UNIVERSIDAD RICARDO PALMA

FACULTAD DE INGENIERÍA

ESCUELA PROFESIONAL DE INGENIERÍA MECATRÓNICA

"SISTEMA DE VISIÓN ARTIFICIAL HUMANOIDE PARA RECONOCIMIENTO DE FORMAS Y PATRONES DE OBJETOS, APLICANDO REDES NEURONALES Y ALGORITMOS DE APRENDIZAJE AUTOMÁTICO"

TESIS

PARA OPTAR EL TÍTULO PROFESIONAL DE INGENIERO MECATRÓNICO

PRESENTADA POR:

Bach. CASTILLO ORTIZ JONATHAN

Asesor: Dr. Ing. HUAMANÍ NAVARRETE PEDRO FREDDY

LIMA - PERÚ

2015

DEDICATORIA A Dios, a mi madre, mi padre, y a mi hermano…

AGRADECIMIENTOS

En primer lugar tengo que agradecer a los profesores, quienes me apoyaron en todo aspecto con sus conocimientos para el desarrollo de esta Tesis.

Al Ing. Velázquez Machuca, por su apoyo que me brindó, respecto a la parte de potencia para el desarrollo de mi Autómata móvil, al desarrollo de los circuitos para los actuadores, que funcionen perfectamente, por brindarme información sobre MOSFETS y aclararme muchas dudas que tenía, por dedicarme un poco de su ajustado tiempo, ante usted muy agradecido.

Dr. Ing. Elmer Córdova Zapata, siempre me prestó apoyo para que la tesis se pueda llevar a cabo, sugerencias, críticas, comentarios y poco a poco fui figurando el tema de desarrollo de mi Tesis. Además siempre estuvo al tanto de todo el plan documentario que tuve que presentar en la universidad, y por agilizar el mismo para que este proyecto de Tesis salga lo antes posible. También por la gran oportunidad que me dio para hacer un intercambio estudiantil en la Universidad Johannes Kepler, por confiar en mí y en mis capacidades, porque gracias a eso, tuve una experiencia la cual nunca olvidaré en la vida, y porque ahora tengo muchas nuevas metas que alcanzar como ingeniero, metas que jamás hubiera pensado, porque gracias a eso, quiero y anhelo ser un ingeniero competitivo a nivel mundial, deseo conocer muchos más idiomas, para así poder acceder a mucha mayor información en la red, capacitarme y prepararme más y llegar a ser un Ingeniero de calidad mundial, y gracias a esta experiencia, estoy muy motivado para lograrlo, y a todo esto, muchas gracias.

Al Mg. Ricardo J. Palomares Orihuela, padrino de Promoción, por su apoyo en lo que va de todos estos años de enseñanza, por su perseverancia y paciencia, y compartir experiencias personales la cual me sirvieron de mucho en mi desarrollo y desenvolvimiento profesional.

A mi asesor Dr. Ing. Huamaní Navarrete, quien me enseñó bastante para poder desarrollar esta tesis que, como bachiller, el desarrollo de un sistema de visión artificial estereoscópica fue muy complejo, pero gracias a su ayuda se pudo realizar.

Y para terminar, a mi hermano, y a mis padres, porque ellos fueron quienes invirtieron varios años de su vida en mí, para que yo sea un buen profesional, una buena persona, con moral y ética, quienes confiaron en mí y estuvieron conmigo en todo momento, y me dieron todas las herramientas necesarias para ahora afrontar por mí mismo a la vida. Agradezco a ellos también porque me enseñaron valores personales, como la humildad, honradez y sencillez, y a mi padre especialmente por siempre inculcarme especialmente la RESPONSABILIDAD, PUNTUALIDAD Y COMPROMISO, que bajo estos tres puntos importantes, yo debo mostrar siempre a las personas para respetarlas y ser respetado, tanto en la vida profesional, como en la vida. A mi hermano, que siempre me trajo esa chispa de alegría a la vida que tanto necesitaba para seguir adelante. A mi madre, porque siempre estuvo insistiéndome, que cumpla y mis metas y objetivos, hasta que al final, un buen día como hoy, puedo decir, familia, ya terminé, y esto, es especialmente dedicado para ustedes.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

Esta tesis, desarrolla un sistema de Visión Artificial el cual detecta objetos bajo un ambiente a campo abierto de terreno arenoso, con un fondo incierto bajo ciertas condiciones de luminosidad, brillo saturación, contraste.

Para ello se desarrolló un sistema el cual permite ver la profundidad y la distancia la cual se encuentra determinado punto de la imagen de la cámara respecto al mundo real, todo por medio de algoritmos computacionales y teoría de Visión estereoscópica, aproximándose así a un sistema de Visión artificial Humanoide al percibir la profundidad y el reconocimiento de objetos bajo el concepto que se presenta en el libro "A Humanoid Vision System for Versatile Interaction" escrita por Yasuo K., Sebastien R., Oliver S., Gorgon C. & Akihiko N. También se desarrolló un sistema el cual permite identificar el objeto objetivo, ya sea para su posterior manipulación, o posterior acción del sistema, lo cual se implementó una red neuronal Multicapa el cual permite diferenciar de entre 3 objetos, el final deseado.

El uso de la Red Neuronal Artificial guarda un papel muy importante. Las conexiones entre neuronas tienen pesos asociados que representan la influencia de una sobre la otra. Si dos neuronas no están conectadas, el correspondiente peso de enlace es cero. Esencialmente, cada una envía su información de estado multiplicado por el correspondiente peso a todas las neuronas conectadas con ella. Luego cada una, a su vez, suma los valores recibidos desde sus dendritas para actualizar sus estados respectivos.

Se emplea normalmente un conjunto de ejemplos representativos de la transformación deseada para "entrenar" el sistema, que, a su vez, se adapta para producir las salidas deseadas cuando se lo evalúa con las entradas "aprendidas".

Para la percepción de profundidad, se realizó con un tablero de ajedrez, al identificar las esquinas de cada cuadrado ubicados cada 2.54 cm a una distancia de la cámara de 79cm medidos en real, obteniendo resultados muy aproximados entre [71 – 75] cm con un error de hasta 4 cm. Esto sin embargo se puede corregir aplicando mayor uso de imágenes para lectura y calibración de cámara y luego aplicar los algoritmos de estereoscopía. Éste error, es aceptable por ejemplo en sistemas de prevención de choques en los autos, los cuales tienen un retrovisor pantalla en su panel principal, mostrándose en el screen la distancia del obstáculo detectado respecto al auto a fin de evitar malas maniobras o choques.

De esta manera en este proyecto de tesis se demuestra la identificación de objetos, y la percepción de la profundidad con el tablero de ajedrez.

Palabras Clave: Redes Neuronales, Visión Artificial, Percepción, estereoscopía, entrenar, aprendizaje

ABSTRACT

This thesis develops a computer vision system that detects objects in an open sandy field environment with an uncertain background under certain lighting conditions, brightness, saturation, contrast.

This requires a system that allows you to see the depth and distance that a particular point in the image of the camera is located respect to the real world, all by computer algorithms and theory of stereoscopic vision, thus approaching a system developed Humanoid computer vision to perceive depth and object recognition under the concept presented in the book "A Vision System for Humanoid Interaction Versatile" written by Yasuo K., Sebastien R., S. Oliver, C. & Akihiko N Gorgon . a system which identifies the target object, either for further handling, or subsequent action of the system, whereby a multilayer neural network which differentiates between 3 objects, was implemented in order to get the final object desired

The use of artificial neural network keeps a very important role. Connections between neurons are associated with synaptic weights which represent the influence on each other. If two neurons are not connected, the corresponding linked weight is zero. Essentially, each state sends its information multiplied by the appropriate weight to all neurons connected with it. Then, each one, adds the values received from their dendrites to update their respective states.

It is normally used a set of representative examples of the desired transformation to "train" the system, which, is adapted to produce the desired outputs when it evaluates the "learned" entries.

For depth perception, it was performed with a checkerboard, to identify the corners of each square located each 2.54 cm and a distance of a camera of 79cm on real, obtaining

very approximate results between [71-75] cm with an error of up to 4 cm. This however can be corrected by applying greater use of images for reading and camera calibration and then applying stereoscopy algorithms. This error is acceptable for example collision avoidance systems in cars, which have a rear screen in your dashboard, showing in the screen the distance of the detected obstacle regarding the car to avoid bad maneuvers or collisions.

Thus in this thesis project, the identification of objects is shown, and depth perception with the checkerboard

Keywords: Neural Networks, Artificial Vision, Perception, Stereoscopy, training, learning.

CAPÍTULO 1:

INTRODUCCIÓN

Actualmente se hace mucha investigación para conocer cómo es el funcionamiento de los órganos humanos, el cómo trabajan, y en particular del cómo diseñar o construir una estructura similar en la realidad.

Bajo este concepto se inspira el tema de Visión Artificial, el cómo el ojo humano percibe la forma de los objetos, cómo los reconoce y maneja esa información la cual es percibido por el ojo humano y la forma de actuar frente a ellas y enviar esta información al cerebro para que las procese y tome acciones frente a diferentes situaciones

Visión Artificial (VA), es una herramienta muy usable y necesitada alrededor de todo el mundo, ya sea para el Reconocimiento (Reconocimiento de Objetos, Identificación, Detección, y Reconocimiento de rostros) también para visión de máquina, donde la información es extraída con el propósito de mejorar un proceso de manufactura y Control de calidad de la misma. [1]

VA, está referida a Machine Visión para aplicaciones industriales, la cual una de las tareas que ejecuta es la extracción de imágenes para procesos de Control de Calidad. La Extracción de imágenes nos puede permitir hacer aplicaciones muy ponderosas como aplicaciones de sistemas de rastreo, o como en nuestra vida diaria podemos ver en Google, donde ellos tienen la búsqueda por imágenes, cargando una imagen en el buscador la cual se deseó encontrar, y ésta a base de una serie de patrones, hacienda lectura de la imagen insertada, nos deriva como resultado a una gamma de imágenes las cuales tienen las mismas o similares características a la imagen deseada a buscar. En cámaras fotográficas, las cuales actualmente, cuentan con una maravillosa herramienta

que es el reconocimiento de rostros, el cual nos permite enfocar la cámara de mejor manera en algunas partes de la imagen para la toma de una mejor foto. Como se puede ver, es un sistema completamente autónomo e inteligente.

Esta rama de VA es completamente amplia, la cual se puede desarrollar, para así poder obtener sistemas muy útiles y en un futuro, tal vez con aplicaciones mucho más complejas que las actuales. Reconocimiento de objetos y Procesamiento de Imágenes, especialmente en robótica, donde el objetivo principal es llegar a construir un ser humano y hacerlo pensar y razonar tal cual como si fuera uno.

En la siguiente Figura: 1, presentamos la relación entre la Visión por computadora y otras ramas afines.

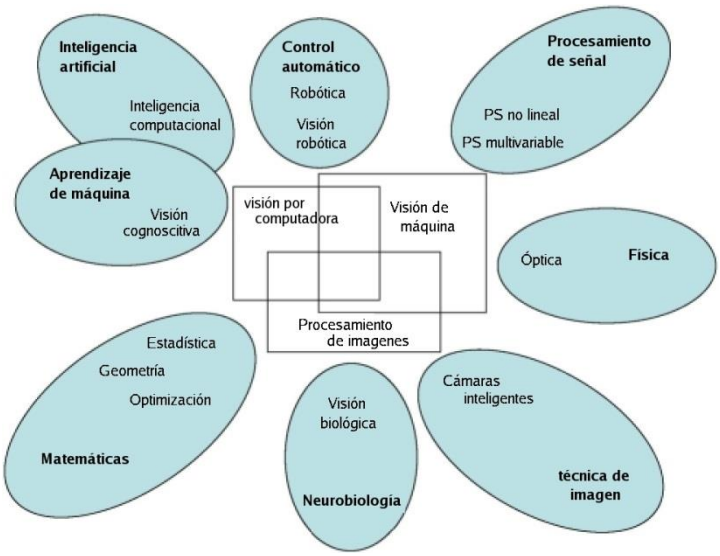


Figura: 1 Esquema de relaciones entre visión por computadora y otras áreas afines
Fuente: [5] Ramesh J., Rangachar K., Gilmore B. Schunck (1995). Machine Vision. McGraw Hill. USA

1.1 Planteamiento del Problema

1.1.1 Formulación del Problema

El uso de la automatización de procesos emulando percepciones humanas es una necesidad que se puede encontrar en diversas áreas: industrial, médica, agropecuaria, etc; sin importar el campo de acción en toda área se realizan procedimientos realizados

por personas y que podrían ser realizados por robots autónomos. El propósito de ello, en muchos casos es mejorar la calidad del proceso realizado, e inclusive hacerlo mucho más eficaz, más autónomo, disminuyendo mermas del proceso, a su mismo, volviéndola mucho más efectiva, produciendo en el tiempo mayores ganancias. El mismo de implementar un sistema que emula la percepción humana en un proceso brinda mayor robustez, por lo que permite realizar mayor diversidad de tareas, más precisas, y hasta pueda ser el caso, en menor tiempo. Esto último depende mucho del tiempo de procesamiento y del procesador usado, sin embargo no quita prestigio en cuanto a la robustez, complejidad, precisión, y mayor alcance de las tareas a poder realizar en el sistema.

Gracias a los avances de la tecnología en el campo de informática, es posible en la actualidad hacer realidad la interacción de los robots y los seres humanos con el medio ambiente, permitiendo así el desarrollo e implementaciones de soluciones que mejoran la calidad de vida de las personas, facilitando procesos en el trabajo, incrementando su productividad, evitando realizar procedimientos peligrosos, mermas, etc.

1.1.1.1 Formulación del problema general

El desarrollo de un sistema de visión Artificial humanoide para reconocimiento de formas y patrones de objetos aplicando redes Neuronales y algoritmos de aprendizaje automático

1.1.1.2 Formulación del problema específico

A. Adquisición de la imagen

¿Qué criterios de selección se debe tomar, para realizar una buena adquisición de la imagen?

B. Procesamiento de la Imagen

¿Qué tipo de filtros y en que qué dominio se deberán usar para realizar un buen procesamiento de imagen y así obtener la imagen requerida a las condiciones sometidas?

C. Redes neuronales

¿Qué tipo de Red Neuronal se debe implementar, y qué arquitectura de Red se debe considerar, y qué tipo de aprendizaje se debe usar para realizar un buen aprendizaje y posteriormente se haga una buena selección del objeto a identificar?

D. Captación y percepción de profundidad en la VA

¿Cuáles son los parámetros intrísecos de la cámara, y qué algoritmos de triangulación y estereoscopía se deben usar, implementar y programar para emular la percepción humana en una máquina?

1.2 Motivación

Este proyecto de tesis tuvo como motivación, el imitar e implementar un sistema de visión humana en máquinas, debido a esto entra el concepto de "Machine Vision". Por ejemplo, hoy en día las mejores cámaras son sólo aparatos de transmisión y grabación, son nuestras facultades de percepción visual lo que nos distingue de ellas.

Los humanos pueden reconocer los bordes entre los objetos, también repeticiones y patrones de texturas. Pueden intuir la naturaleza de la localización de fuentes de luz sin mirarlas directamente. Pueden también reconocer la naturaleza tridimensional de las cosas que ven, pueden ver como las cosas están conectadas conjuntamente, y cuán lejos están las cosas. Pueden también reconocer el movimiento o la moción del objeto que ven, pueden reconocer complejos mecanismos con un sinfín de partes móviles como componentes individuales y distinguirlos del resto de su entorno que lo rodea.

La mejor característica que distingue a los humanos de la Visión Máquina es que ellos manejan muy bien y se adaptan a la capacidad de diferenciar y reconocer todas las características mencionadas arriba, pero se adaptan mejor ante situaciones nuevas. Por ejemplo, un auto nuevo que nunca habían visto, continúa para ellos siendo un carro, porque obviamente se ve como un carro. Los humanos inmediatamente lo catalogan como un carro, instantáneamente catalogan nuevos objetos, y registran diferencias esenciales de los demás autos.

Por ello, es motivo de implementar este sistema de visión artificial Humanoide que se aproxime en gran parte a la de un humano, ya que su uso puede sernos útil para múltiples e infinidad de tareas aplicados a la vida real.

1.3 Importancia

La importancia de este proyecto ratifica en el que puede emplearse de diversas formas, ya sea en la industria, sistemas de rastreo o navegación, en procesos de Automatización como control de calidad, hasta en granjas o ranchos, para reconocer qué frutas están en buen estado o suficientemente maduras como para extraerlas y se comercialice, también para Clasificación se usa el sistema de VA como por ejemplo en las algunas compañías pesqueras en Europa, para clasificar qué clase de pez se capture en cierta zona, porque probablemente haya más de un tipo de pez la cual haya sido capturado.

Sistemas de Visión Artificial está destinados para desempeñar inspecciones visuales que requieran alta velocidad, gran crecimiento y funcionalidad las 24hs.

El objetivo de un sistema de inspección con Visión Artificial es usado para probar la conformidad de una pieza la cual tiene que cumplir ciertos requerimientos y características como dimensión, presencia de componentes ya sea en placas electrónicas, formas, etc.

En Robots Humanoides la VA cumple una gran labor la cual le permite al robot asemejarse mucho más al Humano. Manipulación de objetos e interacción del Humanoide, y toma de decisiones respecto a condiciones inciertas del mismo medio como el bajar y subir escaleras, identificación de obstáculos y localización son otras de las grandes ventajas que se llegaron a desarrollar gracias a la Visón Artificial.

1.4 Objetivos

1.4.1 Objetivos Generales

Diseñar e Implementar un Sistema de Visión Artificial Humanoide para reconocimiento de Formas y Patrones de Objetos, aplicando Redes Neuronales y algoritmos de aprendizaje automático.

1.4.2 Objetivos Específicos

1. Realizar la buena selección de la cámara, y a la misma realizar una buena lectura de las condiciones ambientales las cuales se toma la imagen adquirida con el objetivo obtener una buena imagen para posteriormente se preste a las etapas de procesamiento.
2. Desarrollar algorítmicamente rutinas de programación que permitan la realización de la segmentación, localización y reconocimiento de ciertos objetos en imágenes digitales en ambientes inciertos y reconozca objetos.
3. Implementar algoritmos inteligentes usando redes Neuronales de reconocimiento efectivo de formas y Patrones de objetos en el software Matlab.
4. Determinar e implementar algoritmos de "Percepción de profundidad" para poder percibir la distancia del objeto respecto al punto de visión (las cámaras).

1.5 Justificación

a) Costes de los materiales:

En la mayoría de las aplicaciones, evitar la producción de piezas defectuosas mediante el uso de un sistema de visión industrial tendrá un período de amortización muy corto. Para evitar que se fabriquen piezas defectuosas, el sistema de inspección automática, ya sea muestreando el 100% en la línea de producción o bien usado fuera de línea tomando muestras, debe formar parte del control estadístico de procesos del sistema productivo. Esto significa que el sistema indica cuando un parámetro de control deriva hacia el límite de tolerancia, o es simplemente demasiado errático. El sistema de visión puede tomar medidas correctivas antes de que el límite sea superado.

b) Costes de la mano de obra:

La reducción de la mano de obra es también un importante ahorro de costes, ya que muchas de las tareas realizadas por la visión industrial pueden sustituir a personas

directamente. Además, deben ser considerados los ahorros en selección de personal, prestaciones sociales y los aumentos salariales anuales.

c) Costes de calidad:

La creciente conciencia del costo de la calidad significa que el uso de la Visión Artificial puede ofrecer un estándar más fiable y consistente en la inspección de productos.

El ahorro en la optimización del uso de materiales, seguimiento de la calidad de los proveedores y garantía de calidad de los productos acabados pueden llevar a ahorros tanto tangibles como intangibles. El costo de los trabajos de reparación en garantía se puede reducir y además se mejora la confianza de los clientes consiguiendo pedidos recurrentes y una mayor cuota de mercado.

Primeramente es importante recalcar que parte del nombre de la Tesis presentada le dimos el nombre de "Sistema de Visión Artificial Humanoide" ya que éste presenta una aproximación hacia un Sistema de Visión Humanoide que realiza interacciones humanas reales en un entorno o ambiente REAL., bajo el concepto que se presenta en el libro "A Humanoid Vision System for Versatile Interaction" escrita por Yasuo K., Sebastián R., Oliver S., Gorgon C. & Akihiko N.

El hecho del uso de Visión Estereoscópica, implica y refuerza el hecho de que es un sistema de visión Humanoide ya que la estereoscopía presenta características de visión respecto al ojo humano. Debido a esto nuestra tesis es un sistema de VA Humanoide.

En el mundo de la Visión Artificial hay una gran cantidad de investigaciones, como por ejemplo en las Industria automotrices, en los sistemas de control de calidad de productos diversos, hasta en el ámbito de la robótica en robots humanoides como el

humanoide ASIMO el cual tiene integrado un sistema de Visión Artificial muy avanzado desarrollado por el equipo de la empresa HONDA.

Una de las Fuentes teóricas es el estudio de las Redes Neuronales, el cómo estas se desarrollan y como se aplican para reconocer objetos desde una cámara.

- Consisten de unidades de procesamiento que intercambian datos o información.
- Se utilizan para reconocer patrones, incluyendo imágenes, manuscritos y secuencias de tiempo (por ejemplo: tendencias financieras).
- Tienen capacidad de aprender y mejorar su funcionamiento.
- Aprendizaje adaptativo.

Las Redes Neuronales Artificiales fueron aplicadas satisfactoriamente en reconocimiento de voz, análisis de imagines y también en Control Adaptivo, para los cuales se pudieron crear Agentes software (en computadoras y juegos de video) o robots autónomos. La mayoría de los actuales Redes neuronales Artificiales empleada para Inteligencia Artificial son basados en estimaciones estadísticas, Clasificación óptima y teorías de control de sistemas.

1.6 Alcance de la tesis

Este proyecto de Tesis tiene contemplado la simulación de un sistema de visión artificial Humanoide para reconocimiento de formas y patrones de objetos, aplicando redes neuronales y algoritmos de aprendizaje automático.

Como alcance de este proyecto de tesis no se contempla hardware ya que se realizará en un proyecto futuro en un robot móvil como se muestra en la Figura: 2, Figura: 3 y Figura: 4

Figura: 2 Avance, implementación hardware de proyecto de tesis-Mecánica

Fuente: Elaboración Propia

Figura: 3 Avance, implementación hardware de proyecto de tesis Software

Fuente: Elaboración Propia

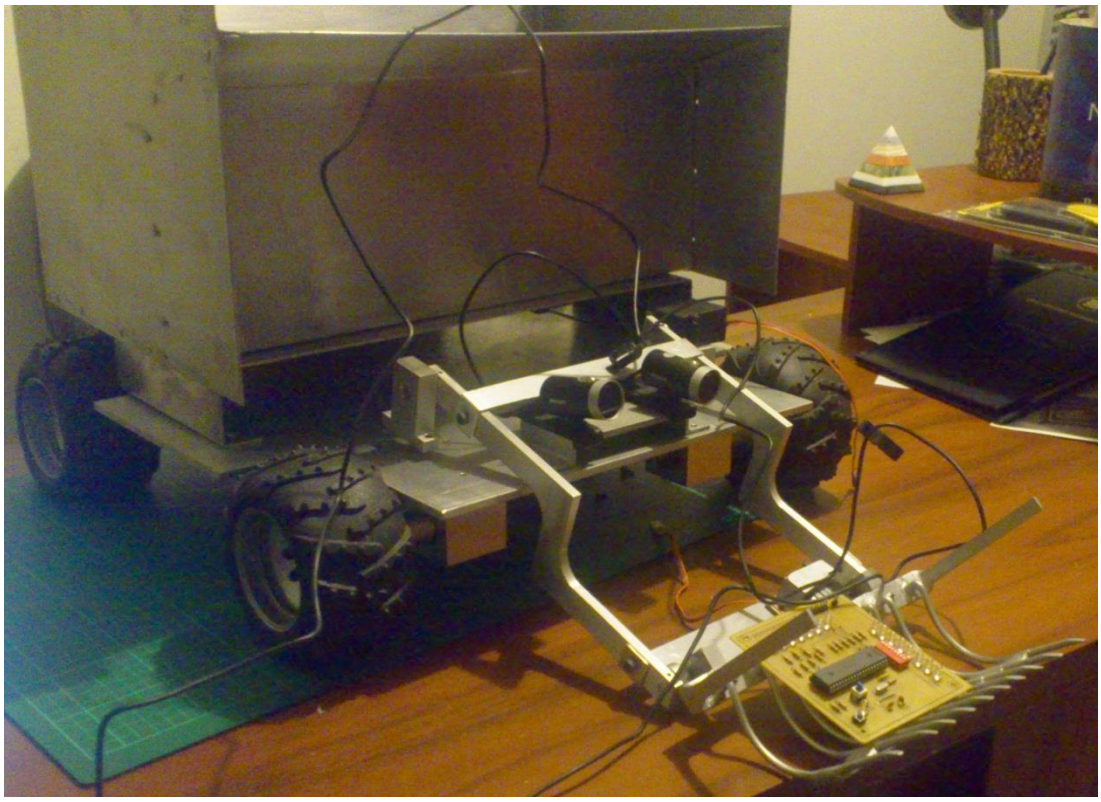


Figura: 4 Avance, implementación hardware de proyecto de tesis-cámara y Control electrónico
Fuente: Elaboración Propia

1.7 Estructura de la Tesis

En esta sección se explica lo desarrollado en cada capítulo de la tesis:

En el Capítulo 1 describe una introducción de lo que es y está compuesto un sistema de visión artificial, también se relata la formulación del problema a fin de que el proyectista desarrolle el sistema en base a esta premisa y termine con un diseño final en base al problema planteado. Se relatan las ventajas y desventajas del sistema, se describe su importancia e impacto, y luego se plantean los objetivos del proyecto.

En el Capítulo 2, en base a los objetivos planteados se plantean las hipótesis, premisas las cuales son base para el desarrollo y diseño del proyecto y definen las variables del mismo. Luego aquí se relata el fundamento teórico el cual sirve de base para entender

los conceptos, definiciones técnicas a usar a lo largo de esta tesis. En esta parte se definen 3 grandes conceptos los cuales son el sistema de Visión artificial, Las redes Neuronales y La estereoscopía. Seguidamente se definen en mayor detalle de que están conceptos estas partes para luego comenzar con el diseño del proyecto.

En el capítulo 3 se comienza a realizar el diseño (Simulación) del Sistema de Visión Artificial Humanoide para reconocimiento de formas y patrones de objetos, aplicando redes neuronales y algoritmos de aprendizaje automático. Se explica a detalle lo que se realizó y algoritmos y programación usada para el desarrollo del sistema.

En el capítulo 4 se explica el desarrollo, algoritmos y programación usada para la estereoscopía del sistema de visión Artificial. Aquí se desarrolla tanto los algoritmos de Calibración de cámara para obtener los parámetros intrínsecos del mismo como el modelamiento de la cámara para obtener la distancia de la imagen u objeto objetivo respecto a la cámara.

En el capítulo 5 se describen las pruebas realizadas en el proyecto de tesis y los resultados obtenidos del mismo a fin de demostrar las hipótesis planteadas en el capítulo 2

Seguidamente narraremos las conclusiones obtenidas una vez obtenidos los resultados. Con esto se confirma o no las hipótesis planteadas para el desarrollo del presente proyecto.

CAPÍTULO 2:

MARCO TEÓRICO

2.1 Antecedentes de la Investigación

Los robots Humanoides han cobrado a gran importancia para su desarrollo a lo largo de estos años. Identificación de formas y patrones de objetos la cual ésta tesis se enfoca, también es una de las tareas la cual la Robótica Humanoide trata de desarrollar a la perfección para poder implementar sistemas muy semejantes al humano.

2.1.1 Internacionales

Gonzalo Ch. (2004) de la Universidad de Barcelona - España, en su tesis para optar el grado de Msc en Electrónica cuyo título es "Visión Artificial: Percepción de la Profundidad" justifica que: el uso de la técnica de apareamiento estereoscópico es útil principalmente para la generación de modelos tridimensionales, manipulación de objetos con robots y sistemas de navegación autómata.

Tamara H. (2010) de la Universidad de Michigan – EE.UU, en su tesis “Sistema Automático de detección y etiquetado de caras en imágenes” justifica que: el uso de la Transformada de Fourier para reconocimiento de patrones es la herramienta más aproximada y óptima para la extracción de rasgos característicos y patrones.

Calvo C. (2000) de la Universidad Autónoma de baja California – México, en su tesis “Contribución al control de un robot móvil guiado mediante visión artificial” justifica el uso de filtros frecuenciales para la detección de objetos bajo ambientes inciertos expuesto variables de luminosidad brillo y contraste, para atenuar ruidos causados por el mismo medio del cual se requiere hacer la adquisición, y poder detectar el objeto requerido.

2.1.2 Nacionales

Sobrado M. (2003), (Univ. Nacional de Ingeniería - Lima)en su tesis para optar el grado de Mg. en Ingeniería de Control y Automatización cuyo título es "Sistema de visión artificial para reconocimiento y manipulación de objetos utilizando un brazo robot" concluye que: los algoritmos de procesamiento de imágenes(ya sean morfológicos) y algoritmos Neuronales han permitido tener una adecuada descripción e identificación de características de los objetos en tiempo real, siendo posible implementarlo en sistemas de control de calidad como en manipulación de objetos para robots autómatas, etc.

Victoria P. (2000) (Pontifica Universidad Católica del Perú – Lima) en su tesis para optar el grado de Ingeniero de Sistemas y Computación cuyo título es "Visión Artificial aplicada al monitoreo automatizado del proceso de cloración para mejorar la calidad del agua" justifica que: el uso de la visión artificial fue óptimo para diferenciar e identificar las diferentes calidades de agua, la cual aplica también en otros tipos de sistemas para diferenciar y reconocer objetos, patrones etc.

Carbajal N. (2011) (Universidad Nacional de Ingeniería - Lima) en su tesis para optar el grado de Ingeniero Electrónico cuyo título es "Clasificación automática de Vinos utilizando Redes Neuronales" justifica que: El uso de la red BackPropagation es óptimo para hacer clasificación de varios objetos ya que la actualización de los pesos de forma automática lo hace a base de gradientes locales y acumulación de energía, por lo que es más aproximado en los cálculos, y sus resultados son óptimos y flexibles en varias tareas de reconocimiento y aprendizaje artificial.

2.2 Óptica de la Investigación

2.2.1 Hipótesis General

Es posible el diseño e implementación de un sistema de visión artificial humanoide para reconocimiento de formas y patrones de objetos, aplicando redes neuronales y algoritmos de aprendizaje automático mediante el uso apropiado de hardware de

adquisición de imágenes, software y algoritmos matemáticos para el sistema de visión artificial.

2.2.2 Hipótesis Específicas

1. Es posible realizar una buena adquisición de la imagen en ambientes al aire libre e inciertos, con una cámara de alta resolución HD para PC, y hacer la lectura de la imagen digital en el ordenador
2. Es posible hacer que el Sistema de VA funcione en ambientes inciertos, y reconozca objetos mediante el uso de filtros y la transformada de Fourier.
3. Es posible la detección de formas y patrones de objetos de forma exitosa, mediante el uso de la red Neuronal BackPropagation, primero tomando un set de entrenamiento de la Neurona, para de esta manera entrenar y actualizar los pesos sinápticos de la red, para posteriormente una vez realizado el aprendizaje, logre diferenciar eficientemente la clase objetivo a identificar de manera robusta.
4. Es posible la detección de profundidad del objeto mediante el uso de 2 cámaras en el Autómata, ya que de esta manera nos permite realizar una visión 3D.

2.3 Selección de Variables

Para poder realizar las **demostraciones** del sistema de visión artificial humanoide se precisan las diversas variables:

A. Procesamiento de imágenes

E_y : Imagen RGB

$W_{i,j}$: valor de pixeles en la máscara escogida (la suma de estos pesos debe ser igual a 1)

$F(u,v)$: Transformada de Fourier de la imagen en el dominio de la frecuencia (Representación de $f(x,y)$ en el dominio de la frecuencia)

$H(u,v)$: Filtro frecuencial escogido para el procesamiento

$G(u,v)$: Resultado de la convolución de el filtro con la imagen en el dominio de la frecuencia.

h : filtro en el dominio del espacio

$e^{j\varphi(u,v)}$: Representación polar de la transformada de Fourier de la imagen

$f(x,y)$: Imagen original

(u,v) : variables continuas que expresan frecuencia

$g(x)$: Función gaussiana 1D (filtro)

$g(x,y)$: Función gaussiana 2D

$g(i,j)$: filtro gaussiano (máscara en el espacio)

B. Redes Neuronales

$\hat{X}_j$: es vector de características, donde n es el número de características que definen a mi patrón de entrada.

$\hat{P}_m$: es vector del patrón a identificar.

$\hat{Y}_q^{(i)}$: es vector de Salida donde q es el número de Salidas e i es el número de Iteración que hace la Red Neuronal

$\hat{Y}_{dq}^{(i)}$: es vector de Salida deseado donde q es el número de Salidas e " i " es el número de Iteración que hace la Red Neuronal.

$\hat{W}_{ij}$: es vector de pesos sinápticos de mi primera capa.

$\hat{C}_{nl}$: es vector de pesos sinápticos de mi capa oculta.

e_q : es el error en la salida “q”

C. Estereoscopía

x y z coordenadas iniciales (alg. Estereoscopía)

x ’, y ’, z ’ coordenadas finales (alg. Estereoscopía)

$W_h \rightarrow$ Pto. en el Mundo en coordenadas homogéneas

Cartesianas: (X, Y, Z)

Homogéneas: $(k.X, k.Y, k.Z, k)$

$G \rightarrow$ Traslación

$R_{-\theta}^{Z} \rightarrow$ Rotación alrededor del eje Z de forma anti horaria un ángulo θ

$R_{-\phi}^{X} \rightarrow$ Rotación alrededor del eje Z de forma anti horaria un ángulo ϕ

$C \rightarrow$ Traslación

$P \rightarrow$ Transformación de Perspectiva (Toma la imagen / foto)

$C_h \rightarrow$ Como resultado se obtiene coordenadas en el sistema de coordenadas de la Imagen

2.4 Bases Teóricas

2.4.1 Sistema de visión artificial

Se puede definir la “Visión Artificial” como un campo de la “Inteligencia

Artificial” que, mediante la utilización de las técnicas adecuadas, permite la obtención, procesamiento y análisis de cualquier tipo de información especial obtenida a través de imágenes digitales.

La visión artificial la componen un conjunto de procesos destinados a realizar el análisis de imágenes. Estos procesos son: captación de imágenes, memorización de la información, procesado e interpretación de los resultados. [1]

2.4.1.1 Adquisición de la imagen

En esta sección se presenta los métodos que se deben seguir para una correcta adquisición de imágenes.

Cabe resaltar que este proceso es muy importante para reducir ruidos y otros factores que posiblemente nos afecten en los siguientes pasos de procesamiento de la imagen. [1]

En este primer paso, se trata de conseguir que la imagen sea lo más adecuada posible para que se pueda continuar con las siguientes etapas.

Una correcta adquisición de la imagen supone un paso muy importante para el proceso de reconocimiento tenga éxito. Dentro de esta etapa existen múltiples factores que atañen directamente al proceso de captura de la imagen, formados fundamentalmente por: el sistema hardware de visión artificial, ya sea cámara, óptica, tarjeta de adquisición o PC, y el entorno de posicionamiento de los elementos, como el fondo, la iluminación, posición correcta de cámara entre otros.[1]

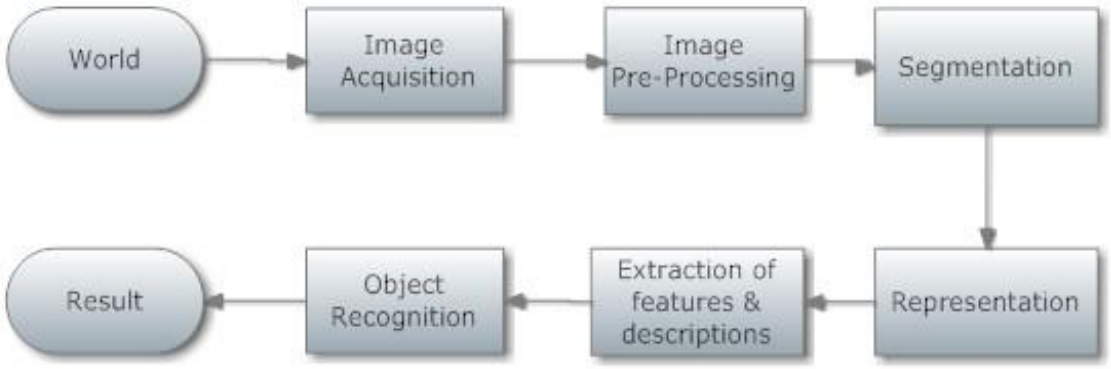


Figura: 5 Pasos de un sistema de Visión Artificial

Fuente: Elaboración Propia

De acuerdo a la Figura: 5 la siguiente etapa después del pre-procesamiento de la imagen es la de Segmentación, la cual de manera sencilla, se puede definir como la partición de

una imagen en un conjunto de regiones no solapadas y homogéneas con respecto a algún criterio, cuya unión cubre la imagen entera. La segmentación es una etapa que conlleva, así como por la importancia de sus resultados. La imagen de salida después de este proceso de segmentación es una imagen etiquetada o simbólica, donde se ha pasado de una imagen digital bruta, cuyos componentes son píxeles, a una imagen más simplificada en lo que se espera que los objetos aparezcan nítidamente distinguidos.[1]

Posterior a la etapa de segmentación las etapas que siguen son las de representación y descripción. La representación y descripción de patrones u objetos que han sido previamente segmentados son los primeros pasos que se realizan en la mayoría se los sistemas de análisis automático de imágenes. Existen diferentes técnicas ara la representación t la descripción. El objetivo de las técnicas de representación es llevar los datos de cada uno de los objetos o regiones segmentados a formas en que la computadora pueda trabajar con ellos de manera más apropiada; una lista ligada circular que represente la información del contorno de la región de un objeto es más fácil de trabajar que un conjunto de pares de coordenadas de los pixeles del contorno de la misma región.[**2**] El objetivo de las técnicas de descripción consiste en capturar las diferencias esenciales entre objetos pertenecientes a clases diferentes; por supuesto se buscaría que estos mismos rasgos se mantuvieran lo más invariantes ante cambios como escalamientos traslaciones t rotaciones.[2]

2.4.1.1.1 Cámara

La cámara es un elemento de adquisición de imágenes la cual de por sí transforma las señales luminosas en señales analógicas. Ésta se divide en dos partes, el sensor, y la óptica, la cual se encarga de proyectar los elementos adecuados de la escena, ajustando una distancia focal adecuada.[1]

Los sensores de visión usados son los basados en matrices de dispositivos acoplados por carga CCD. Estos transductores realizan una digitalización espacial completa en dos dimensiones (líneas y columnas), pues descomponen la imagen en una matriz de puntos.[1]

La selección de la cámara mucho depende de múltiples factores como mencionamos anteriormente, ahora mencionaremos algunos de estos factores:

Cámara de color o monocromática: Debemos decidir si usamos una cámara de color o de escala grises. En la mayoría de aplicaciones industriales se usan cámaras monocromáticas, obteniendo ventajas tanto económicas como computacionales.[1]

Cámara digital o análoga: La principal ventaja que tienen éstas cámaras digitales frente a las análogas es la velocidad y la calidad de imagen que éstas presentan. También las digitales transmiten información directamente en digital ahorrando de esta manera tiempo computacional de procesamiento, ya sea al computador o a una tarjeta de adquisición, y también supone mejor calidad frente a ruidos que puedan aparecer en los elementos de transmisión. [2]

Velocidad de obturador: Para poder obtener imágenes en movimiento, es necesario controlar el tiempo de exposición. He aquí también el concepto de Trigger/Triggering**.**[2]

Resolución: Cuanto mayor resolución tiene una cámara mejor se distinguen los detalles más pequeños de los objetos. Sin embargo también a su resolución también va a la mano el precio de la cámara, y es un costo a pagar si se quieren obtener buenos o mejores resultados en un sistema de VA.[2]

Existen otras cámaras que son las Cámaras progresivas, pero no vamos a ampliar más ese concepto y nos limitaremos a los conceptos anteriormente mencionados.

Se puede concluir que para la elección de la cámara hay que determinar qué proceso se va a realizar y según éste decidir si es necesaria una cámara color o monocromática.

2.4.1.1.2 Óptica

La óptica en un sistema de visión artificial es muy importante, ya que de esto dependerá especialmente de las condiciones ambientales y de la distancia de medición.

Las ópticas se utilizan para transmitir la luz al sensor de la cámara de una forma controlada y de esta forma obtener una imagen enfocada de uno o varios objetos. Para saber exactamente que óptica debe utilizarse para la aplicación que se desea resolver se debe tener en cuenta una serie de parámetros.[2]

Dependiendo de la luminosidad ambiental, puede ser conveniente la utilización de filtros que la eliminen y que permitan capturar convenientemente los objetos a explorar.

En la selección de la óptica, sobre todo cuando se van a determinar características geométricas de los objetos, es necesario evitar todas aquellas lentes que deformen la imagen. [2]

A veces es necesario el uso de filtros espaciales para graduar la luminosidad de la cámara.

2.1.1.2 Pre-procesamiento de la imagen

En algunas aplicaciones no es necesario este proceso, Sin embargo en muchas, esta etapa es fundamental y primordial, y básicamente consiste en la atenuación del ruido presente en la imagen, el mejoramiento del contraste, así como filtrados para eliminación de artefactos, entre otros.

El proceso de pre-procesado de la imagen se realizó mediante el esquema que mostrado en la Figura: 6

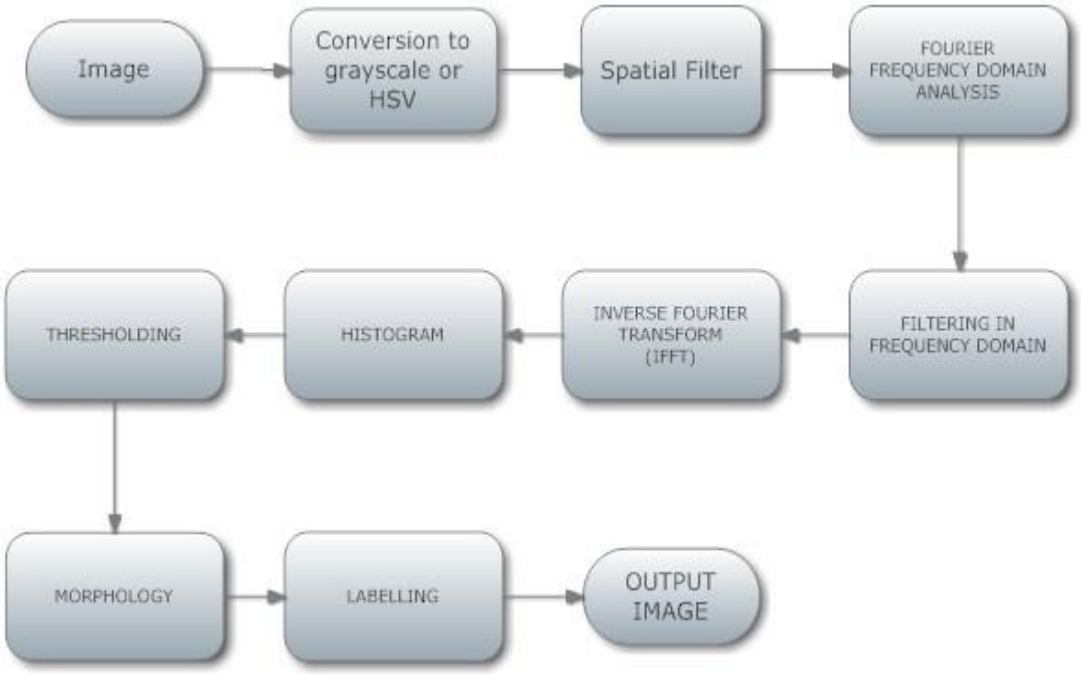


Figura: 6 Etapa de Pre-Procesamiento de una imagen
Fuente: Elaboración Propia

2.1.1.2.1 Escala Grises

Escala de grises (o graylevel) de una imagen no es más que una en la que los únicos colores son tonos de gris. La razón para diferenciar este tipo de imágenes de cualquier otra especie de imagen de color es que menos información necesita ser proveída para cada píxel. De hecho un color gris es uno en el que todos los componentes rojo, verde y azul (RGB) tienen la misma intensidad en el espacio RGB, y por lo que sólo es necesario especificar un valor de intensidad para cada pixel individual, en oposición a las tres intensidades, la cual necesitábamos especificar para una imagen Full color RGB. [3]

Con frecuencia, la intensidad en escala de grises se almacena como un entero de 8 bits que da 256 posibles diferentes tonos de gris desde el negro al blanco. [1]

La escala de grises en las imágenes son muy comunes (tal como muestra la Figura: 7), porque gran parte de las pantallas de hoy y hardware de captura de imagen sólo pueden soportar imágenes de 8 bits. Además, las imágenes en escala de grises son totalmente suficientes para muchas tareas por lo que no hay necesidad de usar imágenes a color que son más difíciles de procesarlas y mucho más complicadas.

Figura: 7 Rango de escala de grises [255 0]

Fuente: Tttps://es.enciclopedia.org/msf/grayscalefeature1&%C7color

Ahora para la conversión a Escala grises de una imagen a color RGB, presentaremos las Curvas de luminancia, con sus factores de ponderación de cada componente de color RED, GREEN, BLUE, las cuales indican su sensibilidad respecto al ojo humano. Ver Figura: 8

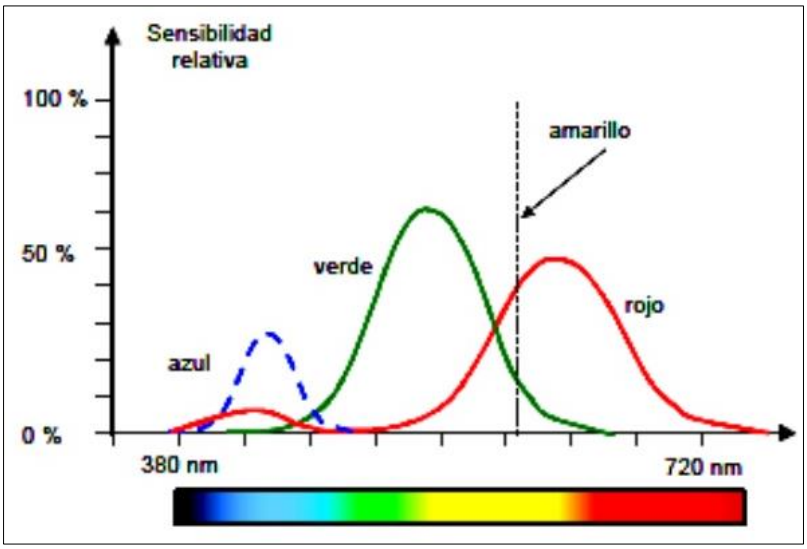


Figura: 8 Curva Universal normalizada de Visión

Fuente: https://es.enciclopedia.org/msf/Percepci%C3%B3n_del_color

La expresión matemática está dada por la ecuación

$$E_y = R\times(0.3)+G\times(0.59)+B\times(0.11) \qquad (2.\ 1)$$

Esta expresión permite realizar la conversión, aplicándolo para cada píxel de la imagen de color, resultando de esta manera una matriz de 8 bits por píxel dando la información de luminancia. Ver Figura: 8

2.1.1.2.2 Hue-Saturation-lightness & Hue-Saturation-Value (HSL & HSV)

HSL viene por sus siglas en inglés Tono, Saturación y Luminosidad, y HSV significa Tono, Saturación y Valor, ésta también es conocida como HSB (B por brightness o brillantez). Son de las representaciones en coordenadas cilíndricas más comunes de los puntos de una imagen o modelo RGB. Esa representación reorganiza la geometría de RGB en un intento de ser más intuitivo y perceptivamente relevante que la representación cartesiana (cubo). Ver Figura: 9

Estas representaciones, son usadas hoy en día para computación gráfica, varios software de edición de imágenes, análisis de imágenes y visión por computadora.

Como son representaciones cilíndricas, el ángulo alrededor del eje vertical corresponde a Tono, la distancia al eje corresponde a Saturación, y la distancia a lo largo del eje corresponde a "Luminosidad", "Valor o Brillantez". Notamos que mientras "Tono", en HSL y HSV refiere al mismo atributo, sus definiciones en "Saturación", difieren en mucho.[4]

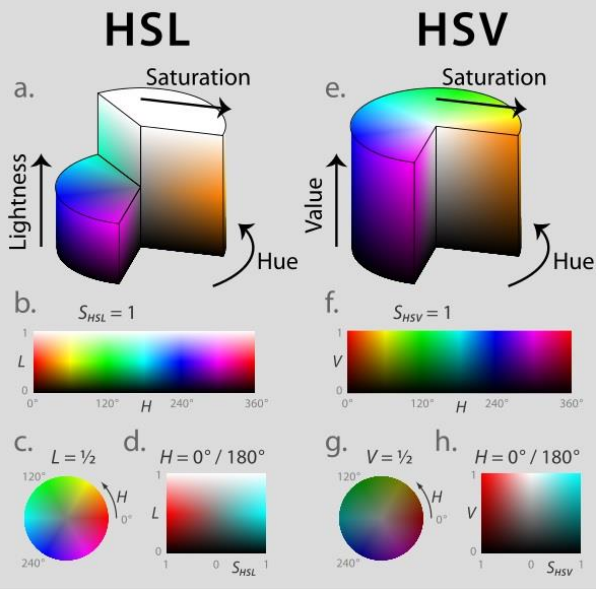


Figura: 9 Representación cilíndrica de los modelos HSL Y HSV, con sus representaciones gráficas 2D

Fuente: https://en.enciclopedia.org/msf/HSL_and_HSV

2.1.1.3 Filtrado en el Dominio del Espacio

Antes de comenzar un sistema de Visión Artificial, debemos usar técnicas adecuadas para disminuir ruidos o disturbios el que tenga la imagen, para corregir errores, y mejorar el tiempo computacional de procesado, por esto su importancia el aplicar esta técnica.

Por lo general este tipo de filtro consiste en recorrer toda la imagen pixel a pixel, respecto a un pixel central, y cada uno de estos realiza alguna operación matemática dependiendo del tipo de filtro que se use, con un número concreto de píxeles vecinos. Estos píxeles vecinos, en conjunto recibe varios nombres, entre los más conocidos son: ventana, kernel, máscara y pueden ser de 3x3, 5x5, 6x6 e incluso pueden llegar a ser kernels no cuadrados. Ver Figura: 10

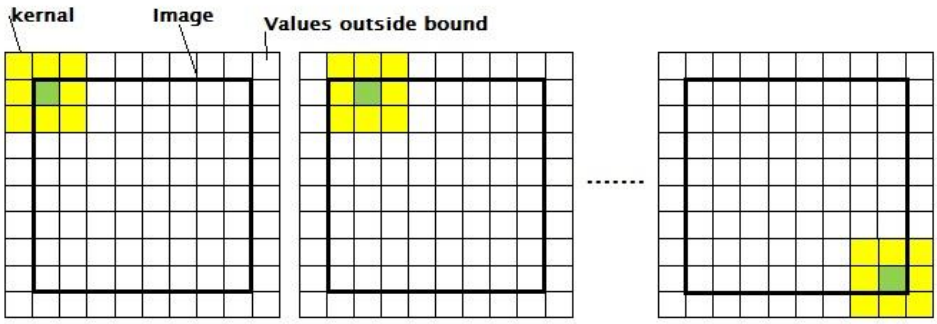


Figura: 10 Representación de máscara para filtrado de imágenes, siendo el píxel verde el píxel central.
Fuente: [5] Ramesh J., Rangachar K., Gilmore B. Schunck (1995). Machine Vision. McGraw Hill. USA

Ahora, hay varios tipos de filtros espaciales, mencionaremos algunos.

Como es en el dominio del espacio, existen técnicas las cuales nosotros podemos mejorar nuestra imagen trabajando directamente con los píxeles de la imagen. Un ejemplo de esto son las llamadas Operaciones Espaciales o de Vecindad (trabajan directamente con los niveles de gris y con las relaciones posicionales de los píxeles de la imagen). [5]

Ahora, entre estas operaciones Espaciales que mejoran la imagen pueden ser de dos tipos:

- Operaciones Globales: Son quienes operan sobre toda la imagen a la vez y solo tienen en cuenta la Intensidad en cada punto.
  - Negativo
  - Aumento de contraste
  - Comprensión del rango dinámico
  - Fraccionamiento del nivel de gris
  - Procesamiento de histogramas (Ecualización, especificación)
- Operaciones Locales: Actúan sobre una zona de la imagen (vecindad). Tienen en cuenta la posición de los pixeles, tanto absoluta como relativa. Hay dos tipos de Operaciones Locales:
  - Operaciones Algebraicas

    Sustracción de imágenes

    Promediado de imágenes
  - Filtrado Espacial

    Suavizantes:

    Filtros Pasa bajo

    Mediana

    Realzantes:

    Paso alto

    High-boot

    Diferenciales:

    Gradiente

2.1.1.3.1 Filtro pasa bajo

Este tipo de filtros como su nombre lo dice, deja pasar las bajas frecuencias, más no las altas. En imágenes las ALTAS FRECUENCIAS se asocian a "cambios bruscos de intensidad en pequeños intervalos espaciales (pixeles)", es decir BORDES. Las BAJAS FRECUENCIAS se asocian a "cambios lentos de intensidad en pequeños intervalos espaciales"[5]

En Filtros Pasa Bajo, al no dejar pasar las Altas frecuencias, éstas se atenuarán (los bordes) y su resultado es un desfiguro de los bordes (desenfoque) o más conocido como efecto blurring.

Un filtro pasa bajo muy utilizado para procesamiento de imágenes es el que presentaremos a continuación en la Figura: 11 Filtros pasa bajo de varios tamaños

$$1/9 \times \begin{bmatrix} 111 \\ 111 \\ 111 \end{bmatrix} \qquad 1/25 \times \begin{bmatrix} 11111 \\ 11111 \\ 11111 \\ 11111 \\ 11111 \end{bmatrix}$$

(a) (b)

Figura: 11 Filtros pasa bajo de varios tamaños
Fuente: Elaboración propia

Las máscaras de suavizado son básicamente promedios de los pixeles en el vecindario. La suma de los coeficientes debe ser 1 como se muestra en la ecuación $\sum_{i,j} w_{i.j} = 1$ (2. 2), para evitar ganancias indeseadas.

$$\sum_{i,j} w_{i.j} = 1 \qquad (2.\ 2)$$

Donde $w_{i,j}$ son los pesos o el valor de los pixeles dentro de la máscara.

2.1.1.3.2 Filtro pasa alto

Este tipo de filtro, deja pasar las altas frecuencias, aislando estos componentes en una imagen. Este filtro tiene el efecto contrario al filtro pasabajos.

Este filtro acentúa componentes en alta frecuencia espacial obteniendo como resultado los bordes en una imagen por ejemplo más resaltados. [6]

Una de las motivaciones de aplicar una operación local es la posibilidad de poder resaltar o atenuar por zonas las frecuencias espaciales de una imagen. En muchos problemas prácticos al capar una imagen se produce una atenuación de sus altas frecuencias y por lo tanto se observa cierta borrosidad de sus contornos, lo cual da una sensación desagradable al observado.

Es conocido que el HSV se adapta mejor a una imagen rica en altas frecuencias (muchos detalles), aun cuando la sensación de estar un poco ruidosa, que a una imagen donde predominen las bajas frecuencias espaciales (borrosidad). En este caso se hace resaltar las altas frecuencias.

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{bmatrix} \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} 1 & -2 & 1 \\ -2 & 5 & -2 \\ 1 & -2 & 1 \end{bmatrix}$$

Figura: 12 Máscaras para realizar un filtrado pasa alto
Fuente: [3] Sahoo P.K.(1988) A survey of Thresholding Techniques. Computer Vision Graphics and Image Processing. Ed. Morgan. England

Uno de los filtros más usados dentro de las operaciones locales para resaltar los detalles en una imagen y producir así un mejoramiento en su contraste consiste en aplicar, por ejemplo, una de las máscaras mostradas en la Figura: 12.

2.1.1.2 Filtrado en el Dominio de la Frecuencia

Procesan una imagen trabajando en el dominio de la frecuencia haciendo uso de la transformada de Fourier de la imagen. Es decir, si se desea filtrar una imagen, se tiene que pasar esta imagen del dominio del Espacio, al dominio frecuencial, y esto solo se logra con la transformada de Fourier.

Para ello, ésta se modifica siguiendo el Teorema de la Convolución correspondiente:

1. Se aplica la Transformada de Fourier,
2. Se multiplica posteriormente por la función del filtro que ha sido escogido,
3. Para concluir re-transformándola al dominio espacial empleando la Transformada Inversa de Fourier.

El Teorema de la Convolución viene dado por la ecuación

$$G(u,v) = F(u,v) * H(u,v) \qquad (2.\ 3)$$

$$G(u,v) = F(u,v) * H(u,v) \qquad (2.\ 3)$$

Donde

$F(u,v)$: es la transformada de Fourier de la imagen

$H(u,v)$: es el filtro frecuencial

Las técnicas basadas en filtros frecuenciales, como ya mencionamos antes se basan en la transformada de Discreta de Fourier (DFT) y la Transformada Discreta de Fourier Inversa (IDFT). Ambas se han desarrollado como un caso particular de enorme interés práctico de la Transformada de Fourier clásica o continua a partir de la aparición de la tecnología informática, que no trabaja con señales continuas.

A continuación en la Figura: 13, muestra un ejemplo de donde se aplicó la Transformada de Fourier en imágenes

(a) (b)

Figura: 13 (a) imagen Original (b) Espectro de Fourier de la Imagen Original
Fuente: [6] Grandon T.(2011) Artificial Vision in the Nao Humanoid Robot. Tesis (Master in Artificial Intelligence). Catalan – Rovira i Virgii University, Department of Computer Science and Mathematics. Spain

A continuación se presenta las etapas de procesamiento de la imagen en el dominio de la frecuencia

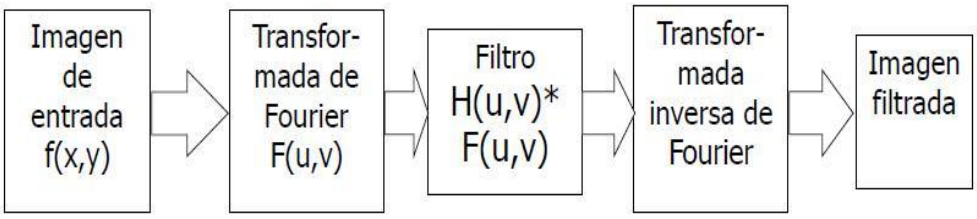


Figura: 14 Etapas de procesamiento de la imagen en el dominio de la frecuencia
Fuente: Fuente: https://es.enciclopedia.org/thz/frecuencydomain1&%C7color

Como vimos anteriormente en el dominio espacial, si se quería hacer el filtrado de una imagen, se tenía que escoger un filtro, con un kernel adecuado, y luego éste, se le aplicaba una Convolución con la imagen, para obtener como resultado la imagen filtrada. A diferencia del dominio del espacio, en la frecuencia no se hace la Convolución, si no, su equivalente que es la "Multiplicación". [1] Ver Figura: 14

Para hacer el filtrado en el dominio de la frecuencia, como ya se explicó, la imagen se tiene que pasar del dominio espacial al dominio frecuencial, pero también el Filtro espacial que se vá a usar, se tiene que pasar al dominio frecuencial.

Por teoría la mayoría de filtros espaciales se pueden representar en el dominio de la frecuencia.

Para hacer esto es necesario que el filtro seleccionado tenga las mismas dimensiones que la imagen, ya que es un requisito para poder hacer la multiplicación como se muestra en la Figura: 15, para el proceso de filtrado en la frecuencia.

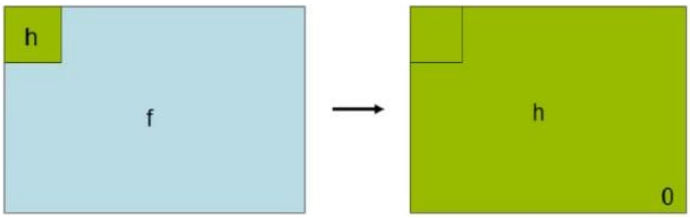
Véase más claro en un ejemplo:

Sea "h" el filtro espacial de suavizado (pasa-bajo) seleccionado en la ecuación (2. 4):

$$h = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{2. 4}$$

Y tenemos una imagen de M X N dimensiones (filas-columnas).

Entonces se hace el siguiente algoritmo de rellenado mostrado en la Figura: 15 para obtener así un filtro de las mismas dimensiones de la imagen. Este algoritmo también es conocido como "Padding-Algorithm" el cual rellena vacíos (en este caso con ceros)



(a) (b)

Figura: 15 Algoritmo Padding, 2.11(a) Filtro "h" de 3x3 espacial vs Imagen de M X N; 12 (b) filtro "h" espacial de MxN dimensiones.
Fuente: [7] Carnajal M. (2010). Robótica y Visión Artificial. [Diapositivas] Universidad de Alicante. Spain

En la Figura: 15 se observa como el filtro “h” pasa a ser de 3x3 a MxN done aloja en sus primeras 3 filas y 3 columnas a los elementos de la ecuación (2. 4), y el resto lo rellena con ceros hasta MxN.

Una vez hecho el algoritmo, para pasarlo al dominio de la frecuencia, se hace lo mismo que mencionamos anteriormente con la imagen, hallamos su Transformada de Fourier (TF), y una vez obtenido la TF de la imagen, y la TF del filtro, se pasa a multiplicar ambas expresiones (pixel por pixel), obteniendo luego una imagen filtrada después de aplicarle la TF Inversa, siguiendo los pasos de la Figura: 15. Recordemos que la imagen con el filtro se multiplican, mas no se convolucionan.[7]

2.1.1.2.1 Transformada de Fourier

El uso de la transformada de Fourier permite pasar una señal continua en el tiempo, o en el espacio en el caso de imágenes, al dominio de la frecuencia.

Cualquier señal periódica puede representarse por una suma de señales basadas en senos y cosenos con diferente amplitud, frecuencia y fase.

Su expresión matemática de la transformada de Fourier 1D está dada por la ecuación

$$F(u) = \int_{-\infty}^{\infty} f(x) e^{-j2\pi ux} dx \qquad (2.\ 5)$$

$$F(u) = \int_{-\infty}^{\infty} f(x) e^{-j2\pi ux} dx \qquad (2.\ 5)$$

En imágenes, la transformada Fourier está dada por la ecuación.

$$F(u,v) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y) e^{-j2\pi(ux+vy)} dxdy \qquad (2.\ 6)$$

$$F(u,v) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y) e^{-j2\pi(ux+vy)} dxdy \qquad (2.\ 6)$$

Donde “u, v” son frecuencias espaciales

En Fourier vemos el término exponencial con un valor complejo. Esto da cuenta que lo podemos representar en su forma polar que ya daremos cuenta a detalle más adelante en parte Real y parte imaginaria.[8]

Se tiene que tener en cuenta que:

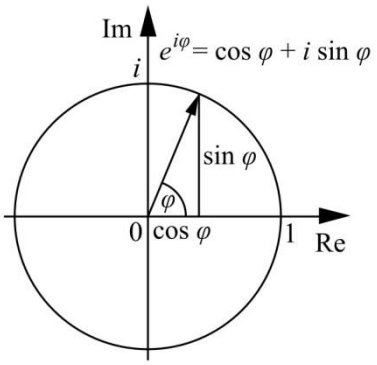


Figura: 16 Representación de un número complejo $\mathbf{e^{i\varphi}}$ en el plano complejo usando la fórmula de Euler (coordenadas polares)
Fuente: [8] Dr. Wolf M.(2002) Quantitative Image Analysis Methods and Limitations. Biomedicalproducts. Ed. Springer. USA

La coordenada angular φ es expresada en radianes a través de esta sección.

Por Euler tenemos la ecuación. (2.7)

$$e^{j\varphi} = cos(\varphi) + jsin(\varphi) \qquad (2.7)$$

Entonces se obtiene la ecuación (2.8)

$$Re = cos(\varphi); Im = sin(\varphi) \qquad (2.8)$$

Lo mismo se tiene en imágenes.

Sea F(u,v) la transformada de Fourier de una imagen, y como ya vimos anteriormente, la transformada de Fourier nos da una respuesta compleja, con parte real y parte imaginaria. Entonces se puede expresar en su forma polar de la siguiente manera:

$$F(u,v) = |F(u,v)| = |e^{j\varphi(u,v)} \qquad (2.\ 9)$$

Donde la magnitud |F(u,v)| recibe el nombre de **"Espectro de Fourier o de frecuencia"** y está dada por:

$$|F(u,v)| = \sqrt{R^2(u,v) + I^2(u,v)} \qquad (2.\ 10)$$

Y su ángulo de fase está dado por la ecuación (2.10)

$$\varphi(u,v) = arctan(\frac{I(u,v)}{R(u,v)}) \qquad (2.\ 11)$$

Como ya vimos la representación del Espectro de Fourier o de frecuencia denotada por $|F(u,v)|$ lo cual es la magnitud de la transformada de Fourier, Ahora vamos a ver el valor de su espectro pero de potencia.

El **"Espectro de potencia"** se define como:

$$\begin{aligned} P(u,v) &= |F(u,v)|^2 \\ &= R^2(u,v) + I^2(u,v) \end{aligned} \qquad (2.\ 12)$$

A continuación se muestra un ejemplo del uso de la transformada de Fourier:

Se tiene un rectángulo de dimensiones X, Y en el espacio centrado en el origen. Ahora calcularemos su Transformada de Fourier Ver ecuación. 2.13 y 2.14.

$$F(u.v) = \int_{X/2}^{-X/2} e^{-j2\pi ux} \int_{Y/2}^{-Y2} e^{-j2\pi vy} \qquad (2.\ 13)$$

$$= \left[\frac{e^{-j2\pi ux}}{-j2\pi u} \right]_{-X/2}^{X/2} \left[\frac{e^{-j2\pi ux}}{-j2\pi u} \right]_{-Y/2}^{Y/2}$$

$$= \frac{1}{-j2\pi u}[e^{-juX} - e^{-juX}]\frac{1}{-j2\pi v}[e^{-jvY} - e^{-jvY}]$$

$$XY\left[\frac{sin(\pi Xu)}{\pi Xu}\right]\left[\frac{sin(\pi Yv)}{\pi Yv}\right] \qquad (2.\ 14)$$

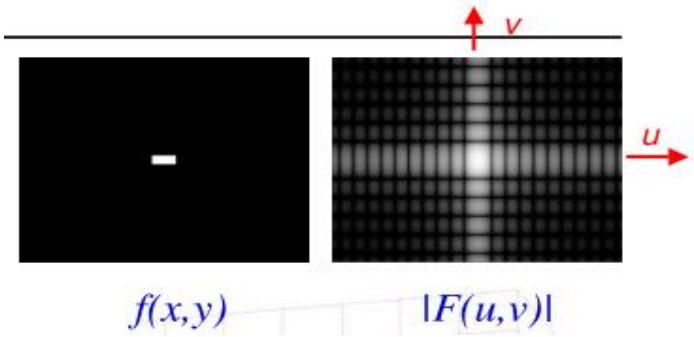


Figura: 17 En la mano izquierda tenemos rectángulo situado en su origen, y a su derecha su transformada de Fourier

Fuente: [9] Moore D. (2003) A real-world system for human motion detection and tracking. California Institute of Technology. USA

Siendo $f(x,y)$: La imagen original

(u,v): variables continuas que expresan frecuencia

$F(u,v)$: Representación de $f(x,y)$ en el dominio de la frecuencia

2.1.1.2.2 Transformada Discreta de Fourier (DFT)

Mediante esta transformada se obtiene el espectro frecuencial de la imagen en cada eje, obteniendo unas frecuencias determinadas, el modulo y argumento de cada componente frecuencial de la imagen. Sea una imagen f(x,y) con dimensiones MxN, se define su Transformada Discreta de Fourier de la siguiente manera expresada en la ecuación (2.15)

$$F(u,v) = \sum_{x=0}^{X-1}\sum_{y=0}^{Y-1} f(x,y)e^{-j2\pi(\frac{u*x}{X})}e^{-j2\pi(\frac{v*y}{Y})} \qquad (2.\ 15)$$

$$Donde:$$

$$u = 0, 1, ..., X - 1$$

$$v = 0, 1, ..., Y - 1$$

X, Y son las dimensiones de la imagen.

(x, y) son las posiciones de los pixeles de la imagen en el dominio espacial desde [0 X-1] x [0 Y-1], ejemplo una imagen de 256x256 entonces x=0,1,...,255, y=0,1,...,255

(u, v) son las variables discretas que expresan frecuencia.

Ahora, también podemos expresar la Transformada de Fourier por sus muestras w1, w2, en F(w1, w2) en la siguiente ecuación (2.16):

$$Sea: \quad w1 = 2\pi u/X \quad w2 = 2\pi v/Y \quad F(w1, w2) = \sum_{x=0}^{X-1} \sum_{y=0}^{Y-1} f(x, y) e^{-j(w1.x)} e^{-j(w2.y)} \tag{2.16}$$

2.1.1.2.3 Transformada Inversa Discreta de Fourier (IDFT)

Dada la transformada F(u,v), es posible obtener f(x,y) utilizando la transformada inversa de Fourier. Esto se usa para volver a obtener la imagen del Dominio de la Frecuencia al Dominio Espacial.

Esta ecuación por la ecuación (2.17)

$$f(x, y) = \frac{1}{XY} \sum_{u=0}^{X-1} \sum_{v=0}^{Y-1} f(x, y) e^{j2\pi(\frac{u.x}{X})} e^{j2\pi(\frac{v.y}{Y})} \tag{2.17}$$

$$Siendo:$$

$$x = 0, 1, ...X - 1$$

$$y = 0, 1, ...Y - 1$$

La Transformada Fourier 2D es en general compleja, y esta tiene tanto parte Real, como Imaginaria, y ésta puede representarse en su forma polar en la ecuación (2.18):

$$F(u, v) = \left|F(u, v)\right| e^{j\Theta(u,v)} \tag{2.18}$$

Donde la magnitud |F (u,v)| recibe el nombre de ESPECTRO DE FOURIER o de frecuencia como se muestra en la ecuación (2.19)

$$\left|F(u,v)\right| = [R^2(u,v) + I^2(u,v)]^{1/2} \quad (2.19)$$

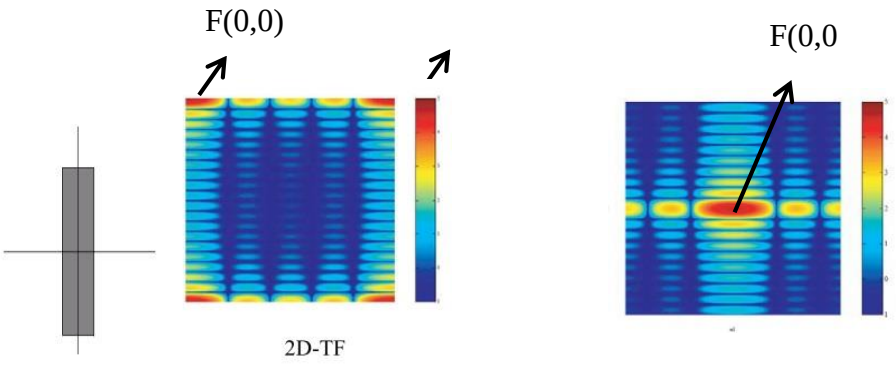
Además, Θ(u,v) es el ángulo de fase, y está dada por la ecuación (2.20):

$$\theta(u,v) = \arctan\left(\frac{I(u,v)}{R(u,v)}\right)$$

Donde
$R(u,v) : Parte\ Real$
$I(u,v) : Parte\ Imaginaria$ F(0,0) (2.20)

Véase un ejemplo de respuesta de la transformada de Fourier discreta bi-dimensional (2D)

$$\boldsymbol{f(x,y)} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad ; \quad \boldsymbol{F(u,v)} = \begin{bmatrix} 4 & -2-j2 & 0 & -2+j2 \\ -2-j2 & j2 & 0 & 2 \\ 0 & 0 & 0 & 0 \\ -2+j2 & 2 & 0 & -j2 \end{bmatrix}$$



(a) (b) (c)

Figura: 18 (a) Imagen Original (b) Transformada de Fourier "D de imagen original (c) 2D-TF con F(0,0) centrada

Fuente: [8] Dr. Wolf M.(2002) Quantitative Image Analysis Methods and Limitations. Biomedicalproducts. Ed. Springer. USA

En la primera Figura: 18(a) como en el ejemplo, tenemos la imagen original, en el dominio del espacio. Luego ésta, la pasamos al dominio de la frecuencia mediante su DFT-2 Dimensión, y notamos que la componente F (0,0) no aparece en el centro, las componentes de altas frecuencias (tiende al color rojo) se ubican a los extremos como el la Figura: 18 (b) mostrada anteriormente, entonces, lo que hacemos es un algoritmo para centrarlo al punto F(0,0) y poder apreciar estas frecuencias (ubicándolas en el centro).

Las imágenes usuales, sin variaciones muy bruscas, suelen tener los mayores valores de la transformada a bajas frecuencias, en torno a la frecuencia (0,0), que situaremos, usualmente, en el centro de la imagen. Figura: 18(c)

2.1.1.2.4 Filtro Pasa-bajo

El filtrado Pasa-Bajo como su nombre lo dice, deja pasar las bajas frecuencias y elimina las altas frecuencias, por lo que vendría a ser en imágenes, los cambios o variaciones bruscas que se producen entre pixeles.[5]

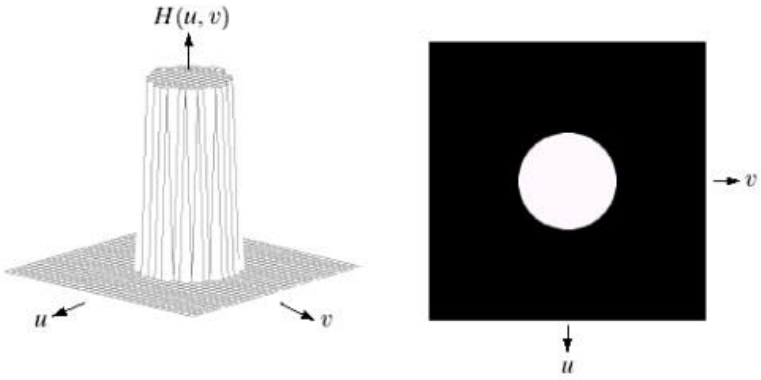
Como resultado de este tipo de filtros es una imagen suavizada, es decir, una imagen borrosa, especialmente en los contornos de la imagen donde por lo general se encuentran los mayores cambios de variación de pixeles o altas frecuencias.[9]

Existen varios tipos de filtros de suavizado, tanto en el dominio espacial, como ya vimos anteriormente como en el dominio frecuencial.

Idealmente un filtro ideal Pasa-bajo esta expresado por la siguiente función:

Un filtro ideal Pasa-bajo está denotado por la expresión:

$$H(u,v)=\begin{Bmatrix}1 & \sqrt{u^2+v^2},, Qt\\ 0 & else\end{Bmatrix}$$ Donde Qt es la frecuencia de corte.



(a) (b)

Figura: 19 (a) Grafica perspectiva de un filtro ideal pasa-bajo; (b) Filtro ideal pasa-bajo mostrado como imagen

Fuente: [8] Dr. Wolf M.(2002) Quantitative Image Analysis Methods and Limitations. Biomedicalproducts. Ed. Springer. USA

Un ejemplo de código de un filtro ideal pasa-bajo lo mostraremos en el Anexo 1

Como este tipo de filtro es difícil implementarlo por circuitos electrónicos u en otros dispositivos entonces se implementan otros tipos de filtros pasa-bajos

2.1.1.2.5 Filtro Gaussiano

El filtro Gaussiano es un tipo de filtro pasa-bajo y esta función Gaussiana en 1D se expresa en la ecuación (2.21):

$$g(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-u}{\sigma}\right)^2} \qquad (2.\ 21)$$

Donde

σ: Desviación

σ²: Varianza

x: Variable independiente

g(x)=y: Variable Dependiente

μ: Media

Si tomamos que su media ***(μ)*** sea cero, entonces nos quedaría la ecuación (2.22)

$$g(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{x^2}{2\sigma^2}}$$

$$ademas$$

$$\frac{1}{\sigma\sqrt{2\pi}} : Amplitud(A)$$

$$Resumiendo\ tenemos\ expresion\ final:$$

$$g(x) = A.e^{-\frac{x^2}{2\sigma^2}} \qquad (2.\ 22)$$

Veamos una gráfica de esta función donde su media ***(μ)*** es cero y su Amplitud se normalizó para A=1 constante y se escogió desviación σ=1/2.

Para x={-2,-1,0,1,2} (Step=1, Window=5)

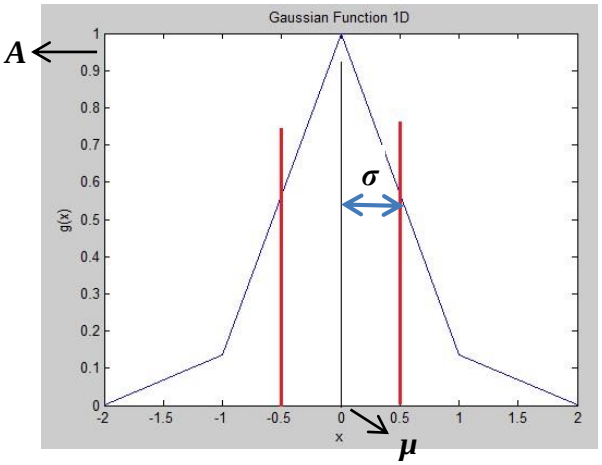


Figura: 20 Función Gaussiana 1D con x=[-2 2]

Fuente: Elaboración propia

Para x={-2,-1.8,-1.6,…,1.6,1.8,2} (Step=0.2, Window=20)

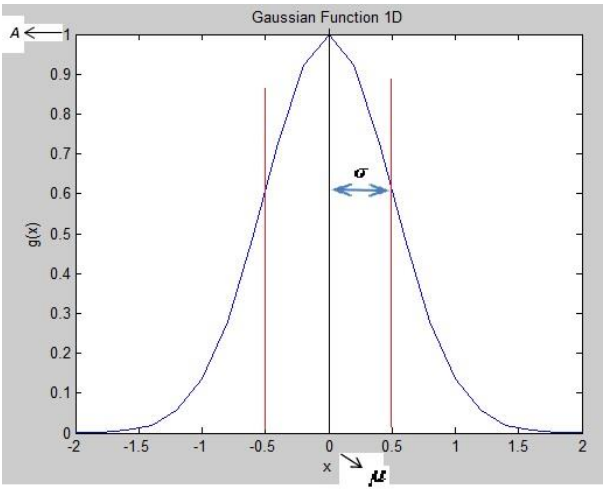


Figura: 21 Función Gaussiana 1D con x=[-2 2]

Fuente: Elaboración propia

Para procesamiento de imágenes, la función Gaussiana 2D, está dada por la ecuación (2.23):

$$g(x,y) = A.e^{-\left(\frac{(x-x_o)^2}{2\sigma_x^2}+\frac{(y-y_o)^2}{2\sigma_y^2}\right)} \quad (2.\ 23)$$

Donde "x" e "y" son las posiciones de los pixeles en las coordenadas X-Y (Euler, no matriciales donde el origen está en la esquina superior izquierda.

x_0 , y_0: Son Medias en los ejes X-Y

σ_x , σ_y: Son desviaciones en los ejes X-Y

Las propiedades del filtro Gaussiano en 2 Dimensiones son:

4. En 2D, las funciones Gaussianas son rotacionalmente Simétricos. Esto quiere decir que, la cantidad de suavizado ejecutado por el filtro será la misma en todas sus direcciones. En general, los bordes en una imagen no se verán orientados hacia una dirección en particular.

5. Esto quiere decir que el filtro Gaussiano reemplaza cada pixel de la imagen con un peso promedio de sus píxeles vecinos, tal que el peso dado decrece monótonamente con la distancia respecto al píxel central.

- El peso de sus píxeles decrece con la distancia al centro
- Cuanto más alejado esté el píxel, se vuelve menos significativo

6. El Fourier de un Gaussiano sigue siendo un Gaussiano porque el espectro de éste también tiene un único lóbulo.
7. El grado de filtrado es controlado por "σ". A mayor "σ" mayor suavizado, y se tiene en cuenta los puntos más alejados a la media "μ".
8. Filtros Gaussianos grandes pueden ser implementados muy efectivamente ya que éstas funciones Gaussianas son separables.

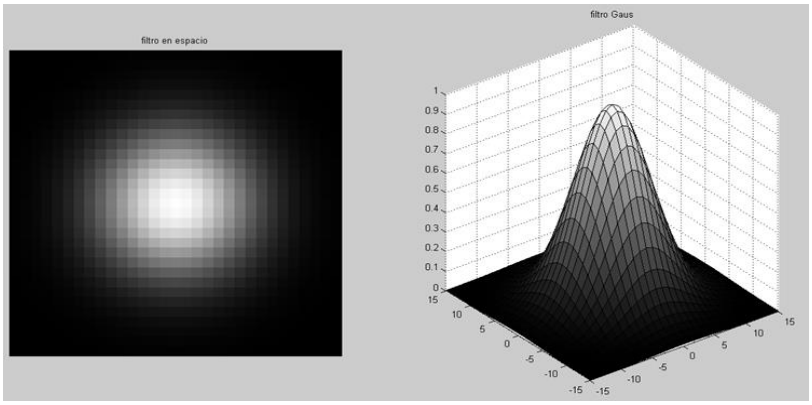


Figura: 22 Representación de Filtro Gaussiano

Fuente: Elaboración propia

- Propiedad "Simetría Rotacional"

La simetría rotacional de una función Gaussiana puede mostrarse al convertir la función de coordenadas rectangulares a coordenadas polares como expresaremos a continuación en la ecuación (2.24):

$$g(i,j) = -e^{\frac{i^2+j^2}{2\sigma^2}} \qquad (2.\ 24)$$

Cabe decir en la ec. 2.21, que tomamos como referencia una Amplitud A=1 constante.

Como el radio en coordenadas polares está definido por: $r^2 = i^2 + j^2$,como se muestra en la Figura 2.12, es fácil ver que la función Gaussiana en coordenadas polares puede definir en la ecuación (2.25):

$$g(r,\theta) = -e^{\frac{r^2}{2\sigma^2}} \qquad (2.\ 25)$$

Además, esta función, en coordenadas polares, no depende de Θ, y por ende, es rotacionalmente simétrico.

- Propiedad “Transformada de Fourier”

Esta propiedad dice que, la transformada de Fourier de una función Gaussiana, es también una función Gaussiana. Como la Transformada de Fourier de un Gaussiano es una función Real, la TF es su propia magnitud.

Ahora, la TF de un Gaussiano se computa por la ecuación (2.26):

$$\begin{aligned}\mathcal{F}\{g(x)\} &= \int_{-\infty}^{\infty} g(x)e^{-j\omega x}dx \\ &= \int_{-\infty}^{\infty} e^{\frac{-x^2}{2\sigma^2}}e^{-j\omega x}dx \\ &= \int_{-\infty}^{\infty} e^{\frac{-x^2}{2\sigma^2}}cos(\omega x)dx \; + j\int_{-\infty}^{\infty} e^{\frac{-x^2}{2\sigma^2}}sin(\omega x)dx\end{aligned} \quad (2.26)$$

El Gaussiano es una función Simétrica, y el Gaussiano es una función anti-simétrica, por ende, el integrando en la segunda integral es Anti-simétrica. Por ello, ésta segunda integral debe ser cero, y la transformada Fourier, según la ecuación. $g(i,j) = -e^{\frac{i^2+j^2}{2\sigma^2}}$ (2. 24) se simplificaría de la siguiente manera en la ecuación (2.27):

$$\begin{aligned}\mathcal{F}\{g(x)\} &= \int_{-\infty}^{\infty} e^{\frac{-x^2}{2\sigma^2}}cos(\omega x)dx \\ &= \sqrt{2\pi}\sigma e^{-\frac{\omega^2}{2\nu^2}}, \qquad \nu^2 = \frac{1}{\sigma^2}\end{aligned} \quad (2.\ 27)$$

Vemos que “ω” es el parámetro de frecuencia espacial, y la desviación (propagación) del Gaussiano en el dominio de la frecuencia está controlada por “υ”, el cual es recíproco del parámetro “σ” en el dominio espacial. Esto quiere decir:

- Estrecho Gaussiano en el dominio espacial, tiene un espectro más ancho (Espectro en frecuencia). Esto quiere decir que hay menos Smoothing (suavizado), y en el dominio frecuencial tiene un mayor ancho de banda y pasan más altas frecuencias (Ancho Espectro)
- Un Ancho Gaussiano en el dominio espacial tiene un estrecho Espectro. Esto quiere decir que hay mucho más Smoothing y en el dominio frecuencial pasan menos altas frecuencias, ruido, etc.

Esta simple relación entre el dominio espacial, y el dominio frecuencial Espectral, mejora la facilidad de uso del filtro Gaussiano en usos prácticos.[10]

- Propiedad de "Separabilidad Gaussiana"

La separabilidad de los filtros Gaussianos la demostré haciendo lo siguiente en las ecuaciones (2.28) y (2.29):

$$\begin{aligned} g(i,j) * f(i,j) &= \sum_{k=1}^{m}\sum_{l=1}^{n} g(k,l) f(i-k, j-l) \\ &= \sum_{k=1}^{m}\sum_{l=1}^{n} e^{-\frac{k^2+l^2}{2\sigma^2}} f(i-k, j-l) \\ &= \sum_{k=1}^{m} e^{-\frac{k^2}{2\sigma^2}} \left\{ \sum_{l=1}^{n} e^{-\frac{l^2}{2\sigma^2}} f(i-k, j-l) \right\} \end{aligned} \qquad (2.28)$$

Donde la suma en llaves es la Convolución de la imagen de entrada f(i,j) con una función vertical Gaussiana de una dimensión (1D-Gaussian vertical function).

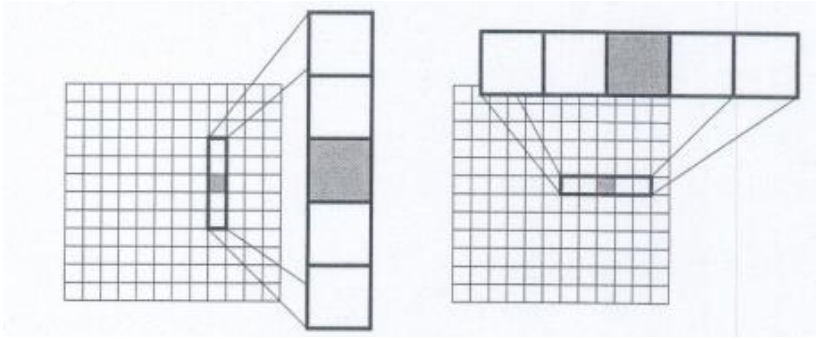


Figura: 23 Ejemplo de la separabilidad de una convolución Gaussiana. A la izquierda, convolución con máscara vertical. A la derecha, convolución con máscara horizontal. El origen de la máscara esta sombreado

Fuente: [1] Guevara A., Jornales F., Alpha V.y Alvarez F. (2006) Técnicas y Algoritmos Básicos de Visión Artificial. Universidad de la Roja. España

Al tener esta propiedad de separabilidad, para una imagen de NxN ya no estaríamos haciendo N^2 operaciones. Con la convolución esto se reduce a 2N operaciones, obteniendo de esta manera un tiempo de procesamiento computacional, mucho más reducido y más efectivo.

Diseño de filtro Gaussiano

En imágenes, debemos computar una máscara espacial el cual vendría a ser nuestro filtro gaussiano en el espacio, luego éste se pasa al dominio de la frecuencia y se aplica el proceso de filtrado en el dominio de la frecuencia como muestra la Figura: 22

$$g(i,j) = ce^{-\frac{i^2+j^2}{2\sigma^2}} \qquad (2.\ 30)$$

Ahora, para saber, o computar los pesos de la máscara directamente de la distribución Gaussiana, debemos primero mirar la expresión de la ecuación. (2.30)

Donde "c" es una constante normalizada. Reescribiendo en la ecuación (2.31):

$$\frac{g[i,j]}{c} = e^{-\frac{i^2+j^2}{2\sigma^2}} \qquad (2.\ 31)$$

Escogiendo una desviación σ^2, nosotros podremos evaluarlo en una ventana de "*n x n*" para obtener el kernel o la máscara la cual, para evaluado en [0,0] es igual a 1.

Por ejemplo escogiendo $\sigma^2 = 2$ y $n = 7$, la expresión de la ecuación 2.30 produce el arreglo:

Tabla: 1 Pesos Gaussianos con $\sigma^2 = 2\ y\ n = 7$

[i,j]	**-3**	**-2**	**-1**	**0**	**1**	**2**	**3**
-3	.011	.039	0.82	.105	.082	.039	.011
-2	.039	.135	.287	.368	.287	.135	.039
-1	.082	.287	.606	.779	.606	.287	.082
0	.105	.368	.779	1.000	.779	.368	.105

1	.082	.287	.606	.779	.606	.287	.082
2	.039	.135	.287	.368	.287	.135	.039
3	.011	.039	.082	.105	.082	.039	.011

Nota. Fuente: Elaboración propia

Es mucho mejor hacer que estos pesos sean enteros, y no decimales como los mostramos anteriormente en la Tabla1, para facilidad computacional, entonces para hacer ello, tomamos un valor de las esquinas de la matriz (el que la mayoría de libros toma es el punto [3,3]) y escogemos c tal que este valor se vuelve 1. Usando el ejemplo anterior sería de la siguiente manera:

$$\begin{aligned} \frac{g[3,3]}{c} &= e^{-\frac{3^2+3^2}{2.2}} \\ &= 0.011 \\ \Rightarrow c &= \frac{g[3,3]}{0.011} \\ &= \frac{1.0}{0.011} \\ &= 91 \end{aligned}$$

Ahora multiplicando el resto de los pesos por "c" tenemos:

Tabla: 2 Pesos enteros para filtro Gaussiano con $\sigma^2 = 2\ y\ n = 7$

[i,j]	**-3**	**-2**	**-1**	**0**	**1**	**2**	**3**
-3	1	4	7	10	7	4	1
-2	4	12	26	33	26	12	4
-1	7	26	55	71	55	26	7
0	10	33	71	91	71	33	10
1	7	26	55	71	55	26	7
2	4	12	26	33	26	12	4
3	1	4	7	10	7	4	1

Nota. Fuente: Elaboración propia

La tabla anterior muestra la máscara convolución resultante para un filtro Gaussiano. Sin embargo los pesos de la máscara no suman 1. Para ello, cuando ejecutamos la convolución, las salidas de los pixeles deben estar normalizados por la suma de los pesos de la máscara para asegurar que las regiones de intensidad uniforme no se vean afectadas. Por lo tanto aplicamos la suma de pesos en la siguiente ecuación (2: 32):

$$\sum_{i=-3}^{3}\sum_{i=-3}^{3} g[i, j] = 1115$$

Por ello,

$$h[i, j] = \frac{1}{1115}\left(f[i, j] * g[i, j]\right) \qquad (2.\ 32)$$

Donde f[i,j] es la imagen a filtrar, y g[i,j] es el filtro gaussiano con sus pesos de valor entero.

Cabe resaltar que el filtrado que expresamos en la ecuación. (2.31) es un filtrado en el dominio del espacio. Ahora si deseamos hacerlo en la frecuencia se pasaría a proceder con los pasos mencionados en la sección 2.2.4 de Filtrado en Frecuencia.

El código fuente para generar este tipo de ventana lo vemos en el Anexo 2

2.1.1.2.6 Filtro Pasa-Alto

El Filtro pasa Alto, como su mismo nombre lo dice, deja pasar las altas frecuencias, mientras que las bajas no las deja pasar. Este tipo de filtro se usa normalmente en la detección de bordes.

El filtro pasa alto trabaja de la misma manera que el filtro pasa bajo, sólo que usa diferente máscara de convolución. Los coeficientes de la máscara deben sumar 0.

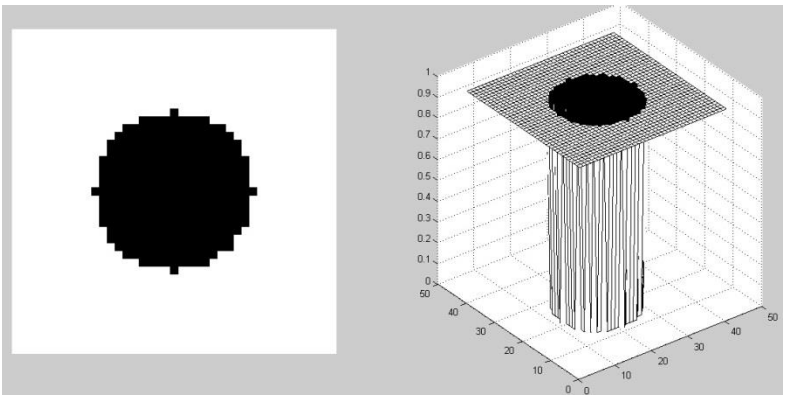
En general el filtro pasa alto reduce mucho el contraste de la imagen.

Ahora bajo este tipo de filtro existen varios tipos de filtros o mascaras a usar como son las de Prewit, Roberts, Sobel, etc.

Un filtro ideal pasa-alto viene dado por la expresión:

$$H(u,v)=\begin{Bmatrix}1 & \sqrt{u^2+v^2}>Qt\\ 0 & else\end{Bmatrix}$$ Donde Qt es la frecuencia de corte.

Los filtros pasa-alto atenúan las componentes de baja frecuencia y dejan intactas las de medias-altas en función de la frecuencia de corte que se elija. Se usan para quedarnos con las propiedades de la imagen en los que los niveles de gris varían bruscamente, por bordes de la imagen.



(a) (b)

Figura: 24 Filtro ideal pasa-bajo mostrado como imagen; (b) Grafica perspectiva de un filtro ideal pasa-bajo
Fuente: Elaboración propia

Un ejemplo de código para generar un filtro ideal Pasa-alto se aprecia en el Anexo 3

2.1.1.3 Operaciones Morfológicas

Las Operaciones Morfológicas principalmente extraen y alteran la estructura de las partículas de una imagen (pixels), para el análisis.

Las operaciones morfológicas que se realizan son principalmente para:

- Expandir o reducir partículas
- Llenar agujeros

- Cerrar inclusiones
- Suavizar bordes
- Remover extensiones

La morfología generalmente se aplica en imágenes binarias, pero hay teorías que ese aplican en imágenes escala grises.

Estos procesamientos morfológicos de las imágenes se basan en la teoría de conjuntos. [1]

2.1.1.3.1 Elemento Estructurante

Es una máscara binaria o a nivel de grises usada para las trasformaciones morfológicas.

Se utiliza para pesar el efecto de esa función en la forma y vecindad de las partículas o pixels.

El tamaño y la forma se escogen de acuerdo a las formas que se deseen extraer.

2.1.1.3.2 Erosión

Elimina los píxeles aislados en el fondo y erosiona (reduce) el contorno de las partículas con respecto al patrón definido por el Elemento Estructurante.

Es generalmente usada para el Adelgazamiento de regiones, y está dada por el siguiente algoritmo matemático:

$A!\ \ B = \{x \mid B_x \subseteq A\}$

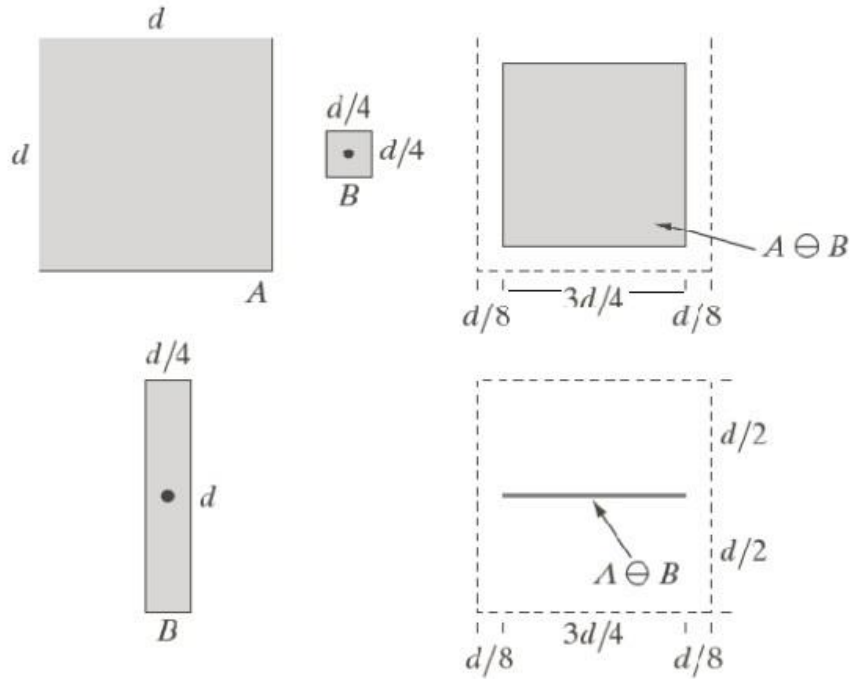
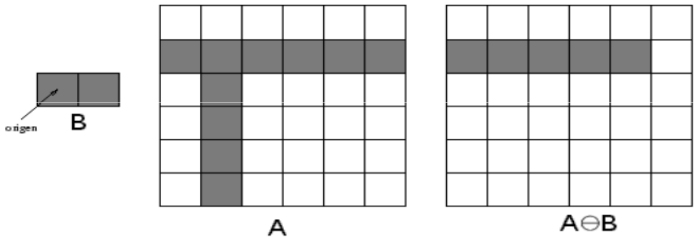


Figura: 25 (a) Conjunto A. (b) Elemento estructural. (c) Erosión de A por B sombreado. (d) Elongación de Elemento Estructurante. (e) Erosión de A por B utilizando este elemento estructurante. El borde punteado en (c) y (e) es el borde del conjunto A, mostrado solo como referencia.

Fuente: [1] Guevara A., Jornales F., Alpha V.y Alvarez F. (2006) Técnicas y Algoritmos Básicos de Visión Artificial. Universidad de la Roja. España

Para la implementación del algoritmo de erosión vemos el código fuente mostrado en el Anexo 4

Ahora mostraremos otro ejemplo de erosión:



(a) (b) (c)

Figura: 26 Ejemplo Erosión: (a) Elemento Estructurante, con origen señalado. (b) Imagen a erosionar. (c) Imagen erosionada con Elemento Estructurante.
Fuente: [1] Guevara A., Jornales F., Alpha V.y Alvarez F. (2006) Técnicas y Algoritmos Básicos de Visión Artificial. Universidad de la Roja. España

La erosión es la operación morfológica dual de la dilatación. La erosión se concibe usualmente como una reducción de la imagen original.

2.1.1.3.3 Dilatación

Sea una imagen A y un elemento estructural B, siendo ambas imágenes BINARIAS, entonces definimos la dilatación como:

$$A \oplus B = \left\{ x \mid (\hat{B})_x \cap A \neq 0 \right\}$$

La dilatación se obtiene en base a la reflexión de B definido como $\hat{B}$ con respecto a su origen y un desplazamiento x.

La salida de la dilatación es el conjunto de puntos barridos (pixels) por el centro del Elemento Estructurante (EE), mientras algún punto de $\hat{B}$ tocaba o coincide con algún punto de A (barriendo los pixels de A por el centro de "B")

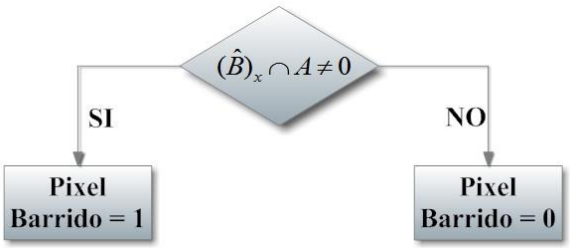
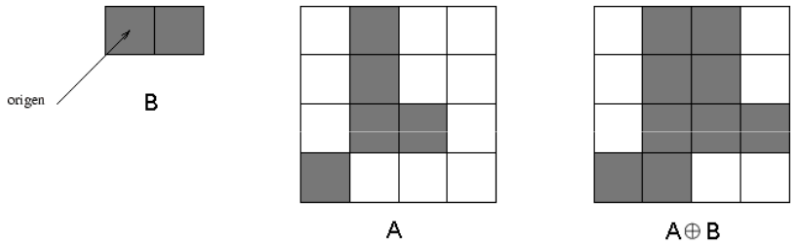


Figura: 27 Diagrama de flujo de algoritmo Dilatación
Fuente: [1] Guevara A., Jornales F., Alpha V.y Alvarez F. (2006) Técnicas y Algoritmos Básicos de Visión Artificial. Universidad de la Roja. España



(a) (b) (c)

Figura: 28 Ejemplo de Dilatación: (a) Elemento Estructurante, con origen señalado. (b) Imagen a Dilatar. (c) Imagen Dilatada con Elemento Estructurante
Fuente: [1] Guevara A., Jornales F., Alpha V.y Alvarez F. (2006) Técnicas y Algoritmos Básicos de Visión Artificial. Universidad de la Roja. España

Para la implementación del algoritmo de Dilatación vemos el código fuente mostrado en el Anexo 5

2.1.1.3.4 Apertura

Sea una imagen A y un elemento estructural B, siendo ambas imágenes BINARIAS, entonces definimos la Apertura en la ecuación (2. 33):

$$A^{\circ}B = (A! \quad B) \oplus B \qquad (2.\ 33)$$

Que, en palabras sencillas, establece que la apertura de A por B es simplemente la erosión de A por B, seguido de la dilatación del resultado por B.

Remueve pequeñas partículas y suaviza los bordes. No altera el tamaño de las partículas. Las pequeñas partículas que desaparecen durante la erosión no vuelven a aparecer después de la dilatación.

Si A no cambia con la apertura con K, se dice que A es abierto respecto a K.

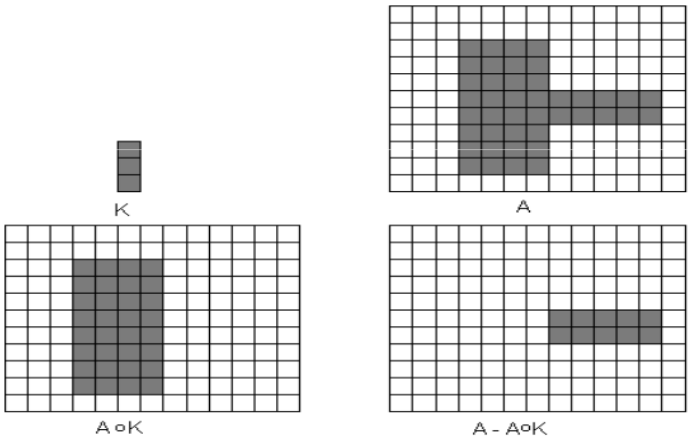


Figura: 29 Ilustra la apertura de una imagen A respecto a un Elemento Estructurante K. En la parte inferior muestra el procedimiento para separar las dos partes de la figura.
Fuente: [1] Guevara A., Jornales F., Alpha V.y Alvarez F. (2006) Técnicas y Algoritmos Básicos de Visión Artificial. Universidad de la Roja. España
Para la implementación del algoritmo de Apertura vemos el código fuente mostrado en el Anexo 6

2.1.1.3.5 Cerradura

Sea una imagen A y un elemento estructural B, siendo ambas imágenes BINARIAS, entonces definimos la Cerradura en la ecuación (2. 34):

$$A \bullet B = (A \oplus B) \,!\quad B \qquad (2.\ 34)$$

Que, en palabras, establece que la clausura de A por B es la dilatación de A por B, seguido de la erosión del resultado por B

Llena pequeños agujeros y suaviza los bordes. No altera el tamaño de las partículas. Al realizar la erosión los agujeros no vuelven a aparecer.

Se obtiene desplazando el EE “B” por el exterior del conjunto y anexionando las zonas por las que B no pueda “pasar”.

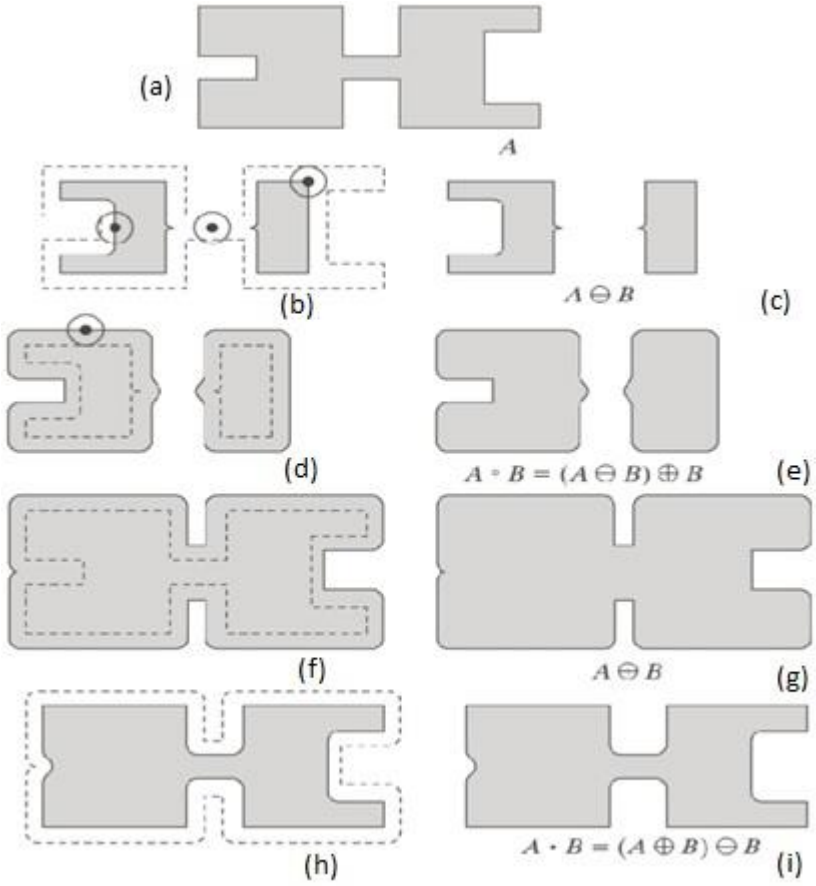


Figura: 30 Operaciones morfológicas de Apertura y Cerradura (Clausura). El elemento estructurante es el pequeño círculo mostrado en varias posiciones en (b). El Elemento Estructurante no estaba sombreado por claridad. El punto oscuro es el centro del Elemento Estructurante.

Fuente: [1] Guevara A., Jornales F., Alpha V.y Alvarez F. (2006) Técnicas y Algoritmos Básicos de Visión Artificial. Universidad de la Roja. España

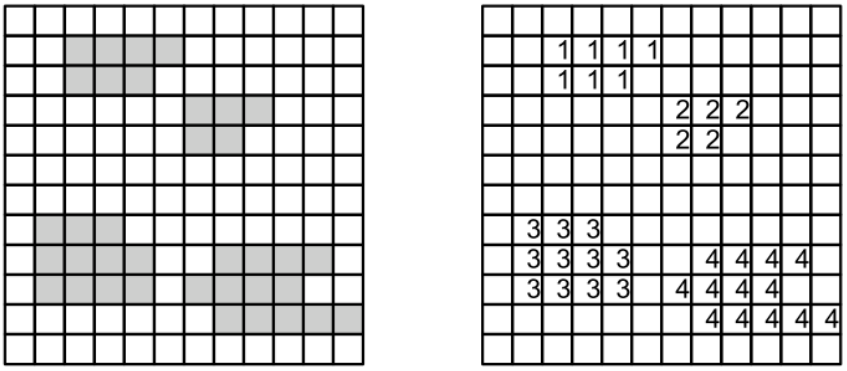
Para la implementación del algoritmo de Dilatación vemos el código fuente mostrado en el Anexo 8

2.1.1.4 Etiquetado

En la práctica llegar a la imagen binaria no suele ser suficiente para realizar una descripción adecuada, por lo que el proceso de segmentación se prolonga aplicando diversas técnicas sobre este tipo de imágenes.

Existe una gran cantidad de técnicas de análisis para imágenes binarias, con propósitos tan variados, entre estas técnicas está el contar, etiquetar objetos y filtrado de objetos según su tamaño.

Una de las operaciones más comunes en visión es encontrar las componentes conectadas dentro de una imagen. En la Figura: 27 vemos un ejemplo de una imagen y su imagen de componentes conectadas, en donde se ha etiquetado con un número a las componentes conectadas.



(a) (b)

Figura: 31 (a) Imagen a etiquetar. (b) Etiquetado y cuenta de objetos
Fuente: [10] Jackson O. Söderkvist O. (2004) Computer Vision Classification of Leaves from Swedish Trees (Master in Science). Linköping University. Computer Vision Laboratory (CVL). Suiza

Se ha de disponer de una definición consistente de conectividad para demarcar todos los objetos en imágenes binarias y ser capaces de idear algoritmos para etiquetarlos y contarlos.

El etiquetado es una técnica que, partiendo de una imagen binaria, nos permite etiquetar cada uno de los objetos conectados presentes en la imagen. Esto va a posibilitar:

- Distinguir los objetos respecto del fondo (propiedad intrínseca de la imagen binaria).
- Distinguir un objeto respecto de los demás objetos.
- Conocer el número de objetos en la imagen.

2.1.1.5 Representación de los Objetos

La representación de objetos se puede hacer de una de las dos formas siguientes:

A. Descripción **interna:** En términos de píxeles que componen la región [1]
B. **Descripción externa:** en términos de los píxeles que componen su contorno [1]

Un ejemplo de representación interna es el esqueleto de un objeto, mientras que la representación externa es una aproximación ya sea poligonal, códigos de cadena, los descriptores de Fourier, entre otras.

En este proyecto de tesis se utilizó como representaciones las aproximaciones por código de cadena y por descriptores de Fourier como Descripción externa ya que éstas representan un mejor resultado para la extracción de características en procesamiento de imágenes.

Para hacer una muy buena representación de un objeto, es necesario que nuestros esquemas de representación del objeto a procesar (código de cadena o Descriptor de Fourier), deben cumplir con las siguientes propiedades:

A. **Unicidad:** Cada objeto debe tener una única representación. [18]
B. **Invarianza ante transformaciones geométricas:** La invarianza ante traslaciones, rotaciones, escalamientos y reflexiones es una característica muy importante para las tareas de reconocimiento visual de patrones.[18]
C. **Sensibilidad:** Es la propiedad del esquema de representación escogido de reflejar diferencias entre objetos similares.[18]
D. **Abstracción de detalles:** Propiedad de la representación de mantener las principales características de la forma del objeto y abstraerse de los detalles. Propiedad relacionada con la robustez de la representación ante el ruido.[18]

Al momento de realizar el procesamiento de la imagen, se puede encontrar el objeto en diferente tamaño, rotado, trasladado, escalado, para lo cual se tienen que emplear algoritmos los cuales permiten identificar el objeto frente a estas variaciones

A. TRASLACIÓN

El coeficiente $F(0)$ representa el centro de gravedad de la curva (en este caso, el contorno del objeto), entonces si existe una traslación de la curva como:*[1]*

$$f_t(n) = f(n) + d_0 \qquad \text{siendo} \quad d_0 = x_0 + iy_0 \qquad (3.\ 1)$$

Ésta variación de traslación única y solamente afecta al término $F(0)$ de Fourier, es decir:

$$F_t(0) = F(0) + d_0 \qquad (3.\ 2)$$

Entonces, nuestra transformada de Fourier se vuelve de la forma:

$$F_t(u) = \sum_{n=0}^{N-1} \left[f(n) + d_0 \right] e^{-j2\Pi nu/N} \qquad (3.\ 3)$$

$$F_t(u) = \frac{1}{N} \sum_{n=0}^{N-1} f(n)\, e^{-j2\Pi nu/N} + \frac{1}{N} \sum_{n=0}^{N-1} d_0\, e^{-j2\Pi nu/N} \qquad (3.\ 4)$$

$$F_t(u) = F(u) + d_0 \Psi(u) \qquad (3.\ 5)$$

Siendo $F(u)$ la transformada Fourier de la función $f(n)$ y d_0 la traslación hecha.

B. ESCALAMIENTO

Si la imagen respecto al origen es escalada por un factor “S” como:[1]

$$g(n) = S\ f(n) \tag{3. 6}$$

Entonces su Transformada Discreta de Fourier estará también escalada por el mismo factor como

$$G(N) = SF(u) \tag{3. 7}$$

C. ROTACIÓN

Si la imagen es rotada respecto a su origen por un ángulo Θ como:[1]

$$g(n) = f(n)\, e^{j\Theta} \tag{3. 8}$$

Entonces su transformada de Fourier está multriplicada por su mismo factor

$$G(N) = F(u)\, e^{j\Theta} \tag{3. 9}$$

Se aplicó el descriptor Fourier según muestra la ecuación (3.16), para describir la imagen según las características que éste presenta, siendo esta invariante ante escalamiento, rotaciones y traslaciones, permitiendo de ésta manera, reconocer en otra imagen, que probablemente haya sido tomada en diferente posición, rotación, acercamiento, el objeto a identificar, sometiéndolo a éste algoritmo descriptor de Fourier.

2.1.1.6 Redes neuronales

Las redes neuronales es una de las técnicas que se pueden usar para el reconocimiento de imágenes.

Para saber la manera del cómo y porqué se aplican las redes neuronales en el procesamiento de imágenes es necesario definir unos conceptos.

Primero se tiene que saber que la red Neuronal presenta dos Capacidades, la de aprendizaje, que es la capacidad de recoger información de las experiencias pasadas y utilizar esa información para poder uno actuar ante situaciones FUTURAS (Learning by Experience),[16] mientras que la capacidad de Generalización es la capacidad por la cual posee para poder abstraer sólo información relevante y útil más allá de los casos particulares. De ésta manera las Redes Neuronales son capaces de responder ante casos completamente desconocidos. [18]

Según este par de definiciones se puede comenzar a describir lo que sería una Red Neuronal Artificial.

Las Redes neuronales Artificiales como lo definimos anteriormente tratan de abstraer capacidades del cerebro para resolver problemas o situaciones complejas como, la visión, y reconocimiento de patrones. También la red neuronal artificial es capaz de almacenar conocimiento experimental.

Como se menciona, el conocimiento experimental, y la abstracción relevante son la base para la aplicación de Redes Neuronales en procesamiento de Imágenes.

Como se vio, una imagen es de dos dimensiones, por lo que se le aplicaron una serie de algoritmos, para poder identificar al objeto y describirlo. Ahora, al aplicar la teoría de Redes Neuronales lo que se hace es identificar esas características que se extrajeron del algoritmo de Fourier de la imagen, cumpliendo con una de las capacidades de la red neuronal, y para que la red neuronal aprenda en base a experiencias, se le somete a este mismo algoritmo, un conjunto de imágenes del objeto a identificar y de otro objeto que no es objetivo de nuestra identificación. Por lo que la base de datos de la red Neuronal se va haciendo mucho más compleja y robusta en la identificación del objeto, haciéndolo de ésta manera muy efectiva en el procesamiento de imágenes.

Las Redes Neuronales Artificiales (RNA) están inspiradas en la biología, esto significa que están formadas por elementos que se comportan de manera análoga a las neuronas (en las funciones más elementales) y están organizadas de una forma similar a la del cerebro, pero las analogías no son muchas más.

Las características fundamentales de las RNA son:

• Aprenden de la experiencia: Las RNA pueden modificar su comportamiento como respuesta a su entorno. Dado un conjunto de entradas (quizá con las salidas deseadas), las RNA se ajustan para producir respuestas consistentes. Una amplia variedad de algoritmos de entrenamiento se han desarrollado, cada uno con sus propias ventajas e inconvenientes.

• Generalizan de ejemplos anteriores a los ejemplos nuevos: Una vez que la RNA esté entrenada, la respuesta de la red puede ser, hasta un cierto punto, insensible a pequeñas variaciones en las entradas, lo que las hace idóneas para el reconocimiento de patrones.

• Abstracción de la esencia de las entradas: Algunas RNA son capaces de abstraer información de un conjunto de entradas. Por ejemplo, en el caso de reconocimiento de patrones, una red puede ser entrenada en una secuencia de patrones distorsionados de una letra. Una vez que la red sea correctamente entrenada será capaz de producir un resultado correcto ante una entrada distorsionada, lo que significa que ha sido capaz de aprender algo que nunca había visto.

Redes de capa simple

2.1.1.6.1 Redes Neuronales Unicapa

A pesar de que una sola neurona puede realizar modelos simples de funciones, su mayor productividad viene dada cuando se organizan en redes. La red más simple es la formada por un conjunto de perceptrones a

los que entra un patrón de entradas y proporcionan la salida correspondiente. Por cada perceptrón que tengamos en la red vamos a tener una salida, que se hallará como se hacía con un perceptrón solo, haciendo el sumatorio de todas las entradas multiplicadas por los pesos. Al representar gráficamente una red, se añade una "capa" inicial que no es contabilizada a efectos de computación, solamente sirve para distribuir las entradas entre los perceptrones. La denominaremos la capa 0.

De esta manera, la representación gráfica de una red de capa simple sería la siguiente como se muestra en la Figura: 32:

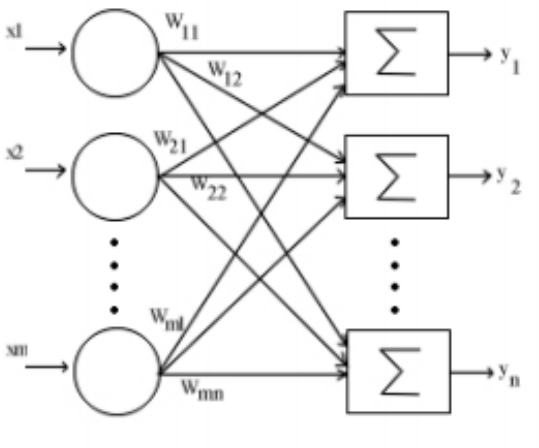


Figura: 32 Ejemplo de Red Neuronal Unicapa

Fuente: [14] Garrick D. (2013) Principles of Artificial Neural Networks. University of Illinois. USA

2.1.1.6.2 Redes Neuronales Multicapa

Estas redes Neuronales tienen una estructura que contempla una capa de entrada, una de salida y una capa oculta, tal y como se muestra en la Figura: 33

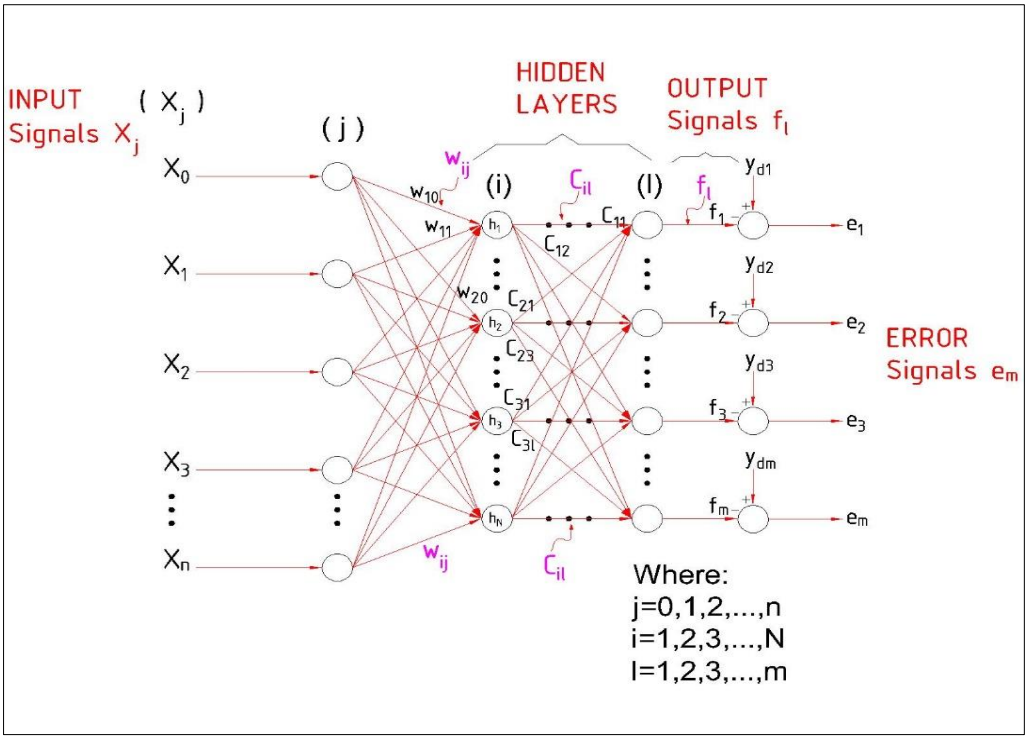


Figura: 33 Modelamiento Red Neuronal MultiCapa

Fuente: Elaboración propia

Generalmente se usan las redes multicapa ya que es una herramienta poderosa al momento de tratar con comportamientos no lineales. Sirve para resolver problemas cuyas soluciones resultaron insuficientes por los métodos tradicionales.[19]

Esto consiste en un método gradiente cuya función de activación es tangente hiperbólica.

En la Figura: 34 se muestra la función de activación usada en la red BackPropagation

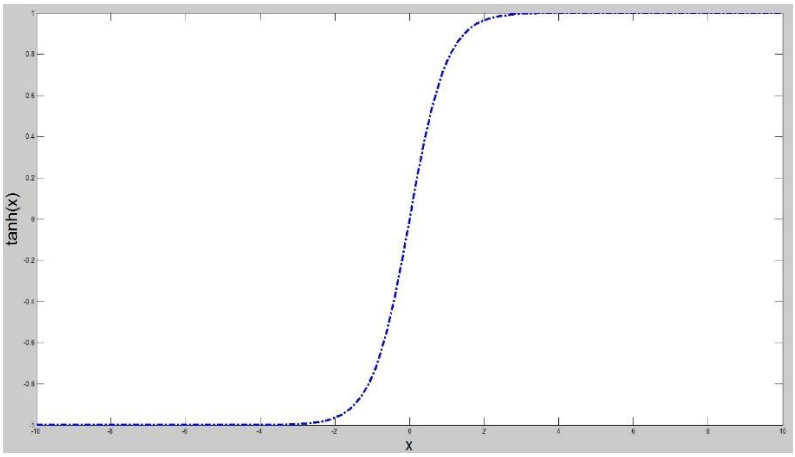


Figura: 34 Función de activación tangente hiperbólica

Fuente: Elaboración propia

A diferencia de la función sigmoidea, esta función de activación permite valores positivos y negativos, por lo que, en el trabajo de las redes neuronales, se trabaja con valores bipolares está función resulta la más adecuada ya que nos permite obtener tal particularidad a diferencia de la sigmoidea. [5]

La función sigmoidea está dada por la siguiente función (3.1):

$$g(a) = sigm(a) = \frac{1}{1 + exp(-a)} \qquad (3.\ 10)$$

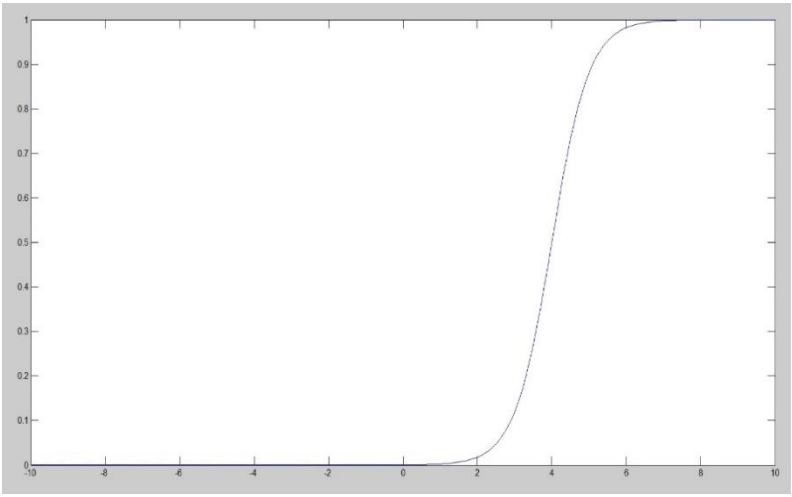


Figura: 35 Función de activación Sigmoidea

Fuente: Elaboración propia

En la Figura: 35 se aprecia la gráfica de una función sigmoidea.

Al derivar la función sigmoidea nos quedaría la función (3.2):

$$g'(a) = g(a)(1 - g(a)) \qquad (3.\ 11)$$

Según se observa, para que su derivada sea 0, $g(a)$ tendría que ser 0 o 1, y para esto sí "a" tiende al infinito negativo $g(a)$ se hace 1, y cuando "a" tiende al infinito positivo, $g(a)$ tiende a 0, por lo que notamos que la función sigmoidea tiene como máximos valores entre 0 y 1, por lo que si queremos tomar valores bipolares en nuestra red neuronal no sería posible aplicar esta función de activación por lo que se optó por la función tangente hiperbólica.

La función tangente hiperbólica está dada por la ecuación (3.17):

$$g(a) = tanh(a) = \frac{exp(a) - exp(-a)}{exp(a) + exp(-a)} = \frac{exp(2a) - 1}{exp(2a) + 1} \qquad (3.\ 17)$$

Y la derivada de la función tangente hiperbólica sería:

$$g'(a) = 1 - g(a)^2 \qquad (3.\ 18)$$

Y de nuevo, si se analiza la ecuación se aprecia que la función tangente hiperbólica toma valores de entre -1 a +1, por lo que permite trabajar con valores bipolares, y su análisis por gradiente (derivada) BackPropagation es mejor para el aprendizaje en la neurona, y rapidez computacional.[5]

Como se dijo, se usó la función tangente hiperbólica, la cual nos permite el trabajo con valores bipolares en nuestra red neuronal.

A continuación se muestra la manera como se hizo el aprendizaje de la red neuronal en este proyecto de tesis.

Se halló vector de entrada de la imagen invariante ante rotaciones, traslaciones y escalamiento denominado como $\hat{x}$ definido como nuestro vector de características de la imagen.

Se tiene definido un vector de salida pre-definido el cual definimos que es una lata en base a mi vector de entrada. Entonces, por lo mismo que conocemos el vector de entrada, y mi salida respectiva, estamos hablando de un aprendizaje Supervisado.

2.1.1.7 Esteoroscopía

La estereoscopía es cualquier técnica capaz de recoger información visual tridimensional y/o crear la ilusión de profundidad mediante una imagen 3D (tridimensional). La ilusión de la profundidad en una fotografía, película, u otra imagen bidimensional se crea presentando una imagen ligeramente diferente para cada ojo, como ocurre en nuestra forma habitual de ver. Muchas pantallas 3D usan este método para transmitir imágenes.

La fotografía estereoscópica tradicional consiste en crear una ilusión 3-D a partir de un par de imágenes 2D. La forma más sencilla de crear en el cerebro la percepción de profundidad es proporcionando a los ojos del espectador dos imágenes diferentes, que representan dos perspectivas del mismo objeto, con una pequeña desviación similar a las perspectivas que de forma natural reciben los ojos en la visión binocular. [20]

Para lograr la estereoscopía como se relató en la breve reseña anterior, se aplicó el uso de 2 cámaras HD Microsoft, simulando el ojo humano, y su forma de percibir los objetos u el ambiente que lo rodea.

Sin embargo no es el simple hecho del uso de 2 cámaras, con 1 tengo el objeto identificado, lo mismo con la otra, sin embargo cada cámara tiene un eje de coordenadas diferente ya que tiene distintas referencias, lo mismo que el ojo humano, y según el ojo podemos ver un objeto a la izquierda u el otro un poco más a la derecha. Es por esto que se tuvo que definir bien estos parámetros tanto de coordenadas como el eje focal del lente de las cámaras.

La estereoscopía en visión artificial es simplemente el uso de la geometría y relación de un eje respecto del otro y de cómo se comportan los sistemas en relación.

CAPÍTULO 3:

DISEÑO DEL SISTEMA DE VISIÓN ARTIFICIAL

3.1 Adquisición de la Imagen y descripción de hardware utilizado

Este proceso es muy importante para reducir ruidos y otros factores que posiblemente afectan al sistema en los siguientes pasos de procesamiento de la imagen.

Como primer paso, se intenta que la imagen sea lo más adecuada posible para que se pueda continuar con las siguientes etapas.

Debido a esto y otras razones, por las que se optó por realizar las pruebas en un lugar donde las condiciones de luminosidad eran las adecuadas, y el medio también era el adecuado como para hacer la captura de las imágenes seguidamente procesar. Tal como se muestra en la Figura: 36

Esto se realizó con Cámaras HD para PC (Webcam) de Microsoft Modelo CINEMA HD, los cuales tienen una resolución de hasta 5M pixels, y sistema Autofocus, por lo que permite una mejor captura y resolución de la imagen, para ir eliminando ruidos ocurrentes del mismo medio adquisitivo de la imagen.

Figura: 36 Ejemplo de Adquisición de la Imagen a Procesar
Fuente: Elaboración propia

3.2 Pre-procesamiento y Segmentación de la imagen

Una vez hecha la Adquisición de la imagen, se pasó a procesar la imagen, primero identificando que tipo de objetos queremos en nuestro proceso final llegar a ver, u obtener mediante los algoritmos de Visión Artificial.

En este tema de Tesis, se requiere que el sistema reconozca latas, no solo de un color de latas, si no, cualquier tipo de color de lata que se encuentre en cualquier posición, ya que en la realidad, podemos encontrarlas tanto en posición vertical, como completamente reclinadas o semi-inclinadas como muestra claramente en la Figura: 36.

Se analizó una serie de imágenes digitales a las cuales debían aplicarse diversas técnicas de procesamiento y ver cuáles de ellas serian implementadas.

La secuencia que se siguió en esta etapa fue la siguiente:

- Conversión a niveles HSV de la imagen inicial RGB
- Corte de imagen
- Filtrado
- Umbralización
- Erosión y dilatación
- Etiquetado de la imagen binarizada

3.2.1 Conversión imagen HSV

En un primer paso para el pre-procesamiento de la imagen, se realizó la conversión de una imagen RGB a una Imagen HSV la cual viene de las palabras Hue-Saturation-Value.

Se realizó esto siguiendo el siguiente algoritmo en C:

Figura: 37 Algoritmo de conversión de imagen RGB a HSV

Fuente: [5] Ramesh J., Rangachar K., Gilmore B. Schunck (1995). Machine Vision. McGraw Hill. USA

Ciertamente en Matlab hay una simple función la cual corrobora el algoritmo implementado anteriormente en la Figura: 37 con la función "rgb2hsv(Image)"

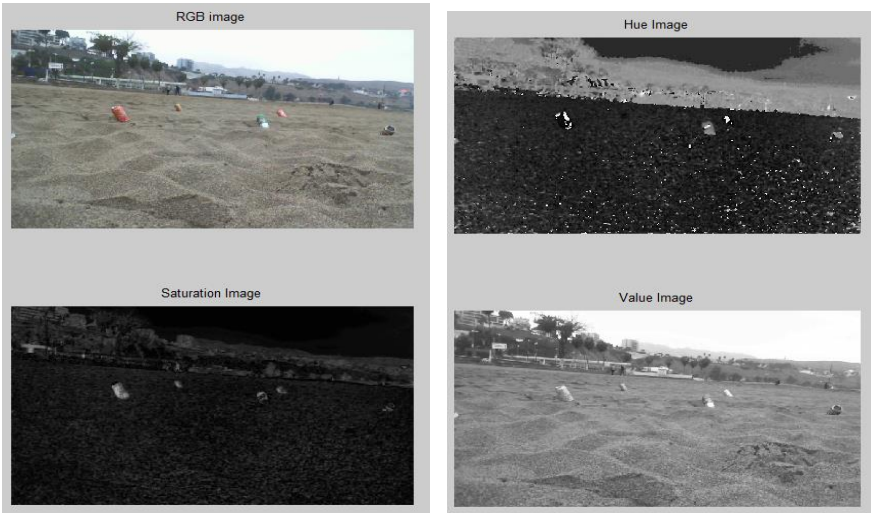


Figura: 38 Imagen Original, seguidamente Imagen Hue, Imagen Saturation e Imagen Value de imagen RGB original

Fuente: Elaboración propia

Seguidamente se empieza a procesar con cada una de estas imágenes HSV y dimos que la mejor respuesta se obtuvo con el uso de la imagen de Saturación ya que se aprecia mejor una diferencia en cuanto al fondo y los objetos a identificar, siendo más sencilla la identificación de los mismos, por la cual la se empieza a usar durante todo el Procesamiento de la imagen. Ver Figura: 38

3.2.2 Corte de la Imagen

Para disminuir el tiempo de procesamiento de la imagen se decide reducir la imagen recortándola aproximadamente en su cuarta parte verticalmente de la imagen original, eliminando partes innecesarias al momento de procesar la imagen y haciendo más rápido su procesamiento. Ver Figura: 39

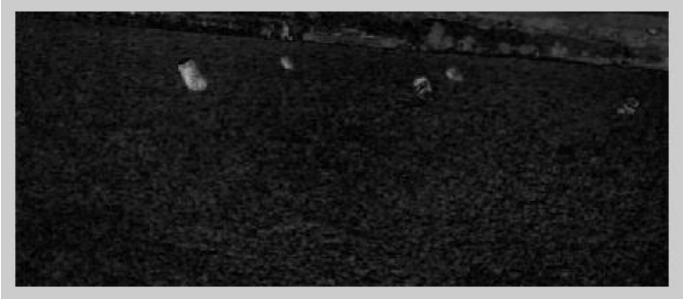


Figura: 39 Imagen Saturación Cortada
Fuente: Elaboración propia
En el Anexo 9 muestra el un algoritmo de corte de imagen que se implementó.

3.2.3 Filtrado

Cuando se hizo la etapa de post procesamiento se notó que la imagen tenía mucho ruido del medio en sí, por lo que fue necesaria la implementación de filtros.

En este caso se realizó el algoritmo de filtrado (con el barrido de máscaras), y luego se le evaluó con filtros de Suavizado, mediana y Pasa bajo. Ver Figura: 40

Para la implementación del algoritmo de filtrado vemos el código fuente mostrado en el Anexo 10

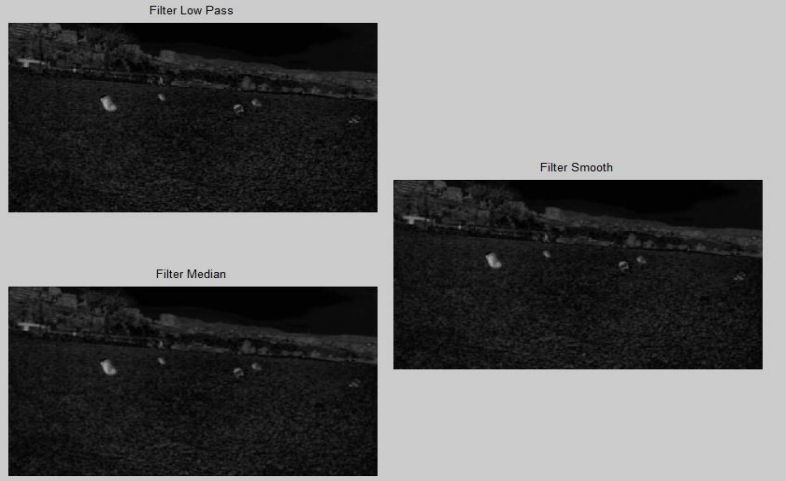


Figura: 40 Resultado de aplicar filtros Pasa Bajo, Media, y Suavizado

Fuente: Elaboración propia

Para el filtrado se usó diferentes tipos de máscaras para los cuales se obtuvo mejores resultados usando las siguientes: Ver Figura: 41

$$\begin{bmatrix} 0 & 0.1 & 0 \\ 0.1 & 0.6 & 0.1 \\ 0 & 0.1 & 0 \end{bmatrix} \qquad 1/16 * \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \qquad 1/9 * \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

LowPass F. Smooth Median

Figura: 41 Filtros usados para procesamiento de imagen.

Fuente: Elaboración propia

Luego, de hacer este filtrado en el dominio del espacio, (Ver Figura: 42) pasamos a hacer lo mismo, pero en el dominio de la frecuencia. Se implementó filtros de Butteworth y Gaussianos de diferentes ventanas, obteniéndose mejores resultados con el filtro Gaussiano expresado en la ecuación. (3.1):

$$g(i, j) = ce^{-\frac{i^2 + j^2}{2\sigma^2}} \qquad (3.\ 12)$$

donde c es una constante normalizada, medias tanto $i_0\ j_0 = 0$ y término denominador dependiente de la desviación(σ) $\rightarrow 2\sigma^2 = 1.2$, ventana n=15x15

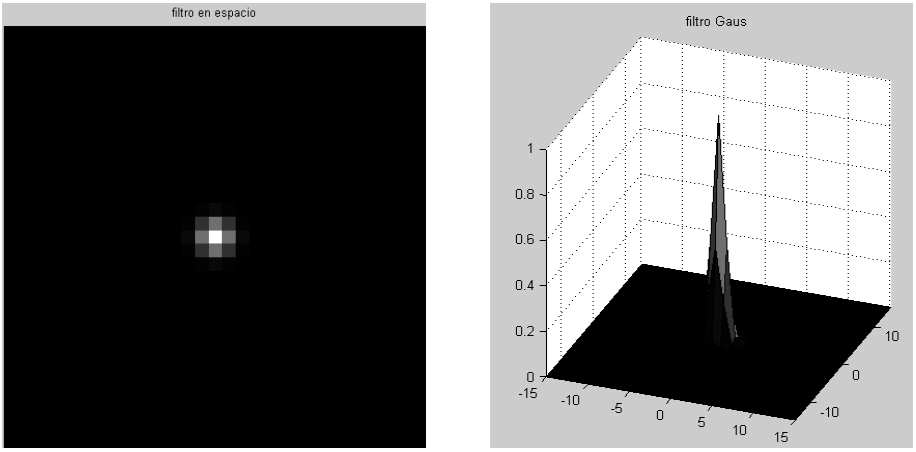


Figura: 42 Filtro Gaussiano Espacial

Fuente: Elaboración propia

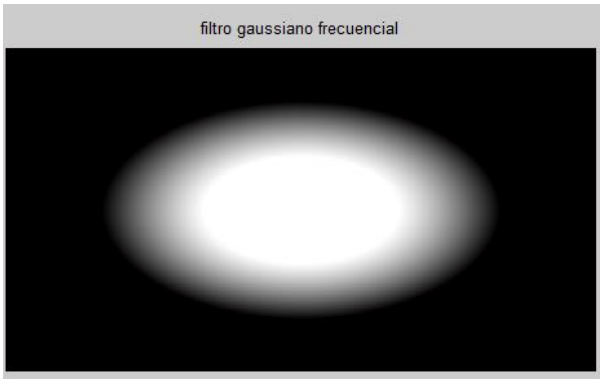


Figura: 43 Representacion de Filtro Gausiano Frecuencial

Fuente: Elaboración propia

En 2D, las funciones Gaussianas son rotacionalmente Simétricos. Esto quiere decir que, la cantidad de suavizado ejecutado por el filtro será la misma en todas sus direcciones. [11] En general, los bordes en una imagen no se verán orientados hacia una dirección en particular.

Además, el criterio principal para hacer uso de este filtrado en frecuencia fue que un estrecho Gaussiano en el dominio espacial, tiene un espectro más ancho (Espectro en frecuencia). Esto quiere decir que hay menos Smoothing (suavizado), y en el dominio

frecuencial tiene un mayor ancho de banda y pasan más altas frecuencias (Ancho Espectro). [12]

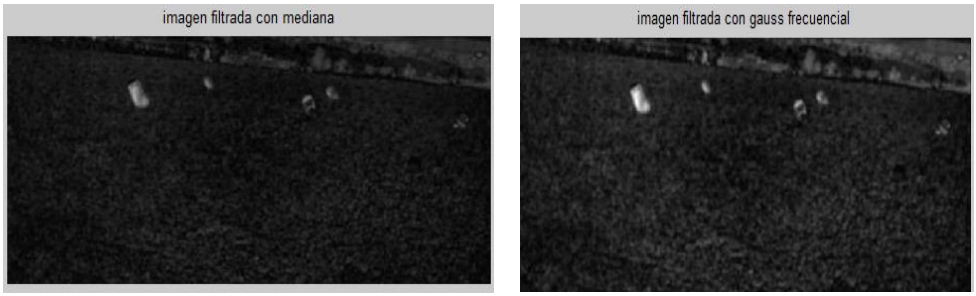
El cálculo de la transformada de Fourier del filtro Gaussiano se realizó en un capítulo anterior expresado en la ecuación (2.26).

En la sección 2.2.4 se menciona la manera del cómo hacer este filtrado en frecuencia, el cual consistía primero en hallar la Transformada de Fourier (TF) de la Imagen, luego una vez seleccionado el filtro, debemos pasar este filtro a las mismas dimensiones de la Imagen por la cual necesitaría un algoritmo de rellenado (Padding Algorithm), para después hallar la TF de filtro rellenado con ceros. Una vez obtenido los 2 se convolucionan (en frecuencia se multiplican punto a punto los pixels) para luego con la Transformada de Fourier Rápida Inversa volver a obtener la Imagen original. Ver Figura: 43

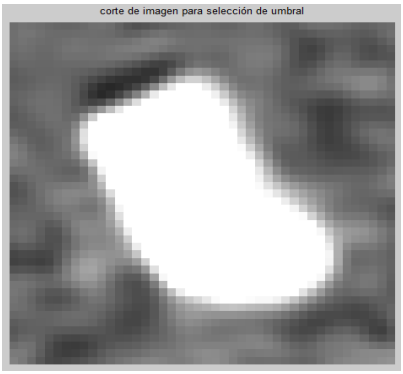
En el Anexo 11 se desarrolló una función para obtener este filtro gaussiano primero en el dominio del espacio.

En el Anexo 12 se desarrolló una función para obtener este filtro gaussiano en el dominio de la frecuencia.

Finalmente en el Anexo 13 se desarrolló una función el cual hace la convolución en frecuencia (multiplica pixel por pixel) del Fourier de la Imagen junto con el Fourier de nuestro filtro gaussiano el cual tiene una ventana en nuestro caso de $n = 15x15$



(a) (b)



(c)

Figura: 44 (a) imagen filtrada con filtro mediana en el espacio (b) Imagen filtrada con filtro Gaussiano en la frecuencia.ds (c) Acercamiento de lata de Figura 36 (b)
Fuente: Elaboración propia

En la Figura: 44 (a) (b) y (c) podemos apreciar una gran mejora en nuestro sistema de procesamiento de imagen, ya que bajo el concepto previamente dicho, el Gauss en su espectro tiene un ancho de banda muchísimo mayor, por lo que deja pasar muchas más altas frecuencias, teniendo de esta manera mejor apreciación de las latas para su identificación.

Luego de esto seguimos con el proceso de umbralización el cual nos permitirá segmentar la imagen para la identificación del objeto respectiva.

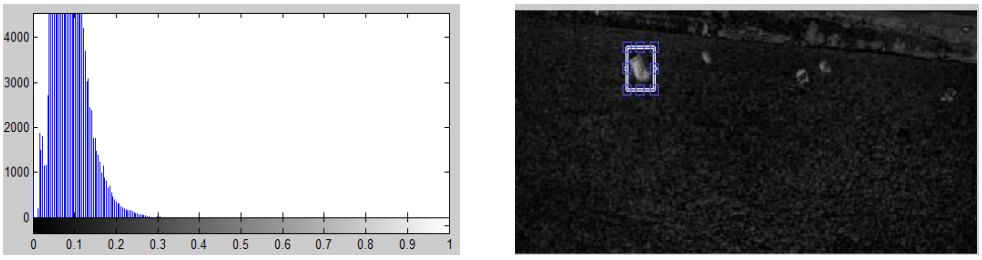
3.2.4 Umbralización

La umbralización es uno de los más importantes métodos de segmentación. El objetivo es convertir una imagen en escala de grises a una nueva con sólo dos niveles, de manera que los objetos queden separados del fondo.

Esto se llevó a cabo mediante el análisis del histograma de la imagen HSV previamente adquirida, cortada y filtrada.

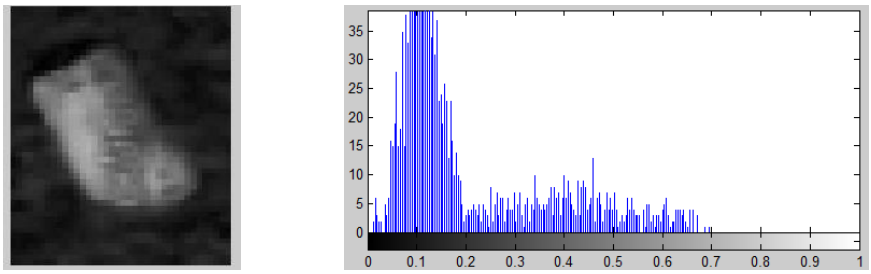
Lo que se hizo fue al tener la imagen HSV se leyó su histograma, pero ésta no mostraba fácilmente la información de la lata, por lo que se cortó la imagen manualmente por donde estaba exactamente la lata, para otra vez realizar el análisis de su histograma y poder obtener el valor óptimo para la umbralización como se muestra en la Figura: 45

Este proceso se hizo con todas las imágenes obteniendo al final un umbral óptimo que funciona para la mayoría de los casos.



(a) (b)

Figura: 45 (a) histograma de imagen recortada (b) Imagen Recortada seleccionando sólo objeto lata para su posterior análisis.
Fuente: Elaboración propia



(a) (b)

Figura: 46 (a) Lata seleccionada manualmente de la Figura: 45 (b) Histograma de lata seleccionada manualmente

Fuente: Elaboración propia

En la Figura: 46 (a), (b) se aprecia que el valor óptimo para seleccionar y diferenciar la lata del fondo, arena, etc. es de 0.19 aproximadamente. Sin embargo al evaluar éste valor con todas las imágenes no dieron buenos resultados por lo que se decidió mejor optar por Seleccionar el umbral de 0.21, porque también nos elimina ruido del medio. Con esto lo que hacemos es poner todos los valores menores a 0.21 a 0 que es negro y el resto a 1 que es blanco, binarizando así la imagen.

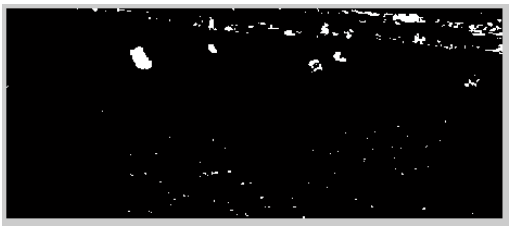
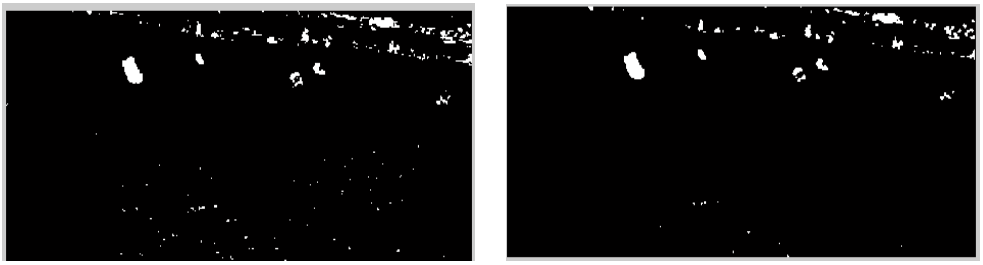


Figura: 47 Imagen binarizada con Umbral=0.21

Fuente: Elaboración propia

El algoritmo de Binarización bajo un Umbral (Threshold) está implementado en el Anexo 14

Una vez seleccionado el umbral en un paso anterior a la Binarización lo que se hace es filtrar la imagen con el filtro de Mediana como el mostrado en la Figura: 44 (a), ya que éste mostró mejor respuesta. Luego se procede a hacer la binarización como lo hicimos anteriormente y obtendremos mejores resultamos como los mostrados en la Figura: 48 (b)

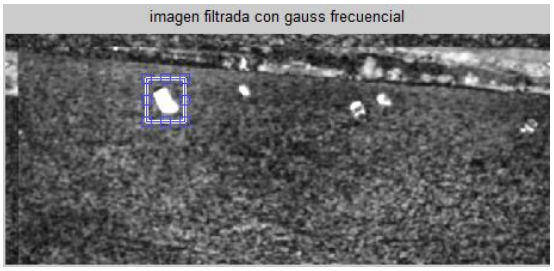


(a) (b)

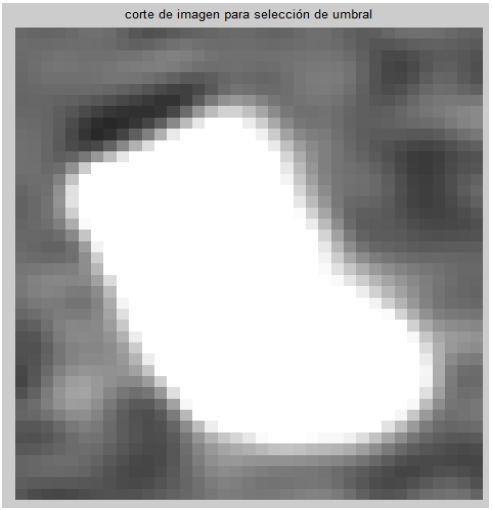
Figura: 48 (a) Muestra binarización sin filtro de Mediana. (b) Binarización con filtro Mediana

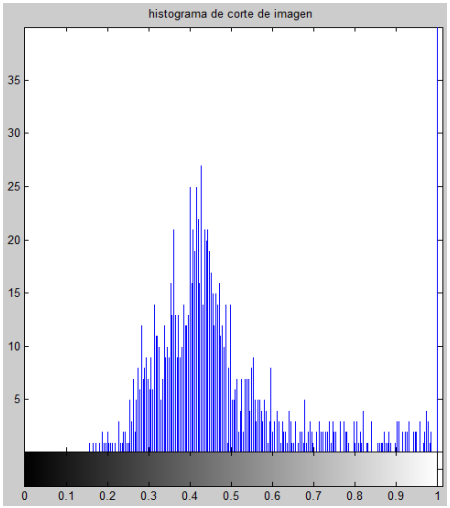
Fuente: Elaboración propia

Ahora, a diferencia de cómo se mostró anteriormente el proceso de umbralización pero con el filtro espacial de media, hacemos este mismo proceso pero con el filtro frecuencial Gaussiano.



(a)





(b) (c)

Figura: 49 (a) Imagen Recortada seleccionando sólo objeto lata para su posterior análisis. (b) Lata seleccionada manualmente de la Figura 41 (b). (c) Histograma de lata seleccionada manualmente
Fuente: Elaboración propia

Vemos que una selección óptima de nuestro umbral (Ver Figura: 49) según el histograma mostrado en la figura anterior sería de 0.85 aproximadamente. Sin embargo después de pruebas se decidió mejor optar por el Threshold de 0.9 para nuestro caso, ya que en otras imágenes también se ven muy buenos resultados. Con esto lo que hacemos es poner todos los valores menores a 0.9 a 0 que es negro (Fondo) y el resto a 1 que es blanco (Objeto), binarizando así la imagen.

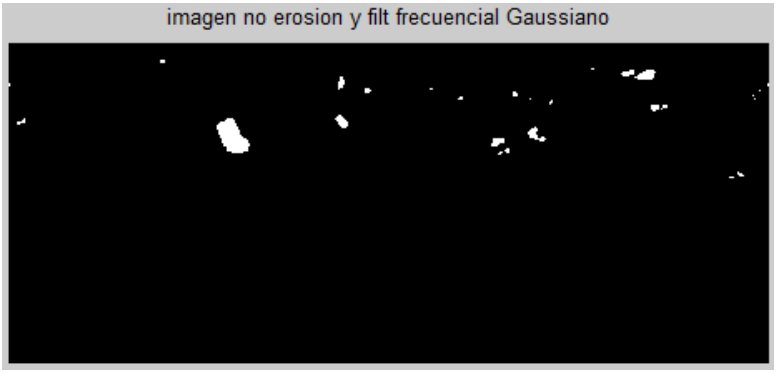


Figura: 50 Imagen binarizada con umbral 0.9 y filtro frecuencial Gaussiano
Fuente: Elaboración propia

Comparando la Figura: 50 con la Figura: 48(b), vemos una gran mejora implementando un filtro frecuencial que sólo con el uso de un filtro espacial de mediana, obteniendo así menor ruido en mi imagen procesada de salida.

3.2.5 Erosión

Una vez hecho el filtrado en el dominio Espacial y frecuencial de la imagen, como aún se tiene ruido en la imagen se optó por aplicar la operación morfológica de la Erosión, la cual permite en este caso difuminar el ruido de fondo el cual no se desea en la imagen.

La erosión está dada por la siguiente operación morfológica:

$$A \ominus B = \{x \mid B_x \subseteq A\} \qquad (3.\ 13)$$

Se probaron varios Elementos Estructurales (EE), cuadrados regulares y no regulares, por lo que se optó con mejores resultados el siguiente:

$$EE = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

obteniendo así una respuesta mostrado en la Figura: 51

Figura: 51 Imagen Erosionada de Figura: 50 (b)

Fuente: Elaboración propia

Al ver el resultado de la imagen después de erosionarla, vemos que aún tiene ruido el cual debemos filtrarlo.

Luego se realizó el mismo proceso pero con la imagen umbralizada después de aplicarle el filtro frecuencial Gaussiano en la Figura: 52

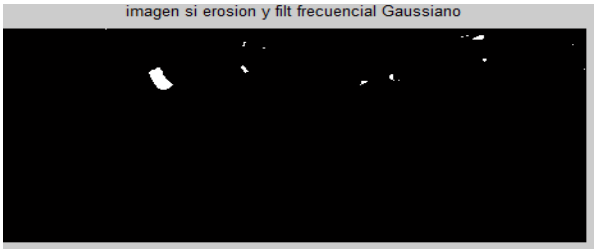
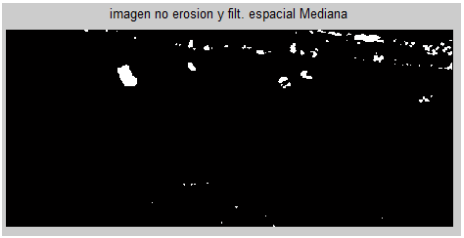
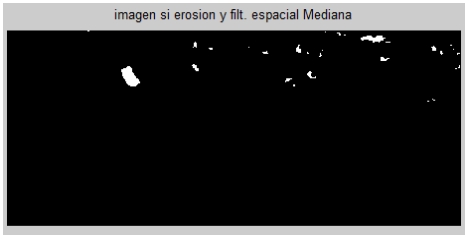


Figura: 52 Imagen erosionada con filtro frecuencial Gaussiano

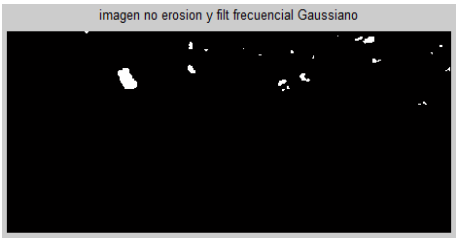
Fuente: Elaboración propia



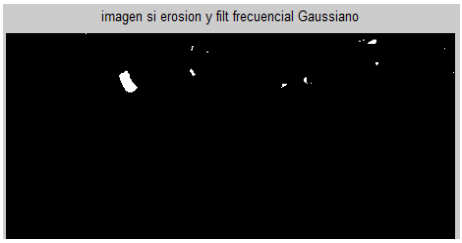
(a)



(b)



(c)



(d)

Figura: 53 Aquí vemos la mejora de respuesta añadiendo filtrado en frecuencia Gaussiana. (a) Imagen sin erosión y filtrado espacial Mediana. (b) Imagen erosionada con filtro espacial Mediana. (c) Imagen sin erosión con filtro frecuencial Gaussiano. (d) Imagen erosionada con filtro Frecuencial Gaussiano.

Fuente: Elaboración propia

En la Figura: 53, se aprecia la mejora de respuesta añadiendo filtrado en frecuencia Gaussiana, sin embargo, es posible notar que aún necesitamos éstas imágenes filtrarlas ya que el ruido no se ha podido difuminar completamente. Se aplicó entonces según la teoría de la erosión, y la dilatación tantas veces como se aplicó la erosión con el fin de mantener por procesamiento la forma del objeto.

Se aplicó el mismo proceso a otras imágenes conjuntas tomadas, pero a la larga se notó que esto me consumía mayor tiempo computacional ya que si erosionaba 5 veces, entonces 5 veces dilataba, y hacer esta cantidad de barridos en la imagen, hablando en procesamiento no es muy bueno computacionalmente, por lo que se optó de manera mucho más sencilla, eliminar este proceso morfológico cíclico hasta difuminar el ruido, y estimar la cantidad de pixels del objeto a identificar, para luego aplicar un filtro tamaño, y eliminar completamente el ruido obtenido.

Para esto, tuve primero que etiquetar la imagen con ruido obtenida para saber qué objeto estoy leyendo, y estimar la cantidad de pixels de dicho objeto.

3.2.6 Etiquetado

Como se mencionó anteriormente, luego de aplicar la morfología, había probado con aplicar la erosión nuevamente, seguido de la dilatación para mantener la forma del objeto tantas veces sean necesarias para que la imagen me salga casi sin ruido, sin embargo la aplicación de la dilatación y erosión varias veces es bastante tiempo computacional por lo que primero etiqueté la imagen para después aplicar un filtro de tamaño, y con esto llegué a obtener resultados óptimos con mi imagen.

En este proceso de etiquetado lo que se hizo fue hacer un barrido a la imagen desde el punto superior izquierdo, cuando vea un objeto entonces a éste lo registro con un número.

Estos objetos registrados con un número en particular son registrados de tal forma que, el primer objeto etiquetado con '1', pertenece al que está ubicado más en la parte izquierda superior, ya que es respecto al barrido el cual hace la lectura, luego la etiqueta sube su valor al siguiente '2', y el segundo objeto que detecte, lo etiqueta con este valor y así sucesivamente hasta etiquetar de objetos toda la imagen.

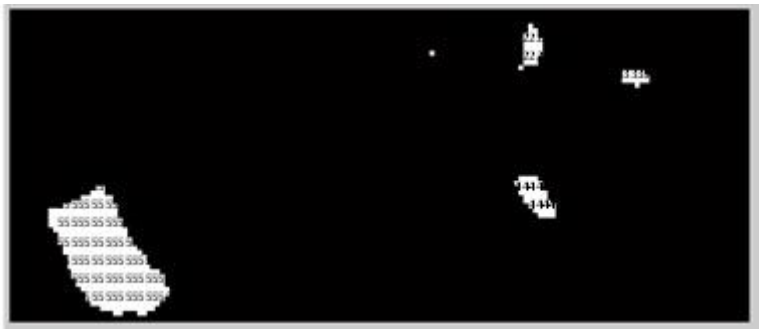


Figura: 54 Imagen etiquetada
Fuente: Elaboración propia

Como se mostró en la Figura: 54 se hizo un acercamiento a la Figura: 43para mostrar el proceso de etiquetado. Aquí se observan los pixels pertenecientes a los objetos que han sido etiquetados con un número, sin embargo se nota que realmente hay un objeto de interés en la imagen mostrada y los demás es simple ruido de la imagen aún después de filtrada que no pudo eliminarse después de hacer la morfología. Por tanto, me fue necesario aplicar un filtro de tamaño para eliminar estos ruidos, y ahorrar tiempo computacional que erosionando y dilatando la imagen X veces.

3.2.7 Filtro Tamaño

Con el uso de la morfología no es suficiente para eliminar el ruido completamente de la imagen, ya que se puede tener algunos bulbos de pixels blancos que no pertenecen al objeto, por lo que se es útil saber la cantidad aproximada de pixels que tiene el objeto, de tal manera que todos los objetos que tengan mucho menor o mayor tamaño que el que se fije, se deberán filtrar.

En la Figura: 55 observamos que efectivamente con este algoritmo, sólo nos queda el objeto de interés luego de aplicar el filtro de tamaño.

Figura: 55 Resultado de aplicar filtro de tamaño.
Fuente: Elaboración propia

En el Anexo 15 se muestra el programa que se encarga de hacer este filtro de tamaño, anulando por completo el ruido restante de las anteriores etapas del procesamiento de la imagen ya procesadas.

3.3 Procesamiento de la Imagen

3.3.1 Representación y Descripción

La representación y descripción de la imagen, que han sido previamente segmentados y/o pre-procesados son los primero pasos que se deben realizar para el análisis de imágenes.

Como al mismo título refiere, primero respecto a la Representación de la imagen, el objetivo es elegir a dicha representación que facilite la manipulación de los objetos que deseemos procesar en nuestro ordenador. Ahora, en cuanto a los Descriptores, se trata de encontrar en los objetos ciertos rasgos que son diferencias esenciales entre los objetos que pertenecen a diferentes clases (ya sea un triángulo, cuadrado, etc.), y que éstas diferencias sean lo más independientemente posible ante cambios como escalamientos, traslaciones y rotaciones.

Un punto muy importante a recalcar es que una representación no nos dice si el objeto es un círculo, cuadrado. Donde se define esa representación es cuando se describe dicha representación.

3.3.1.1 Diseño de representación de los Objetos

Lo más importante que hay que tener en cuenta en la selección de una representación es saber que, se elige una representación externa cuando el objetivo principal se centra en las características de forma del objeto; una representación interna se escoge cuando el principal interés está relacionado con las propiedades de reflexión de la superficie del objeto tales como color y textura.

A continuación se nombra los pasos a seguir para la representación del objeto a identificar:

A. Código de Cadena

Ahora, como lo que interesa son las características de forma del objeto se opta por escoger un esquema de representación externa como el Código de Cadena.

Para realizar dicha representación se realizó un código de cadena de 8 direcciones como la mostrada en la Figura: 56 donde 0 indica la derecha, 1 es la diagonal derecha hacia arriba, 2 es arriba y así sucesivamente hasta que se tenga una completa descripción del contorno de la imagen a describir. Esta es una secuencia conectada de segmentos con dirección y longitud específica, entre puntos sucesivos de los límites de un objeto, describiendo de esta manera su forma. Con el algoritmo de conectividad 8 se obtiene de manera más aproximada la descripción de la forma del objeto incluyendo un trazo en diagonal o en las direcciones 1, 3, 5, 7. Como se muestra en la Figura: 56

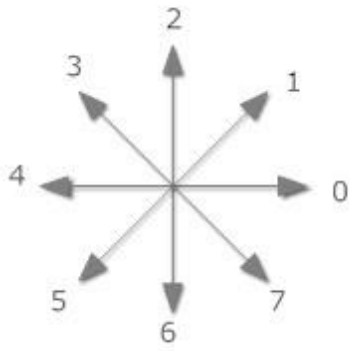


Figura: 56 Código de Cadena de 8 Direcciones
Fuente: Elaboración propia

Una vez que se tiene bien en claro las direcciones del código de cadena, otro asunto bien importante es definir un vector de direcciones los cuales se va a desplazar el punto de referencia ya sea en sentido horario o antihorario.

Para realizar dicho algoritmo de Cadena, hay que tomar ciertas consideraciones, primero tenemos que definir si el barrido que le vamos a hacer a la figura será en sentido horario o anti-horario, en sentido contrario a las manecillas del reloj.[21]

Anteriormente habíamos filtrado la imagen de tal manera que la obteníamos libre de ruido, por lo que nos quedaba una imagen binaria. Ahora nuestro objetivo es esa imagen dejar sólo sus bordes para después obtener una descripción de estos bordes. El algoritmo que se realizó fue el siguiente:

1. Encuentro el primer punto del objeto de la imagen binaria.
2. Defino en qué sentido haré el barrido al objeto a procesar. Se escogió el sentido horario respecto al punto de referencia a barrer.
3. Como el primer punto encontrado es barrido desde la esquina superior izquierda hasta dicho punto entonces, no habrán puntos que pertenezcan al objeto en las direcciones 2,3 y así sucesivamente para ese sentido.
4. Como se escogió sentido horario entonces el primer punto a barrer sería el de la dirección 1. Si existe objeto entonces se le considera en la imagen =1, si no, se continúa con la siguiente dirección en sentido horario (dirección 0).
5. Una vez encontrado el punto se desplaza el punto de referencia respecto al vector de direcciones como especifica la Figura: 57
6. Una vez mi punto de referencia se ubicó en el nuevo punto del objeto, éste empieza a escanear si existe objeto 90° menos desde su última dirección que se desplazó, para hacerlo en el sentido horario el cual se desea, y así sucesivamente hasta terminar de escanear completamente el objeto o hasta que su posición final sea igual a su posición inicial.

f (-1,-1)	f (-1, 0)	f (-1, 1)
f (0,-1)	f (0 , 0)	f (0, 1)
f (1,-1)	f (1, 0)	f (1, 1)

Figura: 57 Vector de Direcciones

Fuente: Elaboración propia

A continuación se muestra el contorno de la imagen procesada en la Figura: 58

Figura: 58 contorno de la Imagen procesada

Fuente: Elaboración propia

Debido a que el código de cadena es una secuencia de direcciones, entonces podemos tener varias secuencias de direcciones, por ejemplo:

Sea el código de cadena $[1 \quad 3 \quad 2]$, entonces podemos tener derivados de este código de cadena otras secuencias de direcciones como: $[3 \quad 2 \quad 1]$ y $[2 \quad 1 \quad 3]$.

Se hizo la programación del código de cadena tal y como se indica en el Anexo 16 y el Anexo 17 la cual es una sub función del Anexo 16

B. Descriptor de Fourier

Como se vio anteriormente la descripción de una imagen por códigos de cadena sólo es invariante ante traslaciones, sin embargo no es invariante ante rotaciones o cambios de escala que es lo que se requiere para obtener una mejor descripción por lo que ahora se opta por usar un mejor descriptor como lo es el descriptor de Fourier.

Para implementar esto, se asume que se tiene la imagen solo y únicamente con sus bordes respectivos la cual se desea describir.

Cada punto del borde de la imagen está representada por un punto en el eje "x"y un punto en el eje "y", es decir, la imagen está descrita en dos dimensiones $I(x,y)$, entonces para implementar el descriptor se pasa de tener un problema en dos dimensiones a uno de una dimensión.[12]

La manera de hacer es colocar las coordenadas de cada punto $x(t), y(t)$ para $t = 1, 2, 3, ..., N-1$ en una función compleja sea $f(t) = x(t) + iy(t)$ cuya parte real está representada por la posición en el eje x y la parte imaginaria está representada por la posición del punto del borde en el eje y.

De forma general, cada par de coordenadas se puede tratar como un número complejo de la forma en la ecuación (3.5):

$$f(t) = x(t) + iy(t), t = 0,1,2,\ldots,N-1 \tag{3.14}$$

De esta manera se convierte de un problema de dos dimensiones en uno de una dimensión, obteniendo así un vector para representar dicha descripción del objeto.

Ahora, se puede representar ésta función con la transformada de Fourier unidimensional, ya que es de una Dimensión, aplicando la transformada discreta de Fourier la cual viene dada por la siguiente expresión mostrada en la ecuación (3.15):

$$F(u) = \frac{1}{N}\sum_{n=0}^{N-1} f(n) e^{-j2\Pi\left(\frac{nu}{N}\right)} \tag{3.15}$$

Los coeficientes complejos $F(u)$ se les conocen como "Descriptores de Fourier" del contorno de la imagen.

[13] Los coeficientes de $F(u)$ presentan variaciones muy pequeñas para formas con contornos suaves, si los contornos tornan una forma más abrupta las variaciones de estos descriptores van a ser mayores dando como resultado una mayor frecuencia.

La transformada Inversa de Fourier permite otra vez restaurar completamente el contorno $f(n)$ la cual vienen dada por la expresión:

$$f(n) = \frac{1}{N}\sum_{n=0}^{N-1} F(u) e^{j2\Pi\left(\frac{nu}{N}\right)} \tag{3.16}$$

Se usó esta descripción de Fourier debido a que representa propiedades interesantes.

3.4 Redes neuronales

Para el diseño primero se planteó el modelo de la red Neuronal Básica como se muestra en la Figura: 59 Modelamiento de Red Neuronal Unicapa

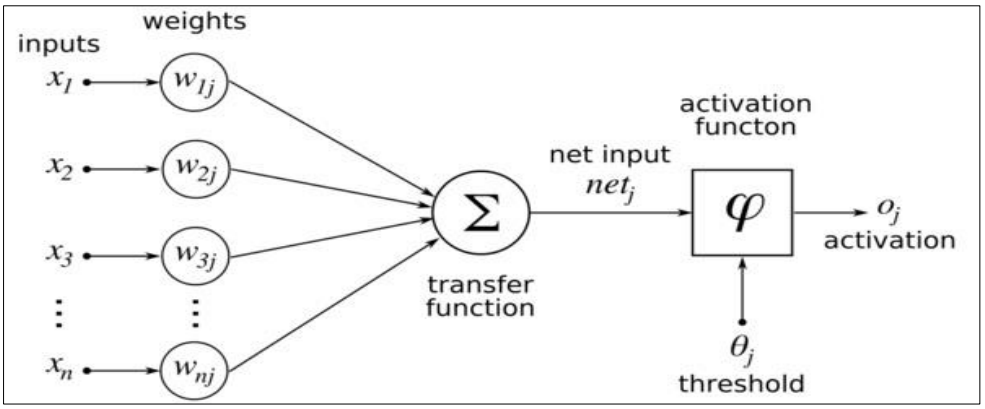


Figura: 59 Modelamiento de Red Neuronal Unicapa

Fuente: [14] Garrick D. (2013) Principles of Artificial Neural Networks. University of Illinois. USA

En la figura anterior se puede apreciar que las variables $x1, x2, \ldots x_n$, éstos vendrían a ser las señales de entrada o los patrones de entrada, normalmente expresados en un vector columna (traspuesta), o en este caso, los valores del descriptor de Fourier que se tiene de la imagen. Luego están los pesos sinápticos, los cuales son los pesos de interconexión que varían gradualmente. Estos pesos pasan a una función de transferencia, que es una simple sumatoria de los patrones de entrada por sus pesos sinápticos, y seguidamente pasa a una función de activación, que es el agente principal en la red Neuronal.USA

[15]

La función de activación determina lo que es y lo que no es objeto de la identificación al momento de hacer el procesamiento, a través de un umbral es la que da la salida de la neurona.[17]

Ésta arquitectura simple de la red Neuronal mostrada es de una red unicapa, las cuales hay de diferentes tipos para su aprendizaje, como son la red Neuronal Perceptrón, Hopfield, Kohonen, etc. Sin embargo según investigaciones referenciadas en el capítulo II, muestran claramente que un sistema más óptimo es usar Redes Neuronales multicapa.

En éste proyecto de tesis, se usó red Neuronal Multicapa debido a que los pesos sinápticos se pueden ajustar de manera más precisa, en base a un SET de entrenamiento (patrones de entrada) que ingreso a mi red, y minimiza el error cuadrático medio en base a todas las salidas y éste set de entrenamiento ingresado

La red multicapa escogida es la red BACKPROPAGATION, ya que presenta propiedades óptimas como las antes mencionadas para la identificación del objeto a procesar.

3.4.1 Redes Neuronales Multicapa

A. Para el diseño de la red neuronal, una vez definido el número de entradas y salidas definimos el tipo de red neuronal que se va a usar.
B. Se escogió red multicapa BackPropagation 5-4-3, donde N es el número de entradas que se coloca en el vector, o el número de características que definen la imagen a reconocer, 5 es el número de enlaces que se usará en la capa oculta y M es el número de enlaces de la capa de salida.
C. Una vez seleccionada la red Neuronal a usar, se procedió a seguir con el esquema de la Figura: 60
D. Se tiene una muestra de 20 imágenes las cuales serán los diferentes patrones del vector de entrada. El vector se normalizó para 5 características más relevantes (vector invariante ante rotaciones, traslaciones, escalamientos, y perímetro y área con respecto al algoritmo código de cadena elaborado anteriormente) que definen la lata X_n
E. Ahora se pasa a definir los pesos sinápticos de mis capas de entrada, oculta y salida.

Según la red definida en la Figura: 60 se tiene lo siguiente:

- Definiremos $\hat{W}_{nj}$ a los pesos de la capa de entrada.
  - $\#W_{weigths} = \#X_n * \#\sum Xn$
  - Donde "$\sum Xn$"es el número de enlaces de mis pesos de la capa de entrada=4.

- Según la red multicapa Backpropagation mostrada en la Figura: 60 tenemos que: $j = 0,1,2..5$ & $\#\sum_{Xn} = 4$
- Por lo que se tiene: $\#W_{weigths} = 6*4 = 24$

- La cantidad de pesos en la capa oculta está definida por:
  - $\#C_{weights} = \#h_l * \#\sum_{hm}$
  - Donde "$\sum_{hm} = 3$,"es el número de enlaces de mis pesos de la capa de oculta=3
  - Según la red multicapa Backpropagation mostrada en la Figura: 60 tenemos que:
  - $l = 0,1,2,3,4$ & $\#\sum_{hm} = 3$
  - Por lo que se tiene: $\#C_{weights} = 5*3 = 15$
  - Seguidamente el algoritmo automáticamente ajusta los pesos respecto a la gradiente del error.

- El vector de Salida $\hat{Y}_q$ es igual al número enlaces de la capa de Salida = 3 Definida por $\hat{Y} = [Y_1\ Y_2\ Y_3]$ (Salidas de la capa oculta). Entonces nuestra arquitectura de la Red Neuronal sería como la siguiente figura a continuación:

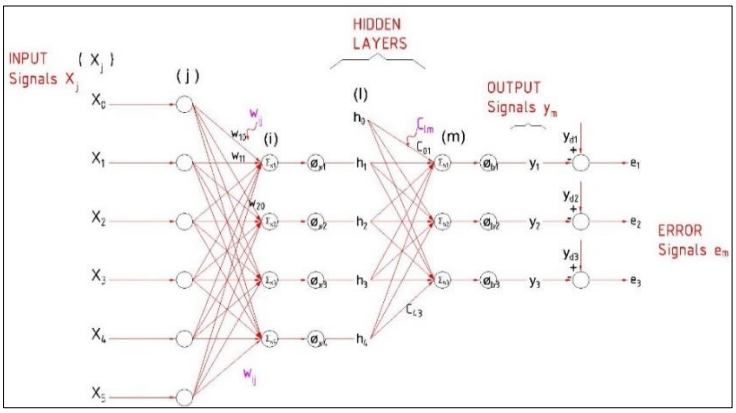


Figura: 60 Arquitectura de Red Neuronal Multicapa usada (5-4-3)

Fuente: Elaboración propia

Como se explicó, se escogió un set de Entrenamiento de 20 imágenes la cual se clasificará entre si es o no una lata. Mi vector de Salida Y1, Y2, Y3 definen los objetos los cuales de desean identificar.

Se escogió 3 salidas Y1, y2, Y3 para poder realizar la clasificación de múltiples clases (o como lo mencionan en la literatura como "Multi-Classification Problem")

Antes se definen algunas variables en referencia a la Figura: 60:

$\hat{X}_j$: es vector de características, donde n es el número de características que definen a mi patrón de entrada.

$\hat{P}_m$: es vector del patrón a identificar.

$\hat{Y}_q^{(i)}$: es vector de Salida donde q es el número de Salidas e i es el número de Iteración que hace la Red Neuronal

$\hat{Y}_{dq}^{(i)}$: es vector de Salida deseado donde q es el número de Salidas e "i" es el número de Iteración que hace la Red Neuronal.

$\hat{W}_{ij}$: es vector de pesos sinápticos de mi primera capa.

$\hat{C}_{nl}$: es vector de pesos sinápticos de mi capa oculta.

e_q: es el error en la salida "q" y se define por

$$e_q = y_{dq} - y_q$$

Ahora se pasa resolver la Red:

Una vez calculado el número de pesos de la capa de entrada y capa oculta se pasa a definir los valores de los mismos los cuales entra a hacer el aprendizaje la neurona.

Según la red escogida los pesos sinápticos vendrían a ser los siguientes:

$$\hat{W}_{ij} = [W_{10}\ W_{11}\ W_{12} \ldots W_{15}\ W_{20}\ W_{21}\ W_{22} \ldots W_{25}\ W_{30}\ W_{31}\ W_{32} \ldots W_{35}\ W_{40}\ W_{41}\ W_{42} \ldots W_{45}]^T \quad (3.17)$$

$$\hat{C}_{ln} = [C_{01}\ C_{11}\ C_{21}\ C_{31}\ C_{41}\ C_{02}\ C_{12}\ C_{22}\ C_{32}\ C_{42}\ C_{03}\ C_{13}\ C_{23}\ C_{33}\ C_{43}]^T \quad (3.18)$$

Seguidamente, se elabora tabla en la cual se extraerán los datos para realizar algoritmos de Red Neuronal como se muestra en la tabla a continuación:

Tabla: 3 Vector de características y vector de salida deseado

P_m	$x_1^{(k)}$	$x_2^{(k)}$	$x_3^{(k)}$	$x_4^{(k)}$		$x_5^{(k)}$	$y_{d1}^{(k)}$	$y_{d2}^{(k)}$	$y_{d3}^{(k)}$
P_1	$x_1^{(1)}$	$x_2^{(1)}$	$x_3^{(1)}$	$x_4^{(1)}$		$x_5^{(1)}$	$y_{d1}^{(1)}$	$y_{d2}^{(m)}$	$y_{d3}^{(1)}$
P_2	$x_1^{(2)}$	$x_2^{(2)}$	$x_3^{(2)}$	$x_4^{(2)}$		$x_5^{(2)}$	$y_{d1}^{(2)}$	$y_{d2}^{(m)}$	$y_{d3}^{(2)}$
P_3	$x_1^{(3)}$	$x_2^{(3)}$	$x_3^{(3)}$	$x_4^{(3)}$		$x_5^{(3)}$	$y_{d1}^{(3)}$	$y_{d2}^{(m)}$	$y_{d3}^{(3)}$
...	...	...	...	...		...	...	...	...
P_{18}	$x_1^{(18)}$	$x_2^{(18)}$	$x_3^{(18)}$	$x_4^{(18)}$		$x_5^{(18)}$	$y_{d1}^{(18)}$	$y_{d2}^{(m)}$	$y_{d3}^{(18)}$
P_{19}	$x_1^{(19)}$	$x_2^{(19)}$	$x_3^{(19)}$	$x_4^{(19)}$		$x_5^{(19)}$	$y_{d1}^{(19)}$	$y_{d2}^{(m)}$	$y_{d3}^{(19)}$
P_{20}	$x_1^{(20)}$	$x_2^{(20)}$	$x_3^{(20)}$	$x_4^{(20)}$		$x_5^{(20)}$	$y_{d1}^{(20)}$	$y_{d2}^{(m)}$	$y_{d3}^{(20)}$

Fuente: Elaboración propia

donde:

$\overrightarrow{P_m}$: Set de entrenamiento

"m": Número de patrón a usar

"k": Es el número de iteración que depende del entrenamiento de la red Neuronal respecto a la actualización de sus pesos.

"y_{dq}": Es el valor deseado que se desea obtener según dice el método de aprendizaje supervisado (Supervised Learning).

Según el vector de entrada y mi vector de salida se procede a hacer el aprendizaje de la neurona de la siguiente manera:

Sea el primer patrón $\hat{P}_1$ v $\vec{P}_1$ el cual está comprendido por vector de características $\overrightarrow{X_j} = [x_0\ x_1\ x_2\ x_3\ x_4\ x_5]$, este vector será vector de entrada de la neurona.

Luego de realiza algoritmo como sigue a continuación tomando la red de la Figura: 60.

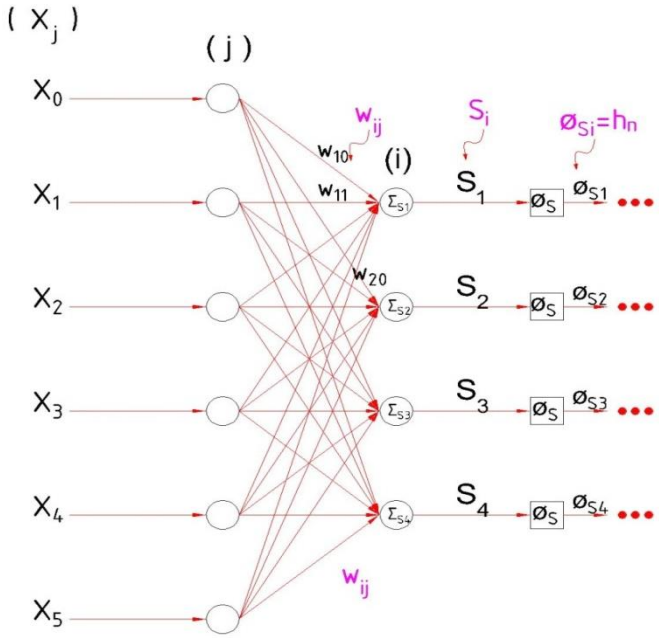


Figura: 61 Input-hidden layer schema
Fuente: Elaboración propia

Según red se realiza siguiente algoritmo:

$$S_i = \sum_{k=0}^{j=5} X_j . W_{ij} \qquad (3.\ 19)$$

Donde:

$$X_0 = 1 \qquad (3.\ 20)$$

$$j \in \square \subset X[0-5] \qquad (3.\ 21)$$

$$i \in \square \subset W[1-4] \qquad (3.\ 22)$$

Cálculo del algoritmo realizado en el Anexo 18

Como se mencionó anteriormente se escogió como función de activación "tangente hiperbólica", y su cálculo se realizó bajo el siguiente algoritmo:

$$h_n = \phi(S_i) = tangh(S_i) \qquad (3.\ 23)$$

Donde:

$$tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \qquad (3.\ 24)$$

Entonces se tiene:

$$h_1 = tanh(S_1) \qquad (3.\ 25)$$

$$h_2 = tanh(S_2) \qquad (3.\ 26)$$

$$h_3 = tanh(S_3) \qquad (3.\ 27)$$

$$h_4 = tanh(S_4) \qquad (3.\ 28)$$

Luego se hace mismo algoritmo para capa oculta como se muestra a continuación para obtener salidas.

Se tiene vector $\overrightarrow{h_n} = [h_0\ h_1\ h_2\ h_3\ h_4]$

Donde

$$h_0 = 1$$

Se tiene vector de pesos sinápticos $\overrightarrow{C_{nl}}$ los cuales se definen según la arquitectura de la Red como se muestra en la Figura: 62

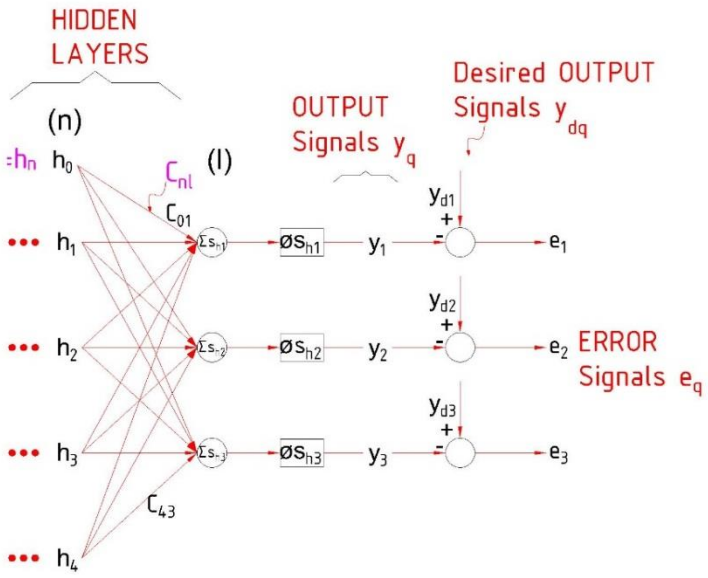


Figura: 62 Output layer schema

Fuente: Elaboración propia

Se aplica algoritmo siguiente:

$$S_{hl} = \sum_{k=0}^{n=4} h_n . C_{nl} \qquad (3.\ 29)$$

Donde:

$$n \in \square \subset h[0-4]$$

$$l \in \square \subset C[1-3] \qquad (3.\ 30)$$

$$h_0 = 1$$

Cálculo del algoritmo realizado en el Anexo 19

Como se mencionó anteriormente se escogió como función de activación "tangente hiperbólica", y su cálculo se realizó bajo el siguiente algoritmo:

$$y_q = \phi(S_{hl}) = tangh(S_{hl}) \quad (3.\ 31)$$

Donde:

$$tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad (3.\ 32)$$

Entonces tenemos:

$$y_1 = tanh(S_{h1}) \quad (3.\ 33)$$

$$y_2 = tanh(S_{h2}) \quad (3.\ 34)$$

$$y_3 = tanh(S_{h3}) \quad (3.\ 35)$$

Seguidamente se calcula el error en base a la tabla obtenida bajo el siguiente algoritmo:

$$e_n^{(q)} = y_{dq}^{(n)} - y_{dq}^{(n)} \quad (3.\ 36)$$

n: numero de iteración

Se tiene:

$$e_1^{(1)} = y_{d1}^{(1)} - y_1^{(1)} \quad (3.\ 37)$$

$$e_2^{(1)} = y_{d2}^{(1)} - y_2^{(1)} \quad (3.\ 38)$$

$$e_3^{(1)} = y_{d3}^{(1)} - y_3^{(1)} \quad (3.\ 39)$$

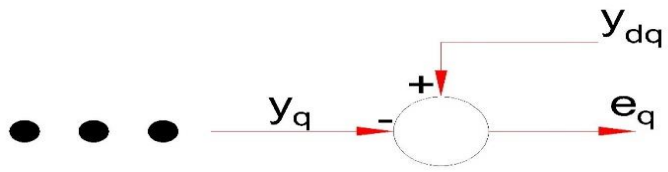


Figura: 63 Representation of Neural Net Error algorithm

Fuente: Elaboración propia

Una vez calculado los Errores se calculó el valor espontáneo de la energía del error (Ver Figura: 63) (el error cuadrático medio) bajo el siguiente algoritmo:

Error Energy:

$$E(n) = \frac{1}{2}\sum_{l=1}^{Q} e_q^2(n) \qquad (3.\ 40)$$

Donde Q pertenece al número de neuronas en la capa de salida.

Se prosiguió con el mismo procedimiento para todos los patrones

Los cálculos se muestran en el Anexo 20

Seguidamente se haya el promedio del error cuadrático, con esto podemos apreciar el coste del algoritmo neuronal, siendo como objetivo minimizar este valor actualizando los pesos. Se realizó el siguiente artificio para hallar el promedio:

$$E_{av}(n) = \frac{1}{N}\sum_{n=1}^{N} E(n) \quad ; \qquad (3.\ 41)$$

Siendo n el # de Iteración y N el # de patrones de nuestra Red. Esto se aplica una vez terminado de realizar algoritmo Neuronal hacia adelante con todos los patrones de entrada según la tabla elaborada.

$$E_{av}(1) = \frac{1}{20}[E(1) + E(2) + E(3) + E(4) + E(5) + E(6) + E(7) + E(8)$$

$$+E(9)+E(2)+E(10)+E(11)+E(12)+E(13)+E(14)+E(15)+E(16)$$

$$+E(17)+E(18)+E(19)+E(20)]$$

Después de esto se aplica algoritmo BackPropagation.

Como su mismo nombre lo dice, es un algoritmo de retropropagación inversa. Como se hizo primero, se computa estos errores en la capa de salida y los propagamos hacia atrás esos errores a la capa oculta anterior, de tal que se tiene una heurística para el error que está en la capa oculta. Entonces se tiene una fracción de los errores de la Salida y luego se propagan estos hacia atrás a las capas ocultas y luego, una vez se tenga las suposiciones de los errores en la capa oculta, se obtiene la regla de actualización para estos pesos de la capa subsiguiente de la entrada como se mostrará a continuación.

Para la retropropagación se empieza de los términos de errores hallados, se calculas las gradientes locales de las capas previas de la salida, luego propagamos estas gradientes una capa anterior, y así subsecuentemente hasta que se llegue a la capa anterior a la entrada, y así se obtiene al final todos los pesos ajustados.[14]

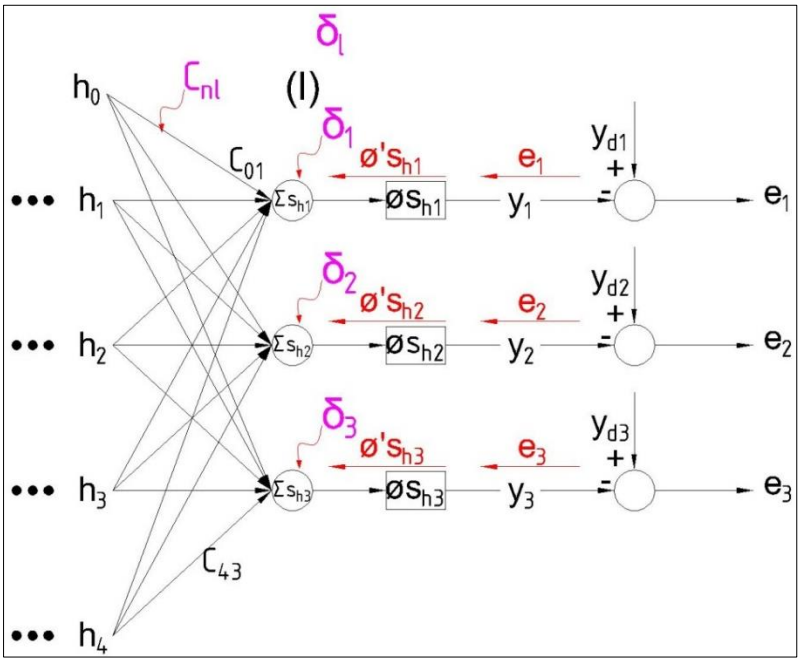


Figura: 64 Detalle, inicio de algoritmo Back Propagation
Fuente: Elaboración propia

En la Figura: 64 anterior se muestra como empieza el algoritmo BackPropagation

Se tiene:

$$\delta_q = e_q \,.\, \phi'_q s_{hl} \qquad (3.\ 42)$$

Entonces:

$$\delta_1 = e_1 \,.\, \phi' s_{h1} \qquad (3.\ 43)$$

$$\delta_2 = e_2 \,.\, \phi' s_{h2} \qquad (3.\ 44)$$

$$\delta_3 = e_3 \,.\, \phi' s_{h3} \qquad (3.\ 45)$$

Donde δ_q de la capa anterior a la capa de salida

Seguidamente se prosigue con la capa anterior.

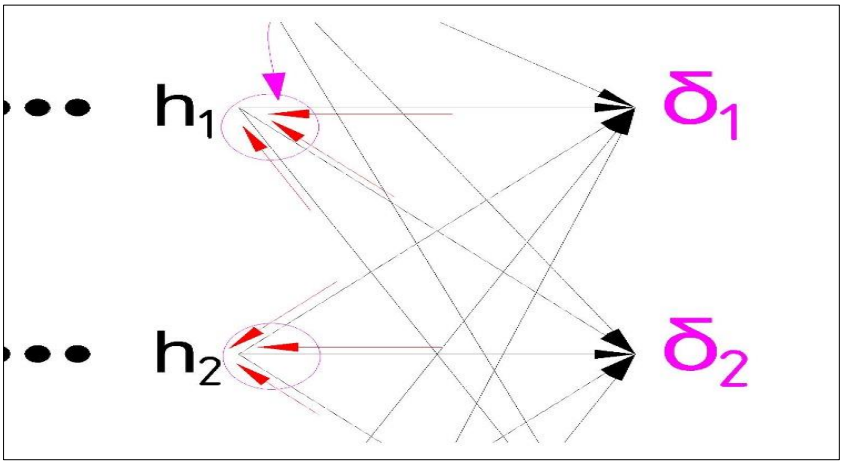


Figura: 65 Detalle, Propagación del error, algoritmo BackPropagation

Fuente: Elaboración propia

En la Figura: 65 muestra la forma como se prosiguió a hacer la propagación del error hacia la capa anterior a detalle.

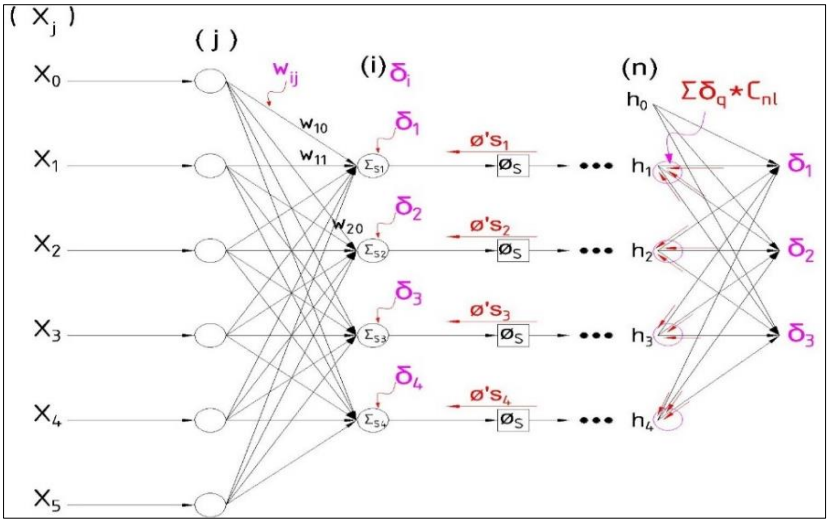


Figura: 66 Detalle, algoritmo BackPropagation, general.

Fuente: Elaboración propia

En la Figura: 66 se muestra de manera más completa el algoritmo backpropagation.

Como se puede apreciar al propagarlo hacia atrás el error, se toma una fracción de éste hacia la capa anterior.

Según muestra la Figura anterior, haciendo el algoritmo hacia atrás, el valor que se obtiene en los puntos h1, h2…, que viene del algoritmo FORWARD de la red que es nada más que la función de activación el campo inducido S_i está dado por la siguiente expresión:

$\sum_{i=1}^{N} \delta_q \cdot C_{nl}$; Donde N es el número de neuronas de mi capa anterior a la capa de salida.

Esto es debido a que en este punto convergen no sólo el error propagado directamente de la salida, como lo fue con la capa de salida que es en un solo punto, si no que en este punto se obtiene el error propagado en fracciones iguales hacia las capas de entrada, en las cuales como se muestran en la

figura convergen más de una red respecto a las capas de salida. Es por esto que no es solo 1 si no la sumatoria de estos con sus respectivos pesos sinápticos de la capa de salida. Se realizó este análisis y se representó de forma gráfica en la Figura: 67

Seguidamente como se hizo anteriormente se toma la derivada de la función de activación obteniendo de esta manera la Gradiente local de la capa de entrada δ_i.

$$\delta_i = \phi'(S_i) \cdot \sum_{i=1}^{N} \delta_q \cdot C_{nl} \qquad (3.\ 46)$$

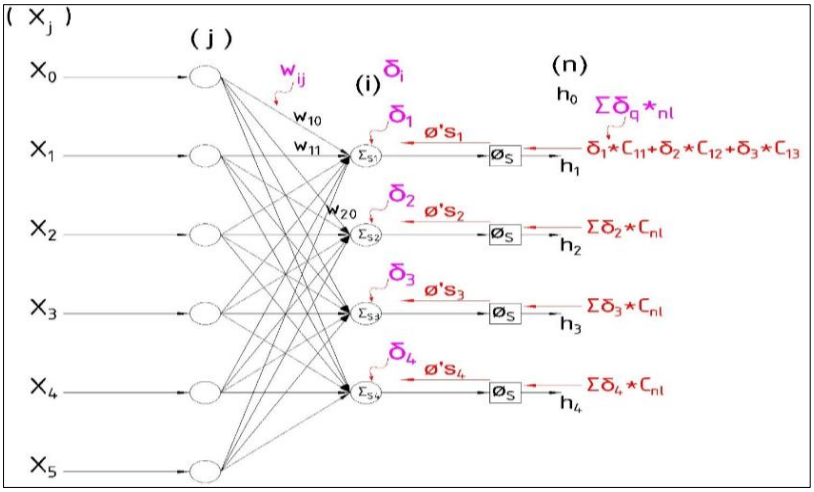


Figura: 67 Detalle, Algoritmo Back propagation, actualizaciones de pesos, y diferenciaon de gradientes locales (Back Step)
Fuente: Elaboración propia

En la figura anterior se elaboró representación más general del algoritmo BackPropagation. Donde la gradiente local δ_1 que viene de capa de salida δ_q, es diferente a δ_1 que viene de capa de entrada δ_i.

Para objetivos de cálculo se diferenció esto con δ_{1in} & δ_1 para gradiente local de la capa oculta y capa de salida respectivamente

Los cálculos se realizaron en el Anexo 21

Los cálculos descritos en esta tesis solo se pusieron para una primera iteración, el resto es proceso cíclico de la red.

Al final se tiene lo siguiente:

a) Del paso Forward

$W_{ij}(n)$, $C_{li}(n)$ pesos sinápticos de la capa de entrada y la capa oculta respectivamente en la iteración "n".

$$S_i(n) = \sum_{j=0}^{k} W_{ij}(n) * X_j(n) \qquad (3.47)$$

Donde k es el número de entradas = 5

$$\phi_i(S_i(n)) = tangh(S_i(n)) = y_i(n) \qquad (3.48)$$

Además

$$tangh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \qquad (3.49)$$

$$S_{hl}(n) = \sum_{h=0}^{k} C_{nl}(n) * y_i(n) \qquad (3.50)$$

Donde k es el número de neuronas de la capa de entrada=4, $C_{li}(n)$ son los pesos sinápticos de la capa oculta en la iteración "n", $y_i(n)$ es la salida o la función de activación resultante de la neurona de la capa de entrada "i"

Seguidamente se hace función de activación como en la capa anterior:

$$\phi_q(S_{hl}(n)) = tangh(S_{hl}(n)) = y_q(n) \qquad (3.51)$$

Se calculó el error según:

$$e_q = y_{dq} - y_q \tag{3. 52}$$

Se obtuvo el Error Energy:

$$E(n) = \frac{1}{2}\sum_{l=1}^{Q} e_q^2(n) \tag{3. 53}$$

Además gradientes locales de capa de salida δ_q y de capa oculta δ_i en el paso de la retropropagación del error hacia los pesos sinápticos del algoritmo Backpropagation:

$$\delta_q(n) = e_q(n) \,.\, \phi_q' s_{hl}(n) \tag{3. 54}$$

$$\delta_i(n) = \phi'_i(S_i(n)) \,.\, \sum_{i=1}^{N} \delta_q(n) \,.\, C_{nl}(n) \tag{3. 55}$$

Para la actualización de pesos se hizo lo siguiente:

La gradiente, poniendo un ejemplo, es la variación de una función respecto a una variable. De ésta manera tenemos en backpropagation gradientes locales.

Viendo de manera más detallada, la corrección o la variación de los pesos sinápticos (por ex. W_{ij}) puede escribirse como lo muestra en la ecuación (3. 58) porque ésta va a ser proporcional a la gradiente del Error respecto al Peso sináptico $\frac{\partial E(n)}{\partial W_{ij}(n)}$. Matemáticamente se expresa en la ecuación (3.40):

$\Delta W_{ij}(n)\ aplicado\ a\ W_{ij}$

$$\Delta W_{ij}(n) = -\eta \frac{\partial E(n)}{\partial W_{ij}(n)} \tag{3. 56}$$

El negativo es porque tiene que ser en sentido opuesta a la dirección de la gradiente.

De la misma manera en la ecuación (3. 57):

$$\Delta C_{nl}(n) = -\eta \frac{\partial E(n)}{\partial C_{nl}(n)} \qquad (3.\ 57)$$

Siendo η el factor o taza de aprendizaje. Entre más grande sea η más rápido aprenderá la neurona. Sin embargo hay que tener cuidado que, si consideramos η muy grande, puede que éste no converja a un Local mínimo para que pueda llegar a hacer el aprendizaje y la Red nunca aprenderá.

Después se aplicó la regla de aprendizaje una vez obtenido la corrección de los pesos sinápticos. Matemáticamente se expresó en la ecuación (3. 58):

$$W_X(n+1) = W_X(n) + \Delta W_X(n) \qquad (3.\ 58)$$

Dónde:

$W_X(n+1)$: es el peso sináptico de la siguiente iteración Actualizada

Ahora para hallar $\frac{\partial E(n)}{\partial W_{ij}(n)}$ se usó regla de la cadena.

Según nuestra arquitectura de Red Backpropagation se tiene la ecuación (3. 59):

$$\frac{\partial E(n)}{\partial W_{ij}(n)} = \frac{\partial E(n)}{\partial e_q(n)} \cdot \frac{\partial e_q(n)}{\partial y_q(n)} \cdot \frac{\partial y_q(n)}{\partial S_{hl}(n)} \cdot \frac{\partial S_{hl}(n)}{\partial y_i(n)} \cdot \frac{\partial y_i(n)}{\partial S_i(n)} \cdot \frac{\partial S_i(n)}{\partial W_{ij}(n)} \qquad (3.\ 59)$$

Del mismo modo en la ecuación (3. 60):

$$\frac{\partial E(n)}{\partial C_{nl}(n)} = \frac{\partial E(n)}{\partial e_q(n)} \cdot \frac{\partial e_q(n)}{\partial y_q(n)} \cdot \frac{\partial y_q(n)}{\partial S_{hl}(n)} \cdot \frac{\partial S_{hl}(n)}{\partial C_{nl}(n)} \qquad (3.\ 60)$$

Primero se halló la variación, del error respecto al peso sináptico de la capa de salida $\Delta C_{nl}(n)$

De la ecuación (3. 53) se tiene la ecuación (3. 61):

$$\frac{\partial E(n)}{\partial e_q(n)} = e_q(n) \tag{3. 61}$$

De la misma manera de ecuaciones (3. 52)(3. 51)(3. 50) se obtiene:

$$\frac{\partial e_q(n)}{\partial y_q(n)} = -1 \tag{3. 62}$$

$$\frac{\partial y_q(n)}{\partial S_{hl}(n)} = \phi'_q(S_{hl}(n)) \tag{3. 63}$$

$$\frac{\partial S_{hl}(n)}{\partial C_{nl}(n)} = y_i(n) \tag{3. 64}$$

Al final, de (3. 60)(3. 61)(3. 62)(3. 63)(3. 64) se obtuvo:

$$\frac{\partial E(n)}{\partial C_{nl}(n)} = -e_q(n).\phi'_q(S_{hl}(n)).y_i(n) \tag{3. 65}$$

Además, haciendo (3. 54) en (3. 65) se obtuvo:

$$\frac{\partial E(n)}{\partial C_{nl}(n)} = -\delta_q(n).y_i(n) \tag{3. 66}$$

Haciendo (3. 66) en (3. 57)

$$\Delta C_{nl} = \eta.\delta_q(n).y_i(n) \tag{3. 67}$$

Además aplicando regla de actualización de pesos en capa de salida según (3. 58) se obtuvo:

$$C_{nl}(n+1) = \Delta C_{nl}(n) + C_{nl}(n) \tag{3. 68}$$

$$C_{nl}(n+1) = \eta.\delta_q(n).y_i(n) + C_{nl}(n) \tag{3. 69}$$

La ecuación (3. 69) se usó como algoritmo de actualización de los pesos sinápticos en la capa de salida, sin embargo todavía se tuvo la incógnita de la gradiente local respecto al error $\delta_q(n)$ por lo que se calculó de la siguiente manera:

$$\delta_q(n) = e_q(n).\phi'_q(S_{hl}(n)) \tag{3. 70}$$

Se sabe que la función de Activación escogida es la tangente hiperbólica por sus propiedades de simetría, a continuación se aprecia que también se escogió la tangente hiperbólica por su simplicidad matemática al efectuar su gradiente, derivada respecto en la actualización de los pesos sinápticos.

Se tiene:

$$\phi(x) = tangh(x) \tag{3. 71}$$

$$tangh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{3. 72}$$

Recordando:

$$e^{f(x)'} = f'(x).e^{f(x)} \tag{3. 73}$$

$$f(x) = \frac{u}{v} \rightarrow f'(x) = \frac{u'.v - u.v'}{v^2} \tag{3. 74}$$

Entonces:

$$\phi'(x) = \frac{(e^x + e^{-x}).(e^x + e^{-x}) - (e^x - e^{-x}).(e^x - e^{-x})}{(e^x + e^{-x})^2} \tag{3. 75}$$

$$\phi'(x) = \frac{(e^x + e^{-x})^2 - (e^x - e^{-x})^2}{(e^x + e^{-x})^2} \tag{3. 76}$$

$$\phi'(x) = \frac{(e^x + e^{-x})^2}{(e^x + e^{-x})^2} - \frac{(e^x - e^{-x})^2}{(e^x + e^{-x})^2} \qquad (3.\ 77)$$

De (3. 69) se tiene:

$$\phi'(x) = 1 - \phi^2(x) \qquad (3.78)$$

$$C_{nl}(n+1) = \eta.e_q(n).\phi'_q(S_{hl}(n)).y_i(n) + C_{nl}(n) \qquad (3.79)$$

Finalmente aplicando (3.78) en (3.79)se obtuvo algoritmo de Actualización de pesos de la capa de salida expresado a continuación en (3. 81):

$$C_{nl}(n+1) = \eta.e_q(n).(1 - \phi_q^2(S_{hl}(n))).y_i(n) + C_{nl}(n) \qquad (3.\ 80)$$

Lo mismo se aplicó para la actualización de pesos sinápticos de la capa anterior como se muestra a continuación:

$$\frac{\partial E(n)}{\partial W_{ij}(n)} = \frac{\partial E(n)}{\partial e_q(n)} \cdot \frac{\partial e_q(n)}{\partial y_q(n)} \cdot \frac{\partial y_q(n)}{\partial S_{hl}(n)} \cdot \frac{\partial S_{hl}(n)}{\partial y_i(n)} \cdot \frac{\partial y_i(n)}{\partial S_i(n)} \cdot \frac{\partial S_i(n)}{\partial W_{ij}(n)} \qquad (3.\ 81)$$

$$\frac{\partial E(n)}{\partial W_{ij}(n)} = \frac{\partial E(n)}{\partial y_i(n)} \cdot \frac{\partial y_i(n)}{\partial S_i(n)} \cdot \frac{\partial S_i(n)}{\partial W_{ij}(n)} \qquad (3.\ 82)$$

Con el mismo concepto se calculó primero $\frac{\partial E(n)}{\partial y_i(n)}$

$$\frac{\partial E(n)}{\partial y_i(n)} = \frac{\partial E(n)}{\partial e_q(n)} \cdot \frac{\partial e_q(n)}{\partial y_q(n)} \cdot \frac{\partial y_q(n)}{\partial S_{hl}(n)} \cdot \frac{\partial S_{hl}(n)}{y_i(n)} \qquad (3.\ 83)$$

De esta ecuación la única incógnita que se tiene es $\frac{\partial S_{hl}(n)}{\partial y_i(n)}$ la cual se calculó de (3. 50) teniendo como resultado:

$$\frac{\partial S_{hl}(n)}{\partial y_i(n)} = C_{nl}(n) \qquad (3.\ 84)$$

En primera instancia simplemente se concluyó según (3. 61)(3. 62)(3. 63)(3. 84) lo siguiente:

$$\frac{\partial E(n)}{\partial y_i(n)} = e_q(n).(-1).\phi'_q(S_{hl}(n)).C_{nl}(n) \qquad (3.\ 85)$$

$$\frac{\partial E(n)}{\partial y_i(n)} = -\delta_q(n).C_{nl}(n) \qquad (3.\ 86)$$

$$\frac{\partial E(n)}{\partial y_i(n)} = -e_q(n).(1-\phi_q^2(S_{hl}(n))).C_{nl}(n) \qquad (3.\ 87)$$

Sin embargo el algoritmo, dio como resultado que la Red nunca aprendió, en éste punto aplicando la propagación inversa, no sólo converge la fracción del error de la primera salida, sino que también convergen la fracción del error respecto a las otras salidas, por lo que en este punto "*i*" no solo se tiene un simple término, sino que es una Sumatoria de todas estas fracciones del error respecto a sus salidas y a sus pesos sinápticos de la iteración anterior.

Debido a este criterio se modificó el algoritmo bajo la siguiente premisa:

$$\frac{\partial E(n)}{\partial y_i(n)} = -\sum_{q=1}^{k=3} \delta_q(n).C_{nl}(n) \qquad (3.\ 88)$$

Donde k es el número de salidas de la red=3

Seguidamente se sabe que:

$$y_i(n) = \phi_i(S_i(n)) \qquad (3.\ 89)$$

$$S_i(n) = \sum_{j=0}^{k} W_{ij}(n) * X_j(n) \qquad (3.\ 90)$$

Por lo que se tiene:

$$\frac{\partial y_i(n)}{\partial S_i(n)} = \phi_i'(S_i(n)) \qquad (3.\ 91)$$

$$\frac{\partial S_i(n)}{\partial W_{ij}(n)} = X_j(n) \qquad (3.\ 92)$$

De (3. 88)(3. 91) y (3. 92) en (3. 86) se obtuvo:

$$\frac{\partial E(n)}{\partial W_{ij}(n)} = (-1).(\sum_{q=1}^{k=3} \delta_q(n).C_{nl}(n)).(\phi_i'(S_i(n))).X_j(n) \qquad (3.\ 93)$$

Luego para la actualización de los pesos de la capa se aplicó algoritmo de la ecuación (3. 56) como sigue:

$$\Delta W_{ij}(n) = -\eta \frac{\partial E(n)}{\partial W_{ij}(n)} \qquad (3.\ 94)$$

$$\Delta W_{ij}(n) = -\eta.(-1).(\sum_{q=1}^{k=3} \delta_q(n).C_{nl}(n)).(\phi_i'(S_i(n))).X_j(n) \qquad (3.\ 95)$$

$$\Delta W_{ij}(n) = \eta.(\sum_{q=1}^{k=3} \delta_q(n).C_{nl}(n)).(\phi_i'(S_i(n))).X_j(n) \qquad (3.\ 96)$$

De (3. 55) se obtuvo finalmente algoritmo de corrección de los pesos sinápticos:

$$\Delta W_{ij}(n) = \eta.\delta_i.X_j(n) \qquad (3.\ 97)$$

Finalmente, el algoritmo de actualización de los pesos anterior a la capa de salida se calculó como se expresó en (3. 58)

$$W_{ij}(n+1) = W_{ij}(n) + \Delta W_{ij}(n)$$

$$W_{ij}(n+1) = W_{ij}(n) + \eta.\delta_i.X_j(n) \qquad (3.\ 98)$$

Donde:

$$\delta_i = \phi_i'(S_i(n)).\sum_{i=1}^{N} \delta_q(n).C_{nl}(n) \qquad (3.\ 99)$$

Seguidamente, una vez hecho el aprendizaje, la máquina detecta eficientemente los objetos, y logra identificar de manera precisa si es o no una lata bajo el procesamiento de señales, definición de características, filtros, identificadores de Fourier, código cadena, redes neuronales. Ver Figura: 68

Sin embargo, faltaría algo más que aplicar como objetivo final para este proyecto de tesis y es la identificación la distancia la cual se encuentra el objeto identificado el cual se explicaría la manera como se hizo en el siguiente capítulo.

cc =

Columns 1 through 23

1 0 1 0 0 1 1 0 1 0 0 7 7 6 7 6 6 7 6 7 6 7 6

Columns 24 through 46

6 7 7 0 7 7 7 7 6 7 5 6 5 4 5 4

5 4 4 4 4 4 4

Columns 47 through 69

3 4 4 3 3 2 3 3 3 2 3 2 2 3 2 2 3 2 3 2 3 2 2

Figura: 68 Resultado de algoritmo de Código de cadena de una lata

Fuente: Elaboración propia

CAPÍTULO 4:

DISEÑO DEL SISTEMA DE PERCEPCIÓN DE PROFUNDIDAD

4.1 Estereoscopía

Para objetivos de esta tesis se analizará la parte geométrica y la manera como llegar a la visión de profundidad bajo éste concepto de estereoscopía, sin embargo no se llegará a tocar muy a fondo éste tema ya que implica un estudio mucho más detallado, que puede ser objeto de otro trabajo de investigación sólo dedicado a la Estereoscopía.

4.2 Calibración de cámara

Como primer paso se tiene que determinar los valores extrínsecos e intrínsecos de la cámara (valores externos e internos).

Extrínsecos: Refiere a la locación y orientación de la cámara.

Intrínsecos: Se refiere al lente focal de la cámara, pixeles, etc.

En cuanto refiere a la calibración de la cámara hay otro concepto el cual se tomó en cuenta, el cual es la Estimación de la posición (Pose Estimation).

4.2.1 Pose Estimation

Se tiene un modelo de objeto 3D y se quiere proyectar ese modelo en el plano de la imagen (Proyección es por ejemplo el determinar la locación y la orientación (traslación/Rotación) del objeto tal que luego, este, proyectado en el plano de la imagen encajará en la imagen)

Entonces, la estimación de Pose es básicamente dado un objeto 3D, se extraen diferentes imágenes (3 cámaras, 3 figuras) de ese objeto, tal que estas son proyectadas, pero éstas se encajan, las juntamos y/o las asociamos de tal manera que se obtenga un modelo 3D del objeto basado en figuras, el cómo éste es rotado y trasladado. [5]

Por lo tanto, puesto que se tiene un objeto 3d, y se tiene imágenes 2D, también se necesita saber cuál sería la distancia focal del lente de la cámara y el cuál sería la escala de los objetos a enfocar para que el objeto basado en imágenes parezca real en 3D (el objeto en 3D basado en sólo imágenes y figuras), pero si sólo se toma el objeto de 2 imágenes, y se pega (como 3D), no parecerá real, porque no tendrá una perspectiva de efecto Real, debido a que se tiene un mundo 3D y se está tomando imágenes 2D. Entonces, ¿Cómo se torna real esto? Para hacer en sí esto, se necesita hablar de las TRANSFORMACIONES BÁSICAS y el cómo se relaciona un sistema de coordenadas respecto a otro sistema de coordenadas.

De aquí se deriva la teoría de la Robótica y sus matrices de transformación Homogéneas respecto a los movimientos que se hace, a la articulación y/o junta que la hace, y el ángulo, posición del que varían un sistema respecto a otro, solo que aquí se usa las cámaras, su posición y desfase ubicados uno respecto de otro, y bajo este concepto de visión Artificial, aparece la matriz de escalabilidad.

También se verá la manera de calcular el eje Focal de la cámara y finalmente se verá la manera como se procedió a calcula la distancia final del objeto respecto a las cámaras en el eje Z en el mundo 3D.

En resumen, hay muchos sistemas de coordenadas en la calibración de la cámara tales como:

- Coordenadas de la escena

- Coordenadas de la cámara
- Coordenadas de la imagen
- Coordenadas de pixeles.

Determinar las relaciones entre estos sistemas de coordenadas es el propósito y objetivo para poder llegar a la estereoscopía propuesta para este proyecto de tesis.

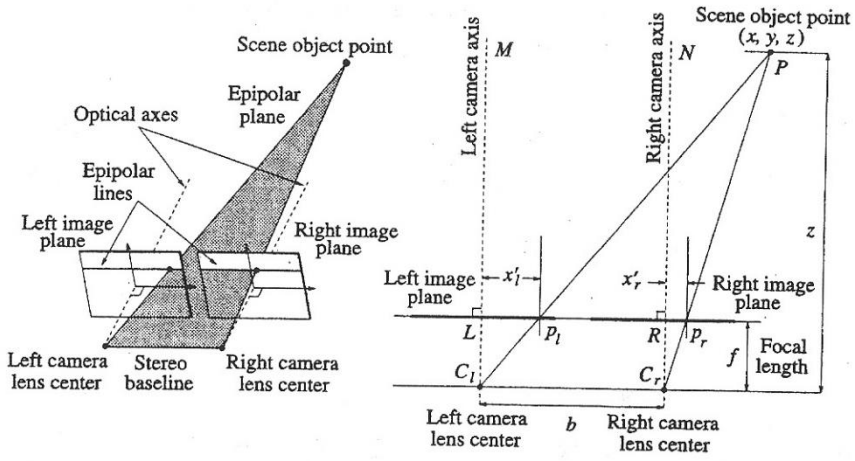


Figura: 69 Cualquier punto de la escena que es visible en ambas cámaras, será proyectado en un par de puntos de la imagen, en 2 imágenes llamados, par conjugado.
Fuente: [5] Ramesh J., Rangachar K., Gilmore B. Schunck (1995). Machine Vision. McGraw Hill. USA

En la Figura: 69 se muestra la geometría de las cámaras respecto al plano de la imagen y a un punto cualquiera en el mundo 3D. [5]

Como se mencionó anteriormente se procede a hallar las matrices de transformación tanto de traslación, Rotación, Escalabilidad y perspectiva de la cámara respecto al objeto (refiere en este caso que la cámara es el objeto móvil mientras que el objeto a identificar es estático, por lo que tanto posición rotación varían en caso que el objeto sea móvil y la cámara estática).

4.2.1.1 Traslación 3D

La traslación 3D se realizó de la siguiente manera:

Se asumió un cierto punto en el espacio (x_1, y_1, z_1), luego se traslada ese objeto una distancia (dx, dy, dz), entonces al final se tuvo un punto final trasladado (x_2, y_2, z_2) como el demostrado en la ecuación 4. 1.

$$\begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix} = \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} + \begin{bmatrix} dx \\ dy \\ dz \end{bmatrix} \qquad 4.\ 1$$

Entonces para calcular este nuevo punto se aplicó la teoría de robótica que habla de las matrices transformaciones básicas de traslación, rotación y sus matrices homogéneas, como también la escalabilidad y perspectiva como se mencionó que recalcan en la teoría de visión artificial.

Para la traslación se hizo:

$$\begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & dx \\ 0 & 1 & 0 & dy \\ 0 & 0 & 1 & dz \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x_1 \\ y_1 \\ z_1 \\ 1 \end{bmatrix} \qquad 4.\ 2$$

$$\begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix} = T * \begin{bmatrix} x_1 \\ y_1 \\ z_1 \\ 1 \end{bmatrix} \qquad 4.\ 3$$

Además su inversa se demuestra por simple

matemática como se expresa en la ecuación 4. 4:

$$T^{-1} = \begin{bmatrix} 1 & 0 & 0 & -dx \\ 0 & 1 & 0 & -dy \\ 0 & 0 & 1 & -dz \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad 4.\ 4$$

Donde “T” es la matriz de transformación homogénea cuando un objeto es trasladado en el espacio.

La matriz homogénea está dada por la ecuación 4. 5:

$$T = \begin{bmatrix} Rotation_{3x3} & Translation_{3x1} \\ Perpective_{1x3} & Scaling_{1x1} \end{bmatrix}_{4x4} \qquad 4.\ 5$$

A través de las coordenadas homogéneas se puede representar la localización de sólidos en un espacio n dimensional a un espacio n+1 dimensional. Es decir, un espacio n-dimensional se encuentra representado en coordenadas homogéneas por (n+1) dimensiones, de tal forma que un vector p(x,y,z) vendrá representado por p(wx,wy,z,w), donde w tiene un valor arbitrario y representan un factor de escala.

A partir de la definición de las coordenadas homogéneas surge inmediatamente el concepto de matriz de transformación homogénea.

Se define como matriz de transformación homogénea T a una matriz de dimensión 4*4 que representa la transformación de un vector de coordenadas homogéneas de un sistema de coordenadas a otro tal y como se representa en la ecuación 4.5

4.2.1.2 Escalamiento (Scaling)

Se refiere al aumento o disminución de proporciones.

Es simplemente la multiplicación de los elementos del vector del punto (x, y) por una constante.

Es la transformación lineal que aumenta de tamaño o reduce los objetos por un factor de Escala que es la misma en todas las direcciones. La expresión se encuentra en la ecuación 4. 6

$$\begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix} = \begin{bmatrix} x_1 * s_x \\ y_1 * s_y \\ z_1 * s_z \end{bmatrix} \qquad 4.\ 6$$

Seguidamente se muestra cómo se expresa la matriz de escalabilidad en la ecuación 4. 7 y 4. 8:

$$\begin{bmatrix} x_2 \\ y_2 \\ z_2 \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x_1 \\ y_1 \\ z_1 \\ 1 \end{bmatrix} \qquad 4.\ 7$$

$$\begin{bmatrix} x_2 \\ y_2 \\ z_2 \\ 1 \end{bmatrix} = S * \begin{bmatrix} x_2 \\ y_2 \\ z_2 \\ 1 \end{bmatrix} \qquad 4.\ 8$$

Donde “S” es la matriz de escalabilidad.

También se tiene la ecuación 4.9:

$$S^{-1} = \begin{bmatrix} \frac{1}{s_x} & 0 & 0 & 0 \\ 0 & \frac{1}{s_y} & 0 & 0 \\ 0 & 0 & \frac{1}{s_z} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad 4.\ 9$$

$$S.S^{-1} = S^{-1}.S = I$$

Dónde:

$$I = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad 4.\ 10$$

4.2.1.3 Rotación

Para entender a lo que esto refiere se realizó gráfica del eje de coordenadas como se muestra en la Figura: 70

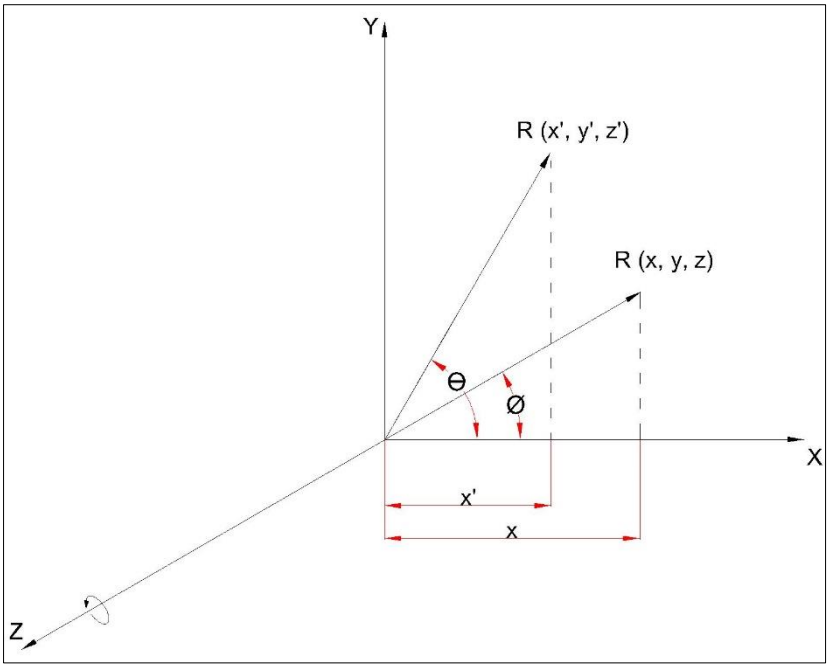


Figura: 70 rotación respecto al eje Z

Fuente: Elaboración propia

Suponiendo que se hace rotación respecto al eje Z

Sea xyz coordenadas iniciales

Sea x'y'z' coordenadas después de rotarlas

Se tiene vector R

Entonces se tiene:

$$x = R.Cos\phi$$

$$y = R.Sin\phi$$

Si se rota respecto a Z un ángulo θ:

$$x' = R.Cos(\theta + \phi) = R.Cos\theta.Cos\phi - R.Sin\theta.Sin\phi$$

$$y' = R.Sin(\theta + \phi) = R.Sin\theta.Cos\phi + R.Cos\theta.Sin\phi$$

Entonces:

$$x' = X.Cos\theta - Y.Sin\theta$$

$$y' = X.Sin\theta + Y.Cos\theta$$

Entonces respeto a Z se tiene Rotación como se demostró:

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} Cos\theta & -Sin\theta & 0 \\ Sin\theta & Cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \\ z \end{bmatrix} = R_\theta^z \qquad 4.\ 11$$

Siendo R_θ^z la rotación respecto al eje Z un ángulo θ

De la misma manera se tiene la ecuación 4.11:

$$(R_\theta^z)^{-1} = \begin{bmatrix} Cos\theta & Sin\theta & 0 \\ -Sin\theta & Cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad 4.\ 12$$

Se notó que la inversa de R_θ^z es la traspuesta de la misma matriz obteniendo la ecuación 4.12:

4. 13

$$(R_\theta^z)(R_\theta^z)^{-1} = I = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Así como se demostró una rotación en el eje Z un ángulo θ , se demostró la matriz de rotación respecto a los ejes X e Y con rotaciones β y φ respectivamente. Ver Figura: 71

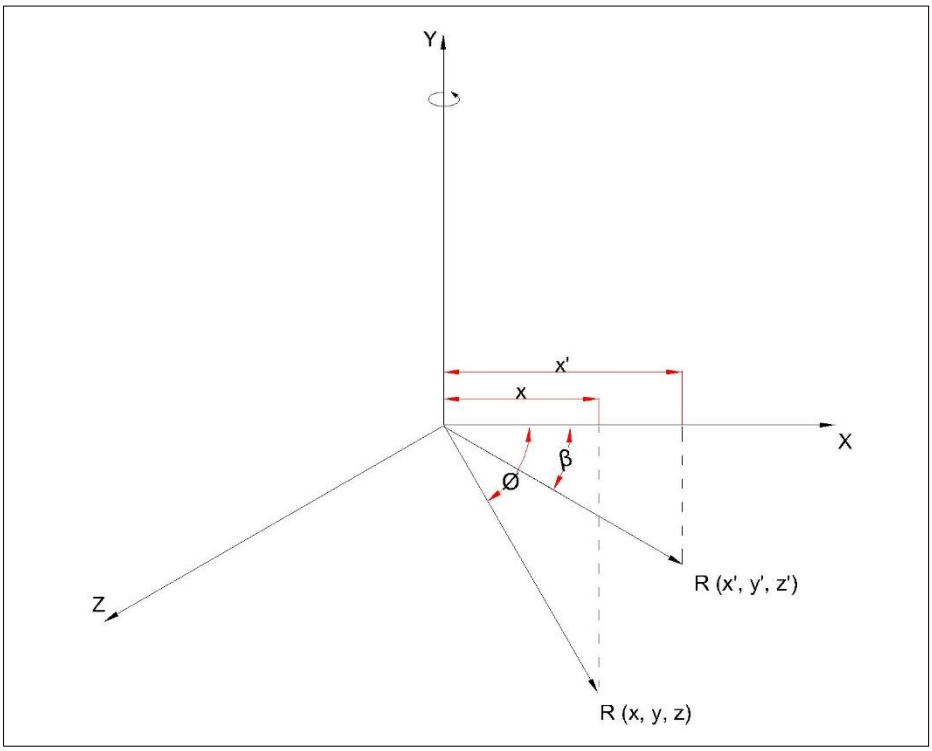


Figura: 71 Rotación respecto al eje Y

Fuente: Elaboración propia

En la Figura anterior se muestra el eje de coordenadas esta vez rotado respecto al eje Y un ángulo β

De la misma manera demostrada anteriormente R_β^y sería:

$$R_{\beta}^{y}=\begin{bmatrix} Cos\beta & 0 & Sin\beta \\ 0 & 1 & 0 \\ -Sin\beta & 0 & Cos\beta \end{bmatrix} \qquad 4.\ 14$$

Seguidamente lo mismo respecto al eje X

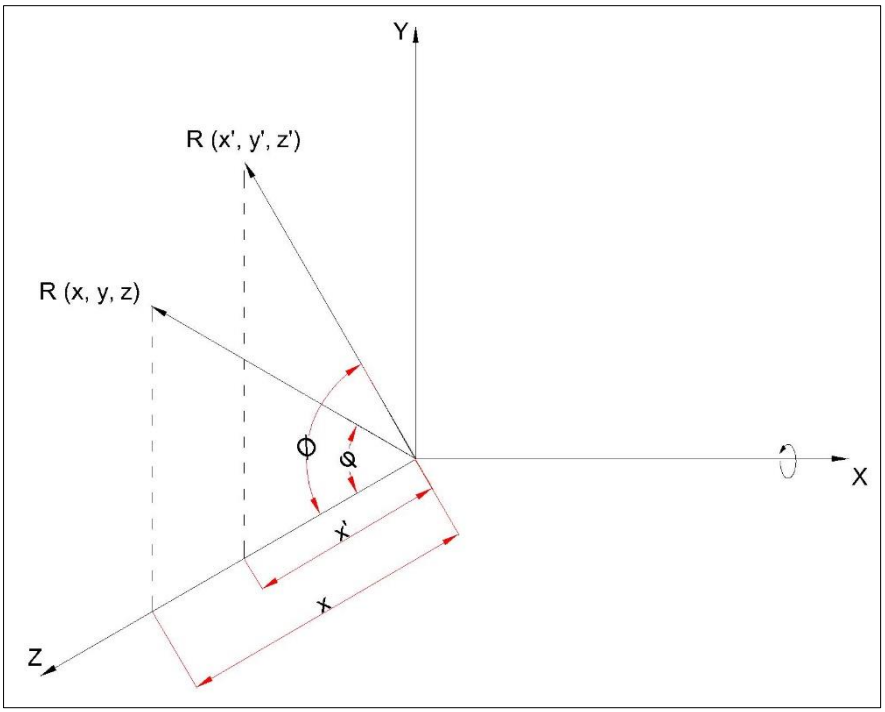


Figura: 72 Rotación respecto al eje X

Fuente: Elaboración propia

En la Figura anterior se muestra el eje de coordenadas esta vez rotado respecto al eje X un ángulo φ

De la misma manera demostrada anteriormente R_{φ}^{x} sería:

$$R_{\varphi}^{x}=\begin{bmatrix} 1 & 0 & 0 \\ 0 & Cos\varphi & -Sin\varphi \\ 0 & Sin\varphi & Cos\varphi \end{bmatrix} \qquad 4.\ 15$$

Entonces, en el caso que se aplicara una rotación a la cámara primero respecto al eje X, luego al eje Y y luego al eje Z su matriz de rotación sería como la expresada en la ecuación 4. 16:

$$R = R_z^{\theta}.R_y^{\beta}.R_x^{\varphi} = \begin{bmatrix} Cos\theta.Cos\beta & Cos\theta.Sin\beta.Sin\varphi - Sin\theta.Cos\varphi & Cos\beta.Sin\beta.Cos\varphi + Sin\theta.Sin\varphi \\ Sin\theta.Cos\beta & Sin\theta.Sin\beta.Sin\varphi + Cos\theta.Cos\varphi & Sin\theta.Sin\beta.Cos\varphi - Cos\theta.Sin\varphi \\ -Sin\beta & Cos\beta.Sin\varphi & Cos\beta.Cos\varphi \end{bmatrix} \quad 4.\ 16$$

Entonces, si los ángulos son pequeños (aprox. 0°) se obtiene lo siguiente:

Si

$$\alpha \to 0 \quad 4.\ 17$$

$$Sin\alpha = \alpha; Cos\alpha = 1$$

Obteniendo:

$$R = \begin{bmatrix} 1 & -\theta & \beta \\ \theta & 1 & -\varphi \\ -\beta & \varphi & 1 \end{bmatrix} \quad 4.\ 18$$

4.2.1.4 Proyección de perspectiva

Es aquel que relaciona los puntos 3D del mundo real con los puntos 2D de la imagen. Se origina en el centro del lente de la cámara.

Seguidamente se elaboró una gráfica para análisis del mismo:

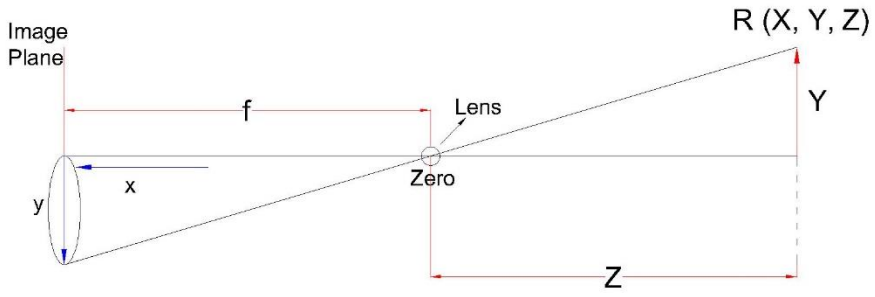


Figura: 73 Relación geométrica desde un punto y objeto en el mundo real respecto a la cámara. (Tomando como punto Zero el lente de la cámara)
Fuente: Elaboración propia

En la Figura: 73 se realizó un análisis de la relación geométrica percibida desde un punto y objeto en el mundo real respecto a la cámara. Se tiene un objeto R ubicado en (X, Y, Z) en el mundo real, luego éste punto, va a través de los lentes y se convierte en una imagen en la cámara en el plano "Image Plane" representado en la Figura 67.

Entonces, mirando a estos 2 triángulos se puede aplicar relación tal como:

$$\frac{-y}{Y} = \frac{f}{Z}$$

También 4. 19

$$-\frac{x}{X} = \frac{f}{Z}$$

Luego se hizo un arreglo a estas ecuaciones para obtener el punto y en las coordenadas del plano de la imagen relacionado con las coordenadas del mundo real, las cuales son (X, Y, Z)

$$y = -\frac{f.Y}{Z} \quad ; \quad x = -\frac{fX}{Z} \qquad 4.\ 20$$

Seguidamente se realizó otro análisis si el punto Zero o el origen se mueve al lado izquierdo, al plano de la imagen tal y como muestra la Figura: 74.

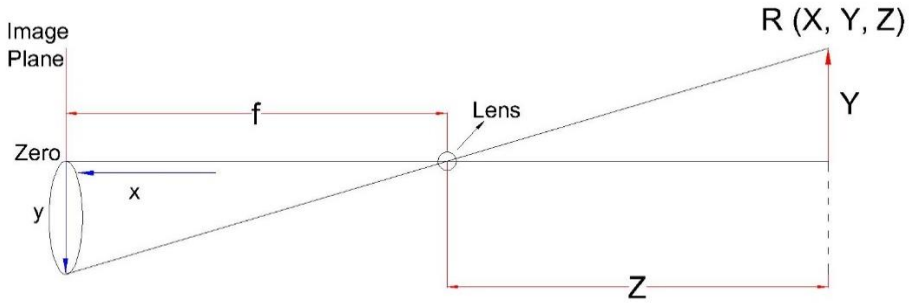


Figura: 74 Relación geométrica desde un punto y objeto en el mundo real respecto a la cámara. (Tomando como punto Zero el plano de la Imagen respecto a la cámara)
Fuente: Elaboración propia
En el análisis se obtuvo lo siguiente:

$$-\frac{y}{Y}=\frac{f}{Z-f} \qquad 4.\ 21$$

$$y=-\frac{f.Y}{Z-f}\ ;\quad x=-\frac{f.X}{f-Z} \qquad 4.\ 22$$

Ahora se hizo el mismo análisis, con la diferencia que el origen se movió a la derecha, en el plano del mundo 3D tal y como muestra la Figura: 75.

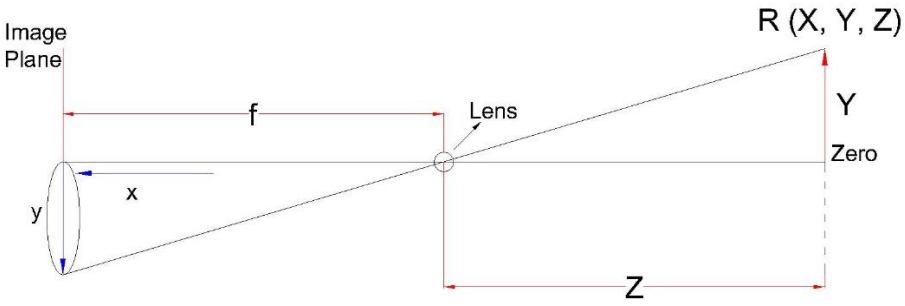


Figura: 75 Relación geométrica desde un punto y objeto en el mundo real respecto a la cámara. (Tomando como punto Zero el plano en el medio real)
Fuente: Elaboración propia

En el análisis se obtuvo lo siguiente:

$$y = \frac{f.Y}{f - Z} \quad ; \quad x = \frac{f.X}{f - Z} \qquad 4.\ 23$$

4.2..1.5 Matriz de transformación de Perspectiva

Lo que se deseó obtener para poder realizar la estereoscopía es la matriz de perspectiva para poder integrarla a la matriz homogénea y al final poder la posición del objeto y su orientación respecto al plano de la imagen en el mundo 3D.

Sin embargo, en primera instancia, respecto al análisis hecho anteriormente no se pudo obtener esta matriz de perspectiva ya que es una expresión NO LINEAL, debido a que aparece un Z en el denominador.

Entonces la matriz que se obtuve fue algo de la forma:

$$\begin{bmatrix} x \\ y \\ - \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} . \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \qquad 4.\ 24$$

Donde (X, Y, Z) es las coordenadas del mundo, (x, y) son las coordenadas en el plano de la Imagen.

Entonces se llegó a esta función de transformación mostrada en la ecuación 4.24, el cual de alguna manera representa la función de Transformación de perspectiva, pero es no lineal, por ende, lo que se hizo es usar las coordenadas homogéneas.

Se tiene (X,Y,Z) en las coordenadas cartesianas. Se convirtió estas en coordenadas homogéneas tan sólo multiplicándolos por una constante k y por último se obtuvo un componente por DEFAULT k como se muestra en la ecuación. 4.26

$$(X,Y,Z) \rightarrow (k.X, k.Y, k.Z, k) \qquad \text{4. 25}$$

Coord. 3D Coord. 4D DEFAULT Comp.

Donde $(k.X, k.Y, k.Z, k)$ representa la transformación homogénea que se realizó a las coordenadas 3D.

Entonces con este simple algoritmo pasamos de 3D a 4D y también por el sentido contrario, se puede convertir de coordenas HOMOGÉNEAS a CARTESIANAS al dividir el 4to elemento de la coordenada homogénea por el 1er, 2do y 3er término como se expresa a continuación en la ecuación 4.27

$$(C_{h1}, C_{h2}, C_{h3}, C_{h4}) \rightarrow \left(\frac{C_{h1}}{C_{h4}}, \frac{C_{h2}}{C_{h4}}, \frac{C_{h3}}{C_{h4}}\right) \qquad \text{4. 26}$$

Homogéneas (4D) Cartesianas 3D

La expresión en coordenadas cartesianas $\left(\frac{C_{h1}}{C_{h4}}, \frac{C_{h2}}{C_{h4}}, \frac{C_{h3}}{C_{h4}}\right)$ también es conocida como la inversa de la homogénea.

Según expresa **"Mubarak Shah"** en su libro **"Fundamentals of Computer Vision"** la matriz de perspectiva se expresa como:

$$P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -\frac{1}{f} & 1 \end{bmatrix} \qquad 4.\ 27$$

Entonces se aplicando lo anterior se obtiene:

$$\begin{bmatrix} C_{h1} \\ C_{h2} \\ C_{h3} \\ C_{h4} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -\frac{1}{f} & 1 \end{bmatrix} * \begin{bmatrix} k.X \\ k.Y \\ k.Z \\ k \end{bmatrix} \qquad 4.\ 28$$

Homogeneous Coordinates

Perpective Transformation Matrix

Image

Resolviendo se obtuvo:

$$\begin{bmatrix} C_{h1} \\ C_{h2} \\ C_{h3} \\ C_{h4} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -\frac{1}{f} & 1 \end{bmatrix} * \begin{bmatrix} k.X \\ k.Y \\ k.Z \\ k \end{bmatrix} = \begin{bmatrix} k.X \\ k.Y \\ k.Z \\ k - \frac{k.Z}{f} \end{bmatrix} \qquad 4.\ 29$$

Entonces con la expresión de la ecuación 4.29 se obtuvo finalmente las coordenadas de la imagen en el sistema de coordenadas homogéneas.

Finalmente, para obtener las coordenadas reales, se requiere obtener las coordenadas cartesianas, entonces simplemente se realiza el algoritmo explicado anteriormente, el cual es tomar el 4to elemento C_{h4} y dividirlo por el 1ro, 2do, 3ro y sucesivamente.

De lo obtenido anteriormente se obtuvo:

$$C_{h1} = k.X \qquad C_{h2} = k.Y \qquad C_{h3} = k.Z \qquad C_{h4} = k - \frac{k.Z}{f}$$

Entonces, siendo (x, y) coordenadas del plano de la Imagen, y (X, Y, Z) coordenadas del mundo se obtuvo:

$$x = \frac{C_{h1}}{c_{h4}} = \frac{k.X}{k - \frac{k.Z}{f}} = \frac{f.X}{f - Z} \qquad \text{4. 30}$$

$$y = \frac{C_{h2}}{c_{h4}} = \frac{k.Y}{k - \frac{k.Z}{f}} = \frac{f.Y}{f - Z} \qquad \text{4. 31}$$

Después de esto, se podrá hallar el modelamiento de la Cámara (o las relaciones de las coordenadas respecto al plano de la imagen, y al mundo.

4.3 Modelamiento de Cámara

Para hallar el Modelamiento de la cámara y comprobar la realidad de los algoritmos demostrados anteriormente, se procedió a ejecutar los siguientes movimientos a la cámara.

Se toma en cuenta que el objeto móvil es la cámara para esta Tesis, más no el objeto objetivo a detectar.

Entonces:

- La cámara está en el origen de las coordenadas del Mundo
- Luego es trasladada una distancia G
- Luego es rotada alrededor del eje Z en contra del sentido de las agujas del reloj (anti horario) un ángulo θ.
- Luego es rotada nuevamente alrededor del eje X en sentido anti horario un ángulo ϕ
- Luego es trasladado una distancia C
- Entonces se obtuvo la siguiente transformación:

$$C_h = P.C.R^X_{-\phi}.R^Z_{-\theta}.G.W_h \qquad 4.32$$

Donde:

$W_h \rightarrow$ Pto. En el Mundo en coordenadas homogéneas

Cartesianas: (X, Y, Z)

Homogéneas: $(k.X, k.Y, k.Z, k)$

$G \rightarrow$ Traslación

$R^Z_{-\theta} \rightarrow$ Rotación alrededor del eje Z de forma anti horaria un ángulo θ

$R^X_{-\phi} \rightarrow$ Rotación alrededor del eje Z de forma anti horaria un ángulo ϕ

$C \rightarrow$ Traslación

$P \rightarrow$ Transformación de Perspectiva (Toma la imagen / foto)

$C_h \rightarrow$ Como resultado se obtiene coordenadas en el sistema de coordenadas de la Imagen

Como objetivo de aplicar lo anterior en esta tesis es encontrar cuál serán las coordenadas de la imagen en la cámara aplicando estas transformaciones respecto al mundo real.

De lo dicho anteriormente se desarrolló lo siguiente:

$$W_h = \begin{bmatrix} k.X \\ k.Y \\ k.Z \\ k \end{bmatrix} \qquad 4.\ 33$$

$$P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -\frac{1}{f} & 1 \end{bmatrix} \qquad 4.\ 34$$

$$T_1^{-1} = G = \begin{bmatrix} 1 & 0 & 0 & -X_0 \\ 0 & 1 & 0 & -Y_0 \\ 0 & 0 & 1 & -Z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad 4.\ 35$$

$$(R_\theta^Z)^{-1} = R_{-\theta}^Z = \begin{bmatrix} C\theta & S\theta & 0 & 0 \\ -S\theta & C\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad 4.\ 36$$

$$(R_\phi^X)^{-1} = R_{-\phi}^X = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & C\phi & S\phi & 0 \\ 0 & -S\phi & C\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad 4.\ 37$$

$$T_2^{-1} = C = \begin{bmatrix} 1 & 0 & 0 & -r_1 \\ 0 & 1 & 0 & -r_2 \\ 0 & 0 & 1 & -r_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad 4.\ 38$$

Debido a que se está moviendo la cámara, y no el objeto (objeto estático) se usa Transformaciones inversas, cuyas resultantes se demostraron los capítulos 4.2.1.1 & 4.2.1.3 de Traslación y Rotación en esta Tesis.

Resolviendo C_h:

$$G.W_h = A$$

$$A = \begin{bmatrix} k.(X - X_0) \\ k.(Y - X_0) \\ k.(Z - X_0) \\ k \end{bmatrix} \qquad 4.\ 39$$

$$R_{-\theta}^{Z}.A = B$$

$$B = \begin{bmatrix} k.C\theta(X - X_0) + k.S\theta(Y - Y_0) \\ -k.S\theta(X - X_0) + k.C\theta(Y - Y_0) \\ k.(Z - Z_0) \\ k \end{bmatrix} \qquad 4.\ 40$$

$$R_{-\phi}^{X}.B = D \qquad 4.\ 41$$

$$D=\begin{bmatrix} k.C\theta(X-X_0)+k.S\theta(Y-Y_0) \\ -k.S\theta(X-X_0)+k.C\theta C\phi(Y-Y_0)+k.S\phi(Z-Z_0) \\ k.S\theta.S\phi(X-X_0)-k.C\theta.S\phi(Y-Y_0)+k.C\phi(Z-Z_0) \\ k \end{bmatrix}$$

$$C.D=E$$

$$E=\begin{bmatrix} k.C\theta(X-X_0)+k.S\theta(Y-Y_0)-k.r_1 \\ -k.S\theta(X-X_0)+k.C\theta C\phi(Y-Y_0)+k.S\phi(Z-Z_0)-k.r_2 \\ k.S\theta.S\phi(X-X_0)-k.C\theta.S\phi(Y-Y_0)+k.C\phi(Z-Z_0)-k.r_3 \\ k \end{bmatrix} \qquad 4.42$$

$$P.E=C_h$$

$$E=\begin{bmatrix} k.C\theta(X-X_0)+k.S\theta(Y-Y_0)-k.r_1 \\ -k.S\theta(X-X_0)+k.C\theta C\phi(Y-Y_0)+k.S\phi(Z-Z_0)-k.r_2 \\ k.S\theta.S\phi(X-X_0)-k.C\theta.S\phi(Y-Y_0)+k.C\phi(Z-Z_0)-k.r_3 \\ \dfrac{-k.S\theta.S\phi(X-X_0)+k.C\theta.S\phi(Y-Y_0)-k.C\phi(Z-Z_0)+k.r_3+k.f}{f} \end{bmatrix} = \begin{bmatrix} C_{h1} \\ C_{h2} \\ C_{h3} \\ C_{h4} \end{bmatrix} = C_h \qquad 4.43$$

Entonces, como se demostró, se obtuvo C_h, lo cual vendría a ser las coordenadas de la imagen en la Cámara, pero esto es en coordenadas HOMOGÉNEAS, lo que se requiere es obtenerlas en coordenadas Cartesianas, por lo que simplemente se divide entre el 4to término tal y como se expresa en 4.26, para obtener coordenadas reales 3D (X,Y,Z).

Entonces las coordenadas de la imagen son simplemente (x,y):

De 4.31 & 4.32 se obtiene:

$$x=\frac{C_{h1}}{C_{h4}}=\frac{k.f[C\theta(X-X_0)+S\theta(Y-Y_0)-r_1]}{k[-S\theta.S\phi(X-X_0)+C\theta.S\phi(Y-Y_0)-C\phi(Z-Z_0)+r_3+f]} \qquad 4.44$$

$$y=\frac{C_{h2}}{C_{h4}}=\frac{k.f[-S\theta.C\phi(X-X_0)+C\theta.C\phi(Y-Y_0)+S\phi(Z-Z_0)-r_2]}{k[-S\theta.S\phi(X-X_0)+C\theta.S\phi(Y-Y_0)-C\phi(Z-Z_0)+r_3+f]} \qquad 4.45$$

Recapitulando, se tiene modelamiento de la Cámara según la expresión 4.33

$$C_h = P.C.\underbrace{R_{-\phi}^{X}.R_{-\theta}^{Z}.G}_{A1_{4x4}}.W_h$$

$$C_h = A1.W_h \qquad 4.\ 46$$

Como $A1_{4x4}$ es resultado de los movimientos, rotaciones, perspectiva de la cámara hechos y expresados por sus matrices de transformación de un arreglo de $4x4$, entonces la C_h se puede expresar como se mostró en 4.47.

De la misma manera, se puede expresar de la siguiente manera:

$$\begin{bmatrix} C_{h1} \\ C_{h2} \\ C_{h3} \\ C_{h4} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} . \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \qquad 4.\ 47$$

Entonces se obtiene:

$$C_{h1} = a_{11}.X + a_{12}.Y + a_{13}.Z + a_{14} \qquad 4.\ 48$$

$$C_{h2} = a_{21}.X + a_{22}.Y + a_{23}.Z + a_{24} \qquad 4.\ 49$$

$$C_{h4} = a_{41}.X + a_{42}.Y + a_{43}.Z + a_{44} \qquad 4.\ 50$$

C_{h3} no es necesario hallarlo ya que en el plano de la imagen no se tiene coordenadas 3D, solo existen (x, y)

Ahora, para regresar a las coordenadas cartesianas reales, se tuvo que dividir C_{h1} & C_{h2} entre C_{h4}

De lo demostrado anteriormente en 4.49, 4.50, 4.51, si se quiere encontrar el modelamiento de la cámara, se tuvo que identificar estos 12 Parámetros $(a_{11}, a_{12}, a_{13}, \ldots, a_{44})$ desconocidos, porque (X, Y, Z) es un punto conocido en el espacio conocido (escogido/ seteado por el mismo operador)

$$x = \frac{C_{h1}}{C_{h4}} \quad ; \quad y = \frac{C_{h2}}{C_{h4}}$$

Entonces en resumen, la manera en la que se halló el modelamiento de la cámara fue, poner la cámara en un punto origen determinado cualquiera, y ese lo considero el $(0, 0)$, luego se Traslada, se Rota, nuevamente se Rota, se Traslada, la distancia focal f de la cámara, coloco esos valores en las ecuaciones halladas en 4.45 & 4.46 y finalmente se tuvo como resultado satisfactoriamente el modelamiento de la cámara.

Gracias a esto, se pudo obtener las coordenadas del escenario 3D en coordenadas cartesianas al plano de la imagen de la cámara.

Una vez realizado esto, se lo colocó en el algoritmo de programación para obtenerlo de forma automática.

Como paso Final, una vez obtenido esta referencia de coordenadas del modelamiento de la cámara, el resto es simple Geometría para poder hallar la profundidad de la cámara la cual se encuentra respecto al objeto.

Como se está usando 2 cámaras, se tiene un lo que se hizo en esta tesis para hallar la profundidad del objeto la cual se ubicaba respecto al plano de la imagen fue la siguiente manera:

Una forma de estimar la profundidad de cada uno de los puntos en la escena es mediante el cálculo de la disparidad entre las imágenes de la misma.USA

[22] La escena es estática, es decir, los objetos visibles en la escena no cambian su posición en la misma ni sufren deformaciones. Para definir la disparidad se elabora una

configuración de dos cámaras de características similares, como la que se muestra en la Figura 70.

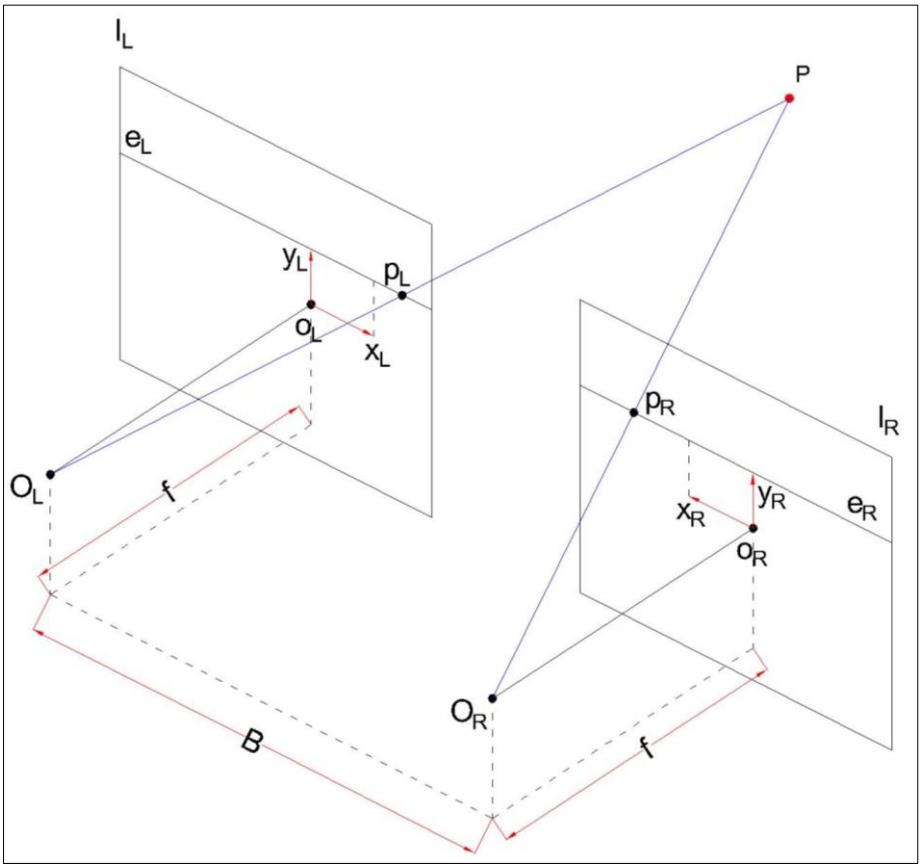


Figura: 76 Geometría estéreo en Proyección respecto al plano de la cámara
Fuente: Elaboración propia

Como se aprecia en la Figura: 76 las 2 cámaras forman un par estéreo.

Los ejes ópticos de las cámaras son paralelos. Ambas cámaras tienen la misma distancia focal f debido a que son el mismo modelo y tipo de cámara Microsoft HD, con centros en 0_L y 0_R separados una distancia B de forma que las imágenes que se forman, I_L e I_R estén planos paralelos. De esta manera la línea base es paralela a la coordenada $\vec{x}$ de las imágenes. Con el modelo considerado de las cámaras, un punto en el espacio tridimensional P con coordenadas homogéneas $(X,Y,Z,1)^T$ se proyecta en cada una de

las imágenes bidimensionales en los puntos p_L y p_R con coordenadas homogéneas $(x_L, y_L, 1)^T$ y $(x_R, y_R, 1)^T$ respectivamente.

El plano que contiene a los puntos P, O_L y 0_R, intersecta a las imágenes en dos rectas epipolares e_L y e_R. Dada la configuración del par estéreo, éstas son rectas epipolares entre sí, es decir, un punto p_L en la recta e_L de la imagen I_L tiene su correspondiente en algún punto de la recta e_R.

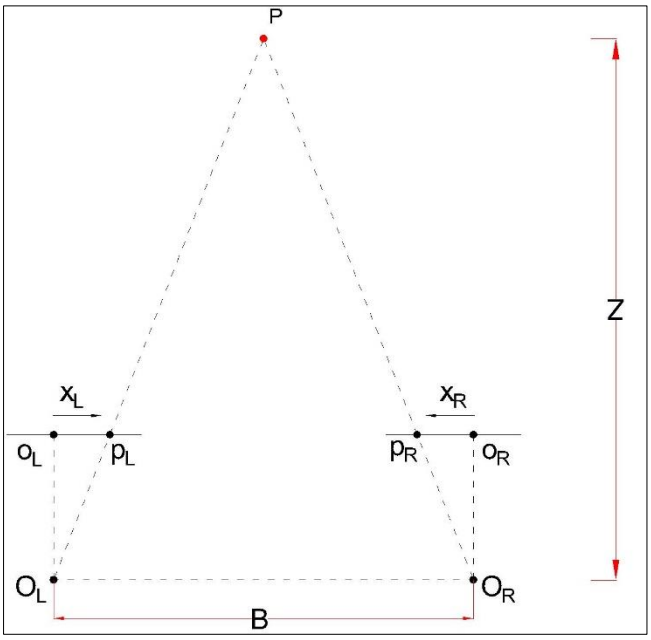


Figura: 77 Geometría de relación estéreo entre Cámaras
Fuente: Elaboración propia

En la Figura: 77 se puede apreciar cómo se relacionó los parámetros definidos en el par estéreo, que permiten obtener la relación entre la disparidad d y la profundidad Z del punto P.

Una vez dicho todo esto, lo único que falta definir es la disparidad, y luego por una simple semejanza de triángulos se pudo sacar la distancia (profundidad) Z la cual está localizado el objeto en el escenario 3D.

La disparidad es la diferencia en las coordenadas horizontales de los puntos p_L y p_R, es decir $d = x_L - x_R$. Las coordenadas de p_L y p_R quedan relacionadas mediante:

$$x_L = x_R + d$$

Además: 4. 51

$$y_L = y_R$$

Finalmente, la distancia Z se obtuvo mediante la semejanza de triángulos PO_LO_R $p_Ro_RO_R$ y $p_Lo_LO_L$ la cual se llega a la relación entre d y Z según la siguiente ecuación:

$$Z.d = f.B$$ 4. 52

Del mismo modo: $Z = \frac{f.B}{d}$ 4. 53

De esta forma se calculó la profundidad la cual se encontraba el objeto a partir de la disparidad calculada, tomando como referencia y parámetro conocido la distancia entre las cámaras de 8cm.

CAPÍTULO 5:

PRUEBAS Y RESULTADOS

5.1 Pruebas

Prueba A:

a) Se desarrolló, programó e implementó algoritmos morfológicos matemáticos con diferentes elementos estructurantes a aplicarse posteriormente en las diferentes imágenes tomadas en el proceso de adquisición de la imagen.
b) Se programó he implementó diversos algoritmos de filtros espaciales, para seguidamente en el software confirmar los mejores resultados bajo las condiciones de la adquisición obtenidas y escoger el mejor filtro para su desarrollo en el dominio de la frecuencia
c) Se programó he implementó diversos algoritmos de filtros frecuenciales para posteriormente realizar la simulación respectiva del procesamiento de imágenes.

Prueba B

a. Se elaboró y programó algoritmo de etiquetado y filtro tamaño con las diferentes imágenes adquiridas en el proceso de adquisición de la imagen.

Prueba C

a. Se planteó diferentes arquitecturas de Redes Neuronales multicapa, con diferentes valores de pesos sinápticos para su actualización y psoterior aprendizaje
b. Se planteó set de aprendizaje de 10, 15 y 20 elementos

Prueba D

a. Para estereoscopía se tomó snapshots bajo los diferentes ángulos del tablero de ajedrez rotados y trasladados.
b. Se realizó calibración de cámaras usando un tablero de ajedrez, colocándolo a 79cm de distancia respecto a la cámara, y luego rotándola 8 veces ligeramente
c. Se tomaron 8 imágenes sucesivas tanto para la cámara derecha como para la cámara izquierda.

d. Se realizó el mismo proceso para diferentes distancias en Z del tablero de ajedrez respecto a las cámaras.

5.2 Resultados

Resultado A:

a. Se nota que este sistema sería óptimo sólo bajo ciertas condiciones de luminosidad. Figura: 78

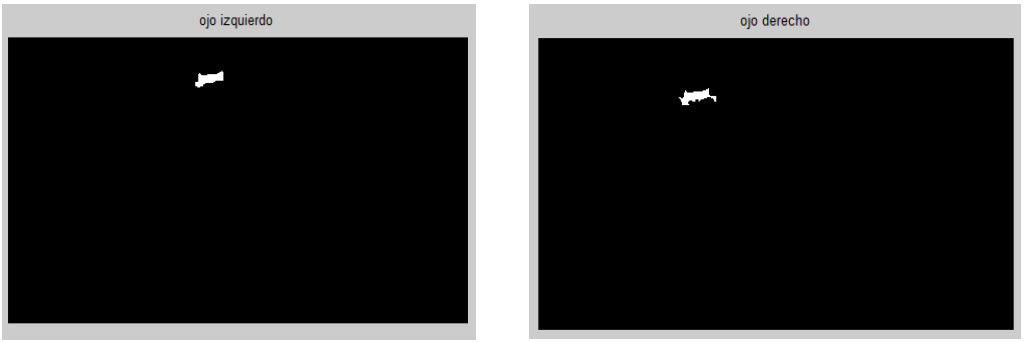


Figura: 78 Imagen tomada bajo malas condiciones de luminosidad

Fuente: Elaboración propia

b. En el presente proyecto de Tesis, se obtuvo mejores resultados aplicando Filtros Gausianos de diferentes ventanas, obtando mejor por una ventana de k=15

c. La Erosión de la imagen permitió difuminar el ruido de fondo el cual no deseamos en la etapa de pre-procesamiento y segmentación de la imagen

Resultado B

a. Con el Filtro de tamaño, se pudo obtener el objeto de interés de una manera mucho más rápido, mejorando el tiempo computacional, que erosionando, dilatando la imagen N veces. Ver Figura: 79

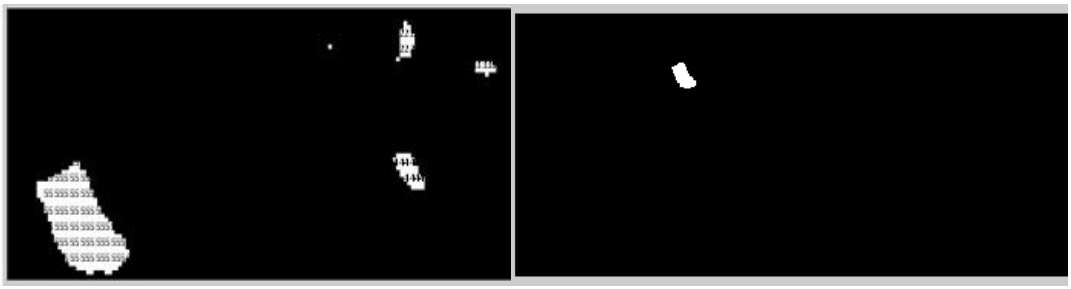


Figura: 79 Aplicación de filtro de tamaño después de etiquetar la imagen en la etapa de post-procesamiento de la imagen

Fuente: Elaboración propia

Resultado C

Figura: 80 Resultado de aplicación Red Neuronal Multicapa Backpropagation

a. En la Figura: 80 se representa el resultado de la Red Neuronal Multicapa BackPropagation después de la aplicación de 20 datasets de aprendizaje.
b. El resultado se obtuvo al aplicarlo de entre 3 diferentes formas de objetos, para el cual se aplicaron valores de salidas deseadas para el objeto a identificar.

Resultado D

a. Al hacer la calibración de la cámara, se llegaron a obtener los parámetros intrínsecos y extrínsecos, llegando a realizar la estereoscopía exitosamente. Sin embargo, cabe decir que la medida en Z Real vs la Medida obtenida después de la simulación, hay un error de +/-50mm. Anexo 23 (Parámetros Intrínsecos de cámaras)
b. Para obtener una muy buena aproximación de los parámetros de la cámara, es mejor realizar varios snapshots al momento de la calibración. De ésta manera tanto el eje focal, disparidad, distorsión, y otros parámetros intrínsecos serán más exactos, y se obtendrá un menor error al momento de convertir las coordenadas de la cámara en coordenadas en el mundo real. Anexo 23

c. Se concluye que con 8 imágenes es suficiente para realizar una buena calibración de la cámara, sin embargo se pudo escoger más imágenes para obtener una buena aproximación. Ver Figura: 81

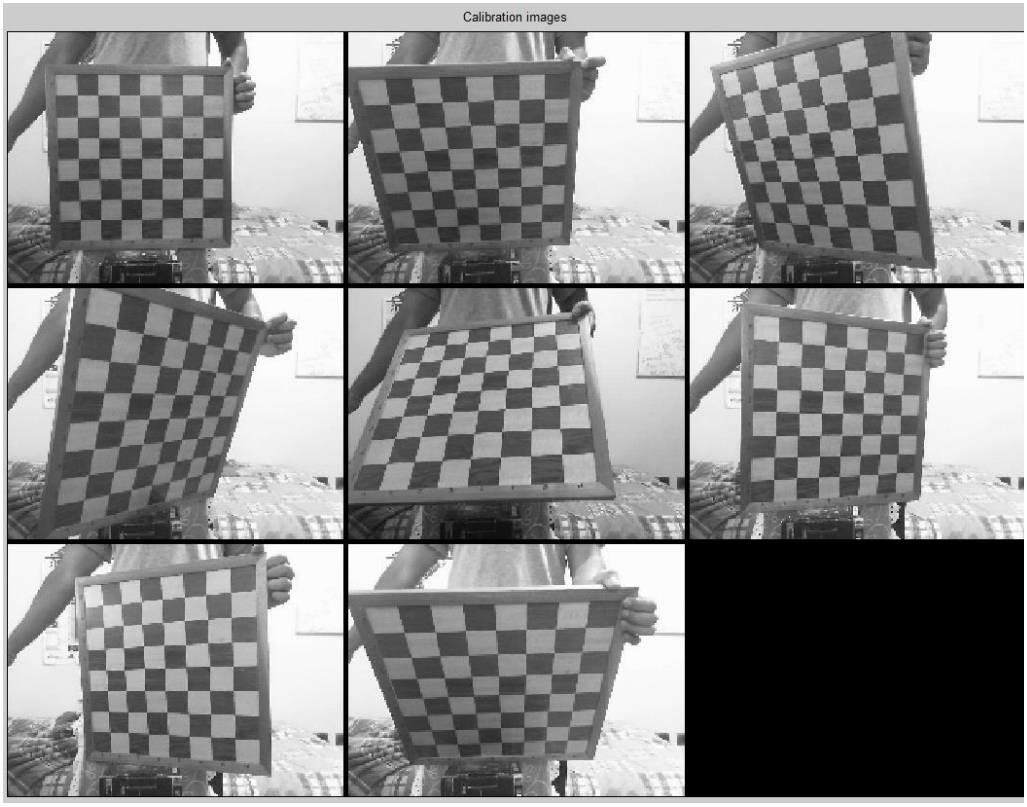


Figura: 81 Tablero para calibración de cámara

Fuente: Elaboración propia

d. Al momento de hacer la calibración se escogió primero las 4 esquinas, para luego el algoritmo realizado detecte automáticamente la cantidad de cuadrados del Checkerboard, y así para cada imagen. Esto con el fin de rotarlo, trasladarlo respecto a la cámara, aplicar los algoritmos mostrados en la sección 4.3 “Camera Model” del capítulo 4, y poder sacar una aproximación de los puntos respecto a las coordenadas de la cámara (considerando el modelo, cámara estática - objeto móvil).

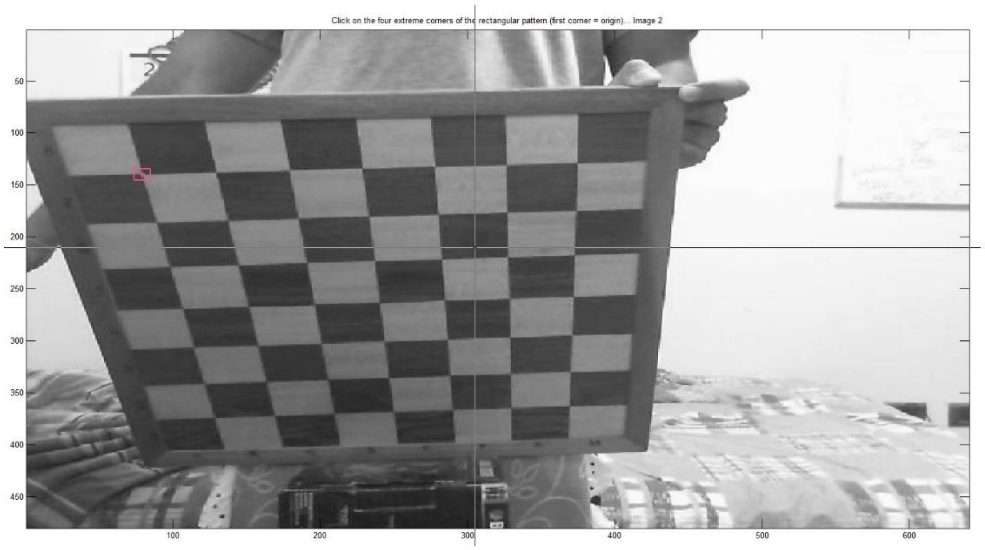


Figura: 82 Primer punto, esquina Superior izquierda

Fuente: Elaboración propia

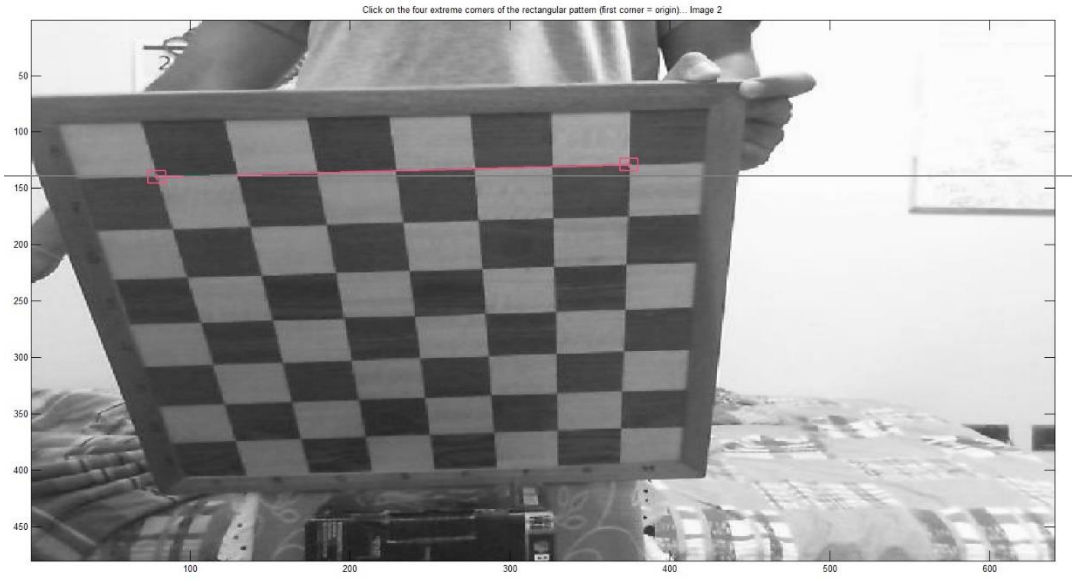


Figura: 83 Segundo Punto esquina Superior Derecha

Fuente: Elaboración propia

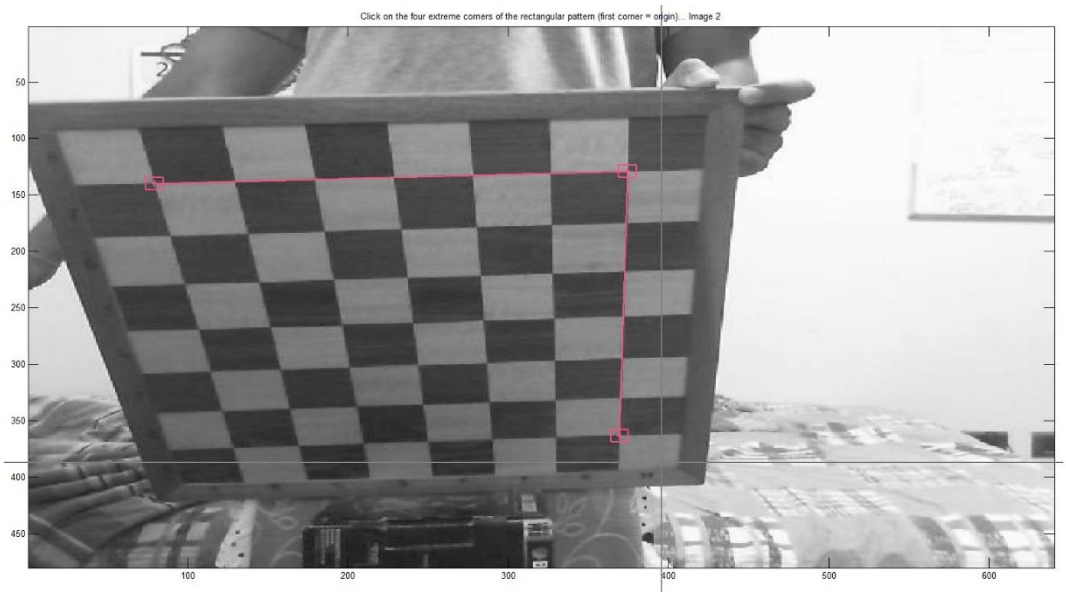


Figura: 84 Tercer punto esquina inferior derecha

Fuente: Elaboración propia

Figura: 85 Cuarto punto esquina inferior izquierda

Fuente: Elaboración propia

e. En la Figura: 86 se muestra que el algoritmo detecta satisfactoriamente los 49 puntos, y así en cada imagen tomada tanto en el ojo derecho como en el Izquierdo en los 8 casos.

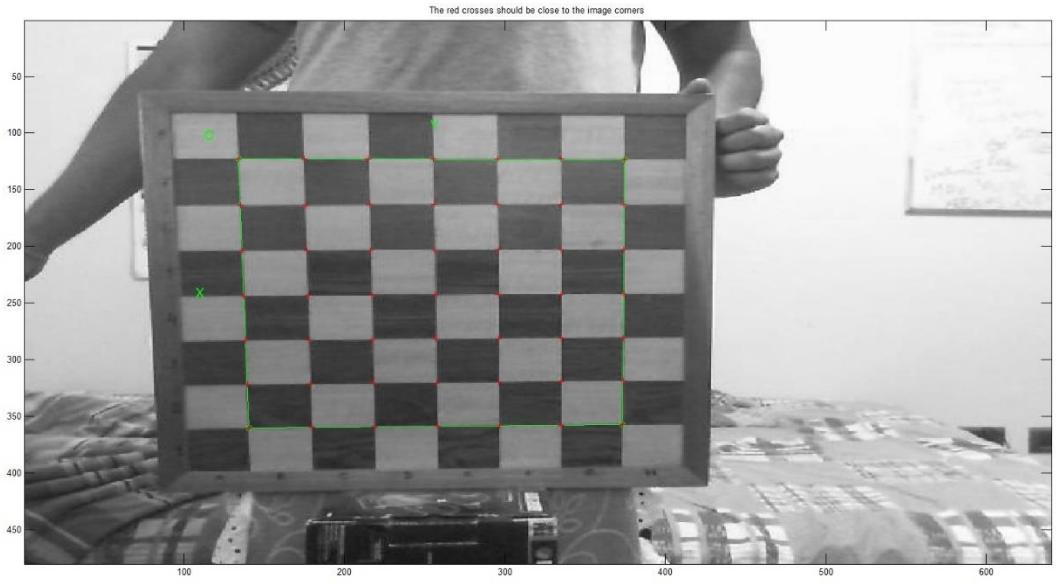


Figura: 86 Vista defrente del sistema de autodeteccion de puntos

Fuente: Elaboración propia

f. Entre las cámaras en ésta prueba se consideró una disparidad de 13cm, y se procedió a realizar los snapshots respectivos tanto para el ojo Derecho como para el Izquierdo Mostrados en Figura: 87 y Figura: 88

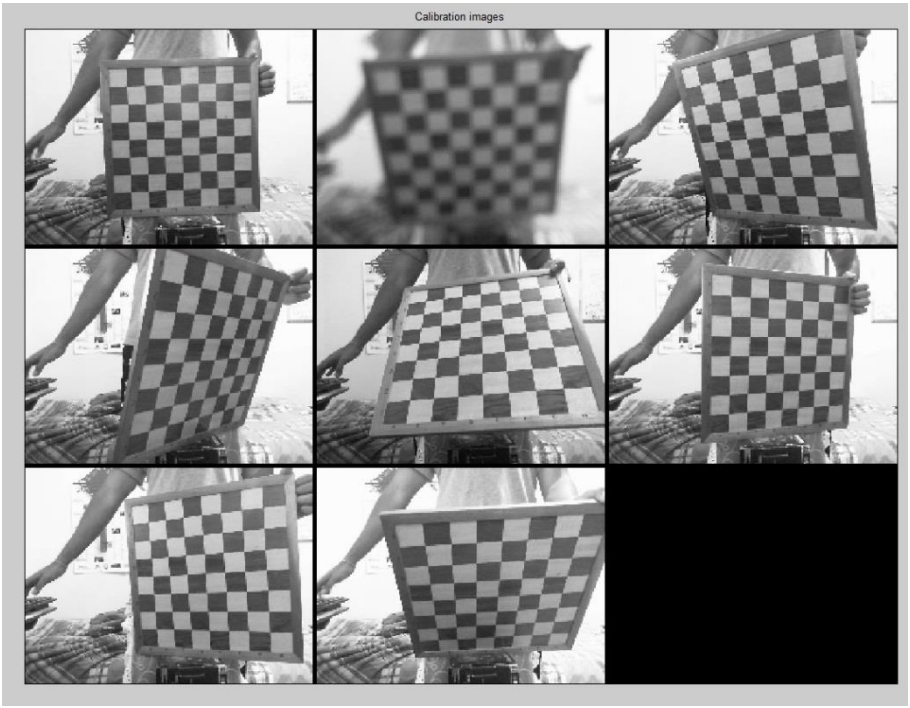


Figura: 87 Captura de imagenes de Cámara izquierda

Fuente: Elaboración propia

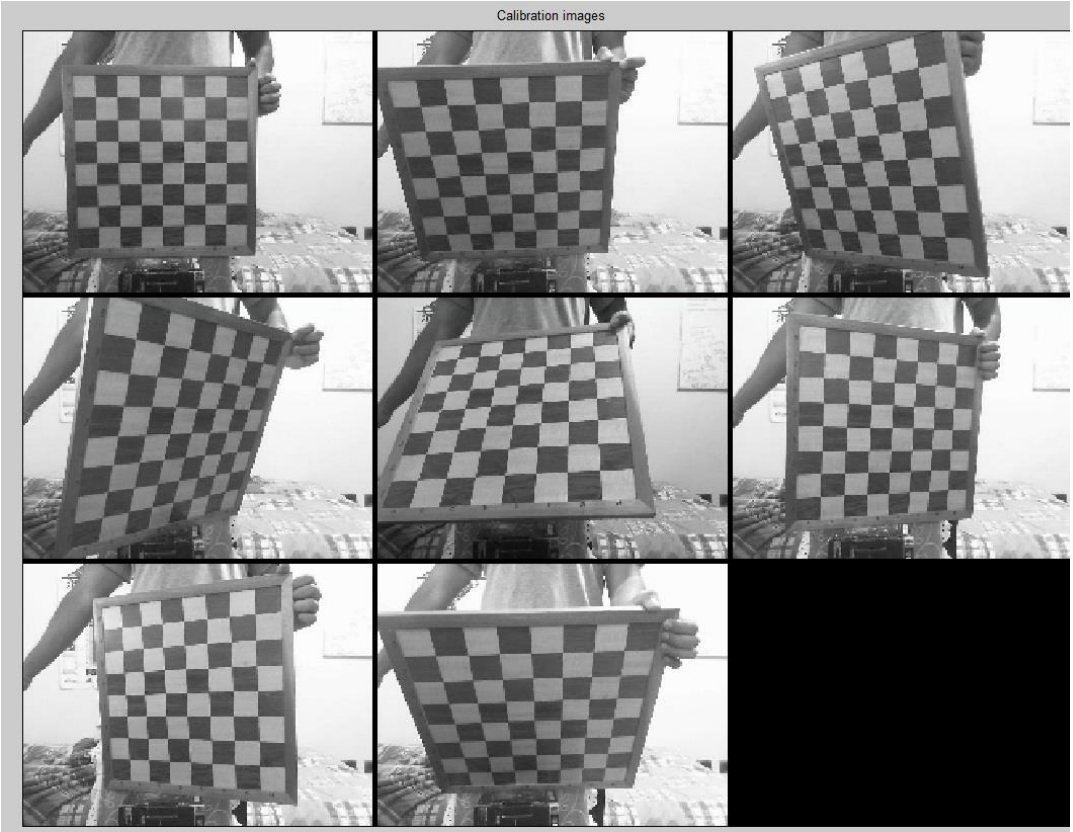


Figura: 88 Captura de imagenes de Cámara derecha

Fuente: Elaboración propia

g. Los parámetros de la cámara obtenidos tanto para el ojo Derecho e ojo Izquierdo resultaron similares, tal y como se esperaba. Ver Anexo 24

h. El tablero se colocó a una distancia de 79cm los cuales se sacaron las imágenes para cada cámara.

i. Cada cuadrado del Checkerboard mide 48cm.

j. En el Anexo 25 se observa las coordenadas en el mundo Real (X,Y,Z) Notando que en Z exitosamente llega a obtener la distancia la cual está cada punto de los 49 muestreados por cada imagen (Xc_1_left es la imagen 1 del ojo izquierdo). La distancia detectada por el algoritmo está en el rango de [710 – 750] mm, mostrando un error de Δ[40-80] mm respecto al medido en Real de 790mm.

k. En el Anexo 26 se observa las coordenadas en el mundo Real (X,Y,Z) mostrando que cada cuadrado está midiendo en un rango de [46-49]mm respecto al real de 48mm defiriendo del real en tan solo Δ2mm aprox. En X lo podemos apreciar simplemente sacando un promedio de los 7 primeros, y luego de los 7 subsiguientes, y luego restando estos 2 promedios como: $((X_1 + X_2 + \ldots + X_7)/7) - ((X_8 + X_9 + \ldots + X_{14})/7)$, y en Y simplemente se hace $Y_1 - Y_2$ siendo Y_1 el punto1 actual detectado de los 49 totales y Y_2 el punto subsiguiente tomando en cuenta la notación de la Figura: 86 y así sucesivamente hasta completar las 7 Columnas (X son Filas=7 y Y son Columnas=7)

Figura: 89 Ancho de Cuadrado de Checkerboard

Fuente: Elaboración propia

Figura: 90 Largo de Cuadrado de Checkerboard

Fuente: Elaboración propia

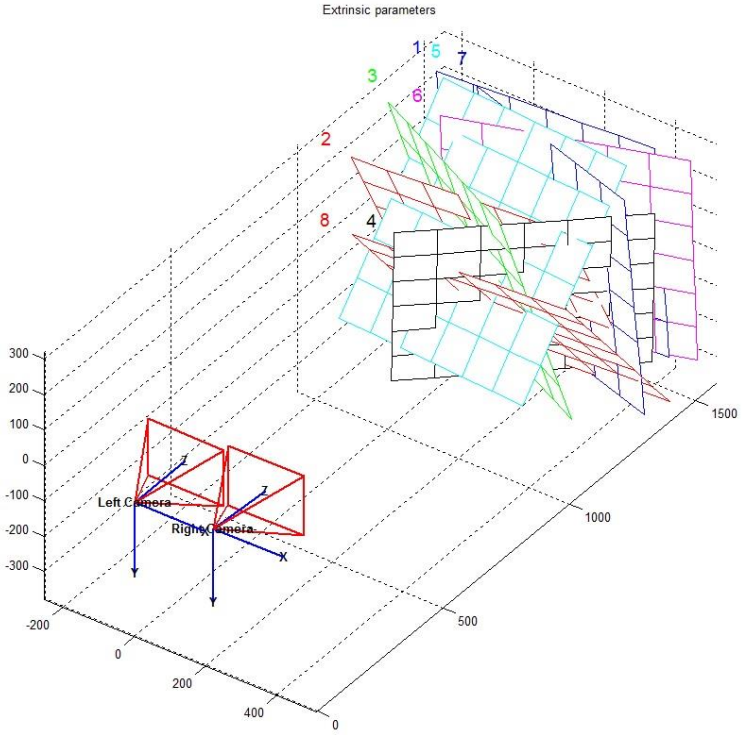


Figura: 91 Representación Visual de los parámetros extrínsecos de las cámaras (Stereo Algorithm)

Fuente: Elaboración propia

a. En la Figura: 91 se observa no solo la representación Visual de los parámetros extrínsecos de las cámaras, si no también el como se movió rotó y trasladó el Checkerboard, para realizar los testeos y obtener los parámetros de la cámara, reflejando efectivamente los movimientos realizados de los 8 snapshots tomados con cada cámara.

CONCLUSIONES

1. Se aceptan las hipótesis específicas planteadas en el capítulo I para el desarrollo de este sistema de Visión Artificial Humanoide.
2. Para la correcta realización de la segmentación localización y reconocimiento de objetos en las imágenes digitales tomadas, se aplicó algoritmos de erosión, dilatación, apertura, cerradura, se realizó un correcto pre-procesamiento de la imagen, como también se aplicaron filtros frecuenciales y filtros de tamaño para finalmente obtener una imagen limpia, con sólo el objeto deseado a identificar para posterior manipulación.
3. Se realizó de manera efectiva el reconocimiento efectivo de Formas y patrones de objetos mediante el uso de la regla de la cadena y algoritmos neuronales multicapa de estructura 5-4-3
4. Para la implementación de algoritmos de Estereoscopía se tuvo que desarrollar algoritmos para la calibración de la cámara, y una vez realizado, se procedió a aplicar geometría respecto al punto de referencia, las cámaras y el eje focal para poder percibir la profundidad la cual se encontraba el objeto (En este caso, el objeto se simuló con un tablero de ajedrez ubicado a 79cm de distancia respecto de las cámaras)

BIBLIOGRAFÍA

[1] Guevara A., Jornales F., Alpha V.y Alvarez F. (2006) *Técnicas y Algoritmos Básicos de Visión Artificial.* Universidad de la Roja. España

[2] Yang M, Kriegman D.J. (2003). *Detecting Faces in Images: A Survey, IEEE Transactions on Pattern Analysis and Machine Intelligence*. NY Magazine Vol. 24 & 83. USA

[3] Sahoo P.K.(1988) *A survey of Thresholding Techniques. Computer Vision Graphics and Image Processing*. Ed. Morgan. England

[4] Kunihito K., Masayuki S. & Kazuhiko Y. (2010). *An Implementation of humanoid Vision - Analysis of Eye Movement and Implementation to Robot*. Gifu University. Japan

[5] Ramesh J., Rangachar K., Gilmore B. Schunck (1995). *Machine Vision*. McGraw Hill. USA

[6] Grandon T.(2011) *Artificial Vision in the Nao Humanoid Robot.* Tesis (Master in Artificial Intelligence). Catalan – Rovira i Virgii University, Department of Computer Science and Mathematics. Spain

[7] Carnajal M. (2010). *Robótica y Visión Artificia*l. [Diapositivas] Universidad de Alicante. Spain

[8] Dr. Wolf M.(2002) *Quantitative Image Analysis Methods and Limitations. Biomedicalproducts*. Ed. Springer. USA

[9] Moore D. (2003) *A real-world system for human motion detection and tracking*. California Institute of Technology. USA

[10] Jackson O. Söderkvist O. (2004) *Computer Vision Classification of Leaves from Swedish Trees* (Master in Science). Linköping University. Computer Vision Laboratory (CVL). Suiza

[11] Thessaloniki Y.(1998) *Digital Image Processing Fundamentals*. Ed. Morgan England

[12] Rossemont D.(1998) *Using Segmentation to Verify Object Hypotheses*. Toyota Technological Institute. USA

[13] Balta D. (2012). *Reconocimiento de Objetos en 3D Utilizando Sensores de Visión y Profundidad Bajo Coste*. Universidad de Zaragoza. Departamento de Informática e Ingeniería de Sistemas. España

[14] Garrick D. (2013) *Principles of Artificial Neural Networks*. University of Illinois. USA

[15] Morse C. (2000). *An Introduction to Practical Neural Networks and genetic Algorithms for engineers and Scientist*. Ed. Pshrenon. Irlanda

[16] Wolftruck M., McCulloch W.(1987). *How we know universal to recurrent neural networks. Physical Review Letters*. Ed. McGraw Hill. USA

[17] Bishop M. (2000). *Neural Networks for Patter Recognition.* Department of Computer Science and Applied Mathematics. Aston University. Birmingham. England

[18] Parker D.(1985) *Learning Logic* [Technical report] MIT University. USA

[19] Wasserman P., Van N. (1993) *Advanced methods in neural Computing*. McGraw Hill. USA

[20] Karper D.(2005) *A Brief Introduction to Neural Networks*. Ed. Düssel. Germany

[21] Gibsander K. Learned-Miller.(2011) *Introduction to Computer Vision*. University of Massachusetts Science. USA

[22] Pirson G. (2000) *Depth Perception in Computer Graphics*. Trinity College University of Cambridge. [Phd Thesis]. USA

ANEXOS

ANEXO 1: Algoritmo Filtro Pasa Bajo

```
clear all, close all, clc

i=0;

j=0;

%Centered image in 0,0 to make it symmetric

for u=-1:.05:1

   i=i+1;

   for v=-1:.05:1

      j=j+1;

      D(i,j)=sqrt(u^2+v^2);

      if D(i,j)<=.5 %where .5 is cutoff frequency

         H(i,j)=1;

      else

         H(i,j)=0;

      end

   end

   j=0;

end

H1=fftshift(H);
```

```
subplot(121);imshow(H);

subplot(122),surf(H)
```

ANEXO 2: Algoritmo Filtro Gaussiano con ventana n=7

```
%% Gauss Simple Filter with n=7 window (matrix) & sigma^2=2

clear all, close all, clc

jj=0;

ii=0;

for i=-3:1:3%Creating a n=7 window Gaussian

   ii=ii+1;

   for j=-3:1:3

      jj=jj+1;

      G(ii,jj)=exp(-(i^2+j^2)/(2*2));

   end

   jj=0;

end

ii=0;

%here G are real Values

G1=round(G);%here G1 are “int” Values of the Gaussian

subplot(121),imshow(G)

subplot(122),surf(-3:3,-3:3,G)


k=1/G(7,7); %we analize k1 value to make it then integer Gaussian

k1=round(k);
```

```
for i=-3:1:3%Creating a n=7 window with k1 value to make Gaussian integer
    ii=ii+1;
    for j=-3:1:3
        jj=jj+1;
        G(ii,jj)=k1*exp(-(i^2+j^2)/(2*2));
    end
    jj=0;
end


G1=round(G);
subplot(121),imshow(G)
subplot(122),surf(-3:3,-3:3,G)
```

ANEXO 3 : Algoritmo Filtro frecuencial pasa-alto

```
clear all, close all, clc

i=0;

j=0;

%Centered image in 0,0 to make it symmetric

for u=-1:.05:1

   i=i+1;

   for v=-1:.05:1

      j=j+1;

      D(i,j)=sqrt(u^2+v^2);

      if D(i,j)>.5 %where .5 is cutoff frequency

         H(i,j)=1;

      else

         H(i,j)=0;

      end

   end

   j=0;

end

H1=fftshift(H);

subplot(121);imshow(H);
```

```
subplot(122),surf(H)
```

ANEXO 4: Algoritmo Morofológico - Erosión

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%
%                  Artificial Vision System                        %%
% ________________________________________________________________%%
%                                          %%
%___________________________TASK________________________________%%
%                                          %%
% Here we're gointo to create a function in order to EROSION an image   %%
%______________________VARIABLES_______________________________%%
% A: Original Image                                  %%
% EE:structured Element                                 %%
% Im_erode:Eroded Image                                  %%
%_______________________________________________________________%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%_EROSION_FUNCTION_%%%%%%%

function [Im_erode]=jon_erode(A,EE)

%----------------------------- Paso 1 -----------------------------------
%------- 1.-Introducir imagen y Elemento Estructurante --------------

[Ar,Ac]=size(A);%Tamaño de imagen original
[sEEi,sEEj]=size(EE);%Tamaño EE
%------------------------------ Fin 1 ---------------------------------

%----------------------------- Paso 2 -----------------------------------
%------- 2.-Algoritmo Hallar el Origen del EE -------------

oEE=floor((size(EE)+1)/2);%hallando origen del EE
oEEi=oEE(1);%Origen de elemento en filas
oEEj=oEE(2);%Origen de elemento en filas
```

```
fprintf('Elemento Estructurante\n')
EE
%------------------------------ Fin 2 --------------------------------

%---------------------------- Paso 4 ----------------------------------
%---- 4.- Crear matriz Aumentada "Au" con matriz imagen A en el medio ----

if (oEEi-1 ~= 0) || (oEEj-1 ~=0) || (sEEi-oEEi~=0) || (sEEj-oEEj~=0)
%el simbolo "~=" siginifica "DIFERENTE DE"

   fup=oEEi-1;%filas aumentar ARRIBA
   fdown=sEEi-oEEi;%filas aumentar ABAJO
   cleft=oEEj-1;%columnas aumentar IZQUIERDA
   cright=sEEj-oEEj;%columnas aumentar DERECHA

   Au_r=(Ar+fup+fdown);%tamaño de nueva matriz en filas
   Au_c=(Ac+cleft+cright);%tamaño de nueva matriz en columnas
   Au(1:Au_r ,1:Au_c )=1;%creamos matriz Au aumentada y la rellenamos matriz Au de full
ceros
end
%------------------------------ Fin 4 --------------------------------

%---------------------------- Paso 5 ----------------------------------
% 5.-Rellenando matriz Au con matriz imagen A para despues hacer morfología

   Au_rr=(Ar+(oEEi-1));%valor final en FILAS a traslapar matriz A en
                %nueva matriz Au
   Au_cc=(Ac+(oEEj-1));%valor final en COLUMNAS a traslapar matriz A en
                %nueva matriz Au
   Au(oEEi:Au_rr,oEEj:Au_cc)=A;%Crea matriz aumentada con ceros
%    alrededor y la imagen A la pone en el medio como coresponde
% Ex: Au=[0 0 0
%         0 A 0   } Según sea el orden de la matriz imagen A
%         0 0 0]
Au=logical(Au);
A=logical(A);
EE=logical(EE);
%------------------------------ Fin 5 --------------------------------
```

```
%----------------------------- Paso 6 ----------------------------------
%--- 6.- Creando ID para Rellenar con nuevos valores según morfología---

Im_erode=zeros(Ar,Ac);%creamos matriz a rellenar con nuevos valores



sum_EE=sum(sum(EE));%Suma de EE
for i=1:1:Au_r-sEEi+1%imagen aumentada, filas
   %Au_r-sEEi->Si Au_r 2 columnas y sEEi=1 entonces solo tenemos que mover el EE
   %1 vez hacia abajo, para que siga encajando en nuestra
   %matriz imagen nueva
   for j=1:1:Au_c-sEEj+1%imagen aumentada columnas
      %Au_c-sEEj->Si Au_c 9 columnas y sEEj=4 entonces solo tenemos que mover el EE
      %5 veces hacia la derecha, para que siga encajando en nuestra
      %matriz imagen nueva
      %Creando matriz H a comparar, H es la máscara a comparar con EE
      H=Au(i:i+(sEEi-1),j:j+(sEEj-1));%Parte de imagen a comparar
      %del mismo tamaño que el EE. y lo guardamos en H

      [Hr,Hc]=size(H);%Tamaño de Máscara a comparar del mismo tamaño que
                  %Elemento Estructural "EE".
      oH=floor((size(H)+1)/2);
      %----------- Algoritmo Principal Erosion --------------------
      suma=0;%Acumulador de SUMA en algoritmo erosión, más abajo
      for k=1:1:Hr
         for m=1:1:Hc
            if (H(k,m) * EE(k,m))==1
               suma=suma+1;
            end
         end
      end
      if suma==sum_EE
         Im_erode(i,j)=1;
      else
         Im_erode(i,j)=0;
      end
   end
end
  %----------------------------- Fin 6 ---------------------------------
```

```
end % END Function Erosion
```

ANEXO 5: Algoritmo morofológico - Dilatación

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%
%              Artificial Vision System                    %%
% ______________________________________________________________%%
%                                            %%
%_________________________________TASK__________________________________%%
%                                            %%
% Here we're gointo to create a function in order to DILATE an image   %%
%___________________________VARIABLES______________________________%%
% A: Original Image                              %%
% EE:structured Element                           %%
% EE_rfl:Reflexive Structured Element                %%
%______________________________________________________________%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%_DILATE_FUNCTION_%%%%%%%%%%%%%

function [ID]=jon_dilate(A,EE)
%---------------------------- Paso 1 ----------------------------------
%------- 1.-Introducir imagen y Elemento Estructurante -------------
[Ar,Ac]=size(A);%Tamaño de imagen original
[sEEi,sEEj]=size(EE);%Tamaño EE
%----------------------------- Fin 1 ---------------------------------


%---------------------------- Paso 2 ----------------------------------
%------- 2.-Pasos para Reflexión de EE -------------

% 2.1.- Algoritmo Hallar el Origen del EE:
oEE=floor((size(EE)+1)/2);%hallando origen del EE
% 2.2.- Algoritmo Reflexion del EE:
for i=1:1:sEEi
   for j=1:1:sEEj
      jrfl=sEEj-(j-1);%jref=ultimaColumna-(ActualColumna-1raColumna)
      irfl=sEEi-(i-1);%iref=ultimaFila-(ActualFila-1raFila)
      EE_rfl(irfl,jrfl)=EE(i,j);%Creamos matriz Reflexiva R del
                      %elemento estructurante EE3
```

```
    end
end
fprintf('Elemento Estructurante\n')
EE
fprintf('Elemento Estructurante Reflexivo\n')
EE_rfl % Matriz reflexiva
%------------------------------ Fin 2 --------------------------------

%---------------------------- Paso 3 ----------------------------------
%------- 3.- Tamaño y Origen de EE reflexivo (EE_rfl) -----------

[sEEi_rfl sEEj_rfl]=size(EE_rfl);%Tamaño EE Reflexivo
oEE_rfl=round((size(EE_rfl)+1)/2);%Origen de Elemento Reflexivo
oEEi_rfl=oEE_rfl(1);%Origen de elemento reflexivo en filas
oEEj_rfl=oEE_rfl(2);%Origen de elemento reflexivo en columnas
%------------------------------ Fin 3 --------------------------------

%---------------------------- Paso 4 ----------------------------------
%---- 4.- Crear matriz Aumentada "Au" con matriz imagen A en el medio ----

if (oEEi_rfl-1 ~= 0) || (oEEj_rfl-1 ~=0) || (sEEi_rfl-oEEi_rfl~=0) || (sEEj_rfl-oEEj_rfl~=0)
%el simbolo "~=" siginifica "DIFERENTE DE"

    fup=oEEi_rfl-1;%filas aumentar ARRIBA
    fdown=sEEi_rfl-oEEi_rfl;%filas aumentar ABAJO
    cleft=oEEj_rfl-1;%columnas aumentar IZQUIERDA
    cright=sEEj_rfl-oEEj_rfl;%columnas aumentar DERECHA

    Au_r=(Ar+fup+fdown);%tamaño de nueva matriz en filas
    Au_c=(Ac+cleft+cright);%tamaño de nueva matriz en columnas
    Au(1:Au_r ,1:Au_c )=0;%creamos matriz Au aumentada y la rellenamos matriz Au de full
ceros
end
%------------------------------ Fin 4 --------------------------------

%---------------------------- Paso 5 ----------------------------------
% 5.-Rellenando matriz Au con matriz imagen A para despues hacer morfología
```

```
   Au_rr=(Ar+(oEEi_rfl-1));%valor final en FILAS a traslapar matriz A en
                %nueva matriz Au
   Au_cc=(Ac+(oEEj_rfl-1));%valor final en COLUMNAS a traslapar matriz A en
                %nueva matriz Au
   Au(oEEi_rfl:Au_rr,oEEj_rfl:Au_cc)=A;%Crea matriz aumentada con ceros
%     alrededor y la imagen A la pone en el medio como coresponde
% Ex: Au=[0 0 0
%         0 A 0   } Según sea el orden de la matriz imagen A
%         0 0 0]
Au=logical(Au);
A=logical(A);
EE=logical(EE);
EE_rfl=logical(EE_rfl);
%------------------------------ Fin 5 --------------------------------


%------------------------------ Paso 6 ---------------------------------
%--- 6.- Creando ID para Rellenar con nuevos valores según morfología---

ID=zeros(Ar,Ac);%creamos matriz a rellenar con nuevos valores
 for i=1:1:Au_r-sEEi_rfl+1%imagen aumentada, filas
   %Rr-f->Si Rr 2 columnas y f=1 entonces solo tenemos que mover el EE
   %1 vez hacia abajo, para que siga encajando en nuestra
   %matriz imagen nueva
   for j=1:1:Au_c-sEEj_rfl+1%imagen aumentada columnas
      %Rc-c->Si Rc 9 columnas y f=4 entonces solo tenemos que mover el EE
      %5 veces hacia la derecha, para que siga encajando en nuestra
      %matriz imagen nueva
      %Creando matriz H a comparar, H es la máscara a comparar con EE_rfl
      H=Au(i:i+(sEEi_rfl-1),j:j+(sEEj_rfl-1));%Parte de imagen a comparar
      %del mismo tamaño que el EE. y lo guardamos en H

      [Hr,Hc]=size(H);%Tamaño de
      ID(i,j)=max(max(EE_rfl.*H));
   end
end
%------------------------------ Fin 6 --------------------------------
end %END Function Dilate
```

ANEXO 6: Algoritmo Morfológico - Apertura

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%
%                Artificial Vision System                    %%
% ___________________________________________________________%%
%                                                  %%
%____________________________TASK____________________________%%
%                                                  %%
% Here we're gointo to create a function in order to OPEN an image   %%
%________________________VARIABLES___________________________%%
% A: Original Image                                 %%
% EE:structured Element                               %%
% Im_erode:Eroded Image                                %%
%____________________________________________________________%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%_IMAGE_OPEN_FUNCTION_%%%%%%%%%%%%%%%%

function [Im_open]=jon_open(A,EE)

%---------------------------- Paso 1 ----------------------------------
%------- 1.-Introducir imagen y Elemento Estructurante -------------

[Ar,Ac]=size(A);%Tamaño de imagen original
[sEEi,sEEj]=size(EE);%Tamaño EE
%----------------------------- Fin 1 --------------------------------

%---------------------------- Paso 2 ----------------------------------
%------- 2.-Algoritmo Hallar el Origen del EE -------------

oEE=floor((size(EE)+1)/2);%hallando origen del EE
oEEi=oEE(1);%Origen de elemento en filas
oEEj=oEE(2);%Origen de elemento en filas
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%----------------------------- Fin 2 --------------------------------
```

```
%----------------------------- Paso 4 ----------------------------------
%---- 4.- Crear matriz Aumentada "Au" con matriz imagen A en el medio ----

if (oEEi-1 ~= 0) || (oEEj-1 ~=0) || (sEEi-oEEi~=0) || (sEEj-oEEj~=0)
%el simbolo "~=" siginifica "DIFERENTE DE"

   fup=oEEi-1;%filas aumentar ARRIBA
   fdown=sEEi-oEEi;%filas aumentar ABAJO
   cleft=oEEj-1;%columnas aumentar IZQUIERDA
   cright=sEEj-oEEj;%columnas aumentar DERECHA

   Au_r=(Ar+fup+fdown);%tamaño de nueva matriz en filas
   Au_c=(Ac+cleft+cright);%tamaño de nueva matriz en columnas
   Au(1:Au_r ,1:Au_c )=1;%creamos matriz Au aumentada y la rellenamos matriz Au de full
ceros
end
%----------------------------- Fin 4 ----------------------------------

%----------------------------- Paso 5 ----------------------------------
% 5.-Rellenando matriz Au con matriz imagen A para despues hacer morfología

   Au_rr=(Ar+(oEEi-1));%valor final en FILAS a traslapar matriz A en
                %nueva matriz Au
   Au_cc=(Ac+(oEEj-1));%valor final en COLUMNAS a traslapar matriz A en
                %nueva matriz Au
   Au(oEEi:Au_rr,oEEj:Au_cc)=A;%Crea matriz aumentada con ceros
%     alrededor y la imagen A la pone en el medio como coresponde

Au=logical(Au);
A=logical(A);
EE=logical(EE);
%----------------------------- Fin 5 ----------------------------------

%----------------------------- Paso 6 ----------------------------------

%--- 6.- Creando ID para Rellenar con nuevos valores según morfología---
```

```
Im_erode=zeros(Ar,Ac);%creamos matriz a rellenar con nuevos valores



sum_EE=sum(sum(EE));%Suma de EE
for i=1:1:Au_r-sEEi+1
   for j=1:1:Au_c-sEEj+1
      H=Au(i:i+(sEEi-1),j:j+(sEEj-1));%Parte de imagen a comparar
      %del mismo tamaño que el EE. y lo guardamos en H

      [Hr,Hc]=size(H);
      oH=floor((size(H)+1)/2);
      %---------- Algoritmo Principal Erosion --------------------
      suma=0;%Acumulador de SUMA en algoritmo erosión, más abajo
      for k=1:1:Hr
         for m=1:1:Hc

            if (H(k,m) * EE(k,m))==1
               suma=suma+1;
            end
         end
      end
      if suma==sum_EE
         Im_erode(i,j)=1;
      else
         Im_erode(i,j)=0;
      end
   end
end

%---------------------------- Fin 6 -------------------------------
```

ANEXO 7: Algoritmo Morofológico - Dilatación (Continuación – Reflexion de elemento estructurante)

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%_DILATE_FUNCTION_%%%%%%%%%%%%%%%%%%%%%%

%------- 2.-Pasos para Reflexión de EE -------------

% 2.2.- Algoritmo Reflexion del EE:
for i=1:1:sEEi
   for j=1:1:sEEj
      jrfl=sEEj-(j-1);%jref=ultimaColumna-(ActualColumna-1raColumna)
      irfl=sEEi-(i-1);%iref=ultimaFila-(ActualFila-1raFila)
      EE_rfl(irfl,jrfl)=EE(i,j);%Creamos matriz Reflexiva R del
                        %elemento estructurante EE3
   end
end

%------------------------------ Fin 2 --------------------------------

%----------------------------- Paso 3 ----------------------------------
%------- 3.- Tamaño y Origen de EE reflexivo (EE_rfl) -----------

[sEEi_rfl sEEj_rfl]=size(EE_rfl);%Tamaño EE Reflexivo
oEE_rfl=round((size(EE_rfl)+1)/2);%Origen de Elemento Reflexivo
oEEi_rfl=oEE_rfl(1);%Origen de elemento reflexivo en filas
oEEj_rfl=oEE_rfl(2);%Origen de elemento reflexivo en columnas
%------------------------------ Fin 3 --------------------------------

%----------------------------- Paso 4 ----------------------------------
%---- 4.- Crear matriz Aumentada "Au" con matriz imagen A en el medio ---

   Au(1:Au_r ,1:Au_c )=0;%creamos matriz Au aumentada y la rellenamos matriz Au de full ceros

%------------------------------ Fin 4 --------------------------------
```

```
%----------------------------- Paso 5 ----------------------------------
% 5.-Rellenando matriz Au con matriz imagen A para despues hacer morfología

   Au_rr=(Ar+(oEEi_rfl-1));%valor final en FILAS a traslapar matriz A en
                %nueva matriz Au
   Au_cc=(Ac+(oEEj_rfl-1));%valor final en COLUMNAS a traslapar matriz A en
                %nueva matriz Au
   Au(oEEi_rfl:Au_rr,oEEj_rfl:Au_cc)=Im_erode;%Crea matriz aumentada con ceros
%     alrededor y la imagen A la pone en el medio como coresponde

Au=logical(Au);
A=logical(A);
EE=logical(EE);
EE_rfl=logical(EE_rfl);
%----------------------------- Fin 5 ----------------------------------

%----------------------------- Paso 6 ----------------------------------
%--- 6.- Creando ID para Rellenar con nuevos valores según morfología---

Im_open=zeros(Ar,Ac);%creamos matriz a rellenar con nuevos valores

for i=1:1:Au_r-sEEi_rfl+1
   for j=1:1:Au_c-sEEj_rfl+1
      H=Au(i:i+(sEEi_rfl-1),j:j+(sEEj_rfl-1));

      [Hr,Hc]=size(H);
      Im_open(i,j)=max(max(EE_rfl.*H));
   end
end
%----------------------------- Fin 6 ----------------------------------

end %END Function OPEN
```

ANEXO 8: Algoritmo Morfológico - Cerradura

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%
%                  Artificial Vision System                          %%
% _________________________________________________________________%%
%                                                      %%
%____________________________TASK__________________________________%%
%                                                      %%
% Here we're gointo to create a function in order to CLOSE an image    %%
%_______________________VARIABLES_______________________________%%
% A: Original Image                                    %%
% EE:structured Element                                    %%
% EE_rfl:Reflexive Structured Element                          %%
%__________________________________________________________________%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%_IMAGE_CLOSE_FUNCTION_%%%%%%%%%%%%%%%%

function [Im_close]=jon_close(A,EE)

%--------- Here we're gonna execute the Dilate Algorithm ----------------
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%---------------------------- Paso 1 ----------------------------------
%------- 1.-Introducir imagen y Elemento Estructurante --------------
[Ar,Ac]=size(A);%Tamaño de imagen original
[sEEi,sEEj]=size(EE);%Tamaño EE
%----------------------------- Fin 1 ---------------------------------


%---------------------------- Paso 2 ----------------------------------
%------- 2.-Pasos para Reflexión de EE -------------

% 2.1.- Algoritmo Hallar el Origen del EE:
oEE=floor((size(EE)+1)/2);%hallando origen del EE
% 2.2.- Algoritmo Reflexion del EE:
for i=1:1:sEEi
    for j=1:1:sEEj
        jrfl=sEEj-(j-1);%jref=ultimaColumna-(ActualColumna-1raColumna)
        irfl=sEEi-(i-1);%iref=ultimaFila-(ActualFila-1raFila)
```

```
        EE_rfl(irfl,jrfl)=EE(i,j);%Creamos matriz Reflexiva R del
                                %elemento estructurante EE3
    end
end
%------------------------------ Fin 2 ---------------------------------

%------------------------------ Paso 3 ---------------------------------
%------- 3.- Tamaño y Origen de EE reflexivo (EE_rfl) -----------

[sEEi_rfl sEEj_rfl]=size(EE_rfl);%Tamaño EE Reflexivo
oEE_rfl=round((size(EE_rfl)+1)/2);%Origen de Elemento Reflexivo
oEEi_rfl=oEE_rfl(1);%Origen de elemento reflexivo en filas
oEEj_rfl=oEE_rfl(2);%Origen de elemento reflexivo en columnas
%------------------------------ Fin 3 ---------------------------------

%------------------------------ Paso 4 ---------------------------------
%---- 4.- Crear matriz Aumentada "Au" con matriz imagen A en el medio --

if (oEEi_rfl-1 ~= 0) || (oEEj_rfl-1 ~=0) || (sEEi_rfl-oEEi_rfl~=0) || (sEEj_rfl-oEEj_rfl~=0)
%el simbolo "~=" siginifica "DIFERENTE DE"

    fup=oEEi_rfl-1;%filas aumentar ARRIBA
    fdown=sEEi_rfl-oEEi_rfl;%filas aumentar ABAJO
    cleft=oEEj_rfl-1;%columnas aumentar IZQUIERDA
    cright=sEEj_rfl-oEEj_rfl;%columnas aumentar DERECHA

    Au_r=(Ar+fup+fdown);%tamaño de nueva matriz en filas
    Au_c=(Ac+cleft+cright);%tamaño de nueva matriz en columnas
    Au(1:Au_r ,1:Au_c )=0;%creamos matriz Au aumentada y la rellenamos matriz Au de full
ceros
end
%------------------------------ Fin 4 ---------------------------------

%------------------------------ Paso 5 ---------------------------------
% 5.-Rellenando matriz Au con matriz imagen A para despues hacer morfología

    Au_rr=(Ar+(oEEi_rfl-1));
    Au_cc=(Ac+(oEEj_rfl-1));
    Au(oEEi_rfl:Au_rr,oEEj_rfl:Au_cc)=A;
```

```
Au=logical(Au);
A=logical(A);
EE=logical(EE);
EE_rfl=logical(EE_rfl);
%------------------------------ Fin 5 --------------------------------

%------------------------------ Paso 6 --------------------------------
%--- 6.- Creando ID para Rellenar con nuevos valores según morfología---

ID=zeros(Ar,Ac);%creamos matriz a rellenar con nuevos valores

for i=1:1:Au_r-sEEi_rfl+1
   for j=1:1:Au_c-sEEj_rfl+1
      H=Au(i:i+(sEEi_rfl-1),j:j+(sEEj_rfl-1));

      [Hr,Hc]=size(H);
      ID(i,j)=max(max(EE_rfl.*H));
   end
end
%------------------------------ Fin 6 Dilate ------------------------------

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%---------------Here we're gonna execute the Erode algorithm -----------

%------------------------------ Paso 1 --------------------------------
%------- 1.-Introducir imagen y Elemento Estructurante --------------

%------- 2.-Algoritmo Hallar el Origen del EE -------------

oEE=floor((size(EE)+1)/2);%hallando origen del EE
oEEi=oEE(1);%Origen de elemento en filas
oEEj=oEE(2);%Origen de elemento en filas
%------------------------------ Fin 2 --------------------------------

%------------------------------ Paso 4 --------------------------------
%---- 4.- Crear matriz Aumentada "Au" con matriz imagen A en el medio --
 Au(1:Au_r ,1:Au_c )=1;
%------------------------------ Fin 4 --------------------------------
```

```
 %----------------------------- Paso 5 ----------------------------------
% 5.-Rellenando matriz Au con matriz imagen A para despues hacer morfología

   Au_rr=(Ar+(oEEi-1));%valor final en FILAS a traslapar matriz A en
                %nueva matriz Au
   Au_cc=(Ac+(oEEj-1));%valor final en COLUMNAS a traslapar matriz A en
                %nueva matriz Au
   Au(oEEi:Au_rr,oEEj:Au_cc)=ID;
Au=logical(Au);
A=logical(ID);
EE=logical(EE);
%----------------------------- Fin 5 ---------------------------------

%----------------------------- Paso 6 ----------------------------------
%--- 6.- Creando ID para Rellenar con nuevos valores según morfología---

Im_close=zeros(Ar,Ac);%creamos matriz a rellenar con nuevos valores
 sum_EE=sum(sum(EE));%Suma de EE
for i=1:1:Au_r-sEEi+1

   for j=1:1:Au_c-sEEj+1
      H=Au(i:i+(sEEi-1),j:j+(sEEj-1));

      [Hr,Hc]=size(H);
      oH=floor((size(H)+1)/2);
      %----------- Algoritmo Principal Erosión --------------------
      suma=0;%Acumulador de SUMA en algoritmo erosión, más abajo
      for k=1:1:Hr
         for m=1:1:Hc

            if (H(k,m) * EE(k,m))==1
               suma=suma+1;
            end
         end
      end
      if suma==sum_EE
         Im_close(i,j)=1;
      else
         Im_close(i,j)=0;
```

```
        end
    end
end

%------------------------------ Fin 6 --------------------------------
end % END Function jon_close
```

ANEXO 9: Proceso de Corte de imagen

```
function [ Image_cut ] = jon_imcut( Image,cutsize )
%UNTITLED2 Summary of this function goes here
%   Detailed explanation goes here
% here we are going to just cut an image by "cutsize" part on vertical axis
% or rows
%if the image has f rows and c columns, let's say we just want to plot
%f/4 or f/5.5 part of the whole Image so cutsize=4 or cusize=5.5
% jon_imcut( Image,cutsize )

[f,c]=size(Image);
f1=round(f/cutsize);
Image_cut = (Image(f1:f,1:c));

end
```

ANEXO 10: Algortimo de filtrado especial de una imagen aplicando un kernel

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%
%                                            Artificial     Vision     System
%%_____________________________________________________________________%%
%                                          %%
%______________________________TASK_____________________________________%
%
%                                          %%
% Here we're gointo to create a function in order to FILTER an image    %%
%_________________________VARIABLES______________________________________%
%
% Image: Original Image                              %%
% filter:Filter kernel                            %%
% Im_filter:filtered Image                          %%
%_______________________________________________________________________%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [ Im_filter ] = jon_imfilter( Image,filter )
%UNTITLED2 Summary of this function goes here
%   Detailed explanation goes here
% Syntax: jon_imfilter( Image,filter )
%%%Algoritmo_Filtrado----
[Ar Ac]=size(Image);%Image size
%------------------------------------------|
[sEEi sEEj]=size(filter);%Filter size

%---------------------------- Step 2 ----------------------------------
%to create a new Increased Matrix (Au) according to Filter mask
% we need to know it's origin
%------- 2.-Algoritmo Hallar el Origen del Filtro -------------
oEE=floor((size(filter)+1)/2);%Origin of Filter
oEEi=oEE(1);
oEEj=oEE(2);
%---------------------------- Step 3 ----------------------------------
```

```
% 4.- Creating INCREASED MATRIX "Au" with A original image in the middle ----

if (oEEi-1 ~= 0) || (oEEj-1 ~=0) || (sEEi-oEEi~=0) || (sEEj-oEEj~=0)
%el simbolo "~=" siginifica "DIFERENTE DE"

   fup=oEEi-1;%filas aumentar ARRIBA
   fdown=sEEi-oEEi;%filas aumentar ABAJO
   cleft=oEEj-1;%columnas aumentar IZQUIERDA
   cright=sEEj-oEEj;%columnas aumentar DERECHA

   Au_r=(Ar+fup+fdown);%tamaño de nueva matriz en filas
   Au_c=(Ac+cleft+cright);%tamaño de nueva matriz en columnas
   Au(1:Au_r ,1:Au_c )=0;%creamos matriz Au aumentada y la rellenamos matriz Au de full
ceros
end
%---------------------------- Step 4 ----------------------------------
% 5.-Rellenando matriz Au con matriz imagen A para despues hacer morfología

   Au_rr=(Ar+(oEEi-1));%valor final en FILAS a traslapar matriz A en
                %nueva matriz Au
   Au_cc=(Ac+(oEEj-1));%valor final en COLUMNAS a traslapar matriz A en
                %nueva matriz Au
   Au(oEEi:Au_rr,oEEj:Au_cc)=Image;%Crea matriz aumentada con ceros
%     alrededor y la imagen A la pone en el medio como coresponde

%luego---

ID=zeros(Ar,Ac);%Creamos matriz para guardar nueva imagen con filtro aplicado
for i=1:1:Au_r-sEEi+1%lo q tiene que recorrer la matriz aumentada en filas
   for j=1:1:Au_c-sEEj+1%lo q tiene que recorrer la matriz aumentada en columnas
      H=Au(i:i+(sEEi-1),j:j+(sEEj-1));%toma la imagen del mismo tamaño de máscara
      [Hr,Hc]=size(H);%Tamaño de Máscara a comparar
      oH=floor((size(H)+1)/2);%origen de imagen a comparar H

      sum_filter=0;%Filter SUM accumulator
       %este algoritmo es de Reflexion, por el Teorema de la convolucion
       %tenemos que invertir 180° o Reflexionar la imagen a comparar
       %en este caso como filtro 3x3 entonces comparamos H de 3x3
      for m=1:1:Hr
```

```
        for n=1:1:Hc
        % Reflexion de Imagen a comparar con estas cadenas FOR m y n

            sum_filter=H(m,n)*filter(m,n)+sum_filter;

        end
      end
      Im_filter(i,j)=sum_filter;%actualmente este punto bendría ha ser el origen mas NO el
ORIGEN DEL REFLEXIVO
   end
end
end %function jon_imfilter()
```

ANEXO 11: Algoritmo Filtro Espacial Gaussiano

```
function [ spatial_gauss,X,Y ] = jon_sgauss( window,sigma1 )
%UNTITLED2 Summary of this function goes here
%   Detailed explanation goes here
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%
% This is my Spatial Gaussian Function
% [ spatial_gauss,X,Y ] = jon_sgauss( window,sigma1 )
% where Gauss -> The function value (output)
% X-> window in rows (output)
% Y-> window in columns (output)
% sigma1=2*sigma^2-> sigma is deviation of the gaussian (INPUT)
% window -> n of Gaussian (INPUT) ex. 15x15,7x7,3x3 Matrix with Median=0
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% example of program          %
% [G,X,Y]=jon_sgauss(15,1.2); %
% figure(1)                   %
% subplot(121),imshow(G)      %
% subplot(122),surf(X,Y,G)    %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
contx=0;% Here starts Gauss
conty=0;

for x=-window:1:window %median in zero
   contx=contx+1;
   X(1,contx)=x;
   for y=-window:1:window %median in zero
      conty=conty+1;
      Y(1,conty)=y;
     spatial_gauss(contx,conty)=exp(-((x^2+y^2))/sigma1);
   end
   conty=0;
end
contx=0;

end %end Gauss function 'jon_spgauss()'
```

ANEXO 12 Algoritmo Filtro Frecuencial de Gauss

```
function [ fft2_gauss,shift_freq_gauss,padded_image ] = jon_fgauss( Image,spatial_gauss )
%UNTITLED3 Summary of this function goes here
%   Detailed explanation goes here
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%
%   This is my Gaussian Function in frequency Domain
% -----------------------------------------------------------------------------
%   [ fft2_gauss,shift_freq_gauss,padded_image ] = jon_fgauss( Image,spatial_gauss )
%   where:
%   fft2_gauss -> My gaussian function in FREQUENCY DOMAIN (OUTPUT)
%   shift_freq_gauss -> shift of fft2_gauss (OUTPUT)
%   padded_image -> padding algorithm of Image (OUTPUT)
%   Image -> Image of size MxN on which we want to Pad our spatial_gauss (INPUT)
%   spatial_gauss -> My gaussian function in SPATIAL DOMAIN (INPUT)
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%
%   example of program                          %
%   [F,SF,PAD]=jon_fgauss(I,G);                 %
%   figure                                      %
%   subplot(221);imshow(SF)                     %
%                                               %
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%
%
spatial_gauss(size(Image,1),size(Image,2))=0;%Padding algorithm
padded_image=spatial_gauss;
fft2_gauss=fft2(spatial_gauss);%Fourier of Gaussian
shift_freq_gauss=log(abs(fftshift(fft2_gauss)));%Fourier de gauss padded (Gauss en frecuencia)

end%end Gauss function 'jon_fpgauss()'
```

ANEXO 13: Algoritmo de convolución entre imagen y filtro aplicado en frecuencia

```
function  freq_filtered_image  = jon_sym_fconv( fourier_image,fourier_filter,win )
%UNTITLED Summary of this function goes here
%   Detailed explanation goes here
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% freq_filtered_image  = jon_sym_fconv( fourier_image,fourier_filter,win )
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This function TO CONVOLVE FOURIER IMAGE WITH FOURIER FILTER.
% This function is when you're using a symmetric filter like Gaussian
% ---------------------------------------------------------------
% fourier_image -> Fourier transform of the Image, 'fft2(Image)' (INPUT)
% fourier_filter -> fourier transform of the filter already padded on same size
%                   of Image, 'fft2(paddedfilter)' (INPUT)
% win -> window of FILTER (INPUT)
% freq_filtered_image -> Image filtered on frequency domain (OUTPUT)
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%
% Example of Program
% ImF=fft2(Image);
% F=fft2(padded_filter);%here filter has win=15 for example
% F3=jon_sym_fconv(ImF,F,15);%our Image convolved on frequency domain
% figure
% subplot(121);imshow(Image);
% subplot(122);imshow(F3,[]);
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 convolved_image=fourier_image.*fourier_filter;
INN=ifft2(convolved_image);
%symmetric part
[INN1,INN2]=size(INN);
freq_filtered_image=zeros(INN1,INN2);
freq_filtered_image(1:INN1-(win),1:INN2-(win))=INN((win+1):INN1,(win+1):INN2);
freq_filtered_image(INN1-(win)+1:INN1,1:INN2-(win))=INN(1:(win),(win+1):INN2);
freq_filtered_image(1:(INN1-(win)),   (INN2-(win)+1):INN2)=INN(((win)+1):INN1,1:win);%primera
parte
freq_filtered_image((INN1-(win)+1):INN1,(INN2-(win)+1):INN2)=INN(1:(win),1:(win));
```

```
end % end function 'jon_sym_fconv()'
```

ANEXO 14: Algoritmo Binarizacion de imagen

```
%%%%%%%%%% Binarization I2 (C Code) %%%%%%%%%%
Threshold=input('Insert Threshold value: ')
for i=1:1:I2_row                    %Just Positioning in all rows
                                    %(not taking the value)
   for j=1:1:I2_column              %Just Positioning in all columns
                                    %(not taking the value)
      if (I2(i,j)<=Threshold)
         I2(i,j)=0;%is Background (black)
       else
          I2(i,j)=1;%is White
      end
   end
end
```

ANEXO 15: Algoritmo de Etiquetado y Filtro de Tamaño

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%               LABELLING:              %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
L = bwlabel(I4f3,8);%defino vecinos 8, of gaussian filt. Image
[Li Lj]=size(L);
maxi=max(max(L));%tell us how many object has detected(the max value)
vmax=0;
for k=1:1:1:maxi
   vmax(1,k)=length(find(L==k));%putting in a vector how many pixels are
      %in each object, so we can have in a vector the length of each object
end
% j=0;%Reset counter from maxi to 0 value so we can use it again later
[rmax,cmax]=find(vmax==(max(vmax)));%drop us the number of object that has
%more pixels on it in cmax variable(is a column of the vector vmax)
% cmax -> AVERAGE QUANTITY PIXELS OF THE OBJECT!!!
       %in gray scale just to see it
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Detecting the object (sweeping-barriendo) FILTRO TAMAÑO (CMAX)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for i=1:1:Li%for3
   for j=1:1:Lj%for2
      if L(i,j)==cmax%CMAX IS QUANTITY OF PIXELS OF THE OBJECT
         PP(i,j)=1;
      else
         PP(i,j)=0;
      end%if1
   end%for2
end%for3
PP=logical(PP);
PP2=PP;
% % subplot(122),imshow(PP);%------------------------------
figure(8)
subplot(121);imshow(I20);
subplot(122);imshow(PP);
%
```

ANEXO 16: Algoritmo código de cadena

```
% |%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% |
% |                    CHAIN CODE, BY JON                              |
% | ---------------------------------------------------------------------------------------- |
% | jon_chaincode(bin_im) is a function where we get the contour of any |
% | binary image.                                         |
% |[cc,border_im]                                        |
% | where:                                               |
% | bin_im       -> binary image (INPUT)                  |
% | cc           -> is my chain code array (OUTPUT)            |
% | border_im    -> is the contour of my binary image,          |
% |              result from chain code (OUTPUT)            |
% | ___________________________________________________________________________ |
function [ cc, border_im ] = jon_chaincode( bin_im )
%UNTITLED4 Summary of this function goes here
%   Detailed explanation goes here
% ////////////////////////////////////////////////////////
% ----finding first point of image to start chain code-----
sz=size(bin_im);
indx=find(bin_im,1)-1;
start(2)=floor(indx/sz(1));
start(1)=indx-(sz(1)*start(2));
start
% -------------------------------------------------------
% -------Directions vector------------------------------
directions = [ 0, 1     %Doing this directions in my array (how Matlab works)
              -1, 1     %  |-1,-1| |-1,0| |-1,1|
              -1, 0     %  | 0,-1| | 0,0| | 0,1|
              -1,-1     %  |-1,-1| | 1,0| | 1,1|
               0,-1     %
               1,-1     %
               1, 0     %
               1, 1];   %
%---------------------------------------------
cc=[];%Initialize chain
coord=start;%Coordinates of the current pixel
dir=1;%First direction we want to go, because it's first point
% ------------------------------------------------------------
```

```
% -----Doing loop for the chain code-------------------
while 1
    newcoord=coord+directions(dir+1,:);
    if all(newcoord>=0) && all(newcoord<sz-1) && bin_im(newcoord(1)+1,newcoord(2)+1)
        cc=[cc,dir];
        coord=newcoord;
        dir=mod(dir+2,8);
    else
        dir=mod(dir-1,8);
    end
    if all(coord==start) && (dir==1)

        break;
    end
end %while
cc; %we've got the chain code;
% ----------------------------------------
% -------if we want to plot the chaincode----
border_im=zeros(sz);
coords=start;
for i=1:length(cc)
    border_im(coords(1)+1,coords(2)+1)=1;
    coords=coords+directions(cc(i)+1,:);
end %for
figure(1)
subplot (122), imshow(border_im)
% ----------------------------------------
% ////////////////////////////////////
v=jon_chainsub(cc);
end%function jon_chain()
```

ANEXO 17: Algoritmo vector invariante a traslacion

```
% This function returns the minor vector of the chain code which is the one
% that is translation invariant
% "This is a sub-function of the jon_chaincode() function"
% --------------------------------------------------------------------
% tcc -> OUTPUT vector translation invariant, the minor value
% cc  -> INPUT Direction vector of boundary
% --------------------------------------------------------------------
function [ v ] = jon_chainsub( cc )
%UNTITLED4 Summary of this function goes here
%   Detailed explanation goes here
k=length(cc);
cc2=0;
for i=1:1:k
    v(i,:)=cc;%;
    cc2(k)=cc(1);%;
    cc2(1:k-1)=cc(2:k);%
    cc=cc2;%
end
v;
[r c]=size(v);
w=0;
for k=1:1:c
    mini=min(v(:,k));
    for i=1:1:r
        cr=v(i,k);
        if cr==mini
            w=w+1;
            f(w,:)= v(i,:);
        end
        if (i==r)
            v=f(1:w,:);
            [r c1]=size(v);
            w=0;
        end
    end
end
% v
end
```

ANEXO 18: Algortimo aplicado de capa de entrada a Capa oculta

$$S_1 = W_{10}X_0 + W_{11}X_1 + W_{12}X_2 + W_{13}X_3 + W_{14}X_4 + W_{15}X_5$$

$$S_2 = W_{20}X_0 + W_{21}X_1 + W_{22}X_2 + W_{23}X_3 + W_{24}X_4 + W_{25}X_5$$

$$S_3 = W_{30}X_0 + W_{31}X_1 + W_{32}X_2 + W_{33}X_3 + W_{34}X_4 + W_{35}X_5$$

$$S_4 = W_{40}X_0 + W_{41}X_1 + W_{42}X_2 + W_{43}X_3 + W_{44}X_4 + W_{45}X_5$$

ANEXO 19: Algoritmo Aplicado de capa oculta a capa de salida de la neurona

Se tiene S_{hl}

Donde: $l \in \square \subset [1-3]$

$$S_{h1} = h_0 C_{01} + h_1 C_{11} + h_2 C_{21} + h_3 C_{31} + h_4 C_{41}$$

$$S_{h2} = h_0 C_{02} + h_1 C_{12} + h_2 C_{22} + h_3 C_{32} + h_4 C_{42}$$

$$S_{h3} = h_0 C_{03} + h_1 C_{13} + h_2 C_{23} + h_3 C_{33} + h_4 C_{43}$$

ANEXO 20: Cálculo de valor espontáneo de la energía del error (Error cuadrático medi – proceso Back del BackPropagation)

$$E(1)=\frac{1}{2}((e_1^2(1)+e_2^2(1)+e_3^2(1))$$

$$E(2)=\frac{1}{2}((e_1^2(2)+e_2^2(2)+e_3^2(2))$$

$$E(3)=\frac{1}{2}((e_1^2(3)+e_2^2(3)+e_3^2(3))$$

$$E(4)=\frac{1}{2}((e_1^2(4)+e_2^2(4)+e_3^2(4))$$

$$E(5)=\frac{1}{2}((e_1^2(5)+e_2^2(5)+e_3^2(5))$$

$$E(6)=\frac{1}{2}((e_1^2(6)+e_2^2(6)+e_3^2(6))$$

$$E(7)=\frac{1}{2}((e_1^2(7)+e_2^2(7)+e_3^2(7))$$

$$E(8)=\frac{1}{2}((e_1^2(8)+e_2^2(8)+e_3^2(8))$$

$$E(9)=\frac{1}{2}((e_1^2(9)+e_2^2(9)+e_3^2(9))$$

$$E(10)=\frac{1}{2}((e_1^2(10)+e_2^2(10)+e_3^2(10))$$

$$E(11)=\frac{1}{2}((e_1^2(11)+e_2^2(11)+e_3^2(11))$$

$$E(12)=\frac{1}{2}((e_1^2(12)+e_2^2(12)+e_3^2(12))$$

$$E(13)=\frac{1}{2}((e_1^2(13)+e_2^2(13)+e_3^2(13))$$

$$E(14)=\frac{1}{2}((e_1^2(14)+e_2^2(14)+e_3^2(14))$$

$$E(15)=\frac{1}{2}((e_1^2(15)+e_2^2(15)+e_3^2(15))$$

$$E(16)=\frac{1}{2}((e_1^2(16)+e_2^2(16)+e_3^2(16))$$

$$E(17)=\frac{1}{2}((e_1^2(17)+e_2^2(17)+e_3^2(17))$$

$$E(18)=\frac{1}{2}((e_1^2(18)+e_2^2(18)+e_3^2(18))$$

$$E(19)=\frac{1}{2}((e_1^2(19)+e_2^2(19)+e_3^2(19))$$

$$E(20)=\frac{1}{2}((e_1^2(20)+e_2^2(20)+e_3^2(20))$$

ANEXO 21: Cálculo de gradient local para la capa oculta (Proceso Back de BackPropagation)

$$\delta_{1in} = \sigma'(s_1)[\delta_1 . C_{11} + \delta_2 . C_{12} + \delta_3 . C_{13}]$$

$$\delta_{2in} = \sigma'(s_2)[\delta_1 . C_{21} + \delta_2 . C_{22} + \delta_3 . C_{23}]$$

$$\delta_{3in} = \sigma'(s_3)[\delta_1 . C_{31} + \delta_2 . C_{32} + \delta_3 . C_{33}]$$

$$\delta_{4in} = \sigma'(s_4)[\delta_1 . C_{41} + \delta_2 . C_{42} + \delta_3 . C_{43}]$$

ANEXO 22: Algoritmo - vector de características de la imagen (invariante ante rotaciones traslaciones y escalamiento)

```
% This function returns a CHARACTERISTIC vector of the chain code which is the one
% that is translation rotation and scale invariant
% "This is a sub-function of the jon_chaincode() function"
% ------------------------------------------------------------------
% fo1, traslation -> OUTPUT traslation momentum
% fo2, rotation -> OUTPUT traslation momentum
% fo3, Scaling -> OUTPUT traslation momentum
% perimeter_char, Perimeter -> OUTPUT traslation momentum
% Imarea, Area -> OUTPUT traslation momentum
% t  -> INPUT jon_chaincode image
% ------------------------------------------------------------------
function [ t ] = jon_characteristics( fo1,fo2,fo3,perimeter_char,Imarea)
%UNTITLED4 Summary of this function goes here
%   Detailed explanation goes here
 x0=coordinates(t,1)
 y0=coordinates (t,2)
 f00=0; d00=0;
 for I=1:1:t
    x= coordinates (I,1);
    y=coordinates(I,2)
    [Image1 dy dx]=jonchaincode(Image)
    pe=dy/dx;
    p=0; q=0;
    d00=d00+x;
    f00=foo+y;
    f00=pe*(x^(p+2)-x0^(p+2));
    d00=d00+f00/((p+2)*(p+1));
 end

 f1=(x^(p+5)-x0^(p+5))/(p+5)
 f2=(x^(p+2)-x0^(p+2))/(p+2);
```

```
 f3=(x^(p+4)-x0^(p+4))/(p+4);
 e=y0-pe*xo;
 fo1=(p^2)*f1+(e^2)*f2+2*p*e*f3;%Translation Invariant
 fo2=(pe^2)*f1+(e^2)*f2+2*pe*e*f3;%Rotation Invariant
 fo3=pe(x^(p+2)-xo^(p+2));%Scaling Invariant
 [image_Per,x1,y1]=jon_chaincode(Image);
 perimeter_char=sum(image_Per);%Perimeter, Characteristic

 %Area
[image_Per,x,y]=jon_chaincode(Image)
Immcoord=[Image_Per x y];
%x y are coordinates of the chaincode points
xmax=max(x);
ymax=max(y);
for i=1:1:xmax
   for j=1:1:ymax
      if Imcoord(i,j)==1
         %desplazando coordenadas
         x=x+1
         y=y+1
         Im1=Imcoord(x,y)
         k=Im1(x,y)
         Imarea=k+k
      end
   end
end
```

ANEXO 23: Parámetros Intrínsecos de Cámaras

Stereo calibration parameters after optimization:

Intrinsic parameters of left camera:

Focal Length: fc_left = [606.32277 601.55612] ± [11.77323 10.97921]

Principal point: cc_left = [311.02300 222.84665] ± [18.61915 12.21605]

Skew: alpha_c_left = [0.00000] ± [0.00000] => angle of pixel axes = 90.00000 ± 0.00000 degrees

Distortion: kc_left = [0.13484 -0.36483 -0.00526 -0.02438 0.00000] ± [0.11689 0.48879 0.00774 0.01524 0.00000]

Intrinsic parameters of right camera:

Focal Length: fc_right = [596.52651 598.38297] ± [11.13107 10.94936]

Principal point: cc_right = [314.52306 251.73694] ± [18.18732 16.04565]

Skew: alpha_c_right = [0.00000] ± [0.00000] => angle of pixel axes = 90.00000 ± 0.00000 degrees

Distortion: kc_right = [0.10568 -0.73821 0.00685 -0.00065 0.00000] ± [0.12709 0.71193 0.00934 0.01201 0.00000]

Extrinsic parameters (position of right camera wrt left camera):

Rotation vector: om = [0.05251 -0.02230 0.02846] ± [0.03230 0.04166

0.00227]

Translation vector: T = [-103.05151 1.08753 -3.50124] ± [2.33896 2.17577 11.09019]

Note: The numerical errors are approximately three times the standard deviations (for reference).

ANEXO 24: Algortimo de Profundidad (Estereoscopía)

```
[Xc_1_left,Xc_1_right] = stereo_triangulation(x_left_1,x_right_1,om,T,fc_left,cc_left,kc_left,alpha_c_left,fc_right,cc_right,kc_right,alpha_c_right);
```

ANEXO 25: Coordenadas en el mundo Real de la primera imagen del ojo Izquierdo (Stereo Algorithm)

Xc_1_left =

Columns 1 through 10

187.6792 189.9522 191.8407 193.4748 195.1722 195.9860 197.4454 139.6669 141.2171 142.4287
-126.1841 -77.6842 -29.0662 19.6360 67.9983 116.0531 164.7932 -124.3091 -76.3269 -27.4866
738.9088 742.5188 747.2982 746.7451 750.4188 750.5509 752.5021 738.4187 741.1438 741.5078

Columns 11 through 20

144.6342 145.9108 147.4352 149.0497 91.0459 92.3512 94.4998 96.0606 97.3255 98.7282
21.0549 69.3402 117.7887 165.6352 -122.0678 -74.1818 -25.7722 22.6061 70.9531 118.8420
745.4846 746.9639 750.5658 750.9507 732.8355 736.3137 739.8168 742.3531 744.9458 745.3998

Columns 21 through 30

100.3390 43.0023 44.0669 46.4102 47.4132 48.8661 50.6290 52.3596 -4.1881 -3.4340
166.2747 -120.2574 -71.9921 -24.3536 24.0783 72.1201 119.6415 166.6984 -117.3610 -70.3017
743.8144 728.7307 730.9529 733.0164 738.6380 741.1414 738.7211 738.8784 722.0640 728.1144

Columns 31 through 40

-1.5523 -0.1109 1.4748 3.3466 5.2143 -51.2239 -49.7222 -48.7338 -46.8829 -45.2436

-22.5914 25.4508 73.0075 120.1630 167.1078 -114.7632 -67.4125 -20.5878 26.8712 74.2037

727.7543 732.8162 733.0547 732.8195 733.4208 716.1367 719.0684 724.1652 725.7529 727.9233

Columns 41 through 49

-42.9490 -40.9062 -95.1527 -94.0201 -93.2528 -91.5520 -89.8661 -86.8027 -85.0418

120.7779 166.6850 -111.0815 -64.8180 -18.5350 28.0747 74.5988 119.4398 165.0856

726.1405 723.8843 702.2317 705.2529 711.1485 714.9968 714.9115 709.9176 711.1578

Anexo 26: Coordenadas en el mundo Real de la primera imagen del ojo Derecho (Stereo Algorithm)

Xc_1_right =

Columns 1 through 14

72.2528 73.0389 73.4122 73.6437 73.8571 73.2728 73.2745 24.2280 24.3259 24.1113 24.8209 24.6634 24.7030 24.9195

-158.6495 -110.3634 -62.0317 -13.3416 34.7877 82.7726 131.3635 -158.0894 -110.2933 -61.5256 -13.2191 34.9376 83.1520 130.9379

731.9473 738.1332 745.4840 747.5094 753.7384 756.3953 760.9188 730.4500 735.7088 738.6474 745.2001 749.2246 755.3825 758.2992

Columns 15 through 28

-24.3060 -24.4663 -23.7995 -23.6985 -23.8936 -23.8915 -23.6242 -72.2823 -72.6672 -71.7525 -72.2769 -72.2732 -71.8386 -71.4782

-156.9115 -109.2582 -61.0600 -12.8584 35.3008 83.1195 130.5961 -156.2259 -108.1343 -60.6244 -12.5476 35.3168 82.9304 129.9433

723.8729 729.8730 735.9446 741.0359 746.1749 749.1581 750.0860 718.7626 723.5230 728.1217 736.2832 741.3215 741.4241 744.0751

Columns 29 through 42

-119.3824 -120.1249 -119.6215 -119.6843 -119.4851 -118.9781 -118.4871 -166.3352 -166.2719 -166.7529 -166.3150 -166.0972 -165.1174 -164.3599

-154.2971 -107.6202 -59.9235 -12.1939 35.3096 82.4455 129.3269 -152.7011 -105.5478 -59.0482 -11.7060 35.4732 82.1224 128.1237

711.1709 719.6832 721.8553 729.4479 732.2027 734.4703 737.5619 704.3054 709.7365 717.2899 721.3929 726.0660 726.7679 726.9566

Columns 43 through 49

ñ -210.0427 -210.3186 -211.0221 -210.7581 -210.4217 -208.5543 -208.1460

-149.5156 -103.4625 -57.5519 -11.1810 35.3116 80.4216 125.9696

689.6033 695.0585 703.3761 709.6881 712.0682 709.4916 713.1508