmo ***trainlm***.

4.3.2. Criterios de parada

Para considerar concluida la fase entrenamiento pueden considerarse dos criterios: alcanzar un error mínimo o alcanzar un número determinado de iteraciones. Pero en el ***Toolbox*** hay otro criterio por el cual se podría dar por concluida la fase de entrenamiento. Este criterio se llama ***Early Stopping*** y trata de evitar que el entrenamiento de una red neuronal caiga en un ***Overfitting***, que es lo que sucede cuando la red pierde su capacidad de generalización, usualmente cuando alcanza un error muy pequeño.

Para lograr esto se divide el set de entrenamiento en tres partes: 60%, 20% y 20%. Por defecto solo el 60% de los datos ingresados es usado en el entrenamiento (datos de entrenamiento), el 20% es usado para evaluar los pesos actualizados calculando el MSE constantemente. Esta división es aleatoria y también puede ser modificada. Si la diferencia entre el error de validación y error de entrenamiento se incrementa en seis iteraciones seguidas querrá decir que la red puede estar frente a un problema de ***Overfitting,*** entonces se da por concluido el entrenamiento y se presentan como resultados los pesos obtenidos cuando se produjo el primer incremento. En la mayoría de entrenamientos realizados fue el motivo de parada que más se presentó.

4.3.3. Topologías usadas y resultados obtenidos

Después de elegir el algoritmo de entrenamiento ***trainlm*** y las funciones de activación ***tansig*** y ***purelin*** se procedió a variar el número de neuronas por capa y el número de capa ocultas. Se sabe que para resolver un problema complejo

es suficiente usar entre dos a tres capas ocultas como máximo y el número de neuronas por capa depende del problema que se resuelve o modela. La tabla 4-2 muestra las topologías usadas para el entrenamiento. Se utilizaron funciones ***tansing*** en las capas ocultas y ***purelin*** o ***tansig*** en la capa de salida.

Estructura de la red		
Nº de neuronas en la capa oculta		Nº de neuronas en la capa de salida
4	8	1
5	10	1
10	15	1
5		1
8		1
10		1

Tabla 4-2. Topologías usadas.

Después del entrenamiento se evaluaron los pesos obtenidos con 100 imágenes sintéticas y se escogieron las topologías que lograron identificar más imágenes sintéticas correctamente (ver tabla 4-3 y tabla 4-4).

Estructura de la red			
Caso		Capa oculta	Capa de salida
Tasa	Función	tansig	tansig
	Nº neuronas	8	1
		10	1
Libro	Función	tansig	purelin
	Nº neuronas	10 15	1
		5	1
Plato	Función	tansig	purelin
	Nº neuronas	10 15	1
		8	1

Tabla 4-3. Topologías escogidas para la tercera forma.

Estructura de la red			
Caso		Capa oculta	Capa de salida
Tasa	Función	tansig	tansig
	Nº neuronas	10	1
Libro	Función	tansig	tansig
	Nº neuronas	5	1
	Función	tansig	purelin
	Nº neuronas	10	1
Plato	Función	tansig	tansig
	Nº neuronas	5	1
	Función	tansig	purelin
	Nº neuronas	10	1

Tabla 4-4. Topologías escogidas para la cuarta forma.

En la figura 4-22 se muestra la inferencia de datos de entrenamiento de las tasas usando la tercera forma de vecindad y una red MLP.

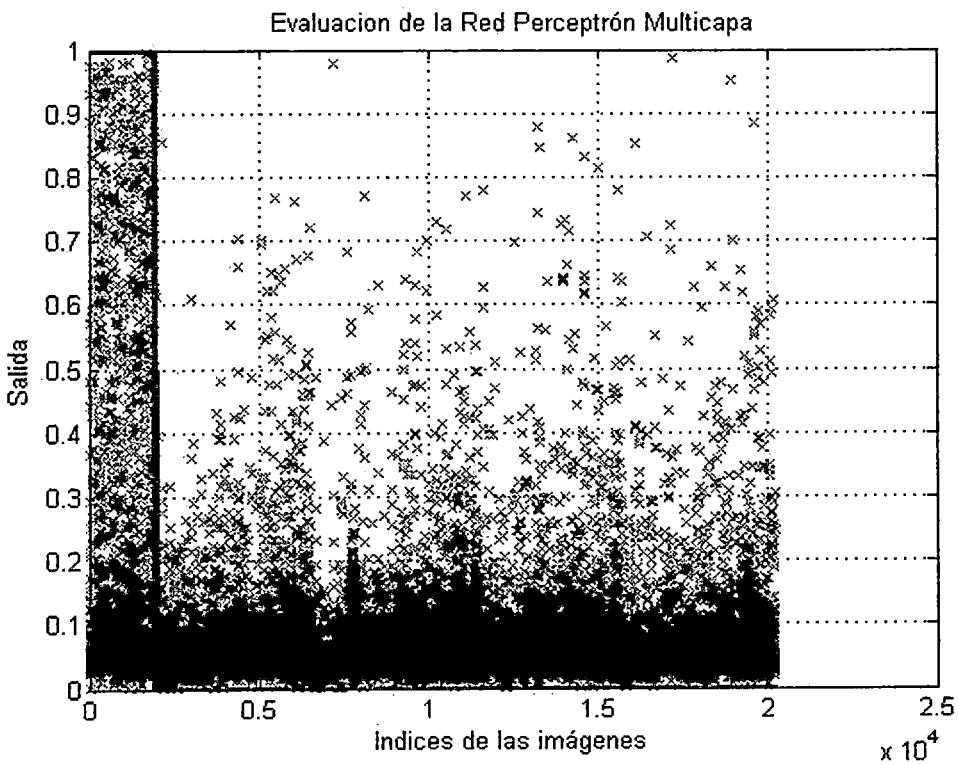


Figura 4-22. Gráfico de la inferencia de datos de tasas usando una MLP.

Si la comparamos la gráfica obtenida del entrenamiento con Regresión Logística (figura 4-12) con la gráfica obtenida del entrenamiento con redes Perceptrón Multicapa (figura 4-22) podemos apreciar que ahora los valores alcanzados por los índices positivos están más cercanos a 1. Además, el MSE alcanzado por las redes Perceptrón Multicapa es menor que el alcanzado por los modelos de Regresión Logística entrenados. En el entrenamiento del caso del libro y del plato obtuvimos las siguientes gráficas (ver figura 4-23 y figura 4-24 respectivamente).

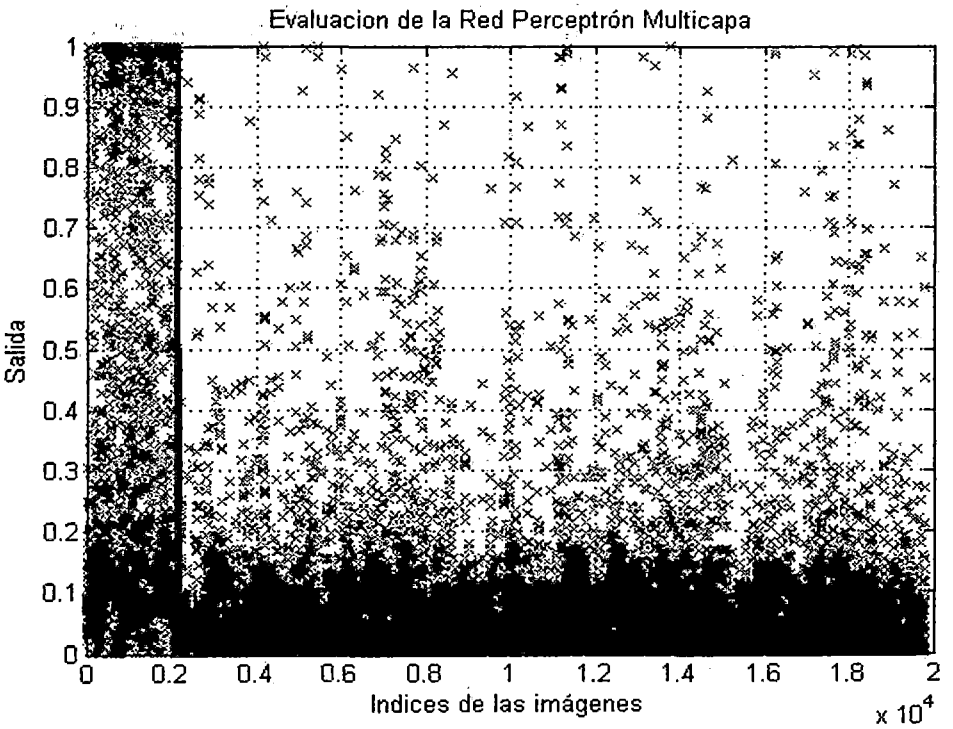


Figura 4-23. Gráfico de la inferencia de datos de libros usando una MLP.

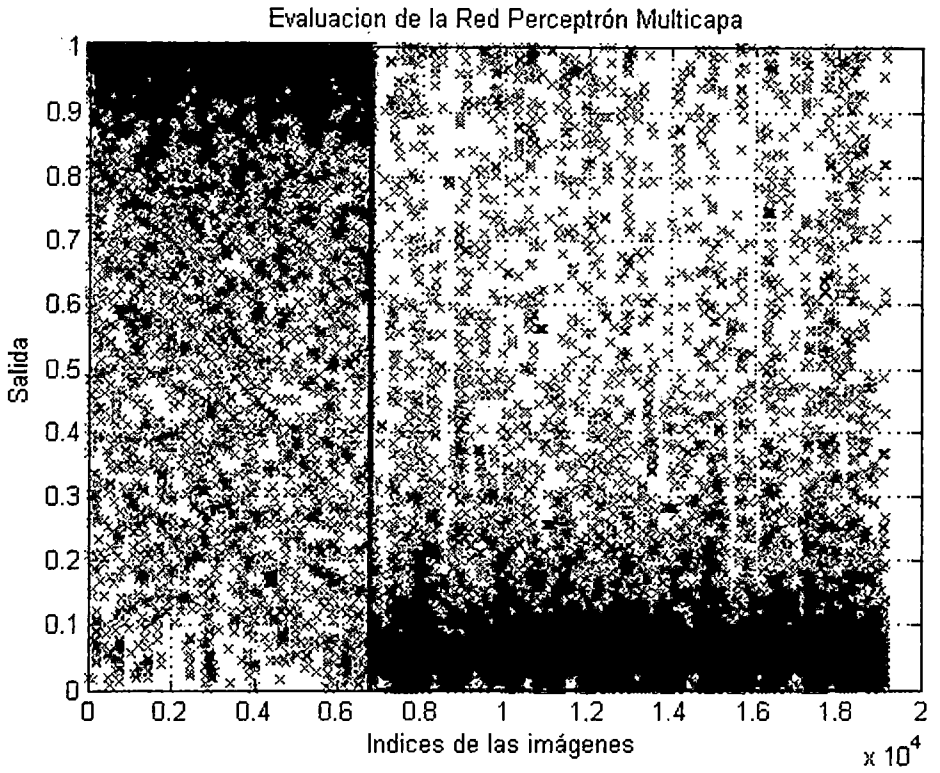


Figura 4-24. Gráfico de la inferencia de datos de platos usando una MLP.

Para estos dos últimos casos (ver figura 4-23 y figura 4-24) también se puede ver gráficamente que las redes Perceptrón Multicapa tienen una mejor clasificación de los datos de entrenamiento a pesar de que las redes Perceptrón Multicapa solo emplean el 60% de ellos. Esto también se puede verificar si revisáramos los MSE alcanzados por la RL (ver tabla 4-5) y las redes MLP (ver tabla 4-6). De ambas tablas se puede observar que los MSE hallados con los algoritmos de Regresión Logística siempre son mayores que los MSE hallados con las redes MLP.

Regresión Logística				
Objeto	Forma 1	Forma 2	Forma 3	Forma 4
	MSE	MSE	MSE	MSE
Tasas	0.0733	0.0733	0.0742	0.0745
Libros	0.0744	0.0751	0.0756	0.0763
Platos	0.1074	0.1082	0.1093	0.1114

Tabla 4-5. MSE alcanzados por los algoritmos RL.

Redes Perceptrón Multicapa			
Objeto	Topología		MSE
	Capa oculta	Capa de salida	
Tasas	tansig	tansig	0.0695
	8	1	
	tansig	tansig	0.0688
	10	1	
Libros	tansig	purelin	0.0675
	8 1	1	
	tansig	purelin	0.0732
	5 10	1	
Platos	tansig	purelin	0.0814
	10	1	
	tansig	purelin	0.0851
	5	1	

Tabla 4-6. MSE alcanzados por las redes Perceptrón Multicapa.

En la figura 4-25 se muestran los resultados de la inferencia de imágenes sintéticas de objetos usando redes Perceptrón Multicapa entrenadas. En estos entrenamientos se usó la tercera forma y en los resultados de la figura 4-26 se usó la cuarta forma.

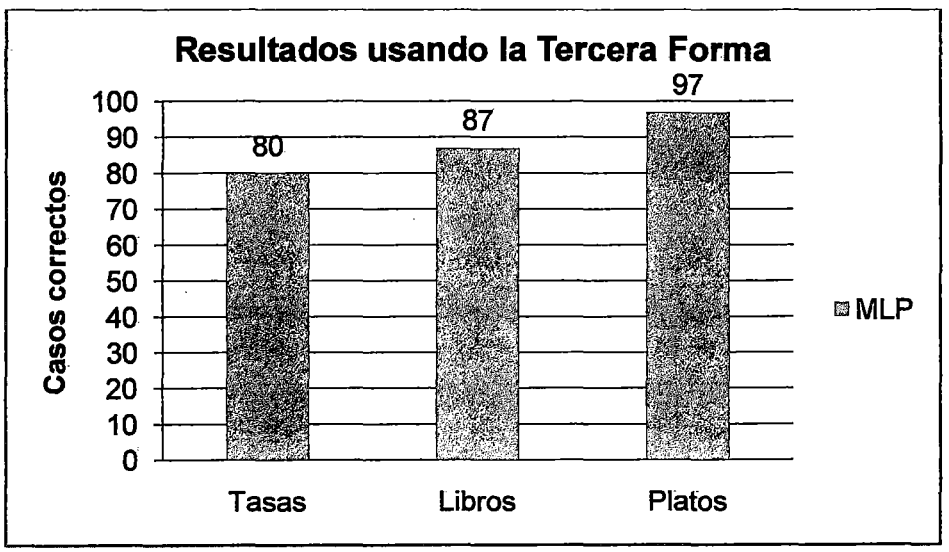


Figura 4-25. Resultados obtenidos con imágenes sintéticas usando la tercera forma.

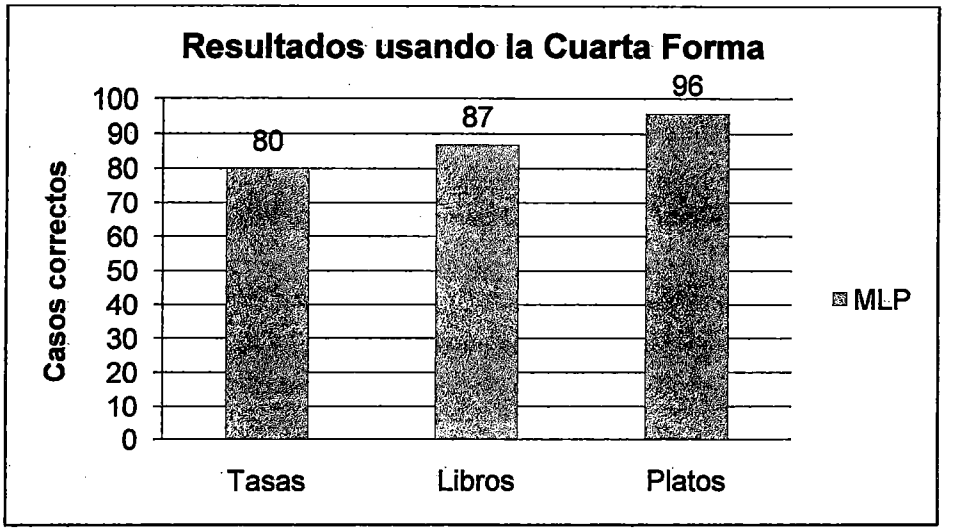


Figura 4-26. Resultados obtenidos con imágenes sintéticas usando la cuarta forma.

Si comparamos los resultados de la inferencia de imágenes sintéticas para la tercera y cuarta forma, las redes Perceptrón Multicapa alcanzan mayor cantidad de imágenes inferidas correctamente con respecto a las imágenes inferidas por Regresión Logística. El siguiente paso es entrenar usando redes neuronales con función de base radial.

4.4. Entrenamiento y Pruebas con Redes con Función de Base Radial

El tercer algoritmo de aprendizaje que se probó fue las redes neuronales con función de base radial, las cuales también han sido usadas para resolver problemas de clasificación. La topología de una red neuronal con función de base radial solo tiene una capa oculta y capa de salida (para nuestro caso una sola salida). En el entrenamiento de estas redes se deben establecer los parámetros de diseño que serán explicados a continuación.

El primero es el MSE a alcanzar que para nuestro caso es el mínimo error alcanzado por las redes neuronales entrenadas anteriormente. El segundo parámetro se denomina ***spread***, el cual es el mismo para todas las neuronas de la capa oculta y se interpreta como la propagación de una función de base radial. El ***spread*** es un parámetro que se tiene que ir variando hasta encontrar el mejor valor que le permita a la red una buena generalización después del entrenamiento. El

problema que se puede presentar es que la red logré clasificar muy bien los datos de entrenamiento pero que presente un error muy alto cuando se presenten datos nuevos. En la capa de salida la función de activación es una función ***purelin***.

El último parámetro es el número máximo de neuronas a implementarse ya que el algoritmo de entrenamiento se inicia sin ninguna neurona en la capa oculta y va agregando neuronas y actualizando los pesos hasta que se alcance el error deseado (MSE) o se alcance el límite máximo de neuronas establecido.

En este caso no hay criterios que puedan asegurar no caer en un ***Overfitting*** o una buena generalización, ya que la red incrementará el número de neuronas hasta alcanzar el error deseado. Por lo que el único camino que queda es probar con diferentes valores de ***spread*** hasta encontrar una buena generalización evaluando siempre con imágenes no usadas en el entrenamiento.

4.4.1. Caso: Libro

Este fue el primer caso con el que se experimentó el entrenamiento para diferentes valores de ***spread***. Se consideró un número máximo de neuronas alto igual a 500 para que se pudiera llegar al error deseado y el parámetro ***spread*** tomó valores como: 1, 10 mil, 50 mil, etc. Se usó la tercera forma de vecindad para extraer las características y un error (MSE) de 0,05 como objetivo para todos los casos.

En el primer entrenamiento se usó un ***spread*** = 1 y un error = 0,05. La figura 4-27 muestra la inferencia de datos usados en el entrenamiento. A partir de la gráfica se puede apreciar que los valores inferidos son casi exactamente los valores deseados (target).

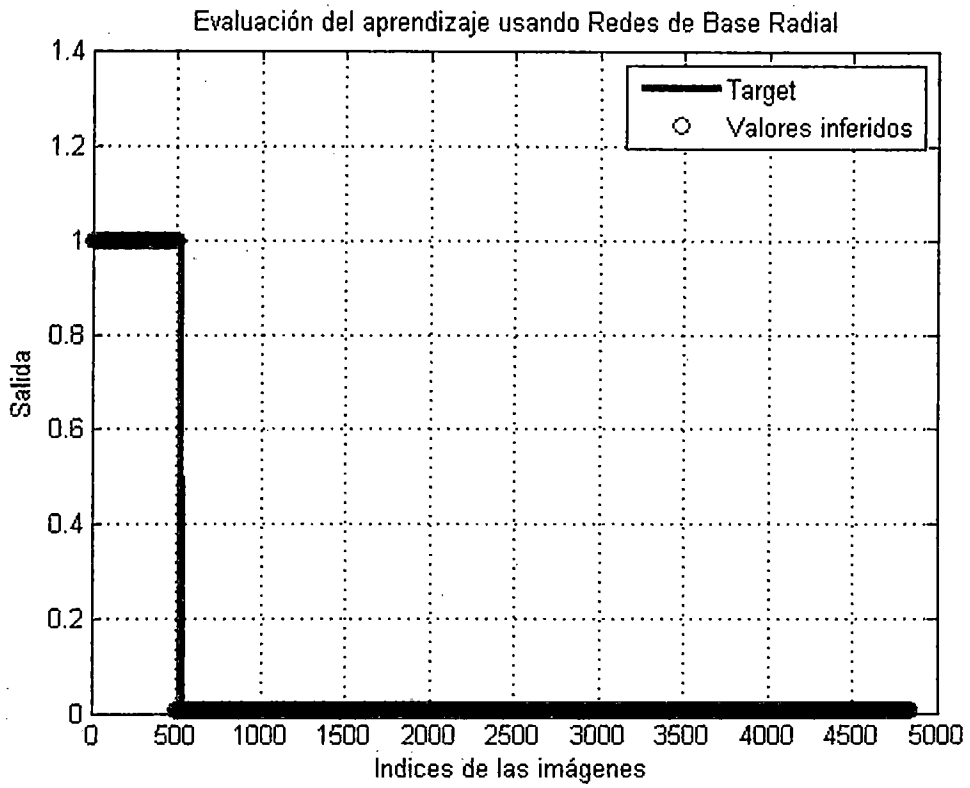


Figura 4-27. Inferencia de datos del primer entrenamiento.

Al probar los pesos obtenidos con los datos de entrenamiento se obtuvo la gráfica de la figura 4-27 y cuando se probó con imágenes sintéticas usadas en el entrenamiento resultó un porcentaje mayor a 97% de imágenes inferidas correctamente; pero cuando se probó la red entrenada con nuevas imágenes sintéticas no usadas en el entrenamiento el porcentaje de imágenes inferidas correctamente era muy bajo. A partir de ello se deduce que la red había aprendido o memorizado los datos de entrenamiento pero no podía inferir correctamente datos nuevos. Esto debido a un bajo valor de ***spread***. Se sabe con anterioridad que el valor de ***spread*** está relacionado con la capacidad de generalización de la red, por esta razón se empezó a aumentar dicho valor y realizar pruebas solo con nuevos datos.

En el segundo entrenamiento se usó un valor de ***spread*** = 10´000, error = 0.05 y el número neuronas alcanzado fue 375. La figura 4-28 muestra la inferencia de datos usados en el entrenamiento. Como se puede apreciar los

datos se empiezan a dispersar con respecto a la salida deseada e incluso algunos índices positivos tienen valores muy cercanos a cero.

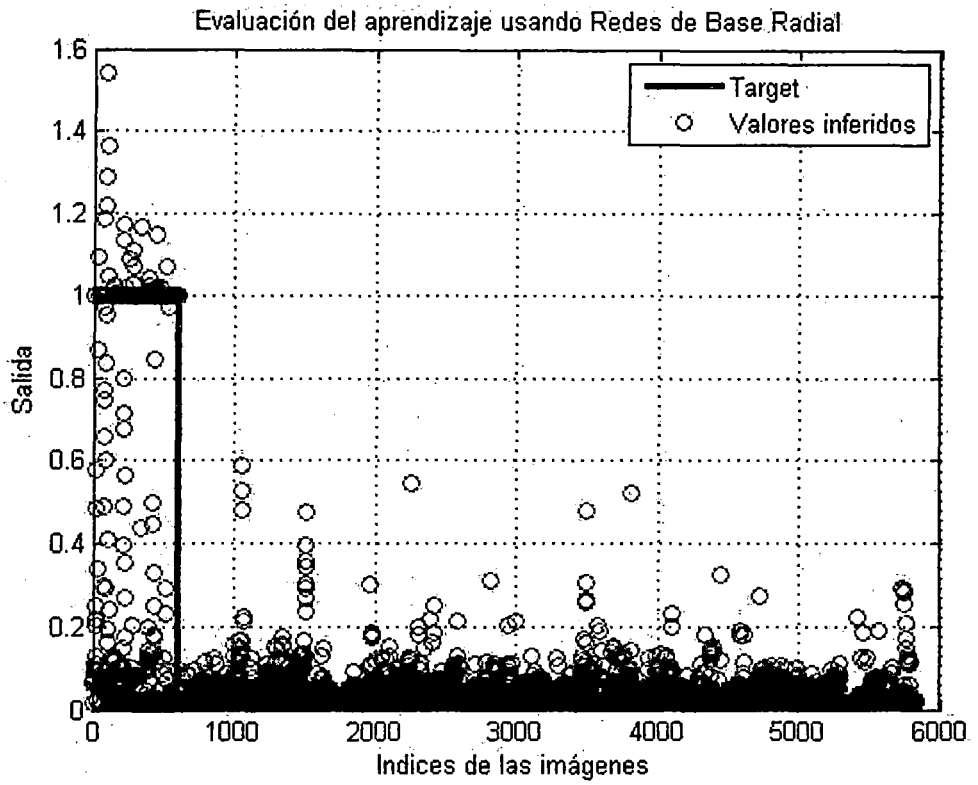


Figura 4-28. Inferencia de datos del segundo entrenamiento.

En el tercer entrenamiento se usó un ***spread*** = 50´000, error = 0.05 y el número neuronas alcanzado fue 375. La figura 4-29 muestra la inferencia de índices usados en el entrenamiento. Los índices negativos empiezan a dispersarse y a tomar valores negativos en algunos casos.

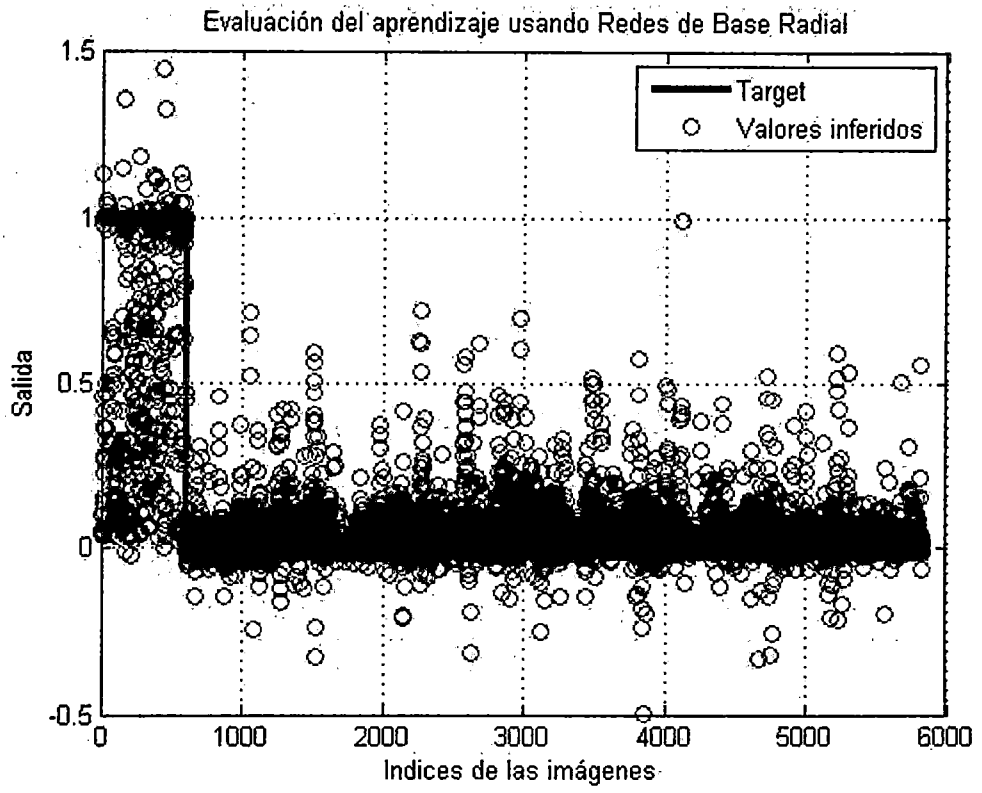


Figura 4-29. Inferencia de datos del tercer entrenamiento.

En el cuarto entrenamiento se usó un valor de ***spread*** = 150´000 y un error = 0.05. La gráfica de la inferencia de datos usados en el entrenamiento se puede apreciar en la figura 4-30. Al incrementar el valor de ***spread***, se incrementa la dispersión de los valores inferidos con respecto a la salida deseada. Para estos valores de ***spread*** las gráficas tienen un mayor parecido a las gráficas obtenidas cuando se realizó en el entrenamiento usando las Redes Multicapa Perceptrón.

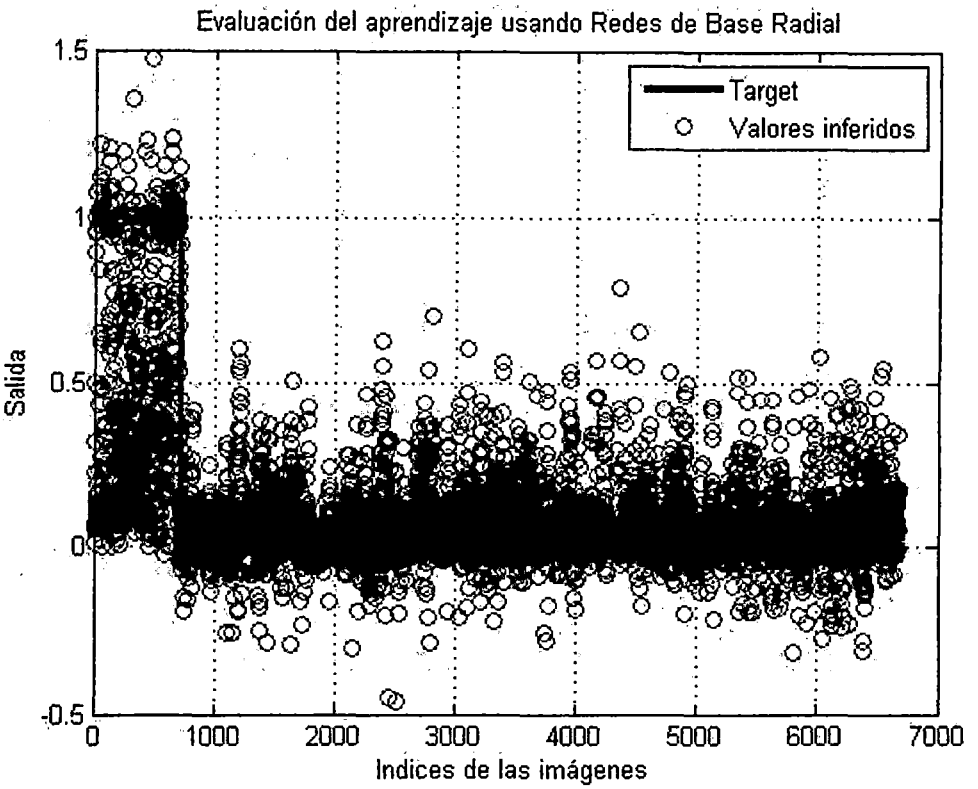


Figura 4-30. Inferencia de datos del cuarto entrenamiento.

Se puede apreciar que las primeras gráficas tienen gran similitud con la salida deseada debido a que tienen bajos valores de ***spread***, pero esto va cambiando cuando incrementamos este valor. Si queremos conocer la generalización de los pesos obtenidos en los entrenamientos se tienen que probar con datos nuevos como lo haremos a continuación.

4.4.2. Evaluación de pesos obtenidos

Después de realizar el entrenamiento para varios valores de spread los pesos obtenidos se guardaron y se evaluaron con data que no fue usada en el entrenamiento para calcular el MSE que se alcanzaba. A continuación, la tabla 4-7 muestra los MSE para diferentes valores de ***spread***.

Caso: Libro	
SPREAD	MSE
10 mil	0,103
50 mil	0,0958
75 mil	0,0929
100 mil	0,0892
150 mil	0,0798
175 mil	0,0773
200 mil	0,0772
300 mil	0,0776
400 mil	0,082
500 mil	0,0888
750 mil	0,1092

Tabla 4-7. MSE alcanzados para los valores de ***spread*** probados.

En la figura 4-31 se gráfica como varía el MSE con datos para diferentes valores de ***spread***. Se puede apreciar que con los valores de ***spread*** de 175´000 y 200´000 se obtienen los menores valores de MSE. En el entrenamiento todas las redes RBF entrenadas alcanzaron un MSE igual a 0,05.

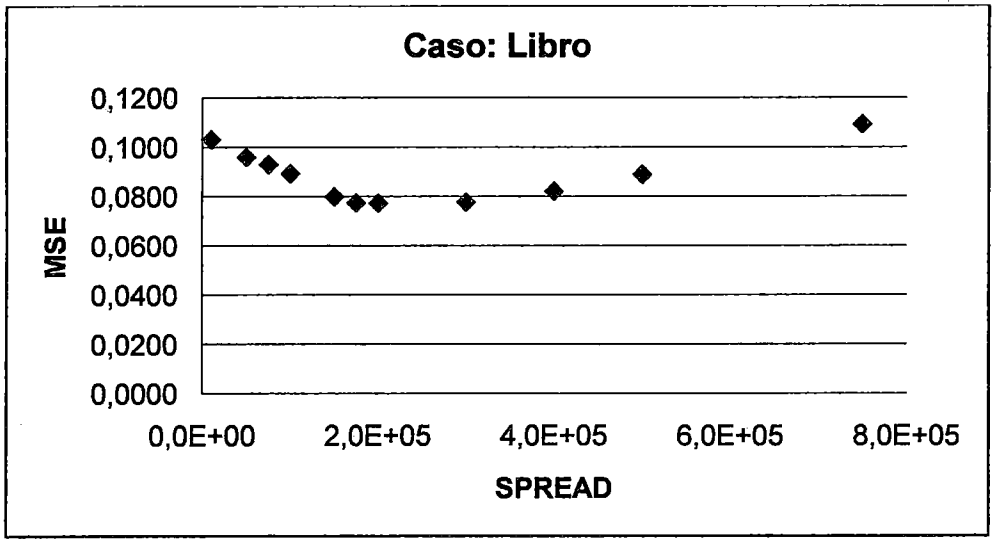


Figura 4-31. Gráfico de MSE obtenido para el caso de los libros.

El procedimiento de entrenamiento para diferentes valores de spread se repitió para el caso de las tasas. Nuevamente se guardaron los pesos y se evaluaron con datos nuevos. En la tabla 4-8 se muestran los valores ***spread*** y los MSE encontrados para este caso. En la figura 4-32 se muestra la gráfica de los valores de MSE alcanzados.

Caso: Tasa	
SPREAD	MSE
150 mil	0,0878
200 mil	0,0873
300 mil	0,0869
400 mil	0,0877
500 mil	0,0896

Tabla 4-8. MSE alcanzados para los valores de ***spread*** probados.

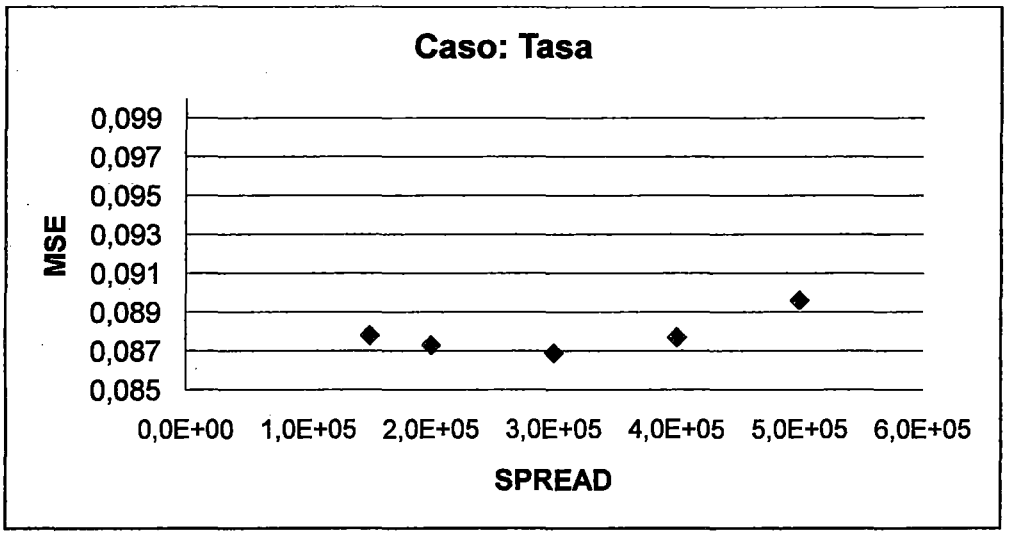


Figura 4-32. Gráfico de MSE obtenido para el caso de las tasas.

Lo siguiente fue probar estos pesos obtenidos con imágenes sintéticas nuevas para escoger el valor de ***spread*** con cual se obtenía la mayor cantidad imágenes correctamente inferidas. De esta forma, se comprueba que se escoge al que mejor generaliza cuando se usan imágenes nuevas. En la figura 4-33 se muestra el resultado de la inferencia de imágenes sintéticas nuevas para el caso

libro. El valor de ***spread*** con el que se obtuvo mayor cantidad de imágenes correctamente inferidas fue 200´000.

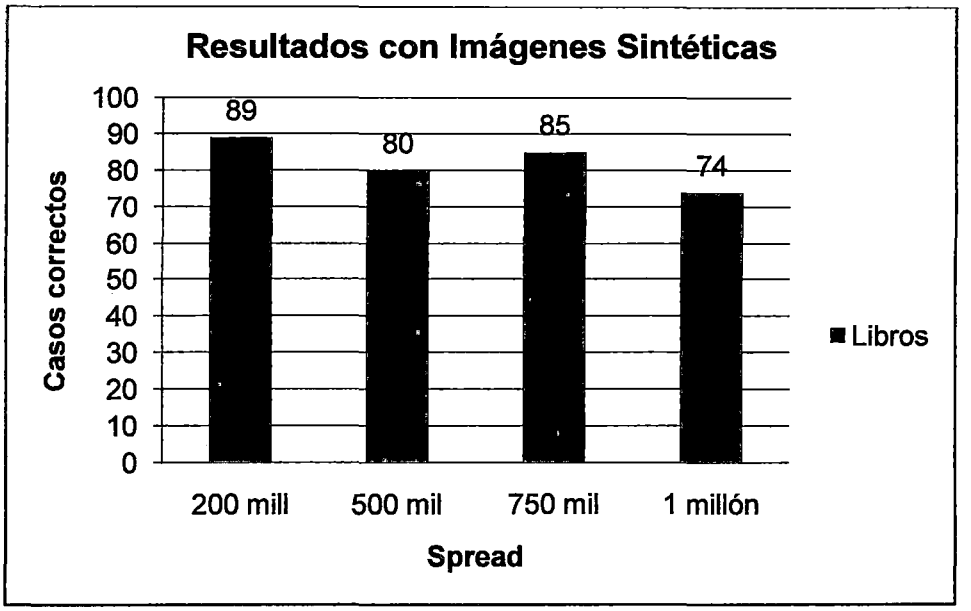


Figura 4-33. Resultado de la inferencia de imágenes sintéticas de libros.

En la figura 4-34 se muestran los resultados de la inferencia de imágenes sintéticas del caso de las tasas. Para este caso el valor spread de 500´000 produjo el mayor número de imágenes correctamente inferidas.

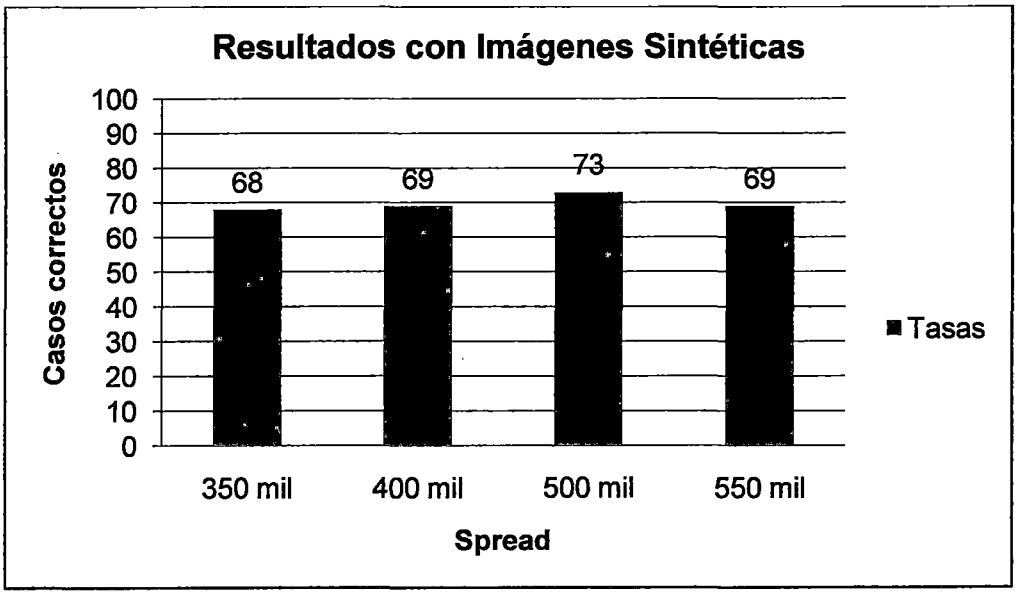


Figura 4-34. Resultado de la inferencia de imágenes sintéticas de tasas.

Los pasos realizados para el caso de los libros y las tasas se repitieron para el caso de los platos encontrando un valor de spread que se usará para la inferencia de imágenes reales. Valores bajos de ***spread*** nos pueden dar buenos resultados con la data de entrenamiento pero no con datos nuevos. Para el caso de las tasas se escogió la red entrenada con valor de ***spread*** de 500 mil, en el caso de los libros la red entrenada con valor de ***spread*** de 200 mil y el caso de los platos la red entrenada con valor de ***spread*** de 500 mil.

El tiempo de entrenamiento es una diferencia resaltante entre los tres algoritmos. Las redes RBF fue el algoritmo que en promedio tomó más en el tiempo de entrenamiento, alrededor de 48 minutos. El tiempo de entrenamiento del algoritmo de Regresión Logística toma un poco menos de un minuto y en el caso de las redes Perceptrón Multicapa entre uno a tres minutos aproximadamente, debido a que en algunos casos convergía más rápidamente que en otros. En la figura 4-35 se muestra la gráfica del tiempo de entrenamiento en función del número de neuronas obtenidas del entrenamiento de una red RBF del caso de las tasas para un valor de ***spread*** de 400´000 y un error a alcanzar de 0,05.

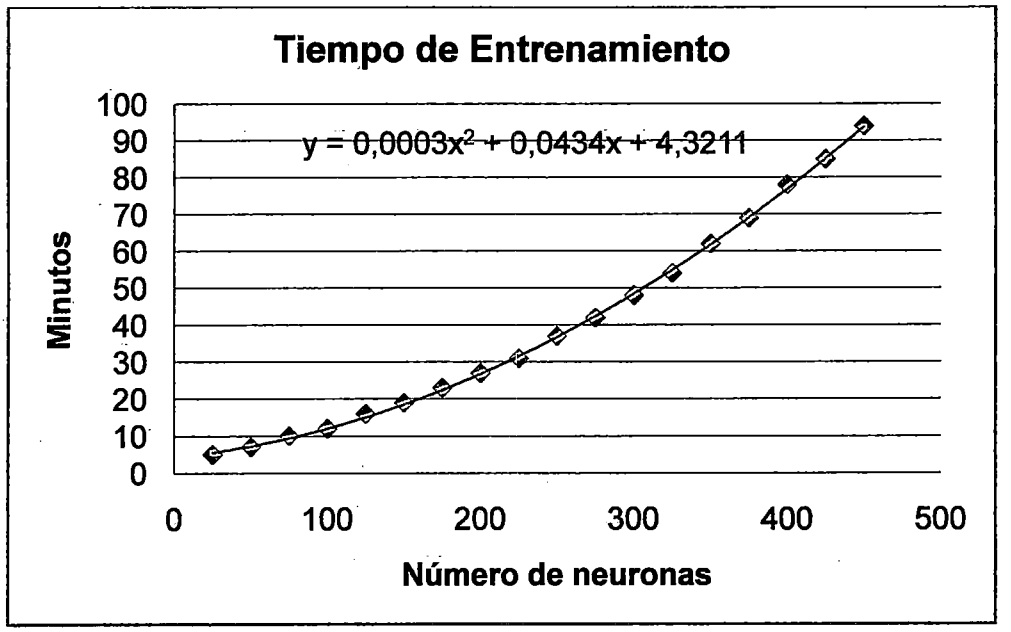


Figura 4-35. Gráfica del tiempo de entrenamiento de una red RBF.

4.5. Comparación de los Resultados de la Inferencia de Imágenes Sintéticas

Los resultados obtenidos de la inferencia de imágenes sintéticas con los algoritmos de aprendizaje de RL, MLP y RBF se presentan en forma comparativa en las siguientes gráficas. El algoritmo RL se entrenó para las cuatro formas, las redes MLP sólo para la tercera y cuarta forma y las redes RBF sólo para la tercera. Las pruebas se realizaron con 100 imágenes de las cuales las cantidades de imágenes inferidas correctamente se indican en las figuras 4-36 y 4-37.

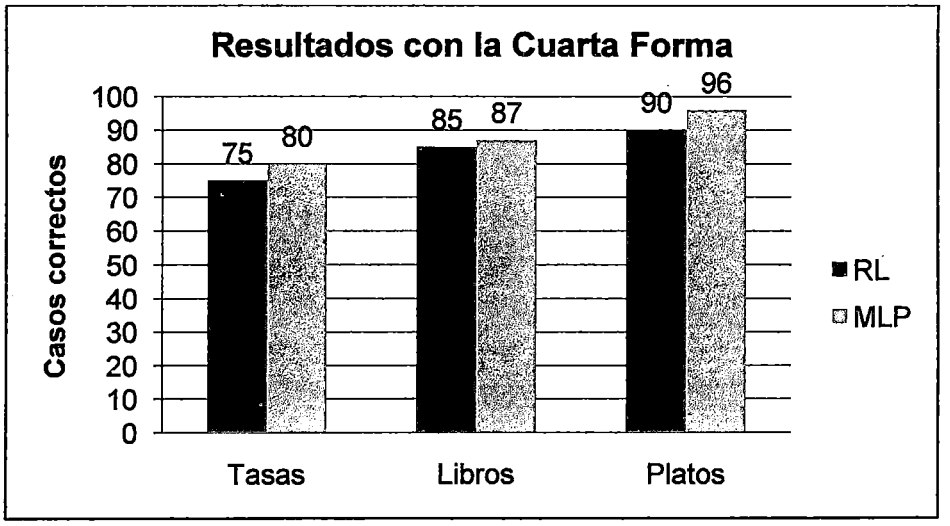


Figura 4-36. Gráfico comparativo entre RL y MLP usando la cuarta forma.

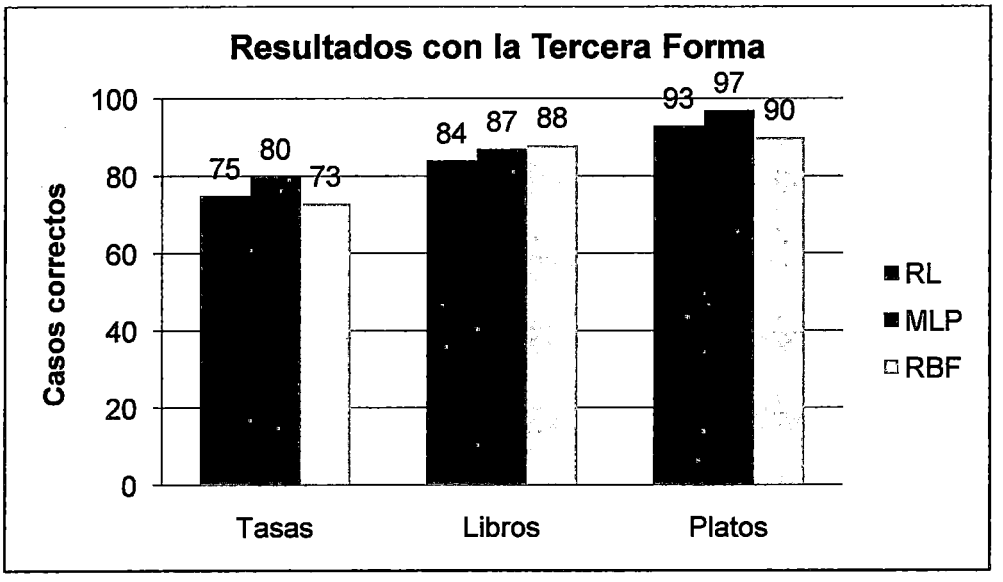


Figura 4-37. Gráfico comparativo entre RL, MLP y RBF usando la tercera forma.

Como se puede apreciar en la gráfica de la figura 4-36, la mayor cantidad de imágenes correctamente inferidas se obtiene con las redes Perceptrón Multicapa para los tres casos de entrenamiento. En la figura 4-37 se observa que las redes Perceptrón Multicapa tienen los mejores resultados en los casos de las tasas y los platos, pero en el caso de los libros los mejores resultados se obtienen con las redes RBF. En las figuras 4-38, 4-39 y 4-40 se muestran algunas imágenes de la inferencia de puntos de agarre usando las imágenes sintéticas.

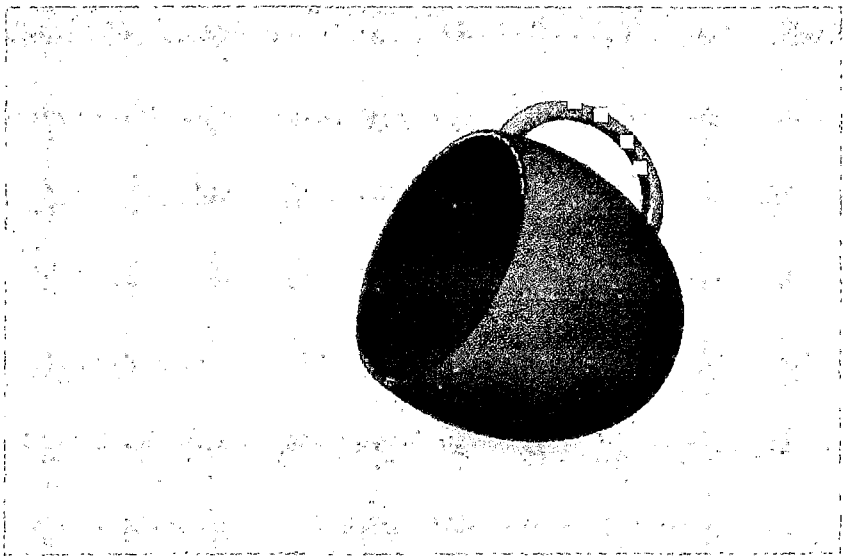


Figura 4-38. Inferencia de la imagen sintética de una tasa.

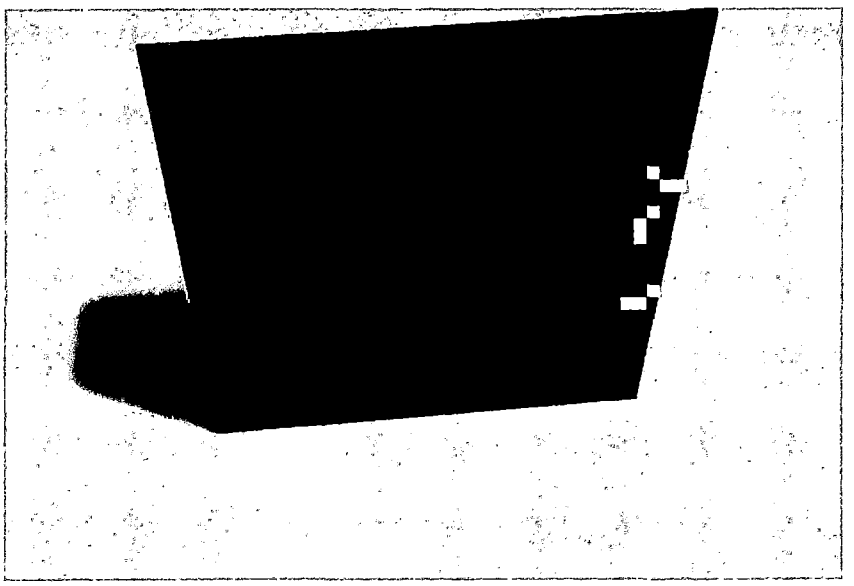


Figura 4-39. Inferencia de la imagen sintética de un libro.

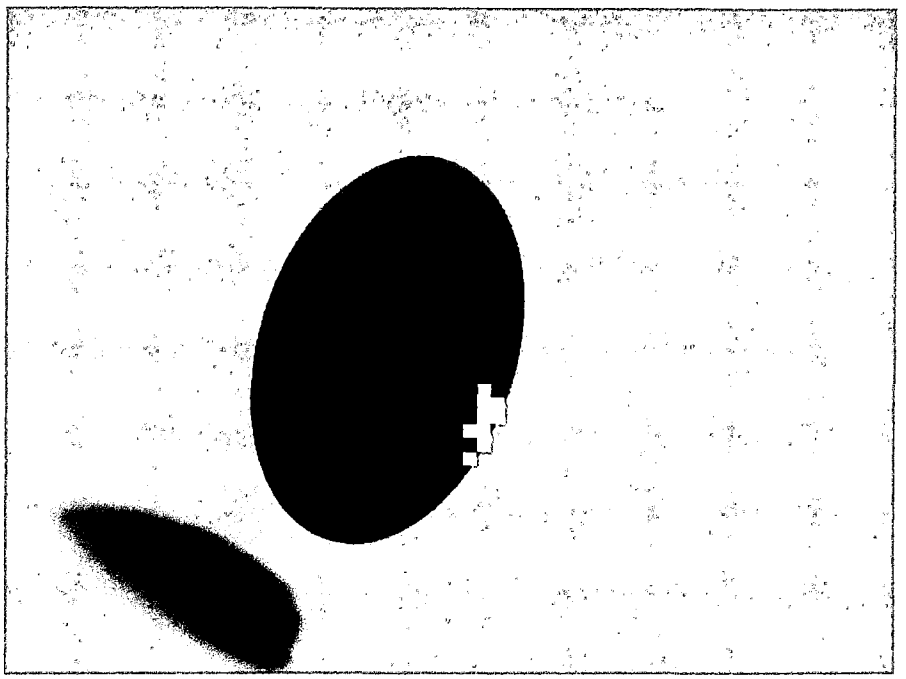


Figura 4-40. Inferencia de la imagen sintética de un plato.

En la inferencia de imágenes sintéticas se observó que se pueden presentar varias regiones inferidas como puntos de agarre formando un área (ver figura 4-40) o dispersas (ver figura 4-38 y 4-39). Esto es posible porque en el etiquetamiento de la data los índices positivos forman un área de agarre. Por lo tanto, para poder realizar la triangulación se tiene que calcular el centroide de dicha área y cuando presentan más de una se tiene que seleccionar la mayor de estas y eliminar las pequeñas. En el caso del plato no se puede calcular el centroide debido a que este se puede ubicar en centro del plato, por lo tanto, se tiene que seleccionar una porción del área inferida.

4.6. Pruebas de Inferencia de Imágenes de Objetos Reales

Después de evaluar los algoritmos de aprendizaje con imágenes sintéticas se evaluaron los algoritmos con 200 imágenes de objetos reales captadas en diferentes posiciones y a diferentes distancias por una cámara web. Los objetos estaban ubicados sobre un fondo verde. La figura 4-41 muestra la imagen de una tasa donde se inferido como punto de agarre un borde de la tasa.

Figura 4-41. Inferencia de la imagen real de una tasa.

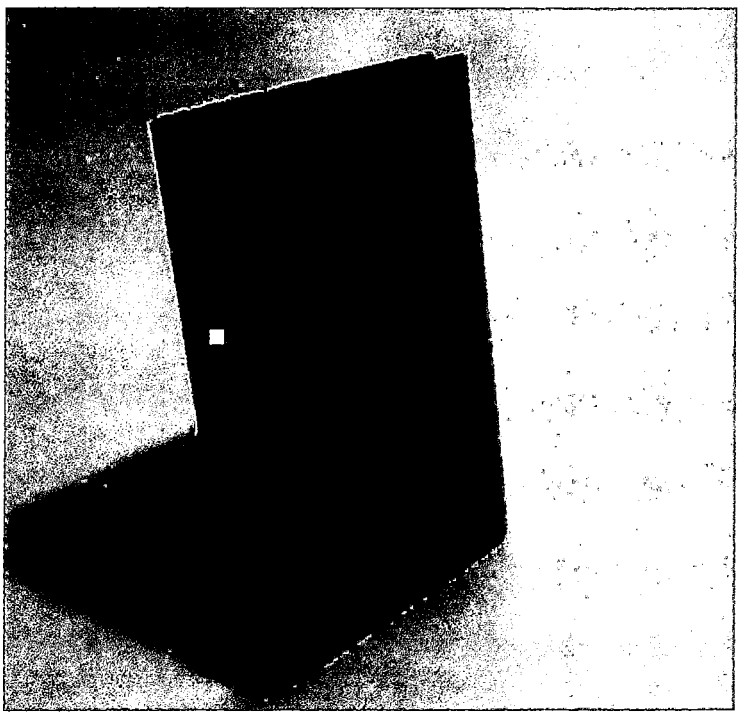


Figura 4-42. Inferencia de la imagen real de un libro.

En la figura 4-42 se muestra la imagen de un libro donde se ha seleccionado como un posible punto de agarre un punto ubicado en el lomo. En la imagen del plato que se muestra en la figura 4-43 se ha inferido como punto de agarre uno de sus bordes.

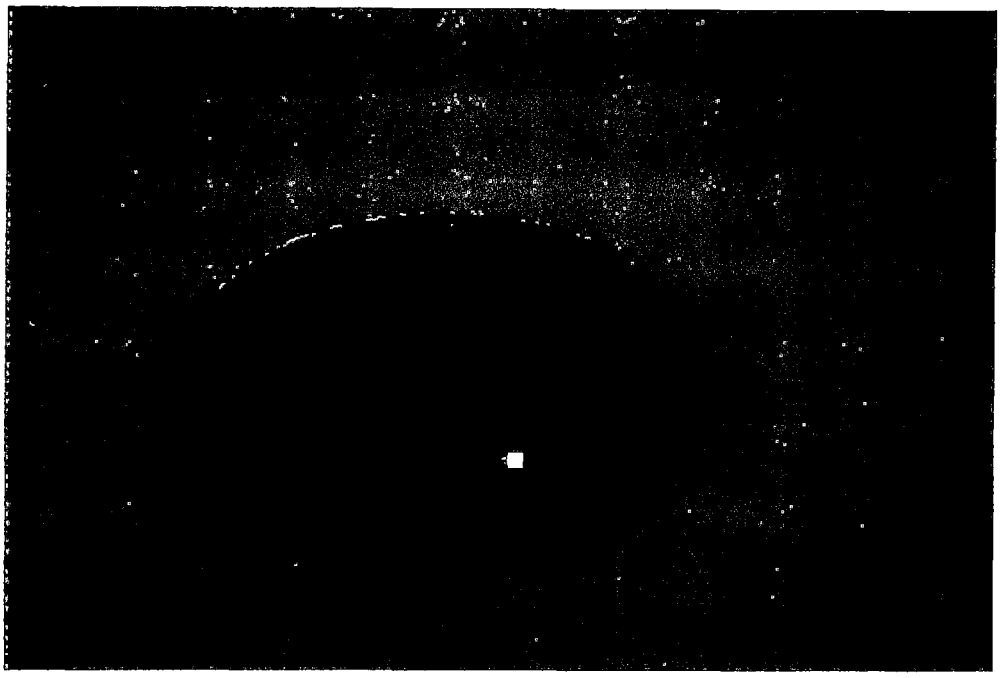


Figura 4-43. Inferencia de la imagen real de un plato.

Los resultados de la inferencia de todas las imágenes de los objetos reales que se capturaron se muestran a continuación (ver figura 4-44). Se probó 85 imágenes de tasas, 57 imágenes de libros y 58 de platos que fueron tomadas desde diferentes posiciones por las dos cámaras del par estéreo.

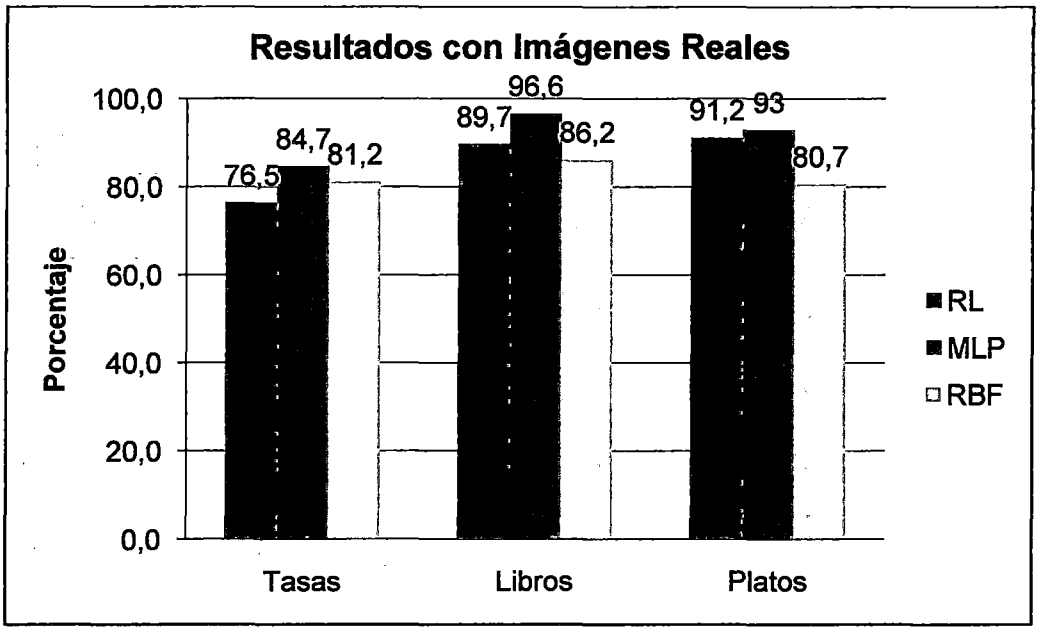


Figura 4-44. Gráfica de los resultados con imágenes reales.

De la figura 4-44 se puede apreciar que para los tres casos se obtuvieron mejores resultados con las redes Perceptrón Multicapa. En el caso de las tasas, las redes Perceptrón Multicapa y las redes con Función de Base Radial tienen resultados cercanos. En el caso de los platos, la Regresión Logística y las redes Perceptrón Multicapa tienen también resultados cercanos. Finalmente, se puede decir que para este problema de clasificación las redes Perceptrón Multicapa tienen mayor capacidad de generalización a pesar de que solo usan el 60% de datos de entrenamiento.

Los resultados de la inferencia de imágenes sintéticas mostraron también una tendencia a obtener mejor clasificación con las redes Perceptrón Multicapa que con los otros dos algoritmos. La figura 4-44 muestra como los algoritmos entrenados han logrado generalizar el aprendizaje para imágenes de objetos reales.

En algunos casos donde no se infirió correctamente se encontró que no había una buena segmentación debido a que el color del objeto era parecido al del fondo. Por último, a veces la presencia de algunas sombras en el caso de los platos hacía difícil la segmentación.

4.7. Obtención de Coordenadas Tridimensionales de objetos reales

Después de mostrar los resultados de evaluar los algoritmos de aprendizaje con imágenes de objetos reales se va a explicar cómo construyó el par estéreo con el cual se capturó las imágenes del objeto (izquierda y derecha) para poder realizar la triangulación. Este par estéreo se colocó sobre un espacio plano cubierto por un fondo verde sobre el cual se colocaron los objetos de los cuales se quiso obtener las coordenadas tridimensionales de sus puntos de agarre.

4.7.1. Formación del par estéreo

El par estéreo está formado por dos cámaras web marca Omega (2 mega pixeles) montadas en una pequeña estructura de aluminio (ver figura 4-45). El campo de visión del par estéreo está formado por la intersección del ángulo de visión de las dos cámaras y la intersección depende de la distancia y el ángulo

entre ellas. Debido a que el ángulo de visión de las cámaras es un parámetro constante; para definir nuestro campo de visión se puede realizar variando los otros dos parámetros.

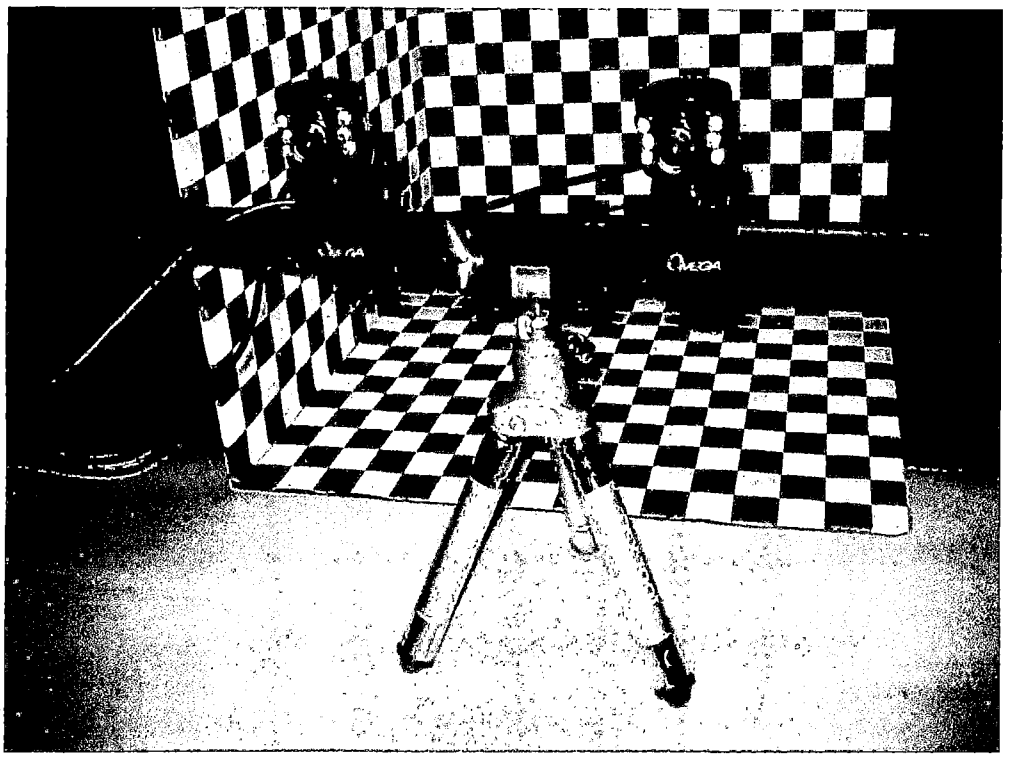


Figura 4-45. Par estéreo.

En primer lugar, el par estéreo debe estar lo más alineado posible vertical y horizontalmente para que las imágenes no estén rotadas. Para esto se sitúa el par estéreo frente a los planos de calibración (ver figura 4-46) a unos 50 centímetros aproximadamente, considerando que este puede ser el rango de alcance de un brazo robótico y también de un ser humano. Para alinear verticalmente el par estéreo se consideró como referencia el eje Z del sistema de coordenadas de calibración y para alinear horizontalmente se tomó como referencia una línea de pendiente -1 perteneciente al plano XY.

Luego de alinear ambas cámaras queda definir la distancia y el ángulo entre ellas. Estas dimensiones se determinaron mediante la experimentación y mediante criterios como los que explican a continuación. Estas dimensiones nos

definen el campo de visión del par estéreo, el cual va a permanecer constante en las pruebas que realicemos.

Para nuestra aplicación, las cámaras deben estar lo suficientemente cerca y formar entre ellas un ángulo pequeño, de esta manera tendrán una imagen de la escena muy similar lo cual facilita la correlación, aunque incrementa el error en la triangulación. Por el contrario, si incrementamos la distancia entre las cámaras y el ángulo entre ellas (cercano a 90 grados), los cálculos de triangulación serán más exactos pero la correlación será más difícil y en algunos casos podría producir error.

Por ejemplo [17] usa un par estéreo con un ángulo aproximado de 90 grados entre las cámaras para medir los pequeños desplazamientos y orientación de un actuador. En nuestro caso se probó distancias entre 11 a 20 centímetros y el ángulo se varió tratando de que ambas cámaras visualicen la misma escena del sistema de calibración ubicado a unos 50 centímetros aproximadamente.

4.7.2. Calibración del par estéreo

Para poder realizar la triangulación de puntos primero se deben hallar los parámetros de ambas cámaras. Para esto se realizó la calibración de ambas en el espacio de trabajo usando una nueva herramienta de calibración: una estructura formada por los planos XY, YZ y ZX (ver figura 4-46). Se eligió esta estructura porque permite abarca un mayor espacio de puntos de calibración y por su facilidad para definir el sistema de coordenadas y obtener medidas. En las pruebas de triangulación se tomarán como referencia el sistema de coordenadas de la nueva estructura de calibración.

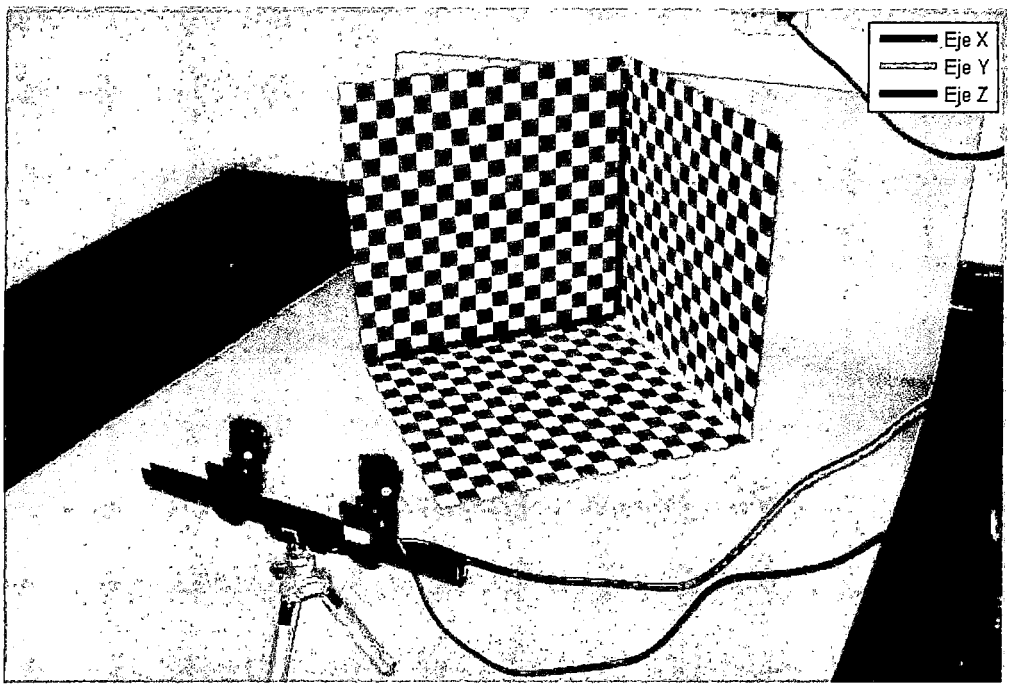


Figura 4-46. Sistema de coordenadas usado para la calibración del par estéreo.

Se realizaron los mismos pasos de calibración mencionados en el capítulo III pero ahora tomando como referencia el nuevo sistema de coordenadas y se encontraron nuevamente los parámetros de ambas cámaras. En la figura 4-47 se muestran los puntos correspondientes a los tres planos XY, YZ y ZX que se usaron para el cálculo de los parámetros.

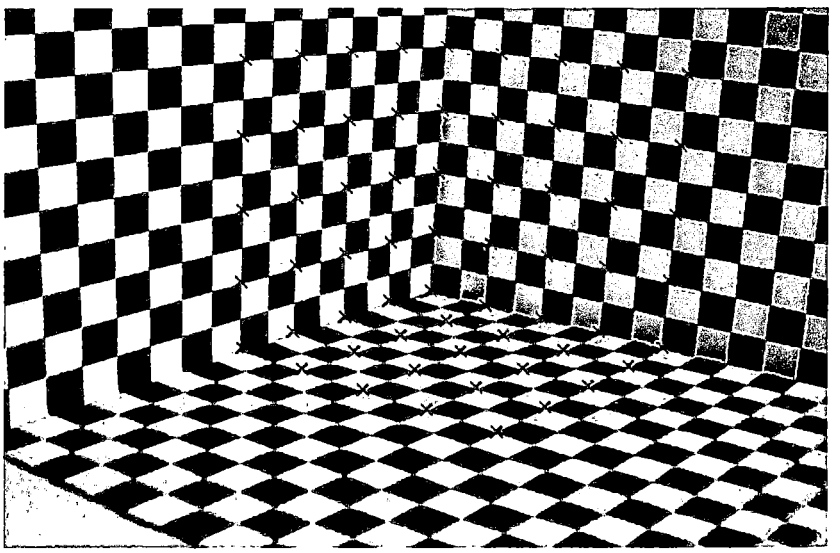


Figura 4-47. Toma de puntos de los tres planos.

4.7.3. Primeras pruebas de triangulación

Se empezó triangulando puntos que no pertenecieran a los planos usados en la calibración de las cámaras. Para esto se desplazó un cubo con puntos fácilmente identificables dentro del espacio de calibración y se calculó sus coordenadas u y v de los puntos en cada imagen de forma visual.

Se tomaron en total 38 puntos y se trianguló las coordenadas de cada uno. Estos puntos se usarán para conocer qué errores se pueden encontrar mediante la triangulación. Los errores de triangulación están relacionados con la precisión de la calibración y la precisión con la que ubican las coordenadas u y v de los puntos en cada imagen. Las unidades de medida en este sistema de calibración equivalen a 5 centímetros. Las coordenadas del punto medio del soporte de la cámara con respecto al sistema de coordenadas de calibración son (50, 50, 28) centímetros y la distancia entre las cámaras fue 16 centímetros.

Nº	Coordenadas medidas			Coordenadas calculadas			Errores		
	X (cm)	Y (cm)	Z (cm)	X (cm)	Y (cm)	Z (cm)	e_x (cm)	e_y (cm)	e_z (cm)
1	17,5	17,5	12,5	17,7	17,9	12,7	0,2	0,3	0,2
2	17,5	12,5	12,5	17,5	12,9	12,6	0	0,4	0,1
3	17,5	7,5	12,5	17,4	7,7	12,6	0,1	0,2	0,1
4	17,5	12,5	7,5	17,7	12,8	7,6	0,2	0,3	0,1
5	17,5	7,5	7,5	17,6	7,8	7,6	0,1	0,3	0,1
6	17,5	12,5	2,5	17,7	12,8	2,7	0,2	0,3	0,2
7	17,5	7,5	2,5	17,6	7,8	2,6	0,1	0,3	0,1
8	17,5	17,5	7,5	17,7	17,9	7,8	0,2	0,4	0,3
9	17,5	17,5	2,5	17,6	17,8	2,8	0,1	0,3	0,3
10	12,5	17,5	7,5	12,7	17,9	7,8	0,2	0,4	0,3
11	12,5	17,5	2,5	12,9	17,9	3	0,4	0,4	0,5
12	7,5	17,5	7,5	7,8	17,9	7,8	0,3	0,4	0,3
13	7,5	17,5	2,5	7,7	17,8	2,8	0,2	0,3	0,3
14	12,5	17,5	12,5	12,8	18	12,8	0,3	0,5	0,3
15	7,5	17,5	12,5	7,9	18,1	12,8	0,4	0,6	0,3
16	12,5	12,5	12,5	12,9	13,2	12,9	0,4	0,7	0,4
17	7,5	12,5	12,5	7,6	13	12,8	0,1	0,5	0,3
18	12,5	7,5	12,5	12,5	8	12,7	0	0,5	0,2
19	7,5	7,5	12,5	7,8	8,3	13	0,3	0,8	0,5

Tabla 4-9. Cálculo de coordenadas tridimensionales parte 1.

La tabla 4-9 y la tabla 4-10 muestran las coordenadas medidas y calculadas de 38 puntos tridimensionales y los errores encontrados. Finalmente la tabla 4-11 muestra el promedio de errores en cada eje.

Nº	Coordenadas medidas			Coordenadas calculadas			Errores		
	X (cm)	Y (cm)	Z (cm)	X (cm)	Y (cm)	Z (cm)	e_x (cm)	e_y (cm)	e_z (cm)
1	12,5	12,5	12,5	12,9	13,1	12,8	0,4	0,6	0,3
2	12,5	7,5	12,5	12,6	7,9	12,6	0,1	0,4	0,1
3	12,5	2,5	12,5	12,6	3	12,6	0,1	0,5	0,1
4	12,5	7,5	7,5	12,7	7,9	7,6	0,2	0,4	0,1
5	12,5	2,5	7,5	12,8	2,9	7,6	0,3	0,4	0,1
6	12,5	7,5	2,5	12,8	7,9	2,7	0,3	0,4	0,2
7	12,5	2,5	2,5	12,8	2,9	2,6	0,3	0,4	0,1
8	12,5	12,5	7,5	12,7	13	7,8	0,2	0,5	0,3
9	12,5	12,5	2,5	12,8	13	2,7	0,3	0,5	0,2
10	7,5	12,5	7,5	7,5	12,7	7,6	0	0,2	0,1
11	7,5	12,5	2,5	7,7	12,7	2,7	0,2	0,2	0,2
12	2,5	12,5	7,5	3,1	13	7,9	0,6	0,5	0,4
13	2,5	12,5	2,5	2,6	12,6	2,6	0,1	0	0,1
14	7,5	12,5	12,5	7,6	12,8	12,6	0,1	0,3	0,1
15	2,5	12,5	12,5	2,9	12,9	12,8	0,4	0,4	0,3
16	7,5	7,5	12,5	7,6	7,8	12,7	0,1	0,3	0,2
17	2,5	7,5	12,5	3	8,1	12,9	0,5	0,6	0,4
18	7,5	2,5	12,5	7,7	3	12,7	0,2	0,5	0,2
19	2,5	2,5	12,5	2,9	3,1	12,9	0,4	0,6	0,4

Tabla 4-10. Cálculo de coordenadas tridimensionales parte 2.

Promedio de Errores		
e_x (cm)	e_y (cm)	e_z (cm)
0,2	0,4	0,2

Tabla 4-11. Promedio de errores encontrados.

Como se puede apreciar en la tabla 4-11 el error promedio máximo es 0,4 centímetros y se presenta en el eje Y. Estas pruebas sirven para conocer los errores que se podrían presentar en una triangulación. En ellas la captura de puntos fue de forma visual, en otras palabras, la persona que hacía la calibración

sabía qué punto tomar en la primera imagen y qué punto le correspondía en la segunda.

4.7.4. Aplicación de la Correlación

En un principio se había propuesto inferir el punto de agarre en las dos imágenes capturadas (izquierda y derecha), pero en la práctica se demostró que en algunos casos había una diferencia en la posición de los dos puntos, y esto producía un error en el cálculo de coordenadas tridimensionales. Es por esta razón que se buscó un método en visión por computador que nos permitiera encontrar este segundo punto considerando que ya se había inferido correctamente el primero. De esta forma se empezó a usar la correlación cruzada normalizada para poder encontrar las coordenadas u y v del punto de agarre en la segunda imagen. Para esto, una vez obtenidas las coordenadas del primer punto se toma una ventana con la cual se buscará una similar en la segunda imagen. El tamaño de la ventana es 200 pixeles x 200 pixeles para que pueda tomar datos característicos ya que si fuera más pequeña es probable que la búsqueda en la segunda imagen se haga más difícil.

4.7.5. Triangulación de puntos de agarre de objetos reales

Después de realizar las primeras pruebas de triangulación se hará la triangulación de los puntos de agarre obtenidos de la inferencia de imágenes de objetos usando los algoritmos de aprendizaje ya entrenados. Se desplazó objetos dentro del espacio de trabajo, se infirió el punto de agarre, se realizó la correlación para encontrar el mismo punto en la segunda imagen y se trianguló estos dos puntos para calcular las coordenadas tridimensionales (ver figura 4-48).

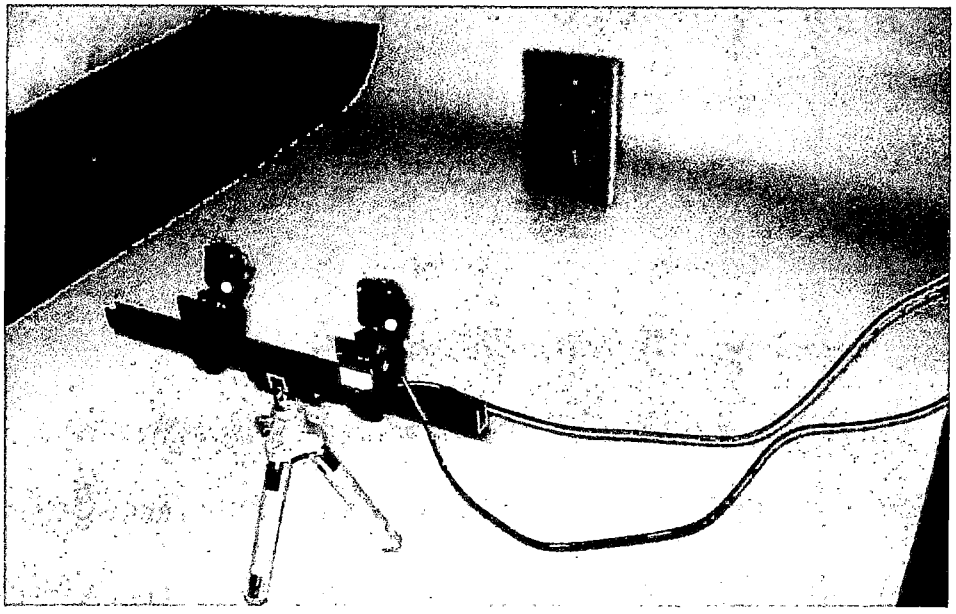


Figura 4-48. Par estéreo y un objeto de prueba.

En la triangulación de objetos como tasas y platos, que presentan cierta simetría, puede suceder que se ve casi la misma imagen cuando se observa desde otro punto cercano. En estos casos las coordenadas del segundo punto (u_2 y v_2 en la segunda imagen) hallado por la aplicación de la correlación están desplazadas con respecto a las verdaderas coordenadas del punto ($u_2 + e_{u_2}$ y $v_2 + e_{v_2}$) debido a la simetría del objeto al ser visualizado desde la segunda cámara (ver figura 4-49).

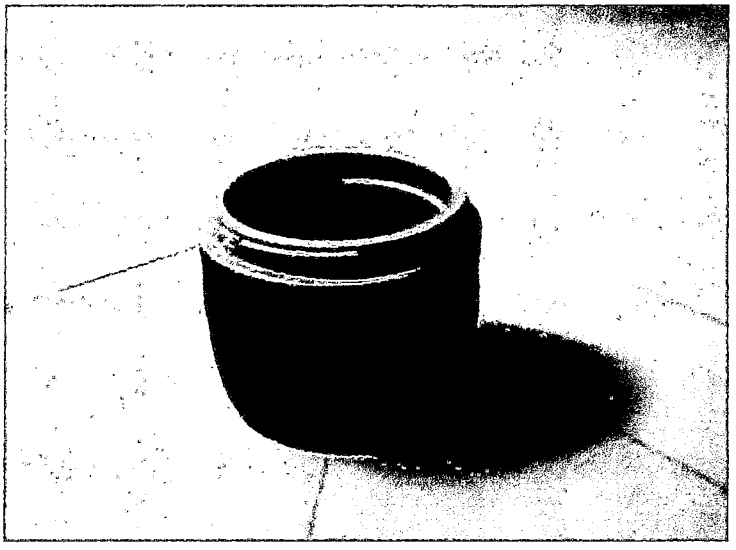


Figura 4-49. Correlación de puntos.

Este desplazamiento o error en el cálculo de las coordenadas del segundo punto produce un error promedio máximo 1,93 centímetros en el cálculo de las coordenadas tridimensionales cuando se triangula objetos cilíndricos (ver figura 4-49) en el área donde se tomó los puntos para calibrar el sistema estéreo. Si se realizara la búsqueda de las coordenadas del segundo punto en forma visual (filas de color celeste en la tabla 4-12), el error promedio máximo se reduciría a 0,53 centímetros. De esta manera, se prueba que la mayor parte del error producido en el cálculo de coordenadas del segundo punto es por el desplazamiento o error al usar la correlación.

Nº	Coordenadas medidas			Coordenadas calculadas			Errores		
	X (cm)	Y (cm)	Z (cm)	X (cm)	Y (cm)	Z (cm)	e_x (cm)	e_y (cm)	e_z (cm)
1	9	4	7,3	7,96	2,66	6,04	1,04	1,34	1,26
				8,95	3,47	6,67	0,05	0,53	0,63
2	7	6	7,2	6,44	5,03	6,32	0,56	0,97	0,88
				7,28	5,63	6,83	0,28	0,37	0,37
3	8,9	3,4	6,7	7,81	2,49	5,95	1,09	0,91	0,75
				8,68	3,2	6,5	0,22	0,2	0,2
4	4,3	10	7,9	7,98	12,38	9,36	3,68	2,38	1,46
				5,03	10,36	7,8	0,73	0,36	0,1
5	7,7	20	7,3	5,86	19,27	6,29	1,84	0,73	1,01
				7,87	20,08	7,47	0,17	0,08	0,17
6	7,7	6,3	7,3	6,18	5,96	6,09	1,52	0,34	1,21
				5,52	5,43	5,71	2,18	0,87	1,59
7	17,2	7,8	7,3	13,8	4,76	4,83	3,4	3,04	2,47
				17,04	7,86	7,33	0,16	0,06	0,03
8	12,7	7,8	7,3	14,98	9,35	9,13	2,28	1,55	1,83
				13,11	7,67	7,92	0,41	0,13	0,62

Tabla 4-12. Cálculo de coordenadas tridimensionales parte 3.

Promedio de Errores		
e_x (cm)	e_y (cm)	e_z (cm)
1,93	1,41	1,36
0,53	0,33	0,46

Tabla 4-13. Promedio de errores encontrados.

El efecto de este error se incrementa cada vez que el punto a triangular se aleja del área en donde se realizó la calibración; es decir, al área donde uno de los ejes X, Y o ambos son negativos (ver tabla 4-14). En esta área el error promedio máximo es 4,05 centímetros y realizando la corrección obtenemos un error promedio de 1,22 centímetros (ver tabla 4-15).

Nº	Coordenadas medidas			Coordenadas calculadas			Errores		
	X (cm)	Y (cm)	Z (cm)	X (cm)	Y (cm)	Z (cm)	e_x (cm)	e_y (cm)	e_z (cm)
1	-3,3	-0,6	6,3	-6,77	-2,02	4,97	3,47	1,42	1,33
				-6,2	-1,81	5,24	2,9	1,21	1,06
2	-8,7	-13	7,4	-13,92	-18,07	4,67	5,22	5,07	2,73
				-8,62	-13,11	6,91	0,08	0,11	0,49
3	3	-6,2	6	0,73	-8,34	4,46	2,27	2,14	1,54
				2,72	-6,3	5,55	0,28	0,1	0,45
4	9,6	-5,5	5,4	5,6	-7,86	3,84	4	2,36	1,56
				7,58	-5,62	5,08	2,02	0,12	0,32
5	-11,3	-8,5	6,4	-17,12	-12,04	3,81	5,82	3,54	2,59
				-12,55	-8,39	5,71	1,25	0,11	0,69
6	-7	8	7,1	-8,84	7,77	5,93	1,84	0,23	1,17
				-6,35	8,93	7,01	0,65	0,93	0,09
7	-13,1	9,6	6,4	-7,41	12,49	8,44	5,69	2,89	2,04
				-14,48	9,21	5,59	1,38	0,39	0,81

Tabla 4-14. Cálculo de coordenadas tridimensionales parte 4.

Promedio de Errores		
e_x (cm)	e_y (cm)	e_z (cm)
4,05	2,52	1,85
1,22	0,42	0,56

Tabla 4-15. Promedio de errores encontrados.

En el caso de los libros este desplazamiento no es muy grande y por lo tanto la influencia en el error no es tan importante. Esto se puede observar en los errores mostrados en la tabla 4-16. La tabla 4-17 muestra que se obtiene un error en promedio máximo de 1,26 centímetros.

Nº	Coordenadas medidas			Coordenadas calculadas			Errores		
	X (cm)	Y (cm)	Z (cm)	X (cm)	Y (cm)	Z (cm)	e_x (cm)	e_y (cm)	e_z (cm)
1	0	9,4	6	1,24	9,44	6,51	1,24	0,04	0,51
2	16	7,4	1,3	13,85	4,52	0,55	2,15	2,88	0,75
3	11	11,5	1	11,62	11,15	1,41	0,62	0,35	0,41
4	6	12,6	0,4	5,76	12,36	0,25	0,24	0,24	0,15
5	-3,5	3	1	-3,87	2,73	0,81	0,37	0,27	0,19
6	9,7	19,4	9,8	8,89	18,97	10,11	0,81	0,43	0,31
7	6	18	12,7	5,55	17,47	12,5	0,45	0,53	0,2
8	6	12,5	8,5	4,71	10,79	7,82	1,29	1,71	0,68
9	8,2	19,4	12	5,2	17,48	11,03	3	1,92	0,97
10	9,5	22	5,5	7,63	20,49	4,52	1,87	1,51	0,98
11	8,3	17	8,5	7,2	15,76	8,04	1,1	1,24	0,46
12	11,5	19,5	12,2	9,56	18,2	11,26	1,94	1,3	0,94

Tabla 4-16. Cálculo de coordenadas tridimensionales parte 5.

Promedio de Errores		
e_x (cm)	e_y (cm)	e_z (cm)
1,26	1,04	0,54

Tabla 4-17. Promedio de errores encontrados.

De acuerdo a los errores obtenidos (ver tabla 4-15 y tabla 4-17) se recomienda trabajar dentro del espacio donde se realizó la calibración. Este espacio tiene que estar incluido dentro del espacio de trabajo del brazo robótico.

4.8. Simulación

Para comprender la incorporación del sistema de visión a un brazo robótico de una manera más fácil se realizó una simulación en video de cual se muestran algunas imágenes. A continuación se describirá paso a paso cada uno de los movimientos del brazo robótico explicando las consideraciones tomadas para esta simulación.

En la simulación se puede apreciar un brazo robótico de 5 grados de libertad con un efector final de dos dedos o pinzas que se desplazan en paralelo. Este brazo está ubicado sobre una mesa en la cual se va a colocar un objeto (plato), el cual va a ser manipulado llevándolo desde su posición original hacia uno de los lados de la mesa. El sistema de visión formado por dos cámaras está montado sobre un eslabón del brazo robótico (ver figura 4-50).

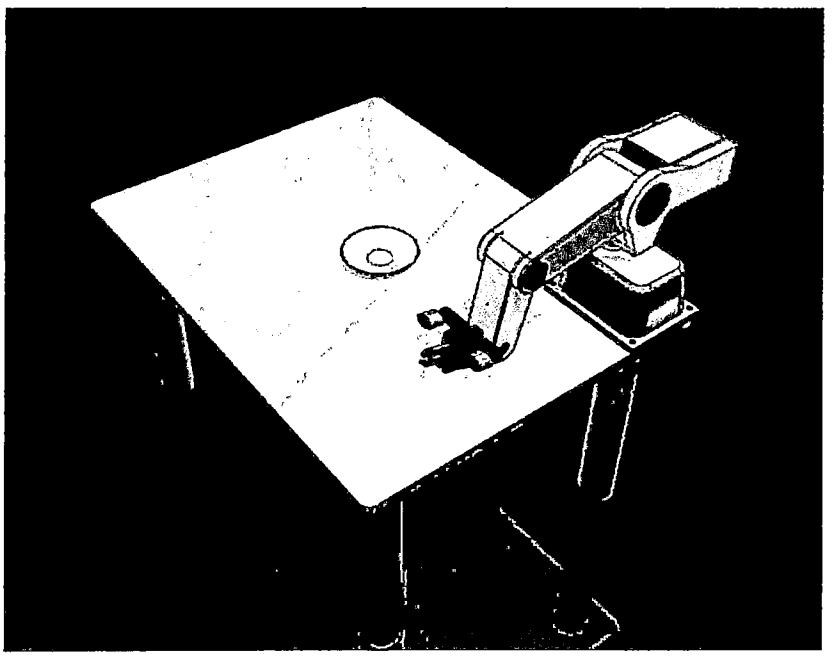


Figura 4-50. Posición inicial del brazo.

Inicialmente se desconoce la ubicación exacta del objeto por esto motivo el brazo robótico ejecutará primero un algoritmo de búsqueda partiendo desde la posición de reposo (una posición cerca a uno de los lados de la mesa) y desplazándose o haciendo un barrido con una de las cámaras del sistema de visión desde un punto superior a la mesa (ver figura 4-51).

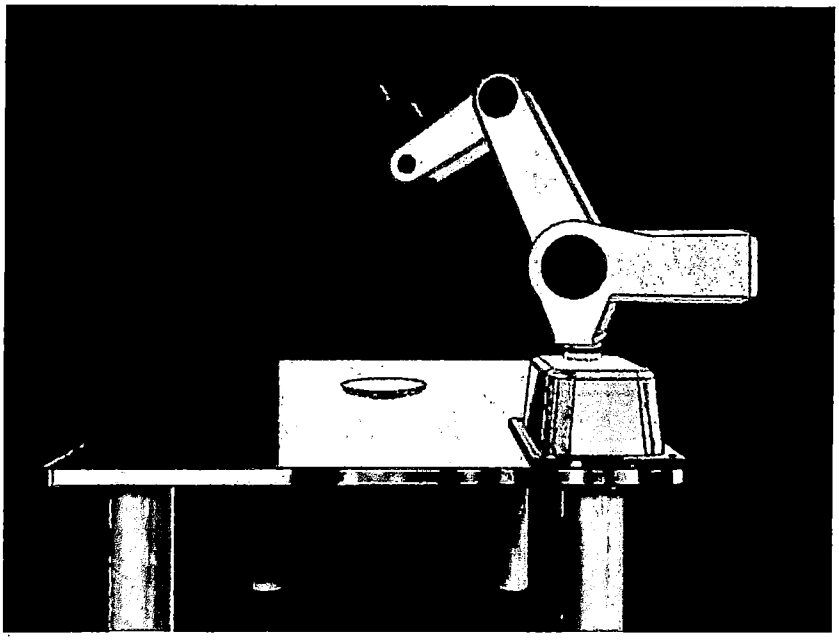


Figura 4-51. Inicio de la búsqueda del objeto.

Al encontrar o ubicar una parte del objeto en las imágenes captadas por la primera cámara (ver figura 4-52), el brazo robótico realizará movimientos tratando de que ambas cámaras tengan una imagen completa del objeto (ver figura 4-53). De esta forma se asegura que el objeto está dentro del campo común de visión de las dos cámaras y que el punto a identificar va a ser ubicable en las dos imágenes.

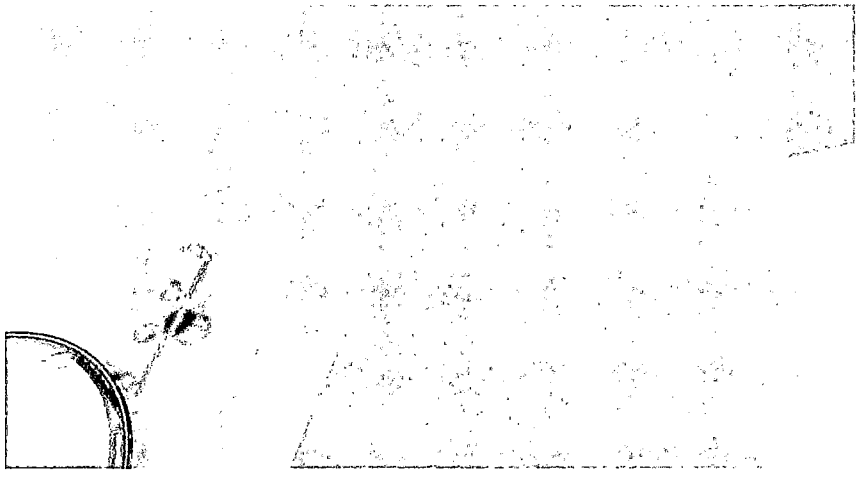


Figura 4-52. Imagen parcial del objeto.

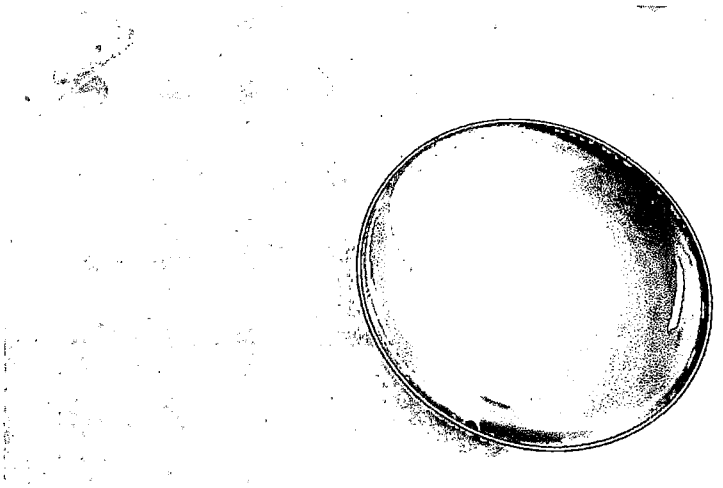


Figura 4-53. Imagen completa del objeto.

Luego de que el objeto se visualiza completamente con las dos cámaras, se procede a identificar el punto de donde se va a manipular el objeto (ver figura 4-53) y calcular las coordenadas tridimensionales de ese punto. Adicionalmente se asume la dirección más conveniente que debe adoptar el efector para capturar el objeto. En la figura 4-54 se observa esquemáticamente el proceso de triangulación para calcular las coordenadas tridimensionales así como también se muestran los sistemas de coordenadas del par estéreo y del brazo robótico.

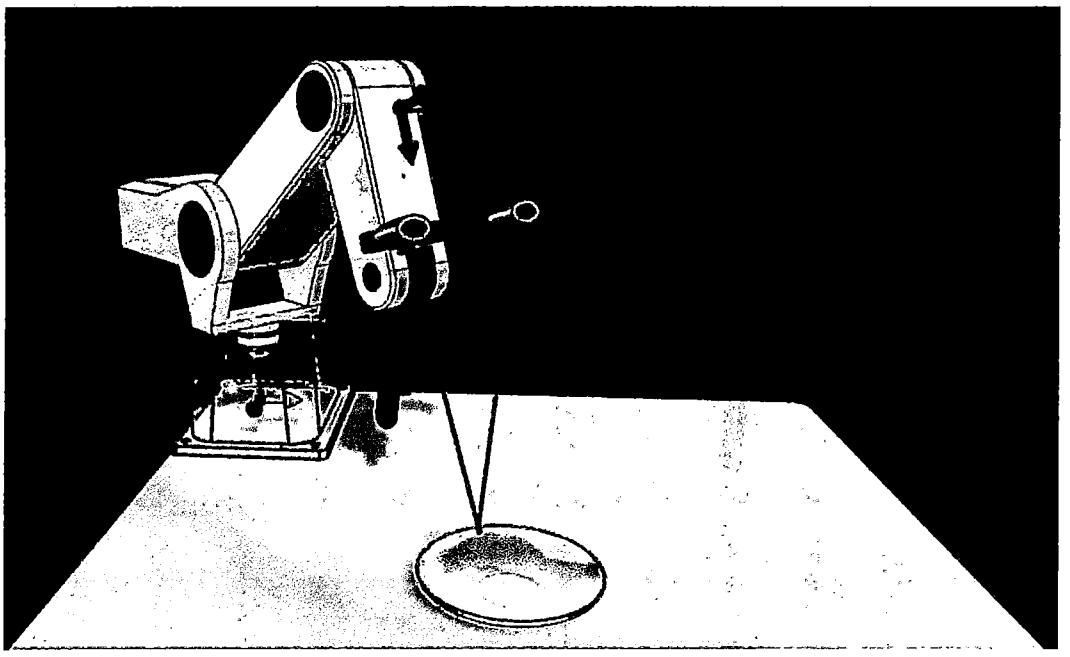


Figura 4-54. Triangulación del punto de agarre.

Lo siguiente es posicionar o configurar el brazo robótico para que el centro del efector coincida con las coordenadas del punto de agarre mediante movimientos de cada uno de sus eslabones. Esta es la parte donde se considera que la incorporación de sensores en el efector final (haptic sensors) podría corregir los errores en su posicionamiento.

Los movimientos en la etapa de captura son lentos debido a que movimientos bruscos y la inestabilidad del brazo pueden ocasionar posibles colisiones o fracturas del objeto (ver figura 4-55). Después de que el efector alcanzó la posición deseada, este se cierra hasta aplicar las fuerzas necesarias para evitar o minimizar el deslizamiento garantizando un agarre estable. Las fuerzas que se presentan en el punto de contacto entre el objeto y el efector final tienen dirección normal y tangencial a la superficie de contacto.

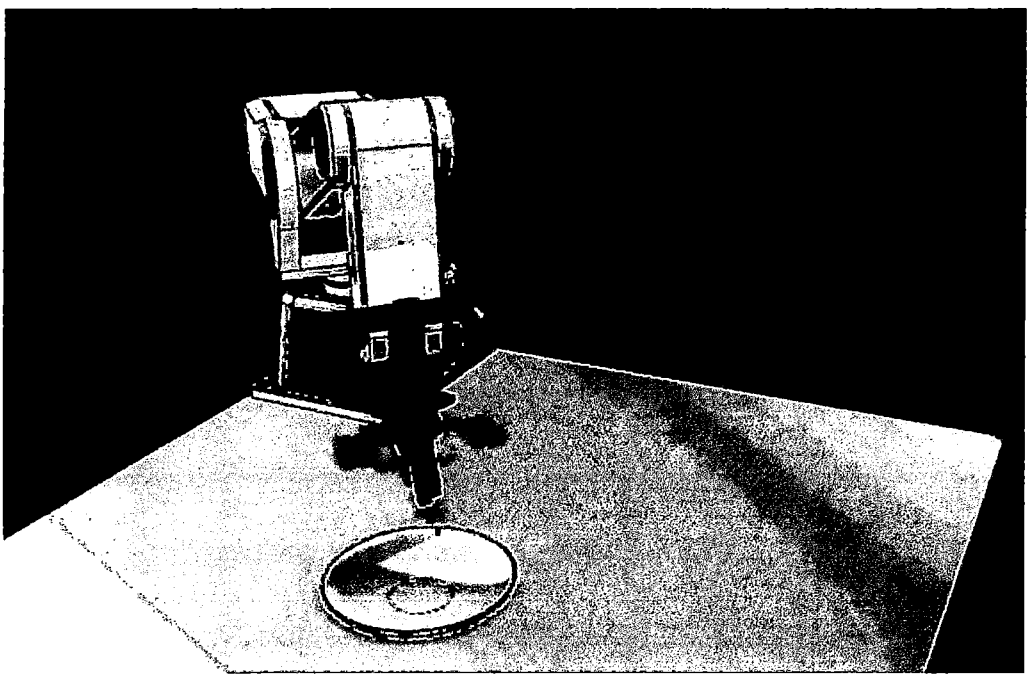


Figura 4-55. Efector final acercándose al objeto.

Luego de que el objeto ha sido elevado lentamente una altura suficiente para ser transportado, este será llevado a la posición deseada y bajado cuidadosamente hasta que este toque la superficie de la mesa. El siguiente paso es alejar el efector final tratando de evitar que colisione con la superficie de la mesa o la del objeto.

En la figura 4-56 se grafican las velocidades angulares de cada uno de los eslabones del brazo robótico durante la simulación. A partir del segundo 26 las velocidades del brazo, antebrazo y mano disminuyen debido a que en esa etapa se están realizando los movimientos para capturar y levantar el objeto.

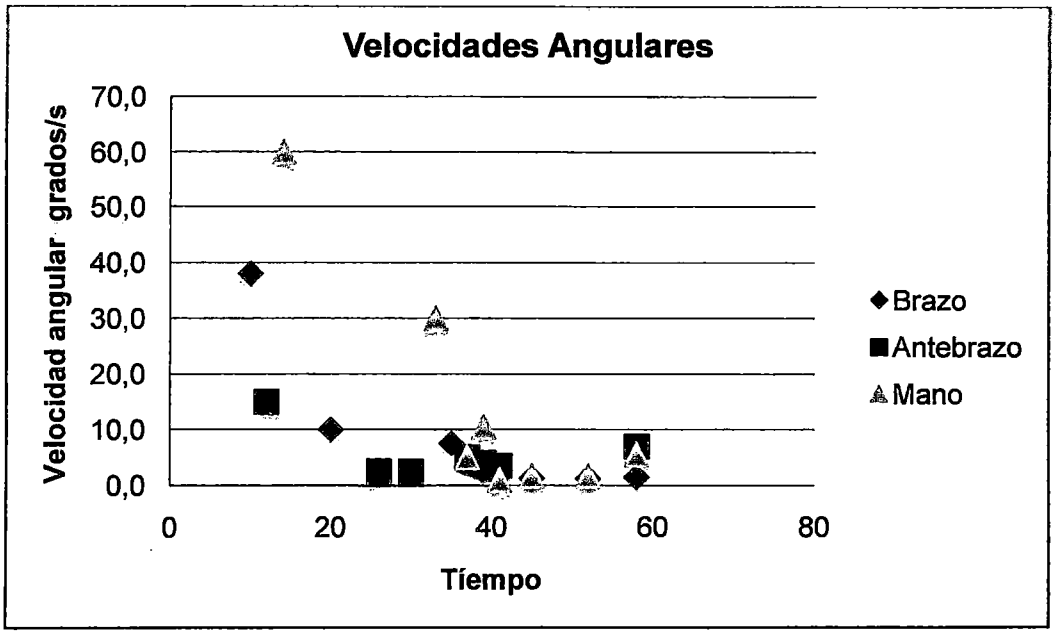


Figura 4-56. Velocidades angulares.

CONCLUSIONES

Las redes Perceptrón Multicapa, por los resultados obtenidos con imágenes sintéticas y de objetos reales y tiempo de entrenamiento, muestran que tienen una mejor generalización en comparación con los algoritmos de Regresión Logística y redes con Función de Base Radial. Por este motivo, se las considera como el modelo más apropiado para ser usado en esta aplicación.

La cantidad de características o patrones a clasificar es importante en el proceso de entrenamiento y evaluación, ya que un exceso de estas incrementa el tiempo de entrenamiento y la cantidad de memoria usada en la obtención del modelo. Por esta razón y por los resultados obtenidos, se demuestra que la tercera forma toma la cantidad suficiente de características para obtener una buena clasificación de las regiones.

El proceso de triangulación calcula las coordenadas tridimensionales con una exactitud que depende además de otros procesos anteriores como son la configuración de las cámaras, la calibración de estas y la correspondencia de los puntos en las dos imágenes, siendo esta última la más influyente en las pruebas que se realizaron. Si se hace una correcta configuración y correspondencia de puntos, los cálculos de esta información tendrán una exactitud aceptable para ser utilizables por el control de un brazo robótico.

La exactitud de la obtención de los parámetros intrínsecos y extrínsecos de las cámaras depende de la exactitud de las medidas de fabricación de la herramienta usada en la calibración, la cantidad de puntos y la exactitud con la que se hace la captura de puntos de imagen. Por esta razón, usar la herramienta de

calibración formada por los tres planos es lo más indicado para este caso ya que se puede abarcar la mayor parte del campo visual del par estéreo y el sistema de coordenadas está orientado hacia a este.

Para que la operación de correlación pueda encontrar el punto correspondiente en la segunda imagen se necesita una ventana lo suficientemente grande para que pueda albergar características singulares de la imagen, en caso contrario se puede encontrar puntos diferentes o no correspondientes.

La exactitud de búsqueda del punto mediante el método de correlación usado se ve afectada por la forma del objeto con el que se trabaja produciendo un error en los casos donde este presenta simetría.

Las experimentaciones, cálculos y resultados demuestran que el sistema de visión desarrollado es aplicable a diferentes brazos robóticos y es un camino válido a seguir en la manipulación robótica.

Por último, al sistema de visión desarrollado se necesita incorporarle algoritmos que infirieran la orientación del objeto y la del punto de agarre en base a técnicas de visión por computador, de esta forma el brazo robótico tendrá la información completa, y logrará capturar el objeto autónomamente. La correlación es un proceso en el cual se pueden desarrollar métodos más complejos y trabajar con ventanas más pequeñas o de tamaño variable.

BIBLIOGRAFÍA

[1] Castiello Umberto. **The Neuroscience of Grasping.**

[2] Kragic and Christense. **Robust visual Servoing.**

[3] Justus Piater. **Learning visual features to predict hand orientations.**

[4] Hsiao and Lozano Perez. **Imitation Learning of Whole Body Grasp.**

[5] Jiang Y., Moseson S. and Saxena A. **Efficient Grasping from RGBD Images.**

[6] Ashutosh Saxena, Justin Driemeyer and Andrew Y. Ng. **Learning 3D Object Orientation from Images.**

[7] M. Quigley, S. Batra and S. Gould. **High-Accuracy 3D Sensing for Mobile Manipulation Improving Object Detection and Door Opening.**

[8] A. Saxena, Lawson and L.S. Wong. **Learning Grasp Strategies with Partial Shape Information.**

[9] Kaijen Hsiao, P. Nangeroni and M. Huber. **Reactive Grasping Using Optical Proximity Sensors.**

[10] Eris Chinellato 2008. **Visual Neuroscience of Robotic Grasping.**

[11] Ramakant Nevatia and K. Ramesh Babu. 1980. **Linear feature extraction and description Computer Graphics and Image Processing.**

[12] 610.4-1990. **IEEE Standard Glossary of Image Processing and Pattern Recognition Terminology.**

[13] Laws K. 1980. **Texture Image Segmentation.**

[14] Suzuki M. and Yaginuma Y. 2007. **A Solid Texture Analysis Based on Three Dimensional Convolution.**

[15] http://www.povray.org.

[16] http://ai.stanford.edu/~asaxena/learninggrasp/data.html

[17] Jiménez R. **Desenvolvimiento de Atuadores Tridimensionais Baseados em Músculos Artificiais Dielétricos de Uma ou Múltiplas Camadas.**

[18] Arnau J., Arce C. y Ato M. **Métodos y Técnicas Avanzadas de Análisis de Datos en Ciencias del Comportamiento.**

[19] Ashutosh Saxena, Justin Driemeyer and Andrew Y. Ng. **Robotic Grasping of Novel Objects using Vision.**

[20] Bicchi and Kumar. **Robotic Grasping and contact.**

[21] Demuth Howard, Beale Mark and Hagan M. **Neural Network Toolbox: User's Guide.**

[22] Flórez R. y Fernández J. **Las Redes Neuronales Artificiales.**

[23] Forsyth David A. and Jean Ponce. **Computer vision: A Modern Approach Pentice Hall 2002.**

[24] Hilera R José, Martínez y Víctor J. 2000. **Redes Neuronales Artificiales: Fundamentos, Modelos y Aplicaciones.**

[25] Howlett R. Lakhmi C. Jain. **Radial Basis Function Networks 2: New Advances in Design.**

[26] Shademan A. y Farahmand A. **Robust Jacobian Estimation for Uncalibrated Visual Servoing.**

APÉNDICE

a) Código de programación para extracción de características

```
% Parte I
% Extracción de características para el entrenamiento
close all; clc;

typeData = 'book';
imrows = 48; %tamaño de la filas
imcols = 64; %tamaño de las columnas
gridSize = 10;
nDimNeighbors = 17;
featureVector = zeros(0, 0);
targetVector = zeros(0, 1);

trainFeatureSet = [ ]; %train feature
trainTargetSet = [ ];  %train target
testFeatureSet = [ ];  %test feature
testTargetSet = [ ];   %test target
weightVector = [];

directoryName = '../Stanford Data/book/';

for  pic = 1:250
   if pic < 1000
      filename = 'book';
   end
   if pic < 100
      filename = 'book0';
   end
   if pic < 10
      filename = 'book00';
   end


filename= [filename num2str(pic)];
A = (imread([directoryName filename], 'png'));
```

```
targetImage = double(imread([directoryName 'grasp_' filename ], 'png'));
H = rgb2hsv(A);
objectMask = H(:,:,2) > mean(mean( H(:,:,2) ));

targetImageScaled = zeros(imrows,imcols);
objectMaskScaled = zeros(imrows,imcols);
 for y=1:imrows
  for x=1:imcols
   possibleGrasps = targetImage(10*y-9:y*10, 10*x-9:x*10, 1);
   possibleGrasps = reshape(possibleGrasps, 100, 1);
   possibleGrasps = sort(possibleGrasps);
   bestVal = 0;
   bestNum = 0;
   numProcessed = 0;
   while numProcessed < 100;
    thisVal = possibleGrasps(numProcessed+1);
    thisNum = sum(possibleGrasps == thisVal);
    numProcessed = numProcessed + thisNum;
    if (thisVal ~= 0) && (thisNum > bestNum)
     bestNum = thisNum;
     bestVal = thisVal;
    end
   end
   targetImageScaled(y, x) = bestVal;
   objectMaskScaled(y, x) = mean(mean(objectMask(10*y-9:y*10, 10*x-9:x*10)))
> 0.5;
  end
 end

 gridImage = zeros(imrows,imcols);
 configurationFile(A, gridImage);
 fV = makeJustinFeatureVector(A);
 fV1 = fV;
 fV_old = fV;
 for r = -2:2
  for c= -2:2
   if (r~=0 || c~=0)
    tempFV = [zeros(imrows,max(0,-c),nDimNeighbors),
fV_old(:,max(1,c+1):(end+min(c,0)),1:nDimNeighbors),
zeros(imrows,max(0,c),nDimNeighbors)];
    tempFV2 = [zeros(max(0,-r),imcols,nDimNeighbors);
tempFV(max(1,r+1):(end+min(r,0)),:,1:nDimNeighbors);
zeros(max(0,r),imcols,nDimNeighbors)];
    fV = cat(3, fV, tempFV2);
    end
  end
 end

 fV = cat(3, fV);
 trainFeatureVectorCut = reshape(fV, [imrows*imcols, size(fV,3)]);
```

```
    trainTargetVectorCut = reshape(targetImageScaled, [imrows*imcols 1]);
    objectMaskVector = reshape( objectMaskScaled, [imrows*imcols 1]);

    RelNumNegExamples = 1.1;
    idealNumElements = floor(RelNumNegExamples*sum(trainTargetVectorCut) + 2);
    negInd = find( (trainTargetVectorCut == 0) & (objectMaskVector == 1) );
    negInd_k = find( (trainTargetVectorCut == 0) & (objectMaskVector == 1) );
    posInd = find(trainTargetVectorCut);
    tmpRnd = cumsum(rand(idealNumElements,1));
    randInd = 1 + floor( (size(negInd,1)-1) * tmpRnd/tmpRnd(end) );
    randInd = randInd( find(filter([1 -1], 1, randInd) ~= 0) );

    trainFeatureVectorCut = trainFeatureVectorCut([posInd; negInd(randInd)],:);
    trainTargetVectorCut = trainTargetVectorCut([posInd; negInd(randInd)]);

    featureVector = [featureVector; trainFeatureVectorCut];
    targetVector = [targetVector; trainTargetVectorCut];
     disp(pic)
end
```

b) Código de programación para el entrenamiento de las redes neuronales

```
%inicializando variables
trainFeatureSet = []; %train feature
trainTargetSet = [];  %train target
testFeatureSet = [];  %test feature
testTargetSet = [];   %test target
weightVector = [];

% cargando datos
featureVector = load('featurebookselec_250.mat');%
targetVector = load('targetbookselec_250.mat'); %

featureVector = featureVector.featureVector;
targetVector = targetVector.targetVector;

% featureVector = featureVector(1:40000,:);
% targetVector = targetVector(1:40000,:);

% selección de formas de vecindad
base = [1:9 12:17]; % 2:9,...
% 1 total, 2 ind_cruz_equi, 3 ind_ajed, 4 ind_equi
masc = 3;
if masc == 1
   masca = [1:24];
end
if masc == 2
   masca = [1 3 5 7 8 9 11 12 13 14 16 17 18 20 22 24];
```

```
end
if masc == 3
   masca = [1 3 5 7 9 11 14 16 18 20 22 24];
end
if masc == 4
   masca = [1 5 7 9 16 18 20 24];
end
feat_base = [];
for i=0:2
   feat_base = [feat_base i*17 + base];
end

for i=1:size(masca,2)
   feat_base = [feat_base (masca(i)+2)*17 + base];
end

RelNumNegExamples = 1.1;
idealNumElements = floor(RelNumNegExamples*sum(targetVector) + 2);

negInd = find(targetVector == 0);
posInd = find(targetVector);
tmpRnd = cumsum(rand(idealNumElements,1));
randInd = 1 + floor( (size(negInd, 1) - 1) * tmpRnd / tmpRnd(end) );
randInd = randInd( find(filter([1 -1], 1, randInd) ~= 0) );
indToKeep = [posInd; negInd(randInd)];
featureVector = featureVector(indToKeep, :);
targetVector = targetVector(indToKeep);
trainIndices = find( mod((1:size(featureVector,1)),6) == 1);
testIndices = find( mod((1:size(featureVector,1)),6) == 0);

trainFeatureSet = [trainFeatureSet; featureVector(trainIndices,feat_base)];
trainTargetSet = [trainTargetSet; targetVector(trainIndices,1)]/255;

thisWeightVector = ones(size(trainIndices));
weightVector = [weightVector thisWeightVector];

clear featureVector;
clear targetVector;

P = trainFeatureSet'; % Caracteristicas
T = trainTargetSet'; % Target

clear trainFeatureSet; % Caracteristicas
clear trainTargetSet; % Target

close all

% Entrenamiento del algoritmo
net = newpr(P,T,10,{'tansig','purelin'},'trainbfg'); %

net.trainParam.epochs = 50;
```

```
net.trainParam.goal = 0.01;
net = train(net,P,T);

Y1 = sim(net,P);

% figure, plot(T,'LineWidth',2.5), hold on, plot(Y1,'ro'), grid on
% title('Evaluación del aprendizaje usando Redes de Base Radial')
% xlabel('Indices de las imágenes')
% ylabel('Salida')
% legend('Target','Valores inferidos')
```

c) Código de programación para la inferencia de imágenes

```
close all;clc;
%Captura de la imagen
vid2 = videoinput('winvideo', 1, 'YUY2_640x480');
src = getselectedsource(vid2);
vid2.ReturnedColorspace = 'rgb';
vid2.FramesPerTrigger = 25;
start(vid2);
cami2 = getdata(vid2);
I2 = cami2(:,:,:,end-1); % imagen del objeto

% inicializando variables
imrows = 48; imcols = 64;
gridSize = 10; nDimNeighbors = 17;
featureVector = zeros(0, 0); targetVector = zeros(0, 1);
testFeatureSet = [ ];  %test feature
testTargetSet = [ ];   %test target
weightVector = [];

% cargando los pesos hallados en el entrenamiento
cassso = 2;
nass = 3;
nass = 150;
if cassso == 1
% TASA
[net masca rc] = callweightstasa(nass);
end
if cassso == 2
% LIBRO
[net masca rc] = callweightslibro(nass);
end
if cassso == 3
% PLATO
[net masca rc] = callweightsplato(nass);
end
base = [1:9 12:17];
```

```
feat_base = [];
for i=0:2
   feat_base = [feat_base i*17 + base];
end
for i=1:size(masca,2)
   feat_base = [feat_base (masca(i)+2)*17 + base];
end

 A = I2;
 H = rgb2hsv(A);
 objectMask = H(:,:,2) > 2.3*mean(mean( H(:,:,2) ));

objectMaskScaled = zeros(imrows,imcols);
targetImageScaled = zeros(imrows,imcols);
  for y=1:imrows
   for x=1:imcols
   objectMaskScaled(y, x) = mean(mean(objectMask(10*y-9:y*10, 10*x-
9:x*10)))>0.5;
   end
   end

    gridImage = zeros(imrows,imcols);
    configurationFile(A, gridImage);
    fV = makeJustinFeatureVector(A);
    fV1 = fV;

    fV_old = fV;
    for r = -rc:rc
      for c= -rc:rc
        if (r~=0 || c~=0)
          tempFV = [zeros(imrows,max(0,-c),nDimNeighbors),
fV_old(:,max(1,c+1):(end+min(c,0)),1:nDimNeighbors),
zeros(imrows,max(0,c),nDimNeighbors)];
          tempFV2 = [zeros(max(0,-r),imcols,nDimNeighbors);
tempFV(max(1,r+1):(end+min(r,0)),:,1:nDimNeighbors);
zeros(max(0,r),imcols,nDimNeighbors)];
          fV = cat(3, fV, tempFV2);

       end
      end
    end

    fV = cat(3, fV);
    trainFeatureVectorCut = reshape(fV, [imrows*imcols, size(fV,3)]);
    trainTargetVectorCut = reshape(targetImageScaled, [imrows*imcols 1]);
    objectMaskVector = reshape( objectMaskScaled, [imrows*imcols 1]);

    RelNumNegExamples = 1.1;
    idealNumElements = floor(RelNumNegExamples*sum(trainTargetVectorCut) + 2);
    % Eliminando ruido de la camara
    for i = 1:size(objectMaskVector)
```

```
    if i < 49
       objectMaskVector(i) = 0;
    end
    if i > 3023
       objectMaskVector(i) = 0;
    end
    end

    negInd_k = find( objectMaskVector == 1); % del objeto
    tmpRnd = cumsum(rand(idealNumElements,1));
    randInd = 1 + floor( (size(negInd_k,1)-1) * tmpRnd/tmpRnd(end) );      %ERROR:
some might be repeated
    randInd = randInd( find(filter([1 -1], 1, randInd) ~= 0) ); % 1,2,3,...
    Indices = find( mod((1:size(negInd_k,1)),2) == 1);
trainFeatureVectorCut_k = trainFeatureVectorCut([negInd_k],feat_base);

% inferencia
if nass < 5
    predictions = glmval(net,trainFeatureVectorCut_k, 'logit');

else
    predictions = sim(net,trainFeatureVectorCut_k');
end

% selección
close all;
maxi = max(max(predictions));
for i = 1:max(size(predictions)) % NN ,2 para probal sin nada
    if predictions(i) < 0.99*maxi
       predictions(i) = 0;
    else
       predictions(i) = 1;
    end
end
abc = find(predictions == 1);
IndR = negInd_k(abc);
%  IndR = posInd;
imk = zeros(imrows*imcols,1);
for i = 1:size(imk)
     imk(IndR)= 1;
end
imkk = reshape(imk,[48 64]);

targetImageScal = zeros(480,640);
for i = 1:48
    for j = 1:64
       if imkk(i,j) == 1
          targetImageScal(10*i-9:10*i,10*j-9:10*j) = 1;
       end
    end
end
```

```
BW3 = targetImageScal;
CC = bwconncomp(BW3);
numPixels = cellfun(@numel,CC.PixelIdxList);
[biggest,idx] = max(numPixels);
if biggest > 1
   BW3 = bwareaopen(BW3, biggest-1);
else
   BW3 = BW3;
end
s  = regionprops(BW3, 'centroid');
centro1 = cat(1, s.Centroid);
figure, imageandtarget(A,BW3);

figure, imshow(A);
hold on; plot(centro1(1,1), centro1(1,2),'bx','LineWidth',2,'MarkerSize',25)
% k = 1;
% if k == 1
   u_a = centro1(1,1); v_a = centro1(1,2); % Camara izquierda
% end
% if k == 2
   u_b = centro1(1,1); v_b = centro1(1,2); % Camara derecha
% end
aa = 200; bb = 200; % Ventana 1
% aa = 100; bb = 100; % Ventana 2

if centro1(1,1) > aa/2
   aaa = centro1(1,1)-aa/2; aaaa = 1;
else
   aaa = centro1(1,1)/2; aaaa = 2;
end
if centro1(1,1)+aa/2 > 640
   aaa = ((centro1(1,1)+640)/2)-aa; aaaa = 3;
end

if centro1(1,2) > bb/2
   bbb = centro1(1,2)-bb/2; bbbb = 1;
else
   bbb = centro1(1,2)/2; bbbb = 2;
end
if centro1(1,2)+bb/2 > 480
   bbb = ((centro1(1,2)+480)/2)-bb; bbbb = 3;
end
area = [aaa bbb aa bb];  %primeros son los índices, los segundos son las
dimensiones
sub_I = imcrop(I2,area);
```

d) Código de Calibración de cámaras digitales

```
close all;clc;
% Puntos en las imágenes de los planos XY, YZ y ZX, cámara izquierda
Ipon1A = [351 255;302 281;245 314;182 350;368 310;313 345;251 385;439
338;386 377;325 422;516 368;468 412;409 463];
Ipon2A = [415 277;484 302;558 328;418 204;489 226;567 251;422 126;496
146;577 168;426 43;502 60;587 78];
Ipon3A = [350 184;353 108;355 29;297 208;297 129;297 44;237 237;235 153;232
63;171 270;165 182;158 86];

IponA = [Ipon1A; Ipon2A; Ipon3A];
figure; imshow(I2); hold on;
plot(IponA(:,1),IponA(:,2),'rx','LineWidth',1.5,'MarkerSize',10)

% Puntos en las imágenes de los planos XY, YZ y ZX, cámara derecha
Ipon1B = [268 242;208 266;142 295;66 326;264 299;196 331;119 366;325 333;256
368;178 409;393 370;325 411;247 457];
Ipon2B = [323 269;385 299;452 332;326 196;388 222;459 253;328 115;393
140;466 166;330 31;397 51;473 72];
Ipon3B = [266 171;267 95;267 14;201 192;199 113;198 28;131 217;125 134;120
44;52 246;43 159;34 63];

IponB = [Ipon1B; Ipon2B; Ipon3B];
figure; imshow(I2); hold on;
plot(IponB(:,1),IponB(:,2),'bx','LineWidth',1.5,'MarkerSize',10)

%Generación de los puntos tridimensionales con respecto al sistema de
%calibración XYZ

Rpon1 = [0 0 0];
for i = 1:4
   for j = 1:3
   Rpon1 = [Rpon1; j, i-1,0];
   end
end
Rpon2 = [0 0 0];
for i = 1:4
   for j = 1:3
   Rpon2 = [Rpon2; 0, j, i-1];
   end
end

Rpon3 = [0 0 0];
for i = 1:4
   for j = 1:3
   Rpon3 = [Rpon3; i-1, 0, j];
   end
end
Rpon = [Rpon1; Rpon2(2:end,:); Rpon3(2:end,:)];%*25;

[Ma Ka Ra ta Par_a]=ParamCamera(Rpon,IponA); % parámetros de la cámara
izquierda
```

```
[Mb Kb Rb tb Par_b]=ParamCamera(Rpon,IponB); % parámetros de la cámara
derecha
```

e) Código de reconstrucción tridimensional

```
% Los puntos a triangular, por ejemplo:
% u_a = 326.3522; v_a = 385.6031; % cámara izquierda
% u_b = 265.9397; v_b = 419.6380; % cámara derecha
pa = [0 -1 v_a;1 0 -u_a;-v_a u_a 0]; % formando las matrices
pb = [0 -1 v_b;1 0 -u_b;-v_b u_b 0]; % formando las matrices

MMa = pa*Ma; % matrices obtenidas de la calibración, cámara izquierda
MMb = pb*Mb; % matrices obtenidas de la calibración, cámara derecha
MM = [MMa;MMb];
[vect val]=eig(MM'*MM);
Pi= vect(:,1)'/vect(4,1);

pointss = Pi(1:3)*5; % Coordenadas XYZ
```