INTEGRACIÓN DE UN SISTEMA DE VISIÓN ARTIFICIAL A UN ROBOT INDUSTRIAL PARA LA SOLUCIÓN DE UN CUBO DE RUBIK

JOSÉ FERNANDO GIL BOTERO

UNIVERSIDAD AUTÓNOMA DE OCCIDENTE
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE AUTOMÁTICA Y ELECTRÓNICA
PROGRAMA DE INGENIERÍA MECATRÓNICA
SANTIAGO DE CALI
2012

INTEGRACIÓN DE UN SISTEMA DE VISIÓN ARTIFICIAL A UN ROBOT INDUSTRIAL PARA LA SOLUCIÓN DE UN CUBO DE RUBIK

JOSÉ FERNANDO GIL BOTERO

Proyecto de grado para optar por el título de Ingeniero Mecatrónico

Director
JUAN CARLOS PERAFÁN VILLOTA
Msc. Ingeniero Electricista

UNIVERSIDAD AUTÓNOMA DE OCCIDENTE
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE AUTOMÁTICA Y ELECTRÓNICA
PROGRAMA DE INGENIERÍA MECATRÓNICA
SANTIAGO DE CALI
2012

Nota de Aceptación:

Aprobado por el comité de grado en cumplimiento de los requisitos exigidos por la Universidad Autónoma de Occidente para optar al título de Ingeniero Mecatrónico.

JESÚS ALFONSO LÓPEZ
Jurado

JUAN DIEGO PULGARÍN
Jurado

Santiago de Cali, 03 de Febrero de 2012

CONTENIDO

LISTA DE FIGURAS

Pág.

LISTA DE ANEXOS

Pág.

GLOSARIO

BIT DE PARIDAD: bit utilizado en comunicación serial que indica si la cantidad de bits enviados con el número 1 es par o impar, es un método de corregir errores de comunicación.

CAPA: capa de un Cubo de Rubik, en el presente documento se considera capa como la sección de un cubo de Rubik que puede girar respecto a un eje determinado. Se compone de una cara y todos los cuadros de colores que tienen un borde en común con dicha cara.

CARA: cara de un Cubo de Rubik, en el presente documento se considera cara al conjunto de nueve cuadros de colores del cubo que pertenecen al mismo plano.

CROMINANCIA: componente del espacio de color YUV la cual contiene la información sobre el color. Es una cantidad bidireccional compuesta por tono y saturación.

EXTEROCEPTIVO: sensor exteroceptivo, sensor que se encarga de detectar las condiciones externas de un dispositivo, da información sobre el entorno.

GRIPPER: herramienta ubicada en el extremo de un robot para manipular objetos del entorno, en el caso particular del proyecto el gripper del robot es una pinza neumática.

HISTOGRAMA: representación gráfica que indica cuantas veces se repite un color en una imagen.

LUMINANCIA: componente del espacio de color YUV la cual contiene la cantidad de luz incidente sobre un color determinado por la crominancia.

PROPIOCEPTIVO: sensor propioceptivo, sensor que se encarga de detectar las condiciones internes de un dispositivo, da información sobre el estado propio.

RGB: espacio de color utilizado en las pantallas, representa los colores en tres colores primarios que son el rojo, verde y azúl.

SINAPSIS: comunicación que se da entre dos neuronas, en el caso natural se da por medio de neurotransmisores, en el caso artificial se da por medio de señales eléctricas.

TASA DE BAUDIOS: medida utilizada para indicar la velocidad de transmisión, no se debe confundir por la tasa de bits

TRAMA: cantidad de información que se envía a un dispositivo por comunicación serial, contiene los datos que desea enviar el usuario e información adicional que entienden los dispositivos para corregir errores de comunicación y entender cuándo termina la trama.

YUV: Espacio de color el cual representa los colores por medio de luminancia y crominancia.

RESUMEN

El presente trabajo muestra el proceso realizado para la elaboración de un sistema de bajo costo el cual integra un robot con una cámara de video que permitirá realizar una tarea asociada a visión artificial.

El proyecto busca demostrar que añadiendo un sistema de visión a un robot tipo industrial se pueden ampliar las posibilidades de tareas realizadas por este. La tarea planteada fue solucionar la capa superior de un Cubo de Rubik la cual es la que requiere de más movimientos, cosa que permite demostrar que si esta cara es solucionada, no hay limitación alguna para que el mismo sistema pueda armar el cubo en su totalidad.

Para poder ejecutar la tarea se requiere inicialmente capturar las imágenes de las diferentes caras del Cubo de Rubik mediante una cámara web de donde se extrae información del color asociado a cada bloque del cubo y se envía al sistema de procesamiento de imagen.

El sistema de procesamiento se hizo mediante una red neuronal que fue entrenada con información de capturas realizadas en diferentes escenarios de modo que se pudiera ampliar un poco la gama luminosidad que ésta pudiera detectar y así tener un sistema mucho más confiable y asegurar la correcta detección de colores del Cubo de Rubik.

Teniendo los colores ya detectados de cada una de las caras del cubo, se envía la información a un sistema de solución basado en el algoritmo de Singmaster, el cual se encarga de decidir qué rotaciones se deben hacer en el cubo.

Por último, esta información es enviada a un planificador de tareas que decide cuales son los movimientos que deberá hacer el robot para conseguir rotar las piezas móviles del cubo. Cabe anotar que el robot usado fue el Intelitek Performer SV3 y que para poder realizar correctamente las rotaciones del cubo, se requiere de una base adicional hecha en madera la cual sostiene la capa inferior del cubo.

PALABRAS CLAVES: CUBO DE RUBIK. ROBOT. ROBÓTICA. MECATRÓNICA. VISIÓN ARTIFICIAL. RED NEURONAL. PROCESAMIENTO DE IMÁGENES. MOTOMAN. DISPOSITIVO MECATRÓNICO. SENSOR. SENSORES.

INTRODUCCIÓN

Los procesos industriales de la actualidad, requieren cada vez más de la automatización, razón por la cual son implementados sistemas que integran mecánica, electrónica, computación y control.

La idea principal de utilizar este tipo de sistemas en procesos industriales es que éstos realicen tareas repetitivas de modo que todas las piezas construidas durante el proceso tengan geometrías estandarizadas y así el proceso es acelerado. Dichos robots poseen en su extremo una herramienta que es la que se adapta al tipo de proceso, algunos son utilizados para soldar, otros son utilizados para pintar con aerógrafo, otros para simplemente transportar objetos de un lado a otro. Por ejemplo, el robot Unimate fue el primer robot industrial el cual se utilizaba en la línea de ensamblaje de la General Motors y tenía 4 grados de libertad.

Para que estos sistemas sean exitosos, deben tener la capacidad de percibir el entorno, y es por esto que han surgido una serie de dispositivos conocidos como sensores. Existen sensores tales como: de ultrasonido, laser, odométricos, capacitivos e inductivos entre otros.

Un dispositivo que viene creciendo en su utilización como elemento sensorial es la cámara de video. Éste dispositivo puede brindar información más completa del entorno, lo cual es una gran ventaja pero esto trae consigo la necesidad de procesar más información. Este problema ha sido solucionado en gran medida gracias a las mejoras computacionales (Procesador y RAM) y a la gran cantidad de algoritmos optimizados presentados por la comunidad científica.

El grupo de Visión Artificial y Robótica de la Universidad de Cambridge actualmente trabaja en un proyecto para modelar un objeto existente en 3D mediante la toma de varias fotografías periféricas[1]. Por otro lado, el Instituto de Robótica de la Universidad Carnegie Mellon trabaja en un proyecto para

[1] COMPUTER VISION AND ROBOTICS GROUP. Universidad de Cambridge. 3D Shape from Uncalibrated Images. [En línea] 2011. [consultado 24 de Marzo de 2011]. Disponible en Internet: http://mi.eng.cam.ac.uk/~cipolla/research.htm.

reconocimientos de rostros humanos en una imagen estática con el propósito de luego implementarlo en imágenes en movimiento[2].

Al reunir las técnicas visuales con los robots, es posible tener sistemas más potentes y con mejores prestaciones, pues estos dispositivos ampliarían la gama de aplicaciones posibles con ellos. Funciones como el control de calidad en medio de la línea de producción pueden disminuir tiempo y dinero al proceso.

Nuestro objetivo con este proyecto es entonces hacer la integración de la parte visual con un robot manipulador industrial y garantizar un mejor rendimiento en una tarea específica, que en el caso particular es la solución de un cubo de Rubik.

[2] ROBOTICS INSTITUTE. The Robotics Institute. Face Detection. [En línea] 2011. [consultado 24 de Marzo de 2011]. Disponible en Internet: http://www.ri.cmu.edu/research_project_detail.html?project_id=416&menu_id=261.

1. PLANTEAMIENTO DEL PROBLEMA

En la actualidad se ha visto la necesidad de que los dispositivos puedan percibir qué es lo que ocurre con el entorno para así interactuar con él. Se han diseñado diferentes sensores exteroceptivos que han permitido esa interacción y constantemente se hacen estudios para mejorarlos.

Uno de los estudios más complejos, ha sido encontrar un sensor que entregue gran cantidad de información sobre el entorno a nivel visual. Por ejemplo, sensores de proximidad los cuales permiten saber si hay un objeto presente o no y su distancia, y sensores de color que permiten tener acercamientos a la visión pero no el reconocimiento exacto del mismo en su geometría y color.

A raíz de esto surge la pregunta de cómo tener un sensor que integre la detección de forma y color de un objeto y la respuesta a ello es la cámara de video. El problema con esto es que si bien la cámara como tal reduce costos en los componentes, se requiere detrás de ella un dispositivo que procese la información que ésta brinda para hacerla entendible al sistema.

Este problema ha ido encontrando solución en la actualidad con el advenimiento de los computadores que cada vez tienen más capacidad de procesamiento y almacenamiento de información. Y en el presente proyecto se pretende contribuir con la solución.

Es claro que este tipo de estudios sobre la visión artificial, no son solamente para hacer algo investigativo, sino también aplicativo, pues no es lógico que se haga un estudio exhaustivo sobre la cámara de video si no se piensa de qué manera se podría aplicar. En el caso particular los estudios realizados, tienen como propósito integrar la visión artificial con los robots industriales para la realización de una tarea específica.

Dado lo anterior, surgen unos cuestionamientos que se deben hacer para darle un punto de partida al proyecto: ¿Qué se necesita para integrar un sistema de visión artificial con un robot industrial? ¿Cómo se va a integrar? ¿Qué aplicaciones se pueden mejorar con esta técnica?

2. JUSTIFICACIÓN

Se buscó diseñar un sistema de visión artificial que fuera de bajo costo para utilizarlo con robots industriales y así ampliar las aplicaciones que se pueden hacer con ellos, ya que no solamente tendrán sensores propioceptivos para saber su estado propio, sino también exteroceptivos para saber con mayor certeza, qué objetos contiene su entorno. De esta manera ya se podrían utilizar dichos robots para clasificar objetos o detectar objetos defectuosos en una misma banda de producción o también ensamblar objetos, entre otros.

El hecho de que el dispositivo sea de bajo costo, significa que éste puede ser mucho más accesible en desarrollos de pequeñas y medianas empresas, las cuales lo pueden llegar a utilizar en sistemas mucho más complejos que integren esta tecnología.

Es claro que esta tecnología es muy útil y no solamente para procesos industriales sino que se puede utilizar en espacios mucho más habituales como por ejemplo en el área de la domótica o en los automóviles para que éstos estén en toda capacidad de avisarle al usuario de cualquier peligro que pueda ocurrir en un momento determinado. O en el área de vigilancia donde no debe estar una persona monitoreando cada una de las cámaras como se hace actualmente sino que el mismo sistema inteligente esté en la capacidad de avisar a la persona encargada si hay algo sospechoso.

Dado esto se necesita garantizar no sólo un dispositivo de bajo costo sino también de buena calidad y por eso se utilizaron los conocimientos adquiridos en la academia sobre electrónica, mecánica, computación y control para así hacer un sistema de visión artificial aplicado a un robot del laboratorio que es el Intelitek Performer SV3, Utilizando diseño concurrente para saber entre las diferentes opciones posibles, cuál era la mejor opción para ello y así llevar a cabo un proceso didáctico que fue el de solucionar la primera cara de un Cubo de Rubik dado que computacionalmente ésta es la cara más difícil de armar y con esta se puede demostrar claramente que el cubo completo puede ser solucionado con el sistema.

3. ANTECEDENTES

El Procesamiento de las imágenes empezó en los años 20's cuando se empezó a ver la necesidad de transmitir una imagen de un lugar a otro por medio de una señal. A Principios de esta década, luego de haber logrado enviar imágenes digitalizadas de un periódico por cable submarino de New York a Londres se buscaron formas de optimizar el sistema de transmisión por medio de algoritmos.

El sistema Bartlane fue aquel algoritmo que logró reducir el tiempo de transmisión de más de una semana a menos de tres horas pero había problemas de impresión ya que las impresoras del momento debían ser adaptadas para imprimir imágenes. Por esto se hizo una mejora al algoritmo Bartlane que diera unas mejoras a las imágenes que contrarrestaran los errores de las impresoras, modificando la imagen a 5 brillos diferentes de acuerdo a la impresora.

La aparición de las computadoras fue la que le dio gran impulso al procesamiento de imágenes cuando los científicos del Laboratorio de Propulsión Espacial recibieron imágenes de la luna tomadas por la sonda espacial Ranger 7 y se vieron en la necesidad de corregir dichas imágenes debido a los errores producidos por la cámara de la sonda. Aquí se solucionaban problemas de curvatura por medio de computadores[3].

En esa época el procesamiento de las imágenes se hacía para una mejor comprensión humana, esto significa que las imágenes eran procesadas y mejoradas para que la imagen pudiera ser humanamente comprendida.

En esta década surge también un concepto nuevo que es el de la realidad virtual. Este fue uno de los primeros acercamientos al manejo de la cámara con los dispositivos electrónicos. El Head-Mounted Display fue el primer dispositivo en lograrlo permitiendo que el hombre pudiera a partir de sus movimientos (Realidad Visual) explorar un nuevo mundo de realidad virtual.

Paralelo a esto, en los Estados Unidos los científicos Warren McCulloch y Walter Pitts realizaban estudios sobre el funcionamiento del cerebro logrando con esto modelar sistemas de redes neuronales por medio de algunos dispositivos que sólo

[3] GONZÁLES, Rafael C. y WOODS, Richard E. Digital Image Processing. Upper Saddle River : Pearson Prentice Hall, 2008. p. 67.

podían dar resultados binarios y funcionaban bajo la lógica fundamental donde cada neurona poseía una función tal como “and” y “or”. Luego de esto surgieron métodos de aprendizaje para las neuronas artificiales. Y sistemas más complejos tales como el Perceptrón y la red ADALINE. Con estos sistemas más complejos se pudo luego utilizar las redes para identificación y separación de patrones.

Fue en los años 70’s cuando el enfoque del procesamiento de imágenes dio un giro radical y se empezó a hacer procesamiento de imágenes no para una mejor comprensión humana sino para que las imágenes fueran más comprensibles a los computadores para así realizar un mejor análisis computacional que podía arrojar resultados mucho más exactos. En el campo de las redes neuronales fue en esta misma época donde las redes empezaron a difundirse por todo el mundo con la aparición de libros y conferencias que permitieron que estos sistemas fueran introducidos en programas académicos de universidades[4].

A finales de la década de los 80’s se empezó a hacer procesamiento de imágenes para editores gráficos utilizando una gran cantidad de filtros los cuales permitían mejorar la imagen para hacerla más agradable o incluso eliminar o agregar cosas en ella. Es aquí donde aparece el tan reconocido photoshop el cual es uno de los programas más completos para edición de imágenes y aún se sigue mejorando cada vez más. Y en esta época se empezaron a integrar redes neuronales con cámaras para generar visión artificial reconociendo patrones como por ejemplo colores o formas en una imagen adquirida[5].

En la década de los 90’s se utilizó por primera vez el término de Realidad Aumentada al diseñar un software de ayuda para los ensambladores de Boeing que contenía un laboratorio virtual para que ellos aprendieran sobre la línea de ensamblaje. La Realidad Aumentada ha sido una de las aplicaciones más importantes de la cámara de video integrada a un sistema de procesamiento ya que mediante ella se podía mezclar elementos reales con elementos virtuales[6].

---

[4] LÓPEZ, Jesús A. y CAICEDO, Eduardo F. Una Aproximación Práctica a las Redes Neuronales Artificiales. Cali : Editorial Universidad del Valle, 2009. p. 13.

[5] LINGGARD, Robert, MYERS, D. y NIGHTINGALE, C. Neural Networks for Vision, Speech and Natural Language. Londres, Reino Unido : Chapman & Hall, Ltd., 1992. p. 156.

[6] University of New Mexico. Electrical and Computer Engineering Department. Faculty Spotlight: Prof. Caudell and the Meeting of Minds. [En línea] 2011. [consultado 22 de Marzo de 2011]. Disponible en Internet: http://www.ece.unm.edu/morenews/profile_caudell.html.

En la actualidad se utiliza el procesamiento digital de imágenes a diario por medio de programas tan conocidos como los pertenecientes a la suite de adobe para generar avisos publicitarios y diferentes productos audiovisuales entre otros. De igual manera las redes neuronales han tenido progresos muy significativos y actualmente se pueden comprar circuitos integrados a base de redes neuronales para realizar distintas aplicaciones. Y en cuanto a robótica con visión artificial se han empezado a utilizar robots que detectan naranjas en los árboles de modo que ellos son los que hacen la recolección y se están haciendo estudios para recolectar también uvas en los viñedos.

4. MARCO TEÓRICO

4.1. VISIÓN ARTIFICIAL

La visión artificial es una tecnología que permite que un computador o algún dispositivo que posea elementos computacionales, pueda percibir imágenes del entorno pudiendo así reconocer con exactitud la forma y colores de todo lo que hay en el mismo.

La visión artificial está siendo utilizada para reconocimiento de letras escritas a mano, Inspección de máquinas, modelado de edificios en 3D, Imágenes médicas y Seguridad Automovilística entre otros[7].

4.2. SENSOR

Un sensor es un dispositivo de detección de cantidades físicas y químicas, transformándolas en cantidades eléctricas se compone de un transductor que es el que hace la detección, y un controlador para el transductor. Igualmente contienen todo tipo de instrumentación que permite adaptar la señal captada y transformada a unos niveles eléctricos más razonables para utilizar en los circuitos. Hay dos tipos de sensores:

Los sensores propioceptivos son utilizados para medir el estado interno de un dispositivo, por ejemplo la posición, velocidad angular y torque de sus propios motores y la temperatura en determinadas áreas internas. Y los sensores exteroceptivos son utilizados para detectar el entorno en el que el dispositivo se inmersa, por ejemplo los sensores de proximidad, de color, de fuerza, de tacto y de visión (la cámara)[8].

4.3. MODELO DE COLOR

Debido a que uno de los componentes más importantes en la visión artificial es el reconocimiento del color, se debe tener establecido un modelo de color que se use como base para la detección de la imagen, un modelo de color describe de qué

[7] AYACHE, Nicholas. Artificial vision for mobile robots: stereo vision and multisensory perception. Cambridge : InterEditions, 1991. p. 25.
[8] SICILIANO, Bruno, y otros, y otros. Robotics: modelling, planning and control. Londres : Springer, 2010. p. 87.

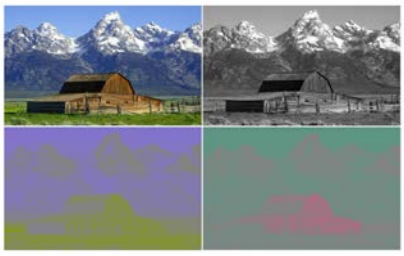
manera son utilizados ciertos colores “primarios” para obtener una gama de colores mucho más amplia.

El modelo de color más conocido y más inexacto es el RYB donde los tres colores principales son el Amarillo, Azul y Rojo; este modelo es utilizado en la pintura hecha en casa. Otro modelo muy común es el que utilizan las pantallas y es el de RGB donde los colores primarios son el Rojo, Verde y azul (ver Figura 1). Para el caso de las impresiones se utiliza un modelo más preciso que es el CMYK donde los colores base son Cian, Amarillo, Magenta y Negro.

Figura 1: Imagen descompuesta en sus componentes roja (R), verde (G) y azúl (B).

Existen otros modelos más complejos que son variaciones de los anteriores que permiten dar mucho más precisión a la gama de colores. Entre estos está el HSL el cual controla los colores RGB de acuerdo a su Matiz, Saturación y Luminancia, y YUV y YIQ que controlan los colores de acuerdo a la luminancia y un espacio cartesiano de crominancias (ver Figura 2)[9].

Figura 2: Imagen descompuesta en sus componentes luminancia (Y), crominancia u (U) y crominancia v (V).



Fuente: The Video Road: What is YUV? [en línea] 2011 [consultado 24 de Marzo de 2011]. Disponible en Internet: http://blogs.adobe.com/VideoRoad/2010/06/what_is_yuv.html

[9] GONZÁLES, Rafael C. y WOODS, Richard E. Op. cit. p. 67.

4.4. PROCESAMIENTO DE IMAGEN

El Procesamiento de imagen es un proceso digital empleado para extraer algunos atributos de las imágenes ya digitalizadas de modo que al analizar dicha información se pueden hacer cambios directos en la imagen para modificarla al gusto del usuario, bien sea mejorándola para una visualización más agradable, para eliminar errores o incluso reconocer objetos dentro de ellas.

Se habla de tres niveles de procesamiento de imágenes, el procesamiento de bajo nivel es utilizado para realizar operaciones primitivas (pre-procesamiento) tales como reducción de ruido, mejorar el contraste, mejorar el enfoque entre otros. El procesamiento de nivel medio es utilizado para operaciones un poco más complejas como la segmentación de áreas dentro de la imagen y descripción de cada segmento reducido a un formato más sencillo de procesar. Y Finalmente el procesamiento de alto nivel es utilizado para el reconocimiento como tal de la imagen, es aquí donde se hacen funciones cognoscitivas que son las que dan relación a este proceso con la visión humana[10].

4.5. ROBOT MANIPULADOR

Un robot manipulador es un robot utilizado generalmente en la industria que, como su nombre lo dice, tiene la función de manipular objetos que se presentan en la línea de producción. Se caracterizan por tener generalmente un brazo unido por articulaciones y una articulación rotacional en la parte final, donde se ubica la herramienta, que facilitan su movilidad[11]. Un ejemplo de robot manipulador se muestra en la Figura 3

Figura 3 : Robot manipulador Intelitek SV3

[10] Ibíd., p. 68.
[11] SICILIANO, Bruno Op. cit. p. 93.

4.6. NEURONA ARTIFICIAL

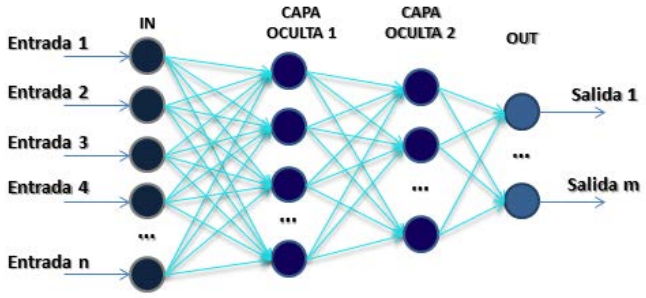
Una Neurona Artificial es un dispositivo que puede ser software o hardware el cual busca simular el funcionamiento de las neuronas cerebrales ya que busca actuar de igual manera teniendo los siguientes aspectos funcionales:

- **Recibe señales de entrada:** Esto implica que cada neurona está conectada a otras neuronas que le dan un estímulo de algún evento que está ocurriendo. Dicha información hace que la neurona actúe de alguna manera específica y a su vez, ésta afecte a oras neuronas. La comunicación que se da en las neuronas es lo que se denomina una sinapsis.
- **Las señales pueden ser modificadas por los pesos sinápticos:** Cada neurona posee unos pesos sinápticos con los cuales ella puede definir qué tan importante es una señal que le esté entrando y así saber de qué manera se comporta. Cuando se lleva a cabo el proceso de aprendizaje, son los pesos sinápticos los que se modifican para así saber qué hacer ante la misma situación en algún otro momento.
- **Suma las entradas afectadas por la sinapsis:** A medida que la neurona va recibiendo información de alguna sinapsis y sabiendo su importancia por medio de los pesos sinápticos, se acumulan todas las señales de entrada dentro de la neurona para así definir qué hacer.
- **En circunstancias apropiadas la neurona transmite una señal de salida:** Cuando la acumulación de señales de entrada llega a cierto umbral, la neurona es activada y se empieza a transmitir una señal de salida. Si esto no ocurre, la neurona permanecerá inactiva.
- **La señal de salida puede ir a varias neuronas:** Cuando la neurona es estimulada, la señal de salida viajará hacia todas las neuronas con las que ella tiene contacto y estas otras neuronas tendrán esta señal como estímulo en su entrada para ellas realizar el mismo proceso.

4.7. RED NEURONAL ARTIFICIAL

Como la neurona artificial posee baja capacidades de procesamiento, sólo sirven para realizar tareas supremamente simples. Para esto se hace lo que ocurre con las neuronas biológicas que es interconectarlas formando estructuras complejas que permiten realizar tareas mucho más difíciles. Existen actualmente diversas estructuras desarrolladas por varios científicos y todas estas estructuras son llamadas redes neuronales. Dichas estructuras poseen varias capas las cuales poseen neuronas que tienen una función en común. Generalmente una red posee 3 capas:

Figura 4: Red neuronal con sus respectivas capas



- **Capa de entrada:** Son las neuronas que reciben las señales que vienen de fuentes externas. Las neuronas de esta capa sólo tienen una entrada, lo que significa que hay tantas neuronas de esta capa como señales entrantes (ver Figura 4).
- **Capa oculta:** En esta capa posee las neuronas internas que no tienen ningún contacto con el exterior. Pueden estar organizadas en varios niveles y son las que analizan las señales para ejecutar las tareas requeridas. De la forma como estén interconectadas estas neuronas depende el tipo de red neuronal que se esté usando (ver Figura 4).
- **Capa de salida:** Las neuronas de esta capa se encargan de transferir la información procesada hacia el exterior. Hay tantas neuronas de esta capa como señales de salida (ver Figura 4).

5. OBJETIVOS

5.1. OBJETIVO GENERAL

Diseñar e implementar un sistema de visión artificial de bajo costo reconociendo colores para la solución de un cubo de Rubik con el robot Intelitek Performer SV3.

5.2. OBJETIVOS ESPECÍFICOS

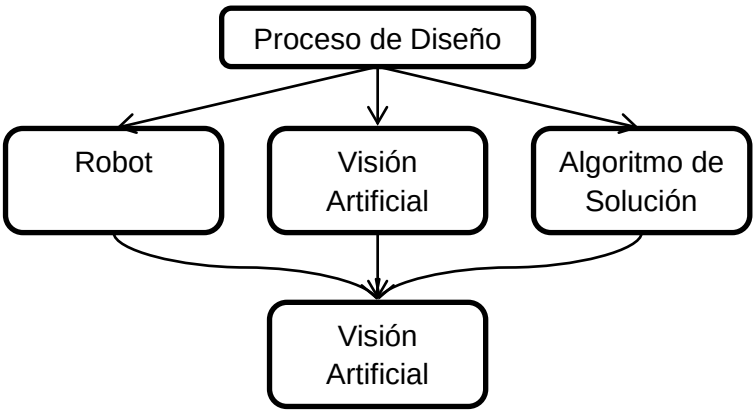
- Diseñar un sistema de acople entre el sistema de visión artificial y el robot Intelitek Performer SV3 utilizando el mejor protocolo de comunicación posible.
- Hallar el modelo de color más apropiado para la optimización del sistema de visión artificial.
- Encontrar el algoritmo más adecuado para procesamiento de imágenes.
- Solucionar la primera cara de un cubo de rubik como una aplicación experimental de la integración del sistema de visión con el robot.

6. METODOLOGÍA

Utilizando las bases del diseño concurrente, se resolvió el problema en diferentes etapas que fueron desarrolladas de forma paralela y por ultimo hubo una etapa donde se hacía la integración de las anteriores, pues a medida que se hacía cada una de las tareas, se podía ver que algunas de las otras que se realizaban paralelamente tenían incompatibilidades. Razón por la cual se diseñaba conforme a una idea y luego de haber terminado, se analizaba dicha tarea con las demás para así saber qué aspectos corregir.

De este modo se trabajó en paralelo los movimientos del robot con el sistema de visión artificial y el algoritmo de solución del cubo de Rubik analizando las incompatibilidades que podían haber entre las etapas y cuando ya se tenía todo esto completo, se hizo la integración de cada una de ellas (ver Figura 5)

Figura 5: Etapas de diseño del sistema



Por esto se fueron realizando inicialmente los movimientos del robot y el sistema de visión artificial, pero en el momento de realizar el algoritmo de solución del cubo de Rubik se pudo ver que se necesitaba hacer cambios en los movimientos, no solo debido a la ejecución de la tarea final sino también por limitaciones que se presentaban en los laboratorios. De modo que era muy importante que a medida que se fueran desarrollando cada uno de los elementos, se hicieran sus respectivas pruebas y así hacer las correcciones.

El diseño del sistema se hizo en cuatro etapas que, como se mencionó anteriormente, fueron realizadas en paralelo. Las etapas son: Programación de los

movimientos del robot, sistema de visión artificial, algoritmo de solución de la primera capa del Cubo de Rubik y acople de todas las etapas anteriores.

6.1. ETAPAS DEL PROYECTO

6.1.1. Robot. Programar los movimientos que debía realizar el robot para la realización de la tarea, la cual es resolver la primera cara del cubo de Rubik, igualmente fue necesario probar que dichos movimientos fueran adecuados al espacio, y al tamaño del cubo y hacer las debidas mejoras.

6.1.2. Visión Artificial. Encontrar el modelo de color y algoritmos de procesamiento adecuado para así utilizando el diseño concurrente, encontrar el mejor sistema de visión artificial posible. Esta metodología de diseño implicaba que constantemente se hacían pruebas sobre lo que se está diseñando.

6.1.3. Algoritmo de Solución. Investigar sobre los algoritmos existentes para solucionar el cubo de Rubik y diseñar un algoritmo optimizado para realizar los movimientos en el robot.

6.1.4. Acople. Una vez terminado el sistema de visión artificial, se hizo su respectiva aplicación en un sistema real que en este caso es un robot y se hicieron las pruebas, mediante la solución de la primera cara de un cubo de Rubik. Esto teniendo en cuenta el sistema de comunicación necesario entre el sistema de visión y el robot.

6.2. DESARROLLO DE LAS ETAPAS DEL PROYECTO

Es importante mencionar que si bien hubo cuatro etapas paralelas para el desarrollo del proyecto, se necesitó buscar información previa sobre los movimientos básicos del cubo de Rubik y utilizar unas convenciones que permitieran darle un orden al proyecto, pues era necesario hacer las diferenciaciones de las caras y ponerles a cada una un nombre y también a sus respectivos giros.

Para esto se utilizó una notación en ejes cartesianos utilizando la ley de la mano derecha donde los ejes X y Y corresponden a profundidad y ancho respectivamente y el eje Z corresponde a la altura (ver Figura 6), de modo que las

capas, entendiéndose por capa como una sección del cubo que puede girar respecto a un eje determinado, le deben sus nombres a cada uno de estos ejes. Como se sabe que para cada eje hay tres capas, se hizo la diferenciación de cada una con números del 1 al 3 en orden creciente desde el origen hacia la dirección de cada eje (ver Figura 6). Como un ejemplo, la capa inferior del Cubo de Rubik corresponde a la capa Z1 (ver Figura 7).

Figura 6: Notación de los ejes Cartesianos y capas del Cubo de Rubik

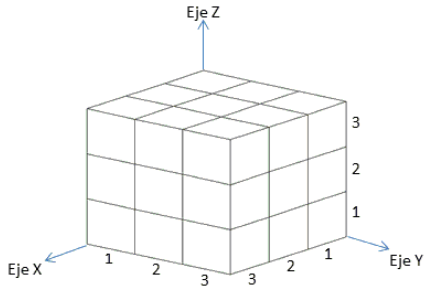


Figura 7: Capa Z1

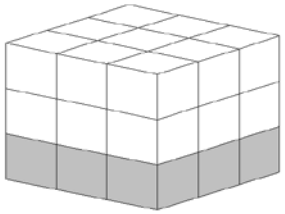
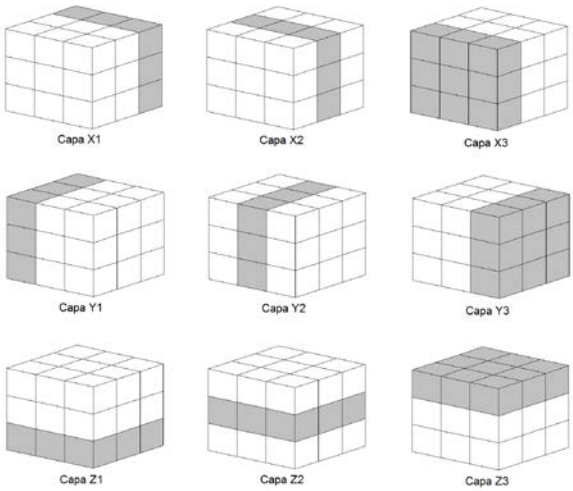


Figura 8: Notación de las capas del Cubo de Rubik



Como en un cubo de Rubik las capas X2, Y2 y Z2 son consideradas capas estáticas, las capas que rotan son las que tienen los números 1 y 3 (ver Figura 8), todos los diseños tanto del algoritmo como de los movimientos del robot, están hechos en base a doce movimientos básicos de las rotaciones posibles de las capas correspondientes a las rotaciones en sentido horario y anti horario de cada una de las seis capas que pueden rotar y seis rotaciones posibles del cubo completo en los tres ejes con sus dos sentidos de rotación para cada uno.

6.2.1. Robot. Teniendo ya las notaciones establecidas para cada capa, se empezó a diseñar el programa pensando en la utilización de un segundo brazo robótico que funcionaría como base y se encargaba de realizar dos tareas simples: sostener el cubo y rotarlo de acuerdo al eje Z. Siendo así, se implementaría un sistema maestro-esclavo donde la información proveniente del computador entraría al robot maestro y éste sería quien le indicara al otro robot qué hacer (ver Resultados y Discusión).

De este modo se ideó un método para realizar la comunicación entre los robots. Como cada uno posee puertos de entrada y salida binarios, se pretendía utilizar varias de esas entradas para enviar datos en paralelo entre uno y el otro de modo que, de acuerdo a lo que el robot "maestro" necesitaba, se enviaba una señal que le dijera al robot "esclavo" si debía girar o no y en qué sentido hacerlo, y también si abrir o cerrar la herramienta del robot. Siendo así, se generaron unos programas que funcionaran como métodos que contuvieran las secuencias de movimientos que necesitaban hacer los robots para rotar las caras del cubo (incluyendo la comunicación entre los robots) y luego se hizo un programa principal que tenía un ciclo de repetición donde se preguntaba por qué movimiento hacer con este diseño surgió el diagrama de flujo presentado en el Anexo B (ver Anexo B). Estos programas quedaban todos almacenados dentro de controlador propio del robot y se escribieron según el lenguaje que éste soporta (ver Resultados y Discusión).

Finalmente se decidió utilizar solamente un robot con una base estática la cual sostenía la capa inferior del cubo y se acoplaron los movimientos del robot para que todas las rotaciones del cubo se hicieran por medio de dicha base y era el computador quien indicaba al robot qué movimiento hacer.

6.2.2. Visión Artificial. Al principio del proceso se hicieron unos análisis referentes al espacio de color para decidir cuál utilizar y se decidió que el mejor espacio de color sería el YUV ya que la cámara lo utilizaba de forma nativa y por las ventajas que tiene en controlar la luminosidad.

Durante este proceso se estaban realizando estudios académicos en sistemas inteligentes y se pudo ver que las redes neuronales son una técnica bastante apropiada para lo que se está tratando, no solo por la facilidad de uso que tienen

estos sistemas sino también por el gran potencial que tienen al ser una imitación de lo que es el funcionamiento del cerebro humano en relación con la visión).

De este modo se diseñó un programa de visión artificial escrito en Matlab, ya que este software IDE tiene soporte para redes neuronales, comunicación serial y manejo de cámaras web. Se hicieron distintas pruebas, entre ellas se analizó el tipo de red neuronal a utilizar, la manera en que la información entregaría el resultado, se analizaron los colores a detectar en distintas condiciones de luz y otras pruebas para encontrar la técnica de entrenamiento de redes neuronales más apropiada para la ocasión haciendo uso de la información adquirida con las pruebas anteriores (ver Anexos).

Todas estas pruebas dieron como resultado un sistema bastante confiable, lo cual hizo que no se buscaran más posibilidades para sistemas de visión sino que se buscó hacer mejoras en el programa ya diseñado.

6.2.3. Algoritmo de solución. Para conseguir el objetivo de demostrar que efectivamente el sistema de visión artificial se puede integrar con un robot llevándolo a ejecutar una tarea de acuerdo a lo sensado, se planteó la idea de solucionar un cubo de Rubik detectado por medio del sistema de visión artificial.

Para esto se necesitó aprender sobre cómo armar el cubo de forma manual y se encontró uno de los algoritmos más sencillos para la solución que es el algoritmo inventado por el matemático David Singmaster[12]. Pues dada su sencillez era más fácil de aprender y mucho más fácil de implementar en un código computacional. Una vez que se tuvo conocimiento sobre el algoritmo se buscó demostrar que con el robot es posible realizarlo armando por lo menos la capa superior que es la que necesita un mayor número de movimientos para solucionarla (ver Anexos).

6.2.4. Acople. Una vez terminado el sistema de visión artificial y los movimientos del robot, se buscó una forma en que se pudieran comunicar el robot y el computador y se pensó inicialmente en una tarjeta de adquisición de datos la cual le enviaría datos al robot en las entradas de éste y se terminó con el sistema de comunicación serial puesto que este es el sistema que el robot utiliza.

Para esto se tuvo que analizar la manera en que el robot recibe los datos, teniendo en cuenta si utiliza bit de paridad, tasa de baudios (Baud Rate), cómo debe ser la trama, qué caracteres definen el final de la trama, entre otros. Con

[12] SINGMASTER, David. Notes on Rubik's Magic Cube. Hillside, New Jersey : Library of Congress, 1981. p. 21-63.

toda esta información se generó un nuevo programa que contenía el sistema de visión artificial y el algoritmo para solucionar el cubo.

Por último, se realizó la parte más importante del proyecto que era conectar directamente el sistema de visión con el robot. Aquí se hace la adquisición de los datos de cada una de las caras que deben ser posicionadas por el robot de una forma correcta frente a la cámara de modo que ésta tome una foto y se envíe al sistema de procesamiento quien se encarga del análisis para determinar qué colores se encuentran en ese momento.

7. RESULTADOS

De acuerdo a la metodología utilizada para el diseño del sistema, a continuación se muestran los resultados:

Al diseñarse los movimientos del robot en una primera instancia donde se tenía en cuenta que serían dos robots manipuladores, se encontró que existen incompatibilidades en el robot SV3 para anidar programas dentro de otros.

Se encontró que la mejor forma de realizar los movimientos del robot, debido a las condiciones actuales, es construyendo una base estática en la cual sostener el cubo y no con dos robots manipuladores.

Al buscar el espacio de color más apropiado para el sistema de visión artificial se encontró que es más conveniente el espacio YUV.

Investigando sobre qué técnica utilizar para el procesamiento de las imágenes se encontró que era muy apropiado utilizar redes neuronales artificiales.

La información saliente de la red neuronal se basó en las probabilidades, de modo que al hacer el análisis sobre un color detectado, las distintas salidas de la red entregaban una probabilidad sobre qué color podría ser y el resultado final entregado era l de mayor probabilidad.

El algoritmo utilizado para el entrenamiento fue el de la clasificación Bayesiana, la cual dio resultados supremamente satisfactorios ya que se logró alcanzar unos niveles mínimos de error (ver Anexos).

Al implementarse la red neuronal en el sistema de procesamiento se vio que el sistema era capaz de reconocer los colores aun así habiendo cambios en la luminosidad.

Haciendo un análisis detallado se encontró que el algoritmo de solución más sencillo para implementar en código es el algoritmo inventado por el matemático David Singmaster.

Se hizo una simulación sobre dicho algoritmo en Matlab y se solucionó correctamente la capa superior.

En cuanto a la comunicación, lo más adecuado es utilizar el sistema que trae el robot por defecto que es RS232. Se buscó sobre las configuraciones que se necesitarían y al hacerse conforme a los parámetros del robot, la comunicación fue exitosa.

Al integrarse todos los sistemas ya probados, se pudo ver que la capa superior del cubo fue concluida correctamente.

8. CONCLUSIONES

Con la realización de este proyecto de grado se pudo concluir que:

La visión artificial es un sistema muy complejo que se ha ido desarrollando con los años y que no ha terminado de desarrollarse. Si bien aquí se hicieron unas detecciones de color en un cubo de Rubik, esto es solamente una de las componentes de lo que es visión artificial

La cámara es un sensor que graba imágenes sobre algo que se encuentra en frente del lente pero lo importante no está en haber detectado o almacenado una imagen sobre lo que hay al frente, sino procesar la imagen para saber exactamente qué es lo que está allí y, de acuerdo a ello, tomar decisiones para efectuar alguna acción.

Si bien el sistema de visión artificial es bastante confiable para el reconocimiento de colores, se pudo ver que, más que tener unos colores determinados, es importante tener unas buenas condiciones de luz.

Analizando la precepción humana, la forma en que el cerebro aprende y todo esto relacionado con la visión, se pudo tener un profundo entendimiento del potencial tan grande que hay en una red neuronal artificial no tanto para detectar colores ni formas sino también para ésta aprenda todas esas teorías que hay sobre la visión que como se dijo anteriormente, más que ingeniería requieren es de conocimiento en psicología para entender qué ocurre con ello y que todo el aprendizaje está dado por la experiencia.

El sistema de red neuronal artificial es un sistema tan robusto que las condiciones en las cuales éste puede trabajar no tienen que ser condiciones perfectas. Se hicieron pruebas con dos cámaras web que son de bajo costo y se hicieron cambios de escenario y aun así el sistema continuó funcionando sin hacer una gran cantidad de ajustes. De hecho el sistema de la forma en la que está actualmente se puede garantizar que la detección tiene un alto porcentaje de fiabilidad en muchos escenarios diferentes.

9. RECOMENDACIONES

Para un buen comienzo, es muy apropiado tener un punto de partida. De modo que si se van a diseñar muchos métodos para integrar entre un robot y un computador, sus nombres sean similares si entre ellos hay una relación.

El robot Intelitek Performer SV3 no tiene soporte para anidar programas dentro de otros, es recomendable diseñar los programas de otra manera o revisar algún método alternativo para activar programas en la guía de referencia del lenguaje ACL.

Se debe tener en cuenta que siempre que se inicie a trabajar con dicho robot, se debe ubicar manualmente de acuerdo a los indicadores que éste tiene y programar el estado “home” por medio del computador.

Cuando se haga una futura comunicación RS232 con otro programa diferente a un terminal de comunicaciones, se debe tener en cuenta que la tasa de baudios (baud rate) para este robot es de 19200 y que el carácter terminal de la trama es el retorno de carro que codificado en ASCII corresponde al número 13. Adicional a esto se debe tener en cuenta el tamaño de los buffers de entrada y de salida.

En el momento de implementar un algoritmo tan tedioso como el de solución de un cubo de Rubik, es recomendable utilizar métodos que se encarguen cada uno de ejecutar una tarea específica para tener más facilidad en programar toda la secuencia de movimientos.

Si se va a hacer una comunicación continua y automática entre el computador y el robot es recomendable que al final de cada programa del robot se envíe una cadena de caracteres como indicador de que éste ha terminado.

Al hacer el entrenamiento de la red neuronal se recomienda que se hagan muchas capturas para los colores, esto permitirá que se amplíe la gama de luminosidades que el sistema admitirá y así el sistema será mucho más confiable.

Se debe tener en cuenta que la comunicación entre un dispositivo y el otro sea pausada para evitar errores de funcionamiento debido al envío de datos en un momento inoportuno.

BIBLIOGRAFÍA

ACL: Advanced Control Languaje, reference Guide. [En línea] 2011 [consultado 24 de Marzo de 2011]. Disponible en Internet: http://elabz.com/wp-content/uploads/2010/04/100083-a-ACLV1.43-F44-Ctrl-A.pdf.

AYACHE, Nicholas. Artificial vision for mobile robots: stereo vision and multisensory perception. Cambridge : InterEditions, 1991. 342 p.

COMPUTER VISION AND ROBOTICS GROUP. Universidad de Cambridge. 3D Shape from Uncalibrated Images. [En línea] 2011. [consultado 24 de Marzo de 2011]. Disponible en Internet: http://mi.eng.cam.ac.uk/~cipolla/research.htm.

GONZÁLES, Rafael C. y WOODS, Richard E. Digital Image Processing. Upper Saddle River : Pearson Prentice Hall, 2008. 797 p.

LINGGARD, Robert, MYERS, D. y NIGHTINGALE, C. Neural Networks for Vision, Speech and Natural Language. Londres, Reino Unido : Chapman & Hall, Ltd., 1992. 442 p.

LÓPEZ, Jesús A. y CAICEDO, Eduardo F. Una Aproximación Práctica a las Redes Neuronales Artificiales. Cali : Editorial Universidad del Valle, 2009. 217 p.

ROBOTICS INSTITUTE. The Robotics Institute. Face Detection. [En línea] 2011. [consultado 24 de Marzo de 2011]. Disponible en Internet: http://www.ri.cmu.edu/research_project_detail.html?project_id=416&menu_id=261.

SICILIANO, Bruno, y otros, y otros. Robotics: modelling, planning and control. Londres : Springer, 2010. 644 p.

SINGMASTER, David. Notes on Rubik's Magic Cube. Hillside, New Jersey : Library of Congress, 1981. 80 p.

University of New Mexico. Electrical and Computer Engineering Department. Faculty Spotlight: Prof. Caudell and the Meeting of Minds. [En línea] 2011. [consultado 22 de Marzo de 2011]. Disponible en Internet: http://www.ece.unm.edu/morenews/profile_caudell.html.

ANEXOS

Anexo A: Código fuente de los programas del robot

El código aquí mostrado es el mismo que hay almacenado en el controlador del robot Intelitek Performer SV3. Los nombres para cada programa están escritos en tres partes, la primera parte que es “rub” el cual corresponde al diferencial que hay entre los programas del Cubo de Rubik y demás programas que puedan haber almacenados. La segunda parte corresponde al tipo de rotación en su respectivo eje, utilizando la notación establecida en la metodología, y la tercera parte corresponde al sentido de giro siento en sentido de las manecillas del reloj (“man”) o contrario a las manecillas del reloj (“cont”) o si no existe tercera parte, significa que se da un giro de 180°

ROTACIONES EN LOS EJES PARA EL CUBO

Los siguientes programas corresponden a las rotaciones que se pueden hacer al cubo respecto a un eje determinado, el formato del nombre está conforme a lo explicado anteriormente.

```
        PROGRAM   RUBX
        **********************
1500: OPEN
1501: MOVED     ARRIBA[1]
1502: MOVED     CAPAMEDIA[1]
1503: CLOSE
1504: MOVED     ARRIBA[1]
1505: MOVED     ARRIBA[4]
1506: MOVED     CAPAMEDIA[4]
1507: OPEN
1508: MOVED     ARRIBA[4]
1509: SPLINED   VECTADEL 1 4
1510: CLOSE
1511: SPLINED   VECTADEL 4 1
1512: MOVED     CAPAMEDIA[1]
1513: OPEN
1514: SPLINED   VECTADEL 1 4
1515: CLOSE
1516: SPLINED   VECTADEL 4 1
1517: MOVED     ARRIBA[2]
1518: MOVED     CAPAMEDIA[2]
1519: OPEN
1520: MOVED     ARRIBA[2]
1521: MOVED     ARRIBA[1]
1522: PRINT     "ya"
1523: END

        PROGRAM   RUBXMAN
        **********************
1545: OPEN
1546: MOVED     ARRIBA[1]
1547: MOVED     ARRIBA[2]
```

```
1548: MOVED    CAPAMEDIA[2]
1549: CLOSE
1550: MOVED    ARRIBA[2]
1551: SPLINED  VECTADEL 1 4
1552: OPEN
1553: SPLINED  VECTADEL 4 1
1554: MOVED    CAPAMEDIA[1]
1555: CLOSE
1556: MOVED    ARRIBA[1]
1557: MOVED    ARRIBA[2]
1558: MOVED    CAPAMEDIA[2]
1559: OPEN
1560: MOVED    ARRIBA[2]
1561: MOVED    ARRIBA[1]
1562: PRINT    "ya"
1563: END
```

PROGRAM RUBXCONT
****

```
1525: OPEN
1526: MOVED    ARRIBA[1]
1527: MOVED    CAPAMEDIA[1]
1528: CLOSE
1529: MOVED    ARRIBA[1]
1530: MOVED    ARRIBA[4]
1531: MOVED    CAPAMEDIA[4]
1532: OPEN
1533: MOVED    ARRIBA[4]
1534: SPLINED  VECTADEL 1 4
1535: CLOSE
1536: SPLINED  VECTADEL 4 1
1537: MOVED    ARRIBA[2]
1538: MOVED    CAPAMEDIA[2]
1539: OPEN
1540: MOVED    ARRIBA[2]
1541: MOVED    ARRIBA[1]
1542: PRINT    "ya"
1543: END
```

PROGRAM RUBY
****

```
1565: OPEN
1566: MOVED    ARRIBA[1]
1567: MOVED    CAPAMEDIA[1]
1568: CLOSE
1569: SPLINED  VECTADEL 1 4
1570: OPEN
1571: SPLINED  VECTADEL 4 1
1572: MOVED    CAPAMEDIA[1]
1573: CLOSE
1574: SPLINED  VECTADEL 1 4
1575: OPEN
1576: SPLINED  VECTADEL 4 1
1577: MOVED    ARRIBA[1]
1578: PRINT    "ya"
1579: END
```

PROGRAM RUBYMAN
****

```
1592: OPEN
1593: MOVED    ARRIBA[1]
1594: SPLINED  VECTADEL 1 4
1595: CLOSE
1596: SPLINED  VECTADEL 4 1
1597: MOVED    CAPAMEDIA[1]
1598: OPEN
1599: MOVED    ARRIBA[1]
1600: PRINT    "ya"
1601: END
```

PROGRAM RUBYCONT
****

```
1581: OPEN
1582: MOVED    ARRIBA[1]
1583: MOVED    CAPAMEDIA[1]
1584: CLOSE
1585: SPLINED  VECTADEL 1 4
1586: OPEN
1587: SPLINED  VECTADEL 4 1
1588: MOVED    ARRIBA[1]
1589: PRINT    "ya"
1590: END
```

```
        PROGRAM   RUBZ
        **********
1603: OPEN
1604: MOVED    ARRIBA[1]
1605: MOVED    CAPAMEDIA[1]
1606: CLOSE
1607: MOVED    ARRIBA[1]
1608: MOVED    ARRIBA[3]
1609: MOVED    CAPAMEDIA[3]
1610: OPEN
1611: MOVED    ARRIBA[3]
1612: MOVED    ARRIBA[1]
1613: PRINT    "ya"
1614: END

        PROGRAM   RUBZMAN
        **********
1616: OPEN
1617: MOVED    ARRIBA[1]
1618: MOVED    CAPAMEDIA[1]
1619: CLOSE
1620: MOVED    ARRIBA[1]
1621: MOVED    ARRIBA[4]
1622: MOVED    CAPAMEDIA[4]
1623: OPEN
1624: MOVED    ARRIBA[4]
1625: MOVED    ARRIBA[1]
1626: PRINT    "ya"
1627: END

        PROGRAM   RUBZCONT
        **********
1629: OPEN
1630: MOVED    ARRIBA[1]
1631: MOVED    CAPAMEDIA[1]
1632: CLOSE
1633: MOVED    ARRIBA[2]
1634: MOVED    CAPAMEDIA[2]
1635: OPEN
1636: MOVED    ARRIBA[2]
1637: MOVED    ARRIBA[1]
1638: PRINT    "ya"
1639: END
```

ROTACIONES DE CADA CAPA DEL CUBO

Los siguientes programas corresponden a las rotaciones que se pueden hacer a cada capa respecto a su propio eje, el formato del nombre está conforme a lo explicado anteriormente y los número 1 y 3 de cada capa corresponden a los mismos establecidos en la metodología.

```
        PROGRAM   RUBX1
        **********
1092: OPEN
1093: MOVED    ARRIBA[1]
1094: SPLINED  VECTADEL 1 4
1095: CLOSE
1096: SPLINED  VECTADEL 4 1
1097: MOVED    CAPAMEDIA[1]
1098: MOVED    CAPAMEDIA[3]
1099: MOVED    ARRIBA[3]
1100: SPLINED  VECTADEL 1 4
1101: OPEN
1102: SPLINED  VECTADEL 4 1
1103: MOVED    ARRIBA[1]
1104: PRINT    "ya"
1105: END

        PROGRAM   RUBX1MAN
        **********
1062: OPEN
```

```
1063: MOVED     ARRIBA[1]
1064: SPLINED   VECTADEL 1 4
1065: CLOSE
1066: SPLINED   VECTADEL 4 1
1067: MOVED     CAPAMEDIA[1]
1068: MOVED     CAPAMEDIA[4]
1069: MOVED     ARRIBA[4]
1070: SPLINED   VECTADEL 1 4
1071: OPEN
1072: SPLINED   VECTADEL 4 1
1073: MOVED     ARRIBA[1]
1074: PRINT     "ya"
1075: END

        PROGRAM   RUBX1CONT
        **********************
1077: OPEN
1078: MOVED     ARRIBA[1]
1079: SPLINED   VECTADEL 1 4
1080: CLOSE
1081: SPLINED   VECTADEL 4 1
1082: MOVED     CAPAMEDIA[1]
1083: MOVED     CAPAMEDIA[2]
1084: MOVED     ARRIBA[2]
1085: SPLINED   VECTADEL 1 4
1086: OPEN
1087: SPLINED   VECTADEL 4 1
1088: MOVED     ARRIBA[1]
1089: PRINT     "ya"
1090: END

        PROGRAM   RUBX3
        **********************
1107: OPEN
1108: MOVED     ARRIBA[1]
1109: MOVED     CAPAMEDIA[1]
1110: CLOSE
1111: MOVED     VECTADEL[1]
1112: SPLINED   VECTADEL 1 4
1113: OPEN
1114: MOVED     VECTADEL[4]
1115: SPLINED   VECTADEL 4 1
1116: MOVED     CAPAMEDIA[1]
1117: CLOSE
1118: MOVED     CAPAMEDIA[3]
1119: MOVED     ARRIBA[3]
1120: MOVED     ARRIBA[1]
1121: MOVED     CAPAMEDIA[1]
1122: OPEN
1123: MOVED     VECTADEL[1]
1124: SPLINED   VECTADEL 1 4
1125: CLOSE
1126: MOVED     VECTADEL[4]
1127: SPLINED   VECTADEL 4 1
1128: MOVED     CAPAMEDIA[1]
1129: OPEN
1130: MOVED     ARRIBA[1]
1131: PRINT     "ya"
1132: END

        PROGRAM   RUBX3MAN
        **********************
1134: OPEN
1135: MOVED     ARRIBA[1]
1136: MOVED     CAPAMEDIA[1]
1137: CLOSE
1138: MOVED     VECTADEL[1]
1139: SPLINED   VECTADEL 1 4
1140: OPEN
1141: MOVED     VECTADEL[4]
1142: SPLINED   VECTADEL 4 1
1143: MOVED     CAPAMEDIA[1]
1144: CLOSE
1145: MOVED     CAPAMEDIA[4]
1146: MOVED     ARRIBA[4]
1147: MOVED     ARRIBA[1]
1148: MOVED     CAPAMEDIA[1]
1149: OPEN
1150: MOVED     VECTADEL[1]
1151: SPLINED   VECTADEL 1 4
1152: CLOSE
1153: MOVED     VECTADEL[4]
```

```
1154: SPLINED   VECTADEL 4 1
1155: MOVED     CAPAMEDIA[1]
1156: OPEN
1157: MOVED     ARRIBA[1]
1158: PRINT     "ya"
1159: END

        PROGRAM   RUBX3CONT
         **********************
1161: OPEN
1162: MOVED     ARRIBA[1]
1163: MOVED     CAPAMEDIA[1]
1164: CLOSE
1165: MOVED     VECTADEL[1]
1166: SPLINED   VECTADEL 1 4
1167: OPEN
1168: MOVED     VECTADEL[4]
1169: SPLINED   VECTADEL 4 1
1170: MOVED     CAPAMEDIA[1]
1171: CLOSE
1172: MOVED     CAPAMEDIA[2]
1173: MOVED     ARRIBA[2]
1174: MOVED     ARRIBA[1]
1175: MOVED     CAPAMEDIA[1]
1176: OPEN
1177: MOVED     VECTADEL[1]
1178: SPLINED   VECTADEL 1 4
1179: CLOSE
1180: MOVED     VECTADEL[4]
1181: SPLINED   VECTADEL 4 1
1182: MOVED     CAPAMEDIA[1]
1183: OPEN
1184: MOVED     ARRIBA[1]
1185: PRINT     "ya"
1186: END

        PROGRAM   RUBY1
        **********************
1217: OPEN
1218: MOVED     ARRIBA[1]
1219: MOVED     ARRIBA[2]
1220: MOVED     CAPAMEDIA[2]
1221: CLOSE
1222: MOVED     ARRIBA[2]
1223: MOVED     ARRIBA[1]
1224: MOVED     CAPAMEDIA[1]
1225: OPEN
1226: SPLINED   VECTADEL 1 4
1227: CLOSE
1228: SPLINED   VECTADEL 4 1
1229: MOVED     CAPAMEDIA[1]
1230: MOVED     CAPAMEDIA[3]
1231: MOVED     ARRIBA[3]
1232: SPLINED   VECTADEL 1 4
1233: OPEN
1234: SPLINED   VECTADEL 4 1
1235: MOVED     CAPAMEDIA[1]
1236: CLOSE
1237: MOVED     ARRIBA[1]
1238: MOVED     ARRIBA[2]
1239: MOVED     CAPAMEDIA[2]
1240: OPEN
1241: MOVED     ARRIBA[2]
1242: MOVED     ARRIBA[1]
1243: PRINT     "ya"
1244: END

          PROGRAM   RUBY1MAN
          **********************
1275: OPEN
1276: MOVED     ARRIBA[1]
1277: MOVED     ARRIBA[2]
1278: MOVED     CAPAMEDIA[2]
1279: CLOSE
1280: MOVED     ARRIBA[2]
1281: MOVED     ARRIBA[1]
1282: MOVED     CAPAMEDIA[1]
1283: OPEN
1284: SPLINED   VECTADEL 1 4
1285: CLOSE
1286: SPLINED   VECTADEL 4 1
1287: MOVED     CAPAMEDIA[1]
```

```
1288: MOVED    CAPAMEDIA[4]
1289: MOVED    ARRIBA[4]
1290: SPLINED  VECTADEL 1 4
1291: OPEN
1292: SPLINED  VECTADEL 4 1
1293: MOVED    CAPAMEDIA[1]
1294: CLOSE
1295: MOVED    ARRIBA[1]
1296: MOVED    ARRIBA[2]
1297: MOVED    CAPAMEDIA[2]
1298: OPEN
1299: MOVED    ARRIBA[2]
1300: MOVED    ARRIBA[1]
1301: PRINT    "ya"
1302: END

      PROGRAM  RUBY1CONT
        *********************
1246: OPEN
1247: MOVED    ARRIBA[1]
1248: MOVED    ARRIBA[2]
1249: MOVED    CAPAMEDIA[2]
1250: CLOSE
1251: MOVED    ARRIBA[2]
1252: MOVED    ARRIBA[1]
1253: MOVED    CAPAMEDIA[1]
1254: OPEN
1255: SPLINED  VECTADEL 1 4
1256: CLOSE
1257: SPLINED  VECTADEL 4 1
1258: MOVED    CAPAMEDIA[1]
1259: MOVED    CAPAMEDIA[2]
1260: MOVED    ARRIBA[2]
1261: PLINED   VECTADEL 1 4
1262: OPEN
1263: SPLINED  VECTADEL 4 1
1264: MOVED    CAPAMEDIA[1]
1265: CLOSE
1266: MOVED    ARRIBA[1]
1267: MOVED    ARRIBA[2]
1268: MOVED    CAPAMEDIA[2]
1269: OPEN
1270: MOVED    ARRIBA[2]
1271: MOVED    ARRIBA[1]
1272: PRINT    "ya"
1273: END

         PROGRAM  RUBY3
         **********************
1188: OPEN
1189: MOVED    ARRIBA[1]
1190: MOVED    ARRIBA[4]
1191: MOVED    CAPAMEDIA[4]
1192: CLOSE
1193: MOVED    ARRIBA[4]
1194: MOVED    ARRIBA[1]
1195: MOVED    CAPAMEDIA[1]
1196: OPEN
1197: SPLINED  VECTADEL 1 4
1198: CLOSE
1199: SPLINED  VECTADEL 4 1
1200: MOVED    CAPAMEDIA[1]
1201: MOVED    CAPAMEDIA[3]
1202: MOVED    ARRIBA[3]
1203: SPLINED  VECTADEL 1 4
1204: OPEN
1205: SPLINED  VECTADEL 4 1
1206: MOVED    CAPAMEDIA[1]
1207: CLOSE
1208: MOVED    ARRIBA[1]
1209: MOVED    ARRIBA[4]
1210: MOVED    CAPAMEDIA[4]
1211: OPEN
1212: MOVED    ARRIBA[4]
1213: MOVED    ARRIBA[1]
1214: PRINT    "ya"
1215: END

         PROGRAM  RUBY3MAN
         **********************
1304: OPEN
1305: MOVED    ARRIBA[1]
```

```
1306: MOVED    ARRIBA[4]
1307: MOVED    CAPAMEDIA[4]
1308: CLOSE
1309: MOVED    ARRIBA[4]
1310: MOVED    ARRIBA[1]
1311: MOVED    CAPAMEDIA[1]
1312: OPEN
1313: SPLINED  VECTADEL 1 4
1314: CLOSE
1315: SPLINED  VECTADEL 4 1
1316: MOVED    CAPAMEDIA[1]
1317: MOVED    CAPAMEDIA[4]
1318: MOVED    ARRIBA[4]
1319: SPLINED  VECTADEL 1 4
1320: OPEN
1321: SPLINED  VECTADEL 4 1
1322: MOVED    CAPAMEDIA[1]
1323: CLOSE
1324: MOVED    ARRIBA[1]
1325: MOVED    ARRIBA[4]
1326: MOVED    CAPAMEDIA[4]
1327: OPEN
1328: MOVED    ARRIBA[4]
1329: MOVED    ARRIBA[1]
1330: PRINT    "ya"
1331: END

        PROGRAM  RUBY3CONT
        **********************
1333: OPEN
1334: MOVED    ARRIBA[1]
1335: MOVED    ARRIBA[4]
1336: MOVED    CAPAMEDIA[4]
1337: CLOSE
1338: MOVED    ARRIBA[4]
1339: MOVED    ARRIBA[1]
1340: MOVED    CAPAMEDIA[1]
1341: OPEN
1342: SPLINED  VECTADEL 1 4
1343: CLOSE
1344: SPLINED  VECTADEL 4 1
1345: MOVED    CAPAMEDIA[1]
1346: MOVED    CAPAMEDIA[2]
1347: MOVED    ARRIBA[2]
1348: SPLINED  VECTADEL 1 4
1349: OPEN
1350: SPLINED  VECTADEL 4 1
1351: MOVED    CAPAMEDIA[1]
1352: CLOSE
1353: MOVED    ARRIBA[1]
1354: MOVED    ARRIBA[4]
1355: MOVED    CAPAMEDIA[4]
1356: OPEN
1357: MOVED    ARRIBA[4]
1358: MOVED    ARRIBA[1]
1359: PRINT    "ya"
1360: END

        PROGRAM  RUBZ1
        **********************
1389: OPEN
1390: MOVED    ARRIBA[1]
1391: MOVED    CAPAMEDIA[1]
1392: CLOSE
1393: MOVED    CAPAMEDIA[3]
1394: MOVED    ARRIBA[3]
1395: MOVED    ARRIBA[1]
1396: MOVED    CAPAMEDIA[1]
1397: OPEN
1398: MOVED    ARRIBA[1]
1399: PRINT    "ya"
1400: END

        PROGRAM  RUBZ1MAN
        **********************
1402: OPEN
1403: MOVED    ARRIBA[1]
1404: MOVED    CAPAMEDIA[1]
1405: CLOSE
1406: MOVED    CAPAMEDIA[4]
1407: MOVED    ARRIBA[4]
1408: MOVED    ARRIBA[1]
```

```
1409: MOVED     CAPAMEDIA[1]
1410: OPEN
1411: MOVED     ARRIBA[1]
1412: PRINT     "ya"
1413: END
```

```
        PROGRAM   RUBZ1CONT
         **********************
1415: OPEN
1416: MOVED     ARRIBA[1]
1417: MOVED     CAPAMEDIA[1]
1418: CLOSE
1419: MOVED     CAPAMEDIA[2]
1420: MOVED     ARRIBA[2]
1421: MOVED     ARRIBA[1]
1422: MOVED     CAPAMEDIA[1]
1423: OPEN
1424: MOVED     ARRIBA[1]
1425: PRINT     "ya"
1426: END
```

```
        PROGRAM   RUBZ3
         **********************
1428: OPEN
1429: MOVED     ARRIBA[1]
1430: SPLINED   VECTADEL 1 4
1431: CLOSE
1432: SPLINED   VECTADEL 4 1
1433: MOVED     CAPAMEDIA[1]
1434: OPEN
1435: SPLINED   VECTADEL 1 4
1436: CLOSE
1437: SPLINED   VECTADEL 4 1
1438: MOVED     CAPAMEDIA[1]
1439: MOVED     CAPAMEDIA[3]
1440: MOVED     ARRIBA[3]
1441: SPLINED   VECTADEL 1 4
1442: OPEN
1443: SPLINED   VECTADEL 4 1
1444: MOVED     CAPAMEDIA[1]
1445: CLOSE
```

```
1446: SPLINED   VECTADEL 1 4
1447: OPEN
1448: SPLINED   VECTADEL 4 1
1449: MOVED     ARRIBA[1]
1450: END
```

```
        PROGRAM   RUBZ3MAN
         **********************
1452: OPEN
1453: MOVED     ARRIBA[1]
1454: SPLINED   VECTADEL 1 4
1455: CLOSE
1456: SPLINED   VECTADEL 4 1
1457: MOVED     CAPAMEDIA[1]
1458: OPEN
1459: SPLINED   VECTADEL 1 4
1460: CLOSE
1461: SPLINED   VECTADEL 4 1
1462: MOVED     CAPAMEDIA[1]
1463: MOVED     CAPAMEDIA[4]
1464: MOVED     ARRIBA[4]
1465: SPLINED   VECTADEL 1 4
1466: OPEN
1467: SPLINED   VECTADEL 4 1
1468: MOVED     CAPAMEDIA[1]
1469: CLOSE
1470: SPLINED   VECTADEL 1 4
1471: OPEN
1472: SPLINED   VECTADEL 4 1
1473: MOVED     ARRIBA[1]
1474: END
```

```
        PROGRAM   RUBZ3CONT
         **********************
1476: OPEN
1477: MOVED     ARRIBA[1]
1478: SPLINED   VECTADEL 1 4
1479: CLOSE
1480: SPLINED   VECTADEL 4 1
1481: MOVED     CAPAMEDIA[1]
1482: OPEN
```

```
1483: SPLINED   VECTADEL 1 4
1484: CLOSE
1485: SPLINED   VECTADEL 4 1
1486: MOVED     CAPAMEDIA[1]
1487: MOVED     CAPAMEDIA[2]
1488: MOVED     ARRIBA[2]
1489: SPLINED   VECTADEL 1 4
1490: OPEN
1491: SPLINED   VECTADEL 4 1
1492: MOVED     CAPAMEDIA[1]
1493: CLOSE
1494: SPLINED   VECTADEL 1 4
1495: OPEN
1496: SPLINED   VECTADEL 4 1
1497: MOVED     ARRIBA[1]
1498: END
```

MOVIMIENTOS PARA POSICIONAR CADA CARA DEL CUBO FRENTE A LA CÁMARA

```
        PROGRAM   CARA1
        **********************
1641: OPEN
1642: MOVED     ARRIBA[1]
1643: MOVED     CAPAMEDIA[1]
1644: CLOSE
1645: MOVED     ARRIBA[1]
1646: DELAY     120
1647: PRINT     "ya"
1648: END

        PROGRAM   CARA1DEV
        **********************
1708: MOVED     CAPAMEDIA[1]
1709: OPEN
1710: MOVED     ARRIBA[1]
1711: DELAY     120
1712: PRINT     "ya"
1713: END

        PROGRAM   CARA2
        **********************
1650: OPEN
1651: MOVED     ARRIBA[1]
1652: MOVED     ARRIBA[2]
1653: MOVED     CAPAMEDIA[2]
1654: CLOSE
1655: MOVED     ARRIBA[2]
1656: MOVED     ARRIBA[1]
1657: PRINT     "ya"
1658: END

        PROGRAM   CARA2DEV
        **********************
1715: MOVED     ARRIBA[2]
1716: MOVED     CAPAMEDIA[2]
1717: OPEN
1718: MOVED     ARRIBA[2]
1719: MOVED     ARRIBA[1]
1720: PRINT     "ya"
1721: END

        PROGRAM   CARA3
        **********************
1660: OPEN
1661: MOVED     ARRIBA[1]
1662: MOVED     CAPAMEDIA[1]
1663: CLOSE
1664: MOVED     ARRIBA[1]
1665: MOVED     ARRIBA[3]
1666: DELAY     120
1667: PRINT     "ya"
1668: END

        PROGRAM   CARA3DEV
        **********************
```

```
1723: MOVED     ARRIBA[1]
1724: MOVED     CAPAMEDIA[1]
1725: OPEN
1726: MOVED     ARRIBA[1]
1727: DELAY     120
1728: PRINT     "ya"
1729: END

        PROGRAM   CARA4
        **********************
1670: OPEN
1671: MOVED     ARRIBA[1]
1672: MOVED     ARRIBA[4]
1673: MOVED     CAPAMEDIA[4]
1674: CLOSE
1675: MOVED     ARRIBA[4]
1676: MOVED     ARRIBA[1]
1677: PRINT     "ya"
1678: END

        PROGRAM   CARA4DEV
        **********************
1731: MOVED     ARRIBA[4]
1732: MOVED     CAPAMEDIA[4]
1733: OPEN
1734: MOVED     ARRIBA[4]
1735: MOVED     ARRIBA[1]
1736: PRINT     "ya"
1737: END

        PROGRAM   CARA5
        **********************
1680: OPEN
1681: MOVED     ARRIBA[1]
1682: MOVED     CAPAMEDIA[1]
1683: CLOSE
1684: SPLINED   VECTADEL 1 4
1685: OPEN
1686: SPLINED   VECTADEL 4 1
1687: MOVED     CAPAMEDIA[1]
1688: CLOSE
1689: MOVED     ARRIBA[1]
1690: PRINT     "ya"
1691: END

        PROGRAM   CARA5DEV
        **********************
1739: MOVED     CAPAMEDIA[1]
1740: OPEN
1741: SPLINED   VECTADEL 1 4
1742: CLOSE
1743: SPLINED   VECTADEL 4 1
1744: MOVED     CAPAMEDIA[1]
1745: OPEN
1746: MOVED     ARRIBA[1]
1747: PRINT     "ya"
1748: END

        PROGRAM   CARA6
        **********************
1693: OPEN
1694: MOVED     ARRIBA[1]
1695: SPLINED   VECTADEL 1 4
1696: CLOSE
1697: SPLINED   VECTADEL 4 1
1698: MOVED     ARRIBA[1]
1699: PRINT     "ya"
1700: GET       X
1701: SPLINED   VECTADEL 1 4
1702: OPEN
1703: SPLINED   VECTADEL 4 1
1704: MOVED     ARRIBA[1]
1705: PRINT     "ya"
1706: END

        PROGRAM   CARA6DEV
        **********************
1750: SPLINED   VECTADEL 1 4
```

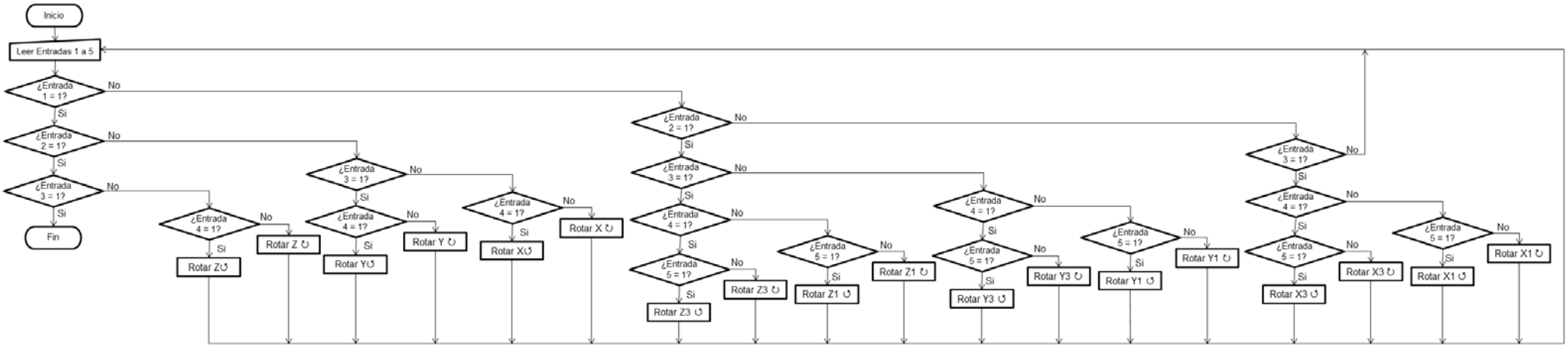
```
1751: OPEN                          1752: SPLINED   VECTADEL 4 1
1753: MOVED     ARRIBA[1]
1754: PRINT     "ya"
1755: END
```

ANEXOS

Anexo B: Diagrama de flujo para el diseño de los métodos del robot

El diagrama de flujo que se muestra en la Figura corresponde al programa que se pensó inicialmente para los movimientos del robot el cual está pasmado en la metodología donde se buscaba anidar todos los movimientos en un programa principal y las rotaciones de cada capa se consideraban métodos. Las entradas de la 1 a la 5 correspondían no a la comunicación serial sino a 5 de las 8 entradas digitales disponibles que tiene el robot las cuales se conectaban por medio de cables separados uno para cada entrada.

Figura 9: Diagrama de flujo para los métodos iniciales del robot

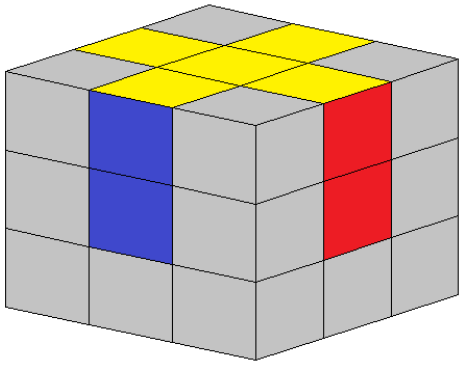


ANEXOS

Anexo C: Pasos para Solucionar el Cubo de Rubik por medio del algoritmo de Singmaster

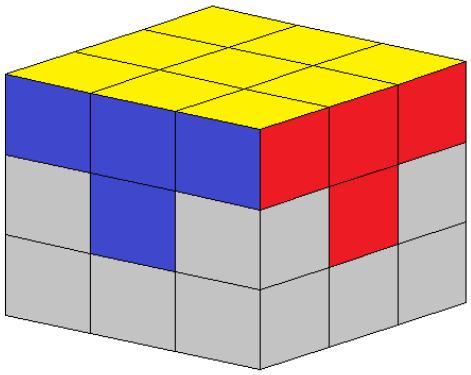
El primer paso para solucionar el cubo de Rubik es hacer que los colores de la cara superior queden en forma de cruz y los colores de los cuadros de las otras caras que coinciden con los bordes de la cruz deben ser iguales al color central de sus respectivas caras (ver Figura).

Figura 10: Primer paso para solucionar el Cubo de Rubik.



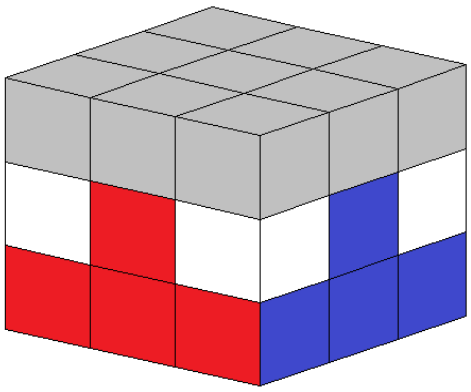
El Segundo paso es poner las esquinas que coinciden con los tres centros de las caras a las que pertenecen en su lugar (ver Figura).

Figura 11: Segundo paso para solucionar del Cubo de Rubik.



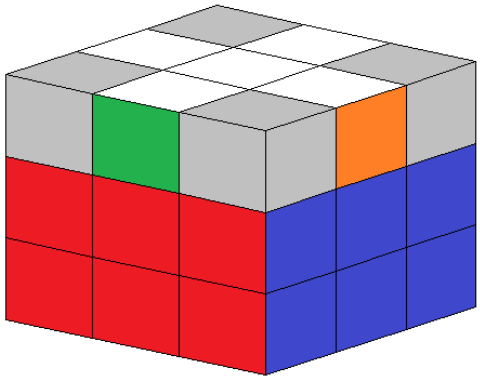
El tercer paso consiste en armar la segunda capa poniendo los cuatro bordes que faltan de esta capa, se debe primero rotar el cubo dejando la capa superior en la parte de abajo y ubicar las fichas demarcadas con color blanco en la Figura de modo que sus colores coincidan con los centros de las caras.

Figura 12: Paso 3 para solucionar el cubo de Rubik.



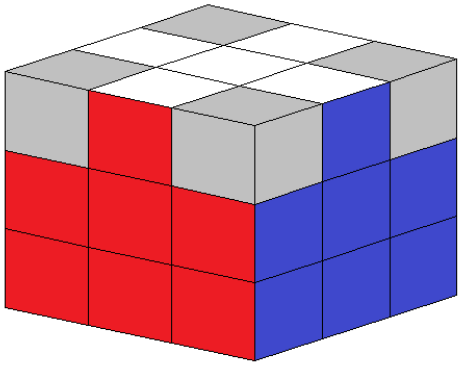
Formar la cruz de la capa de arriba sin importar si coinciden los colores de los lados con los centros de las caras como ocurre en el ejemplo de la Figura.

Figura 13: Paso 4 para solucionar el Cubo de Rubik



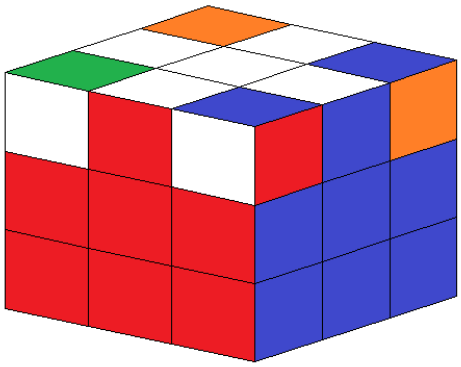
El quinto paso consiste en acomodar la cruz de modo que coincidan los centros del mismo modo en que coinciden los centros con la cruz del primer paso (ver Figura).

Figura 14: Paso 5 para solucionar el Cubo de Rubik.



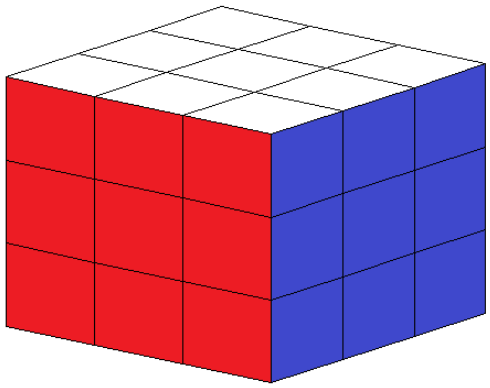
En el sexto paso como lo único que falta son las esquinas de la última capa, se busca que las esquinas coincidan con los centros de las tres caras a la que pertenecen, no importa si la esquina está girada como en el ejemplo de la Figura.

Figura 15: Sexto paso para solucionar el cubo de Rubik



En el último paso se hace que las esquinas que ya están ubicadas en el lugar indicado, se giren hasta que coincidan los colores con cada centro, de este modo quedará terminado el Cubo de Rubik (ver Figura).

Figura 16: Séptimo paso, cubo solucionado.



Nota: Si se desea encontrar información precisa sobre los movimientos exactos que se deben realizar para realizar cada paso diríjase a:

http://www.rubiks.com/solving-center/pdf/Rubiks_Spanish.pdf

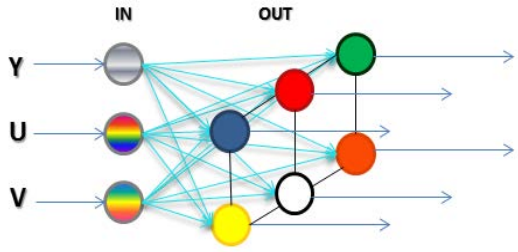
ANEXOS

Anexo D: Pruebas Realizadas para un Óptimo reconocimiento de los Colores en la Red Neuronal

Para la elaboración del sistema de visión artificial en su parte perceptiva, se necesitó generar una red neuronal la cual se logro a partir de distintos análisis.

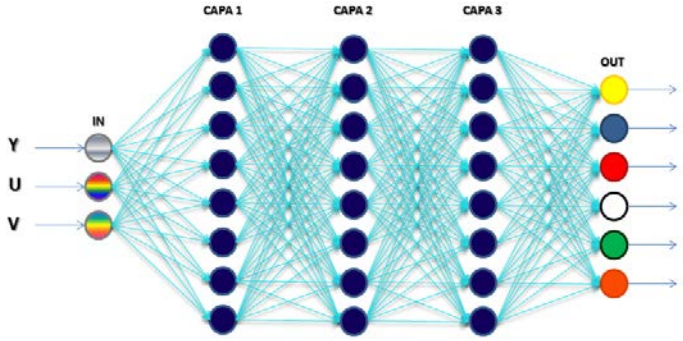
Al diseñarse la red neuronal, se necesitó mirar qué tipo de red se implementaría para esto se analizaron dos tipos de red neuronal, la primera son los mapas auto-organizados de Kohonen los cuales son los más apropiados a este tipo de cosas pero matlab presentó limitaciones en cuando al manejo de tres entradas a las red las cuales en el presente caso serían los tres canales provenientes de la cámara con el modelo de color YUV (ver Figura).

Figura 17: Mapa Auto-organizado de Kohonen aplicado al presente sistema.



Como este tipo de red neuronal se basa en encontrar la neurona más cercana a uno de los seis colores del cubo, la cual es considerada neurona ganadora, se diseño entonces una red neuronal multicapa que diera también como resultado una neurona ganadora, de este modo se seleccionaron seis neuronas de salida de las cuales se debería activar sólo una neurona al poner como entrada un color determinado (ver Figura).

Figura 18: Red neuronal multicapa aplicada al presente sistema.



Como esta red neuronal no da resultados binarios indicando si una neurona se activó o no sino que arroja unos resultados probabilísticos donde pueda estar el color dentro de sus rangos, lo que se buscó fue encontrar cuál de estas neuronas de salida dio la mayor probabilidad y así determinar el color detectado.

Una vez diseñada la red neuronal, se hizo adquisición de datos en varias condiciones de luz para tener mucha más información con la cual entrenar la red, para esto se tomó una foto a cada cara del cubo mientras éste estaba solucionado de modo que se adquirieron los 9 pixeles centrales de cada uno de los 9 cuadros que hay en cada cara, o sea que en cada una de las adquisiciones de datos realizadas se adquirieron 81 muestras de cada color, dado que a pesar de ser en una misma cara, los colores pueden variar.

La figura a continuación muestra claramente las diferencias que hay entre cada uno de los cuadros de color en cada adquisición, así como también las diferencias que hay al realizar cambios en la iluminación (ver Figura).

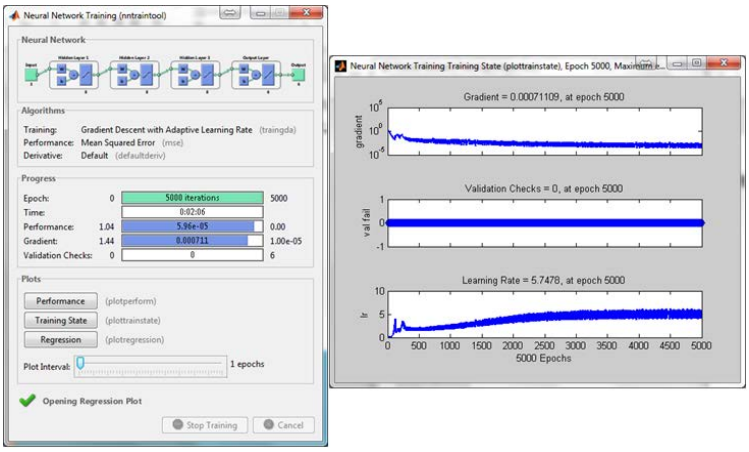
Figura 19: Resultados obtenidos de las adquisiciones realizadas.

Al tener toda la información necesaria para entrenar la red neuronal se probaron varias técnicas de entrenamiento las cuales fueron, entrenamiento con alfa variable, con momento variable, alfa y momento variable y por último por medio del clasificador bayesiano.

Los parámetros utilizados para evaluar estas técnicas fueron: la cantidad de iteraciones hasta el fin del entrenamiento, tiempo de entrenamiento, y finalmente el gradiente el cual es el que indica si se alcanzó un nivel máximo de aprendizaje en cuyo caso el valor del gradiente tiende a cero.

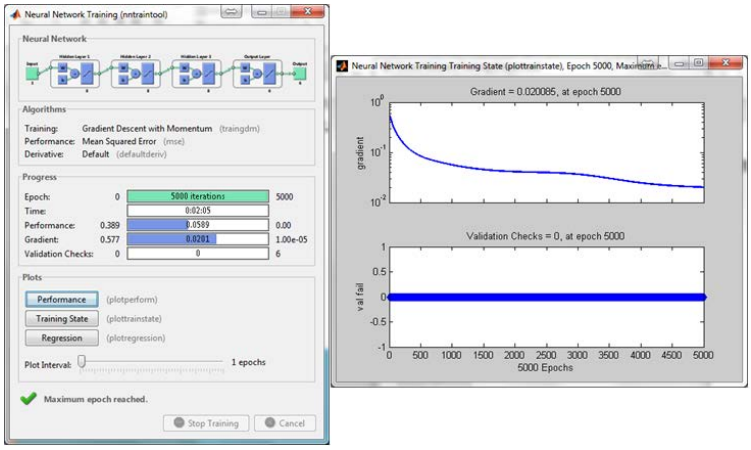
La figura siguiente muestra el desempeño de la técnica de entrenamiento con alfa variable, en este caso se hizo el entrenamiento con 5mil iteraciones el cual tardó 2:06 minutos y el gradiente mínimo fue alcanzado en la iteración número 500 el cual dio un valor de 7.1×10^{-4} (ver Figura).

Figura 20: Desempeño del entrenamiento con alfa variable.



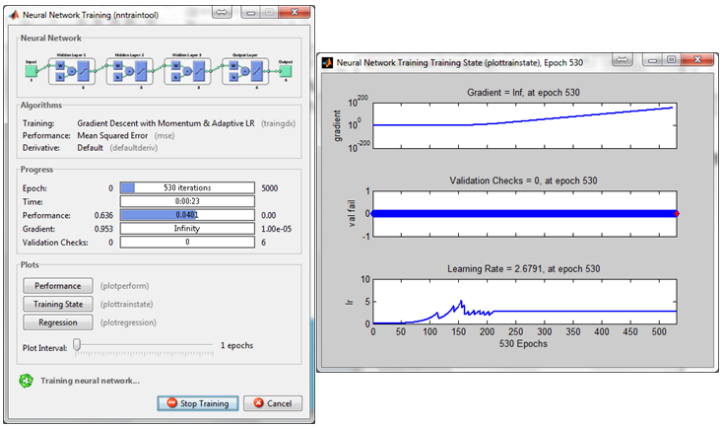
En el caso del entrenamiento con momento variable, se realizó el proceso en 5mil iteraciones y tardó 2:05 minutos, pero si bien estos dos parámetros fueron muy similares, se puede ver que el entrenamiento fue menos efectivo que el anterior, pues el gradiente mínimo que fue alcanzado en la iteración número 5000 dio un valor de $2x10^{-2}$ (ver Figura 21).

Figura 21: Desempeño del entrenamiento con momento variable.



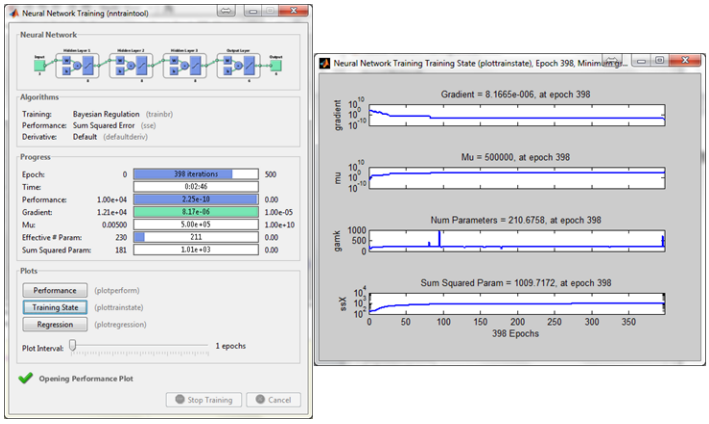
Luego se realizó un entrenamiento con alfa y momento variable y fue programado para realizar 5mil iteraciones de entrenamiento pero a los 23 segundos de haber iniciado el entrenamiento en la iteración número 530 el valor del gradiente fue tan alto que ya se consideró como infinito lo cual pausó el proceso de entrenamiento demostrando así que éste método no es el apropiado para realizar la presente tarea (ver Figura).

Figura 22: Desempeño del entrenamiento con alfa y momento variable.



Finalmente se utilizó el Clasificador Bayesiano el cual es una mejora al algoritmo de Levenberg-Marquardt y dio los mejores resultados, pues fue programado para realizarse en 5mil iteraciones pero en la iteración número 398 se alcanzó un nivel tan bajo que se consideró que ya había llegado a cero y el algoritmo terminó su entrenamiento. Finalmente el valor para el gradiente fue de 8.2x10-6 y el tiempo de entrenamiento fue de 2:46 minutos (ver Figura).

Figura 23: Desempeño del algoritmo de entrenamiento con Clasificador Bayesiano.



Si bien la relación iteración/tiempo del entrenamiento con esta técnica es mucho mayor que en los demás casos, se puede ver que el entrenamiento fue el que dio mejores resultados que, como en este caso se hace entrenamiento antes de solucionar el cubo y no durante el proceso, no es tan importante el tiempo que demore el sistema en entrenar.

Ya teniendo la red lo mejor entrenada posible, se procede a obtener los datos de los 9 cuadros de cada cara pero ya con el cubo desordenado y se ingresaban a la red neuronal para que ésta, con los datos del entrenamiento, pudiera determinar el color correcto.