REDES NEURONALES APLICADAS AL ANÁLISIS DE IMÁGENES PARA EL DESARROLLO DE UN PROTOTIPO DE UN SISTEMA DE SEGURIDAD

GABRIEL FELIPE JARAMILLO GONZÁLEZ

UNIVERSIDAD TECNOLÓGICA DE PEREIRA
PROGRAMA DE INGENIERÍA EN SISTEMAS
PEREIRA
2009

REDES NEURONALES APLICADAS AL ANÁLISIS DE IMÁGENES PARA EL DESARROLLO DE UN PROTOTIPO DE UN SISTEMA DE SEGURIDAD

GABRIEL FELIPE JARAMILLO GONZÁLEZ

Trabajo de grado presentado como requisito para optar al título de ingeniero en sistemas

Director
Ing. Carlos Augusto Meneses Escobar

UNIVERSIDAD TECNOLÓGICA DE PEREIRA
PROGRAMA DE INGENIERÍA EN SISTEMAS
PEREIRA
2009

Nota de aceptación:

_____
_____
_____
_____
_____
_____

_____
Firma del presidente del jurado

_____
Firma del jurado

_____
Firma del jurado

Pereira, Septiembre 2009

Dedicatoria

A mis padres y mis hermanos por ser el impulso que necesito para cumplir todas mis metas.

A mis queridos amigos Héctor y Marcela por escucharme en los momentos de mayor necesidad y por ayudarme a disfrutar de los pequeños detalles de la vida.

RESUMEN

El presente trabajo muestra el desarrollo de un prototipo de un sistema que permita detectar la presencia de una figura humana a partir de imágenes tomadas en un ambiente controlado tanto en el fondo como en la iluminación.

Este sistema tiene dos partes fundamentales: La primera es el módulo en el que se hace la detección de la silueta del objeto a analizar. Para esta labor es necesario diferenciar el fondo del objeto que se va a analizar por medio de la comparación pixel a pixel que se hace entre dos imágenes (una que contiene simplemente el fondo y la que se desea analizar); posteriormente por medio de un grupo de operaciones morfológicas conocidas como dilatación y erosión, se elimina el ruido que pueda resultar; luego se aplica un algoritmo de detección de bordes para obtener solo el contorno del objeto a analizar. Finalmente, partiendo de esta imagen se crea una cadena de códigos del recorrido de la silueta que luego será utilizada para generar las entradas a la red neuronal.

La segunda parte es la red neuronal que se encarga de ser el detector de patrones, por medio de un entrenamiento previo con imágenes anteriormente cargadas en una base de datos.

Después de que la red neuronal ha sido entrenada y trabaja correctamente, se une con el primer modulo para crear un prototipo que debe detectar la presencia de una figura humana basado en un grupo de imágenes de las cuales dispone el operador.

El sistema cuenta con dos tipos de aplicaciones: una para trabajo individual en un computador y otra basada en la arquitectura cliente/servidor que puede ser usada con varios computadores conectados a una red común.

AGRADECIMIENTOS

Al Ing. Carlos Augusto Meneses por el acompañamiento y la orientación que presto a lo largo del desarrollo de este proyecto.

A los profesores del programa de ingeniería en sistemas por todos los conocimientos que me brindaron a lo largo de la carrera.

OBJETIVOS

OBJETIVO GENERAL

Diseñar e implementar un prototipo de un sistema de seguridad aplicable a zonas cerradas, desarrollando redes neuronales para analizar imágenes e indicar la presencia de formas que puedan ser interpretadas como humanos.

OBJETIVOS ESPECÍFICOS

- Diseñar una red neuronal que tenga como principales cualidades: la facilidad para entrenarse, buen uso de recursos y tiempos de respuesta en concordancia con un sistema de seguridad en tiempo real.

- Analizar las distintas técnicas de procesamiento de imágenes y seleccionar aquellas que faciliten el trabajo de la red neuronal.

- Seleccionar el conjunto de características que describan correctamente el patrón que se desea reconocer.

- Implementar la red neuronal.

- Entrenar la red neuronal con un grupo de imágenes para que cumpla con las metas del proyecto.

- Integrar los elementos para crear un prototipo funcional.

CONTENIDOS

Pág.

LISTA DE FIGURAS

Pág.

LISTA DE TABLAS

Pág.

INTRODUCCIÓN

En la actualidad, nos encontramos en una sociedad que demanda más seguridad tanto es su hogar como en sus entidades bancarias y sitios públicos, pero no está dispuesta a pagar el precio extra que esto genera. A pesar de que existen soluciones a estas inquietudes ellas por lo general sólo están al alcance de un pequeño grupo de personas que cuentan con el dinero para pagarlas.

Por lo general los sistemas de seguridad cuentan con unas cámaras de vigilancias controladas por unos operadores que muchas veces son los mismos guardias de seguridad que deben desempeñar varias tareas al mismo tiempo. De igual manera el proceso de mirar una cámara de seguridad es aburrido y repetitivo; esto genera que el operador deba soportar un carga extra que genera fallas de seguridad y pueden poner en peligro lo que se desea vigilar.

Al mismo tiempo, en distintos sectores se han desarrollado sistemas más complejos, principalmente motivamos por las amenazas que se presentan actualmente en un mundo más globalizado. Como consecuencia el campo de la biométrica ha mejorado considerablemente hasta el punto en el cual ha sido posible identificar las distintas métricas que son comunes en la especie humana.

Una de las áreas que más se ha beneficiado con el desarrollo de la biométrica es la identificación de rostros, lo que no solo ha mejorado la velocidad con la cual es posible identificar un rostro sino que ahora es posible predecir el envejecimiento natural de un rostro. Otra área que ha mejorado constantemente es la identificación de seres humanos ya sea en videos o imágenes hasta el punto que es posible contar la cantidad de personas en el campo de visión de la cámara e identificar diferentes medidas de cada uno de los cuerpos. Pero estos desarrollos no están al alcance de todo el mundo y en algunos casos son sumamente costos limitando su uso para gobiernos y algunos sectores privados.

Por esta razón es que se piensa en darle una solución simple y económica a este problema por medio de la creación de un sistema autónomo que se encargue del análisis de la imagen. Este tipo de sistemas ya se ha probado con otras actividades como el análisis de imágenes diagnosticas para la detección de enfermedades, de sistemas de control para matriculas de vehículos y para la compresión de la letra escrita a mano.

En este tipo de enfoque no se elimina al operario sino que simplemente se le ofrece una herramienta que le permite dedicarse a las otras labores que deba desempeñar mientras el trabajo repetitivo queda en manos de un sistema que no sufre de problemas inherentes a la condición humana como el cansancio, la pereza o la falta de atención.

Para crear este tipo de sistemas es necesario tener en cuenta tres partes, la adquisición de la imagen, el procesamiento de la imagen y la clasificación de estas. Es necesario que todas estas partes sean afinadas correctamente ya que una falla en cualquiera de ellas generaría una falla total en el sistema.

Se espera de tal manera crear un sistema que cumpla con las condiciones que se deben tener para detectar una figura humana dentro de un grupo de imágenes y que lo haga con una carga computacional baja y sin la necesidad de un hardware muy especializado.

1. INTRODUCCIÓN AL RECONOCIMIENTO DE PATRONES EN IMÁGENES

En la actualidad, con el interés de poner en manos de la computación procesos que antes eran controlados por seres humanos se ha llegado a un punto en el cual existen procesos que no pueden ser fácilmente implementados usando técnicas computacionales, tal como lo es el reconocimiento de patrones que a pesar de que nos resulta tan fácil a los seres humanos, no es tan fácil de solucionar de forma normal.

Cuando se habla de reconocimiento de patrones, es necesario explicar que un patrón es un conjunto de reglas que pueden ser usadas para identificar un objeto en particular. Luego, reconocer un patrón es la capacidad de identificar objetos por sus similitudes a pesar de que estos no sean totalmente iguales al modelo base.

Dado que para los seres vivos el reconocimiento de patrones es una tarea relativamente sencilla, no es similar entre distintas criaturas, los seres humanos tienden a reconocer con facilidad los objetos utilizando la vista como sentido principal, mientras los perros hacen mayor énfasis en el reconocimiento por medio del olfato.

Utilizando técnicas normales de computación es difícil encontrar respuestas satisfactorias en tiempos razonables y es en este punto donde algunas técnicas de inteligencia artificial han mostrado tener una de sus más importantes aplicaciones. Aplicaciones como reconocimiento de escritura, clasificaciones de objetos en imágenes, reconocimiento de voz, etc. [FaK99].

1.1 Reconocimiento de patrones en imágenes.

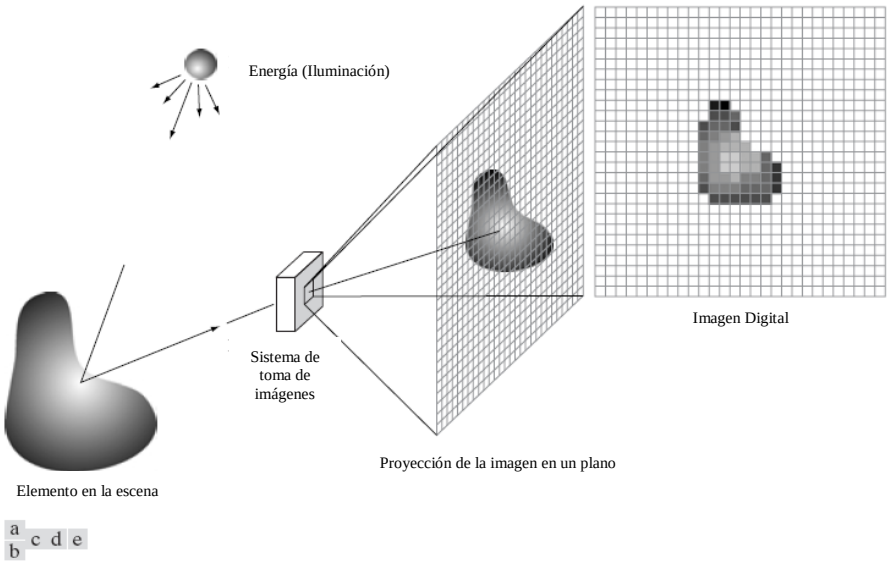
El reconocimiento de patrones en imágenes ha sido de gran interés para la comunidad científica, ya que emular el principal sentido que poseen los seres humanos es de gran importancia para poder automatizar ciertas tareas que actualmente necesitan de la visión y que son altamente repetitivas como la verificación de firmas, reconocimiento de caras, reconocimiento de objetos, etc. [VaB08].

Normalmente el reconocimiento de patrones en imágenes suele presentar las siguientes etapas: (Captura de la imagen, Pre-Procesamiento de la imagen, Extracción de características, clasificación y decisión).

1.1.1 Captura de Imágenes.

El proceso por el cual se logra una imagen digital, empieza con sensores que leen

la intensidad de la luz usando para eso diferentes filtros de color y luego dispositivos de memoria digital que se encargan de guardar la imagen digital, ya sea en formato RGB o como simple información para su posterior análisis.



(a)Energía (iluminación) (b)Elemento en la escena (c)Sistema de toma de imágenes (d)Proyección de la imagen en un plano (e)Imagen digital

Figura 1.1 Un ejemplo del proceso de adquisición de una imagen digital.

Para poder crear una imagen digital, es necesario que un sensor capture la imagen, los principales tipos de sensores son el CCD (siglas en inglés de *charge-coupled device:* 'dispositivo de cargas [eléctricas] interconectadas') el cual es uno de los elementos principales de las cámaras digitales; este sensor está compuesto de diminutas células fotoeléctricas que registran la imagen. Luego la cámara procesa la imagen y la guarda en el dispositivo de almacenamiento.

También está el APS (**Active Pixel Sensor**) que detecta la luz basado en la tecnología CMOS.

1.1.2 Pre-procesamiento.

A pesar de que para cualquier criatura es fácil determinar patrones a partir de la imagen que ven, a nivel computacional, es necesario realizar un procesamiento de la imagen que se va a utilizar para eliminar de ella todo aquello que pueda generar una falla a la hora de tomar una decisión y esto se debe a que en el momento de capturar la imagen el resultado puede sufrir ruido, perdidas de calidad, desenfoque, etc.

El procesamiento de bajo nivel, también conocido como ***pre-procesamiento*** es utilizado primordialmente para reducir ruido, mejorar el contraste, etc. Este tipo de

funciones se pueden considerar como reacciones automáticas y que no requieren inteligencia por parte del sistema de análisis de imágenes; este procesamiento se puede comparar con el proceso de percepción y adaptación que sufre una persona cuando trata de encontrar una asiento en medio de la oscuridad de un teatro, tras haber estado expuesta durante cierto periodo de tiempo a la claridad del sol. Este tipo de procesamiento se caracteriza por el hecho que tanto la entrada como la salida son imágenes [GaW02].

Figura 1.2 Mejora del contraste de una imagen.

1.1.3 Segmentación.

El procedimiento de ***segmentación*** particiona una imagen en los objetos o las partes que la constituyen. En general, la segmentación autónoma es una de las tareas más difíciles en el procesamiento digital de imágenes. Un procedimiento de segmentación desigual genera que el proceso tome un largo tiempo hacia la solución acertada de los problemas de imagen que requieren que los objetos sean identificados individualmente.

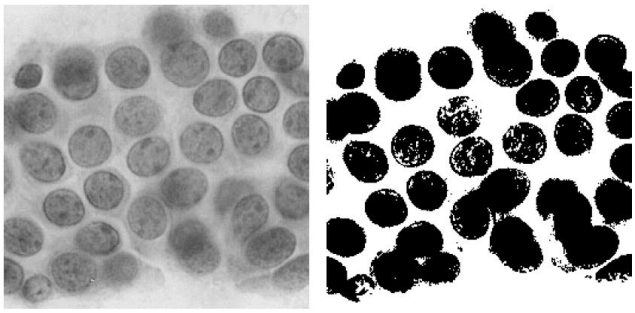


Figura 1.3 Separación de los objetos de interés del fondo.

Por el otro lado, un algoritmo de segmentación débil y errática casi siempre genera una eventual falla. En general, entre más acertada sea la segmentación, existe una mayor posibilidad de que el reconocimiento sea correcto [GaW02]

1.1.4 Extracción de características.

En el reconocimiento de patrones la ***extracción*** de características se ha vuelto sin duda la parte principal ya que al seleccionar las características que mejor representen el problema, se puede lograr que la respuesta no varié al existir cambios de luz, de punto de vista, color, etc.

Por lo general una imagen se puede representar en función de los bordes de la imagen utilizando ya sea una cadena de códigos o con una aproximación poligonal. También es posible representar la imagen en función de sus características internas.

1.1.5 Clasificación.

La meta final en el reconocimiento de patrones es la ***clasificación*** de un patrón. Utilizando la información original con la que se cuenta, se busca identificar los rasgos relevantes y luego extraerlos para poder medirlos. Estas medidas deben pasar a un clasificador el cual se debe encargar de la clasificación, al determinar a cual de las clases existentes este patrón pertenece [FaK99].

Existen distintas técnicas de clasificación como son las siguientes:

- Funciones de decisión.
- Clasificadores de la distancia mínima.
- Clasificadores estadísticos.
- Redes neuronales artificiales (ANN).
- Maquinas de soporte vectorial (SVM).

Como cualquier otro problema de clasificación, se debe tener un conjunto de entrenamiento el cual provee información necesaria que asocie un futuro dato de

entrada con la decisión que debe ser tomada.

1.1.6 Decisión.

A partir de un buen trabajo en el proceso de clasificación, una nueva entrada que sea ingresada al clasificador debe estar en posición de dar una respuesta que satisfaga la necesidad aunque la nueva entrada no sea igual a los ejemplos utilizados en la parte del entrenamiento.

Es en esta parte donde se verifica si todo lo anterior ha trabajado de manera correcta o si es necesario hacer un cambio para lograr resultados más satisfactorios. Es posible que después de llegar a esta parte, sea necesario al proceso de clasificación si la respuesta que se busca no está apareciendo correctamente.

También es importante en esta parte, que los resultados presentados sean fácilmente entendibles para cualquier persona que se encargue del sistema, es posible que a pesar de que la respuesta sea satisfactoria, no sea entendible para un usuario normal. En este punto será necesario descubrir la mejor forma de interpretar los datos.

2. FUNDAMENTO TEORICO

2.1 Imágenes digitales.

En términos informáticos, una ***imagen*** es un caso particular de *señal*, más exactamente, una función que especifica una determinada distribución de intensidades lumínicas. Dicho de otro modo, una imagen es entendida, desde este punto de vista, como la serie de valores atribuidos a una función bidimensional que asigna a todos los puntos de un segmento de un plano un valor visual determinado [Mon99].

En el caso de una imagen monocromática este valor vendría dado por una función simple de dos variables *f(x,y)*, en donde *x,y* denota coordenadas espaciales y f un valor en cada punto que es proporcional a la intensidad de iluminación en ese punto o "nivel de gris", en el caso de una imagen acromática. Las coordenadas x, y están referidas a un ámbito espacial determinado, por ejemplo, por un extremo inferior (x_0, y_0) y un extremo superior (x_{max}, y_{max}). En el caso de una imagen cromática esta valor vendría dado por tres funciones simples de dos variables, *fr(x,y), fg(x,y), fb(x,y),* que expresarían la intensidad de iluminación de un punto *x, y,* en el mismo ámbito, y para los tres componentes cromáticos primarios rojo (R), verde (G) y azul (B). [Mon99].

2.1.1 Relaciones básicas de píxeles.

Cada coordenadas *x, y* contiene un píxel y es importante notar que existen algunas pequeñas pero importantes relaciones entre cada una de ellas.

Vecinos de un píxel: Cada píxel *p* de coordenada *(x, y)* tiene cuatro vecinos horizontales y verticales cuyas coordenadas son

$$(x+1,y),\ (x-1,y),\ (x,y+1),\ (x,y-1)$$

este conjunto de píxeles, que reciben el nombre de ***4-vecinos de p*** se nota $N_4(p)$. Cada píxel esta a distancia unitaria de *(x, y)*.

Los vecinos diagonales de *p* tienen coordenadas

$$(x+1,y+1),\ (x+1,y-1),\ (x-1,y+1),\ (x-1,y-1)$$

y se notan $N_D(p)$. Estos puntos, junto con los cuatro vecinos, se llaman ***8-vecinos de p*** y se notan $N_8(p)$.

Conectividad: La ***conexión*** entre píxeles es un concepto importante usado para

establecer las fronteras de objetos y las regiones componentes de una imagen. Para establecer si dos píxeles están conectados se ha de establecer si son adyacentes en algún sentido (por ejemplo si son 4-vecinos y si sus niveles de gris cumplen algún criterio de similaridad, por ejemplo ser iguales). Así o una imagen binaria con valores 0 y 1 dos píxeles pueden ser 4-vecinos y no estar conectados salvo que tengan el mismo valor [Mol98].

Sea V el conjunto de valores de niveles de gris que se usa para definir la conectividad: por ejemplo en las imágenes binarias, $V= \{1\}$ o un rango como $V= \{33, 34,\ldots,50\}$ en las imágenes de niveles de gris.

Consideremos tres tipos de conectividad:

1. ***4 – Conectividad***. Dos píxeles *p* y *q* con valores en *V* se dicen 4 – conectados si q pertenecen $N_4(p)$.

2. ***8 – Conectividad***. Dos píxeles *p* y *q* con valores en *V* se dicen 8 – conectados si q pertenecen $N_D(p)$.

3. ***m – conectividad***. Dos píxeles *p* y *q* con valores en *V* se dicen m – conectados si

   a. *q* pertenece a $N_4(p)$ o
   b. *q* pertenece a $N_D(p)$ y $N_4(p) \cap N_D(p)$ es vacio. (Este es el conjunto 4-vecinos de *p* y *q* con valores en *V*).

   Es importante notar que la m-conectividad se introduce para eliminar la ambigüedad en los posibles caminos que unen dos píxeles.

Se dice que un píxel p es adyacente a q si están conectados. Es obvio que este concepto depende del tipo de conectividad que se use. Dos subconjuntos de una imagen *S,* S_1 y S_2 diremos que son adyacentes si algún píxel es S_1 es adyacente a alguno en S_2.

Un camino del píxel *p* con coordenadas *(x, y)* a *q* con coordenadas *(s, t)* es una sucesión distinta de píxeles con coordenadas

$$(x_0,y_0), (x_1,y_1),\ldots., (x_n,y_n)$$

Donde $(x_0,y_0) =(x,y)$ y $(x_n,y_n) = (s,t)$, siendo (x_i,y_i) adyacente a (x_{i-1},y_{i-1}) para $1 \leq i \leq n$, siendo entonces *n* la longitud del camino. Es obvio que el tipo de camino depende del tipo de adyacencia utilizado.

Si *p* y *q* son píxeles en un subconjunto *S* de una imagen, entonces *p* está conectado a *q* en *S* si existe un camino de *p* a *q* que esté conectado en *S*.

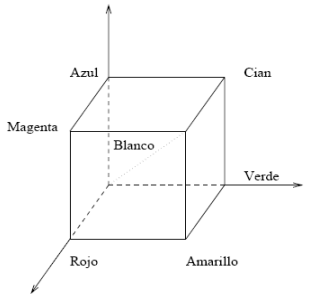
2.1.2 Tipos de imágenes.

Imágenes a color

Son imágenes compuestas por cuatro valores almacenados en tres planos de píxeles o matrices, que se encuentran definidos por ***modelos de color***. El propósito de un modelo de color es una especificación de un modelo de coordenadas 3-D y un sub-espacio dentro de ese sistema donde cada color se representa con un punto único. Los modelos más usados en el procesamiento de imágenes son RGB, YIQ y HSI.

En el modelo RGB cada color aparece en sus componentes primarias espectrales rojo, verde y azul. El modelo está basado en un sistema de coordenadas cartesiano. El sub-espacio de interés es el cubo que se muestra en la figura 2.1.

Las imágenes en el modelo de color RGB están formadas por tres planos de imágenes independientes, cada uno de los colores primarios. Cuando son introducidas en un monitor RGB, las tres imágenes se combinan en la pantalla de fosforo para producir una imagen de color compuesta. Por tanto, el uso del modelo RGB para procesamiento de imágenes tiene sentido cuando las imágenes viene expresadas en términos de los tres planos de colores. Alternativamente, la mayoría de las cámaras en color que se usan para adquirir imágenes digitales utilizan el formato RGB [Mol98] [GaW03].



Los puntos en la diagonal principal tienen niveles de gris desde el negro en el origen al blanco en el punto (1,1,1)

Figura 2.1 Cubo de color RGB.

El modelo YIQ se usa en las emisiones de TV en color. Básicamente, YIQ es una re-codificación de RGB para la eficiencia en la transmisión y para mantener la compatibilidad con los estándares de TV monocromo. De hecho, la componente Y del sistema YIQ proporciona toda la información de video que se requiere para una televisión monocromo. La conversión RGB a YIQ viene dada por

$$\begin{pmatrix} Y \\ I \\ Q \end{pmatrix} = \begin{pmatrix} 0{,}299 & 0{,}587 & 0{,}114 \\ 0{,}596 & -0{,}275 & -0{,}321 \\ 0{,}212 & -0{,}523 & 0{,}311 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix}$$

Este sistema fue diseñado teniendo en cuenta las características del sistema visual humano, en particular la mayor sensibilidad a los cambios de luminancia que a los cambios de matiz o saturación. Así pues, este estándar utiliza más bits (o ancho de banda) para representar Y y menos para I y Q. Además de ser ampliamente usado, una de sus ventajas más importantes es que la informaciones de luminancia (Y) y color (I y Q) están separadas. De esta forma se puede procesar la luminancia sin afectar el color [Mol98].

El matiz es un atributo que describe la pureza del color (puro amarillo, naranja o rojo), mientras que la saturación da la medida del grado en el que un color puro se diluye con luz blanca. La utilidad del modelo HSI se debe a dos hechos principales. Por una parte, la componente de intensidad se separa de la información de color y además las otras dos componentes están muy relacionadas con la forma en que el ser humano percibe el color. Estas características hacen este modelo muy apropiado para el desarrollo de algoritmos de procesamiento de imágenes basados en propiedades del sistema de visión humano [Mol98].

Figura 2.2 Imagen RGB.

Imágenes en escala de grises

Son imágenes compuestas de un simple plano o matriz de píxeles, en donde sus valores están representados entre [0-255] para una imagen de 8-bits, en donde el 0 representa el color negro y el 255 el color blanco, dado que en ese intervalo se ubican todas las intensidades de grises disponibles en la imagen [VyC07].

Figura 2.3 Imagen en escala de grises.

Imágenes binarias

Son aquellas donde cada píxel asume valores discretos de unos o ceros. Esencialmente estos valores corresponden a encendido o apagado; en donde una imagen binaria es almacenada con un orden lógico de ceros o píxeles apagados y unos o píxeles prendidos. Estos se encuentran en la frontera entre dos regiones diferentes, siendo su detección, un paso importante para la recuperación de información [VyC07].

Figura 2.4 Imagen binaria

2.2 Pre-procesamiento.

2.2.1 Detección de bordes.

Los primeros pasos de la visión tratan de identificar rasgos en la imagen que son relevantes para estimar la estructura y propiedades de los objetos en la escena. Los ***bordes*** corresponden a cambios locales significativos en la imagen y son, probablemente, los rasgos más importantes para el análisis de la misma.

Simplificando la definición se puede decir que un borde es una discontinuidad de algún tipo en la función de intensidad de la imagen. Pero, existen muchos tipos de bordes diferentes, algunos son debidos a sombras que se producen sobre los objetos, otros a la variación de la reflactancia de los objetos o incluso por la textura de los objetos. Como se ve en la figura 2.5 los bordes etiquetados *d* son debido a discontinuidades entre los objetos y la cámara, junto con discontinuidades en la normal a la superficie del objeto. Los etiquetados *dc* son debidos a discontinuidades de la distancia a la cámara pero no a la normal. Los etiquetados *n* son debido a discontinuidad en la normal, mientras los etiquetados *r* se deben a cambios en la reflactancia de los objetos, pero sin cambios en las propiedades geométricas. Por último los etiquetados *s* se deben a la sombra que el cilindro produce en la mesa [Mol98].

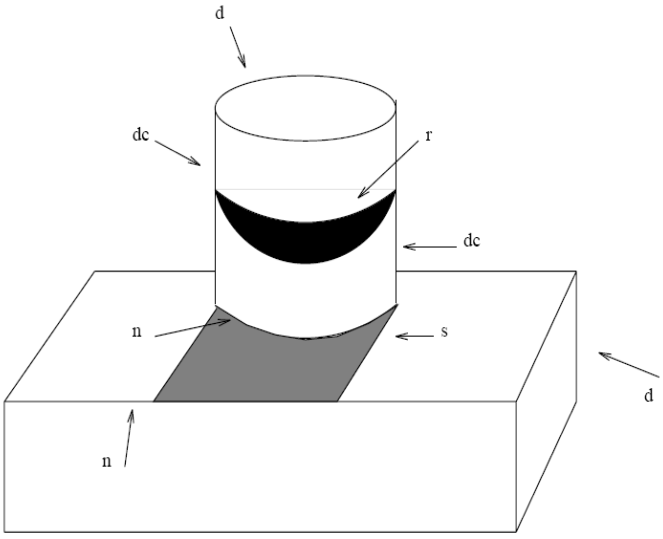


Figura 2.5 Discontinuidades en imágenes.

Tales discontinuidades son detectadas usando derivadas del primer y segundo orden. La elección en derivada del primer orden en el procesamiento de imágenes es el gradiente. El gradiente de una función en 2-D, *f(x, y)*, es definida como el vector

$$\nabla F = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \dfrac{\partial f}{\partial x} \\ \dfrac{\partial f}{\partial y} \end{bmatrix}$$

La magnitud de este vector es

$$\nabla^2 f = mag(\nabla F) = \left[G_x^2 + G_y^2\right]^{-1/2} = \left[\left(\partial f/_{\partial x}\right)^2 + \left(\partial f/_{\partial y}\right)^2\right]^{-1/2}$$

Para simplificar a nivel computacional, esta cantidad es aproximada algunas veces omitiendo la raíz cuadrada,

$$\nabla f \approx G_x^2 + G_y^2$$

O usando valores absolutos,

$$\nabla f = |G_x| + |G_y|$$

Estas aproximaciones aun se comportan como derivadas; por esta razón, estas

aproximaciones son cero en áreas de intensidad constante y sus valores son proporcionales al grado del cambio de intensidad en áreas en las cuales el valor de píxel sea variable. Existe una buena cantidad de detectores de fronteras [Mol98].

El operador de Roberts es uno de los más antiguos. Es fácil de calcular ya que usa sólo una ventana de tamaño 2x2. Sus mascaras de convolución son

$$G_x = \begin{array}{|c|c|}\hline 1 & 0 \\ \hline 0 & -1 \\ \hline \end{array} \qquad G_y = \begin{array}{|c|c|}\hline 0 & 1 \\ \hline -1 & 0 \\ \hline \end{array}$$

La magnitud del gradiente se calcula mediante la siguiente aproximación

$$G\big(f(i,j)\big) = |f(i,j) - f(i+1,j+1)| + |f(i,j+1) - f(i+1,j)|$$

Puesto que G_x y G_y son aproximaciones de las derivadas parciales con respecto a *x* e *y* respectivamente. Una desventaja del operador de Roberts es que es muy sensible al ruido, ya que se usan muy pocos píxeles para aproximar el gradiente.

Una forma de evitar que el gradiente se calcule en un punto intermedio, tal y como ocurre en el operador de Roberts, es usar entornos de tamaño 3x3. Considérese la distribución de píxeles que se muestra alrededor de *(i, j)* en la figura 2.6.

El ***operador de Sobel*** es la magnitud del gradiente calculado mediante

$$M = \sqrt{S_x^2 + S_y^2}$$

Donde las derivadas parciales se calculan mediante

$$S_x = (a_2 + ca_3 + a_4) - (a_0 + ca_7 + a_6)$$
$$S_y = (a_0 + ca_1 + a_2) - (a_6 + ca_5 + a_4)$$

Con una constante *c* = 2.

Es claro que estos operadores de gradiente pueden implementase utilizando mascaras de convolución, que es una forma que se mostrar la función con base a los a la conectividad de los pixeles. Es importante notar que este operador le da más peso a los píxeles más cercanos al centro de la máscara. Este operador es uno de los más usados.

a_0	a_1	a_2
a_7	(i, j)	a_3
a_6	a_5	a_4

Figura 2.6 Etiquetado de nivel de gris para explicar los detectores de borde.

Derivadas del segundo orden en el procesamiento de imágenes son generalmente computadas usando el laplaciano. El laplaciano de una función en 2-D *f(x, y)* es formada por las derivadas de segundo orden, como se muestra a continuación:

$$\nabla^2 f = {\partial^2 f}/{\partial x^2} + {\partial^2 f}/{\partial y^2}$$

La segunda derivada en las direcciones *x* e *y* se aproximan utilizando las ecuaciones en diferencias

$$\frac{\partial^2 f}{\partial x^2} = \frac{G_x}{\partial_x} = \frac{\partial(f(i+1,j) - f(i,j))}{\partial_x} = f(i+2,j) - 2f(i+1,j) + f(i,j)$$

Análogamente se obtendría

$$\frac{\partial^2 f}{\partial y^2} = f(i,j+2) - 2f(i,j+1) + f(i,j)$$

Sin embargo, estas dos aproximaciones no están centradas en la posición *(i, j)* de modo que la aproximación más razonable seria

$$\frac{\partial^2 f}{\partial x^2} = f(i+1,j) - 2f(i,j) + f(i-1,j)$$
$$\frac{\partial^2 f}{\partial y^2} = f(i,j+1) - 2f(i,j) + f(i,j-1)$$

Combinando estas dos ecuaciones en un único operador, se puede usar la siguiente mascara para aproximar al laplaciano

$$\nabla^2 \approx \begin{array}{|c|c|c|}\hline 0 & 1 & 0 \\ \hline 1 & -4 & 1 \\ \hline 0 & 1 & 0 \\ \hline \end{array}$$

Algunas veces es deseable darle más peso a los píxeles centrales del entorno.

Una aproximación del laplaciano que hace esto es

$$\nabla^2 \approx \begin{array}{|c|c|c|}\hline 1 & 4 & 1 \\ \hline 4 & -20 & 4 \\ \hline 1 & 4 & 1 \\ \hline \end{array}$$

El operador laplaciano marca la presencia de un borde cuando la salida del operador realiza una transición por cero. Los ceros triviales, es decir, las regiones de cero uniformes tienen que ser ignoradas. En principio, las localizaciones de los cruces por cero pueden ser estimadas a nivel de sub-píxeles usando interpolación lineal; pero los resultados pueden no ser exactos.

2.2.2 Segmentación de la imagen.

El primer paso en el análisis de imágenes es generalmente ***segmentar*** la imagen en regiones. *Una región, en una imagen, es un grupo de píxeles conectados (un concepto que necesita ser definido precisamente) que tienen propiedades similares*. Es obvio que las regiones son importantes para la interpretación de las imágenes pues pueden corresponder a objeto en la escena. Una imagen puede contener varios objetos y además cada objeto puede contener varias regiones que corresponden a partes del mismo [Mol98] [GaW03].

Para que una imagen pueda ser interpretada correctamente, tiene que ser dividida en regiones que correspondan a objetos o partes de ellos. Sin embargo, debido a los errores de segmentación, la correspondencia entre regiones y objetos no será perfecta y deberemos utilizar conocimiento específico en etapas posteriores de la interpretación de las imágenes.

Regiones y Bordes

El primer paso en el análisis para entender una imagen es particionar la imagen de forma que las regiones que representan diferentes objetos son marcadas explícitamente. Estas particiones puede ser obtenidas a partir de las características de los niveles de gris en la imagen, o utilizando alguna otra. Así pues, podría decirse que los valores de gris en las posiciones de la imagen son observaciones y que los otros atributos, como la pertenencia a regiones, deben obtenerse a partir de los niveles de gris. Existen dos aproximaciones para particionar una imagen en regiones: *segmentación basada en regiones y estimación de la frontera utilizando detectores de bordes* [Mol98].

En la aproximación basada en regiones todos los píxeles que corresponden a un objeto se agrupan juntos y son marcados para indicar que pertenecen a una región. Este proceso recibe el nombre de *segmentación*. Los píxeles son

asignados a regiones según algún criterio que los distingue del resto de la imagen. Dos principios muy importantes en la segmentación son valor de similitud y proximidad espacial. Dos píxeles pueden ser asignados a la misma región si tienen características similares de intensidad y si están próximas. La varianza de los niveles de gris en una región y la compactificación de un región pueden ser también usadas como valores de similitud y proximidad entre píxeles respectivamente [Mol98].

Al igual que en el problema de la segmentación por similitud entre los niveles, es posible también realizar este proceso mediante lo que se denomina *disparidad*. El objetivo es encontrar los píxeles que se encuentran en las fronteras de las regiones. Estos píxeles, llamados bordes, pueden localizarse examinando los píxeles vecinos. Puesto que los píxeles llamados bordes están en la frontera de las regiones y estas suelen tener diferentes niveles de gris a cada lado, lo que se necesita es medir la diferencia entre los píxeles vecinos. La mayoría de detectores de bordes usan solo las características de la intensidad para su detección, aunque algunos aspectos como la textura y el movimiento también aportan información.

Es necesario definir que se entiende por segmentación. Una segmentación completa de una imagen R es un conjunto finito de regiones $R_1,\ldots,R_s$ tales que

$$R = \cup_{I=1}^{S} R_I \qquad\qquad R_I \cap R_J = 0 \quad i \neq j$$

Segmentación basada en umbralización

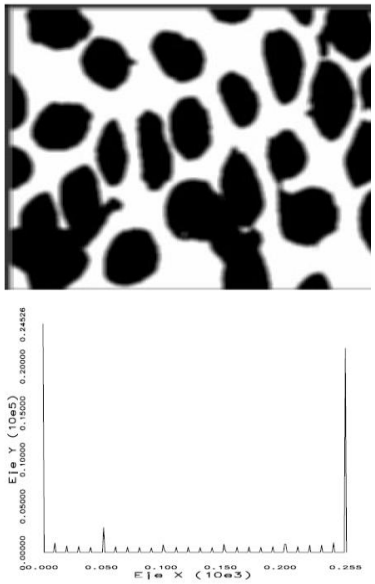


Figura 2.7 Histograma de niveles de gris.

Considérese el ***histograma*** de niveles de gris mostrado en la figura 2.7

correspondiente a una imagen *f(x, y)* compuesta de objetos oscuros sobre fondo luminoso, de forma que los píxeles en los objetos y fondo tienen niveles de gris agrupados en dos modas dominantes. Una forma obvia de extraer los objetos del fondo es seleccionar un umbral *T* que separe estas modas. Es decir, escribir

$$g(i,j) = \begin{cases} 1 & \text{si} \quad f(i,j) \geq T \\ 0 & \text{si} \quad f(i,j) < T \end{cases}$$

Esta ecuación tiene muchas modificaciones. Una posibilidad es segmentar la imagen en regiones de píxeles con nivel de gris en un conjunto *D* y el resto como fondo. También se puede utilizar umbrales múltiples, lo que produce imágenes no binarias con un conjunto muy limitado de niveles de gris, el proceso sería

$$\begin{aligned} g(i,j) &= 1 \quad \text{si } f(i,j) \in D_1 \\ &= 2 \quad \text{si } f(i,j) \in D_2 \\ &= 3 \quad \text{si } f(i,j) \in D_3 \\ &\ldots \\ &= n \quad \text{si } f(i,j) \in D_n \\ &= 0 \quad \text{en el resto} \end{aligned}$$

Este tipo de umbralización multinivel es generalmente menos fiable que la umbralización en dos regiones. La razón está en que es difícil establecer múltiples umbrales que aíslen regiones correctamente, especialmente cuando hay muchas modas (máximos locales).

Es claro que para hacer la segmentación más robusta, el umbral debería ser seleccionado automáticamente por el sistema, para esta labor, tanto el conocimiento sobre la escena como el del problema a resolver y cualquier otro conocimiento debería ser usado para fijar el umbral. También se puede disponer de la información proporcionada por las características de los objetos, el tamaño de los mismos, la fracción de la imagen ocupada por los objetos y los diferentes tipos de objetos que pueden aparecer en la imagen [Mol98].

Segmentación basada en bordes

Históricamente el primer grupo de métodos de segmentación, que es todavía hoy muy importante es el basado en información sobre ***bordes*** de la imagen. Las técnicas de segmentación basada en bordes utilizan los bordes encontrados por

los detectores. El objetivo final es alcanzar al menos una segmentación parcial, es decir, agrupar bordes locales en una imagen donde sólo cadenas de bordes con una correspondencia con objetos en la imagen o parte de la imagen están presentes.

Existen diferentes métodos para la segmentación basada en bordes que difieren en la estrategia para la construcción de frontera, así como en la cantidad de información previa que se incorpora en el método. Es obvio, que a mayor información mejor segmentación.

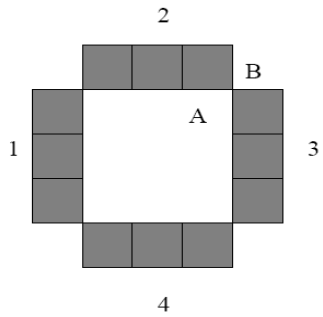


Figura 2.8 Problemas con la conectividad.

Las fronteras son bordes unidos que caracterizan la forma de un objeto. Son, por tanto útiles para calcular rasgos geométricos como tamaño y orientación. Conceptualmente, las fronteras pueden encontrarse trazando los bordes conectados (bien sea 4 u 8 conectados). Sin embargo, hay dificultades asociadas con estas definiciones de conectividad, como muestra la figura 2.8. Si utilizamos la 4-conectividad los segmentos 1, 2, 3 y 4 serían clasificados como disjuntos, aunque claramente parecen formar un anillo. Bajo la 8-conectividad estos segmentos estarían conectados, pero también lo estarían el interior con el exterior del anillo. En principio, esto podría resolverse utilizando tipos de conectividad distinta para los objetos y para el fondo.

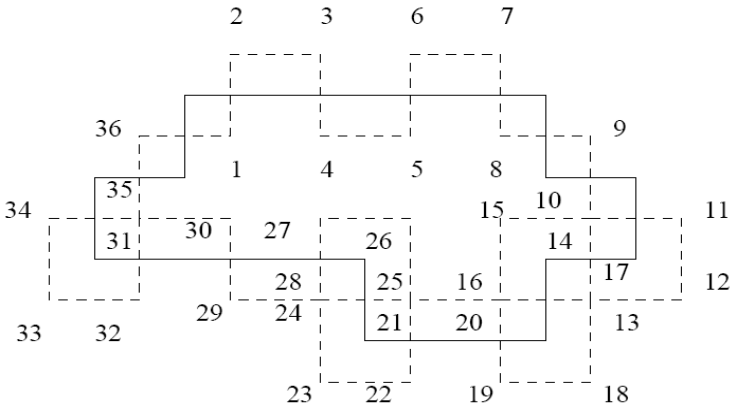


Figura 2.9 Seguimiento del contorno.

Por otro lado, los algoritmos de seguimiento de contorno trazan las fronteras ordenando los puntos de tipo borde sucesivo. Un algoritmo para trazar fronteras cerradas en imágenes binarias es el siguiente:

1. Comenzar dentro de la región A, (por ejemplo, el primer píxel que encontramos de la región cuando hacemos un rastreo por filas),
2. Girar a la izquierda y pasar al píxel siguiente si estamos dentro de la región A, en caso contrario girar a la derecha y pasar el píxel siguiente,
3. Continuar hasta que se llegue al punto del que se partió

La figura 2.9 muestra un ejemplo de cómo se sigue un contorno. En la imagen se ve como al recorrer los bordes se debe tener en cuenta los puntos por donde se entra y se sale de la región.

Segmentación orientada a regiones

El método más natural para el crecimiento de las ***regiones*** es comenzar considerando cada píxel de la imagen como una región, obviamente en este caso cada región es homogénea, pero no necesariamente cumple el criterio de ser las regiones máximas, el proceso deberá de repetirse hasta que las regiones sean máximas. En algoritmo, el método sería:

1. Definir una segmentación inicial que cumpla el criterio de homogeneidad,
2. Definir un criterio para unir regiones adyacentes,
3. Unir las regiones adyacentes si cumplen el criterio de unión. Parar cuando no puedan unirse dos regiones sin romper el criterio de homogeneidad.

0	0	5	6	7
1	1	5	8	7
0	1	5	6	7
2	0	7	6	6
0	1	5	6	5

(a)

a	a	b	b	b
a	a	b	b	b
a	a	b	b	b
a	a	b	b	b
a	a	b	b	b

(b)

a	a	a	a	a
a	a	a	a	a
a	a	a	a	a
a	a	a	a	a
a	a	a	a	a

(c)

Figura 2.10 Ejemplo de crecimiento de regiones a partir de puntos semilla.

La implementación más simple de este algoritmo empieza por la unión de regiones comenzando la segmentación con regiones 2x2, 4x4 u 8x8. Las descripciones de las regiones se basan entonces en propiedades estadísticas de los niveles de gris, como por ejemplo en el histograma, la descripción de una región se compara con

la de otra adyacente, si coinciden se unen las regiones y se computa el nuevo descriptor de la región. En caso contrario las regiones se marcan como que no se pueden unir [Mol98]. El proceso continua.

En general podría decirse que lo más importante es seleccionar el criterio para realizar la unión ya que de esta forma se logra obtener una mejor segmentación con la cual se trabajará. Un criterios de tipo heurístico utilizado con el fin de brindar una buena unión de pixeles de una región es

1. Unir dos regiones R_i y R_j si $w/P_m > \theta_1$ donde $P_m = min\ (P_i, P_j)$ P_i y P_j son los perímetros de R_i y R_j y W es el número de localizaciones de fronteras débiles, (píxeles en los cuales la diferencia es menor a un cierto umbral).

El crecimiento de regiones es un procedimiento que agrupa píxeles o subregiones en regiones mayores. La forma más sencilla de este proceso es la llamada agregación de píxeles que comienza con un conjunto de píxeles semilla y a partir de ellos hace crecer la región añadiendo a dichos píxeles semilla aquellos vecinos que tienen propiedades similares (nivel de gris, color, textura).

Para ilustrar ese procedimiento se puede considerar la figura 2.10, donde los números dentro de las celdillas representan niveles de gris. Sean los puntos coordenadas (3,2) y (3,4) ubicados en la matriz (a) y subrayados en la misma, las ***semillas*** que se consideren. Si se utilizan estos dos puntos semillas se tiene una segmentación que consiste en dos regiones a lo más: una región R_1 asociada con la semilla (3,2) y otra R_2 asociada con (3,4). La propiedad P que se utilizará para incluir un píxel en una región es que la diferencia en valor absoluto entre los niveles de gris del píxel y la semilla sean menor que un umbral T. Cualquier píxel que cumpla la propiedad dada con relación a los dos píxeles es asignado arbitrariamente a una de las regiones. La figura 2.10 (b) muestra el resultado de $T = 3$. Las dos regiones segmentadas se notan a y b. con $T = 8$ se tendría una única segmentación como se muestra en 2.10 (c)

2.2.3 Procesamiento morfológico de la imagen.

La aproximación ***no morfológica*** de imágenes esta próxima al cálculo, basándose en conceptos como el delta de Dirac, PSF y transformaciones lineales como la convolución, en cambio la morfología matemática se basa principalmente en la geometría y la forma, las operaciones morfológicas simplifican imágenes y conservan las principales características de formas de los objetos.

Un sistema de operadores como los de la morfología matemática es útil porque pueden formarse composiciones de sus operadores, que cuando actúan sobre formas complejas, son capaces de descomponerlas en sus partes que tienen sentido y separarlas de las partes que le son extrañas. Un sistema de operadores de este tipo y su composición permite que las formas subyacentes sean

identificadas y reconstruidas de forma óptima a partir de sus formas distorsionadas y ruidosas. Además permite que cada forma se entienda en función de una descomposición, siendo cada entidad de esa descomposición una forma simple apropiada [Mol98] [GaW03].

Las ***transformaciones morfológicas*** constituyen usualmente una parte intermedia de la secuencia de procesamiento de imágenes, a continuación se presenta un resumen de cuales son las fases del procesamiento:

1. En primera fase, la imagen es digitalizada y pre-procesada usando operadores de convolución locales y luego es segmentada para obtener una imagen binaria en la que se separan los objetos del fondo.
2. Las operaciones morfológicas pueden formar una segunda fase que opera sobre la forma de esos objetos.
3. El último paso del procesamiento evalúa los resultados de la morfología usando descriptores numéricos o sintácticos.

La morfología matemática se puede usar, entre otros, con los siguientes objetivos:

1. Pre-procesamiento de imágenes (supresión de ruido, simplificado de formas),
2. Destacar la estructura de los objetos (extraer el esqueleto, marcado de objetos, envolvente convexa, ampliación, reducción),
3. Descripción cualitativa de objetos (área, perímetro, etc.).

El lenguaje de la ***morfología matemática binaria*** es basado en la teoría de conjuntos, por esta razón es necesario conocer algunos conceptos básicos de las operaciones básicas sobre conjuntos.

Operaciones básicas sobre conjuntos

Sean A y B conjuntos en un n-espacio E^n con elementos $a = (a_1,\ldots, a_n)$ y $b = (b_1,\ldots, b_n)$ respectivamente siendo ambos n-uplas.

1. La translación de A por $x \in E^n$ que se nota A_x se define como

$$A_x = \{c | c = a + x, \text{ Para algún } a \in A\}$$

2. La reflexión de B notada $B\hat{}$ se define como

$$B\hat{} = \{x | x = -b, \text{ Para algún } b \in B\}$$

3. La diferencia de dos conjuntos A y B, notada $A - B$, se define mediante

$$A - B = \{x | x \in A, x \notin B\}$$

Morfología binaria

Como anteriormente se dijo, el lenguaje de la morfología matemática binaria se basa en la teoría de conjuntos. Los conjuntos en morfología matemática representan las formas presentes en imágenes binarias o de nivel de gris. El conjunto de todos lo píxeles blancos de una imagen en blanco y negro (binaria) constituye una descripción completa de la imagen.

Los puntos en un conjunto sobre los que se aplica la transformación son el conjunto de puntos seleccionados y el complementario de los no seleccionados. En las imágenes binarias los puntos seleccionados son los que no pertenecen al fondo.

Las operaciones primarias morfológicas son la ***erosión*** y la ***dilatación***. La dilatación es la transformación morfológica que combina dos vectores utilizando la suma. La dilatación binaria fue usada primero por Minkowski, y en la literatura matemática recibe el nombre de la suma de Minkowski. Si *A* y *B* son conjuntos en un *n*-espacios E^n con elementos $a = (a_1, \ldots, a_n)$ y $b = (b_1, \ldots, b_n)$ respectivamente siendo ambos *n*-uplas., entonces la dilatación de *A* por *B* es el conjunto de todos los posibles vectores que son la suma de pares de elementos, uno de *A* y otro de *B*.

Más formalmente, la dilatación de *A* por *B* se nota $A \oplus B$ y se define mediante

$$A \oplus B = \{c \in E^n | c = a + b \text{ para algún } a \in A \text{ y } b \in B\}$$

Al ser la suma conmutativa, la dilatación también lo es: $A \oplus B = B \oplus A$.

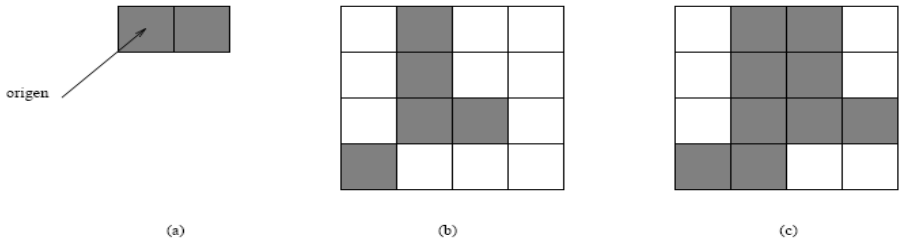


Figura 2.11 Ejemplo de dilatación.

En la práctica los conjuntos *A* y *B* no son simétricos. El primer elemento de la dilatación, *A*, está asociado con la imagen que se está procesando y el segundo recibe el nombre de elemento estructural, la forma que actúa sobre *A* en la dilatación para producir $A \oplus B$.

Cuando se realiza una dilatación con un elemento estructural que contiene el cero, lo que realiza es la expansión de una imagen y es fácil pensar en una implementación paralela. Un ejemplo se muestra en la figura 2.11, en donde al recorrer la imagen por filas, al encontrar un pixel negro, se agrega otro a la derecha del mismo. Es importante tener en cuenta que el sistema de coordenadas que se usará en este tema es (fila, columna).

La dilatación tiene las siguientes propiedades

1. La dilatación por el traslado de un elemento estructural es el traslado de la dilatación:
   $A \oplus B_t = (A \oplus B)_t$.

2. Propiedad distributiva:
   $A \oplus (B \cup C) = (A \oplus B) \cup (A \oplus C)$.

3. Asociatividad (iteración):
   $A \oplus (B \oplus C) = (A \oplus B) \oplus C$.

4. Crecimiento:
   $A \subseteq B \Rightarrow A \oplus K \subseteq B \oplus K \quad \forall K$.

La erosión es la operación morfológica dual, un concepto que se definirá formalmente a partir de la dilatación. Es la transformación morfológica que combina dos conjuntos utilizando el concepto de inclusión. Si A y B son conjuntos en el espacio euclídeo n-dimensional, entonces la erosión A por B es el conjunto de todos los elementos x para los que $x + b \in A$ para todo $b \in B$. la sustracción de Minkowski está muy relacionada con la erosión [Mol98].

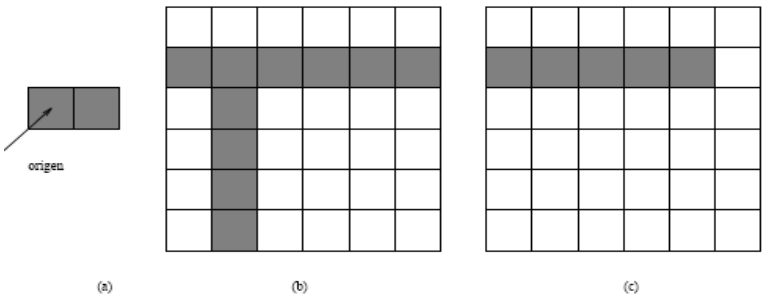


Figura 2.12 Ejemplo de erosión.

La erosión de A por B se nota $A \ominus B$ y su definición es

$$A \ominus B = \{x \in E^n | x + b \in A \text{ para todo } b \in B\}$$

Un ejemplo de erosión se encuentra en la figura 2.12, al encontrar un pixel negro se comprueba si a su lado derecho está acompañado de otro pixel negro, si lo esta se deja el pixel como esta, si no, se cambia por un pixel blanco.

La utilidad de la erosión puede apreciarse mejor cuando ésta se expresa de forma diferente. La erosión de una imagen, *A*, por un elemento estructural, *B*, es el conjunto de todos los elementos $x \in E^n$ para los cuales *B* trasladado por *x* está contenido en *A*. la demostración es inmediata y se tiene.

$$A \ominus B = \{x \in E^n | \, B_x \subseteq A\}$$

Mientras que la dilatación puede representarse como la unión de los traslados, la erosión puede representarse como la intersección de los traslados negativos.

$$A \ominus B = \cap_{b \in B} A_{-b}$$

La erosión se concibe usualmente como una reducción de la imagen original. En términos de teoría de conjuntos, el conjunto erosionado se suele pensar que siempre está contenido en el original.

2.3 Extracción de Características.

Una vez que una imagen ha sido segmentada en regiones por los métodos discutidos en los temas anteriores, es necesario representar y describir dichos píxeles de forma que puedan ser procesados posteriormente. Podría decirse que existen dos opciones

1. Representar una región en función de su frontera.

2. Realizar la representación en función de sus características internas.

Una vez elegida la representación se ha de buscar una forma de describirla. Por ejemplo, se puede escoger una representación basada en la frontera y describirla mediante su longitud, el número de concavidades que tiene, etc.

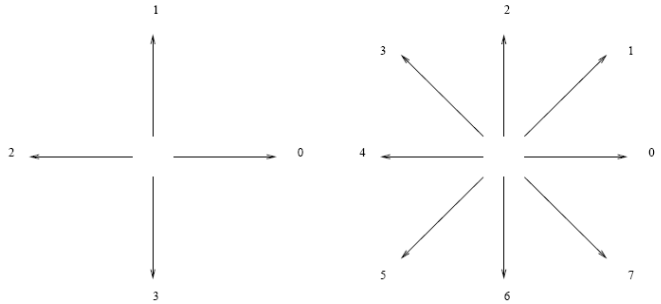


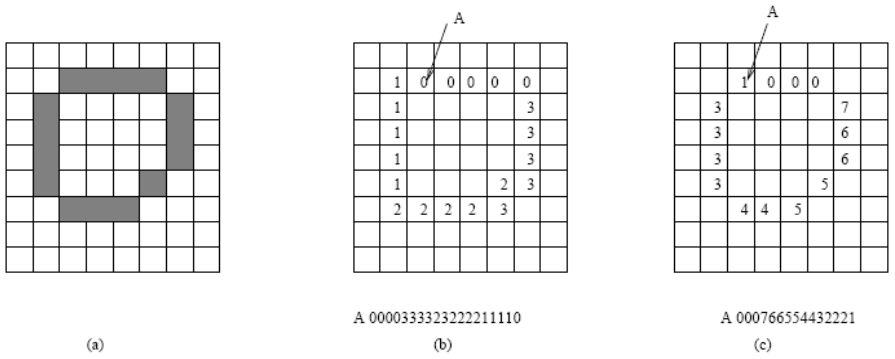
Figura 2.13 Direcciones usadas para clasificar la frontera.

2.3.1 Representación de la frontera.

La representación basada en la frontera se utiliza más cuando el énfasis es en la forma. La representación basada en la frontera se usa más cuando se buscan propiedades como el color o la textura. En cualquier caso, los rasgos que se seleccionen como descriptores deberán de ser insensibles a cambios como variaciones de tamaño, traslación, rotación, etc.

Las ***cadenas de códigos*** se utilizan para representar fronteras mediante una sucesión conectada de segmentos de líneas rectas de una longitud y dirección dadas. Normalmente esta representación se basa en la 4- u 8- conectividad. La dirección de cada segmento se codifica mediante un esquema de numeración como los que se muestran en la figura 2.13, en la primera figura sólo se puede construir una cadena de códigos horizontal y verticalmente lo que provocaría una descripción menos detallada del objeto en cambio la segunda figura muestra que se puede construir la cadena teniendo en cuenta los recorridos en diagonal [Mol98].

La figura 2.14 muestra un ejemplo de representación de una frontera utilizando cadenas de códigos. Obviamente, la cadena de códigos depende del punto de partida y es necesaria una normalización para una futura comparación.



(a) frontera, (b) 4-conectividad, (c) 8-conectividad

Figura 2.14 Codificaciones de frontera.

Una frontera digital puede ser aproximada mediante un polígono. Si la curva es cerrada, la aproximación es exacta cuando el número de segmentos en el polígono coinciden con el número de puntos en la frontera, de forma que cada par de puntos adyacente define un segmento del polígono. En la práctica el objeto de una aproximación poligonal es capturar la esencia de la forma de la frontera con el menor número posible de segmentos poligonales. El problema no es trivial y puede necesitar una enorme potencia de cálculo. Sin embargo, varias técnicas de aproximación poligonal son al mismo tiempo simples y buenas [Mol98].

Una aproximación frecuentemente utilizada subdivide un segmento sucesivamente en dos partes hasta que se cumpla un determinado criterio. Por ejemplo, se puede exigir que la distancia perpendicular máxima desde un segmento de frontera a la línea que une sus dos puntos finales no exceda un umbral dado. Si excede el umbral, el punto más alejado se convierte en vértice y divide el segmento en dos sub-segmentos. Esta aproximación tiene la ventaja de que busca puntos de inflexión prominentes. El ejemplo se ilustra en la figura 2.15.

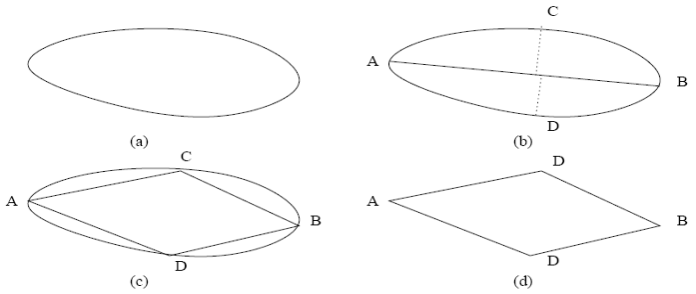


Figura 2.15 Representado una frontera por aproximaciones poligonales.

2.3.2 Representación de las regiones.

La forma de un objeto puede representarse directamente por la región que ocupa. Por ejemplo la matriz binaria

$$u(m,n) = \begin{cases} 1\ si\ (m,n) \in R \\ 0\ en\ el\ caso\ contraio \end{cases}$$

Es una representación de la región R, aunque obviamente la representación por fronteras es más eficiente que este procedimiento.

Cualquier región o imagen variable puede ser vista como una sucesión en la que se alternan hileras de ceros y unos. Los códigos de longitud variable representan estas hileras. Para una imagen que es escaneada por filas hasta llegar a la región de interés, el código consiste en la dirección de comienzo de cada hilera seguida del número de unos o ceros que la integran.

En el método de quad-trees, la región dada se incluye en un área rectangular conveniente. Esta área es dividida en cuatro cuadrantes y se examina cada uno de ellos para ver si todos sus píxeles pertenecen a la misma región o no. Si tiene pertenecientes y no pertenecientes a la región, se sigue subdividiendo. Una estructura de arbole se genera hasta que cada sub-cuadrante solo contiene píxeles de un tipo. Si se supone que se utiliza el 1 para la región y el 0 para la no región, entonces el árbol puede codificarse de forma única como una hilera de ceros, unos y gris para denotar no región, región y subdivisión necesaria [Mol98], ver figura 2.16.

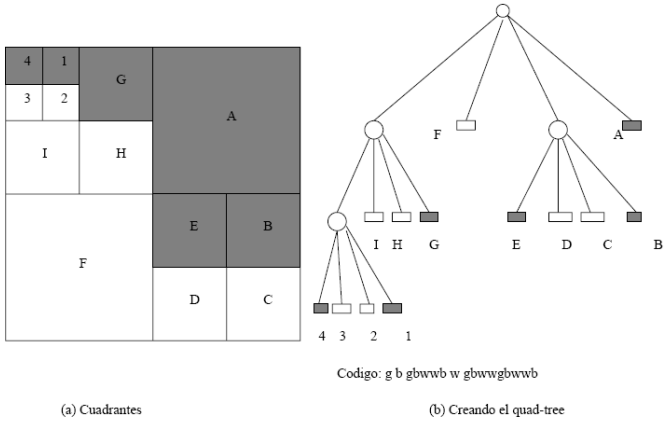


Figura 2.16 Ejemplo del método quad-trees.

2.4 Redes Neuronales.

2.4.1 La neurona artificial.

La primera ***red neuronal*** formulada por McCulloch y Pitts en 1943, era un circuito eléctrico que realizaba la suma ponderada de las diferentes señales binarias y produce una salida binaria según el resultado de la suma con relación al umbral. Una buena representación de lo que es una neurona artificial se basa en los parámetros mostrados a continuación [Kas98].

- Entradas: *x1, x2,. . ., xn*. A cada entrada se encuentra atado un peso: *w1, w2,. . ., wn*; una entrada a la neurona, llamada *bias*, tiene un valor constante de 1 y usualmente es representado como una entrada independiente, pero por simplicidad tratada como una entrada normal con valor constante.

- Una función de ***excitación*** *f*, que por lo general calcula la sumatoria de cada entrada por el peso que tiene asignado $u = \sum()$.

- Una función de activación *s* que calcula el nivel de activación de una neurona a = s(u).

- Una función de transferencia que calcula el valor que debe ser enviado por la salida (el axón) de la neurona o = g(a); por lo general la salida se supone igual al nivel de activación de la neurona, o = a.

Dependiendo del tipo de valores que cada uno de los parámetros pueda tener cada diferente tipo de neurona, la entrada y la salida pueden ser binarios, {0, 1}; bivalentes, {- 1, 1}; continuos, [0, 1]; o números discretos en un intervalo definido [Kas98].

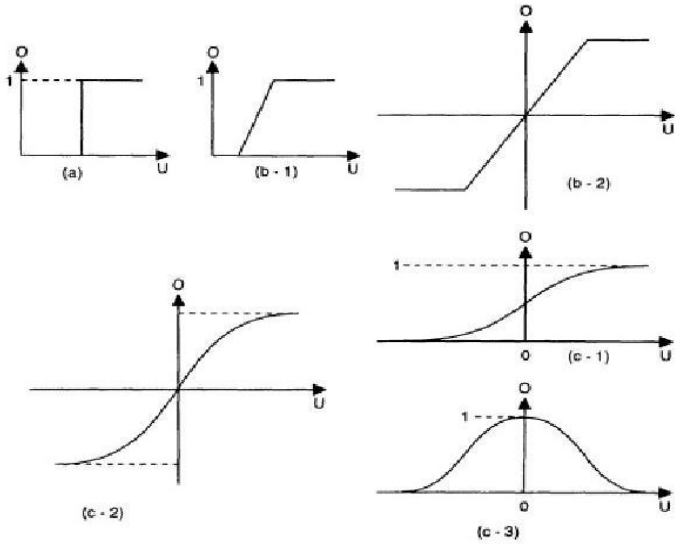


Figura 2.17 Las funciones de activación.

Las funciones de activación más usadas que se muestran en la figura son:

1. La función de umbral ***escalón unitario***. Si el valor de la sumatoria u de la neurona está por encima del umbral, la neurona se vuelve excitadora (valor de activación de 1); sino se vuelve inhibidora (valor de activación de 0).

2. La función de umbral ***lineal***. El valor de activación incrementa linealmente con el valor u, pero cuando alcanza un umbral la salida se satura (con un valor de 1); pero existen diferentes variable de esta función dependiendo del rango que deba tomar la salida de la neurona. (Figura 2.17 [b-1] and [b-2]).

3. La función ***sigmoide***. Es una función de transformación no lineal en forma de S que se caracteriza por lo siguiente:

A. Tener sus valores restringidos entre dos puntos, por ejemplo, [0, 1], [- 1, 1].

B. El valor de ***s(u)*** nunca decrece a pesar de que u aumente.

C. Continua y suave, por lo tanto diferenciable en cualquier punto del dominio. Diferentes tipos de funciones sinusoidales son utilizadas en la práctica. La mayoría de ellas son la función logística: $a = 1/(1 + e^{-u})$. En forma mas general la función logística se puede escribir como: $a = 1/(1 + e^{-c \cdot u})$, con *c* como constante.

D. Existen alternativas a esta función como lo son la logística bipolar:

h(u)= (1 - e^{-u})/(1 + e^{-u}) = 2g(u) -1; esta función tiene un rango entre [- 1, 1] y la tangente hiperbólica: ***tanh(u) = (e^{u} - e^{-u})/(e^{u} + e^{-u})***.

4. La función ***gaussiana*** (en forma de campana) (figura 2.17 [c-3]).

Una salida de la neurona pude ser representada por un solo potencial estático o por un pulso, el cual puede ocurrir (se codifica como 1) o no (se codifica como 0).

A pesar de que una sola neurona es capaz de realizar algún tipo de función simple, es en la unión de varias neuronas la que puede solucionar una gran cantidad de problemas complejos.

2.4.2 Red neuronal artificial.

Por lo general una red neuronal artificial esta modelada computacionalmente por cuatro parámetros:

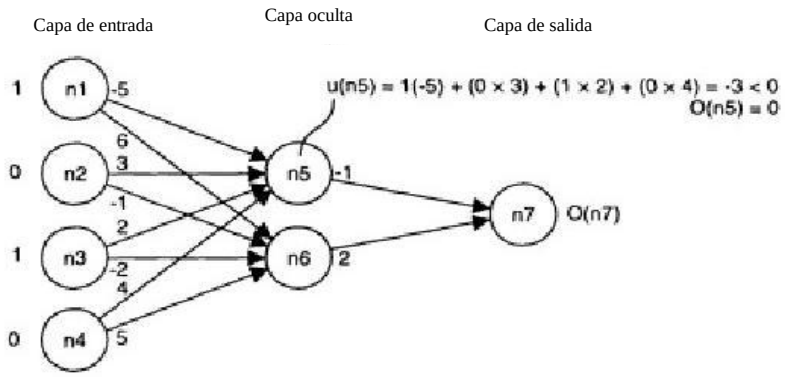


Figura 2.18 Ejemplo de red neuronal.

1. Tipo de neuronas (también conocidos como nodos)
2. La organización de las conexiones entre neuronas
3. Algoritmo de entrenamiento
4. Algoritmo de memoria

En una red neuronal, la forma en la que las neuronas estén conectadas define su topología, las neuronas pueden estar totalmente conectadas, donde cada neurona esta conectada a cada una de las otras neuronas o parcialmente conectada, por ejemplo, cuando solo se aceptan conexiones entre neuronas en diferentes capas o en general que no todas las conexiones posibles entre neuronas estén conectadas [Kas98].

Dependiendo de la presencia o ausencia de retroalimentación en la red neuronal,

se distinguen dos tipos de arquitecturas:

1. Arquitectura ***Feedforward***. No existen conexiones de la salida de la red neuronal a la entrada; este tipo de redes no guardan memoria de su estado anterior, el estado siguiente sólo depende de las nuevas entradas, las redes del tipo perceptrón son un ejemplo.

2. Arquitectura ***Feedback***. Existen conexiones de la salida de la red neuronal a la entrada; este tipo de redes guardan memoria de su estado anterior, el estado siguiente no sólo depende de las nuevas entradas sino también del estado anterior, las redes Hopfield son un ejemplo de este tipo de redes.

La mayor característica de las redes neuronales es su capacidad de ***aprender***, esto hace posible la modificación del comportamiento como respuesta al medioambiente. Una red neuronal es entrenada para que después al ingresar un vector de entrada *X* se produzca una salida deseada(o al menos consistente) de un vector *Y*, o que la red aprenda sobre las características internas y estructuras de datos de un vector de entrada *X*.

El conjunto *X*, utilizado para entrenar la red es llamado el conjunto de entrenamiento, los elementos *x* del conjunto *X* son llamados ejemplos de entrenamiento. El proceso de entrenamiento se refleja en el cambio de pesos entre las conexiones de las distintas neuronas. Durante el entrenamiento la red neuronal va convergiendo gradualmente a valores tales que a cada vector de entrenamiento *x*, la red produce la salida deseada.

La habilidad de aprender de una neurona es alcanzada a través de la utilización de un algoritmo de aprendizaje. Los algoritmos de entrenamiento pueden ser en su mayoría clasificados en los siguientes tres grupos:

1. Supervisado: Para el entrenamiento supervisado se tiene tanto los vectores de entrada como de salida El entrenamiento es logrado cuando la red asocia a cada uno de los vectores de entrada da una salida deseada.

2. No supervisado: Solo se le es suministrada a la red el vector de entrada, la red neuronal aprende algunas características de todo el conjunto de entrada que se le presenta.

3. Aprendizaje con refuerzo: Algunas veces llamado aprendizaje de recompensa y castigo, es una combinación de los dos paradigmas anteriores; esta basado en presentar un vector de entrada x a la red neuronal y mirando en vector de salida calculado por red. Si se considera “bueno”, se da una “recompensa” en el sentido de que se refuerzan las conexiones, de otra manera, la red es “castigada”, las conexiones que “se consideran no apropiadas” son inhibidas. De esta manera el aprendizaje con refuerzo es un aprendizaje con un crítico y no con un maestro.

Aprender no es una habilidad de una sola neurona. Es un proceso colectivo de toda la red neuronal y es como resultado de una actividad de aprendizaje. La matriz de pesos *W* tiene sentido como un patrón global; y representa "conocimiento". Aunque realmente no se sabe la manera en la cual el cerebro aprende. En las redes neuronales artificiales se ha llegado a unas leyes genéticas de aprendizaje [Kas98].

La ley de aprendizaje mas favorecida actualmente en el modelo conectivista es la ley de aprendizaje de Hebbian, la idea es que la conexión sináptica entre dos neuronas *i* y *j* incrementa su conexión w_{ij} si las dos neuronas *i* y *j* son simultáneamente activadas en repetidas ocasiones por un estimulo. El cambio del peso sináptico ∇w_{ij} es luego calculado como una función del producto de dos valores de activación a_i and a_j de las neuronas *i* y *j*:

$$\nabla w_{ij} = c * a_i * a_j$$

El proceso general de aprender en una red neuronal es descrito por una característica llamada ***convergencia***. La red reacciona cada vez mejor al de entrada *x* a medida que más vectores son introducidos en la red neuronal y finalmente llevando a la respuesta esperada. Después de que el entrenamiento es completado los pesos no se modifican mas, eso es $\nabla w_{ij} = 0$, para cada conexión (i, j) cuando nuevos datos o los mismos ejemplos sean entrados luego a la red. La red puede para de entrenarse por dos razones: (1) La red ha aprendido de los ejemplos, y (2) la red se ha saturado. El teorema de saturación de Grossberg indica que grandes señales de entrada saturan una neurona cuando es sensible a pequeñas señales de entrada, pero si es sensible a grandes entradas, las pequeñas entradas son ignoradas como ruido [Kas98].

$$\nabla w_{ij} = c * a_i * a_j + n,$$

donde n es la señal de ruido.

Cuando se presenta ruido durante el entrenamiento, la red neuronal alcanza una convergencia cuando los pesos cambian conforme a la magnitud del ruido (equilibrio estocástico).

$$\nabla w_{ij} \leq n$$

El estado de convergencia puede ser también alcanzado de una forma oscilatoria, que significa, que los pesos sinápticos oscilan entre dos o más estados.

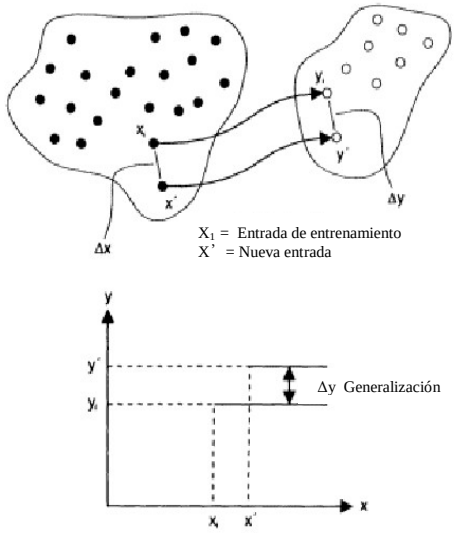


Figure 2.19 El principio de generalización.

Otra característica general de las redes neuronales, similar a la habilidad del cerebro humano, es la ***generalización***. Esto pasa cuando, después de haber entrenado la red neuronal, un nuevo vector de entrada x es presentado a la red y una función de memoria es activada. La red debe producir una salida similar a la salida del entrenamiento si la entrada nueva es similar a una de las entradas del entrenamiento. El principio general es que un estimulo similar debe producir una respuesta similar. La figura 2.19 muestra gráficamente como una red neuronal generaliza una nueva entrada.

Generalizar podría tomar una gran cantidad de iteraciones de cálculos consecutivos de estados de la red, lo cual es el caso para redes recurrentes. Eventualmente la red alcanza un estado de equilibrio, cuando la red no cambia su estado al pasar a una siguiente iteración, como si se congelara en ese estado. Este fenómeno es visto en las redes Hopfield. Es análogo al proceso de tratar de asociar una cara con una que se ha visto en el pasado, en orden de recordar donde hemos visto a la persona que nos saludó en una fiesta.

2.4.3 Entrenamiento supervisado.

El entrenamiento supervisado se basa principalmente en tener un antecedente para determinar cómo las entradas deben ser correctamente mapeadas con las salidas. De forma que el entrenamiento supervisado puede ser visto como una relación entre un espacio dominio y un espacio solución de un problema: *X -> Y*, donde muestras (ejemplos) del par (vector entrada, vector salida); *(x, y)* son conocidos, $x \in X$, $y \in Y$, *x = (x1, x2, . . .,xn)*, *y = (yl, y2, . . ., ym)*. El cómo lograr una aproximación *F'* de datos etiquetados usando redes neuronales que

generalice lo suficiente para dar una respuesta satisfactoria a nuevos entradas es el problema que concierne al entrenamiento supervisado. Por lo general este tipo de entrenamiento utiliza usualmente la siguiente secuencia:

1. Organizar una estructura apropiada para la red neuronal, teniendo, por ejemplo, *(n + 1)* neuronas de entrada (n por las variable de entrada y 1 por el bias, x_0) y m neuronas de salida, luego seleccionar unos pesos iníciales para toda la red.

2. Ingresar a la red un vector *x* del conjunto de entrenamiento.

3. Calcular el vector de salida *o* producido por la red neuronal.

4. Comparar el vector *o* con el vector deseado *y*. Si es posible, evalué el error.

5. Corregir el peso de la conexión de tal manera que la próxima vez que un vector de entrada *x* sea presentado a la red, la salida producida esté un poco más cerca de la salida deseada.

6. Si es necesario, repita del paso 2 al 5 hasta que la red alcance un estado de convergencia.

Evaluar el error es una parte importante de este tipo de entrenamiento, esta evaluación puede ser hecha de distintas maneras, las más usadas son, el error instantáneo:

$$err = (o - y), o\ err = |o - y|$$

El error cuadrático (MSE):

$$err = \frac{(o-y)^2}{2}$$

Una suma total de todos los errores cuadráticos sobre todos los ejemplos individuales y todas las neuronas de salida de la red:

$$err = \frac{\left(\sum_{k=1}^{p}\sum_{j=1}^{m}\left(o_j^{(k)} - oy_j^{(k)}\right)^2\right)}{p * m}$$

Donde

Dependiendo de cómo es calculado el error, dos tipos de error pueden ser evaluados en una red neuronal. El primero, llamado *el error aparente,* estima que

tan bien una red entrenada se aproxima a los ejemplos de entrenamiento. El segundo, llamado *el error de prueba,* estima que tan bien una red entrenada puede generalizar, esto significa, reaccionar satisfactoriamente a nuevas entradas [Kas98].

Existen distintos algoritmos de entrenamiento supervisado, a pesar de que en principio todos tienen las mismas bases, su diferencia principal reside en la forma en la cual los pesos entre las conexiones son cambiadas a través del entrenamiento. Algunos de estos algoritmos son el entrenamiento del perceptrón (Rosenblatt 1958); ADALINE (Widrow and Hoff 1960); y el algoritmo backpropagation (Rumelhart et al. 1986b; and others) [Kas98].

El perceptrón

El ***perceptrón***, que al principio fue propuesto por Rosenblatt (1958), como una simple neurona que es utilizada para clasificar una entrada entre una o dos categorías linealmente diferenciables. Puede tener varias entradas que se organizan en forma de cuadricula. Esta organización puede ser utilizada para representar una imagen, o un campo de visión, y es por esto que el perceptrón puede ser usado para lograr una clasificación simple de imágenes o tareas de reconocimiento [Kas98]. Un diseño básico del perceptrón puede ser visto en la figura 2.20.

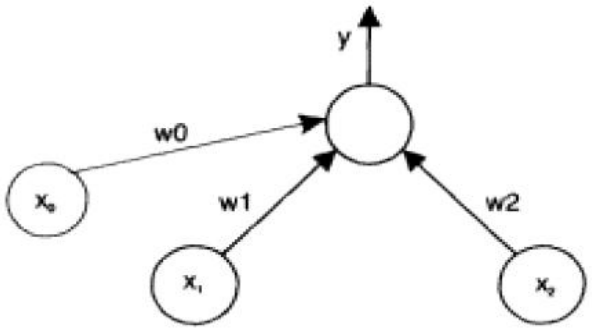


Figura 2.20 Un perceptrón simple.

El entrenamiento del perceptrón se logra cuando este clasifica mal una entrada, luego se cambian lo pesos de forma tal que si se desea una salida 1 y el valor producido por la red es 0, el valor de los pesos aumenta o disminuye en el caso contrario. Si el valor producido a la salida $o_j^{(k)}$ es igual al valor deseado $y_j^{(k)}$ los valores de los pesos no cambian. A continuación se describen los pasos del algoritmo general de entrenamiento del perceptrón:

P1. Organice un perceptrón con *(n+1)* entradas y m salidas. A todos los pesos entre las conexiones w_{ij}, *i=1, 2,…, n* y *j=1, 2,…, m*; ingréseles pequeños valores

aleatorios.

P2. Ingrese a la red un vector *x* del conjunto *X* de entrenamiento y calcule la función de excitación u_j para cada salida de la neurona j usando la fórmula:

$$u_j = \sum (x_j * w_{ij})$$

Para *i=1, 2,…, n* y *j=1, 2,…, m* donde x_0 es el *bias*.

P3. Aplique una función de umbral escalón unitario al resultado de la función de excitación, de la siguiente forma:

$o_j = 1$ *si* $u_j >$ *umbral, sino* $o_j = 0$.

P4. Calcule el error para cada neurona al comparar el valor actual de la salida con el deseado:

$$Err = (o - y).$$

P5. Modifique cada peso w_{ij} al calcular el nuevo valor $w_{ij}(t+1)$ basándose en el anterior $w_{ij}(t)$ y en el error evaluado Err_j:

$$w_{ij}(t+1) = w_{ij}(t) + \alpha x_i * Err_j,$$

donde α en el coeficiente de aprendizaje, un numero entre 0 y 1.

P6. Repetir entre el paso P2 y P5 hasta que el error sea lo suficientemente pequeño.

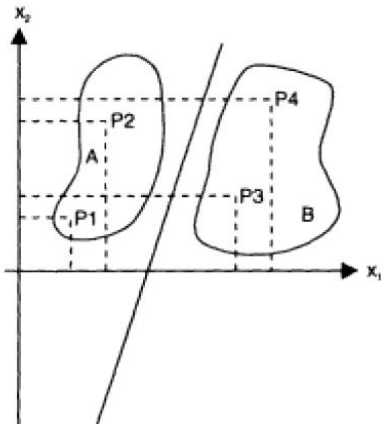


Figura 2.21 Ejemplo de dos clases linealmente separables.

Como se muestra en la figura 2.21 el perceptrón sólo puede clasificar clases linealmente separables, como se muestra en la figura 2.22, este tipo de separación puede ser utilizado para solucionar la función OR, pero presenta problemas a la hora de separa la función XOR

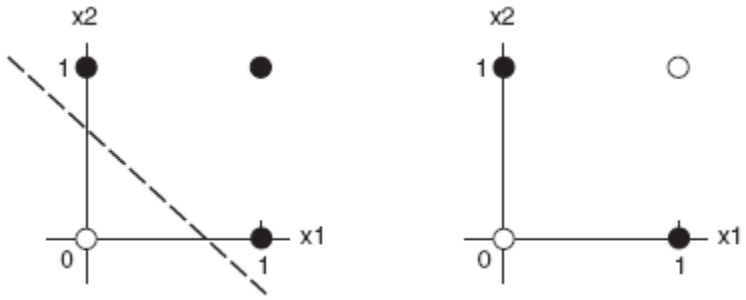


Figura 2.22 Problema del XOR.

La razón por la cual un sólo perceptrón solo puede modelar funciones que sean linealmente separables se puede ver al revisar la función de transferencia de este tipo de neurona.

$$X = \sum_{i=1}^{n} x_i * w_i$$

$$Y = \begin{cases} 1 = para\ X \geq Umbral \\ 0 = para\ X < Umbral \end{cases}$$

Usando estas funciones, se está dividiendo exitosamente el espacio de búsqueda haciendo uso de una línea en la cual X = Umbral. De esta forma, un perceptrón con dos entradas, la línea que divide una clase de la otra esta definida por

$$w_1 x_1 + w_2 x_2 = umbral$$

Teniendo en cuenta lo anterior se nota que la forma como trabaja el perceptrón es simple, a pesar de esto, este todavía es ampliamente usado por su arquitectura sencilla y la convergencia incondicional cuando se consideran clases linealmente separables. Pero para problemas más complejos es necesario utilizar otro tipo de redes que pueden lograr separaciones no lineales y múltiples clasificadores.

Redes neuronales multicapa y el algoritmo Backpropagation

La mayoría de problemas en el mundo real no son linealmente separables, es por esta razón por la cual a pesar de que el perceptrón es un buen modelo de estudio

para entender como funcionan las redes neuronales, pero para trabajar en problemas reales es necesario algo que pueda trabajar con sistemas no lineales. Como se ha descrito anteriormente las redes neuronales consisten de un número de neuronas conectadas, usualmente arregladas en capas.

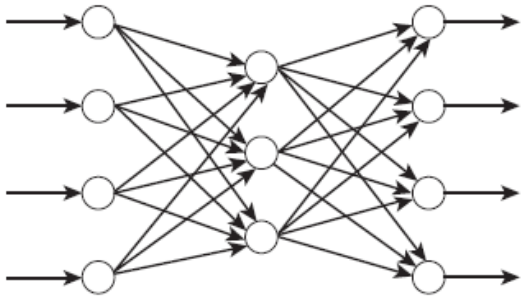


Figura 2.23 Ejemplo de red neuronal feed-forward.

Un solo perceptrón puede ser pensado como un perceptrón de una capa. Perceptrónes conectados en multicapa son capaces de modelar funciones más complejas, incluidas aquellas que no son linealmente separables como la función XOR.

Como se ve en la figura 2.23, donde se muestra un ejemplo de una red neuronal feed-forward que consiste de 3 capas. La primera capa, la capa de entrada. Cada nodo (neurona) recibe una señal. De hecho, en algunos casos estos primeros nodos no son neuronas, sino que simplemente dejan pasar las entradas a la siguiente capa, en este caso, la capa oculta. Es normal que una red neuronal tenga más de una capa oculta, que es donde generalmente se hace el verdadero trabajo. Al final se pasa a la última capa, la capa de salida. Esta se encarga de enviar la señal de salida. Es de gran importancia notar que las neuronas de una capa pueden estar total o parcialmente unidas a las neuronas de la capa siguiente, según el tipo de problema a tratar [Cop04].

Las redes neuronales multicapa aprenden de forma similar que un perceptrón. La mayor diferencia es que cada neurona tiene asociadas a cada entrada un peso, lo que genera una mayor cantidad de pesos que ajustar cuando existe un error. Claramente la pregunta más importante es cómo seleccionar los pesos que mas contribuyen al error. Un método bastante utilizado es el algoritmo de backpropagation.

En vez de usar la función escalón unitario, las redes neuronales backpropagation usualmente usan la función sigmoide, ilustrada en la figura 4.1 (c-1). Esta función se define como:

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

Se usa porque es muy fácil de diferenciar:

$$\frac{d\sigma(x)}{dx} = \sigma(x) * (1 - \sigma(x))$$

De la misma manera que inicia el entrenamiento del perceptrón, el algoritmo ***backpropagation*** empieza inicializando los pesos de la red con valores aleatorios, los cuales son generalmente pequeños valores; en el rango [-0.5, 0.5].

Como en el entrenamiento del perceptrón, en el algoritmo backpropagation cada iteración involucra primero alimentar la red con entradas y observar las repuestas. En la siguiente fase, la cual le da el nombre al algoritmo, el cual involucra mirar cómo se propaga el error desde la capa de salida hasta la capa de entrada. Estos valores del error retro actúan a través de la red generando cambios en el valor de los pesos en todas las neuronas. Este algoritmo se repite hasta que las salidas de la red neuronal sean lo suficientemente cercanas a los valores deseados, en otras palabras, hasta que el error sea lo suficientemente pequeño [Cop04].

Como la función sigmoidea no puede alcanzar los valores de 0 o 1, es usual que se acepten valores como 0.9 para 1 y 0.1 para 0.

Ahora se debe ver la fórmula que es usada para ajustar los pesos en este algoritmo. Considérese una red neuronal con tres capas, *i* representa las neuronas de la capa de entrada, *j* representa las neuronas en la capa oculta, y *k* representa las neuronas de la capa de salida. Por lo tanto, w_{ij} se refiere a los pesos de una conexión entre una neurona de la capa de entrada y una neurona de la capa oculta.

La función que es usada para derivar el valor de una salida para un nodo *j* es la siguiente:

$$X_j = \sum_{i=1}^{n} x_i * w_{ij} - \theta_j$$

$$Y_j = \frac{1}{1 + e^{-X_j}}$$

Donde *n* es el número de entradas del nodo *j*; w_{ij} es el peso entre el nodo *i* y el nodo *j*; $\theta_j$es el valor umbral que es usado para el nodo *j*, el cual es un valor al azar entre 0 y 1; x_i es el valor de entrada para el nodo *i*; y y_j es el valor de salida producido por el nodo *j*.

Una vez que cada nodo de la red va generando una salida, el gradiente del error

es calculado para cada nodo *k* en la capa de salida. La señal de error en *k* se conoce como la diferencia entre el valor deseado d_k y el valor actual para ese nodo y_k.

$$e_k = d_k - y_k$$

El gradiente del error para la salida del nodo *k* es definida como el valor del error para el nodo *k* por la derivada de la función de activación:

$$\delta_k = \frac{\partial y_k}{\partial x_k} * e_k$$

x_k es la suma ponderada de los valores de entrada del nodo *k*.

Debido a que *y* se define como una función sigmoidea de *x*, se puede usar la fórmula que fue dada anteriormente para derivar la función sigmoidea para obtener la siguiente fórmula para el error gradiente:

$$\delta_k = y_k * (1 - y_k) * e_k$$

Similarmente, se calcula un error gradiente para cada nodo *j* en la capa oculta, como se muestra a continuación:

$$\delta_j = y_j * \left(1 - y_j\right) * \sum_{k=1}^{n} w_{ik} * \delta_k$$

Donde *n* es el número de nodos en la capa de salida, y de este modo el número de salidas para cada nodo en la capa oculta. Ahora cada peso en la red, w_{ij} o w_{jk}, es actualizado de acuerdo a la siguiente fórmula:

$$w_{ij} \leftarrow w_{ij} + \alpha * x_i * \delta_j$$
$$w_{jk} \leftarrow w_{jk} + \alpha * y_j * \delta_k$$

Donde x_i es el valor de entrada del nodo *i*, y α es la tasa de aprendizaje, la cual es un número positivo por debajo de 1, el cual no debe ser muy alto [Cop04].

Este método es conocido como gradiente descendiente, porque implica seguir la senda empinada hacia abajo de la superficie que representa la función de error para encontrar el mínimo error en el espacio, lo que representa el conjunto de pesos que ofrece el mejor rendimiento de la red. De hecho, las iteraciones terminan usualmente en el momento cuando la suma de los errores cuadráticos de los valores de salida para todos los datos de entrenamiento en un momento dado es menor que algún umbral.

Nótese pues que este método le asigna un castigo individual a cada uno de los

nodos de la red al asociar los pesos adjuntos a cada nodo con el error asociado a tales nodos. En el caso de nodos de la capa oculta, no hay valor de error para esos nodos ya que no se sabe cuál es el valor deseado. En este caso, los pesos de cada conexión entre la capa oculta y la capa de salida; son multiplicados por el error de ese nodo de salida de manera que este se distribuya entre los nodos de la capa oculta de acuerdo con el porcentaje según el cual cada nodo contribuye con el error [Cop04].

3. ANÁLISIS DEL SISTEMA

En este capítulo se mostrará cómo será la estructura del sistema que se desea desarrollar y se explicara las razones por las cuales se escogieron los diferentes algoritmos que serán utilizados en la creación del mismo.

3.1 Estructura del sistema.

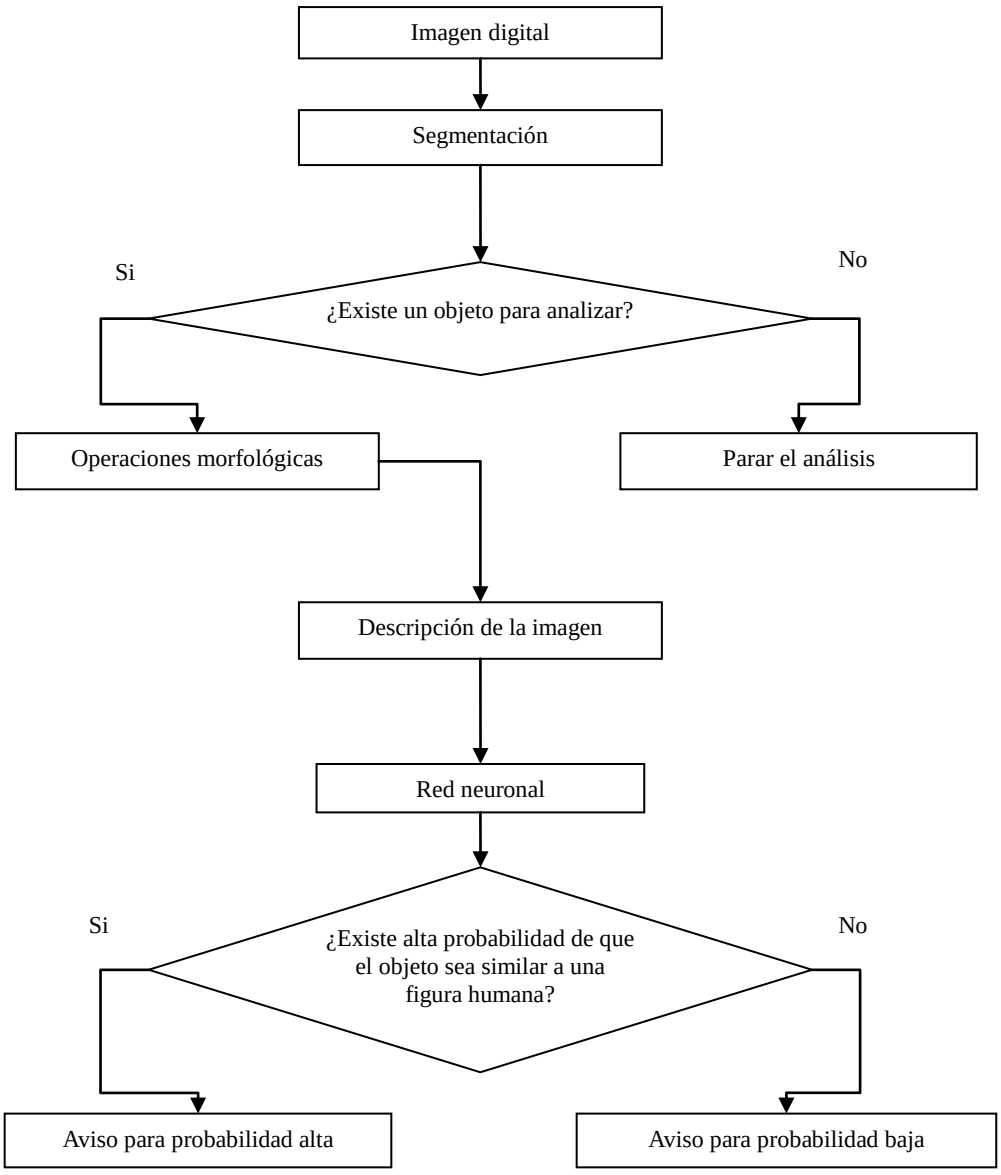


Figura 3.1 Estructura del sistema.

3.2 Análisis de la imagen.

3.2.1 Imagen digital.

A la hora de tratar con tipos similares de sistemas es necesario escoger con que tipo de imagen se quiere trabajar, ya que tomar una buena decisión podría facilitar el resto de proceso necesario para analizar la imagen.

Primero que todo se debe pensar en los estándares que son usados por los equipos fotográficos y de video más comunes en el mercado, tomando esto como punto de partida rápidamente se nota que los tipos de imágenes más comunes en cualquier dispositivo son las imágenes RGB y las imágenes en escala de grises.

Ya en este punto la decisión a tomar se basa en la facilidad con la que cada tipo de imágenes permita trabajar. Las imágenes RGB representan muy bien los objetos y sus colores reales, pero es problemático segmentar la imagen al tener una gran cantidad de colores (16.777.256) que revisar. Las imágenes en escala de grises representan la imagen con 256 colores lo que hace que analizar este tipo de imágenes sea más fácil que con imágenes RGB; además existen una gran cantidad de algoritmos que permite a partir de una imagen en formato RGB obtener una en escala de grises.

3.2.2 Segmentación.

Es necesario que al analizar una imagen se distinga entre el objeto sobre el cual se va a trabajar y el fondo de la imagen, este proceso como se vio en el capítulo anterior se conoce como segmentación. Esta es posiblemente la parte más importante del análisis de una imagen.

Para tomar esta decisión simplemente se ha pensado en lo que sea más fácil de implementar y que pueda ofrecer un desempeño aceptable; por estas razones se pensó en la comparación pixel a pixel entre dos imágenes, una que contenga solo el fondo y otra con el fondo y la imagen a analizar. Se conoce las fallas de este tipo de opción como el ruido y los problemas con cambios extremos de iluminación, pero esto es tratable con una buena captura de la imagen.

3.2.3 Descripción de la imagen.

Como se vio en el capítulo anterior, se debe buscar la forma de describir la imagen de acuerdo con lo que se desee identificar de ella. Por esta razón se ha tomado la decisión de trabajar con el contorno de la imagen ya que esto encaja de manera aceptable con la solución que se le desea dar al problema al proponer este proyecto; y luego representar la silueta usando una cadena de códigos.

Para obtener el contorno de la imagen se han propuesto distintos algoritmos como

se vio en el capítulo anterior, la mayoría de ellos basados ya sea en la primera o la segunda derivada; al toparse con esta decisión se optó por el aquel cuya implementación fue fácilmente encontrada en la red, el algoritmo de Sobel que ha sido usado con éxito en distintos proyectos. Si la extracción de la frontera es exitosa se procederá a representarla mediante una cadena de códigos y luego trabajando sobre la misma se creará un vector que será utilizado por la red neuronal.

3.3 Red neuronal.

Primero que todo, en base a las especificaciones del problema se llega a la conclusión de que se debe usar una red neuronal con entrenamiento supervisado. Es fácil descartar el uso de un perceptrón en este sistema ya que difícilmente este problema sea linealmente diferenciable, aunque no se puede asegurar con absoluta firmeza.

La siguiente opción es una red neuronal multicapa, opción que normalmente se suele utilizar y que cuenta con una gran documentación de sus usos prácticos, lo que la hace ideal para este proyecto. Este tipo de red es capaz de resolver problemas no lineales a su vez no tiene dificultad de tratar con problemas lineales, lo que la hace una herramienta destacada en el campo del reconocimiento de patrones.

Por último se puede recalcar que más que a la par con un buen diseño de la red neuronal esta un buen algoritmo para entrenarla, en el capítulo anterior se dio una amplia explicación del funcionamiento del algoritmo backpropagation, al mismo tiempo su eficiencia está bien documentada y es soportado por varias librerías, lo que lo convierte en una solida elección para el desarrollo del sistema.

3.4 Arquitectura.

A la hora de desarrollar un nuevo sistema de seguridad se debe tener en cuenta en que ambientes este podría ser utilizado, por una lado un escenario en el cual una familia desea ser avisada si existe la posibilidad de una persona entrando sin permiso a su propiedad o un escenario en el cual se deben revisar múltiples cámaras como en el caso de grandes empresas, conjuntos cerrados o edificios públicos.

Por estas razones se tomo la decisión de resolver el mismo problema para dos arquitecturas diferentes; una aplicación de escritorio que sea una opción viable para pequeñas empresas y ambientes domésticos, donde es preferible algo sencillo y fácil de configurar y una aplicación sobre una plataforma cliente-servidor más útil para analizar una gran cantidad de información proveniente de distintas cámaras ya se encuentren estas en el mismo sector o en zonas geográficamente separadas.

4. DISEÑO DEL SISTEMA

A continuación se describe como se ha diseñado el sistema que se encargará de de resolver el problema que se ha descrito al principio de este documento. Para la implementación del prototipo se hará uso de un paradigma orientado objetos con el cual se crearan dos clases principales, la primera encargada del análisis de la imagen y la segunda se encargará del reconocimiento de patrón con la red neuronal.

4.1 Captura de la imagen.

Para este prototipo en la captura de imágenes se usa una cámara digital de 32 bits, acondicionada para tomar fotos en un tamaño de 640 píxeles por 480 píxeles en formato RGB. Estas fotos se guardan en el equipo y se leen cuando el operador lo considere necesario para permitir el procesamiento y la toma de decisiones del software, además se guardó una imagen de control con sólo la imagen del fondo.

Para evitar una alta complejidad en el problema, estas imágenes serán tomadas en un ambiente controlado en el cual se controlan variables como la luminosidad y la presencia de objetos que puedan dificultar el proceso de decisión.

En esta primera versión del prototipo se trabajará de esta manera pero la idea final es utilizar una cámara digital que cada cierto tiempo tome una fotografía del sector que está vigilando.

4.2 Procesamiento de la imagen.

4.2.1 Conversión de RGB a escala de grises.

Para simplificar el procesamiento de la imagen, esta debe (si esta en formato RGB) ser convertida a una escala de grises, de esta manera solo hay 256 valores a comparar.

4.2.2 Separación del fondo.

Lo primero que se va a hacer es diferenciar el fondo de los objetos importantes de la imagen, para esto se computa la media y la variancia de la intensidad de cada píxel utilizando unas imágenes iníciales (en el cual no hay ningún objeto presente). Se asume que el fondo es relativamente estacionario y el uso de un umbral adaptativo $\tau(x, y)$ para cada pixel $p(x, y)$, debido a que puede existir ruido. Cada píxel de la imagen es comparado con un píxel de la imagen de control correspondiente para extraer los objetos a analizar [SBaM].

Para obtener el objeto se clasifica cada pixel en la imagen de acuerdo a la siguiente desigualdad:

Si $|p(x, y) - u(x, y)| < \tau(x, y)$
Es un pixel del fondo $p(x, y) = 0$ *(negro)*
Si no
Es un pixel de un objeto $p(x, y) = 255$ *(blanco)*

Al final se obtiene una imagen binaria en la cual se puede distinguir el fondo y los objetos a analizar.

4.2.3 Operaciones morfológicas.

En orden de poder extraer la silueta de la imagen, el ruido presente es removido al utilizar operaciones morfológicas como la dilatación y la erosión [SBaM]. Primero que todo se realiza una erosión que está destinada a eliminar ruido que se haya producido por el proceso anterior, luego una dilatación que evita que el objeto que se va analizar este desconectado de una de sus partes, y luego otra erosión para que el objeto tenga las mismas proporciones que el original.

4.2.4 Extracción de la frontera.

Para la extracción se aplica el algoritmo de Sobel utilizando una variación mostrada en [Gop97]. En esta variación en vez de solo utilizar dos mascaras de convolución para la horizontal y la vertical, se utilizan 4 mascaras; una para la horizontal, otra para la vertical, otra para la diagonal derecha y otra para la diagonal izquierda. Las ecuaciones parciales se pueden escribir como:

$$S_V = (a_2 + ca_3 + a_4) - (a_0 + ca_7 + a_6)$$
$$S_H = (a_0 + ca_1 + a_2) - (a_4 + ca_5 + a_6)$$
$$S_{DI} = (a_1 + ca_2 + a_3) - (a_7 + ca_6 + a_5)$$
$$S_{DD} = (a_1 + ca_0 + a_7) - (a_3 + ca_4 + a_5)$$

Con un *c = 2*;

Las anteriores ecuaciones pueden ser expresadas con las siguientes máscaras de convolución.

$$S_H = \begin{matrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{matrix} \qquad S_V = \begin{matrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{matrix}$$

$$S_{DI} = \begin{matrix} 0 & 1 & 2 \\ -1 & 0 & 1 \\ -2 & -1 & 0 \end{matrix} \qquad S_{DD} = \begin{matrix} 2 & 1 & 0 \\ 1 & 0 & -1 \\ 0 & -1 & -2 \end{matrix}$$

Estas cuatro medidas son conocidas para la magnitud y la fase del gradiente. La magnitud del gradiente puede ser estimada como [Gop97].

$$Magnitud = Max[|S_H|, |S_V|, |S_{DI}|, |S_{DD}|] + 1/8[|S_\perp|]$$

Donde $S_\perp$ es la medida en la dirección perpendicular al máximo. La magnitud es comparada con un umbral para determinar si es un píxel de borde, si el píxel es detectado se representa con un 255 (blanco), de lo contrario se representa con 0 (negro).

4.3 Extracción de características.

4.3.1 Representación de la frontera.

La técnica que se utiliza es la de representar el borde como una cadena de códigos. Si se asume que cada píxel está conectado a sus ocho vecinos (8-conectividad), cada vez que se encuentra un pixel vecino, en la cadena se ingresa un número que representa la dirección de ese pixel [SBaM]. Se recorre el contorno desde un punto cualquiera en el sentido de las agujas del reloj para generar una cadena de códigos del contorno de la imagen.

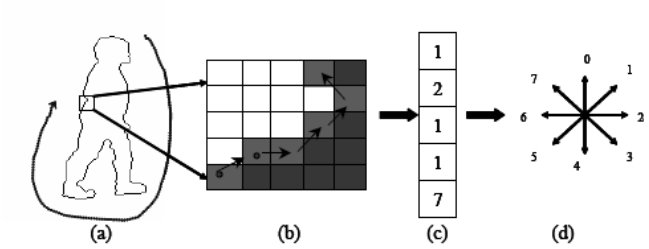


Figura 4.1 Representación del contorno de silueta humana.

La clasificación requiere que los rasgos relevantes sean extraídos de la cadena de código que representa el contorno de la silueta [SBaM]. La frecuencia $f_j(k)$ de una valor $v_j(k)$ es el numero de tuplas en la relación R con el atributo $X_j = v_j(k)$. la representación de la silueta puede ser aproximada como una distribución de datos D_j. En el caso donde la silueta es representada por una cadena de códigos, la distribución D_j puede ser evaluada como el histograma de una cadena de código generada [SBaM] para cada imagen. La distribución de datos se construye a partir del atributo de direccionabilidad.

Para lograr que no exista una varianza basada en la cercanía o lejanía del objeto a analizar es necesario normalizar el vector de direcciones de manera que la media fuera cero, de esta forma que todos los valores de este vector se encuentran entre:

$$\left[-\frac{1}{k}, 1-\frac{1}{k}\right]$$

Si *f(i)* es la frecuencia de la i-esima tupla de la distribución de datos, el i-esimo componente del vector, la frecuencia normalizada puede ser escrita como:

$$\emptyset_i = \frac{f(i)}{\sum_{i=1}^{k} f} - \frac{1}{k} \; para \;\; 1 \leq i \leq k$$

Pero para facilitar el trabajo se utilizará con una fórmula simplificada:

$$\emptyset_i = \frac{f(i)}{\sum_{i=1}^{k} f} para \;\; 1 \leq i \leq k$$

Al final, el vector que se construya en base a la frontera será usado como entrada de la red neuronal.

4.4 Red Neuronal.

4.4.1 Diseño.

Para dar solución al problema se empleará una red neuronal multicapa por cada posición que se desea detectar (caminando, de frente, etc.). El diseño es el mismo pero para cada posición se entrena de manera individual, la capa de entrada está compuesta de 8 neuronas que reciben la información de la distribución de datos normalizada luego se ingresa a las neuronas de la capa oculta y al finalizar la capa de salida sólo cuenta con una neurona que es la encargada de informar si la imagen analizada corresponde a la posición que esa red neuronal debe detectar.

Esta aproximación al problema permite que se evite la creación de una compleja red neuronal que podría generar problemas como no poder acomodar fácilmente sus parámetros y requerir una gran cantidad de datos para ser entrenadas y en cambio hacer uso de un grupo de redes neuronales más sencillas que simplemente se entrenan de forma diferente para que cada una solamente identifique un sola figura.

4.4.2 Entrenamiento.

Dentro de la base de datos se encuentran dos tipos de datos, unos para el entrenamiento y los otros para la prueba. Los datos para el entrenamiento están conformados por aquellos correspondientes a las distintas posiciones a detectar, por los datos de distintos objetos que puedan estar presentes en un espacio cerrado y por ultimo datos validos pero que no hacen referencia a ningún objeto ni

a ninguna posición.

La cantidad mínima de datos para entrenar está dada por la afirmación "*Si la red tiene **n** entradas, **h** unidades ocultas y una salida → el número de vectores de entrenamiento debe ser superior a n*h + h = (n+1) * h pesos variables*" [Nil00].

Si la red no converge a una solución se probarán con distintos números de neuronas en la capa oculta hasta lograr una que pueda desempeñar la labor de una manera aceptable.

4.5 Interfaz.

mmmm

Como una de las partes de la implementación de un proyecto, el diseño de la interfaz es de gran importancia debido a que esta representa la forma en la cual el usuario interactúa con la aplicación. Como se dijo en el capítulo anterior este proyecto será desarrollado teniendo en cuenta dos arquitecturas distintas, a continuación se explicará la forma en la cual el usuario se comunicara con cada una de ellas.

4.5.1 Escritorio.

Por lo general las interfaces para aplicaciones de escritorio son diseñadas para permitirle al usuario trabajar fácilmente y con un ambiente visual. Este proyecto no es la excepción, como se ha dicho anteriormente, esto es un prototipo por lo que la interfaz no será muy pulida, pero lo suficiente para permitir que un usuario con poca instrucción pueda manejarlo.

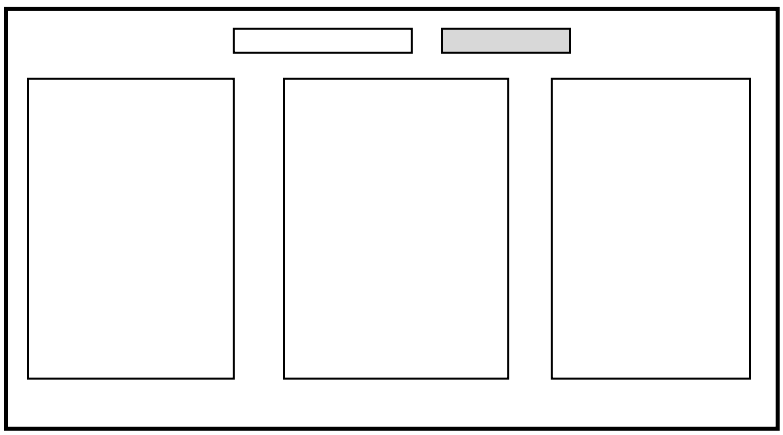


Figura 4.2 Diseño de la interfaz para la aplicación de escritorio.

Como se muestra en la figura 4.2 es un diseño sencillo, en la parte superior de la aplicación se encontrará el espacio donde el usuario ingresará el nombre de la imagen que desea analizar, debajo de este espacio se mostrarán 3 imágenes, que representan las diversas etapas del análisis de la imagen. Luego dependiendo de

la respuesta de la red neuronal se lanzara una alarma que será tanto visual como sonora para dar alerta al usuario que lo está usando.

4.5.2 Cliente-Servidor.

Para esta arquitectura el diseño es distinto que el que sería usado en la aplicación de escritorio; como es usual en este tipo de arquitecturas, algunas veces es necesario sacrificar el atractivo visual para enfocar el uso el procesador al análisis del problema. Por esta razón se ha decidido que el funcionamiento de la aplicación sea en consola, lo que limita el desarrollo visual de la misma.

Tanto el cliente como el servidor obligarán al usuario a tener un mejor conocimiento de la aplicación y a trabajar con una mínima ayuda visual, en la parte del cliente se mostrará unos espacios donde debe digitar el nombre de la imagen a usar y la respuesta que brinda el servidor. El servidor mostrará una reducida cantidad de información sobre lo que el sistema está procesando como lo que le envió el cliente y la respuesta que este produce. Para el cliente la forma de dar una alarma será principalmente sonora.

5. IMPLEMENTACION DEL SISTEMA

Para la implementación del sistema se hizo uso de un computador portátil compaq v2250 con un procesador **Turion 64** de 1.8 Ghz y 1Gb de RAM con un sistema operativo **Ubuntu 8.04**. Para lograr que el sistema sea fácilmente migrado a otros sistemas operativos se utiliza el lenguaje de programación **C++** con el compilador GNU GCC 4.2. Para el análisis de imágenes se utiliza una librería libre conocida como **CImg** y para el trabajo con redes neuronales otra librería libre conocida como **FANN.**

- **CImg** (C++ Template Image procesing tool) es una librería para el procesamiento de imágenes que tiene como principales características el ser simple de usar, portable entre distintos sistemas operativos.

- **FANN** (Fast Artificial Neural Network) es una librería de **C** para el manejo de redes neuronales, permite el uso de redes multicapa total o parcialmente conectadas.

Estas librerías fueron seleccionadas sobre otras que realizan un trabajo similar principalmente por que poseen una gran cantidad de documentación que facilita la tarea del aprendizaje, además ambas librerías son fáciles de utilizar y tienen licencias no privativas que pueden ser usadas sin costo alguno por cualquiera que desee utilizarlas.

Como se dijo en el capítulo anterior la implementación del prototipo fue desarrollado en un paradigma orientado a objetos en el lenguaje C++, el prototipo tiene como base dos clases, la primera encargada del análisis de la imagen y la segunda de la red neuronal. Este diseño permite que sea fácil utilizar el mismo proyecto para distintos enfoques. El primer enfoque es orientado a una aplicación que corre en un ordenador y el segundo para una arquitectura cliente-servidor.

Por ser un prototipo, las fotos son tomadas previamente en un ambiente controlado con una cámara digital, con un fondo blanco, los objetos de color obscuro y una iluminación que impide la formación de sombras, ya que estas dificultan la extracción de características de la imagen. La aplicación analiza estas imágenes e ingresa en la base de datos las características a analizar de la imagen, a qué posición pertenecen y si la imagen va a ser utilizada para el entrenamiento de la red neuronal o para probarla.

5.1 Entrenamiento.

Antes de empezar cualquier trabajo con el prototipo, fue necesario entrenar la red neuronal para que esta funcione de acuerdo con las necesidades del problema. Para el entrenamiento de la red neuronal primero fue necesario tomar las imágenes e ingresar a la base de datos las características con las cuales se va a identificar una posición de la figura humana. El sistema tiene un modulo sencillo el cual trabaja en consola para este propósito. A continuación unos datos experimentales de las dos posiciones que se tienen en cuenta para ser detectadas por el prototipo y de algunos objetos sin parecido a una silueta humana.

Posición número 1(Caminando)

	F1	F2	F3	F4	F5	F6	F7	F8
11.jpg	0,057656	0,300567	0,0727788	0,0482042	0,0604915	0,335539	0,0359168	0,0888469
35.jpg	0,0577617	0,283394	0,0812274	0,0478339	0,0749097	0,307762	0,0406137	0,106498
54.jpg	0,0413369	0,299912	0,0879508	0,0474934	0,0466139	0,335092	0,0483729	0,0932278

Tabla 5.1 Cadena de códigos de siluetas humanas caminando.

Posición número 2 (Parado de frente)

	F1	F2	F3	F4	F5	F6	F7	F7
90.jpg	0,0351035	0,369937	0,0675068	0,0207021	0,0423042	0,369937	0,060306	0,0342034
103.jpg	0,0402299	0,326765	0,0648604	0,0517241	0,0484401	0,342365	0,0418719	0,0837438
106.jpg	0,0478943	0,345995	0,0528489	0,0396367	0,0561519	0,355904	0,0355078	0,0660611

Tabla 5.2 Cadena de códigos de siluetas humanas paradas de frente.

Objetos variados

	F1	F2	F3	F4	F5	F6	F7	F7
131.jpg	0.00341297	0.00341297	0.0477816	0.221843	0.204778	0.450512	0.0443686	0.0238908
167.jpg	0.0241228	0.111842	0.166667	0.192982	0.0197368	0.127193	0.157895	0.199561
162.jpg	0.0798226	0.125277	0.0388027	0.252772	0.0587583	0.172949	0.0133038	0.258315

Tabla 5.3 Cadena de códigos de objetos no similares a siluetas humanas.

Como se explicó en capítulos anteriores, el vector que ingresa a la red neuronal está compuesto de la frecuencia con la cual cada una de las 8 direcciones que puede tomar un pixel para unirse con otro al recorrer el contorno aparece en la cadena de códigos.

Al revisar los datos experimentales de las tablas 5.1 y 5.2 es posible ver una similitud entre los datos que conforman cada una de las tablas lo que indica indudablemente la presencia de un patrón, pero también es visible que entre las tablas los valores tienen sus diferencias.

Al mirar la tabla 5.3 es evidente la diferencia que tienen ciertos objetos con una silueta humana en base a la frecuencia de las direcciones, algo que se puede inferir es que a pesar de que los datos de las tablas 5.1 y 5.2 no son completamente similares, si lo son a la luz de los otros objetos que se han

analizado. Por este razón y por la falta de datos para entrenar la red neuronal se decidió trabajar con una sola red neuronal la cual se encarga de determinar si existe una alta o baja probabilidad de que se encuentre una figura humana ya sea en posición de frente o caminando.

Al final, después del entrenamiento se creó un archivo de configuración que guarda la información de la red, para que esta pueda ser transportada a distintos equipos sin la necesidad de ser reprogramada.

5.2 Resultados experimentales.

Después de finalizado el entrenamiento se pone a prueba el sistema con distintas imágenes, unas de simples objetos o dibujos y otras en las cuales se encuentra presente una figura humana. Estas imágenes están en formato .jpg con una resolución de 480x640 pixeles. Dependiendo del alto o bajo grado de probabilidad de encontrar una forma humana, se lanza una alarma y se guarda la imagen para su verificación indicando en el nombre de la foto el grado de probabilidad.

En este primer prototipo el sistema no actúa de forma autónoma tomando las fotos que le sean entregada por una cámara, en cambio las imágenes ya están listas y simplemente se abren cuando se desean para la prueba.

En la figura 5.1 se muestra la alarma que sigue a una imagen con un alto grado de probabilidad de pertenecer a una figura humana, esta alarma está acompañada de un sonido de alarma para captar la atención del usuario. También se guarda una imagen con el prefijo (Pr_Alta_) para que sea verificado por un operador luego. En la figura 5.2 se muestra la alarma para una imagen con un bajo grado de probabilidad de pertenecer a una figura humana esta no está acompañada de sonido alguno pero si se guarda una imagen con el prefijo (Pr_Baja_) para una posterior verificación.

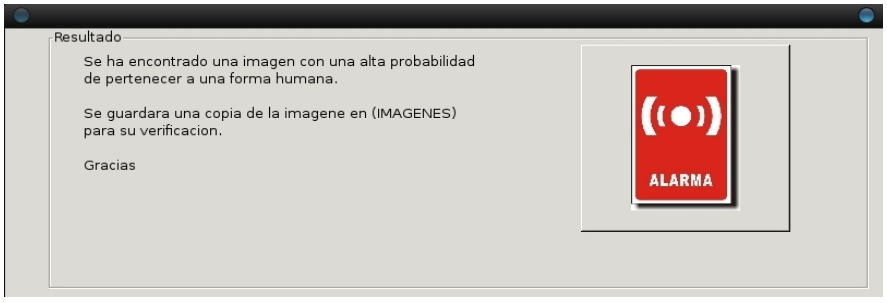


Figura 5.1 Alarma para un alto grado probabilidad.

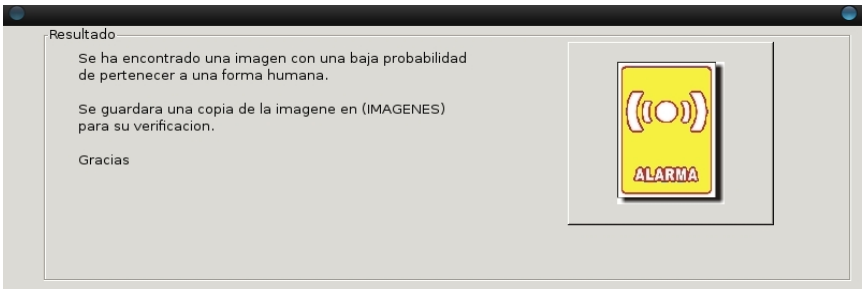


Figura 5.2 Alarma para un bajo grado probabilidad.

En la figura 5.3 se muestra el set de imágenes que muestra el sistema de una imagen que contiene una figura humana, la primera es una imagen en escala de grises, la segunda es la diferenciación entre el fondo y la figura. Por último está la imagen del contorno de donde sale la información que es ingresada a la red neuronal.

Figura 5.3 Imagen con silueta humana.

La figura 5.4 muestra el set de imágenes de una imagen que no contiene una figura humana.

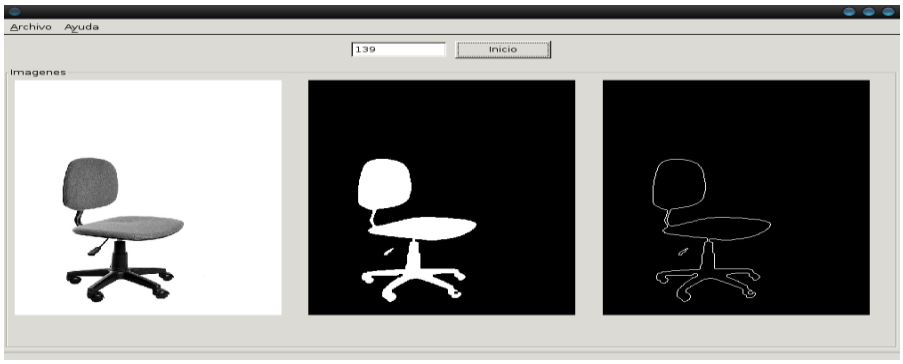


Figura 5.4 Imagen sin silueta humana.

El sistema ha probado que todas las figuras humanas generan una respuesta de la red neuronal por encima de 0.97 por debajo se encuentran las demás, pero es posible que una cierta colección de objetos puedan generar una falsa alarma, como aquellos que tenga parecido con una silueta humana o porque simplemente al recorrer el contorno de la imagen y crear el vector de direcciones este fue similar al de una silueta humana.

En la tabla 5.4 se muestra una tabla con la respuesta de la red neuronal a 15 imágenes. 5 imágenes de personas en posición de caminar, estos datos se muestran en las primeras 2 columnas; 5 de frente con los pies abiertos, estos datos se muestran en la 3 y 4 columna; y las demás de objetos no relacionados con una figura humana.

Nombre imágenes	Resultado	Nombre imágenes	Resultado	Nombre imágenes	Resultado
2.jpg	1	80.jpg	1	129.jpg	0.933336
10.jpg	0.987883	88.jpg	1	137.jpg	0.881704
12.jpg	1	92.jpg	1	138.jpg	0.86979
18.jpg	1	100.jpg	1	140.jpg	0.886426
23.jpg	0.988212	112.jpg	1	150.jpg	0.927546

Tabla 5.4 Respuestas a distintas imágenes.

Como se muestra con esta información es posible determinar el valor del punto en el cual claramente se diferencian las siluetas humanas de las siluetas de otros objetos. Pero esta información también permite cambiar el valor de ese punto para aumentar la probabilidad de detectar más formas humanas aunque de esta manera también aumente el número de falsas alarmas.

5.3 Comunicación cliente-servidor.

Anteriormente se explicaron las razones por las cuales se decidió desarrollar el sistema para dos arquitecturas distintas, la aplicación cliente-servidor permite tener un servidor central encargado exclusivamente de manejar la red neuronal y un gran número de clientes que se encargan del análisis de la imagen. Pero era necesario decidir la forma en que el cliente se comunicaría con el servidor y viceversa.

Gracias a experiencias previas se decidió que para la comunicación entre las dos aplicaciones lo mejor era usar mensajes en formato xml, que son fáciles de construir y de los cuales es fácil recuperar la información. Par la implementación de esta parte se utilizó una librería que es fácil de usar y cuenta con una gran documentación, además de ser liberada bajo una licencia no privativa.

A continuación se muestra un ejemplo del tipo de mensaje que se envía desde el

cliente al servidor, este contiene la información de las 8 entradas de la red neuronal.

```
<?xml version="1.0"?>
<rna>
        <f1>0.90000</f1>
        <f2>0.91000</f2>
        <f3>0.92000</f3>
        <f4>0.93000</f4>
        <f5>0.94000</f5>
        <f6>0.95000</f6>
        <f7>0.96000</f7>
        <f8>0.97000</f8>
</rna>
```

El siguiente ejemplo es el de un mensaje devuelto por el servidor a uno de sus clientes, contiene la respuesta de la red neuronal y un campo que en el futuro puede ser utilizado para identificar a qué tipo de posición individual pertenece.

```
<?xml version="1.0"?>
<clienterna>
        <res>0.98</res>
        <pos>1</pos>
</clienterna>
```

5. CONCLUSIONES

Se crea un sistema basado en redes neuronales para la detección de figuras humanas en imágenes. Este fue probado usando imágenes que contenían tanto personas en posición de caminando y de frente, y de otros objetos. Por ser un prototipo este sistema todavía no actúa de manera autónoma, por otro lado la red neuronal ya es funcional lo que permite su transporte a otros equipos para comenzar a trabajar aunque si por necesidades particulares el sistema necesita otro tipo de entrenamiento simplemente se requiere de una buena cantidad de imágenes para iniciar el entrenamiento.

El hecho de que el sistema trabaje con herramientas del software libre que se encuentran en distintos sistemas operativos permite que la aplicación pueda ser migrada fácilmente entre sistemas operativos. Por otra parte al existir una versión que trabaja bajo la arquitectura cliente/servidor es posible trabajar con el servidor en un sistema operativo con sus clientes en otros sistemas, además por medio del uso de mensajes xml permitirá que en un futuro sea posible trabajar con clientes escritos en distintos lenguajes de programación usando un servidor común.

El método para diferenciar el fondo del objeto a analizar es muy bueno cuando se encuentra en un ambiente controlado con una iluminación constante pero trabaja pobremente cuando se generan sombras o hay cambio de iluminación. Para un futura versión ha de ser necesario encontrar un mejor método de segmentación, por el momento cuando la aplicación deba trabajar autónomamente debe hacerse en un ambiente muy controlado que no permita cambios bruscos en las condiciones necesarias para un optimo trabajo.

El trabajo con redes neuronales ha mostrado ser una buena opción al momento de crear un identificador de características, pero también ha mostrado ser una opción muy difícil de diseñar. Escoger el número de entradas, el número de neuronas de la capa oculta, el algoritmo de entrenamiento puede llegar a ser una tarea dispendiosa. Para futuras aplicaciones será necesario hacer uso de otro tipo de clasificadores como las maquinas de soporte vectorial que han probado ser más fáciles de configurar y trabajar con una menor cantidad de parámetros y tienden a generalizar mejor una menor cantidad de datos de entrenamiento como se muestra en ([Lin04]).

El sistema presenta problemas con imágenes que generen una cadena de códigos similar a la que genera una silueta humana aunque la imagen no tenga un parecido obvio con la figura humana, para trabajos futuros ha de ser necesario revisar como contrarrestar este problema.

A. REFERENCIAS DE LAS FIGURAS

A continuación se muestra la tabla con las referencias bibliográficas de los libros de los que se adquirieron las figuras incluidas dentro de la tesis.

REFERENCIAS BIBLIOGRÁFICAS	
Número de la figura	**Referencia**
1.1 1.2	R. Gonzalez and R. Woods, *Digital Image Processing*, USA: Prentice Hall, 2002.
1.3 2.1 2.2 2.3.2.4 2.5 2.6 2.7 2.8 2.9 2.10 2.11 2.12 2.13 2.14 2.15 2.16	R. Molina, *Introducción al procesamiento y análisis de imágenes digitales*, España: Universidad de Granada, 1998.
2.17 2.18 2.19 2.20 2.21	N. Kasabov, *Foundations of Neural Networks, Fuzzy Systems, and Knowledge Engineering*, UK: The MIT Press, 1998.
2.22 2.23	B. Coppin, *Artificial Intelligence Illuminated*, USA: Jones and Bartlett Publishers, 2004
3.1	M. Singh, A. Basu and M. Mandal, *"Human Activity Recognition based on Silhouette Directionality"*.

Tabla A.1 Referencias bibliográficas de figuras.

BIBLIOGRAFÍA

[Mon99] J. Monedero, "*Conceptos Fundamentales de Teoría de Imagen Digital"*, monografía, UPC, 1999.

[Mol98] R. Molina, *Introducción al procesamiento y análisis de imágenes digitales*, España: Universidad de Granada, 1998.

[FaK99] M. Friedman and A. Kandel, *Introduction to pattern recognition. Statistical, structural, neural and fuzzy logic approaches*, UK: Imperial College Press, 1999.

[GaW02] R. Gonzalez and R. Woods, *Digital Image Processing*, USA: Prentice Hall, 2002.

[GaW03] R. Gonzalez, R. Woods and S. Eddins, *Digital Image Processing using MATLAB*, USA: Prentice Hall, 2003.

[SBaM] M. Singh, A. Basu and M. Mandal, *"Human Activity Recognition based on Silhouette Directionality"*.

[Cop04] B. Coppin, *Artificial Intelligence Illuminated*, USA: Jones and Bartlett Publishers, 2004

[Car07] M. Carter, *Minds and Computers*, Scotland: Edinburgh University Press Ltd, 2007.

[VaB08] B. Verma and M. Blumenstein, *Pattern Recognition Technologies and Aplicaions: Recent Advance*, USA: IGI Global, 2008.

[Kas98] N. Kasabov, *Foundations of Neural Networks, Fuzzy Systems, and Knowledge Engineering*, UK: The MIT Press, 1998.

[VyC07] J. Valencia y M. Cañas, *"Registro de transeúntes en tiempo real utilizando un sistema de visión artificial sobre un ambiente controlado"*, Tesis, UTP, 2007.

[Gop97] S. Gopalakrishnan, *"Development of Web-Based Educational Modules for Testing VHDL Models of Digital Systems",* Thesis*,* Virginia Polytechnic Institute and State University, 1997.

[Lin04] W. Lin, *"A Case Study on Support Vector Machines Versus Artificial Neural Networks"*, Thesis, University of Pittsburgh, 2004.

[Nil00] J Nilsson, "*Inteligencia Artificial. Una nueva síntesis*", España: McGraw-Hill, 2000.

[Arb03] M. Arbib, *The Handbook of Brain Theory and Neural Network*, UK: The MIT Press, 2003.