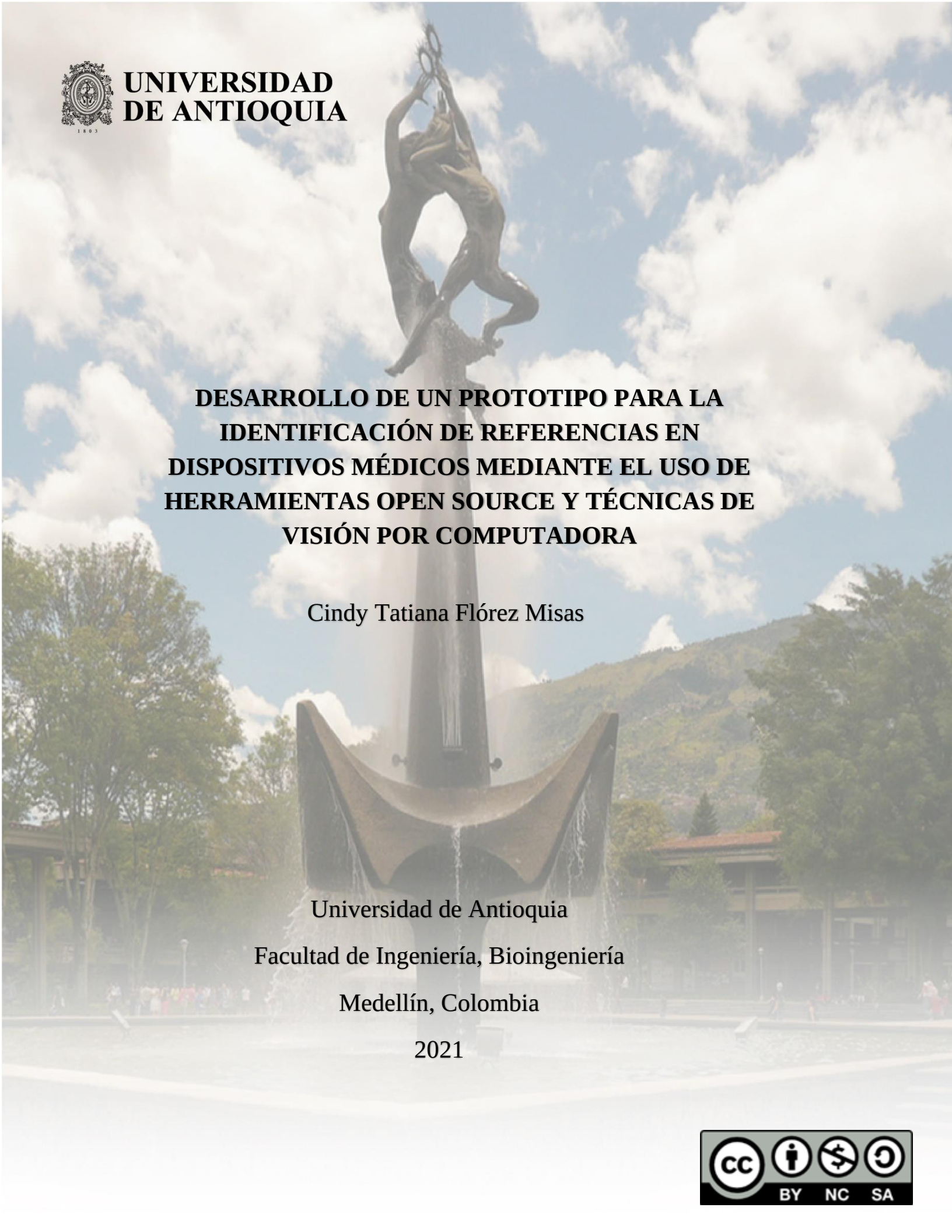


UNIVERSIDAD DE ANTIOQUIA

DESARROLLO DE UN PROTOTIPO PARA LA IDENTIFICACIÓN DE REFERENCIAS EN DISPOSITIVOS MÉDICOS MEDIANTE EL USO DE HERRAMIENTAS OPEN SOURCE Y TÉCNICAS DE VISIÓN POR COMPUTADORA

Cindy Tatiana Flórez Misas

Universidad de Antioquia

Facultad de Ingeniería, Bioingeniería

Medellín, Colombia

2021

DESARROLLO DE UN PROTOTIPO PARA LA IDENTIFICACIÓN DE REFERENCIAS EN DISPOSITIVOS MÉDICOS MEDIANTE EL USO DE HERRAMIENTAS OPEN SOURCE Y TÉCNICAS DE VISIÓN POR COMPUTADORA

Cindy Tatiana Flórez Misas

Trabajo de grado presentado como requisito parcial para optar al título de:

Bioingeniera

Asesores (a):

Aura Cristina Puche Sarmiento, Bioingeniera

Servio Leonel Mera Jiménez, Ingeniero Físico

John Fredy Ochoa Gómez, PhD Ingeniería Electrónica

Diego Alejandro Vélez, Ingeniero Mecánico

Línea de Investigación:

Informática

Universidad de Antioquia

Facultad de Ingeniería, Bioingeniería

Medellín, Colombia

2021

CONTENIDO

LISTA DE TABLAS

LISTA DE FIGURAS

RESUMEN

En la industria es muy común usar etiquetas sobre los productos para mantener un registro y trazabilidad de estos en el tiempo, y en los dispositivos médicos no es la excepción. Los lectores o sensores se encargan de obtener la información de las diferentes codificaciones existentes y con ellos se aumenta la confiabilidad y precisión en la información guardada. Por medio de la visión por computadora, esta lectura y captura de información se puede dar de manera más precisa y por medio de dispositivos alternativos, como los smartphones, permitiendo mayor versatilidad en la mejora de procesos.

En el presente trabajo se planteó el desarrollo de un prototipo para la captura y procesamiento de imágenes por medio de la visión por computadora, con el método de OCR (*Optical Character Recognition*), con software *open source* como Python, OpenCV y Tesseract para el reconocimiento de referencias (etiquetas) de productos de la empresa Industrias Médicas Sampedro. Se estableció un protocolo para la adquisición de imágenes; 20 fotos fueron tomadas por cada referencia, con un total de 100 fotos para probar el algoritmo. Se usaron técnicas de preprocesamiento (binarización, transformación morfológica, filtrado/remoción de ruido, umbralización) para la obtención de información de las palabras extraídas por medio de la librería compatible con Python, Pytesseract. Se evaluó la precisión del algoritmo; es decir, la cantidad de veces que acertó en el reconocimiento de caracteres. Después se realizó una comparación de la información de entrada con una base de datos predefinida. Como resultados en la parte de precisión del algoritmo, éste tuvo fallos en reconocer fotos asociadas a las dos últimas referencias establecidas, lo cual hizo que el prototipo tuviera una precisión del 89%; se realizaron algunas validaciones y correcciones. Al final, la precisión global aumentó al 100%.

Palabras clave: *Etiquetas, OCR, Open Source, Visión por computadora, trazabilidad*

ABSTRACT

In the industry, it is very common to use labels/marking on products to maintain a record and traceability of these over time, and in medical devices it is no exception. The readers or sensors are responsible for obtaining the information from the different existing encodings and with them the reliability and precision of the stored information is increased. Through computer vision, this reading and capturing of information can be given more precisely and through alternative devices, such as smartphones, allowing greater versatility in improving processes.

In the current work, the development of a prototype for the capture and processing of images by means of computer vision was proposed, with the OCR (Optical Character Recognition) method, with opensource software such as Python, OpenCV and Tesseract for the recognition of references (labels) of products from Industrias Médicas Sampedro. A protocol for image acquisition was established; 20 photos were taken for each referral, giving a total of 100 photos to test the algorithm. Preprocessing techniques were used (binarization, morphological transformation, noise filtering / removal, thresholding) for to obtain the information of extracted words through the Python-compatible library, Pytesseract. Evaluation of the algorithm's accuracy was performed; that is, the number of times it hits the character recognition. The input information was then compared with a predefined database. As a result of the precision part of the algorithm, it had failures to recognize photos associated with the last two established references (which made the prototype have an accuracy of 89%); Some validations and corrections were carried out. And at the end, the global precision increased to 100%.

Keywords: Labels/marking, OCR, Open Source, Image acquisition, Traceability

INTRODUCCIÓN

1.1. Planteamiento del Problema

En la actualidad es muy frecuente observar cualquier tipo de producto con un marcado o etiqueta especial, la cual sirve no sólo para identificar el lote, fecha de creación y referencia única de un producto (ej. Información en los códigos de barras), sino también en general para llevar control y trazabilidad sobre éste (Bogotá, 2019). Estas etiquetas son "leídas" de forma automática, normalmente con un lector (dispositivo con un sensor), quién envía la información a una base de datos digital y en donde se almacena el registro de ésta. La gestión y la forma en cómo se guarda la información se da a través de software, y, junto con trabajos en el área de la informática se han generado nuevas tecnologías de codificación para nuevos tipos de etiquetado y herramientas para una lectura más fluida, dinámica y confiable. El uso de esta nueva tecnología ha impactado industrias relacionadas al área de la salud, dentro de la cual es importante un correcto etiquetado de los dispositivos médicos (Videojet, 2016).

Tomando un enfoque en el área de la informática y desarrollo de software, la visión por computadora es un método que se sigue explorando mucho y la cual tiene potencial dentro de la industria: ésta consiste en extraer y analizar la información derivada de imágenes o secuencia de imágenes (Szeliski, 2011). Sus aplicaciones son muchas: van desde el reconocimiento de rostros, hasta la detección de objetos y personas, inclusive en el reconocimiento de patrones impresos. De esto último, el reconocimiento óptico de caracteres (OCR) es una de las herramientas que se desarrollan y aplican dentro de la informática: con el OCR se busca extraer la información deseada de una imagen (normalmente caracteres o símbolos) para objetivos concretos (Pedersen et al., 2016). Aquello es muy útil cuando en la industria se busca detectar etiquetados o referencias asociadas a productos y su correcto registro. Por ejemplo, se suelen estudiar y/o desarrollar

diferentes métodos para optimizar la tarea de procesamiento: estudios previos han propuesto métodos para corregir los ángulos de inclinación que se suelen presentar al tener un documento impreso que no está completamente "plano" (Sawant & Chougule, 2015). El desenfoque de la imagen también es un problema recurrente: otro estudio se enfocó en este problema y desarrollaron un "Método de estimación de desenfoque local" (Kieu et al., 2016). El método trata de hallar la relación entre el desenfoque y el tamaño del carácter obtenido, esto con el objetivo de sacar indicadores y poder clasificar la información como: legible, intermedio, no legible. Por último, en un artículo publicado en la "*International Journal of Computer Science Issues*" (Mollah et al., 2011), establecieron un sistema OCR para la detección de texto integrado en imágenes para dispositivos de mano.

Con base en lo previamente mencionado, en este trabajo se describe la metodología usada en la creación de un prototipo para la identificación de referencias en dispositivos médicos mediante el uso de herramientas *open source* y técnicas de visión por computadora. Para ello se hizo uso de las herramientas/librerías del lenguaje de programación PYTHON y el software TESSERACT, del cual se usó la librería adaptable en Python: pytesseract. Los resultados del prototipo son presentados, y se explican las dificultades durante el procesamiento de las imágenes, factores que afectaron la precisión del algoritmo. Finalmente se exponen algunas recomendaciones para mejorar y conclusiones.

1.2. Objetivos

1.2.1 Objetivo General

Desarrollar un prototipo para el reconocimiento de referencias para dispositivos médicos por medio de herramientas *open source* y técnicas de visión por computadora.

1.2.3 Objetivos Específicos

- Construir una base de datos con las referencias que serán la base de comparación.
- Determinar el esquema de adquisición de imágenes con información de las referencias.
- Diseñar un algoritmo para el reconocimiento de referencias mediante el uso de código *open source*.
- Determinar la precisión del prototipo.

2. MARCO TEÓRICO

2.1 Etiqueta/Etiquetado

El etiquetado es un método de identificación o marcado que se asocia a la identidad de un producto. Las etiquetas suelen tener elementos como la fecha del producto, número de lote, y una referencia o código único (Bogotá, 2019). El código puede ser mediante un serial alfanumérico o más gráficos como código de barras, código QR, DataMatrix, etc.

2.1.1. Serial o referencia

Serial, referencia o número de serie es aquel número o código alfanumérico que se le asigna a un producto, el cual es una identificación única y trazable. Normalmente este código contiene información relacionada a la fecha de fabricación, lote, o descripción del producto (DELSOL, 2021).

2.1.2. Código de barras

Código gráfico, el cual consiste en unas líneas o “barras” negras y blancas de diferentes anchos y espacios, y debajo, unos números asociados (Logyca, 2015) (ver figura 1). Este método de marcado sirve para codificar la información importante asociado al producto y se captura la información por medio de dispositivos lectores (escáner).

Figura 1. Representación código de barras

Por medio de estos códigos también se puede identificar de forma automática:

- Artículos comerciales
- Ubicaciones
- Unidades logísticas
- Documentos
- Activos
- Relaciones entre socios de negocio

2.1.3. Código QR

Código gráfico bidimensional (2D) cuadrado, empleado para almacenar datos o información (ver figura 2). Actualmente es ampliamente empleado en servicios digitales y para almacenar enlaces a sitios web (URL) (Unitag, 2020). Los dispositivos lectores o escáner usados para la lectura de esta codificación son los smartphones principalmente.

Figura 2. Representación código QR

2.1.4. DataMatrix

Es un código gráfico bidimensional (2D) que puede ser cuadrado o rectangular, y su estructura es matricial (KEYENCE, 2021). La versión actual de este código presenta módulos pares (ECC200). Esta versión incluye corrección de errores para reducir problemas de distorsión.

El área de datos del código es rodeada por un marco con forma de L denominado “patrón de alineación” y líneas punteadas denominadas “patrón de reloj” (ver figura 3). Los lectores (ej. Smartphones) capturan estos patrones para determinar la posición del código con procesamiento de imágenes. Por lo tanto, el código DataMatrix se puede leer en cualquier dirección.

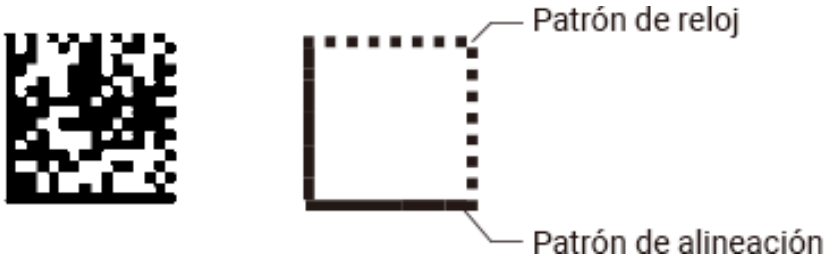


Figura 3. Representación código DataMatrix

2.2 Trazabilidad

Proceso en el cual se busca mantener el historial de un producto que será destinado al consumidor, esto con el fin de mantener un control y, si hay algún problema asociado al producto, poder identificar de manera rápida y oportuna el origen de éste, el fabricante, lote o la fecha de fabricación asociada (DELSOL, 2021).

2.3 Open Source (Código abierto)

Término usado dentro del campo de la informática; hace alusión a código y a veces a software libre, es decir, que es de acceso público y gratuito, o que tiene licencia gratuita. Respecto al código, al ser *open source* significa que el código fuente puede ser modificado por cualquier persona (Perandones, 2019), lo que es muy útil para los programadores y desarrolladores.

2.4 Visión por computadora

Disciplina dentro del campo de la informática la cual busca extraer, analizar y procesar información obtenida de imágenes, incluso de videos (ya que un video es una secuencia de imágenes) (Szeliski, 2011).

2.4.1. Reconocimiento óptico de caracteres (OCR – Optical Character Recognition)

Método informático, el cual busca reconocer y/o extraer información en formato texto (caracteres alfanuméricos) de imágenes (Pedersen et al., 2016).

2.4.2. Validación óptica de caracteres (OCV – Optical Character Validation)

Complementario al OCR; consiste en el uso de métodos para validar la información extraída en OCR, muchas veces mediante símbolos o codificación como códigos de barras, código QR, etc (Pedersen et al., 2016).

2.5 Procesamiento de imágenes

Conjunto de tratamientos que se le hacen a una imagen de entrada; normalmente al obtener y/o capturar información de interés, ésta posee "ruido", u otras características que interfieren en la precisión del procesamiento posterior. Los tratamientos básicos para cualquier preprocesamiento son: escalamiento de imagen (si aplica), binarización, eliminación de ruido mediante filtros digitales, umbralización, entre otros.

2.5.1. Escalamiento de imagen

El escalamiento de imagen hace alusión al redimensionamiento de ésta respecto a su estado original, esto puede ser útil para ahorrar especio en memoria o para eliminar información innecesaria (Filip Zelic, 2021; OpenCV, 2020d).

2.5.2. Binarización

Proceso por el cuál una imagen en escala de colores es reducida a pixeles con dos valores posibles: verdadero y falso, o, en términos digitales 0 (negro) y 255 (blanco). Este cambio facilita el procesamiento y análisis de la imagen, debido a que muchas funciones o herramientas lo utilizan de esta manera por facilidad de cálculo (Molina et al., 2017).

2.5.3. Eliminación de ruido/filtros digitales

La eliminación de ruido de manera digital es muy útil cuando se requiere un procesamiento más preciso y con mayor capacidad respecto a datos digitales. Para eliminar el ruido de la imagen capturada se puede usar filtros pasa altas (high-pass filters), es decir, la información que posea una frecuencia mayor a cierta frecuencia se "deja pasar", el resto se descarta o se trata de reducir (OpenCV, 2020d). En algunos casos, para realizar un filtro digital se debe primero definir un Kernel (núcleo); matemáticamente el kernel puede ser definido de la siguiente manera:

$$K = \frac{1}{mxn}\begin{bmatrix} 1 & 1 \dots \\ \vdots & 1 \dots \end{bmatrix}$$

Donde *m* y *n* serían la cantidad de filas y columnas, respectivamente. Ésta matriz de unos, puede ser mayor y al ser cuadrada m y n deben valer lo mismo. Por ejemplo, el proceso de filtrado sería el siguiente: si tomamos un kernel de 5x5 significa que, en una región, una "ventana" de 5x5 se centra en ésta, todos los pixeles dentro de esta ventana se agregan y el resultado es dividido por 25. Este procedimiento se aplica a toda la imagen hasta obtener finalmente la información filtrada.

2.5.3.1. Filtro Gaussiano

Tipo de filtro digital, el cual puede ajustarse según casos específicos. El kernel para este filtro debe ser con coeficientes impares preferiblemente, es decir, tanto las filas como columnas deben ser números enteros positivos e

impares. Este filtro se usa para eliminar el llamado “ruido Gaussiano”, es decir, el que resulta de la radiación electromagnética presente durante la captura de la imagen. En la figura 4 se puede ver una aplicación de este filtro sobre una imagen:

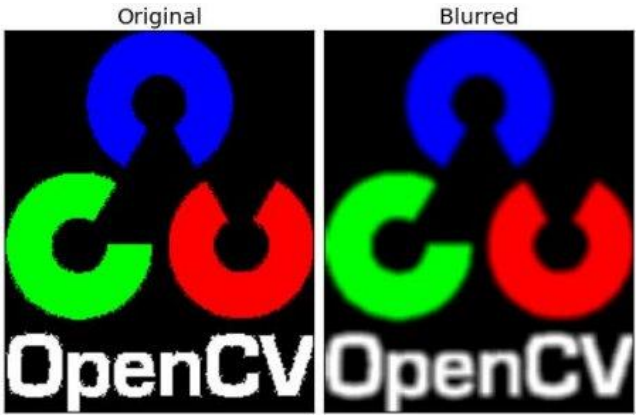


Figura 4. Aplicación filtro digital sobre imagen

Se puede observar en la imagen que se “diluyen” un poco los bordes, esto es debido a las aproximaciones realizadas con un Kernel específico.

2.5.4. Transformaciones Morfológicas

Transformaciones en las cuáles se altera la forma de la imagen y preferiblemente deben estar binarizadas para un mejor desempeño (OpenCV, 2020c). Las operaciones morfológicas básicas son: Erosión (Erosion) y Dilatación (Dilation). De éstos se derivan las operaciones de Apertura (Opening), Cierre (Closing), y Gradiente (Gradient). En la figura 5 se muestra una imagen la cuál servirá de ejemplo para las demás definiciones.

Figura 5. Imagen de prueba

2.5.4.1 Erosión (Erosion)

Con esta transformación, se “erosiona” los bordes de la imagen. Esta erosión o reducción usa de igual forma que los filtros un Kernel, sólo que sin la operación de división: el Kernel se “desliza” a través de la imagen, el píxel de la imagen original (ya sea 1 ó 0) será 1 sólo si todos los pixeles dentro del Kernel son 1, de lo contrario lo “erosiona” (lo convierte en cero). El ejemplo de la aplicación de esta transformación puede verse en la figura 6.

Figura 6. Imagen con Erosión

2.5.4.2 Dilatación (Dilation)

Opuesto al proceso de Erosión. La diferencia es que un píxel será 1 si al menos un píxel dentro del Kernel es 1. Esto genera el incremento en la región blanca o aumenta el tamaño del objeto en primer plano. El ejemplo de la aplicación de la transformación puede verse en la figura 7.

Figura 7. Imagen con Dilatación

2.5.4.3 Apertura (Opening)

Derivado del proceso de Erosión seguido de Dilatación y se puede usar para remover ruido. En la figura 8 se puede ver la aplicación de esta transformación.

Figura 8. Imagen con Apertura

2.5.4.4 Cierre (Closing)

Opuesto al de apertura; Dilatación seguida de Erosión. Esta transformación es usada para cerrar pequeños huecos o espacios dentro de la imagen. La aplicación de esta operación puede verse en la figura 9.

Figura 9. Imagen con Cierre

2.5.4.5 Gradiente Morfológico

Al aplicar esta operación a una imagen creará una diferencia (resta), entre las operaciones Erosión y Dilatación. La imagen mostrará el contorno del objeto en la imagen (ver figura 10).

Figura 10. Imagen con Gradiente.

2.5.5. Umbralización de imágenes (Thresholding)

Relacionado a procesos de segmentación de la imagen. Por medio de ciertos valores límite (de referencia) máximo o mínimo, se delimita una zona específica, separándola de los demás elementos de la imagen ya que se basa en la clasificación a partir del nivel de gris, teniendo en cuenta el valor de referencia (umbral) (OpenCV, 2020a). Se definirán tres métodos de umbralización: umbralización simple, umbralización adaptativa, y la binarización de Otsu.

2.5.5.1 Umbralización Simple

El proceso de umbralización clasifica todos los pixeles de una imagen en dos posibles valores. Si un píxel es mayor que un valor umbral o límite, se le asigna un valor dado (puede ser blanco), de lo contrario se le asigna otro valor (puede ser negro). La imagen debe estar binarizada o en escala de grises para aplicar los diferentes métodos de umbralización.

2.5.5.2 Umbralización Adaptativa

Usada para imágenes que poseen diferentes tipos de iluminación o variación en la escala de grises. Con este método se calcula el umbral para diferentes regiones o porciones de la imagen. Al tener diferentes valores de umbrales intermedios mejora los resultados. En la figura 11 se puede observar dos diferentes tipos de umbralización y cómo varía el resultado respecto a la imagen original: el umbral medio o promedio (Mean Thresh) calcula el umbral promediando las áreas vecinas y el umbral Gaussiano (Gaussian Thresh) se obtiene de la suma ponderada de los valores vecinos por medio de una ventana Gaussiana.

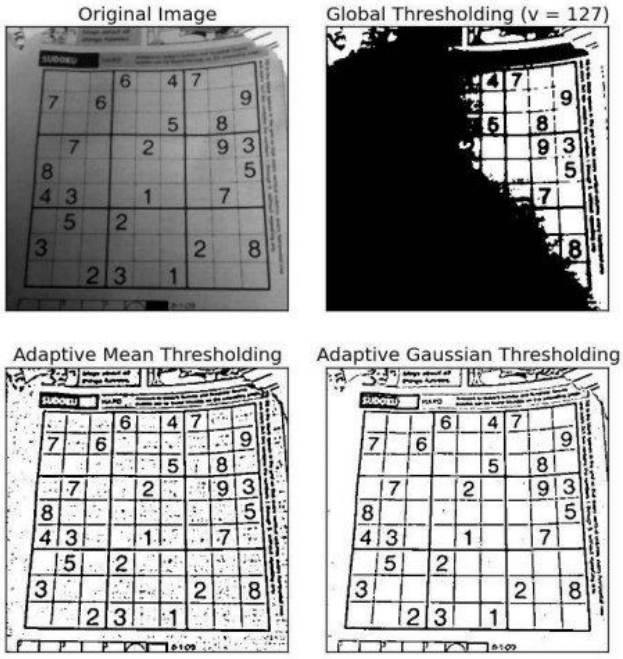


Figura 11. Imagen comparativa de los diferentes métodos de umbralización

2.5.5.3 Binarización de Otsu

Este método calcula el valor umbral usando el histograma de la imagen. El cálculo será lo más preciso posible si la imagen es bimodal (es decir, que el histograma de la imagen tiene dos picos o valores máximos). En la figura 12 se pueden observar las diferencias en la implementación de la umbralización global o simple, y la de Otsu, con la imagen original con ruido y filtrada y sus respectivos histogramas.

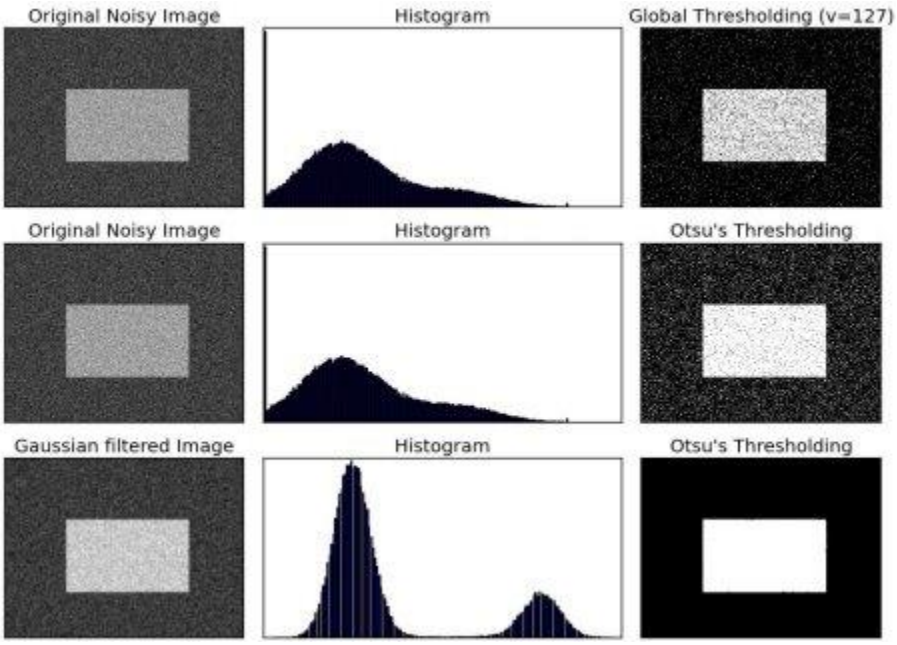


Figura 12. Diferencias entre umbralización global y de Otsu con la imagen sin filtrar y filtrada.

2.6 OpenCV

Biblioteca libre de código abierto especializada en el desarrollo de visión por computadora y machine learning, originalmente desarrollada en Intel en 1999 por Gary Bradsky (OpenCV, 2020b). Sus librerías son muy populares debido a su amplia dispersión o uso en diferentes sistemas operativos (GNU/Linux, Mac OS X, Windows y Android) y soporta diferentes lenguajes de programación como: C++, Java, Python, etc.

2.7 Python

Python es un lenguaje de programación orientado a objetos, con un amplio rango de herramientas y versatilidad. Su estructura sencilla y continuo desarrollo lo han vuelto muy atractivo tanto en las esferas académicas como las industriales (Chris, 2019; OpenCV, 2020b). Fue desarrollado inicialmente por Guido van Rossum y al ser un lenguaje de

programación libre y adaptable (como OpenCV), posee una amplia variedad de librerías y herramientas, y una basta documentación, lo cual lo hace muy asequible a las personas.

2.7.1. Numpy

Librería de Python, altamente optimizada para cálculos numéricos y manejo/operaciones con vectores y matrices (arreglos). Los vectores y matrices estructurados en OpenCV son compatibles con Numpy, por lo que la comunicación y uso de las librerías se hace más sencillo.

2.7.2. OpenCV-Python

Librería de OpenCV compatible con Python; esta librería posee la mayoría de las funciones disponibles en OpenCV y sirve para el desarrollo de algoritmos basados en la visión por computadora, usando el lenguaje versátil de Python.

2.7.3. Pandas

Una de las bibliotecas básicas más importantes de Python; desde análisis y operaciones estadísticas de datos y manipulación (Mokhtar, 2019). Lo más importante de Pandas son los Dataframe; un tipo de estructura matricial donde los datos son almacenados y de donde se pueden realizar diferentes tipos de cálculos y/o manipulación, dependiendo del tipo de dato.

2.8 Tesseract OCR

Tesseract es un motor open source para reconocimiento óptico de texto (OCR), soporta una amplia variedad de lenguajes y puede ser usado para reconocer texto en documentos largos, o usarse junto con un detector de caracteres externo para extraer texto de imágenes o una simple línea de palabras (Filip Zelic, 2021). En la figura 13 se esquematiza el proceso de manera general, en cuanto al cálculo o extracción del texto en la imagen original, y al final del procesamiento se genera un documento en formato editable. Esto es útil para las personas que buscan digitalizar sus documentos físicos.

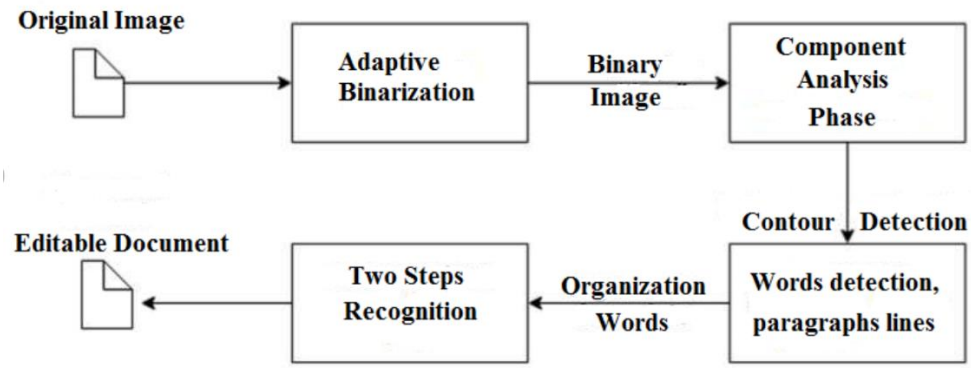


Figura 13. Diagrama de flujo del proceso de reconocimiento de caracteres/texto.

Tesseract también puede:

- Encontrar palabras específicas en una imagen
- Encontrar o rastrear líneas específicas de texto
- Clasificar caracteres

2.8.1. Pytesseract

Librería de Tesseract compatible con Python. En ella están la mayoría de las funciones básicas de reconocimiento de texto.

2.9 Microsoft Excel

Excel es un programa de hojas de cálculo desarrollado por Windows, para MacOS, Android e iOS. Es ampliamente usado en la industria ya que posee herramientas de cálculo, de gráficas y un lenguaje de programación llamado Visual Basic. Debido a su integración con Visual Basic pueden programarse estas hojas para diferentes actividades, incluyendo su uso como Base de Datos; almacenando y gestionando la información (Microsoft, 2021).

3. METODOLOGÍA

A continuación, se define el procedimiento empleado en el proyecto. Para el procesamiento de las imágenes se usó el lenguaje de programación Python 3.8, integrado en el software/entorno de desarrollo Anaconda. Como librerías empleadas se usaron las librerías externas OpenCV-Python y PyTesseract. Como base de datos se empleó el programa Excel.

3.1 Desarrollo de la base de datos

Para el desarrollo de la base de datos se tomó la documentación otorgada por la empresa, de la cual se extrae la información de interés, teniendo cuidado con la información confidencial.

3.1.1. Hoja de Remisión

El tipo de codificación que es obtenida de las imágenes fue de tipo serial/referencia. La hoja de datos de la figura 14 es otorgada por Industrias Médicas Sampedro y de ella fueron seleccionadas las referencias para la realización del proyecto.

CÓDIGO	DESCRIPCIÓN	CANT.
SAF	SISTEMA AFFINITY RADIO DISTAL	1
SPRDAF/012	SET PLACA RD AFFINITY + TORN. / CON |CON	1
PRDEE2SD	PLACA RADIO DISTAL EXTRAARTICULAR ESTRECHA 2S DER. |AF109093X1	1
PRDEE2SI	PLACA RADIO DISTAL EXTRAARTICULAR ESTRECHA 2S IZQ. |AF106738X1	1
PRDEI2SD	PLACA RADIO DISTAL EXTRAARTICULAR INTERMEDIA 2S DER. |AF104113_1	1
PRDEI2SI	PLACA RADIO DISTAL EXTRAARTICULAR INTERMEDIA 2S IZQ. |AF108818X2	1
PLBCRD4SD-	PLACA EN L BORDE CUBITAL Y RADIAL DISTAL 4S DER.- |AF85260_1	1
PLBCRD4SI-	PLACA EN L BORDE CUBITAL Y RADIAL DISTAL 4S IZQ.- |AF95351_1R1	1
PLBCRD6SD-	PLACA EN L BORDE CUBITAL Y RADIAL DISTAL 6S DER.- |AF82730_1	1
PLBCRD6SI-	PLACA EN L BORDE CUBITAL Y RADIAL DISTAL 6S IZQ.- |AF58943_1	1
PRDEA3SD-	PLACA RADIO DISTAL EXTRAARTICULAR ANCHA 3S DERECHA- |AF109266X1	1
PRDEA3SI-	PLACA RADIO DISTAL EXTRAARTICULAR ANCHA 3S IZQUIERDA- |AF107955X1	1
PRDEE3SD-	PLACA RADIO DISTAL EXTRAARTICULAR ESTRECHA 3S DERECHA- |AF104700X1	1
PRDEE3SI-	PLACA RADIO DISTAL EXTRAARTICULAR ESTRECHA 3S IZQUIERDA- |AF107956X1	1
PRDEI3SD-	PLACA RADIO DISTAL EXTRAARTICULAR INTERMEDIA 3S DERECHA- |AF107260X1	1
PRDEI3SI-	PLACA RADIO DISTAL EXTRAARTICULAR INTERMEDIA 3S IZQUIERDA- |AF107571X1	1
PRDEI7SD-	PLACA RADIO DISTAL EXTRAARTICULAR INTERMEDIA 7S DERECHA- |AF103027_1	1
PRDEI7SI-	PLACA RADIO DISTAL EXTRAARTICULAR INTERMEDIA 7S IZQUIERDA- |AF103479_1R1	1
PRER4S-	PLACA RECTA ESTILOIDES DE RADIO 4S- |AF95090_1	1

Figura 14. Fragmento hoja de datos con referencias producto Sampedro.

3.1.2. Base de Datos

Para la base de datos se escogieron de manera aleatoria cinco (5) referencias, junto con su descripción y cantidad inicial. En la tabla 1 se encuentra la pequeña base de datos realizada con las muestras aleatorias en Excel. El prototipo propuesto, una vez reconocido las referencias en las fotos/imágenes, accede a la base de datos y realiza la búsqueda y comparación. Se eligen sólo cinco referencias por simplicidad, y para evaluar la precisión del algoritmo.

Tabla 1. Base de datos de las referencias de producto Sampedro.

Referencia	**Descripción**	**Cantidad**
PRDEE2SD	PLACA RADIO DISTAL EXTRAARTICULAR ESTRECHA 2S DER.	1
PLBCRD4SD-	PLACA EN L BORDE CUBITAL Y RADIAL DISTAL 4S DER.	1
PRDEA3SD-	PLACA RADIO DISTAL EXTRAARTICULAR ANCHA 3S DERECHA-	1
TC2510RT	TORN. CORTICAL 2.5 X 10MM RT	1
TCC2520RP	TORN. CORTICAL PENTALOCK CANULADO 2.5 X 20 MM RP	1

3.2. Adquisición de imágenes

Debido a que el algoritmo requiere de imágenes como datos de entrada, la adquisición u obtención de estas se hizo por medio de fotografías tomadas por un celular comercial (Xiaomi Redmi Note 9). Las imágenes "crudas", es decir, sin aún haber pasado por preprocesamiento fueron almacenadas de manera local en un computador para su posterior tratamiento. Los formatos de imagen aceptados son de tipo .PNG, .JPG, .JPEG, o .TIFF.

Nota. Las imágenes contienen las referencias de interés almacenadas en la base de datos creada.

3.2.1. Protocolo de captura de imágenes

Con el fin de garantizar una buena calidad en las imágenes para un mejor procesamiento, se plantea el siguiente protocolo:

- El área en donde se toma la foto debe tener una iluminación uniforme (sin cambios de intensidad, mezcla de luces de otros colores, sombras), y

preferiblemente luz blanca, similar a la usada en el interior de los hospitales.

- El plano de la cámara del celular debe estar, en la medida de lo posible, de manera paralela al plano de la imagen a capturar (evitar angulación, ya que puede afectar el procesamiento). La imagen debe estar sobre un sitio plano.
- Las fotos como son tomadas de una lista con diferentes referencias, es imperativo el enfoque de la cámara lo suficiente para que sólo tome una de las referencias (código serial/ código de barras), o preferiblemente que ocupe por lo menos un 70% de la imagen completa para luego hacer recorte o redimensionamiento (ver figura 15).
- Para evaluación y validación de la imagen procesada, se toma por lo menos veinte (20) fotos de cada referencia. Debido a que son cinco (5) referencias diferentes, se obtienen cien (100) imágenes en total para procesamiento.

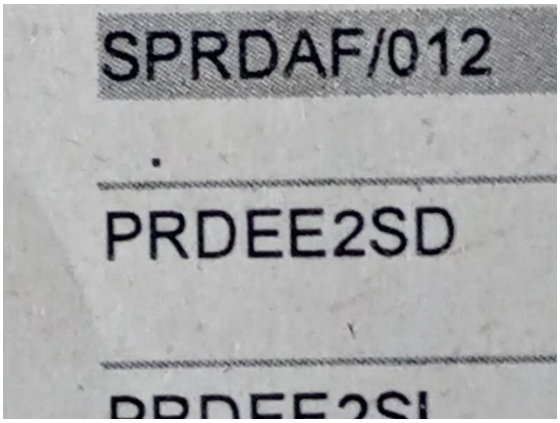
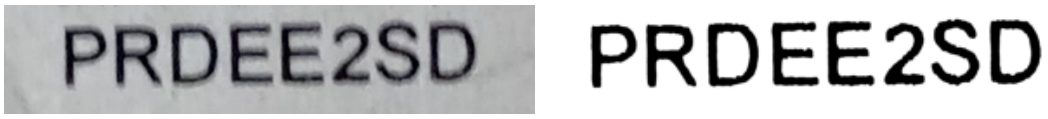


Figura 15. Captura imagen de referencia PRDEE2SD.

3.3. Preprocesamiento

Según el protocolo planteado se tomó cada foto con características similares a la de la figura 15 (realizando un zoom o acercamiento) y se recorta de manera manual usando el

lector de imágenes de Windows 10, para que quede sólo la referencia deseada. Luego se binarizaron las imágenes usando la librería de OpenCV-Python (todo el preprocesamiento se hizo con esto). El filtro usado para eliminar ruido fue de tipo Gaussiano. Posteriormente se realiza transformación morfológica con la función "Closing" y finalmente se implementa el "Thresholding" con la binarización de Otsu. En la figura 16 se puede observar como quedó la imagen antes y después de realizado el preprocesamiento.



(a) **(b)**

Figura 16. Imagen recortada: a). Antes de preprocesamiento, b). Después de preprocesamiento.

3.4. Selección de método(s) de segmentación y caracterización

Después de realizada la respectiva búsqueda bibliográfica, se optó por usar junto con Python, la parte de segmentación por medio de la librería Pytesseract; las funciones otorgadas por la librería realizan de manera interna la extracción de las características representada en los pixeles, al igual que el histograma de la imagen para, posteriormente encerrar cada caracter (o toda una palabra) en "recuadros" con el objetivo de validar visualmente qué información fue reconocida. En la figura 17 se puede ver cómo por medio de las funciones facilitadas por Pytesseract, éstas detectan o reconocen cada carácter presente en la imagen.

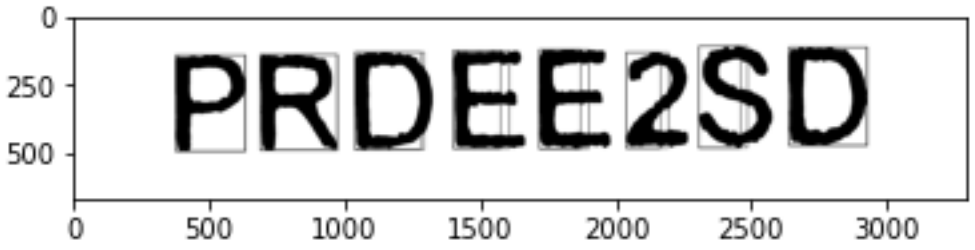


Figura 17. Reconocimiento de caracteres.

También la librería cuenta con la opción para reconocimiento de palabras o texto completo. En este caso, como fue sólo por cada referencia entonces simplemente encierra en un recuadro la única presente. En la figura 18 se visualiza como queda.

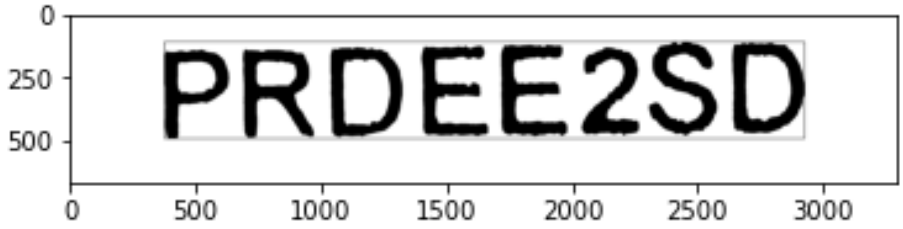


Figura 18. Reconocimiento de palabras.

3.5. Realización de pruebas de precisión al prototipo propuesto

Como se mencionó en la subsección “3.2.1. Protocolo de captura de imágenes”, se tomaron 20 fotos por cada referencia y se probó el algoritmo con cada una de ellas para evaluar la precisión de este, por medio de porcentaje de acierto local y después global; es decir, se evaluó el porcentaje de acierto por cada referencia y al final se halló el porcentaje de precisión total.

3.6. Registro y almacenamiento de referencias

De las imágenes procesadas se extrae las referencias ya en formato digital y la información es guardada en una nueva base de datos, con las mismas características que la original por medio de la librería Pandas. Durante el proceso el algoritmo busca en la base de datos original si cada palabra coincide (es decir, verifica que la referencia exista), si es afirmativo entonces en la nueva base de datos se guarda esta palabra y la cantidad respectiva. Si la siguiente referencia es igual entonces la cantidad del producto debe aumentar (ya que las referencias hacen alusión a productos reales pertenecientes a Industrias Médicas Sampedro), de lo contrario la guarda en el siguiente campo vacío. La idea es que simule un caso de registro real y la información de interés pueda ser guardada, basada en los productos existentes. Como forma simplificada de resumir todas las fases del algoritmo, en la figura 19 se encuentra el diagrama de flujo asociado y las librerías importantes utilizadas.

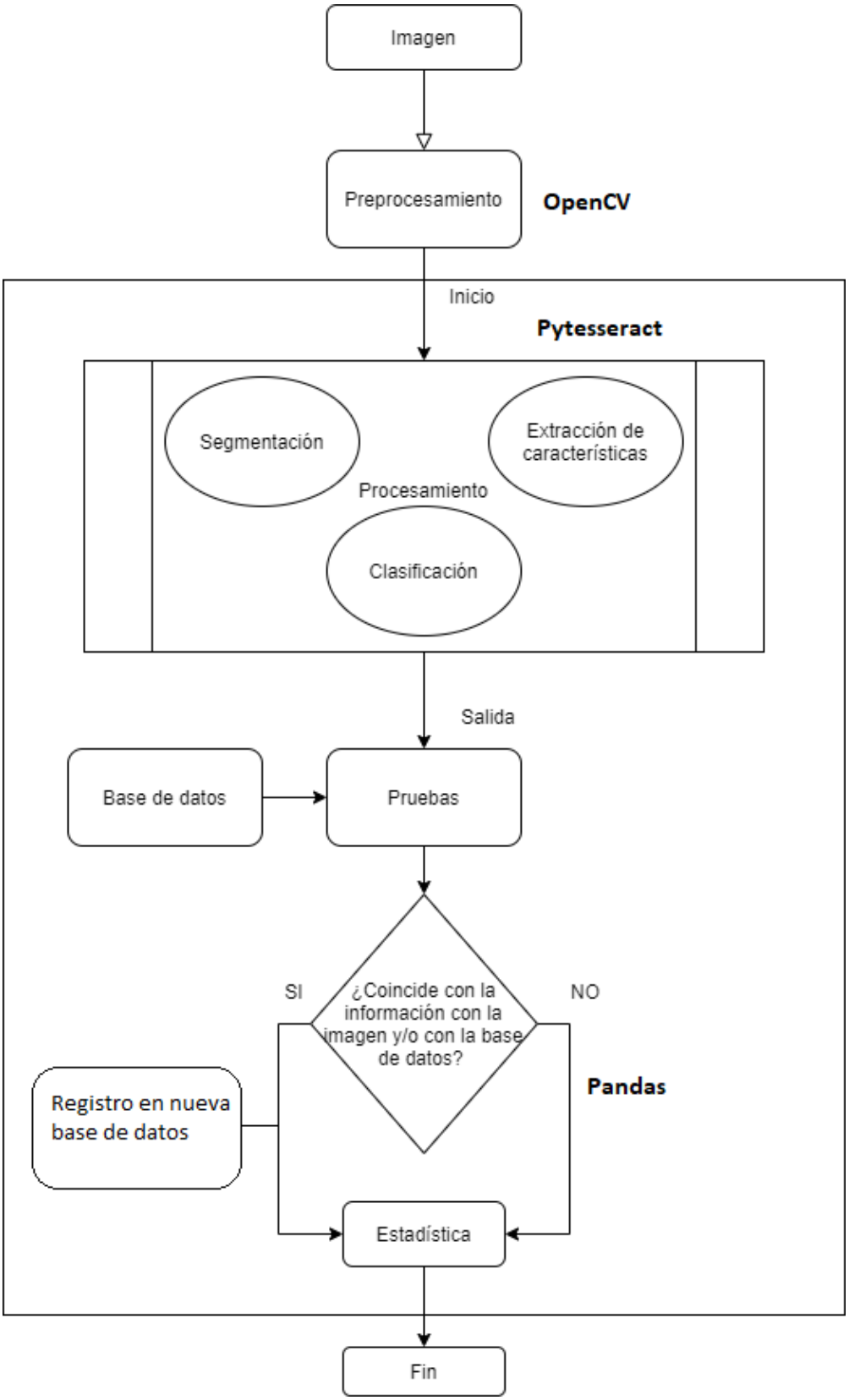


Figura 19. Diagrama de flujo prototipo OCR y registro.

4. RESULTADOS Y ANÁLISIS

4.1 Realización de pruebas de precisión al prototipo propuesto

Según los objetivos planteados para la realización de este trabajo, la parte más importante del mismo fue la de evaluar la precisión del prototipo realizado. En la tabla 2 se resume los porcentajes de precisión para cada referencia y el valor final de toda la evaluación.

Tabla 2. Resultados precisión del algoritmo.

Referencia	**Descripción**	**% Aciertos**
PRDEE2SD	PLACA RADIO DISTAL EXTRAARTICULAR ESTRECHA 2S DER.	100
PLBCRD4SD-	PLACA EN L BORDE CUBITAL Y RADIAL DISTAL 4S DER.	100
PRDEA3SD-	PLACA RADIO DISTAL EXTRAARTICULAR ANCHA 3S DERECHA-	100
TC2510RT	TORN. CORTICAL 2.5 X 10MM RT	95
TCC2520RP	TORN. CORTICAL PENTALOCK CANULADO 2.5 X 20 MM RP	50
Total % Aciertos		89

La información de las dos últimas referencias presentó algunos errores como: caracteres erróneos, ruido, vacío, referencia fragmentada o referencia con ruido. En este caso el ruido se asocia a caracteres no pertenecientes o especiales (@, %, “, /, &, etc.). Debido a ello se decide volver a realizar la toma de las cien (100) imágenes teniendo un mayor cuidado al momento de hacer la adquisición, debido a que, al revisar las fotos estas presentaron problemas de angulación y la intensidad de luz varió ligeramente en algunas (ver figuras 20 y 21). También fue identificado que, al momento de realizar las capturas de

las fotos, se hizo de manera muy rápida, por lo que la cámara no tuvo el tiempo suficiente para enfocar adecuadamente.

Figura 20. Fotos que presentaron errores en el procesamiento.

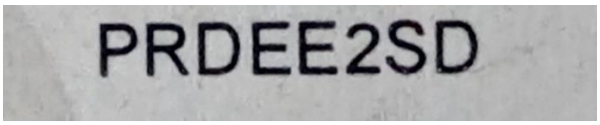


Figura 21. Foto reconocida exitosamente.

Una vez realizada la segunda ronda de fotos con las consideraciones realizadas, la precisión del algoritmo fue del 100%.

4.2. Registro y almacenamiento de referencias

Una vez validado el algoritmo de reconocimiento de texto, se procedió a realizar ensayos escogiendo referencias al azar y cantidades al azar. Por medio de otra librería de Python para manejar directorios (librería OS) se escogió diferentes fotos al azar y se ingresaron al código. Como se puede observar en la figura 22, se almacenó de manera correcta las referencias y cantidades de prueba en el Dataframe creado en Pandas.

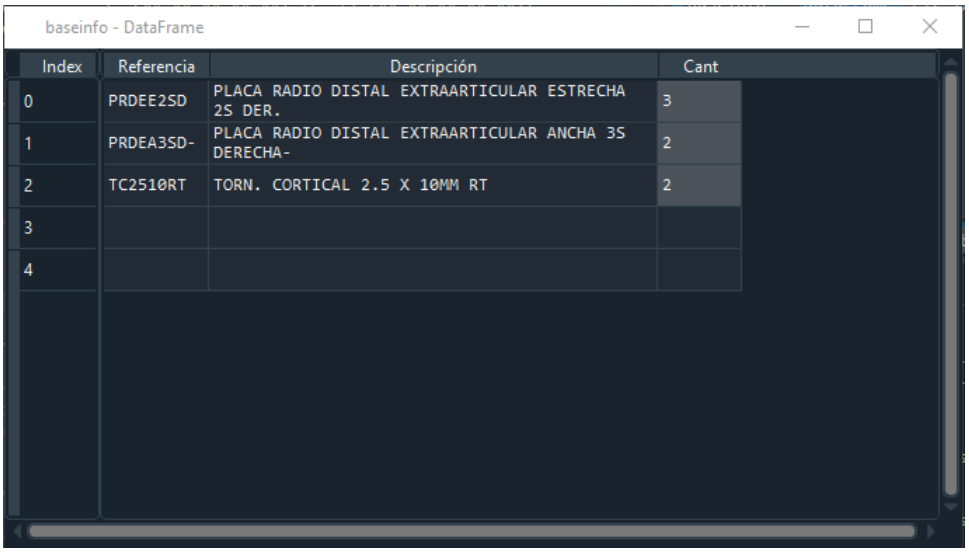


Figura 22. Ensayo base de datos con referencias aleatorias.

Esta parte del algoritmo fue validado para que las referencias ingresaran en desorden e igual hiciera el conteo de manera correcta.

5. CONCLUSIONES Y RECOMENDACIONES

El prototipo propuesto presentó una alta fiabilidad en su función de reconocer los caracteres presentes en las fotos/imágenes que se le enviaba, incluso si en el primer intento hubo errores. El objetivo del protocolo de adquisición de imágenes es precisamente el reducir los factores que pueden ocasionar malas lecturas y durante la primera ronda de capturas no hubo tanta rigurosidad y tampoco no se había considerado el factor de tiempo de captura entre fotos. Al realizar una mejor captura de las imágenes se logró una precisión del 100% en contraste del 89% inicial. La parte crítica en este trabajo fue la adquisición de las imágenes; la calidad de la cámara del celular también es importante para lograr una imagen y qué tanto zoom se le realiza la imagen, ya que, si la cámara no es buena, y a la imagen se le hace mucho acercamiento, la calidad de la imagen disminuye considerablemente, y para este caso se recomendaría, en la medida de lo posible, no realizar mucho aumento con cámaras de celulares de gama baja.

El prototipo en conjunto es altamente escalable y puede ser integrado con otros software o aplicaciones ya existentes; respecto a la escabilidad, es decir, su capacidad para ampliarse y adaptarse, se pueden agregar más funciones de las librerías aplicadas para mejorar su respuesta ante diferentes situaciones, ya que en la vida real las condiciones van a ser más variables. Para este proyecto hubo un mayor control de las variables lumínicas y de posicionamiento de la cámara, pero en una situación cotidiana puede que no sea tan sencillo, y ahí el protocolo planteado puede ser una ayuda, debido a su simplicidad. Como consideraciones a futuro, se puede seguir mejorando el prototipo, con énfasis en la adquisición de las imágenes con aplicaciones para captura de fotos, como diseñar un rectángulo o marco delimitador que, al tomar la foto, sólo guarde lo que se encuentra dentro de ésta geometría, lo que ayudaría a reducir errores de angulación o de interferencia con otra información que no es de interés. También, conforme vaya mejorando el algoritmo se puede realizar un escalamiento a realizar captura y procesamiento en tiempo real, ya que, por los limitantes de este proyecto, sólo pudo ser realizado de manera estática.

6. REFERENCIAS

Bogotá, C. de comercio de. (2019). *Guía Práctica: Sistema de Empaque, Envase, Embalaje y Etiquetas*. Cámara de Comercio de Bogotá. https://bibliotecadigital.ccb.org.co/bitstream/handle/11520/14382/Guía Práctica Sistema de Empaque Envase Embalaje y Etiqueta para una Exportación %28002%29.pdf?sequence=5&isAllowed=y

Chris, M. (2019). *BeginnersGuide/Overview - Python Wiki*. https://wiki.python.org/moin/BeginnersGuide/Overview

DELSOL. (2021). *Diferencias entre Número de Serie y Lote*. https://www.sdelsol.com/glosario/numeros-de-serie-y-lote/

Filip Zelic, S. A. (2021). *[Tutorial] OCR in Python with Tesseract, OpenCV and Pytesseract*. https://nanonets.com/blog/ocr-with-tesseract/#preprocessingfortesseract

KEYENCE. (2021). *¿Qué es el código Data Matrix? | Conceptos básicos de los códigos 2D ╱ Información y consejos sobre códigos de barras y códigos 2D*. https://www.keyence.com.mx/ss/products/auto_id/barcode_lecture/basic_2d/datamatrix/

Kieu, V., Cloppet, F., & Vincent, N. (2016). OCR Accuracy Prediction Method Based on Blur Estimation. *2016 12th IAPR Workshop on Document Analysis Systems (DAS)*, 317–322.

Logyca. (2015). *¿Qué es el Código de Barras?* http://blog.logyca.com/noticias/que-es-el-codigo-de-barras/

Microsoft. (2021). *Software de hojas de cálculo Microsoft Excel | Microsoft 365*. https://www.microsoft.com/es-co/microsoft-365/excel?rtc=1

Mokhtar, E. (2019). *Tutorial de Python pandas: Iniciando con DataFrames - Like Geeks*. https://likegeeks.com/es/tutorial-de-python-pandas/#Leer-un-archivo-de-Excel

Molina, E., Diaz, J., Hidalgo-Silva, H., & Chávez, E. (2017). Algoritmos de Binarización Robusta de Imágenes con Iluminación No Uniforme. *Revista Iberoamericana de Automática e Informática Industrial*, *15*. https://doi.org/10.4995/riai.2017.8847

Mollah, A., Majumder, N., Basu, S., & Nasipuri, M. (2011). Design of an Optical Character Recognition System for Camera-based Handheld Devices. *International Journal of Computer Science Issues*, *8*.

OpenCV. (2020a). *Image Thresholding — OpenCV-Python Tutorials 1 documentation*. https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_thresholding/py_thresholding.html#thresholding

OpenCV. (2020b). *Introduction to OpenCV-Python Tutorials — OpenCV-Python Tutorials 1 documentation*. https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_setup/py_intro/py_intro.html#intro

OpenCV. (2020c). *Morphological Transformations — OpenCV-Python Tutorials 1 documentation*. https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_morphological_ops/py_morphological_ops.html#morphological-ops

OpenCV. (2020d). *Smoothing Images — OpenCV-Python Tutorials 1 documentation*. https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_filtering/py_filtering.html#filtering

Pedersen, J. B., Nasrollahi, K., & Moeslund, T. B. (2016). Quality inspection of printed texts. *International Conference on Systems, Signals, and Image Processing*, *2016-June*, 6–9. https://doi.org/10.1109/IWSSIP.2016.7502718

Perandones, S. M. (2019). *¿Qué es el open source y cómo puede ayudarte? - OpenExpo Europe 2020*. https://openexpoeurope.com/es/open-source-puede-ayudarte/

Sawant, A., & Chougule, D. (2015). *Script independent text pre-processing and segmentation for OCR*. https://doi.org/10.1109/EESCO.2015.7253643

Szeliski, R. (2011). *Computer vision : algorithms and applications*. Springer. http://cataleg.upc.edu/record=b1387199~S1*cat

Unitag. (2020). *¿Qué es un código QR?* https://www.unitag.io/es/qrcode/what-is-a-qrcode

Videojet. (2016). *Dispositivos médicos y productos farmacéuticos*. https://global.videojet.com/wp-content/uploads/dam/pdf/Spain - Spanish/Brochure/br-pharmaceutical-and-medical-devices-es.pdf

ANEXOS

N/A