UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

PROGRAMA DE MAESTRÍA Y DOCTORADO EN INGENIERÍA

INGENIERÍA ELÉCTRICA – SISTEMAS ELECTRÓNICOS

“INTEGRACIÓN DE ALGORITMO EN SISTEMA EMBEBIDO PARA EL RECONOCIMIENTO Y CLASIFICACIÓN DE FORMAS RÍGIDAS CON EL MODELO MANO-OJO EN UN ROBOT ANTROPOMÓRFICO”

TESIS

QUE PARA OPTAR POR EL GRADO DE:

MAESTRO EN INGENIERÍA

PRESENTA:

ING. LUIS MANUEL AMÉZQUITA VILLEGAS

TUTOR PRINCIPAL

DR. JUAN MARIO PEÑA CABRERA

MÉXICO, CDMX, OCTUBRE 2019

Jurado Asignado

Cargo

Presidente…….. Dr. Prado Molina Jorge

Secretario…… Dra. Moumtadi Fátima

1er. Vocal…….. Dr. Peña Cabrera Juan Mario

2do. Vocal…… Dr. De La Rosa Nieves Saúl

3er. Vocal…….. Dr. Pérez Alcázar Pablo Roberto

Lugar o lugares donde se realizó la tesis: Facultad de Ingeniería (FI), Instituto de Investigaciones en Matemáticas Aplicadas y Sistemas (IIMAS).

TUTOR DE TESIS:

NOMBRE

FIRMA

Agradecimientos

Quiero agradecer al Consejo Nacional de Ciencia y Tecnología (CONACYT), pues gracias al apoyo brindado a lo largo de este posgrado, ayudó a facilitar el desarrollo de este trabajo.

Contenido

Contenido de ilustraciones

Contenido de tablas

Resumen

El presente trabajo plantea el proceso de integración en un sistema embebido de un algoritmo de reconocimiento y clasificación de formas rígidas, haciendo uso de un robot antropomórfico de tipo industrial con un modelo Mano-Ojo que se podría incorporar a una celda de manufactura (ilustración 1).

El proceso de integración en el sistema embebido se puede separar en 3 partes:

- Captura de la imagen.
- Reconocimiento del objeto.
- Comunicación con el robot industrial.

El algoritmo de reconocimiento se basa en el uso de la función de frontera de objetos (BOF), con el cual podemos reconocer y clasificar figuras geométricas para sentar una base en el reconocimiento de formas más complejas que se asemejen a objetos que usualmente se encuentran en la industria.

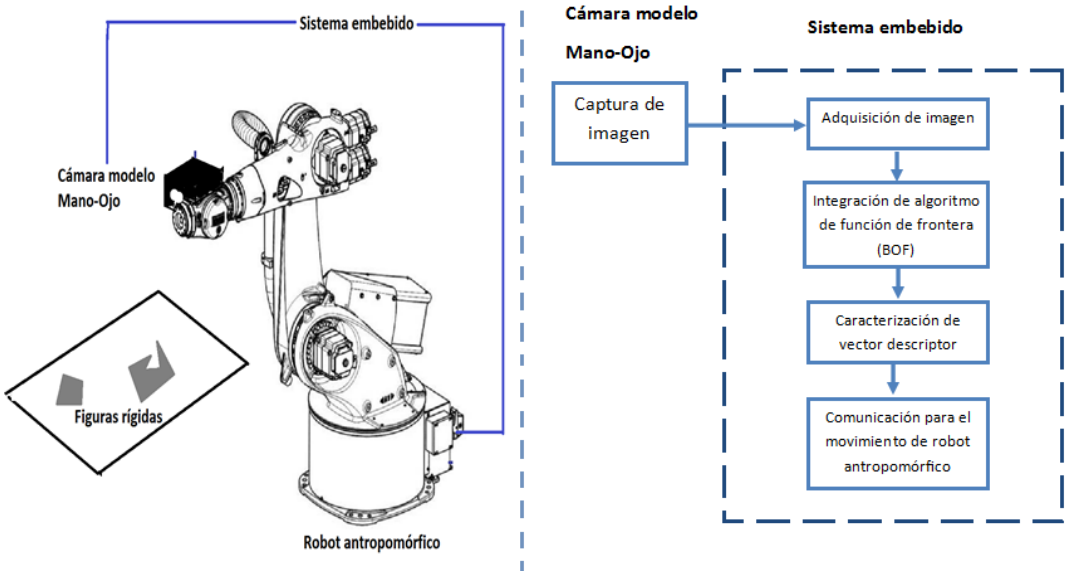


Ilustración 1 Representación del sistema de reconocimiento.

El robot antropomórfico es un robot KUKA industrial de 6 ejes y su módulo de control KRC4. Este robot mediante el uso de una cámara y el modelo mano-ojo realizará la captura de imágenes en diferentes ángulos de dichos objetos con la finalidad de obtener el mayor número de características para su reconocimiento y ubicación.

Capítulo 1

Instituciones Participantes

Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas (IIMAS)

Meta

Realizar la integración de un algoritmo de reconocimiento de formas rígidas en un sistema embebido con la manipulación de un robot de tipo industrial para su implementación en una celda de manufactura.

Objetivos

Objetivo general

Integración de algoritmo de reconocimiento en sistema embebido y la manipulación de un robot antropomórfico.

Objetivos específicos

- Reconocimiento y clasificación de formas rígidas.
- Manipulación de un robot antropomórfico con cámara y modelo Mano-Ojo.
- Programación en procesadores.
- Reconocimiento del entorno KUKA mediante el KRC.
- Programación de robot KUKA.

Justificación

En la actualidad, el creciente desarrollo en el área de la manufactura, nos ha llevado a lo que ahora conocemos como Industria 4.0, la cual nos exige continuar mejorando sistemas para que puedan actuar de una manera autónoma, por lo cual el presente trabajo contribuye con la investigación que actualmente lleva acabo el IIMAS en el desarrollo de celdas flexibles de manufactura. Donde se han realizado diversas investigaciones concluyendo en avances en el reconocimiento de formas rígidas definidas.

El presente proyecto busca contribuir en la caracterización y el reconocimiento de formas rígidas y la manipulación de un robot tipo industrial que se usará para la inspección de objetos, todo esto integrado en un sistema embebido. La diferencia de nuestro proyecto a los proyectos realizados en otros países, radica principalmente en la integración de estos algoritmos y la comunicación con el robot industrial haciendo uso de un solo sistema embebido, que permita dejar de lado el uso de software que requieren un mayor número de prestaciones computacionales [1].

Metodología

- Realizar una revisión bibliográfica sobre el estado del arte de las celdas de manufactura flexible.
- Curso introductorio al ambiente KUKA.
- Realizar una revisión bibliográfica sobre la visión artificial.
- Realizar una revisión bibliográfica del modelo Mano-Ojo.
- Estudiar el tipo de programación para realizar movimientos con el robot KUKA.
- Definir el entorno del área de experimentación.
- Conocer especificaciones del software y hardware.
- Definir sistema embebido a utilizar.
- Realizar la programación del algoritmo de la función de frontera (BOF).
- Asociar el vector descriptor para cada forma rígida.
- Implementar los algoritmos en el sistema embebido.
- Integrar al sistema embebido con los algoritmos la comunicación para la manipulación del robot KUKA.
- Realizar pruebas finales con la integración del sistema embebido.

Infraestructura

- Robot industrial KUKA.
- Controlador de robot CR4.
- Raspberry Pi 3.
- Red LAN.
- Cámara web Logitec y cámara basler.
- Equipo de cómputo.

Estado del Arte

Las revoluciones, por ejemplo, la revolución industrial, se han producido a lo largo de la historia cuando nuevas tecnologías y formas novedosas desencadenan un cambio profundo. En el caso de la industria 4.0, es un concepto nacido en 2011 en Alemania, el cual no se define por sus tecnologías emergentes, sino por el alcance de sus nuevos sistemas y su tendencia a la automatización, a lo que llamaremos “fabricas inteligentes”, que serán aquellas que tendrán la capacidad de aprender y adaptarse mediante uso de la inteligencia artificial (AI) y el uso de las redes cibernéticas (ilustración 2) [2] [3] [4] [5].

Ilustración 2 Ejemplificación de fábrica interconectada. Ver referencia [3]

El concepto de la industria 4.0 trae consigo el desarrollo de métodos para transformar líneas de producción de un esquema basado en el sistema, a uno basado en el producto [2], es decir, de pasar de una producción masiva donde la fábrica produzca grandes cantidades en una línea de producción hasta su etapa final, a pasar a tener pequeños subsistemas (celdas de manufactura) que puedan crear partes específicas de los productos y que puedan ser realizadas desde diferentes partes del mundo con un fin común. Los sistemas actuales de producción industrial pueden ser transformados a lo que se llama industria 4.0 aplicando metodológicamente una combinación de las siguientes innovaciones tecnológicas:

(1) Internet de las cosas, (2) la seguridad cibernética, (3) la computación en la nube, (4) análisis de grandes datos, (5) simulación, (6) la fabricación aditiva, (7) realidad aumentada, (8) robots autónomos e (9) inteligencia artificial. Según lo indicado por la última revisión en la industria 4.0, uno de los mayores desafíos será el de interconectar el mundo físico y el virtual. Una fábrica que aplique las anteriores tecnologías será llamada sistema de “manufactura flexible” o “celda de manufactura flexible” la cual consta de básicamente una estación de trabajo, manejo y transporte automatizado de piezas y un sistema de control (ilustración 3) [4] [6] [7] [8].

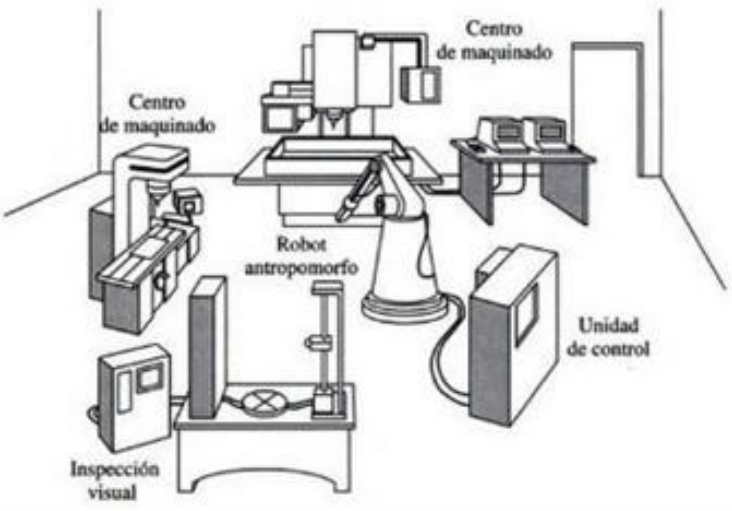


Ilustración 3 Celda de manufactura. (Kalpakjian & Schmid, 2002)

La transformación de esta industria se viene desarrollando y aplicando gradualmente en las fábricas que en sus procesos puedan ser sustituidos por sistemas inteligentes ya sea por la reducción de costos o por la seguridad de los trabajadores. En la producción de automóvil, como un ejemplo, el proceso de producción debe estar compuestos de muchas líneas de producción. Debido a que estas líneas del sistema de gestión de ejecución en la fabricación se componen de un número de líneas de producción con limitaciones mecánicas y en hardware, la flexibilidad se reduce enormemente, por lo tanto, es difícil diversificarlo.

La Industria 4.0 propone un modo de configuración dinámica, ya no en una línea de producción fija, si no, en la producción modular y la combina de forma dinámica y orgánicamente (ilustración 4). Cada módulo de producción puede ser considerado como un sistema, en donde, si se detiene la producción o suministro de piezas por un “cuello de botella”, el módulo de producción puede ser programado en otros modelos para continuar los procesos [9].

Ilustración 4 "Fábrica inteligente" automotriz de Tesla. Ver referencia [9]

En la industria láser, recientemente uno de los fabricantes de láser de alta potencia introdujo en su línea de producción la capacidad de recoger y compartir todos los datos de estado y parámetros de proceso de cientos de sensores de medición, incluidos los de salida del láser y el estado de los componentes adicionales. El objetivo final es analizar los parámetros de datos, optimizar los consumos de energía, llevar a cabo análisis de tendencias basadas en algoritmos, y tomar medidas específicas para determinar el riesgo de falla potencial para la prevención de paros no programados [3].

Gracias a la capacidad de análisis de datos, las fabricas pueden operar en pequeños lotes, hasta lotes de tamaño uno como la impresión láser 3D que actualmente eligió SpaceX para su cámara de motor SuperDraco (ilustración 5) [5].

Ilustración 5 Ensayo de motores SuperDraco de SpaceX. Ver referencia [5]

Para seguir siendo competitivos en un mundo que así lo requiere, las empresas deben aumentar la eficiencia operativa donde considere posible. En este sentido la visión artificial ha sido ampliamente reconocida como una tecnología innovadora para la automatización de los controles de calidad, medición y control en diferentes sectores industriales. En 1993 investigaron la viabilidad de utilizar la tecnología de visión artificial para localizar las plantas de maíz e inspeccionar la calidad de la semilla de maíz. Wen y Tao (1999) desarrollaron un sistema de inspección de visión artificial basado en normas para la clasificación de manzana. Jianchend Jia (2009) desarrolló un sistema de inspección de visión para el reconocimiento de calidad en jeringas de uso médico (ilustración 6) y, en los actuales años (2017), Liu Tao de la Universidad Case Western Reserve y el Instituto de Ingeniería de la Información y la Facultad de Ciencias de Ingeniería de Taiwán crearon sistemas de visión para la inspección de electrodomésticos y la detección e identificación de diferentes metales. Estos avances nos dan la certeza de hacer uso de la visión computacional para dotar a las máquinas de este sentido y llegar a una industria cada vez más autónoma [3] [11] [12].

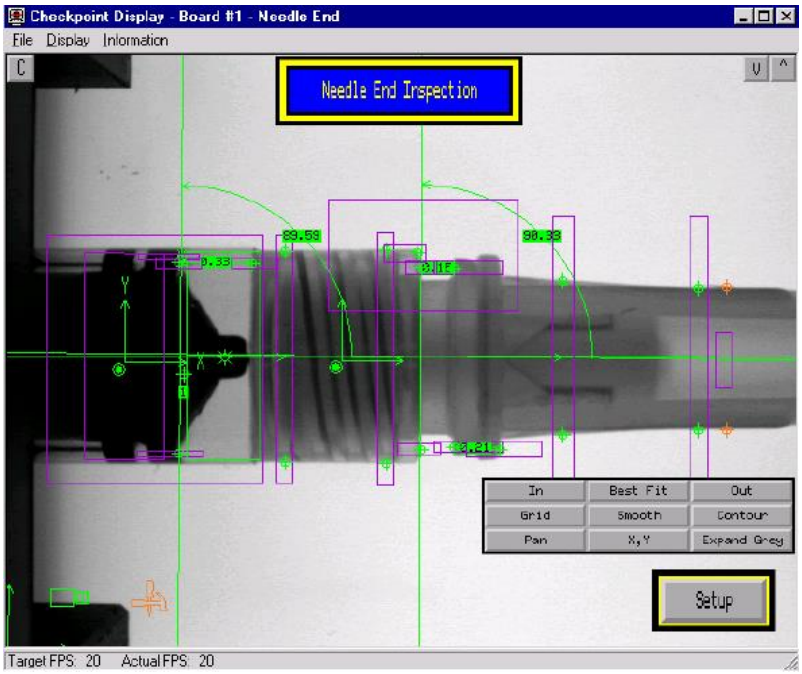


Ilustración 6 Inspección de calidad en una jeringa utilizando visión por computadora. Ver referencia [11].

En todos los procesos para formar una fábrica inteligente, la visión por computadora, entre varios aspectos, se utiliza para los procesos de inspección en productos, reconocimiento de aditamentos o para la auto calibración de los brazos robóticos.

En los procesos de inspección de productos se hace uso de celdas para facilitar el trabajo en el movimiento de piezas de grandes volúmenes y que resultan pesadas, así como el uso de la visión artificial para el reconocimiento de defectos que puedan

tener las mercancías. Una celda de inspección, como la desarrollada en Case Western Reserve University, donde implementan un sistema de servicio en la nube de AI (inteligencia artificial) en un ordenador donde se entrenan algoritmos y se actualizan modelos de inspección, hace uso de un brazo antropomórfico con un modelo mano-ojo que permite tener una mayor flexibilidad al momento de inspeccionar productos de gran o pequeño tamaño (ilustración 7) [8].

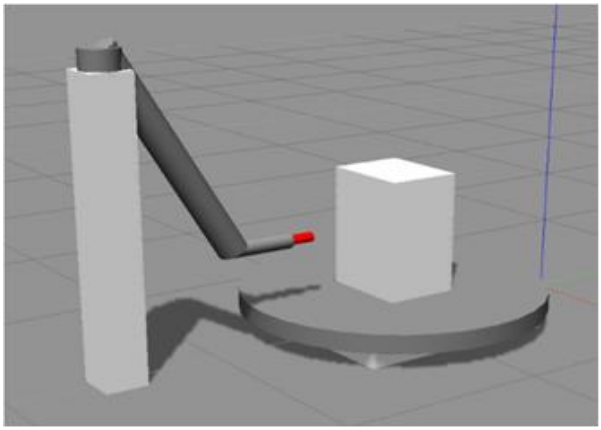


Ilustración 7 Simulación de una inspección de objeto. Ver referencia [8].

Retomando otro trabajo en la inspección de objetos haciendo uso de robots podemos mencionar el realizado en la Universidad Nacional de Taiwán, el cual mediante un robot paralelo realiza la inspección y seguimiento de objetos. La especialidad de este trabajo recae en el análisis de la velocidad con la que se mueven las piezas en el área de trabajo a través de un modelo mano-ojo y la precisión del robot en ubicarlas (ilustración 8) [13].

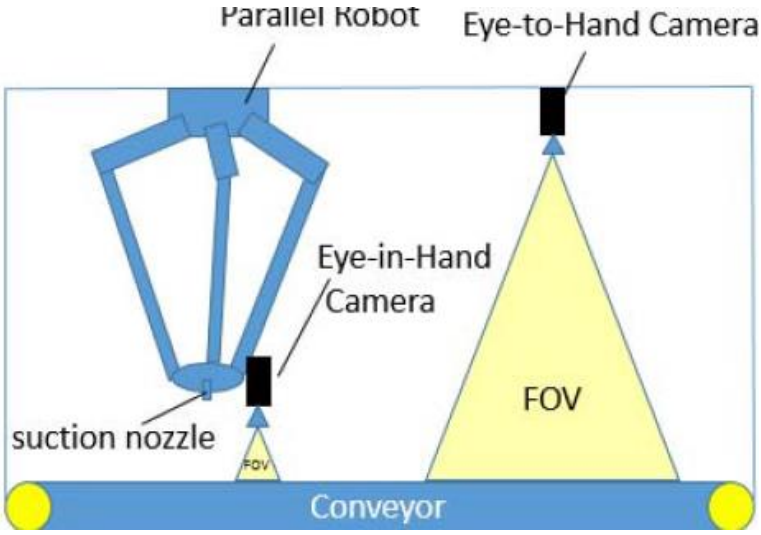


Ilustración 8 Inspección con modelo mano-ojo con un robot tipo paralelo. Ver referencia [13]

En los procesos de inspección se desarrollan técnicas para obtener un mejor análisis de piezas a reconocer, desde el uso de una cámara fija hasta el uso de

visión estéreo como el utilizado en el CINVESTAV, donde a partir de dos cámaras realizan el reconocimiento de la profundidad en un objeto para lograr obtener mediciones más precisas del objeto (ilustración 9) [14].

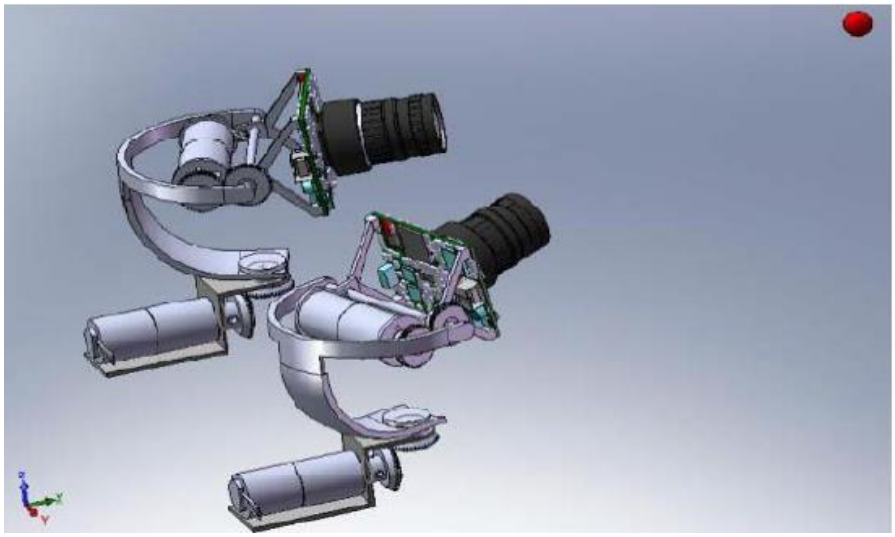


Ilustración 9 Visión estéreo para el reconocimiento de la profundidad de un objeto. Ver referencia [14].

En la Universidad Santo Tomás, Colombia, realizaron el proceso de reconocimiento mediante una cámara fija basado en cuatro parámetros: tamaño, forma, color y temperatura. El reconocimiento de la figura se realiza a través de una red neuronal mediante el aprendizaje de resultados obtenidos por las pruebas de MATLAB (ilustración 10) [12].

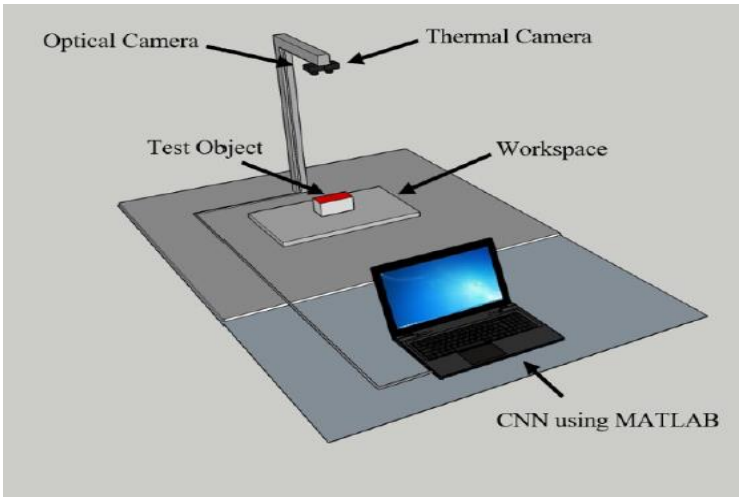


Ilustración 10 Reconocimiento de objeto haciendo uso de MATLAB. Ver referencia [12].

Siguiendo con el proceso de reconocimiento, el Instituto Nacional de Investigación de Inteligencia Integral en Corea, se propone a realizar el reconocimiento, haciendo uso de una fuente de iluminación y la rotación de objetos. El objetivo es la realización del reconocimiento del objeto utilizando una red neural, la cual entrenó con clases que se clasifican por tipo de objeto y el ángulo de rotación (ilustración 11) [15].

Ilustración 11 Reconocimiento de objetos en Corea. Ver referencia [15]

En la Universidad Nacional de Tecnología de Taipéi, se desarrolló una celda de manufactura para el reconocimiento de objetos, haciendo uso de un robot manipulador tipo industrial. Esta celda de manufactura se compone de un microprocesador que realiza la adquisición de la imagen, un robot antropomórfico que se encarga de la manipulación de las piezas y un ordenador que se encarga del procesamiento de la imagen (ilustración 12) [1].

Ilustración 12 Inspección de objetos con brazo robótico Taipéi. Ver referencia [1]

En la Universidad Nacional Chung Cheng Chiayi, se aplicó una celda de manufactura flexible en el reconocimiento de objetos, dando como resultado un

sistema de bajo costo. El proyecto se llevó a cabo mediante un robot antropomórfico, una cámara fija, un microcontrolador y un ordenador para el procesamiento de las imágenes (ilustración 13) [16].

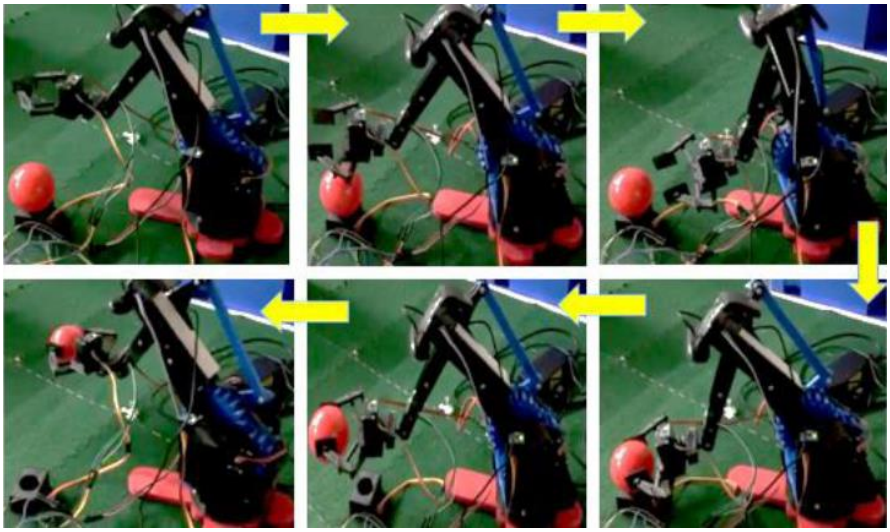


Ilustración 13 Inspección de objetos con brazo robótico Universidad Chung Cheng Chiayi. Ver referencia [16].

En la Universidad de Sousse, investigan los procesos necesarios para la extracción visual y el control remoto de un robot industrial KUKA KR-125. El sistema crea una comunicación mediante Ethernet gracias al protocolo TCP/IP. Para ello, se realizó una aplicación cliente/servidor donde se almacena el control de los movimientos más relevantes que podría realizar el robot. Con un Kinect, usado como cámara fija, detecta los objetos e identifican sus posiciones gracias a un algoritmo de procesamiento de imágenes ejecutado mediante un PC. El conjunto de todos estos elementos tiene como resultado determinar las coordenadas de ubicación del robot y el movimiento necesario para su manipulación (ilustración 14) [17].

Ilustración 14 Reconocimiento de objeto mediante Kinect. Ver referencia [17].

Actualmente, en el país se han investigado procesos de celdas de manufactura en diferentes aplicaciones, por ejemplo, en módulos para procesos de soldadura; como las desarrolladas por un equipo de académicos mexicanos del CINVESTAV, IPN y el IIMAS. Esta celda de manufactura, que consta de un control para el suministro de alambre a la antorcha, un sistema de adquisición de datos basado en un ordenador y un depósito de gas inerte, como se representa en la ilustración 15, puede modificar líneas de trayectoria de soldadura en piezas [18].

Además, esta celda de manufactura a través de una cámara y un sensor láser recopilan información en tiempo real y mediante el uso de algoritmos de lógica difusa puede corregir el error en la línea de soldadura de dos placas.

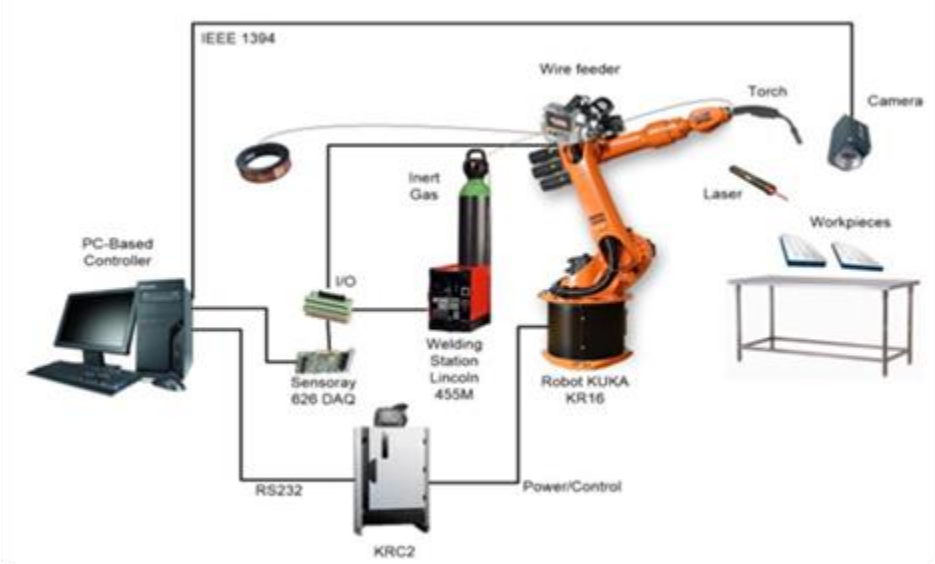


Ilustración 15 Celda de Manufactura para soldadura. Ver referencia [18].

En el IIMAS, se desarrolló una metodología para el reconocimiento y clasificación de formas rígidas, la cual pueden ser usada en una celda de manufactura, con la intención de crear un sistema de visión. La metodología consiste en la creación e implementación de un vector único en diferentes figuras, el cual describe características específicas del objeto, que resulta independiente a la escalabilidad y rotación, esto mediante una cámara de video fija en un área de trabajo (ilustración 16) [19].

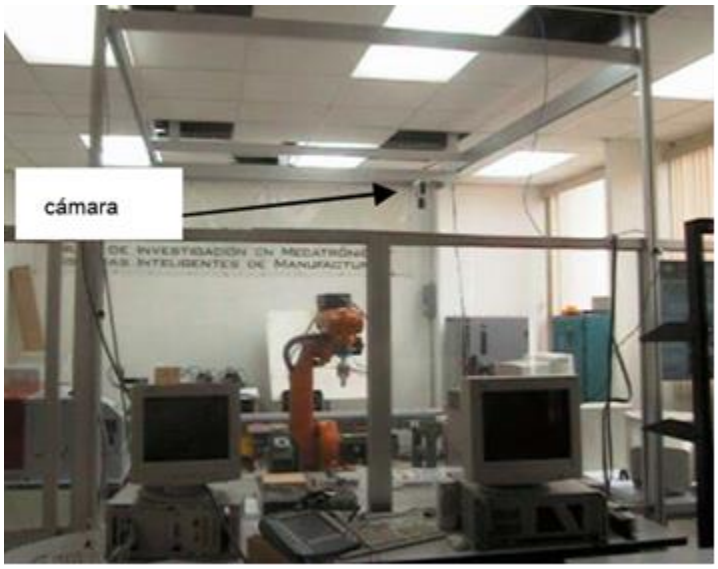


Ilustración 16 Inspección de objetos mediante cámara fija. Ver referencia [19].

Otro trabajo relacionado, llevado a cabo en el CINVESTAV, realizó la manipulación de objetos haciendo uso de un sistema Kinect, un robot tipo industrial y un sistema de reconocimiento de 12 cámaras. El trabajo consistió en la manipulación de un objeto mediante señales enviadas por un operador a través del sistema Kinect. Las señales captadas por el Kinect se relacionaban a los movimientos que el robot podía realizar y con el sistema de reconocimiento de 12 cámaras capturaba la posición del objeto (ilustración 17) [20].

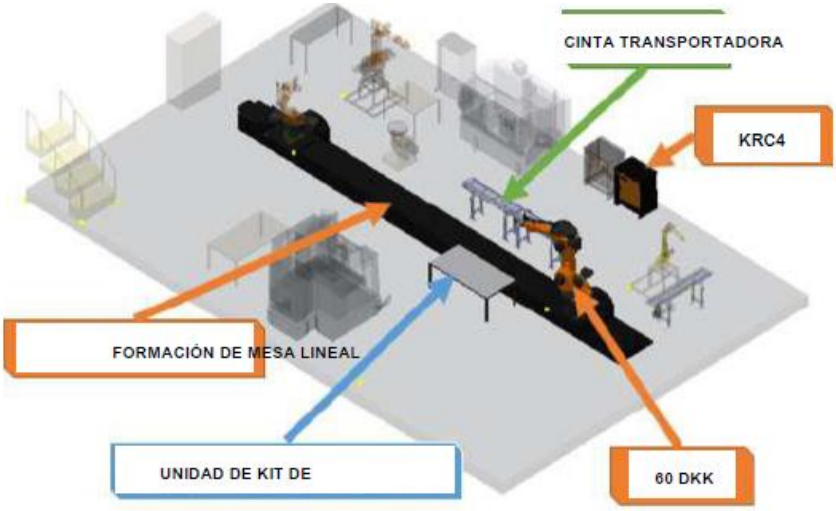


Ilustración 17 Celda de manufactura realizada en el CINVESTAV. Ver referencia [20].

Con base en estos proyectos podemos realizar una comparación con el actual proyecto que se realiza, resaltando las ventajas de nuestro sistema como se observa en la siguiente tabla (tabla 1).

Comparativa de celdas de manufactura.

Instituto	Robot	Conexión en red cibernética	Procesamiento de imagen	Modelo de captura de imagen
Case Western Reserve University	Antropomórfico tipo industrial.	Uso de un ordenador.	Uso de un ordenador para el procesamiento de imagen.	Modelo mano-ojo.
Universidad Nacional de Taiwán	Tipo paralelo.	N/P	Uso de ordenador.	Modelo mano-ojo.
CINVESTAV	2 Robots antropomórficos.	N/P	Uso de ordenador.	Visión estéreo.
Universidad Santo Tomas Colombia	Brazo Fijo.	Uso de ordenador.	Procesamiento en MATLAB.	Cámara fija.
Instituto Nacional de Investigación de Inteligencia Integral en Corea	Brazo fijo.	Uso de ordenador.	Procesamiento en ordenador mediante red neuronal.	Cámara fija.
Universidad Nacional de Tecnología de Taípe	Antropomórfico tipo industrial.	Uso de un ordenador.	Uso de un microcontrolador y un ordenador para el procesamiento de imagen.	Modelo mano-ojo.
Universidad Nacional Chung Cheng Chiay	Antropomórfico.	Uso de un sistema embebido.	Uso de un ordenador para el procesamiento de imagen.	Cámara fija.
Universidad de Sousse	Antropomórfico tipo industrial.	Uso de ordenador.	Uso de ordenador para el procesamiento de imágenes.	Cámara fija.
CINVESTAV, IPN , IIMAS	Antropomórfico tipo industrial.	N/P	Utiliza un ordenador para el procesamiento de la imagen.	Cámara fija.
IIMAS	Antropomórfico tipo industrial.	N/P	Uso de un ordenador para el procesamiento de imagen.	Cámara fija.
CINVESTAV	Antropomórfico tipo industrial.	Uso de ordenador.	Uso de ordenador para procesamiento de imágenes.	Cámara fija.
Proyecto actual	Antropomórfico tipo industrial.	Integración de un algoritmo en un sistema embebido para el procesamiento de imagen y conexión a red.		Modelo mano-ojo

Tabla 1 Comparativa de Proyectos.

Se puede identificar, con base en la tabla, que nuestro sistema de reconocimiento que se implementa en un sistema embebido y realiza la manipulación de un robot antropomórfico, se distingue por la integración de todos los procesos en un solo dispositivo. Esto con la finalidad de implementar el concepto que define a la industria 4.0, a lo que se llaman “celdas flexibles de manufactura”, el cual, cambiando un solo elemento, se pueda reconfigurar una o varias celdas de manufactura, agregando o quitando en dado caso otro tipo de sensores.

Capítulo 2

Antecedentes

En este capítulo trataremos acerca de los conocimientos previos que necesitamos conocer para el desarrollo del proyecto.

Sistema embebido

Se pueden encontrar diferentes definiciones de un sistema embebido:

“Un sistema integrado o embebido es un sistema basado en microprocesador que está diseñado para controlar una función o rango de funciones [8].”

“Un sistema embebido, es un sistema de computación diseñado para realizar un conjunto limitado de funciones específicas. Frecuentemente en un sistema informático en tiempo real. Los sistemas embebidos se utilizan para usos concretos, muy diferentes a los usos generales a los que se suelen dedicar los ordenadores [61].”

“Un sistema embebido cumple un propósito especial. Tiene requerimientos muy específicos y cumple un número de tareas predeterminadas. Es una combinación de hardware y software en el mismo paquete [62].”

Con base en estas definiciones, podemos definir a un sistema embebido como un dispositivo que suelen tener recursos limitados y funciones específicas para cumplir tareas concretas y predeterminadas.

Los procesadores que dominan a los sistemas embebidos son los procesadores ARM, pues su relativa simplicidad los hace ideales para aplicaciones de baja potencia. Estos procesadores, se han convertido en dominantes en el mercado de la electrónica móvil e integrada. Desde el 2005, alrededor del 98% de los más de mil millones de teléfonos móviles vendidos utilizaban al menos un procesador ARM. Desde 2009, los procesadores ARM son aproximadamente el 90% de todos los procesadores RISC de 32 bits integrados y se utilizan ampliamente en la electrónica de consumo, incluyendo PDA, tabletas, teléfono inteligente, teléfonos móviles, videoconsolas portátiles, calculadoras, reproductores digitales de música y medios (fotos, vídeos, etc.), y periféricos de ordenador como discos duros y routers [21].

Arquitectura ARM

Los procesadores tipo ARM (Advanced RISC Machine) fueron diseñados para permitir implementaciones de tamaño reducido y alto rendimiento. Estos procesadores fueron originalmente una arquitectura RISC de 32 bits y actualmente existen procesadores ARM de 64 bits, todos estos están licenciados por la compañía Britania ARM Holdings, que realizó su primer prototipo en el año 1985.

Las características esenciales de una arquitectura RISC pueden resumirse como sigue:

- Estos microprocesadores siguen tomando como base el esquema moderno de Von Neumann.
- Tamaño fijo de instrucciones de 32 bits.
- Arquitectura del tipo load-store (carga y almacena). Las únicas instrucciones que tienen acceso a la memoria son 'load' y 'store'.
- Casi todas las instrucciones pueden ejecutarse dentro de un ciclo de reloj.
- Pipeline (ejecución simultánea de varias instrucciones). Posibilidad de reducir el número de ciclos de máquina necesarios para la ejecución de la instrucción, ya que esta técnica permite que una instrucción puede empezar a ejecutarse antes de que haya terminado la anterior.
- Un banco grande de 32 registros. Todos ellos se pueden usar para cualquier propósito, permitiendo que la arquitectura de almacenamiento y carga funcione eficientemente

El hecho de que la estructura simple de un procesador RISC conduzca a una notable reducción de la superficie del circuito integrado, se aprovecha con frecuencia para ubicar en el mismo, funciones adicionales:

- Unidad para el procesamiento aritmético de punto flotante.
- Unidad de administración de memoria.
- Funciones de control de memoria cache.
- Implantación de un conjunto de registros múltiples.

La relativa sencillez de la arquitectura de los procesadores RISC conduce a ciclos de diseño más cortos cuando se desarrollan nuevas versiones, lo que posibilita siempre la aplicación de las más recientes tecnologías de semiconductores. Por ello, los procesadores RISC no solo tienden a ofrecer una capacidad de procesamiento del sistema de 2 a 4 veces mayor, sino que los saltos de capacidad que se producen de generación en generación son mucho mayores que en los CISC [22].

Características que posteriormente agregó ARM:

- Todas las instrucciones se ejecutan en un ciclo de reloj.
- Modos de direccionamiento simples.
- El procesamiento de datos solo opera con contenidos de registros, no directamente en memoria.
- Control sobre la unidad aritmética lógica (ALU, Arithmetic Logic Unit) y el “shifter”, en cada instrucción de procesamiento de datos para maximizar el uso de la ALU y del “shifter”.

- Modos de direccionamiento con incremento y decremento automático de punteros, para optimizar los lazos de los programas.
- Carga y almacenamiento de múltiples instrucciones, para maximizar el rendimiento de los datos.
- Ejecución condicional de todas las instrucciones, para maximizar el rendimiento de la ejecución.
- Set de instrucciones ortogonal, regular o simétrico. En este tipo de set no hay restricciones en los registros usados en las instrucciones, son todos registros de propósitos generales, con muy pocas excepciones (por ejemplo, el contador de programa, PC).
- Técnica “pipeline”. Esta técnica consiste en comenzar la próxima instrucción antes de que la actual haya finalizado. El objetivo es economizar tiempo.
- Excepciones vectorizadas. Las excepciones son condiciones inusuales o inválidas asociadas con la ejecución de una instrucción particular.
- Arquitectura “Thumb”. Algunos procesadores ARM tienen esta arquitectura para aplicaciones que necesiten mejorar la densidad de código. Consiste en usar un set de instrucciones de 16 bits que es una forma comprimida del set de instrucciones ARM de 32 bits. Estas mejoras sobre la arquitectura RISC básica permiten a los procesadores ARM adquirir un buen equilibrio entre alto rendimiento, escaso tamaño de código, bajo consumo y poca área de silicio. [23]

La familia ARM ha ido evolucionando con el tiempo, desde su arquitectura inicial (basada en el famoso 6502 de Rockwell) hasta los modernos serie v8 (ilustración 18) que equipan a dispositivos de tan amplia difusión como los Apple iPad.

Arquitectura	*Familia*
ARMv1	ARM1
ARMv2	ARM2, ARM3
ARMv3	ARM6, ARM7
ARMv4	StrongARM, ARM7TDMI
ARMv5	ARM7EJ, ARM9E,XScale
ARMv6	ARM11, ARM Cortex-M
ARMv7	ARM Cortex-A, ARM Cortex-R
ARMv8	ARM Cortex-A50

Ilustración 18 Evolución de los procesadores de la familia ARM.

Tarjeta de desarrollo Raspberry pi3

Raspberry Pi es un ordenador de placa reducida, ordenador de placa única u ordenador de placa simple (SBC) de bajo coste. Se trata de una diminuta placa de desarrollo que contiene un procesador Broadcom BCM2837 System-on-Chip (SoC). Esto quiere decir que la mayor parte de los componentes del sistema, incluidos la unidad central de procesamiento y la de gráficos junto con el audio y el hardware de comunicaciones, se encuentran integrados dentro de un único componente (ilustración 19) [24].

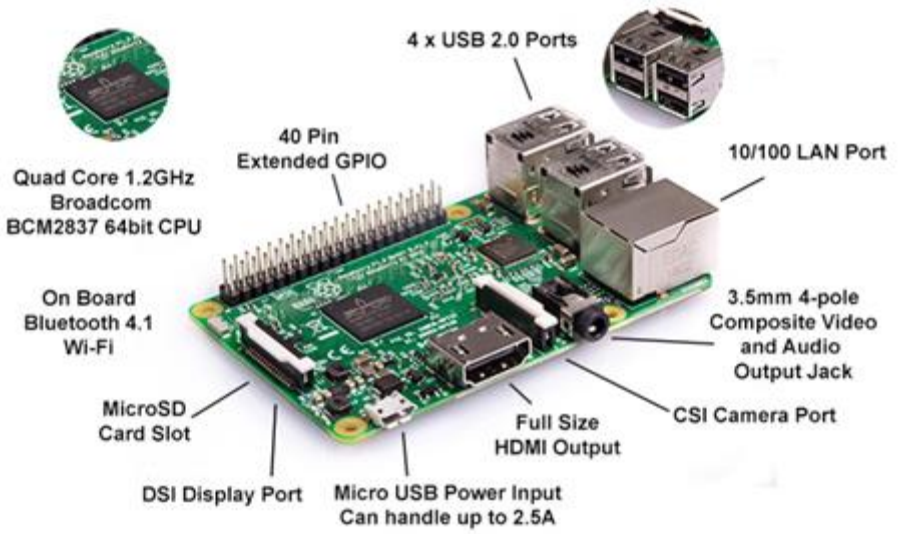


Ilustración 19 Componentes que integran la Raspberry pi 3. Ver referencia [24].

Como se menciona, Pi es un procesador multimedia Broadcom BCM2837 system-on-chip (SoC). No es sólo el diseño del SoC lo que hace al BCM2837 diferente del procesador de su PC o laptop. Lo que lo hace también diferente es que utiliza una arquitectura de conjunto de instrucciones (Instruction Set Architecture, ISA) distinta, conocida como ARM.

El BCM2837 utiliza una generación del diseño del procesador ARM conocida como ARM11, que a su vez está diseñada en torno a una versión de la arquitectura de conjunto de instrucciones conocida como ARMv6. Vale la pena recordar que ARMv6 es una arquitectura ligera y potente, pero tiene un rival en la arquitectura más avanzada ARMv7 utilizada por la familia de procesadores ARM Cortex.

El Raspberry Pi 3 Modelo B es un modelo de la tercera generación de Raspberry Pi con las siguientes características (ilustración 20):

- Quad Core 1.2GHz Broadcom BCM2837 64bit CPU
- 1 GB de RAM

- BCM43438 LAN inalámbrica y Bluetooth Low Energy (BLE)
- 100 Base Ethernet
- GPIO extendido de 40 pines
- 4 puertos USB 2
- Salida de 4 polos estéreo y puerto de video compuesto
- HDMI de tamaño completo
- Puerto de cámara CSI para conectar una cámara Raspberry Pi
- Puerto de visualización DSI para conectar una pantalla táctil Raspberry Pi
- Puerto micro SD para cargar su sistema operativo y almacenar datos
- Fuente de alimentación Micro USB conmutada actualizada de hasta 2.5[A]

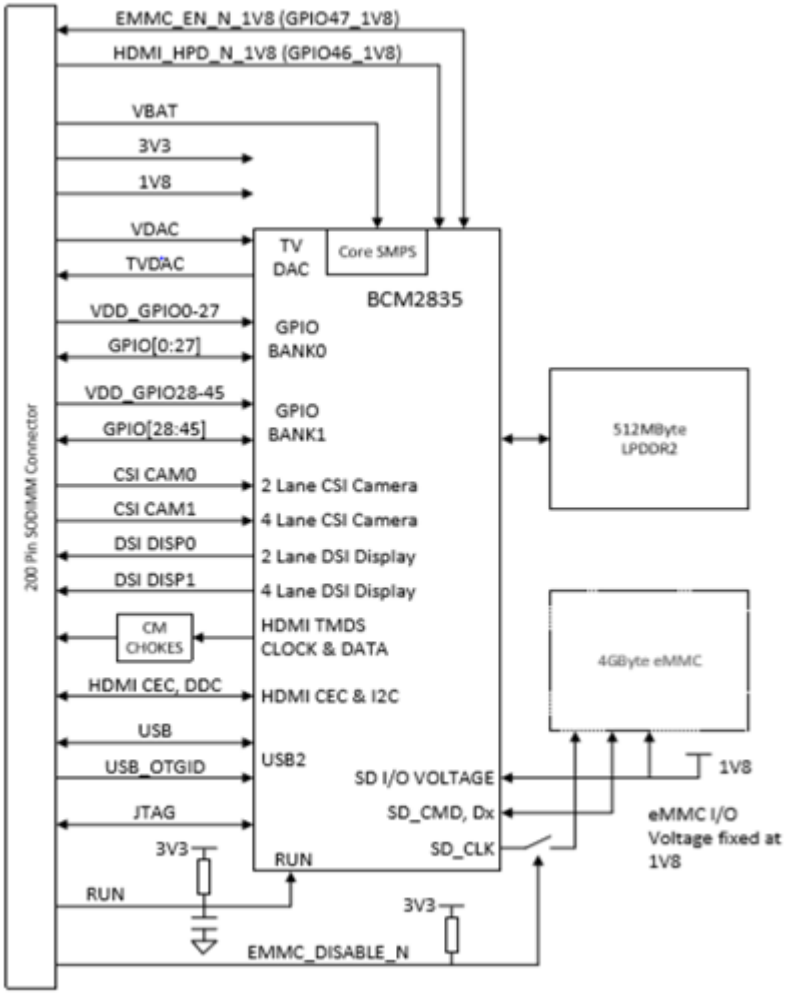


Ilustración 20 Diagrama electrónico de la Raspberry pi 3. Ver referencia [24].

El Raspberry Pi usa principalmente sistemas operativos GNU/Linux. Raspbian, una distribución derivada de Debian que está optimizada para el hardware de Raspberry Pi.

Visión Artificial

Entre los distintos modos sensoriales, la tecnología que se encarga de dotar del sentido de la vista a un robot se denomina “visión artificial” o “visión por computadora”, a esto lo podemos definir como las técnicas para el procesamiento, análisis e interpretación de imágenes para que la “máquina pueda comprender” y con esto realizar una acción y tomar decisiones.

Proporcionar capacidades sensoriales a las computadoras es una tarea sumamente complicada. A pesar de los inconvenientes, existe un interés especial por dotar a los ordenadores de uno de los cinco sentidos del hombre, el cual se refiere a la habilidad de ver. La visión artificial nos permite conocer ciertas características de un mundo de tres dimensiones en uno de dos dimensiones. Esta transformación de datos del mundo real al mundo digital las observamos mediante características como pueden ser brillo, forma, color, orientación, etc. [25].

La visión, tanto para un ser humano como para una computadora, consta principalmente de dos fases: captar una imagen e interpretarla. Por lo que la Visión Artificial lo que hace es interpretar las imágenes, distinguir los objetos de una escena, extraer información de ellos e interpretar esa información en un código digital [26].

Podemos dividir la visión artificial como una serie de procesos: la adquisición de la imagen, procesamiento de mejora de la imagen, obtención de características e interpretación de los resultados.

La primera etapa en el proceso de la visión artificial es adquirir una imagen digital, para ello se necesita de una cámara de video y un ordenador que tenga la capacidad de digitalizar la señal producida [27] [28].

La segunda etapa consiste en el uso de diferentes métodos para mejorar ciertos rasgos de la imagen, para lo cual hacer uso de métodos como la binarización, la realización de umbrales dinámicos, la ecualización de la imagen, etc.

La tercera etapa consiste en realizar un algoritmo que obtenga las características más importantes que necesita la aplicación, por ejemplo, obtención de pixeles negros, cantidad de pixeles en un área, distribución de pixeles en la imagen, nivel de grises en los pixeles, etc.

La última etapa consiste en interpretar esos resultados, es decir, clasificar cada una de las características antes recabadas para darle un sentido a la aplicación, saber cuántos objetos hay en la imagen, que tamaño tiene, etc.

Adquisición de la imagen

Una imagen es una representación de una realidad, la cual proporciona información sobre colores, brillo, formas, etcétera.

La digitalización se obtiene a través de una cámara fotográfica o de video, un escáner o directamente desde una computadora por medio de una cámara y utilizando algún programa para el tratamiento de imágenes. La información digital que generan los dispositivos electrónicos es almacenada en una memoria.

Una imagen es una matriz o arreglo bidimensional que la podemos representar como (ilustración 21):

$$r = f(x, y) \dots \dots \dots Ecuación\ 1$$

Donde:

r es la intensidad luminosa del píxel.

x, y son las coordenadas del pixel.

90	67	68	75	78	98	185	180	153	139	132	106	70	80	81	69	69	67	35	34
92	87	73	78	82	132	180	152	134	120	102	106	95	75	72	63	75	42	19	29
63	102	89	76	98	163	166	164	175	159	120	103	132	96	68	42	49	46	17	22
45	83	109	80	130	158	166	174	158	134	105	71	82	121	80	51	12	50	31	17
39	69	92	115	154	122	144	173	155	105	98	86	82	106	83	76	17	29	41	19
34	80	73	132	144	110	142	181	173	122	100	88	141	142	111	87	33	18	46	36
37	93	88	136	171	164	137	171	190	149	110	137	168	161	132	96	56	23	48	49
66	117	106	147	188	202	198	187	187	159	124	151	167	158	138	105	80	55	59	54
127	136	107	144	188	197	188	184	192	172	124	151	138	108	116	114	84	46	67	54
143	134	99	143	188	172	129	127	179	167	106	118	111	54	70	95	90	46	69	52
141	137	96	146	167	123	91	90	151	156	121	93	78	82	97	91	87	45	66	39
139	137	80	131	162	145	131	129	154	161	158	149	134	122	115	99	84	35	52	30
137	133	56	104	165	167	174	181	175	169	165	162	158	142	124	103	67	19	31	23
135	132	65	86	173	186	200	198	181	171	162	153	145	135	121	104	53	14	15	33
132	132	88	50	149	182	189	191	186	178	166	157	148	131	106	78	28	10	15	44

Ilustración 21 Matriz de valores de una imagen.

Al localizar las coordenadas de un pixel podemos conocer sus relaciones entre pixeles; pixeles vecinos horizontales y verticales y pixeles vecinos diagonales. Los primeros pixeles mencionados se les conoce como vecindad 4, cuyas coordenadas son (x+1, y), (x-1, y), (x, y-1) y (x, y+1) (ilustración 22), de la misma forma se les conoce a los pixeles diagonales, cuyas coordenadas son (x-1, y-1), (x+1, y-1), (x-1, y+1) y (x+1, y-1) (ilustración 23) [29].

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

Ilustración 22 Pixeles vecinos verticales y horizontales. Ver referencia [29].

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

Ilustración 23 Pixeles vecinos diagonales. Ver referencia [29].

Uno de los parámetros de mayor importancia en una imagen digital es su resolución. La resolución es la cantidad de píxeles que contiene una imagen. La resolución total expresa el número de píxeles que forman una imagen de mapa de bits. La calidad de una imagen depende directamente de su resolución.

Formato RGB

En el formato RGB, cada píxel se representa como un color creado a partir de ciertas intensidades en los colores rojo, verde y azul. Esta representación se puede interpretar como una matriz de tres niveles de intensidad, donde cada nivel corresponde a la intensidad de color de las componentes rojo, verde y azul (ilustración 24).

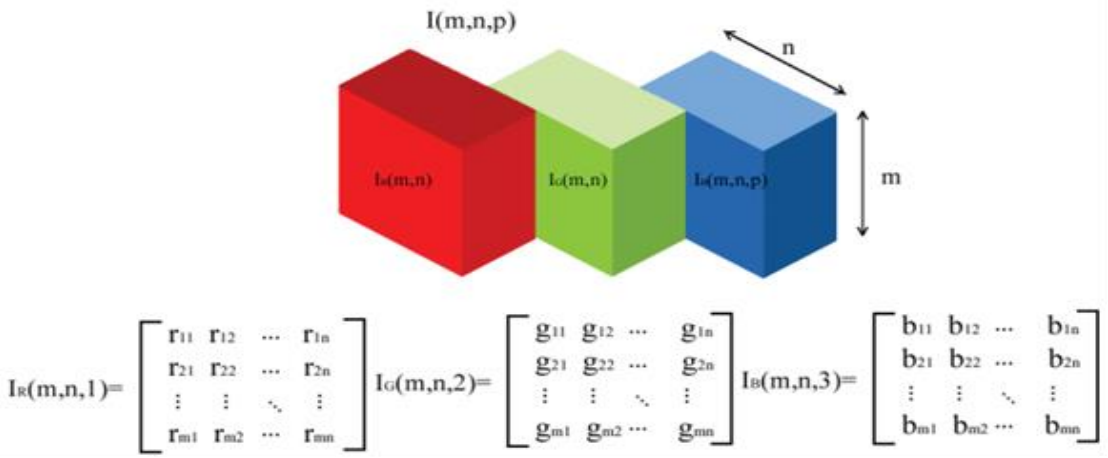


Ilustración 24 Formato RGB de una imagen. Ver referencia [24].

La forma en que el programa convierte a grises se realiza haciendo una mezcla de los tres canales (RGB) para obtener uno solo en gris tomando los siguientes porcentajes: Rojo 30%, Verde 59%, Azul 11%, que es según los expertos lo más parecido a como nuestros ojos captan la intensidad de luz dependiendo del color [30] [31].

$$grises(i,j) = \frac{R(i,j) * 0.3 + G(i,j) * 0.59 + B(i,j) * 0.11}{3} \dots\dots\dots Ecuación\ 2$$

Donde:

R (i, j) es el valor del pixel de la matriz de intensidad roja.

G (i, j) es el valor del pixel de la matriz de intensidad verde.

B (i, j) es el valor del pixel de la matriz de intensidad azul.

i, j son las coordenadas del pixel.

Procesamiento de mejora de la imagen

El propósito de las técnicas de realce de la imagen es mejorar la apariencia de la misma para el observador. Es un proceso subjetivo, realizado habitualmente de una forma interactiva. La selección de los métodos apropiados y la elección de los parámetros adecuados dependen de la calidad de la imagen original y de la aplicación.

Existe una gran cantidad de transformaciones u operaciones que se pueden realizar sobre las imágenes con el propósito de realzarlas y mejorarlas, donde existen varios criterios para clasificar estas operaciones.

Histograma

El histograma de la imagen consiste en una representación gráfica, donde se muestra el número de píxeles de cada nivel de gris que aparecen en la imagen, es decir, las frecuencias relativas de cada una de las intensidades de grises en una imagen (ilustración 25). Por lo general, las abscisas o eje “X” representa la tonalidad de grises en una imagen mientras que las ordenadas o eje “Y” representa la frecuencia que tiene tal tonalidad [32].

$$P_k = \frac{n_k}{n} \dots\dots\dots Ecuación\ 3$$

$$\begin{cases} 0 \leq P_k \leq 1 \\ k = 0, 1, \dots, L-1 \end{cases} \dots\dots\dots Ecuación\ 4$$

Dónde:

P_k = Probabilidad de grises.

n_k = Cantidad de pixeles en el nivel k

n = Cantidad total de pixeles.

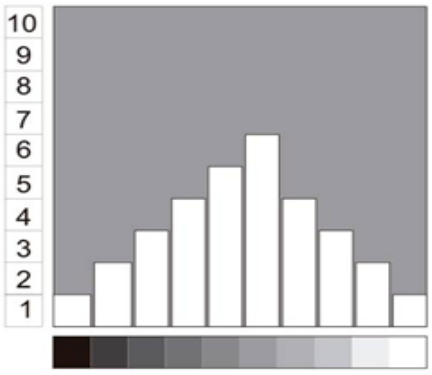


Ilustración 25 Histograma de la escala de grises.

Un histograma nos muestra la distribución de grises en nuestra imagen, donde podemos conocer si predomina el color negro o blanco, el grado de contraste que puede tener una imagen o la división de zonas oscuras o claras (ilustración 26).

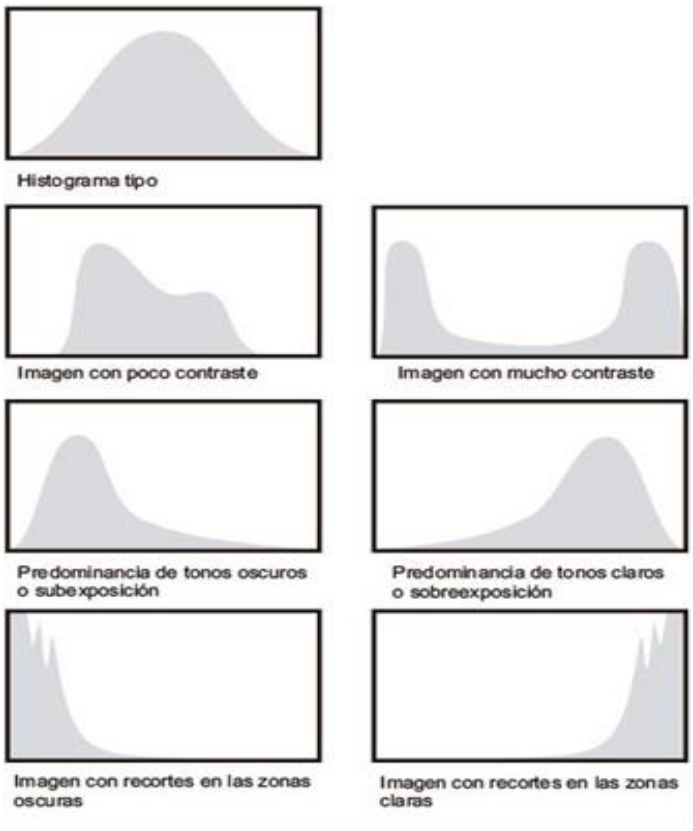


Ilustración 26 Distribución de grises en una imagen.

Binarización

La binarización es una técnica que consiste en la realización de un barrido en la matriz de la imagen digital, por medio de bucles o recursividad, con el fin de que el proceso produzca la reducción de la escala de grises a dos únicos valores. Negro cuando el valor del pixel es igual a 0 y blanco cuando el valor del pixel es igual a 255, o lo que es lo mismo, un sistema binario de ausencia y presencia de color 0 y 1. La comparación de cada píxel de la imagen viene determinada por el umbral de sensibilidad de un valor T (Threshold) (ilustración 27) [26].

$$Binaria(i,j) = \begin{cases} 1, & si \quad grises(i,j) < T \\ 0, & si \quad grises(i,j) \geq T \end{cases} \ldots\ldots\ldots Ecuación\ 5$$

Existen dos formas de realizar la binarización global o estática y local o dinámica: la primera cuyo valor del umbral T no cambia para todas las imágenes y la segunda el valor del umbral se adapta a cada píxel, dependiendo de las características locales de la imagen segmentada.

Ilustración 27 Distintos umbrales en la binarización de una imagen.

Ecualización

Es una transformación que pretende obtener un histograma con una distribución uniforme (ilustración 28). Es decir, la ecualización tiene la finalidad de cambiar una imagen de tal manera que se produzca una imagen donde todos los niveles de grises son equiprobables [33].

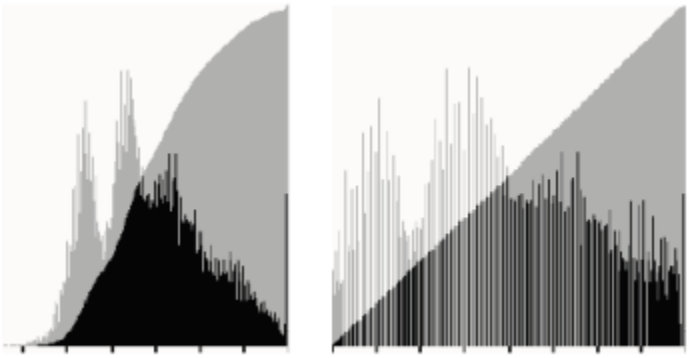


Ilustración 28 Comparación entre histograma e histograma ecualizado.

El proceso de la ecualización consiste en que todos los pixeles de un mismo nivel de gris se transformarán a otro nivel de gris, y el histograma se distribuirá en todo el rango disponible de todas las tonalidades de grises. La forma de lograr la ecualización es normalizando el histograma acumulado al rango de niveles de intensidad de la imagen [34]:

$$P_{ac} = P_{ac}(n-1) + P_k \dots\dots\dots Ecuación\ 6$$

$$I_{ecu}(i,j) = (L-1) * P_{ac}\left(I_{in}(i,j)\right) \dots\dots\dots Ecuación\ 7$$

Dónde:

P_{ac} = Probabilidad acumulada.

P_k = Probabilidad de cada nivel de grises.

I_{ecu} = Imagen ecualizada.

I_{in} = Imagen de entrada.

L = Es el nivel de grises.

Obtención de características uso del BOF

La detección de la frontera de un objeto es parte de un proceso de segmentación o aislamiento que consiste en la identificación de elementos dentro de una imagen.

La frontera es el contorno de un objeto que lo diferencia del fondo de la imagen y se obtiene al analizar los cambios en los niveles de grises que ocurren en una ubicación en específico.

La representación de un contorno se hace mediante polilíneas las cuales son líneas continúas compuestas de varios segmentos que describen una frontera. Cada segmento se especifica mediante un punto inicial y hasta un punto final. Por lo que la concatenación de dichos puntos describe un contorno:

$X_1, X_2, X_3, \ldots, X_n$

En donde Xn corresponde a las coordenadas (i, j), de cada punto del contorno. En la ilustración 29 se muestra una representación de un contorno mediante polilíneas [40].

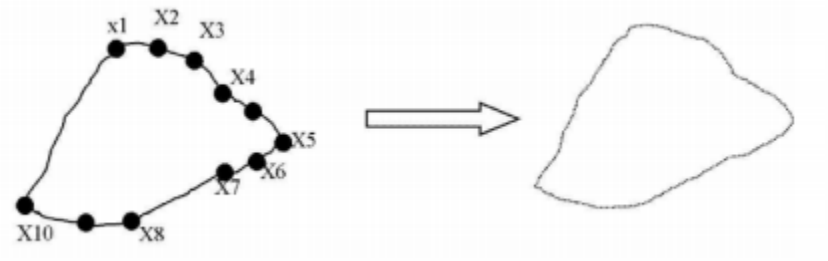


Ilustración 29 Puntos que conforman el BOF.

La función de objeto de frontera considera solo el contorno del objeto para reconocer diferentes objetos. Es muy importante obtener las propiedades métricas con la mayor precisión posible, como área, perímetro, centroide y distancia desde el centroide a los puntos del contorno del objeto.

Las desventajas de utilizar el BOF se hace notoria cuando se tienen objetos de dimensiones grandes, la función de frontera del objeto tiende a ser bastante larga, por lo que cualquier perturbación pequeña a lo largo del contorno puede causar cambios en el código, lo que representaría una mala secuencia.

Matriz de pesos

En una imagen binarizada obtenemos una matriz que nos representa los pixeles negros y blancos mediante 1 y 0, en este proceso se crea una matriz interna de 3x3 pixeles la cual recorrerán nuestra matriz binarizada haciendo la suma de cada valor que contienen en sus pixeles y sustituyéndolos en una nueva matriz que se denominará matriz de pesos (ilustración 30).

$$V_p = \sum_{i=0}^{3} \sum_{j=0}^{3} M_{aux}(i,j) \ldots\ldots\ldots Ecuación\ 8$$

Dónde:

V_p = Valor de la suma interna de la matriz.

M_{aux} = Matriz auxiliar de 3x3.

i, j = las coordenadas del pixel.

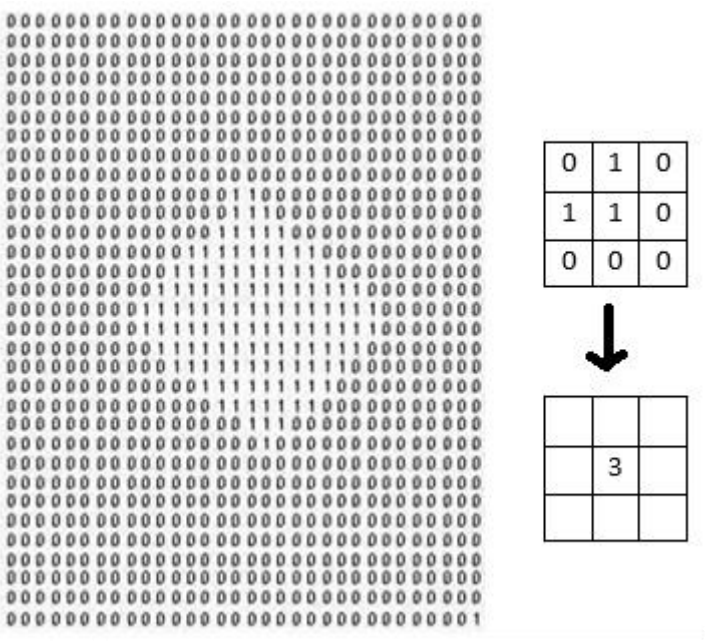


Ilustración 30 Matriz binarizada de una imagen.

Esta transformación obtiene una matriz con valores numéricos distribuidos en la imagen de acuerdo a su peso y partiendo de una imagen binaria (ilustración 31), cada número está relacionado con un par de coordenadas para establecer su localización.

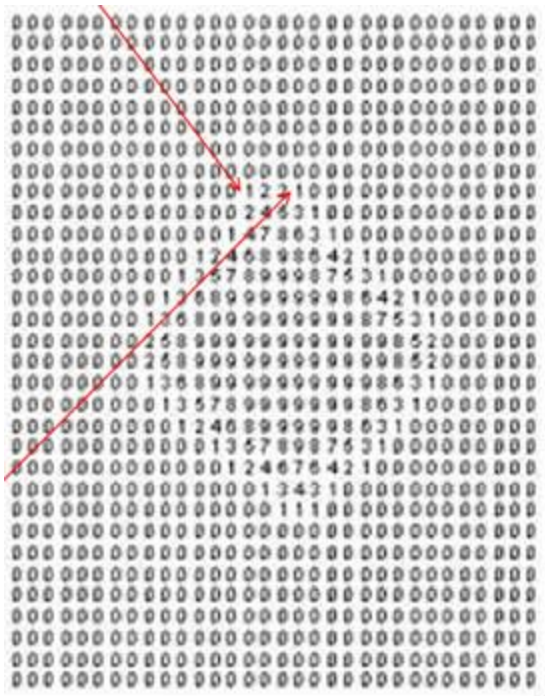


Ilustración 31 Matriz de pesos de una imagen.

En la matriz que se denomina; matriz de pesos, los valores más bajos serán los que se encuentren en los contornos de nuestra imagen y a medida que estos valores crezcan serán el centro de la imagen.

Matriz de pesos

En una imagen binarizada obtenemos una matriz que nos representa los pixeles negros y blancos mediante 1 y 0, en este proceso se crea una matriz interna de 3x3 pixeles la cual recorrerán nuestra matriz binarizada haciendo la suma de cada valor que contienen en sus pixeles y sustituyéndolos en una nueva matriz que se denominará matriz de pesos (ilustración 30).

$$V_p = \sum_{i=0}^{3}\sum_{j=0}^{3} M_{aux}(i,j) \dots\dots\dots Ecuación\ 8$$

Dónde:

V_p = Valor de la suma interna de la matriz.

M_{aux} = Matriz auxiliar de 3x3.

i, j = las coordenadas del pixel.

Calculo de área, perímetro y centroide.

La definición de perímetro es el conjunto de puntos que conforman la forma del objeto; en forma discreta es la suma de todos los píxeles que se encuentran en el contorno, que se puede expresar como:

$$P = (\sum_{i=0}^{n} + \sum_{j=0}^{m}) ImagenBinarizada(i,j) \dots\dots\dots Ecuación\ 10$$

La ecuación muestra cómo calcular el perímetro, el problema es saber cuáles son los píxeles de las imágenes que pertenecen al perímetro es ahí la importancia de calcular la matriz de pesos.

El área de los objetos se define como el espacio entre ciertos límites, es decir, la suma de todos los píxeles que forman el objeto, que puede definirse por:

$$A = \sum_{i=0}^{n}\sum_{j=0}^{m} ImagenBinarizada(i,j) \dots\dots\dots Ecuación\ 11$$

El centro de masa de una forma arbitraria es un par de coordenadas (Xc, Yc) en las que toda su masa se considera concentrada y también en la que actúa la resultante de todas las fuerzas. En otras palabras, es el punto donde un solo soporte puede equilibrar el objeto. Matemáticamente, para el dominio discreto de cualquier forma se definen como:

$$Xc = \frac{1}{A}\sum_{i=0}^{n}\sum_{j=0}^{m} i \dots\dots\dots Ecuación\ 12$$

$$Yc = \frac{1}{A}\sum_{i=0}^{n}\sum_{j=0}^{m} j \dots\dots\dots Ecuación\ 13$$

Obtención del vector descriptor

Esta fase proporciona información valiosa para el reconocimiento invariable de los objetos por parte del BOF, encontrando la distancia desde el centroide al perímetro o los píxeles del límite. Si suponemos que P_{cen} (X1, Y1), son las coordenadas del centroide Xc, Yc y P_{con} (X2, Y2) un punto en el perímetro, entonces, esa distancia está determinada por la distancia Euclidiana, la cual es deducida a partir del Teorema de Pitágoras:

$$P0 = D(P_{cen}, P_{con}) = \sqrt{(X2 - X1)^2 + (Y2 - Y1)^2} \dots\dots\dots Ecuación\ 14$$

Entonces nuestro vector descriptor estará determinado por el valor de cada distancia encontrada en nuestra imagen (ilustración 32):

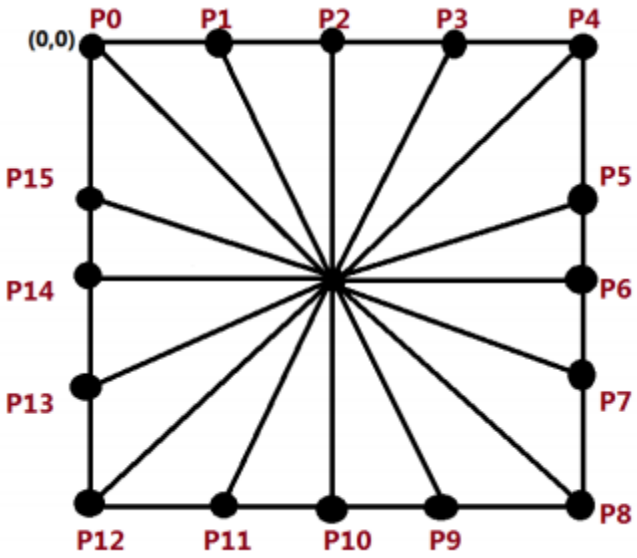


Ilustración 32 Obtención de distancias para el BOF de un cuadrado.

$$BOF = [P0, P1, P2, P3, P4, \dots P15]$$

Este vector representará una función única, independiente de la rotación o el tamaño de cada una de nuestras imágenes analizadas (ilustración 33).

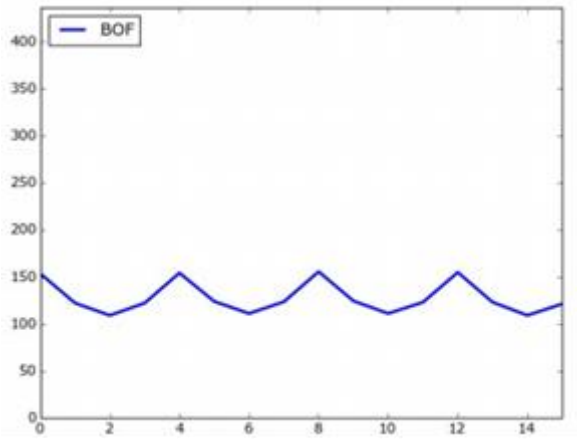


Ilustración 33 Representación del BOF de un cuadrado.

Robot Antropomórfico

Los robots industriales actualmente se encuentran en todas las empresas con largas y complejas líneas de producción, sustituyendo la mano de obra humana, reduciendo costos y maximizando ganancias; además, la programación automática de sus funciones beneficia la precisión en las tareas y disminuye la posibilidad de errores en el proceso.

Podría decirse que un robot industrial es una máquina multifuncional capaz de mover materias u objetos, manipular herramientas y piezas, los cuales son programables para hacer diversas tareas automáticamente, es decir, sin la necesidad de la intervención constante de una persona.

Existen varios y diversos conceptos que tratan de definir a los robots industriales:

El instituto de Robot Industry Association (RIA)

"Se entiende por robot industrial a un manipulador multifuncional reprogramable, diseñado para desplazar materiales, piezas, herramientas o dispositivos especiales mediante movimientos programados, variables, que permiten llevar a cabo tareas diversas".

La Swedish Industrial Robot Association (SWIRA)

"una máquina manipuladora automáticamente controlada, reprogramable, multipropósito con o sin locomoción para uso en aplicaciones industriales de automatización"

Asociación Francesa de Normalización (AFNOR)

"Manipulador automático, con servo sistema de posición, reprogramable, polivalente, capaz de posicionar y desplazar materiales, piezas útiles o dispositivos especiales a lo largo de movimientos variables y programables para la ejecución de tareas variadas. Estas máquinas polivalentes son generalmente concebidas para efectuar la misma función de manera cíclica y pueden ser adaptadas a otras funciones sin modificación permanente del material".

Japón

"Un robot es todo aquel dispositivo mecánico con algún grado de libertad, destinado a su utilización como manipulador"

Un manipulador robótico consta de una secuencia de elementos estructurales rígidos, denominados enlaces o eslabones, conectados entre sí mediante juntas o articulaciones, que permiten el movimiento relativo de cada dos eslabones consecutivos (ilustración 34).

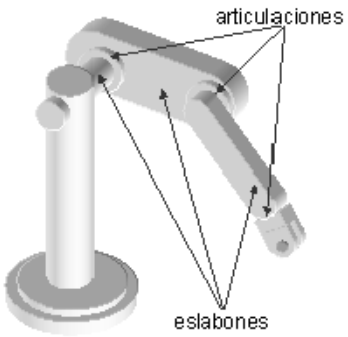


Ilustración 34 Elementos estructurales de un robot manipulador. Imagen tomada de http://platea.pntic.mec.es/vgonzale/cyr_0708/archivos/_15/Tema_5.4.htm

A los robots manipuladores también se le conocen como brazos robóticos o antropomórficos por su similitud en los movimientos de una extremidad superior humana (ilustración 35).

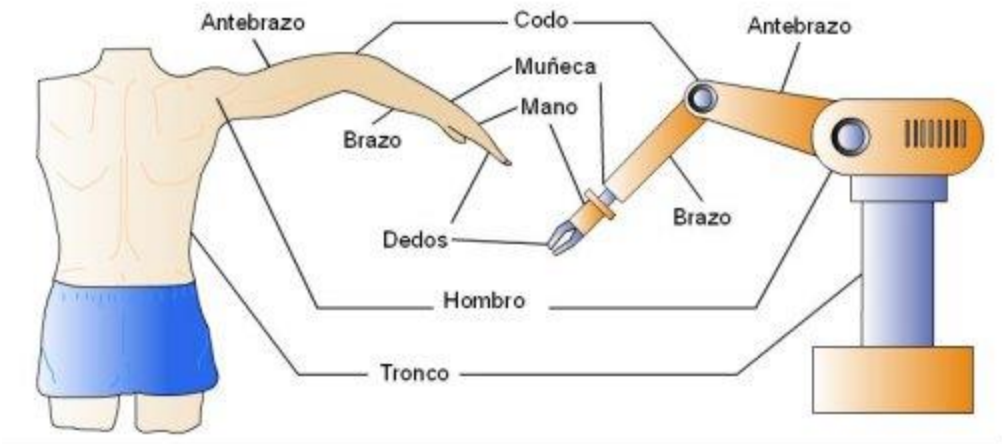


Ilustración 35 Similitudes entre un brazo humano y un robot manipulador. Imagen tomada de http://lrobotc.blogspot.com/2015/06/morfologia-de-un-robot.html

KUKA

EL robot KUKA se puede dividir en 4 componentes: (1) La parte mecánica o manipulador, (2) El KUKA Control Robot v4 (KCR), (3) El panel de control y (4) cables de conexión (ilustración 36).

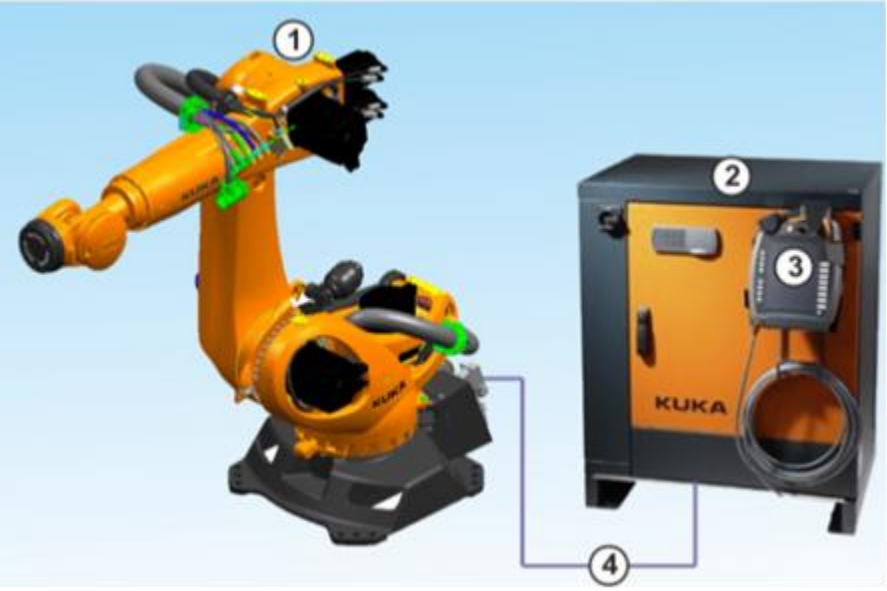


Ilustración 36 Elementos que componen a un robot KUKA Industrial. Ver referencia [39].

EL Robot KUKA es un robot KUKA KR5 arc HW de 6 grados de libertad, del tipo antropomórfico debido a que su movimiento se asemeja a los de un brazo humano (ilustración 37).

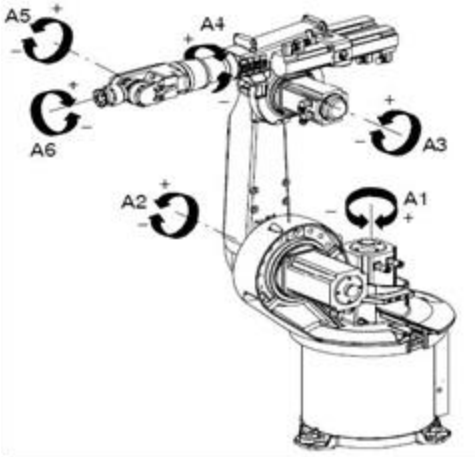


Ilustración 37 Ejes de movimiento de un robot KUKA KR5. Ver referencia [42].

Cada eje tiene un rango de movimiento limitado por software como medida de seguridad y de carga ligera el cual es adecuado para actividades tales como montar piezas pequeñas, lijar, pulir y pegar (ver tabla2 y tabla 3).

Datos Técnicos	
Alcance máximo	1423 mm
Carga nominal	5 kg
Repetitividad	0.04 mm
Número de ejes	6
Peso	126 kg aprox.

Tabla 3 Datos técnicos de un robot KUKA KR5 arc HW.

Datos de los Ejes		
Eje	**Desplazamiento**	**Velocidad**
A1	± 155°	156 °/s
A2	-180° / 65 °	156 °/s
A3	-110° / 170°	227 °/s
A4	± 165°	390 °/s
A5	± 140°	390 °/s
A6	Rotación sin fin	858 °/s

Tabla 2 Características de desplazamiento de los ejes de un robot KUKA KR5 arc HW.

El KUKA control robot (KRC) facilita la programación por medio de la interfaz de usuario Microsoft Windows. Es ampliable, se puede integrar en redes por medio de un bus y contiene paquetes de software pre-elaborados. El KRC tiene los siguientes componentes (ilustración 38) [39]:

1. Filtro de Red
2. Interruptor Principal
3. Controller System Panel (CSP)
4. PC de control
5. Alimentación Ejes 7, 8 y 9
6. Alimentación Ejes 1, 2 y 3
7. Alimentación Ejes 4, 5 y 6
8. Filtro de Freno
9. Cabinet Control Unit (CCU)
10. Safety Interface Board (SIB)
11. Fusibles
12. Acumuladores
13. Panel de Conexiones
14. Rodillos
15. SmartPAD

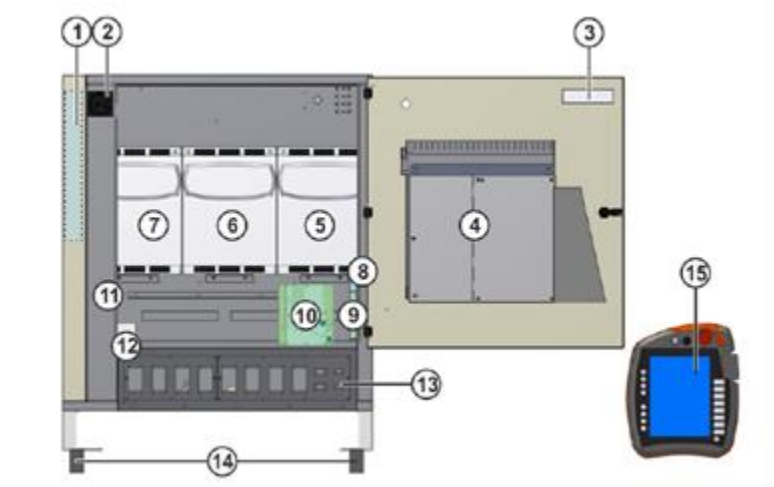


Ilustración 38 Elementos que componen el KRC. Ver referencia [39].

El software o teach pendal es la unidad manual de programación del sistema del robot. El KCP contiene todas las funciones necesarias para el manejo y la programación del sistema del robot (ilustración 39).

Este panel se compone de los siguientes elementos:

1. Desconectar SmartPAD
2. Selección de modo
3. Paro de Emergencia
4. Mouse
5. Teclas de Jog
6. Program Override
7. Jog Override
8. Menu Principal
9. Teclas de estado
10. Tecla de inicio
11. Tecla de inicio en reversa
12. Tecla de Stop
13. Teclado

Ilustración 39 Estructura de panel de control. Ver referencia [39].

Mientras que la parte trasera está compuesta por (ilustración 40):

1. Pulsador de hombre muerto
2. Tecla de inicio
3. Pulsador de hombre muerto
4. Conexión USB
5. Pulsador de hombre muerto

6. Placa de identificación

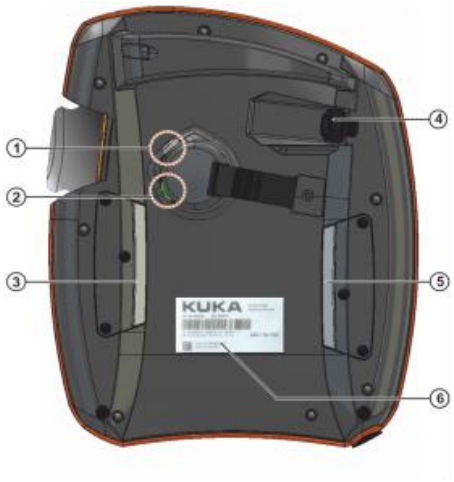


Ilustración 40 Parte trasera del panel de control. Ver referencia [39].

El software que contiene el teach pendal es el KUKA System Software 8.3 – HMI, la cual es nuestra interfaz gráfica donde podemos visualizar todas las opciones para controlar el robot, hacer movimientos manuales e ingresar a diferentes configuraciones o carpetas de programas (ilustración 41).

1. Barra de estado
2. Contador de mensajes
3. Ventana de mensajes
4. Indicador de estado del mouse
5. Indicador de alineación del mouse
6. Indicador de estado de las teclas de Jog
7. Etiquetas de las teclas de Jog
8. Program Override
9. Jog Override
10. Barra de botones
11. Icono de WorkVisual
12. Reloj
13. Bateria

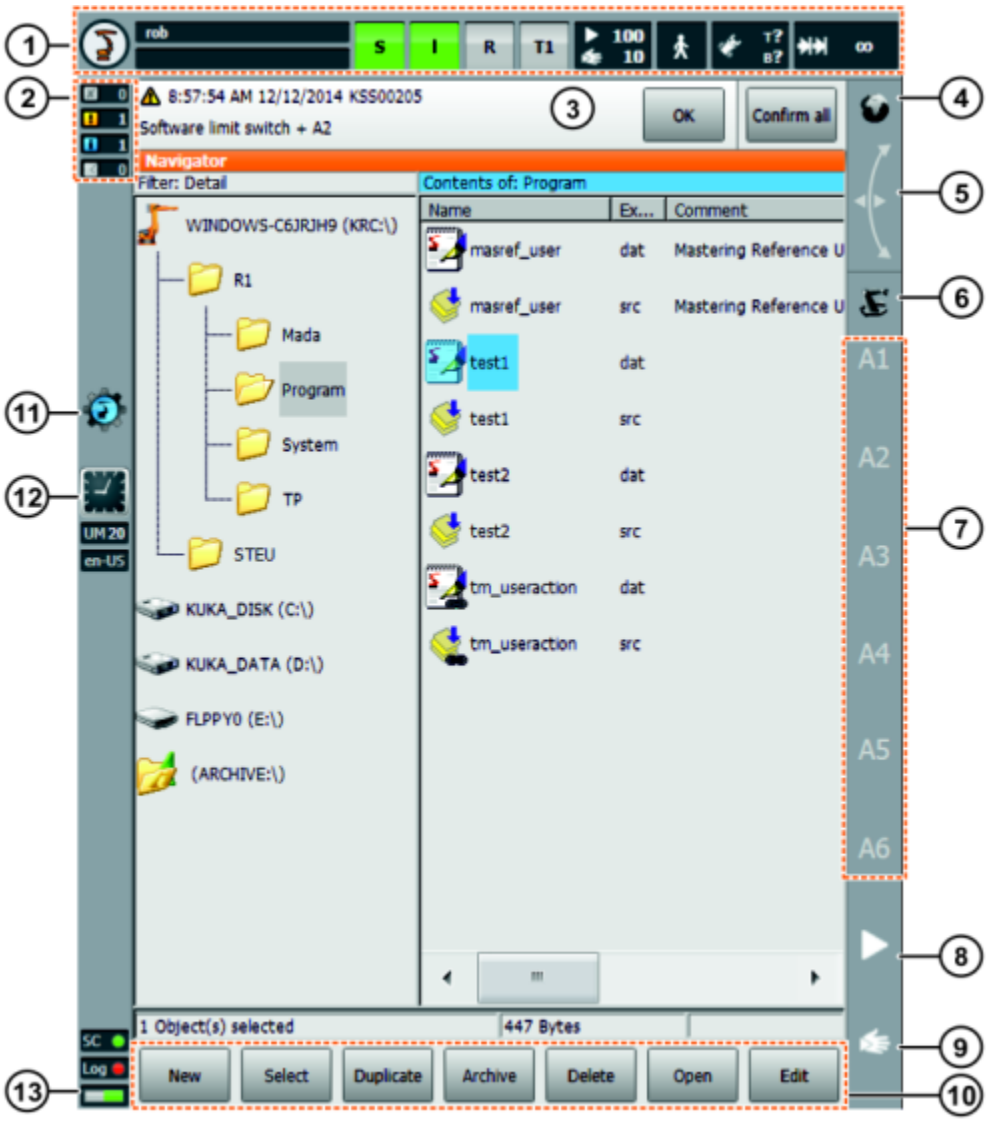


Ilustración 41 Software del robot KUKA.

Sistema de coordenadas

El robot KUKA nos proporciona 3 sistemas de coordenadas para trabajar: Coordenadas MUNDO, que se encuentra por defecto en la base del robot, coordenadas BASE, que es un sistema de coordenadas cartesiano con el cual podemos hacer una transformación hacia la pieza de trabajo y coordenadas HERRAMIENTA, la cual se encuentra en el centro de la herramienta en el eje 6 del robot, por defecto se encuentra en la brida (ilustración 42).

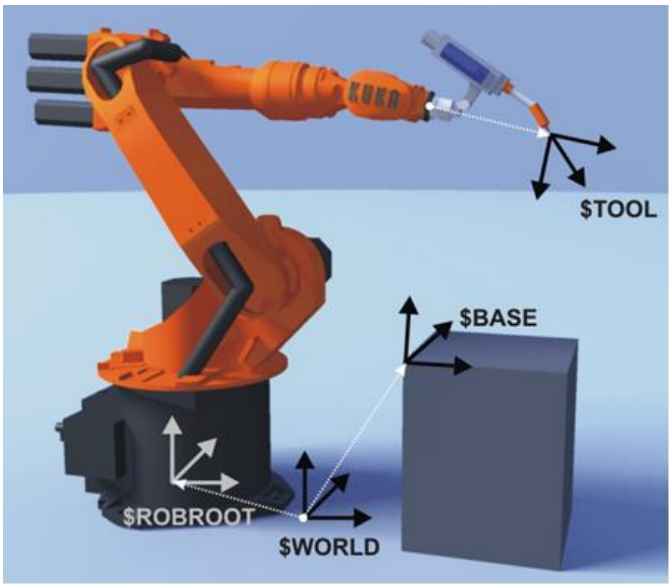


Ilustración 42 Bases de trabajo del robot KUKA. Ver referencia [39].

Calibración

El proceso de calibración se lleva a cabo en dos sistemas de coordenadas; en HERRAMIENTA, para conocer las dimensiones del dispositivo a utilizar, ya que será una extensión del robot manipulador, y en coordenadas BASE, ya que aquí sabremos las dimensiones de nuestra área de trabajo. En la calibración de la herramienta se pueden almacenar hasta 16 calibraciones. Este proceso de calibración almacena 6 valores: X, Y, Z, A, B y C, (ilustración 43) donde:

- X, Y y Z almacenan valores de los ejes cartesianos.
- A, B y C almacenan los valores de rotación con respecto a los ejes.

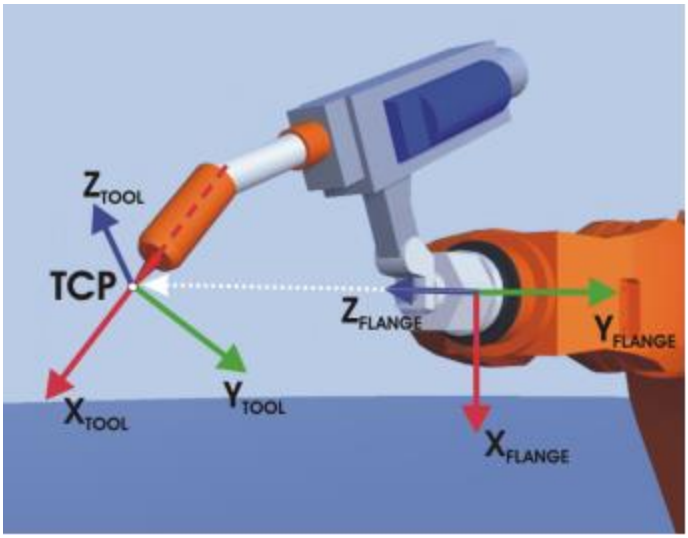


Ilustración 43 Valores a calibrar del herramental del robot KUKA. Ver referencia [39].

El proceso de calibración consta de dos pasos:

1. Definición del origen de la herramienta (TCP).
2. Definición de la orientación de la herramienta.

Para definir el origen de la herramienta se utiliza el método XYZ 4-Puntos. Este método consiste en aproximar el TCP a un punto de referencia desde 4 posiciones diferentes (ilustración 44). El robot calcula automáticamente el TCP de la herramienta.

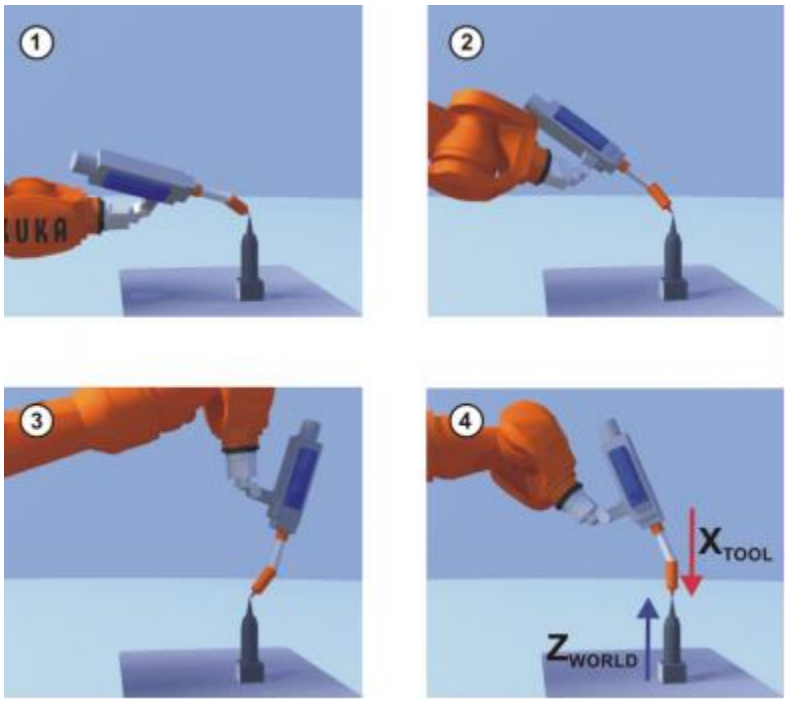


Ilustración 44 Calibración por método de 4 puntos. Ver referencia [39].

En cuanto a definir la orientación de la herramienta, se utiliza el método ABC Mundo. Este método consiste en alinear la herramienta a un patrón en posición vertical en donde (ilustración 45):

- Eje X herramienta = - Z calibración mundo
- Eje Y herramienta = Y calibración mundo
- Eje Z herramienta = X calibración mundo

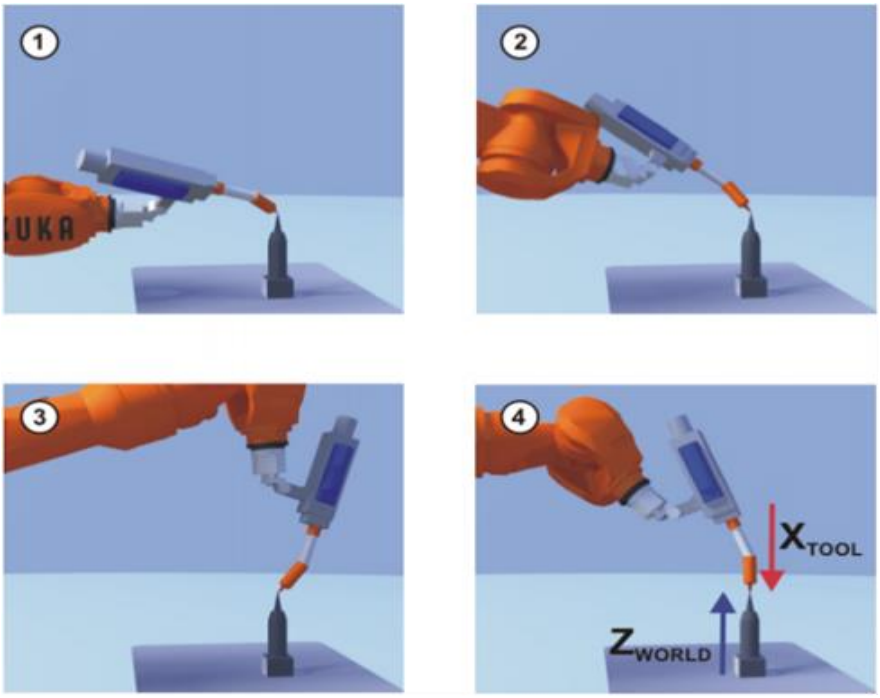


Ilustración 45 Calibración por método ABC mundo del robot KUKA. Ver referencia [39].

En cuanto a la calibración de la base, se pueden utilizar dos opciones: método directo de 3 puntos y el método indirecto de 4 puntos. El primero consiste en aproximar el TCP al origen del nuevo sistema de coordenadas, después mover el TCP a un punto sobre el eje X positivo de la nueva base y por último mover el TCP a un punto sobre el plano XY con un valor de Y positivo (ilustración 46).

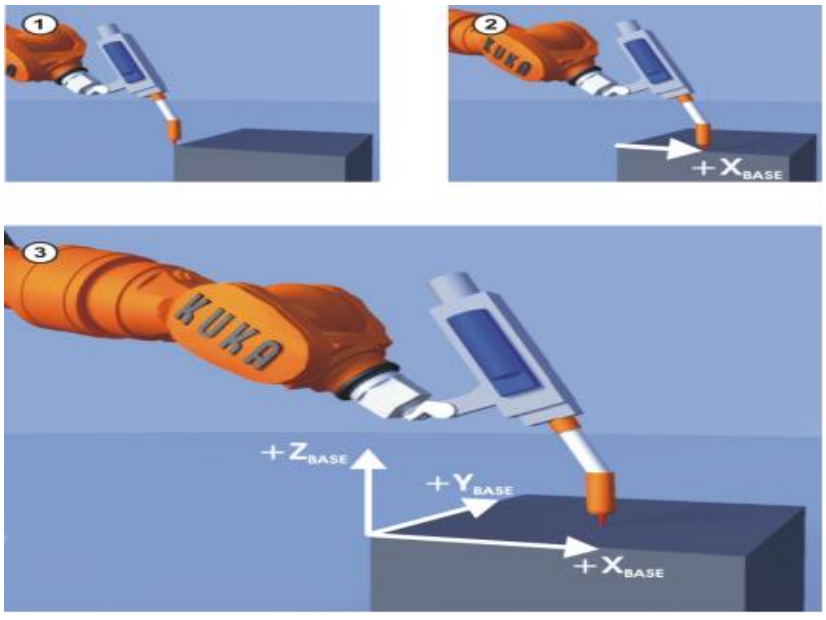


Ilustración 46 Calibración de una base de trabajo por método directo de 3 puntos. Ver referencia [39].

El segundo método consiste en mover el TCP a 4 puntos sobre la pieza de trabajo, de las cuales se conocen las coordenadas. Este método se utiliza solo cuando el origen de la pieza de trabajo no se encuentre accesible (ilustración 47).

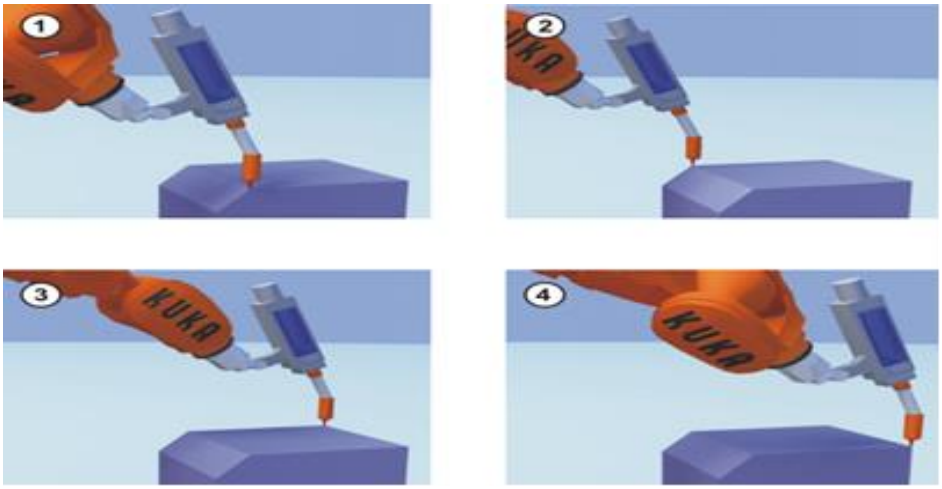


Ilustración 47 Calibración de base de trabajo por método de 4 puntos. Ver referencia [39].

Programación, usuario/experto

La programación tipo usuario es la programación básica con la que cuenta KUKA, en este tipo de programación solo se crean líneas de código donde guardamos movimientos del robot de un punto a otro mediante 3 tipos de movimientos disponibles: TCP, LIN y CIRC.

Para el movimiento TCP el robot se mueve a lo largo de la ruta más rápida, no necesariamente en línea recta pues no se puede predecir la trayectoria exacta. La velocidad de movimiento es programada en porcentaje de las velocidades articulares (ilustración 48).

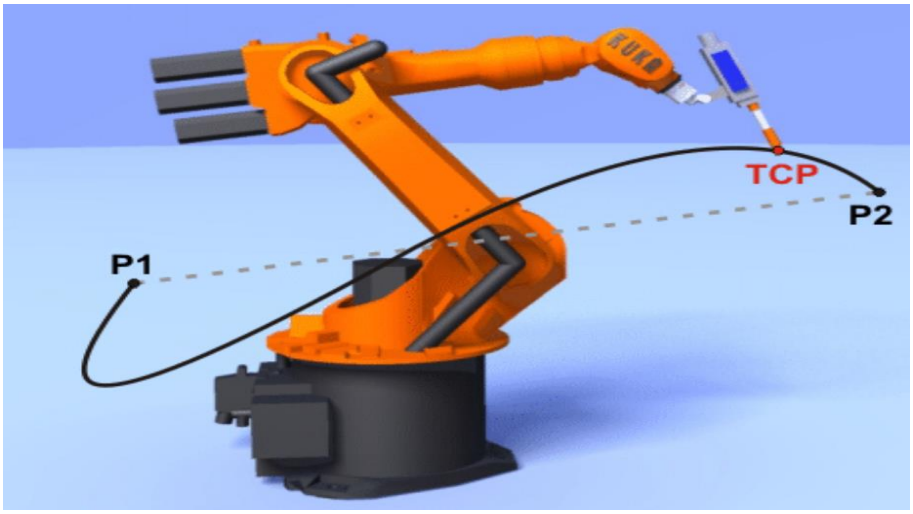


Ilustración 48 Movimiento TCP del robot KUKA. Ver referencia [39].

En la programación LIN el robot se desplaza en línea recta a una velocidad constante (ilustración 49).

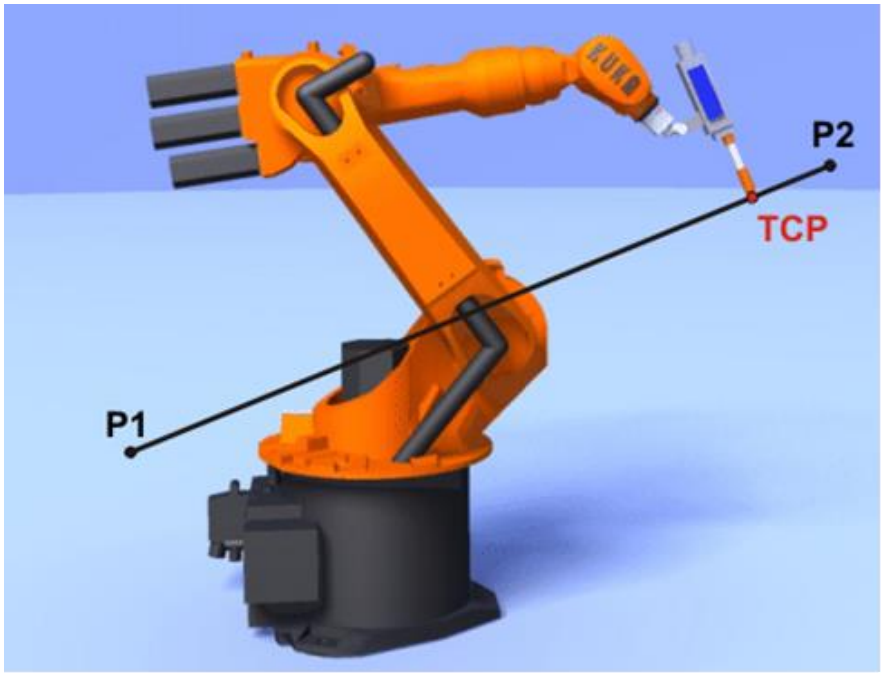


Ilustración 49 Movimiento LIN del robot KUKA. Ver referencia [39].

Para la programación CIRC, el robot se desplaza a velocidad constante a lo largo de una trayectoria circular. La trayectoria circular está definida por un punto inicial, un auxiliar y un punto final (ilustración 50).

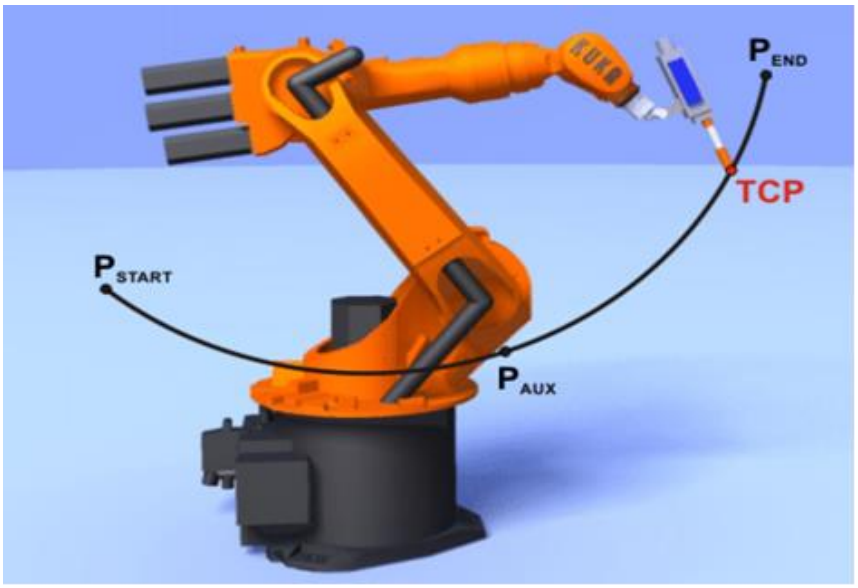


Ilustración 50 Movimiento CIRC del robot KUKA. Ver referencia [39].

A diferencia de la programación usuario, al ingresar a la programación experto se habilitan archivos generales del sistema para poder editarlos. Los ficheros que se habilitan son los siguientes (tabla 4):

Fichero	Uso
$MACHINE.DAT	Lista de datos del sistema con variables del sistema para la adaptación de la unidad de control del robot.
$ROBCOR.DAT	Lista de datos del sistema con datos para el modelo dinámico del robot.
MACHINE.UPG	Fichero para futuras actualizaciones
ROBCOR.UPG	Fichero para futuras actualizaciones.
CONFIG.DAT	Lista de datos del sistema con datos generales de configuración.
BAS.SRC	Paquete básico para el comando de movimientos.
IR_STOPM.SRC	Programa para el tratamiento de errores cuando se producen averías.
SPS.SUB	Fichero submit para controles paralelos.

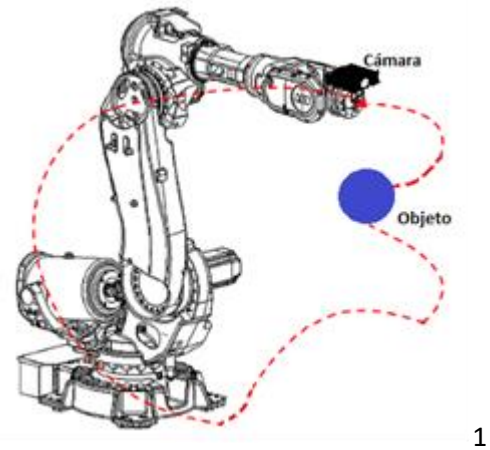
Tabla 4 Ficheros desbloqueados en programación experto del KUKA.

Al ingresar en este modo de programación y al habilitar los ficheros de programas, podemos hacer uso de operadores de programación:

- Variables y declaraciones
- Programación de movimientos
- Asistente KRL
- Control de ejecución de programa
- Instrucciones entradas/salidas
- Subprogramas y funciones
- Tratamiento de interrupciones
- Trigger -- Acciones de conmutación referentes a la trayectoria
- Lista de datos
- Editor externo

Modelo mano ojo

A diferencia del uso de cámaras fijas para la identificación de objetos, en la manufactura el uso del modelo mano-ojo permite una inspección del objeto más detallada sin importar el tamaño del mismo. Con el modelo mano-ojo se podría analizar cualquier cuerpo en cualquier ángulo y obtener diferentes imágenes del mismo con la finalidad de tener un mayor número de características y con esto que el reconocimiento pueda ser independiente de la rotación y posición, esto debido a que uno de los problemas en el reconocimiento de formas rígidas es la identificación de objetos con rotación (ilustración 51) [42].



1

Ilustración 51 Robot Antropomórfico con modelo de cámara mano-ojo.

Al obtener un análisis más exhaustivo de las figuras podemos disminuir el uso de varias cámaras, como por ejemplo en el uso de la visión estéreo donde ya no es necesario contar con dos cámaras para analizar la profundidad del objeto.

Capítulo 3

En este capítulo se muestra el desarrollo del proyecto, realizando en primera instancia un análisis de los requerimientos del hardware y software disponible.

Desarrollo

Considerando que el proyecto se puede dividir en 3 etapas como son: la captura de la imagen, la obtención del vector descriptor y la comunicación con el robot industrial, se pueden definir las limitantes que se tendrán en la realización del proyecto.

Para la etapa de captura de imagen, se tiene que hacer uso únicamente de dos cámaras de video: una Basler ace acA1300-30gc y una logitec C525, con distintas características mostradas en la siguiente tabla [36] [37]:

Características	Basler ace acA1300-30gc	Logitec C525
Conexión	ETHERNET	USB
Tipo de sensor	CCD	CMOS
Resolución	1.3 MP	8 MP
Cuadros por segundo	30 fps	30 fps
Consumo eléctrico	2.2 w	0.12 w

Tabla 5 Comparación entre cámara basler y logitec.

Con base en la tabla, uno de los aspectos más importantes a considerar, será que nuestro dispositivo tendrá que contar con un puerto ethernet, así como acceso para la conexión por USB (Universal Serial Bus). Además, ambas cámaras tendrán que ser utilizadas mediante los controladores y las librerías que provee el fabricante, por lo cual, el sistema embebido tendrá que soportar la implementación de un sistema operativo basado en Linux.

Respecto a la comunicación entre el robot tipo industrial y nuestro sistema embebido, se tiene que considerar que el robot KUKA cuenta con un software privado, por lo cual, se tiene acceso únicamente a una librería para realizar la comunicación entre ambos. Esta librería se encuentra escrita en dos lenguajes de programación: Python y Java, por lo que se tendrá que tomar en consideración para la selección del sistema embebido.

Otro aspecto importante en la comunicación es la interconectividad, el tener acceso a la comunicación entre nuestro dispositivo y el robot industrial con un tercer equipo, para que este pueda tener control de nuestro sistema. Esta interconexión se realizará mediante la conexión con una red LAN (Local Area Network), por lo tanto,

el dispositivo tendrá que tener acceso a un protocolo de comunicación 802.11, también conocido como WIFI (Wireless Fidelity). Además, el dispositivo tendrá que soportar la realización de un protocolo de comunicación TCP/IP (Transmission Control Protocol/Internet Protocol), para su conexión con Internet.

En la tabla 6 se puede observar algunas de las especificaciones más relevantes en distintos sistemas embebidos.

	UDOO	*LATTEPANDA*	*BEAGLEBONE*	*RASPBERRY PI*
Velocidad de procesador	1 GHz	1.8 GHz	1 GHz	1.2 GHz
Puerto Ethernet	Si	Si	Si	Si
Wifi	No	Si	Si	Si
RAM	512 MB	1 GB	512 MB	1 GB
Sistema operativo	Linux	Windows	Linux	Linux
Costo	50 USD	110 USD	55 USD	50 USD

Tabla 6 Características de distinto sistemas embebidos.

Considerando los requisitos antes mencionados como aspectos principales para trabajar, podemos descartas el sistema embebido LATTEPANDA y UDOO. En el caso de UDOO, se descarta por no contar con conexión WIFI para realizar de una manera más sencilla la interconexión con otros dispositivos, el cual es un concepto fundamental en la industria 4.0, mientras que la LATTEPANDA, por no tener como sistema nativo un sistema operativo basado en Linux, ya que en ese sistema se tendrán que instalar los controladores y librerías de las cámaras, en especial la cámara basler, así como la librería de comunicación con el robot industrial.

Podremos utilizar una RASPBERRY PI o una BEAGLEBONE, ya que ambos cuentan con los principales requisitos para trabajar, pero por tener un procesador que puede realizar un mayor número de instrucciones por segundo, así como también por tener un costo económico más reducido, haremos uso de la Raspberry Pi.

Estructura del proyecto

El proceso para realizar el trabajo se muestra en la ilustración 52. Se tendrá que realizar en primera instancia la calibración de la herramienta y la base de trabajo del robot KUKA, es necesario como primer punto porque nos garantiza tener las medidas de seguridad necesarias para trabajar con un robot tipo industrial. El siguiente punto es realizar la captura de la imagen, ya que esta nos brindará las características necesarias para la obtención de nuestro BOF y, por último, se realizará el movimiento del robot KUKA, esto debido a que nuestro BOF nos proporcionará las coordenadas para la ubicación de nuestro objeto en el área de trabajo.

Ilustración 52 Diseño de bloques del sistema de reconocimiento.

Calibración de la herramienta y base.

El primer proceso a realizar será la calibración de una herramienta y una base, las cuales serán nuestra herramienta y área de trabajo. La herramienta se calibrará por el método XYZ y la base se calibrará por el método de los 3 puntos.

Para realizar la calibración de la herramienta se utilizará un objeto patrón vertical, para que el robot pueda calcular la distancia que tiene nuestra herramienta con respecto al último eje del KUKA (ilustración 53).

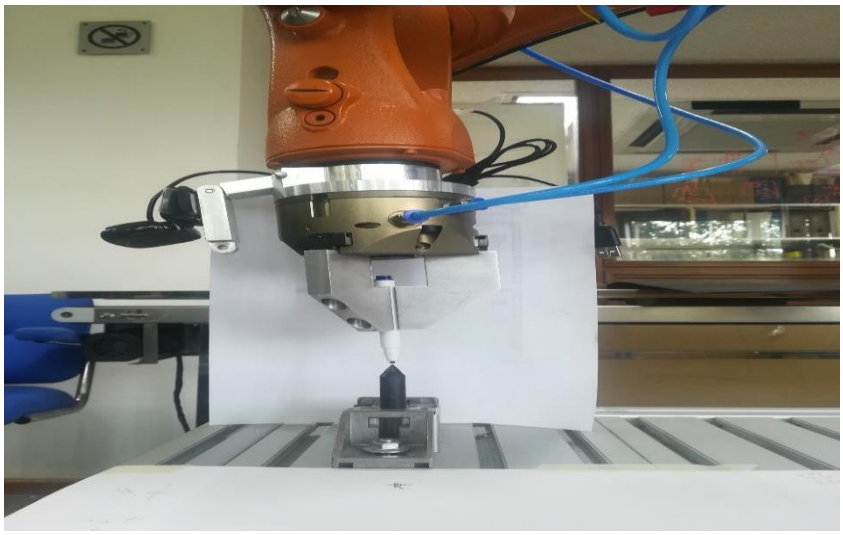


Ilustración 53 Herramienta patrón para calibración de herramental del KUKA.

El siguiente paso de este método consiste en ubicar en diferentes posiciones el herramental en el objeto patrón, esto para obtener los ángulos de inclinación respecto a nuestra área de trabajo (ilustración 54).

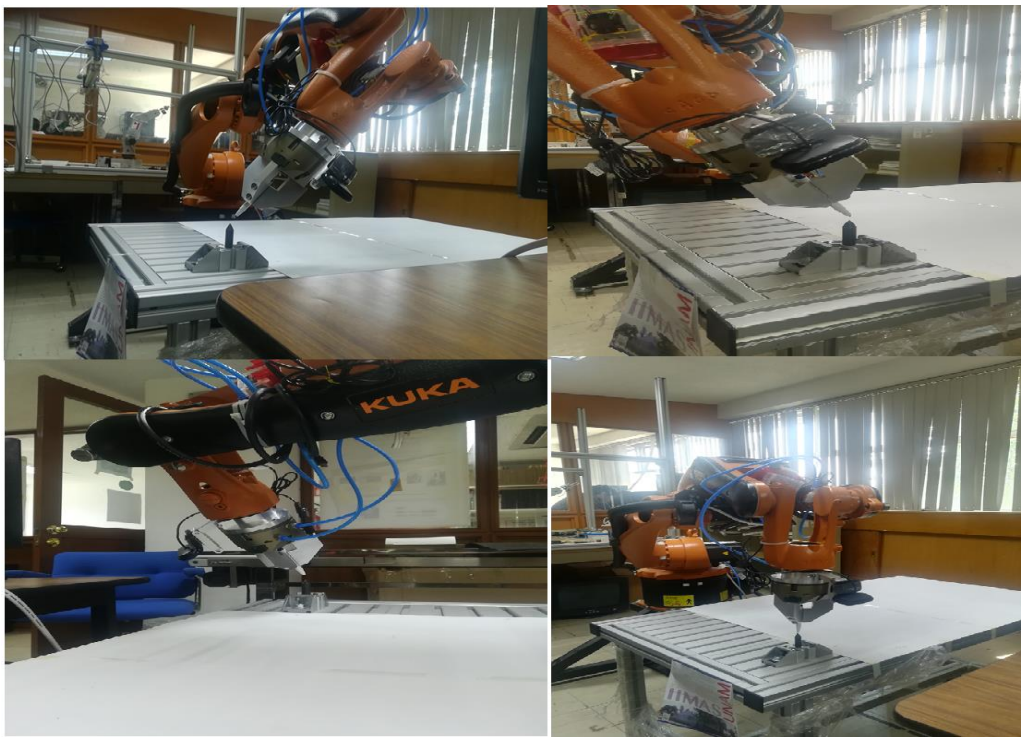


Ilustración 54 Proceso de calibración del herramental.

Realizada la calibración de la herramienta se prosigue a calibrar nuestra área de trabajo, esto por medio del método de 3 puntos, que consiste en indicarle al robot el punto de origen de nuestro sistema de coordenadas y, consecutivamente, indicarle la dirección del eje X, así como el desplazamiento que tendrá en el eje Y,

considerando cual será la mayor distancia que alcance el robot en estos ejes (ilustración 55).

Ilustración 55 Proceso de calibración de área de trabajo.

Sistema operativo de la Raspberry Pi 3

La raspberry pi 3 utiliza raspbian, que es un sistema operativo GNU/Linux basado en Debian 9.4 y optimizado para el hardware Raspberry Pi. Como lenguaje de programación nativo con el que cuenta raspbian, se encuentra “Python”, el cual es un lenguaje de programación de alto nivel y permite escribir programas legibles y más compactos, si se compara con lenguajes como “C”, “C++” o “Java”, ya que no es necesario declarar variables ni argumentos, tampoco es necesario abrir y cerrar llaves, ya que la agrupación de instrucciones se hace por sangría [35].

Adquisición de la imagen

Como se mencionó se cuentan con dos cámaras de video para la realización del proyecto, el uso de alguna de estas cámaras depende de la aplicación de la celda de manufactura, en este caso se utilizarán ambas cámaras: la cámara logitec, se utilizará para la ubicación de la figura en el área de trabajo, esto debido a que posee un ángulo de visión mayor respecto a la cámara basler (ver anexo I), mientras que la cámara basler, se utilizará para la inspección de figuras, ya que al poseer un sensor CCD, cuenta con una mejor sensibilidad en la captura de luz, lo cual resulta en un mejor contraste entre pixeles, comparándolo con el sensor CMOS que posee la cámara logitec y con esto facilita el procesamiento de la imagen (ilustración 56).

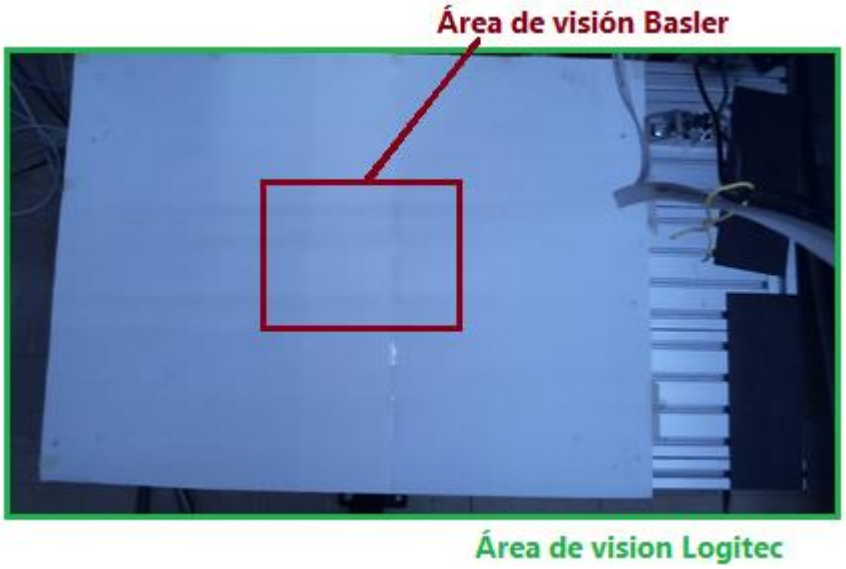


Ilustración 56 Comparación del área de visión basler vs logitec.

El proceso de adquisición de imagen iniciará con el uso de la cámara logitec para ubicar el objeto en el área de trabajo. Para hacer uso de esta cámara se tendrá que utilizar los controladores para la raspberry "fswebcam". Los controladores se tendrán que instalar desde la terminal de la raspberry dando permisos de administrador con la palabra reservada "sudo": sudo apt-get install fswebcam.

La imagen se obtiene mediante la función *fswebcam -r 1280x720 –no—banner NombreImagen.jpg*, donde: *fswebcam* hace referencia al paquete de instalación, *-r* define la resolución de la imagen, *-- no – banner* quita todo título dentro de la imagen y NombreImagen.jpg define la ruta y el nombre con la extensión de imagen sea JPG, BITMAP, PNG como se encontrará la imagen (ilustración 57).

```
################
# cargar librerías
import math
import cv2
import numpy as np
from matplotlib import pyplot as plt
import os
import time
import sys
from kukavarproxy import *
from time import sleep
#############################################################################
os.system("fswebcam -r 1280x720 --no-banner /home/pi/imagenPrueba.jpg")
image = cv2.imread('imagenPrueba.jpg')
```

Ilustración 57 Código para captura de imagen con cámara logitec.

Para el uso de la cámara basler será necesario instalar los controladores *"Basler pylon camera"*, estos son paquetes de instalación para el sistema operativo Linux, los cuales están adaptados para el uso de la raspberry pi. También se realizará la instalación de la librería para desarrolladores "pypylon", la cual nos proporciona el fabricante, esto para poder manipular la cámara desde líneas de código en Python, ya que el fabricante solo permite el uso de esta cámara mediante ciertos programas autorizados (ilustración 58).

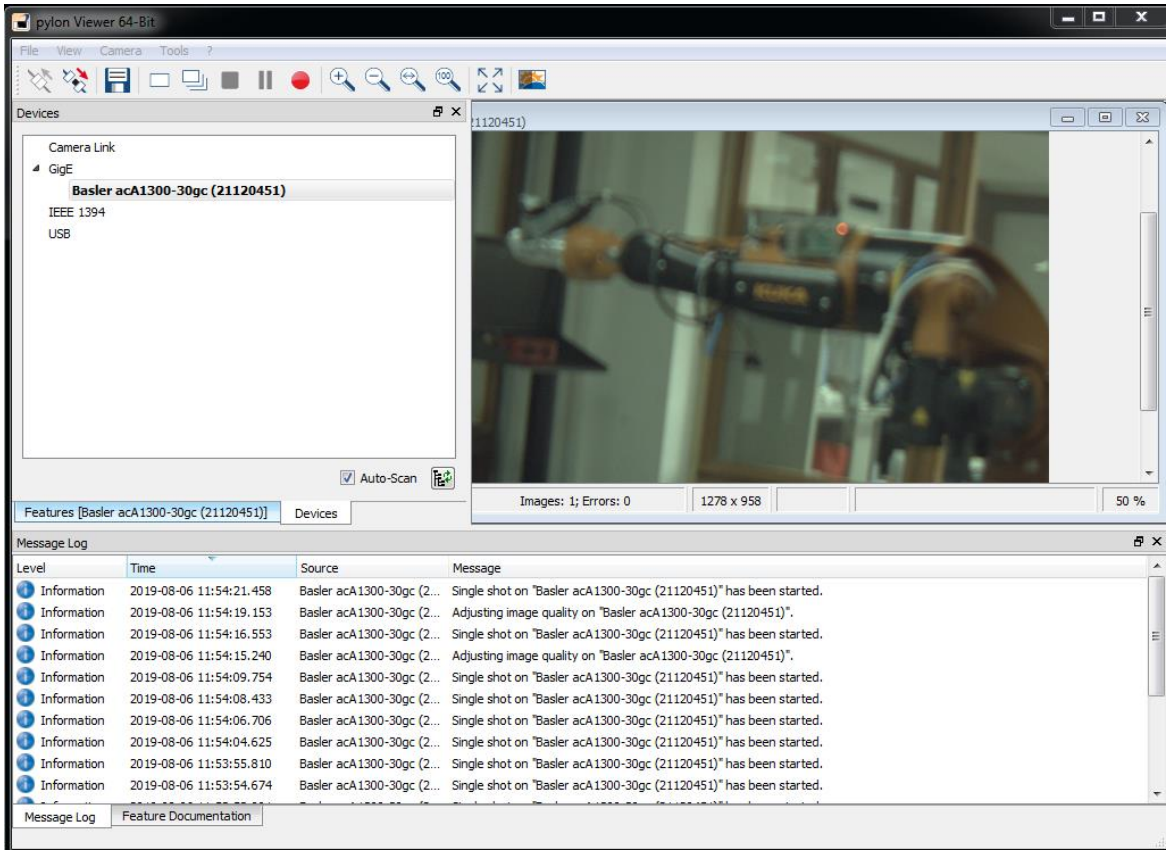


Ilustración 58 Programa para uso de la cámara Basler.

La comunicación de la cámara con la raspberry se realizará por medio del protocolo de comunicación TCP/IP. Se realizará una conexión fija, creando una IP estática en la raspberry pi en un rango de trabajo definida por la cámara, este rango de valores se encuentra entre: 172.16.0.1 a 172.32.255.254, con una máscara de sub red de 255.255.0.0 o también de una IP de 192.168.0.1 a 192.168.255.254, con una máscara de sub red de 255.255.255.0 (ilustración 59) [38].

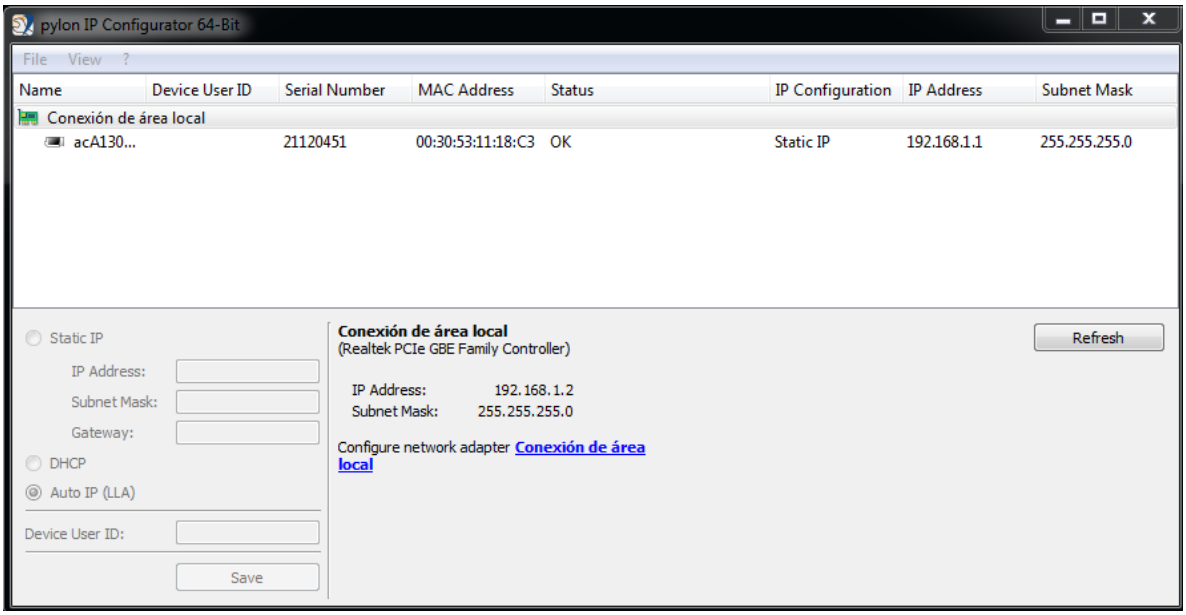


Ilustración 59 Estado de conexión de cámara basler.

Una vez establecida la conexión y teniendo las librerías “pypylon”, se podrá hacer uso de la cámara por medio de un programa ejecutado en python. La cámara basler se utilizará para la inspección de las figuras una vez ubicadas en el espacio de trabajo.

```
from pypylon import pylon
import cv2

# conecting to the first available camera
camera = pylon.InstantCamera(pylon.TlFactory.GetInstance().CreateFirstDevice())

# Grabing Continusely (video) with minimal delay
camera.StartGrabbing(pylon.GrabStrategy_LatestImageOnly)
converter = pylon.ImageFormatConverter()

# converting to opencv bgr format
converter.OutputPixelFormat = pylon.PixelType_BGR8packed
converter.OutputBitAlignment = pylon.OutputBitAlignment_MsbAligned

while camera.IsGrabbing():
    grabResult = camera.RetrieveResult(5000, pylon.TimeoutHandling_ThrowException)

    if grabResult.GrabSucceeded():
        # Access the image data
        image = converter.Convert(grabResult)
        img = image.GetArray()

    grabResult.Release()

# Releasing the resource
camera.StopGrabbing()
```

Ilustración 60 Algoritmo para captura de imagen con cámara basler en formato RGB.

El algoritmo de la ilustración 60, en primera instancia realiza la selección de la cámara instalada, en este caso la cámara basler, para posteriormente inicializarla. El siguiente paso define el formato de la imagen capturada por la cámara, el cual será en formato RGB. Finalmente, el algoritmo realiza un ciclo de capturas de

imagen con la finalidad de grabar y dar tiempo a la cámara de hacer el enfoque automático de acuerdo a la cantidad de luz en el ambiente.

Obtención del BOF

El procedimiento para realizar el algoritmo de obtención del BOF es el siguiente [40]:

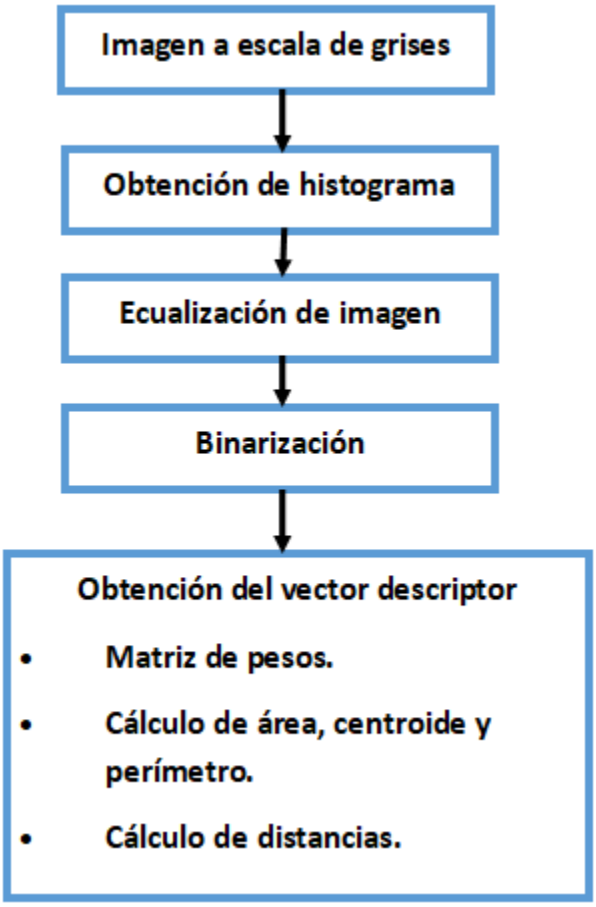


Ilustración 61 Procedimiento para la obtención del BOF.

Escala de grises

Uno de los **problemas más importante que representa este trabajo es el poder obtener el reconocimiento de figuras en un sistema embebido**, en este caso nuestra raspberry pi, que a diferencia de una PC tiene menos prestaciones computacionales, por lo cual será necesario proponer métodos para obtener una mayor optimización y pueda realizar todo el proceso necesario para reconocer los objetos.

Para iniciar con la obtención del BOF será necesario considerar la resolución con la que obtendremos nuestra imagen, ya que una mayor resolución (ilustración 62) tendrá como consecuencia un mayor análisis de pixeles de nuestra imagen, que a su vez representa un mayor tiempo de procesamiento.

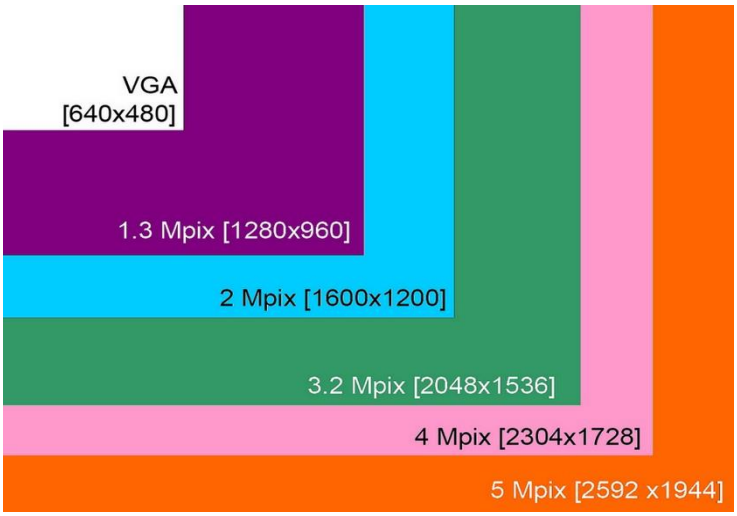


Ilustración 62 Escala de las resoluciones de imágenes.

El siguiente paso para el procesamiento de imágenes consiste en obtener una imagen en escala de grises a partir de una imagen a color (ilustración 63). La idea de trabajar con una imagen a escala de grises radica en obtener una mayor velocidad en el procesamiento de la imagen, esto se debe a que, a diferencia de una imagen a color, la imagen a grises trabaja con una sola matriz con los valores de los pixeles y no con tres matrices que componen a una imagen a color. La desventaja de trabajar con una matriz a escala de grises se encuentra en diferenciar el contraste de lo que representa el fondo de la imagen con la figura a reconocer.

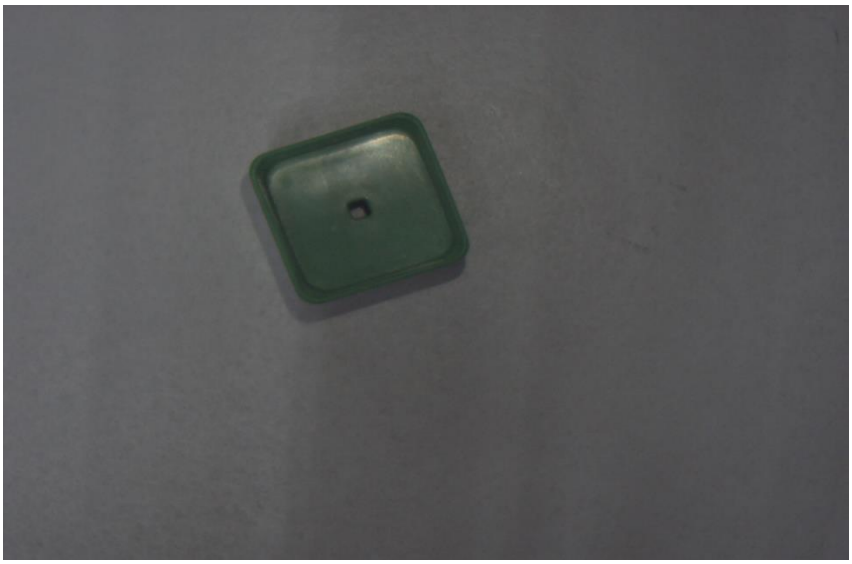


Ilustración 63 Captura de imagen con cámara basler a color.

Al convertir la imagen a escala de grises mediante la ecuación 2, se trabaja únicamente con una matriz de 8 bits al promediar las 3 matrices (ilustración 64).

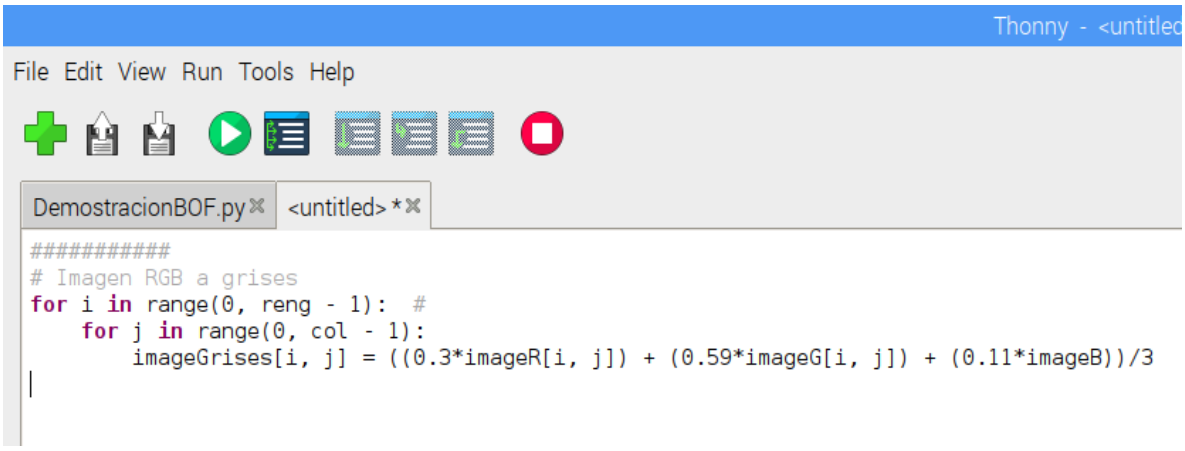


Ilustración 64 Algoritmo para convertir imagen de color a escala de grises.

En este algoritmo se realiza la lectura de cada uno de los pixeles haciendo un recorrido en todas las matrices, se realiza un promedio multiplicando cada pixel por un factor que nos indica la sensibilidad del ojo humano a las frecuencias del espectro cercanas al rojo, verde y azul. Al realizar este algoritmo se tiene una matriz donde cada pixel estará representado en una escala de grises con valores de 0 a 255 (ilustración 65).

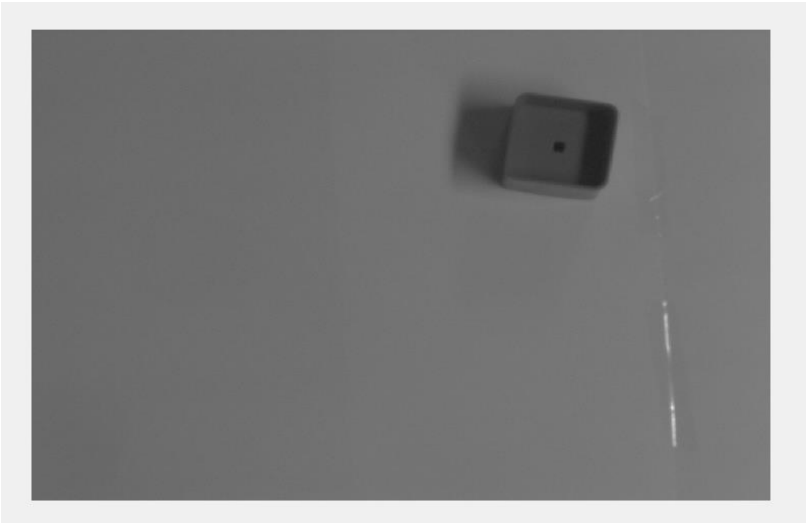


Ilustración 65 Imagen de un cubo a escala de grises.

Histograma y ecualización

Obteniendo una imagen a escala de grises es necesario conocer la distribución de los grises en nuestra matriz, lo cual lo logramos mediante las ecuaciones 3 y 4 (ilustración 66).

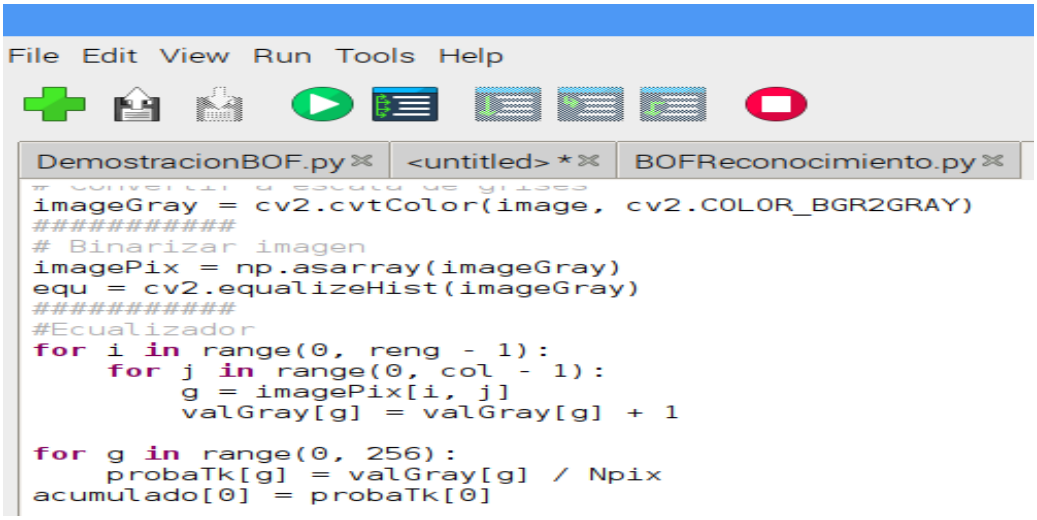


Ilustración 66 Algoritmo para la realización del histograma.

El algoritmo realiza un recorrido en toda la matriz y guardamos en un arreglo llamado "valGray[g]" la cantidad de pixeles que conforman cada tonalidad de grises.

Este vector de 255 datos contiene la cantidad de pixeles de acuerdo al nivel de gris de la imagen.

$$valGray[g] = [0, 0, 0, 0 \ldots \ 23, 45, 743, \ldots 0{,}0]$$

Se puede observar en la ilustración 67, la distribución de los grises en nuestra imagen, donde, los valores más bajos representan grises más oscuros y los valores más altos representan grises más claros.

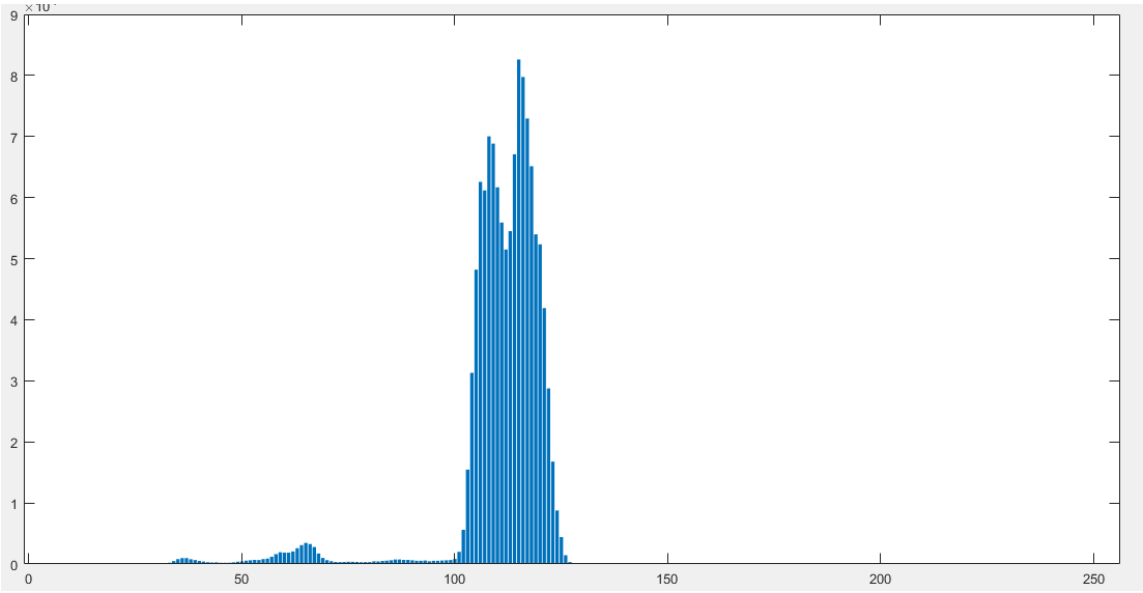


Ilustración 67 Distribución de grises de la imagen de un cubo.

A partir del arreglo “valGray[g]”, el cual contiene el histograma con la cantidad de pixeles referente a cada rango de grises de nuestra imagen, podremos realizar el proceso de ecualización que consiste en asignar la misma probabilidad para encontrar cualquier pixel en todo nuestro rango del histograma mediante las ecuaciones 6 y 7 (ilustración 68).

```
#Ecualizador
for i in range(0, reng - 1):
    for j in range(0, col - 1):
        g = imagePix[i, j]
        valGray[g] = valGray[g] + 1

for g in range(0, 256):
    probaTk[g] = valGray[g] / Npix
acumulado[0] = probaTk[0]

for g in range(1, 256):
    acumulado[g] = acumulado[g-1] + probaTk[g]

for g in range(0, 256):
    ecualizacion[g] = acumulado[g] * 256

for i in range(0, reng - 1):
    for j in range(0, col - 1):
        g = imagePix[i, j]
        imageEcualiza[i, j] = ecualizacion[g]
```

Ilustración 68 Algoritmo para realizar la ecualización.

Lo que se realiza en el código es encontrar la probabilidad de una tonalidad del gris de un pixel, para tener un vector de probabilidad acumulada, el cual consiste en que la probabilidad acumulada “K” fuera igual a la acumulada anterior y posteriormente realizar su transformación a su nuevo nivel de gris.

Al finalizar el proceso de ecualización obtenemos una mejor distribución de los grises en toda nuestra imagen (ilustración 69).

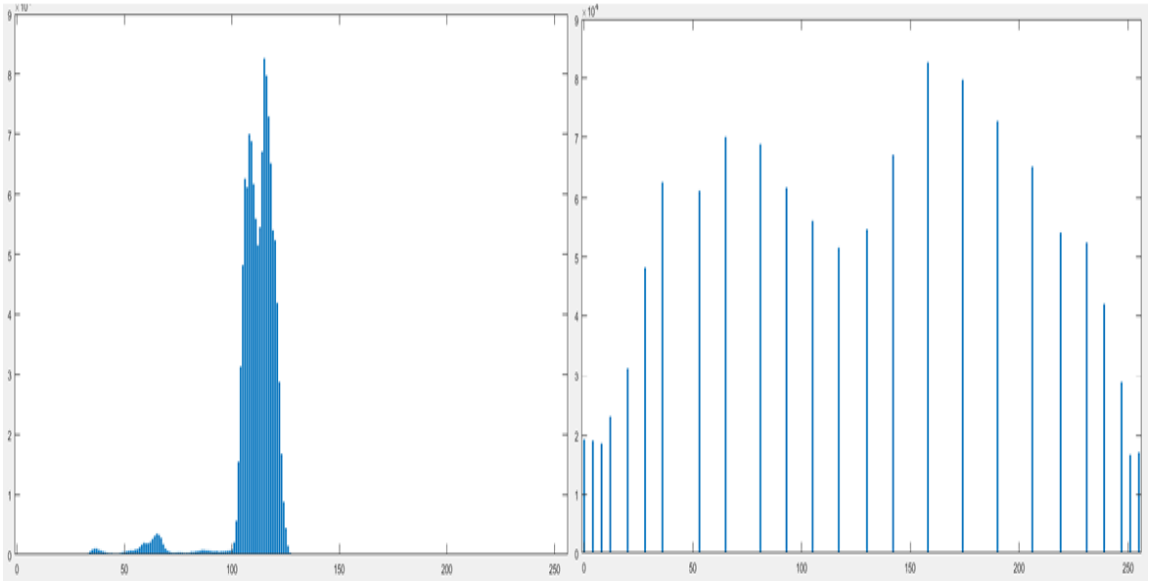


Ilustración 69 Comparación entre el histograma y el histograma ecualizado de un cubo.

El resultado del proceso de ecualización se observa en la ilustración 70, donde podemos definir un mejor contraste entre lo que representa el fondo de la imagen con el objeto que se tiene que reconocer, de esta manera se corrige el problema de no utilizar una imagen a color para realizar este proceso de reconocimiento.

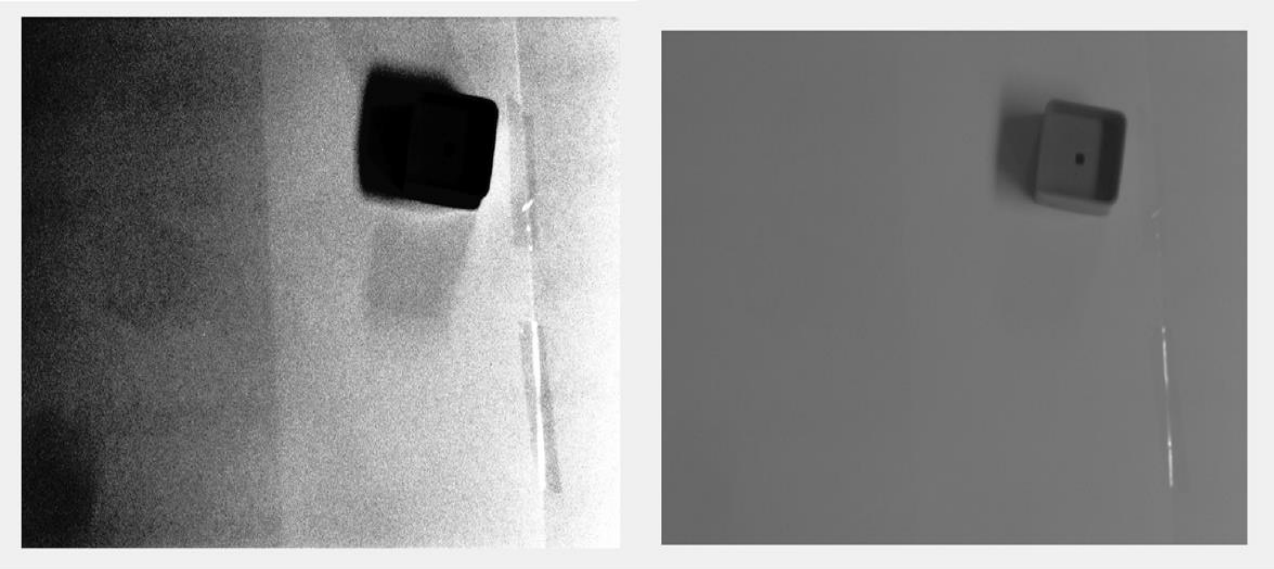


Ilustración 70 Comparación de imagen ecualizada y a escala de grises.

binarización

La binarización consiste en trabajar una imagen con colores blanco y negro que se representa en nuestra matriz con “0” y “1”, al pasar de 8 bits a 1 bit. Este proceso lo logramos al comparar nuestro histograma ecualizado con un umbral “T”, para definir en nuestra matriz ecualizada que valores corresponderán a 0 y 1.

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

Los umbrales pueden ser estáticos o dinámicos, por nuestra aplicación y los diferentes usos que se pueden tomar una celda de manufactura exponiéndola a diferentes fuentes de luz en los procesos de fabricación, el umbral se realizará de manera dinámica. A partir de nuestro primer histograma, correspondiente a la imagen a escala de grises, obtendremos los valores máximos de las dos crestas más pronunciadas y con esto la distancia media de ambas crestas será el umbral “T” (ilustración 71).

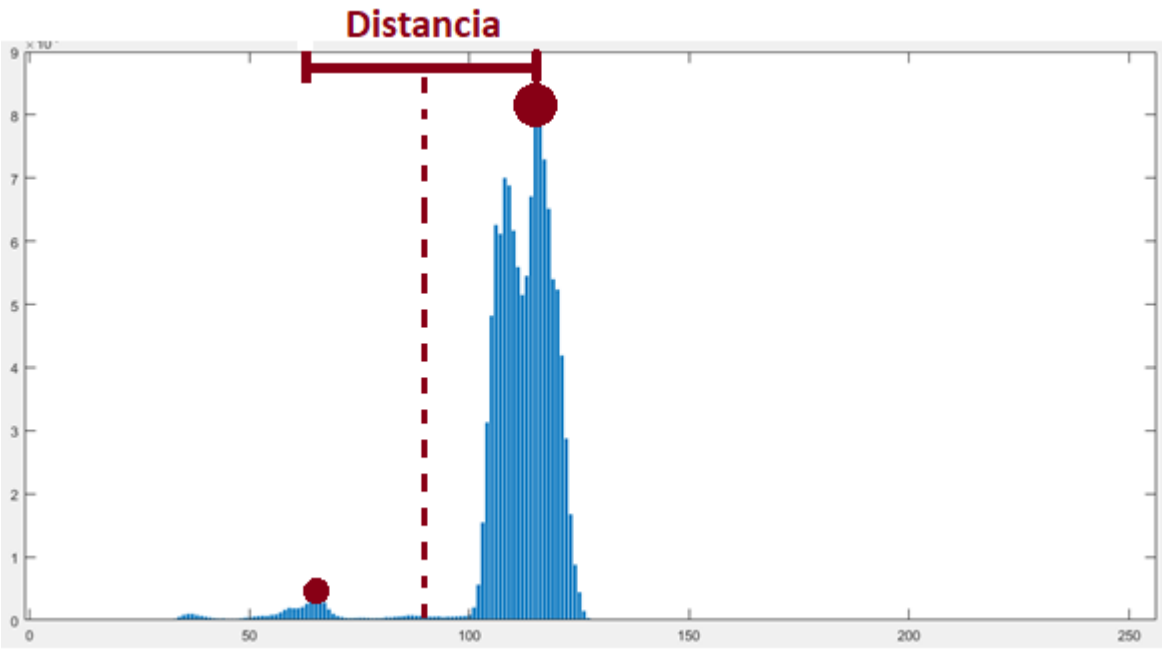


Ilustración 71 Proceso para un umbral dinámico.

Al obtener el valor de umbral “T”, analizaremos toda la imagen para realizar la comparación de cada uno de los pixeles y obtener una imagen a blanco y negro (ilustración 72).

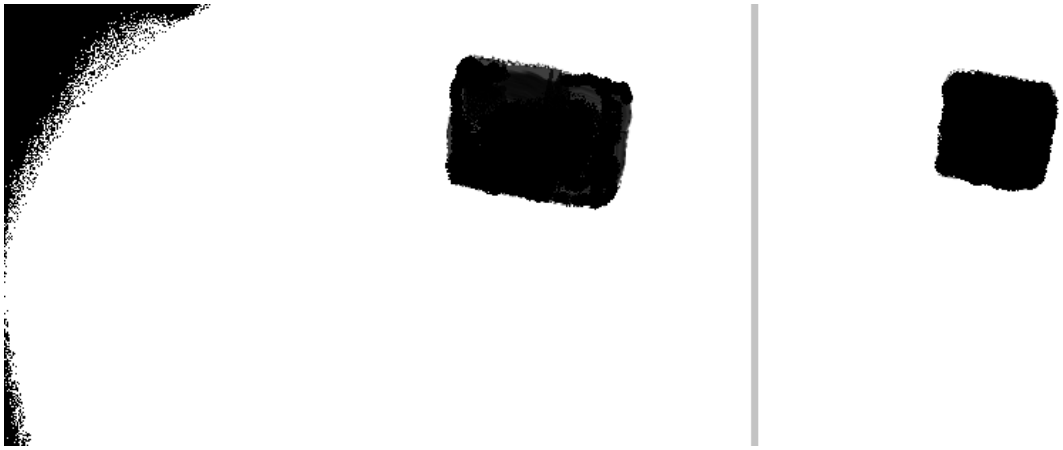


Ilustración 72 Imagen binarizada sin ecualizar y ecualizada.

Obtención de vector descriptor

Para la obtención del vector que definirá cada figura, es necesario calcular ciertas características en el objeto de la imagen como son: matriz de pesos, el área, perímetro y centroide.

Para calcular lo que llamaremos la matriz de pesos, que es una matriz que representa la distribución de los pixeles que se encuentran más cerca del contorno de la imagen, a los que se encuentran al centro de la misma (ilustración 73).

```
# Matriz de peso de contorno
for i in range(0, reng - 1):
    for j in range(0, col - 1):
        for m in range(i - 1, i + 2):
            for n in range(j - 1, j + 2):
                imageCont_Aux[i, j] = imageCont_Aux[i, j] + imagePixBin[m, n]
```

Ilustración 73 Código para la obtención de la matriz de pesos.

El algoritmo realiza por medio de una matriz de 3x3 un recorrido en toda la imagen, sumando para cada pixel de posición P(i, j) de la imagen, los valores de los pixeles vecinos y el resultado colocándolo en una nueva matriz en la misma posición (ilustración 74).

$$\mathrm{Pcentro}(i,j) = \mathrm{P}(i,j) + \mathrm{P}(i-1,j-1) + \mathrm{P}(i-1,j) + \mathrm{P}(i-1,j+1) + \cdots$$

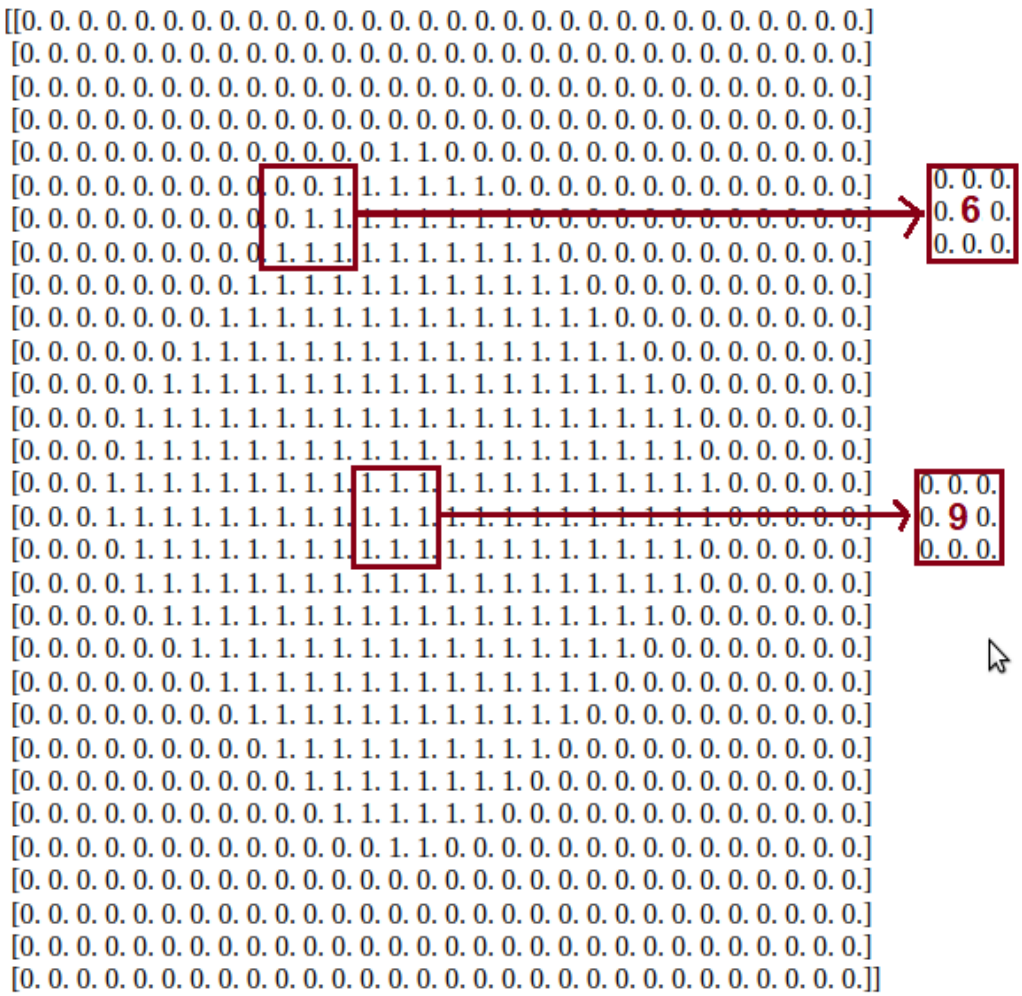


Ilustración 74 Proceso gráfico para obtención de matriz de pesos.

El resultado de este proceso es obtener una nueva matriz, donde los valores de los pixeles más grandes se encuentran en el centro de la figura, mientras que los pixeles con valores más pequeños se encuentran ubicados en el contorno (ilustración 75).

```
[[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 2. 2. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 2. 4. 5. 5. 4. 2. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 3. 5. 7. 8. 8. 7. 5. 3. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 1. 3. 6. 8. 9. 9. 9. 9. 8. 6. 3. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 1. 3. 6. 8. 9. 9. 9. 9. 9. 9. 8. 6. 3. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 1. 3. 6. 8. 9. 9. 9. 9. 9. 9. 9. 9. 8. 6. 3. 1. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 1. 3. 6. 8. 9. 9. 9. 9. 9. 9. 9. 9. 9. 9. 8. 6. 3. 1. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 1. 3. 6. 8. 9. 9. 9. 9. 9. 9. 9. 9. 9. 9. 9. 9. 8. 6. 3. 1. 0. 0. 0. 0. 0.]
 [0. 0. 0. 1. 3. 6. 8. 9. 9. 9. 9. 9. 9. 9. 9. 9. 9. 9. 9. 9. 9. 8. 6. 3. 1. 0. 0. 0. 0.]
 [0. 0. 0. 2. 5. 8. 9. 9. 9. 9. 9. 9. 9. 9. 9. 9. 9. 9. 9. 9. 9. 9. 8. 5. 2. 0. 0. 0. 0.]
 [0. 0. 1. 4. 7. 9. 9. 9. 9. 9. 9. 9. 9. 9. 9. 9. 9. 9. 9. 9. 9. 9. 9. 7. 4. 1. 0. 0. 0.]
 [0. 0. 2. 5. 8. 9. 9. 9. 9. 9. 9. 9. 9. 9. 9. 9. 9. 9. 9. 9. 9. 9. 9. 8. 5. 2. 0. 0. 0.]
 [0. 0. 2. 5. 8. 9. 9. 9. 9. 9. 9. 9. 9. 9. 9. 9. 9. 9. 9. 9. 9. 9. 9. 8. 5. 2. 0. 0. 0.]
 [0. 0. 1. 4. 7. 9. 9. 9. 9. 9. 9. 9. 9. 9. 9. 9. 9. 9. 9. 9. 9. 9. 9. 7. 4. 1. 0. 0. 0.]
 [0. 0. 0. 2. 5. 8. 9. 9. 9. 9. 9. 9. 9. 9. 9. 9. 9. 9. 9. 9. 9. 9. 8. 5. 2. 0. 0. 0. 0.]
 [0. 0. 0. 1. 3. 6. 8. 9. 9. 9. 9. 9. 9. 9. 9. 9. 9. 9. 9. 9. 9. 8. 6. 3. 1. 0. 0. 0. 0.]
 [0. 0. 0. 0. 1. 3. 6. 8. 9. 9. 9. 9. 9. 9. 9. 9. 9. 9. 9. 8. 6. 3. 1. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 1. 3. 6. 8. 9. 9. 9. 9. 9. 9. 9. 9. 9. 8. 6. 3. 1. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 1. 3. 6. 8. 9. 9. 9. 9. 9. 9. 9. 8. 6. 3. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 1. 3. 6. 8. 9. 9. 9. 9. 9. 8. 6. 3. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 1. 3. 6. 8. 9. 9. 9. 9. 8. 6. 3. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 3. 5. 7. 8. 8. 7. 5. 3. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 2. 4. 5. 5. 4. 2. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 2. 2. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]]
```

Ilustración 75 Matriz de pesos de un cubo.

Como se mencionó la matriz de pesos, distingue una relación de los pixeles que se encuentran más alejados del centro de la figura, es decir, los pixeles con un menor valor numérico son los que se encuentran en el contorno y los que contienen un mayor valor numérico se encuentran en el centro. Entonces, definimos un valor “c” que permita comparar y hacer una elección de los pixeles que se encuentran en el contorno y al realizar la sumatoria de éstos tendremos el valor del perímetro (ilustración 76).

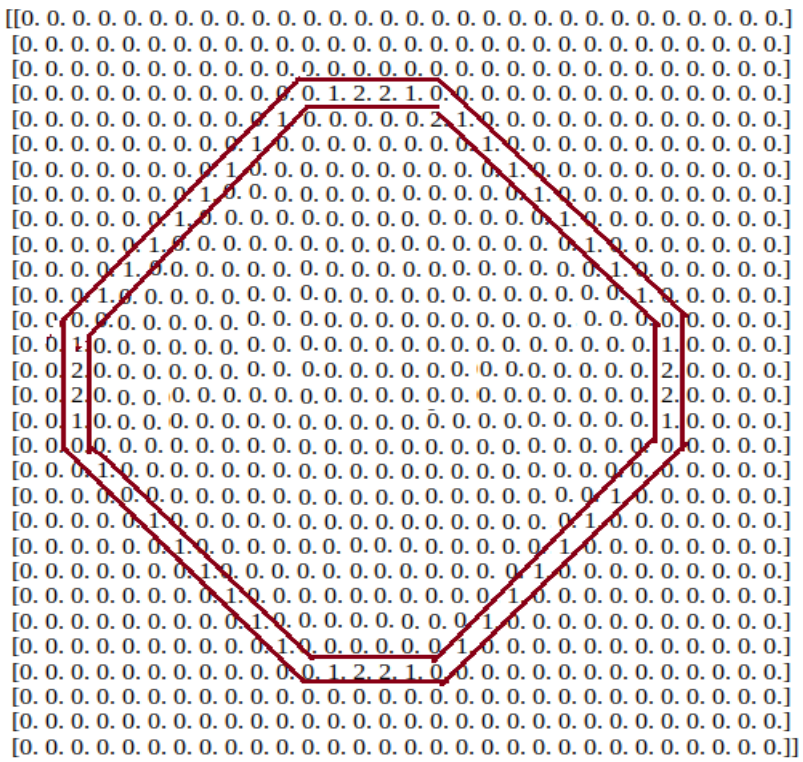


Ilustración 76 Obtención del contorno de un cubo.

El siguiente paso para definir el vector descriptor, será calcular el centroide de la figura mediante el siguiente algoritmo, haciendo uso de las ecuaciones 10, 11, 12 y 13:

```
##########
# Cálculo de área
for i in range(0, reng - 1):
    for j in range(0, col - 1):
        if imagePixBin[i, j] == 0:
            area = area + 1
            x = x + i
            y = y + j

centrox = x / area
centroy = y / area
centrox = math.floor(centrox)
centroy = math.floor(centroy)
print('Cx =', centrox, 'Cy =', centroy, 'área =', area)
###########
```

Ilustración 77 Algoritmo para el cálculo del centroide.

El algoritmo de la ilustración 77 muestra el cálculo de área y centroide. El cálculo del área, se realiza recorriendo los pixeles de la imagen y realizando la sumatoria de la cantidad de pixeles diferentes de 1 (imagen binarizada), mientras que al mismo tiempo vamos obteniendo la mayor distancia que tiene la figura en los ejes X y Y.

Al realizar el cálculo del centroide, lo siguiente será encontrar distancias que vayan del centroide a un punto del contorno de la figura mediante la ecuación 14. Para tener una distribución equitativa de toda la figura estas distancias serán calculadas cada grado “n”, dependiendo del número de puntos que el vector característico tendrá. Hay que tener en cuenta que entre mayor número de puntos la figura será mejor representada en nuestro vector en comparación de tener una menor cantidad de puntos (ilustración 78). El querer realizar este algoritmo en un sistema embebido como la Raspberry hay que tener en cuenta que un mayor número de puntos requerirá de un mayor procesamiento que deberá realizar la Raspberry.

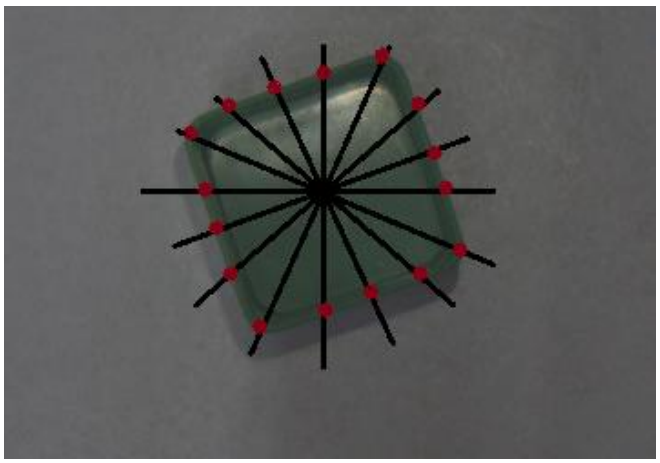


Ilustración 78 Puntos representativos de un cubo para la obtención del BOF.

Actualmente existen diferentes métodos para dibujar o distinguir líneas a diferentes grados en una imagen, el más utilizado por su precisión y rapidez es el algoritmo de Bresenham, el cual permite dibujar pixeles consecutivos en una imagen a cualquier ángulo (ilustración 79).

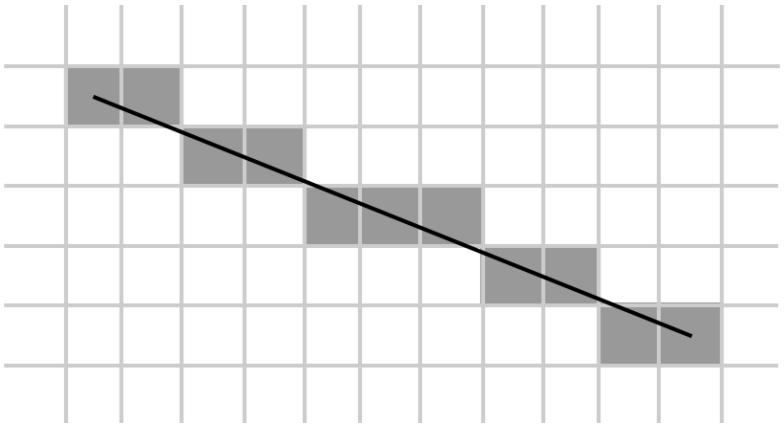


Ilustración 79 Línea recta usando algoritmo Bresenham.

El problema de utilizar este algoritmo en una raspberry repercute en gran medida en el tiempo de procesamiento de la imagen, ya que, el algoritmo tiene que leer todos los pixeles de la imagen para dibujar la trayectoria de la recta.

Ante las limitaciones de procesar toda la imagen en la raspberry y entendiendo el algoritmo que representa el BOF, se propone utilizar un sistema de referencia cartesiano. La idea de utilizar el sistema de referencia es trabajar con menos datos, ya que, solo se analizarán pixeles puntuales, el sistema toma como referencia el centroide de la figura como nuestra coordenada de origen, mientras que el movimiento de pixeles desde el centroide a lo largo de las columnas de la imagen harán referencia a nuestro eje X, mientras que el movimiento del centroide a lo largo de los renglones de la imagen serán nuestro eje Y. De igual manera representaremos nuestro eje X como el inicio de grado 0° para el análisis de los puntos del BOF, así únicamente mediante el cálculo de ángulos y distancias podremos obtener los puntos de interés para nuestro vector descriptor (ilustración 80).

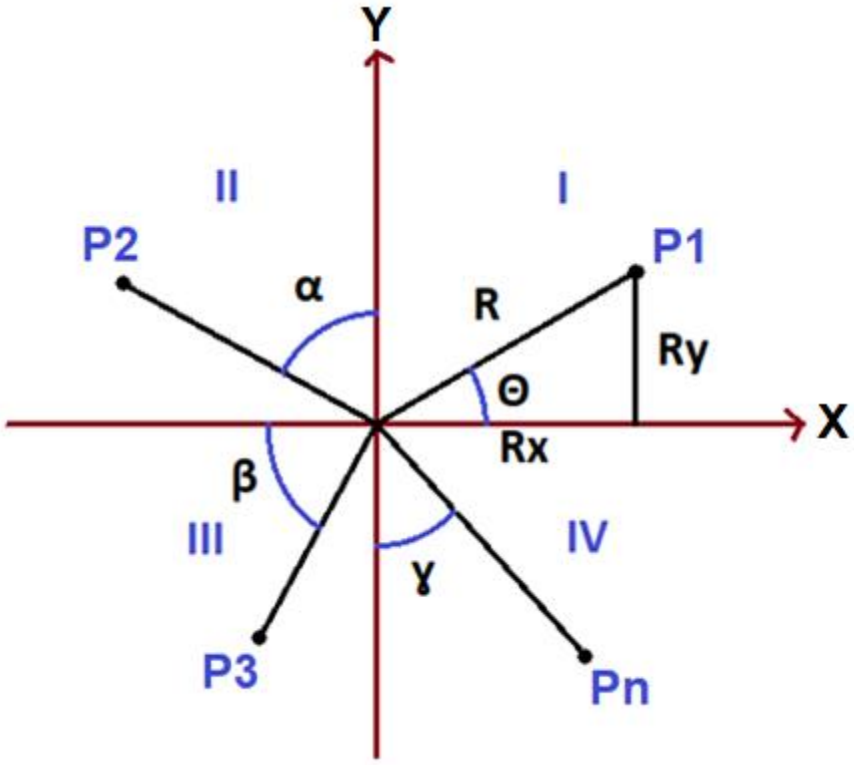


Ilustración 80 Obtención de puntos del BOF de acuerdo a su ángulo.

Mediante el siguiente algoritmo, dividimos a los cuadrantes tomando como referencia a nuestro centroide y dependiendo del cuadrante en que se ubique el punto de interés, se agrega una compensación de grados para poder distinguir cada distancia de acuerdo a su ángulo (ilustración 81).

```
# Matriz de contorno__Distancia centroide
for i in range(3, reng - 3):
    for j in range(3, col - 3):
        if imageCont_Aux[i, j] < 7:
            if imageCont_Aux[i, j] > 3:
                if imageCont_Aux[i, j]:
                    imageCont[i, j] = 1
                    if i < centrox and j < centroy:
                        Xref = centrox - i
                        Yref = centroy - j
                        ang = 180 - round(((math.atan(Yref / Xref) * 180) / 3.1416))
                    if i > centrox and j < centroy:
                            Xref = i - centrox
                            Yref = centroy - j
                            ang = round(((math.atan(Yref / Xref) * 180) / 3.1416))
                    if i < centrox and j > centroy:
                                Xref = centrox - i
                                Yref = j - centroy
                                ang = 180 + round(((math.atan(Yref / Xref) * 180) / 3.1416))
                    if i > centrox and j > centroy:
                                    Xref = i - centrox
                                    Yref = j - centroy
                                    ang = 360 - round(((math.atan(Yref / Xref) * 180) / 3.1416))
                    if i == centrox:
                        if j < centroy:
                            ang = 90
                        else:
                            ang = 270
                    if j == centroy:
                        if i < centrox:
                            ang = 180
                        else:
                            ang = 0
                    moduloVector = round(pow((Xref ** 2) + (Yref ** 2), 0.5))
                    for grados in range(24):
                        if ang / 15 == grados:
                            vectorGrados[grados] = moduloVector
                            """corX[grados] = i
                            corY[grados] = j"""
            else:
                imageCont[i, j] = 0
##########
```

Ilustración 81 Código para la obtención del BOF.

De acuerdo a la cantidad de puntos que queramos analizar será el tamaño de nuestro vector descriptor, tomando en cuenta que un mayor número de puntos representaría de una mejor manera a nuestra figura. El vector descriptor estará acomodado de acuerdo a su ángulo en orden creciente con la finalidad de poder conocer la rotación de pudiera tener la figura.

$$VecDescri[n] = [D1_{0°}, \quad D2_{0°+n}, \quad D3_{0°+2n}, \dots, Dm_{0°+Kn}]$$

Reconocimiento de la forma

Para el reconocimiento de las figuras a partir del vector descriptor, será necesario tomar en consideración que el realizar un proceso que involucre una red neuronal que comúnmente se utiliza para resolver este tipo de problemas resulta complicado, ya que el uso de este tipo de instrumentos requiere una gran prestación de recursos computacionales, así que será necesario otras alternativas.

Tomando en cuenta que el vector descriptor es independiente a la rotación, puesto que, para cualquier ángulo de inclinación en que se encuentre la figura las distancias del centroide al contorno estarán representadas en el mismo vector descriptor, pero en una posición diferente. Conociendo esta propiedad de nuestro algoritmo podemos proponer el uso de una base de datos, en donde tengamos un vector de referencia de las imágenes a analizar y podamos comparar un vector obtenido con estas mismas comparando cada una de sus posiciones (ilustración 82).

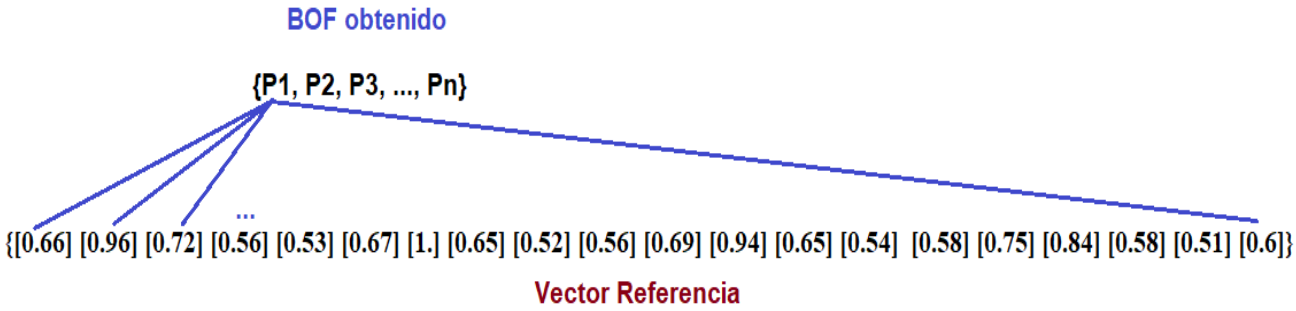


Ilustración 82 Identificación de BOF resultante con base de datos.

Una manera para lograr que el vector sea independiente a la escala de la figura será normalizándolo, para que mismas figuras de diferentes tamaños tengan el mismo vector. Esto se realiza tomando el mayor valor del vector que será un 1 en un nuevo vector y haciendo la comparación de todos los demás valores para obtener su valor correspondiente (ilustración 83).

```
############
#Normalizar
Nmayor = vectorGrados[0]
for n in range(24):
    if vectorGrados[n] > Nmayor:
        Nmayor = vectorGrados[n]
for n in range(24):
    vectorGrados[n] = round(vectorGrados[n] / Nmayor, 2)
    if vectorGrados[n] > 0.9:
        puntos = puntos + 1
```

Ilustración 83 Algoritmo para normalización del BOF.

Al obtener el vector característico de referencia ya normalizado podremos realizar un análisis de los próximos vectores comparando posición por posición para encontrar las similitudes en las distancias.

BOF_Cuadrado = [0.99 0.83 0.75 0.73 0.76 0.85 0.98 0.81 0.72 0.7 0.72 0.8 0.95 0.82 0.74 0.72 0.75 0.85 1.0 0.83 0.74 0.7 0.72 0.81]

BOF_Estrella = [0.66 0.96 0.72 0.56 0.53 0.67 1.0 0.65 0.52 0.56 0.69 0.94 0.65 0.54 0.58 0.75 0.84 0.58 0.51 0.63 0.91 0.75 0.55 0.52]

BOF_Cruz = [1.0 0.95 0.97 0.68 0.55 0.62 0.99 0.96 0.97 0.63 0.51 0.64 0.95 0.93 0.99 0.73 0.51 0.59 1.0 0.96 0.99 0.66 0.5 0.64]

BO_Triángulo = [0.58 0.55 0.56 0.63 0.76 1.0 0.81 0.65 0.58 0.55 0.56 0.62 0.84 0.96 0.83 0.66 0.59 0.56 0.57 0.62 0.74 0.99 0.83 0.65]

BOF_Círculo = [0.94 0.91 0.91 0.94 0.99 1.0 0.98 0.96 0.95 0.95 0.91 0.91 0.96 0.96 0.97 0.94 0.96 0.98 0.98 0.97 0.97 0.96 0.96 0.95]

Red de trabajo

El robot industrial **KUKA se encuentra integrado en una red LAN,** esto con el fin de integran varios equipos de trabajos o también conocidos como módulos, para enlazar una comunicación y poder realizar diferentes procesos con un mismo robot.

Nos integraremos a la red del robot mediante el protocolo TCP/IP, tomando un rango de nuestra IP entre: 192.168.0.0 a 192.168.0.255, 172.16.0 a 172.16.255.255, 172.31.0 a 172.31.255.255, esto de acuerdo al manual del robot (ilustración 84) [39].

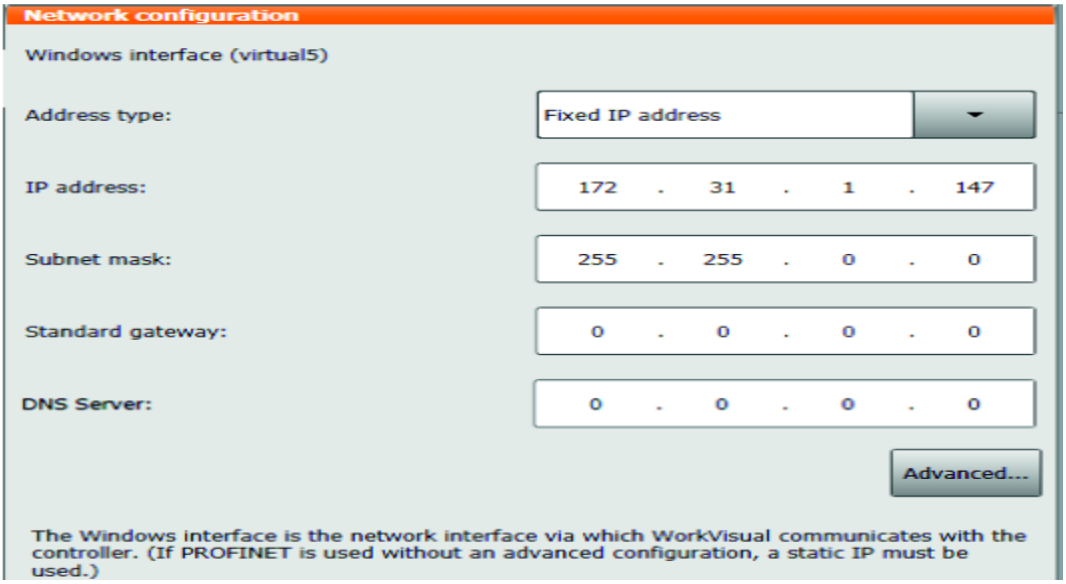


Ilustración 84 Dirección de conexión KUKA.

Comunicación del robot

Una vez obtenidas las calibraciones el robot y éste sea capaz de reconocer el tamaño del instrumental y el tamaño del área donde trabajaremos, lo siguiente será realizar la comunicación de nuestra raspberry con el robot para tomar el control de los movimientos del brazo robótico.

En primera instancia, tendremos que estar conectados a la red LAN donde se encuentra el robot mediante el protocolo TCP/IP. Ingresaremos a esta red con la dirección previamente definida que es 172.31.2.52, esto mediante la configuración de ficheros dentro de la raspberry pi (ilustración 85).

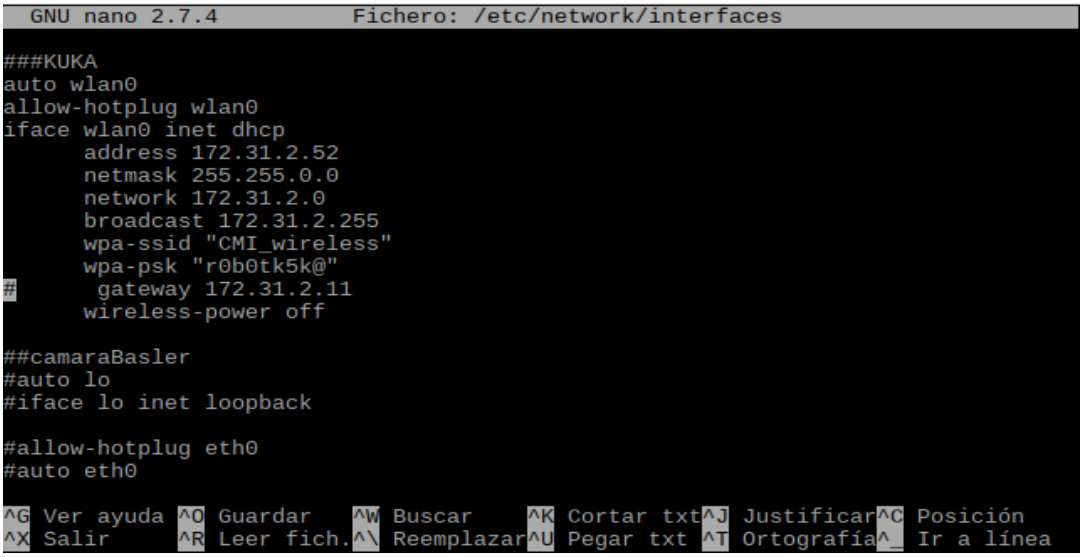


Ilustración 85 Modificación de fichero en raspberry pi 3 para conexión.

Para lograr la comunicación del robot mediante la raspberry será necesario abrir archivos del sistema KCR4, siendo más precisos se modificará el programa config.dat (ilustración 86) que se encuentra disponible al iniciar como usuario experto. Este programa contiene variables globales de la programación KUKA, es decir, contiene todas aquellas palabras reservadas de la programación del robot para hacer uso de los movimientos de los motores, aumentar o reducir la velocidad, leer las posiciones de los ejes, etc.

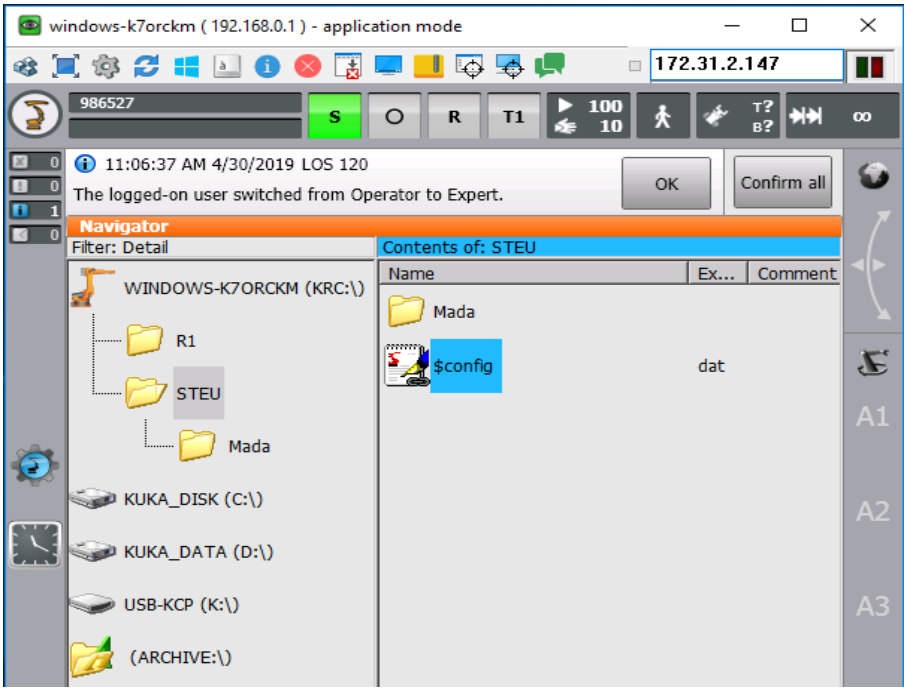


Ilustración 86 Archivo con variables globales.

En este archivo crearemos una **variable global** que pueda comunicarse con la raspberry y pueda modificar valores del robot, será una variable que funcionará como un enlace entre nuestro sistema embebido y el robot KUKA. El robot KUKA al tener diferentes tipos de movimiento PTP, LIN, CIRC, también cuenta con variables que se definen de acuerdo a las aplicaciones que se quieran programar en el robot, entre las más utilizadas se encuentra el definir a la variable para acceder a cada eje de manera individual (movimiento de cada uno de los motores individualmente) o definirla para controlar al robot en un sistema de coordenadas cartesianas. En nuestro caso al calibrar una base como nuestra área de trabajo definiremos estas variables “UBICACIÓN”, “BOF” y “ESCANEO” para que trabajen en un sistema coordenadas cartesianas, entonces, los parámetros que deberá leer esta variable serán las posiciones de los ejes X, Y y Z, así como también las posiciones del ángulo de inclinación de nuestra herramienta con respecto a nuestra área de trabajo que se representarán como A, B y C (ilustración 87).

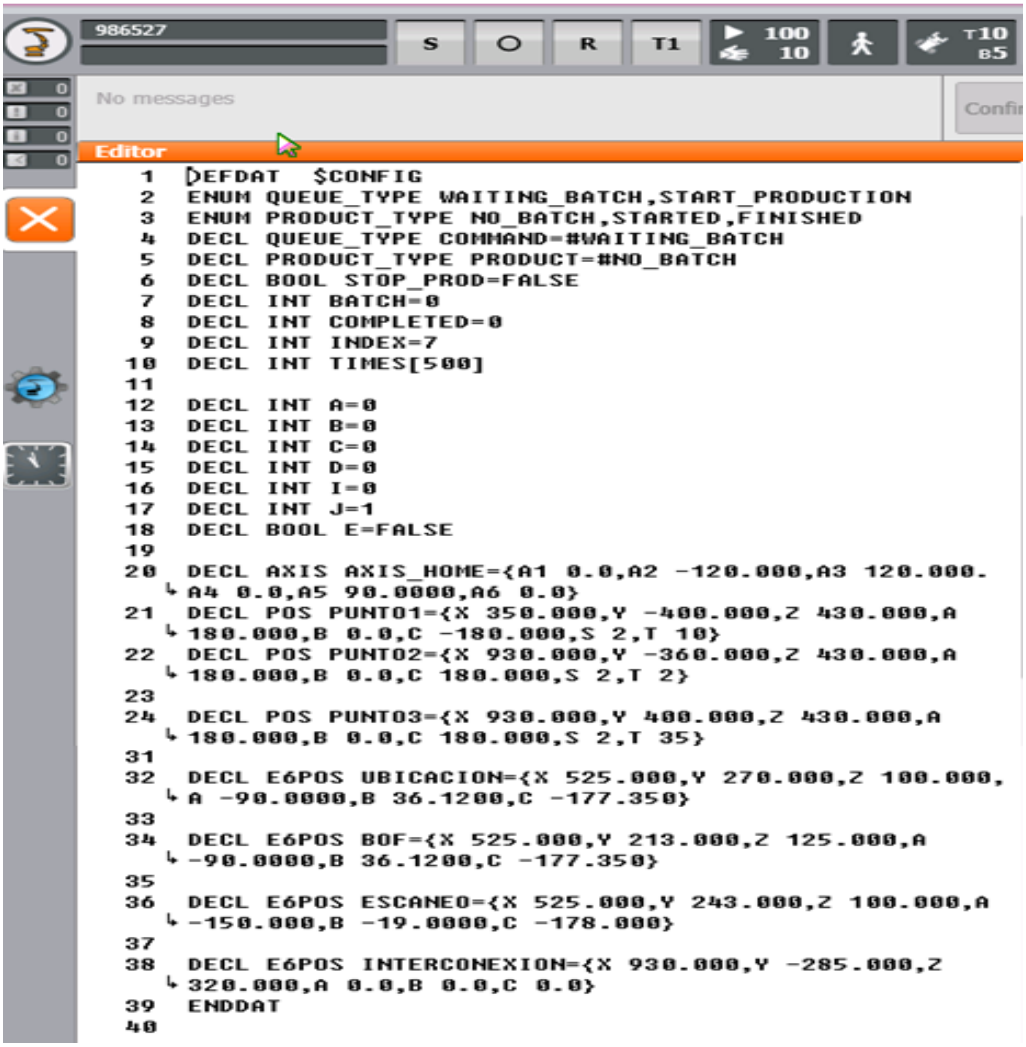


Ilustración 87 Formato de la variable global a utilizar.

Una vez definidas estas variables como variables de enlace se utilizará el modo de trabajo automático del robot KUKA, en el cual el sistema KCR4 libera todos los modos de seguridad del robot, con la finalidad de que una vez iniciado el programa el robot puede trabajar sin la necesidad de un operador. Para lograr que el robot lea las variables de enlace que manda la raspberry, realizaremos un programa en el cual definamos la herramienta y el área de trabajo calibrados haciendo referencia a las variables creadas en el fichero config.dat (ilustración 88).

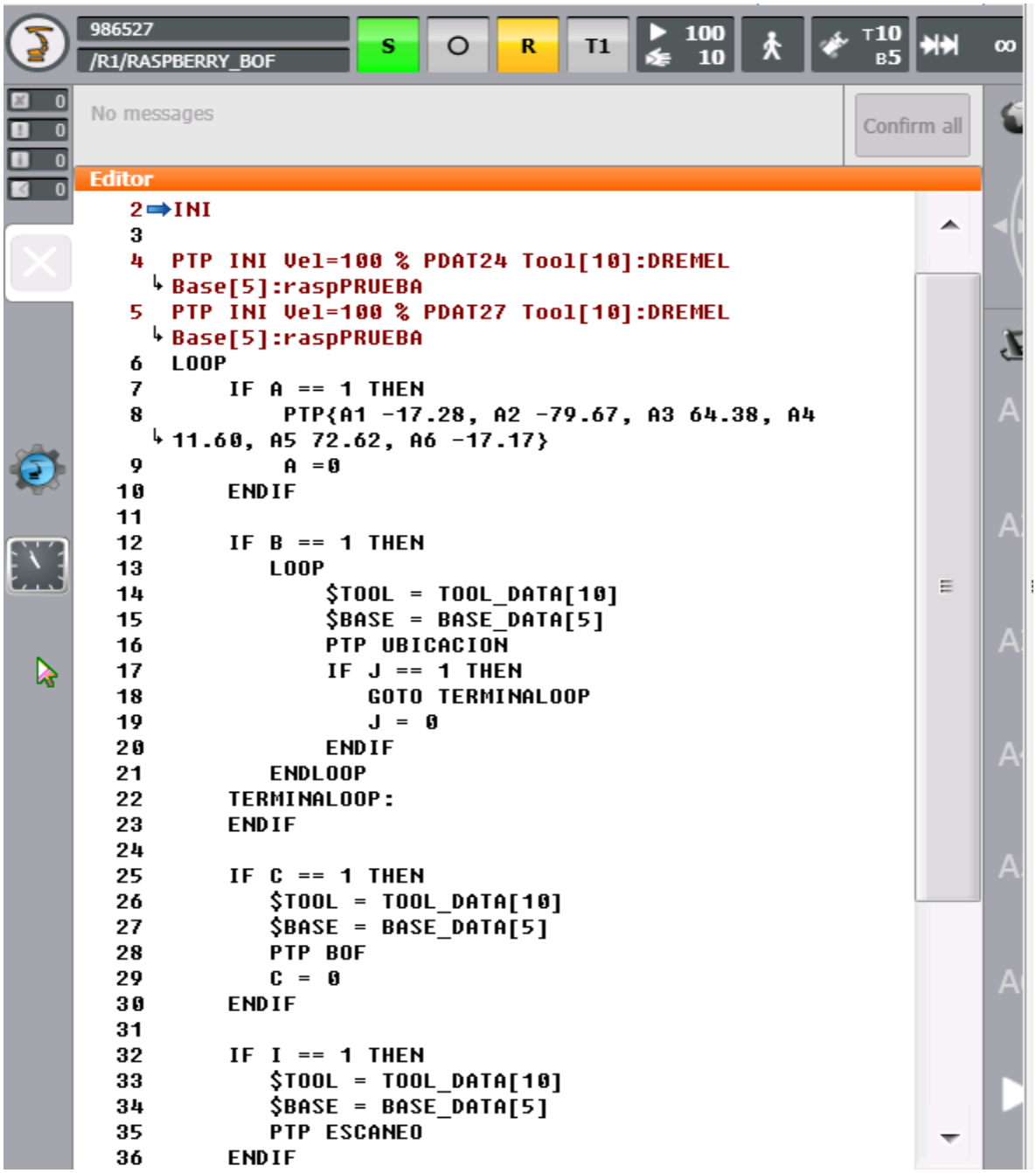


Ilustración 88 Programa de ejecución en KUKA para la lectura de la variable global.

En primera instancia el programa inicia con un movimiento PTP guardado previamente, que sirve como punto de partida del robot. En este programa asignamos la herramienta y la base calibrada para que el robot tenga una referencia de movimiento y en un ciclo leemos la información que contienen las variables de enlace “UBICACIÓN”, “BOF” y “ESCANEO”.

En nuestro programa de la raspberry pi tendremos que pasar cada uno de los parámetros que se definieron las variables de enlace, que serían las posiciones de los ejes, así como el ángulo de inclinación de la herramienta (ilustración 89).

```
elif opcion == 4:
    robot.write("I", 1)
    #Escaneo
    robot.write("ESCANEO.X",dxS)
    robot.write("ESCANEO.Y",dyS+30)
    robot.write("ESCANEO.Z",100)
    robot.write("ESCANEO.A",-90)
    robot.write("ESCANEO.B",35)
    robot.write("ESCANEO.C",-178)
    sleep(2)
    """robot.write("ESCANEO.X",dxS)
    robot.write("ESCANEO.Y",dyS+30)
    robot.write("ESCANEO.Z",100)
    robot.write("ESCANEO.A",-90)"""
    robot.write("ESCANEO.B",-19)
    """robot.write("ESCANEO.C",-178)"""
    sleep(1)
    os.system("fswebcam -r 640x480 --no-banner /home/pi/BOF_Proyecto/ESC_100.jpg")
    robot.write("ESCANEO.A", -100)
    sleep(1)
    os.system("fswebcam -r 640x480 --no-banner /home/pi/BOF_Proyecto/ESC_110.jpg")
    robot.write("ESCANEO.A", -110)
    sleep(1)
    os.system("fswebcam -r 640x480 --no-banner /home/pi/BOF_Proyecto/ESC_120.jpg")
    robot.write("ESCANEO.A", -120)
    sleep(1)
    os.system("fswebcam -r 640x480 --no-banner /home/pi/BOF_Proyecto/ESC_130.jpg")
    robot.write("ESCANEO.A", -130)
    sleep(1)
    os.system("fswebcam -r 640x480 --no-banner /home/pi/BOF_Proyecto/ESC_140.jpg")
    robot.write("ESCANEO.A", -140)
    sleep(1)
    os.system("fswebcam -r 640x480 --no-banner /home/pi/BOF_Proyecto/ESC_150.jpg")
    robot.write("ESCANEO.A", -150)
    robot.write("I", 0)
elif opcion == 5:
    print ("Regresar HOME....")
    robot.write("D",1)
```

Ilustración 89 Algoritmo para la actualización de la variable local KUKA desde Raspberry.

Para tener acceso a la variable de enlace en la raspberry, será necesario contar con la librería “KUKAVARPROXY”. Esta librería realiza la conexión entre nuestro sistema embebido y la variable de enlace del KUKA utilizando el protocolo de comunicación TCP/IP con una dirección “172.31.2.147”.

Ubicación de objeto

Para realizar la ubicación de objetos en nuestra área de trabajo se usará la base de trabajo como un sistema de coordenadas cartesianas. Este sistema de coordenadas se adapta a la calibración “BASE” con la que cuenta el robot KUKA, que es el mismo con el área de trabajo que se calibró previamente. Esto quiere decir que el mismo punto de origen de la calibración será la coordenada (0, 0) del sistema de referencia y los ejes X y Y serán los alcances máximos que se validaron al calibrar el área de trabajo (ilustración 90).

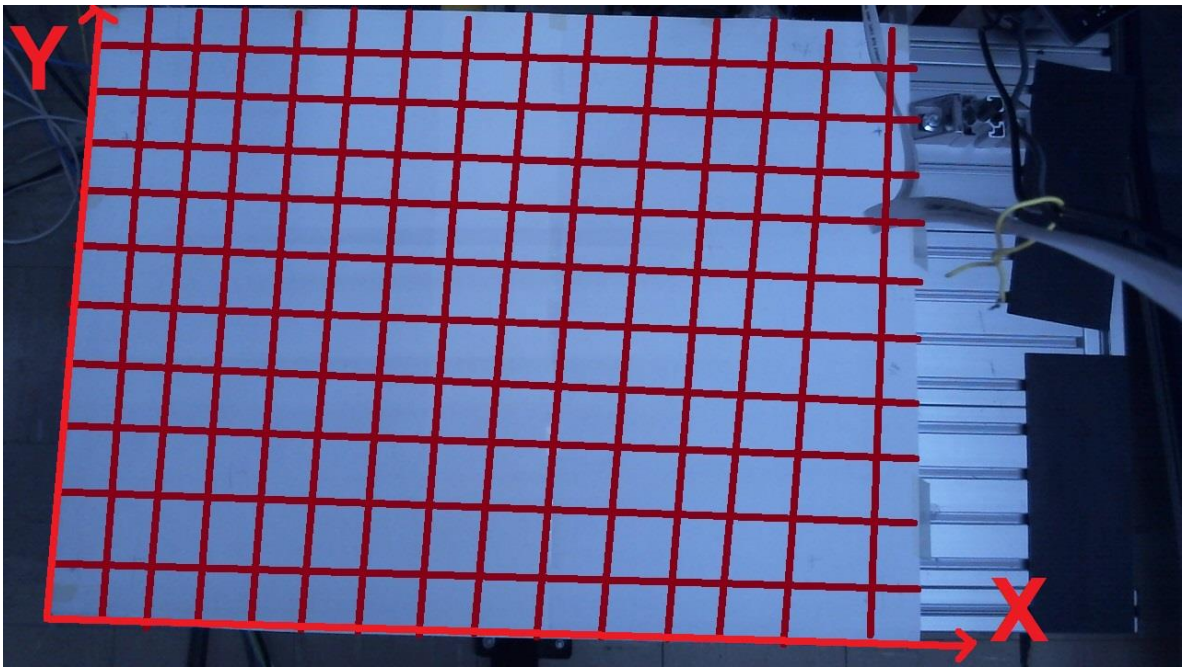


Ilustración 90 División de base de trabajo como un sistema de coordenadas cartesianas.

Lo siguiente a realizar será el proceso de transformar una distancia digital a su correspondiente distancia real. Para conocer las coordenadas de la figura que corresponden al área de trabajo se tomará como referencia la resolución de la imagen, el centroide de la misma y el tamaño de la base.

$$Dx = \frac{(Distancia\ X\ base * Centroide\ X)}{Resolución\ imagen\ X}$$

$$Dy = \frac{(Distancia\ Y\ base * Centroide\ Y)}{Resolución\ imagen\ Y}$$

La idea es conocer la relación de un pixel con la distancia de la base de trabajo, es decir, la distancia que le corresponde el movimiento de un pixel a nuestra área de

visión de la imagen. En el caso de obtener una imagen de 1280x960 pixeles y conociendo las medidas de nuestra área de trabajo de 70x40 [cm] tendríamos que el movimiento de un pixel en el eje X equivaldría a 0.54 [mm] y en el eje Y a 0.41 [mm]. Se debe tener en cuenta que entre mejor sea la calibración del robot la precisión con la que ubique el centroide de la figura será mejor.

Como se mencionó la calibración de la herramienta nos proporciona las variables necesarias para lograr realizar la rotación de la herramienta respecto a los ejes (ilustración 91), esta misma información tendrá que ser enviada en las variables de enlace.

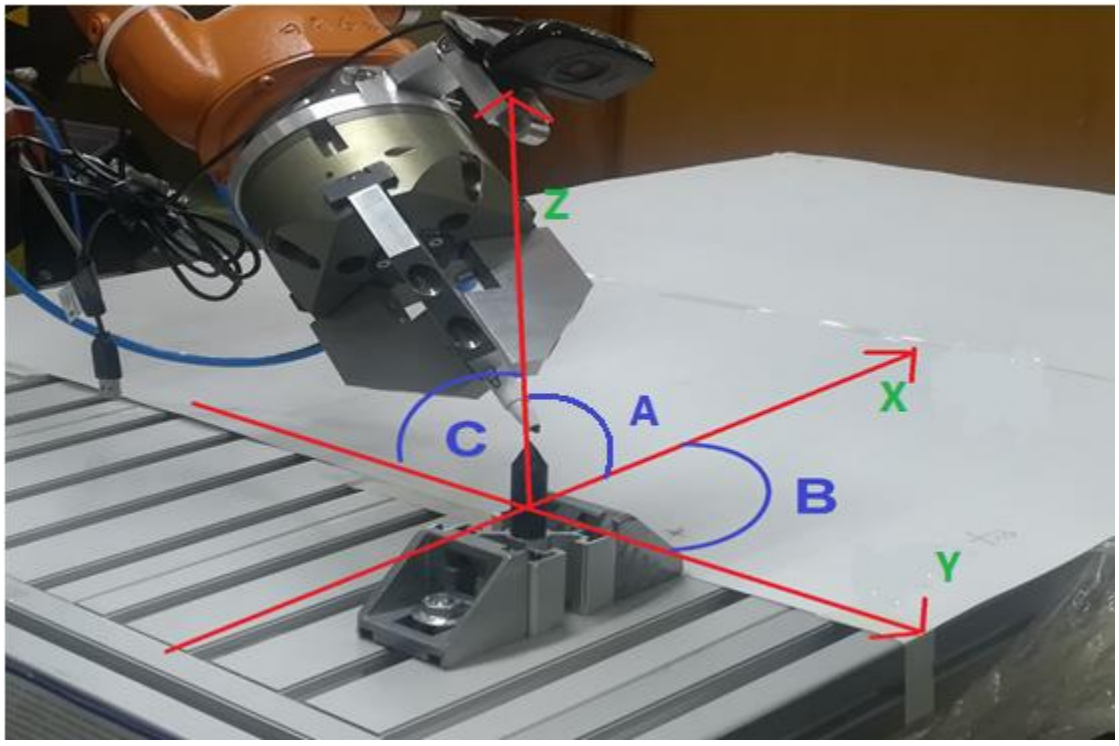


Ilustración 91 Referencia de ángulos del robot KUKA y la variable "ESCANEO".

Capítulo 4

Resultados experimentales

Calibración mundo digital a mundo físico

La primera prueba a realizar es la precisión con la que nuestro código genera las coordenadas del centroide de la figura, comparándolo con el mundo físico, es decir, el porcentaje de error de la calibración cámara-área de trabajo (ilustración 92). La resolución de la imagen con la que se obtiene el área de trabajo es de 1280x960 pixeles y el área de trabajo es de 70x40 [cm].

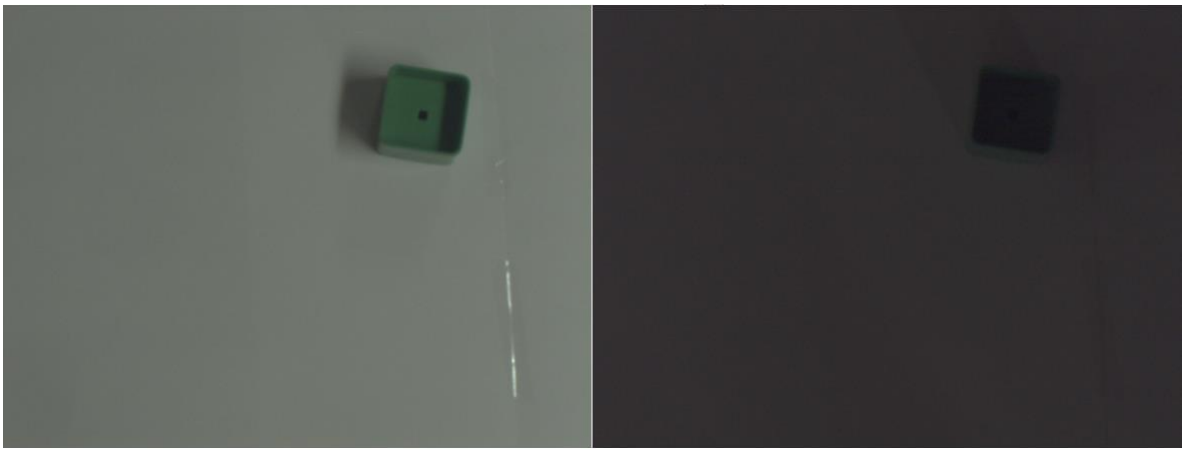


Ilustración 92 Foto de un cubo tomada con dos diferentes fuentes de iluminación.

En la tabla 7, se comprueba el porcentaje de error obtenido del centroide con una buena iluminación.

N° Prueba	*Centroide real (x, y) [cm]*	*Centroide calculado (i, j) [cm]*	*% Error*
Prueba 1	(32, 30)	(31.998, 30.008)	(0.006, 0.026)
Prueba 2	(40, 12)	(39.997, 12.005)	(0.007, 0.041)
Prueba 3	(65, 14)	(64.999, 14.009)	(0.001, 0.064)
Prueba 4	(10, 35)	(9.999, 35.009)	(0.01, 0.025)
Prueba 5	(10, 20)	(9.999, 20.001)	(0.01, 0.005)
Prueba 6	(22, 22)	(21.997, 22.006)	(0.013, 0.027)
Prueba 7	(33, 17)	(32.998, 17.005)	(0.006, 0.029)
Prueba 8	(65, 5)	(64.990, 5.003)	(0.015, 0.06)
Prueba 9	(60, 32)	(59.992, 32)	(0.013, 0)
Prueba 10	(15, 27)	(14.996, 27.004)	(0.026, 0.014)

Tabla 7 Porcentaje de error en la detección del centroide con una buena fuente de iluminación.

Se realiza una segunda prueba (tabla 8) de ubicación de objeto, con un ambiente de iluminación más escasa para comprobar el contraste que nos proporciona el ecualizar la imagen.

N° Prueba	*Centroide real (x, y) [cm]*	*Centroide calculado (i, j) [cm]*	*% Error*
Prueba 1	(32, 30)	(31.730, 30.200)	(0.84, 0.66)
Prueba 2	(40, 12)	(39.772, 12.197)	(0.57, 1.64)
Prueba 3	(65, 14)	(64.671, 14.187)	(0.5, 1.33)
Prueba 4	(10, 35)	(9.807, 35.260)	(1.93, 0.74)
Prueba 5	(10, 20)	(9.869, 20.274)	(1.31, 1.37)
Prueba 6	(22, 22)	(21.800, 22.300)	(0.9, 1.36)
Prueba 7	(33, 17)	(32.791, 17.243)	(0.63, 1.42)
Prueba 8	(65, 5)	(64.812, 5.101)	(0.28, 2.02)
Prueba 9	(60, 32)	(59.773, 32.298)	(0.37, 0.93)
Prueba 10	(15, 27)	(14.897, 27.231)	(0.68, 0.85)

Tabla 8 Porcentaje de error en la detección del centroide con una escasa fuente de iluminación.

La siguiente prueba a realizar es para encontrar el error que proporciona el robot KUKA al ubicar el centroide de trabajo, es decir, el error que tiene el robot en cuanto a la ubicación del centro del herramental con el centroide en el espacio de trabajo (ilustración 93).

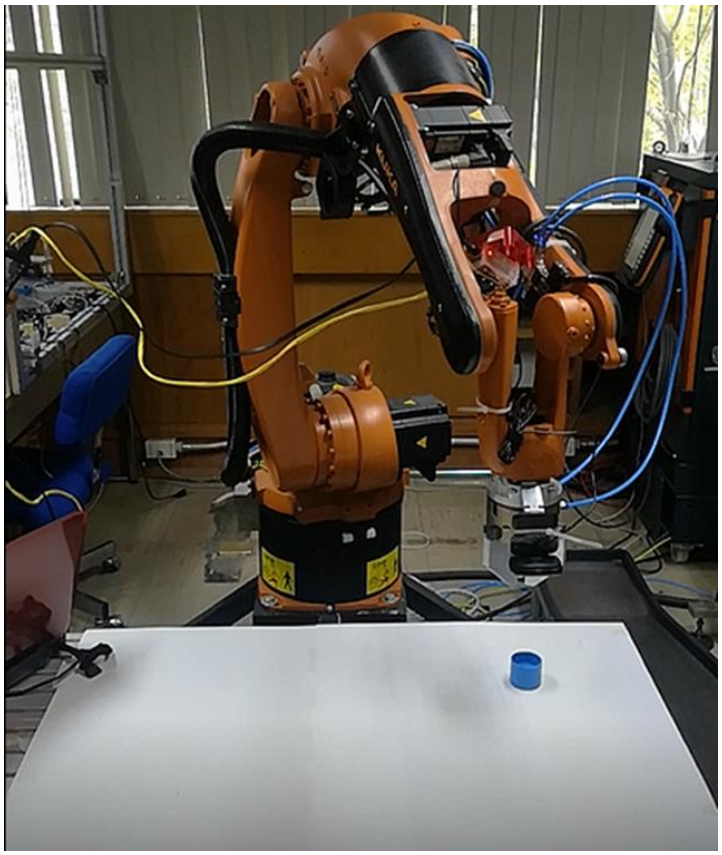


Ilustración 93 Ubicación de figura en el área de trabajo.

En la siguiente tabla 9, se comparan puntos propuestos en nuestra área de trabajo con los puntos en las que se ubica el robot después del procesamiento de la imagen.

Punto	*Punto propuesto (x, y) [cm]*	*Ubicación obtenida (x, y) [cm]*	*Error (x, y)%*
1	(5, 2)	(5.2, 2.2)	(4, 10)
2	(10, 4)	(10.2, 4.2)	(2, 5)
3	(15, 6)	(15.1, 6.2)	(0.6, 3.3)
4	(20, 12)	(20.1, 12.1)	(0.5, 0.8)
5	(25, 16)	(25.1, 16.2)	(0.4, 1.2)
6	(30, 20)	(30, 20.1)	(0, 0.5)
7	(40, 24)	(40.1, 24.1)	(0.2, 0.4)
8	(50, 28)	(50.2, 28.2)	(0.4, 0.7)
9	(60, 34)	(60.2, 34.2)	(0.3, 0.5)
10	(70, 40)	(70.3, 40.2)	(0.4, 0.5)

Tabla 9 Porcentaje de error entre los puntos propuestos y el obtenido con el robot KUKA.

Obtención de BOF

Obtenido los errores referentes a la calibración del robot con la ubicación del centroide de la figura, se realizan pruebas para corroborar el error del vector característico. Se tienen 5 figuras para analizar un cuadrado, una estrella, una cruz, un triángulo y finalmente un círculo, el vector descriptor generado cuenta con 24 puntos para su análisis.

BOF Cuadrado.

Se toma una foto con una resolución de 640x480 pixeles, con el cual se realiza el proceso de transformar la imagen a color a su correspondiente escala de grises, realizando la ecualización par finalmente obtener el contorno de la figura, este caso un cuadrado (ilustración 95).

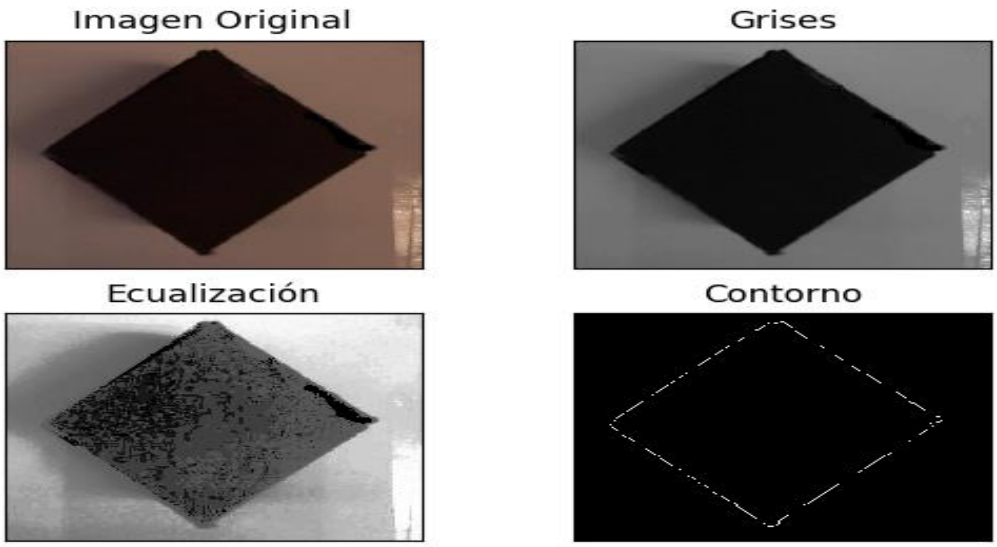


Ilustración 94 Procesamiento de imagen de un cubo

Posteriormente, a partir del contorno de la imagen obtenernos nuestro vector de referencia del cuadrado (ilustración 95).

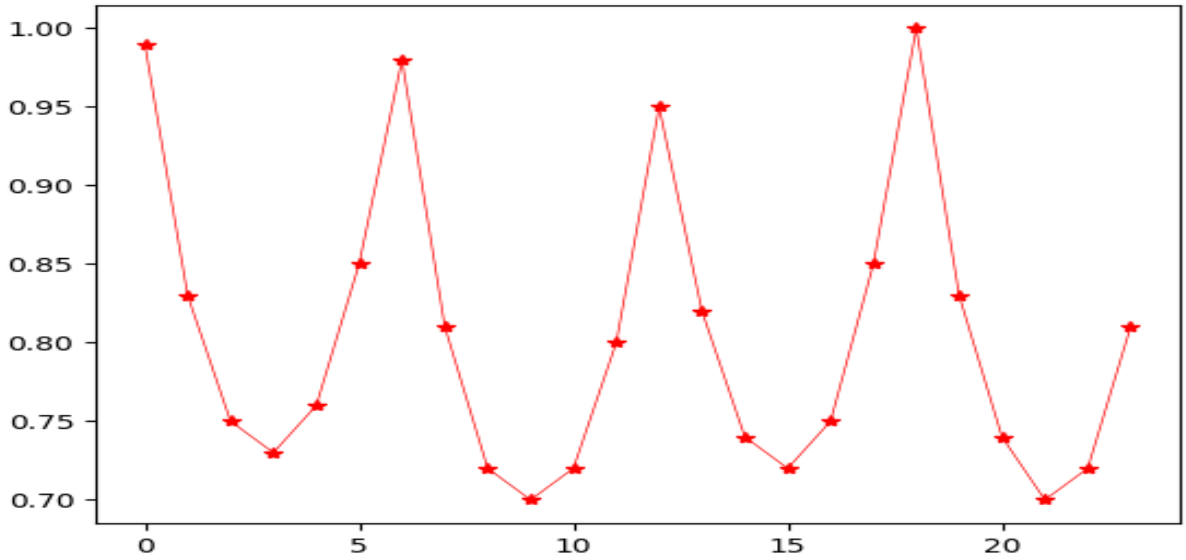


Ilustración 95 BOF de referencia de un cuadrado.

El siguiente proceso a realizar es obtener diferentes BOF de cuadrados, para comprobar la repetitividad con la que obtenemos los vectores descriptivos (tabla 10).

Grados [°]	*BOF Referencia*	*BOF 1*	*BOF 2*	*BOF 3*	*BOF 4*	*BOF 5*	*BOF 6*	*BOF 7*	*BOF 8*	*BOF 9*	*BOF 10*	*ERROR MAX [%]*
0	0.99	1	0.99	1	1	0.99	0.99	0.99	1	0.99	0.99	1
15	0.83	0.83	0.84	0.88	0.83	0.83	0.85	0.83	0.84	0.83	0.85	6
30	0.75	0.76	0.77	0.72	0.75	0.76	0.77	0.75	0.77	0.75	0.74	2.6
45	0.73	0.74	0.75	0.73	0.73	0.7	0.76	0.77	0.73	0.74	0.74	10
60	0.76	0.76	0.72	0.77	0.79	0.79	0.75	0.77	0.77	0.77	0.77	3.9
75	0.85	0.84	0.81	0.85	0.85	0.92	0.88	0.84	0.83	0.85	0.84	8.2
90	0.98	0.99	0.96	0.99	0.98	0.99	0.98	0.96	0.98	0.99	0.99	2
105	0.81	0.81	0.81	0.8	0.84	0.81	0.81	0.88	0.8	0.81	0.8	8.6
120	0.72	0.72	0.73	0.73	0.72	0.73	0.73	0.76	0.74	0.72	0.72	5.5
135	0.7	0.7	0.75	0.71	0.74	0.7	0.69	0.72	0.65	0.71	0.71	7.14
150	0.72	0.7	0.77	0.71	0.72	0.69	0.73	0.68	0.73	0.72	0.72	6.9
165	0.8	0.7	0.78	0.82	0.8	0.8	0.82	0.76	0.81	0.83	0.82	12.5
180	0.95	0.98	0.9	0.95	0.95	0.94	0.95	0.9	0.96	0.96	0.95	5.2
195	0.82	0.84	0.89	0.83	0.79	0.83	0.79	0.88	0.88	0.82	0.81	8.5
210	0.74	0.77	0.77	0.74	0.74	0.71	0.74	0.78	0.75	0.74	0.73	5.4
225	0.72	0.72	0.66	0.72	0.72	0.7	0.72	0.72	0.72	0.71	0.72	8.3
240	0.75	0.75	0.71	0.76	0.73	0.77	0.77	0.76	0.75	0.75	0.77	5.3

255	0.85	0.85	0.89	0.85	0.85	0.86	0.85	0.85	0.88	0.85	0.9	5.8
270	1	0.98	1	0.99	0.99	1	1	1	1	1	1	2
285	0.83	0.84	0.87	0.83	0.85	0.89	0.83	0.88	0.82	0.84	0.85	7.2
300	0.74	0.74	0.66	0.74	0.74	0.8	0.74	0.76	0.75	0.74	0.71	10.8
315	0.7	0.7	0.79	0.7	0.69	0.75	0.72	0.74	0.71	0.7	0.72	12.8
330	0.72	0.77	0.7	0.72	0.72	0.72	0.72	0.7	0.73	0.72	0.77	6.9
345	0.81	0.83	0.78	0.81	0.81	0.9	0.81	0.82	0.79	0.81	0.82	3.7

Tabla 10 Análisis del BOF de un cuadrado

En la ilustración 95, podemos observar el comportamiento de nuestro algoritmo para la obtención del BOF, se puede observar que la tendencia de los puntos es prácticamente la misma con un promedio de error máximo de 6.5%.

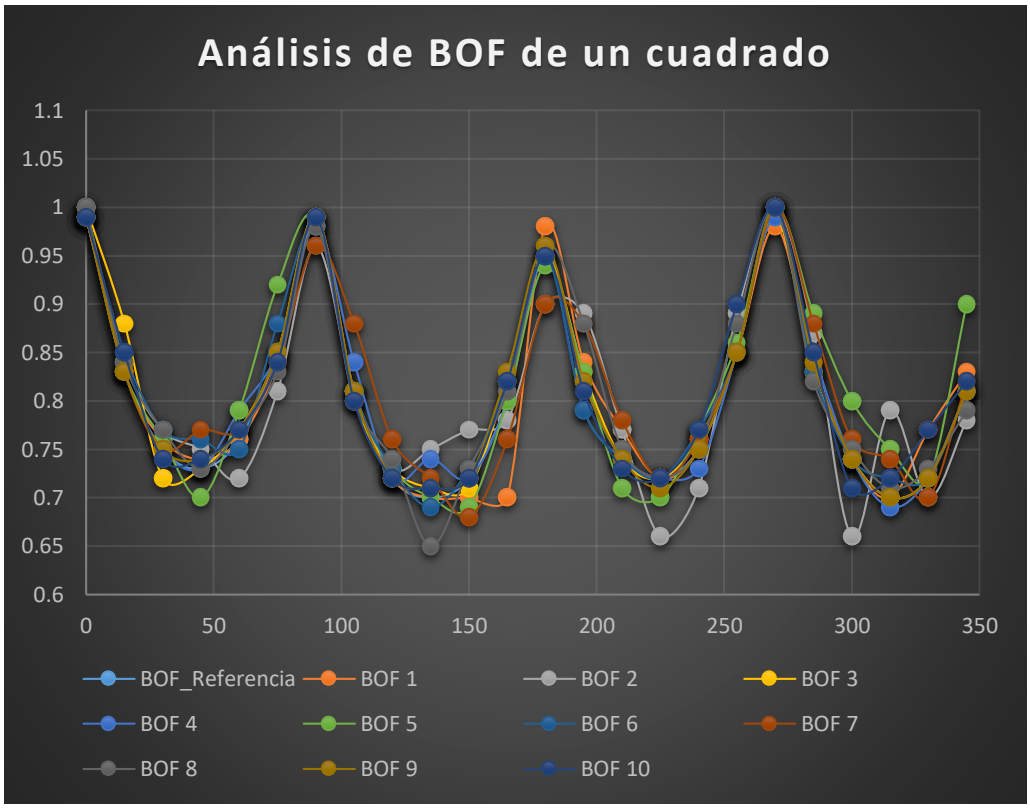


Ilustración 96 Gráficas de varias muestras del BOF de un cuadrado.

BOF Estrella.

Realizaremos el mismo proceso para cada una de las demás figuras (ilustración 97).

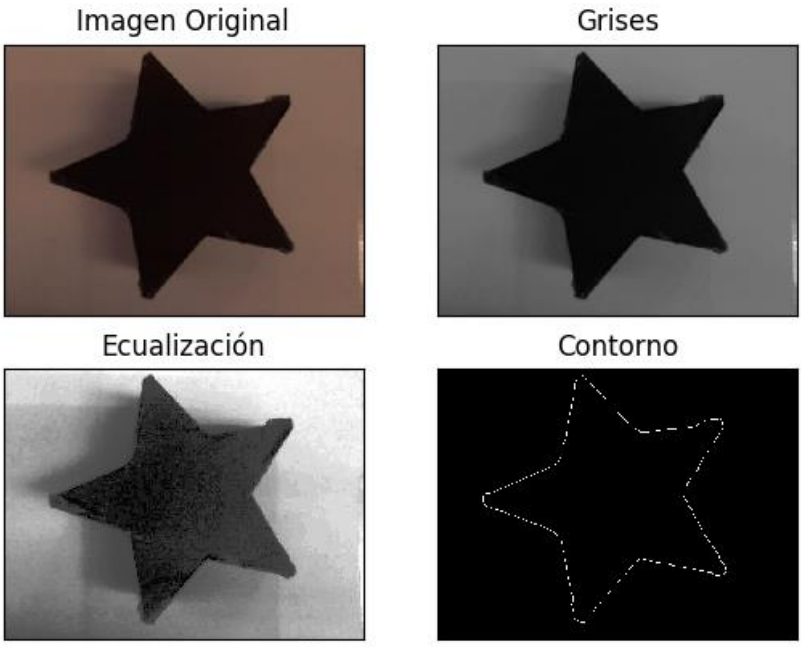


Ilustración 97 Procesamiento de imagen de una estrella.

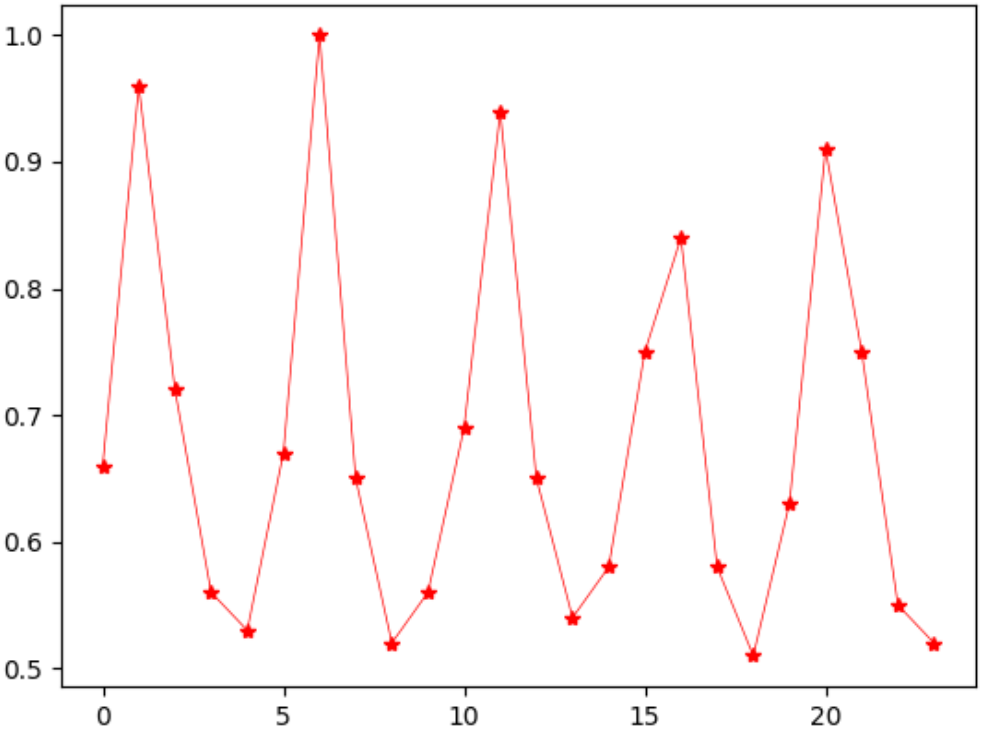


Ilustración 98 BOF de referencia de una estrella.

Obtenemos en la tabla 11 distintos vectores descriptores, para conocer el error del BOF de la figura de una estrella.

Grados	*BOF Referencia*	*BOF 1*	*BOF 2*	*BOF 3*	*BOF 4*	*BOF 5*	*BOF 6*	*BOF 7*	*BOF 8*	*BOF 9*	*BOF 10*	*Error MAX [%]*
0	0.66	0.67	0.66	0.69	0.65	0.66	0.7	0.68	0.68	0.66	0.67	6
15	0.96	1	0.97	0.92	1	0.96	0.9	0.96	0.96	1	0.96	4.1
30	0.72	0.7	0.7	0.72	0.7	0.69	0.75	0.72	0.75	0.72	0.7	4.1
45	0.56	0.58	0.56	0.55	0.58	0.58	0.58	0.55	0.58	0.59	0.55	5.3
60	0.53	0.57	0.54	0.53	0.53	0.54	0.53	0.51	0.5	0.55	0.53	7.5
75	0.67	0.68	0.68	0.66	0.65	0.66	0.65	0.68	0.69	0.66	0.69	2.9
90	1	0.92	1	1	0.99	1	1	1	1	0.95	1	8
105	0.65	0.65	0.64	0.67	0.69	0.65	0.66	0.64	0.63	0.66	0.64	6.15
120	0.52	0.52	0.51	0.57	0.5	0.51	0.53	0.52	0.52	0.53	0.54	9.6
135	0.56	0.55	0.54	0.52	0.54	0.59	0.56	0.58	0.54	0.57	0.58	7.1
150	0.69	0.68	0.69	0.71	0.67	0.66	0.69	0.69	0.7	0.72	0.69	2.8
165	0.94	0.99	0.95	0.91	0.97	0.97	0.94	0.92	0.96	0.91	0.94	5.3
180	0.65	0.62	0.65	0.63	0.64	0.63	0.65	0.67	0.62	0.66	0.65	3
195	0.54	0.49	0.55	0.55	0.56	0.51	0.54	0.54	0.53	0.52	0.52	9.2
210	0.58	0.59	0.52	0.58	0.6	0.61	0.6	0.56	0.6	0.58	0.55	10.3
225	0.75	0.75	0.77	0.79	0.79	0.73	0.73	0.75	0.73	0.77	0.75	5.3
240	0.84	0.84	0.82	0.79	0.79	0.81	0.84	0.84	0.86	0.81	0.84	5.9
255	0.58	0.58	0.57	0.61	0.56	0.6	0.58	0.59	0.58	0.61	0.59	5.1
270	0.51	0.52	0.51	0.51	0.51	0.49	0.51	0.51	0.52	0.52	0.51	3.9
285	0.63	0.62	0.63	0.6	0.65	0.59	0.63	0.62	0.63	0.61	0.64	6.3
300	0.91	0.9	0.91	0.95	0.88	0.92	0.95	0.92	0.92	0.9	0.89	3.2
315	0.75	0.75	0.74	0.74	0.77	0.74	0.73	0.71	0.77	0.77	0.77	5.3
330	0.55	0.56	0.55	0.6	0.55	0.54	0.55	0.56	0.55	0.54	0.56	1.8
345	0.52	0.52	0.53	0.5	0.49	0.53	0.52	0.55	0.5	0.55	0.54	5.7

Tabla 11 Análisis del BOF de una estrella.

De la misma manera obtenemos un comportamiento similar para todos los vectores con un error máximo promediado de 5.5%.

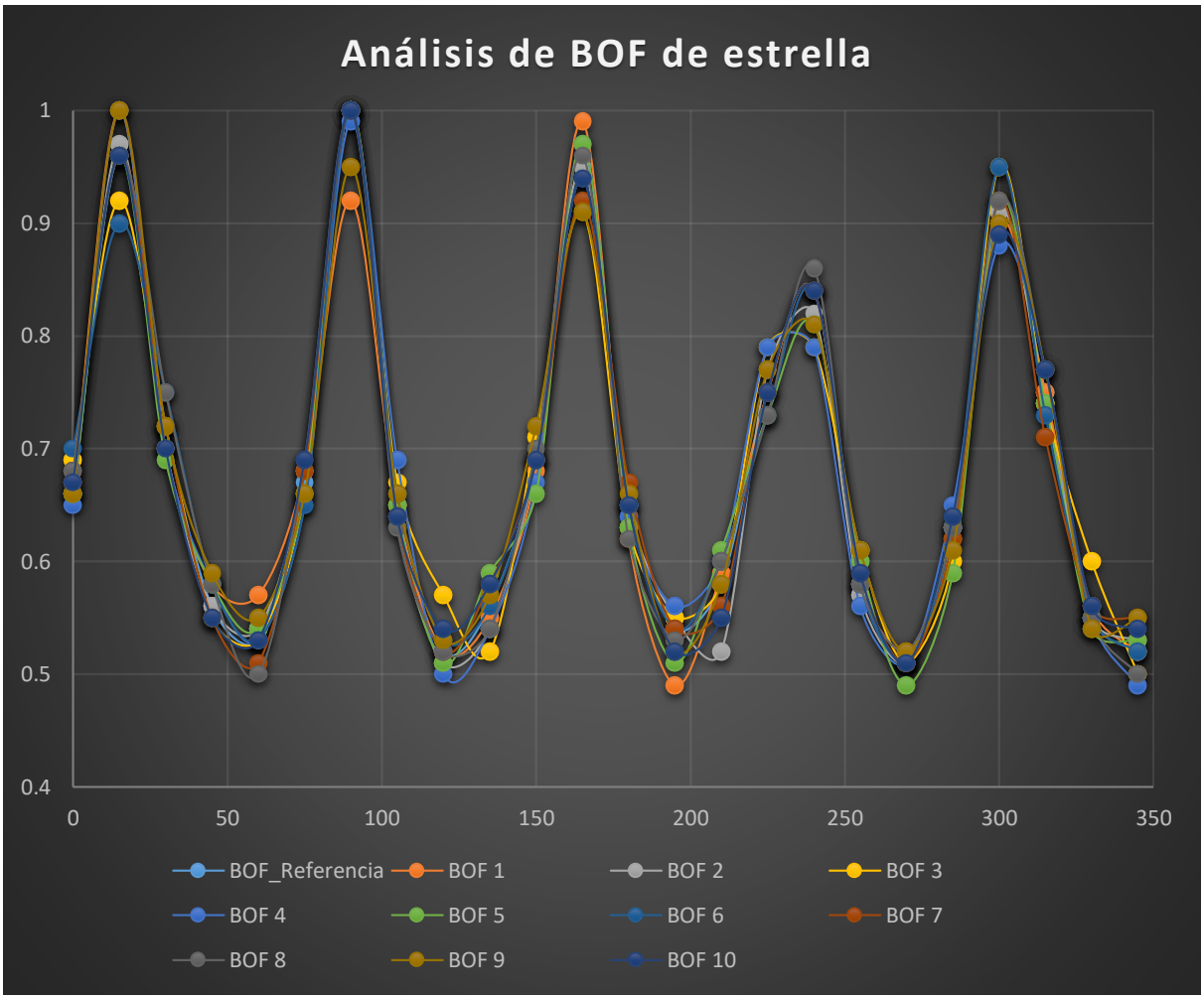


Ilustración 99 Gráficas de varios vectores de una estrella.

BOF Cruz

Obtenemos el procesamiento de nuestra figura llama cruz, encontrando su respectivo vector de referencia (ilustración 100, ilustración 101).

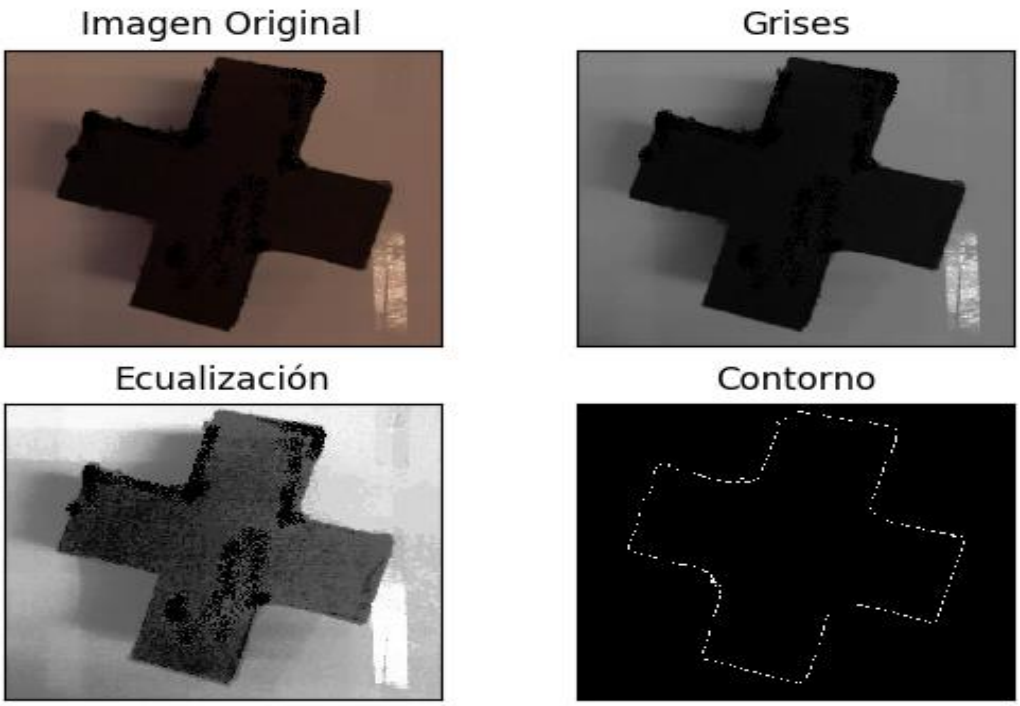


Ilustración 100 Procesamiento de imagen de una cruz.

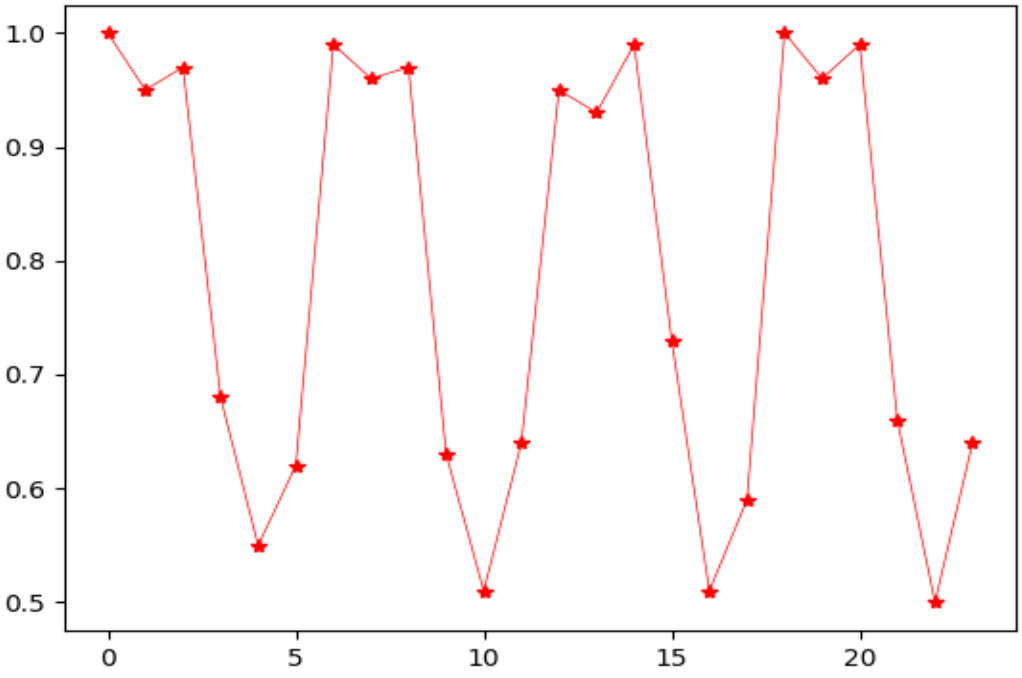


Ilustración 101 Vector de referencia de una cruz.

Realizamos el proceso de obtención del BOF varia veces para conocer el error del vector de la figura (tabla 12).

Grado Que representa	*BOF Referencia*	*BOF 1*	*BOF 2*	*BOF 3*	*BOF 4*	*BOF 5*	*BOF 6*	*BOF 7*	*BOF 8*	*BOF 9*	*BOF 10*	*Error MAX [%]*
0	1	1	1	0.99	1	0.99	1	1	0.98	1	1	2
15	0.95	0.96	0.94	0.95	0.91	0.98	0.92	0.97	0.97	0.96	0.95	4.2
30	0.97	0.96	0.98	0.98	0.92	0.96	0.97	0.99	0.99	0.95	0.94	5.1
45	0.68	0.67	0.7	0.65	0.63	0.69	0.68	0.69	0.71	0.69	0.67	7.3
60	0.55	0.53	0.58	0.54	0.57	0.51	0.54	0.56	0.57	0.51	0.57	7.2
75	0.62	0.66	0.65	0.66	0.6	0.59	0.62	0.62	0.62	0.63	0.62	4.8
90	0.99	0.95	0.92	0.97	0.96	0.99	0.99	0.98	0.95	0.99	0.97	7
105	0.96	0.92	0.91	0.97	0.94	0.96	0.98	0.96	0.94	0.96	0.98	5.2
120	0.97	0.96	0.99	0.95	0.98	0.97	0.99	0.97	0.98	0.97	0.99	2
135	0.63	0.63	0.65	0.64	0.6	0.63	0.64	0.64	0.63	0.63	0.63	4.7
150	0.51	0.51	0.54	0.49	0.54	0.51	0.51	0.51	0.49	0.54	0.53	5.8
165	0.64	0.66	0.61	0.63	0.65	0.64	0.64	0.6	0.64	0.69	0.64	7.8
180	0.95	0.95	0.92	0.94	0.95	0.95	0.94	0.91	0.94	0.91	0.96	4.2
195	0.93	0.89	0.94	0.96	0.91	0.92	0.93	0.93	0.93	0.93	0.94	4.3
210	0.99	0.95	0.97	0.97	0.98	0.99	0.99	0.98	0.97	0.99	0.97	4
225	0.73	0.72	0.71	0.74	0.71	0.73	0.73	0.73	0.73	0.73	0.73	2.7
240	0.51	0.49	0.55	0.51	0.52	0.51	0.49	0.52	0.51	0.52	0.51	3.9
255	0.59	0.59	0.57	0.59	0.61	0.6	0.59	0.62	0.59	0.57	0.61	3.3
270	1	0.98	1	1	1	1	0.96	1	1	1	1	4
285	0.96	0.97	0.95	0.96	0.9	0.96	0.96	0.96	0.96	0.94	0.97	6.2
300	0.99	0.98	0.96	0.99	0.94	0.99	0.99	0.97	0.97	0.99	0.99	5
315	0.66	0.65	0.69	0.72	0.64	0.66	0.65	0.66	0.66	0.66	0.68	9
330	0.5	0.48	0.52	0.53	0.49	0.5	0.5	0.49	0.5	0.56	0.5	12
345	0.64	0.65	0.63	0.61	0.66	0.64	0.64	0.63	0.64	0.66	0.66	4.6

Tabla 12 Análisis del BOF de una cruz.

El comportamiento del BOF tiene la misma tendencia para cualquier prueba (ilustración 102), de igual manera obtenemos que el mayor error encontrado en un punto ya promediado es del 5.5%.

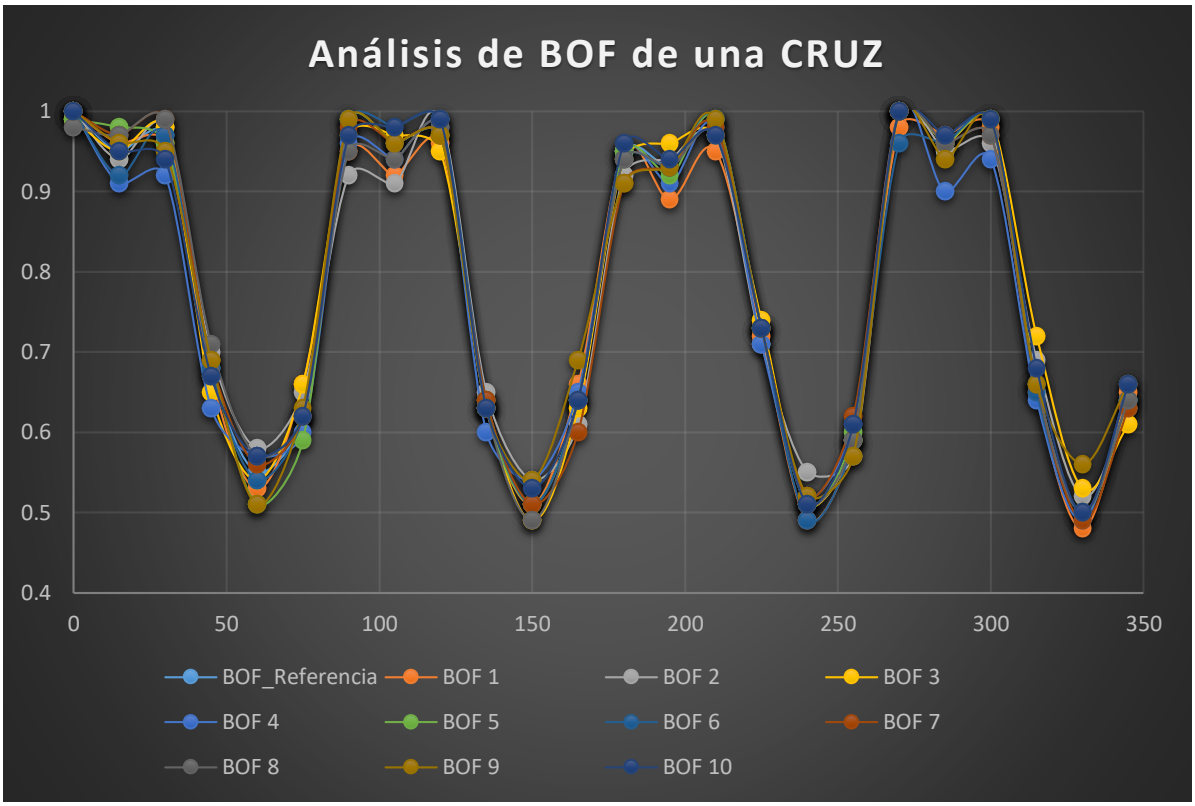


Ilustración 102 Gráficas de varios vectores de una cruz.

BOF Triángulo

Continuamos el proceso con un triángulo (ilustración 103), realizando el procesamiento de la imagen y obtenido su vector de referencia.

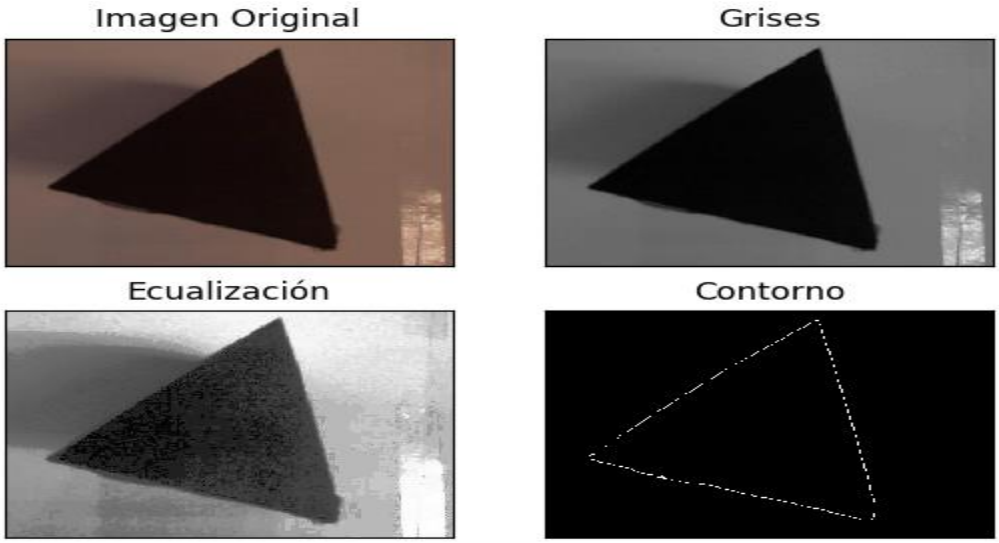


Ilustración 103 Procesamiento de imagen de un triángulo.

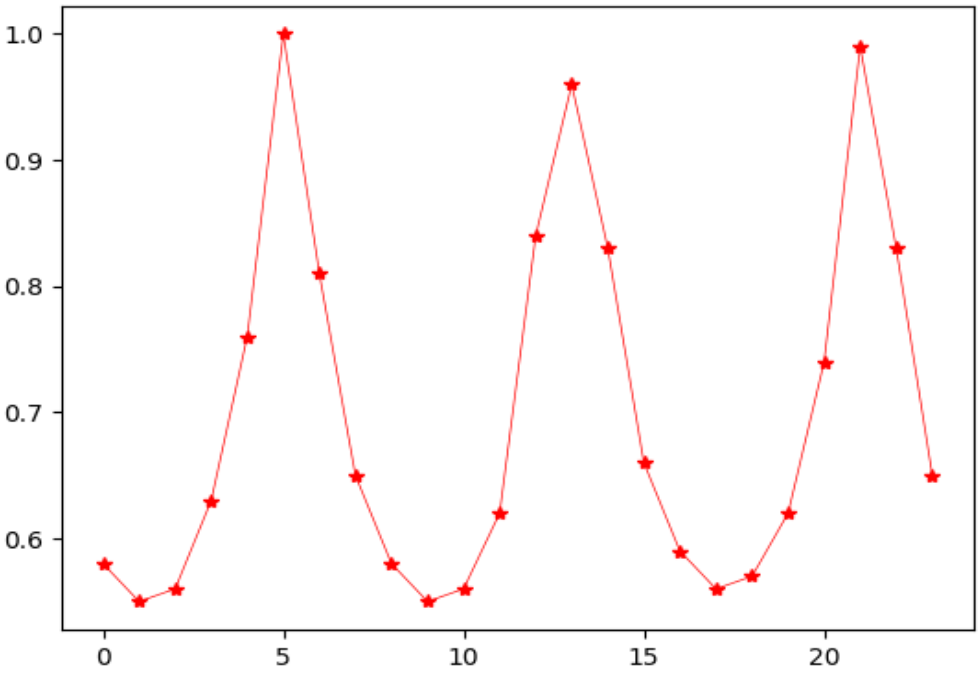


Ilustración 104 Vector de referencia de un triángulo.

Realizamos el proceso de obtención de BOF varia veces (tabla 13).

Grados	*BOF Referencia*	*BOF 1*	*BOF 2*	*BOF 3*	*BOF 4*	*BOF 5*	*BOF 6*	*BOF 7*	*BOF 8*	*BOF 9*	*BOF 10*	*Error MAX [%]*
0	0.58	0.57	0.58	0.58	0.59	0.58	0.59	0.58	0.58	0.58	0.59	1.7
15	0.55	0.54	0.57	0.53	0.51	0.56	0.55	0.55	0.5	0.53	0.58	5.4
30	0.56	0.57	0.51	0.6	0.56	0.57	0.57	0.58	0.54	0.57	0.57	7.1
45	0.63	0.59	0.63	0.64	0.61	0.65	0.62	0.65	0.6	0.67	0.6	6.3
60	0.76	0.76	0.78	0.75	0.79	0.76	0.78	0.79	0.76	0.75	0.75	3.9
75	1	1	0.99	1	1	1	1	1	1	1	1	1
90	0.91	0.95	0.88	0.94	0.92	0.89	0.91	0.88	0.91	0.87	0.92	4.3
105	0.65	0.65	0.69	0.63	0.64	0.64	0.68	0.68	0.69	0.68	0.65	6.1
120	0.58	0.61	0.58	0.58	0.53	0.57	0.59	0.61	0.54	0.61	0.58	6.8
135	0.55	0.51	0.54	0.51	0.57	0.55	0.54	0.57	0.52	0.54	0.54	7.2
150	0.56	0.56	0.59	0.57	0.54	0.59	0.55	0.6	0.54	0.57	0.55	7.1
165	0.62	0.62	0.61	0.62	0.62	0.63	0.6	0.63	0.61	0.63	0.59	1.6
180	0.84	0.84	0.83	0.84	0.85	0.87	0.84	0.86	0.84	0.8	0.84	4.7
195	0.96	0.96	0.97	0.96	0.96	0.98	0.96	0.95	0.96	0.96	0.97	2
210	0.83	0.84	0.83	0.83	0.83	0.83	0.82	0.82	0.82	0.87	0.83	4.8
225	0.66	0.64	0.62	0.69	0.65	0.65	0.67	0.66	0.67	0.7	0.66	6
240	0.59	0.61	0.59	0.56	0.59	0.61	0.6	0.6	0.58	0.63	0.61	5
255	0.56	0.56	0.51	0.59	0.57	0.58	0.58	0.56	0.51	0.54	0.53	5
270	0.57	0.55	0.58	0.55	0.58	0.59	0.57	0.59	0.58	0.57	0.56	3.5
285	0.62	0.62	0.62	0.59	0.63	0.62	0.58	0.62	0.6	0.66	0.59	4.8

300	0.74	0.77	0.73	0.72	0.74	0.74	0.77	0.79	0.7	0.74	0.73	6.7
315	0.99	0.99	1	0.99	0.99	0.99	1	0.99	0.99	1	0.99	1
330	0.83	0.83	0.8	0.81	0.83	0.82	0.86	0.87	0.82	0.83	0.83	4.8
345	0.65	0.68	0.63	0.64	0.65	0.68	0.67	0.6	0.69	0.64	0.64	4.6

Tabla 13 Análisis de varios vectores de un triángulo.

Obtenemos las gráficas para corroborar que todos tengan la misma tendencia (ilustración 105) y obtenemos un error del 4.6%.

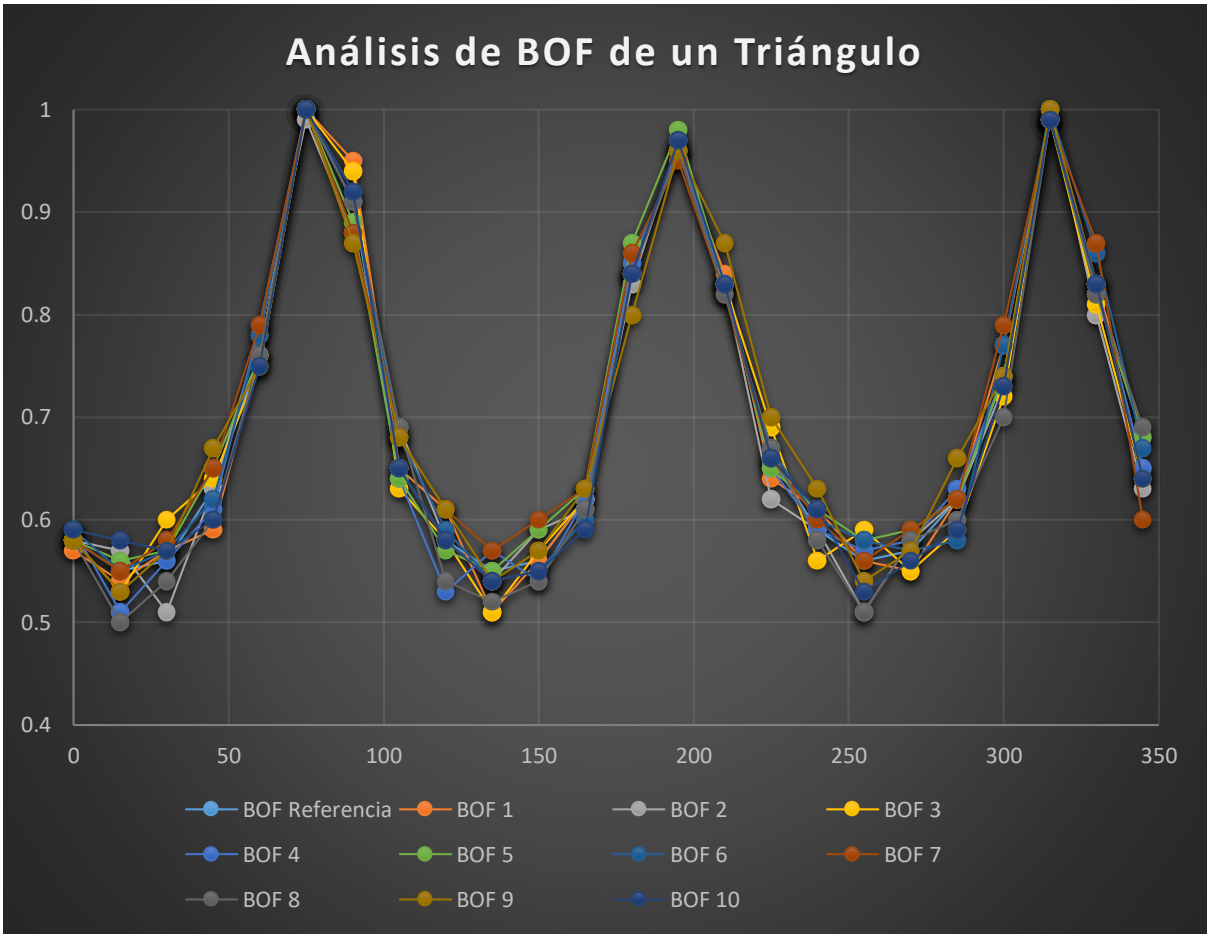


Ilustración 105 Gráficas de varios vectores de un triángulo.

BOF Círculo

Finalmente realizamos el procesamiento de imagen de un círculo (ilustración 106) y obtenemos su vector de referencia (ilustración 107).

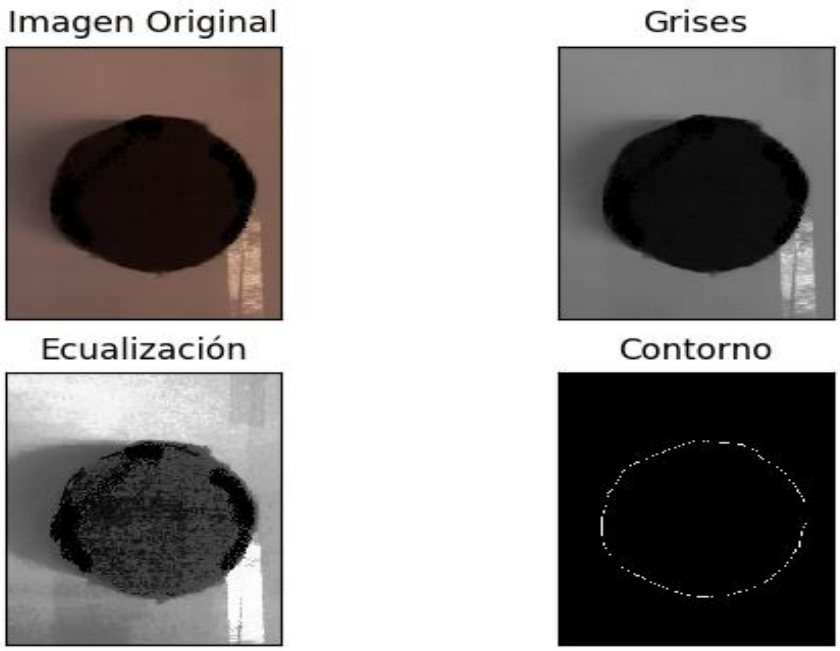


Ilustración 106 Procesamiento de imagen de un círculo.

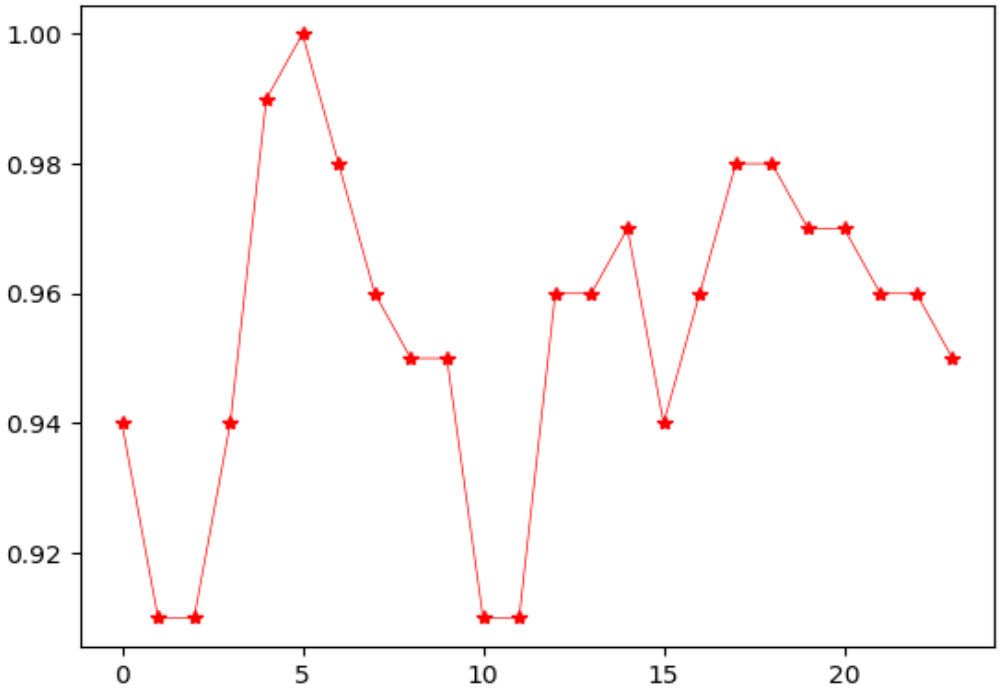


Ilustración 107 Vector de referencia de un círculo.

Realizamos el procedimiento varias veces más, para realizar una comparación en la tabla 14.

Grados	*BOF Referencia*	*BOF 1*	*BOF 2*	*BOF 3*	*BOF 4*	*BOF 5*	*BOF 6*	*BOF 7*	*BOF 8*	*BOF 9*	*BOF 10*	*Error MAX [%]*
0	0.94	0.93	0.94	0.93	0.95	0.94	0.94	0.94	0.94	0.93	0.95	1
15	0.91	0.91	0.92	0.9	0.92	0.91	0.91	0.91	0.9	0.91	0.9	1
30	0.91	0.91	0.9	0.9	0.92	0.91	0.91	0.92	0.9	0.91	0.9	1
45	0.94	0.95	0.94	0.93	0.95	0.95	0.94	0.95	0.93	0.94	0.96	2.1
60	0.99	0.99	0.99	0.98	0.99	0.99	0.99	0.99	0.98	0.99	0.99	1
75	1	1	1	1	1	1	1	1	1	1	1	1
90	0.98	0.97	0.98	0.98	0.99	0.99	0.99	0.99	0.99	0.98	0.99	1
105	0.96	0.96	0.97	0.96	0.96	0.96	0.96	0.96	0.97	0.96	0.99	3.1
120	0.96	0.95	0.97	0.96	0.95	0.96	0.95	0.96	0.96	0.96	0.96	1
135	0.95	0.95	0.96	0.95	0.95	0.95	0.95	0.95	0.96	0.95	0.95	1
150	0.95	0.94	0.96	0.95	0.94	0.96	0.94	0.95	0.96	0.95	0.95	1
165	0.91	0.91	0.93	0.92	0.91	0.92	0.9	0.91	0.92	0.94	0.94	3.2
180	0.91	0.9	0.93	0.92	0.9	0.92	0.9	0.91	0.92	0.95	0.94	4.3
195	0.96	0.96	0.97	0.97	0.95	0.96	0.95	0.96	0.97	0.96	0.97	1
210	0.96	0.97	0.97	0.97	0.96	0.96	0.95	0.97	0.97	0.97	0.97	1
225	0.97	0.97	0.97	0.97	0.96	0.98	0.97	0.97	0.98	0.97	0.97	1
240	0.94	0.95	0.95	0.95	0.93	0.95	0.94	0.94	0.93	0.96	0.94	2.1
255	0.96	0.97	0.96	0.97	0.96	0.96	0.97	0.96	0.95	0.97	0.96	1
270	0.98	0.98	0.98	0.98	0.98	0.99	0.98	0.99	0.98	0.98	0.99	1
285	0.98	0.98	0.98	0.98	0.98	0.99	0.98	0.98	0.98	0.97	0.99	1
300	0.97	0.98	0.97	0.97	0.99	0.98	0.97	0.98	0.97	0.97	0.98	1
315	0.97	0.97	0.97	0.96	0.97	0.98	0.97	0.97	0.97	0.96	0.98	1
330	0.96	0.97	0.96	0.96	0.97	0.97	0.97	0.97	0.96	0.94	0.97	2.1
345	0.96	0.96	0.96	0.95	0.97	0.96	0.96	0.95	0.96	0.94	0.96	2.1

Tabla 14 Análisis de varios vectores de un círculo.

Obtenemos que la tendencia representa una línea recta (ilustración 108) y, observamos que esta figura contiene el menor error de todas al obtener un 1.5%.

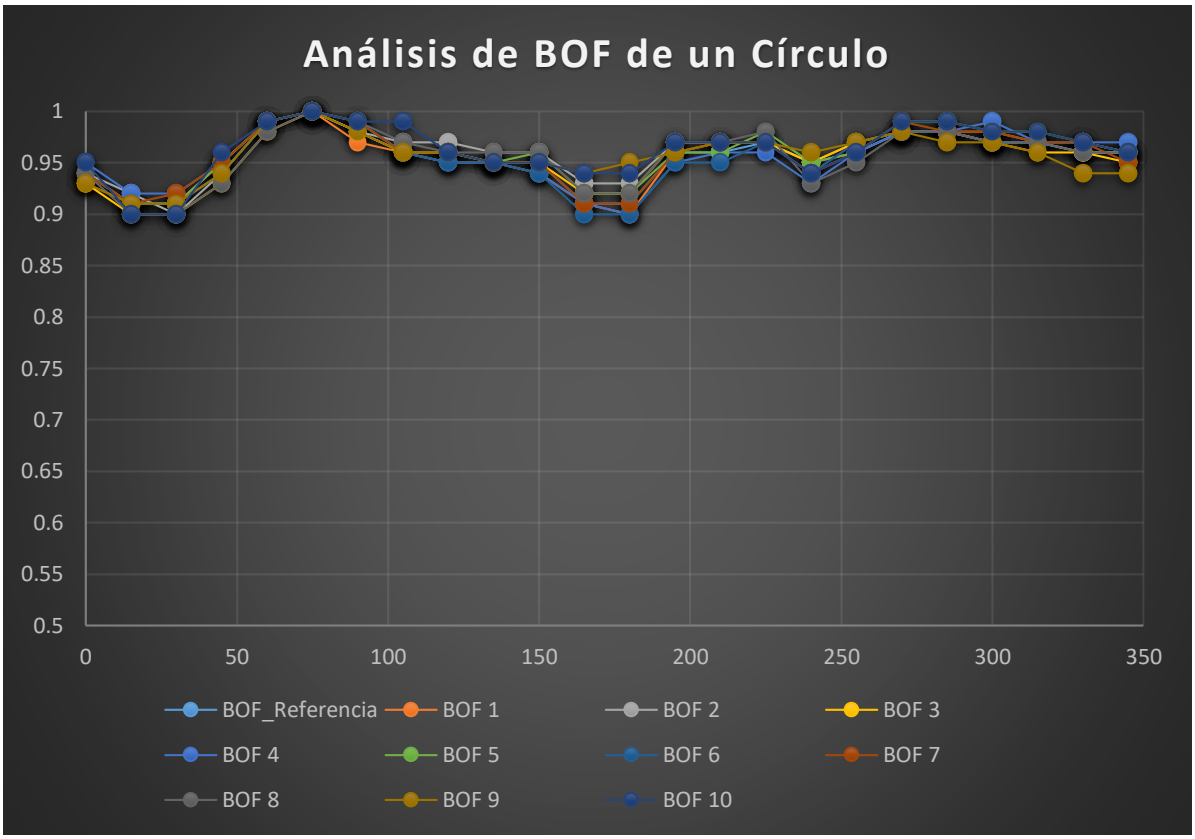


Ilustración 108 Gráficas de varias vectores de un círculo.

Velocidad de procesamiento.

Un análisis importante es conocer el tiempo de procesamiento de la imagen, que inicia desde la obtención de la imagen a través de la cámara hasta la identificación de la figura, este valor se obtiene mediante el uso de librería “time” de la raspberry. Se realizan pruebas con 3 diferentes resoluciones de imágenes 320x320 pixeles, 640x480 pixeles y 1280x960 pixeles para conocer los tiempos de procesamiento, tomando en consideración distintas cantidades de puntos para formar nuestro vector descriptor (Tabla 15).

Puntos de BOF	*Resolución 320X320*	*Resolución 640x480*	*Resolución 1280x960*
10	3 [s]	7 [s]	68 [s]
12	3 [s]	7 [s]	68 [s]
15	3 [s]	7 [s]	69 [s]
20	3 [s]	7 [s]	68 [s]
24	3 [s]	7 [s]	70 [s]
30	3 [s]	7 [s]	70 [s]
36	3 [s]	7 [s]	70 [s]

Tabla 15 Velocidad de procesamiento de 3 diferentes resoluciones de imágenes.

En la ilustración 109, podemos observar que la mejor resolución le lleva a la raspberry un tiempo de procesamiento que sobrepasa un minuto esto por tener una mayor cantidad de pixeles en la imagen.

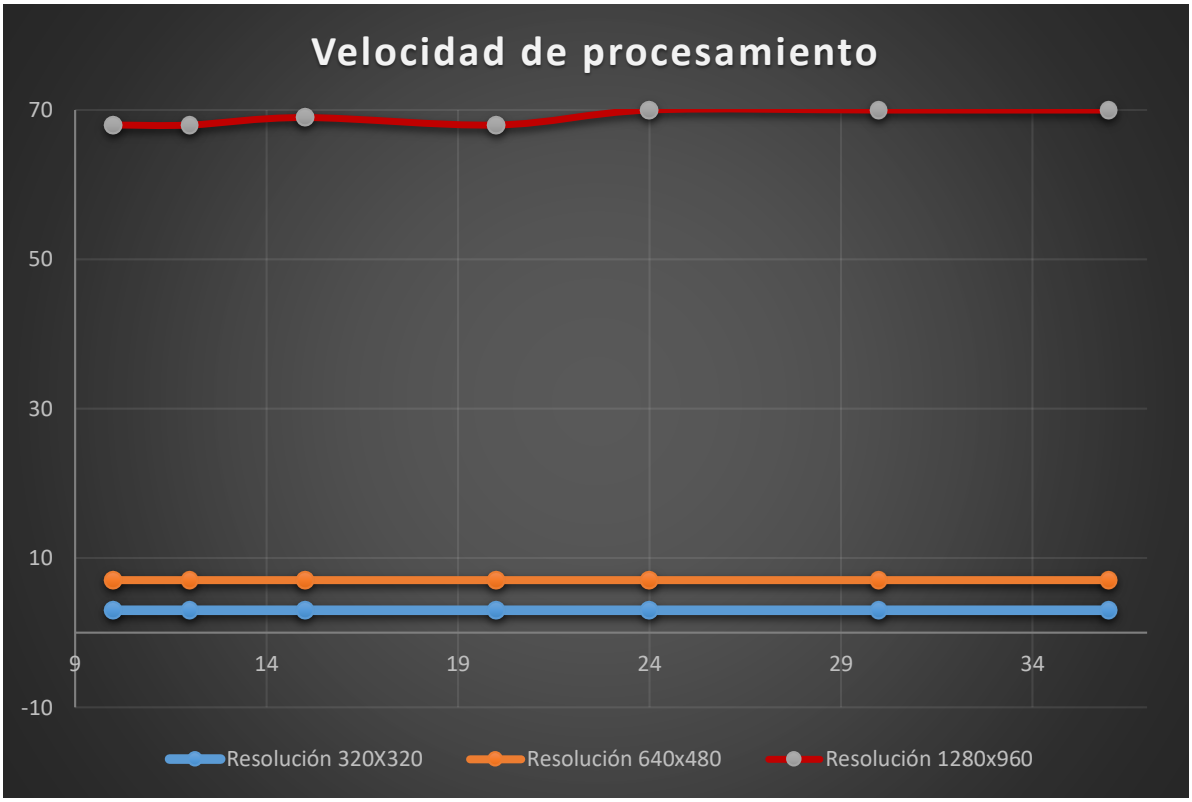


Ilustración 109 Tiempo de procesamiento con 3 diferentes resoluciones.

Para realizar una comparación, se realiza la experimentación del porcentaje de reconocimiento de las figuras, tomando en consideración la cantidad de puntos de nuestro vector descriptor (tabla 16).

Puntos de BOF	*Reconocimiento 268x268*	*Reconocimiento 640x480*	*Reconocimiento 1280x960*
10	60	80	83
12	60	90	93
15	62	99	99
20	78	99	99
24	85	99	99
30	89	99	99
36	91	99	99

Tabla 16 Porcentaje de reconocimiento de las figuras analizadas.

En la ilustración 110, se muestran los resultados del proceso de reconocimiento de las 3 diferentes resoluciones de imágenes, tomando en cuenta diferentes cantidades de puntos del vector.

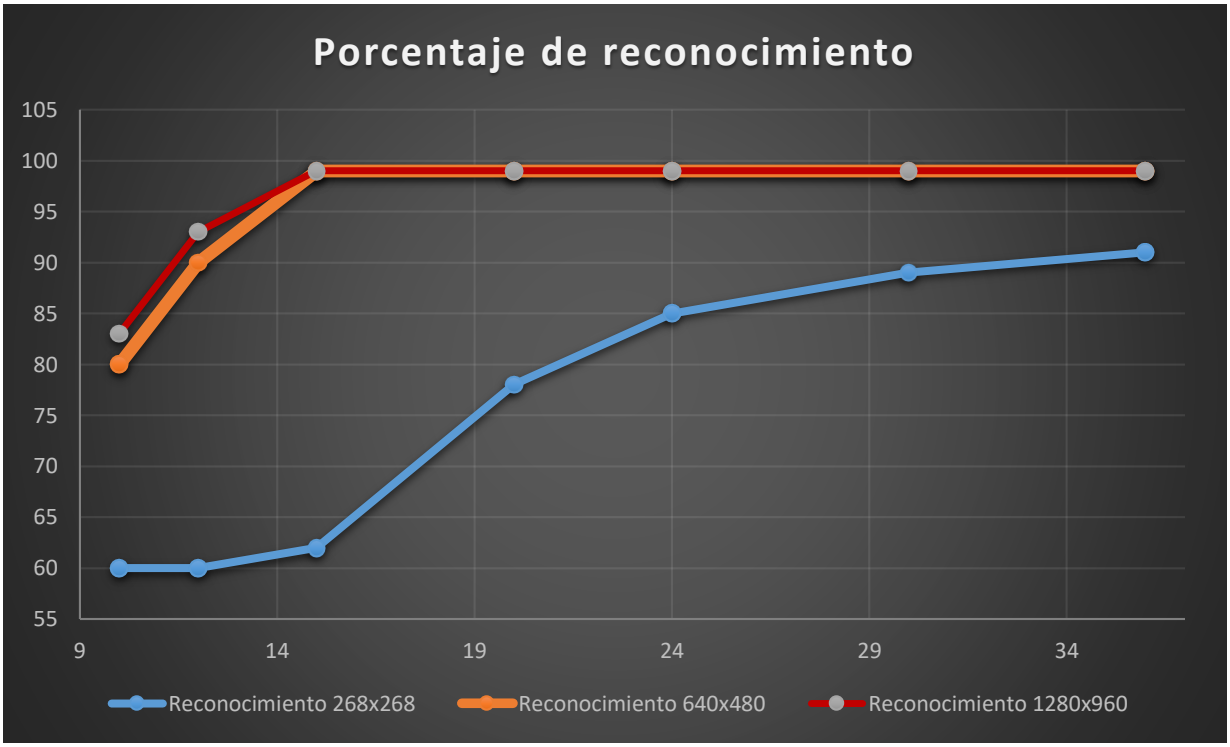


Ilustración 110 Porcentaje de reconocimiento con 3 distintas resoluciones de imágenes.

Escaneo de figuras

Al hacer uso de un modelo mano-ojo, se puede realizar una inspección más detallada de las figuras realizando un barrido en diferentes ángulos, esto se logra rotando la cámara haciendo uso de las variables de la calibración de la "HERRAMIENTA", esto con la finalidad de identificar figuras respecto a otras parecidas (ilustración 111, 112, 113).

Se realiza la inspección de las figuras, realizando inclinaciones de la cámara respecto a los ejes del área de trabajo. Para las pruebas se realiza una inclinación de 45° de la cámara sobre el ángulo Z respecto a la base de trabajo, mientras que se realiza una rotación de la cámara en el eje X desde 0° hasta 90°. En las siguientes ilustraciones 111, 112 y 113, se muestran los resultados del escaneo de un cubo, mientras que los resultados de las figuras restantes se pueden observar en el anexo III.

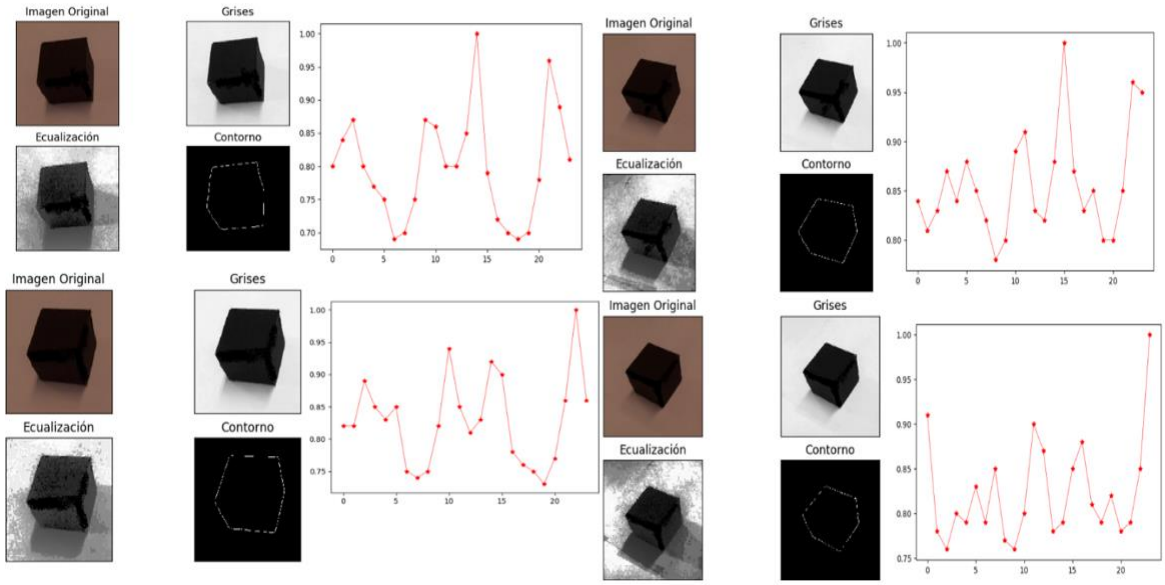


Ilustración 111 Escaneo de figura de un cubo con rotación cámara.

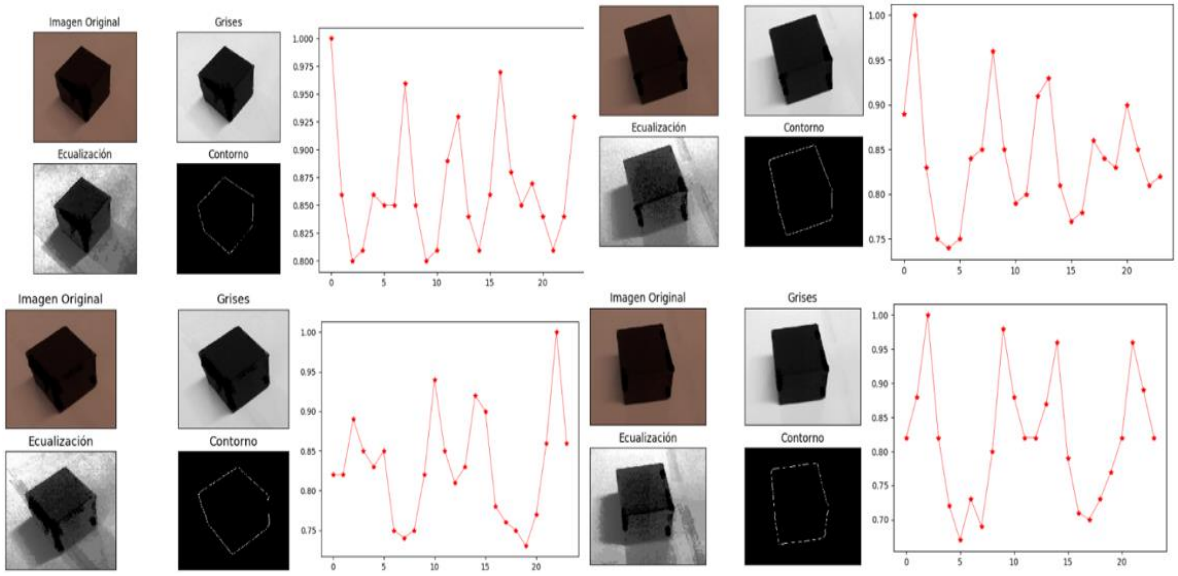


Ilustración 112 Escaneo de figura de un cubo con rotación de cámara.

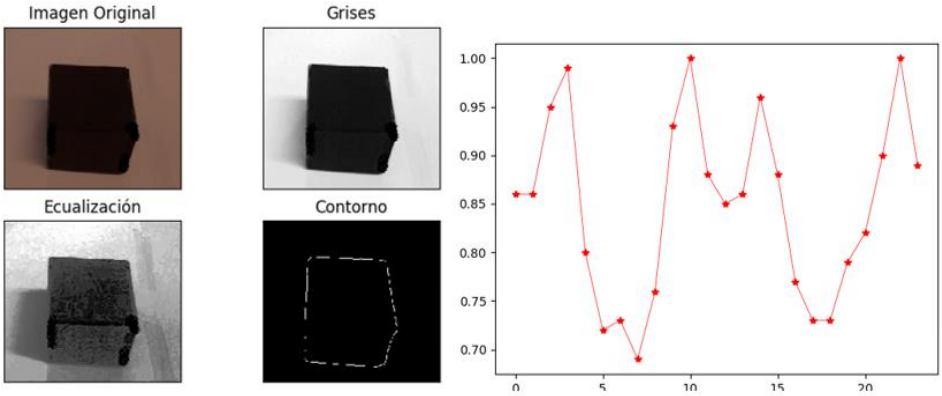


Ilustración 113 Escaneo de figura de un cubo con rotación de cámara.

Conclusiones

Podemos empezar dividiendo las conclusiones en tres aspectos: El primero, referente a los resultados de calibración del robot de la cámara-área de trabajo y la calibración cámara-robot. El segundo aspecto a comentar es con referencia al procesamiento de la imagen y los resultados del BOF. Por último, mencionar los resultados del escaneo de las imágenes mediante la rotación de la cámara.

En los resultados referentes a la cámara y el área de trabajo, se puede observar que el porcentaje de error en cuanto a la ubicación del centroide no sobrepasa el 0.5%, lo que significa que el procesamiento de la imagen es muy bueno, al no involucrar una gran cantidad de ruido que pueda modificar la posición del centroide. De igual manera se realizó el proceso de obtención del centroide con una mala fuente de iluminación, se ejecutó el proceso obstruyendo las fuentes de iluminación y se corroboró que el proceso de ecualización mejora el contraste de la imagen, siendo su porcentaje de error menor al 2% en las peores condiciones ambientales. En cuanto a la ubicación del centroide y el posicionamiento del herramental, en éste se comprobó que el máximo error fue de un 4% sobre el eje X y un 10% sobre el eje Y y aunque parezca bastante el error en estos puntos, esta cantidad simboliza un desplazamiento no mayor a los 2 [mm] por cada eje desde centroide al momento de realizar su medición físicamente.

En los resultados referentes al procesamiento de la imagen y reconocimiento de figuras, podemos concluir que la resolución de la imagen impacta en gran medida al tiempo de procesamiento de la raspberry, también se observó que para las resoluciones de 320x320 pixeles y 640x480 pixeles la cantidad de puntos del vector descriptor era insignificante, pues el obtener una mayor o menor cantidad de puntos no repercutía en el tiempo. Se puede corroborar que el tomar una mayor cantidad de puntos para construir el vector define de una mejor manera la figura y por consecuencia se obtiene un mayor porcentaje de aciertos en el reconocimiento. Analizando estos resultados se puede constatar que las mejores características para realizar el BOF serían con una imagen de resolución de 640x480 pixeles y 24 puntos del BOF y con esto tener el menor tiempo de procesamiento. Analizando los resultados de obtención de BOF, donde se revisó la tendencia que tenía el realizar varias veces el experimento, se corroboro que el mayor porcentaje de error en un punto del vector no era mayor al 6.5%, este resultado nos llevó a proponer en el análisis de las bases de datos una tolerancia en la identificación de los puntos de la figura para asegurar que la mayoría de puntos estuvieran dentro del intervalo del vector de referencia y con esto lograr casi el 100% en el reconocimiento.

Por último, se realizaron pruebas para obtener diferentes vectores descriptores de las figuras a diferentes ángulos, esto con la finalidad de reconocer figuras que puedan ser parecidas dependiendo de su ángulo de visión, por ejemplo, distinguir una esfera de un cilindro. Este último punto abre una puerta para nuevas

aplicaciones que se pueden llevar a cabo con la manipulación del robot y el modelo mano-ojo.

Como conclusión general, se logró la meta del trabajo al realizar la integración de un algoritmo de reconocimiento de formas rígidas en un sistema embebido, así como realizar la comunicación del mismo con un robot tipo industrial llevando a cabo las calibraciones correspondientes, ya que para los trabajos revisados todos usan computadoras de grandes prestaciones para realizar el mismo proceso que nosotros realizamos en la raspberry pi. El lograr llevar acabo esta integración abre la posibilidad a realizar celdas de manufacturas que puedan ser reconfigurables, al reproducir el sistema embebido para que varias celdas puedan realizar el mismo proceso o adecuarlo para nuevas instrucciones añadiendo otros tipos de sensores, tomando en cuenta que el sistema se puede conectar a una red de trabajo y puede ser operado desde cualquier parte del mundo.

Trabajo futuro

El trabajo abre la posibilidad a la instalación en el sistema embebido de diferentes tipos de sensores, que puedan ayudar a la capacidad de visión que se logró dar al robot, por ejemplo: sensores de presión, distancia, u otros que puedan contribuir a la interacción del robot con el mundo físico.

Actualmente la evolución de la industria 4.0 trae consigo el desarrollo de más y mejores sistemas embebidos para el desarrollo de este sector, por consiguiente, se pueden realizar pruebas migrando este código en sistemas más potentes y mejorar la velocidad de procesamiento como lo sería una Dev Board de google, en la cual tienen una GPU dedicada al procesamiento de este tipo de información.

En la inspección de objetos, cuando se proporciona una rotación a la cámara usando el modelo mano-ojo, abre la posibilidad a realizar investigaciones para obtener las dimensiones reales de las figuras, puesto que, conocer de una manera autónoma las dimensiones físicas de un objeto resulta complicado, además, se podría obtener a partir de varios vectores descriptores a diferentes ángulos una imagen en 3D de la figura.

Bibliografía

[1]J. Shaw, KY Cheng. (2016). Object Identification and 3-D Position Calculation Using Eye-in-Hand Single Camera for Robot Gripper. Institute of Electrical and Electronic Engineers (IEEE).

[2]Schwab Klaus. (3 Nov 2016). La cuarta revolución industrial. Ed. Grupo España.

[3]CHUN-Tai Yen, Yu-Chi Liu. (2014). Advanced Manufacturing Solution to Industry 4.0 trend through Sensing Network and Cloud Computing Technologies. Institute of Electrical and Electronic Engineers (IEEE).

[4]Mingwei Wang, Jingtao Zhou. (2012). Cloud Manufacturing: Needs, Concept and Architecture. Institute of Electrical and Electronic Engineers (IEEE).

[5]Lorenzo Bassi. (2017). Industry 4.0: hope, hype or revolution?. Institute of Electrical and Electronic Engineers (IEEE).

[3]Fotios K. Konstantindis, Antonios Gasteratos. (2018). Vision-Based Product Tracking Method for Cyber-Physical Production Systems in Industry 4.0. Institute of Electrical and Electronic Engineers (IEEE).

[7]Shigeo, Abe. Support Vector Machines for Pattern Classification. Springer, 2005.

[8]Ontiveros Mauricio. (2016). Reconocimientos de objetos y manejo de un brazo robótico mediante un procesador sitara, IIMAS, UNAM. pp3.

[9]Santhosh Krishna, Iviya J. (2016). Cloud Robotics in industry using raspberry pi. Institute of Electrical and Electronic Engineers (IEEE).

[10]Guo-jian Cheng, Li-ting Liu. (2016). Industry 4.0 Development and Application of Intelligent Manufacturing. Institute of Electrical and Electronic Engineers (IEEE).

[11]Liu Tao, Haisong Gu, Dongyan Wang. (2017). Visual Inspection System. Institute of Electrical and Electronic Engineers (IEEE).

[12]Reuben Babao, Frazanti Bianzon, Miguel Lorenzo. (2017). Integration of Visual and Thermographic Images in an Artificial Neural Network for Object Classification. Institute of Electrical and Electronic Engineers (IEEE).

[13]Ren C. Luo, Shih-Che Chou. (2014). Hybrid Eye-to-hand and Eye-in-hand Visual Servo System for Parallel Robot Conveyor Object Tracking and Fetching. Institute of Electrical and Electronic Engineers (IEEE).

[14]Emerson J. Olaya, Luz A. Torres. (2009). A Foveated Stereo Vision System for Active Depth Perception. CINVESTAV.

[15]Kyekyung Kim, Joongbae Kim. (2012). Object Recognition for Cell Manufacturing System. Institute of Electrical and Electronic Engineers (IEEE).

[16]Yu-SinWu, Min-Liang Wang, N. Michael Mayer. (2016). A New Type of Eye-on-hand Robotic Arm System Based on a Low-cost Recognition System. Institute of Electrical and Electronic Engineers (IEEE).

[17]Hassine BELHADJ, Saber BEN HASSEN. (2013). KUKA Robot control based Kinect image analysis. Institute of Electrical and Electronic Engineers (IEEE).

[18]Ignacio Dávila, Román Osorio, Ismael López. (2016). A Fuzzy Approach for on-line Error Compensation During Robotic Welding. IEEE

[19]Steve Heath. (2003). Embedded Systems Design. Edit. Newnes. pp 254.

[20]Ismael Lopez, Emmanuel Rojas. (2018). Grounding the Lexicon for Human-Robot Interaction During the Manipulation of Irregular Objects. Institute of Electrical and Electronic Engineers (IEEE)..

[21]Jim Turley. (2002). The Two Percent Solution. Wayback Machine

[22]José Vega, Roberto Sánchez. (2019). Arquitectura RISC vs CISC. 2019, de UNIVERSIDAD AUTONOMA METROPOLITANA Sitio web: https://www.azc.uam.mx/publicaciones/enlinea2/num1/1-2.htm

[23]Susana Canel. (2007). Microcontroladores ARM 32 bits.

[24]Raspberry. (2016). Manual de raspberry pi. Raspberry.org

[25]Peña, M.et al. A Learning Approach for On Line Object Recognition Tasks, Proceedings of the Fifth Mexican International Conference in Computer Science (ENC'04), 2004.

[26]M. Tim Jones, Artificial Intelligence A System Approach, Infinity Science Press LLC, Hingham Massachusetts, 2008, pp 1-7.

[27]Sparsh Mittal, Saket Gupta, S. Dasgupta, Fpga: An efficient and promising platform for real-time image processing applications, Proceedings of the National Conference on Research and Development in Hardware & Systems, 2008.

[28]M. Brady L.A. Gerhardt and H. F. Davidson. Robotics and Artificial Intelligence. Proccedings of the NATO Advanced Study Institute on Robootics and Artificial Intelligence held Castelvecchio Pascoli (Barga). Italy, June 26-July 1983.

[29]José Jaime Esqueda Elizondo, Luis Enrique Palafox Maestre. Fundamentos de procesamiento de imágenes. Universidad Autónoma de Baja California, 2005.

[30]R. Hall. (2000). Illumination and Color in Computer generated Imagery. Springer Verlag.

[31]José Cortés Parejo. (2000). La percepción del color.

[32]Zhe-Ming Lu, Shi-Ze Guo. (2017). Lossless Information Hiding in Images on Transform Domains. Columbia University

[33]Dorothy, Joany & Rathish, Joseph & Santhana Prabha. (2015). Image enhancement by Histogram equalization. International Journal of Nano Corrosion Science and Engineering.

[34]Javier Ruiz. (2015). Procesamiento avanzado de imágenes. Universidad de Chile.

[35]Lentin Joseph. (2015). Learning Robotics Using Python. Packt.

[36]Logitec. (2019). SPECIFICATIONS. 2019, de Logitec Sitio web: https://support.logitech.com/en_us/product/hd-webcam-c525/specs

[37]Basler AG. (2019). acA1300-30gc - Basler ace. 2019, de Basler the power of sight Sitio web: https://www.baslerweb.com/en/products/cameras/area-scan-cameras/ace/aca1300-30gc/#tab=specs

[38]Basler. (2015). Basler cameras.

[39]KUKA Roboter. KUKA System Software 8.2 Operating and Programming Instructions for end users, 2011, Vol. 1.

[40]M. Peña Cabrera, R. Osorio, I. Lopez. (2010). Real Time Object Recognition Methodology. Institute of Electrical and Electronic Engineers (IEEE). Conference, pp 439-444.

[41]Hyungwon Sung, Daesik Kim. A Robot-Camera Hand/Eye Self-Calibration System Using a Planar Target. Institute of Electrical and Electronic Engineers (IEEE).

[42]KUKA Robot Group. Kr 5, Specification, 2016, Vol. 3.

[43]Rafael Magro. (2013). BINARIZACIÓN DE IMÁGENES DIGITALES Y SU ALGORITMIA COMO HERRAMIENTA APLICADA A LA ILUSTRACIÓN ENTOMOLÓGICA. Sociedad Entomológica Aragonesa.

[44]Serope Kalpakjian, Steven R. Schmid. Manufactura, Ingeniería y tecnología. Ed. Pearson Educación, 2014.

[45]Tao Lio. (2017). Visual Inspection System for Smart Manufacture of Home Appliances. Institute of Electrical and Electronic Engineers (IEEE).

[46]Jianchend Jia.(2009) A Machine Vision Application for Industrial Assembly Inspection. Institute of Electrical and Electronic Engineers (IEEE).

[47]Suwoo Park, SunHee Baek. (2018). A Study on Development of th e Blind Spot Detection System for the IoT-Based Smart Connected Car. Institute of Electrical and Electronic Engineers (IEEE).

[48]Amy Tabb, Khalil Ahmad. (2015). Parameterizations for Reducing Camera Reprojection Error for Robot-World Hand-Eye Calibration. Institute of Electrical and Electronic Engineers (IEEE).

[49]Seung-ho Kim, Hyeong-jun Cho. (2015). Effective Object Recognition based on CUDA. Institute of Electrical and Electronic Engineers (IEEE).

[50]Kwang.Hee Lee, Hyun-Su Kim. High Precision Hand-Eye Self-Calibration for Industrial Robots. Institute of Electrical and Electronic Engineers (IEEE).

[51]Ahmad Ghasemi, Wen-Fang Xie. (2017). Decoupled Image-Based Visual Servoing for Robotic Manufacturing Systems Using Gain Scheduled Switch Control. Institute of Electrical and Electronic Engineers (IEEE).

[52]Marcos A. Pisching, Marcosiris A.O. Pessoa. (2018). PFS/PN Technique to Model Industry 4.0 Systems based on RAMI 4.0. Institute of Electrical and Electronic Engineers (IEEE).

[53]Simone Pasinetti, Giovanna Sansoni. (2018). In-line monitoring of laser welding using a smart vision system. Institute of Electrical and Electronic Engineers (IEEE).

[54]Ivan Lundberg, Marten Bjorkman and Petter Ogren. (2014). Intrinsic Camera and Hand-Eye Calibration for a Robot Vision System using a Point Marker. Institute of Electrical and Electronic Engineers (IEEE).

[55]Kyekyung Kim, Sangseung Kang. (2013). Multiple Objects Recognition for Industrial Robot Applications. Institute of Electrical and Electronic Engineers (IEEE).

[56]Liang Zhang, Jian-Zhou Zhang. (2017). Object-Oriented Stripe Structured-Light Vision-Guided Robot. Institute of Electrical and Electronic Engineers (IEEE).

[57]Gridsada Phanomchoeng, Ratchatin Chanchareon. (2017). Projected Pattern on Three-dimensional Objects for Image Feature Classification and Recognition. Institute of Electrical and Electronic Engineers (IEEE).

[58]Ryuki Funakubo, Khaing Win Phyu. (2016). Recognition and Handling of Clothes with Different Pattern by Dual Hand-eyes Robotic System. Institute of Electrical and Electronic Engineers (IEEE).

[59]Hyungwon Sung, Sukhan Lee. A Robot-Camera Hand/Eye Self-Calibration System Using a Planar Target. Institute of Electrical and Electronic Engineers (IEEE).

[60]M. Peña, I. Lopez, R, Osorio. (2007). Robot Vision Methodology for Assembly Manufacturing Tasks. Institute of Electrical and Electronic Engineers (IEEE).

[61]InstitutoPolitécnico Nacional (IPN). (2007). Tecnología informática contra plagas. Conversus, vol. 60, pp. 74.

[62]Josefina Lopez Herrera. (4 de octubre del 2011). Programación en tiempo real y bases de datos: Un enfoque práctico. Universidad politecnica de cataluña: Iniciativa Digital.

ANEXO I

Ángulo de visión de la cámara

El ángulo de visión de la cámara depende del objetivo y sus parámetros, y sobre todo de la distancia focal del objetivo. Esta es la distancia entre el centro óptico del objetivo en que se enfoca la luz y el sensor sobre el cual incide la luz; suponiendo que el enfoque está ajustado a infinito (Ilustración 114).

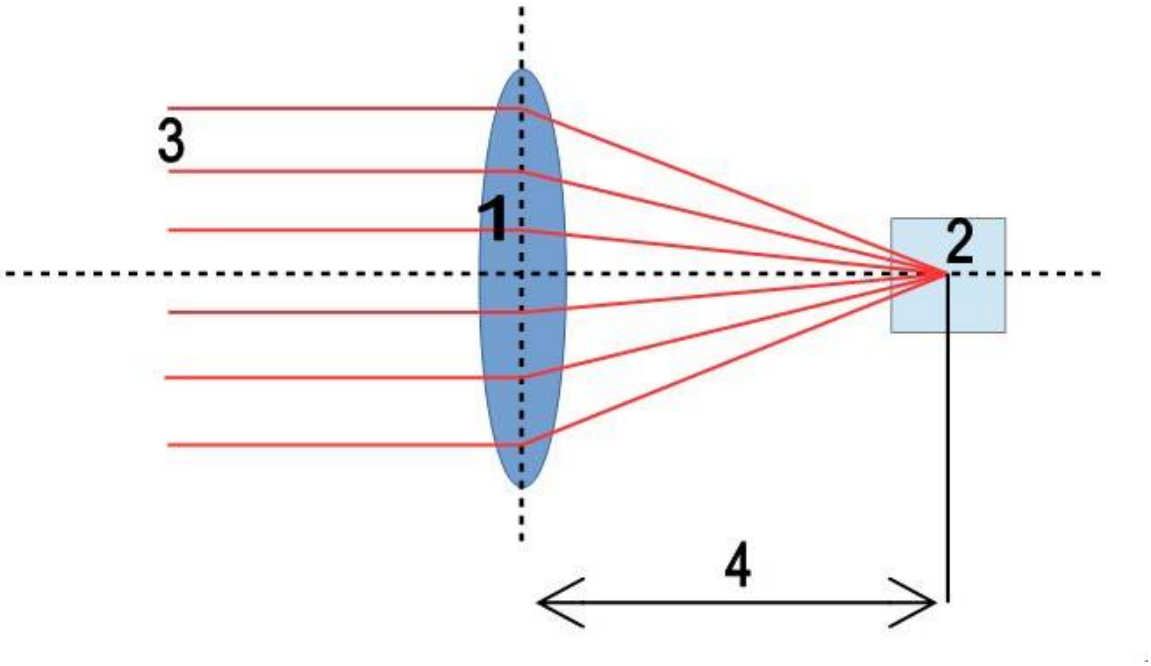


Ilustración 114 Ángulo de visión de cámara.

1 - Objetivo

2 - Enfoque de la imagen (sensor)

3 - Rayos de luz

4 - Distancia focal

Por lo tanto, como se muestra en la figura, el ángulo de visión (2) no sólo depende de la distancia focal (4), sino también del tamaño del sensor (5) sobre el cual incide la luz. Cuanto mayor sea la distancia focal, menor será el ángulo de visión de la cámara. En cuanto al sensor, cuanto mayor sea el elemento fotosensible utilizado, mayor será el ángulo de visión. Cabe señalar, sin embargo, que cada objetivo está diseñado para las dimensiones específicas del sensor. Por lo tanto, el mismo objetivo puede tener un ángulo de visión diferente en diferentes cámaras.

ANEXO II

Calibración

Para una celda de manufactura que pueda ser flexible y reconfigurable tendrá que hacer uso de múltiples áreas de trabajo (ilustración 115).

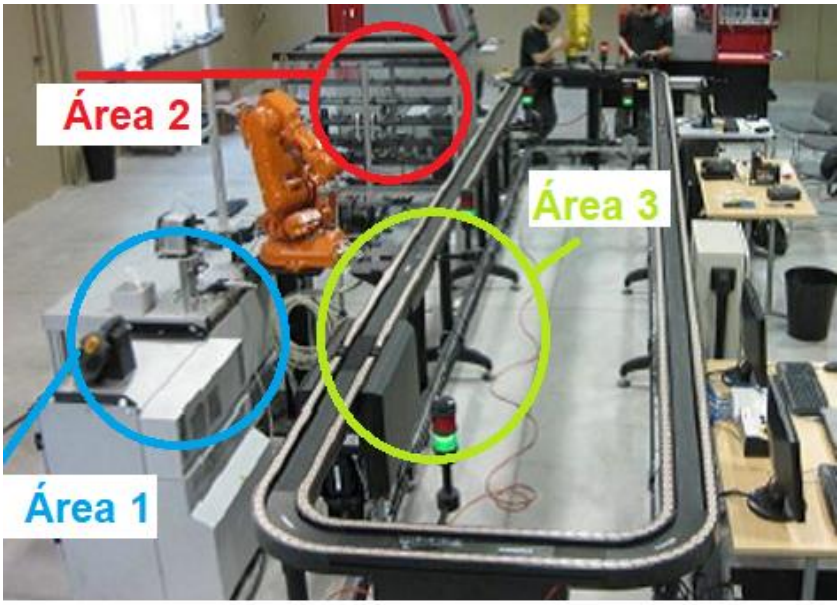


Ilustración 115 Celda de manufactura con múltiples áreas de trabajo.

En el caso de requerir más áreas de trabajo, se tendrá que realizar la calibración de cada una de estas, asignándole las mismas características de la herramienta. Se tiene que tomar en consideración los límites definidos por software, así como también los límites físicos del mismo robot (ilustración 116).

Ilustración 116 Verificación de los límites físicos del robot KUKA.

Otro aspecto importante a tomar en consideración para realizar un múltiple uso de bases de trabajo, será la relación de los ángulos que pueda tomar la herramienta con cada una de las bases de trabajo.

ANEXO III

Escaneo de Figuras

Se realiza una inspección de las figuras rotando la cámara que maneja el robot, con la intención de lograr identificar aquellas figuras que bajo una misma perspectiva puedan parecer idénticas. Este análisis se realizó para analizar figuras como los son: una estrella, cruz, triangulo y un cilindro.

En la siguiente tabla 18, se muestra el resultado obtenido del escaneo de una estrella mediante la inclinación de la cámara a 45° sobre el eje Z, con referencia a la base de trabajo y un recorrido de la cámara de 0° a 90°, sobre el eje X, con referencia a la misma base de trabajo.

Estrella

La siguiente tabla muestra los BOF, correspondientes a distintas rotaciones de la cámara.

Rotación 0°	Rotación 10°	Rotación 20°	Rotación 30°	Rotación 40°	Rotación 50°	Rotación 60°	Rotación 70°	Rotación 80°	Rotación 90°
0.66	0.7	0.98	0.94	0.85	0.93	0.65	0.66	0.8	0.97
0.96	0.67	0.67	0.94	0.96	0.94	0.72	0.73	0.63	0.71
0.72	0.82	0.79	0.8	0.79	0.99	0.97	0.92	0.74	0.72
0.56	0.95	0.93	0.76	0.76	0.9	0.85	0.94	0.94	0.89
0.53	0.99	1	0.75	0.67	0.89	0.81	0.91	0.86	0.97
0.67	0.95	0.92	0.81	0.7	0.78	0.78	0.95	0.87	0.93
1	0.98	0.72	0.95	0.73	0.98	0.87	0.98	0.89	0.97
0.65	0.64	0.64	1	1	0.96	0.76	0.69	0.7	0.84
0.52	0.76	0.81	0.76	0.75	1	1	0.88	0.67	0.72
0.56	0.97	0.93	0.68	0.64	0.79	0.81	0.93	1	0.89
0.69	0.76	0.73	0.79	0.68	0.71	0.68	0.76	0.8	0.99
0.94	0.7	0.71	0.92	0.88	0.83	0.69	0.72	0.68	0.79
0.65	0.91	0.86	0.8	0.76	0.98	0.87	0.89	0.73	0.76
0.54	1	0.97	0.72	0.65	0.72	0.74	0.91	0.91	0.87
0.58	0.79	0.79	0.88	0.76	0.77	0.67	0.69	0.71	0.94
0.75	0.68	0.66	0.96	0.92	0.96	0.77	0.77	0.68	0.7
0.84	0.82	0.79	0.89	0.85	0.99	0.94	0.98	0.8	0.78
0.58	0.93	0.92	0.9	0.85	0.96	0.88	0.95	0.91	0.98
0.51	0.98	0.99	0.75	0.73	0.98	0.87	0.98	0.89	0.97
0.63	0.85	0.87	0.71	0.62	0.78	0.8	0.9	0.95	0.98
0.91	0.69	0.7	0.97	0.79	0.69	0.65	0.72	0.76	1
0.75	0.7	0.7	0.9	0.91	0.96	0.66	0.72	0.64	0.78
0.55	0.91	0.92	0.71	0.69	0.93	0.99	1	0.74	0.69
0.52	0.98	0.98	0.74	0.65	0.73	0.75	0.83	0.98	0.91

Tabla 17 Vectores descriptores de una estrella con rotación.

Siguiendo con el análisis de las figuras, se muestran en las ilustraciones 118, 119, 120 y 121 el resultado de una figura de estrella obtenida mediante las rotaciones.

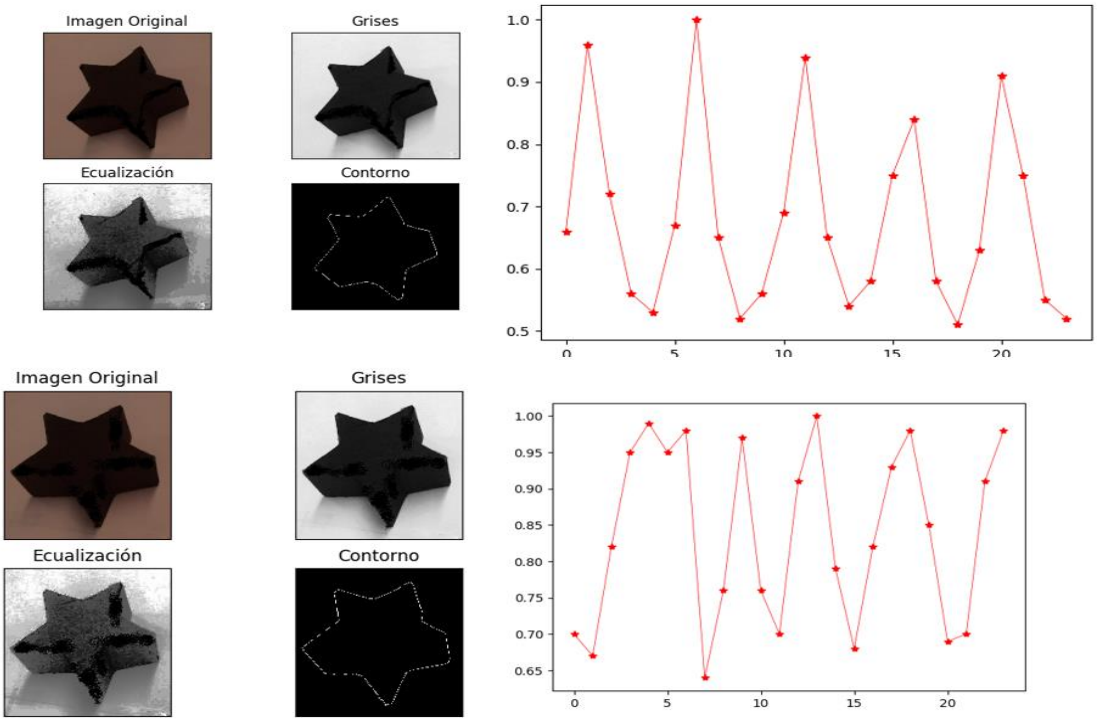


Ilustración 117 Vectores característicos de una estrella con rotación.

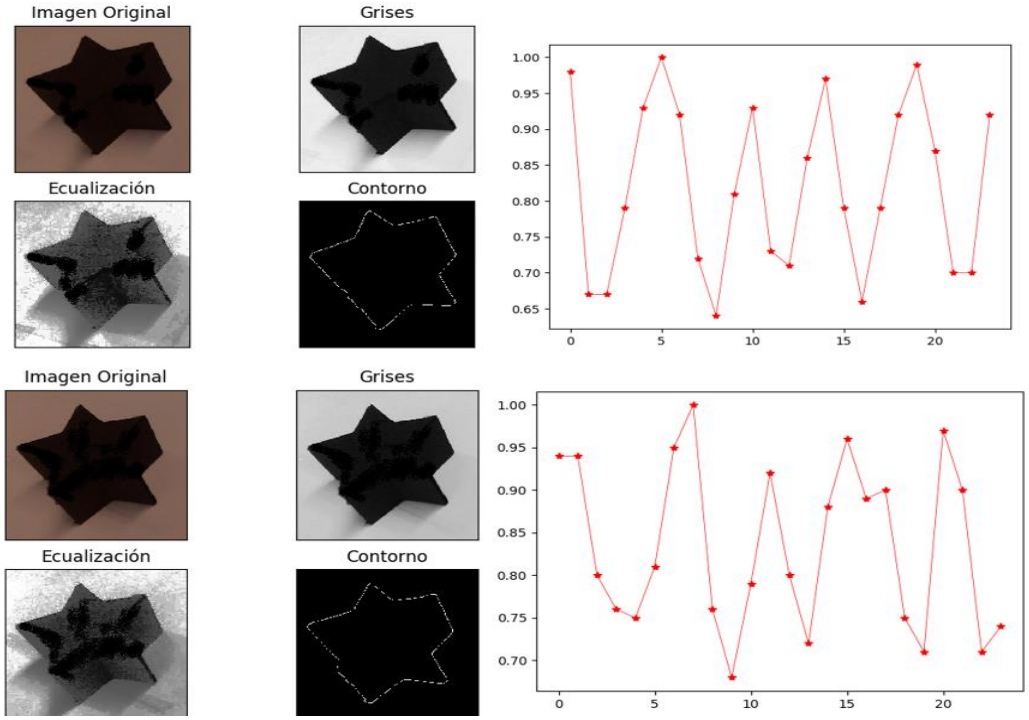


Ilustración 118 Vectores característicos de una estrella con rotación.

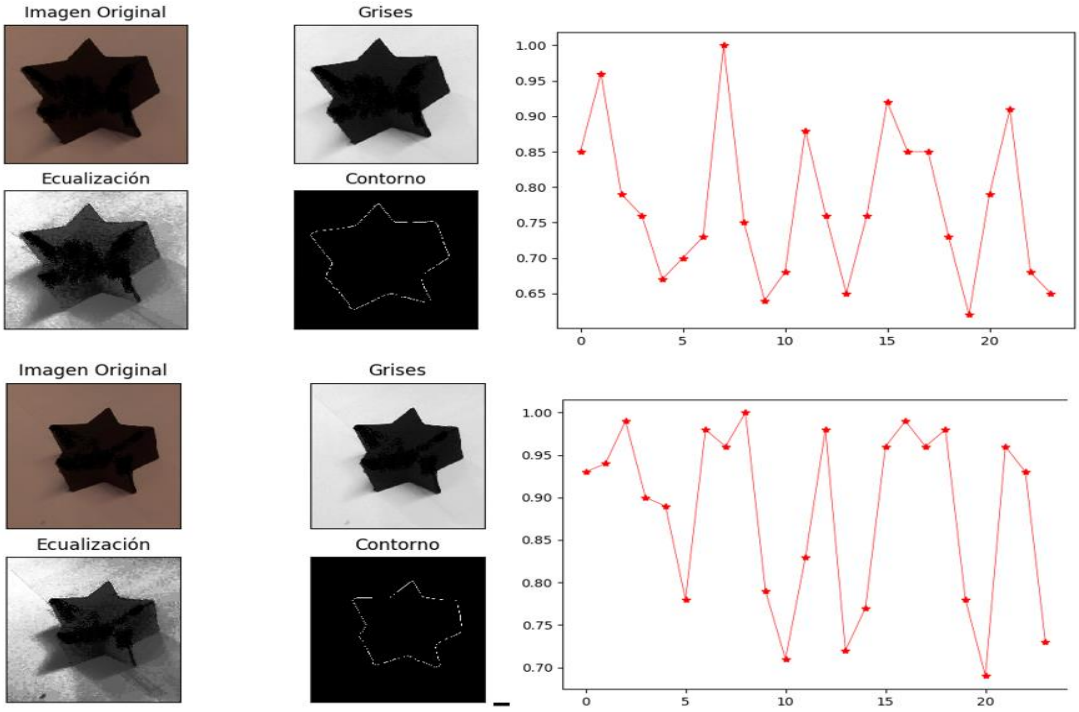


Ilustración 119 Vectores característicos de una estrella con rotación.

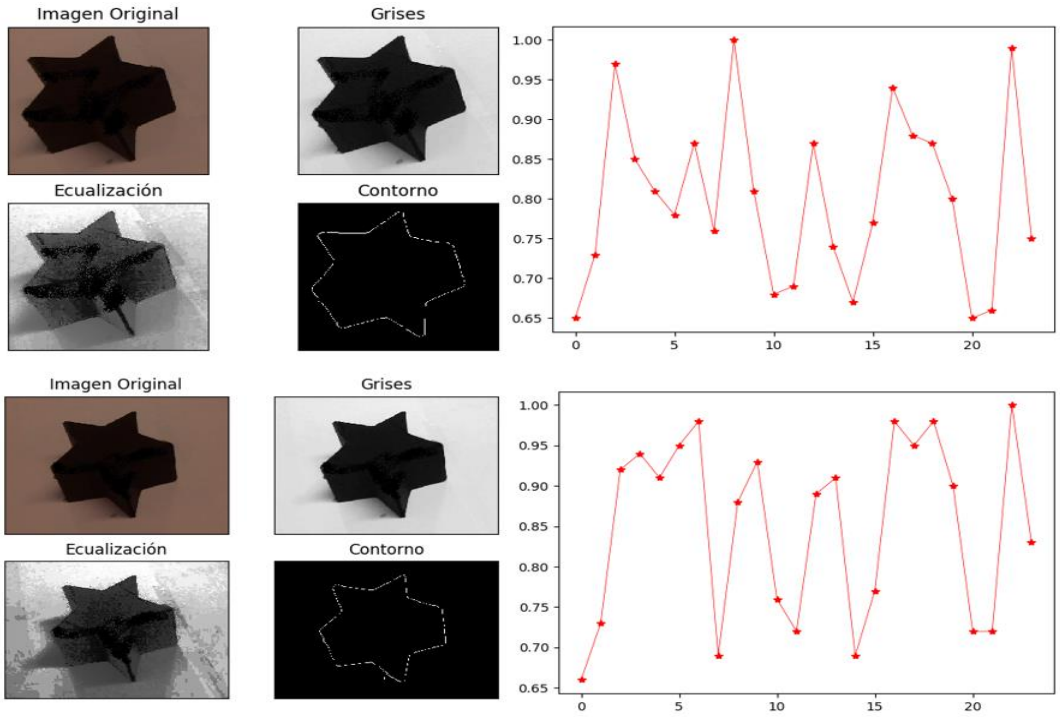


Ilustración 120 Vectores característicos de una estrella con rotación.

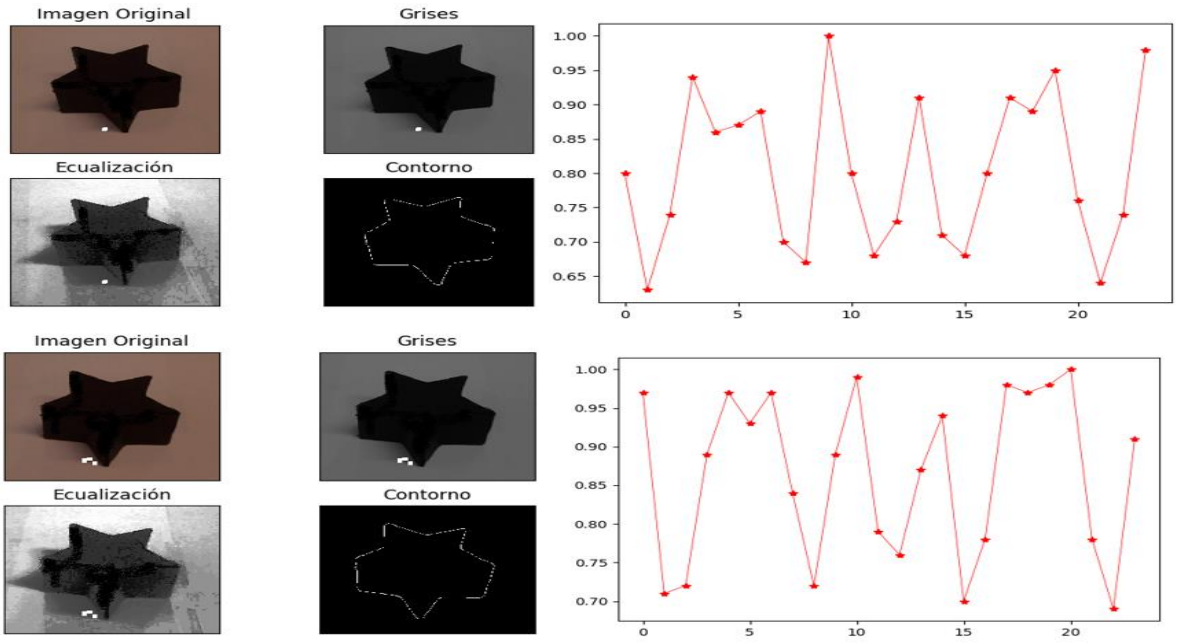


Ilustración 121 Vectores característicos de una estrella con rotación.

CRUZ

En la siguiente tabla 18, se muestra el resultado obtenido del escaneo de una cruz, mediante la inclinación de la cámara a 45°, sobre el eje Z, con referencia a la base de trabajo y un recorrido de la cámara de 0° a 90°, sobre el eje X, con referencia a la misma base de trabajo.

Rotación 0°	Rotación 10°	Rotación 20°	Rotación 30°	Rotación 40°	Rotación 50°	Rotación 60°	Rotación 70°	Rotación 80°	Rotación 90°
1	0.67	0.7	0.98	0.99	1	0.97	0.99	0.85	0.94
0.95	0.93	0.72	0.65	0.73	0.84	0.99	0.99	0.99	0.97
0.97	0.95	1	0.87	0.7	0.65	0.71	0.93	0.96	0.94
0.68	0.91	0.93	0.94	0.96	0.86	0.69	0.76	0.93	0.94
0.55	0.88	0.89	0.9	0.9	0.91	0.91	0.75	0.82	0.85
0.62	0.94	0.9	0.9	0.88	0.92	0.88	0.86	0.75	0.78
0.99	0.71	0.97	0.96	0.92	0.96	0.91	0.9	0.91	0.92
0.96	0.71	0.68	0.83	0.99	0.98	0.93	0.99	0.89	0.81
0.97	0.95	0.81	0.65	0.73	0.84	0.99	0.98	0.99	0.92
0.63	0.95	0.98	0.87	0.7	0.66	0.73	0.9	0.97	0.95
0.51	0.99	0.97	0.96	0.98	0.84	0.64	0.3	0.83	0.97
0.64	0.71	0.9	0.98	0.94	0.94	0.95	0.64	0.66	0.71
0.95	0.66	0.65	0.99	0.94	0.96	0.92	0.95	0.82	0.68
0.93	0.89	0.76	0.67	0.64	0.77	0.98	0.92	0.95	0.91
0.99	0.97	0.99	0.91	0.74	0.68	0.63	0.98	0.97	0.92
0.73	0.98	0.98	0.98	1	0.92	0.76	0.63	0.9	0.98

0.51	0.88	0.95	0.98	0.96	0.96	0.99	0.76	0.7	7
0.59	0.82	0.84	0.91	0.94	0.93	0.92	0.99	0.87	0.69
1	0.71	0.76	0.91	0.89	0.93	0.91	0.92	0.91	0.92
0.96	0.83	0.8	0.73	0.88	0.99	0.92	0.91	0.92	0.88
0.99	0.97	0.88	0.73	0.65	0.78	0.96	0.92	0.93	0.92
0.66	0.96	0.97	0.98	0.79	0.62	0.69	0.92	1	0.94
0.5	1	0.97	0.98	0.99	0.97	0.66	0.69	0.8	1
0.64	0.79	0.97	1	0.97	1	1	0.66	0.62	0.69

Tabla 18 Vectores descriptores de una cruz con rotación.

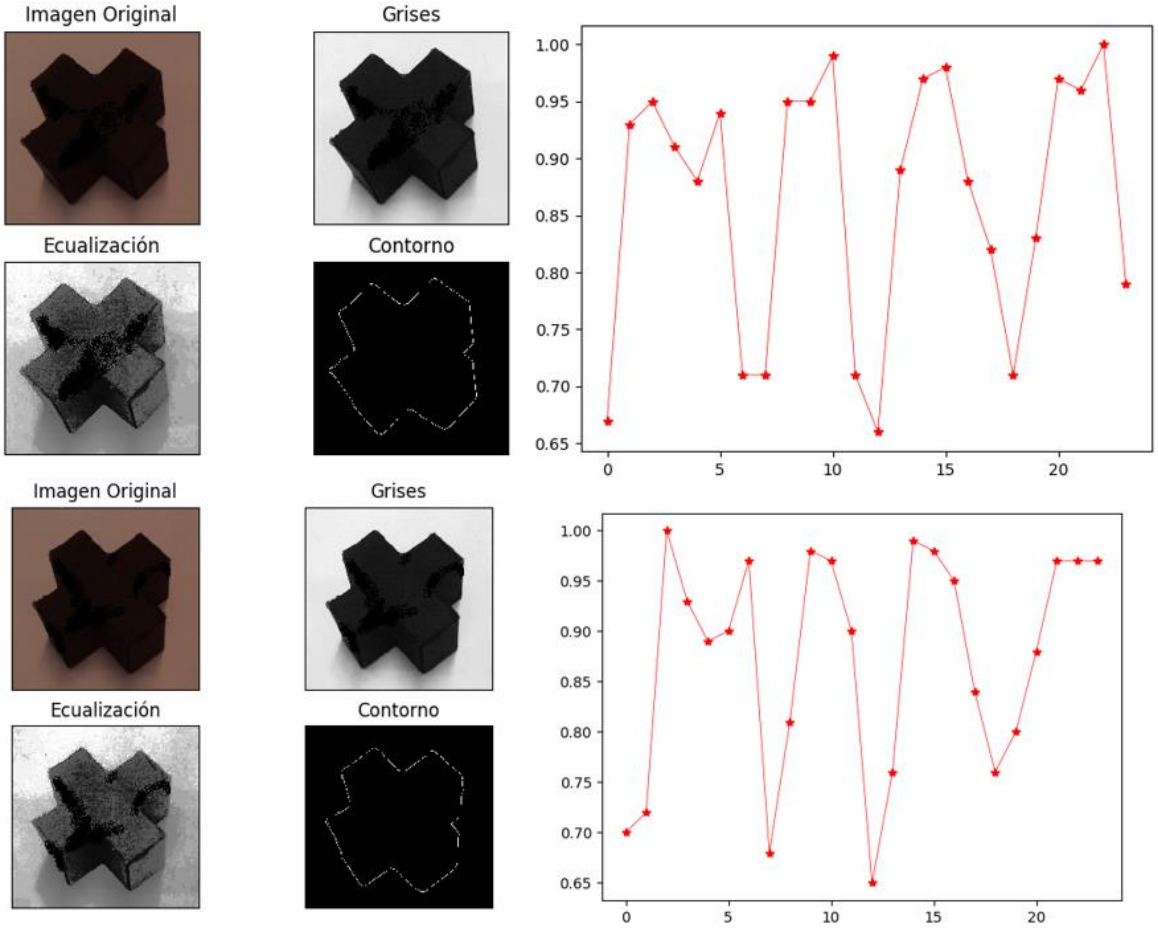


Ilustración 122 Vectores característicos de una cruz con rotación.

En las ilustraciones 122, 123, 124, 125 y 125, se puede observar el análisis del procesamiento de la imagen de una cruz y su respectivo BOF.

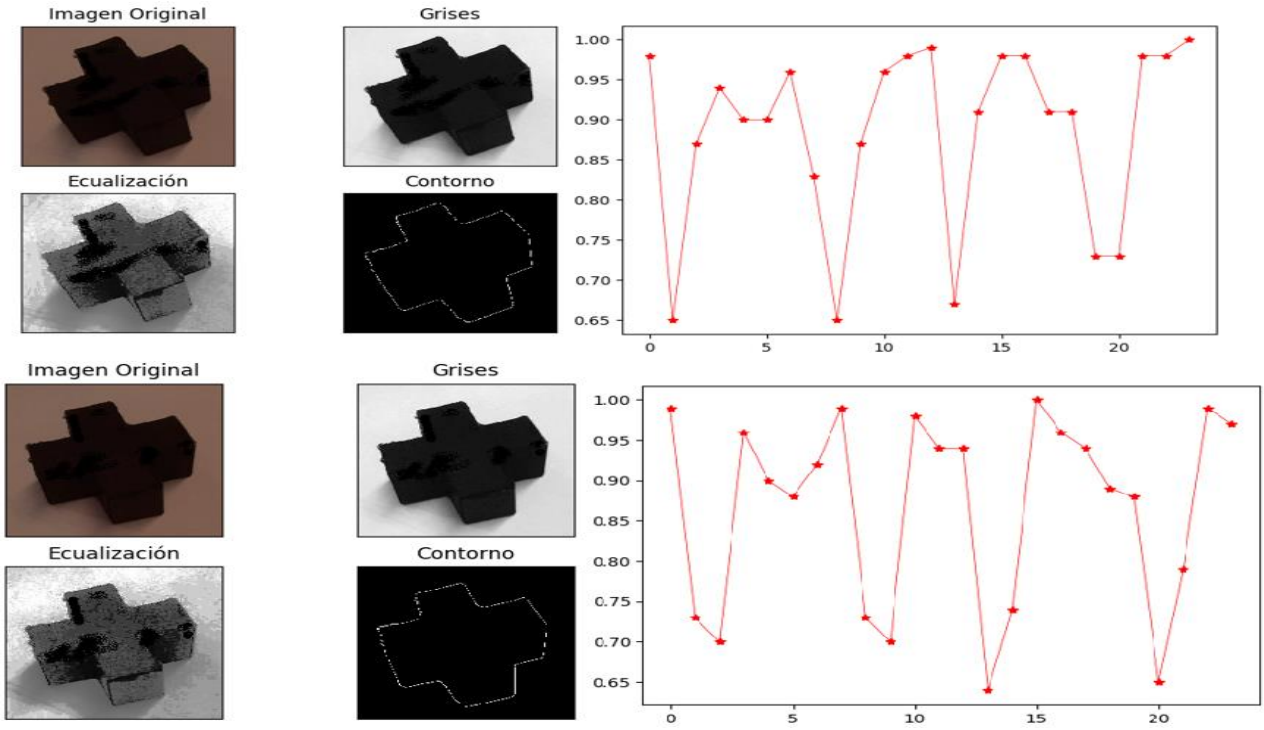


Ilustración 123 Vectores característicos de una cruz con rotación.

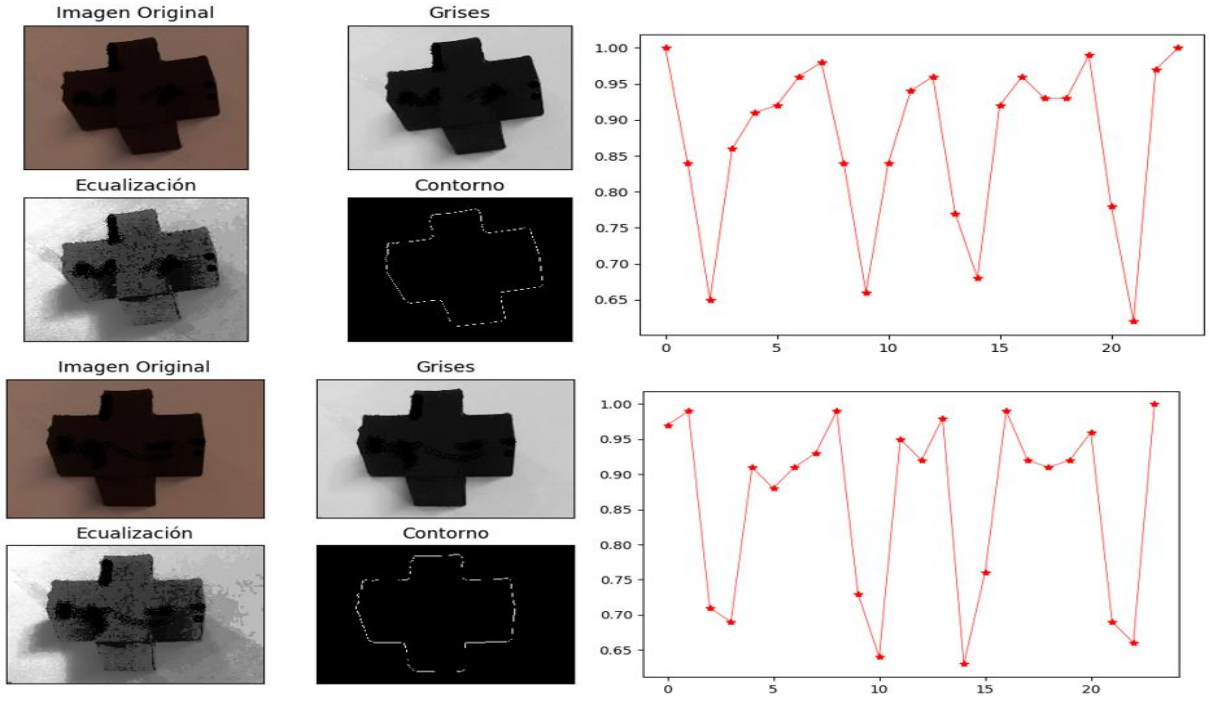


Ilustración 124 Vectores característicos de una cruz con rotación.

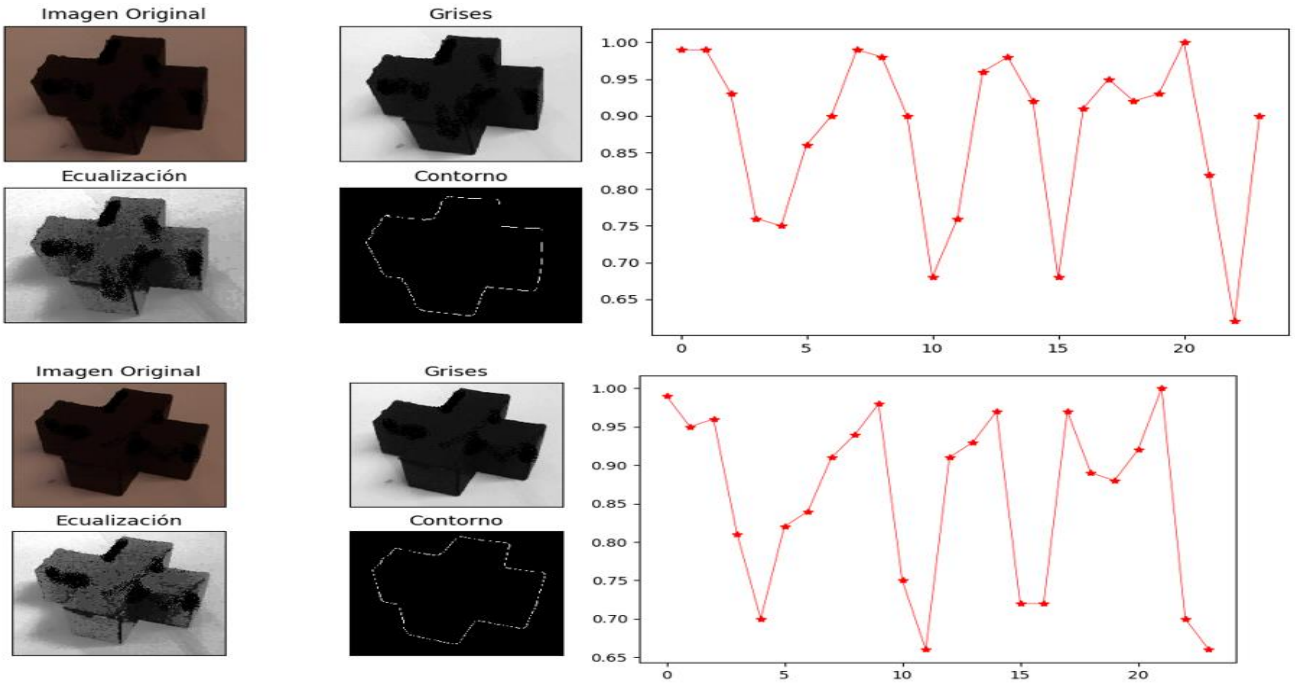


Ilustración 125 Vectores característicos de una cruz con rotación.

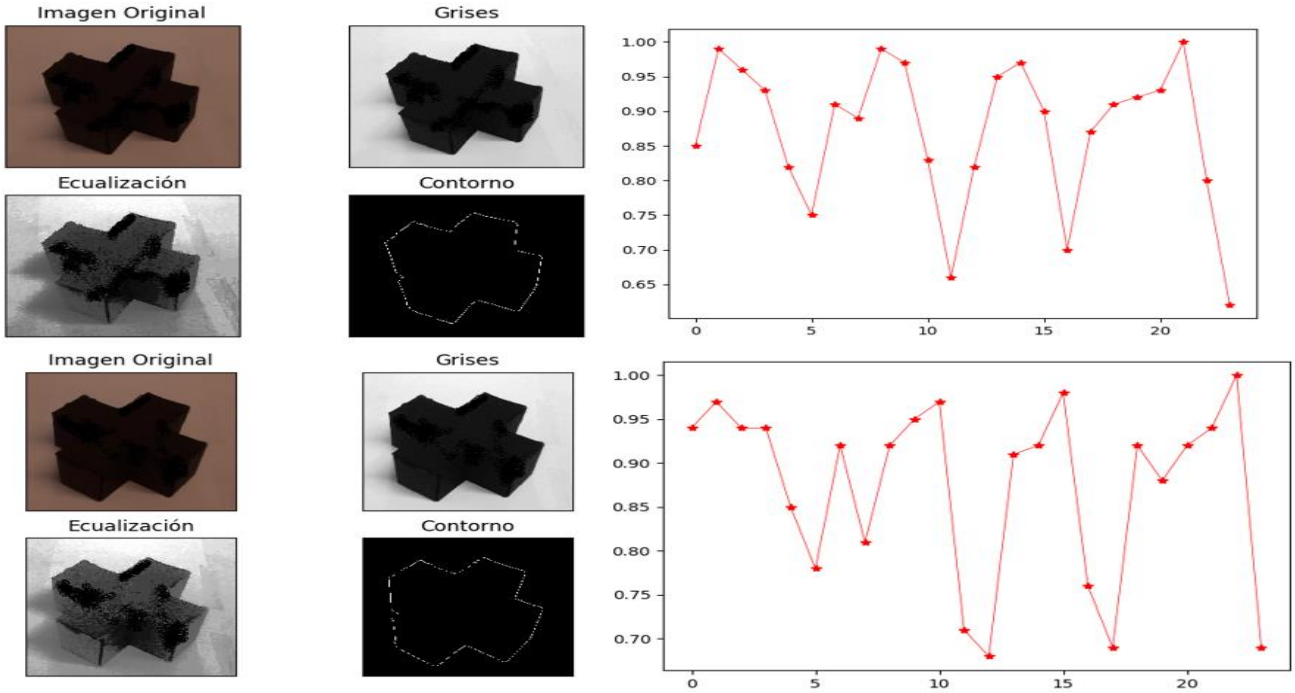


Ilustración 126 Vectores característicos de una cruz con rotación.

Triángulo

En la tabla 19, se muestra el resultado obtenido del escaneo de un prisma triangular, mediante la inclinación de la cámara a 45°, sobre el eje Z, con referencia a la base de trabajo y un recorrido de la cámara de 0° a 90°, sobre el eje X, con referencia a la misma base de trabajo.

Rotación 0°	Rotación 10°	Rotación 20°	Rotación 30°	Rotación 40°	Rotación 50°	Rotación 60°	Rotación 70°	Rotación 80°	Rotación 90°
0.73	0.95	0.91	0.84	0.97	1	1	0.74	0.77	0.66
74	0.77	0.73	0.78	0.75	0.76	0.97	1	1	0.77
0.81	0.8	0.75	0.77	0.66	0.64	0.73	0.76	0.88	1
0.64	0.74	65	0.7	0.51	0.59	0.63	0.62	0.67	0.73
0.61	0.67	0.53	0.6	0.49	0.47	0.56	0.55	0.57	0.58
0.61	0.64	0.52	0.57	0.49	0.45	0.48	0.45	0.47	0.51
0.64	0.67	0.51	0.53	0.49	0.45	0.54	0.5	0.56	0.56
0.7	0.8	0.61	0.6	0.58	0.49	0.53	0.45	0.47	0.42
0.66	0.72	0.75	0.71	0.73	0.57	0.61	5	0.51	0.45
0.65	0.69	0.7	0.75	0.87	0.75	0.79	0.6	0.63	0.52
0.69	0.71	0.68	0.72	0.76	0.74	0.88	0.87	0.91	0.65
0.81	0.79	0.72	0.73	0.72	0.66	0.77	0.73	0.86	0.86
1	0.94	0.88	0.8	0.73	0.65	0.73	0.67	0.74	0.79
0.76	0.94	1	0.95	0.79	0.68	0.74	0.65	0.7	0.66
0.64	0.74	0.7	0.88	0.93	0.77	0.81	0.67	0.71	0.65
0.59	0.65	0.63	0.67	0.68	0.74	0.89	0.76	0.78	0.58
0.57	0.63	0.52	0.59	0.57	0.56	0.68	0.71	0.8	0.77
0.6	0.63	0.5	0.55	0.52	0.49	0.57	0.56	0.64	0.68
0.64	0.67	0.51	0.53	0.49	0.45	0.54	0.5	0.56	0.56
0.62	0.65	0.5	0.52	0.52	0.49	0.53	0.46	0.53	0.51
0.69	0.7	0.58	0.56	0.62	0.57	0.55	0.47	0.53	0.49
0.86	0.8	0.7	0.65	0.66	0.61	0.65	0.53	0.59	0.5
0.85	1	0.99	0.84	0.76	0.74	0.69	0.59	0.66	0.61
0.76	0.86	0.85	1	1	0.74	0.77	0.64	0.69	0.62

Tabla 19 Vectores descriptores de un primas triangular con rotación.

En las ilustraciones 127, 128, 129, 130 y 131, se observa el análisis del procesamiento de la imagen de un prisma triangular y sus respectivos vectores descriptores.

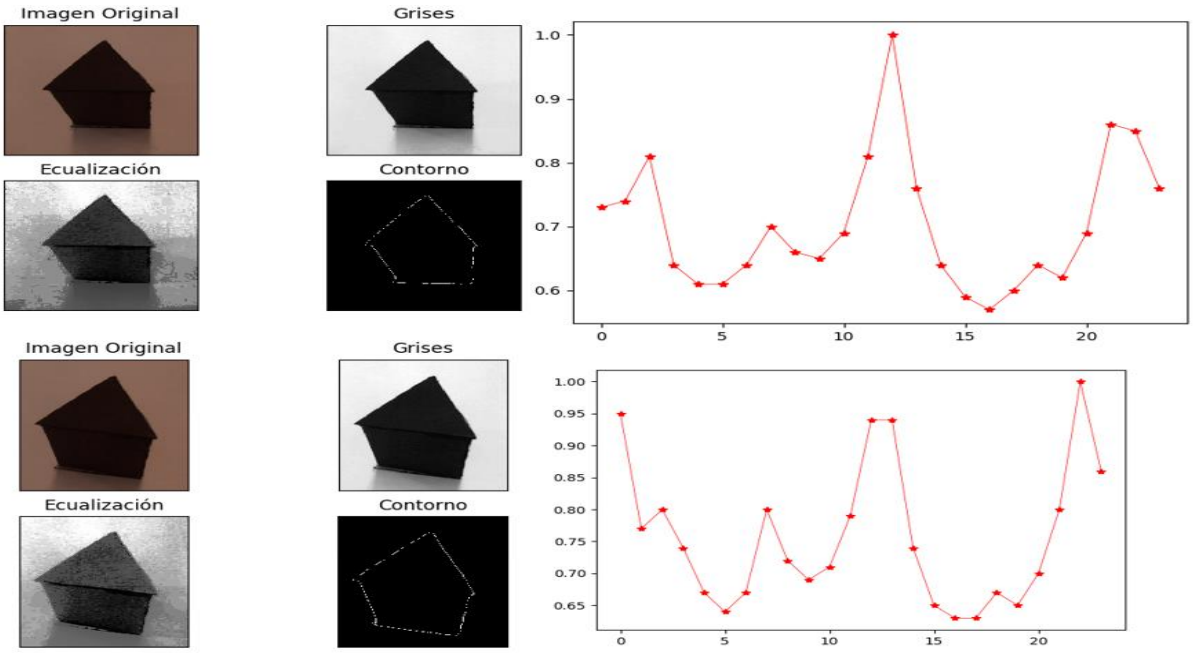


Ilustración 127 Vectores característicos de un prima triangular con rotación.

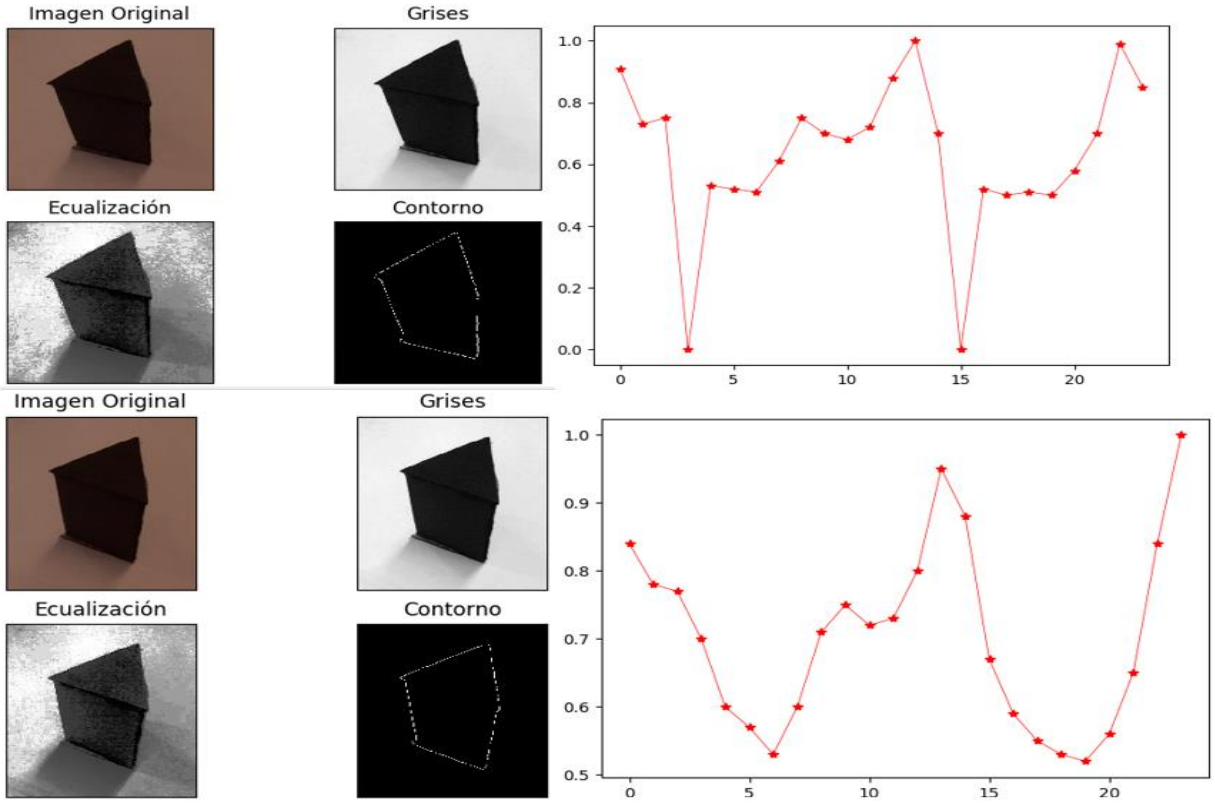


Ilustración 128 Vectores característicos de un prima triangular con rotación.

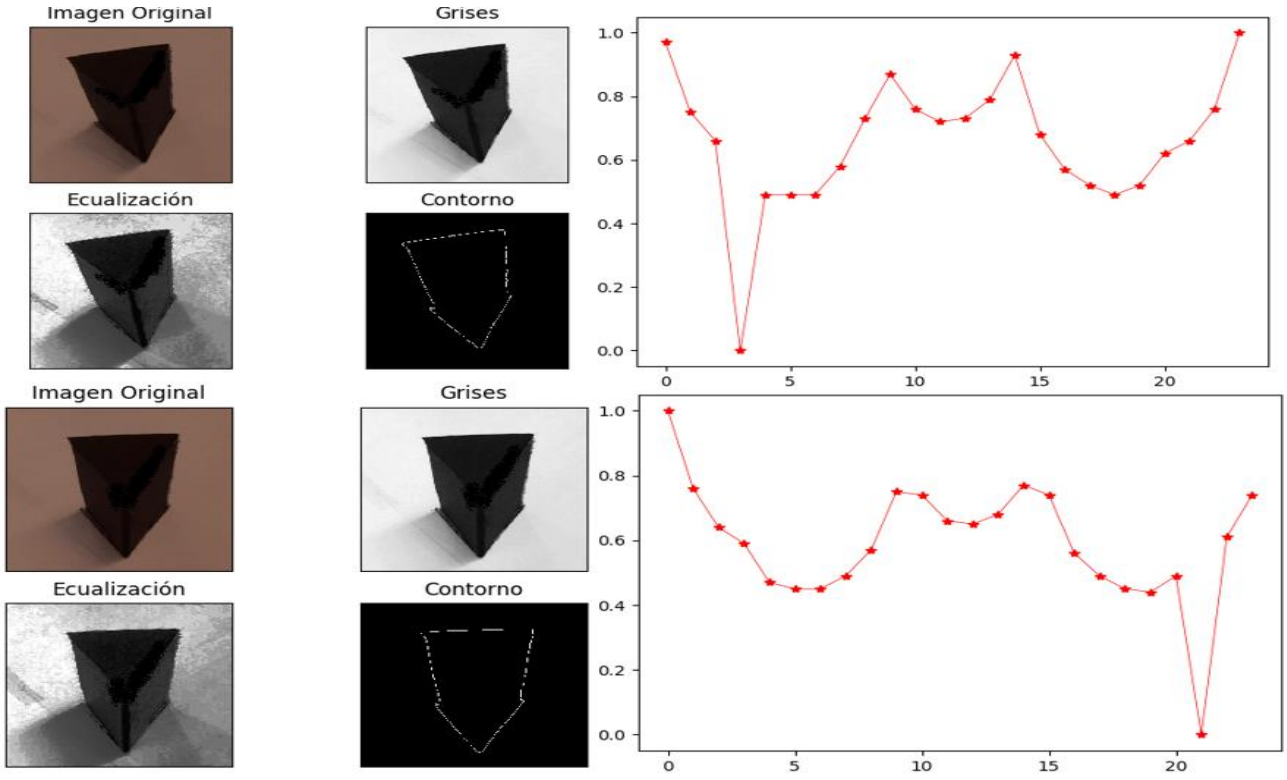


Ilustración 129 Vectores característicos de un prima triangular con rotación.

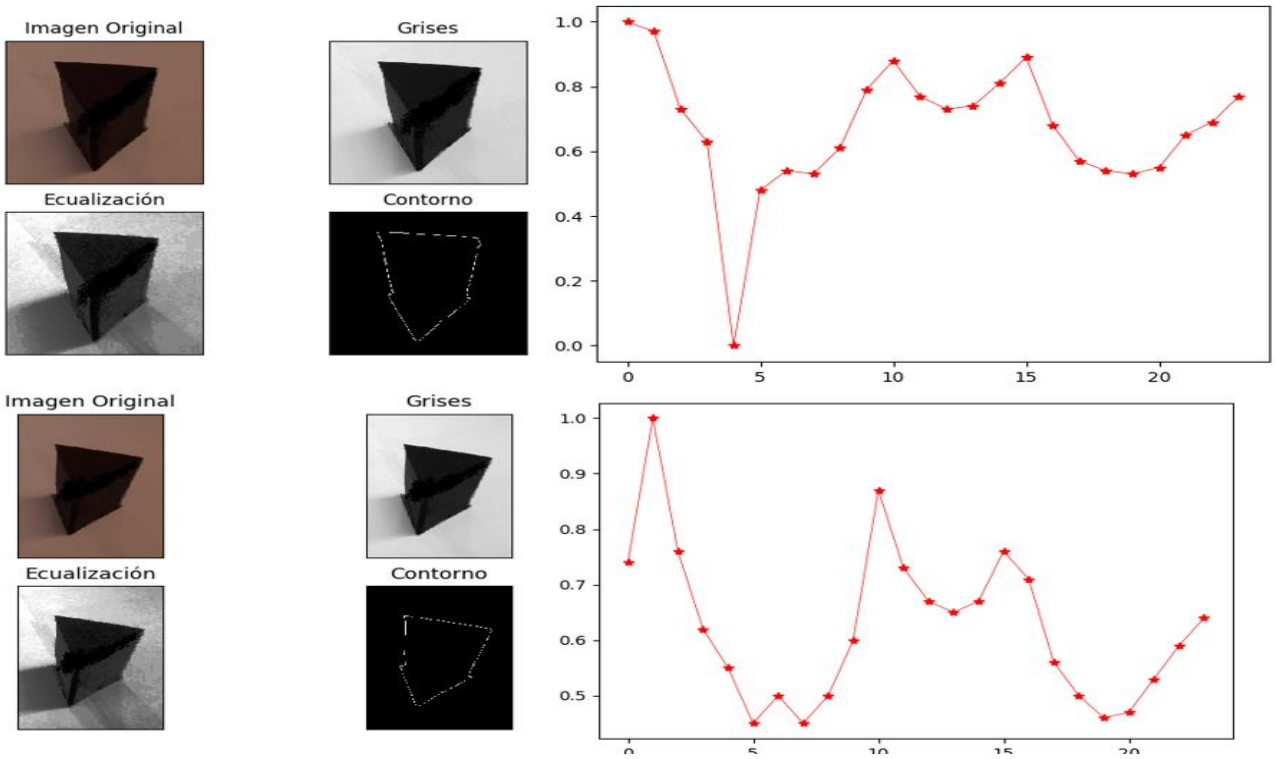


Ilustración 130 Vectores característicos de un prisma triangular con rotación.

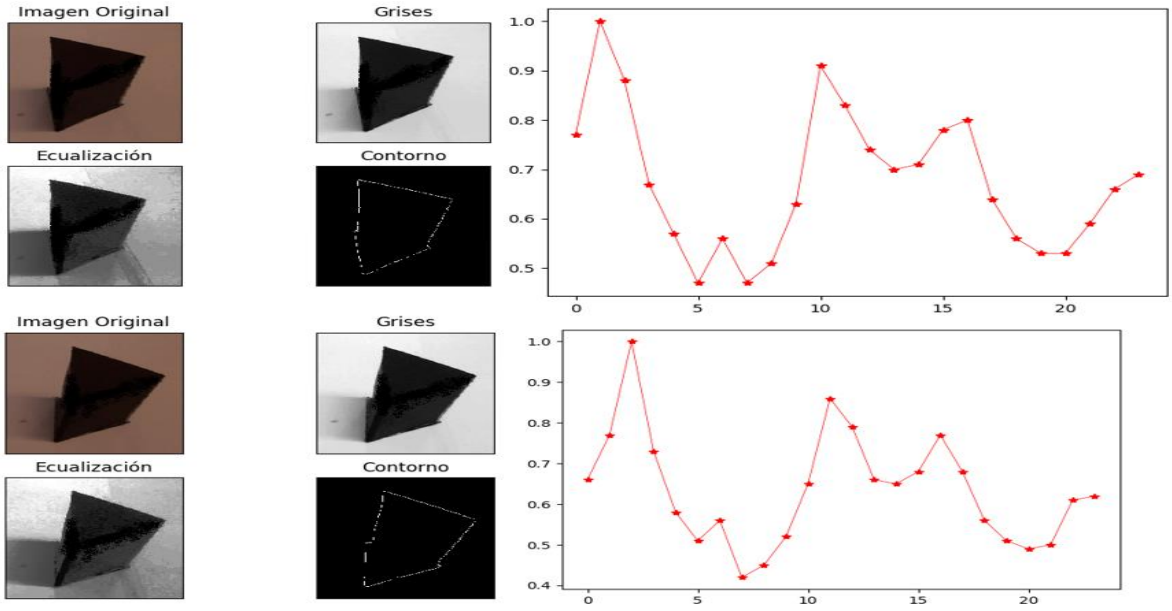


Ilustración 131 Vectores característicos de un prisma triangular con rotación.

Cilindro

En la tabla 20 se muestra el resultado obtenido del escaneo de un cilindro, mediante la inclinación de la cámara a 45° sobre el eje Z con referencia a la base de trabajo y un recorrido de la cámara de 0° a 90° sobre el eje X con referencia a la misma base de trabajo.

Rotación 0°	Rotación 10°	Rotación 20°	Rotación 30°	Rotación 40°
0.94	0.99	0.95	0.91	0.95
0.91	0.93	0.94	0.94	0.95
0.91	0.91	0.94	0.94	0.94
0.94	0.93	0.94	0.95	0.97
0.99	0.98	0.96	0.96	0.98
1	0.91	0.97	0.94	0.96
0.98	0.95	0.92	0.97	0.96
0.96	0.9	0.91	0.98	0.93
0.95	0.96	0.91	0.97	0.93
0.95	0.98	0.9	0.99	0.94
0.91	0.97	0.91	0.99	0.93
0.91	0.97	0.95	0.99	0.92
0.96	0.92	0.99	0.97	0.97
0.96	0.94	0.99	0.97	0.97

0.97	0.91	0.99	0.97	0.98
0.94	0.92	0.99	0.97	0.93
0.96	0.91	0.96	0.92	0.9
0.98	0.9	0.96	0.91	0.89
0.98	0.93	0.97	0.91	0.95
0.97	0.91	0.98	0.92	0.9
0.97	0.96	0.99	0.94	0.94
0.96	0.98	0.93	0.93	0.97
0.96	0.99	0.93	0.93	0.99
0.95	0.99	0.91	0.99	0.98

Tabla 20 Vectores descriptores de un cilindro con rotación.

En las siguientes ilustraciones 132 y 133, se observa el análisis de procesamiento de imagen de un cilindro y su respectivo vector descriptor.

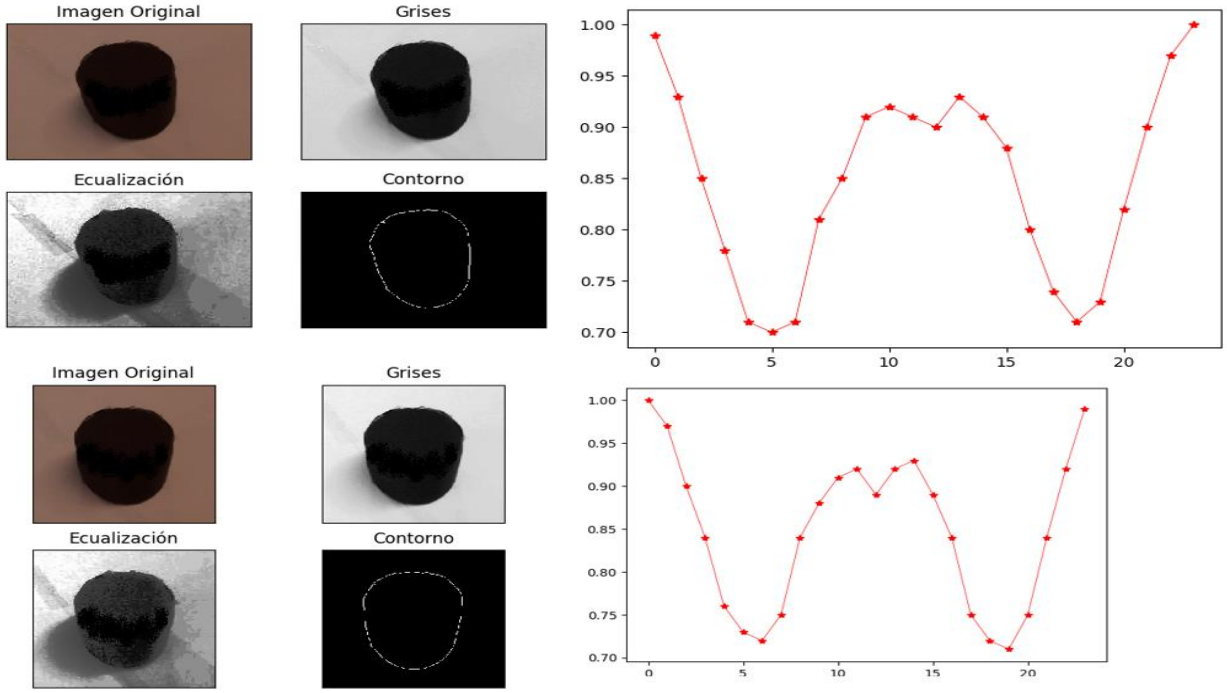


Ilustración 132 Vectores característicos de un cilindro con rotación.

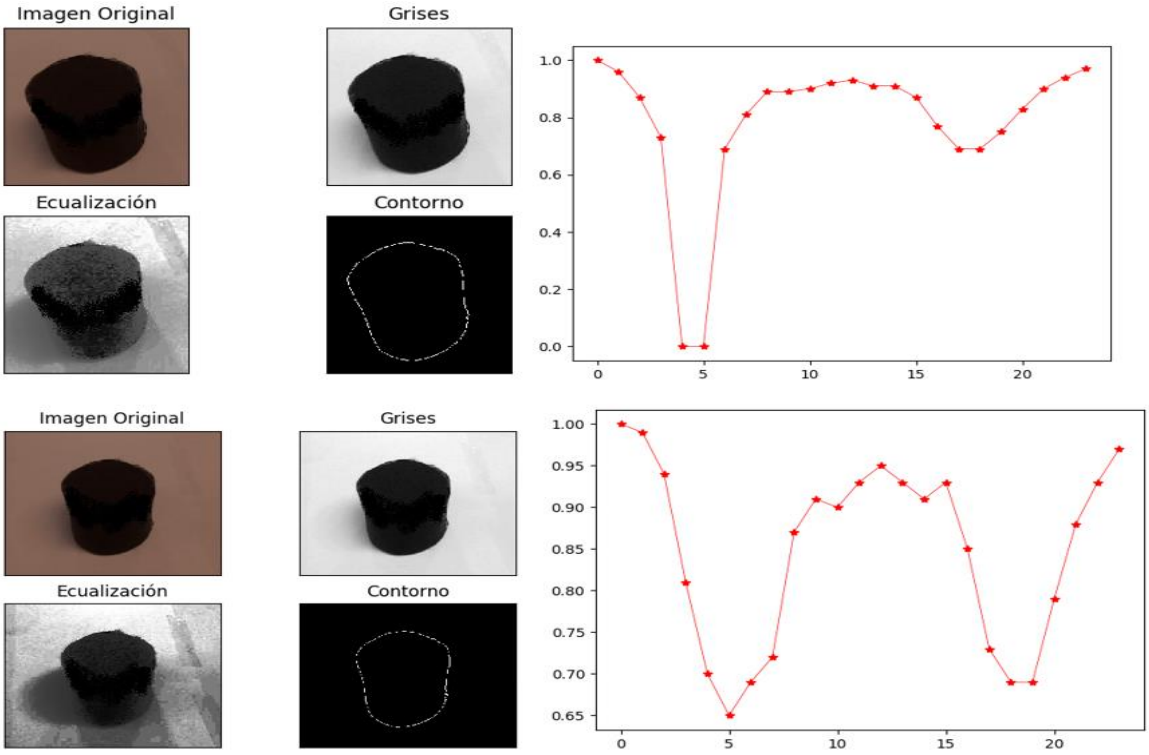


Ilustración 133 Vectores característicos de un cilindro con rotación.