Universidad Nacional Autónoma de México

Posgrado en Ciencia e Ingeniería de la Computación

Reconocimiento de espejos planos para automatizar el ensamble de concentradores solares

Tesis

Que para optar por el grado de

Maestro en ciencia e ingeniería de la computación

Presenta

Miguel Angel Robles Roldan

Tutor:

Dr. Tetyana Baydyk - CCADET, UNAM

Ciudad de México, México Enero 2017

A mis padres y hermanos por todo el apoyo incondicional que siempre me han brindado.

A Alejandra y Sayuri por soportarme y apoyarme todo este tiempo.

A los proyectos PAPIIT IT102814 y PAPIIT IN102014
que apoyaron la realización de esta tesis.

Agradecimientos

A mi tutora, la Dra. Baydyk por todas sus enseñanzas, consejos, correcciones y el enorme tiempo dedicado.

Al comité de sinodales: Dr. Ángel Fernando Kuri Morales, Dr. Francisco Javier García Ugalde, Dr. Ernst Kussul y Dr. Caleb Antonio Rascón Estebané, por sus invaluables comentarios y aportes a esta tesis.

Al CONACYT por los apoyos brindados, en específico para la asistencia al congreso CONISOFT 2016.

Al M.I. Ulises Martín Peñuelas Rivas, al Dr. Ricardo Torres Jardón, al Ing. Jorge Antonio Escalante González por el apoyo brindado para la realización de mi maestría.

A los ingenieros Wilfrido Gutiérrez López y Manuel García Espinosa del Centro de Ciencias de la Atmósfera y al mismo centro por las facilidades y apoyo en la realización de esta maestría.

A la UNAM y en especial al posgrado de Ciencia e Ingeniería de la Computación por todas las enseñanza proporcionadas.

A l@s compañer@s del posgrado, con los que tuve la oportunidad de intercambiar conocimientos.

Índice general

Resumen

Una de las energías renovables más abundantes es la energía solar y México es considerado uno de los países que cuentan con las mejores condiciones para aprovecharla. Sin embargo, esto no ocurre y es en gran medida por el alto costo que tiene la tecnología necesaria. Los concentradores solares son una de las principales herramientas para el uso eficiente de la energía solar. Con objeto de reducir los costos de producción, se pueden utilizar espejos planos en lugar de espejos parabólicos para la fabricación de concentradores solares. Sin embargo, es necesaria una reducción mayor de costos y esto puede lograrse por medio de la automatización de su manufactura y ensamble.

La visión computacional y las redes neuronales son herramientas importantes que han tenido un creciente uso en la automatización. Ambas se aplicaron en el presente trabajo utilizando la red neuronal Random Threshold Classifier (RTC) y su variante Random Subspace Classifier (RSC). Se realizaron diferentes pruebas con RTC y RSC utilizando imágenes en color para poder detectar espejos planos. Para RTC se utilizó un vector de entrada basado en la moda, mientras que para RSC se utilizó un vector de entrada basado en un histograma.

Las pruebas realizadas con RTC y moda requirieron un menor tiempo de procesamiento, sin embargo, las pruebas que usaron RSC y el histograma mostraron una mayor estabilidad. En ambos casos las pruebas realizadas muestran que, en la etapa de entrenamiento, la red converge rápidamente y los errores en la detección son menores de 10 %. Se espera que a corto plazo, la detección realizada, permita calcular la posición de los espejos dentro de su contenedor de manera que facilite la automatización del ensamble de concentradores solares.

Capítulo 1

Introducción

Actualmente el uso de energías renovables tiende a aumentar y, la energía solar es una de las más abundantes en el planeta. Sin embargo, no en todos los lugares del planeta puede ser aprovechada esta energía de manera redituable. Afortunadamente, México se encuentra entre los países considerados como más apropiados para utilizar la energía solar.

Una manera de mejorar la eficiencia en el aprovechamiento de la energía solar es con el uso de concentradores solares. Los concentradores solares dirigen los rayos de Sol en un área más reducida, de manera que la energía concentrada en este punto puede ser aprovechada como calefacción, para cocinar y para generar energía eléctrica. Si bien los concentradores solares representan una ganancia en eficiencia, la tecnología para fabricarlos es muy costosa debido a que requieren espejos con forma parabólica.

Existen diferentes propuestas que se han hecho para reducir dichos costos de fabricación, algunas de éstas propuestas se mencionan en el capítulo 3. Una de las técnicas que se han estudiado para reducir el costo que implica la construcción de los concentradores solares, es aproximar su superficie parabólica con pequeños espejos planos. Esta técnica sumada a la automatización en la fabricación representarían un enorme reducción en el costo de fabricación.

Dos herramientas muy útiles en la automatización (y en otras áreas) son: los sistemas de visión computacional y las redes neuronales. Los sistemas de visión computacional se detallan en el capítulo 4, éstos intentan reproducir las tareas que realiza la visión humana.

Una de estas tareas, que además es básica en la automatización, es el reconocimiento y clasificación de objetos. En este campo las redes neuronales artificiales están siendo ampliamente usadas actualmente. Las redes neuronales artificiales son modelos del funcionamiento del cerebro y se ahonda en este tema en el capítulo 5.

En este trabajo se presenta la simulación por software de la red neuronal artificial *Random Threshold Classifier* (Clasificador de Umbrales Aleatorios) y su aplicación en el reconocimiento de objetos en imágenes de color. Específicamente aplicada al reconocimiento de espejos planos que se utilizan en la construcción de concentradores solares, el objetivo es sentar las bases que permitan a corto plazo la automatización del ensamble de éstos.

Una descripción más completa del problema planteado se desarrolla en el capítulo 6, mientras que en los capítulos 7, 8 y 9 se detalla el principio de operación del Random Threshold Classifier, así como su software de simulación y la base de imágenes necesaria.

Finalmente, en los capítulos 10 y 11 se explica la metodología seguida en los experimentos realizados y las conclusiones obtenidas, respectivamente.

Capítulo 2

Energías Renovables

La organización mundial Renewable Energy Policy Network for the 21st Century (REN21) [1] reporta un importante crecimiento a nivel mundial en el uso y desarrollo de energías renovables. Un ejemplo de ello es aumento en el uso de energías renovables en la generación de energía eléctrica, en la Fig. 2.1 se muestra la evolución en el periodo 2004-2013. Este crecimiento se atribuye principalmente al interés global por reducir la cantidad de emisiones de gases de efecto invernadero.

Total Installed Capacity		2004	2005	2006	2007	2008	2009	2010	2011	2012	2013
Solar Photovoltaic	GW	2.6	3.1	4.6	7.6	13.5	21	40	71	100	139
Concentrating Solar Power	GW	0.4	0.4	0.4	0.4	0.5	0.7	1.1	1.6	2.5	3.4
Wind Power	GW	48	59	74	94	121	159	198	238	283	318
Bio Power	GW	39	41	43	45	46	51	70	74	78	88
Geothermal Power	GW	8.9	9.8	10	10.4	10.7	11	11.2	11.4	11.7	12
Hydro Power	GW	715	–	–	920	950	980	935	960	990	1,000

Growth Rates		2004	2005	2006	2007	2008	2009	2010	2011	2012	Estimation 2013
Solar Photovoltaic	GW	–	35%	32%	40%	44%	36%	48%	44%	29%	28%
Concentrating Solar Power	GW	–	5%	0%	23%	14%	24%	54%	31%	36%	26%
Wind Power	GW	–	19%	20%	21%	22%	24%	20%	17%	16%	11%
Bio Power	GW	–	4%	5%	6%	2%	10%	27%	7%	4%	12%
Geothermal Power	GW	–	0%	2%	4%	3%	3%	2%	2%	2%	3%
Hydro Power	GW	–	–	–	–	–	–	–	3%	3%	1%

Figura 2.1: Incremento en la capacidad de generación a partir de energías renovables a nivel mundial en el periodo 2004-2013 según la REN21

Aunque la generación por energía hidroeléctrica es la que cuenta con mayor capacidad instalada, la solar (tanto en rama fotovoltaica como en la de concentradores solares) es la que aparece con mayor crecimiento. Este crecimiento puede observarse de una manera más detallada en la gráfica de la Fig. 2.2 [2]. En esta gráfica se presenta la capacidad de generación de energía eléctrica a partir de concentradores

solares a nivel mundial en el periodo comprendido de 2004 a 2014. Es muy notable que al 2014, el líder en capacidad instalada es España, seguido por Estados Unidos y entre éstos dos acumulan casi la totalidad de la capacidad global.

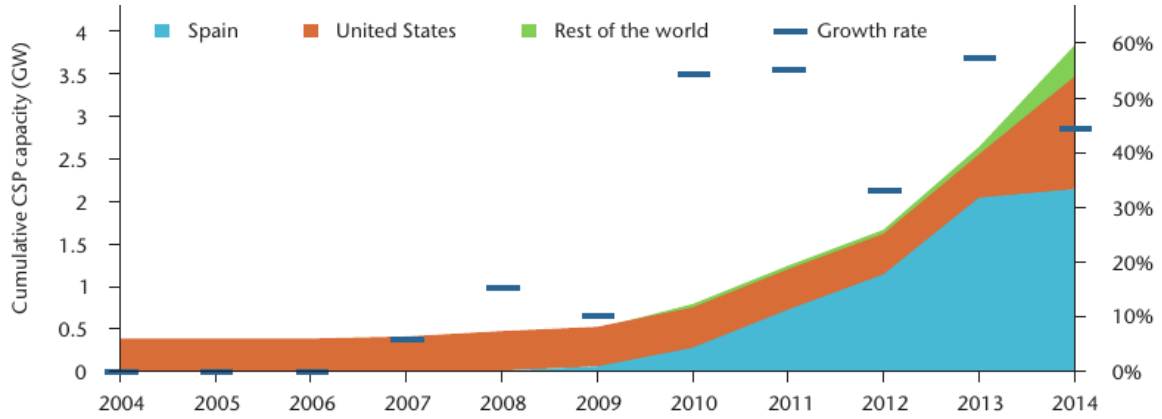


Figura 2.2: Crecimiento en la capacidad de generación de energía eléctrica a partir de concentradores solares (Concentrating Solar Plants) a nivel mundial

La energía solar es considerada la más abundante en el planeta con alrededor de 885 millones TWh en la superficie del planeta cada año [2]. Sin embargo, para poder aprovechar la energía es necesario evitar su dispersión en la atmósfera y el bloqueo por parte de nubes y aerosoles. Un parámetro que nos ayuda a determinar esto es la radiación directa normal. En la Fig. 2.3 podemos ver un mapa global de la radiación directa normal [3].

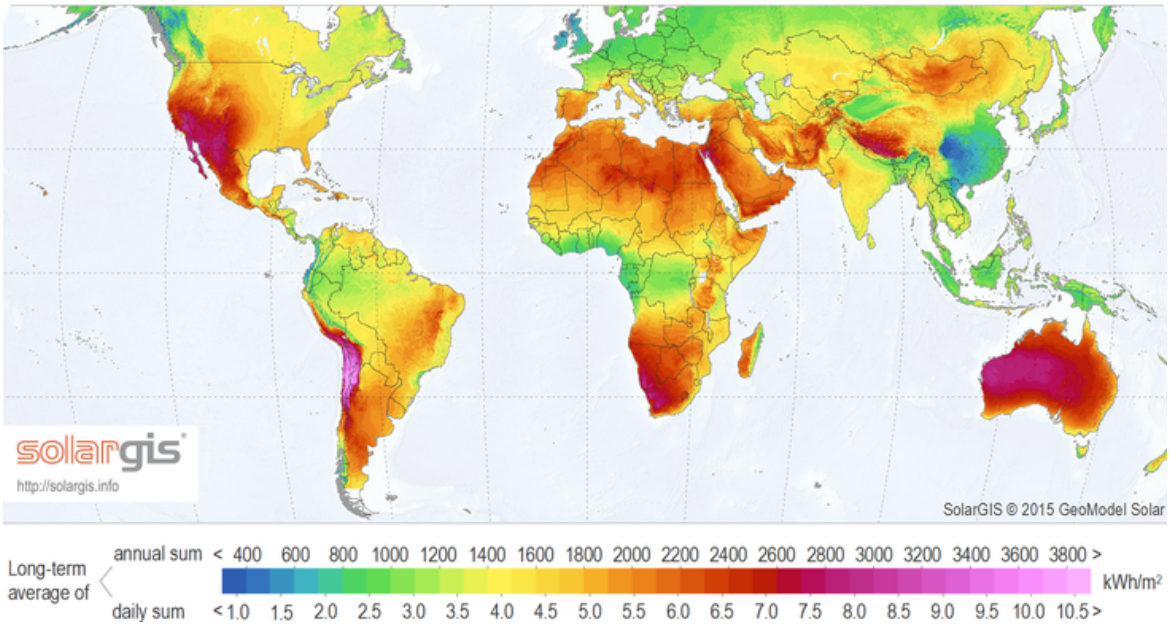


Figura 2.3: Mapa mundial de radiación directa normal

Las mejores zonas para aprovechar la energía solar por medio de concentradores se encuentran entre las latitudes 15 y 40 tanto norte como sur [2]. Se observa un alto nivel de esta radiación en los países

que encabezan la capacidad instalada de generación de electricidad a partir de energía solar (E.U. y España). Pero también se indica que, principalmente en la zona norte, México recibe cantidades de radiación similares a la de estos países.

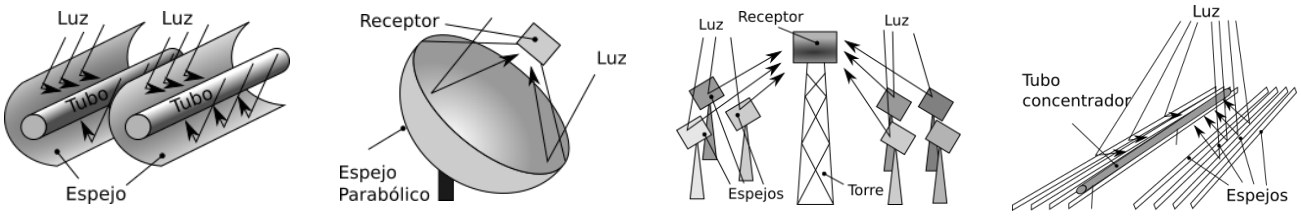
Tomando en cuenta lo anterior, podemos decir que la generación de energía eléctrica a partir de concentradores solares se encuentra en pleno apogeo. Además, dada la ubicación geográfica de nuestro país, éste se encuentra entre las regiones consideradas como más favorables para el uso de Concentradores Solares. Por lo tanto, el desarrollo de este tipo de tecnología en México promete ser muy redituable y no debe dejarse desperdiciar.

A continuación hablaré un poco más acerca de los concentradores solares y algunas de las tecnologías que se han desarrollado para la disminución de su costo.

Capítulo 3

Concentradores Solares

Los concentradores solares toman la energía del Sol y la concentran en un punto (receptor) que para generación eléctrica, en la mayoría de los casos, contiene un fluido que hace funcionar un generador eléctrico. Para otras aplicaciones en el receptor puede variar. Existen diferentes tipos de concentradores, los más comunes son: de canal parabólico, de disco parabólico, de torre de helióstatos y concentrador lineal de tipo Fresnel (ver Fig. 3.1).



(a) Canal parabólico (b) Disco parabólico (c) Torre de helióstatos (d) Tipo Fresnel

Figura 3.1: Tipos de concentradores solares

Los concentradores de canal parabólico utilizan grandes espejos en forma de 'u' colocados uno junto a otro en grandes líneas que siguen el Sol a lo largo del día (Fig. 3.1a). Los concentradores de disco parabólico son conformados por espejos en forma de disco que se montan en un soporte capaz de seguir el Sol a lo largo del día (Fig. 3.1b). Las torres de helióstatos constan de una gran cantidad de espejos planos que siguen al Sol de manera que reflejen sus rayos en lo alto de una torre (Fig. 3.1c). Finalmente, los concentradores de tipo Fresnel se componen de una gran cantidad de espejos planos, largos y delgados que reflejan los rayos del Sol hacia un receptor igualmente largo (Fig. 3.1d).

Una manera de reducir los costos es con diseños de tipo caseros como el desarrollado por Judith Franco et al. [4]. Ellos presentaron el diseño de un concentrador solar de bajo costo para ser utilizado en la pasteurización de la leche. El control de temperatura es importante, ya que se debe tener más de $62°C$ para inactivar a los microorganismos dañinos pero no debe pasar de $65°C$ por más de 5min para evitar la pérdida de las propiedades de la leche.

El concentrador se construyó utilizando conos de aluminio y colocándolos de manera consecutiva. Es fabricado manualmente y tomando como base uno construido previamente para ser usado como

cocina solar. El soporte es un rectángulo metálico horizontal que permite el movimiento vertical del concentrador. Asímismo tiene ruedas que le permiten moverse y ser orientado de manera adecuada. El rastreo es efectuado de manera manual por el operador, se calcula que se debe reajustar la posición aproximadamente cada media hora.

Otro diseño de este tipo es el presentado por Escobedo et al. [5]. Este concentrador solar es plano y cuenta con un control electrónico de tres grados de libertad. Uno permite el movimiento azimutal, otro el movimiento en elevación y el último permite el ajuste manual de la inclinación del concentrador. Para el diseño del controlador de posición se utilizó la ecuación del ángulo solar. La estructura del prototipo está compuesta por un eje que realiza el movimiento de seguimiento. Tiene una base que tiene dos brazos fijos y dos brazos de longitud variable, que sirven para ajustar la inclicación del prototipo de acuerdo a la inclinación del lugar.

El concentrador consta de un espejo de $60x60cm^2$. El motor que realiza el movimiento del espejo es uno que se utiliza para el movimiento de antenas de televisión. Se realizó un sistema de control basado en un dsPIC que permite la programación de la hora de inicio por medio de botones y un LCD.

Este tipo de concentradores son de bajo costo pero su eficiencia es poca y en general no son susceptibles de producción en masa, dado que muchas veces son fabricados reutilizando partes.

Una propuesta más comercial enfocada al uso en celdas fotovoltáicas es la hecha por Nina Khuchua et al. [6]. Diseñaron un concentrador solar basado en capas de lentes y reflectores. Consta de cuatro capas rectangulares: la primera es una arreglo de lentes, la segunda y tercera son capas reflectivas que se encargan de enfocar y concentrar los rayos; la última capa sirve como sustrato para proveer enfriamiento.

En este caso el diseño no es nada casero, por el contrario requiere de lentes especiales que seguramente elevan el costo de manera considerable.

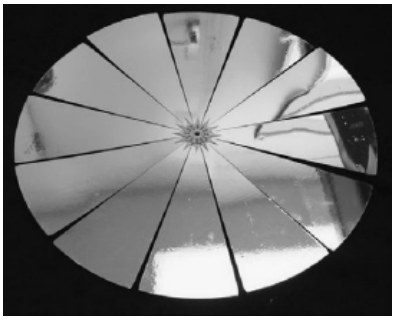
También existen algunos diseños para ser utilizados con diferentes tecnologías de conversión, como es el caso del publicado por M. Vivar et al. [7]. Su prototipo está diseñado especialmente para la India, lo que incluye un análisis climático para determinar las zonas aptas y no aptas para su instalación. El prototipo consiste de tres espejos y tres receptores diseñados para proveer un sistema multifuncional apropiado para receptores fotovoltáicos, térmicos, híbridos y químicos.

Se utilizó un receptor combinado en la posición central, mientras que los receptores laterales son térmicos. El área del colector es de $6.6m^2$ que corresponden a tres espejos de $2.2m^2$. Se calcula una potencia de salida de 220 W de potencia eléctrica y 3507 W de potencia térmica.

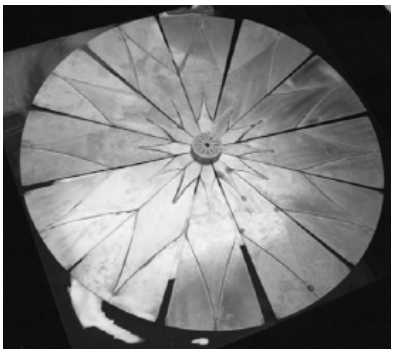
Finalmente, existen prototipos pensados para ser diseñados en masa, con buena eficiencia y disminución de costos. A continuación mencionaré tres de ellos.

Lifang Li et al. [8] diseñaron un concentrador de disco parabólico construido a partir de pétalos de material flexible y con una superficie altamente reflectiva. Para aproximar de manera más adecuada la forma parabólica el grosor del pétalo debe cambiar en diferentes zonas. Variar el grosor de cada pétalo es impráctico en cuestiones de fabricación, por lo que se aproxima esta variación fabricando pétalos hechos de varias capas. Cada capa tiene una forma especialmente diseñada para simular la

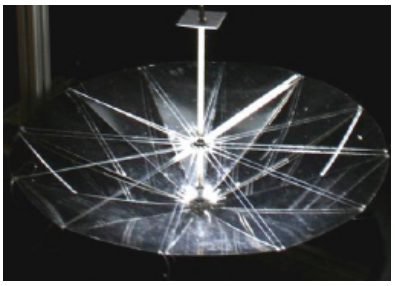
forma parabólica de manera óptima. Cada capa que conforma los pétalos es plana, por lo que es fácil de transportar y una vez en el punto de instalación debe de armarse. Dado que el material es flexible y plano, para mantener la forma, cada pétalo debe ser conectado a los demás por medio de cables (Fig. 3.2).



(a) Vista frontal



(b) Vista trasera

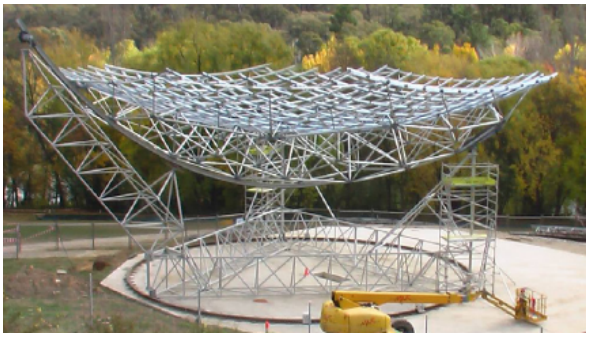


(c) Concentrador armado

Figura 3.2: Concentrador solar de material flexible

La Australian National University (ANU) publicó, en 2011 [9], el diseño de un concentrador solar de disco parabólico de $500m^2$ como una versión susceptible a producción comercial de un concentrador de $400m^2$ previamente diseñado. El nuevo diseño incluye un soporte especial fabricado a partir de una plantilla que permita la fabricación en masa. La construcción comenzó con la preparación de una bloque de concreto un poco mayor a los $500m^2$ que permitiera una superficie estable para la construcción de la plantilla. La plantilla se encuentra conformada por varillas parabólicas sobre soportes ajustables conformando una malla cúbica. Sobre la plantilla se armó y soldó el soporte y una vez terminado se separó de la plantilla (Fig. 3.3a) y se colocó sobre una base triangular (Fig. 3.3b). La base permite el movimiento necesario para el rastreo a partir de motores eléctricos de AC.

(a) Soporte y su plantilla

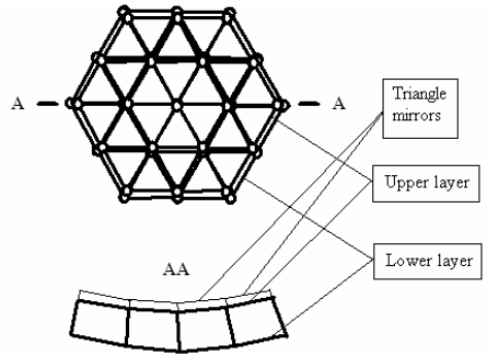


(b) Soporte colocado sobre su base

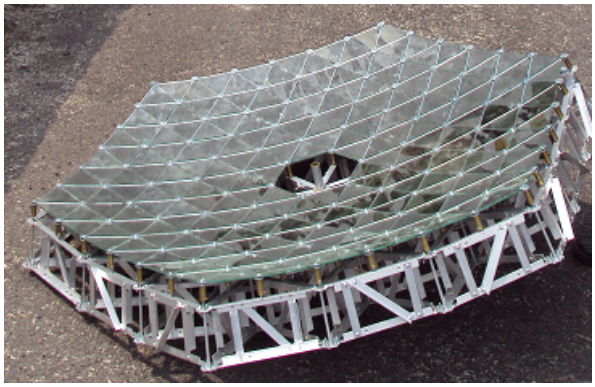
Figura 3.3: Concentrador solar diseñado por la Australian National University

Sobre el soporte se colocaron uno a uno los espejos y un soporte con forma de tetrahedro que permite una carga de hasta 2000 Kg para el receptor. Este concentrador tiene una distancia focal de 13.4m y un sistema de rastreo altitud-azimutal; utiliza 380 paneles con espejos esféricos de 1.17 m x 1.17 m. El prototipo será utilizado en investigación de conversión de energía con diferentes tecnologías.

Por último Kussul et al. [10] han demostrado que es posible construir un concentrador de gran eficiencia y costo bajo a partir de espejos planos triangulares, como el que se muestra en la Fig. 3.4. El diseño consta de dos capas de celdas triangulares. Cada celda se compone de tres barras y tres nodos. Las celdas de la capa baja son más grandes que las celdas de la capa superior pero se encuentran unidas con varias barras con la misma longitud. Eligiendo de manera adecuada los tamaños es posible obtener una forma parabólica de la capa superior.



(a) Diseño del concentrador



(b) Foto del concentrador solar

Figura 3.4: Concentrador solar formado por espejos triangulares

Si bien el costo calculado por Kussul et al. es bajo comparado con costos de otros concentradores solares [10], es posible bajar aun más el precio utilizando un sistema automático de manufactura y ensamble. Por esta razón en las secciones siguientes hablaré de dos herramientas muy utilizadas en la automatización: sistemas de visión computacional y redes neuronales artificiales.

Capítulo 4

Sistemas de visión computacional

La visión computacional se define como una rama de la inteligencia artificial y el procesamiento de imágenes que tiene que ver con el procesamiento de imágenes del mundo real [11]. La visión computacional intenta emular a la visión humana.

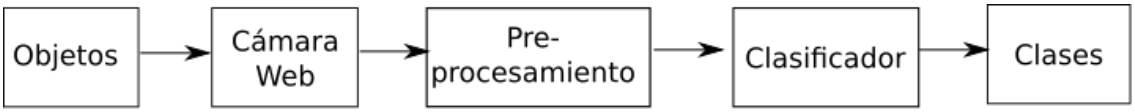


Figura 4.1: Estructura general de un sistema de visión computacional

El diagrama de bloques de un sistema de visión computacional básico se muestra en la Fig. 4.1. Un sistema de visión computacional básico consiste, en primer lugar, de una cámara para poder detectar la luz y convertirla en una imagen digital. Una vez que la imagen ha sido obtenida, se pasa por una etapa de pre-procesamiento que sirve para resaltar o extraer sus características. Como resultado de la etapa de pre-procesamiento se genera información relevante (características) de la imagen. Finalmente, las características son entonces alimentadas a un clasificador para que pueda separar la imagen en clases u objetos.

Los sistemas de visión computacional pueden ser bastante económicos y versátiles, ya que el hardware necesario se reduce a una cámara y una computadora. Actualmente existe una amplia variedad, tanto de computadoras como de cámaras, lo que se ve reflejado en una gran facilidad en su adquisición. Por otro lado la computadora, que por lo general es el componente más caro, no necesariamente tiene que estar dedicado únicamente al proceso de visión sino puede aprovecharse para otras actividades, ya sea de manera paralela o en diferentes momentos. Es por esto que los sistema de visión computacional son utilizados en diversas áreas, a continuación se mencionan algunos ejemplos.

Un sistema genérico para inspección automática de texturas fue propuesto por T.A. Mitchell y M. Sarhadi [12]. En este modelo, a la imagen se le aplican diferentes filtros, cada uno de los cuales corresponde con una característica de textura. Una vez aplicadas las máscaras, son rectificadas y suavizadas. Finalmente se les aplica una regla de clasificación para segmentar la imagen. Para elegir las máscaras adecuadas se utilizó un algoritmo iterativo en el que en cada iteración existe una máscara actual, una

nueva máscara aleatoria y una máscara que es un promedio ponderado de las dos. La máscara que brinda la mejor separación de texturas se convierte en la máscara para la siguiente iteración. Esto se repite por un número predeterminado de iteraciones. La separación de texturas es calculada como la relación de las energías de texturas generadas para cada región. Este proceso tiene el incoveniente de requerir de muchas iteraciones (alrededor de 1000). Los sistemas de visión computacional son utilizables en aplicaciones al aire libre como lo es la recolección agrícola. A continuación se describen dos ejemplos.

Xiangqin Wei et al. [13] desarrollaron un método de extracción de objetos para un robot recolector de fruta. Se utilizaron fotografías tomadas a cuatro tipos de frutas en su periodo de madurez, bajo condiciones de luz natural. Para cada fruta se tomaron 20 imágenes realizando variaciones en el fondo.

En el espacio de color OHTA se definen parámetros básicamente a partir de la diferencia de las componentes RGB. La segmentación propuesta se basa en la elección del mejor de éstos parámetros y el umbral correspondiente. El operador con el que se obtuvieron los mejores resultados fue $I_1' = R - G$, logrando un reconocimiento mayor al 95 %.

Por otro lado, Kanae Tanigaki et al. [14] desarrollaron un robot con sistema de vision 3D para recolectar cerezas. El robot consiste de un manipulador con 4 grados de libertad, un sensor de vision 3D, un efector, una computadora y un dispositivo de desplazamiento. El sensor de visión 3D está colocado en el manipulador para que le permita explorar desde diferentes puntos vista. Como parte del sensor de visión se utilizó un rayo infrarojo de 830 nm para detectar un árbol de cereza y un rayo de 690 nm para detectar el fruto rojo (Fig. 4.2).

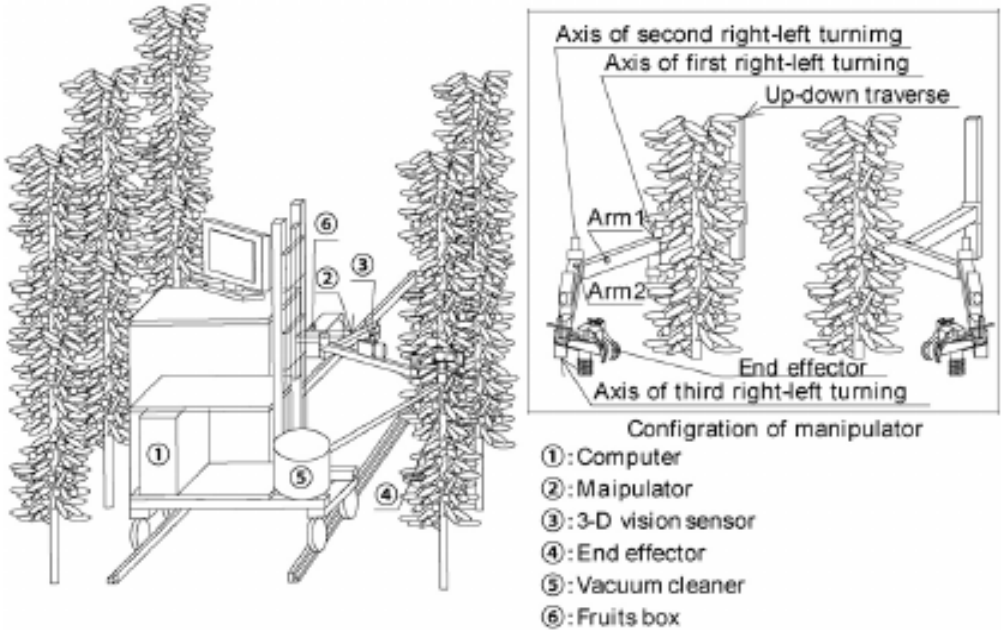


Figura 4.2: Sistema de Visión Computacional para automatizar la detección de cerezas

Para detectar el centro de la cereza se utiliza la reflexión especular que ésta produce. Al ser esférica, el fenómeno de luz especular es mayor cuando el rayo ilumina el centro de la cereza. Sin embargo,

esté efecto no se da si la cereza se encuentra oculta entre las hojas. En pruebas realizadas en un árbol con 80 cerezas se detectaron 47 (59 %).

De los ejemplos anteriores es notable que la detección no es muy eficiente, esto es debido a las diversas condiciones del ambiente. Es por esto que la mayoría de las veces se eligen medios con condiciones controlables. A continuación mencionaré dos ejemplos que entran en este rubro.

Emanuelle Morais de la Oliveira et al. [15] desarrollaron un sistema de visión computacional para clasificar granos de café de acuerdo a su color. El sistema consiste de: una caja de metal para eliminar influencia luminosa del exterior; una cámara digital de 10Mpx de resolución instalada a 40 cm sobre el plano de muestreo; un sistema de iluminación con lámparas de LED de 3W, también a 40cm pero con un ángulo de 45°; y una computadora personal (Fig. 4.3).

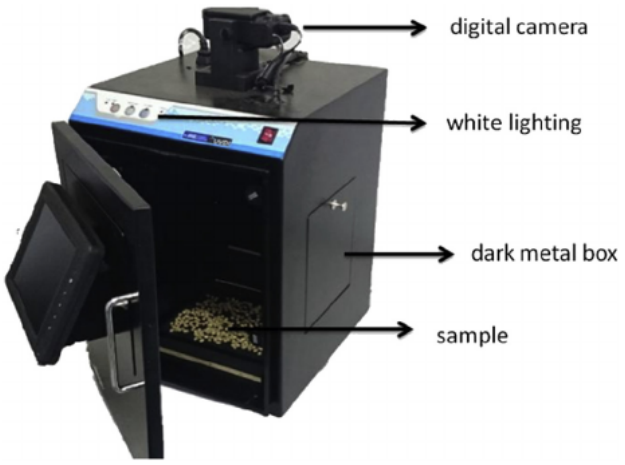


Figura 4.3: Sistema de visión computacional utilizado para clasificar granos de café

Para obtener las imágenes se colocaron los granos correspondientes a una clase en la superficie negra. Posteriormente se removió el fondo por medio de una segmentación por umbral. Una vez que únicamente existen los granos en la imágen, se procedió a convertir su color al espacio CIELab por medio de tres redes neuronales y dicho color se alimentó al clasificador. Las clases requeridas fueron cuatro: whitish, cane green, green y bluish green. Para las pruebas se escogieron 30 granos por color de cada una de las clases, agrupandose de manera notoria en el espacio CIELab.

Una aplicación muy similar a la anterior fue desarrollada por Sahameh Shafie et al. [16]. Su objetivo fue la caracterización no destructiva de la miel basada en el color y su correlación con sus atributos químicos. El sistema de visión computacional consta de cuatro partes: una cámara oscura, un sistema de iluminación, una cámara digital y una computadora (Fig. 4.4).

En la etapa de preprocesamiento se recortó la imagen extrayendo la parte central y se removieron las burbujas y la cera.

En este caso se utilizaron cinco espacios de color: HSV, YIQ, CIELab y YCbCr. Fueron necesarias dos redes neuronales: una para calcular las componentes CIELab; y otra realizar la caracterización de la miel. Las entradas de esta última fueron las 15 componentes calculadas y como única salida la clase a la que correspondía (TPC, AA o AC).

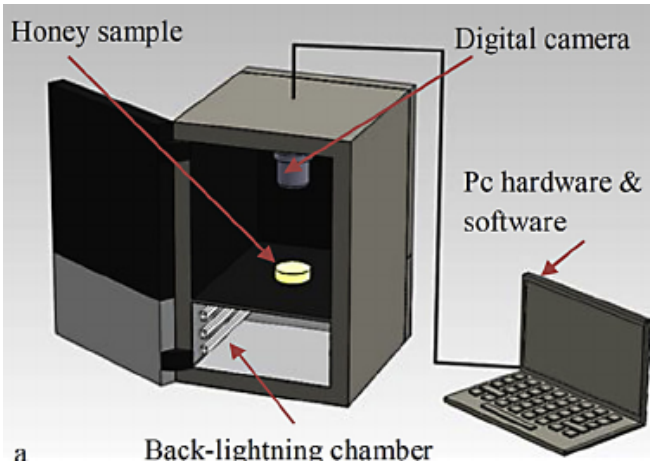


Figura 4.4: Sistema de visión computacional para caracterización no destructiva de la miel

Aunque tener condiciones controlables es altamente deseable para obtener buenos resultados a partir de visión computacional, no siempre es posible y en ocasiones puede entorpecer los procesos (de producción, análisis, recolección, etc.).

En general, un tipo de sistema de visión computacional más eficiente es el que permite tener un balance entre el control de las condiciones y la intervención en el proceso. A continuación mencionaré tres sistemas que caen en esta categoría.

H Celik et al. [17] proponen un sistema que permite la detección y clasificación de cuatro tipos de defectos en telas utilizando redes neuronales. El sistema consiste de una cámara y un sistema de iluminación montados sobre una banda en la que corre la tela (Fig. 4.5).

Figura 4.5: Sistema de visión computacional para detectar defectos en telas

El pre-procesamiento tiene varias etapas, entre ellas se encuentran filtrado, umbralización y operaciones morfológicas para finalmente extraer características mediante la matriz de co-ocurrencia.

El vector de características utilizado para alimentar la red neuronal tiene una longitud de 32. La red neuronal consta de 37 neuronas en la capa oculta y cuatro neuronas de salida (una para cada clase). Para la etapa de entrenamiento se utilizaron 25 imágenes con defectos de cada tipo. El entrenamiento se realizó por medio de backpropagation. Para la etapa de prueba se utilizaron 20 muestras de cada tipo de defectos, obteniendo un promedio de precisión en la clasificación del 96.3 % y del 94.38 % en la detección de defecto.

Joao Cunha et al. [18] desarrollaron un sistema de vision computacional y manipulación robotica para alimentar de manera automática perforadores de corcho. El sistema consta de una banda, un brazo robótico, una herramienta de sujeción, un sistema de visión y una unidad de proceso (Fig. 4.6).

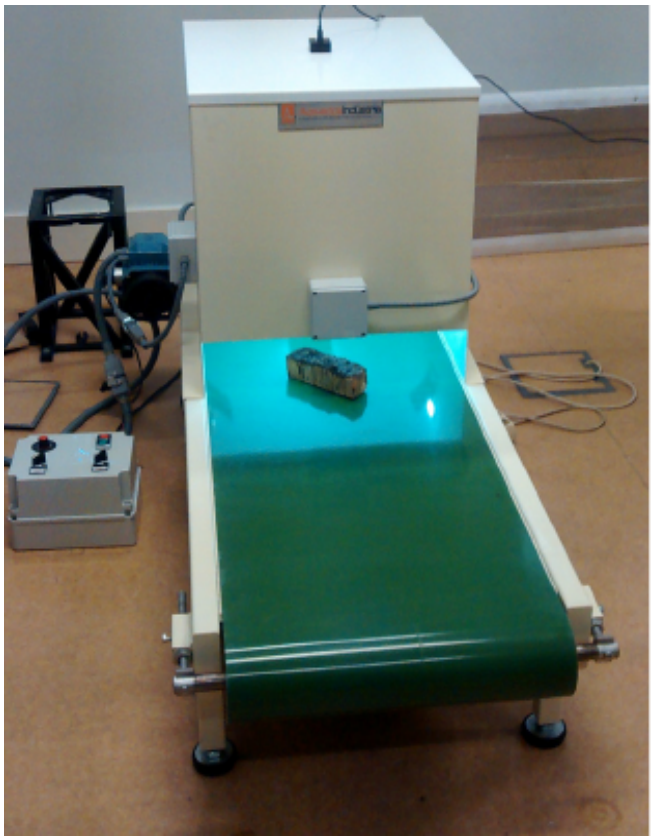


Figura 4.6: Sistema de visión computacional para alimentar una perforadora de corcho

La banda transporta las tiras de corcho y es de un único color que contrasta con el corcho. El brazo robótico debe tomar las tiras de corcho y colocarlas dentro del perforador. Se utilizó el 4 Dof SCARA robotic ARM de OMRON. Al brazo se le agregó una herramienta que le permitiera tomar las tiras y rotarlas de manera que sean colocadas de manera correcta. Esta herramienta funciona por medio de aire a presión, lo que permitetomar el corcho sin dañarlo. El sistema de visión consta de una cámara y un sistema de iluminación. Una computadora personal es utilizada como unidad de proceso. Es responsable de procesar las imágenes capturadas, extraer la información de la posición y orientación de las tiras. Así como del movimiento del brazo robótico y la herramienta sujetadora.

Para lograr la extracción de la tira de corcho en la imagen se definieron tres etapas: en la primera se construye un modelo del fondo, en la segunda se extrae el fondo para obtener únicamente el corcho, y la tercera valida que se esté detectando de manera correcta la tira de corcho.

El espacio de trabajo elegido fue el HSV y escala de grises.

Para cada canal se calcula la matriz de co-ocurrencia (GLMC). A partir de esta matriz se obtuvo la energía, la correlación, la homogeneidad, el contraste y la entropía. Adicionalmente para cada componente se calculó el promedio y la varianza. Estas características son alimentadas a un clasificador. Se utilizó un clasificador del tipo Support Vector Machine (SVM). Se entrenó un SVM para cada clase y la clase es determinada por el SVM que tenga una mayor respuesta.

La base de imágenes fue dividida en aproximadamente 100 imágenes para entrenamiento y aproximadamente 30 para prueba. Se tomaron en cuenta tres clases: belly, back y side. En la etapa de entrenamiento se realizó una búsqueda sobre los parámetros obtenidos para encontrar los óptimos. Como resultado, en la etapa de entrenamiento todas las imágenes fueron clasificadas de manera correcta.

Por último, describiré un sistema que permite la clasificación de engranes sobre una banda propuesta por Wenqui Wu et al. [19]. El sistema consiste de una cámara, una fuente de luz LED, seis sensores foto-eléctricos, una computadora, cinco actuadores neumáticos y una bomba de aire. Al inicio de la banda se encuentran unas guías que colocan los engranes en una fila. La cámara se encuentra montada sobre la banda. Y la fuente de luz tiene forma de anillo y se encuentra colocada debajo de la cámara para proveer la iluminación necesaria (Fig. 4.7).

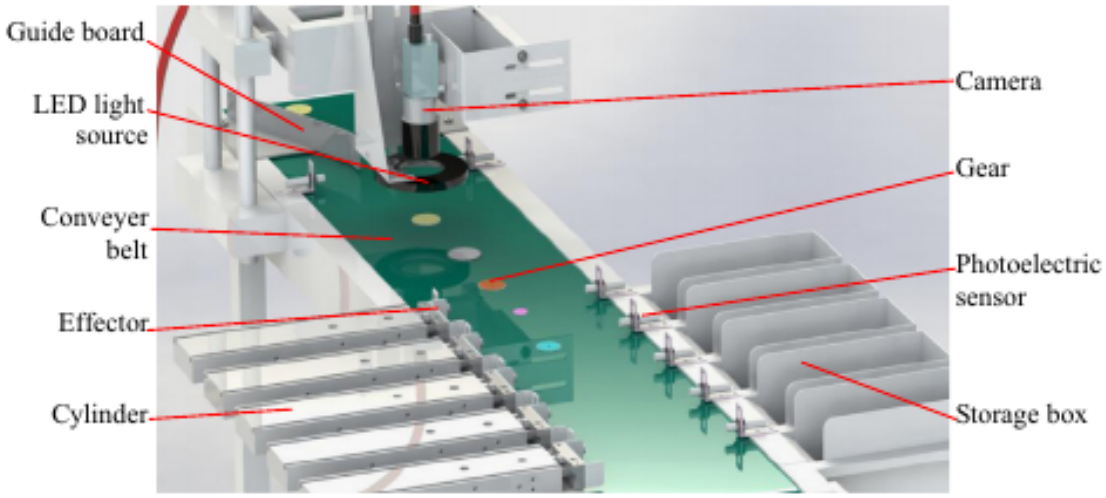


Figura 4.7: Esquema del sistema de ordenamiento de engranes

En cuanto a los sensores, uno de ellos es colocado de manera que detecte al engrane y provoque el disparo de la cámara. Los otros cinco sensores indican la posición de los contenedores e indican que actuador se activará para colocar al engrane en el contenedor adecuado. La computadora se utiliza para disparar la cámara y procesar la imagen, así como para la activación de los actuadores. Las características a detectar de los engranes son: el número de hoyos, el número de dientes y el color; extrayendo cada una por separado.

Para la detección de color, primero se separa al engrane del fondo. Después se calcula el promedio del color y se asigna a una de las clases (color de engrane) utilizando la distancia euclidiana mínima.

Para la extracción del número de hoyos primero se obtiene la imagen en escala de grises. Se realiza una segmentación por umbral para separar el fondo y se elimina el ruido por medio de la operación cerradura. Por medio del algoritmo Floodfill se obtiene una nueva figura en la cual se han eliminado los hoyos. A esta figura sin hoyos se le resta la figura con hoyos, de manera que el resultado son únicamente los hoyos. Finalmente se utiliza un algoritmo de etiquetado de regiones con conexión 8, obteniendo así el número de regiones conectadas (hoyos).

Para la extracción del número de dientes se utiliza la imagen sin hoyos previamente obtenida. Se aplican las operaciones morfológicas de cerradura y erosión con el fin de eliminar ruido y resaltar la forma del contorno respectivamente. Se realiza una operación XOR entre las dos imágenes, obteniendo de esta manera el contorno del engrane. Finalmente se calcula la distancia que existe entre el contorno y el centro de manera que si se grafica la orientación contra la distancia se obtiene una gráfica con picos que corresponden a los dientes.

Los ejemplos anteriores denotan la inmensa variedad de aplicaciones que tienen los sistemas de visión computacional. También cabe resaltar que en varios de estos ejemplos, las redes neuronales son parte fundamental permitiendo resultados excelentes. Es por esto que se ha elegido su uso en el desarrollo del presente trabajo y en la sección siguiente abordaré dicho tema.

Capítulo 5

Redes Neuronales artificiales

Para poder realizar los objetivos planteados por la visión computacional existen diversas técnicas que se emplean: algoritmos genéticos, sistemas difusos, redes neuronales artificiales, etc.

Las redes neuronales artificiales (ANN) son modelos del funcionamiento del cerebro. Actualmente, por lo general, esto implica una simulación por software. Mediante software se crean representaciones de unidades básicas llamadas neuronas artificiales. Las neuronas artificiales se conectan entre sí para formar una red. La intención de esta red es recibir una señal de entrada, realizar un proceso interno y entonces obtener una señal de salida como respuesta a la entrada.

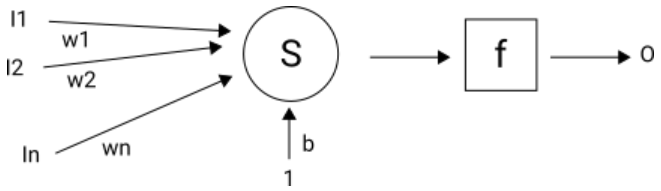


Figura 5.1: Estructura clásica de una neurona

Una representación de la estructura clásica de una neurona artificial se puede observar en la Fig. 5.1. Una neurona clásica consiste de las siguientes partes: señales de entrada l, son las conexiones provenientes de otras neuronas o la señal de interés; conexiones con pesos w, estas conexiones son modificadas permitiendo a la red aprender por medio de un proceso de entrenamiento; unidad de acumulación S, que suma las entradas de la neurona; una función de excitación f, que en general sirve como un método de umbralización que se aplica a la salida de la unidad de acumulación, tiene asociado un offset b; salida O, es la respuesta de la neurona y corresponde con el resultado de la función de excitación.

Para poder entrenar una ANN se realiza un proceso iterativo en el que se ajustan los pesos de las conexiones a partir de un valor inicial dado. Este proceso se repite hasta que se alcanza algún criterio de convergencia o de paro. Por lo general el criterio de paro es un número determinado de iteraciones o cuando la cantidad de errores disminuye por debajo de cierto límite.

Las ANN son utilizadas en una gran variedad de aplicaciones, a continuación se resumen algunos trabajos a modo de ejemplo.

Una aplicación muy interesante es en el reconocimiento del habla, en este campo Jin Kyung Ryeu y Ho Sun Chung [20] propusieron una red neuronal en la que utilizaron como entrada los coeficientes Mel-Frequency Cepstrum (MFCC) correspondientes a monosílabas coreanas. Los pesos de la red se inicializan con valores aleatorios. La red corre por un periodo de tiempo bajo las condiciones iniciales y dadas las señales de entrada. En este periodo los pesos de las conexiones de la red son fijos y el error es acumulado. El error total de la salida es tomado por la función de aprendizaje $E(W) = \int_{T1}^{T2} \sum_{i=1}^{N} \frac{1}{2}(X_i(t) - Q_i(t))^2$, donde Q es la señal correcta y X es la salida de la ANN.

Los pesos se cambian utilizando el metodo de pasos descendentes, en el cual se disminuye el valor de los pesos conforme el error calculado por la regla de aprendizaje. Si el error total es menor que un umbral se termina el entrenamiento, de otro modo se modifican los pesos utilizando una función multiplicativa. En las pruebas realizadas se reporta un reconocimiento por arriba del 86 %.

Otra aplicación muy interesante es el uso en imágenes en movimiento. Osman Günay y A. Enis Cetin [21] propusieron un método para reconocer texturas dinámicas en tiempo real utilizando proyecciones en hiperplanos aleatorios y redes neuronales profundas. Las texturas dinámicas son regiones con imágenes en movimiento como flamas, humo, olas, etc. La mayoría de los métodos utilizados para esta tarea tienen requerimentos computacionales muy altos para poder ser utilizados en tiempo real. Una manera de reducir la carga de procesamiento es utilizar proyecciones en hiperplanos aleatorios y otro el uso de filtros basados en redes neuronales para reducir el tamaño del vector de características. Los resultados obtenidos muestra un mejor desempeño utilizando el muestreo aleatorio de hiperplanos que al usar redes neuronales pero se atribuye al reducido grupo de muestras de entrenamiento que se utilizaron.

Un uso más práctico fue realizado por Ikramullah Khosa y Eros Pasero [22], quienes desarrollaron una red neuronal para clasificar nueces. El objetivo era identificar las nueces no saludables por medio de imágenes de rayos X. El primer paso fue obtener las imágenes en rayos X de nueces compradas en el mercado. Se analizaron las nueces y se etiquetaron de forma manual las imágenes correspondientes.

Para cada imagen se calcularon las siguientes características: media, desviación estandar, uniformidad, tercer momento, suavidad y entropia, Además de cuatro matrices de co-ocurrencia.En total se utilizaron 44 características como entradas de la red neuronal. Como salida se usó únicamente una neurona que nos devuelve la clase reconocida. La red neuronal tiene una capa oculta de 16 neuronas y se usó una función de activación sigmoidal tanto en la capa oculta como en la capa de salida. Para la etapa de entrenamiento se implementó el algoritmo de Levenberg-Marquardt observando resultados positivos en mas del 95 % de las pruebas.

Un ejemplo de aplicación en el área de clasificación es la red propuesta por Amel Hebboul et al [23] (INNCC). Ellos proponen un modelo auto-organizado de aprendizaje semi-supervisado e incremental y no impone restricción en la estructura de la red neuronal.

Un entrenamiento semisupervisado es una combinación de entrenamiento supervisado y no supervisado. Aprendizaje incremental es la habilidad para aprender nueva información sin necesidad de eliminar lo ya aprendido. Actualizando el modelo de manera continua por medio de un aprendizaje competitivo.

Las reglas de construcción de la red neuronal son: una nueva neurona es insertada si es necesario, es

decir cuando un dato no es representado por las neuronas existenetes; neuronas en la región de baja densidad son eliminadas; y aplicar un control de eliminación que evite olvidar lo aprendido.

Cuando se presenta una muestra a la red neuronal, se encuentra la neurona más cercana y se evalua si pertenecen al mismo grupo que la neurona por medio de un criterio de umbral de similaridad. El umbral se calcula utilizando la distancia mínima entre las neuronas. Si la distancia entre la neurona ganadora y la entrada es mayor que el umbral, se inserta una nueva neurona. Si la distancia es menor, la neurona ganadora es actualizada.

En el aprendizaje supervisado se utilizan clases etiquetadas, por otro lado nn el aprendizaje no supervisado las muestras sin etiqueta son agrupadas. Primero una red neuronal con aprendizaje incremental semi-supervisado aprende la estructura básica y un clasificador es entrenado utilizando muestras etiquetadas.

La red neuronal produce las probabilidades de cada que cada dato no etiquetado pertenezca a una clase. Entonces el dato no etiquetado es clasificado por un clasificador (Support Vector Machine). El clasificador es utilizado para evaluar las etiquetas asignadas por la red neuronal durante el entrenamiento para minimizar los errores.

En aplicaciones en visión computacional comenzaré con la propuesta por S. H. Ong et al. [24], que utilizan una red neuronal artificial de dos etapas para la segmentación de imagenes a color, basados en el mapa auto-organizado propuesto por Kohonen (SOM). Se tiene un vector de entrada de tamaño $n = 16 \cdot 16$ que es conectado en paralelo a todas las neuronas de la primer etapa. Para la segunda etapa de la red se tienen 20 neuronas de salida en un arreglo de una dimensión (Fig. 5.2). El espacio de color

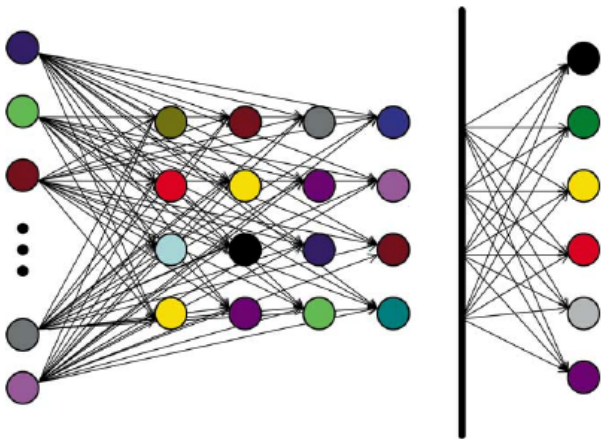


Figura 5.2: Esquema de la red neuronal de dos etapas propuesta por Ong et al.

utilizado es el CIELuv. La primer etapa de la red emplea una región de $16x16$ para crear un mapa de características. Una vez que la primer etapa ha sido entrenada se obtienen los colores dominantes de manera no supervisada.

Para las dos etapas el proceso de entrenamiento es el mismo. En el proceso de inicialización, la única condición que se tiene que cumplir es que los pesos de las neuronas deben ser diferentes. En la etapa de entrenamiento, cada neurona tiene tres pesos y son comparados con las componentes RGB de la

entrada. Para evaluar la distacia se utiliza la distancia euclidiana. Los pesos de la neurona ganadora son modificados con una suma que la acerca al valor de la entrada multiplicada por un factor α. El vecindario de la neurona ganadora es modificada de la misma manera pero la suma es afectada por un valor β. Los pesos de las demás neuronas no son afectados. Tanto α como β son función del número de iteración. El criterio de convergencia es que los cambios en los pesos son menores a una constante ϵ.

Un modelo más general fue propuesto por K. Fukushima [25] cuyo objetivo es el reconocimiento visual de patrones. La red tiene la capacidad de auto-organizarse por medio de un aprendizaje sin maestro, y adquiere la habilidad de reconocer patrones de estímulos basados en la similaridad geométrica de las formas sin ser afectados por sus posiciones.

La estructura de la red consiste de una capa de entrada bidimensional U_0(fotoreceptores), seguido por una estructura modular conectada en cascada. Cada módulo está formada por dos capas de células: la primer capa consiste de células de tipo S o células simples. La segunda capa consiste de células de tipo C o células complejas. Únicamente las sinapsis de entrada de las células S son modificables, las conexiones entre capas S y C son fijas. (Fig. 5.3).

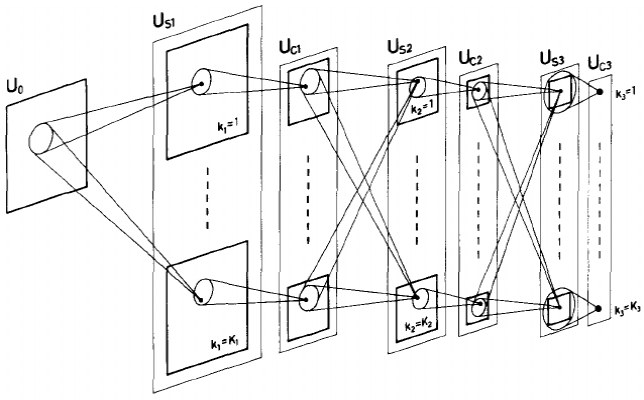


Figura 5.3: Estructura de la red neuronal propuesta por Fukushima

En cada capa las neuronas se dividen en grupos de acuerdo a las características a las cuales responden, llamados planos. En los planos S las neuronas se diferencían únicamente por la posición de sus receptores. De manera similar, agrupando células que se encuentran en posiciones cercanas, tenemos columnas S. Las columnas contienen células de todos los planos S y además se intersectan entre sí.

En el proceso de auto-organización se presenta un patrón de estímulos a la capa U_0 de manera repetida, pero no es necesario alimentar otro tipo de información. Se selecciona la célula más representativa de cada plano S en cada capa cada vez que es presentado un estímulo, asegurando así que no se creen conexiones redundantes en la que dos planos detecten la misma característica. La red fue simulada en una computadora utilizando una estructura de tres módulos, dando un total de siete capas: $U_0, U_{S1}, U_{C1}, U_{S2}, U_{C2}, U_{S3}, U_{C3}$. La capa U_0 es de tamaño 16x16 y el número de planos en cada capa es de 24. Para realizar la auto-organización de la red se presentaron cinco patrones veinte veces cada uno.

Para cada patrón se obtuvo únicamente una salida en la capa U_{C3}, asímismo cada célula en la capa U_{C3} se volvió selectiva para no responder a más de un patrón. También se confirmó que las respuestas

de esta última capa no son afectadas por cambios en la posición del patrón, ni por pequeños cambios en la forma o tamaño.

Además se experimentó utilizando diez patrones diferentes (números del 0 al 9), logrando que la red reconociera de manera correcta los patrones. Sin embargo, pequeñas variaciones de los valores de los parámetros o pequeños cambios en la presentación de los patrones afectaban de manera importante la capacidad de auto-organización de la red.

Una propuesta similar a la de Fukushima es la de Mohamed A. El-Sayed et al. [26], que desarrollaron un modelo para la detección de bordes en imágenes. La imagen de entrada es sub-escalada de manera recursiva y en la última capa de escalamiento cuenta con campos receptivos de 3x3 que hacen la detección de los bordes.

Para mejorar la generalización se pueden hacer manualmente pequeñas transformaciones al conjunto de entrenamiento. De esta manera la red neuronal aprende a ser invariante a este tipo de transformaciones. Para el entrenamiento se utilizaron un total de 17 patrones, 8 que representan un borde y 9 que representan la ausencia de borde. La red neuronal fue programada en VC++ y el proceso de entrenamiento se realizó con diferente número de épocas: desde 100 hasta 100,000, obteniendo para éste último valor los mejores resultados.

Siguiendo dentro del área de visión computacional pero ahora en cuanto a la detección de esquinas, Du ming Tsai [27] propone dos ANN: una para detectar esquinas con una curvatura muy grande (como en los polígonos) y otra para detectar puntos de tangencia e inflexión en los que generalmente se tiene una curvatura baja. Únicamente describiré la primera de éstas. Su estructura consiste de una capa de entrada, dos capas ocultas y una capa de salida. Para las pruebas se utilizaron 18 nodos de entrada; 9 nodos en la primer capa oculta y 4 en la segunda; en la capa de salida hay un único nodo que corresponde con el ángulo Θ (Fig. 5.4).

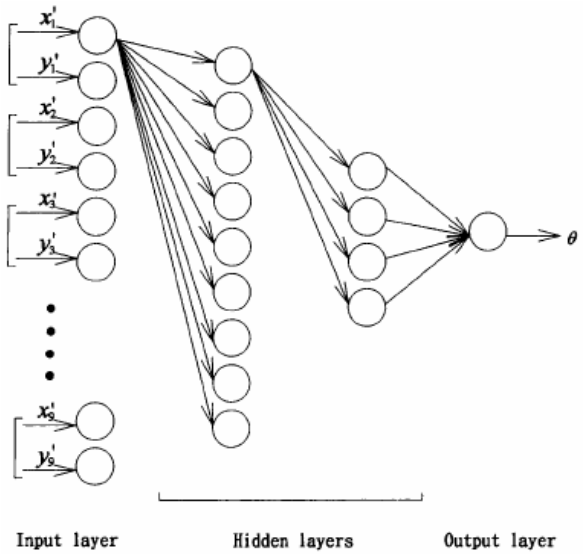


Figura 5.4: Estructura de una de las redes neuronales artificiales propuestas por Tsai

Para la etapa de entrenamiento se utilizaron patrones angulares entre 0° y 360° con pasos de 5°. Se

dice que un punto p_i es una esquina si su curvatura excede un umbral predeterminado. Además cada esquina se encuentra separada al menos por un espacio u. Para la etapa de reconocimiento se utilizaron imágenes binarias de un triángulo con 30°,60° y 90° en dos diferentes inclinaciones. Para evaluar la detección de esquinas se observaron dos puntos: la detección, que consiste en maximizar la detección de esquinas minimixando la detección de falsos; y la localización, que implica la detección lo más cercana posible a su posición verdadera.

Otra propuesta enfocada a la detección de esquinas pero utilizando el código de cadena de Freeman fue propuesto por Subri et al. [28]. El código de cadena de Freeman permite convertir una imagen en una cadena de texto que es mucho más fácil de analizar, utiliza ocho diferentes valores asignados a los ocho posibles vecinos de un pixel. El proceso de entrenamiento fue supervisado y consistió en alimentar el clasificador con 197 conjuntos. Cada conjunto es un fragmento de longitud 9 de un código de cadena extraído de una figura en 2D.

El criterio de convergencia fue que el error cuadrático medio (MSE) fuera menor a 0.1 o se alcanzaron 200,000 etapas de entrenamiento. Se hizo un análisis para determinar la mejor arquitectura de la red por medio de prueba y error de diferentes parámetros, funciones de entrenamiento y estructuras.

Se reportan 71 modelos de estructuras de los cuales 13 (18 %) no alcanzaron el MSE propuesto. En cambio, 36 de los modelos (50 %) obtuvieron una precisión mayor del 90 %. El mejor resultado fue obtenido con una red de tres capas, con una capa oculta y nueve nodos de entrada. Los nodos en la capa oculta utilizan una función de transferencia log-sigmoide.

Un enfoque alternativo es el desarrollado B. Meftah et al. [29], quienes utilizan redes neuronales por impulsos (Spiking Neural Networks) para segmentar imágenes y detectar bordes. En esta red las neuronas acumulan las señales que reciben y cuando superan un umbral determinado emiten una señal por su salida. Entonces la neurona con la mejor respuesta no está determinada por una señal más grande, sino por la que primero responde. La arquitectura de la red es totalmente conectada y consiste de una capa de entrada, una capa oculta y una capa de salida. La capa de entrada está compuesta de tres neuronas de entrada, una para cada valor RGB (Fig. 5.5).

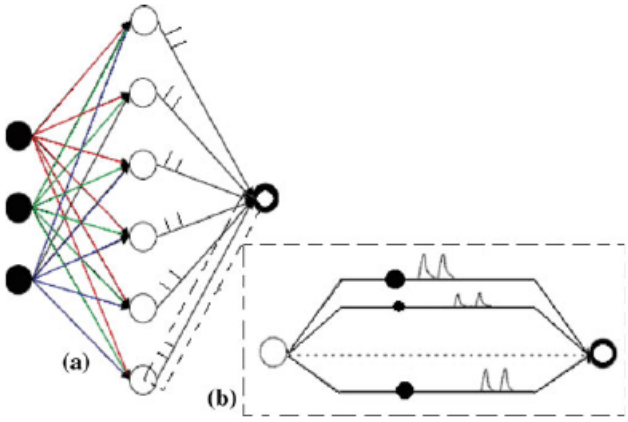


Figura 5.5: Estructura de la red neuronal propuesta por B. Meftah et al.

En lugar de una sóla sinapsis con un retardo y peso, se utiliza un modelo con muchas sub-sinapsis,

cada uno con su propio peso y retardo d^k.

Se implementó un método de aprendizaje no supervisado basado en el método de Hebb. Este algoritmo modifica los pesos entre las neuronas de entrada y la primer neurona en disparar un pulso en la capa de salida. La función de aprendizaje es una curva Gausiana que refuerza las sinapsis entre las neuronas i y j si $\Delta t_{ij} < \nu$ y las debilita en otro caso.

La red propuesta se probó en imágenes de la base de datos de Berkeley, observando quelos parámetros utilizados son específicos para cada imágen y se deben elegir de manera cuidadosa para obtener resultados eficientes.

Una característica que se encuentra presente en muchos tipos de señales es el ruido, la visión computacional no escapa de éste pero existen trabajos en redes neuronales que también intentan atacarlos. A continuación describo dos de ellos.

M. Emin Yüksel [30] presentó un operador neuro-difuso para detección de bordes en imágenes digitales afectadas por ruido. El operador propuesto es construido por la combinación de un número deseado de subdetectores neuro-difusos (NF subdetector), cada uno con tres entradas y una salida; y una etapa de post-procesamiento (Fig. 5.6).

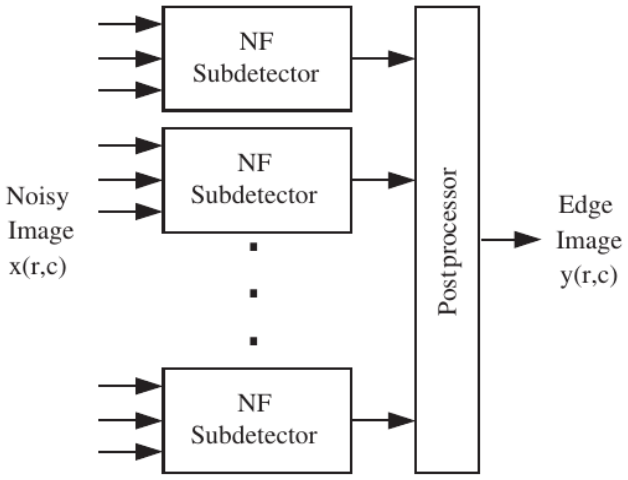


Figura 5.6: Diagrama de bloques del sistema de detección de bordes basado en una red neuronal propuesto por Emin Yüksel

Cada NF subdetector opera sobre una ventana de $3x3$ pixeles y evalúa la relación del pixel central con dos de sus vecinos. Existen muchas combinaciones de vecinos que se pueden utilizar y cada una representa una característica diferente. Entre más NF subdetector se utilicen el desempeño del operador será mejor, pero también el costo computacional será mayor. Para cada entrada existen tres funciones de membresía, por lo tanto la base de reglas contiene 27 reglas. Los parámetros que definen estas reglas de pertenencia son determinados por medio de un entrenamiento. La salida de cada NF subdetector es un promedio ponderado de las salidas de sus reglas.

La etapa final consiste en pasar las salidas de los NF subdetectors a la etapa de post-procesamiento. El post-procesador calcula el promedio de sus entradas y lo compara con un valor de umbral. El valor

de umbral es la mitad del rango disponible para los valores de luminancia. Esto es, para imagenes de 8 bits el valor de umbral es 128. Si el valor es menor al umbral, la salida es 0; en otro caso la salida es 255.

El entrenamiento para obtener los parámetros adecuados de los NF subdetectors es individual. Se parte de tres imágenes: la primera es la imagen base de entrenamiento y corresponde con la imagen original; la segunda es imagen de entrada de entrenamiento y es la imagen base pero que ha sido corrompida por ruido; y finalmente la imagen objetivo, que es la imagen que contiene la salida que debería obtenerse si se detectaran correctamente los bordes. En el entrenamiento se optimizan los valores de manera iterativa por medio del algoritmo Levenberg-Marquardt.

En la etapa de prueba, la imagen es recorrida por la ventana de $3x3$ con el operador previamente entrenado. El desempeño del operador fue comparado los detectores de bordes Sobel y Canny utilizando dos estructuras: una con 2 NF detectors y otra con 8. El detector sobel fue muy afectado por el ruido y tuvo un desempeño muy pobre. El detecto Canny tuvo un mejor desempeño pero también fue afectado de manera importante por el ruido. El operador con 2 NF detectors tuvo un mejor desempeño que el operador Canny pero su versión con 8 NF tuvo mucho mejores resultados.

Por otro lado, Terumitsu Ohyama [31] propuso una técnica para la extracción de regiones en una imagen ruidosa. Utilizó una red neuronal de tres capas con campos receptivos sobrepuestos y una capa oculta. La capa de entrada es un arreglo de $12 \cdot 12$, la capa oculta es un arreglo de $9 \cdot 9$ y la capa de salida es de $12 \cdot 12$. Una célula de la capa oculta conecta con un arreglo de $4x4$ de la capa de entrada, sobrelapandose con el siguiente (ver Fig. 5.7).

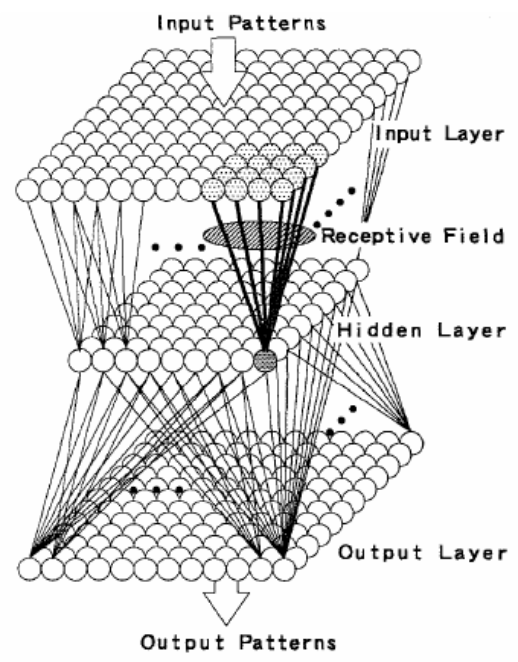


Figura 5.7: Esquema de la red neuronal propuesta por Terumitsu Ohyama.

Cada célula de la capa de salida se conecta con todas las células de la capa oculta. La función de las células en la capa de entrada es lineal, mientras que la que corresponde a la capa oculta es una

sigmoide. El aprendizaje se realiza atraves de Backpropagation.

Para hacer una verificación del desempeño de la red neuronal propuesta se realizó la comparación con otra red neuronal que tiene la misma estructura pero las conexiones de la capa oculta no se sobrelapan. En las pruebas realizadas se observa que mientras la red neuronal sin sobrelape oscila en su etapa de entrenamiento, la red neuronal propuesta converge a partir de los 2000 ciclos.

Finalmente B. Sowmya y B. Sheela Rani [32] realizaron la comparación de una red neuronal con técnicas de agrupación difusa.

La red se compone por un vector de entrada y una matriz de pesos. La entrada de la capa competitiva es compuesta encontrando el negativo de la distacia entre el vector de entrada y el vector de pesos sumando una offset b (Fig. 5.8).

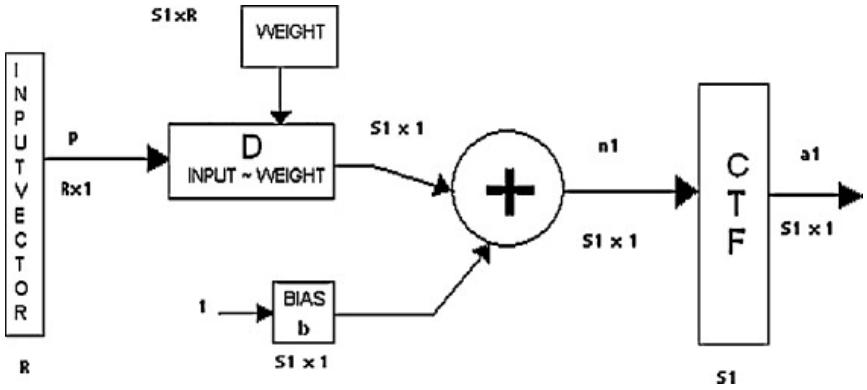


Figura 5.8: Esquema de la red neuronal propuesta por Sowmya y Sheela.

Después el valor obtenido se pasa a un bloque de función de transferencia que convierte en cero todos los valores excepto el de la neurona ganadora. La neurona ganadora es aquella que tiene el valor más positivo. Para segmentación de imágenes a color, el vector de entrada se forma con los colores primarios. El número de salidas corresponde con el número de grupos deseados. Se hicieron comparaciones utilizando un algoritmo de Fuzzy C means(FCM) y otro de Possibilistic Fuzzy C means (PFCM). FCM es el modelo más simple pero los centros de los grupos dependen de los valores iniciales elegidos. PFCM resultó ser el algoritmo más rápido pero requiere algún método para inicializarlo. Por otro lado, la red neuronal competitiva resultó ser mejor definiendo los grupos pero requirió de más tiempo. Aunque una vez que la red es entrenada, el tiempo que requiere para hacer la segmentación es el menor de todos.

Después de esta breve revisión de trabajos enfocados en las redes neuronales es notable que existe una infinidad de modelos, desde los más sencillos hasta los más complejos. Algunos de ellos es posible combinarlos con algunas otras técnicas para beneficiarse una de la otra.

Por lo tanto, es viable utilizar redes neuronales para identificar objetos en sistemas de automatización, que es el objetivo de este trabajo y sobre el cual entraré a detalle en el siguiente capítulo.

Capítulo 6

Declaración del problema

A pesar de que el uso de la energía solar se encuentran en pleno auge, son pocos los diseños orientados al aprovechamiento de ésta abarcando tanto una alta eficiencia como un bajo costo. En México se cuenta con un importante potencial en energía solar, sin embargo el desarrollo de tecnología en ésta área es mínimo y por lo tanto este potencial no ha sido aprovechado. Por esta razón el desarrollo de tecnología orientada al aprovechamiento de la energía solar es muy importante, y más si adicionalmente permite la realización de su objetivo a bajo costo. En este trabajo, se propone el uso de visión computacional y redes neuronales para realizar un sistema de detección de espejos planos, que a su vez, servirá como base a un sistema de automatización que permita la construcción de concentradores solares de bajo costo.

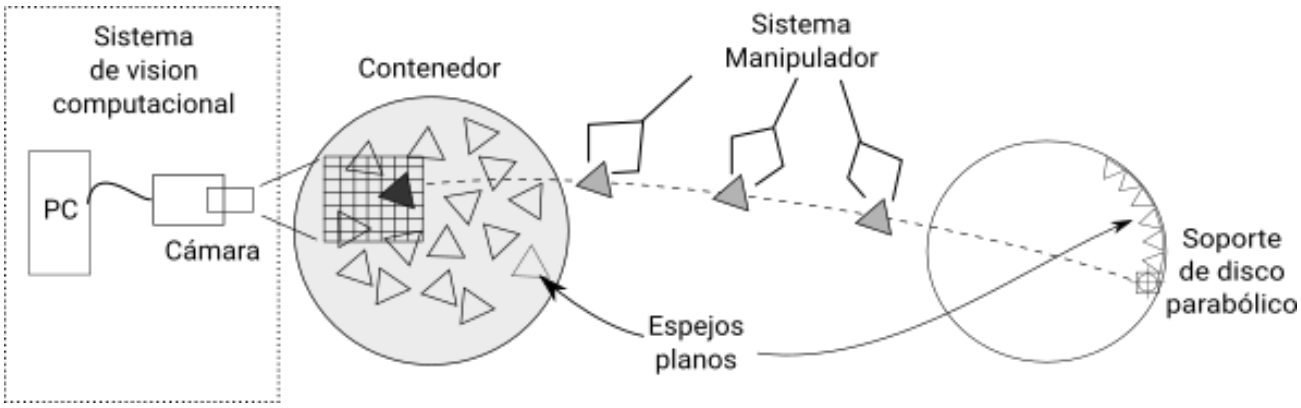


Figura 6.1: Esquema del funcionamiento del sistema de automatización para el ensamble de concentradores solares de bajo costo

En la Fig. 6.1 se muestra un esquema que ejemplifica el sistema de automatización completo. En la parte izquierda se encuentra el hardware del sistema de visión computacional compuesto por una computadora personal y una cámara. Junto a éste se encuentra el contenedor con varios de los espejos planos triangulares que se tiene que reconocer. Un sistema manipulador (a diseñarse en una etapa posterior a este trabajo) debe tomar cada uno de los espejos en el contenedor y colocarlo en la posición que le corresponde en el concentrador solar.

Como ya se mencionó anteriormente, este trabajo se enfoca únicamente en la primer parte de este sistema de automatización: la detección de los espejos. Para lograrlo, el sistema de visión computacional tomará una imagen de la escena y la dividirá en pequeñas ventanas, de menor tamaño que los espejos a detectar. Entonces se clasificarán las ventanas por medio de una red neuronal artificial indicando si pertenece a un espejo o no.

En una etapa posterior a este trabajo, se utilizará la información generada por la clasificación para calcular la posición del espejo. Finalmente, se transmitirá está posición a un sistema manipulador que se encargará de asistir el ensamble automatizado de concentradores solares.

Si bien se ha mostrado en capítulos anteriores que las ANN permiten obtener muy buenos resultados, también hay que hacer notar que existen algunas complicaciones muy importantes a tomar en cuenta. Una de ellas es la elección del modelo para la red neuronal, esto es debido a la existencia de muchos diseños y la flexibilidad para crear nuevos.

Otro inconveniente es la elección de los parámetros con los que trabajará la red, como se mencionó en los trabajos descritos, dependiendo del modelo de red, una pequeña variación puede afectar de manera considerable la respuesta. Por último se tiene que, en general, la mayor carga de cómputo utilizada en una red neuronal es utilizada en la etapa de entrenamiento. En esta etapa, en algunos casos se llegan a realizar miles de iteraciones lo que puede requerir de un tiempo considerable aun con los equipos de cómputo actuales.

Un modelo de red neuronal que presenta un diseño sencillo, y además requiere pocos ciclos de entrenamiento, es el denominado clasificador de umbrales aleatorios (RTC por sus siglas en inglés). En el siguiente capítulo se detalla la estructura de esta red y su implementación.

Capítulo 7

Clasificador de Umbrales Aleatorios

El clasificador de umbrales aleatorios (RTC por sus siglas en inglés: Random Threshold Classifier), es una red neuronal desarrollada por Kussul *et al.*[33]. Esta red neuronal tiene un buen desempeño tanto en la etapa de entrenamiento como en el procesamiento [34]. Ha sido aplicada de manera exitosa en el reconocimiento de texturas en imágenes en escala de grises [35], pero su diseño esta pensado para poder ser utilizado en diversas aplicaciones. A continuación se detalla su arquitectura.

7.1. Estructura del clasificador

El RTC consiste de una gran cantidad de bloques de neuronas (S bloques con 3 capas cada uno), una capa de neuronas de salida que representa las clases a reconocer (capa c) y conexiones modificables (entrenables) w entre los bloques y la capa c. En la Fig. 7.1 se muestra la estructura del RTC.

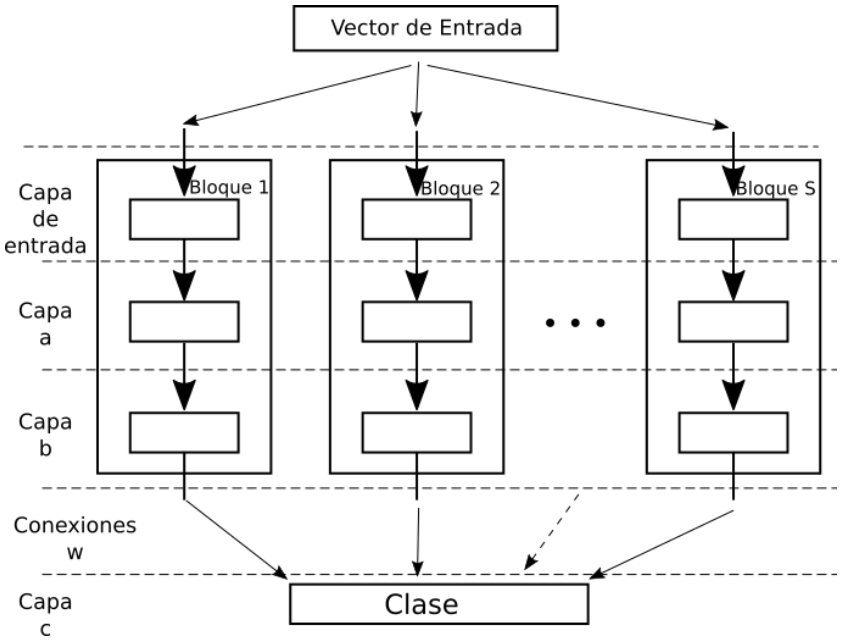


Figura 7.1: Estructura del RTC

Todos los bloques son alimentados por el mismo vector de entrada X de tamaño n y con elementos x_j. Este vector de entrada o estímulo, representa las características que se desean clasificar. La elección de estas características depende de lo que se desee clasificar. Por ejemplo, para el caso de imágenes estas características pueden ser: brillo, contraste, contornos, etc.

Una vez que se ha tiene el vector de entrada, se presenta al RTC. Esto implica presentar el vector de entrada a todos los bloques que conforman la red neuronal.

La estructura de los bloques se describe a continuación.

7.1.1. Descripción de la estructura de los bloques

Los bloques están formados de tres capas de neuronas cada uno: la capa de entrada, la capa a y la capa b. En primer lugar se tiene la capa de entrada y la forman un par de neuronas para cada elemento del vector de entrada, es decir existen n pares de neuronas. Cada par se compone por una neurona l_{ij} y una h_{ij}. Donde: i representa el bloque: $1 < i < s$ y; j representa el elemento en el vector de entrada: $1 < j < n$ (ver Fig. 7.2).

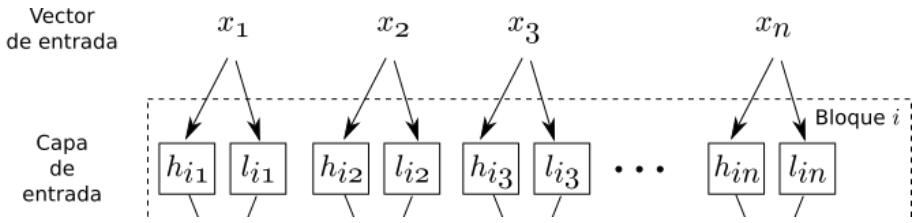


Figura 7.2: Capa de entrada para bloque i del RTC

Tanto las neuronas l_{ij} como las neuronas h_{ij} se activan (su salida es 1) cuando su entrada es mayor a un valor (umbral). El valor de estos umbrales se eligen de manera aleatoria, la única restricción que debe observarse es: $l_{ij} < h_{ij}$.

La salida de las neuronas de la capa de entrada se conectan a las neuronas de la capa a. Las neuronas de la capa a tienen una salida y dos entradas (una excitadora y una inhibidora). La salida se activa únicamente cuando la entrada excitadora es activada (vale 1) y la entrada inhibidora está desactivada (vale 0). La salida de las neuronas l_{ij} se conectan a la entrada excitadora de la neurona a_{ij}, mientras que la salida de las neuronas h_{ij} se conectan a la entrada inhibidora de la misma (ver Fig 7.3).

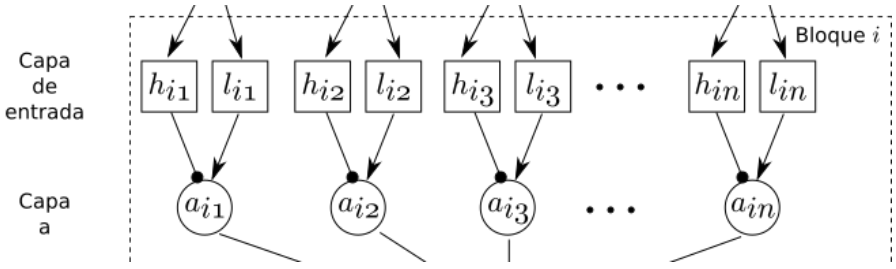


Figura 7.3: Capa a del bloque i

Gracias a esta conexión, la salida de la neurona a se activará (será uno) cuando el valor correspondiente del vector de entrada se encuentre entre los valores de los umbrales de las neuronas l_{ij} y h_{ij}, y será cero en otro caso. Es decir:

$$a_{ij} = \begin{cases} 1 & \text{si } l_{ij} < x_j < h_{ij} \\ 0 & \text{si } x_j \geq h_{ij} \quad o \quad x_j \leq l_{ij} \end{cases} \tag{7.1}$$

Las salidas de todas las neuronas de la capa a de un bloque son conectadas a la única neurona de la capa b del bloque (neurona b_i). Esto es, la neurona b tiene tantas entradas como neuronas a en el bloque y una salida (ver Fig. 7.4). La salida de la neurona b es la salida del bloque.

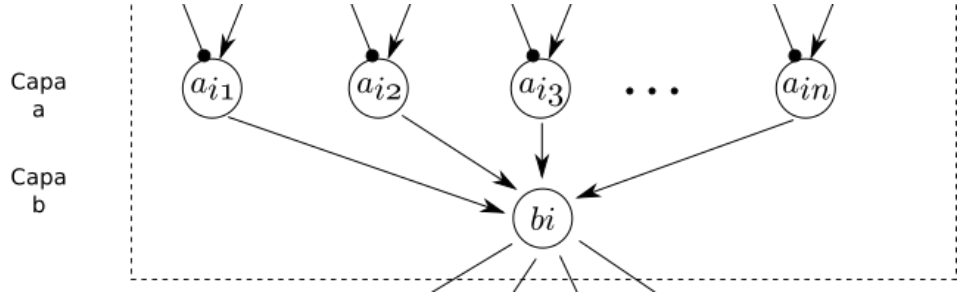


Figura 7.4: Capa b del bloque i

La salida de la neurona b_i únicamente se activa (es uno) si todas sus entradas son uno. Por lo tanto, la salida sólo se activará cuando todos los valores en el vector de entrada se encuentran dentro de los umbrales definidos por las neuronas l_{ij} y h_{ij} del bloque. Es decir:

$$b_i = \begin{cases} 1 & \text{si } l_{ij} < x_j < h_{ij} \; \forall \; x_j \\ 0 & \text{si } \exists \; x_j : x_j \geq h_{ij} \;\vee\; x_j \leq l_{ij} \end{cases} \tag{7.2}$$

Finalmente la salida de la neurona b_i se conecta con todas las neuronas de la capa c (que representan las clases a reconocer) por medio de conexiones con peso entrenables w_{ik}, donde: i representa el bloque y k representa la clase (Fig. 7.5).

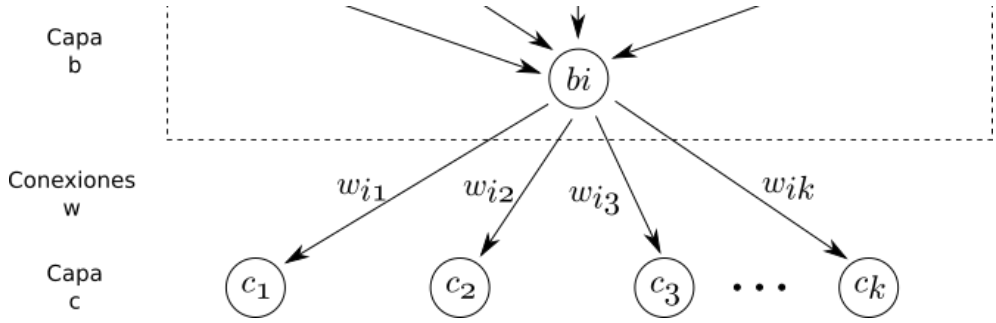


Figura 7.5: Capa c y sus conexiones con el bloque i

En la Fig. 7.5 podemos observar como la neurona b_i se conecta con las k neuronas de la capa c. Es decir, para cada neurona b existirán k pesos, haciendo un total de $i \times k$ pesos.

En la Fig. 7.6 se muestra el esquema completo correspondiente con el bloque i-ésimo. Por sencillez, este bloque representa a todos los bloques de la red neuronal. Por lo cual $1 < i < s$, donde s es el número de bloques.

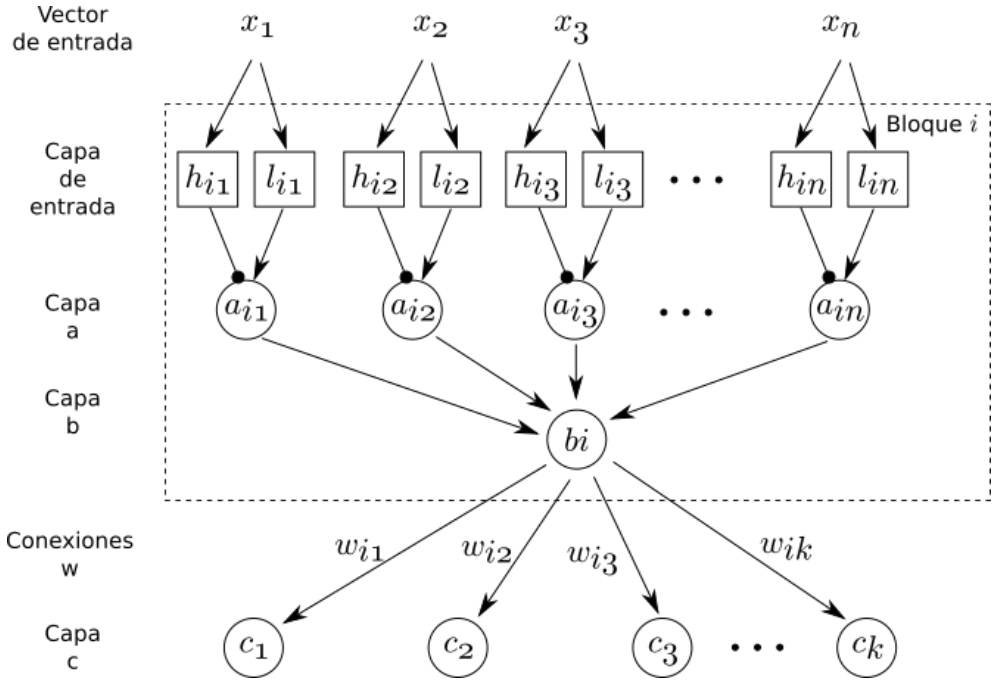


Figura 7.6: Esquema del bloque i-ésimo de la red neuronal RTC

7.1.2. Cálculo de clasificación

Dadas las conexiones descritas, el valor de cada neurona c_k de la capa c estará dado por la sumatoria de los productos de los pesos w_{ik} por la salida de la neurona b_i correspondiente. Es decir:

$$c_k = \sum_{i=1}^{s} w_{ik} b_i \tag{7.3}$$

Entonces la clasificación depende de dos valores: la salida de cada bloque b_i y los pesos w_{ik}. A su vez, la salida de cada bloque depende del vector de entrada X y de los umbrales h_{ij} y l_{ij}. En cambio los valores de los pesos deben ser modificados en la etapa de entrenamiento, de manera que la respuesta a cada estímulo sea la correcta. La clase reconocida C_R corresponderá con aquella neurona de la capa c que tenga el valor mayor, es decir:

$$C_R = max(c_k) \tag{7.4}$$

7.1.3. Etapa de entrenamiento

Los pesos comienzan con un valor inicial w_{ik0} y utilizando estos valores se presenta un vector de entrada al clasificador, cuya clase C_T es conocida. Los valores iniciales pueden ser cualquiera, por ejemplo 1.

Si la respuesta C_R del clasificador es diferente a la esperada C_T, se disminuyen los valores de los pesos w_{iI} correspondientes a la clase R incorrecta y se aumenta el valor de los pesos w_{iC} correspondientes a la clase correcta.

Este procedimiento se repite varias veces presentando el mismo y otros vectores de entrada hasta que se alcanza un criterio de paro. Se consideran dos criterios de paro: 1) un número determinado de iteraciones y 2) obtener menos de una cierta cantidad de errores. El entrenamiento termina cuando se da cualquiera de los dos.

7.2. Clasificador de Subespacio Aleatorio (Random Subspace Classifier)

Cuando el vector de entrada crece en el RTC, las posibles combinaciones que existen de ellos se incrementa de manera exponencial, por lo que la cantidad de neuronas necesarias para que la red funcione se incrementa de la misma manera.

Como una manera de solucionar este problema, existe una variante al RTC llamada Clasificador de Subespacio Aleatorio (RSC). El RSC escoge de manera aleatoria un subconjunto (subespacio) de elementos pertenecientes al vector de entrada original. A partir de este subespacio se forma un nuevo vector de entrada que es de mucho menor tamaño que el original y con el cual puede trabajar RTC. De esta manera es posible utilizar RTC con vectores de entrada de longitud considerable. Cabe mencionar que, para cada bloque del RTC se genera un subespacio diferente. Es decir, cada bloque del RTC tiene un subespacio asociado.

Ya que se ha detallado la estructura del RTC y RSC, en el siguiente capítulo se explica su simulación por medio de software así como algunos detalles de su implementación.

Capítulo 8

Software de simulación de RTC/RSC

A continuación se presenta la descripción del software que sirve para simular el clasificador de umbrales aleatorios y el clasificador de subespacio aleatorio.

El software está escrito en Borland C++ 6 y los parámetros deben ser modificados desde el código fuente. Se cuenta con la interfaz gráfica para visualización y verificación e interacción con los procesos que se ejecuten. Existen dos zonas en la interfaz gráfica (Fig. 8.1) la zona de menú y la zona de despliegue. En la zona de menú se encuentran las acciones a realizar. En la zona de despliegue se muestran los resultados de la ejecución de las acciones.

Figura 8.1: Zonas del software de simulación

Consta de cuatro etapas básicas:

1. Generación de Máscaras (Mask Generation)
2. Abrir imagen y codificar (Open image;Coding)
3. Entrenamiento (Training)
4. Reconocimiento (Recognition)

Dichas etapas se corresponden con los menús existentes en la pantalla gráfica como se observa en la Fig. 8.2.

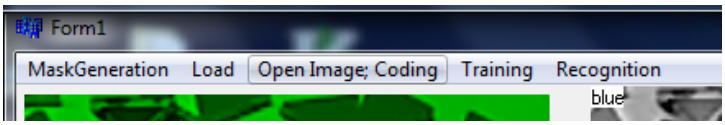
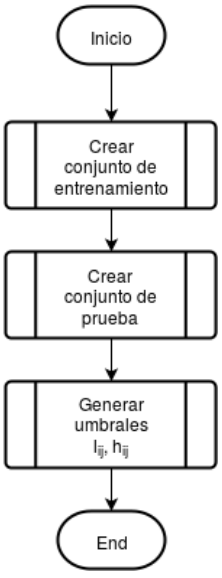


Figura 8.2: Menú del software de simulación

8.1. Generación de máscaras

En esta parte básicamente se realiza la inicialización del modelo. Se elige de manera aleatoria que imágenes se utilizarán para entrenamiento y que imágenes serán para la fase de prueba (reconocimiento) También se generan los umbrales de las neuronas de la capa de entrada (l_{ij} y h_{ij}) para cada bloque. Con esto definimos la estructura del clasificador. Un ejemplo del resultado obtenido se muestra en la Fig. 8.3a y el diagrama de flujo del proceso se puede ver en la Fig. 8.3b

(a) Captura de pantalla de generación de máscaras con 10 imágenes. (5 para entrenamiento y 5 para pruebas)



(b) Diagrama de flujo

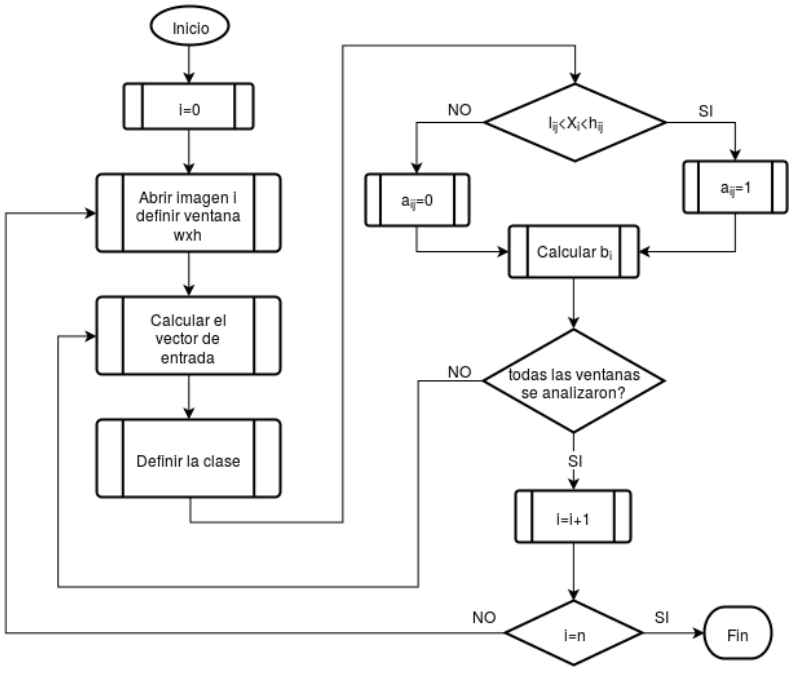
Figura 8.3: Etapa de generación de máscaras.

8.2. Codificación

En esta etapa se obtienen los vectores de entrada a partir de las imágenes seleccionadas. Para poder realizar la codificación primero se define una ventana (área de la imagen) y de ésta se extrae el vector de entrada.

Entonces se le aplican los umbrales calculados previamente para cada neurona l_ij y h_ij y se obtiene la respuesta de las neuronas de la capa b. Se crea una lista con las neuronas de la capa b activas (neuronas cuya salida es diferente de cero) y se almacena en un archivo. En la Fig. 8.4a se puede observar el proceso de codificación y como se va mostrando en el software.

(a) Captura de pantalla.



(b) Diagrama de flujo

Figura 8.4: Etapa de codificación

Mientras las imágenes son procesadas, en el software se van mostrando el avance marcando la ventana procesada con verde. El software muestra el nombre de la imagen procesada, la clase calculada de acuerdo con la imagen marcada y el número de neuronas de la capa b activas. El diagrama de flujo reducido puede verse en la Fig. 8.4b

8.2.1. Entrenamiento

El objetivo de esta etapa, como su nombre lo indica, es entrenar a la red neuronal para que responda de manera correcta a los estímulos presentados. Es decir que para un vector de entrada dado, la c_k con el valor mayor sea la correcta.

En este caso, el entrenamiento consiste en actualizar de manera iterativa los pesos w_{ik} de manera al calcular las c_k se maximice la correcta. Existen dos criterios de paro: uno es el número de ciclos de entrenamiento, el otro es la obtención de cero errores.

El entrenamiento a realizar es un entrenamiento supervisado, es decir, es necesario indicarle a la red neuronal cual es la clase correcta para cada vector de entrada presentado. Con el objetivo de facilitar esta tarea se realizó previamente una imagen marcada que corresponde con la imagen a procesar, que nos indica la clase a reconocer.

Como ya se tiene una lista con las neuronas activas para cada imagen, ya no es necesario volver a procesar las imágenes. Simplemente se calculan los valores de pertenencia a las clases c_k a partir de la lista de neuronas de la capa b activas y los pesos de la capa w como se indicó anteriormente. Se compara la clase calculada con la clase definida previamente (marcada). Si estas clases son diferentes, los pesos correspondientes a la clase correcta son aumentados y los pesos de la clase incorrecta es disminuido. Además se aumenta el contador de errores. Si las clases calculada y marcada coinciden no se realiza ningún cambio en los pesos. Este proceso actualiza los valores de los pesos w de manera que se ajusten a la clase a identificar. El entrenamiento se aplica a todas las imágenes marcadas para entrenamiento y se repite hasta que se alcanza uno de los dos criterios de paro. En este caso el software únicamente muestra la cantidad de errores que se tienen en cada ciclo.

El diagrama de flujo se presenta en la Fig. 8.5

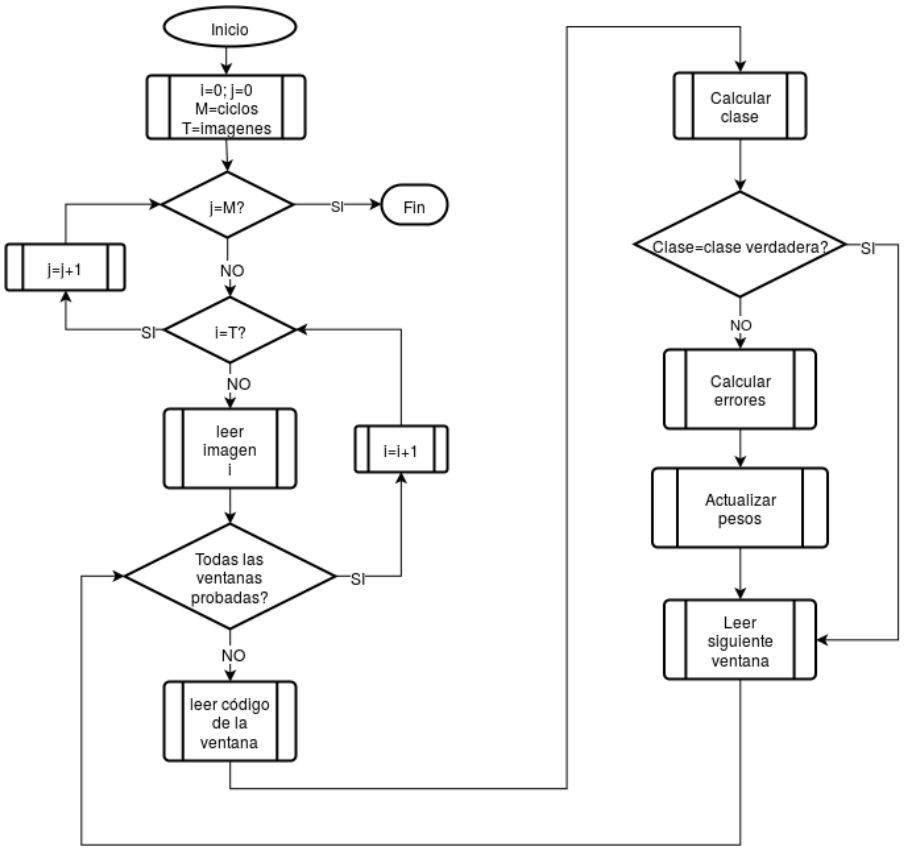


Figura 8.5: Diagrama de flujo de la etapa entrenamiento

8.3. Reconocimiento

Esta es la etapa de prueba y en ella se verifica el clasificador neuronal. Básicamente los pasos a seguir son los mismos que en la etapa de entrenamiento excepto que ahora se presentan imágenes diferentes a las del entrenamiento y sólo hay un ciclo en el cual no se modifica el vector de pesos. Se presenta una imagen y es recorrida por una ventana. De la ventana se extrae el vector de entrada que alimentará a la red neuronal. Se calculan los valores de las neuronas c_k, el valor mayor corresponde a la clase reconocida. La clase reconocida se compara con la clase previamente marcada de la misma manera que en la fase de reconocimiento. Si la clase calculada es diferente a la clase marcada, se cuenta un error y se continúa hasta recorrer toda la imagen. Este procedimiento se realiza para todas las imágenes elegidas. Al final se obtiene el conteo de los errores obtenidos y se presenta en pantalla. El diagrama de flujo reducido se presenta en la Fig. 8.6.

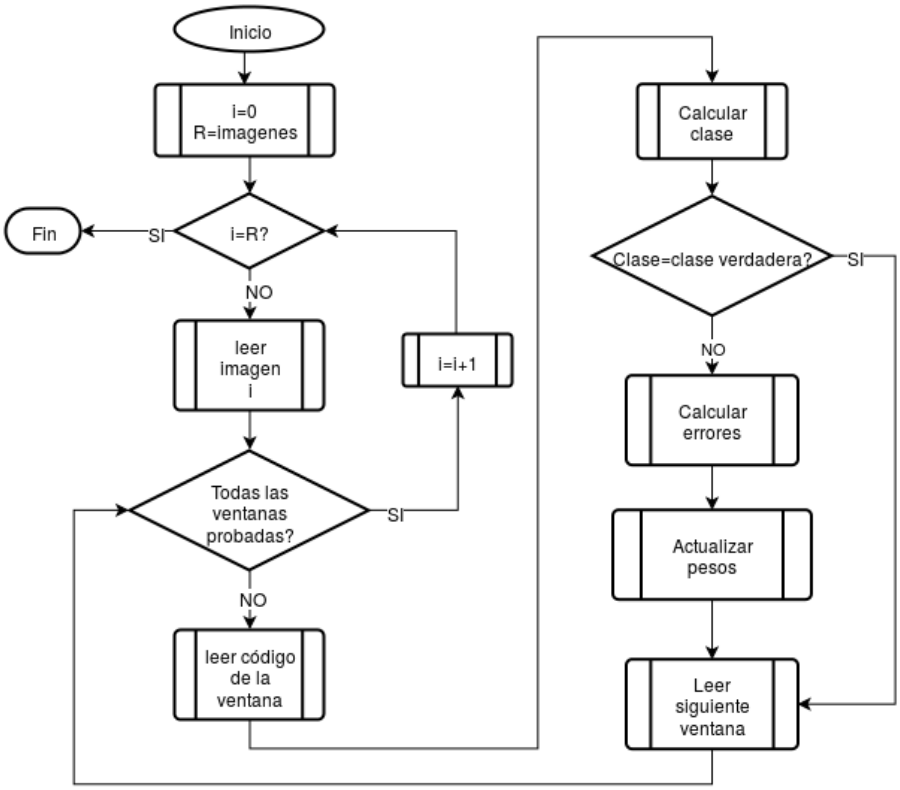


Figura 8.6: Diagrama de flujo de la etapa de reconocimiento

En este capítulo se ha descrito el software básico de simulación para RTC y RSC, sin embargo un aspecto fundamental para que éste funcione es una base de imágenes con que alimentar el modelo. El siguiente capítulo describe la base de imágenes utilizada así como los pasos seguidos para crearla.

Capítulo 9

Base de imágenes

Para que una red neuronal artificial trabaje adecuadamente debe ser entrenada. En la fase de entrenamiento se utilizan imágenes representativas de las que se desea realizar el reconocimiento. Para esto es necesario tener un grupo de imágenes al cual llamaremos base de imágenes. Para el caso de tareas como reconocimiento de caracteres, existen bases de imágenes públicas que pueden ser utilizadas facilitando el trabajo y permitiendo una comparación directa de los resultados obtenidos con diferentes metodologías. Dado que la aplicación de que se trata aquí es muy particular, la base de datos también es muy particular y fue necesario generarla a partir de fotografías de los espejos a reconocer.

La base de imágenes se generó a partir de 32 fotografías tomadas de los espejos triangulares en un contenedor (Fig. 9.1). Sus características son las siguientes: Formato: JPG, Tamaño: 640x480px, Canales: 3x8bits (RGB).

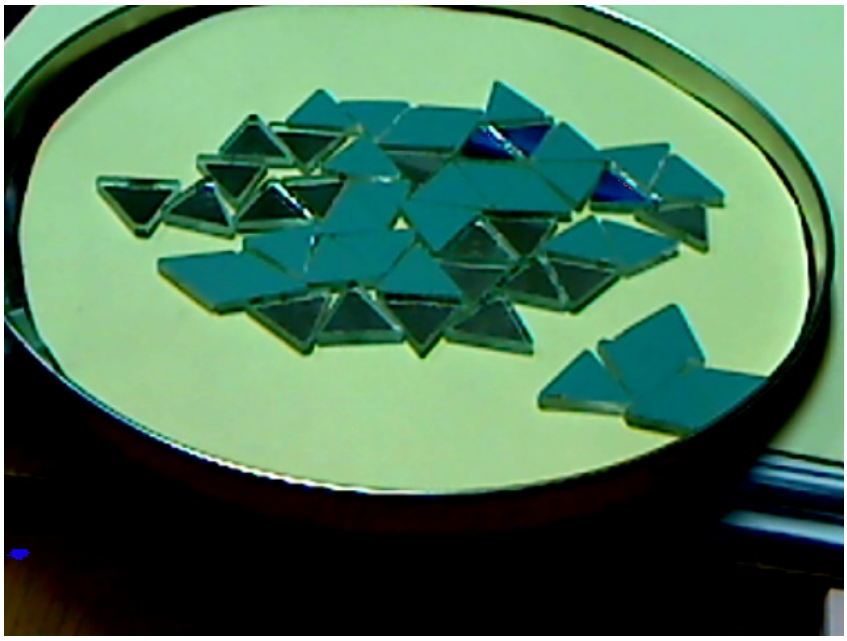


Figura 9.1: Una de las fotos de los espejos en el contenedor.

Con objeto de entrenamiento de la red neuronal fue necesario un pre-procesamiento que marque las imágenes: los bordes con líneas de color negro (RGB[0,0,0]) de 2px y el interior blanco (RGB[255,255,255]). Además, como puede observarse en la Fig. 9.1, en las imágenes aparecen objetos ajenos a los destinados a reconocer, por lo que también se requiere de otra etapa de pre-procesamiento que haga un recorte recortarlas.

Este pre-procesamiento puede hacerse de manera manual pero requeriría de mucho tiempo. Para reducir el tiempo invertido en el pre-procesamiento se decidió realizar programas que permitieran realizarlo de manera semi-automática. Fue necesario realizar 4 programas en Python que modificaran las imágenes:

1. Modifica los píxeles de la imagen de manera que ninguna tenga las componentes RGB[0,0,0];
2. Realiza el marcado de los bordes de las aristas de los triángulos a partir de los vértices;
3. Rellena de blanco la zona delimitada por los bordes previamente marcados;
4. Recorta de manera paralela las imágenes marcadas y las originales de manera que coincidan.

A continuación se describe de manera más detallada el funcionamiento de cada uno de estos programas.

9.1. Programa de modificación de píxeles

Ya que en el marcado de la imagen se utilizaron los colores negro y blanco, debemos asegurar que estos colores no aparezcan en zonas no deseadas (marcadas). Para lograr este objetivo se debe buscar en la imagen estos colores y sustituirlos con otro color parecido.

Estrictamente el color negro corresponde a RGB[0,0,0]. Sin embargo para dejar una distancia en relación a otras combinaciones de componentes RGB, se cambiaron todos los píxeles que tuvieran alguna componente RGB igual a cero. Es decir, si para algún píxel de la imagen la suma de sus componentes RGB es menor que 3, todas sus componentes se cambiaron a 1.

Esto se realizó utilizando las librerías glob, numpy, opencv y matplotlib. Mediante la función glob de la librería glob se obtuvo una lista con los nombres de todos los archivos con terminación JPG. Después para cada elemento de esta lista (imagen) se realizó lo siguiente:

- Cargar la imagen en un arreglo de dimensiones (640x480x3) mediante la función imread de opencv
- Se recorre cada píxel del arreglo y se suman sus componentes.
- Si la suma es menor a 3, el píxel toma el valor [1,1,1].
- Se guarda la imagen modificada en formato PNG.

Con esta modificación se asegura que los únicos píxeles con todas sus componentes igual a cero son los utilizados para marcar la imagen.

9.1.1. Programa de marcado

El programa de marcado realiza su tarea de manera semi-automática: abre una imagen y permite indicar mediante clicks del mouse el lugar en donde se encuentran los vértices de cada triángulo. A medida que se indican los vértices también van apareciendo las aristas que los unen en color negro. Cada triángulo se compone de tres vértices (V_1, V_2 y V_3). Cada vértice se caracteriza por dos valores o coordenadas (X, Y) que definen su posición dentro de la imagen. Estos valores se almacenan en un arreglo de 3 dimensiones de tamaño $t * 3 * 2$, donde t es el número de triángulos.

El arreglo se almacena en un archivo para su posterior uso y por facilidad se le asigna el mismo nombre que el de la imagen a partir de la cual se generó.

Una vez marcados todos los triángulos, se crea una imagen nueva que contiene a la imagen original mas las aristas de cada triángulo en negro. Básicamente se utilizan cuatro variables:

triangles es la lista de los triángulos;

point_ctr indica el vértice del triángulo actual a definir (de 0 a 2);

flag_edit es una bandera que indica si se desea editar el último triángulo definido (re-editar);

img es un arreglo en donde se ha cargado la imagen.

El resultado obtenido se muestra en la Fig. 9.2 y el algoritmo utilizado se describe a continuación:

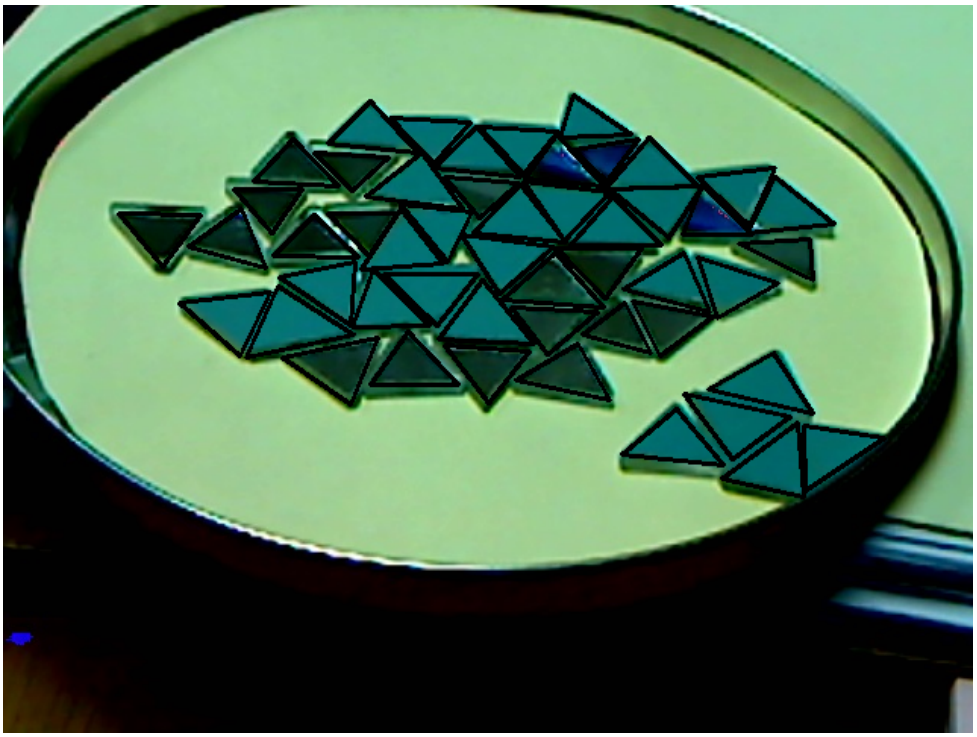


Figura 9.2: Ejemplo de una fotografía con los bordes de los triángulos marcados

- Inicializar *point_ctr* a cero y la lista de triángulos con un triángulo.
- Abrir el archivo con la imagen y cargarlo en *img*.
- Mostrar la imagen.
- Asignar una función (callback) a los eventos de click del mouse.
- Iniciar un bucle que responderá a las señales del teclado de la siguiente manera:
  - 'q' corresponde a salir del programa.
  - 'e' edita un vértice del triángulo actual.
  - 's' guarda la lista de triángulos en un archivo.
  - 'l' carga una lista de triángulos previamente guardada y dibuja las aristas.
  - 'i' almacena la imagen marcada

9.1.2. Programa de rellenado

Una vez que se tienen las imágenes con los bordes marcados es posible el rellenado automático de color blanco. El programa de rellenado únicamente necesita la lista (arreglo) de triángulos creada previamente. De esta lista, se toma cada triángulo y calcula su centroide $C(C_X, C_Y)$ mediante las ecuaciones (9.1) y (9.2) para las componentes X y Y respectivamente.

$$C_X = \frac{X_1 + X_2 + X_3}{3} \tag{9.1}$$

$$C_Y = \frac{Y_1 + Y_2 + Y_3}{3} \tag{9.2}$$

Donde: C_X es la componente X del centroide; C_Y es la componente Y del centroide; X_1 y Y_1 son las componentes de las coordenadas del vértice 1 del triángulo; X_2 y Y_2 son las componentes de las coordenadas del vértice 2 del triángulo; X_3 y Y_4 son las componentes de las coordenadas del vértice 3 del triángulo.

Una vez que se tiene el píxel correspondiente al centroide C, este se pinta de blanco (RGB[255,255,255]) y se comienza a verificar en los píxeles vecinos. Si alguno de los píxeles vecinos es de color negro (RGB[0,0,0]) quiere decir que se ha encontrado con el borde marcado, si es blanco quiere decir que este píxel ya ha sido previamente marcado. En estos dos casos el programa deja el píxel intacto, en cualquier otro caso cambia las componentes del píxel para que sea de color blanco.

Este procedimiento se realiza de manera iterativa hasta que se han marcado todos los píxeles correspondientes al área delimitado por los vértices V_1, V_2 y V_3. El algoritmo descrito está basado en el algoritmo *Breadth-Fisrt Search*, y se muestra en seguida.

1. Se coloca el centroide en una pila.
2. Se obtiene un elemento de la pila.

3. Se calcula el vecindario del elemento.
4. Para cada elemento del vecindario.
    - Si no es blanco ni negro, se pinta de blanco y se coloca en la pila
5. Regresa al paso 2 mientras existan elementos en la pila

En la Fig. 9.3 se puede observar el resultado obtenido después de correr el programa sobre la imagen de la Fig. 9.2.

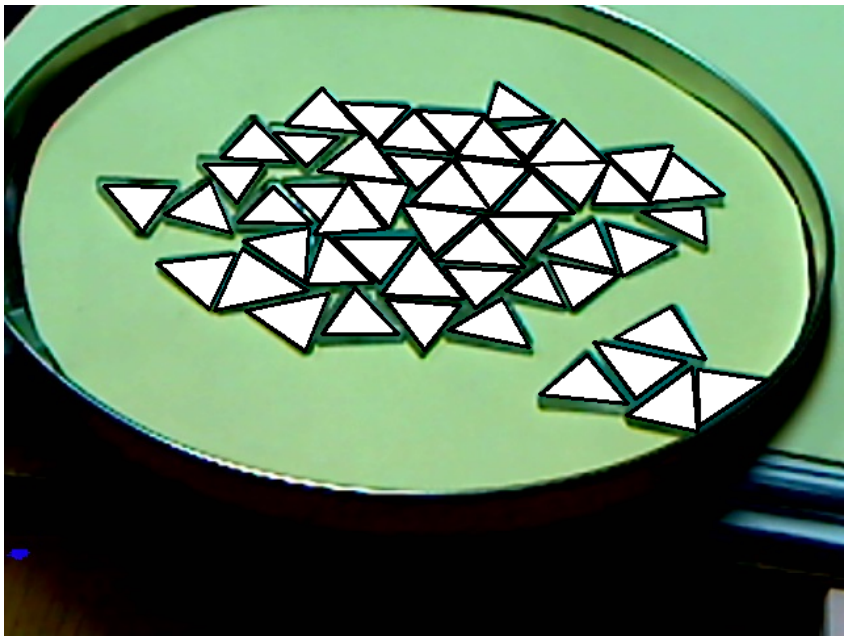


Figura 9.3: Ejemplo de una fotografía con los bordes de los triángulos marcados con negro y su interior rellenado de blanco.

9.1.3. Programa de recorte

Finalmente, para la parte de recorte de la imagen se realizó un programa que carga cada una de las imágenes originales y dibuja un rectángulo de 320x240px. Este rectángulo puede moverse por la imagen por medio del teclado. Una vez ubicado en la posición deseada se hace el recorte de la imagen de acuerdo con los límites indicados por este rectángulo y se almacena en un archivo.

Hasta este punto ya se tiene la imagen original recortada de manera que se han eliminado objetos no deseables, sin embargo es necesaria una imagen correspondiente pero que se encuentre marcada para realizar el entrenamiento. Para esto se carga la imagen correspondiente marcada y se hace el recorte basándose en el rectángulo establecido anteriormente. Este recorte hecho en la imagen marcada es el correspondiente al recorte hecho en la imagen original y se guarda en otro archivo. El algoritmo se lista a continuación.

- Crear una lista con los nombres de las imágenes.
- Para cada imagen
  - Cargar los píxeles de la imagen en un arreglo y mostrarla en pantalla.
  - Dibujar un rectángulo de 320x240px
  - Esperar a que se presione una tecla
    - 'q' termina el programa
    - 'h' mueve el rectángulo a la izquierda
    - 'l' mueve el rectángulo a la derecha
    - 'j' mueve el rectángulo hacia arriba
    - 'k' mueve el rectángulo hacia abajo
    - 's' almacena la imagen encerrada por el rectángulo, carga la imagen correspondiente marcada y hace el recorte en las mismas coordenadas.
    - 'b' carga la imagen anterior en la lista
    - 'n' carga la imagen siguiente en la lista

De esta manera se tiene un programa que carga una imagen sin marcar y nos permite elegir la posición en la que deseamos hacer un recorte. Una vez elegida la posición, se realiza el recorte tanto en la imagen sin marcar como en la imagen marcada. Se pueden hacer más de un recorte de cada imagen, de manera que se incrementan las imágenes disponibles para el entrenamiento. Así se obtuvieron 97 pares de recortes de 320x240px en los que no aparecen texturas diferentes a las que se interesa reconocer, un ejemplo de ellas se muestran en la Fig. 9.4.

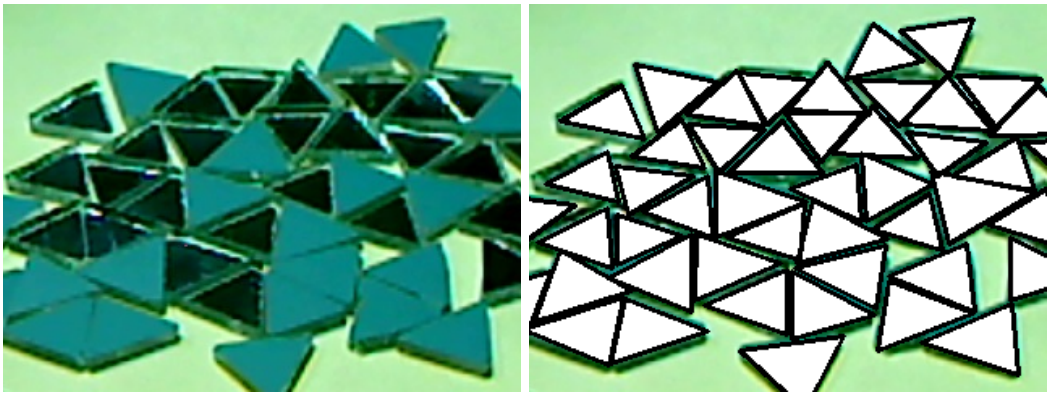


Figura 9.4: Pares de recortes de 320x240px en los que únicamente aparecen las texturas de interés

Capítulo 10

Experimentos y resultados

Una de las partes fundamentales que caracterizan a una red neuronal son los parámetros que se aplican para su funcionamiento. Aunque la red utilizada es sencilla, se cuenta con un número de parámetros considerable para elegir. Recordemos que de ellos depende el funcionamiento de la red, por lo que la adecuada elección es de suma importancia. En este capítulo se describen las pruebas más sobresalientes realizadas con el RTC y RSC, manteniendo algunos parámetros fijos y modificando otros para obtener el reconocimiento de los espejos.

Los parámetros que se pueden modificar al RTC/RSC son: el vector de entrada: tanto en tamaño como en tipo; la cantidad de bloques de neuronas; la cantidad de ciclos de entrenamiento (máximo); las imágenes utilizadas para entrenamiento; las imágenes utilizadas para reconocimiento y; el número de clases.

Dentro de estos parámetros tenemos dos que son de vital importancia ya que definen gran parte el flujo de información en la red, por lo tanto son los primeros que se deben elegir: el vector de entrada y el número de clases.

En lo que respecta al vector de entrada, éste es generado a partir de una región o ventana en la imagen. Entonces el primer parámetro a tomar en cuenta es el tamaño de esta ventana. Se realizaron pruebas con tamaños desde $2px \times 2px$ hasta $40px \times 40px$. Sin embargo, se encontraron mejores resultados con un tamaño de ventana de $15px \times 15px$. Por lo tanto, el tamaño de ventana utilizada en los experimentos presentados a continuación son de $15px \times 15px$.

Ahora esta ventana tiene que hacer una exploración de la imagen, de donde tenemos otro parámetro: el tamaño del paso con el que se moverá la ventana para realizar esta exploración. Si el paso es mayor al tamaño de la ventana, tendremos zonas que no son exploradas entre cada ventana. Por el contrario, si el paso es menor al tamaño de la ventana, tendremos que las ventanas se translapan, esto proporciona más información de la imagen pero requiere un mayor procesamiento de datos. Un compromiso entre el tiempo de procesamiento y una cantidad de datos aceptable, se da al hacer el paso igual al tamaño de la ventana, por lo cual el paso usado fue de $15px$.

Por otro lado, tenemos al número de clases, en la construcción de la base de imágenes se definieron tres clases: el fondo, el espejo y el borde del espejo. No obstante, únicamente se utilizaron dos de estas

clases: la clase que corresponde al fondo y la clase que corresponde al espejo.

Tabla 10.1: Valores utilizados en los parámetros

Parámetro	Valores
Tamaño de ventana	$15px \times 15px$
Paso de ventana	$15px$
Clases	Espejo/Fondo
Vector de entrada	moda, histograma
Bloques de neuronas	10K, 5K
Imágenes de entrenamiento/reconocimiento	1/3, 3/6, 5/25
Ciclos de entenamiento	10, 20, 30

Los demás parámetros se variaron con el objeto de observar el comportamiento de la red, utilizando los valores que se muestran en la tabla 10.1. En lo que respecta al vector de entrada, se utilizaron dos tipos: El primero usa un vector de entrada construido a partir del color moda en la ventana. El segundo usa un vector de entrada construido a partir del histograma de la ventana. Para la cantidad de bloques se realizaron experimentos con 10 mil y 5 mil bloques. En lo que corresponde a la cantidad de imágenes usadas para entrenamiento y reconocimiento, utilicé tres parejas de valores: una imagen para entrenamiento y tres para reconocimiento; tres imágenes para entrenamiento y seis para reconocimiento; cinco imágenes para entrenamiento y veinticinco para reconocimiento.

Los experimentos se pueden dividir por el tipo de vector de entrada, quedando en dos categorías: el vector de entrada formado por la moda y el vector de entrada formado por el histograma. Recordemos que el valor de moda se conforma por un valor, mientras que un histograma se conforma por varios valores o grupos de valores. por lo tanto, el vector de entrada moda es ideal para usarlo en RTC, en cambio para el vector de entrada histograma es necesario utilizar RSC. En ambos tipos de experimentos se utilizaron los mismos valores y cambios (de parámetros) para poder comparar su desempeño. Cada prueba se repitió diez veces, utilizando los mismos parámetros pero generando nuevos umbrales y usando un conjunto de imágenes diferente. En las gráficas presentadas, cada línea representa una repetición del experimento en cuestión. A continuación describo más detalladamente estos experimentos y cómo se construyeron los vectores de entrada.

10.1. Experimento con moda

En esta prueba el vector de entrada se conforma por el valor RGB moda en la región procesada, es decir, el valor RGB mas repetido en la región. Para obtener el valor moda se debe realizar un conteo de cada valor RGB que existe en la región de interés, obteniendo dos listas asociadas: una de los valores RGB y; una de la cantidad de veces que aparece cada valor en la ventana. Finalmente se obtiene el máximo en lista que contiene cantidad de veces y el valor RGB asociado a ese máximo será el valor moda. Por lo tanto el vector de entrada será conformado por 3 valores (RGB) en el rango $[0, 255]$.

10.1.1. Experimento moda 1 (5000 bloques, 1 entrenamiento, 3 prueba)

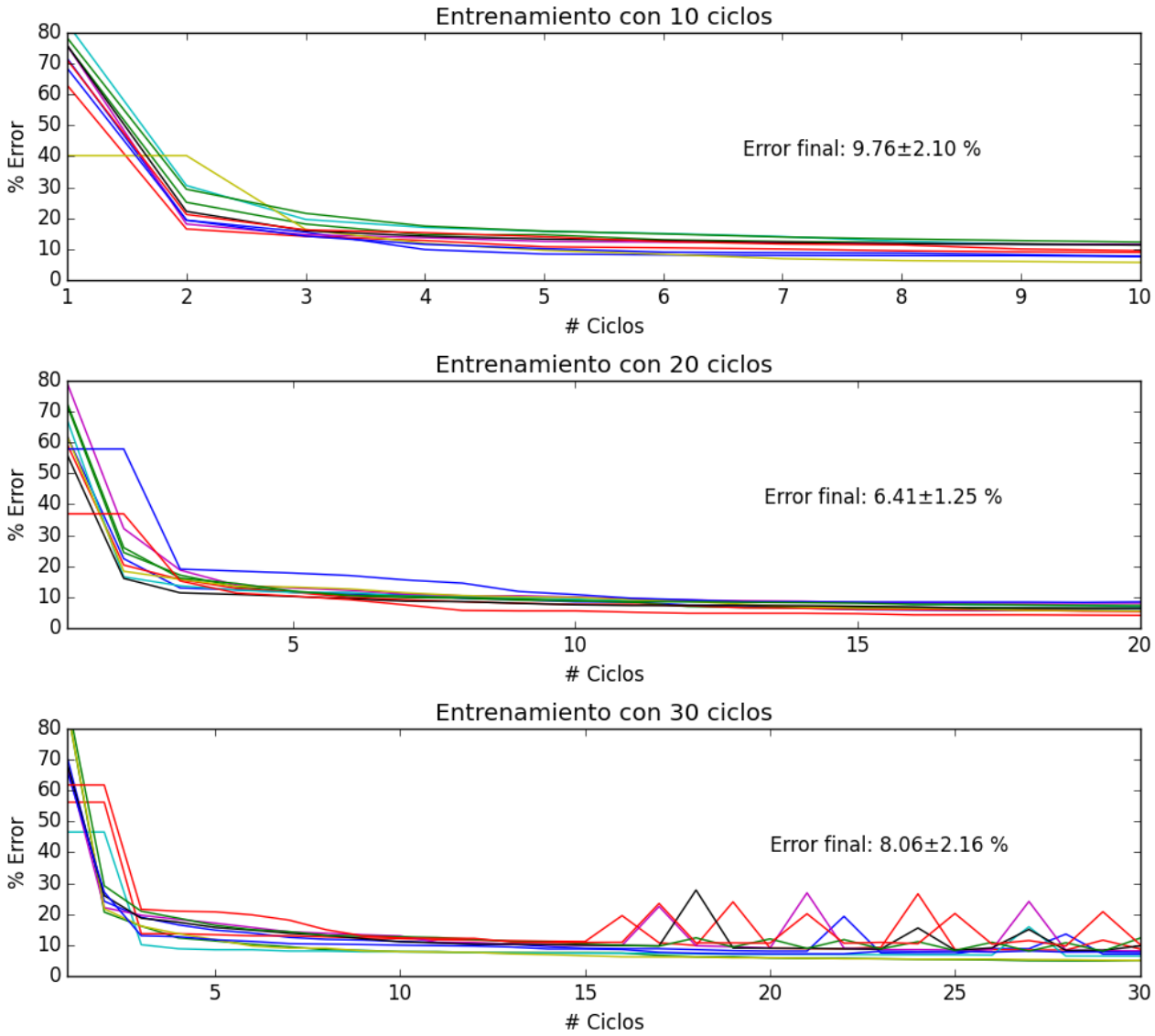


Figura 10.1: Gráfica del % error contra ciclos de entrenamiento usando 5 mil bloques de neuronas, 1 imagen para entrenamiento y 3 imágenes para reconocimiento

En la Fig. 10.1 se muestra el comportamiento del error en entrenamiento contra ciclos de entrenamiento utilizando 5 mil bloques de neuronas, 1 imagen para entrenamiento y 3 imágenes para reconocimiento. En los experimentos con 10 y 20 ciclos de entrenamiento se observa una gran estabilidad y terminan con una baja cantidad de errores (menos del 10 %). Sin embargo en el experimento con 30 ciclos de entrenamiento se observa una oscilación bastante notable a partir del ciclo 15 y hasta el final. No obstante el error final se encuentra alrededor del 10 %.

Cabe notar que únicamente se utilizó una imagen para entrenamiento, lo cual es poco representativo, sin embargo los resultados son buenos.

10.1.2. Experimento moda 2 (5000 bloques, 3 entrenamiento, 6 prueba)

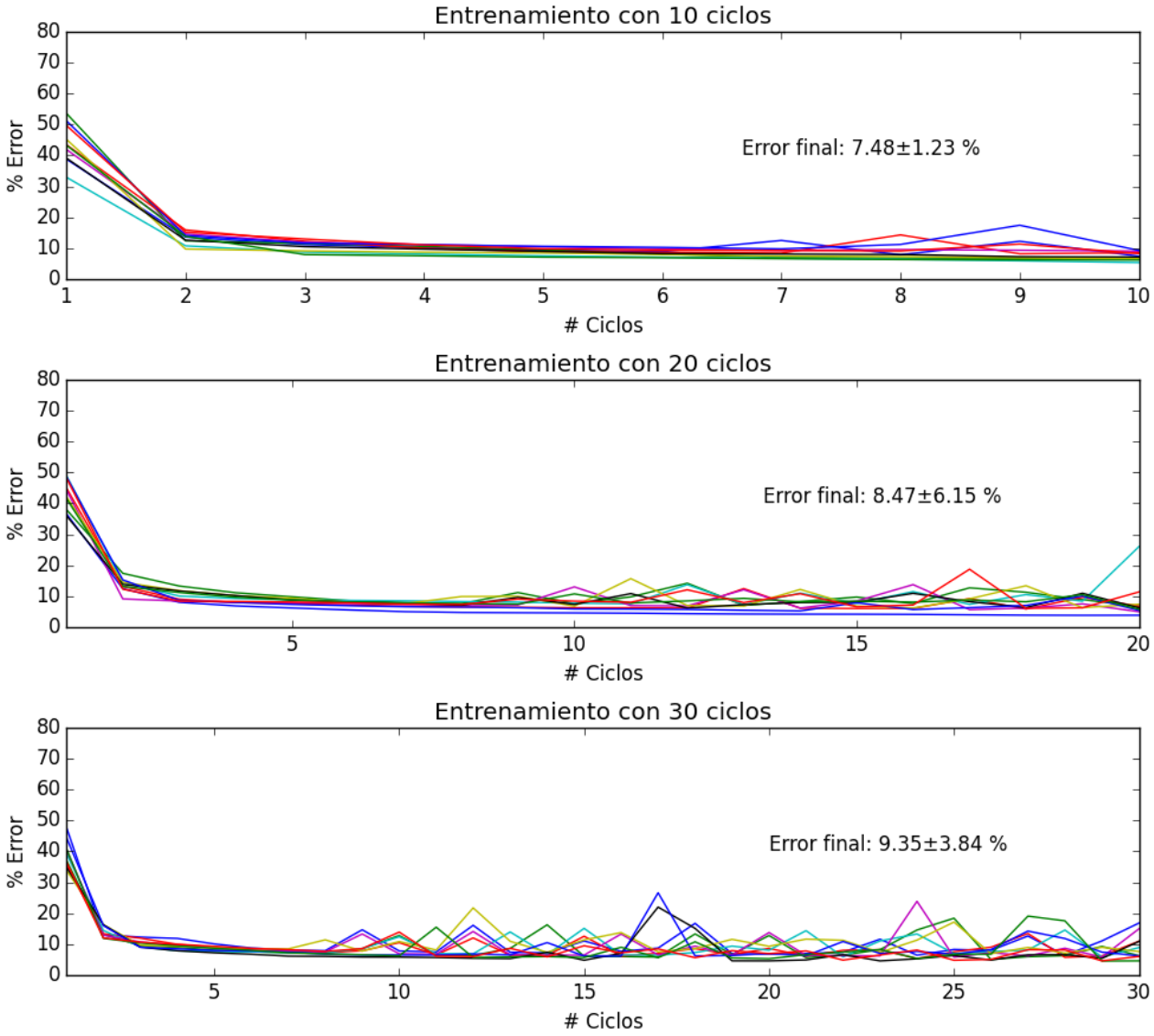


Figura 10.2: Gráfica del % error contra ciclos de entrenamiento usando 5 mil bloques de neuronas, 3 imagenes para entrenamiento y 6 imágenes para reconocimiento

En la Fig. 10.2 se muestra el comportamiento del error en entrenamiento contra ciclos de entrenamiento utilizando 5 mil bloques de neuronas, 3 imagen para entrenamiento y 6 imágenes para reconocimiento. En los experimentos con 10 ciclos de entrenamiento se observa una gran estabilidad y terminan con una baja cantidad de errores (menos del 10 %). En lo que consta a los experimentos con 20 ciclos de entrenamiento, aunque en general también se termina con errores por debajo del 10 %, en algunos casos no es así. Provocando una variabilidad muy grande en cuanto a porcentaje de error final. Por otro lado, en el experimento con 30 ciclos de entrenamiento se observa una oscilación bastante notable a partir del ciclo 7 y hasta el final. No obstante el error promedio final se encuentra alrededor del 10 %.

10.1.3. Experimento moda 3 (5000 bloques, 5 entrenamiento, 25 prueba)

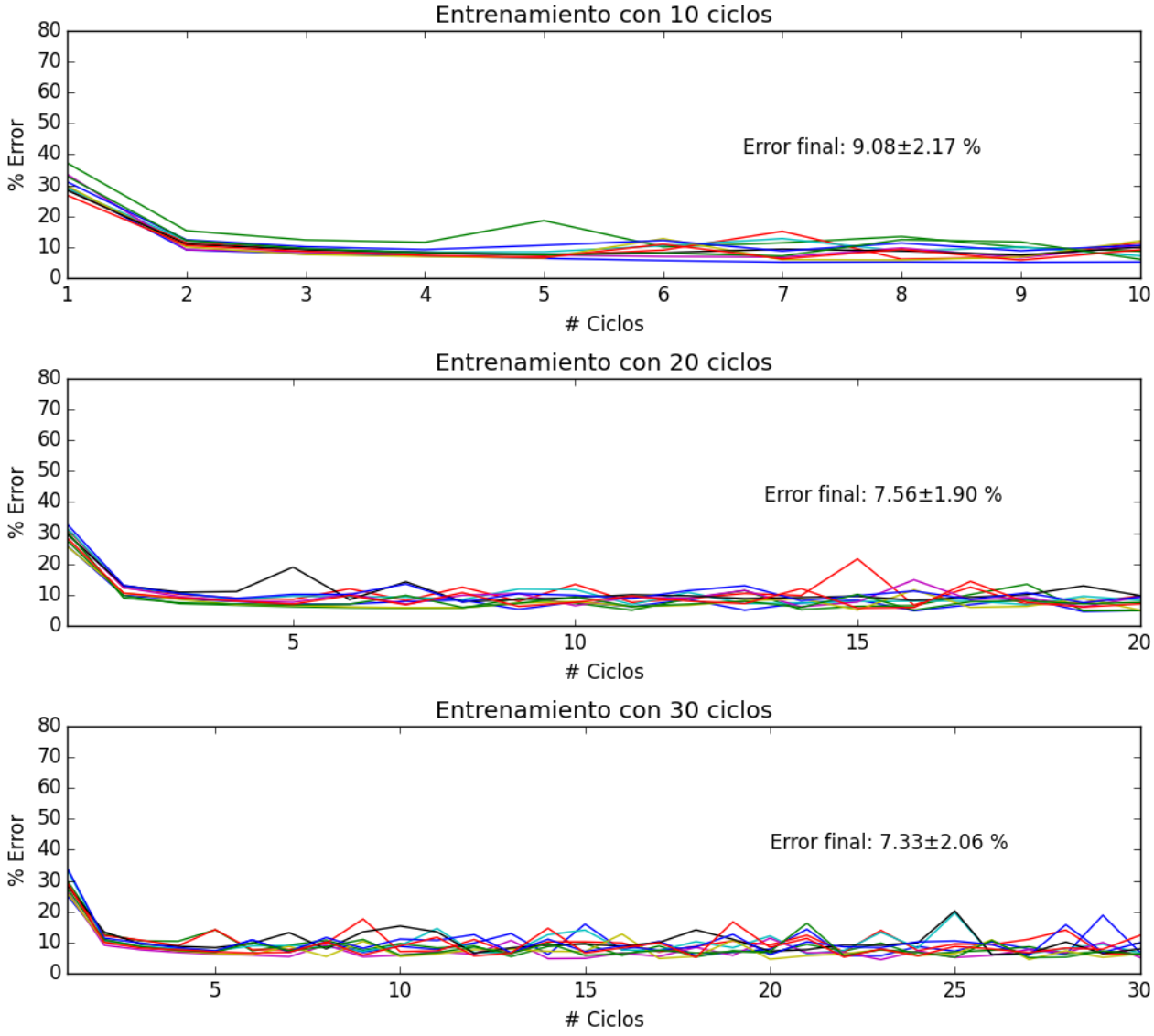


Figura 10.3: Gráfica del % error contra ciclos de entrenamiento usando 5 mil bloques de neuronas, 5 imágenes para entrenamiento y 25 imágenes para reconocimiento

En la Fig. 10.3 se muestra el comportamiento del error en entrenamiento contra ciclos de entrenamiento utilizando: 5 mil bloques de neuronas, 5 imágenes para entrenamiento y 25 imágenes para reconocimiento. En los experimentos con 10 ciclos de entrenamiento se observa estabilidad pero ligeramente menor al experimento moda 2. En los experimentos con 20 ciclos existe un incremento en las oscilaciones pero no son muchas ni muy grandes y al final son menores al 10 %. En el experimento con 30 ciclos se observa una oscilación notable prácticamente desde el inicio y hasta el final. Esta oscilación es mayor que en el experimento anterior y probablemente se debe a que se utilizan más imágenes para entrenamiento, estas imágenes tienen diferentes características que incluso lleguen a generar cierta contradicción (vectores de entrada similares corresponden a clases diferentes en diferentes imágenes).

10.1.4. Experimento moda 4 (10000 bloques, 1 entrenamiento, 3 prueba)

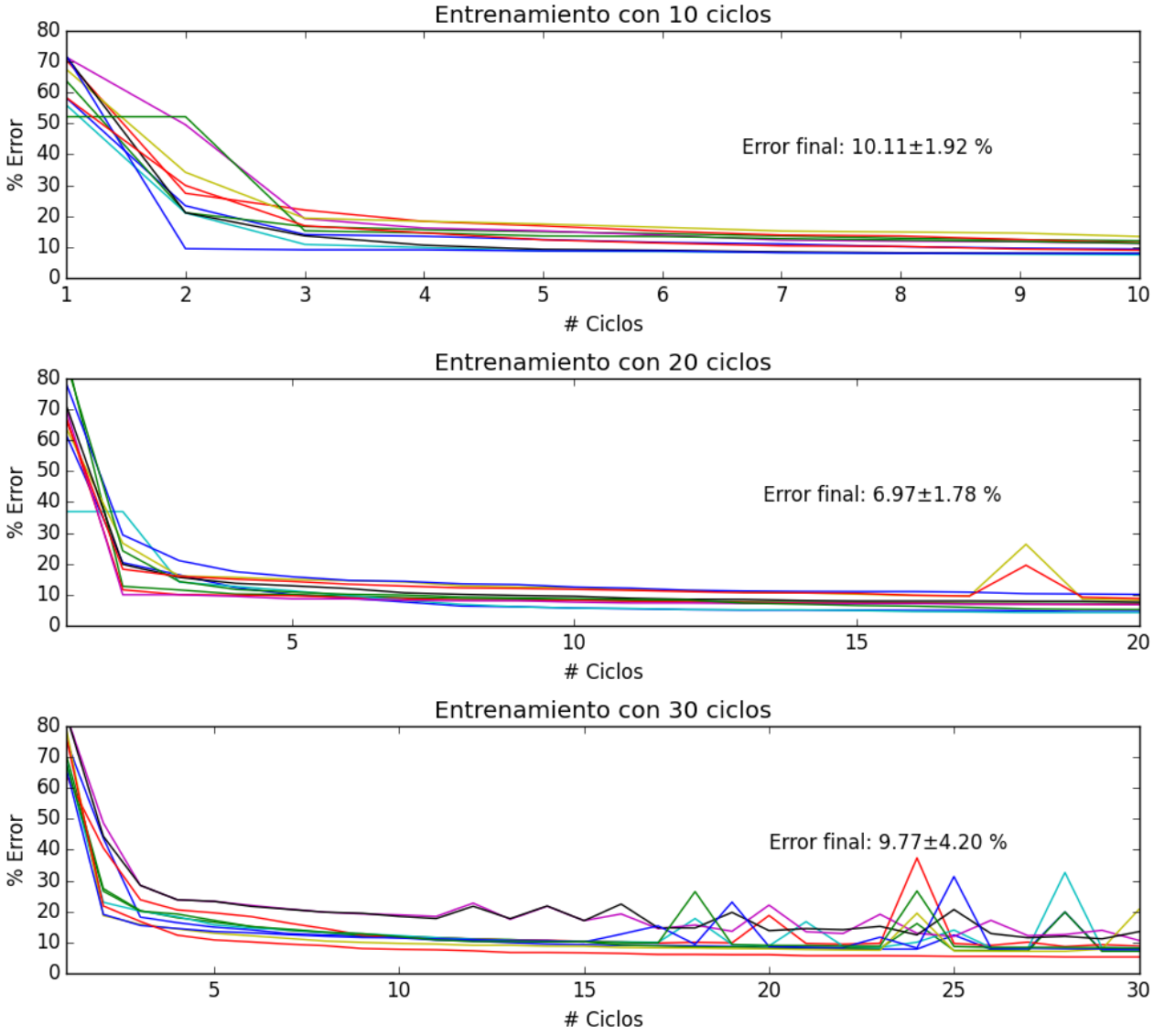


Figura 10.4: Gráfica del % error contra ciclos de entrenamiento usando 10 mil bloques de neuronas, 1 imagen para entrenamiento y 3 imágenes para reconocimiento.

En la Fig. 10.4 se muestra el comportamiento del error en entrenamiento contra ciclos de entrenamiento utilizando: 10 mil bloques de neuronas, 1 imagen para entrenamiento y 3 imágenes para reconocimiento. En los experimentos con 10 y 20 ciclos de entrenamiento se observa una gran estabilidad y terminan con errores menores al 15 %. Sin embargo, como en el experimento con 5 mil bloques, al utilizar 30 ciclos de entrenamiento se observa una oscilación. En este caso la oscilación parece empezar en el ciclo 11 incluso y se mantiene hasta el final. Adicionalmente se observa otra oscilación en la parte final de dos pruebas pertenecientes los experimentos con 20 ciclos. Una posible razón es que al existir mas bloques, se extraen más características de la imagen y algunas de ellas pueden ser comunes a ambas clases. No obstante el error final se encuentra alrededor del 10 %.

10.1.5. Experimento moda 5 (10000 bloques, 3 entrenamiento, 6 prueba)

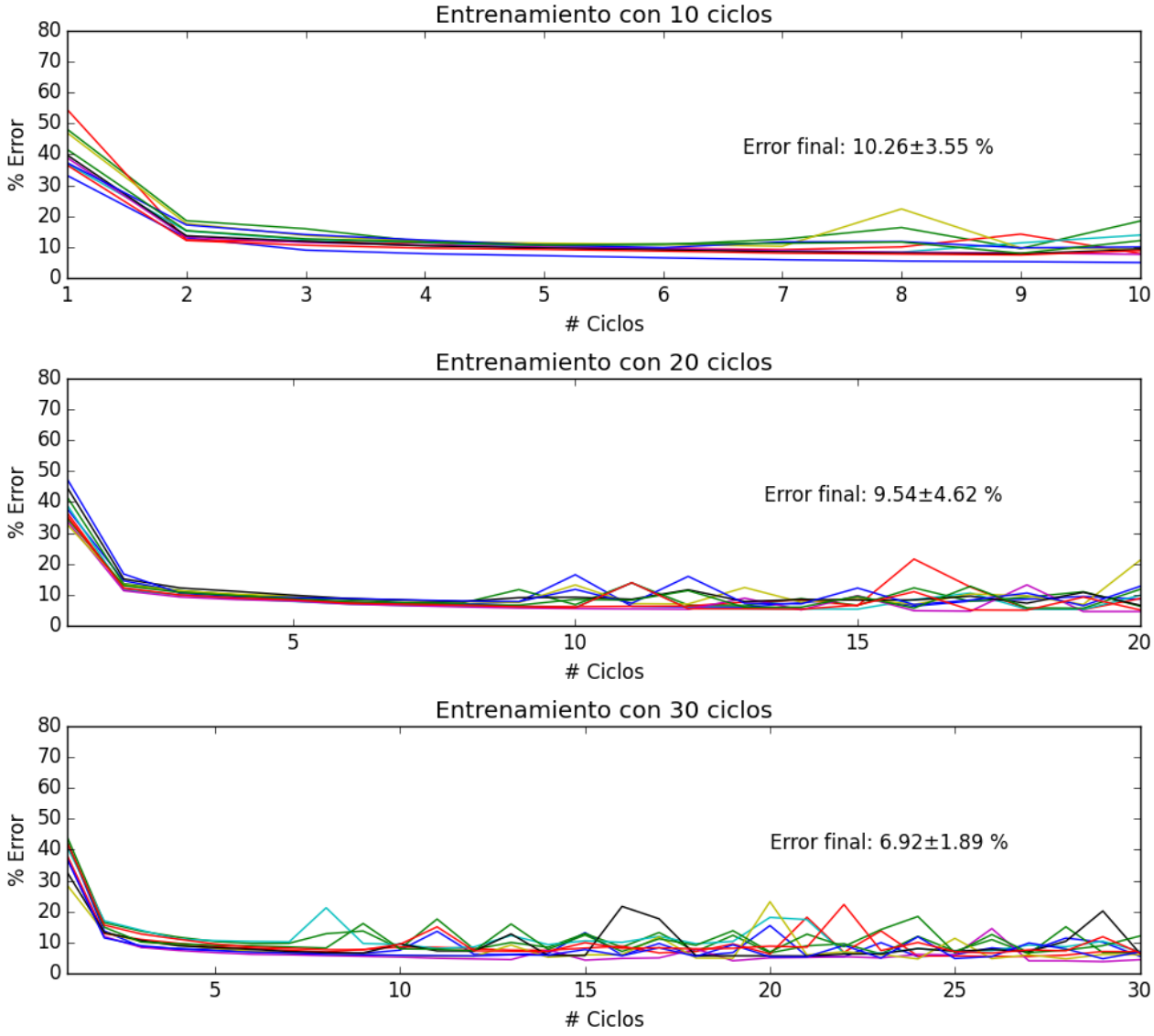


Figura 10.5: Gráfica del % error contra ciclos de entrenamiento usando 10 mil bloques de neuronas, 3 imágenes para entrenamiento y 6 imágenes para reconocimiento.

En la Fig. 10.5 se muestra el comportamiento del error en entrenamiento contra ciclos de entrenamiento utilizando: 10 mil bloques de neuronas, 3 imágenes para entrenamiento y 6 imágenes para reconocimiento. En los experimentos con 10 ciclos se presenta un buena estabilidad aunque al final presenta cierta oscilación. En los experimentos con 20 ciclos de entrenamiento el comportamiento es parecido pero las oscilaciones duran una mayor cantidad de ciclos. Al final terminan con una cantidad de errores menores al 15 %. Como en los experimentos anteriores, al utilizar 30 ciclos de entrenamiento se observa una oscilación, pero en este caso es más marcada y parece empezar un poco antes, en el ciclo 7 incluso y se mantiene hasta el final. Una posible razón es que se combinen la existencia de más bloques con la presencia de una variedad más grande de imágenes, provocando que se encuentren características

comunes a ambas clases. No obstante el error final se encuentra alrededor del 10 %.

10.1.6. Experimento moda 6 (10000 bloques, 5 entrenamiento, 25 prueba)

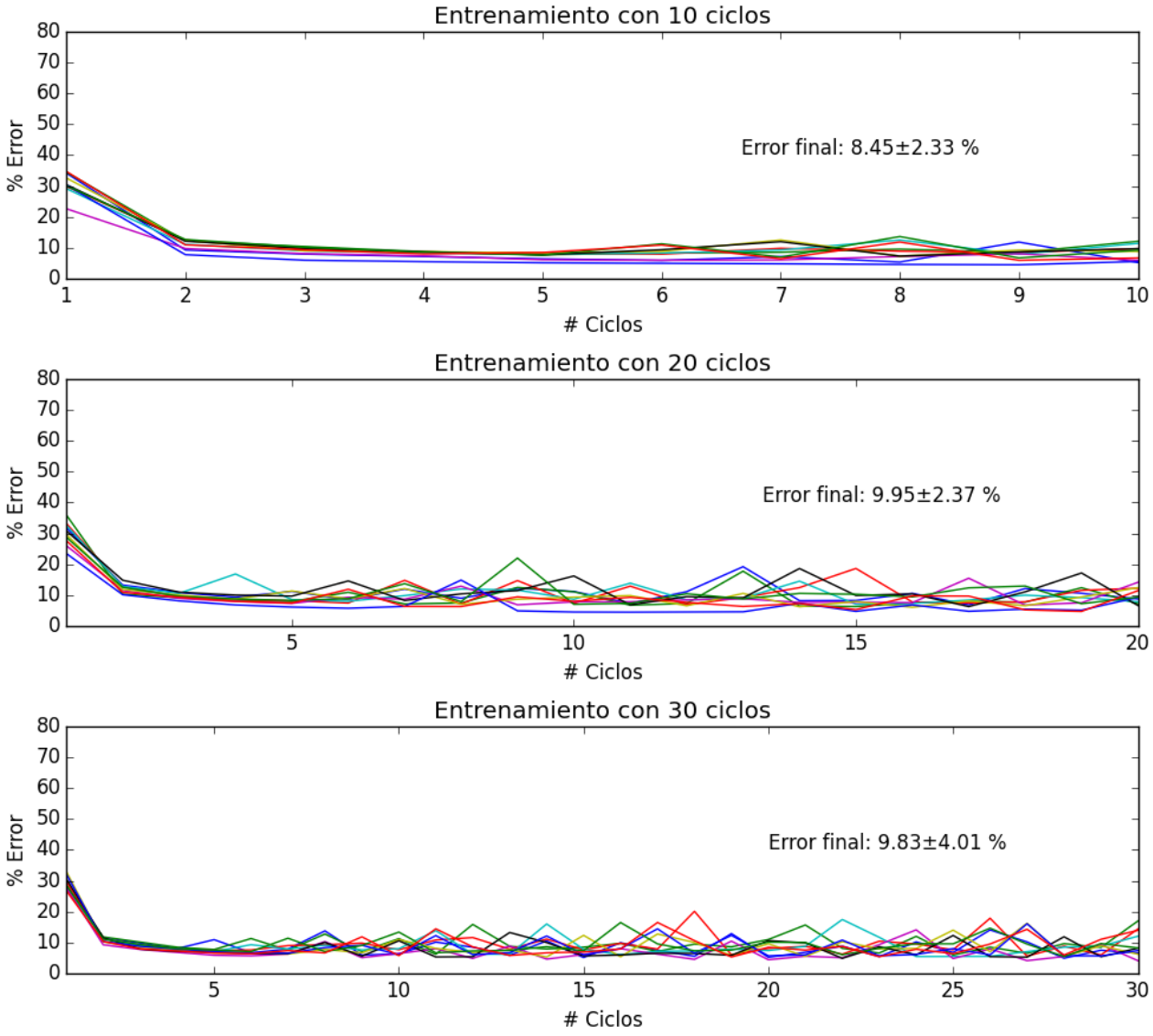


Figura 10.6: Gráfica del % error contra ciclos de entrenamiento usando 10 mil bloques de neuronas, 5 imágenes para entrenamiento y 25 imágenes para reconocimiento.

En la Fig. 10.6 se muestra el comportamiento del error en entrenamiento contra ciclos de entrenamiento utilizando: 10 mil bloques de neuronas, 5 imágenes para entrenamiento y 25 imágenes para reconocimiento. En los experimentos con 10 ciclos se presenta un buena estabilidad aunque al final presenta cierta oscilación. En los experimentos con 20 ciclos de entrenamiento el comportamiento es parecido pero las oscilaciones duran una mayor cantidad de ciclos. Al final terminan con una cantidad de errores alrededor del 10 %.

Como en los experimentos anteriores, al utilizar 30 ciclos de entrenamiento se observa una oscilación,

aunque en este caso tiene una magnitud menor también parece empezar un poco antes, en el ciclo 4 incluso y se mantiene hasta el final.

Una posible razón es que se combinen la existencia de más bloques con la presencia de una variedad más grande de imágenes, provocando que se encuentren características comunes a ambas clases.

No obstante el error final se encuentra alrededor del 10 %.

10.1.7. Resumen de resultados con moda

Tabla 10.2: Errores en entrenamiento y reconocimiento para moda y 5k bloques de neuronas

*	1 entrena/3 reconoce		3 entrena/6 reconoce		5 entrena/25 reconoce	
ciclos	entrena[%]	reconoce[%]	entrena[%]	reconoce[%]	entrena[%]	reconoce[%]
10	9.76 ± 2.10	10.31 ± 2.64	7.48 ± 1.23	9.77 ± 2.11	9.08 ± 2.17	8.92 ± 0.94
20	6.41 ± 1.25	7.85 ± 1.98	8.47 ± 6.15	8.46 ± 1.17	7.56 ± 1.90	9.93 ± 0.83
30	8.06 ± 2.16	12.54 ± 4.82	9.35 ± 3.84	10.53 ± 3.14	7.33 ± 2.06	9.78 ± 1.00

Tabla 10.3: Errores en entrenamiento y reconocimiento para moda y 10k bloques de neuronas

*	1 entrena/3 reconoce		3 entrena/6 reconoce		5 entrena/25 reconoce	
ciclos	entrena[%]	reconoce[%]	entrena[%]	reconoce[%]	entrena[%]	reconoce[%]
10	10.11 ± 1.92	12.76 ± 4.18	10.26 ± 3.55	8.60 ± 1.71	8.45 ± 2.33	9.53 ± 0.93
20	6.97 ± 1.78	10.79 ± 3.13	9.54 ± 4.62	9.35 ± 3.15	9.95 ± 2.37	9.68 ± 1.02
30	9.77 ± 4.20	9.38 ± 3.27	6.92 ± 1.89	9.50 ± 2.34	9.83 ± 4.01	9.36 ± 0.66

En las tablas 10.2 y 10.3, se presenta el porcentaje de error en la última etapa de entrenamiento y la etapa de reconocimiento para las pruebas con 5 mil y 10 mil bloques de neuronas respectivamente. Se puede observar que el porcentaje de error para la etapa de reconocimiento siempre es mayor que en la etapa de entrenamiento. Este resultado es esperado, dado que en la etapa de reconocimiento se utilizan imágenes que no se presentan en la etapa de entrenamiento.

Tanto en las pruebas con 5 mil bloques como con 10 mil, se observan dos características: 1) el porcentaje de error se mantiene alrededor del 10 %; y 2) existen oscilaciones para entrenamientos con una cantidad de ciclos mayor a 10. Estas oscilaciones parecen no cambiar mucho al utilizar una mayor cantidad de imágenes en la etapa de entrenamiento.

Dado que el error no cambia de manera considerable en ninguno de los dos casos, se puede decir que para los experimentos presentados, utilizar una cantidad mayor a 5 mil bloques no representará un cambio considerable al obtenido usando 5 mil bloques.

10.2. Prueba con histogramas de color

Los histogramas aplicados en imágenes, nos brindan información acerca de la distribución de los colores en los píxeles. El primer paso para construir un histograma es definir en cuantos grupos se dividirán los píxeles, así como sus límites. Una vez que se tienen estos grupos se realiza el conteo del número de píxeles que pertenecen a cada grupo. Como resultado para cada grupo tenemos un valor asociado que representa la cantidad de píxeles que se considera pertenecen a éste.

En general se tienen dos formas de hacer un histograma de color a partir de las componentes RGB: Hacer un histograma que tome en cuenta las tres componentes; o hacer tres histogramas, uno para cada componente (RGB).

En esta prueba se optó por utilizar los tres histogramas de las componentes RGB. Para cada histograma se utilizaron 8 grupos de igual tamaño, como cada componente es de un byte, se tienen 256 posibles valores y por lo tanto los grupos son de tamaño $256/8 = 32$. Esto es, el primer grupo se conformará por los valores entre 0 y 32; el segundo por los valores entre 32 y 64; el tercer grupo por los valores entre 64 y 96; y así hasta abarcar los 256 valores.

Por otro lado, al tener 8 grupos y 3 componentes, el vector de entrada tendrá un tamaño total de $8 \times 3 = 24$ valores. En este caso se aplicó RSC con un subespacio de 3. Recordando que se utilizaron ventanas de 15×15 píxeles, tenemos que cada valor del vector de entrada puede ser a lo mas $15 \times 15 = 225$. Por lo tanto, para almacenar cada grupo del histograma es suficiente un byte.

A continuación, se muestran las gráficas correspondientes a las pruebas con este vector de entrada. Con objeto de poder comparar los resultados con los obtenidos en las pruebas con moda, se realizaron pruebas con los mismos parámetros.

10.2.1. Prueba histograma 1 (5000 bloques, 1 entrenamiento, 3 prueba)

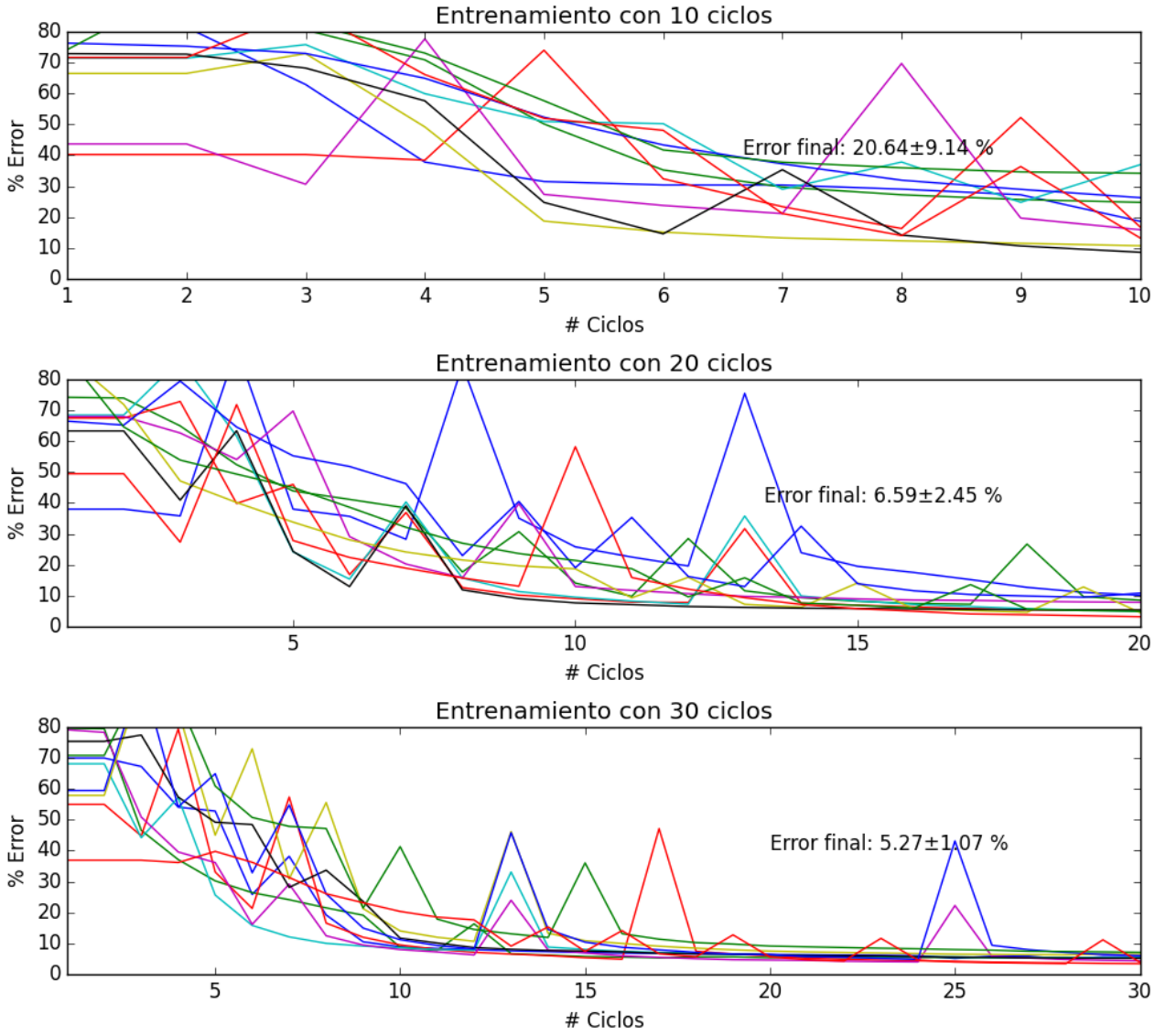


Figura 10.7: Gráfica del %error contra ciclos de entrenamiento, usando 5 mil bloques, 1 imagen para entrenamiento y 3 para reconocimiento.

En la Fig. 10.7 se muestran las gráficas del porcentaje de error contra ciclos de entrenamiento correspondientes a 5 mil bloques de neuronas, 1 imagen para entrenamiento y 3 imágenes para reconocimiento. En los experimentos correspondientes a 10 ciclos se observan grandes oscilaciones durante todo el proceso de entrenamiento. En este caso el error final es grande, en algunos casos incluso mayor al 40 %.

En los experimentos correspondientes a 20 ciclos también se observan oscilaciones, sin embargo tienden a desaparecer hacia la parte final de la prueba. En este caso el error final es bajo, menor al 20 %.

Finalmente en los experimentos correspondientes a 30 ciclos, aunque también se presentan oscilaciones, éstas son de menor magnitud y tienden a desaparecer hacia el final de las pruebas. En este caso el error final es mas bajo que en los casos anteriores, incluso menor al 10 %.

10.2.2. Prueba histograma 2 (5000 bloques, 3 entrenamiento, 6 prueba)

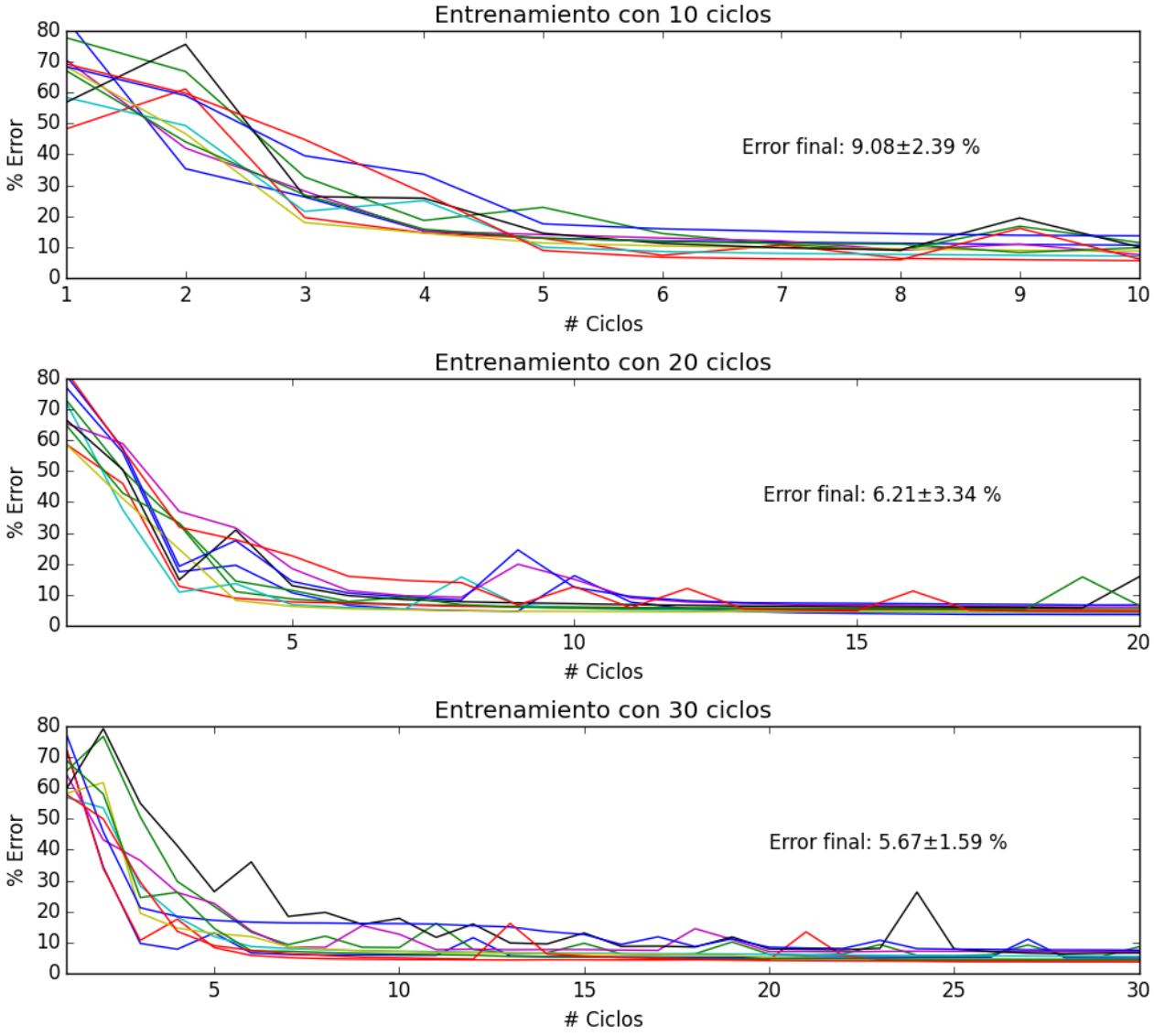


Figura 10.8: Gráfica del %error contra ciclos de entrenamiento, usando 5 mil bloques, 3 imágenes para entrenamiento y 6 para reconocimiento

En la Fig. 10.8 se muestran las gráficas del porcentaje de error contra ciclos de entrenamiento para pruebas correspondientes a 5 mil bloques de neuronas, 3 imágenes para entrenamiento y 6 imágenes para reconocimiento. En los experimentos correspondientes a 10 ciclos se observa un comportamiento estable a pesar de tener pequeñas variaciones al inicio del proceso de entrenamiento. En este caso el error final es menor al 15 %. En los experimentos correspondientes a 20 ciclos también se observan ligeras oscilaciones, sin embargo tienden a desaparecer hacia la parte final de la prueba. En este caso el error final es ligeramente menor al presentado al usar 10 ciclos. Finalmente en los experimentos correspondientes a 30 ciclos, aunque también se presentan oscilaciones, éstas son de menor magnitud. El error final es mas bajo que en los casos anteriores, quedando apenas arriba del 5 %.

10.2.3. Prueba histograma 3 (5000 bloques, 5 entrenamiento, 25 prueba)

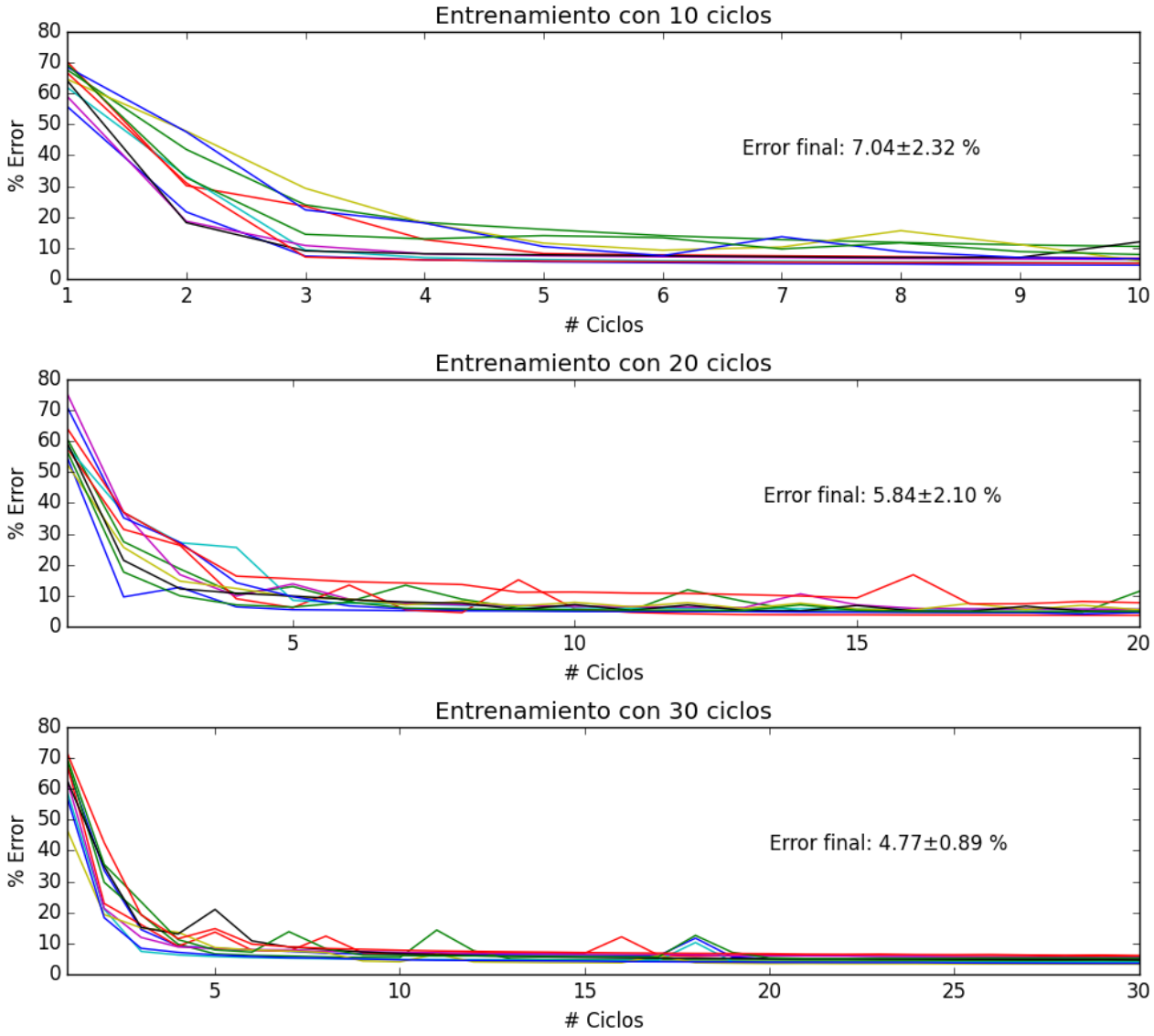


Figura 10.9: Gráfica del % error contra ciclos de entrenamiento, usando 5 mil bloques, 5 imágenes para entrenamiento y 25 para reconocimiento

En la Fig. 10.9 se muestran las gráficas del porcentaje de error contra ciclos de entrenamiento para las pruebas correspondientes a 5 mil bloques, 5 imágenes para entrenamiento, 25 imágenes para reconocimiento. En los experimentos correspondientes a 10 ciclos se observa un comportamiento estable a pesar de tener pequeñas variaciones al inicio del proceso de entrenamiento. En este caso el error final es menor al 10 %.

En los experimentos correspondientes a 20 ciclos también se observan ligeras oscilaciones que tienden a desaparecer hacia la parte final de la prueba. En este caso el error final se encuentra alrededor del 5 %. Finalmente en los experimentos correspondientes a 30 ciclos, tanto las oscilaciones presentadas, como el error final son aún de menor magnitud que en el experimento anterior.

10.2.4. Prueba histograma 4 (10000 bloques, 1 entrenamiento, 3 prueba)

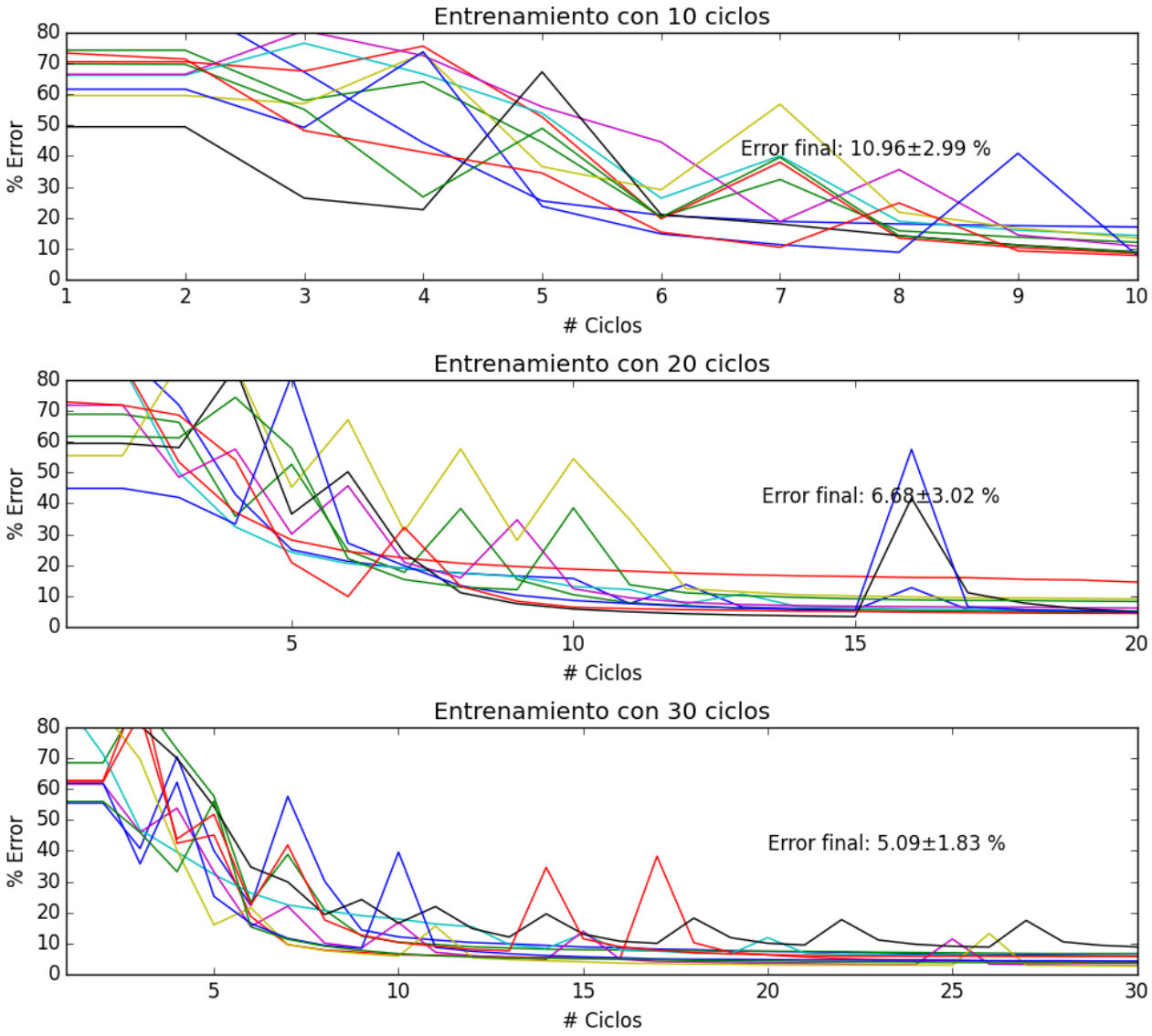


Figura 10.10: Gráfica del %error contra ciclos de entrenamiento, usando 10 mil bloques, 1 imagen para entrenamiento y 3 para reconocimiento

En la Fig. 10.10 se muestran las gráficas del porcentaje de error contra ciclos de entrenamiento para pruebas correspondientes a 10 mil bloques de neuronas, 1 imagen para entrenamiento y 3 imágenes para reconocimiento. En los experimentos correspondientes a 10 ciclos se observan grandes oscilaciones durante todo el proceso de entrenamiento. En este caso el error final es menor del 15 %.

En los experimentos correspondientes a 20 ciclos también se observan oscilaciones, sin embargo tienden a desaparecer hacia la parte final de la prueba. En este caso el error final es bajo, menor al 10 %.

Finalmente en los experimentos correspondientes a 30 ciclos, aunque el comportamiento es parecido, el error final es mas bajo que en los casos anteriores (alrededor de %5).

10.2.5. Prueba histograma 5 (10000 bloques, 3 entrenamiento, 6 prueba)

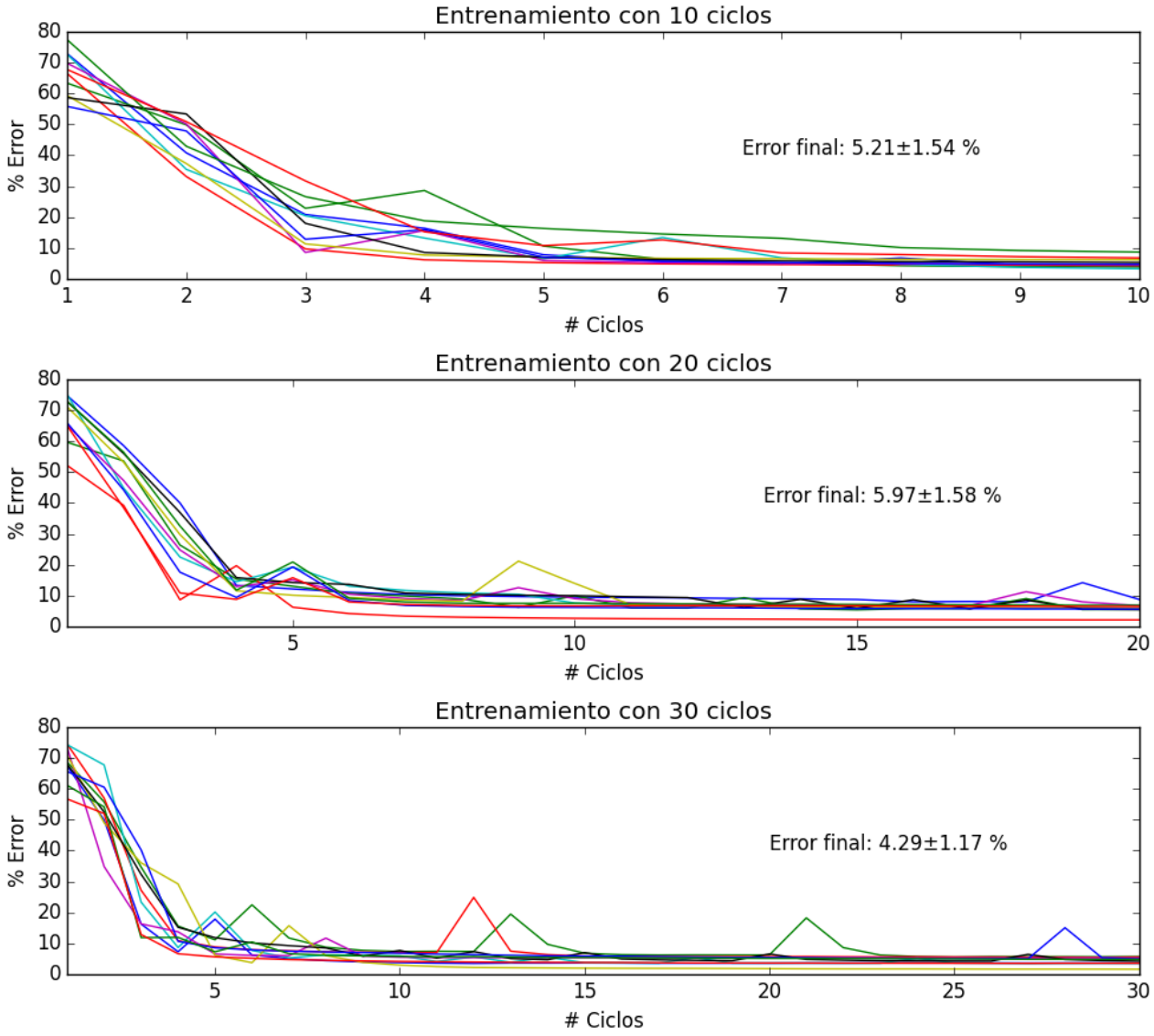


Figura 10.11: Gráfica del %error contra ciclos de entrenamiento, usando 10 mil bloques, 3 imágenes para entrenamiento y 6 para reconocimiento

En la Fig. 10.11 se muestran las gráficas del porcentaje de error contra ciclos de entrenamiento para pruebas correspondientes a 10 mil bloques de neuronas, 3 imágenes para entrenamiento y 6 imágenes para reconocimiento. En los experimentos correspondientes a 10 ciclos se observan oscilaciones durante la primera mitad del proceso de entrenamiento. En este caso el error final es menor al 10 %.

En los experimentos correspondientes a 20 ciclos también se observan oscilaciones que en su mayoría se encuentran en los primeros 5 ciclos. El error final aumenta levemente con respecto a la prueba con 10 ciclos. Finalmente en los experimentos correspondientes a 30 ciclos, además de las oscilaciones iniciales se pueden observar algunas otras de manera aleatoria a lo largo del entrenamiento. No obstante, el error final es menor a los casos con 10 y 20 ciclos quedando debajo del 5 %.

10.2.6. Prueba histograma 6 (10000 bloques, 5 entrenamiento, 25 prueba)

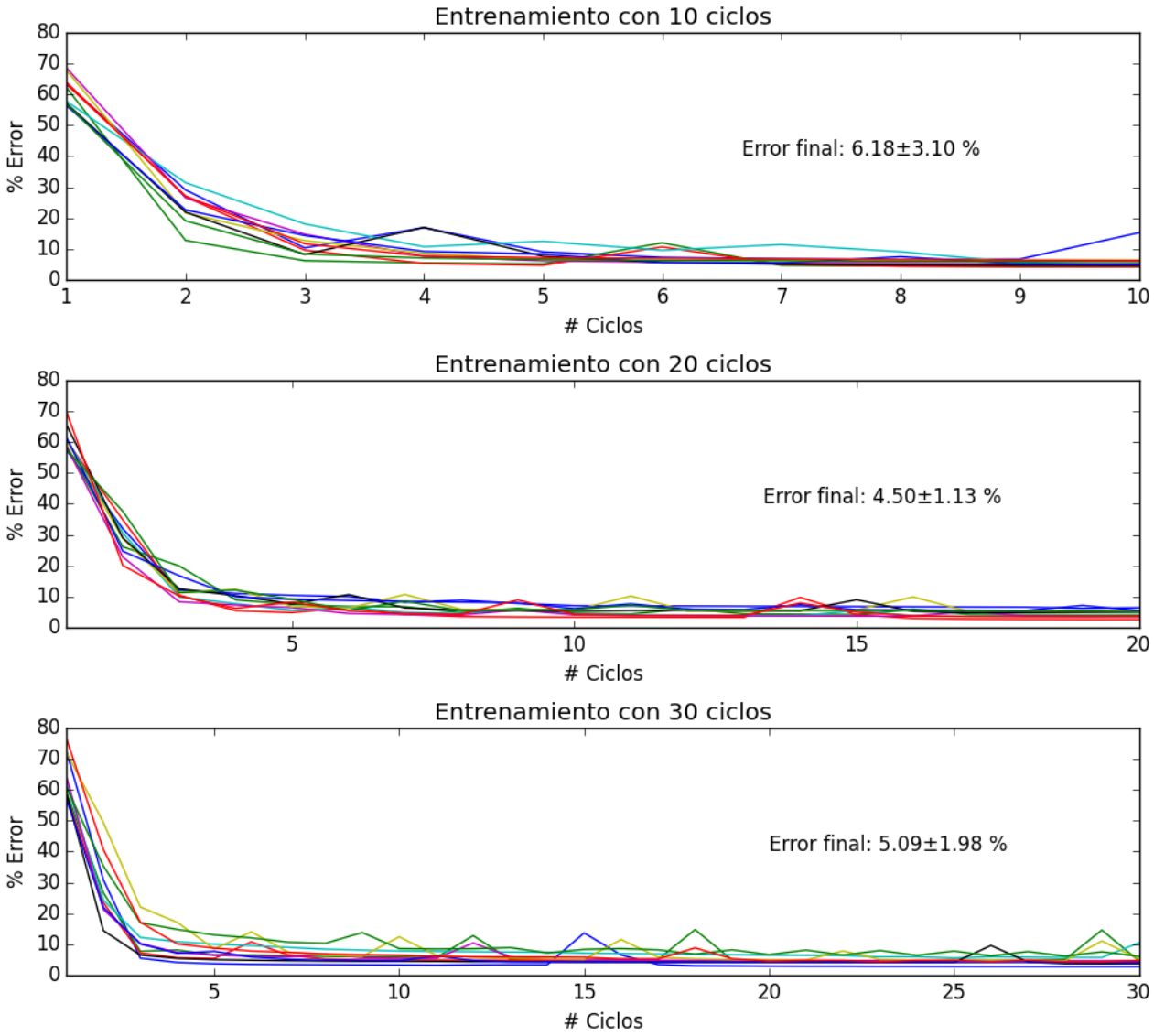


Figura 10.12: Gráfica del %error contra ciclos de entrenamiento, usando 10 mil bloques, 5 imágenes para entrenamiento y 25 para reconocimiento

En la Fig. 10.12 se muestran las gráficas del porcentaje de error contra ciclos de entrenamiento para las pruebas correspondientes a 10 mil bloques, 5 imágenes para entrenamiento, 25 imágenes para reconocimiento. En los experimentos correspondientes a 10, 20 y 30 ciclos se observan oscilaciones muy pequeñas que le permiten a la red disminuir los errores de una manera mas acelerada. Prácticamente en el ciclo 5 de todos los casos el error se mantiene hasta el final de la prueba.

En la mayoría de los casos el error final está alrededor del 5 % y presenta pocas variaciones, lo que representa un buen resultado.

10.2.7. Resumen de resultados con histograma

Tabla 10.4: Errores en entrenamiento y reconocimiento para histograma y 5k bloques de neuronas

*	1 entrena/3 reconoce		3 entrena/6 reconoce		5 entrena/25 reconoce	
ciclos	entrena[%]	reconoce[%]	entrena[%]	reconoce[%]	entrena[%]	reconoce[%]
10	20.64 ± 9.14	19.77 ± 7.98	9.08 ± 2.39	11.16 ± 6.36	7.04 ± 2.32	9.50 ± 2.20
20	6.59 ± 2.45	17.39 ± 9.85	6.21 ± 3.34	12.41 ± 5.16	5.84 ± 2.10	8.12 ± 1.91
30	5.27 ± 1.07	15.21 ± 10.02	5.67 ± 1.59	10.69 ± 4.60	4.77 ± 0.89	7.74 ± 1.96

Tabla 10.5: Errores en entrenamiento y reconocimiento para histograma y 10k bloques de neuronas

*	1 entrena/3 reconoce		3 entrena/6 reconoce		5 entrena/25 reconoce	
ciclos	entrena[%]	reconoce[%]	entrena[%]	reconoce[%]	entrena[%]	reconoce[%]
10	10.96 ± 2.99	19.59 ± 9.54	5.21 ± 1.54	10.63 ± 2.71	6.18 ± 3.10	9.20 ± 2.10
20	6.68 ± 3.02	14.10 ± 10.96	5.97 ± 1.58	9.76 ± 3.53	4.50 ± 1.13	6.51 ± 1.20
30	5.09 ± 1.83	15.12 ± 7.37	4.29 ± 1.17	11.27 ± 3.45	5.09 ± 1.98	7.39 ± 1.79

En las tablas 10.4 y 10.5 se muestra el porcentaje de error para la última etapa de entrenamiento y la etapa de reconocimiento para las pruebas con 5 mil y 10 mil bloques de neuronas respectivamente. Como pasó en el caso de las pruebas con moda, los errores en la etapa de reconocimiento son mayores que los presentados en la última etapa de entrenamiento. La razón es la misma, en la etapa de reconocimiento se presentan imágenes que no se presentaron en la etapa de entrenamiento, por lo que son desconocidas por la red.

En este caso se observa que la cantidad de errores disminuye con el aumento de ciclos de entrenamiento, mientras que la cantidad de imágenes utilizadas en el entrenamiento no influye de manera importante.

10.3. Discusión de los resultados

En las pruebas con moda se observa una gran estabilidad en los primeros ciclos de entrenamiento, mientras que mas allá del ciclo 5 se empiezan a observar oscilaciones. En cambio, en las pruebas con histograma se muestran oscilaciones en los primeros ciclos y una mejor estabilidad en los últimos ciclos.

En ambos casos, se presentan comportamientos muy similares tanto al usar 5 mil bloques como al usar 10 mil, por lo que esta variación no parece afectar las pruebas realizadas. Sin embargo, el procesamiento necesario para utilizar 10 mil bloques de neuronas es mucho mayor que el requerido al utilizar únicamente 5 mil.

En el caso de las pruebas con moda no parece haber cambios significativos al usar una mayor cantidad de imágenes para entrenamiento, mientras que en las pruebas con histograma, al usar más imágenes para entrenamiento se obtiene una reducción en las oscilaciones presentadas.

En ambos casos se obtienen resultados aceptables, pero cabe hacer dos observaciones importantes: 1) las pruebas con histograma parecen ser más estables que las realizadas con moda y; 2) el procedimiento para realizar el cálculo de la moda es más sencillo y requiere menor tiempo de procesamiento que el tiempo requerido para procesar los histogramas.

Dadas estas características, la combinación RTC/moda puede ser más adecuado cuando se requiera un menor tiempo de procesamiento y entrenamiento; pero se tolere cierta cantidad de errores. En cambio la combinación RSC/histograma será más adecuado cuando se requiera un mejor desempeño con un nivel muy bajo de errores; pero se disponga de los recursos necesarios para llevar a cabo un procesamiento más intenso.

Dado que el sistema que realizará el cálculo de la posición de los espejos, así como el control del manipulador se definirán en una etapa posterior, ambas opciones son posibles y se definirá cual usar posteriormente.

Capítulo 11

Conclusiones

El desarrollo de tecnología que permita el aprovechamiento de la energía solar en México es fundamental dado las condiciones privilegiadas condiciones con que cuenta nuestro país.

El uso de visión computacional y las redes neuronales se encuentra muy difundida actualmente en diversas áreas, entre ellas la automatización, sin embargo algunos modelos utilizados pueden llegar a ser bastante complejos dificultando su uso. Modelos sencillos como el presentado por el RTC/RSC son capaces de resolver problemas prácticos de manera eficiente.

La red neuronal RTC tiene rápida convergencia al aplicarla en características derivadas de imágenes de color, lo que permite un rápido entrenamiento. En las pruebas realizadas se obtienen errores del orden del 10 % o menores desde 7 ciclos de entrenamiento, para ambos tipos de vectores de entrada.

En los experimentos realizados se observan resultados similares utilizando 5 mil bloques de neuronas que al usar 10 mil bloques de neuronas. Por lo tanto, el uso de 5 mil bloques es suficiente para estos vectores de entrada. Por otro lado, para el caso del vector de entrada histograma se tuvo que entre más imágenes se utilicen en la etapa de entrenamiento mejores resultados se obtienen.

Las simulaciones realizadas prueban que, mediante la red neuronal RTC/RSC, es posible el reconocimiento de los espejos triangulares planos utilizados para la construcción de concentradores solares. Dicho reconocimiento se realizó a partir de características obtenidas de imágenes a color. Sin embargo, antes de poder calcular la posición de los espejos en el contenedor, se debe poder detectar que dos o más espejos se encuentran juntos. Algunas posibles soluciones se proponen mas adelante en la sección trabajo futuro.

Se obtuvieron dos posibles maneras de reconocimiento de espejos: una que presenta oscilaciones en cuanto a sus resultados pero requiere poca capacidad de procesamiento (RTC/moda); y otra que es mucho más estable pero sus necesidades de procesamiento son mayores (RSC/histograma).

Trabajo futuro

El mayor inconveniente encontrado es que los espejos tienden a juntarse en el contenedor, dificultando la delimitación entre ellos y provocando que, a la salida del clasificador, dos o más espejos aparezcan como un solo objeto.

Dado este problema, se proponen dos posibles soluciones en las cuales trabajar. La primera es encontrar un vector de entrada que permita una mejor delimitación entre espejos. La segunda es desarrollar un post procesamiento que realice la separación de los espejos y facilite el cálculo de su posición.

Referencias

[1] "The First Decade:2004-2014, 10 Years of Renewable Energy Progress," Renewable Energy Policy Network for the 21st Century. [Online]. Available: http://www.ren21.net/spotlight/10-years-report

[2] "Technology Roadmap Solar Thermal Electricity 2014," International Energy Agency. [Online]. Available: https://www.iea.org/publications/freepublications/publication/technology-roadmap-solar-thermal-electricity---2014-edition.html

[3] (2016, May) Maps of direct normal irradiation (dni). [Online]. Available: http://solargis.info/doc/free-solar-radiation-maps-DNI

[4] J. Franco, L. Saravia, V. Javi, R. Caso, and C. Fernandez, "Pasteurization of goat milk using a low cost solar concentrator," *Solar Energy*, vol. 82, pp. 1088–1094, 2008.

[5] J. Escobedo-Alatorre, M. Tecpoyotl-Torres, O. G. Martínez, J. G. Vera-Dimas, J. Campos-Alvarez, M. Torres-Cisneros, and Sánchez-Mondragón, "A Prototype of Plannar Autonomous Solar Concentrator," in *III Conference of University of Guanajuato IEEE students chapter*, Salamanca, Gto, Nov 2009, pp. 33–36.

[6] N. Khuchua, R. Melkadze, and A. Moseshvili, "New-type Solar Concentrator Concept - Approach to Reduced-Cost CPV Module Technology," in *Photovoltaic Specialist Conference (PVSC) 2015 IEEE 42nd*, 2015.

[7] M. Vivar, J. Daniel, I.L.Skryabin, V. A. Everett, A. W. Blakers, L.Suganthi, and S. Iniyan, "A Hybrid Solar Linear Concentrator Prototype in India," in *Photovoltaic Specialist Conference (PVSC), 35th IEEE*, Jun 2010, pp. 3092–3097.

[8] L. Li and S. Dubowsky, "A New Design Approach for Solar Concentrating Parabolic Dish Based on Optimized Flexible Petals," *Mechanism and Machine Theory*, vol. 46, pp. 1536–1548, 2011.

[9] K. Lovegrove, G. Burgess, and J. Pye, "A New 500m2 Paraboidal Dish Solar Concentrator," *Solar Energy*, vol. 85, pp. 620–626, 2011.

[10] E. Kussul, T. Baidyk, O. Makeyev, F. J. Lara Rosano, J. M. Saniger Blesa, and N. Bruce, "Development of micro mirror solar concentrator," in *Proceedings of the 2nd IASME/WSEAS International Conference on Energy and Enviroment*, Portovoz, Slovenia, may 2007, pp. 293–298.

[11] (2016, january) www.webster-dictionary.org/definition/computer%20vision.

[12] T. Mitchell and M. Sarhadi, "A machine vision system using fast texture analysis for automated visual inspection," *Systems Engineering*, pp. 399–402, 1992.

[13] X. Wei, K. Jia, J. Lan, Y. Li, Y. Zeng, and C. Wang, "Automatic Method of Fruit Object Extraction Under Complex Agricultural Background for Vision System of Fruit Picking Robot," *Optik*, vol. 125, pp. 5684–5689, 2014.

[14] K. Tanigaki, T. Fujiura, A. Akase, and J. Imagawa, "Cherry-harvesting robot," *Computers and Electronics in Agriculture*, vol. 63, pp. 65–72, 2008.

[15] E. M. de Oliveira, D. S. Leme, B. H. G. Barbosa, M. P. Rodarte, and R. G. F. A. Pereira, "A computer Vision System for Coffe Beans Classification Based on Computational Intelligence Techniques," *Journal of food engineering*, vol. 171, pp. 22–27, 2016.

[16] S. Shafiee, S. Minaei, N. Moghaddam-Charkari, and M. Barzegar, "Honey characterization Using Computer Vision System and Artificial Neural Networks," *Food Chemistry*, vol. 159, pp. 143–150, 2014.

[17] H. Celik, L. Dülger, and M. Topalbekiroglu, "Development of a machine vision system: real-time fabric detection and classification with neural networks," *The Journal of the Textile Institute*, vol. 105, no. 6, pp. 575–585, 2014.

[18] J. Cunha, R. Ferreira, and N. Lau, "Computer Vision and Robotic Manipulation for Automated Feeding of Cork Drillers," *Materials and Design*, vol. 82, pp. 290–296, 2015.

[19] W. Wu, X. Wang, G. Huang, and D. Xu, "Automatic gear sortin system based on monocular vision," *Digital Communications and Networks*, vol. 1, pp. 284–291, 2015.

[20] J. K. Ryeu and H. S. Chung, "Chaotic recurrent neural networks and their application to speech recognition," *Neurocomputing*, vol. 13, pp. 281–294, 1996.

[21] O. Günay and A. E. Cetin, "Real-time dynamic texture recognition using random sampling and dimension reduction," *Image Processing (ICIP)*, pp. 3087–3091, 2015.

[22] I. Khosa and E. Pasero, "Artificial neural network classifier for quality inspection of nuts," in *International Conference on Robotics and Emerging Allied Technologies in Engineering (iCREATE). Islamabad, Pakistan*, April 2014, pp. 103–108.

[23] A. Hebboul, F. Hachouf, and A. Boulemnadjel, "A new incremental neural network for simultaneous clustering and classification," *Neurocomputing*, vol. 169, pp. 89–99, 2015.

[24] S. Ong, N. Yeo., K. Lee, and Y. V. D. Cao, "Segmentation of color images using a two-stage self-organizing network," *Image and Vision Computing*, vol. 20, pp. 279–289, 2012.

[25] F. K., "Neocognitron: A self-organizing neural for a mechanism of pattern recognition unnafected by shift in position," *Biological Cybernetics*, vol. 36, pp. 193–202, 1980.

[26] M. A. El-Sayed, Y. A. Estaitia, and M. A. Khafaghy, "Automated edge detection using convolutional neural network," *International Journal of Advanced Computer Science and Applications*, vol. 4, no. 10, pp. 11–17, 2013.

[27] D. M. Tsai, "Boundary-based corner detection using neural networks," *Pattern Recognition*, vol. 30, no. 1, pp. 85–97, 1997.

[28] S. H. Subri, H. Haron, and R. Sallenhuddin, "Neural network corner detection of vertex chain code," *AIML Journal*, vol. 6, no. 1, pp. 37–43, 2006.

[29] B. Meftah, O. Lezoray, and A. Benyettou, "Segmentation and edge detection based on spiking neural network model," *Neural Processing Letters*, vol. 32, no. 2, pp. 131–146, October 2010.

[30] W. E. Yüksel, "Edge detection in noisy images by neuro-fuzzy processing," *International Journal of Electronics and Communications*, vol. 61, pp. 82–89, 2007.

[31] T. Ohyama, "Neural network-based regions detection," *Neural Networks*, pp. 1456–1459, 1995.

[32] B. Sowmya and B. S. Rani, "Color image segmentation using fuzzy clustering techniques and competitive neural network," *Aplied Soft Computing*, vol. 11, pp. 3170–3178, 2011.

[33] E. Kussul, T. Baidyk, and D. Wunsch, *Neural Networks and Micro Mechanics.* Springer, 2010.

[34] T. Baidyk and E. Kussul, *Redes neuronales, visión computacional y micromecánica.* Itaca, 2009.

[35] T. Baidyk, E. Kussul, and O. Makeyev, "Texture recognition with random subspace neural classifier," *WSEAS Transactions on Circuits and Systems*, vol. 4, no. 4, pp. 319–325, 2005.