



---

---

**Universidad Autónoma del Estado de México**

**Centro Universitario UAEM Texcoco**

**“Reconocimiento de objetos utilizando *OpenCV* y**

***Python* en una *Raspberry pi 2* en una tlapalería”**

## **Tesis**

Que para obtener el título de:

**Licenciado en Informática Administrativa**

Presenta:

**González Osorio Guadalupe Jonathan**

Director

**Dr. en I. S. José Sergio Ruiz Castilla**

Revisores

**M. en C. Yedid Erandini Niño Membrillo**

**Dr. en C. Adrián Trueba Espinosa**

## Oficio de digitalización


Texcoco, México a 11 de Enero de 2017.

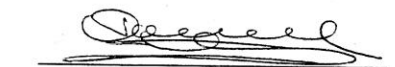
M. EN C. E. VIRIDIANA BANDA ARZATE  
SUBDIRECTORA ACADEMICA DEL  
CENTRO UNIVERSITARIO UAEM TEXCOCO  
PRESENTE:

AT'N L. EN D. MARCO RODRIGO LÓPEZ GONZÁLEZ  
RESPONSABLE DEL DEPARTAMENTO DE TITULACION.

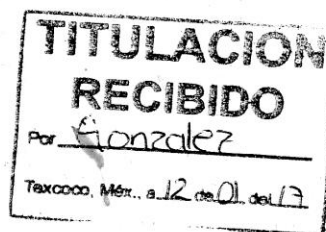
Con base en las revisiones efectuadas al trabajo escrito titulado "Reconocimiento de objetos utilizando OpenCV y Python en una Raspberry pi 2 en una tlapalería" que para obtener el título de Licenciado en Informática Administrativa presenta el sustentante González Osorio Guadalupe Jonathan, con número de cuenta 1124675 respectivamente, se concluye que cumple con los requisitos teórico-metodológicos por lo que se le otorga el voto aprobatorio para su sustentación, pudiendo **continuar con la etapa de digitalización** del trabajo escrito

ATENTAMENTE

  
FIRMA  
Dr. en C. Adrián Trueba Espinosa

  
FIRMA  
M. en C. Yedid Erandini Niño Membrillo

  
FIRMA  
Dr. en I. S. José Sergio Ruiz Castilla



c.c.p González Osorio Guadalupe Jonathan  
c.c.p Director Dr. en I. S. José Sergio Ruiz Castilla  
c.c.p Titulación L. en D. Marco Rodrigo López González

## **Agradecimientos**

A mi mamá por todo el apoyo y esfuerzo dedicado para mi desarrollo profesional y personal.

A mi hermano por ser un apoyo incondicional en todo momento.

A mi director de tesis Dr. en I. S. José Sergio Ruiz Castilla por la dedicación que invirtió en el desarrollo de este proyecto.

A los revisores M. en C. Yedid Erandini Niño Membrillo y Dr. en C. Adrián Trueba Espinosa por la aportación de sus conocimientos para la mejora de este proyecto.

# Índice

## Tabla de contenido

Oficio de digitalización.....	I
Agradecimientos.....	II
Índice.....	III
Índice de figuras.....	IV
Resumen.....	V
1 Capítulo 1 Introducción.....	1
1.1. Introducción .....	1
1.2. Planteamiento del problema .....	2
1.3. Justificación.....	2
1.4 Objetivos .....	3
1.4.1 Objetivo general .....	3
1.4.2 Objetivos específicos.....	3
2 Capítulo 2: Herramientas utilizadas y actividades de una tlapalería .....	4
2.1 Introducción .....	4
2.2 <i>Raspberry Pi 2</i> .....	4
2.3 <i>OpenCV</i> .....	6
2.3.1 Estructura .....	7
2.3.2 Funciones de <i>OpenCV</i> .....	8
2.4 <i>Python</i> .....	11
2.4.1 Lenguaje interpretado o de script .....	11
2.5 <i>Adobe Photoshop</i> .....	13
2.6 Ciclo de vida en cascada .....	14
2.7 Concepto y productos de una tlapalería .....	15
2.7.1 Modelos de negocio de una tlapalería .....	16
2.7.2 Proceso de venta de un artículo en específico .....	17
3 Capítulo 3 Imágenes digitales .....	18
3.1. Conceptos básicos del procesamiento digital de imágenes .....	18
3.1.1 Proceso de captación y formación de una imagen.....	19

3.2	Modelos de representación.....	21
3.2.1	Mapa de bits .....	21
3.2.2	Gráfico vectorial.....	22
3.3	Principales aplicaciones .....	23
3.4	Problemática en el análisis automatizado de imágenes .....	24
3.5	Textura .....	25
3.6	Clasificadores .....	27
3.7	Introducción a la edición paramétrica de imágenes .....	29
3.8	Paradigma en el análisis de imágenes .....	30
3.8.1	Etapa 1: Captación de la Imagen .....	32
3.8.2	Etapa 2: Pre procesamiento: .....	34
3.8.2.1	Filtros .....	37
3.9	Etapa 3: Segmentación .....	44
3.9.1	Segmentación por el clasificador de Bayes .....	46
3.9.2	Formula de Bayes aplicada a la segmentación supervisada .....	46
3.9.3	Detección de bordes .....	48
3.9.4	Operador de Canny.....	50
3.10	Técnicas actuales de segmentación .....	51
3.10.1	Modelos deformables .....	51
3.11	Etapa 4: Representación .....	52
3.11.1	Modelos en dos dimensiones.....	54
3.11.2	Polilíneas .....	54
3.12	Etapa 5: Descripción .....	55
3.12.1	Obtención de características del objeto por color.....	55
3.12.2	Modelo sensorial .....	56
3.12.3	Modelo perceptual.....	56
3.12.4	Extracción de puntos .....	57
3.12.5	Puntos característicos a partir de aristas: puntos de fuga. ....	58
3.12.6	Puntos característicos a partir de la propia imagen: puntos esquina. ....	59
3.13	Extracción de líneas .....	60
3.13.1	Transformada de Hough para líneas.....	61
3.14	Extracción de círculos .....	62

3.15	Etapa 6: Reconocimiento .....	63
3.15.1	Clasificador “K Vecinos más Próximos” .....	63
4	Capítulo 4. Técnicas de tratamiento de imágenes .....	64
4.1	<i>Sift</i> .....	64
4.2	<i>Surf</i> .....	69
4.3	<i>Orb</i> .....	74
4.4	<i>Freak</i> .....	74
4.5	<i>Flann</i> .....	75
5	Capítulo 5. Metodología.....	78
5.1	Propuesta de solución.....	78
5.1.1	Ciclo de vida en cascada .....	78
5.1.1.1	Etapa 1 Análisis de requerimientos.....	78
5.1.1.2	Etapa 2 Diseño .....	82
5.1.1.3	Etapa 3 Implementación.....	87
5.1.1.4	Etapa 4 Integración.....	91
6	Capítulo 6 Resultados.....	92
6.1	Etapa 5 Pruebas .....	92
7	Capitulo 7 Conclusiones.....	105
7.1	Temas futuros .....	105
7.2	Referencias .....	106

## Índice de figuras

<i>Figura 1: Estructura Raspberry pi 2 (raspberrypi, 2016)</i> .....	5
<i>Figura 2: Estructura de la librería OpenCV. (Arevalo, Gonzalez, &amp; Ambrosio, 2016)</i> .....	7
<i>Figura 3: Funciones de OpenCV (Ruiz, 2015)</i> .....	8
<i>Figura 4: Modelo en cascada. (Braude, 2003)</i> .....	15
<i>Figura 5: Proceso de venta de un artículo (Giraldo, Roldan, &amp; Garro, 2009)</i> .....	17
<i>Figura 6: Modelo de formación de la imagen digital (Rodríguez &amp; Sossa, 2012)</i> .....	20
<i>Figura 7: Etapas de un sistema de visión por computadora (García F. , 2009)</i> .....	30
<i>Figura 8: Pasos en un sistema de análisis de imágenes incorporando una base de conocimientos sobre el problema a resolver. (Rodríguez &amp; Sossa, 2012)</i> .....	32
<i>Figura 9: Histograma de una imagen. (Escalante, 2006)</i> .....	36
<i>Figura 10: Tipos de histograma de una imagen. (Escalante, 2006)</i> .....	36
<i>Figura 11: Histograma de una imagen digital (Escalante, 2006)</i> .....	37
<i>Figura 12: Imagen original y con ruido. (Escalante, 2006)</i> .....	38
<i>Figura 13: Cambios en la señal de una imagen. (Escalante, 2006)</i> .....	39
<i>Figura 14: Señal original y sobremuestreada (Escalante, 2006)</i> .....	40
<i>Figura 15: Transformaciones lineales (Sucar &amp; Gomez, 2016)</i> .....	41
<i>Figura 16: Selección de propiedades. (Ruiz, 2015)</i> .....	43
<i>Figura 17: Selección individual (Ruiz, 2015)</i> .....	43
<i>Figura 18: Esquema que representa el proceso de segmentación. (Rodríguez &amp; Sossa, 2012)</i> .....	45
<i>Figura 19: Reconocimiento de caracteres en base a su codificación radial. (Sucar &amp; Gomez, 2016)</i> .....	53
<i>Figura 20: Estructura de un sistema de visión basado en modelos. (Sucar &amp; Gomez, 2016)</i> .....	54
<i>Figura 21: Obtención de características del objeto por color. (Flores, 2016)</i> .....	55
<i>Figura 22: Intersección de rectas. (González, 2016)</i> .....	58
<i>Figura 23: Método de la minimización del área del triángulo. (González, 2016)</i> .....	59
<i>Figura 24: Clasificador KNN. (García E. , 2008)</i> .....	64
<i>Figura 25: Formación de la Diferencia de Gaussianas (DOG). (Juárez, 2011)</i> .....	68
<i>Figura 26: Algoritmo SURF. (Patiño, 2012)</i> .....	69
<i>Figura 27: Esquema del proceso de clasificación SURF. (Aracil, 2012)</i> .....	71
<i>Figura 28: Se puede observar la representación de la derivada parcial de segundo orden de un filtro gaussiano discretizado y la aproximación de la derivada implementada en el caso del descriptor SURF. (Aracil, 2012)</i> .....	73
<i>Figura 29: Algoritmo ORB, Definición del Patrón. (Espitia, 2014)</i> .....	74
<i>Figura 30: Algoritmo FREAK, Patrón de muestreo (Espitia, 2014)</i> .....	75
<i>Figura 31: Algoritmo de búsqueda del vecino más cercano. (Bueno, 2012)</i> .....	77
<i>Figura 32: Diseño del sistema de reconocimiento de objetos (Elaboración propia)</i> .....	82
<i>Figura 33: Diagrama de casos de uso (Elaboración propia)</i> .....	83
<i>Figura 34: Diagrama de componentes (Elaboración propia)</i> .....	84
<i>Figura 35: Proceso de incorporación de un objeto al banco de imágenes. (Elaboración propia)</i> .....	85
<i>Figura 36: Espacio físico. (Elaboración propia)</i> .....	86
<i>Figura 37: Proceso de identificación del objeto. (Elaboración propia)</i> .....	87
<i>Figura 38: Generación de descriptores</i> .....	88

<i>Figura 39: Búsqueda de objeto</i> .....	90
<i>Figura 40: Prueba del sistema de reconocimiento de objetos</i> .....	91
<i>Figura 41: Banco de imágenes</i> .....	93
<i>Figura 42: Banco de imágenes procesado</i> .....	94
<i>Figura 43: Espacio físico de captación de la imagen del objeto</i> .....	95
<i>Figura 44: Proceso de captación de la imagen del objeto con raspberry pi cámara</i> .....	95
<i>Figura 45: Instalación eléctrica</i> .....	95
<i>Figura 46: Prueba de generación de descriptores</i> .....	96
<i>Figura 47: Accediendo al ambiente virtual</i> .....	97
<i>Figura 48: Segunda prueba de generación de descriptores</i> .....	97
<i>Figura 49: Imagen del objeto del banco de imágenes</i> .....	98
<i>Figura 50: Resultados de la ejecución del script para generar descriptores</i> .....	99
<i>Figura 51: Clavo</i> .....	99
<i>Figura 53: Generación del descriptor del objeto a buscar</i> .....	100
<i>Figura 52: Clavo procesado</i> .....	100
<i>Figura 54: Incorporación del descriptor de la imagen de entrada al banco de imágenes</i>	101
<i>Figura 55: Resultado de búsqueda de la imagen de entrada</i> .....	101
<i>Figura 56: Pantalla de error de segmentación de imagen</i> .....	102
<i>Figura 57: Pantalla de error de imagen</i> .....	103
<i>Figura 58: Dimensiones de la imagen de entrada</i> .....	103
<i>Figura 59: Imagen original segmentada por el editor e Imagen retocada posteriormente.</i> .....	104
<i>Figura 60: Resultados de la búsqueda de la imagen de entrada.</i> .....	104



## **Resumen**

Con el paso del tiempo la tecnología va teniendo un auge rápidamente, lo que da origen a la creación de nuevas tecnologías que son de gran utilidad en la vida diaria de las personas.

Estas nuevas ideas son de gran utilidad en el ámbito informático debido a que el auge en la tecnología trae consigo la creación de nuevos sistemas, aplicaciones entre otras cosas, las cuales al hacer uso de ellas conlleva a la creación de nuevos conocimientos.

En este contexto se presenta un sistema de reconocimiento de objetos basado en el algoritmo *FLANN* y *SIFT* con ayuda de una *Raspberry pi 2*.

El sistema fue puesto a prueba con un banco de imágenes almacenadas previamente realizando una comparación entre las imágenes almacenadas y una imagen de entrada, mostrando buenos resultados al momento de clasificar y reconocer objetos.

# 1 Capítulo 1 Introducción

## 1.1. Introducción

Un problema clásico sucede en tiendas que venden herramientas, refacciones o componentes cuyos nombres podrían ser difíciles de aprender y recordar, por lo que, los clientes llevan físicamente un objeto y piden uno nuevo. Por otro lado, puede haber empleados de las tiendas que tampoco saben los nombres de los artículos. Por lo anterior, se planteó resolver el problema para reconocer objetos por computadora.

El objetivo del trabajo consiste en, reconocer objetos a partir de una imagen que puede ser una fotografía. Con la idea de que, se proporcione la imagen al sistema y pueda reconocer el objeto para darnos el nombre, existencia, precio, ubicación, etc.

Se partió de un conjunto de imágenes obtenidas de una cámara fotográfica, además de estandarizar las imágenes en tamaño de pixeles y establecer en tonos grises. Una vez pre-procesada la imagen se aplica el reconocimiento del objeto usando una computadora *Raspberry Pi 2* y la librería *OpenCV*. Adicionalmente, se agregan programas de *Python*.

Se probó con 20 imágenes de las cuales reconoció 15 y las otras 5 fue necesario aplicar un pre-procesamiento adicional como cambiar el tamaño. Agregar que, se estableció el 80% de grado de similitud para el reconocimiento de objetos. Al final se lograron reconocer los 20 objetos.

Se concluyó que mediante el uso de una computadora *Raspberry Pi 2* y la librería *OpenCV* es posible el reconocimiento de objetos con la posibilidad de usarse en tiendas de partes o artículos que se venden en refaccionarias, tlapalerías, etc.

## **1.2.Planteamiento del problema**

Un problema que se presenta de las tlapalerías, entre otros tipos de negocios similares, sucede cuando un cliente se presenta con un objeto que desea comprar, por lo que, lo muestra en la tlapalería sin conocer el nombre y especificaciones del mismo, por otro lado, el dependiente del mostrador también podría desconocer dicho objeto. Lo anterior, obliga a una búsqueda comparando la pieza con las existentes en todo el almacén con una considerable pérdida de tiempo y posible fracaso.

Cuando el cliente llega a la sucursal y lleva una muestra del producto que desea adquirir, en la mayoría de las veces, la tlapalería no ha podido ayudar al cliente a encontrar lo que busca, debido a que, tanto el trabajador como el cliente ignoran el nombre del artículo buscado.

## **1.3.Justificación**

Con la implementación de dicho sistema la tlapalería podría obtener un ahorro de tiempo, capital y, a la vez, salvaguardar la retención de sus clientes para darle una mejor atención al momento en que realicen sus compras, otorgándoles productos específicamente que ellos estaban buscando.

Con la implementación de este sistema de reconocimiento digital de imágenes, los clientes podrán llevar algún producto que deseen adquirir ya sea para reemplazo o bien para conocer y observar si la tlapalería cuenta con cierto material.

Cabe mencionar que, el uso de este tipo de sistemas no solo beneficia a microempresas de este tipo, sino también, contribuye a otros tipos de negocio, como tiendas de herramienta especializadas, material para hospitales, material de electrónica, entre otro tipo de negocio.

Por otra parte, esto conlleva a una mayor atención personalizada a los clientes debido a que las imágenes de los productos almacenados en una base de datos acorde a la empresa, dando lugar a mayores innovaciones tecnológicas, trayendo consigo, la generación de ventajas competitivas sobre las demás empresas.

## **1.4 Objetivos**

### **1.4.1 Objetivo general**

Desarrollar un sistema de reconocimiento de objetos para buscar y en su caso encontrar un objeto dentro de un conjunto de objetos almacenados como imágenes digitales de 500 x 500 pixeles.

### **1.4.2 Objetivos específicos**

1. Definir el tipo de imagen y formato para el catálogo de artículos de la tlapalería
2. Crear un catálogo de imágenes de artículos de la tlapalería, con la posibilidad de agregar objetos nuevos.
3. Desarrollar un algoritmo que permita identificar una imagen, a partir de, una fotografía.
4. Generar las pruebas necesarias y obtener la mayor eficiencia posible.
5. Implementar el sistema.

### **1.4. Hipótesis**

Si se implementa un sistema de reconocimiento de objetos en las tlapalerías será posible reconocer objetos a partir de una fotografía así como su existencia y características del objeto buscado.

## 2 Capítulo 2: Herramientas utilizadas y actividades de una tlapalería

### 2.1 Introducción

La inteligencia artificial ha evolucionado en todas sus áreas. En este caso, la visión artificial no es la excepción. En los últimos tiempos, esta área permite diversas aplicaciones entre ellas el reconocimiento de objetos, huellas dactilares y rostros, entre otros.

### 2.2 *Raspberry Pi 2*

Según (raspberrishop, 2016) *Raspberry Pi* es una computadora de placa simple (SBC) de bajo costo desarrollada en Reino Unido por la Fundación *Raspberry Pi*, con el objetivo de estimular la enseñanza de ciencias de la computación en las escuelas.

En realidad, se trata de una diminuta placa base de 85 x 54 milímetros (del tamaño aproximado de una tarjeta de crédito) en el que se aloja un chip *Broadcom BCM2835* con procesador *ARM* hasta a 1 *GHz* de velocidad (modo Turbo haciendo *overclock*), *GPU VideoCore IV* y 512 *Mbytes* de memoria *RAM* (Las primeras placas contaban con sólo 256MB de *RAM*). Las últimas placas como la *Raspberry Pi 2* y *Raspberry Pi 3* tienen un 1GB de memoria *RAM*.

Para que funcione, se necesita de un medio de almacenamiento (*Raspberry Pi* utiliza tarjetas de memoria *SD* o *microSD* dependiendo del modelo), conectarlo a la corriente utilizando cualquier cargador *microUSB* de al menos 1000mah para las placas antiguas y de al menos 2500mah para las modernas, y si se desea, guardarlo todo utilizando una carcasa para que todo quede a buen recaudo y su apariencia sea más estética.

En función del modelo que se elija, se dispone de una o más opciones de conexión, aunque siempre se dispone de al menos un puerto de salida de video *HDMI* y otro de tipo *RCA*, *minijack* de audio y un puerto *USB 2.0* (modelos A y A+, el modelo B dispone de dos *USB* y B+, *Raspberry Pi 2* y *Raspberry Pi 3* disponen de 4 *USB*) al que conectar un teclado y ratón.

En cuanto a la conexión de red, se dispone de un puerto *Ethernet* (los modelos A y A+ no disponen de puerto *Ethernet*) para enchufar un cable RJ-45 directamente al *router* o se puede recurrir a utilizar cualquier adaptador inalámbrico *WiFi* compatible. En este caso, eso sí, se debe decidir por la *Raspberry Pi model B* que incorpora dos puertos *USB*, ya que de lo contrario, no se puede conectar el teclado y el ratón.

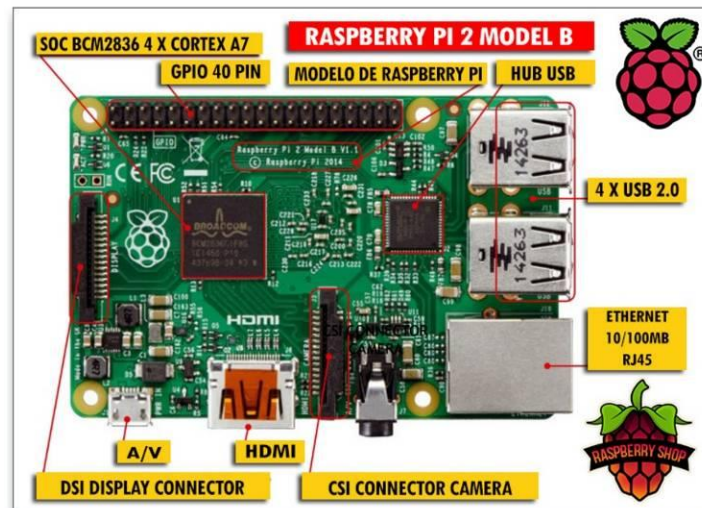


Figura 1: Estructura Raspberry pi 2 (raspberrypi, 2016)

Como se muestra en la Figura 1 la *raspberrypi 2* cuenta con unas características muy interesantes. En su corazón se encuentra con un chip integrado *Broadcom BCM2835*, que contiene un procesador *ARM11* con varias frecuencias de funcionamiento y la posibilidad de subirla (haciendo *overclocking*) hasta 1 *GHz* sin perder la garantía, un procesador gráfico *VideoCore IV*, y 512 *MB* de memoria *RAM* (la *Raspberry Pi 2* y la *Raspberry Pi 3* cuentan con 1 *GB* de memoria *RAM*). Todo ello equivale en la práctica a un ordenador con unas capacidades gráficas similares a la *XBOX* de *Microsoft* y con la posibilidad de reproducir vídeo en 1080p.

En la placa se encuentra además con una salida de vídeo y audio a través de un conector *HDMI*, con lo que se consigue conectar la tarjeta tanto a televisores como a monitores que cuenten con dicha conexión. En cuanto a vídeo se refiere, también cuenta con una salida de vídeo compuesto y una salida de audio a través de un *minijack*. Posee una conexión *ethernet* 10/100 y, si bien es cierto que podría echarse en falta una conexión *Wi-Fi*, gracias a los dos puertos *USB* incluidos, se puede suplir dicha carencia con un adaptador *WIFI* si se necesita.

La placa *Raspberry Pi* se entrega sin ningún Sistema Operativo; éste se debe descargar e instalarlo sobre una tarjeta *SD / microSD* que se introduce en la ranura de la *Raspberry Pi*.

### 2.3 *OpenCV*

Menciona (Arevalo, Gonzalez, & Ambrosio, 2016) *OpenCV The Open Computer Vision Library*, el 13 de Junio del 2000, *Intel® Corporation* anunció que estaba trabajando con un grupo de reconocidos investigadores en visión por computadora para realizar una nueva librería de estructuras/funciones en lenguaje C. Esta librería proporcionaría un marco de trabajo de nivel medio-alto que ayudaría al personal docente e investigador a desarrollar nuevas formas de interactuar con los ordenadores. Este anuncio tuvo lugar en la apertura del *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*. Había nacido *The Open Computer Vision Library* y lo hacía bajo licencia *BSD* (Software Libre).

La librería *OpenCV* es una API de aproximadamente 300 funciones escritas en lenguaje C que se caracterizan por lo siguiente:

- Su uso es libre tanto para su uso comercial como no comercial

No utiliza librerías numéricas externas, aunque puede hacer uso de alguna de ellas, si están disponibles, en tiempo de ejecución.

- Es compatible con *The Intel® Processing Library (IPL)* y utiliza *The Intel® Integrated Performance Primitives (IPP)* para mejorar su rendimiento, si están disponibles en el sistema.

Dispone de interfaces para algunos otros lenguajes y entornos: *EiC* - intérprete ANSI C escrito por Ed Breen. *Hawk* y *CvEnv* son entornos interactivos (escritos en MFC y TCL, respectivamente) que utilizan el intérprete *EiC*; *Ch* - intérprete ANSI C/C++ creado y soportado por la compañía *SoftIntegration*; *Matlab®* - gran entorno para el cálculo numérico y simbólico creado por *Mathworks*; y muchos más.

### 2.3.1 Estructura

La librería *OpenCV* está dirigida fundamentalmente a la visión por computador en tiempo real. Entre sus muchas áreas de aplicación destacarían: interacción hombre-máquina; segmentación y reconocimiento de objetos; reconocimiento de gestos; seguimiento del movimiento; estructura del movimiento; y robots móviles. Ver figura 2.

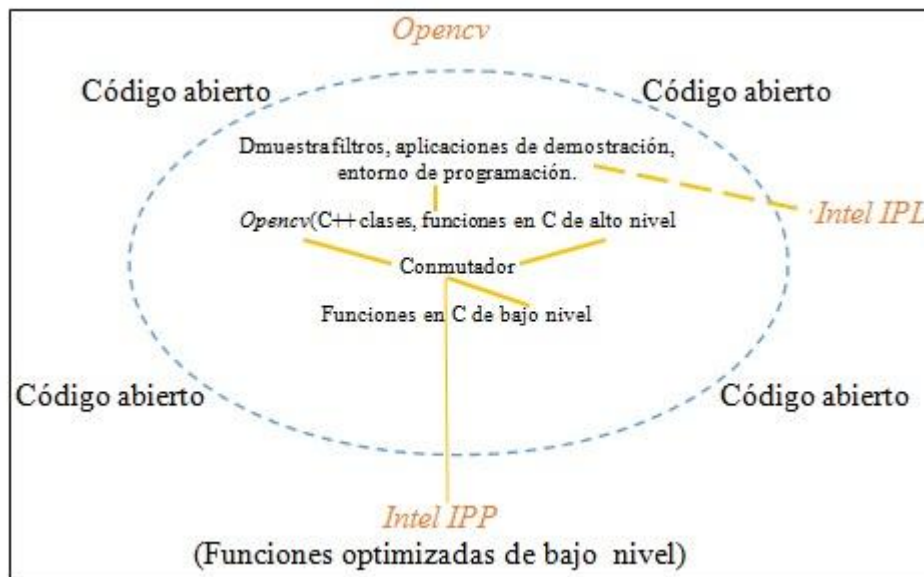


Figura 2: Estructura de la librería *OpenCV*. (Arevalo, Gonzalez, & Ambrosio, 2016)



### 2.3.2 Funciones de *OpenCV*



Figura 3: Funciones de *OpenCV* (Ruiz, 2015)

Según (Ruiz, 2015) como se muestra en la figura 3 *OpenCV* tiene una estructura modular. Los módulos principales de *OpenCV* son:

- *core*: Este es el módulo básico de *OpenCV*. Incluye las estructuras de datos básicas y las funciones básicas de procesamiento de imágenes. Este módulo también es usado por otros módulos como *highgui*.
- *highgui*: Este módulo provee interfaz de usuario, códecs de imagen y vídeo y capacidad para capturar imágenes y vídeo, además de otras capacidades como la de capturar eventos del ratón...etc.
- *imgproc*: Este módulo incluye algoritmos básicos de procesado de imágenes, incluyendo filtrado de imágenes, transformado de imágenes...etc.
- *video*: Este módulo de análisis de vídeo incluye algoritmos de seguimiento de objetos, entre otros

– *objdetect*: Incluye algoritmos de detección y reconocimiento de objetos para objetos estándar.

### **2.3.1 Instalación de *OpenCV 3.0.0* y *Python 2.7* en *raspberry pi 2 raspbian Jessie***

*Menciona (Rosebrock, 2015) el paso 1 es instalar las dependencias*

- Lo primero que se debe hacer es actualizar los paquetes existentes desde la terminal, una vez concluida esta tarea se debe actualizar el firmware de la *raspberry pi 2*.
- Una vez terminado dichos procesos se debe reiniciar la *raspberry pi 2*.
- Se procede a instalar algunas herramientas de desarrollo.
- Ahora se pasa a la instalación de los paquetes de E/S que permiten cargar los formatos de archivo de imagen como *JPEG, PNG, TIFF*, etc.
- Al igual que los paquetes de E/S, también se tienen paquetes de video de E/S. Estos paquetes permiten cargar varios formatos de archivos de vídeo.
- Se necesita instalar la biblioteca de desarrollo “*GTK*” para que se pueda compilar el sub-módulo *highgui* de *OpenCV*, lo que permite visualizar imágenes de la pantalla y construir interfaces con gráficas simples.
- Diversas operaciones dentro de *OpenCV* (tales como operaciones de la matriz) se pueden optimizar añadiendo dependencias.
- Por último, se tienen que instalar los archivos de *Python 2.7* para que se puedan compilar los enlaces de *OpenCV + Python*.

*Paso 2 Descargar el código fuente de OpenCV 3.0.0*

- Hasta este punto se tienen todos los requisitos previos instalados, así que se descargará la versión 3.0.0 de *OpenCV* desde el repositorio de *OpenCV*.

NOTA: Para la instalación completa de *OpenCV* 3.0.0 (que incluye características tales como *SIFT* y *SURF*), se tendrá que descargar *OpenCV\_contrib* en tal caso las versiones de *OpenCV* y *OpenCV\_contrib* tienen que ser las mismas (*OpenCV* 3.0.0, *OpenCV\_contrib* 3.0.0).

### *Paso 3 Instalar Python 2.7*

El primer pasó en la instalación *Python* para compilarlo en *OpenCV*, es instalar *pip*, un gestor de paquetes de *Python*

- Usando *virtualenv* y *virtualenvwrapper* permite crear a *Python* entornos aislados, separados de su sistema de instalación de *Python*. Esto significa que se pueden ejecutar múltiples versiones de *Python*, con diferentes versiones de los paquetes instalados en cada entorno virtual - esto soluciona el problema "El Proyecto A depende de la versión 1.x, pero el Proyecto B necesita 4.x", problema que surge con frecuencia en la ingeniería de *software*. La instalación de estos paquetes no es un requisito para instalar *OpenCV* y *Python* pero se recomienda instalarlos.
- Después de que *virtualenv* y *virtualenvwrapper* han sido instalados, se necesita actualizar el archivo de perfil (*~/.profile*). *Nota: Es probable que se tenga que ejecutar el comando source ~/.profile cada vez que se abre una nueva terminal para asegurarse de que el entorno se ha configurado correctamente.*
- El siguiente paso es crear el entorno virtual de *Python*, donde se va a hacer el trabajo de visión por computadora.
- Se tiene que asegurar de que se está en el entorno virtual *cv*.
- Suponiendo que se está en el entorno virtual *cv*, se procede a instalar *NumPy*, una dependencia importante a la hora de compilar los enlaces *Python* para *OpenCV*.

### *Paso 4 Compilar e instalar OpenCV 3.0.0*

- Primeramente se tiene que verificar que se está en el entorno virtual *cv*.

- Crear el directorio *build*
- Cambiar al directorio *build* y instalar la serie de instrucciones y módulos extras (en dado caso que así lo deseemos)
- Compilar *OpenCV*.

Paso 5 Terminar la instalación

Siempre y cuando el Paso # 4 termine sin error, *OpenCV* 3.0.0 debería estar instalado.

## 2.4 Python

*Menciona* (Gonzalez R. , <https://launchpadlibrarian.net>, 2016) *python* es un lenguaje de programación creado por Guido Van Rossum a principios de los años 90 cuyo nombre está inspirado en el grupo de cómicos ingleses “*Monty Python*”. Es un lenguaje similar a Perl, pero con una sintaxis muy limpia y que favorece un código legible.

Se trata de un lenguaje interpretado o de script, con tipado dinámico, multiplataforma y orientado a objetos.

### 2.4.1 Lenguaje interpretado o de script

Según (Gonzalez R. , <https://launchpadlibrarian.net>, 2016) un *lenguaje interpretado o de script* es aquel que se ejecuta utilizando un programa intermedio llamado intérprete, en lugar de compilar el código a lenguaje máquina que pueda comprender y ejecutar directamente una computadora (lenguajes compilados).

La ventaja de los lenguajes compilados es que su ejecución es más rápida. Sin embargo los lenguajes interpretados son más flexibles y más portables.

*Python* tiene, no obstante, muchas de las características de los lenguajes compilados, por lo que se podría decir que es semi interpretado. En *Python*, como en *Java* y muchos otros lenguajes, el código fuente se traduce a un pseudo código máquina intermedio llamado *bytecode* la primera vez que se ejecuta, generando archivos *.pyc* o *.pyo* (*bytecode* optimizado), que son los que se ejecutarán en sucesivas ocasiones.

### *Tipado dinámico*

La característica de *tipado dinámico* se refiere a que no es necesario declarar el tipo de dato que va a contener una determinada variable, sino que su tipo se determinará en tiempo de ejecución según el tipo del valor al que se asigne, y el tipo de esta variable puede cambiar si se le asigna un valor de otro tipo.

### *Fuertemente tipado*

No se permite tratar a una variable como si fuera de un tipo distinto al que tiene, es necesario convertir de forma explícita dicha variable al nuevo tipo previamente. Por ejemplo, si tenemos una variable que contiene un texto (variable de tipo cadena o *string*) no podremos tratarla como un número (sumar la cadena "9" y el número 8). En otros lenguajes el tipo de la variable cambiaría para adaptarse al comportamiento esperado, aunque esto es más propenso a errores.

### *Multiplataforma*

El intérprete de *Python* está disponible en multitud de plataformas (*UNIX, Solaris, Linux, DOS, Windows, OS/2, Mac OS, etc.*) por lo que si no utilizamos librerías específicas de cada plataforma nuestro programa podrá correr en todos estos sistemas sin grandes cambios.

### *Orientado a objetos*

La *orientación a objetos* es un paradigma de programación en el que los conceptos del mundo real relevantes para nuestro problema se trasladan a clases y objetos en nuestro programa. La ejecución del programa consiste en una serie de interacciones entre los objetos.

*Python* también permite la programación imperativa, programación funcional y programación orientada a aspectos.

*Python* es un lenguaje que todo el mundo debería conocer. Su sintaxis simple, clara y sencilla; el tipado dinámico, el gestor de memoria, la gran cantidad de librerías disponibles y la potencia del lenguaje, entre otros, hacen que desarrollar una aplicación en *Python* sea sencillo, muy rápido y, lo que es más importante, divertido.

La sintaxis de *Python* es tan sencilla y cercana al lenguaje natural que los programas elaborados en *Python* parecen pseudocódigo. Por este motivo se trata además de uno de los mejores lenguajes para comenzar a programar.

*Python* no es adecuado sin embargo para la programación de bajo nivel o para aplicaciones en las que el rendimiento sea crítico.

Algunos casos de éxito en el uso de *Python* son *Google*, *Yahoo*, la *NASA*, Industrias *Light & Magic*, y todas las distribuciones *Linux*, en las que *Python* cada vez representa un tanto por ciento mayor de los programas disponibles.

## **2.5 Adobe Photoshop**

Menciona (Alegsa, 2016) *Adobe Photoshop* es una aplicación para la creación, edición y retoque de imágenes. Fue desarrollado por la compañía *Adobe Systems*. Se lanzó originalmente para computadoras *Apple*, pero luego saltó a la plataforma *Windows*.

Los formatos propios de *Photoshop* son *PSD* y *PDD*, que guardan capas, canales, guías y en cualquier modo de color. Su uso es libre tanto para su uso comercial como no comercial.

*Photoshop* también soporta otros formatos como *PostScript*, *EPS*, *DCS*, *BMP*, *GIF*, *JPEG*, *PICT*, *PIFF*, *PNG*, *PDF*, *IFF*, *PCX*, *RAW*, *TGA*, *Scitex CT*, *Filmstrip*, *FlashPix* por mencionar algunos.

El programa comenzó a ser escrito para Macintosh en 1987 por Thomas Knoll, un estudiante de la Universidad de Michigan, con el objetivo de mostrar imágenes en escala de grises en pantallas monocromáticas.

Este programa, que fue llamado *Display*, llamo la atención de su hermano John Knoll, empleado de Industrial *Light & Magic*, quien recomendó convertirlo en un editor de imágenes.

Para su trabajo Thomas se tomó un receso de seis meses de sus estudios en 1988 y, junto con su hermano, crearon el programa *ImagePro*.

Finalmente ese mismo año Thomas renombró al programa *Photoshop*, logrando un acuerdo con los fabricantes de escáneres *Barneyscan* para que distribuya este programa con sus dispositivos. Apenas unas 200 copias del programa fueron distribuidas de esta manera.

Mientras tanto, John viajó a Silicon Valley y mostró su programa a ingenieros de *Apple Computer* y a Russell Brown, director de arte de *Adobe*.

Ambos estuvieron interesados y compraron la licencia para distribuirlo en Septiembre de 1988.

John se quedó en California desarrollando *plugins*, mientras que Thomas permaneció en Ann Arbor escribiendo el código del programa, *Photoshop* 1.0 fue lanzado en 1990 para Macintosh.

Características:

- 1) Editor de gráficos raster
- 2) Editor de gráficos vectoriales
- 3) Licencia software propietario
- 4) Escrito en C++

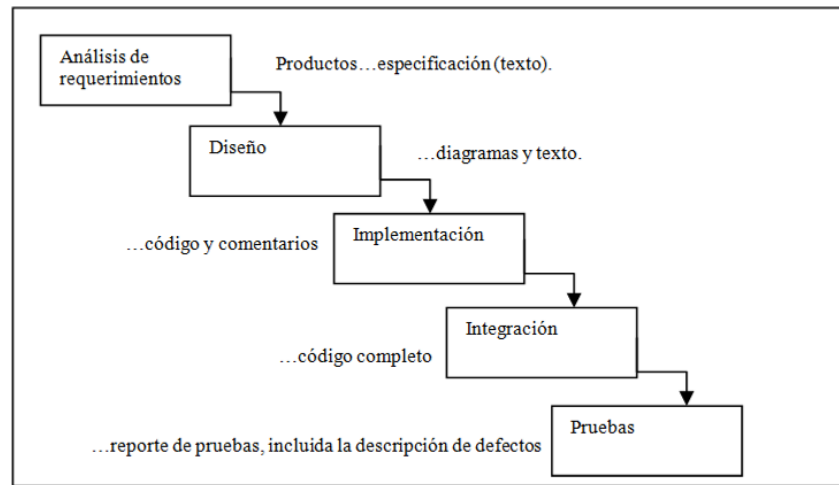
Las funcionalidades de *Photoshop* pueden ser extendidas empleando *add-ons* o *plugins*, especialmente filtros para realizar diferentes efectos en las imágenes.

Disponible en portugués, chino simplificado, chino tradicional, checo, danés, español, neerlandés, inglés, finlandés, francés, alemán, húngaro, italiano, japonés, coreano, noruego, polaco, rumano, ruso, sueco, turco y ucraniano.

## **2.6 Ciclo de vida en cascada**

Menciona (procesosdesoftware, 2016) este modelo de ciclo de vida fue propuesto por Winston Royce en el año 1970. Es un ciclo de vida que admite iteraciones, contrariamente a

la creencia de que es un ciclo de vida secuencial como el lineal. Después de cada etapa se realiza una o varias revisiones para comprobar si se puede pasar a la siguiente. Ver figura 4.



*Figura 4: Modelo en cascada. (Braude, 2003)*

## 2.7 Concepto y productos de una tlapalería

Según (ahorayasabes, 2011) el concepto de la palabra “tlapalería” proviene de la voz náhuatl tlapalli que significa “color” y la terminación “eria” se le da a las tiendas donde se vende algún producto.

Menciona (construramablanca, 2016) los productos que se pueden encontrar dentro de una tlapalería son:

- Cuerdas, sogas e hilos
- Escaleras y andamios
- Misceláneos
- Primers
- Silicones y selladores estructurales
- Cepillos y lijas
- Aditivos multiuso
- Accesorios de limpieza
- Cables y cadenas
- Electrodo y soldaduras



- Mallas y cercas
- Cintas adhesivas
- Lonas
- Anti fungicidas de madera
- Artículos de sujeción.

### **2.7.1 Modelos de negocio de una tlapalería**

Según (Giraldo, Roldan, & Garro, 2009) los modelos de negocio principales dentro de una tlapalería integran los siguientes procesos:

*Administrativo:* Liderar proyectos que aumenten la rentabilidad de la empresa, crear ideas competitivas de organización laboral para su máximo desempeño e incorporar nuevos modelos corporativos y buenas prácticas de negocios disponibles.

*Financiero:* Automatizar procesos de compras y de inventarios así como reducir tiempos en la contabilización sin requerir doble digitación.

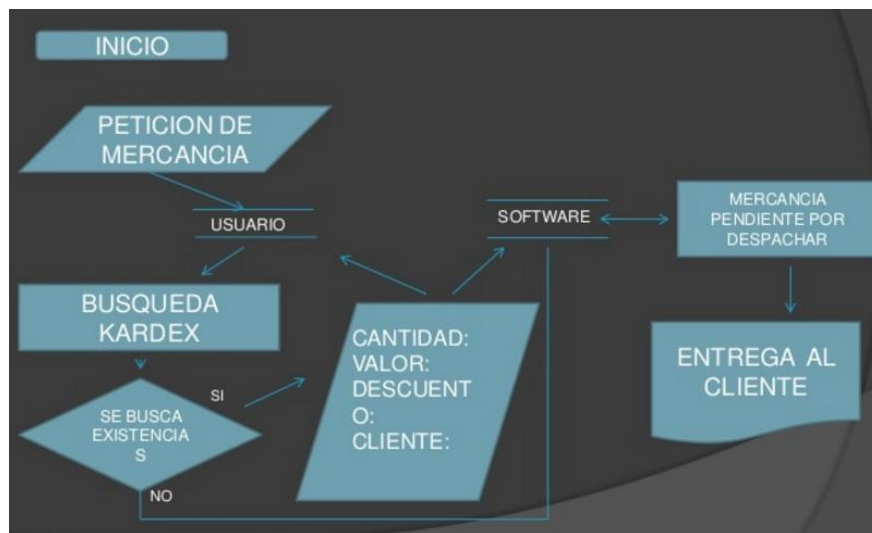
*Mercadeo y ventas:* Mayor eficiencia en la operación del negocio, establecer un programa de publicidad innovador, cierre contable y gestión bancaria.

*Atención al cliente:* Aumentar el número de clientes a través de la disponibilidad de productos y ofrecer un mejor servicio en el momento que se necesite.

*Inventario:* Traslado, entrada y salida de mercancía.

*Logística:* Transporte de la mercancía, despacho y recibo de mercancía.

## 2.7.2 Proceso de venta de un artículo en específico



*Figura 5: Proceso de venta de un artículo (Giraldo, Roldan, & Garro, 2009)*

En la figura 5 se muestra el proceso de venta cuando el cliente llega a la sucursal por la búsqueda de algún producto, el vendedor será el encargado de buscar en almacén si tiene algún producto en existencia o parecidos, esto lo hará por medio de la utilización de algún software que facilitara la búsqueda e información de dicho producto.

### 3 Capítulo 3 Imágenes digitales

#### 3.1. Conceptos básicos del procesamiento digital de imágenes:

De acuerdo a (Rodríguez & Sossa, 2012) el tratamiento o procesamiento digital de imágenes (TDI o PDI) es una de las etapas primarias del proceso de la visión por computadora. De manera sencilla, el TDI o PDI puede definirse como un conjunto de procedimientos que se realizan sobre una imagen (señal obtenida mediante un captor conectado a la computadora) para su almacenamiento, transmisión o tratamiento. Esta técnica es una de las ramas del llamado *tratamiento digital de señales* que más desarrollo ha experimentado en las últimas cuatro décadas. Cabe señalar que entre algunos ejemplos que evidencian los beneficios del PDI se encuentran la utilización de la robótica en la industria, la clasificación automática del terreno y de diferentes recursos naturales a través de imágenes de satélites, el empleo de procedimientos avanzados en la medicina y, en general, en el mejoramiento y análisis de imágenes biomédicas, en el estudio de las condiciones ambientales y meteorológicas, en la defensa, etc.

En el procesamiento digital de imágenes deben tomarse en cuenta varios aspectos como la percepción psicovisual del ser humano. Éste es un factor importante porque independientemente del tratamiento que se le aplique a una imagen, el observador será quien, según su percepción, decidirá si dicha imagen le agrada o no.

El desarrollo de los métodos de procesamiento digital de imágenes tiene su origen en dos áreas principales de aplicación: el mejoramiento de la información pictórica para la interpretación humana, y el procesamiento de datos de la imagen para la percepción de máquina autónoma en el que se incluyen etapas de transmisión y/o almacenamiento de estos datos.

La herramienta usada en el tratamiento digital de las imágenes son las matemáticas; los conceptos que se verán son básicos. La computadora y los algoritmos que se implementan sobre éstas también tienen un papel muy importante en la manipulación de las imágenes.

El término "*imagen monocromática*" o imagen simplemente, se refiere a una función de intensidad de luz bidimensional  $f(x,y)$ , donde  $x$  e  $y$  indican las coordenadas espaciales y el

valor de  $f$  en cualquier punto  $(x,y)$  es proporcional a la luminosidad (o nivel de gris) de la imagen en dicho punto.

Una *imagen digital* es una imagen (función)  $f(x,y)$  que ha sido discretizada tanto en coordenadas espaciales como en luminosidad. Una imagen digital puede ser considerada como una matriz cuyos índices de renglón y columna identifican un punto (un lugar en el espacio bidimensional) en la imagen y el correspondiente valor de elemento de matriz identifica el nivel de gris en aquel punto. Los elementos de estos arreglos digitales son llamados elementos de imagen.

Algunos de los problemas característicos en el diseño de estos subsistemas que involucran el uso de representaciones de señales son las siguientes:

Los dispositivos sensoriales realizan un número limitado de mediciones sobre las señales de entrada; estas mediciones deben ser adecuadas para obtener aproximaciones útiles. Decidir que mediciones realizar y como usarlas de tal manera que aproximen mejor a la señales de entrada son los problemas que deben ser resueltos.

Para la selección del procesamiento y/o codificación que se hará sobre una señal, es necesaria una interpretación de las componentes de la señal. El modelo del sistema de visión humano puede ser utilizado en ciertas etapas de procesamiento para dicha interpretación.

Los dispositivos de despliegue sintetizan una imagen usando un número finito de respuestas básicas de despliegue, como los puntos de fósforo utilizados en un tubo de rayos catódicos. Seleccionar el tamaño y la forma de éstas respuestas de despliegue, la configuración (número y posición relativa) y como pueden ser controlados de la mejor manera óptima para obtener imágenes con la calidad/fidelidad requerida son aspectos que deben ser cubiertos.

### **3.1.1 Proceso de captación y formación de una imagen**

*Según* (Rodríguez & Sossa, 2012) *una imagen*: Es la representación óptica de uno o más objetos iluminados por una o más fuentes de radiación. Así en general, los elementos que forman parte del proceso de formación de una imagen son los siguientes: objeto u objetos, fuente o fuentes de iluminación y sistema de formación de la imagen. Es por ello que todo modelo matemático que pretenda reflejar de forma fiel esta realidad física debe tomar en

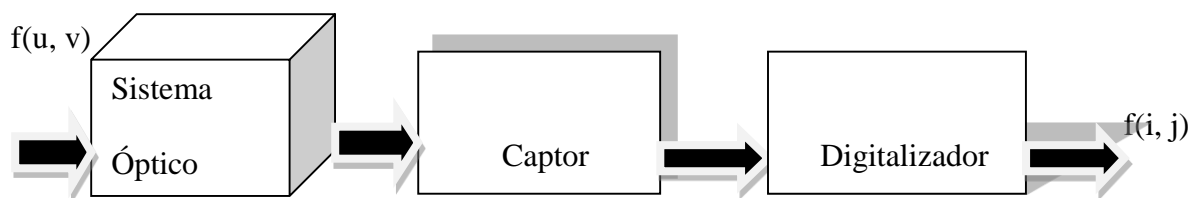
consideración la fuente de radiación (la cual puede ser luz visible, de rayos X, radiación ultrasónica, infrarroja u otra), la física de la interacción de la radiación y el objeto y como es lógico el sistema de adquisición empleado.

Cualquier sistema de análisis de imágenes no es tan robusto como el sistema visual humano. Estos sistemas pueden fácilmente conducir a resultados errados debido a factores como el número, tipo y material de los objetos en la escena, su interrelación, las sombras, los tipos de fuentes iluminación y, por supuesto, al proceso no lineal de formación de la imagen.

La luz reflejada  $f(u, v)$  es la imagen óptica que sirve de entrada al sistema de formación de la imagen digital. Tal sistema está constituido, en general, por lentes ópticos, un captor óptico y un digitalizador o numerizador de imagen. El sistema óptico  $H$  se modela como un sistema de desplazamiento lineal e invariante el cual tiene una respuesta tipo impulso:  $h(x, y)$ . Este sistema óptico ( $H$ ), generalmente, es un sistema pasa bajo que tiende a suprimir las altas frecuencias contenidas en la imagen de entrada  $f(u, v)$ . Por tal motivo la imagen de salida  $s(x, y)$  es, en general, una versión borrosa o no fielmente enfocada de la imagen original  $f(u, v)$ . Como estas dos señales,  $f(u, v)$  y  $s(x, y)$ , representan valores de intensidades ópticas ellas deben tomar valores no negativos, es decir:

$$f(u, v) \geq 0 \quad (1)$$

$$s(x, y) \geq 0 \quad (2)$$



*Figura 6: Modelo de formación de la imagen digital (Rodríguez & Sossa, 2012)*

La figura 6 muestra el flujo secuencial para formar una imagen digital desde que el sistema óptico permite obtener imágenes de los objetos en estudio hasta su digitalización.

Existen otras muchas modalidades un poco más efectivas que las cámaras a tubo para la percepción y generación de la imagen, las cuales pueden ser desde un simple captor hasta

captorees en filas o arreglos de captorees, entre otros. En la actualidad las cámaras de estado sólido (*CCD*) y los exploradores de imágenes juegan un papel muy importante en los sistemas de visión por computadora. Sin embargo, el empleo de uno u otro dispositivo de captación están estrechamente relacionados con la aplicación. Por ejemplo, si la aplicación consiste en la lectura de películas fotográficas el captor más apropiado será un densitómetro. Si se quiere realizar, a través de imágenes, el estudio ambiental de una zona determinada un sensor adecuado podría ser una cámara en cualquiera de las siguientes versiones; fotográfica, de televisión o de digital. Aquí en esta aplicación el explorador o escáner queda excluido. Si en otra situación lo que se desea es realizar la restauración de documentos antiguos un explorador con una alta resolución sería el captor más apropiado.

No obstante, no existen fronteras absolutas para el uso de uno u otro dispositivo ya que muchas veces, en distintos casos prácticos, algunos de ellos han sido utilizados de forma ingeniosa fuera de sus fronteras. Otro aspecto al cual no se ha hecho mención y que juega un rol de peso importante en la formación de la imagen digital es la cuantificación y el muestreo.

### **3.2 Modelos de representación**

De acuerdo con (Jimenez, 2016) a la hora de representar el contenido de una imagen existen básicamente dos métodos que a su vez sirven para dividir las imágenes en dos grandes grupos: mapa de bits y gráficos vectoriales.

#### **3.2.1 Mapa de bits**

Menciona (Jimenez, 2016) en el grupo de imágenes denominado mapa de bits, el contenido se representa mediante pequeños puntos rectangulares denominados píxeles.

El segundo grupo de imágenes, los gráficos vectoriales, sigue un método de representación totalmente distinto. Cada uno de los elementos que componen la imagen se tratan como objetos, y sus propiedades se definen mediante fórmulas matemáticas.

El uso de gráficos vectoriales se encuadra normalmente dentro de trabajos que requieren un alto nivel de detalle a cualquier tamaño, como diseños complejos o mapas

Un *pixel* es la unidad mínima a partir del cual se forma cualquier imagen en mapa de bits. Para la representación de un pixel se puede utilizar un *bit*, dos *bits*, cuatro *bits* de información, y proviene de la abreviatura de *picture element* (elemento de imagen). Sus propiedades son:

- Posición relativa respecto al resto de píxeles.
- Almacenamiento de color, en *bits*. Para la representación de un píxeles puede utilizar 1 *bit*, 2 *bits*, 4 *bits*,... de información. Este valor determina el número de colores que puede representar dicho píxel. A más bits, más colores y mejor calidad de imagen, pero también mayor tamaño de la misma. (Tamaño = peso en *kb*.  $1000\text{ Kb} = 1\text{ Mb}$ .  $1000\text{ Mb} = 1\text{ Gb}$ ).
- Tamaño relativo, siempre en relación con una unidad de longitud fija, cuyas unidades más habituales son pulgadas y centímetros. Lo más usual es trabajar en píxeles por pulgada (ppi). A mayor número de ppi, menor será el tamaño del píxel. Esto está directamente relacionado con el concepto de resolución, cualidad de la que dependen todas las imágenes de mapas de bits. A menor número de ppi, menor es la resolución, lo que produce una menor definición de la imagen, con la correspondiente pérdida de calidad y detalle.

Es muy importante a la hora de crear o tratar imágenes, saber cuál va a ser su utilización para tener la resolución óptima. Toda imagen de mapas de *bits* tiene un número invariable de píxeles a determinada resolución según su tamaño, y éste variará si varía la resolución.

### **3.2.2 Gráfico vectorial**

De acuerdo con (redgrafica, 2016) un *gráfico vectorial* es toda imagen digital formada por diferentes objetos geométricos independientes. Cada uno de estos elementos definido por parámetros matemáticos como la forma, posición, color, el tipo y grosor de contorno, etc.

Los gráficos vectoriales son en su formato completamente distintos a los gráficos de mapas de bits o también llamados matriciales, los cuales están constituidos por píxeles.

Los vectores tienen la ventaja sobre los píxeles de ser escalables, es decir, se puede aumentar su tamaño conservando su calidad original. Lo que no ocurre con los gráficos a base de píxeles que presentan degradación de la calidad al aumentar el tamaño.

Otra diferencia importante entre estos dos tipos de imágenes digitales es el peso, los vectoriales por ser solamente parámetros matemáticos, suelen ser mucho menos pesados que una imagen rasterizada o de píxeles. De la misma manera los gráficos vectoriales pueden ser transformados (estirar, rotar, mover, distorsionar) de una manera más sencilla y con menos requerimientos de memoria en el equipo.

Sin embargo todos los gráficos vectoriales o vectorizados tienen que ser transformados a píxeles cuando se presentan en una pantalla o cuando se requiere su impresión.

### **3.3 Principales aplicaciones**

#### *Diseño animación e ilustración*

Menciona (redgrafica, 2016) en la actualidad, muchos ambientes de tercera dimensión son compuestos por gráficos vectoriales que conforman distintos tipos de aplicaciones.

También son ampliamente utilizados para animaciones 2D en diseño web por su facilidad de uso y su bajo peso.

#### *Documentos digitales*

Permiten la descripción gráfica de un documento digital, sin la pérdida de calidad de la imagen por aplicaciones o transformaciones de forma.

#### *Videojuegos*

Utilizados generalmente en la producción de ambientes virtuales de tres dimensiones.

#### *Tipografía*

Las fuentes son archivos tipográficos que utilizan en la mayoría de las ocasiones imágenes vectoriales. Algunos de los formatos más conocidos son *TrueType*, *OpenType* y *PostScript*.

Principales formatos de los gráficos vectoriales:

*SWF Adobe flash*: Utilizado para gráficos y animaciones 2D para internet.

*AI Adobe Illustrator*: Para gráficos e ilustración para web e impresión.

*CDR Corel Draw*: Utilizado generalmente para diseño y autoedición para impresos.



*PDF Adobe Acrobat:* Para intercambio de documentos y flujos de trabajo.

*Post Script:* Generalmente se utiliza en impresión e intercambio de archivos.

Es importante resaltar que las fotografías no pueden ser vectorizadas sin una pérdida considerable de calidad en la imagen; por lo que en este caso sigue siendo recomendable el uso de formatos de píxeles (*jpeg, png, tiff, etc.*).

De acuerdo con (Delgado, 2009) la resolución de una imagen resulta determinante en su aspecto, ya que a más resolución, más píxeles y, por lo tanto, mayor nivel de detalle.

La resolución indica el número de píxeles por pulgada utilizados para su representación.

En una cámara analógica la luz entra a través de la lente e incide sobre la película que a su vez tiene un tratamiento químico fotosensible que permite registrar la imagen. En el caso de los sistemas digitales de fotografía la primera parte no varía, es decir, la luz llega a través de la lente de la cámara, pero en esta ocasión, es un pequeño dispositivo electrónico denominado CCD (*coupling Charge Device*) el que se encarga de convertir la luz en una señal eléctrica.

Este sensor dispone de miles de pequeñas células fotosensibles de distintas formas que recogen la luz recibida y la envía a un circuito electrónico que interpreta esta formación, mostrando en cada caso la imagen que deseamos capturar con la cámara.

### **3.4 Problemática en el análisis automatizado de imágenes**

*Según (Krogh, 2010) 1) el reconocimiento de objetos:* Consiste en la determinación en forma automática de la identidad de los objetos en una imagen a partir de un conjunto de rasgos descriptivos extraídos a partir de una o más imágenes de dicho(s) objeto(s). Desde el punto de vista de la clasificación, el reconocimiento de objetos puede ser visto como el problema de encontrar las clases a las cuales cada uno de los objetos que se encuentran en la imagen.

*2) Detección de objetos:* Tiene que ver con la determinación de si un objeto dado se encuentra o no en una imagen. Para esto se puede utilizar, por ejemplo, una descripción del objeto a detectar o una imagen recortada de dicho objeto.

*3) Interpretación de una imagen:* Se trata de la etapa de más alto nivel en el análisis de una imagen. Se busca no solo determinar la identidad de los objetos en una imagen sino además

poder contestar, en el sentido más amplio, a la pregunta: ¿de qué habla la imagen? A través de procedimientos como este se intentaría determinar las relaciones entre los objetos en la imagen, el estado de ánimo, el género o edad de una persona en el caso de análisis de una imagen del rostro de una persona, por citar algunos.

*La edición de imágenes no destructivas* es un término muy amplio que hace referencia a cualquier tecnología de edición de imágenes que permite al usuario realizar ajustes en la imagen sin alterar de forma permanente la imagen original.

*La edición de imágenes no destructivas* es el área de desarrollo en el ajuste imagen más importante, particularmente desde el punto de vista de la gestión y tratamiento de archivo fotográfico digital. Ofrece ventajas significativas en cuanto al almacenamiento requerido para a imagen y la libertad que nos proporciona para explorar todas las interpretaciones posibles para nuestras imágenes.

### **3.5 Textura**

Menciona (bibing, 2016) la *textura* es una característica importante utilizada en segmentación, identificación de objetos o regiones de interés en una imagen y obtención de forma. El uso de la textura para identificar una imagen proviene de la habilidad innata de los humanos para reconocer diferencias texturales. Por medio de la visión y el tacto, el ser humano es capaz de distinguir en forma intuitiva diversos tipos de textura. La textura, por consiguiente, es una característica de difícil definición siendo la más extendida la siguiente la dada por Haralick: “Una textura está definida por la uniformidad, densidad, grosor, rugosidad, regularidad, intensidad y direccionalidad de medidas discretas del tono en los píxeles y de sus relaciones espaciales”

En una imagen digital, la textura depende de tres conceptos:

- 1) La frecuencia de cambio de los tonos en los píxeles.
- 2) La dirección o direcciones de cambio.
- 3) El contraste entre un píxel y sus vecinos. Sin embargo, es independiente del tono o color de la imagen.

Para llevar a cabo el procesamiento digital de imágenes es de gran utilidad, para ciertas aplicaciones, obtener información sobre la textura. Este problema se puede abordar desde diferentes puntos de vista, dependiendo de la técnica específica utilizada para extraer de forma cuantitativa la información contenida en la textura de la imagen. Para caracterizar las texturas existen básicamente tres formas de procesar la imagen y extraer su información:

1) *Descriptores en frecuencia*: Los modelos espectrales o de frecuencia consisten en obtener la transformada en frecuencia de la imagen y a partir de ésta, obtener ciertas características. Dichas características son más fáciles de obtener del espectro en coordenadas polares y sirven de base para la clasificación.

Los descriptores en frecuencia se basan principalmente en el análisis del espectro de Fourier mediante la transformada del mismo nombre. El espectro de Fourier está especialmente indicado para describir la direccionalidad de patrones bidimensionales periódicos de una imagen, ya que estos patrones de textura son fácilmente distinguibles como concentraciones altas de energía en el espectro. Las características más importantes vienen dadas por:

- Magnitud de “picos” prominentes en frecuencia: proporciona información sobre la direccionalidad de los patrones de textura.
- Localización de los “picos”: proporciona información sobre el periodo espacial fundamental de los patrones.
- Aplicar técnicas estadísticas a partes a-periódicas, una vez separada de la parte periódica mediante un filtro.

2) *Descriptores estructurales*: En cuanto a los descriptores estructurales, su desarrollo se basa en obtener patrones primitivos, *texel* (del inglés “*textura element*”), de la textura para generar una descripción de cómo se agrupan éstos para formar la textura en sí.

3) *Descriptores probabilísticos*: Estos métodos se basan en construir de un modelo de la imagen, cuyos parámetros estimados sirven para describir y sintetizar una textura, ya que reúnen las características esenciales que caracterizan a la textura. Se consideran como tales las cadenas de Markov y los modelos fractales.

### 3.6 Clasificadores

De acuerdo con (Sanz, 2008) el objetivo del clasificador es analizar los datos obtenidos en la fase anterior para aprender a distinguir entre diferentes objetos. Este conocimiento puede ser expresado como reglas, patrones, asociaciones, relaciones o de forma general como modelo.

En general el error de combinar varios clasificadores se explica por lo que se conoce como *bias-variance decomposition*. El sesgo (*bias*) de cada clasificador viene dado por su error intrínseco y mide lo bien que un clasificador explica el problema. La varianza está dada por los datos que se usan para construir el modelo. El error esperado total de clasificación está dado por la suma del sesgo y la varianza. Al combinar múltiples clasificadores se reduce el error esperado ya que se reduce la varianza.

*Ejemplos de modelos utilizando combinaciones homogéneas:*

1. *Bagging*: El algoritmo de *Bagging* (*Bootstrap Aggregating*) genera clasificadores de un sub-conjunto de muestras.

Es especialmente útil en algoritmos de aprendizaje inestables (cambian mucho sus estructuras al cambiar un poco los ejemplos), por ejemplo, los árboles de decisión.

Una muestra de ejemplos *bootstrap* se genera al muestrear uniformemente ‘m’ instancias del conjunto de entrenamiento con reemplazo. Se generan T muestras, B1, ..., BT y se construye un clasificador Ci para cada muestra. Con estos, se construye un clasificador final C\* de todos los C1 a Ct cuya salida es la salida mayoritaria de los clasificadores.

Para una de las muestras, un ejemplo tiene la probabilidad de  $1 - (1 - 1/m)^m$  de ser seleccionado por lo menos una vez en las ‘m’ veces que se selecciona una instancia. Para valores grandes de ‘m’ se aproxima a  $1 - 1/e = 63.2\%$ . Por lo que cada muestra tiene aproximadamente un 63% de aparecer en los ejemplos de entrenamiento. En general, se puede mejorar el resultado si se quita la opción de podado en los árboles de decisión, lo que los hace más inestables.

Algoritmo *K-NN* (*K-Nearest Neighbor*): Para explicar su funcionamiento utilizaremos un ejemplo: Supongamos que tenemos un conjunto de entrenamiento de N vectores de

características. Estos vectores están agrupados en distintas clases ( $C1, C2, C3, \dots$ ). Supongamos también que se introduce en el sistema un nuevo vector de características. Nuestro objetivo sería determinar la clase a la que pertenece. Para determinarla el algoritmo  $K$ - $NN$  busca los  $K$  vecinos más próximos al vector de entrada, y posteriormente realiza una valoración de las clases a las que pertenecen estos vectores vecinos. La valoración consiste en comparar las distancias existentes entre las clases. Los algoritmos utilizados en esta valoración pueden ser distintos, en nuestro sistema hemos utilizado la Distancia Euclídea.

*Adaboost* es un algoritmo utilizado para construir clasificadores sólidos utilizando la combinación lineal de clasificadores simples. El primer paso consiste en generar los ejemplos, a éstos se les asigna el mismo peso ( $1/m$ ). A medida que se genera un nuevo modelo se cambian los pesos de los nuevos ejemplos utilizados para el siguiente clasificador. El objetivo consiste en minimizar en cada iteración el error esperado. Es por ello que se asignan pesos superiores a los ejemplos mal clasificados.

De acuerdo con (alojamientos, 2016) la segmentación de imágenes divide la imagen en sus partes constituyentes hasta un nivel de subdivisión en el que se aíslen las regiones u objetos de interés.

Los algoritmos de segmentación se basan en una de estas dos propiedades básicas de los valores del nivel de gris: discontinuidad o similitud entre los niveles de gris de píxeles vecinos.

**Discontinuidad.** Se divide la imagen basándose en cambios bruscos de nivel de gris:

- a) Detección de puntos aislados
- b) Detección de líneas
- c) Detección de bordes

**Similitud.** Se divide la imagen basándose en la búsqueda de zonas que tengan valores similares, conforme a unos criterios prefijados:

- 1) Crecimiento de región

2) Umbralización.

### **3.7 Introducción a la edición paramétrica de imágenes**

De acuerdo con (Krogh, 2010) es una clase de edición de imagen no destructiva en la que el software de edición no altera los archivos originales, registrando en su lugar los cambios realizados en la imagen, como conjunto de instrucciones o parámetros.

En el mundo de la imagen digital tradicional existe una forma objetivamente correcta de presentar una imagen. Cuando se utiliza con un perfil de color, cada valor RGB de un pixel en un archivo de imagen describe un color exacto.

La cámara captura lo que se conoce como datos con referencia a la escena, o la información de color y brillo creada a partir de la respuesta del sensor de la cámara la luz que está rebotando en el mundo real. Los datos con referencia a la escena son los que crean la imagen sin procesar (una forma de describir la luz que choca contra el sensor de la cámara, pero que no se parecen para nada a una fotografía final).

Para que se parezca a una fotografía, el software tiene que transformar los datos referidos a la escena en datos referidos a la impresión, que es la información del color que parece una fotografía cuando se envía a un monitor o a una impresora.

Un programa puede leer e interpretar una imagen, pero no puede guardar los nuevos datos de pixeles de nuevo en el archivo sin procesar.

Menciona (McMahon & Nichols, 2007) las cámaras digitales se presentan en todos los formatos y resoluciones aunque, para ser justo, la mayoría ofrece características muy similares. Las grandes diferencias se encuentran en las posibilidades de resolución de la cámara, el alcance de sus lentes de ampliación y la capacidad de la tarjeta de memoria.

Si se desea conseguir la más alta calidad de su fotografía con vista a la impresión comercial, se tiene que buscar una cámara con la resolución más alta que su presupuesto pueda permitirse.

Otros factores como la calidad de lentes, dimensiones del sensor de imagen (*CCD*), posibilidades de software de procesamiento (color y contraste) y características físicas de manejo también afectan al resultado.

Los ordenadores son más fáciles de entender. Puesto que las fotografías contienen numerosos datos por lo que necesitan mucha información, contienen numerosos datos por lo que necesitan mucha potencia.

### 3.8 Paradigma en el análisis de imágenes

Según (García F. , 2009) la visión por computadora actualmente comprende tanto la obtención como la caracterización e interpretación de las imágenes. Esto supone algoritmos de muy diversos tipos y complejidades. En un sistema de visión por computador actual se pueden distinguir seis etapas o fases:

- *Captación*: Es el proceso a través del cual se obtiene una imagen visual.
- *Pre procesamiento*: Incluye técnicas tales como la reducción de ruido y realce de detalles.
- *Segmentación*: Es el proceso que divide a una imagen en objetos que sean de nuestro interés.
- *Descripción*: Es el proceso mediante el cual se obtienen características convenientes para diferenciar un tipo de objeto de otro, por ejemplo tamaño y forma.
- *Reconocimiento*: Es el proceso que identifica a los objetos de una escena. Por ejemplo las diferentes tipos de piezas en un tablero de ajedrez.
- *Interpretación*: Es el proceso que asocia un significado a un conjunto de objetos reconocidos.

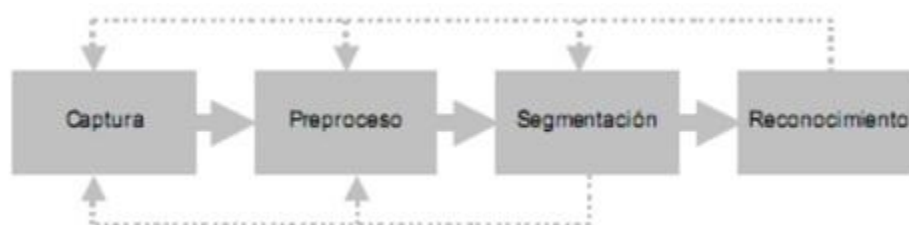
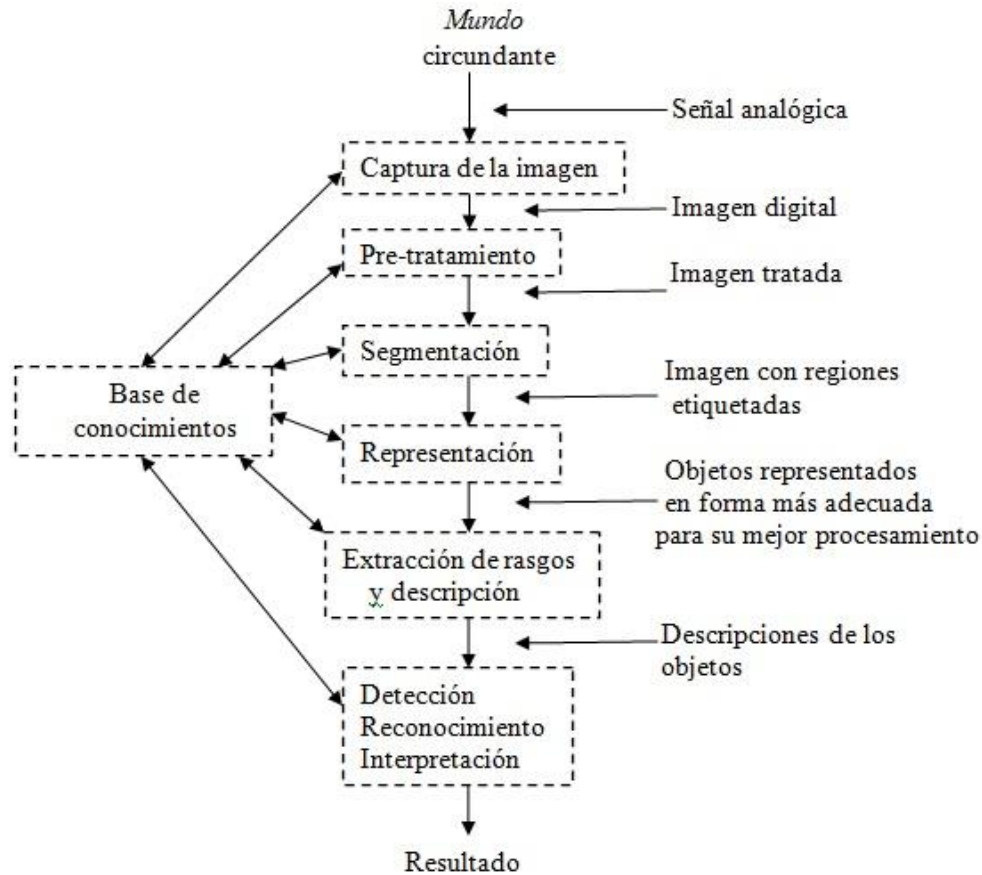


Figura 7: Etapas de un sistema de visión por computadora (García F. , 2009)

Como se muestra en la figura 7 las etapas de un sistema de visión por computadora supone distintos tipos de procesamiento en función del nivel en el que se mueva.

- *Visión de bajo nivel:* comprende la captación y el pre-procesamiento. Ejecuta algoritmos típicamente de filtrado, restauración de la imagen, realce, extracción de contornos, etc.
- *Visión de nivel intermedio:* comprende la segmentación, descripción y reconocimiento, con algoritmos típicamente de extracción de características, reconocimiento de forma y etiquetado de éstas.
- *Visión de alto nivel:* comprende la fase de interpretación, normalmente estos algoritmos se refieren a la interpretación de los datos generalmente mediante procedimientos típicos de la Inteligencia Artificial para acceso a bases de datos, búsquedas, razonamientos aproximados, etc.





*Figura 8: Pasos en un sistema de análisis de imágenes incorporando una base de conocimientos sobre el problema a resolver. (Rodríguez & Sossa, 2012)*

La figura 8 muestra los pasos de un sistema de análisis de imágenes incorporando una base de conocimientos sobre el problema a resolver.

Según (Gonzalez & Woods, Procesamiento Digital de Imágenes, 2002) las etapas son:

### **3.8.1 Etapa 1: Captación de la Imagen**

La cual se puede llevar a cabo a través de variados dispositivos, a saber: una cámara (analógica o digital), un escáner o cualquier otro dispositivo de captación de imágenes.

Los días algo nublados son perfectos para realizar fotografías al aire libre ya que no hacen tan pronunciadas las sombras y ofrecen unos colores bastante neutros.

En situaciones de mucha luz, las sombras y los reflejos pueden estropear cualquier fotografía, por lo que debemos intentar buscar distintos ángulos hasta encontrar el más adecuado.

Menciona (Delgado, 2009) por lo general, las cámaras digitales tienen un mejor comportamiento que las analógicas en situaciones de poca luz. En cualquier caso, es conveniente usar un trípode o apoyar la cámara sobre un lugar firme para evitar que una velocidad lenta de obturación, necesaria para capturar todos los detalles en este tipo de situaciones, provoque una foto movida o poco nítida.

Hay dos tipos principales de cámaras que se utilizan en sistemas de visión por computadoras: Cámaras Matriciales y Cámaras Lineales.

#### *Cámaras Matriciales:*

Término que se refiere a que el sensor de la cámara cubre un área o que está formado por una matriz de píxeles.

Una cámara matricial produce una imagen de un área, normalmente con una relación de aspecto de 4 a 3. Esta relación viene de los tiempos de las cámaras Vidicon y de los formatos de cine y televisión. Actualmente existen muchas cámaras que ya no mantienen esta relación y que no siguen los formatos de la televisión.

*Cámaras Lineales:* El concepto de barrido lineal se asocia a la construcción de una imagen línea a línea, utilizando un sensor lineal de forma que la cámara se desplaza con respecto al objeto a capturar, o bien el objeto se desplaza con respecto a la cámara. La utilización de cámaras lineales sin ser compleja requiere de una mayor experiencia en los entornos de visión que la utilización de cámaras matriciales.

La tecnología de cámaras lineales hace mucho tiempo que fue desarrollada para aplicaciones de inspección de materiales fabricados en continuo, como papel, tela, planchas metálicas, etc. Sin embargo en la actualidad se está imponiendo en muchos otros procesos productivos y de inspección, que requieren alta resolución y / o alta velocidad a un precio competitivo.

Las cámaras lineales utilizan sensores lineales que acostumbran a tener entre los 512 y 8192 elementos (píxeles), con una longitud lo más corta posible, y con una gran calidad con el fin de obtener la mejor sensibilidad y prestaciones.

Con la tecnología de cámaras lineales es posible capturar objetos de grandes dimensiones en una sola pasada, mientras que con cámaras matriciales este mismo objeto debería ser dividido en una secuencia de imágenes parciales

De acuerdo con (Valera, 2015) muchos son los aspectos que pueden influir en la elección de una cámara a otra. Sin embargo, algunas consideraciones han de tenerse en cuenta para no arrepentirse de la compra o para no gastarse más dinero del que realmente hace falta. Normalmente, la elección de la cámara va íntimamente ligada a la de la óptica.

Las características técnicas son las más importantes y las que siempre se tendrá presente a la hora de adquirir una cámara. Por ejemplo:

- **Formato de video:** El formato de la señal es de suma importancia ya que tanto las cámaras como las digitalizadoras y, a veces, los monitores deben ser del mismo formato. El formato cambia de un país a otro.
- **Resolución:** Un factor a tener en cuenta es el número de bits con que se digitaliza el píxel. Lo normal en cámaras digitales es emplear 8 bits, aunque existen cámaras de hasta 16 bits por píxel.

Para la adquisición digital de imágenes se necesitan dos elementos. El primero de ellos es un dispositivo físico sensible a una determinada banda del espectro de energía electromecánica (como las bandas de rayos X, ultravioleta, visible o infrarrojo) y que produzca una señal eléctrica de salida proporcional al nivel de energía detectado.

El segundo, denominado *digitalizador*, es un dispositivo para convertir la señal de salida del sistema sensible a forma digital. Ésta es transmitida hasta el computador, ya sea para su monitorización o procesamiento.

### **3.8.2 Etapa 2: Pre procesamiento:**

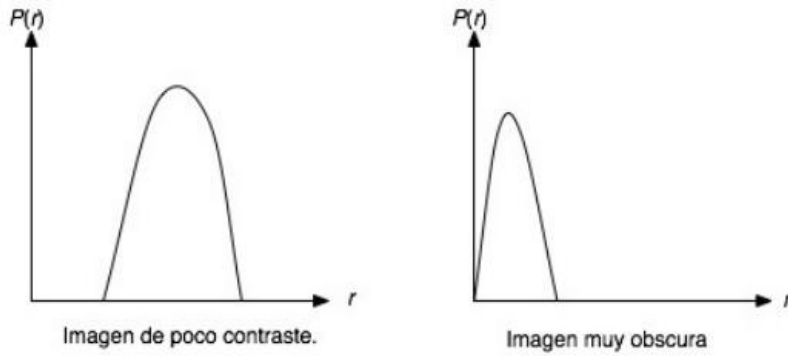
Según (Rodríguez & Sossa, 2012) la cual no es siempre necesario llevar a cabo. Sin embargo, en muchas aplicaciones esta etapa tiene un peso importante y básicamente consiste en la atenuación del ruido presente en la imagen, el mejoramiento del contraste, así como filtrados para la eliminación de artefactos, entre otros.

Menciona (Escalante, 2006) los procesos de realce de imágenes consisten de una serie de técnicas cuyo objetivo es mejorar la apariencia visual de una imagen, ya sea en contraste, ruido, escala de grises, distorsiones, luminosidad, falta de nitidez, etc., o bien convertir o mapear la imagen a una mejor forma para su análisis. El principal objetivo del realce de la imagen es procesar una imagen de tal manera que el resultado obtenido sea el apropiado para una aplicación específica. Los métodos de realce de imágenes se pueden dividir en dos categorías: los métodos de realce en el dominio espacial y los métodos de realce en el dominio de la frecuencia. Los métodos de la primera categoría consisten en la manipulación directa de los píxeles de la imagen mientras que los métodos de la segunda categoría corresponden a técnicas basadas en la representación de los píxeles, a través de una transformación hacia el dominio frecuencial y usa como operador de mapeo o transformación a la DFT.

Dentro de los métodos de realce espacial se pueden encontrar dos técnicas: El realce radiométrico donde las operaciones son efectuadas directamente sobre un píxel sin importar o tomar en cuenta a los píxeles vecinos. Sirve para mejorar condiciones de bajo contraste, baja luminosidad o demasiada oscuridad. Ejemplo: ecualización de histograma.

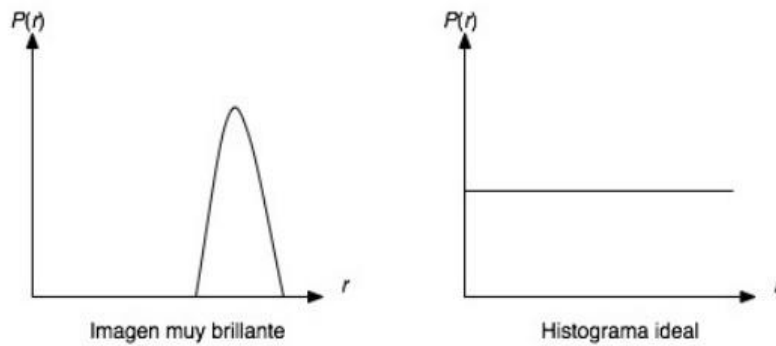
El realce con operaciones entre vecinos que, a diferencia del realce radiométrico, las operaciones son efectuadas sobre un píxel pero tomando en cuenta a los píxeles que lo rodean. Sirve para eliminar ruido o para el mejoramiento de la nitidez. Ejemplo: kernel correspondiente a un filtro paso-bajas usando la convolución para realizar un filtrado espacial.

El *histograma* de una imagen es la representación gráfica de la distribución que existe de las distintas tonalidades de grises con relación al número de píxeles o porcentaje de los mismos. La representación de un histograma ideal sería la de una recta horizontal, ya que eso indicaría que todos los posibles valores de grises están distribuidos de manera uniforme en la imagen.



**Figura 9: Histograma de una imagen. (Escalante, 2006)**

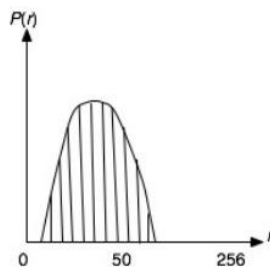
La figura 9 muestra gráficamente la diferencia entre una imagen con poco contraste y una imagen oscura.



**Figura 10: Tipos de histograma de una imagen. (Escalante, 2006)**

En la figura 10 se muestra gráficamente la diferencia entre una imagen con mucho brillo y el histograma ideal para una imagen.

El histograma de una imagen digital con niveles de gris en el rango  $[0, L - 1]$  es una función discreta  $P(r) = nr$  donde  $r$  representa un nivel de gris y  $nr$  representa el número de pixeles que tienen ese valor de gris como se muestra en la figura 11:



*Figura 11: Histograma de una imagen digital (Escalante, 2006)*

Es muy frecuente normalizar un histograma dividiendo cada uno de sus valores por el número total de píxeles en la imagen, denotado por  $n$ . De tal modo que el histograma nos quedaría  $P(rk) = nk/n$ , para  $k = 0, 1, \dots, L-1$ . De esta manera,  $P(rk)$  da una estimación de la probabilidad de la ocurrencia del nivel de gris  $rk$ . Hay que notar que la suma de todos los componentes de un histograma normalizado es igual a 1.

### **3.8.2.1 Filtros**

Según (Gonzalez & Woods, Procesamiento Digital de Imágenes, 2002) los filtros espaciales tienen como objetivo modificar la contribución de determinados rangos de frecuencias de una imagen. El término *espacial* se refiere al hecho de que el filtro se aplica directamente a la imagen y no a una transformada de la misma, es decir, el nivel de gris de un píxel se obtiene directamente en función del valor de sus vecinos.

Los filtros espaciales pueden clasificarse basándose en su linealidad: filtros lineales y filtros no lineales. A su vez los filtros lineales pueden clasificarse según las frecuencias que dejen pasar: los filtros paso bajo atenúan o eliminan las componentes de alta frecuencia a la vez que dejan inalteradas las bajas frecuencias; los filtros paso alto atenúan o eliminan las componentes de baja frecuencia con lo que agudizan las componentes de alta frecuencia; los filtros paso banda eliminan regiones elegidas de frecuencias intermedias.

La *convolución* es una operación por la cual se lleva a cabo una acción de filtrado. Como veremos más adelante y como se ha visto en las propiedades de la Transformada de Fourier, existe una relación entre el filtrado espacial y el filtrado en el dominio de la frecuencia (con una restricción a tomar en cuenta). En general, el filtrado de señales posee tres categorías, según el resultado que se busque:

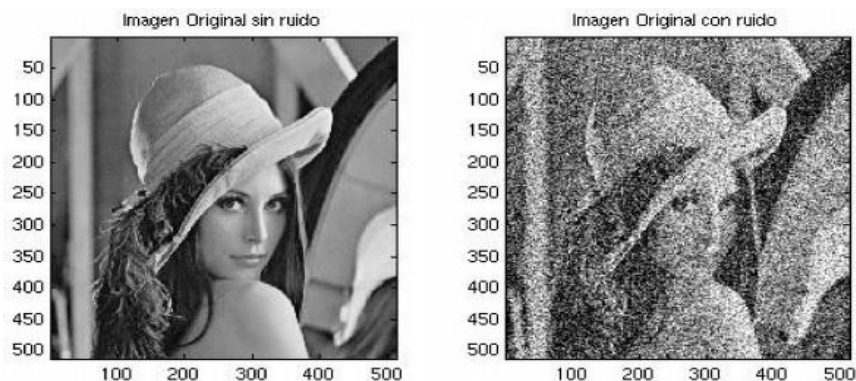
1) *Filtros Paso-Bajas*: Son utilizados en la reducción de ruido; suavizan y aplanan un poco las imágenes y como consecuencia se reduce o se pierde la nitidez. En inglés son conocidos como *Smoothing Spatial Filters*.

2) *Filtros Paso-Altas*: Estos filtros son utilizados para detectar cambios de luminosidad. Son utilizados en la detección de patrones como bordes o para resaltar detalles finos de una imagen. Son conocidos como *Sharpening Spatial Filters*.

3) *Filtros Paso-Banda*: Son utilizados para detectar patrones de ruido. Ya que un filtro paso-banda generalmente elimina demasiado contenido de una imagen casi no son usados. Sin embargo, los filtros paso-banda son útiles para aislar los efectos de ciertas bandas de frecuencias seleccionadas sobre una imagen. De esta manera, estos filtros ayudan a simplificar el análisis de ruido.

*Suavizado*:

De acuerdo con (Escalante, 2006) los filtros paso-bajas son utilizados para difuminar y reducir ruido en las imágenes, a este proceso se le conoce en inglés como *smoothing*. El difuminado (*blurring*) es usado en etapas de pre procesamiento desde la eliminación de pequeños detalles hasta la extracción de objetos y rellenado de pequeños huecos en líneas y curvas. La reducción de ruido puede ser completada por el difuminado usando filtros lineales o bien con un filtrado no lineal. Ver figura 12.



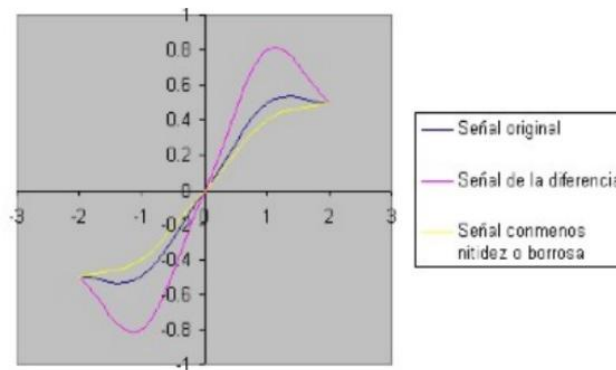
*Figura 12: Imagen original y con ruido. (Escalante, 2006)*

La salida (respuesta) de un filtro paso-bajas lineal simplemente es un tipo de promedio de los píxeles contenidos en la vecindad de la máscara del filtro. Estos filtros son frecuentemente

llamados filtros promediadores. La idea detrás de estos filtros es la de reemplazar cada pixel en la imagen por un promedio de los niveles de gris de los vecinos definidos por la máscara del filtro.

*Los filtros basados en derivadas de Gaussianas* fueron en un principio obtenidos de manera heurística. Sin embargo estos filtros tienen un fundamento matemático y un comportamiento muy similar al sistema de visión humano (HVS); por ello son muy importantes ya que se especializan en la detección de cambios bruscos como los bordes.

El mejoramiento de la nitidez o de la calidad visual de una imagen basado en los filtros *unsharp masking* tiene bastante importancia en el procesamiento digital de imágenes. Si se quisiera traducir al español, *unsharp masking* podría interpretarse como enmascaramiento (*masking*) de imagen borrosa (*unsharp*). Se basa en el hecho de que se tiene una imagen borrosa (con pendiente pequeña), y se le resta una pendiente aún más pequeña. A su vez esto va multiplicado por un factor  $k$  el cual se recomienda tome valores entre 1 y 3 como se muestra en la figura 13.

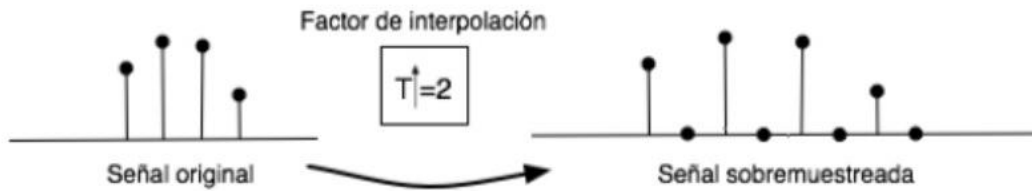


**Figura 13: Cambios en la señal de una imagen. (Escalante, 2006)**

*Interpolación:* El proceso de interpolación está relacionado con el hecho de desear cambiar la tasa de muestreo de una señal discreta a una tasa menor o mayor con la que fue muestreada originalmente. El proceso con el cual se disminuye la tasa de muestreo de una señal discreta por un factor  $D$  se denomina decimación o submuestreo (*downsampling*) y el proceso con el cual se aumenta la tasa de muestreo de una señal discreta por un factor  $T$  se denomina interpolación (*upsampling*).



Una de las más simples formas de interpolación es elegir la amplitud de un pixel de salida como la amplitud de su vecino más cercano. Esta interpolación es llamada interpolación del vecino más cercano o bien interpolación de orden cero. El factor de interpolación  $T$  indicará cuantos pixeles se desean interpolar. Por ejemplo, en una dimensión podríamos ver lo siguiente, se tiene una señal compuesta por cuatro muestras y se desea interpolar con un filtro  $h(x)$  de orden cero con un factor de interpolación  $T = 2$  como se muestra en la figura 14:



*Figura 14: Señal original y sobremuestreada (Escalante, 2006)*

La señal interpolada será el resultado de hacer la convolución de la señal original sobre muestreada (agregando ceros) con el filtro interpolador.

### *Transformaciones de intensidad*

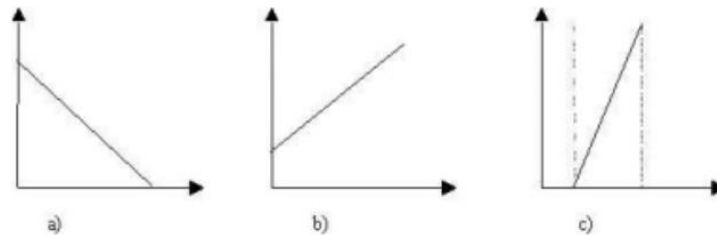
De acuerdo con (Sucar & Gomez, 2016) una *transformación de intensidad* consiste en mapear los valores de intensidad de cada pixel a otros valores de acuerdo a cierta función de transformación. Las funciones de transformación pueden ser de dos tipos:

1. lineales.
2. no-lineales.

En las *transformaciones lineales*, se tiene una relación o función lineal de los valores de intensidad de los pixeles de la imagen de salida respecto a la imagen de entrada. Los tipos de transformaciones lineales más comúnmente utilizados son:

- Obtener el negativo de una imagen.
- Aumentar o disminuir la intensidad (brillo de la imagen).
- Aumento de contraste

Las funciones de transformación para cada uno de estos tipos se especifica gráficamente en la figura 15. Por ejemplo, para el negativo, el pixel de entrada (*eje X*) de intensidad 0 se transforma en un pixel de salida (*eje Y*) de intensidad máxima, y el de entrada de intensidad máxima se transforma en intensidad 0.



**Figura 15: Transformaciones lineales (Sucar & Gomez, 2016)**

Transformaciones lineales. (a) Negativo. (b) Aumento de intensidad. (c) Aumento de contraste.

Las *transformaciones no-lineales* normalmente son funciones monotónicas de forma que mantienen la estructura básica de la imagen. Algunos ejemplos de transformaciones no-lineales son los siguientes:

- Expansión (o aumento) de contraste. Se incrementa el contraste, en forma diferente para distintos rangos de intensidades.
- Compresión de rango dinámico. Se reduce el rango de niveles de gris o intensidades de la imagen.
- Intensificación de un rango de niveles. Se aumenta la intensidad de un rango de niveles de gris de la imagen.

#### *Aumento lineal del contraste*

Utilizando el valor de intensidad mínimo y máximo en una imagen, podemos aumentar su contraste. La idea básica es llevar el valor mínimo (*min*) a cero y el máximo (*máx*) a 255, pensando en imágenes monocromáticas (0-255). Esta transformación genera que las

intensidades se espacien de acuerdo a cierto factor o pendiente; el factor para este aumento lineal de contraste es:

$$C(x, y) = \left( \frac{I(x, y) - \min}{\max - \min} * 255 \right) \quad (3)$$

Donde  $I(x, y)$  es la imagen a procesar y  $C(x, y)$  es la imagen con aumento lineal del contraste. Se puede verificar fácilmente que para  $I(x, y)$  en *min*,  $C(x, y)$  resulta cero (el numerador es cero); para  $I(x, y)$  en *máx*,  $C(x, y)$  resulta en 255 (cociente 1).

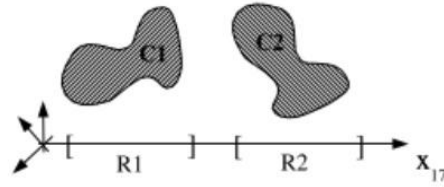
### *Selección de Propiedades*

Menciona (Ruiz, 2015) es un problema de búsqueda combinatoria. Dada una medida de calidad de un conjunto de propiedades (p. ej., el error de clasificación obtenido por un método determinado sobre un conjunto de ejemplos de prueba), tenemos que encontrar el subconjunto que posee mayor calidad. Como hay un número exponencial de subconjuntos, este tipo de selección ‘global’ de propiedades es computacionalmente intratable.

Si la medida de calidad es ‘monótona’ (esto significa que la incorporación de una nueva propiedad no empeora la calidad de un subconjunto, lo que cumplen algunos algoritmos de clasificación, pero no todos) se puede realizar una búsqueda eficiente, ordenada, basada en *branch and bound*. (Se construye un árbol de búsqueda eliminando propiedades en cada rama; no se desciende por ramas en las que se garantiza un empeoramiento de la calidad).

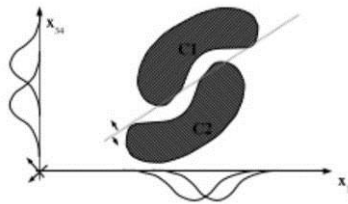
Una alternativa es la selección ‘individual’: elegimos una medida de calidad de cada propiedad por sí misma (sencilla de calcular y que indique lo mejor posible la ‘separación’ de las clases de interés), y nos quedamos con las mejores propiedades que sean, a la vez, estadísticamente independientes.

La selección individual de propiedades es, aparentemente, una idea muy prometedora. La calidad del conjunto será tan buena, al menos, como la del mejor componente, y, posiblemente, mejor. Si encontramos una propiedad discriminante, el problema de clasificación puede considerarse resuelto tal como se muestra en la figura 16:



*Figura 16: Selección de propiedades. (Ruiz, 2015)*

El problema está en que la selección individual puede descartar propiedades poco informativas por separado pero que usadas en combinación son altamente discriminantes como se muestra en la figura 17:



*Figura 17: Selección individual (Ruiz, 2015)*

Además, la selección individual produce a veces resultados poco intuitivos: p. ej., dadas tres propiedades  $p1$ ,  $p2$  y  $p3$ , ordenadas de mayor a menor calidad individual, existen situaciones (debido a las dependencias entre ellas) en que el conjunto  $\{p1, p2\}$  es peor que el conjunto  $\{p2, p3\}$ . Las medidas de calidad individual pueden ser útiles para fabricar árboles de clasificación.

#### *Tipos de textura:*

Define (Valentine & Moughamian, 2010) los siguientes conceptos:

*1.1.- Difusa:* Estos son los datos de pixeles actuales colocados sobre la malla de la superficie 3D. La textura puede ser una imagen o un color sólido.

*1.2.- Auto iluminación:* Es un color que no necesita las luces de la escena para verse. Le da al objeto el efecto de estar iluminado desde dentro, o resplandecer. Observe que si los lados del objeto son 100 por 100 opacos, esta textura o tendrá efecto.

1.3.- *Rugosidad*: Es un mapa de rugosidad que afecta a la topología local o al relieve de la superficie del modelo. Para los modelos creados como mallas a partir de capas de escala de grises, también tendrá acceso al mapa de profundidad.

1.4.- *Resplandor*: Este mapa define la cantidad de luz que refleja la superficie de un objeto. Negro es la sombra más reflectante, mientras que blanco es el menos.

1.5.- *Brillo*: Es diferente de Resplandor en que define las propiedades especulares de la superficie del objeto, o la dispersión de la luz reflejada. Con valores altos produce una reflexión pequeña de baja dispersión, mientras que valores bajos crean una reflexión dispersa más difusa.

1.6.- *Opacidad*: Controla la opacidad local de un objeto. Las regiones blancas son 100 por 100 opacas, mientras que las regiones negras son 100 por 100 transparentes.

1.7.- *Reflectividad*: Cambia la capacidad de la superficie del material para reflejar otros objetos en la escena y el mapa del entorno.

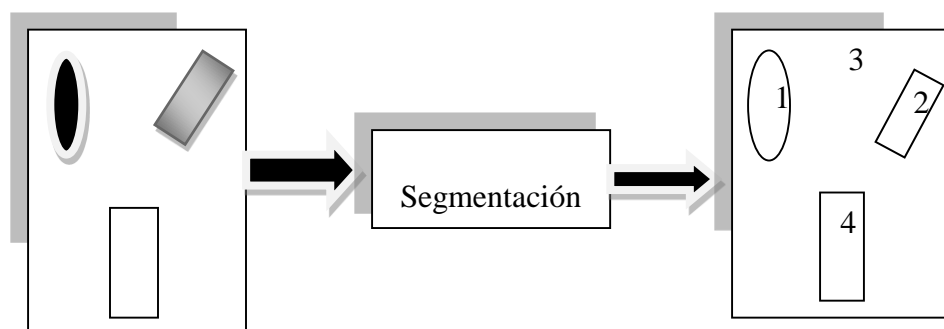
1.8.- *Entorno*: Este mapa contiene la imagen del "mundo" alrededor del objeto. Es lo que reflejara un objeto si no hay otros objetos en la escena. El mapa de entorno se hace como un panorama esférico.

### **3.9 Etapa 3: Segmentación**

De acuerdo con (Rodríguez & Sossa, 2012) la segmentación es una de las etapas cruciales de cualquier sistema de análisis de imágenes, tanto por las dificultades que conlleva como por la importancia de sus resultados. Básicamente, la segmentación puede considerarse como la partición de una imagen, digamos  $f(x, y)$  en un conjunto de regiones  $R$  no solapadas, homogéneas respecto a algún criterio cuya unión cubra la imagen completa. En otras palabras, el objetivo fundamental de la segmentación, en el proceso del análisis de imágenes, es el de separar los objetos de interés del resto no relevante el cual es considerado como fondo.

En ocasiones la segmentación suele considerarse como un proceso de clasificación de los objetos presentes en una imagen y también, en cierta medida, equivale a su reconocimiento

puesto que como consecuencia de la segmentación los diferentes objetos (entendidos como realizaciones físicas de clases o patrones abstractos) se encuentran perfectamente ubicados dentro de la imagen digital. El realizar dicha clasificación no es una tarea trivial. El nivel al que se lleva a cabo esta subdivisión depende del problema a resolver y, por supuesto, de la imagen de entrada. Es por ello que se han originado diferentes técnicas de segmentación. Hasta el presente no se tiene conocimiento de una técnica única que pueda ser utilizada para segmentar cualquier imagen. A decir de muchos autores la segmentación termina cuando se satisfacen los intereses u objetivos de la aplicación.



*Figura 18: Esquema que representa el proceso de segmentación. (Rodríguez & Sossa, 2012)*

De manera gráfica en la figura 18 se presenta el proceso de segmentación de una imagen simple. De dicha figura se aprecia que en virtud de la segmentación se ha pasado de una imagen digital bruta, con toda su información en forma de los niveles de intensidad luminosa, a una imagen mucho más simplificada en lo que deben de estar nítidamente distinguidos entre sí los diferentes objetos existentes.

En general, los métodos clásicos de segmentación se pueden categorizar como se indica a continuación:

*Métodos basados en el umbralado a partir del histograma de la imagen:* En este caso, a partir del histograma de una imagen es posible obtener un umbral de comparación para el agrupamiento de los píxeles.

*Métodos basados en la detección de discontinuidades:* En este caso, la imagen en cuestión es dividida a partir de cambios bruscos de los niveles de grises.

*Métodos basados en la propiedad de similitud de los valores de los niveles de grises:* En este caso se usan criterios de homogeneidad para la agrupación de los píxeles.

*Método heurístico de segmentación:* Estos métodos basan su operación en el conocimiento previo de la imagen a segmentar y en la experiencia del observador, e incluyen en muchas ocasiones los métodos supervisados de segmentación.

### **3.9.1 Segmentación por el clasificador de Bayes**

Menciona (Larrañaga, Inza, & Moujahid, 2016) en muchas ocasiones cuando se utiliza un clasificador automático basado, por ejemplo, en la distancia Euclideana se utilizan prototipos fijos. Estos prototipos representan cada una de las clases u objetos a clasificar. Al hacer esto se está suponiendo una hipótesis determinística en el comportamiento de los elementos de las clases. Sin embargo, en aquellas situaciones en las cuales los vectores de algunas de las clases presentan una dispersión significativa con respecto a su media es conveniente utilizar un enfoque estadístico en vez de una hipótesis determinística. Así, por ejemplo, si se supone que se tienen dos tipos de frutas, por ejemplo naranjas y plátanos, las cuales pasan delante de un sistema de captación de imágenes con el objetivo de ser reconocidas para posteriormente desviarlas a diferentes lugares según las clases a la que pertenezca cada fruta.

### **3.9.2 Formula de Bayes aplicada a la segmentación supervisada**

Menciona (Diaz, Montoya, & Boulanger, 2007) sea un elemento con un valor  $v$  y para cada clase  $i$  sea  $q(i/v)$  la probabilidad de que el elemento sea de la clase  $i$ , entonces la mejor clase para asignar dicho elementos es la clase para la cual  $q(i/v)$  es máxima. Esto viene dado por la expresión siguiente:

$$q(v/i)q(i) \tag{4}$$

---

$$q(i/v) = q(v) \tag{5}$$

Dónde:  $q(i)$  es la probabilidad a priori de que cualquier elemento pertenezca a la clase  $i$ . Esta es la probabilidad que se asume conocida como antes del comienzo de la segmentación o clasificación.

$q(v/i)$  es la probabilidad de que dada la clase  $i$ , el valor de la variable aleatoria tenga precisamente el valor  $v$ . Esta es la función de densidad de probabilidad condicional para la clase  $i$ , o la probabilidad que un elemento de la clase  $i$  tenga valor  $v$ .

$q(i/v)$  es la probabilidad de que un elemento con valor  $v$  este en la clase  $i$ .

$q(v)$  es la suma de las probabilidades  $q(v/i)$  sobre toda las clases  $i$ . Esto se puede considerar sencillamente como un factor de normalización o un factor de escala.

La expresión anterior aporta la solución al problema de la segmentación de imágenes ya que dado el elemento con valor  $v$ , el cual también puede ser considerado como un vector descriptivo, este pertenece a la clase  $j$  si se cumpla la siguiente condición:

$$v \in \text{clase } j \text{ si } p(\text{clase } j/v) > p(i/v) \quad (6)$$

$$\forall i \neq j, i = 1, 2, 3, \dots, N \quad (7)$$

Los pasos genéricos del algoritmo para segmentar una imagen según el clasificador de Bayes son los siguientes:

- 1.- Seleccionar las áreas de entrenamiento para cada una de las clases candidatas. Estas áreas se suponen conocidas y, en general, se debe escoger una gran cantidad de datos por clase con el fin de que estos representen lo mejor posible dicha clase.
- 2.- Calcular el histograma  $H_{i(v)}$  de los datos de entrenamiento para cada una de las áreas seleccionadas y normalizarlos. Esta estimación es usada como una función de densidad de probabilidad condicional:  $q(v/i)$ . Existe una  $q(v/i)$  para cada una de las clases  $i$  seleccionadas.
- 3.- Estimar la probabilidad a priori  $q(i)$  la cual será usada como un factor de escala



4.- Clasificar cada uno de los píxeles de la imagen calculando, para la clase  $i$ ,  $q(i/v)$  mediante la expresión anterior. En la práctica se omite el denominador en la expresión anterior.

5.- Clasificar el valor del píxel en la clase  $i$  si el valor de  $q(i/v)$  es máximo. Los pasos 1, 2 y 3 se realizan una sola vez, el paso 4 se aplica para cada uno de los píxeles de la imagen para producir la imagen segmentada.

Los métodos de segmentación supervisados han sido muy utilizados en la teledetección de los recursos naturales donde muchas veces se tiene, a partir de las pruebas de campo realizadas, algún conocimiento del terreno y de los objetos.

Otros métodos clásicos de segmentación supervisada se han reportado en la literatura. Como ejemplos se tienen el método basado en el cálculo de los centroides y el método basado en el cálculo de la mínima distancia.

Una técnica clásica opuesta a la de crecimiento de regiones es la de división y fusión de regiones. En este caso se comienza bajo el supuesto que la imagen completa es homogénea, de no cumplirse esta condición acorde algún criterio, la imagen se divide en 4 sub imágenes. En forma iterativa cada una de las sub imágenes es subdividida en otras 4 sub imágenes si no satisface dicho criterio. Al final se tiene una imagen dividida en sub imágenes cuyos elementos satisfacen el criterio de homogeneidad establecido.

### **3.9.3 Detección de bordes**

Según (Martin, 2013) un *borde* puede definirse como un cambio significativo en el valor de la intensidad de los píxeles en una región de la imagen. Durante el proceso de detección de bordes se puede obtener la siguiente información:

- Orientación local de los elementos del borde.
- Intensidad de los elementos del borde, es decir, contraste en la brillantez entre regiones vecinas.
- Ancho de los elementos del borde (puesto que los bordes no son pasos ideales, estos en general varían considerablemente en su ancho).

- Colocación de la representación del borde (puesto que pueden tener más de un pixel de ancho, es importante determinar dónde poner el punto del borde).
- Polaridad del elemento del borde (cuál de las regiones es la más brillante).
- Valor de gris del elemento del borde.
- Valor de gris de las regiones del entorno.

En la detección de bordes surgen problemas durante el proceso de selección, a saber:

- Ruido (dado por los pixeles del fondo u otros factores)
- Manchas o emborronamiento (algunas partes del contorno son más espesas o densas que otras).
- Fragmentación (algunas partes del contorno se pierden).
- Mala colocación de los puntos.

Es importante tener en cuenta que un buen detector de bordes debe contar con las propiedades siguientes:

- Debe operar localmente.
- Debe ser sensitivo a la orientación y magnitud del lado.
- Debe trabajar adecuadamente en presencia de ruido.
- Debe ser insensible al valor del umbral.
- No debe dar múltiples respuestas a un mismo borde

En la práctica, debido a la presencia del ruido y a otros problemas en la captación de la escena, es muy poco probable que los detectores de bordes clásicos cumplan en forma eficiente con los puntos anteriormente expuestos. De ahí, la necesidad de la propuesta de nuevas técnicas de detección de bordes menos rígidas.

La importancia que tiene la determinación de los contornos de una figura dentro de una imagen radica en que muchos algoritmos requieren que se demarquen los objetos que están contenidos en una imagen. Algunas de las aplicaciones más importantes que requieren de ésta operación como paso previo son las siguientes:

- Conteo de objetos
- Localización de objetos
- Medición de las características métricas de los objetos
- Determinación de las características geométricas de los objetos
- Discriminación por tamaño o forma de objetos
- Reconocimiento óptico de caracteres (OCR: *Optical Character Recognition*).

Es posible formular otros modelos para el cálculo de bordes a partir del concepto de derivada.

### **3.9.4 Operador de Canny**

Menciona (ehu, 2016) el operador de Canny basa su operación en los siguientes criterios:

1. *Buena detección:* El detector debe distinguir con suficiente Margen entre bordes verdaderos y bordes falsos. Esto implica, por un lado, marcar con alta probabilidad puntos tipo bordes reales. Por otro lado, el detector debe marcar, con baja probabilidad, puntos que no son bordes.

Este criterio corresponde a maximizar la relación señal/ruido.

2. *Buena localización:* Los puntos marcados como bordes deben corresponder en lo posible con los centros de los bordes verdaderos.
3. *Respuesta única:* En lo posible, el detector debe responder con un solo borde. Esto, de alguna manera, queda implícito en el primer criterio, ya que cuando existen dos respuestas para un mismo borde una de ellas se debe considerar como falsa. Sin

embargo, la forma matemática del primer criterio no toma en cuenta el requisito de múltiples respuestas y por tanto esto se debe especificar de manera explícita.

### **3.10 Técnicas actuales de segmentación**

Menciona (Ramiro, 2016) la segmentación que utiliza técnicas tradicionales, tales como: el umbralado, por cálculo de gradiente, por crecimiento de regiones y otras operaciones clásicas, requieren de una considerable cantidad de iteraciones para la obtención de resultados satisfactorios. La automatización de estos modelos libre de aproximaciones es difícil debido a muchos factores, entre otros a la complejidad de los objetos en escena, los tipos de iluminación, las sombras producidas por el acomodo de los objetos y la forma en que estos son iluminados.

Además el ruido y otros artefactos pueden originar regiones incorrectas o discontinuidades en las fronteras de los objetos extraídos. Tomando en cuenta todo esto se han propuesto métodos más elaborados que, en muchos casos, superan a los métodos tradicionales. Enseguida se exponen algunos de estos métodos.

#### **3.10.1 Modelos deformables**

De acuerdo con (Rodríguez & Sossa, 2012) estos modelos son capaces de acomodarse a la frecuente variabilidad que tienen los objetos y además soportan altamente el mecanismo de interacción intuitiva, el cual permite que el especialista pueda aportar su experiencia cuando el modelo de interpretación de imagen así lo requiera. Los modelos existentes pueden agruparse en general en dos grandes grupos: modelos deformables de formas activas y modelos deformables de contornos activos. Por su uso más variado y práctico.

*Los modelos de contorno activo* son curvas definidas dentro del dominio de la imagen, las cuales pueden moverse bajo la influencia de fuerzas internas que se producen dentro de la misma curva y de fuerzas externas computadas sobre los datos de la imagen. Las fuerzas externas e internas son definidas de forma tal que los contornos activos vayan acomodándose en forma iterativa a la frontera de un objeto u otras características deseadas dentro de la imagen. Los contornos activos son ampliamente usados en muchas aplicaciones, incluyendo la detección de bordes, el modelado de formas, el seguimiento de movimiento, entre otras.

Existen básicamente dos tipos de modelos deformables de contornos activos, a saber: los contornos activos paramétricos y los contornos activos geométricos.

*Los modelos deformables paramétricos* son curvas y superficies explícitamente representadas en sus formas paramétricas durante la deformación. Los modelos deformables geométricos manejan, de forma natural, los cambios topológicos en la imagen. Sin embargo, a pesar de esta diferencia fundamental los principios que subyacen en ambos modelos son muy similares.

*Media desplazada (“MEAN SHIFT”)*

Las tareas de análisis de imágenes de alto nivel son verdaderamente difíciles y pueden llevar a resultados no reales o incorrectos. Esto se debe básicamente a que las técnicas empleadas utilizan modelos paramétricos inadecuados que frecuentemente presuponen que el usuario “adivina” correctamente los valores de los parámetros de afinación. Para mejorar el rendimiento de estas técnicas la ejecución de las tareas deben ser manejables; es decir, apoyadas por información independiente de alto nivel.

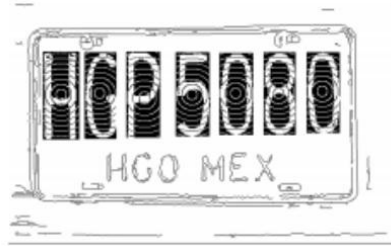
La *media desplazada* es un procedimiento no paramétrico para el análisis de datos multimodales, que ha demostrado su superioridad en varias aplicaciones.

Un *modelo computacional* basado en el cálculo de la media desplazada es una herramienta extremadamente versátil para el análisis del espacio de características y puede dar una solución confiable para muchas tareas de análisis de imágenes.

### **3.11 Etapa 4: Representación**

Según (Sucar & Gomez, 2016) su objetivo principal es llevar los datos de cada uno de los objetos o regiones segmentados a formas en que la computadora pueda trabajar con ellos de manera más apropiada; una lista ligada circular que presente la información del contorno de la región de un objeto es más fácil de trabajar que un conjunto de pares de coordenadas de los píxeles del contorno de la misma región.

Una *representación* es “un sistema formal para hacer explícitas ciertas características o tipos de información, junto con una especificación de como el sistema hace esto” (definición de David Marr). Ver Figura 19. Hay dos aspectos importantes de una representación para visión:



*Figura 19: Reconocimiento de caracteres en base a su codificación radial. (Sucar & Gomez, 2016)*

1. *Representación del modelo.* El tipo de estructura utilizada para modelar la representación interna del mundo.

2. *Proceso de reconocimiento.* La forma en que dicho modelo y la descripción de la imagen es utilizada para el reconocimiento.

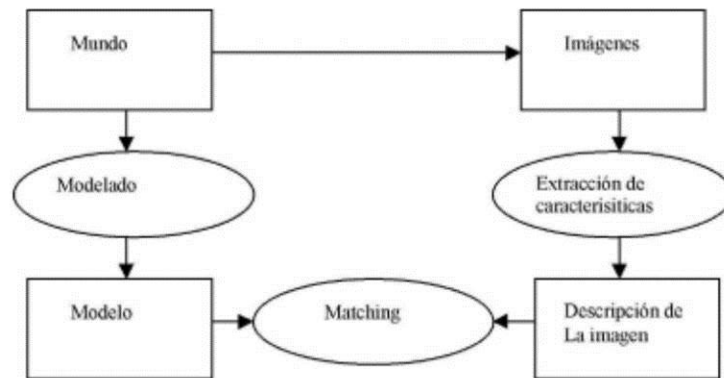
Las representaciones apropiadas para reconocimiento en visión deben de buscar tener las siguientes propiedades:

- Genéricas
- Eficientes, en espacio y tiempo
- Invariantes, independientes de traslación, rotación y escalamiento
- Robustas, tolerantes a ruido e información incompleta.

La estructura general de un sistema de visión basado en modelos se muestra en la figura 20. Tiene 3 componentes principales:

- Extracción de características: Obtención de información de forma de la imagen para construir una descripción geométrica.
- Modelado: Construcción de los modelos geométricos internos de los objetos de interés (a priori).

- Correspondencia o *Matching*: apareamiento geométrico de la descripción con el modelo interno.



*Figura 20: Estructura de un sistema de visión basado en modelos. (Sucar & Gomez, 2016)*

Las técnicas para correspondencia dependen del tipo de representación. Para modelos que utilizan parámetros globales se utilizan técnicas de reconocimiento estadístico de patrones. Con modelos en base gráficas relacionales, se usan algoritmos de grafos (isomorfismo). En modelos paramétricos se aplican técnicas de optimización paramétrica.

### 3.11.1 Modelos en dos dimensiones

Menciona (Gambini, 2006) los modelos en dos dimensiones (*2-D*) están orientados al modelado y reconocimiento de objetos en función de su representación a nivel imagen, es decir, en dos dimensiones. Para representar un objeto en *2-D*, existen básicamente dos alternativas:

- *Contornos*. El objeto se representa en base a su borde o contorno.
- *Regiones*. El objeto se representa en base a la región que define.

### 3.11.2 Polilíneas

Según (Gambini, 2006) la representación de *polilíneas* consiste en una descripción de contornos en base a segmentos de línea, donde cada segmento (*X*) se especifica mediante el punto inicial y final. La concatenación de estos puntos, con el mismo punto inicial y final, describe un contorno:

$$X_1 X_2 \dots X_n, X_1$$

(8)

Donde  $X_i$  corresponde a las coordenadas  $x$ ,  $y$  de cada punto.

### 3.12 Etapa 5: Descripción

Menciona (Quiles & Garrido, 1996) el objetivo principal consiste en capturar las diferencias esenciales entre objetos pertenecientes a clases diferentes; por supuesto se buscaría que estos mismos rasgos se mantuvieran lo más invariantes ante cambios como escalamientos, traslaciones y rotaciones. Es el proceso mediante el cual se obtienen características convenientes para diferenciar un tipo de objeto de otro, por ejemplo tamaño y forma.

La mayoría de procesamiento de imágenes se aplica directamente sobre imágenes en escala de grises, debido al bajo consumo de cómputo de éste, la mayoría de métodos matemáticos deterministas que se utilizan en su procesamiento, están basados en la diferencia de niveles de grises, por lo que no existen muchos métodos para procesamiento de imágenes en color, aun cuando estos pueden ser utilizados en este tipo de formato de imagen.

#### 3.12.1 Obtención de características del objeto por color

Según (Florencia, 2016) existen varias representaciones o modelos de color. Estos modelos se dividen en dos clases de modelos. Unos son modelos sensoriales y otros se denominan modelos preceptuales.

La idea principal de obtener características por color es sacar datos relevantes en formato *RGB* de objetos, para crear características que puedan ser utilizadas por otros procesos para llegar al resultado. Ver figura 21.

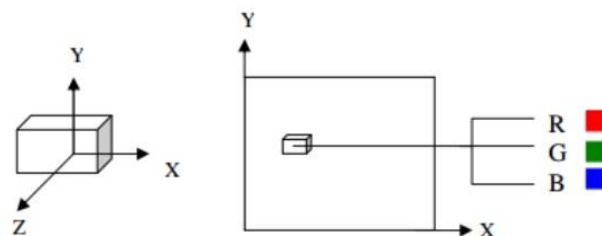


Figura 21: Obtención de características del objeto por color. (Florencia, 2016)



Por la naturaleza de la captura de la imagen, se trabaja en formato *RGB*, por lo cual se trabaja con un formato sensorial en lugar de un formato de percepción como el *HSI*.

Además se pueden aplicar las mismas técnicas que se utilizan en imágenes monocromáticas. Para esto se toma cada componente (*R,G,B*) como una imagen monocromática y se aplica algún operador o método a cada una independientemente.

Después de combinan todas (considerando normalmente el máximo o el promedio).

### **3.12.2 Modelo sensorial**

Menciona (Flores, 2016) el *modelo RGB* es el modelo básico que utiliza los componentes primarios rojo, verde y azul, normalizadas. En este modelo se basan las cámaras y receptores de televisión. Sin embargo se tienen problemas al aplicarlo al procesamiento de imágenes (ecualización).

### **3.12.3 Modelo perceptual**

Según (Martinez, 2016) el *modelo HSI (Hue, Saturation, Intensity)* se puede ver como una transformación del espacio *RGB* al espacio perceptual. En principio los modelos perceptuales deben ser mejores ya que nosotros detectamos los cambios en estos componentes. Sin embargo, es compleja la implementación de la detección de orillas en croma por no ser lineal.

Otra alternativa es definir técnicas especiales para detección de orillas en imágenes a color. El procesamiento de imágenes a color es relativamente reciente, por los altos requerimientos de memoria y cómputo necesarios.

La extracción de información de las imágenes '*Information from imagery*' a través del procesamiento digital constituye hoy en día un inmenso campo de estudio e investigación en diversas disciplinas con múltiples aplicaciones. En este sentido, fotogrametristas, matemáticos, físicos, informáticos y demás, continúan investigando en cuestiones que van desde la aplicación de simples filtros lineales hasta la automatización del reconocimiento semántico de objetos. No obstante, ha sido la disciplina de la Visión Computacional donde se han conseguido los mayores logros, de hecho la detección automática de características sobre imágenes tiene aquí una dilatada tradición y cuenta multitud de métodos para tal

propósito. Desafortunadamente, a pesar de la gran proliferación de métodos, no existe un ‘método universal’ para la detección automática de características, sino que serán los requerimientos del propio problema los que nos obligarán a desarrollar y personalizar nuestro método.

Una imagen contiene una gran cantidad de datos la mayoría de los cuales proporciona muy poca información para interpretar la escena. Un sistema que incorpore visión artificial debe, en un primer paso, extraer de la forma más eficaz y robusta posible determinadas características que nos proporcionen la máxima información posible. Estas características deben cumplir, entre otras, las siguientes condiciones:

- *Su extracción a partir de la imagen no debe suponer un coste excesivo al sistema en el cual está integrado.* El tiempo total de extracción debe ser lo más pequeño posible.
- *Su localización debe ser muy precisa.* El error cometido en la estimación de las características también debe ser lo más pequeño posible.
- *Deben ser robustas y estables.* Deberían permanecer a lo largo de una secuencia.
- *Contendrán la máxima información posible de la escena,* es decir, debemos ser capaces de extraer información de tipo geométrico a partir de ellas.

La extracción de características más importantes en el análisis de imagen son: los puntos, las líneas y los círculos como geometrías básicas en la extracción de características.

#### **3.12.4 Extracción de puntos**

Según (Martinez, 2016) desde el punto de vista computacional se han propuesto dos enfoques para la detección de este tipo de características geométricas:

- Métodos que obtienen los puntos como intersección de aristas o como cambio de pendiente sustancial entre dos aristas y por tanto vienen precedidos de una extracción de bordes.
- Métodos que trabajan directamente sobre imágenes de gris, es decir, y no requieren una extracción previa de aristas.

### 3.12.5 Puntos característicos a partir de aristas: puntos de fuga.

De acuerdo con (González, 2016) los puntos de fuga constituyen el soporte estructural y geométrico de una imagen en perspectiva u oblicua. En este sentido representan puntos de interés que vendrán determinados por la intersección de aristas o líneas de fuga. A continuación se comentan brevemente los dos métodos más sencillos para su cálculo:

*Método de la intersección de rectas.* Se trata del método más sencillo de todos, ya que su cálculo computacional se limita a determinar la intersección de dos rectas perspectivas.

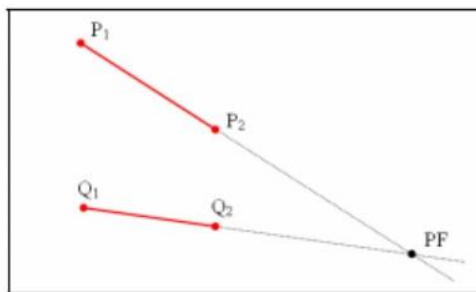


Figura 22: Intersección de rectas. (González, 2016)

A partir de la figura 22, en el que una recta queda identificada geoméricamente mediante su distancia al origen y el ángulo que dicha recta forma con los ejes cartesianos se obtiene la expresión de la misma y sus versiones lineales:

$$\begin{array}{l} \text{Ecuación punto pendiente} \\ y = ax + b \end{array}$$

$$\begin{array}{l} \text{Ecuación normal} \\ Ax + By + 1 = 0 \end{array}$$

Con valores  $(a, l, b)$  para los coeficientes de la  $X$ , de la  $Y$  y del término independiente, respectivamente ó con valores  $(A, B, l)$  para los mismos términos.

La resolución matemática del problema pasa por el planteamiento y resolución del siguiente sistema de ecuaciones:

$$[-a \ 1] \cdot \begin{bmatrix} x \\ y \end{bmatrix} = b \quad \text{ó} \quad [A \ B] \cdot \begin{bmatrix} x \\ y \end{bmatrix} = [-1] \quad (9)$$

Siendo  $x$ , y las coordenadas del punto de fuga a calcular.

*Método de la minimización del área del triángulo.* El método consiste en el cálculo y minimización de la superficie de los triángulos formados por cada uno de los segmentos extraídos y por el punto de fuga, como vértice opuesto de cada triángulo y común a todos ellos.

El área de cada triángulo ( $S$ ) se calcula a través del determinante formado por las coordenadas de los dos puntos extremos de la línea de fuga ( $P1$ ,  $P2$ ) y las coordenadas del punto de fuga ( $PF$ ). Ver Figura 23.

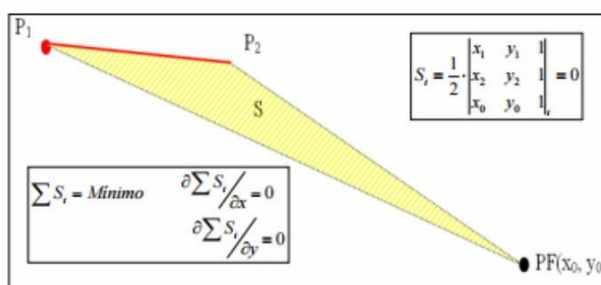


Figura 23: Método de la minimización del área del triángulo. (González, 2016)

En general, la eficiencia de estos métodos dependen directamente de la calidad del método empleado para la obtención de aristas: si este último no localiza correctamente los puntos de arista difícilmente podremos detectar puntos de fuga de forma exacta. Además se añade un tiempo extra de procesamiento previo de las aristas.

### 3.12.6 Puntos característicos a partir de la propia imagen: puntos esquina.

Menciona (González, 2016) existen otros métodos que obtienen los puntos de máxima curvatura de una imagen utilizando directamente los valores de la imagen, sin realizar el paso previo de obtención de aristas. Estos métodos definen una medida que suelen denominar de ‘esquinidad’ la cual se calcula para todos los puntos de la imagen. Cuando esta medida supera un cierto umbral se considera que el punto es una esquina. La mayoría de estos métodos utilizan operadores diferenciales.

### 3.13 Extracción de líneas

De acuerdo con (Gomez & Guerrero, 2016) los métodos de gradiente y laplaciano vistos hasta ahora no proporcionan por sí solos una solución de calidad para la extracción de líneas, de ahí que tengamos que recurrir a estrategias más complejas. A continuación, se describen tres metodologías para la extracción automática de líneas:

- Canny + Burns
- RANSAC + MMCC
- Transformada de Hough

*Canny & Burns.* Extracción de líneas siguiendo un proceso multifase jerárquico que se basa en la extracción de bordes mediante el algoritmo de Canny y en la segmentación posterior de dichos bordes mediante el algoritmo de Burns.

Detección de bordes: Filtro de Canny. El detector de bordes de Canny (Canny, 1986) resulta el más idóneo para la detección de bordes en imágenes donde existe la presencia de geometrías regulares, ya que mantiene tres criterios vitales para nuestros propósitos:

- Precisión en la localización del contorno, garantizando la mayor proximidad de los bordes detectados a los bordes verdaderos.
- Fiabilidad en la detección de los puntos de contorno, minimizando la probabilidad de detección de falsos bordes causados por el ruido, así como también la pérdida de bordes reales.
- Unicidad en la obtención de un único borde para el contorno, garantizando bordes con una achura máxima de un píxel.

*RANSAC+MMCC.*

*RANSAC (RANdom SAmple Consensus)*, es un estimador robusto desarrollado por (Fischler and Bolles, 1981) que se basa en la aplicación de una técnica de votación resultante de un muestreo aleatorio, con el objetivo de determinar el número de observaciones válidas

'*inliers*' y el número de observaciones erróneas '*outliers*'. En nuestro caso la aplicación de *RANSAC* servirá para determinar posibles errores groseros que de ser incluidos en el proceso mínimo cuadrático depararía gravísimos errores en la extracción de las líneas.

Los pasos a considerar por parte de *RANSAC* serían los siguientes:

1. Selección aleatoria de dos puntos aleatorios para constituir una posible recta candidata.
2. Validación de la recta, en función del número de puntos que más menos una cierta tolerancia tengan una variación mínima de su distancia ortogonal a la recta candidata.
3. Repetición del paso 1 y 2 un determinado número de veces.
4. El máximo resultante del proceso de votación después de un número aleatorio de combinaciones se corresponderá con la recta candidata y por consiguiente los votos favorables '*inliers*' del máximo contendrán los puntos favorables a constituir una recta. Los segmentos que superen la tolerancia en los tres grupos generados serán considerados como segmentos erróneos '*outliers*' y serán eliminados.

### **3.13.1 Transformada de Hough para líneas.**

De acuerdo con (Gomez & Guerrero, 2016) la transformada de Hough está diseñada especialmente para encontrar líneas. Definimos una línea como una colección de puntos de borde que son adyacentes y que tienen la misma dirección. La *transformada de Hough* es un algoritmo que tomará una colección de puntos de borde, encontrados mediante un detector de bordes y buscará todas las líneas sobre las cuales estos puntos de borde se encuentran.

La idea básica es convertir los puntos de bordes al espacio de parámetros.

La estrategia para la extracción de líneas mediante Hough es la siguiente:

- Detección de los píxeles de borde de las rectas mediante un filtro de bordes.
- Establecimiento de un espacio de parámetros de dimensiones el espacio de búsqueda y una cuantización suficientemente precisa.

- Se barre la imagen de manera que cada pixel de borde da lugar a una recta: las celdas por las que "pasa" esta recta reciben un "voto".
- En teoría todos los píxeles que pertenecen a una misma recta (en la representación espacial) son rectas (en el dominio de parámetros) que se cortan en una misma celda (en la representación de los parámetros): la recta resultará ser la celda más votada.

### 3.14 Extracción de círculos

Según (González, 2016) *transformada de Hough para círculos*. Al igual que se explotaba la dualidad punto-línea en la transformada de Hough para líneas se podrá hacer algo similar en el caso de los círculos.

Espacio de parámetros. Si el radio es conocido, el dominio de parámetros de cada círculo es bidimensional: coordenadas del centro de cada círculo. En este dominio, cada círculo del espacio se representa con un punto y simétricamente, un punto del dominio espacial se representa en el dominio de parámetros mediante un círculo formado por todos los puntos (dominio de parámetros) que representan a todos los círculos (dominio espacial) que pueden pasar por el punto (dominio espacial).

Menciona (García P. , 2012) después de procesar adecuadamente la imagen se obtienen las características más significativas que permita diferenciar los objetos a clasificar. Si las imágenes fueran adquiridas mediante un escáner, la inclinación podría ser muy pequeña, incluso tanto que podría despreciarse. Sin embargo, cuando las imágenes son adquiridas con un móvil o una cámara fotográfica, la inclinación entra a ser un factor importante. Por esta razón es necesario encontrar características que sean invariantes a la rotación y a la escala, es decir, que al rotar o escalar una imagen, el valor numérico de las características sea similar.

Para lograr un cierto grado de independencia de las características a la rotación y a la escala se pueden tomar dos enfoques: un primer enfoque es transformar la imagen, esto significa rotarla y escalarla para hacerla lo más similar posible a un patrón estándar, después de esto se pueden extraer características numéricas de la imagen. Evidentemente en cualquiera de los dos casos, las características buscadas deben ser lo más similares posibles para objetos de la misma clase y lo más diferentes posibles para objetos de distintas clases. Un segundo

enfoque consiste en trabajar con la imagen tal cual es tomada por la cámara, y utilizar características en las que se obtengan valores numéricos similares para una misma imagen rotada y escalada, es decir que sean invariantes a la rotación y a la escala.

### **3.15 Etapa 6: Reconocimiento**

Según (Rodríguez & Sossa, 2012) el *reconocimiento* es el procedimiento que consiste en decir si una instancia de un objeto se encuentra presente en una imagen.

*1.- Análisis:* Es el procedimiento que asigna una determinada etiqueta a un objeto a partir de la información proporcionada por sus descriptores.

*2.- Interpretación de la imagen:* Es la que determina el significado de un conjunto de objetos previamente reconocidos.

Hay diversos métodos para llevar a cabo dicha etapa, una de ellas es el clasificador “K vecinos más próximos”.

#### **3.15.1 Clasificador “K Vecinos más Próximos”**

Según (García E. , 2008) la clasificación por los “K” vecinos más próximos (*K Nearest Neighbor* o *K-NN*) es ampliamente utilizada en el reconocimiento de formas. Dado un vector a clasificar (rasgos característicos del objeto a clasificar) y un conjunto de vectores prototipo asignados a las diversas clases existentes (base de conocimiento). La regla consiste en calcular la distancia del primero a cada uno de los segundos, seleccionar los “K” vecinos más próximos y decidir por la clase más votada entre los mismos.

Sea  $x$  un vector de dimensión “ $n$ ” a clasificar, sea  $M$  una base de datos de referencia construida de  $N$  vectores de dimensión “ $n$ ” y se conoce la clase  $C_i$  a la cual pertenecen los vectores de la base de referencia  $M$ . Como se muestra en la figura 24 el clasificador  $K$  vecinos más próximos está basado en la estimación local de la densidad de probabilidad de la muestra  $x$  a partir de los “ $K$  próximos vecinos” de la base de referencia.



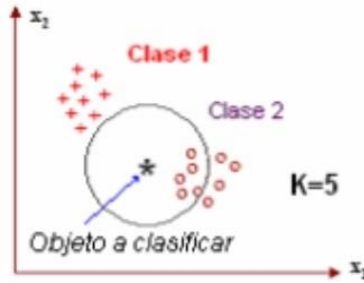


Figura 24: Clasificador KNN. (Garcia E. , 2008)

Sea  $p(x/C)$  la densidad de probabilidad. A partir de esta estimación, la regla de BAYES nos permite expresarlo en términos de la probabilidad a posteriori que la muestra  $x$  pertenezca a la clase  $C_i$ , tal que:

$$p_r(C_i | \vec{x}) = \frac{p(\vec{x}|C_i) * p_r(C_i)}{p(\vec{x})} = \frac{p_r(\vec{x}|C_i) * p_r(C_i)}{\sum_{k=1}^c p_r(\vec{x}|C_k) * p_r(C_k)} \quad (10)$$

Dónde:  $P_r(C_i)$  = probabilidad de aparición de la clase  $C_i$ ,  $P_r(\vec{x})$  = probabilidad de que la muestra  $\vec{x}$  pertenezca a la clase  $C_i$ . y  $P_r(C_i / \vec{x})$  = densidad de probabilidad condicional de la muestra  $\vec{x}$  conociendo la clase  $C_i$ .

## 4 Capítulo 4. Técnicas de tratamiento de imágenes

### 4.1 Sift

De acuerdo a (Juarez, 2011) el objetivo principal del algoritmo *SIFT* es la extracción de unas características que describan de forma correcta los objetos que aparecen en las distintas imágenes que se tengan recopiladas. De esta forma se podrán observar las similitudes existentes entre las imágenes almacenadas en la base de datos y la imagen introducida al programa para su reconocimiento. El coste de la extracción de dichos descriptores debe ser el mínimo posible, y esto se consigue mediante un filtrado en cascada, en los que los algoritmos más pesados sólo se realizan en los lugares que han pasado los filtrados anteriores.

En 1999, David Lowe propuso una técnica de descripción de objetos invariante a escala y rotaciones, el *Scale-Invariant Feature Transform (SIFT)*. Hasta ese momento todos los detectores de objetos creados consideraban un único nivel de detalle (escala) y no tenían en

cuenta que los objetos de interés pueden tener diferentes tamaños o estar a diferentes distancias de la cámara. La técnica se dividía en 2 partes diferenciadas:

1) *El detector SIFT* que utilizaba la combinación de varias técnicas para detectar objetos:

- *La representación espacio-escala de una imagen*: Esta técnica consiste en crear una familia de imágenes suavizadas a diferentes niveles de detalle (definido por un parámetro de escala  $t$ ). El detector *SIFT* creaba el conjunto de imágenes en un rango entre  $t$  y  $2t$  llamado octava.

- *Pirámides gaussianas*: Formadas por la familia de imágenes suavizadas donde la base contiene la más alta resolución. En el caso del detector *SIFT*, a partir de la escala  $2t$  se divide a la mitad de resolución y se usa esta resolución para la siguiente octava.

- *Filtro Log*: Teniendo en cuenta que el espacio escala representaría la familia de imágenes y la pirámide gaussiana es el núcleo de suavizado de la imagen que ha de ser Gaussiana, suponemos que los cruces por cero de estas ecuaciones representarían los bordes del objeto, y los extremos permiten detectar manchas. En el caso del detector *SIFT* sustraen dos escalas consecutivas. Se compara cada píxel con todos sus vecinos y sólo se seleccionan los extremos de espacio-escala que sean mayores o menores que sus vecinos.

2) *Descriptor SIFT* basado en Histogramas de orientación del gradiente: Este descriptor es invariante a rotación ya que se realiza una normalización en rotación que permite comparar puntos en varias orientaciones, donde la orientación de un punto es la orientación dominante del gradiente en su vecindad.

El *gradiente* es un vector bidimensional cuyos componentes están dados por las primeras derivadas de las direcciones verticales y horizontales. Podemos definir el gradiente para cada punto de la imagen, como el vector que apunta en dirección del incremento máximo posible de intensidad, y la magnitud del gradiente del vector corresponde a la cantidad de cambio de intensidad en esa dirección.

Los pasos realizados para obtener el descriptor *SIFT* son:

- Calcular la magnitud y la orientación del gradiente en la vecindad del punto utilizando la primera imagen suavizada.
- Una vez tenemos la orientación de todos sus vecinos, creamos un histograma, donde su máximo determinará la orientación del punto. De aquí podremos obtener los ejes locales para cada punto de interés.
- En el caso de existir en el histograma picos superiores al 80% del máximo se genera un nuevo punto para cada pico con su orientación correspondiente.
- Se segmenta la vecindad en regiones de 4x4 píxeles.
- Se genera un histograma de orientación de gradiente para cada región usando una ponderación *Gaussiana* de ancho 4 píxeles.
- Hay que tener en cuenta que un pequeño desplazamiento espacial provoca que la contribución de un píxel pase de un segmento a otro y cambie la descripción. En el caso de una pequeña rotación la contribución podría pasar de una orientación a otra.
- Para evitar este problema un píxel contribuye a todos sus vecinos, multiplicando la contribución por un peso  $1-d$ , donde  $d$  es la distancia al centro del segmento.
- Por último se obtiene un histograma tridimensional de  $4 \times 4 \times 8 = 128$  casillas considerando 8 direcciones principales y formando un vector con el valor de todas las casillas. Este vector se genera para cada punto de interés

El descriptor *SIFT* creado por David Lowe es parcialmente robusto a cambios de iluminación, puntos de vista, se calcula rápido y es muy distintivo; pero también tiene inconvenientes ya que no es robusto a deformaciones rígidas o elásticas de objetos. Esto es debido a que a la hora de describir se basa en la creación del vector descriptor a partir de las normales de los puntos de interés del objeto y eso provocaría que pequeños cambios de curvatura varíen radicalmente el descriptor.

Menciona (Guitart, 2009) en 2004 apareció una modificación a *SIFT* propuesta por Ke y Sukthankar cuya idea era obtener un descriptor que sea tan distintivo y robusto como *SIFT*, pero con un vector de menos componentes. Para reducir la dimensionalidad usaron la técnica de *Análisis de Componentes Principales (PCA)*. El cálculo de la orientación de los puntos de interés se realiza igual que en *SIFT*, pero se trabaja con ventanas de  $39 \times 39$ , girando la ventana según la orientación dominante y formando un vector concatenando los mapas de gradientes horizontales y verticales.

En este caso se obtiene un vector de  $2 \times 39 \times 39 = 3042$  elementos, pero no toda la información es significativa. Para reducir el número de casillas se recopilan muchos ejemplos de puntos de interés y utilizando sus vectores calculamos la matriz de covarianza, obteniendo así el promedio. Después se diagonaliza la matriz y se organiza por eigenvalores decrecientes, donde los primeros serán los más altos.

A partir de aquí, para cada nuevo punto de interés se obtiene el vector inicial y se convierte al sistema de ejes de los eigenvectores. Finalmente el descriptor estará constituido por las 20 primeras componentes, los 20 ejes más distintivos.

De esta manera se consiguió crear una versión simplificada del descriptor *SIFT*, pero aun así continuaba presentando carencias respecto a deformaciones rígidas o elásticas de objetos.

#### *Detección de extremos en la escala-espacio*

Según (Juarez, 2011) es necesario detectar una serie de puntos clave mediante una serie de algoritmos en cascada. Estos algoritmos, irán filtrando los puntos de la imagen que más posibilidades tengan de llegar a ser un *keypoint*, para realizar posteriormente en torno a ellos un examen más exhaustivo.

En un primer momento, se deben localizar los lugares y escalas que pueden repetirse en distintas vistas del objeto. La localización de estos lugares puede hacerse mediante una búsqueda de características estables en todas las escalas que se den en la imagen, utilizando una función continua conocida como “Escala-espacio”, desarrollada en 1983 por Witkin y descrita en su trabajo de investigación en 1986.

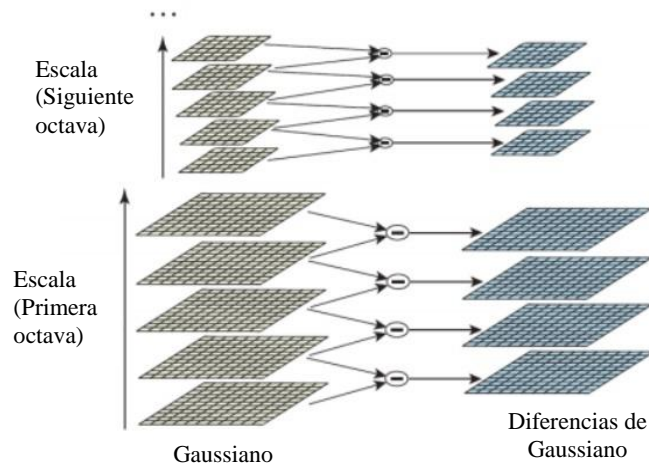
Koenderink en 1984 y Linderberg en 1994 demostraron que bajo algunas circunstancias, la única posibilidad de utilizar una función del tipo “Escala-Espacio” es mediante la función Gaussiana. Por lo tanto, la función que define la “Escala-Espacio” de una imagen sería del tipo  $L(x, y, \sigma)$ .

Esta función se produce tras la convolución de la variable de escala Gaussiana  $G(x, y, \sigma)$  y una imagen de entrada  $I(x, y)$  :

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) ; \quad (11)$$

Donde  $*$  es la operación de convolución en  $x$  e  $y$ .

Una buena forma de construir la *Diferencia de Gaussianas* es convolucionar la imagen inicial con Gaussianas para producir distintas imágenes separadas por un factor constante de escala y espacio,  $k$ ; como se muestra en la figura 25:



**Figura 25: Formación de la Diferencia de Gaussianas (DOG). (Juárez, 2011)**

Posteriormente se dividirá cada octava del “Escala-Espacio” en un número entero de intervalos,  $s$ , tal que  $k = 2^{1/s}$ . Se deberán producir  $s+3$  imágenes borrosas para cada octava. Las imágenes de escalas adyacentes serán restadas para producir la Diferencia de Gaussianas necesaria.

Una vez que la octava completa ha sido procesada se volverá a muestrear la imagen Gaussiana que tiene el doble del valor inicial de  $\sigma$  que tomen el segundo píxel de cada

columna y fila. La precisión de las muestras relativas de  $\sigma$  no es diferente de las del comienzo de la octava previa, mientras que la cantidad de cálculos computacionales se reducen bastante de esta forma.

## 4.2 Surf

De acuerdo con (Patiño, 2012) el algoritmo *SURF* (*Speeded Up Robust Features*), fue empleado por Herbert Bay en el año 2006, se basa en SIFT pero a diferencia de este, es más robusto puesto que utiliza wavelets y determinantes Hessianas, además *SURF* está libre de patentes y existen numerosas implementaciones mucho más eficientes que *SIFT*.

Hace réplicas de la imagen para así buscar los puntos que estén en todas las réplicas asegurando la invariancia de escala. Las réplicas pueden ser piramidales o del mismo tamaño a la original.

En las piramidales la principal idea es tener una imagen más pequeña que la anterior, para ello se realiza un filtro de la imagen y se submuestra ya que si se realiza solo el submuestreo se pierde la información de la imagen como se muestra en la figura 26:

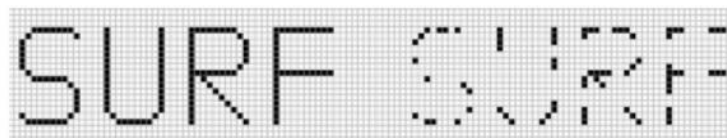


Figura 26: Algoritmo SURF. (Patiño, 2012)

*SURF* ofrece la detección de objetos con menor coste computacional, es más rápido y eficaz que *SIFT* en lo que se refiere a realizar cálculos para la generación de puntos claves, todo esto gracias a la reducción de la dimensionalidad y dificultad en el cálculo de vectores.

Menciona (Aracil, 2012) *SURF* es otro de los algoritmos más utilizado para la extracción de puntos de interés en el reconocimiento de imágenes. La extracción de los puntos la realiza detectando en primer lugar los posibles puntos de interés y su localización dentro de la imagen.

Es mucho más rápido que el método *SIFT*, ya que los *keypoints* contienen muchos menos descriptores debido a que la mayor cantidad de los descriptores son 0.

Este descriptor se puede considerar una mejora debido a que las modificaciones que supondría en el código no serían excesivas, ya que el descriptor *SURF* utiliza la gran mayoría de las funciones que utiliza el descriptor *SIFT*.

En la figura 27 se describe el procedimiento del algoritmo *SURF* para detectar puntos de interés (*keypoints*), asignación de la orientación y por último obtención del descriptor *SURF*.

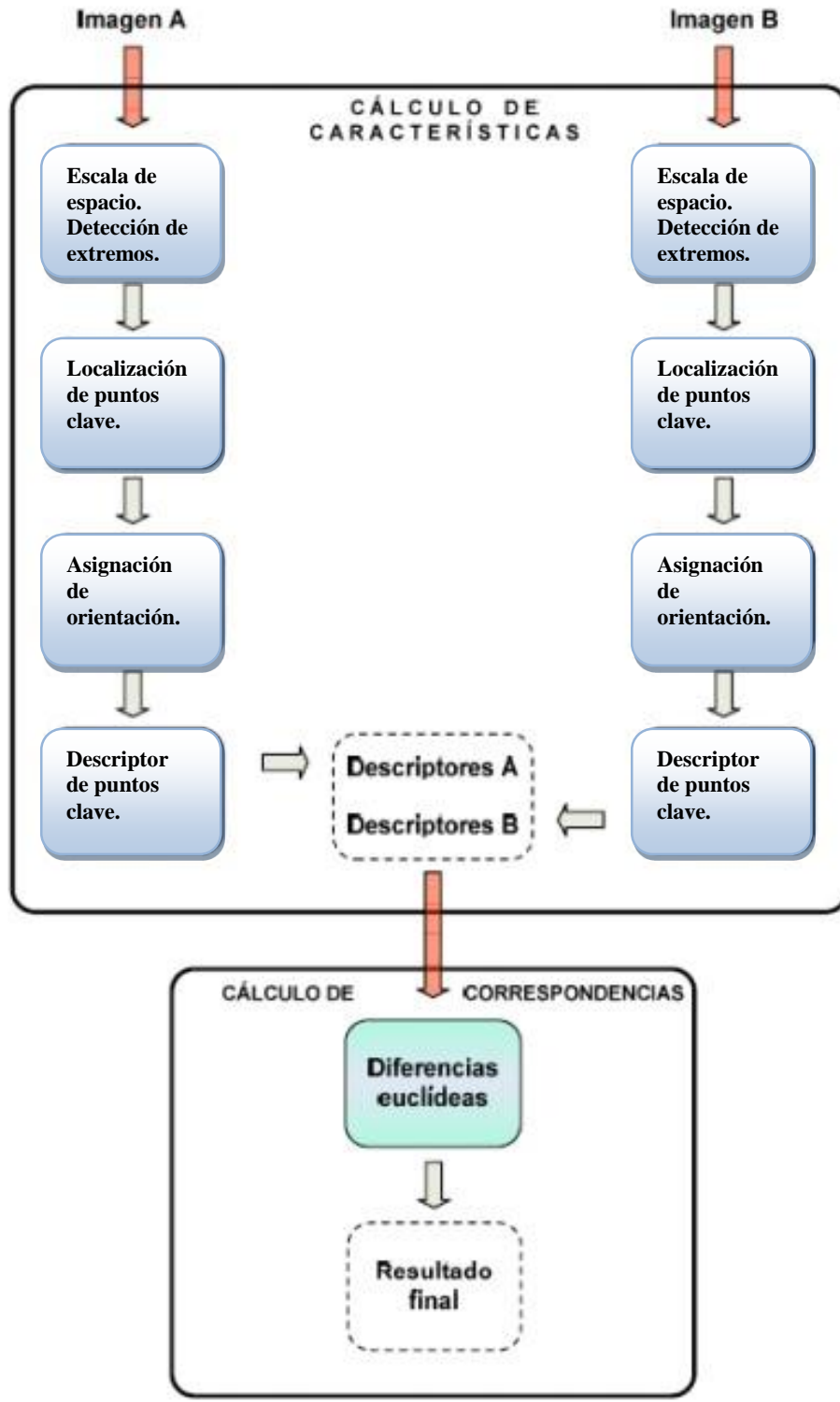


Figura 27: Esquema del proceso de clasificación SURF. (Aracil, 2012)



### *Detección de puntos de interés.*

Menciona (Benito, 2016) la primera de las etapas del descriptor *SURF* es idéntica a la del descriptor *SIFT* en cuanto a la detección de puntos de interés se refiere.

El descriptor *SURF* hace uso de la matriz Hessiana, más concretamente, del valor del determinante de la matriz, para la localización y la escala de los puntos. El motivo para la utilización de la matriz Hessiana es respaldado por su rendimiento en cuanto a la velocidad de cálculo y a la precisión.

Lo realmente novedoso del detector incluido en el descriptor *SURF* respecto de otros detectores es que no utiliza diferentes medidas para el cálculo de la posición y la escala de los puntos de interés individualmente, sino que utiliza el valor del determinante de la matriz Hessiana en ambos casos. Por lo tanto dado un punto  $p = (x, y)$  de la imagen, la matriz Hessiana  $H(p, \sigma)$  del punto  $p$  perteneciente a la escala  $\sigma$  se define como:

$$H(p, \sigma) = \begin{bmatrix} L_{xx}(p, \sigma) & L_{xy}(p, \sigma) \\ L_{xy}(p, \sigma) & L_{yy}(p, \sigma) \end{bmatrix} \quad (12)$$

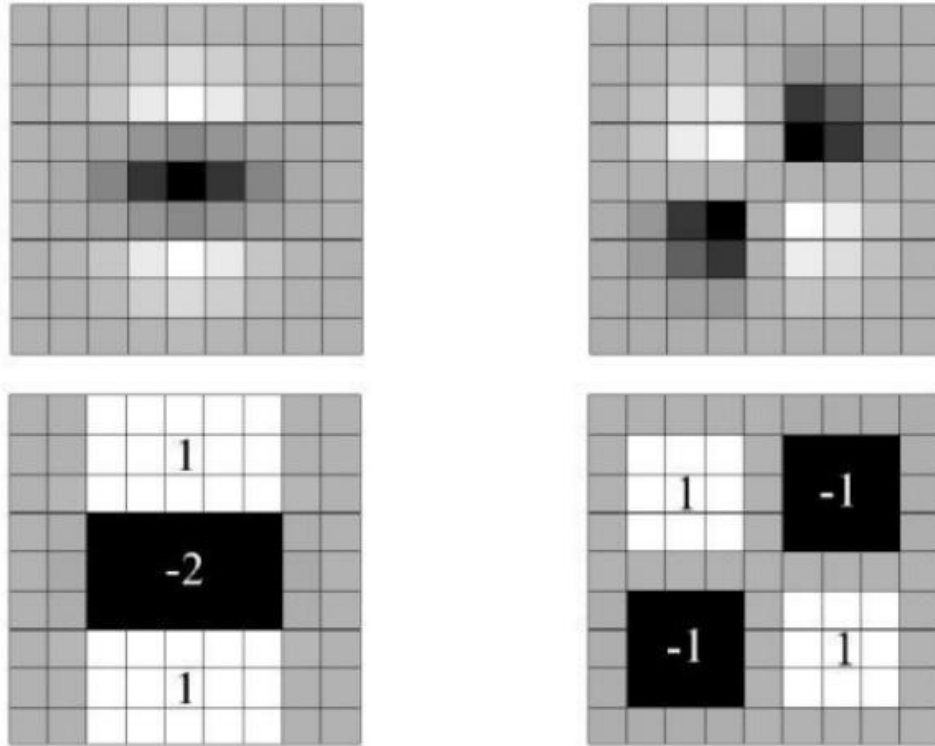
Donde  $L_{xx}(x, \sigma)$  Es la convolución de segundo orden de la Gaussiana.

Las aproximaciones de las derivadas parciales se denotan como  $D_{xx}$ ,  $D_{xy}$  y  $D_{yy}$ , y el determinante se calcula de la siguiente manera:

$$\det(H_{approx.}) = D_{xx}D_{yy} - (0.9D_{xy})^2 \quad (13)$$

Donde el valor de 0,9 está relacionado con la aproximación del filtro Gaussiano.

En la figura 28 se puede observar la representación de la derivada parcial de segundo orden de un filtro gaussiano discretizado y la aproximación de la Derivada implementada en el caso del descriptor *SURF*.



**Figura 28:** Se puede observar la representación de la derivada parcial de segundo orden de un filtro gaussiano discretizado y la aproximación de la derivada implementada en el caso del descriptor SURF. (Aracil, 2012)

La imagen de salida obtenida tras la convolución de la imagen original con un filtro de dimensiones  $9 \times 9$ , que corresponde a la derivada parcial de segundo orden de una gaussiana con  $\sigma = 1;2$ , es considerada como la escala inicial o también como la máxima resolución espacial ( $s = 1;2$ , correspondiente a una gaussiana con  $\sigma = 1;2$ ). Las capas sucesivas se obtienen mediante la aplicación gradual de filtros de mayores dimensiones, evitando así los efectos de *aliasing* en la imagen. El espacio escala para el descriptor SURF, al igual que en el caso del descriptor SIFT, está dividido en octavas. Sin embargo, en el descriptor SURF, las octavas están compuestas por un número fijo de imágenes como resultado de la convolución de la misma imagen original con una serie de filtros cada vez más grande. El incremento o paso de los filtros dentro de una misma octava es el doble respecto del paso de la octava anterior, al mismo tiempo que el primero de los filtros de cada octava es el segundo de la octava predecesora.

Finalmente para calcular la localización de todos los puntos de interés en todas las escalas, se procede mediante la eliminación de los puntos que no cumplan la condición de máximo

en un vecindario de  $3 \times 3 \times 3$ . De esta manera, el máximo determinante de la matriz Hessiana es interpolado en la escala y posición de la imagen.

### 4.3 Orb

Según (Espitia, 2014) este algoritmo de detector/descriptor es invariante a la rotación y resistente al ruido, aplicado a la detección de objetos y seguimientos en dispositivos móviles, construido a partir del detector *FAST* y del descriptor *BRIEF*, se decidió de esta manera porque tiene buen rendimiento a un bajo costo, las características de este algoritmo son que añade la componente de orientación precisa al *FAST*, tiene una computación eficiente del detector de características *BRIEF*.

$$m_{pq} = \sum_{x,y} x^p y^q I(x, y)$$

Figura 29: Algoritmo ORB, Definición del Patrón. (Espitia, 2014)

En la figura 29 se muestra la fórmula matemática del algoritmo ORB y la definición del método secuencial que utiliza. Es una extensión del descriptor *BRIEF*, pero añade dos importantes mejoras, una es aumentar los datos del descriptor con datos de orientación del detector *FAST*, esto hace que el detector sea robusto a la rotación, la segunda mejora es selección de un esquema de pares de puntos, lo cual permite que las comparaciones contribuyan con el descriptor, antes se usaba un muestreo aleatorio, ahora se usa un esquema de muestreo que usa el aprendizaje de la máquina para correlacionar la características de *BRIEF* sobre la rotación invariante.

### 4.4 Freak

De acuerdo con (Peralta, 2013) *FREAK (Fast Retina Keypoint)* es un descriptor que usa las ideas propuestas en *BRISK* y además se inspira en el sistema visual humano, principalmente en la topografía que tienen los conos y los bastones en la retina del ojo. Este descriptor usa un patrón similar a *BRISK*, es un patrón circular basado en los campos receptivos de la retina humana que imita las densidades de los receptores al tener mayor representación en el área central. Esta concentración decrece exponencialmente cuando se aleja del centro, y además

existe un traslape en los campos receptivos. Un filtro Gaussiano con diferentes desviaciones estándar por cada nivel es aplicado para reducir el ruido.

El descriptor se forma usando diferencias entre pares de puntos que comparten el mismo filtro, si la resta es positiva se asigna un valor de 1, en caso de ser igual o menor a cero se deja el valor de cero. Los autores diseñaron el descriptor considerando pares cuyas restas puedan representar mejor al punto, para ello consideran las parejas con varianza más alta. Experimentalmente se encontraron 512 pares que dan la mejor representación al punto, agregar más pares no mejora el desempeño del descriptor. Finalmente, se determina la orientación del punto usando el mismo concepto de *BRISK* al usar pares con distancias largas sobre una configuración simétrica.



*Figura 30: Algoritmo FREAK, Patrón de muestreo (Espitia, 2014)*

En la figura 30 se muestra el patrón basado en *BRISK* y en la retina humana usado en *FREAK*.

#### **4.5 Flann**

Menciona (Morante, 2012) *FLANN* (*fast library for approximate nearest neighbours*). Nos ofrece varias ventajas: Permite el tratamiento de imágenes a color. Esto no afecta al rendimiento del algoritmo pero de cara al operador el resultado es más sencillo de interpretar.

El comparador *FLANN* de *OpenCV* permite el acceso a la distancia euclídea obtenida de la comparación. Esto nos permite trabajar solo con aquellos puntos "de calidad", con distancias pequeñas (lo que indica gran similitud).

Se puede definir el problema de la búsqueda del vecino más cercano de la siguiente manera: dado un conjunto de puntos  $P=p_1+p_2+\dots+p_n$  en un espacio  $X$ , estos puntos deben ser pre-

procesados de tal forma que dado un nuevo punto de búsqueda  $q \in X$ , encontrar el punto en  $P$  que es más cercano a  $q$  puede ser realizado rápidamente.

El problema de la búsqueda del vecino más cercano es uno de mayor importancia en una variedad de aplicaciones tal como reconocimiento de imágenes, compresión de datos, reconocimiento de patrones y clasificación, aprendizaje de máquina, sistemas de recuperación de documentos, estadística y análisis de datos. Como sea, resolver este problema en un espacio multidimensional parece ser una tarea bastante ardua y no existe ese algoritmo que realice la tarea significativamente mejor que los buscadores por fuerza bruta. Esto ha llevado a un interés creciente en esa clase de algoritmos que realicen búsquedas aproximadas del vecino más cercano, el cual ha probado ser una aproximación bastante buena en la mayoría de las aplicaciones prácticas.

*FLANN* es una librería para realizar aproximaciones rápidas de los vecinos más cercanos. *FLANN* está escrito en lenguaje *C++*. *FLANN* puede ser fácilmente usado en la mayoría de los contextos que usen el lenguaje *C*, como *MATLAB* y *Python*.

La precisión de la aproximación es medida en términos de precisión, el cual es definido en términos de porcentaje de los puntos de búsqueda para la cual el vecino más cercano es encontrado. En los experimentos realizados por Marius Muja y David Lowe uno de dos algoritmos obtuvo el mejor rendimiento, dependiendo de los datos y a precisión deseada. Estos algoritmos se usan tanto para árboles  $k$ -ésimos jerárquicos “*k-means tree*” o múltiples árboles  $k$ -ésimos aleatorios “*kd-tree*”.

#### *Algoritmo de kd-tree aleatorio*

Según (Pacheco, 2011) el clásico algoritmo de *kd-tree* es eficiente en bajas dimensiones, pero en altas dimensiones el rendimiento rápidamente decae. Para obtener un impulso sobre búsqueda lineal se hace necesaria una aproximación al vecino cercano. Esto mejora la velocidad de búsqueda al costo de que el algoritmo no estará retornando el vecino cercano exacto. El algoritmo original de *kd-tree* separa los datos a la mitad en cada nivel del árbol en la dimensión en la cual los datos muestran una gran varianza. Por comparación, los árboles

aleatorios son construidos escogiendo la dimensión a partir al azar desde la primera dimensión  $D$  en la que el dato tiene gran varianza.

*El algoritmo de  $k$ -means tree jerárquico.*

Como se muestra en la figura 31 el árbol  $kd$ -tree jerárquico es construido dividiendo los datos en cada nivel en  $K$  distintas regiones usando agrupamiento  $k$ -ésimo y luego aplicando el mismo método recursivamente a los puntos de cada región. Se detiene la recursión cuando el número de puntos en la región es más pequeño que  $K$ .

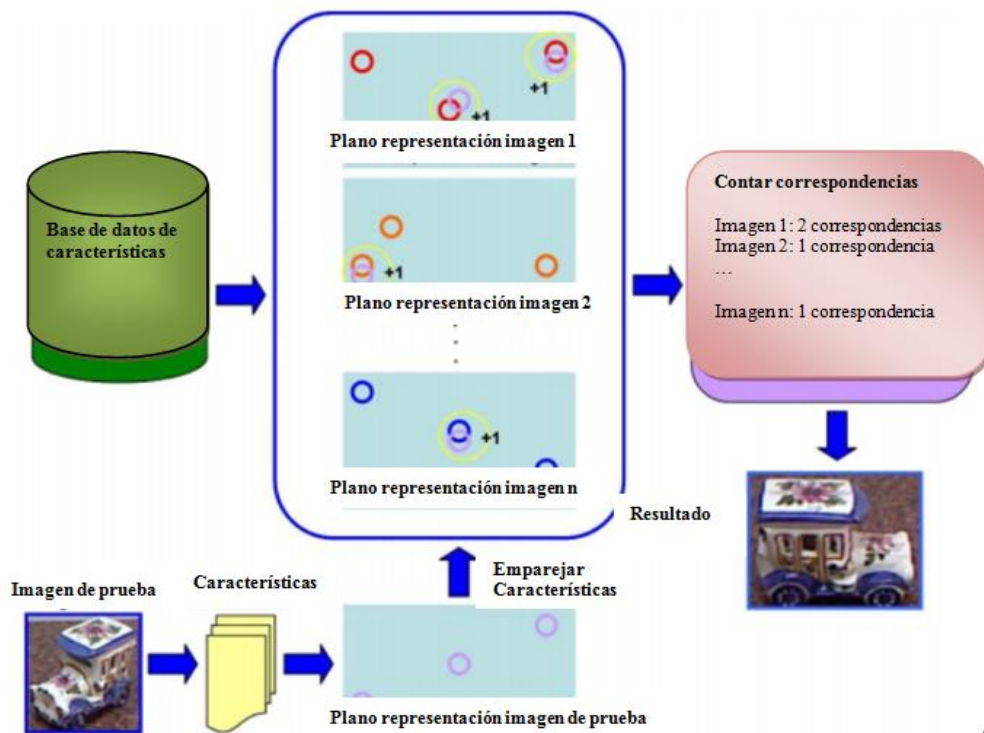


Figura 31: Algoritmo de búsqueda del vecino más cercano. (Bueno, 2012)

## **5 Capítulo 5. Metodología**

### **5.1 Propuesta de solución**

Contextualizando, en caso de que, el cliente no lleve consigo el producto que desea adquirir, el vendedor será el encargado de buscar en la base de datos si tiene en existencia algún producto similar por el cual el cliente pregunto y se lo mostrará, en caso contrario se le notificará que dicho producto no está en venta en la tlapalería.

Por otro lado, cuando se realice la búsqueda de un objeto en específico y existan varios modelos del mismo, el sistema tomará el objeto con más porcentaje de similitud y lo mostrará en la pantalla del usuario.

Por lo anterior, se propone un sistema de visión artificial para reconocer solo objetos existentes dentro de una tlapalería para su venta, con la posibilidad de incorporar nuevos objetos al banco de imágenes. El caso de estudio está enfocado a las tlapalerías en general e insumos que no excedan el tamaño de 37.4 cm de ancho x 30cm de alto.

#### **5.1.1 Ciclo de vida en cascada**

El ciclo de vida utilizado en el desarrollo del sistema de reconocimiento de objetos es el de cascada. Se ha optado por este ciclo ya que es adecuado para los proyectos en los que se dispone de todos los requerimientos al comienzo, para el desarrollo de un producto con funcionalidades conocidas o para proyectos, que aun siendo muy complejos, se entienden perfectamente desde el principio.

Además de su planificación sencilla, provee un producto con un elevado grado de calidad sin necesidad de un personal altamente calificado.

##### **5.1.1.1 Etapa 1 Análisis de requerimientos**

En esta etapa se realizó una investigación previa donde se consultaron diferentes libros acerca del procesamiento digital de imágenes y de *OpenCV* en bibliotecas universitarias y páginas web, así como también artículos y trabajos referentes a estos temas, los cuales fueron necesarios para comprender los conceptos básicos del PDI y de *OpenCV*, para posteriormente generar el algoritmo que permita reconocer un objeto a partir de una imagen.

Una vez comprendido los conceptos básicos del PDI y haber analizado las diferentes técnicas y funciones de *OpenCV* para reconocer objetos, se procede a generar los requerimientos funcionales y no funcionales del sistema de reconocimiento de objetos.

Requerimientos Funcionales:

- El sistema debe reconocer un objeto a partir de una imagen de entrada solo si esta se encuentra en el banco de imágenes.
- El vendedor buscará los objetos que desee para saber si se encuentran en existencia.
- El sistema debe permitir al vendedor buscar y consultar la información sobre los objetos almacenados en el banco de imágenes.
- El sistema debe tener un banco de imágenes pre-procesado para poder realizar una búsqueda de un objeto en específico.
- El sistema debe permitir agregar nuevos objetos al banco de imágenes ya establecido.
- El sistema debe contar con un espacio físico pre-diseñado para la captación del objeto a buscar.

Requerimientos no funcionales:

Interfaces

- Hardware:

*Raspberry pi 2*: Equipo de cómputo.

*Raspberry pi cámara*: Para la captación de los objetos a buscar.

*Teclado*: Para la ejecución de comandos y texto.

*Mouse*: Para los movimientos del cursor.

*Monitor*: Representación de los resultados.



*Banco de imágenes:* Imágenes de los objetos captados por la *raspberry pi* cámara que serán examinados.

1 caja de cartón con dimensiones de 30 cm x 37.4 cm pre-diseñada.

2 Focos de luz cálida de 4 W.

- *Software*

*Librería OpenCV 3.0.0:* Para hacer uso de sus funciones para el reconocimiento de objetos.

*Sistema operativo Raspbian Jessie:* Para que OpenCV 3.0.0 pueda ser instalado eficazmente.

*Lenguaje de programación Python 2.7:* Para el desarrollo del sistema de reconocimiento de objetos.

*Adobe Photoshop CS6:* Para el pre-procesamiento del banco de imágenes así como también de los objetos a buscar.

Según el estándar internacional de Especificación de Requerimientos IEEE830, los documentos de definición y especificación de requerimientos deben contemplar los siguientes aspectos resumidos por (Pleeger, 2002) como se indica a continuación:

**Ambiente físico:** El equipo deberá permanecer en una zona en específica, lejos de la luz natural y con una buena ventilación e iluminación, debido al rígido proceso de captación del objeto a buscar. Como ya se ha mencionado anteriormente el contexto físico es un factor muy importante al momento de tomar una fotografía a determinado objeto.

**Usuarios:** El sistema será operado solo por el vendedor de la tlapalería, siendo capacitado anteriormente para que pueda manejar el sistema sin ningún tipo de problema. En relación al sistema, su estructura no es muy compleja siendo fácil de manejar y entendible para la persona que lo use.

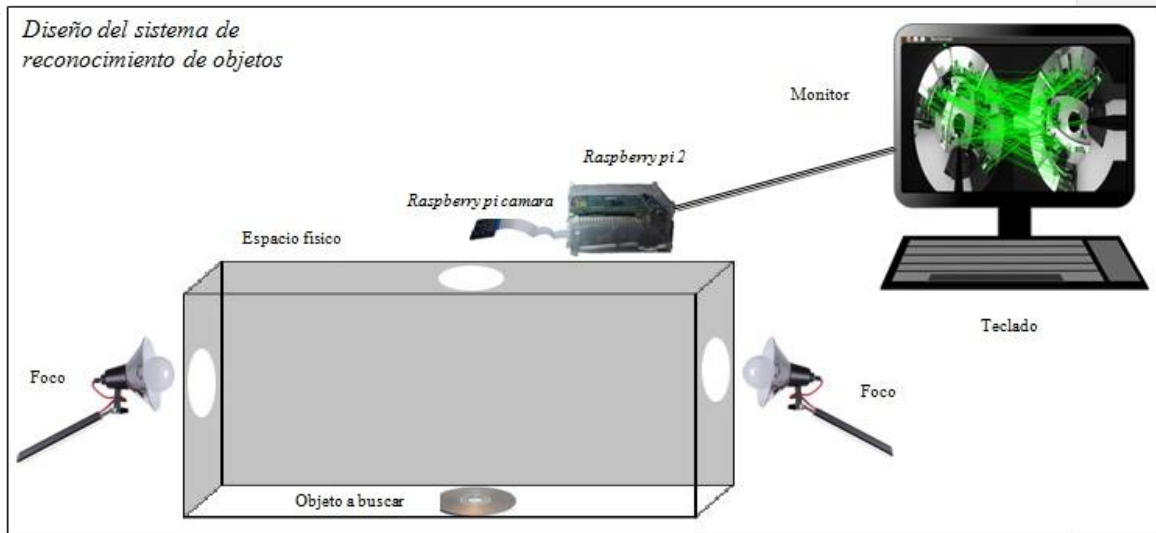
Datos: Debido a que el sistema contará con un banco de imágenes de todos los objetos que estén en venta, estos datos deberán estar almacenados hasta que objetos nuevos lleguen a la tlapalería para ser procesados junto con los actuales. Solo en este caso el banco de imágenes deberá ser actualizado.

Seguridad: Es importante mencionar que el banco de imágenes cuente con uno o más respaldos para que de ese modo se prevean contingencias que puedan surgir en un momento determinado.

Funcionalidad: El sistema reconocerá solo los objetos en existencia de la tlapalería, lo hará cuando el cliente lleve una muestra del producto que desea adquirir y no tenga conocimiento de su nombre en específico, el vendedor hará uso del sistema para buscar dicho objeto, el tiempo de respuesta dependerá del tamaño del banco de imágenes de la tlapalería.

#### **5.1.1.2 Etapa 2 Diseño**

Una vez recabados todos los requisitos para desarrollar el sistema, se procede a realizar el diseño del sistema. En la figura 32 se muestra el prototipo que se ha creado para las pruebas del sistema en cuestión.



**Figura 32: Diseño del sistema de reconocimiento de objetos (Elaboración propia)**

En la figura 33 se muestra el Diagrama de Caso de uso del requerimiento más importante que consiste en obtener la imagen y buscará la identificación del objeto

*Diagrama de casos de uso*

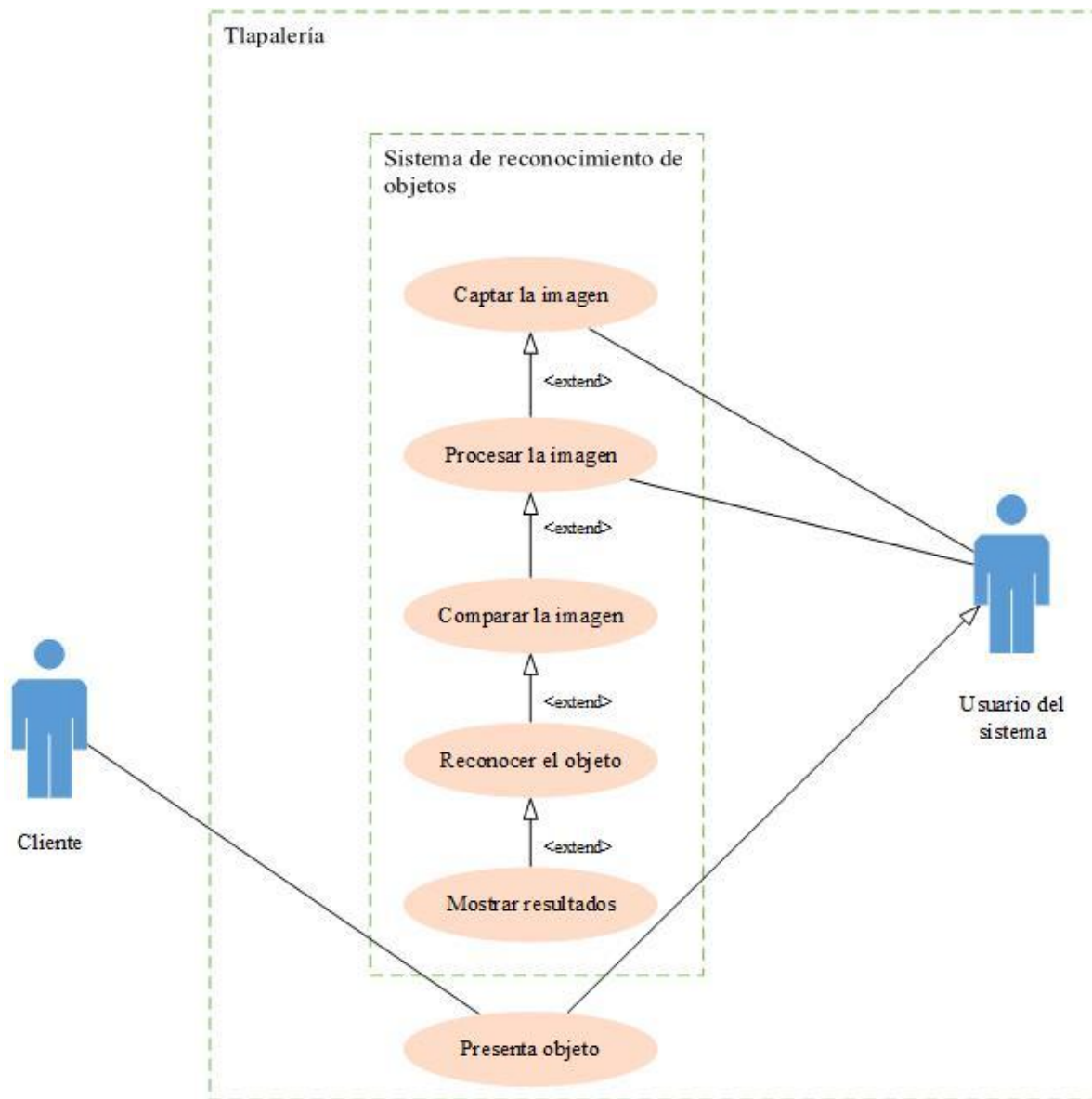


Figura 33: Diagrama de casos de uso (Elaboración propia)

El usuario del sistema es el encargado de tomar la foto del objeto que quiere buscar en el sistema y lo pre-procesa para poder buscarlo y saber sus características con ayuda del sistema, para poder presentarlo físicamente al cliente.

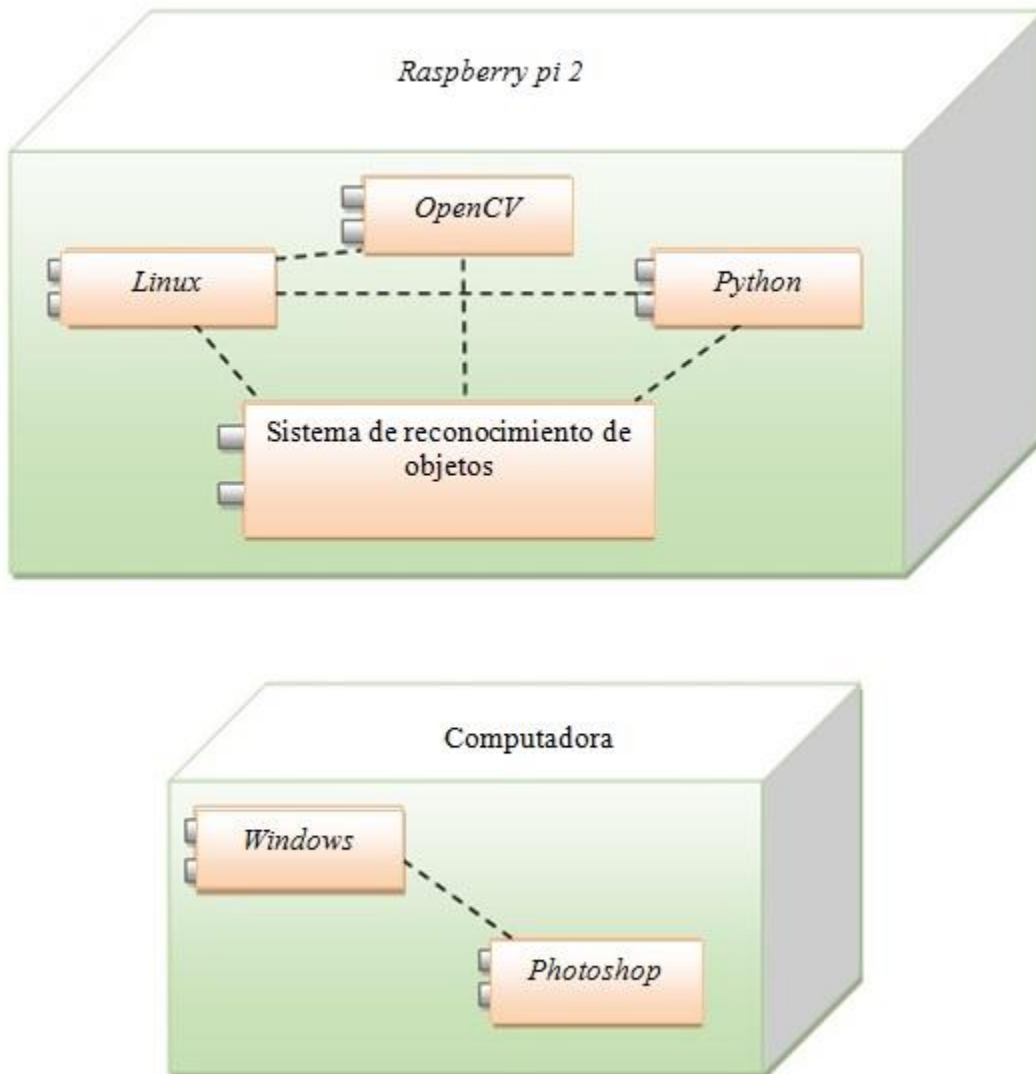
El sistema es el encargado de obtener una imagen del objeto a buscar, lo procesa y compara en el banco de imágenes y si lo reconoce y encuentra, muestra al usuario del sistema las características e imagen de dicho objeto.

El cliente es quien muestra el objeto que está buscando al encargado de la tlapalería.

La tlapalería es el espacio físico donde el sistema y el usuario del sistema son utilizados para la búsqueda de objetos a petición del cliente.

En la figura 34 se presentan el Diagrama de despliegado de los componentes del sistema. Algunos son componentes existentes que solo se instalan mientras que otros se desarrollan para este sistema.

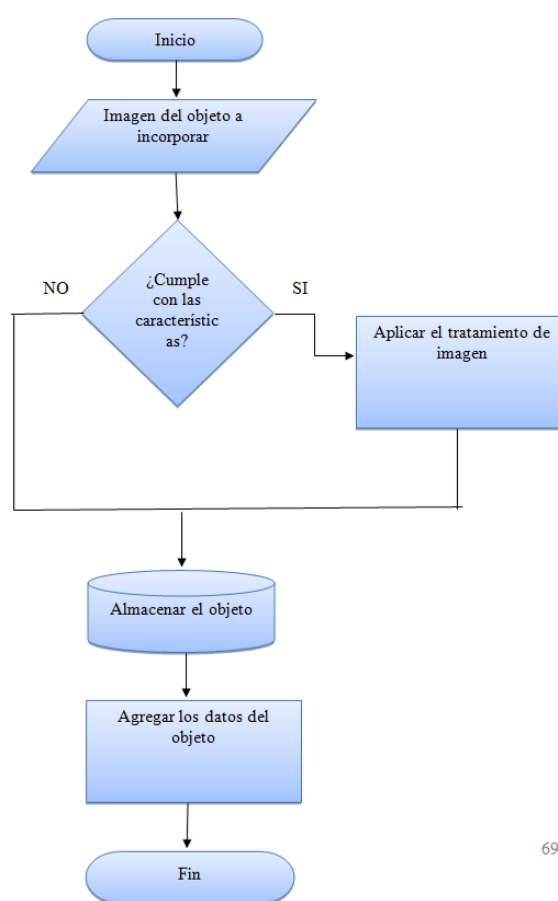
*Diagrama de componentes*



**Figura 34: Diagrama de componentes (Elaboración propia)**

Para incorporar un objeto nuevo se llevará a cabo este proceso. Ver figura 35.

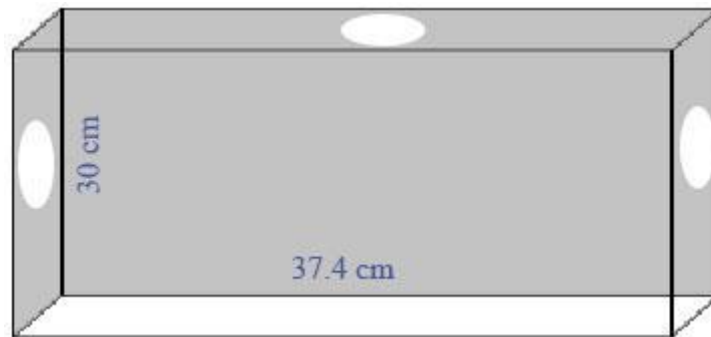
- 1) Obtener la fotografía del objeto con dimensiones de máximo 30 cm x 37.4 cm.
- 2) Aplicar el tratamiento de imagen necesario
- 3) Incorporar al banco de imágenes
- 4) Agregar los datos del objeto.



**Figura 35: Proceso de incorporación de un objeto al banco de imágenes. (Elaboración propia)**

*Para identificar un objeto.*

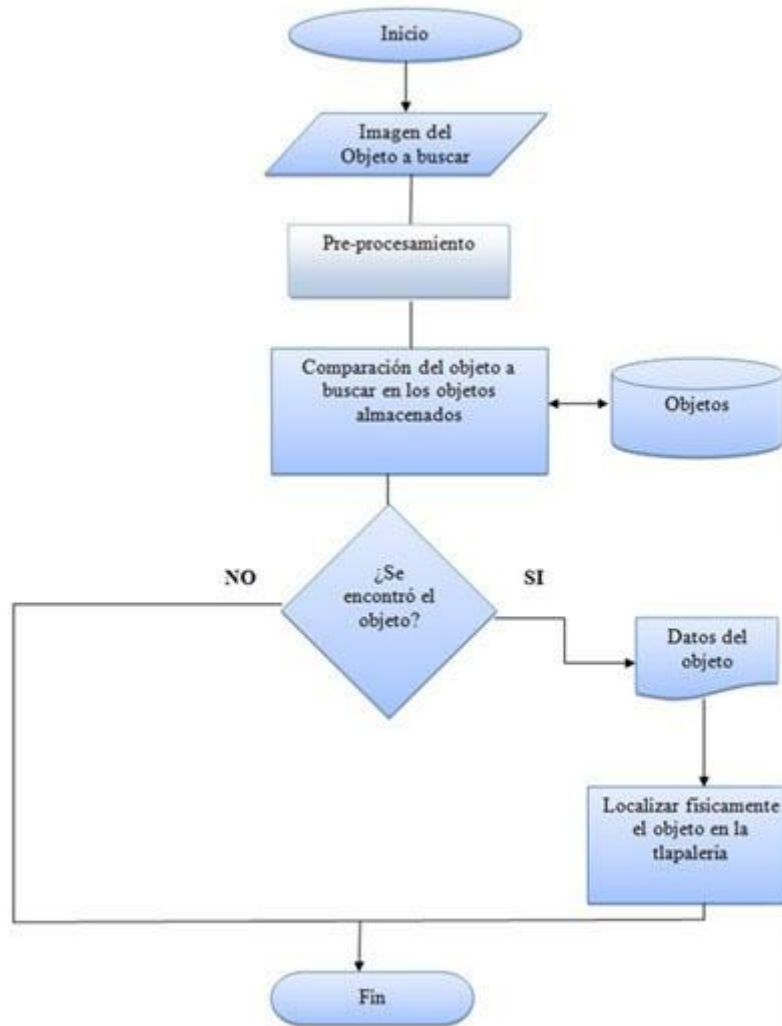
- 1) Obtener la imagen del objeto a buscar
- 2) Pre-procesamiento de la imagen del objeto.
- 3) Iniciar la búsqueda donde se comparará la “imagen a buscar” con las “imágenes del banco de imágenes” ya procesadas.
- 4) Obtener los resultados que consistirán en una o más imágenes con grado de similitud.
- 5) Determinar si se encontró el objeto o no.



*Figura 36: Espacio físico. (Elaboración propia)*

En la figura 36 se muestra el espacio físico y las dimensiones establecidas para que el proceso de captación del objeto a buscar pueda ser llevado a cabo eficazmente.

En la Figura 37 se muestra el proceso para la identificación del objeto. Una vez que el objeto se identifica se busca para ver si existe en la base de datos de objetos.



*Figura 37: Proceso de identificación del objeto. (Elaboración propia)*

### 5.1.1.3 Etapa 3 Implementación

Para tener un mejor entorno de desarrollo el sistema será dividido en dos partes: guardar los descriptores de las imágenes de los objetos de la tlalpería en un archivo externo para luego examinar dichos descriptores para encontrar imágenes, en este caso objetos, similares con la imagen de entrada (objeto a buscar).

Los descriptores le permiten al sistema analizar las características de las imágenes y de ese modo puedan ser procesadas y encontrar posibles similitudes con otros objetos.



El primer proceso que se realizó fue guardar los descriptores de las imágenes en un archivo externo. Esto es para que no se tenga que recrear los descriptores de cada imagen cada vez que se ejecute una búsqueda de imágenes parecidas.

En el sistema se examinará una carpeta con imágenes, dentro de la misma se crearán los descriptores correspondientes para futuras búsquedas.

Para crear los descriptores y guardarlos en un archivo se usaron las funciones *Load an image*, *feature detector*, *detect*, *compute* y *NumPy*.

Se define primeramente la función *create\_descriptors*, en la cual se crearán los descriptores de las imágenes.

Seguido de la función *save\_descriptors* donde se cambia el formato de las imágenes almacenadas a *.npy*, para luego hacer uso de la función *np.save* para guardar las imágenes en ese formato dentro de la misma carpeta que contiene a todas las imágenes.

Una vez terminado el *script*, su función será la de pasar el nombre de la carpeta que contienen todas nuestras imágenes, creando los respectivos descriptores y guardarlos en la misma carpeta. Ver figura 38.



**Figura 38: Generación de descriptores**

Una vez teniendo los descriptores guardados debidamente, el siguiente paso es ejecutar el proceso de búsqueda sobre todos los descriptores y observar si existe alguna imagen parecida a la imagen de entrada (objeto a buscar).

Este es el proceso que se llevó a cabo:

- 1) Cargar una imagen de entrada (objeto a buscar) y la creación de un descriptor para él.
- 2) Examinar la carpeta con los descriptores (objetos).
- 3) Para cada descriptor, se calculan las posibles coincidencias basadas en *FLANN*. (Como ya se mencionó anteriormente, este algoritmo ayuda a buscar posibles similitudes entre los objetos del banco de imágenes, en este caso tomando como mayor porcentaje de parentesco a un objeto sobre otros y a este llamarlo objeto encontrado o con mayor número de similitud con respecto al objeto de entrada).
- 4) De todos los descriptores, la elección de la que tiene el mayor número de coincidencias es el más parecido.
- 5) Si el descriptor posee un 80% de parecido por sobre todos los demás descriptores del banco de imágenes al objeto de entrada se considera como objeto encontrado.

Crear código para la búsqueda del objeto:

Primeramente se crea una matriz con el nombre de los archivos y utilizar la función *cv2.imread* para leer la imagen de entrada (objeto a buscar).

Después se crean los archivos, imágenes y los descriptores globales, también se hará uso del bucle *for* para detectar a los descriptores de las imágenes contenidas en la carpeta.

Se crea el detector *SIFT* mediante la función *sift = cv2.xfeatures2d.SIFT\_create()*

Se crea *FLANN* mediante la función *flann = cv2.FlannBasedMatcher*, estableciendo también los parámetros del índice así como también la búsqueda de los mismos.

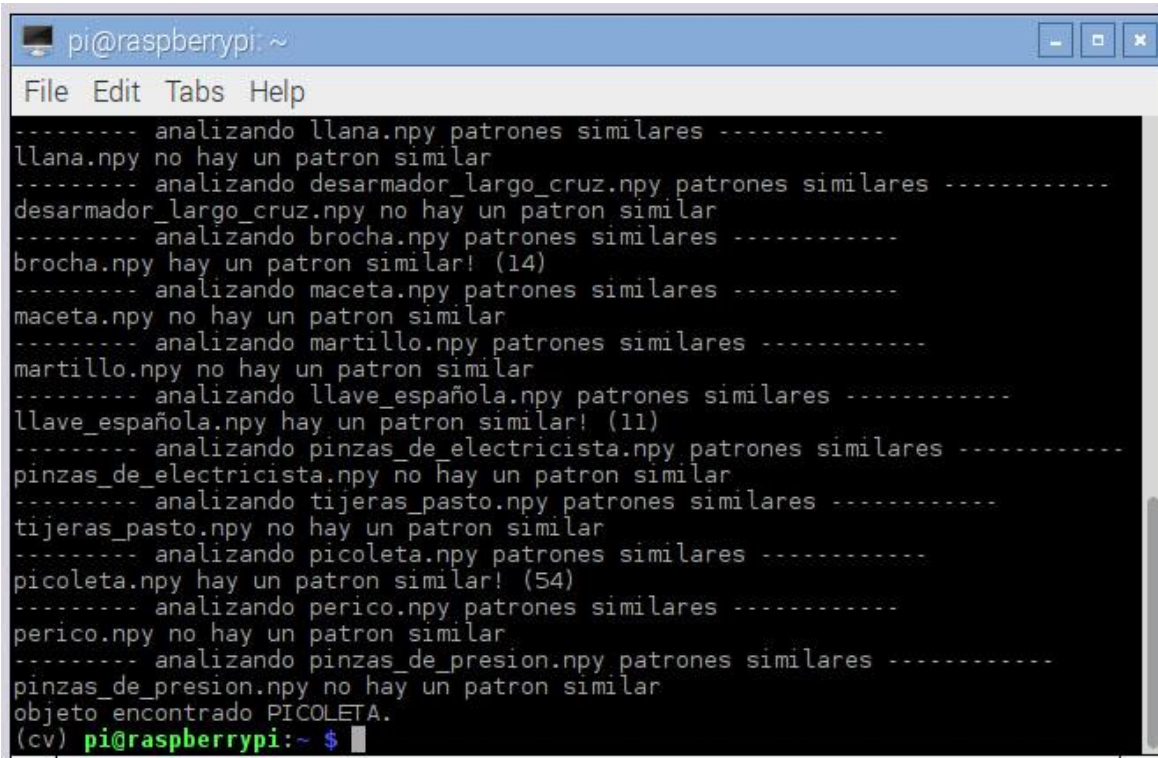
Se declara el número mínimo de coincidencias.

Después se hace uso de la función *print* para indicar el proceso de búsqueda de las imágenes, a la vez de utilizar un bucle *for* para iterar nuestros descriptores e ir buscando en cada descriptor algún parecido con la imagen de entrada (objeto a buscar) con la ayuda de *FLANN*.

Para poder cargar un archivo tipo *numpy dentro de una matriz* se utilizó la función *numpy.load* (nombre del archivo).

En cada iteración el sistema realizará una comparación entre la imagen de entrada con los descriptores almacenados, evaluando si existe o no un parecido, informando si hay una o más coincidencias en cada descriptor.

Para terminar el sistema tomará el mayor número de similitudes de todos los descriptores y en dado caso que un descriptor tenga un muy alto grado de parecido con la imagen de entrada, este se tomará como objeto o imagen encontrada. Ver figura 39.



```
pi@raspberrypi: ~
File Edit Tabs Help
----- analizando llana.npy patrones similares -----
llana.npy no hay un patron similar
----- analizando desarmador_largo_cruz.npy patrones similares -----
desarmador_largo_cruz.npy no hay un patron similar
----- analizando brocha.npy patrones similares -----
brocha.npy hay un patron similar! (14)
----- analizando maceta.npy patrones similares -----
maceta.npy no hay un patron similar
----- analizando martillo.npy patrones similares -----
martillo.npy no hay un patron similar
----- analizando llave_española.npy patrones similares -----
llave_española.npy hay un patron similar! (11)
----- analizando pinzas_de_electricista.npy patrones similares -----
pinzas_de_electricista.npy no hay un patron similar
----- analizando tijeras_pasto.npy patrones similares -----
tijeras_pasto.npy no hay un patron similar
----- analizando picoleta.npy patrones similares -----
picoleta.npy hay un patron similar! (54)
----- analizando perico.npy patrones similares -----
perico.npy no hay un patron similar
----- analizando pinzas_de_presion.npy patrones similares -----
pinzas_de_presion.npy no hay un patron similar
objeto encontrado PICOLETA.
(cv) pi@raspberrypi:~ $
```

Figura 39: Búsqueda de objeto

#### 5.1.1.4 Etapa 4 Integración

Una vez terminado el sistema, es importante aclarar que una parte del programa fue desarrollada principalmente para generar los descriptores de todas las imágenes almacenadas en una carpeta. Este script se guardó con el nombre de `descriptores.py`, para ser identificado más fácilmente y tener un rol en específico a la hora de ejecutar el programa y mostrar sus funcionalidades.

Por otra parte se generó otro script llamado `búsqueda de objeto.py` el cual es el encargado de buscar objetos parecidos en el banco de imágenes, para que de este modo puedan ser examinados todos los objetos almacenados.

El sistema clasifica debidamente arrojando el nombre del objeto (si es que este se encuentra en nuestro banco de imágenes), así que para volverlo un poco más específico se le añadieron nuevas funciones, por mencionar algunas: que nos muestre aparte del nombre del objeto, el precio y la ubicación, además de que nos muestre la imagen del objeto.

Para esto se le añadió la función `imshow` la cual nos permite mostrar la imagen del objeto a buscar. Ver Figura 40.

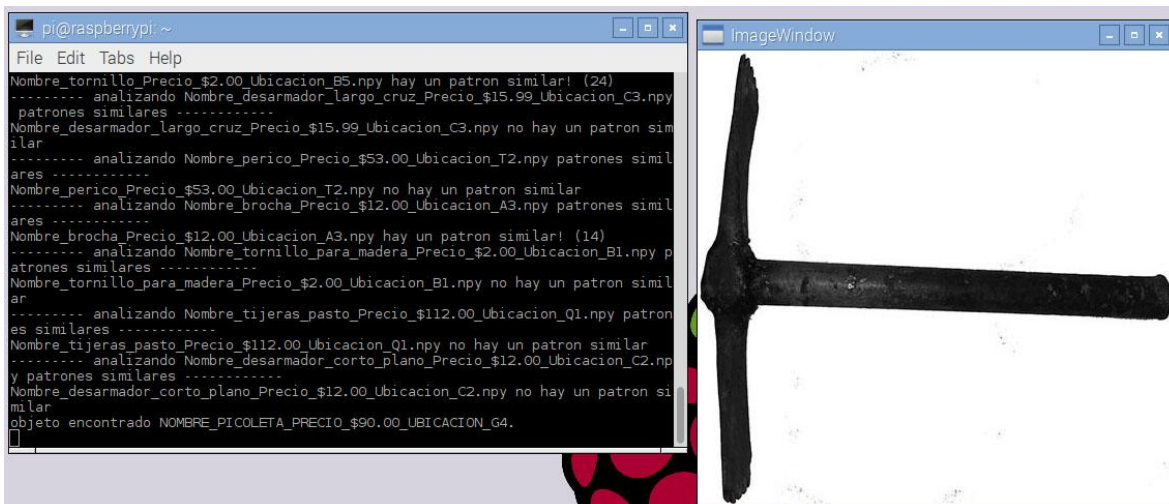


Figura 40: Prueba del sistema de reconocimiento de objetos

## 6 Capítulo 6 Resultados

### 6.1 Etapa 5 Pruebas

En esta parte se necesitará tanto del banco de imágenes previamente procesadas, así como del espacio físico, donde el proceso de captación de imagen se llevará a cabo:

Primeramente se mostrará el procedimiento para almacenar y procesar las imágenes.

Se han seguido los siguientes pasos:

Las imágenes almacenadas tienen un tamaño de 500 x 500

Formato de 8 bits, *RGB*.

Se han escogido 20 objetos diferentes para entrenar la red neuronal que está integrada en *OpenCV*.

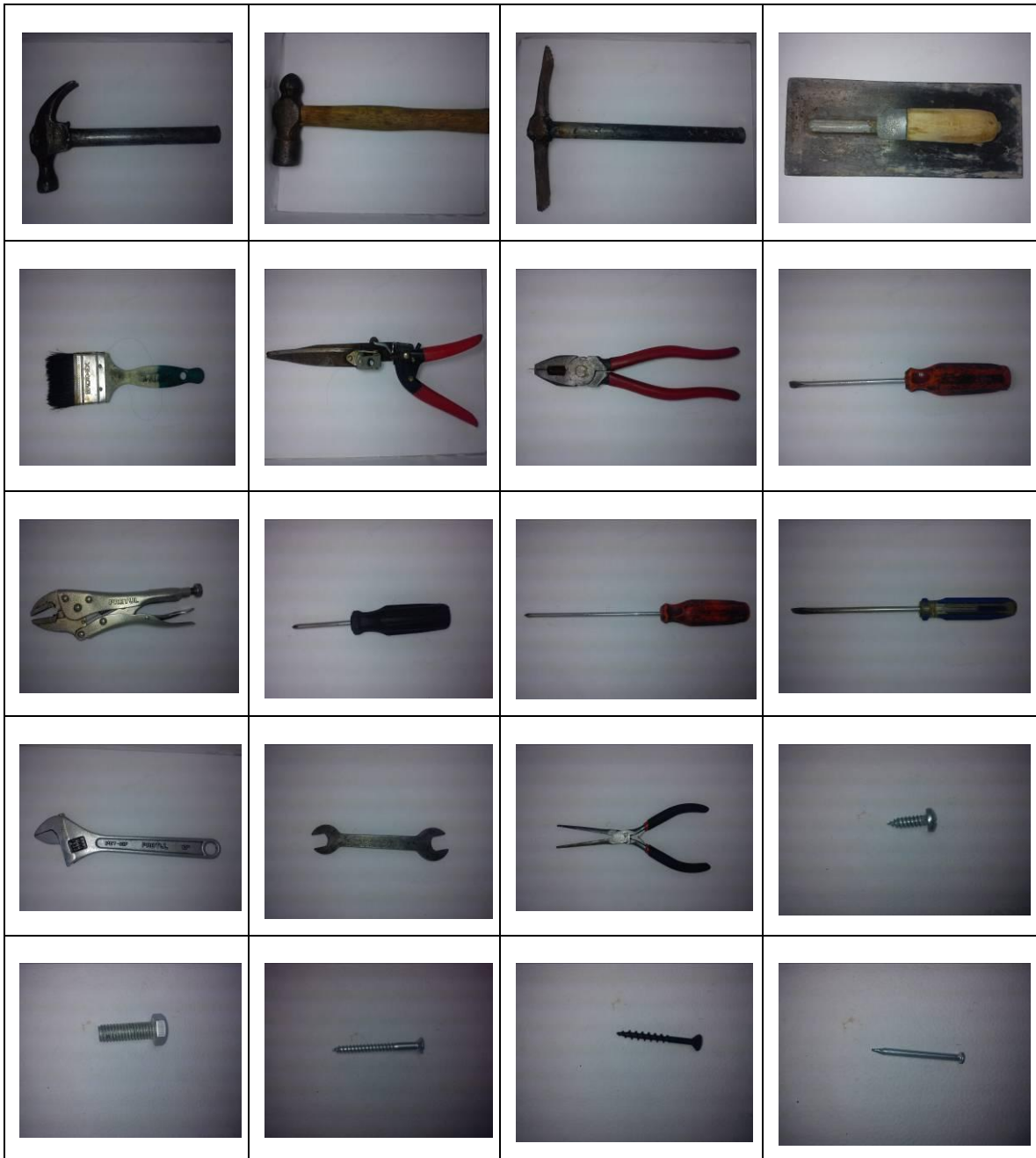
Los objetos fueron captados por la *raspberrypi cámara* con el comando *raspistill -o <nombre de la imagen>*.

Todas las imágenes se almacenaron en una carpeta llamada “tlapalería”

Después de que se juntaron las imágenes de prueba, se les da el tratamiento, por mencionar algunas (recorte de la fotografía para obtener solo el contorno de la figura, eliminar el fondo, cambiar su color a escala de grises), por medio del programa Adobe Photoshop cs6.

La figura 41 muestra las imágenes de todos los objetos que serán almacenados en el banco de imágenes antes de ser procesados con la herramienta *Photoshop*.

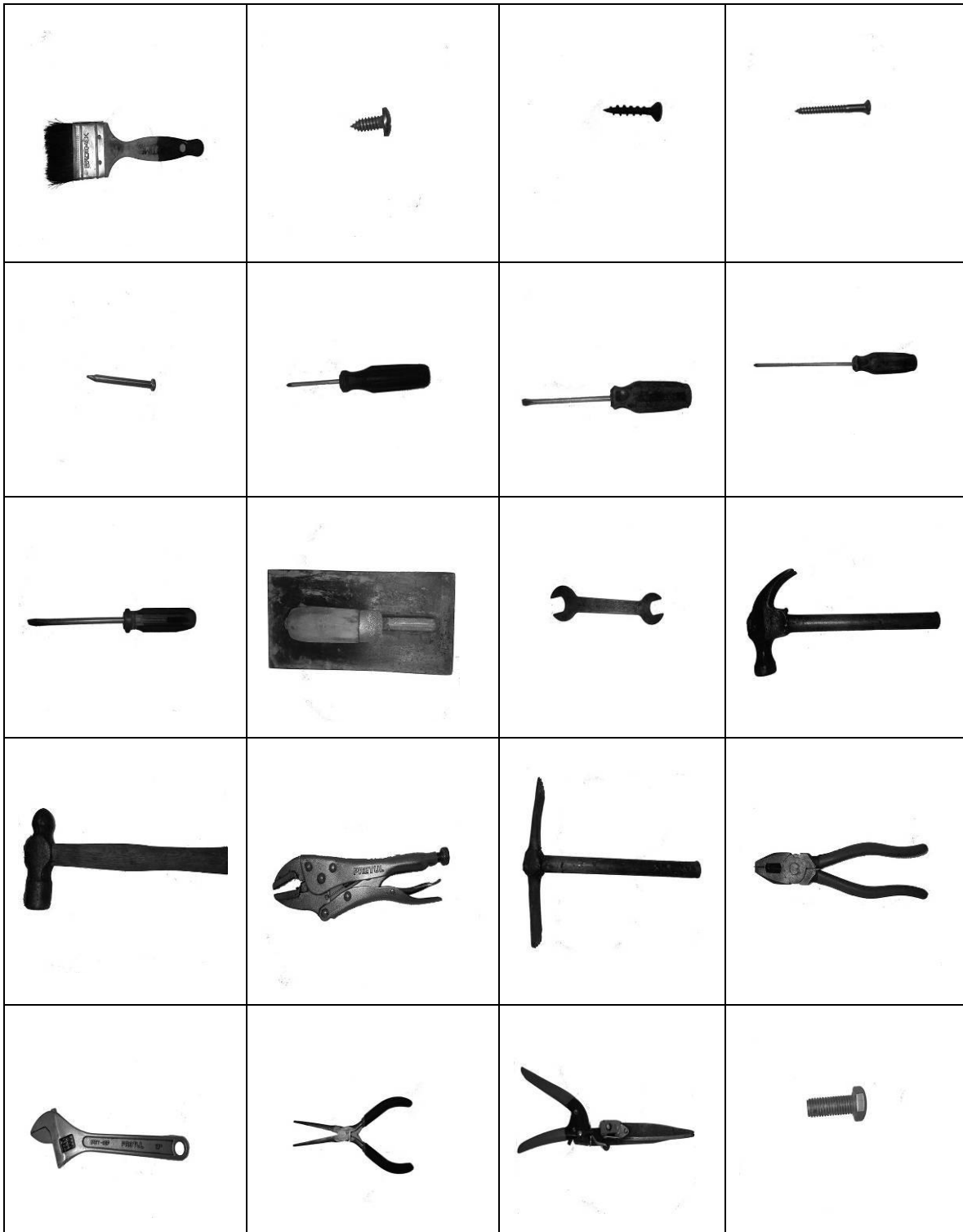
*Imágenes originales*



*Figura 41: Banco de imágenes*

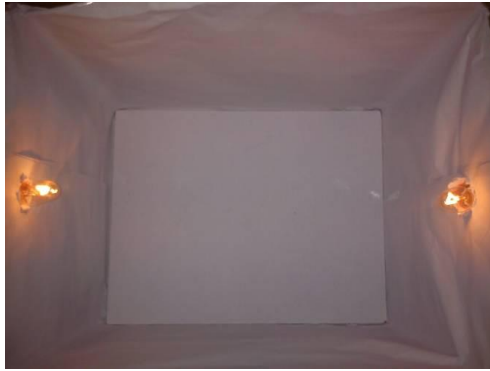
La figura 42 muestra el resultado del procesamiento de las imágenes de todos los objetos que serán almacenados en el banco de imágenes con ayuda de la herramienta *Photoshop*.

*Imágenes procesadas*



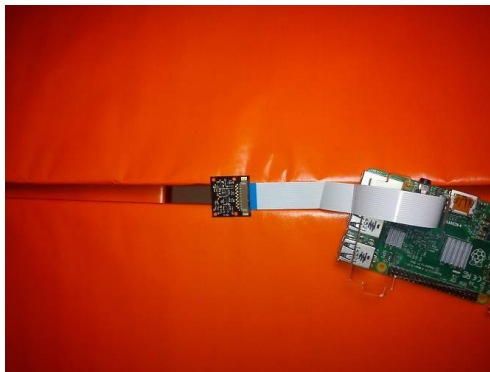
*Figura 42: Banco de imágenes procesado*

### *Espacio físico*



***Figura 43: Espacio físico de captación de la imagen del objeto***

La figura 43 muestra el espacio físico mediante el cual se lleva a cabo el proceso de captación de la imagen de los objetos.



***Figura 44: Proceso de captación de la imagen del objeto con raspberry pi cámara***

La figura 44 muestra la función de la *raspberry pi cámara* para captar las imágenes de los objetos.



***Figura 45: Instalación eléctrica***

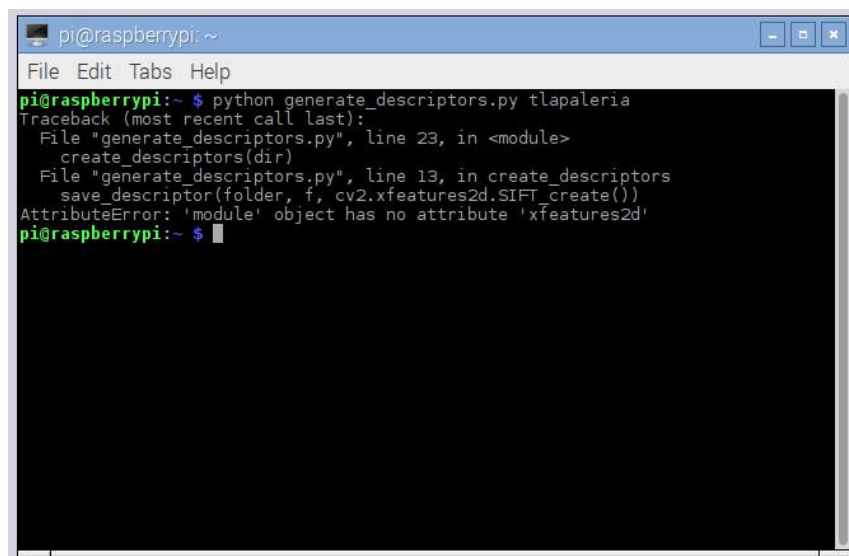
La figura 45 muestra la instalación eléctrica del espacio físico.



Una vez teniendo el banco de imágenes de una manera óptima, se procede a observar el funcionamiento del sistema al momento de buscar y encontrar objetos en base a una imagen de entrada:

El primer objeto a buscar fue un clavo largo, el sistema deberá de reconocerlo debido a que este objeto está almacenado en el banco de imágenes, así como sus datos (nombre, precio, ubicación, fotografía por mencionar algunos).

Entonces primeramente se procede a generar los descriptores del banco de imágenes:

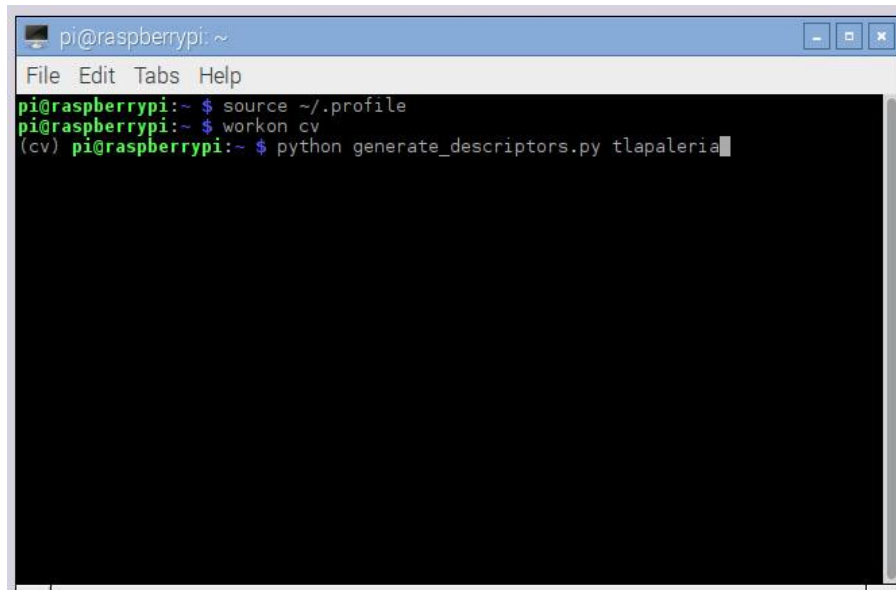


```
pi@raspberrypi:~  
File Edit Tabs Help  
pi@raspberrypi:~ $ python generate_descriptors.py tlapaleria  
Traceback (most recent call last):  
  File "generate_descriptors.py", line 23, in <module>  
    create_descriptors(dir)  
  File "generate_descriptors.py", line 13, in create_descriptors  
    save_descriptor(folder, f, cv2.xfeatures2d.SIFT_create())  
AttributeError: 'module' object has no attribute 'xfeatures2d'  
pi@raspberrypi:~ $
```

**Figura 46: Prueba de generación de descriptores**

La figura 46 Muestra un error de ejecución del script de generación de descriptores, debido a que no se está accediendo desde el ambiente virtual.

Es importante recordar que se está usando la versión 3.0.0 de *OpenCV* y para poder usar *SIFT* se tiene que acceder desde el ambiente virtual.

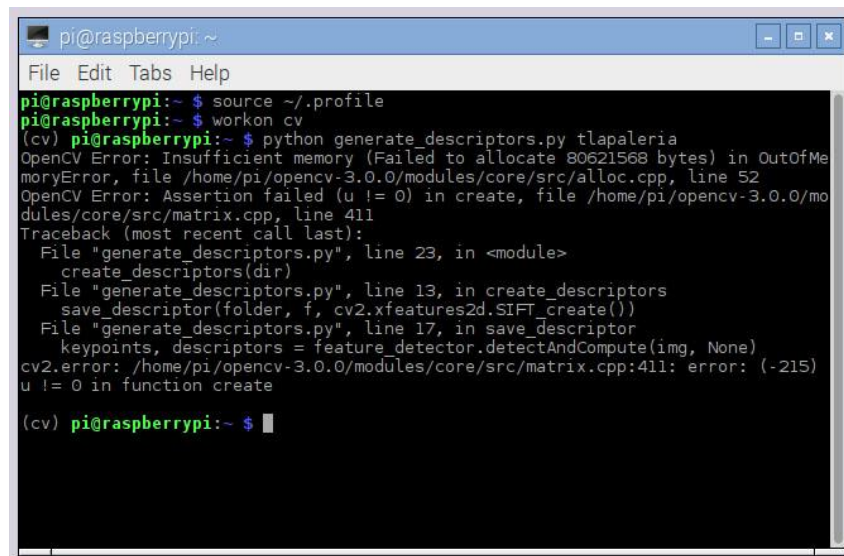


```
pi@raspberrypi: ~  
File Edit Tabs Help  
pi@raspberrypi:~ $ source ~/.profile  
pi@raspberrypi:~ $ workon cv  
(cv) pi@raspberrypi:~ $ python generate_descriptors.py tlapaleria
```

*Figura 47: Accediendo al ambiente virtual*

La figura 47 Muestra la manera de acceder al ambiente virtual para poder usar eficazmente SIFT.

Una vez aclarado este punto se procede con la ejecución del script:



```
pi@raspberrypi: ~  
File Edit Tabs Help  
pi@raspberrypi:~ $ source ~/.profile  
pi@raspberrypi:~ $ workon cv  
(cv) pi@raspberrypi:~ $ python generate_descriptors.py tlapaleria  
OpenCV Error: Insufficient memory (Failed to allocate 80621568 bytes) in OutOfMemoryError, file /home/pi/opencv-3.0.0/modules/core/src/alloc.cpp, line 52  
OpenCV Error: Assertion failed (u != 0) in create, file /home/pi/opencv-3.0.0/modules/core/src/matrix.cpp, line 411  
Traceback (most recent call last):  
  File "generate_descriptors.py", line 23, in <module>  
    create_descriptors(dir)  
  File "generate_descriptors.py", line 13, in create_descriptors  
    save_descriptor(folder, f, cv2.xfeatures2d.SIFT_create())  
  File "generate_descriptors.py", line 17, in save_descriptor  
    keypoints, descriptors = feature_detector.detectAndCompute(img, None)  
cv2.error: /home/pi/opencv-3.0.0/modules/core/src/matrix.cpp:411: error: (-215) u != 0 in function create  
(cv) pi@raspberrypi:~ $
```

*Figura 48: Segunda prueba de generación de descriptores*

La figura 48 Muestra un error al tratar de generar los descriptores de las imágenes contenidas en la carpeta tlapalería.

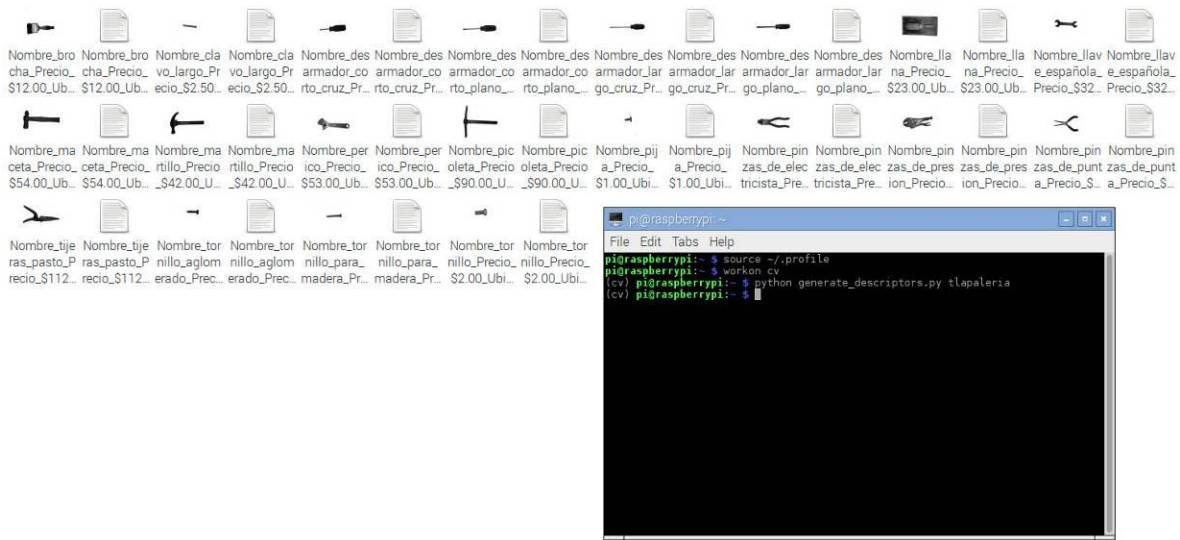
La salida fue un error debido a que las imágenes tenían unas dimensiones realmente grandes en cuanto a anchura y altura en pixeles,



*Figura 49: Imagen del objeto del banco de imágenes*

La figura 49 Muestra el objeto fotografiado por la *raspberry pi 2* cámara la cual nos da las dimensiones que poseen todos los objetos del banco de imágenes.

Por tal motivo se decidió trabajar con esas imágenes cambiándolas de tamaño a 500 x 500 pixeles, de este modo el problema quedó resuelto:



**Figura 50: Resultados de la ejecución del script para generar descriptores**

La figura 50 Muestra el resultado de la generación de los descriptores de las imágenes contenidas en la carpeta tlapalería de manera eficaz.

Ahora que ya se tienen los descriptores de los respectivos objetos, lo siguiente es que a partir de una imagen de entrada (objeto a buscar) sea comparado con el banco de imágenes, para así poder encontrar posibles similitudes en base a la imagen de entrada. En dado caso que un objeto de nuestro banco de imágenes posea un alto grado de patrones similares se tomara como un objeto parecido y por ende encontrado.

De este modo se procede a probar la red neuronal que está integrada en *OpenCV*:

El primer objeto a buscar fue un clavo largo:

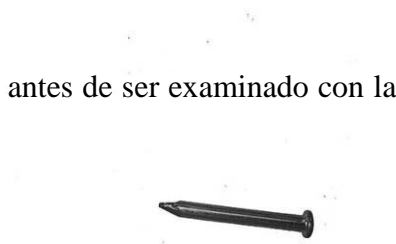
Se toma la foto del clavo:



**Figura 51: Clavo**

La figura 51 Muestra la imagen del objeto a buscar en el banco de imágenes antes de ser pre-procesado.

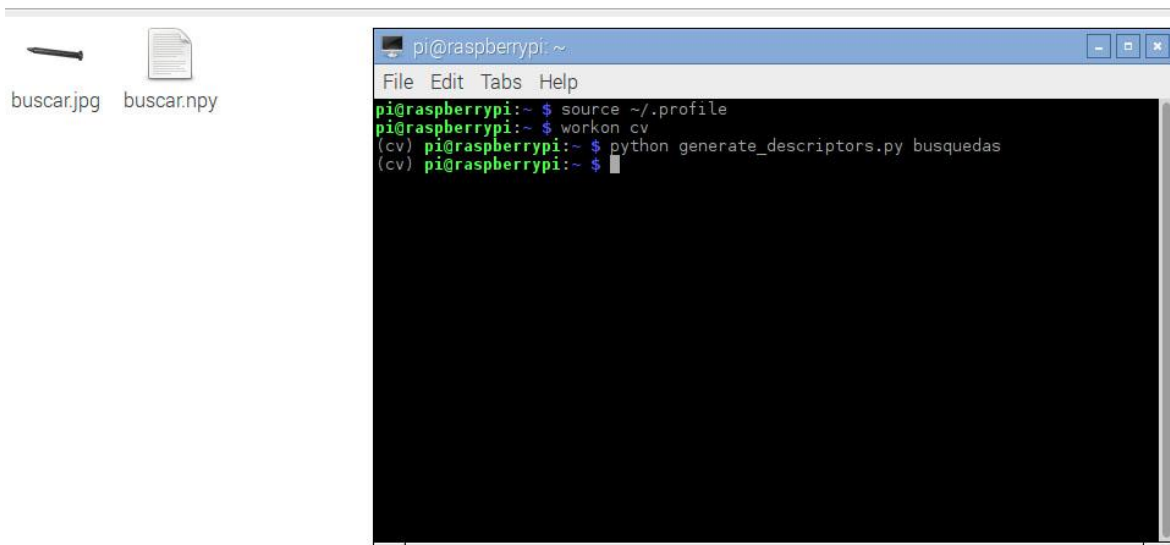
Se le aplica su procesamiento antes de ser examinado con la herramienta *Adobe photoshop cs6*:



**Figura 52: Clavo procesado**

La figura 52 Muestra el resultado del procesamiento de la imagen del objeto de entrada.

Se procede a generar el descriptor de la imagen para poder ser examinada en búsqueda de similitudes con el banco de imágenes:



**Figura 53: Generación del descriptor del objeto a buscar**

La figura 53 Muestra el resultado de la generación de los descriptores del objeto a buscar.

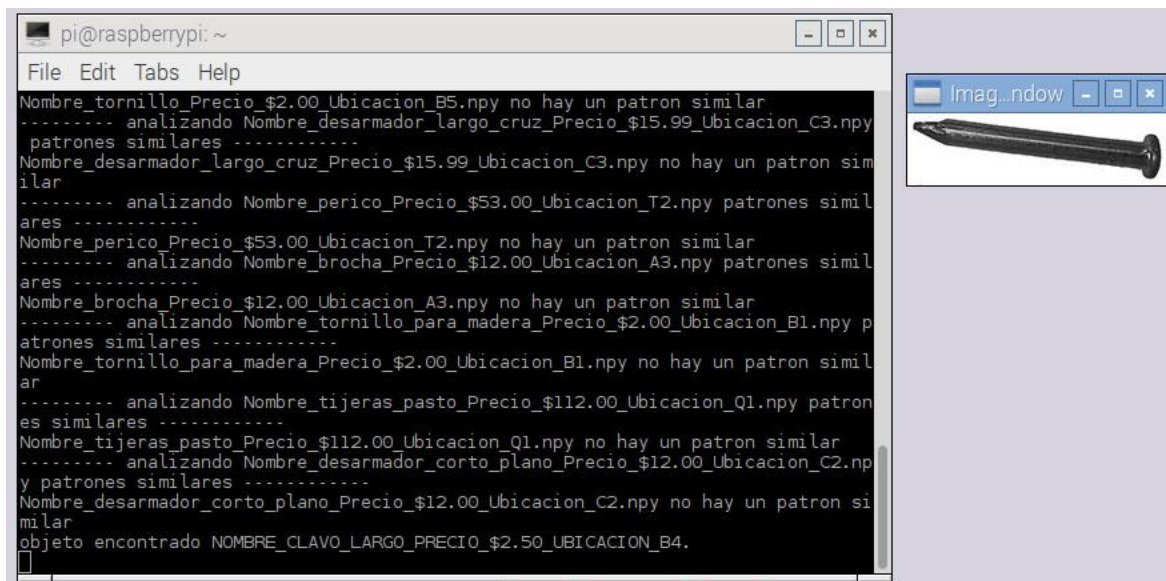
Se pasa el descriptor e imagen del objeto a buscar en el banco de imágenes:



**Figura 54: Incorporación del descriptor de la imagen de entrada al banco de imágenes**

La figura 54 Muestra la incorporación del descriptor del objeto a buscar al banco de imágenes.

Acto seguido se ejecuta el script para poder saber si en el banco de imágenes existe un objeto con algún parentesco a este.

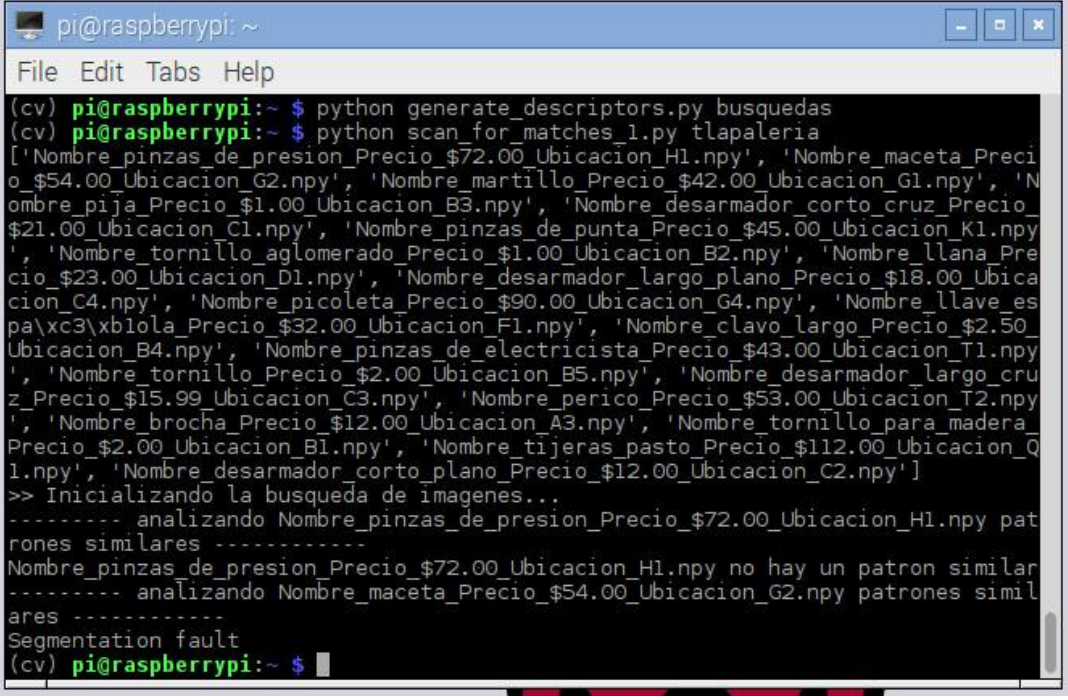


**Figura 55: Resultado de búsqueda de la imagen de entrada**

La figura 55 Muestra el resultado de la búsqueda del objeto a buscar en el banco de imágenes. El cual arroja los datos específicos sobre el objeto.

El sistema lo reconoció debidamente: Entonces se procede a examinar todas nuestras demás imágenes con el mismo procedimiento antes mencionado:

De los 20 objetos almacenados el sistema reconoció 15 sin ningún problema, el sistema generó errores cuando intentó reconocer a una pija, una brocha, un perico, unas pinzas de presión y una llana, en el caso de la pija el script arrojó la siguiente salida:

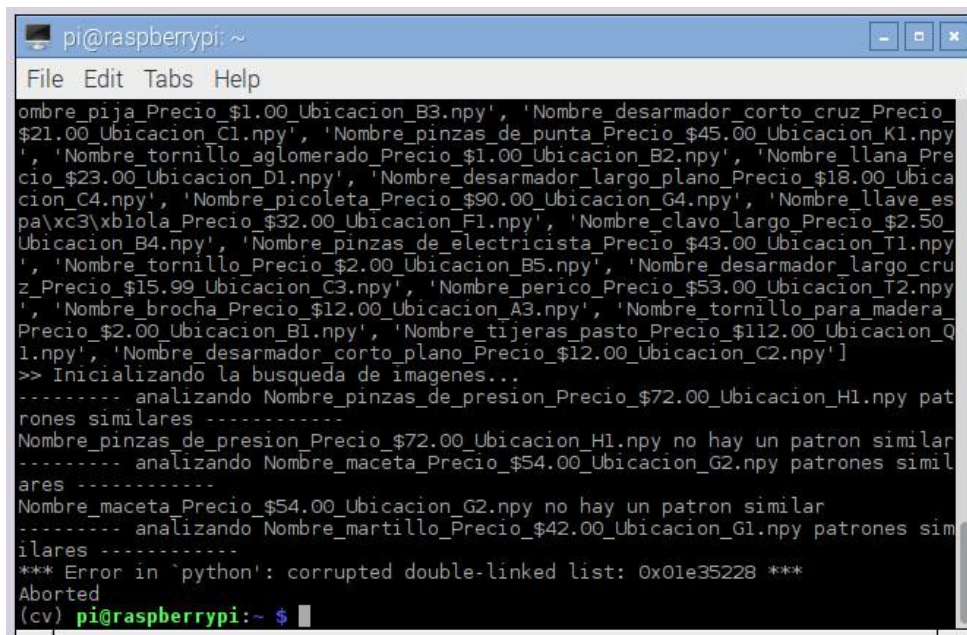


```
pi@raspberrypi: ~
File Edit Tabs Help
(cv) pi@raspberrypi:~ $ python generate_descriptors.py busquedas
(cv) pi@raspberrypi:~ $ python scan_for_matches_1.py tlapaleria
['Nombre_pijas_de_presion_Precio_$72.00_Ubicacion_H1.npy', 'Nombre_maceta_Precio_$54.00_Ubicacion_G2.npy', 'Nombre_martillo_Precio_$42.00_Ubicacion_G1.npy', 'Nombre_pija_Precio_$1.00_Ubicacion_B3.npy', 'Nombre_desarmador_corto_cruz_Precio_$21.00_Ubicacion_C1.npy', 'Nombre_pijas_de_punta_Precio_$45.00_Ubicacion_K1.npy', 'Nombre_tornillo_aglomerado_Precio_$1.00_Ubicacion_B2.npy', 'Nombre_llana_Precio_$23.00_Ubicacion_D1.npy', 'Nombre_desarmador_largo_plano_Precio_$18.00_Ubicacion_C4.npy', 'Nombre_picoleta_Precio_$90.00_Ubicacion_G4.npy', 'Nombre_llave_espa\nc3\xblola_Precio_$32.00_Ubicacion_F1.npy', 'Nombre_clavo_largo_Precio_$2.50_Ubicacion_B4.npy', 'Nombre_pijas_de_electricista_Precio_$43.00_Ubicacion_T1.npy', 'Nombre_tornillo_Precio_$2.00_Ubicacion_B5.npy', 'Nombre_desarmador_largo_cruz_Precio_$15.99_Ubicacion_C3.npy', 'Nombre_perico_Precio_$53.00_Ubicacion_T2.npy', 'Nombre_brocha_Precio_$12.00_Ubicacion_A3.npy', 'Nombre_tornillo_para_madera_Precio_$2.00_Ubicacion_B1.npy', 'Nombre_tijeras_pasto_Precio_$112.00_Ubicacion_Q1.npy', 'Nombre_desarmador_corto_plano_Precio_$12.00_Ubicacion_C2.npy']
>> Inicializando la busqueda de imagenes..
----- analizando Nombre_pijas_de_presion_Precio_$72.00_Ubicacion_H1.npy patrones similares -----
Nombre_pijas_de_presion_Precio_$72.00_Ubicacion_H1.npy no hay un patron similar
----- analizando Nombre_maceta_Precio_$54.00_Ubicacion_G2.npy patrones similares -----
Segmentation fault
(cv) pi@raspberrypi:~ $
```

*Figura 56: Pantalla de error de segmentación de imagen*

La figura 56 Muestra el resultado de la búsqueda de una pija en el banco de imágenes. El cual es un error de segmentación ocasionado por un procesamiento equivocado de la imagen

En los demás casos la salida fue la siguiente:

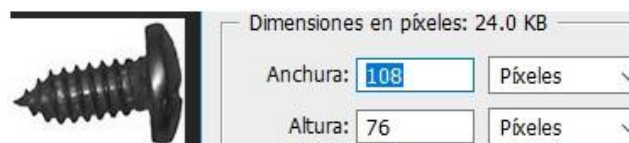


```
pi@raspberrypi: ~
File Edit Tabs Help
Nombre_pija_Precio_$1.00_Ubicacion_B3.npy', 'Nombre_desarmador_corto_cruz_Precio_
$21.00_Ubicacion_C1.npy', 'Nombre_pinzas_de_punta_Precio_$45.00_Ubicacion_K1.npy
', 'Nombre_tornillo_aglomerado_Precio_$1.00_Ubicacion_B2.npy', 'Nombre_llana_Pre
cio_$23.00_Ubicacion_D1.npy', 'Nombre_desarmador_largo_plano_Precio_$18.00_Ubica
cion_C4.npy', 'Nombre_picoleta_Precio_$90.00_Ubicacion_G4.npy', 'Nombre_llave_es
pa\xc3\xblola_Precio_$32.00_Ubicacion_F1.npy', 'Nombre_clavo_largo_Precio_$2.50_
Ubicacion_B4.npy', 'Nombre_pinzas_de_electricista_Precio_$43.00_Ubicacion_T1.npy
', 'Nombre_tornillo_Precio_$2.00_Ubicacion_B5.npy', 'Nombre_desarmador_largo_cru
z_Precio_$15.99_Ubicacion_C3.npy', 'Nombre_perico_Precio_$53.00_Ubicacion_T2.npy
', 'Nombre_brocha_Precio_$12.00_Ubicacion_A3.npy', 'Nombre_tornillo_para_madera
Precio_$2.00_Ubicacion_B1.npy', 'Nombre_tijeras_pasto_Precio_$112.00_Ubicacion_Q
1.npy', 'Nombre_desarmador_corto_plano_Precio_$12.00_Ubicacion_C2.npy']
>> Inicializando la busqueda de imagenes...
----- analizando Nombre_pinzas_de_presion_Precio_$72.00_Ubicacion_H1.npy pat
rones similares -----
Nombre_pinzas_de_presion_Precio_$72.00_Ubicacion_H1.npy no hay un patron similar
----- analizando Nombre_maceta_Precio_$54.00_Ubicacion_G2.npy patrones simil
ares -----
Nombre_maceta_Precio_$54.00_Ubicacion_G2.npy no hay un patron similar
----- analizando Nombre_martillo_Precio_$42.00_Ubicacion_G1.npy patrones sim
ilares -----
*** Error in `python': corrupted double-linked list: 0x01e35228 ***
Aborted
(cv) pi@raspberrypi:~ $
```

*Figura 57: Pantalla de error de imagen*

La figura 57 Muestra el resultado de la búsqueda de los objetos: brocha, perico, pinzas de presión y una llana. El cual fue un error por el mal procesamiento de las imágenes de los objetos.

Esto se originó porque al momento de extraer el contorno del objeto a buscar la herramienta de-procesamiento modificó las dimensiones de las imágenes por lo que no pudieron ser examinadas debidamente:

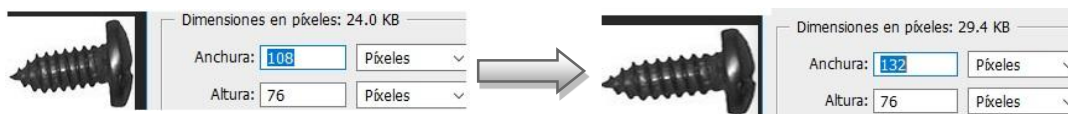


*Figura 58: Dimensiones de la imagen de entrada*

La figura 58 muestra el resultado del procesamiento de las imágenes de entrada posteriormente.



La solución fue modificar las dimensiones de cada una de las imágenes:



*Figura 59: Imagen original segmentada por el editor*

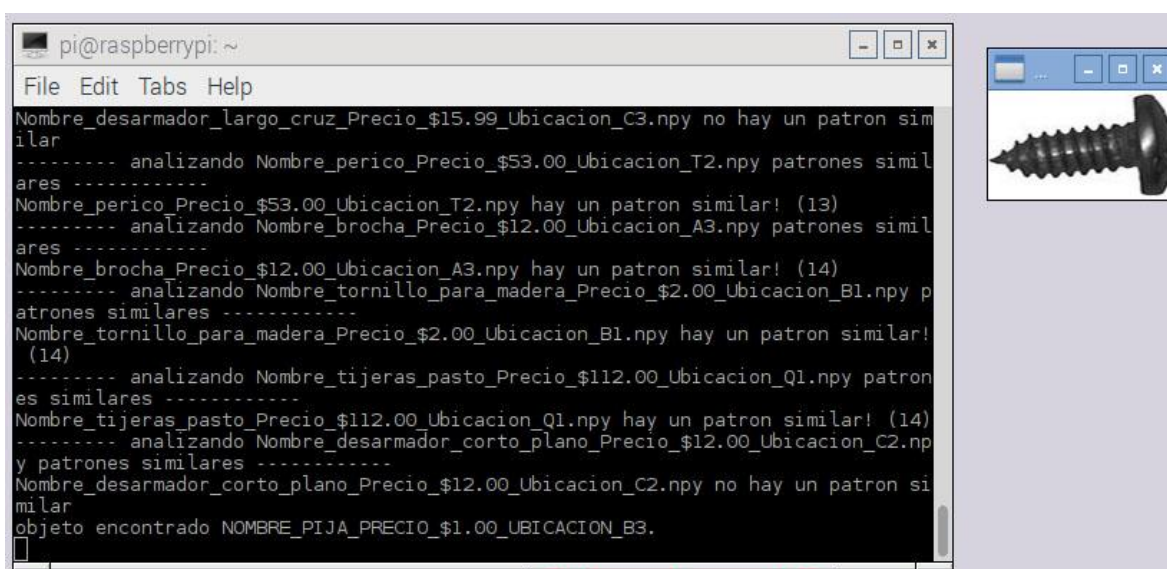
*Imagen retocada posteriormente.*

*Antes*

*Después*

La figura 59 Muestra el retoque de las dimensiones en pixeles a las imágenes de entrada posteriormente.

Ejecutando el script se obtuvo la siguiente salida:



*Figura 60: Resultados de la búsqueda de la imagen de entrada.*

La figura 60 Simplifica el resultado final de la búsqueda de las imágenes de los objetos faltantes de entrada. Arroja los datos específicos sobre el objeto.

De este modo los cinco objetos faltantes fueron reconocidos satisfactoriamente.

Es importante mencionar que en el caso de objetos pequeños tales como clavo, pijas, etc. la distancia debe ser más aproximada al objeto, de este modo se puede obtener la forma del objeto más detalladamente.

## 7 Capítulo 7 Conclusiones

Después de haber aplicado la metodología y obtenido resultados se ha llegado a las siguientes conclusiones.

El problema planteado consistió en reconocer objetos usando una computadora *Raspberry Pi 2* y la Librería *OpenCV*, por lo que, una vez que se llevó a cabo la metodología se obtuvieron resultados que muestran la posibilidad de reconocer objetos de una tlapalería, por lo tanto, se concluye que el problema podría quedar resuelto si se implementa la propuesta de solución.

Mientras que, el objetivo que se planteó de reconocer objetos a partir de imágenes digitales se logró usando la librería *OpenCV* y dos programas desarrollados en *Python* que fueron: El que genera los descriptores de las imágenes y el otro que busca el objetos en el banco de imágenes. Por lo que, el objetivo queda cumplido.

Por otro lado, la hipótesis que dice: “*Si se implementa un sistema de reconocimiento de objetos en las tlapalerías será posible reconocer objetos a partir de una fotografía así como su existencia y características del objeto buscado*”, y que de acuerdo a los resultados la hipótesis ha resultado verdadera al lograrse el reconocimiento de objetos.

Las características de los objetos del banco de imágenes fueron almacenadas en los descriptores de los mismos.

Es importante mencionar que, en la metodología se usó un conjunto de 20 imágenes que se tuvieron que estandarizar en tamaño de pixeles y se cambiaron a escala de grises. Con lo anterior fue posible el reconocimiento del 75% de objetos, con un 80% de grado de similitud. Sin embargo, falló en algunos objetos, y fue necesario un procesamiento de cambio de tamaño y una nueva segmentación. Con lo anterior fue posible el reconocimiento de los 20 objetos.

Se requirió obtener la imagen a partir de una fotografía del objeto, luego se procesó para su segmentación y luego pasar a la escala de grises y finalmente el proceso de reconocimiento de imágenes. Todo lo anterior, se requiere de al menos 4 minutos para obtener la imagen y la búsqueda del objeto tardó segundos, pero cambia al aumentar el banco de imágenes.

Este tipo de sistema una vez implementado en alguna tlapalería sustituiría la búsqueda de objetos en la base de datos, obteniendo una búsqueda de objetos más detallada y a la vez brindando a los clientes una mayor atención personal.

### 7.1 Temas futuros

Los pendientes rescatados después de concluir este trabajo son:

1. Probar el sistema con un gran conjunto de imágenes, considerando que una tlapalería puede contar con cientos e incluso miles de artículos.
2. Probar el reconocimiento de objetos con un equipo de hardware de alto rendimiento, digamos con un servidor.
3. Resolver el problema de reconocimiento de objetos considerando el tamaño, es decir unos son muy grandes y otros muy pequeños, por ejemplo clavos, tornillos, desarmadores, palas, escaleras, etc.
4. Tomar la foto con la cámara de la *Raspberry* y se haga el pre-procesamiento de la imagen de forma automática para hacer el reconocimiento sin mayor intervención.
5. Comparar otras técnicas de reconocimiento de objetos.

## 7.2 Referencias

ahorayasabes. (20 de Julio de 2011). <http://ahorayasabes.blogspot.mx/>. Recuperado el 10 de Septiembre de 2016, de <http://ahorayasabes.blogspot.mx/>:  
<http://ahorayasabes.blogspot.mx/2011/07/tlapaleria.html>

Alegsa, L. (6 de Julio de 2016). <http://www.alegsa.com.ar/>. Recuperado el 12 de Septiembre de 2016, de <http://www.alegsa.com.ar/>:  
<http://www.alegsa.com.ar/Dic/photoshop.php>

alojamientos. (2016). <http://alojamientos.us.es/>. Recuperado el 10 de Septiembre de 2016, de <http://alojamientos.us.es/>: <http://alojamientos.us.es/gtocom/pid/tema4.pdf>

Aracil, R. (28 de Julio de 2012). <https://riunet.upv.es>. Recuperado el 15 de Septiembre de 2016, de <https://riunet.upv.es>:  
<https://riunet.upv.es/bitstream/handle/10251/17010/memoria.pdf?sequence=1>

Arevalo, M., Gonzalez, J., & Ambrosio, G. (2016). <http://mapir.isa.uma.es/>. Recuperado el 15 de Septiembre de 2016, de <http://mapir.isa.uma.es/>:  
<http://mapir.isa.uma.es/varevalo/drafts/arevalo2004lva1.pdf>

Benito, M. (2016). [uc3m.es](http://uc3m.es). Recuperado el 14 de Octubre de 2016, de [uc3m.es](http://uc3m.es): [http://e-archivo.uc3m.es/bitstream/handle/10016/17821/Resumen\\_Espanol\\_PFC\\_Manuel\\_Sayago\\_Benito.pdf?sequence=1](http://e-archivo.uc3m.es/bitstream/handle/10016/17821/Resumen_Espanol_PFC_Manuel_Sayago_Benito.pdf?sequence=1)

bibing. (2016). <http://bibing.us.es/>. Recuperado el 10 de Septiembre de 2016, de <http://bibing.us.es/>:  
<http://bibing.us.es/proyectos/abreproy/11494/fichero/PROYECTO%252FCapitulo+3.pdf>

Braude, E. (2003). *Ingenieria de Software una perspectiva orientada a objetos*. Mexico: Alfaomega.

Bueno, D. (Mayo de 2012). <http://invenio2.unizar.es/>. Recuperado el 15 de Septiembre de 2016, de <http://invenio2.unizar.es/>: <http://invenio2.unizar.es/record/7131/files/TAZ-PFC-2012-183.pdf>

construamablanca. (2016). <http://construamablanca.com/>. Recuperado el 10 de Septiembre de 2016, de <http://construamablanca.com/>:  
[http://construamablanca.com/ver\\_categoria.php?id=2](http://construamablanca.com/ver_categoria.php?id=2)

Delgado. (2009) Manual avanzado de Photoshop cs4, Anaya multimedia.

Diaz, I., Montoya, D., & Boulanger, P. (Diciembre de 2007). [scielo.org.co](http://scielo.org.co). Recuperado el 11 de Octubre de 2016, de [scielo.org.co](http://scielo.org.co):  
[http://www.scielo.org.co/scielo.php?script=sci\\_arttext&pid=S1692-33242007000200011](http://www.scielo.org.co/scielo.php?script=sci_arttext&pid=S1692-33242007000200011)

ehu. (2016). [sc.ehu.es](http://sc.ehu.es). Recuperado el 12 de Octubre de 2016, de [sc.ehu.es](http://sc.ehu.es):  
<http://www.sc.ehu.es/ccwgrrom/transparencias/pdf-vision-1-transparencias/capitulo-6.pdf>

Escalante, B. (6 de Octubre de 2006). <http://verona.fi-p.unam.mx/>. Recuperado el 10 de Septiembre de 2016, de <http://verona.fi-p.unam.mx/>: <http://verona.fi-p.unam.mx/boris/teachingnotes/Capitulo4.pdf>

Espitia, W. (2014). <http://repository.unimilitar.edu.co/>. Recuperado el 15 de Septiembre de 2016, de <http://repository.unimilitar.edu.co/>: <http://repository.unimilitar.edu.co/bitstream/10654/12956/1/Tesis%20de%20Grado%20Wilson%20Espitia%20Manta.pdf>

Florencia. (2016). <http://catarina.udlap.mx/>. Recuperado el 13 de Septiembre de 2016, de <http://catarina.udlap.mx/>: [http://catarina.udlap.mx/u\\_dl\\_a/tales/documentos/msp/florencia\\_y\\_an/capitulo3.pdf](http://catarina.udlap.mx/u_dl_a/tales/documentos/msp/florencia_y_an/capitulo3.pdf)

Gambini, M. (2006). [dc.uba.ar](http://dc.uba.ar/). Recuperado el 12 de Octubre de 2016, de [dc.uba.ar](http://dc.uba.ar/): <http://www-2.dc.uba.ar/grupinv/imagenes/archivos/tesisGambini.pdf>

Garcia, E. (Mayo de 2008). <http://newton.azc.uam.mx/>. Recuperado el 14 de Septiembre de 2016, de <http://newton.azc.uam.mx/>: [http://newton.azc.uam.mx/mcc/01\\_esp/11\\_tesis/tesis/terminada/080513\\_garcia\\_santillan\\_elias.pdf](http://newton.azc.uam.mx/mcc/01_esp/11_tesis/tesis/terminada/080513_garcia_santillan_elias.pdf)

Garcia, F. (30 de Noviembre de 2009). <http://e-archivo.uc3m.es/>. Recuperado el 10 de Septiembre de 2016, de <http://e-archivo.uc3m.es/>: [http://e-archivo.uc3m.es/bitstream/handle/10016/10007/PFC\\_FranciscoJavier\\_Garcia\\_Fernandez.pdf?sequence=3](http://e-archivo.uc3m.es/bitstream/handle/10016/10007/PFC_FranciscoJavier_Garcia_Fernandez.pdf?sequence=3)

Garcia, P. (2012). <http://eprints.sim.ucm.es/>. Recuperado el 13 de Septiembre de 2016, de <http://eprints.sim.ucm.es/>: <http://eprints.sim.ucm.es/23444/1/ProyectoFinMasterPedroPablo.pdf>

Giraldo, D., Roldan, M., & Garro, D. (13 de Noviembre de 2009). [slideshare](http://slideshare.net/). Recuperado el 12 de Septiembre de 2016, de [es.slideshare.net](http://es.slideshare.net/): <http://es.slideshare.net/FERRETERIALA87/proyecto-ferreteria-la-87>

Gomez, V., & Guerrero, A. (Mayo de 2016). [utp.edu.co](http://utp.edu.co/). Recuperado el 14 de Octubre de 2016, de [utp.edu.co](http://utp.edu.co/): <http://repositorio.utp.edu.co/dspace/bitstream/handle/11059/6494/00642G633.pdf?sequence=1>

González, D. (2016). <http://ocw.usal.es/>. Recuperado el 13 de Septiembre de 2016, de <http://ocw.usal.es/>: <http://ocw.usal.es/eduCommons/enseanzas-tecnicas/procesamiento-avanzado-de-imagenes-digitales/contenidos/Tema2.pdf>

Gonzalez, R. (2016). [https://launchpadlibrarian.net](https://launchpadlibrarian.net/). Recuperado el 15 de Septiembre de 2016, de [https://launchpadlibrarian.net](https://launchpadlibrarian.net/): <https://launchpadlibrarian.net/18980633/Python%20para%20todos.pdf>

Gonzalez, R., & Woods, R. (2002). *Procesamiento Digital de Imagenes*. Estados Unidos: Prentice Hall.

Guitart, M. (18 de Febrero de 2009). <http://www.maia.ub.es/>. Recuperado el 15 de Septiembre de 2016, de <http://www.maia.ub.es/>: <http://www.maia.ub.es/~sergio/linked/mario09.pdf>

Jimenez, C. (2016). <http://www.nebrija.es/>. Recuperado el 10 de Septiembre de 2016, de <http://www.nebrija.es/>: <http://www.nebrija.es/~cjimenez/teoria/herramientas/mapasbits.pdf>

Juarez, C. (Diciembre de 2011). <http://e-archivo.uc3m.es/>. Recuperado el 15 de Septiembre de 2016, de <http://e-archivo.uc3m.es/>: [http://e-archivo.uc3m.es/bitstream/handle/10016/13624/PFC\\_Cesar\\_Juarez\\_Megias.pdf?sequence=2](http://e-archivo.uc3m.es/bitstream/handle/10016/13624/PFC_Cesar_Juarez_Megias.pdf?sequence=2)

Krogh, P. (2010). *Gestion del archivo digital para fotografos*. Madrid: Ediciones Anaya multimedia.

Larrañaga, P., Inza, I., & Moujahid, A. (2016). [sc.ehu.es](http://sc.ehu.es). Recuperado el Octubre de 10 de 2016, de [sc.ehu.es](http://sc.ehu.es): <http://www.sc.ehu.es/ccwbayes/docencia/mmcc/docs/t6bayesianos.pdf>

Martin, M. (Enero de 2013). <http://mmartin.cs.buap.mx/>. Recuperado el 13 de Septiembre de 2016, de <http://mmartin.cs.buap.mx/>: <http://mmartin.cs.buap.mx/notas/PDI-MM-Rev.2013.pdf>

Martinez, E. (2016). [turing.iimas.unam.m](http://turing.iimas.unam.m). Recuperado el 12 de Octubre de 2016, de [turing.iimas.unam.m](http://turing.iimas.unam.m): [turing.iimas.unam.mx/~elena/PDI-Mast/Tema\\_6\\_C.pp](http://turing.iimas.unam.mx/~elena/PDI-Mast/Tema_6_C.pp)

McMahon & Nichols, (2007) *Paint shop pro x para fotografos*. Ediciones Anaya multimedia.

Morante, S. (Mayo de 2012). <http://asrob.uc3m.es/>. Recuperado el 15 de Septiembre de 2016, de <http://asrob.uc3m.es/>: [http://asrob.uc3m.es/images/7/71/Tfg\\_completo\\_SMorante.pdf](http://asrob.uc3m.es/images/7/71/Tfg_completo_SMorante.pdf)

Pacheco, J. (Junio de 2011). <http://repository.javeriana.edu.co/>. Recuperado el 15 de Septiembre de 2016, de <http://repository.javeriana.edu.co/>: <http://repository.javeriana.edu.co/bitstream/10554/7073/1/tesis535.pdf>

Patiño, T. (Febrero de 2012). <http://dspace.ups.edu.ec/>. Recuperado el 15 de Septiembre de 2016, de <http://dspace.ups.edu.ec/>: <http://dspace.ups.edu.ec/bitstream/123456789/1709/14/UPS-CT002311.pdf>

Peralta, R. (Mayo de 2013). <http://www.ptolomeo.unam.mx/>. Recuperado el 15 de Septiembre de 2016, de <http://www.ptolomeo.unam.mx/>:

<http://www.ptolomeo.unam.mx:8080/xmlui/bitstream/handle/132.248.52.100/6876/Tesis.pdf?sequence=1>

Pleeger, S. (2002). *Ingeniería de Software. Teoría y práctica*. Buenos Aires: Pearson Education.

procesosdesoftware. (2016). <https://procesosdesoftware.wikispaces.com>. Recuperado el 22 de Septiembre de 2016, de <https://procesosdesoftware.wikispaces.com>: <https://procesosdesoftware.wikispaces.com/file/view/ciclosdevidadelsoftware.pdf/579330701/ciclosdevidadelsoftware.pdf>

Quiles, F., & Garrido, A. (1996). *Computadores paralelos y evaluación de prestaciones*. Cuenca: Ediciones de la Universidad de Castilla-La Mancha.

Ramiro, J. (2016). *esi2.us.es*. Recuperado el 12 de Octubre de 2016, de [esi2.us.es](http://www.esi2.us.es): [http://www.esi2.us.es/~jdedios/asignaturas/Master\\_1.pdf](http://www.esi2.us.es/~jdedios/asignaturas/Master_1.pdf)

raspberrystore. (2016). <http://www.raspberrystore.es/>. Recuperado el 15 de Septiembre de 2016, de <http://www.raspberrystore.es/>: <http://www.raspberrystore.es/>

redgrafica. (2016). <http://redgrafica.com/>. Recuperado el 10 de Septiembre de 2016, de <http://redgrafica.com/>: <http://redgrafica.com/Que-son-los-graficos-vectoriales>

Rodríguez, A. (18 de Agosto de 2015). <https://osl.ull.es>. Recuperado el 15 de Septiembre de 2016, de <https://osl.ull.es>: <https://osl.ull.es/software-libre/opencv-libreria-vision-computador/>

Rodríguez, A. (2016). *wixsite.com*. Recuperado el 8 de Octubre de 2016, de [wixsite.com](http://andresrod97.wixsite.com/photoshopcs5/contact): <http://andresrod97.wixsite.com/photoshopcs5/contact>

Rodríguez, R., & Sossa, J. (2012). *Procesamiento y análisis digital de imágenes*. Madrid: Alfaomega.

Rosebrock, A. (26 de Octubre de 2015). <http://www.pyimagesearch.com/>. Recuperado el 10 de Septiembre de 2016, de <http://www.pyimagesearch.com/>: <http://www.pyimagesearch.com/2015/10/26/how-to-install-opencv-3-on-raspbian-jessie/>

Ruiz, A. (2 de Febrero de 2015). <http://dis.um.es/>. Recuperado el 12 de Septiembre de 2016, de <http://dis.um.es/>: <http://dis.um.es/~alberto/material/percep.pdf>

Sanz, J. (10 de Junio de 2008). <http://www.maia.ub.es/>. Recuperado el 10 de Septiembre de 2016, de <http://www.maia.ub.es/>: <http://www.maia.ub.es/~sergio/linked/julian08.pdf>

Sucar, E., & Gomez, G. (2016). <http://ccc.inaoep.mx/>. Recuperado el 10 de Septiembre de 2016, de <http://ccc.inaoep.mx/>: <http://ccc.inaoep.mx/~esucar/Libros/vision-sucar-gomez.pdf>

Valentine & Moughamian, (2010) Fotomontaje con photoshop cs4. Anaya multimedia.

Valera, B. (30 de Julio de 2015). *http://documents.mx/*. Recuperado el 10 de Septiembre de 2016, de *http://documents.mx/*: *http://documents.mx/documents/sistema-de-vision-por-computadora.html*