UNIVERSIDAD NACIONAL DE SAN ANTONIO ABAD DEL CUSCO

FACULTAD DE INGENIERÍA ELÉCTRICA, ELECTRÓNICA, INFORMÁTICA, Y MECÁNICA.

ESCUELA PROFESIONAL DE INGENÍERIA ELECTRÓNICA

TESIS

"DISEÑO DE UN SISTEMA DE CAPTACIÓN DE PARÁMETROS NUMÉRICOS DE MONITORES MEDIANTE PROCESAMIENTO DIGITAL DE IMÁGENES"

PRESENTADO POR:
Br. HAROLD PABEL CRUZ TTITO

PARA OPTAR EL TÍTULO PROFESIONAL DE INGENIERO ELECTRÓNICO

ASESOR:
Ing. LUIS JIMÉNEZ TRONCOSO

CUSCO – PERÚ
2019

INFORME DE ORIGINALIDAD

(Aprobado por Resolución Nro.CU-303-2020-UNSAAC)

El que suscribe, **Asesor** del trabajo de investigación/tesis titulada: Diseño de un sistema de captacion de parámetros numéricos de monitores mediante procesamiento digital de imágenes

presentado por: Harold Pabel Cruz Tito con DNI Nro.: 72948847

presentado por: ..... con DNI Nro.: .....

para optar el título profesional/grado académico de Ingeniero Electrónico

.....

Informo que el trabajo de investigación ha sido sometido a revisión por 4 veces, mediante el Software Antiplagio, conforme al Art. 6° del ***Reglamento para Uso de Sistema Antiplagio de la UNSAAC*** y de la evaluación de originalidad se tiene un porcentaje de 7 %.

Evaluación y acciones del reporte de coincidencia para trabajos de investigación conducentes a grado académico o título profesional, tesis

Porcentaje	Evaluación y Acciones	Marque con una (X)
Del 1 al 10%	No se considera plagio.	X
Del 11 al 30 %	Devolver al usuario para las correcciones.	
Mayor a 31%	El responsable de la revisión del documento emite un informe al inmediato jerárquico, quien a su vez eleva el informe a la autoridad académica para que tome las acciones correspondientes. Sin perjuicio de las sanciones administrativas que correspondan de acuerdo a Ley.	

Por tanto, en mi condición de asesor, firmo el presente informe en señal de conformidad y **adjunto** la primera página del reporte del Sistema Antiplagio.

Cusco, 13 de diciembre de 2023

.....

Firma

Post firma... Luis Jiménez Troncoso

Nro. de DNI... 08275751

ORCID del Asesor... 0000-0001-6414-9742

Se adjunta:

1. Reporte generado por el Sistema Antiplagio.
2. Enlace del Reporte Generado por el Sistema Antiplagio: oid: 27259:295333201

NOMBRE DEL TRABAJO
tesis_HaroldPabel_Cruz_Ttito_correccion_final_12_12.pdf

AUTOR
Harold Pabel Cruz Ttito

RECUENTO DE PALABRAS
40953 Words

RECUENTO DE CARACTERES
209489 Characters

RECUENTO DE PÁGINAS
207 Pages

TAMAÑO DEL ARCHIVO
3.6MB

FECHA DE ENTREGA
Dec 13, 2023 10:22 AM GMT-5

FECHA DEL INFORME
Dec 13, 2023 10:28 AM GMT-5

7% de similitud general

El total combinado de todas las coincidencias, incluidas las fuentes superpuestas, para cada base d

- 7% Base de datos de Internet
- 3% Base de datos de publicaciones
- Base de datos de Crossref
- Base de datos de contenido publicado de Crossr

Excluir del Reporte de Similitud

- Base de datos de trabajos entregados
- Material bibliográfico
- Material citado
- Material citado

DEDICATORIA

Dedico este trabajo a mis padres Juana Ttito Pauccar y Faustino Cruz Quispe, por su incondicional apoyo en mi vida y la consecuente realización de este trabajo. Está dedicado al empeño que pusieron en formarme como persona y al gran esfuerzo que significó acompañarme en esta difícil travesía.

AGRADECIMIENTOS

Me gustaría expresar mi profundo agradecimiento a todos los que me han apoyado en este trabajo, así como aquellos que me guiaron en su desarrollo, en especial al Ingeniero Luis Jiménez Troncoso, mi asesor, quien encamino y me brindo su conocimiento y su tiempo en el buen desarrollo del presente trabajo.

También me gustaría agradecer a mi familia por su apoyo, a la Dra. Tatiana del Castillo por todos los consejos brindados desde el principio de este trabajo, a mis profesores de la universidad San Antonio Abad por transmitir el afán investigativo y el conocimiento necesario para poder desarrollar este trabajo, también me gustaría agradecer a mis amigos y colegas que me apoyaron con mensajes de motivación y optimismo hasta la culminación del mismo.

RESUMEN

El presente trabajo propone un sistema de detección de parámetros numéricos en imágenes obtenidas a través de cámaras en diferentes tipos de monitores, guardando los datos obtenidos en archivos "*.xls ".

El sistema propuesto consta de tres etapas, la primera etapa consta de la generación de la matriz de predicción del sistema, la segunda la etapa de reconocimiento y, por último, la etapa de reconstrucción.

La etapa de la generación de la matriz de predicción consiste en el tratamiento de las imágenes; para posteriormente ser procesadas: binarizarlas y segmentarlas; extraerles características comunes entre caracteres iguales, y características particulares entre caracteres diferentes; y por último guardarlas en una base de datos que será usada en la siguiente etapa.

La etapa de reconocimiento consta de tres partes: La primera es la de captación de la imagen y su lectura en el software empleado, en este caso MATLAB, la segunda también es un procesado de la imagen; selección de la región, recorte, binarización y segmentación. Y por último la extracción de sus características para luego ser comparadas con la base de datos y obtener el carácter reconocido.

La etapa de reconstrucción consta de la reorganización de los caracteres reconocidos en información útil, adquisición de las variables como el nombre y las unidades, y su posterior almacenamiento en una base de datos.

El sistema tiene alto grado de precisión y con un alto grado de adaptabilidad hacia otros tipos de letras (fuentes), lo que lo hace útil para futuros proyectos que requieran del reconocimiento de los parámetros numéricos en monitores.

Palabras clave: Reconocimiento de parámetros numéricos, ORC, cámaras, monitor de signos vitales, método K-NN.

ABSTRACT

The present work presents a detection system of numerical parameters in images obtained through cameras for any types of monitors, saving the obtained data in files "* .xls".

The proposed system consists of three stages, the first one consists in the creation of the prediction matrix of the system, the second is the recognition and the final stage is reconstruction.

The creation of the prediction matrix consists of the treatment of the images, to be later processed: binarize and segment them, to extract common characteristics between similar characters, and some particular between different characters, and finally save them to a database that will be used in the next stage.

The recognition stage consists of three parts: the first one, consist in capture the image and reading it in the software used, in this case MATLAB; the second is also an image processing: region selection, cropping, binarization and segmentation. And finally, the extraction of characteristics and then be compared with the database, and get the recognized character.

The reconstruction stage consists in the reorganization of the characters recognized in useful information, acquisition of the variables such as the name and the units, and their later storage in the database.

The system has a high degree of accuracy and a high degree of adaptability to other types of letters (fonts), which makes it useful for future projects that require the recognition of numerical parameters in monitors.

Key words: Numerical parameters recognition, ORC, cameras, vital signs monitor, K-NN method.

INTRODUCCIÓN

En la era actual de los avances tecnológicos, el procesamiento de imágenes se ha convertido en una herramienta esencial para el análisis computarizado de diversos procesos, ya sean biológicos o mecánicos. Esta herramienta desarrolló nuevos motores de análisis como lo es el reconocimiento óptico de caracteres, que está estrictamente ligado a reconocer símbolos pertenecientes al abecedario y los números fundamentales, esto lo convierte en una herramienta muy versátil aplicable en innumerables ramas de la ciencia, que requiera de la conversión de una imagen que contenga caracteres, en un archivo de datos modificable.

En diferentes áreas de la ciencia están presentes aparatos de medición los cuales muestran los valores medidos en pantallas de todo tipo ya sean monitores de alta definición hasta pequeños visualizadores que usan displays de 7 segmentos. Muchos de estos aparatos solo poseen la capacidad de medir los datos más no de almacenarlos y menos de conectarse con otros de su misma clase y tener una base de datos organizada y completa. Y aunque los aparatos más modernos ya poseen este tipo de funciones (almacenamiento e integración con otros equipos) estos dejan fuera a los equipos de su clase que son de fabricación anterior.

Por lo que surge la necesidad de ver una manera de no desechar equipos en buen funcionamiento pero que no pueden integrarse a una red moderna ya que no fueron fabricados con esas características. Y acoplar equipos de medición que muestren sus resultados en monitores, pero no tengan un protocolo que permita almacenar estos datos, y en su defecto si los tienen son muy específicos para cada uno de ellos.

Así pues, en este proyecto, pretende obtener estos valores numéricos, organizarlos y almacenarlos en una base de datos. Con la que se podrá unificar los valores de lectura generados por diferentes monitores sin importar la marca ni el fabricante que estos tengan.

ÍNDICE GENERAL

CAPÍTULO 3: DISEÑO DEL SISTEMA.

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

CAPÍTULO 1: PLANTEAMIENTO DEL PROBLEMA

1 PLANTEAMIENTO DEL PROBLEMA

1.1 Título de la investigación.

"DISEÑO DE UN SISTEMA DE CAPTACIÓN DE PARÁMETROS NUMÉRICOS DE MONITORES MEDIANTE PROCESAMIENTO DIGITAL DE IMÁGENES".

1.2 Ámbito de la investigación.

El proyecto se realizará con el equipo monitor desfibrilador del laboratorio de biomédica de la Escuela Profesional de Ingeniería Electrónica, y el sujeto de prueba será el simulador de cuerpo humano E-STAN, también del mismo laboratorio.

1.3 Identificación del problema.

En la región del Cusco no hay una manera sencilla y económica de obtener los datos de distintos monitores y administrarlos en una sola base de datos sin interrumpir el adecuado monitoreo de los pacientes. Por lo que el proyecto deberá solucionar las siguientes interrogantes, teniendo como base de prueba los equipos de la Universidad:

• ¿El proyecto podrá captar los parámetros del monitor desfibrilador, ordenarlos y guardarlos adecuadamente?

• ¿El método de reconocimiento de caracteres que se utilice podrá responder a variaciones del entorno como la iluminación o el desenfoque?

• ¿El sistema será lo suficientemente preciso, y se podrá reconocer los tipos de errores?

1.4 Justificación e importancia del problema.

• Este proyecto busca la generación de bases de datos con valores obtenidos de diversos aparatos de medición digital de manera visual, para su posterior uso.

• El proyecto también pretende unificar sistemas de medición mostrados en monitores o pantallas que tengan la función de almacenar sus datos medidos, con otros que cumplen el mismo rol de medición pero que no disponen de un hardware y/o software que permita el almacenamiento de sus datos medidos.

• El proyecto tiene un amplio campo de aplicación puesto que va dirigido a reconocer caracteres numéricos en monitores, por lo que se puede emplear para guardar datos de monitores de funciones vitales, simuladores de cuerpo humano, aparatos de medición de diferentes áreas, pantallas de monitoreo climático, etc.

• El proyecto ayuda a no desechar equipos de medición en buen estado pero que no posean la capacidad de almacenamiento de sus datos en bases de datos.

• Este proyecto no requerirá de ninguna alteración física de los aparatos de medición de los que se quieran extraer los datos para la base de datos por lo que no alterará su funcionamiento normal en su área correspondiente.

• Este proyecto es de aplicación inmediata por lo que se adapta a cualquier sistema de monitoreo.

• Y por último el sistema tiene alto potencial para la detección de otros factores en la medición como por ejemplo gráficos (curvas o estadísticas), por lo que sirve de punto de apoyo para futuras investigaciones que requieran mejorar el proyecto o también hacerlo más específico en determinadas áreas.

1.5 Limitaciones de la investigación.

• El proyecto no captará los gráficos presentes en los monitores.

• El proyecto no tendrá la cámara ubicada en posiciones que se acomoden al operario, sino que tendrá que estar fija cerca del monitor para una buena captación de las imágenes.

• El proyecto solo podrá captar parámetros que usen métodos no invasivos, de los monitores de funciones vitales.

• El proyecto solo será probado en laboratorio por lo que sus repercusiones en un ambiente médico, solo se mencionarán mas no se probarán.

1.6 Objetivo de la investigación.

1.6.1 Objetivo general.

Desarrollar un sistema de captación de los valores numéricos mostrados en el monitor desfibrilador del laboratorio de biomédica para ser almacenados en un archivo *xls.

1.6.2 Objetivos específicos.

- Escoger un método adecuado para la clasificación de los datos obtenidos y ver la eficiencia de este ante variaciones del entorno como el de iluminación o enfoque de la cámara.
- Contrastar la información captada por el sistema, determinar su precisión y clasificar los errores resultantes del sistema.

1.7 Hipótesis.

Con una cámara de video se captará las variables numéricas presentes en un monitor desfibrilador con un error menor al 10%.

1.8 Definición de variables.

-Variable dependiente.

Imágenes obtenidas de la cámara.

-Variable independiente.

Valores numéricos obtenidos

CAPÍTULO 2: MARCO TEÓRICO

2 MARCO TEÓRICO

2.1 Antecedentes

2.1.1 OCR en la detección de textos en documentos a través de cámaras.

Como se ocupará más adelante es importante revisar las ramas y usos que se le dio al reconocimiento óptico de caracteres (OCR), es así que veremos cómo es empleado en la detección de textos para su posterior digitalización.

Según el trabajo de (Faruk Mollah, Majumder, Basu, & Nasipuri, 2011) [1]. El reconocimiento de texto en documentos se realizaba con scanner, pero hasta no hace mucho se hizo este trabajo que realizaba este proceso a través de la cámara de un celular y era capaz de procesarlo con el hardware del propio celular, con la tecnología cambiante y más potente, los celulares de gama alta se prestan para este tipo de procesos, al ser un proceso autónomo desde el propio dispositivo móvil se requería de celulares con buen procesador y memorias dinámicas grandes.

La precisión del reconocimiento adquirido fue de 92,74%. Los experimentos mostraron que el sistema de reconocimiento es eficiente desde el punto de vista computacional, lo que lo hace aplicable a arquitecturas de bajo nivel de informática, tales como teléfonos móviles, asistentes digitales personales (PDA), etc.

Su modelo planteado fue el siguiente:

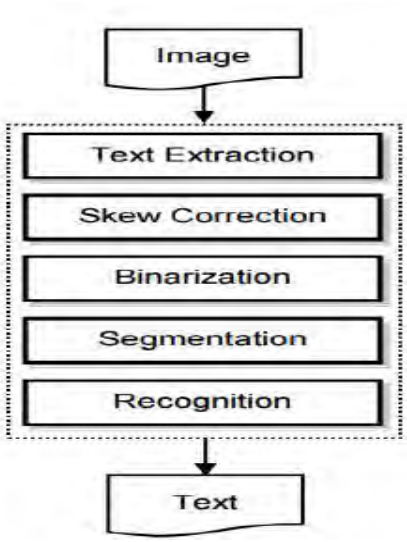


Figura 1: **Modelo del OCR empleado en la detección de textos en documentos.**

Fuente: Mollah et al., (2011) [1].

Se han realizado experimentos para evaluar el rendimiento y la aplicabilidad de esta técnica de reconocimiento de caracteres en un conjunto de 100 imágenes de tarjetas de visita de gran variedad. Las imágenes fueron capturadas con una cámara de teléfono celular (Sony Ericsson K810i).

La aplicabilidad de la técnica en dispositivos portátiles / móviles se estudiaron en términos de requisitos computacionales. El consumo de tiempo medio para reconocer una imagen de tarjeta de visita de 3 megapíxeles de resolución fue de 1,25 segundos con respecto a una computadora moderadamente potente (DualCore T2370, 1,73 GHz, 1 GB de RAM, 1 MB de caché L2).

2.1.2 OCR en la detección de placas de vehículos

Se han hecho diferentes trabajos de investigación que buscan aplicar el reconocimiento óptico de caracteres en la detección de las placas vehiculares, basados en redes neuronales, obteniendo buenos resultados, donde se ven afectado por la luminosidad al momento de capturar la imagen, la posición de la placa y otros, quedando a pesar de esto con un buen índice de acierto. (Por encima del 90%). Según

el trabajo de *N. Vázquez, M. Nakano & H. Pérez-Meana (2002)* [2] el proceso viene dado de la siguiente forma:

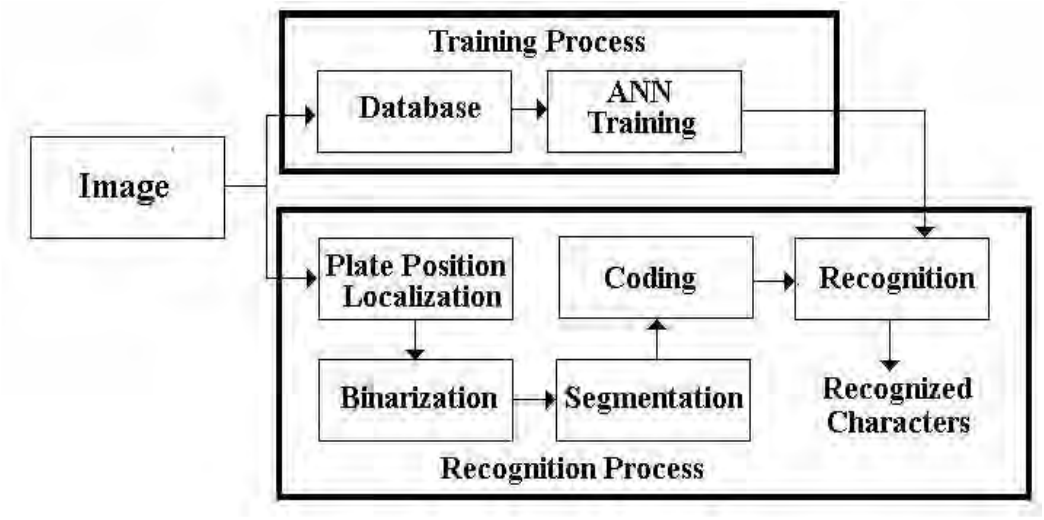


Figura 2: **Modelo del proceso OCR en la detección de placas.**

Fuente: Vázquez et al., (2002) [2].

Se evaluaron dos aspectos en el sistema propuesto: La capacidad de estimación de posición de placa y la velocidad de reconocimiento de caracteres de placa. En ambos casos se evaluó el rendimiento del sistema utilizando 310 imágenes, la mayoría de ellas correspondientes a vehículos traseros. En el primer caso, es decir la estimación de la posición de la placa, se encontró una media de localización correcta del 91,3% Obtenido, que es mejor que otros sistemas previamente propuestos. En el segundo caso se evaluó la capacidad del sistema para reconocer los caracteres de la placa segmentada de forma independiente con una tasa de reconocimiento del 95,5% cuando se requirió al sistema reconocer sólo dígitos, 91,6% cuando se requirió reconocer sólo letras y 91,2% Sistema se requiere para reconocer o identificar una placa de ciudad completa de Ciudad de México que conste de 3 dígitos y 3 letras. El sistema propuesto fue evaluado utilizando MatLab en una estación de trabajo SUN, así como en un ordenador personal. Los resultados de la evaluación muestran que los sistemas propuestos se desempeñaron bastante bien cuando se requiere identificar las placas del vehículo obtenidas de las imágenes de los vehículos.

2.1.3 Reconocimiento óptico de caracteres en entornos reales utilizando redes neuronales y k-vecino más cercano.

Como se pudo observar en los trabajos previos el OCR es una tecnología que ya se vino desarrollando en la última década por lo que nuevas técnicas son aplicadas en nuevos entornos, por ello en el trabajo de O. Matei, P.C. Pop, H. V˘alean (2013)[3] emplearon los métodos de redes neuronales junto con el algoritmo KNN para poder reconocer caracteres tomados de imágenes obtenidas por una cámara de pantallas en medidores de gas y medidores eléctricos.

Para ello emplearon el siguiente diagrama de flujo:

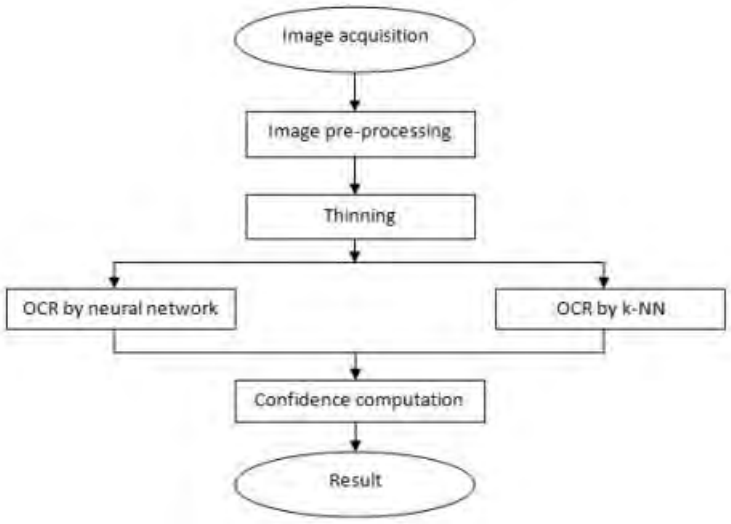


Figura 3: **Modelo del proceso OCR en entornos reales.**

Fuente: O. Matei et al. (2002) [3].

Como se puede observar antes de poder clasificar los caracteres se realiza un proceso de adelgazamiento que consistió en convertir un carácter obtenido a su forma más simple conformado por una sola línea, para recién con esto entrar tanto en la red neuronal para OCR como al algoritmo K-NN aplicado en OCR. Esto para tener un mayor grado de confiabilidad, teniendo como base si los resultados coinciden pasa a mostrarse como resultado, y en caso difieran emplearon métodos heurísticos.

Obteniendo una precisión casi del 99% el estudio mostró un gran desempeño en el OCR planteado dando un especial énfasis en sus resultados que los caracteres como 1, 3 y 5 son los mas propensos a ser identificados erróneamente con dichos métodos OCR.

2.1.4 Clasificación automática de Tweets utilizando K-NN y K-Means como algoritmos de clasificación automática.

Por último, para ver mejor el desenvolvimiento del algoritmo K-NN se verá su eficacia en la clasificación de opiniones, en la minería de datos de opinión, en este trabajo de Felipe Alberto Cifuentes Ramos (2016) [4] muestra la aplicación del algoritmo KNN y los KNN Means. Para la clasificación de Tweets en la minería de opinión, para ello empleó ponderaciones TF-IDF y TF-RFL. Para poder aplicar el algoritmo tubo que clasificar los textos en unigramas, bi-gramas o tri-gramas, que vendrían a ser la separación en conjuntos unitarios, de dos y de tres palabras respectivamente, una vez obtenido los datos se entreno al algoritmo para que pueda clasificar correctamente la información en los tweets, para ello separo los grupos de entrenamiento de 3 formas diferentes, de acuerdo a la ponderación, por la aplicación de lematización de los grupos de palabras, y por la cantidad de K vecinos.

Aplicado estos tres criterios se muestra como el sistema puede clasificar las opiniones de manera acertada, a menor numero de K vecinos, y con el número de ponderación TF-IDF, además de teniendo un desempeño muy similar en la agrupación de palabras ya sea lematizado o sin lematizar, siendo superior por un margen muy pequeño si no se lematiza.

Con esto concluye que el algoritmo funciona mejor con la ponderación TF-IDF ya que solo tiene dos estados (positivo o negativo) y baja la precisión considerablemente con la ponderación TF-RFL ya que agrega un tercer estado (positivo, negativo y neutro)

además que al incrementar las muestras de entrenamiento no mejoran notablemente el rendimiento del algoritmo por lo que concluye que no es necesario entrenar con muchas muestras al sistema.

2.2 Aspectos teóricos pertinentes.

2.2.1 Monitores de funciones vitales.

Según Aguilar (R., 2006) es un dispositivo médico electrónico que es utilizado para el registro exacto de: La frecuencia cardiaca, la frecuencia respiratoria, la saturación de oxígeno en sangre, los gráficos de la señal electrocardiográfica, la presión arterial mínima no invasiva y la temperatura periférica, este cumple por funciones lo siguiente:

• Obtener un registro exacto y de forma continua de las variantes de los parámetros nombrados por medio de módulos que facilitarán su medición y que se colocarán al paciente.

• Evaluar en todo momento y de forma completa las condiciones fisiológicas del paciente.

• Realizar valoraciones adecuadas para la toma de decisiones del diagnóstico y tratamiento.

Figura 4: **Monitor de signos vitales de 4 parámetros.**

Fuente: https://medidyne.dk/wp-content/uploads/nihon-kohden-bsm-3000.png

2.2.2 Cámaras web.

La captación de las imágenes vendrá a ser realizada por cámaras web. Que tienen una prestación adecuada con Matlab, sin contar que son de conexión directa con el ordenador. La fácil conexión que presentan las cámaras web permite que el proceso de reconocimiento de caracteres se realice de forma eficaz e inmediata, otro factor importante es que podemos modificar varios parámetros de este tipo de cámara desde Matlab, que se adecua a las necesidades del proyecto.

Figura 5: **Cámara Web "Genius 1000x".**

Fuente: http://tienda.piensads.com.sv/283-large_default/camara-web-hd-cmic-fc-1000x-720p-genius-.jpg

2.2.3 Procesamiento de imágenes.

Como se evidencia en los antecedentes una parte importante del reconocimiento de caracteres es el procesamiento de imágenes. Para ello se desarrollará las técnicas de procesamiento de imágenes necesarias en el reconocimiento de caracteres.

2.2.3.1 Definición de una imagen digital.

Según (González, Woods, & Eddins, 2003), [6] una imagen se define como una función bidimensional f(x,y) donde x e y son las coordenadas del plano que contiene todos los puntos de la misma, y f(x,y) es la amplitud en el punto (x,y) a la cual se le llama intensidad o nivel de gris de la imagen en ese punto. Si las

coordenadas ‘x’ e ‘y’, y los valores de intensidad de la función f son discretos y finitos, la llamamos imagen digital.

Las imágenes digitales se componen de innumerables elementos, cada uno con una posición y un valor específicos. Estos elementos se denominan puntos base o píxeles de la imagen, que a menudo se utilizan para representar la unidad de medida más pequeña en imágenes digitales.

2.2.3.2 Imágenes en Color.

La base para describir imágenes digitales en color es la misma que la descrita anteriormente, excepto que cada elemento o píxel se describe y codifica de manera diferente, según el espacio de color utilizado. Entonces, por ejemplo, para el espacio de color RGB (que a menudo se usa más comúnmente para representar imágenes), cada píxel se representa como un color creado a partir de una cierta cantidad de colores rojo, verde y azul (Echenique, s.f.). [7] Esta representación se puede interpretar como una matriz de tres niveles de intensidad, donde cada nivel corresponde a la intensidad del color de los componentes rojo, verde y azul, como se muestra en la Figura 6.

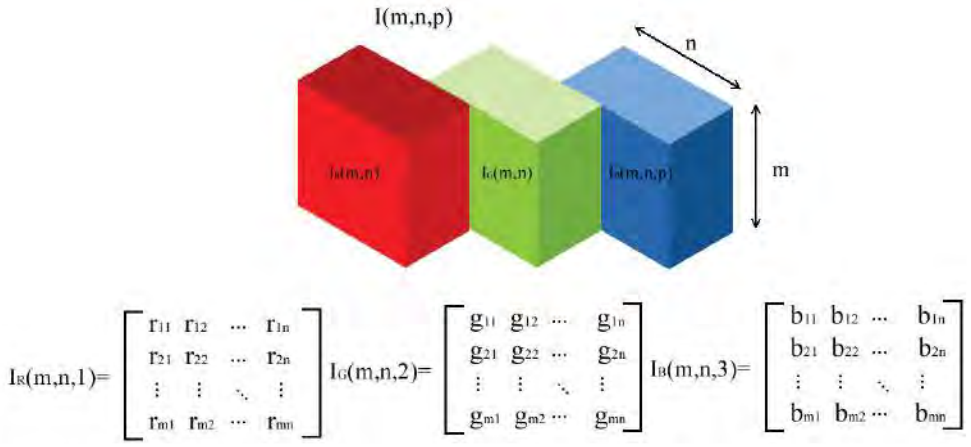


Figura 6: **Planos de color RGB representados como tres matrices bidimensionales.**

Fuente: Nicolás Aguirre Dobernack (2013) [8].

2.2.3.3 Variables de Color.

Para comprender mejor por qué existen diferentes espacios de color, primero describimos brevemente las propiedades inherentes del color según (Aguirre Dobernack, 2013), tales como, tono, brillo, y saturación.

> El matiz (Hue). Es el valor cromático de un color, la frecuencia del espectro donde se encuentra. Depende de la longitud de onda dominante, y es la cualidad que permite clasificar a los colores como amarillo, rojo, violeta, etc.
>
> La luminosidad (Lightness). Es el resultado de la mezcla de los colores con blanco o negro y tiene referencia de matiz. Representa la cantidad de luz presente en un color, más blanco o más negro, según sea el caso. Cuanto mayor es la luminosidad, mayor es la cantidad de luz en un color, es decir, más color blanco posee.
>
> La saturación (Saturation). Se refiere al grado de pureza de un color y se mide con relación al gris. Los colores con menor saturación se muestran más agrisados, con mayor cantidad de impurezas y con menor intensidad luminosa. (p. 44) [8]

2.2.3.4 Espacios de Color.

Los espacios de color son una herramienta importante en el procesamiento digital de imágenes, ya que permiten analizar cada píxel desde una perspectiva diferente, y así aprovechar toda la información contenida en la imagen. Los trabajos más recientes realizados en este campo involucran segmentación de imágenes en color, la localización de objetos, análisis de texturas, morfología matemática, estandarización de imágenes a color, etc. Los sistemas no lineales son

frecuentes en los espacios de color, porque buscan resaltar ciertas características de una imagen. (Vega Uribe & Reyes Figueroa, 2006) [9]

2.2.4 Análisis y procesamiento de imágenes.

El objetivo del análisis digital de imágenes en ingeniería es obtener las medidas, datos o información contenida en una imagen. La lista comprende técnicas que están diseñadas principalmente para simplificar la exploración y comparación de la información que contiene. Un sistema de análisis de imágenes se caracteriza por tener una imagen como parámetro de entrada y su resultado es un valor numérico, en lugar de alguna otra imagen. Esta salida transmite el contenido de una imagen de entrada. (B., 2002) [10]

Para acceder al conjunto de parámetros e información extraída de una imagen, es necesario pasar por etapas separadas de procesamiento y filtrado que analizan y ajustan la imagen para un propósito específico. Es evidente que el resultado del procesamiento está fuertemente influenciado por el problema que se está resolviendo. (González, Woods, & Eddins, 2003) [6]

El procesamiento y análisis de imágenes se desarrolló para resolver los tres mayores problemas con las imágenes:

- 'La digitalización y codificación de imágenes que facilite la transmisión, representación y almacenamiento de las mismas'.
- 'Mejora y restauración de una imagen para interpretar más fácilmente su contenido'.
- 'Descripción y segmentación de imágenes para aplicaciones de visión robótica o visión artificial'.

Todos los algoritmos de procesamiento de imágenes que tienen como objetivo enfatizar, mejorar o contrastar características específicas de una imagen mientras

minimizan cualquier ruido no deseado (incluidos los aditivos/sustractivos y otros) se conocen como técnicas de mejora de imágenes. (Bosdogianni, 1999) [11]

El conjunto de métodos de procesamiento de imágenes está dividido en tres grandes grupos según (Aguirre Dobernack, 2013):

- ‘Algoritmos en el dominio espacial. Se refiere a métodos que procesan una imagen píxel por píxel, o también tomando en cuenta un conjunto de píxeles vecinos’.
- ‘Algoritmos en el dominio de la frecuencia. Frecuentemente, estos métodos son aplicados sobre los coeficientes resultantes de la Transformada de Fourier de una imagen’.
- ‘Algoritmos de extracción de características. A diferencia de los dos grupos anteriores, los algoritmos de extracción de características están enfocados al análisis de imágenes para la extracción de atributos y regiones de interés, separación de objetos del fondo, detección de bordes o formas, entre otros’. (p. 50) [8]

2.2.4.1 Operaciones morfológicas.

Según (Aguirre Dobernack, 2013) El origen de las operaciones morfológicas se remonta a la teoría de conjuntos. El procesamiento de imágenes normalmente lo emplea cuando se utilizan imágenes binarias, donde ya se ha realizado una segmentación previa, distinguiendo el fondo (marcado con '0') de los objetos de interés (marcados como ‘1’). Una imagen que es binaria generalmente muestra una cuadrícula de valores, donde cada píxel tiene sólo dos opciones, 0 o 1 En estas condiciones, es fácil identificar y diferenciar características estructurales de una imagen.

Se utilizan operaciones morfológicas para procesar las imágenes binarias considerando la forma de sus objetos. Por lo general, toman una imagen binaria y producen otra imagen binaria. Al realizar operaciones no lineales en el píxel de entrada y sus vecinos, se puede obtener cada pixel en la imagen de salida. Generalmente, las operaciones morfológicas se emplean para:

- 'Supresión de ruidos'.
- 'Simplificación de formas'.
- 'Destacar la estructura de los objetos' (detección de envolvente, ampliación, reducción).
- 'Descripción de objetos' (área, perímetro).

Las dos operaciones morfológicas más reconocidos, la erosión y la dilatación, están diseñados para simplificar las imágenes para futuros análisis manteniendo todas sus propiedades.

Dilatación. Si consideramos una imagen A binaria, que es una combinación de los elementos de la cuadrícula Z^2, y que para un elemento estructural B, la dilatación de A por B resulta en:

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

Que se entiende como "aquellos píxeles x tales que la intersección de la estructura B situada sobre x y la imagen A es distinto del conjunto vacío". Sólo los píxeles que corresponden a los objetos de primer plano (píxeles en '1') se consideran cuando se cruzan A y B. La dilatación se ejecutará según el elemento estructural B.

La dilatación de una imagen se logra mediante el crecimiento (o "dilatación") de píxeles alrededor de los bordes del objeto. El método normalmente marca todos los píxeles del fondo de la imagen como '1' cuando están en contacto directo con el

objeto. El nivel de píxeles en el perímetro de cada objeto se puede aumentar en uno y los posibles huecos dentro del mismo.

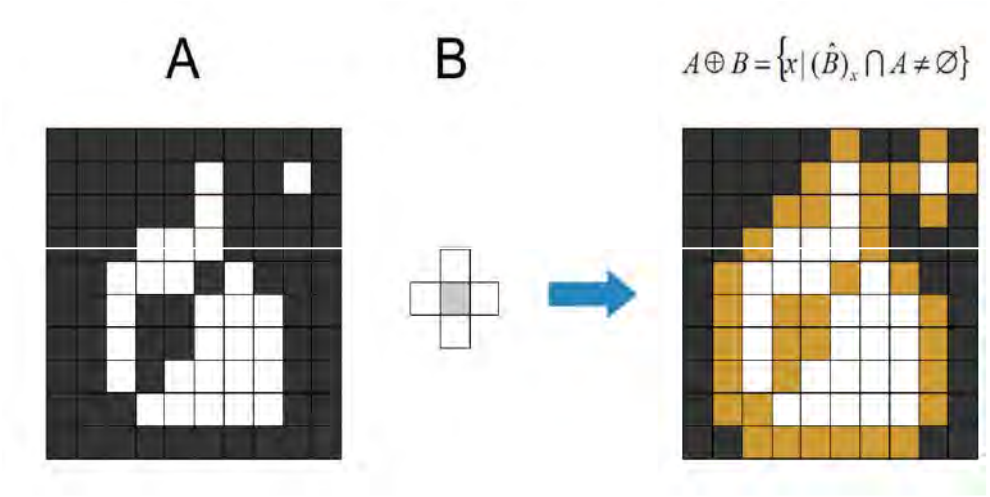


Figura 7: **"Resultados de la Dilatación sobre Imágenes Binarias."**

Fuente: Nicolás Aguirre Dobernack (2013) [8].

Cualquier píxel de entrada y sus píxeles vecinos están sujetos a la misma regla que se aplica a cada pixel de salida. Se obtuvo una imagen dilatada mediante el siguiente procedimiento: "Si cualquier píxel vecino del píxel de entrada es '1', entonces el píxel de salida es también '1'. En cualquier otro caso el píxel de salida será '0'".

Erosión. Si consideramos a una imagen binaria A, que es una combinación de los elementos de la cuadrícula Z^2, y para un elemento estructural B, la erosión de A por B resulta en:

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

Que se entiende como "aquellos píxeles x tales que la estructura B situada sobre x pertenezca en su totalidad a la imagen A". Los pixeles de los objetos en primer plano marcados con '1' son los únicos píxeles que se tienen en cuenta cuando se considera que Bx pertenece a A. La erosión es una operación

morfológica dual a la dilatación y normalmente se considera un intento de reducir la imagen original.

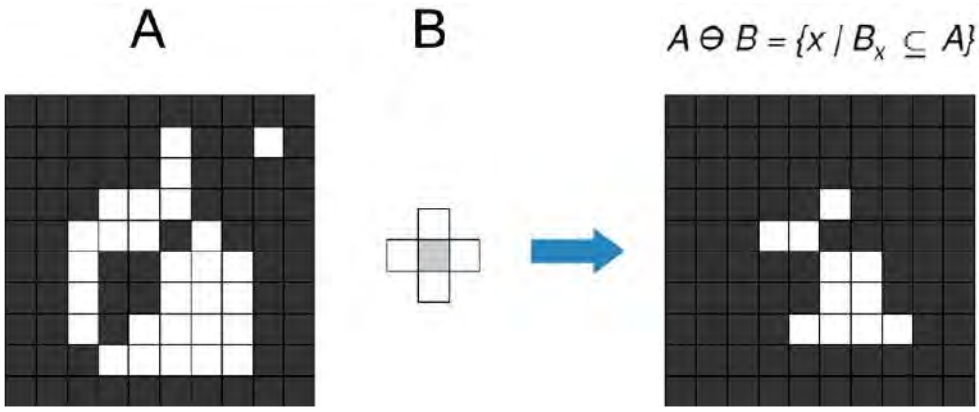


Figura 8: **"Resultado de la Erosión sobre una Imagen Binaria."**

Fuente: Nicolás Aguirre Dobernack (2013) [8].

Cualquier píxel de entrada y sus píxeles vecinos están sujetos a la misma regla que se aplica a cada pixel de salida. Se obtuvo una imagen erosionada mediante el siguiente procedimiento: "Si todos los píxeles vecinos del píxel de entrada están a '1', entonces el píxel de salida será '1'. En cualquier otro caso, el píxel de salida será '0'."

Aplicando en conjunto estas dos operaciones, erosión y dilatación, se obtienen interesantes resultados en el análisis de imágenes, suavizando los contornos, rellenando huecos para hacer los objetos más homogéneos, eliminando ruido y puntos de tamaño demasiado pequeños para resultar de interés, etc. (Figura 8). Cuando estas dos operaciones morfológicas se aplican en cadena, los objetos más grandes permanecen constantes mientras que los más pequeños desaparecen. La modificación del elemento estructural B puede revelar el tamaño mínimo de los objetos, lo que da como resultado una apariencia inalterada. (pp. 57-59) [8]

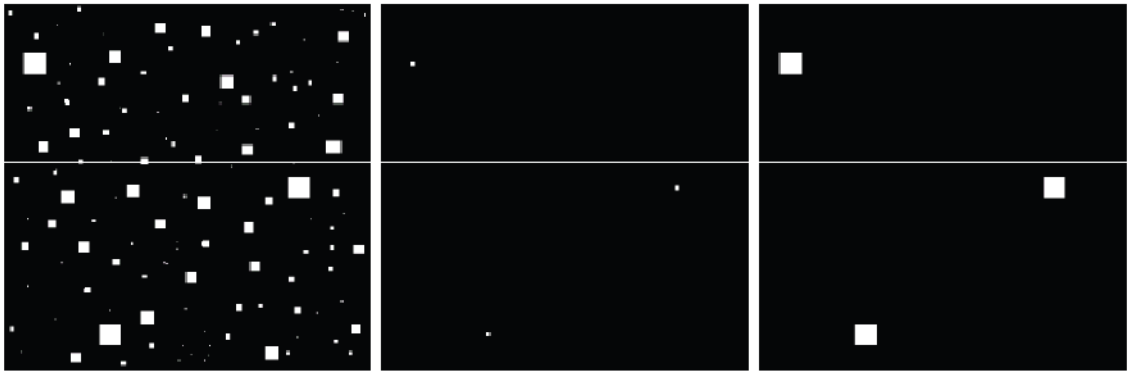


***Figura 9:* "Resultado de aplicar erosión seguida de dilatación en una Imagen Binaria."**

Fuente: Nicolás Aguirre Dobernack (2013) [8].

2.2.4.2 Métodos de extracción de características.

Para la extracción de características según define (Aguirre Dobernack, 2013) es un método distinto que se diferencia de otras técnicas en que utiliza imágenes como entradas y extrae atributos específicos que son relevantes, como coordenadas de objetos que cumplen ciertos criterios, identificación de curvas y formas y etiquetado de componentes. Este enfoque es un aspecto vital del análisis de imágenes y sirve como etapa inicial en el desarrollo de la inteligencia de un sistema de visión artificial.

Las imágenes son depósitos de grandes cantidades de datos, pero la mayoría de estos datos generalmente no son informativos y no ayudan en la interpretación de la escena. El paso inicial de un sistema de visión artificial implica la extracción de características clave de la escena que sean robustas, eficientes y rápidas por naturaleza, y que proporcionen la información necesaria para su posterior interpretación. El sistema debe cumplir varias condiciones, entre ellas:

- Sacando información útil de la imagen no debería ser un gran gasto para el sistema, y el tiempo que se necesita para hacer esto debería ser lo más corto posible.

- Es necesario que la ubicación de los detalles de la imagen sea exacta. El margen de error en la estimación debe ser lo más bajo posible.
- El método utilizado para la extracción de características debe ser robusto y estable.
- Es necesario extraer la mayor cantidad de datos de la escena, incluyendo información geométrica.

2.2.4.2.1 Segmentación.

La segmentación divide una imagen en zonas o en sus partes constitutivas, donde los píxeles de esas áreas tienen características similares, como tonos de gris, contraste o texturas.

"La mayoría de los algoritmos de segmentación están basados en dos propiedades básicas de intensidad de la imagen: la discontinuidad y la similitud. En la categoría de segmentación mediante discontinuidad, el proceso se realiza dividiendo la imagen por cambios abruptos en intensidad, como es el caso de la detección de bordes en una imagen. Con respecto a la segmentación con base en la similitud, ésta se logra mediante la partición de una imagen en regiones que son similares de acuerdo a un conjunto de criterios predefinidos" (Echenique, s.f.) [6].

Hay varios tipos de segmentación, como los que se nombran a continuación:

- Segmentación según las características de los píxeles.
  - "Segmentación por niveles de gris."
  - "Segmentación de imágenes en color."
  - "Segmentación por texturas."
- Segmentación ajustada a transiciones.
  - "Detección de bordes."

- Segmentación ajustada a modelos.
    - “Transformada de Hough.”
- Segmentación ajustada a la homogeneidad.
    - “Fusión de regiones.”
    - “Zonas planas.”
    - “Propagación de Marcadores.”
- Segmentación ajustada a la Morfológica Matemática

Para nuestro caso solo se mostrará la segmentación basada en características de píxel ya que es la que se usará más adelante en el tratamiento de las imágenes.

Segmentación ajustada a las características de los píxeles. El proceso de evaluación de cada píxel de una imagen se basa en sus características locales, así como en las de los píxeles circundantes. Esta evaluación determina la región o segmento a la que pertenece el píxel. Este método de segmentación se emplea a menudo cuando es necesario separar objetos que tienen características de color o intensidad similares de un fondo variado. Lo ideal es que los objetos a separar tengan una gama de colores o valores de escala de grises muy limitada, mientras que el fondo debe ser uniforme. En tales casos, se puede determinar un umbral de segmentación para separar el objeto del fondo. Este método de establecer un umbral se llama thresholding (literalmente "umbralización"). (pp 62-64) [8]

Figura 10: **Segmentación basada en umbral de intensidad de gris.**
Fuente: Nicolás Aguirre Dobernack (2013) [8].

"En el thresholding se define un valor umbral y se toman los píxeles en este rango según pertenezcan o no al fondo: se toman los que no pertenecen al fondo y se rechazan todos los demás. Una imagen de este tipo se muestra como una imagen binaria (de dos niveles) utilizando blanco y negro u otros colores para distinguir las regiones (no hay una convención estándar sobre cuáles son los rasgos de interés, si los blancos o los negros, así que la elección varía en cada caso)." (Echenique, s.f.) [6]

2.2.4.2.2 Etiquetado de componentes conectados.

El etiquetado de componentes, también conocido como etiquetado, es un proceso que agrupa píxeles correspondientes a un mismo objeto y les asigna una etiqueta, separando así un objeto de otro. Esta operación normalmente se realiza después de binarizar la imagen (por ejemplo, después de la segmentación del umbral, como en la Figura 11). El resultado es una imagen donde cada objeto está separado por una etiqueta diferente, lo que permite extraer características como el centroide, las coordenadas, el tamaño o la cantidad de objetos en una imagen. (Rosenfeld & Pfaltz, 1966). [12]

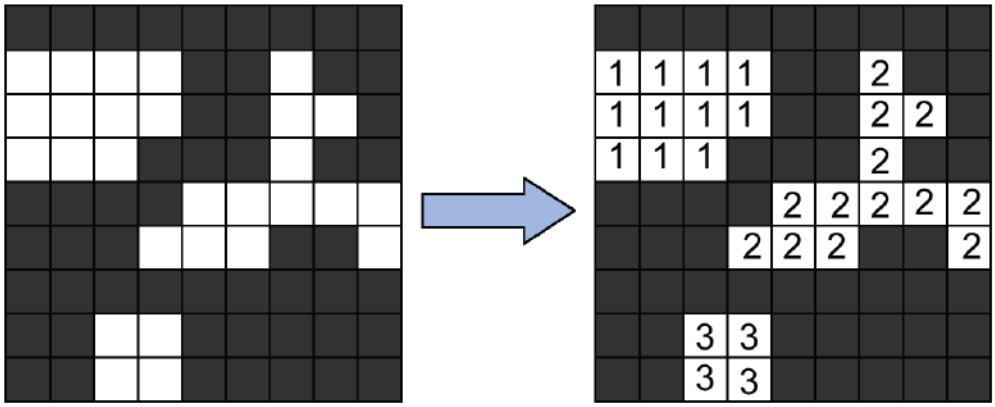


***Figura 11:* "Imagen original y resultado del etiquetado."**

Fuente: Nicolás Aguirre Dobernack (2013) [8].

2.2.5 Procesamiento de imágenes en MATLAB.

El software MATLAB es un entorno de desarrollo de código abierto que incluye un lenguaje M. También se le conoce como 'MAtrix LABoratory'. Soporta Sistemas Operativos como Unix, Windows y Mac OS X, así como para los sistemas operativos GNU/Linux.

Las características de MATLAB son diversas e incluyen: manipulación de matrices, representación de datos, implementación de algoritmos, creación de diseño GUI (GUI), interfaz con programas en otros lenguajes y dispositivos de hardware. Además, hay dos herramientas adicionales disponibles para expandir las capacidades de MATLAB: Simulink (una plataforma de simulación multidominio) y GUIDE (un editor de interfaces de usuario - GUI). Las capacidades de MATLAB también pueden ampliarse con "Toolboxes", y las de Simulink con "Blocksets". (Cuevas Jimenez & Zaldivar Navarro, S.f) [13]

MATLAB desarrolla un entorno de programación secuencial en donde cada línea de código es ejecutada de manera secuencial de arriba abajo, no requiere la

declaración de variables y, puede interactuar con múltiples plataformas de programación lo que lo hace una herramienta robusta y versátil.

Las variables que en muchos casos suelen ser matrices (en este caso imágenes) pueden tener varios tipos de datos, acá se muestran algunos de ellos, pudiendo pasar de una u otro a través de comandos propios de MATLAB. De acuerdo a la documentación de (MathWorks , 2017) [14] tenemos las siguientes definiciones:

- "double: Doble precisión, números en punto flotante que varían en un rango aproximado de -10308 a 10308 (8 bytes por elemento)".

- "uint8: Enteros de 8 bits en el rango de [0,255] (1 byte por elemento)".

- "uint16: Enteros de 16 bits en el rango de [0, 65535] (2 bytes por elemento)".

- "uint32: Enteros de 32 bits en el rango de [0, 4294967295] (4 bytes por elemento)".

- "int8: Enteros de 8 bits en el rango de [-128, 127] (1 byte por elemento)".

- "int16: Enteros de 16 bits en el rango de [-32768, 32767] (2 bytes por elemento)".

- "int32: Enteros de 32 bits en el rango de [-2147483648,2147483647] (4 bytes por elemento)".

- "logical: Los valores son 0 ó 1 (1 bit por elemento)".

Para convertir una variable con un tipo de dato predefinido a otro se emplearán los siguientes comandos:

Comando	Descripción
gray2ind	Crea una imagen indexada a partir de una imagen de intensidad en escala de gris.
im2bw	Crea una imagen binaria a partir de una imagen de intensidad, imagen indexada o RGB basado en un umbral de luminancia.
ind2rgb	Crea una imagen RGB a partir de una imagen indexada
rgb2gray	Crea una imagen de intensidad en escala de gris a partir de una imagen RGB
rgb2ind	Crea una imagen indexada a partir de una imagen RGB

Comandos de conversión de imágenes en Matlab

Figura 12: **Tabla de conversión de formatos.**

Fuente: asignatura.us.es/imagendigital/Matlab_PID_1314.pdf

En este trabajo se empleará este software por lo que será necesario conocer el funcionamiento de sus comandos para obtener los resultados deseados.

2.2.5.1 Herramientas de procesado de imagen de MATLAB.

Estos comandos son funciones preestablecidas por Matlab y son las más usadas para procesar una imagen, alterando sus características y así poder trabajar de mejor manera con ellas, es por este motivo que veremos su sintaxis y el algoritmo matemático que manejan estos comandos internamente y así entender mejor su función según (MathWorks , 2017): [14]

- *IMREAD*: Este comando permite leer un archivo de imagen. "La imagen a leer debe encontrarse en la carpeta de trabajo de Matlab. Los formatos de imagen soportados por Matlab son: TIFF, JPEG, GIF, BMP, PNG, XWD". Su sintaxis es la siguiente:

"A = imread(filename)"

"A = imread(filename,fmt)"

"A = imread(___,idx)"

"A = imread(___,Name,Value)"

"[A,map] = imread(___)"

"[A,map,transparency] = imread(___)"

Este comando convierte la imagen en una matriz de ‘uint8’, ‘uint16’ o incluso ‘logical’, dependiendo del tipo de imagen a importar.

- *IMSHOW*: Muestra a la imagen deseada en una figura. Su sintaxis es la siguiente:

“imshow(I)”

“imshow(X,map)”

“imshow(filename)”

“imshow(I,[low high])”

“imshow(___,Name,Value)”

“himage = imshow(___)”

- *IMWRITE*: “Crea un archivo de imagen que este en la variable indicada. Su sintaxis es la siguiente”:

“imwrite(A,filename)”

“imwrite(A,map,filename)”

“imwrite(___,fmt)”

“imwrite(___,Name,Value)”

- *GINPUT*: Permite identificar las coordenadas en los ejes “x” e “y” del punto o puntos hechos en una figura por medio del click del cursor. Su sintaxis es la siguiente:

“[x,y] = ginput(n)”

“[x,y] = ginput”

“[x,y,button] = ginput(...)”

Este comando guarda las coordenadas obtenidas al hacer click en una figura siendo ‘x’ e ‘y’ vectores de dos dimensiones y ‘n’ la cantidad de coordenadas a ser guardadas.

- *SIZE:* Da como respuesta los valores del tamaño de una matriz, su sintaxis es la siguiente:

"sz = size(A)"

szdim = size(A,dim) % dim = retorna el tamaño de la dimensión 'dim'

"[m,n] = size(A)" % m = filas, n = columnas.

"[sz1,...,szN] = size(A)"

- *RGB2GRAY: "*Convierte una imagen RGB a escala de grises, su sintaxis es la siguiente":

"I = rgb2gray(RGB)"

"newmap = rgb2gray(map)"

"Una matriz de imagen RGB tiene 3 dimensiones por lo que el comando utiliza el siguiente algoritmo, la conversión a escala de grises viene dada de la siguiente manera:

0.2989 * R + 0.5870 * G + 0.1140 * B

- *IMADJUST: "*Ajusta la intensidad en una imagen, se emplea para contrastar la imagen, su sintaxis es la siguiente":

"J = imadjust(I)"

"J = imadjust(I,[low_in; high_in],[low_out; high_out])"

"J = imadjust(I,[low_in; high_in],[low_out; high_out],gamma)"

"newmap = imadjust(map,[low_in; high_in],[low_out; high_out],gamma)"

"RGB2 = imadjust(RGB1,___)"

"gpuarrayB = imadjust(gpuarrayA,___)"

"Por defecto este comando satura un 1% inferior y el 1% superior de todos los valores de los pixeles, gracias a esto la operación incrementa el contraste de la imagen de salida" (MathWorks , 2017).

"Si se considera una imagen digital discreta en escala de grises I, sea entonces la probabilidad de ocurrencia de un nivel de intensidad ik dentro de la imagen una aproximación de la forma":

'$p_r(i_k) = \frac{n_k}{M \times N}$' '$k = 0,1,2, \dots, L-1$'

"donde M × N es el número total de pixeles de la imagen, n_k es el número de pixeles que poseen el nivel de intensidad i_k, y L es número de pixeles representables en la imagen. Se busca una función de transformación de los niveles de intensidad de los pixeles de la forma:

Entonces, una imagen resultante se obtiene a partir del mapeo de cada pixel de nivel de intensidad i_k de la imagen de entrada con un pixel correspondiente de nivel de intensidad i_k utilizando la ecuación anterior. Nótese que CDF(i_k) es la Función de Distribución Acumulada (CDF, por sus siglas en inglés) de la función de distribución de probabilidades $p_r(i_j)$. Finalmente, el nuevo valor de intensidad i'_k correspondiente a la imagen digital transformada se obtiene multiplicando CDF(i_k) por L − 1, es decir": (Moré Rodríguez, 2017) [15]

'$i'_k = [CDF(i_K)] \times (L-1)$' '$con\ i'_k \leq L-1$'

- *MULTITHRESH:* "Umbralización de una imagen multinivel utilizando el método de Otsu. El resultado de este comando es empleado para convertir a imágenes binarias, Su sintaxis es de la siguiente forma": *"thresh = multithresh(A)"*

"thresh = multithresh(A,N) % N número de umbrales".

"[thresh,metric] = multithresh(___)"

Según (Nobuyuki, 1979) El método de Otsu busca un valor de umbral tal que la variación dentro de cada segmento sea mínima, mientras que la variación entre los segmentos sea lo más grande posible. Se parte de dos segmentos de puntos ($K_0(t)$ y $K_1(t)$), definidos por el umbral t. Tratando de determinar el valor de *t*, los dos segmentos son el resultado deseado en la segmentación. La probabilidad de ocurrencia de los valores de gris (0 < g < G, donde G es el valor de gris máximo) se representa como *p(g),* la probabilidad de los píxeles en los dos segmentos se expresa como:

$$"K_0: P_0(t) = \sum_{g=0}^{t} p(g) \; y \; K_1: P_1(t) = \sum_{g=t+1}^{G} p(g) = 1 + P_0(t)";$$

Si tomamos dos segmentos (o sea un solo valor umbral) la suma de sus probabilidades será claramente 1.

Suponiendo que $\bar{g}$ es la media aritmética de los valores de gris en toda la imagen, y $\overline{g_0}$ y $\overline{g_1}$ los valores medios dentro de cada segmento, las varianzas dentro de cada segmento se pueden calcular como:

$$"\sigma_0^2(t) = \sum_{g=0}^{t} (g - \overline{g_0})^2 p(g) \; y \; \sigma_1^2(t) = \sum_{g=t+1}^{G} (g - \overline{g_1})^2 p(g) ";$$

El objetivo es mantener la variancia dentro de cada segmento lo más reducida posible y lograr que la variancia entre los dos segmentos sea lo más grande posible. De ello se obtiene:

$$"Q(t) = \frac{\sigma_{zw}^2(t)}{\sigma_{in}^2(t)} ";$$

La variancia entre los segmentos es:

$$"\sigma_{zw}^2(t) = P_0(t).(\overline{g_0} - \bar{g})^2 + P_1(t).(\overline{g_1} - \bar{g})^2";$$

La variancia se logra de la adición entre amabas dentro de los segmentos:

$$"\sigma_{in}^2(t) = P_0(t).\sigma_0^2(t) + P_1(t).\sigma_1^2(t)";$$

El valor umbral '*t'* se escoge obteniendo el máximo valor del cociente *Q(t). Q(t)* será el valor buscado. Así se escogerá el valor optimo del umbral entre dos segmentos en función de la varianza. (pp. 62-66) [16]

• IMQUANTIZE: Cuantifica una imagen utilizando los niveles de cuantificación y valores de salida detallados. La sintaxis de este comando es de la siguiente manera:

"quant_A = imquantize(A,levels)"

"quant_A = imquantize(___,values)"

"[quant_A,index] = imquantize(___)"

"La imagen de salida quant_A tiene el mismo tamaño que A y contiene N + 1 valores enteros discretos en el intervalo 1 a N + 1, que están determinados por los siguientes criterios":

"Si A(k) ≤ levels(1), quant_A(k) = 1".

"Si levels(m-1) < A(k) ≤ levels(m), quant_A(k) = m".

"Si A(k) > levels(N), quant_A(k) = N + 1".

"Observe que imquantize asigna valores a los dos intervalos finales definidos de manera implícita":

"A(k) ≤ levels(1)"

"A(k) > levels(N)"

• *IMBINARIZE:* Binariza una imagen a partir de umbrales, su sintaxis es la siguiente:

“BW = imbinarize(I)”

BW = imbinarize(I,method) % method = método de umbralizacion puede ser ‘global’ o ‘adaptativo’

BW = imbinarize(I,T) % T = indica el umbral

“BW = imbinarize(I,'adaptive',Name,Value)”

El método de Umbralización es el de Otsu anteriormente explicado.

• *STREL:* Este comando no trabaja por sí solo, es un elemento de estructuración morfológica, que coge los datos de una matriz de acuerdo al arreglo escogido. Su sintaxis es la siguiente:

“SE = strel(‘arrangement’,’Properties’)”

Este comando presenta diversos arreglos sin embargo solo se mencionará el que se esta usando en el presente trabajo:

“SE = strel('disk',R,N) Crea un elemento estructurador de forma de disco, R es el radio y N las líneas de deformación”.

El resultado de esta operación morfológica es generalmente usado por los comandos de dilatación y erosión explicados en el capítulo anterior.

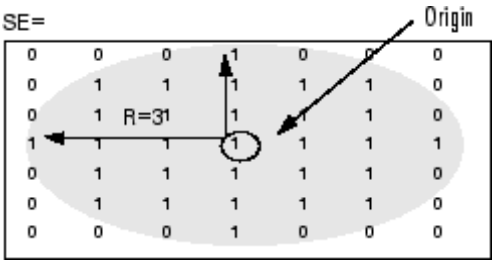


Figura 13: **Elemento Estructurante en forma de disco (‘disk’).**

Fuente: Matlab Product Help [16]

• *IMDILATE:* Dilata una imagen binaria de acuerdo a un elemento estructurador (dados por el comando 'STREL'), su sintaxis es la siguiente:

"IM2 = imdilate(IM,SE)"

"IM2 = imdilate(IM,NHOOD)"

"IM2 = imdilate(___,PACKOPT)"

"IM2 = imdilate(___,SHAPE)"

"gpuarrayIM2 = imdilate(gpuarrayIM,___)"

• *IMERODE:* Erosiona una imagen binaria de acuerdo a un elemento estructurador (dados por el comando 'STREL'), su sintaxis es la siguiente:

"IM2 = imerode(IM,SE)"

"IM2 = imerode(IM,NHOOD)"

"IM2 = imerode(___,PACKOPT,M)"

"IM2 = imerode(___,SHAPE)"

"gpuarrayIM2 = imerode(gpuarrayIM,___)"

• *IMOPEN:* Erosiona y luego dilata una imagen, empleando el mismo elemento estructurador, su sintaxis es la siguiente:

IM2 = imopen(IM,SE)

IM2 = imopen(IM,NHOOD)

gpuarrayIM2 = imopen(gpuarrayIM,___)

• *IMCLOSE:* Dilata y luego erosiona una imagen, empleando el mismo elemento estructurador, su sintaxis es la siguiente:

"IM2 = imclose(IM,SE)"

“IM2 = imclose(IM,NHOOD)”

“gpuarrayIM2 = imclose(gpuarraryIM,___)”

- *REGIONPROPS:* Mide las propiedades de las regiones de una imagen, su sintaxis es la siguiente:

“stats = regionprops(BW,properties)”

“stats = regionprops(CC,properties)”

“stats = regionprops(L,properties)”

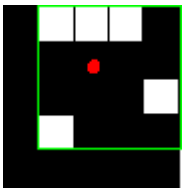
“stats = regionprops(___,I,properties)”

“stats = regionprops(output,___)”

“stats = regionprops(gpuarrayImg,___)”

Este comando tiene varias propiedades que pueden ser medidas, sin embargo, solo se describirá las que serán de utilidad al trabajo:

Nombre de la Propiedad	**Descripción**
‘Area’	“Devuelve un escalar que especifica el número real de píxeles en la región. (Este valor puede diferir ligeramente del valor devuelto por bwarea, que pesa diferentes patrones de píxeles de forma diferente.)”
‘BoundingBox’	“Devuelve el rectángulo más pequeño que contiene la región, especificado como un vector 1-por-Q * 2, donde Q es el número de dimensiones de la imagen, por ejemplo, [ul_corner width]. Ul_corner especifica la esquina superior izquierda del cuadro delimitador en la forma [x y z …]. width especifica el ancho del cuadro delimitador a lo largo de cada dimensión en la forma [x_width y_width …]. regionprops utiliza ndims para

	obtener las dimensiones de la matriz de etiqueta o la imagen binaria, ndims (L) y numel para obtener las dimensiones de los componentes conectados, numel (CC.ImageSize)".
'Centroid'	"Devuelve un vector 1-por-Q que especifica el centro de masa de la región. El primer elemento de Centroid es la coordenada horizontal (o coordenada x) del centro de masa, y el segundo elemento es la coordenada vertical (o coordenada y). Todos los demás elementos de Centroid están en orden de dimensión. Esta figura ilustra el centroide y la caja delimitadora para una región no contigua. La región consiste en los píxeles blancos; el cuadro verde es el cuadro delimitador y el punto rojo es el centroide". 
'EulerNumber'	"Devuelve un escalar que especifica el número de objetos en la región menos el número de agujeros en esos objetos. Esta propiedad sólo es compatible con matrices de etiquetas 2-D. regionprops utiliza la conectividad 8 para calcular la medición del número de Euler. Para obtener más información sobre la conectividad, consulte Conectividad de píxeles".
'Orientation'	"Devuelve un escalar que especifica el ángulo entre el eje x y el eje mayor de la elipse que tiene los mismos segundos momentos que la región. El valor es en grados, que van desde -90 a 90 grados.

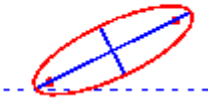
	Esta figura ilustra los ejes y la orientación de la elipse. El lado izquierdo de la figura muestra una región de imagen y su correspondiente elipse. El lado derecho muestra la misma elipse con las líneas azules sólidas que representan los ejes, los puntos rojos son los focos, y la orientación es el ángulo entre la línea punteada horizontal y el eje mayor”.  

Tabla 1: **Propiedades del comando “Regionprops”**

Fuente: Matlab Product Help [16]

• *BWMORPH:* Realiza Operaciones morfológicas en imágenes binarias, al igual que en el comando anterior, este tiene varias operaciones por lo que se procederá a mostrar solo los que nos competen:

Operación	**Descripción**
'branchpoints'	Encuentra puntos de rama del esqueleto. Por ejemplo: 0 0 1 0 0 — 0 0 0 0 0 0 0 1 0 0 sería 0 0 0 0 0 1 1 1 1 1 — 0 0 1 0 0 0 0 1 0 0 — 0 0 0 0 0 0 0 1 0 0 — 0 0 0 0 0 “Nota: Para encontrar puntos de ramificación, la imagen debe estar esqueletizada. Para crear una imagen esqueletizada, utilice bwmorph (BW, 'skel')”.

'clean'	“Elimina los píxeles aislados (1s individuales que están rodeados por 0s), como el píxel central en este patrón”. 0 0 0 0 1 0 0 0 0
'close'	“Realiza cierre morfológico (dilatación seguida de erosión)”.
'diag'	“Utiliza relleno diagonal para eliminar la conectividad 8 del fondo. Por ejemplo”: 0 1 0 0 1 0 1 0 0 sería 1 1 0 0 0 0 0 0 0
'endpoints'	“Encuentra los puntos finales del esqueleto. Por ejemplo”: 1 0 0 0 1 0 0 0 0 1 0 0 sería 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 “Nota: Para encontrar puntos finales, la imagen debe estar esqueletizada. Para crear una imagen esqueletizada, utilice bwmorph (BW, 'skel')”.
'fill'	“Rellena los píxeles interiores aislados (0s individuales que están rodeados por 1s), como el píxel central en este patrón”. 1 1 1 1 0 1 1 1 1
'hbreak'	“Elimina los píxeles conectados en H. Por ejemplo”: 1 1 1 1 1 1 0 1 0 sería 0 0 0 1 1 1 1 1 1

'open'	“Realiza apertura morfológica (erosión seguida de dilatación)”.
'remove'	“Elimina los píxeles interiores. Esta opción establece un píxel en 0 si todos sus 4 vecinos conectados son 1, dejando sólo los píxeles de límite en 1”.
'shrink'	“Con n = Inf, encoge objetos a puntos. Elimina los píxeles para que los objetos sin agujeros se contraigan hasta un punto, y los objetos con agujeros se encogen a un anillo conectado a medio camino entre cada agujero y el límite exterior. Esta opción conserva el número de Euler”.
'skel'	“Con n = Inf, elimina los píxeles en los límites de los objetos, pero no permite que los objetos se separen. Los píxeles restantes constituyen el esqueleto de la imagen. Esta opción conserva el número de Euler”.
'spur'	“Elimina píxeles de spur. Por ejemplo”: 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 sería 0 0 0 0 0 1 0 0 0 1 0 0 1 1 0 0 1 1 0 0

Tabla 2: **Operaciones del comando “Bwmorph”**

Fuente: Matlab Product Help.[16]

2.2.6 Reconocimiento óptico de Caracteres (OCR).

La tecnología tiene como objetivo imitar la percepción de los objetos por parte del ojo humano. Al utilizar el reconocimiento óptico, el software puede identificar secuencias de caracteres en imágenes escaneadas, haciéndolas visibles para los humanos y creando un archivo de texto editable para la computadora. (Kulturaren Euskal, 2011) [17]

Habiendo ya visto los conceptos básicos de procesamiento digital de imágenes y las herramientas de Matlab que sirven a este propósito ahora nos enfocaremos en el proceso de reconocimiento de caracteres, y los pasos a seguir.

Este sigue el siguiente proceso:

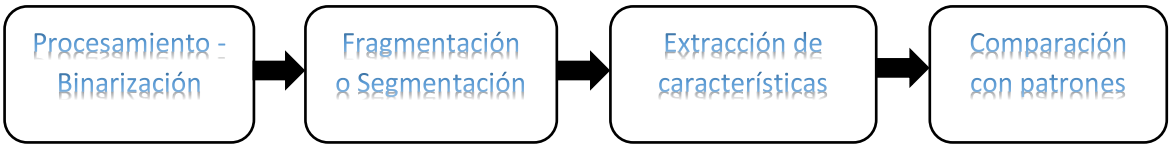


Figura 14: **Diagrama de flujo del proceso del OCR**

Fuente: Elaboración propia.

2.2.6.1 Procesamiento.

Según (Sánchez Fernández & Sandonís Consuegra, s.f.) [18], la fase de preprocesamiento o adaptación de la imagen tiene como objetivo eliminar cualquier ruido o imperfección de la imagen y normalizar su tamaño. Esto es importante.

¿Cuáles son estas técnicas? Además de la normalización de imágenes, el OCR también puede incluir la binarización de imágenes. Se emplea una variedad de algoritmos para eliminar el ruido de las imágenes digitales, que puede ser causado por manchas reales o gráficos incorrectos, o por defectos técnicos en la adquisición o binarización de la imagen.:

- Etiquetado: para dividir la imagen en regiones de componentes conectados.
- Erosión / expansión: para la eliminar pequeños grupos de píxeles.
- Umbralizado de histograma: para eliminar/seleccionar objetos más brillantes o más oscuros dentro de la imagen.

Figura 15: **Proceso de binarización.**

Fuente: Vázquez et al., (2002) [2].

2.2.6.2 Segmentación.

Una vez que esta imagen se procesa por adelantado, se debe dividir o segmentar en las partes conectadas (parte de la imagen donde todos los píxeles están juntos) que forman la imagen completa.

Para identificar correctamente todos los caracteres de una imagen binaria, uno de los mayores desafíos del reconocimiento es la fragmentación o segmentación de una imagen. Los elementos lógicos deben ser independientes y no dependientes del escritor, y también lo suficientemente significativos como para ser reconocidos.

Localizar caracteres en el texto de un documento es crucial para recrearlos correctamente. Esto requiere considerar su disposición, incluidos los espacios vacíos y los finales de línea, para identificar posibles errores y reutilizar el texto original. (Sánchez Fernández & Sandonís Consuegra, s.f.) [18]

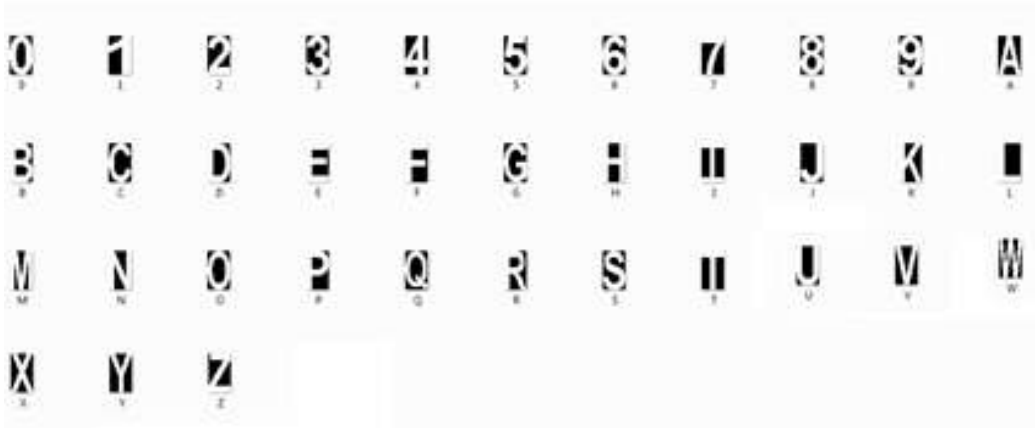


Figura 16: **Segmentación de caracteres.**

Fuente: Vázquez et al., (2002). [2]

2.2.6.3 Extracción de Características.

Cose se ve en el trabajo de (CHOQUE GARCÍA, 2015). [19] Elegir el conjunto de características más adecuado es una tarea desafiante durante la extracción de características, lo que plantea desafíos importantes para los sistemas de reconocimiento de caracteres. Una calidad sólo se consigue cuando tiene:

- Discriminativas: Las características deben poder diferenciarse suficientemente, una clase de otra.
- Las clases deberán tener los mismos valores.
- Independencia: Las características deben estar no correlacionadas unas de otras.
- Reducida cantidad de características: La cantidad de características debe ser mínima para la rapidez y facilidad de clasificación.

2.2.6.4 Comparación con patrones.

De la Figura 16 se tiene una recopilación de imágenes de tipos de caracteres que luego se espera que sean reconocidos. El conjunto de datos se denomina conjunto de entrenamiento. Además de recopilar datos, la fase de

entrenamiento también exige una cantidad significativa de preprocesamiento de los datos originales para producir representaciones compactas y coherentes. En el ejemplo de OCR, las imágenes deben segmentarse (eliminación de ruido y selección del cuadro de inclusión mínimo), normalizarse y transformarse (extracción de características) para producir vectores de baja dimensión que finalmente se guardan como un conjunto de entrenamiento. (Hilera González, Romero Villaverde, & Gutiérrez de Mesa, S.f.) [20]

2.2.7 Algoritmo K-N-N.

En la actualidad, hay una gran cantidad de sistemas de aprendizaje que han surgido del campo de investigación sobre el reconocimiento de formas. Esta variedad es tan extensa que se ha creado una taxonomía para clasificar los distintos métodos de acuerdo a sus características (paramétricos/no-paramétricos, supervisados/no-supervisados, etc.). (Sánchez Fernández & Sandonís Consuegra, s.f.) [17]

Para entender mejor un sistema de aprendizaje, se verá las tareas que pueden realizar, de acuerdo a (García Cambronero & Gómez Moreno, S.f.): [21]

- Descripción: se utiliza con frecuencia como análisis preliminar de datos (resumen, características, casos extremos, etc.). Esto ayuda al usuario a comprender la estructura de los datos. Busca obtener descripciones simples de características de datos (como medias, desviaciones estándares, etc.).
- Predicción: Se divide en dos: Clasificación y Estimación.
  - Clasificación: Los datos se clasifican como objetos que pertenecen a diferentes clases (etiquetas discretas). El objetivo es inducir un modelo con la capacidad de predecir una clase según los valores de sus atributos. Los árboles de decisión, las reglas, el análisis de discriminantes y otros recursos se los que normalmente se utilizan.

- Estimación o Regresión: las clases siguen siendo consistentes. El objetivo es inducir un modelo con la capacidad de predecir el valor de la clase en función de los valores de los atributos. Las redes neuronales, kNN, los árboles de regresión y la regresión lineal son algunos de los recursos que se utilizan.

- Segmentación: dividir los datos en clases o subgrupos interesantes. Las clases pueden ser completas y mutuamente exclusivas, o pueden ser jerárquicas y con desplazamientos. Se puede combinar con otras técnicas de minera de datos, como considerar cada subgrupo de datos por separado, etiquetarlos y usar un algoritmo de clasificación. Se utilizan algoritmos de clasificación, SOM (mapas de organización propia), EM (optimización de expectativas), k-means, etc. El usuario generalmente es bueno en la creación de clases, y se han creado herramientas visuales interactivas para ayudar al usuario.
- Análisis de dependencias: El valor de un elemento puede predecir el valor de otro elemento. La dependencia puede ser probabilística, funcional o definir una red de dependencias. También se ha centrado en determinar si existe una alta proporción de valores de ciertos atributos que ocurren con cierta medida de confianza junto con valores de otros atributos. Las reglas de asociación, las redes bayesianas y las redes causales son algunas de las opciones disponibles.
- Detección de desviaciones, casos extremos o anomalías: Identificar los cambios más significativos en los datos en comparación con valores normales o previos. Sirve para filtrar grandes cantidades de datos que rara vez son interesantes. El desafío radica en determinar si una variación es significativa y de interés.
- Aprendizaje de cuál es la mejor acción a tomar a partir de experiencia: Esto implica buscar y explorar el entorno. Esto está relacionado principalmente con el aprendizaje

por refuerzo, pero también con métodos como el EBL, el chunking y el aprendizaje de macrooperadores.

- Optimización y búsqueda: Existe una amplia gama de algoritmos de búsqueda determinísticos y aleatorios, individuales y poblacionales, locales y globales, que se utilizan principalmente para resolver problemas de optimización. Aquí podemos incluir algoritmos genéticos, simulaciones de recocido, técnicas ant-colony y técnicas de búsqueda local.

Dentro de toda esta amplia gama de sistemas de aprendizaje, no todos son adecuados para cualquier problema; sin embargo, en función de los tipos de datos con los que se debe tratar, algunos sistemas son superiores a los demás. En el caso de las aplicaciones OCR, se utiliza ampliamente un método estadístico, no paramétrico y supervisado conocido como los "vecinos más cercanos de K" (knn, k-nearest-neighbours). Debido a su sencillez y a una serie de propiedades estadísticas bien conocidas, este método es muy popular. Estas propiedades estadísticas le brindan un buen comportamiento para afrontar una variedad de problemas de clasificación, incluido el de OCR. El método funciona de manera básica de la siguiente manera: se da un conjunto de objetos prototipo cuya clase ya se conoce (es decir, un conjunto de caracteres de muestra) y se da un nuevo objeto cuya clase no se conoce (una imagen de un carácter a reconocer), y se buscan los "k" más similares al nuevo objeto entre el conjunto de prototipos. Entre los "k" objetos prototipos seleccionados, a este se le asigna la clase más numerosa.

2.2.7.1 Modelo Matemático del algoritmo K-NN.

(MathWorks , 2017) [22], define al algoritmo K-NN de la siguiente manera. Si se tiene un conjunto X de *n* puntos y una función de distancia, la búsqueda de k-vecino más cercano (kNN) permite localizar los *k* puntos más

cercanos en *X* a un punto de consulta o conjunto de puntos *Y*. Esta técnica y algoritmos basados en ella son ampliamente usados como reglas de aprendizaje de referencia, ya que su relativa sencillez facilita la comparación de los resultados de otros métodos de clasificación con los obtenidos por el método kNN. Esta técnica se ha empleado en áreas variadas como:

- La 'bioinformática'.
- El 'procesamiento de imágenes y compresión de datos'.
- La 'recuperación de documentos'.
- La 'visión por computador'.
- Las 'bases de datos multimedia'.
- Los 'análisis de datos de marketing'.

Se puede usar la búsqueda kNN para otras técnicas de aprendizaje automático, tales como:

- La 'clasificación kNN'.
- La 'regresión local ponderada'.
- La 'imputación e interpolación de datos faltantes'.
- La 'estimación de densidad'.

El algoritmo kNN también emplea dos métodos de búsqueda para clasificación de datos, una por búsqueda exhaustiva y otra por kd-tree. Esto dependerá de las condiciones iniciales de los datos que se quieran examinar.

2.2.7.1.1 Cálculo de distancias.

Para analizar correctamente ambos casos, debemos definir primero las distancias que se pueden emplear en el algoritmo kNN

Sea la matriz de datos *X* de *mx*-por-*n*, las cuales son tratadas como vectores fila *mx*(1-por-*n*) $x_1, x_2, x_3, \ldots, x_{mx}$, y la matriz de datos *Y* de *my*-

por-n que son tratadas como vectores fila *my*(1-por-*n*) $y_1, y_2, y_3, \ldots, y_{my}$, las diferentes distancias entre los vectores '*x'* e '*y'* se determinan de la siguiente manera:

- Distancia Euclidiana.

  "$d_{st}^2 = (x_s - y_t)(x_s - y_t)'$; "

  La distancia euclidiana es un tipo particular de la distancia Minkowski, donde p = 2.

- Distancia Euclidiana Estandarizada.

  "$d_{st}^2 = (x_s - y_t)V^{-1}(x_s - y_t)'$; "

  Donde *V* es la matriz diagonal *n*-por-*n* cuyo *jth* elemento de la diagonal es $(S(j))^2$, donde *S* es un vector de factores de escala para cada dimensión.

- Distancia Mahalanobis.

  "$d_{st}^2 = (x_s - y_t)C^{-1}(x_s - y_t)'$;"

  Donde *C* es la matriz de covarianza

- Distancia 'City Block'.

  "$d_{st} = \sum_{j=1}^{n} |x_{sj} - y_{tj}|$;"

  La distancia 'City Block' es un caso particular de la distancia Minkowski, donde p = 1.

- Distancia Minkowski.

  "$d_{st} = \sqrt[p]{\sum_{j=1}^{n} |x_{sj} - y_{tj}|^p}$;"

- Distancia Chebychev.

  "$d_{st} = max_j\{|x_{sj} - y_{tj}|\}$"

La distancia Chebychev es un caso particular de la distancia Minkowski, donde p = ∞.

- Distancia Coseno.

  "$d_{st} = \left(1 - \frac{x_s y'_t}{\sqrt{(x_s x'_s)(y_t y'_t)}}\right)$;"

- Distancia Correlación.

  "$d_{st} = 1 - \frac{(x_s - \bar{x}_s)(y_t - \bar{y}_t)'}{\sqrt{(x_s - \bar{x}_s)(x_s - \bar{x}_s)'}\sqrt{(y_t - \bar{y}_t)(y_t - \bar{y}_t)'}}$;"

  Donde "$\bar{x}_s = \frac{1}{n}\sum_j x_{sj}$ Y, $\bar{y}_t = \frac{1}{n}\sum_j y_{tj}$;"

- Distancia Hamming.

  "$d_{st} = \left(\#(x_{sj} \neq y_{tj})/n\right)$;"

- Distancia Jaccard.

  "$d_{st} = \frac{\#\left[(x_{sj} \neq y_{tj}) \cap ((x_{sj} \neq 0) \cup (y_{tj} \neq 0))\right]}{\#\left[(x_{sj} \neq 0) \cup (y_{tj} \neq 0)\right]}$;"

- Distancia Spearman.

  "$d_{st} = 1 - \frac{(r_s - \bar{r}_s)(r_t - \bar{r}_t)'}{\sqrt{(r_s - \bar{r}_s)(r_s - \bar{r}_s)'}\sqrt{(r_t - \bar{r}_t)(r_t - \bar{r}_t)'}}$;"

  Donde:

  - r_{sj} es el rango de x_{sj} tomado sobre $x_{1j}, x_{2j}, \ldots, x_{mx,j}$, calculado por tiedrank.

  - r_{tj} es el rango de y_{tj} tomado sobre $y_{1j}, y_{2j}, \ldots, y_{my,j}$, calculado por tiedrank.

  - r_s y r_t son los vectores de rango coordinado de x_s y y_t, que es, $r_s = (r_{s1}, r_{s2}, \ldots r_{sn})$ y $r_t = (r_{t1}, r_{t2}, \ldots r_{tn})$.

  - $\bar{r}_s = \frac{1}{n}\sum_j r_{sj} = \frac{(n+1)}{2}$;

  - $\bar{r}_t = \frac{1}{n}\sum_j r_{tj} = \frac{(n+1)}{2}$.

2.2.7.1.2 Búsqueda exhaustiva de k-vecino más cercano.

Si los datos cumplen cualquiera de los requisitos especificados, la búsqueda kNN usará el método de búsqueda exhaustiva por defecto para encontrar los k vecinos más próximos.:

- La cantidad de columnas de X es mayor que 10.
- X es limitado.
- La métrica de distancia es:
    - ‘Euclidiana Estandarizada’
    - ‘Mahalanobis’
    - ‘Coseno’
    - ‘Correlación’
    - ‘Spearman’
    - ‘Hamming’
    - ‘Jaccard’
    - ‘Una función de distancia personalizada’

En la búsqueda kNN, se emplea el método exhaustivo si el objeto de búsqueda es un objeto de modelo de búsqueda exhaustiva. El método de búsqueda exhaustiva determina la distancia entre cada punto de consulta y cada punto en X, los organiza en orden ascendente y luego devuelve los k puntos con los valores más bajos. Este diagrama sirve como ilustración de la ecuación k = 3 y sus vecinos más cercanos.

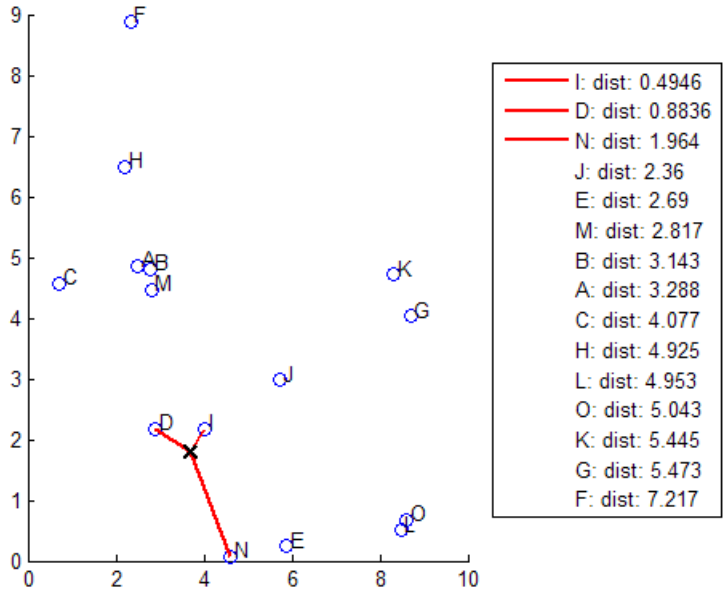


Figura 17: **Calculo de distancias a *x* de los *k=3* vecinos más cercanos.**

Fuente: MatLab, MathWorks (2017) [22].

2.2.7.1.3 Búsqueda de vecino más cercano k usando un Kd-tree.

Cuando los datos de entrada cumplen con todos los criterios, la búsqueda kNN produce un árbol Kd para identificar a sus vecinos más cercanos de forma predeterminada:

- El número de columnas de X es inferior a 10.
- X no es limitado.
- La métrica de distancia es:
  - ‘Euclidiana’ #por defecto
  - ‘City Block’
  - ‘Minkowski’
  - ‘Chevichev’

El algoritmo kNN también usa un árbol Kd si lo que se está buscando es un objeto del modelo KDTree.

Los árboles Kd dividen datos en nodos con un tamaño máximo de 50 puntos por nodo, de acuerdo con coordenadas, en lugar de categorías. Esto se ve en los diagramas, que usan objetos de parche para codificar con colores los diferentes "cuadros".

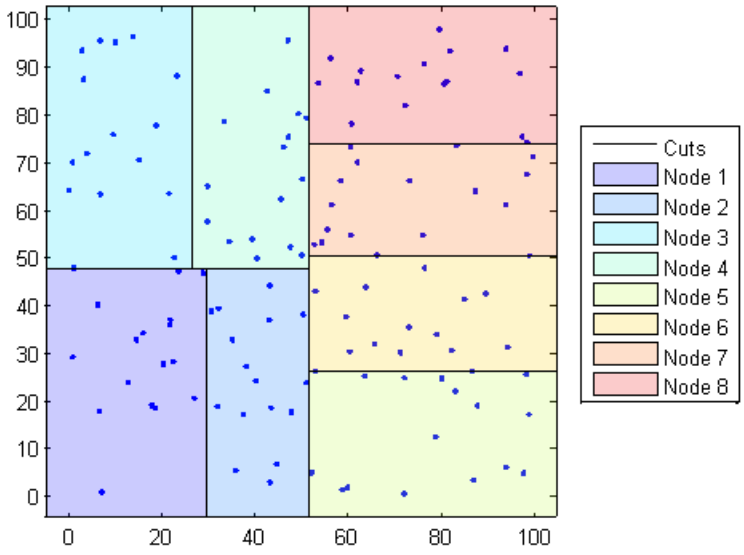


Figura 18: **Diagrama de nodos en la búsqueda de Kd-trees.**

Fuente: MatLab, MathWorks (2017) [22].

2.2.7.2 Fase de entrenamiento y fase de test.

Como lo describe (García Cambronero & Gómez Moreno, S.f.) [21], Para comenzar a utilizar el método de clasificación de 'k vecinos más cercanos', es esencial recopilar un conjunto de datos etiquetados que incluya muestras prototipo y las clases a las que pertenecen. Como se mencionó anteriormente en el OCR se encuentra normalizada la fase de entrenamiento, para luego pasar a la de test. Sin embargo, en el método k-NN la fase de entrenamiento se denominará como creación de la “Matriz de Predicción” puesto que las características que se escojan y se extraigan de las muestras prototipo se almacenarán en una base de datos o “matriz”, además de que el método denomina como “predictores” a las

características extraídas de las muestras prototipo. Y tal y como se mencionó párrafos atrás, en esta etapa se debe obtener vectores de baja dimensionalidad, para ello las imágenes deben segmentarse, normalizarse y transformarse mediante la eliminación de ruido y la selección de características.

2.2.7.3 Elección del valor de 'k'.

Siempre que en él se incluya un espacio bidimensional e innumerables prototipos de la misma clase. Si consideramos cualquier patrón X, los prototipos más cercanos a X se colocarán en un círculo con punto central en X, En las figuras 18 y 19 que se muestran en la siguiente página resaltamos los 7 vecinos más cercanos a tres patrones.

Encontramos que, en regiones con alta densidad de población, el área del círculo alrededor de un número determinado de puntos, k disminuye a medida que el recuento de puntos aumenta más que donde los puntos están dispersos. La base para la estimación de k vecino más cercano es esta sencilla técnica'. En el espacio multidimensional, una hiperesfera es lo que es el círculo.

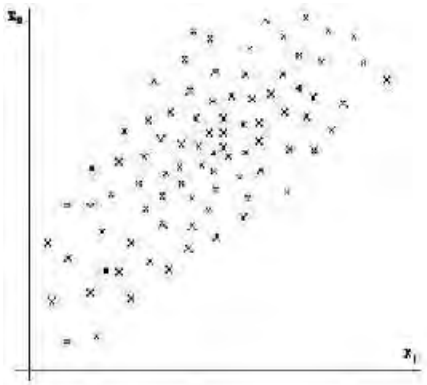


Figura 19: **Patrones en un espacio bidimensional.**

Fuente: Cristina García et al. [20]

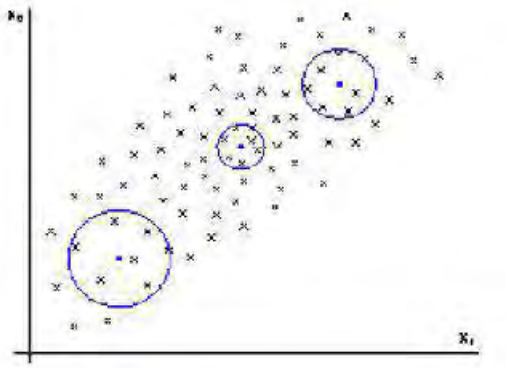


Figura 20: **Estimación de los vecinos más cercanos.**

Fuente: Cristina García et al. [21]

Si $K_i(X)$ $i = 1, 2, 3..., J$ es el número de prototipos de clase w_i que se encuentran en una vecindad de X, el valor de

"$\hat{p}(\frac{X}{w_i})$";

Puede calcularse como la proporción de los N_i prototipos de la clase que se encuentran en esa vecindad:

"$\hat{p} = \frac{K_i(X)}{N_i}$;"

si tomamos en cuenta que:

Si tomamos en cuenta que un área con mucha población tendrá un radio de vecindad más pequeño, mientras que una zona con menos población tendrá un radio más grande, necesitamos modificar la ecuación de manera que el área de la región sea ponderada en sentido inverso. Por tanto, obtenemos:

"$\hat{p} = \frac{K_i(X)}{N_i V(X)}$;"

Ahora nos enfrentamos a la pregunta: ¿hay un valor óptimo para k, o al menos hay algunas reglas para establecerlo? La cantidad de k depende de la densidad de los puntos y debe basarse en N_i.

Se puede demostrar que, si se cumplen ciertas condiciones sobre k, el estimador anterior no es sesgado y es consistente:

" $\lim_{N_i \to \infty} k(N_i) = \infty$;"

" $\lim_{N_i \to \infty} \frac{k(N_i)}{N_i} = \infty$;"

Entonces, una elección adecuada de $k(N_i)$ es: $N_i = cte\sqrt{N_i}$.

2.2.7.4 Consideraciones Computacionales.

La aplicación práctica de las reglas de clasificación de vecindad requiere estar familiarizado con ciertas limitaciones de convergencia y la carga computacional asociada con su implementación.

Para empezar, es importante señalar que el costo computacional de las reglas 1-NN y k-NN son similares. Una diferencia clave es que cuando se buscan *k* vecinos más cercanos se debe mantener una estructura auxiliar para garantizar que los *k* vecinos estén siempre en orden, cualquiera que sea el valor de *k*, la búsqueda debe considerar todo el conjunto de referencia.

Esto significa que el coste de la búsqueda depende linealmente de *N*. Calculando la distancia Euclídea nos llevaría un tiempo lineal respecto a *d* (de forma global, O(Nd)), pero con la distancia de Mahalanobis sería cuadrático (de forma global, O(Nd2)). Además, tendríamos que usar espacio para guardar todos los prototipos, que es alrededor de O (Nd). Si el conjunto de referencia es bastante grande, usar la fuerza bruta para aplicar las reglas k-NN no sería práctico, especialmente si hay muchos datos de alta dimensionalidad.

Todo lo que hemos hablado hasta ahora nos lleva a la conclusión de que hay dos cosas que determinan cuánto cuesta computacionalmente usar el algoritmo de k-NN:

- La dimensionalidad de los datos, *d*.

- El tamaño del conjunto de referencia, *N*.

2.2.7.5 Ventajas.

- No hay ningún coste de aprendizaje.

- No necesitamos adivinar qué cosas se tienen que aprender.

- Podemos adquirir conceptos complicados usando métodos simples como forma de acercarnos a ellos.

- Podemos ampliar el sistema para predecir un resultado sin interrupción (regresión).

- Es muy bueno para lidiar con el ruido.

2.2.7.6 Desventajas.

- El precio de encontrar los k mejores vecinos es alto (usando estructuras kd-trees).

- No hay una forma exacta de determinar cuál es el valor ideal para k (depende de cada set de datos).

- Si hay más descriptores, su desempeño se ve afectado.

- Su interpretabilidad es nula (No hay explicación de lo que se ha aprendido).

CAPÍTULO 3: DISEÑO DEL SISTEMA

3 DISEÑO DEL SISTEMA

Como se mencionó anteriormente se aplicará el reconocimiento óptico de caracteres a monitores de prueba, para ello se debe diseñar un sistema que cumpla con los requisitos de los objetivos, así se siguió el siguiente diagrama de flujo:

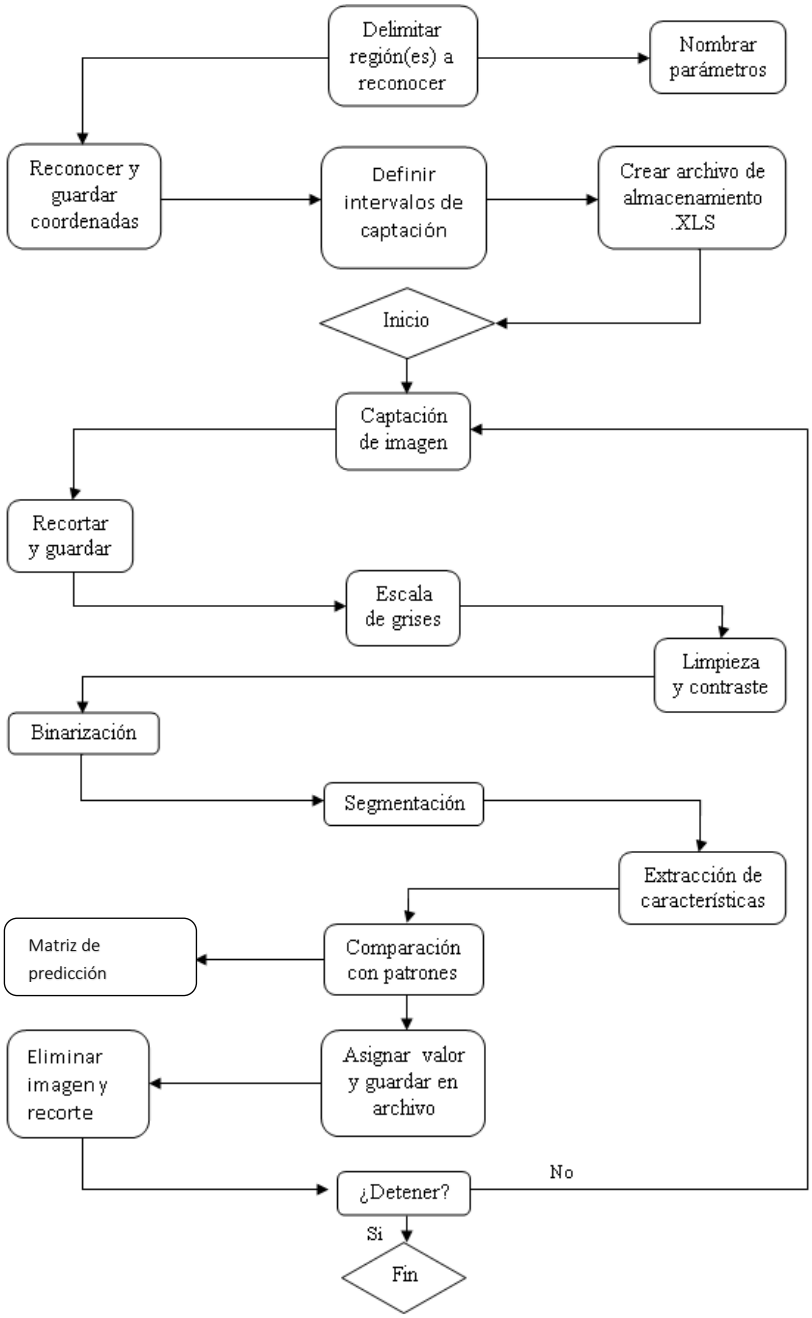


Figura 21: **Diagrama de flujo general del sistema.**

Fuente: Elaboración propia. (2017)

3.1 Diseño de la interface de usuario.

La primera etapa que se desarrollará es la de adquisición de imágenes y aunque la interface de usuario fue la última que se diseñó ya que contiene los demás bloques del proceso dentro de ella, será la que se expondrá a continuación ya que nos permite ver con claridad como fue el diseño de obtención de imágenes a través de la cámara y MatLab así como el diseño de una interface amigable con el usuario para que pueda operar el sistema de Reconocimiento de Caracteres.

La interface de usuario es una herramienta muy útil si lo que se quiere es introducir o modificar variables dentro de un proceso ya establecido, por ello es perfecto para analizar en la primera etapa del proyecto, hasta antes de la etapa que se repetirá para el reconocimiento de los caracteres. Como se describió en el diagrama de flujo, la interface de usuario nos permitirá modificar en cada sesión las regiones a reconocer, nombrar los parámetros dentro de dichas regiones, delimitar con un click los límites de las regiones y, definir el intervalo de captación de las imágenes a través de la cámara.

3.1.1 Interface de selección.

Se implementó una interface de usuario, donde éste pueda hacer modificaciones de ciertos parámetros de manera rápida y sencilla.

Específicamente la interface de selección solo busca que el usuario decida cuántas regiones de la imagen captada por la cámara van a ser procesadas. En ésta se podrá hacer el o los recortes necesarios según se haya seleccionado. Para ello Matlab ofrece herramientas dinámicas para la creación de interfaces de usuario. Para la realización de la etapa inicial se debe obtener los datos que son enviados por la cámara para ello se crea una variable que contendrá a la cámara:

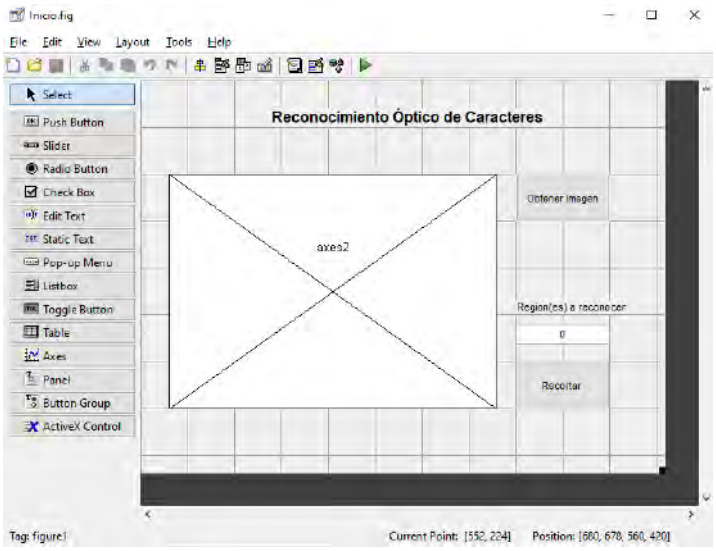


Figura 22: **Interface de selección de regiones (Vista de Diseño).**

Fuente: Elaboración propia. (2017)

Una vez que se reciba los datos de la cámara en “axes2” se procede a obtener una captura para su etapa de calibración, esto mediante el botón de “Obtener Imagen”:

Por último, se realiza la selección de la región o regiones que se quieran reconocer con el botón “Recortar”:

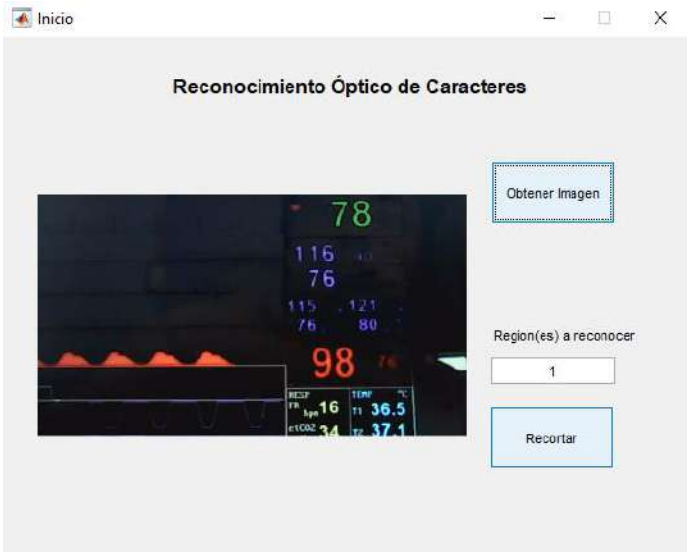


Figura 23: **Interface de selección de regiones (Vista de Usuario).**

Fuente: Elaboración propia. (2017)

El comando “ginput” permite seleccionar coordenadas de una imagen, en este caso se configuró para seleccionar el punto superior izquierdo, y el inferior derecho que

limitan a la región, puntos que se emplearán para recortar automáticamente dicha región cuando se inicie el bucle.

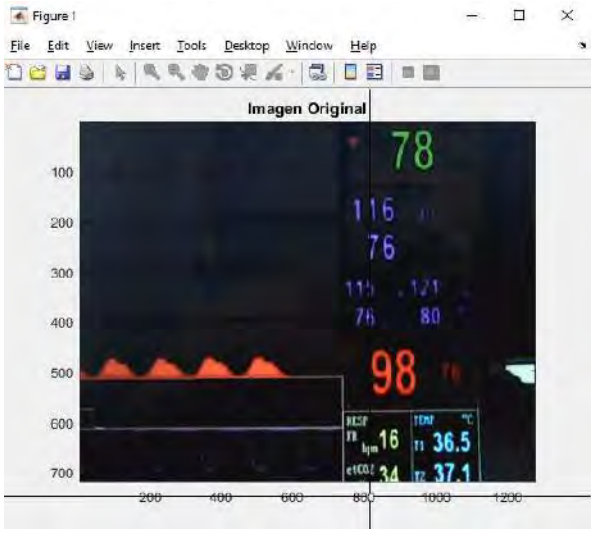


Figura 24: **Figura de selección de coordenadas de la región deseada.**

Fuente: Elaboración propia. (2018)

Una vez realizado el corte se entra en una nueva interface de usuario que es la etapa de calibración en la que se ajustan los parámetros para la correcta binarización de la(s) región(es) seleccionada(s)

3.1.2 Interface de calibración.

Esta interface busca obtener los valores adecuados para una correcta binarización siendo este un paso crucial para un correcto OCR, dentro de esta interface se define si el fondo de la región es “clara” u “oscura” además de ejecutar iteraciones de erosión para evitar que en la binarización los caracteres del parámetro se junten y tener un segmentado correcto de la imagen.

Esta interface muestra la imagen resultante del recorte hecho en el paso anterior que se mostrara en “axes1” y seleccionando las opciones de “claros” u “oscuro” se genera la imagen binarizada en “axes2”

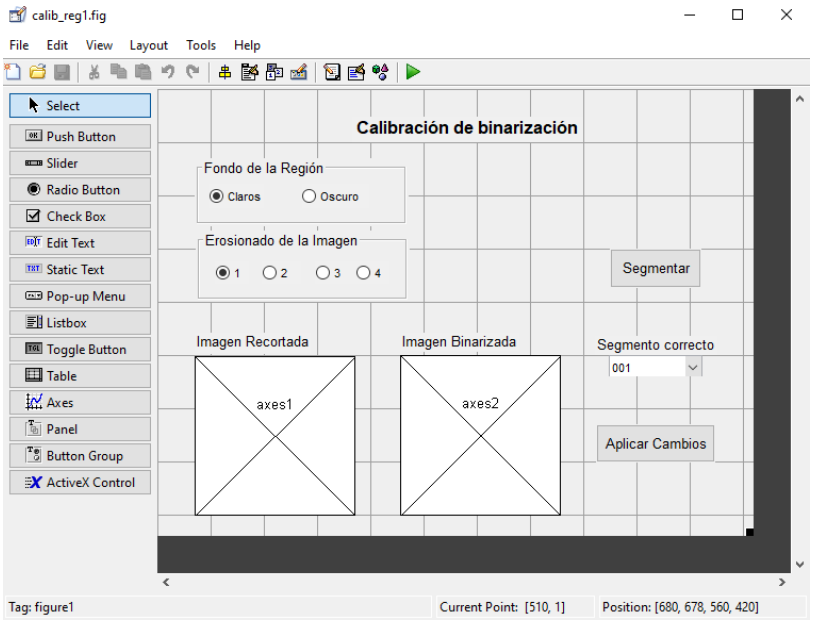


Figura 25: **Interface de selección de regiones (Vista de Diseño).**

Fuente: Elaboración propia. (2017)

Figura 26: **Calibración de una región (Vista de Usuario).**

Fuente: Elaboración propia. (2018)

El botón de segmentar, segmenta la región binarizada y abre la carpeta en la que se contiene los elementos segmentados para luego seleccionar el nombre de una región con un carácter correcto, esto guardará el tamaño que tiene un carácter para ser usado

en el recorte de imágenes cuando se inicie el proceso de reconocimiento de nuevas capturas.

Figura 27: **Carpeta abierta por el programa.**

Fuente: Elaboración propia. (2018)

Figura 28: **Selección del carácter según la carpeta abierta.**

Fuente: Elaboración propia. (2018)

Por último, esta interface permite guardar los cambios que se aplicaran en el bucle que se aplicará, para posteriormente abrir otra ventana para dar los ajustes finales

3.1.3 Interface de inicio del bucle.

Esta es una interface en las que podremos almacenar variables como: el nombre de las regiones seleccionadas, así como las unidades correspondientes a dichos datos.

Además, esta interface posee un botón que hace que inicialice el proceso de reconocimiento y almacenamiento de los valores dentro del ordenador. Para ello la interface creará un archivo .XLS en donde se almacenarán los valores obtenidos.

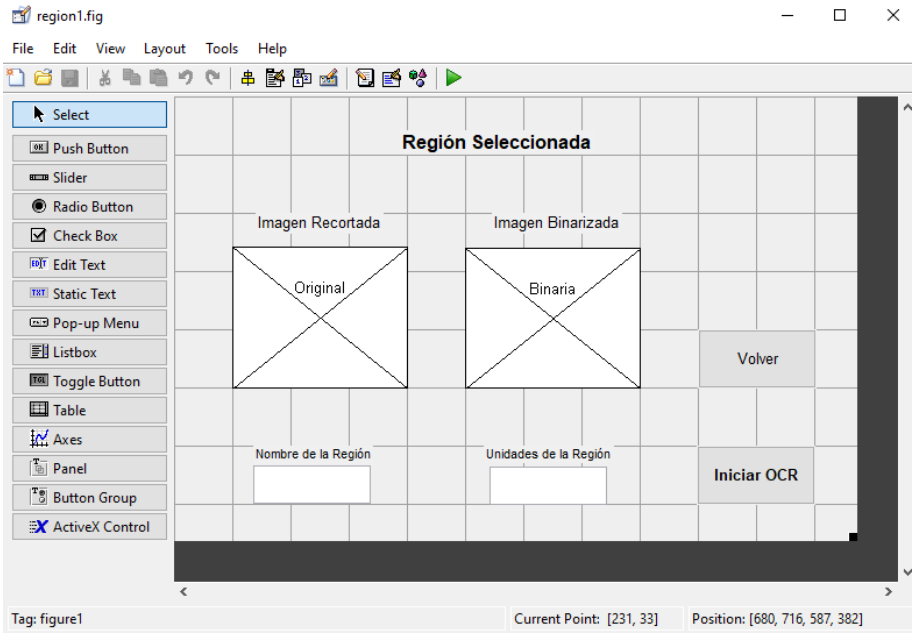


Figura 29: **Interface de Inicio.**

Fuente: Elaboración propia (2018).

Terminada la etapa inicial en que se guardan los datos invariables, se procede a iniciar el bucle de captación de imágenes y su procesamiento para el reconocimiento de los parámetros numéricos presentes en las regiones.

3.2 Diseño de la etapa de adquisición de imágenes.

Continuando el proceso, la captación de las imágenes es la primera parte del bucle de reconocimiento, por lo que se tendrá que lidiar con la comunicación de la cámara y el software Matlab para cumplir con los objetivos propuestos.

3.2.1 Selección de componentes.

Para la sección de adquisición de las imágenes se tomaron en cuenta los diferentes tipos de cámaras y su protocolo de comunicación con el ordenador. Para esto recurrimos a los objetivos específicos, en los que se señala que un sistema deba

reconocer y grabar parámetros de monitores además de que pueda interconectarse con varios equipos de medición en simultáneo. Por ello se comparó una cámara IP con una webcam

Cámaras Web	**Cámaras Ip**
Las cámaras web son como pequeñas cámaras digitales conectadas a una computadora. Puedes usarlas para tomar fotos y enviarlas a través de Internet, tanto a páginas web como a computadoras particulares. Estas usualmente se conectan mediante cable USB y son de tecnología "Plug and Play", que no requiere software extra, para su ejecución.	Las cámaras IP están pensadas para ser visualizadas mediante internet o desde una red local. Por ejemplo, si tengo una red de computadoras conectadas entre sí usando un switch o router, estas cámaras se unirían como si fuera otra computadora. Las cámaras más clásicas y profesionales tienen un enchufe Ethernet con una conexión RJ45 y se conectan al switch o router a través de un cable UTP. También hay cámaras IP inalámbricas que se conectan a la red WIFI a través de una antena.
Características. - Comunicación: Cableada. - Tipo de cable: USB - Estándar: USB 2.0 - Tipo de conexión: Directa	**Características.** - Comunicación: Inalámbrica, Cableada. - Tipo de cable: UTP. - Estándar: IEEE 802.3, IEEE 802.11. - Tipo de conexión: Directa.
Ventajas frente al sistema. - Económico. - De fácil conexión con Matlab. - Variedad de modelos. - La imagen es estable. - La imagen es Fiable. Buena Resolución.	**Ventajas frente al sistema.** - Algo económico. - De fácil instalación. - No requiere sistemas externos de almacenamiento. - Largo alcance de transmisión.
Desventajas frente al sistema.	**Desventajas frente al sistema.**

- No tiene gran alcance - Es de difícil reubicación. - Requiere de un sistema externo de almacenamiento.	- La imagen es propensa a interferencias (solo en modo inalámbrico) - Su conexión con Matlab varía de acuerdo al modelo de la cámara. - Imagen de calidad media.

Tabla 3: **Comparación entre cámaras IP y webcam.**

Fuente: Elaboración propia (2018).

Características del modelo de captación de parámetros numéricos.

- El sistema debe captar las imágenes de manera rápida.
- Las imágenes capturadas deben ser leídas por Matlab sin problemas.
- El sistema requerirá que la cámara sea de fácil conexión y transporte.
- El sistema debe ser capaz de admitir las imágenes de varias cámaras simultáneamente.
- Las imágenes no necesitan ser guardadas permanentemente.
- El sistema necesita reconocer que la cámara se encuentra bien colocada.
- Las imágenes obtenidas no requieren una gran resolución ya que la cámara captará las imágenes desde una posición fija y cercana al monitor esto para evitar la interferencia de las imágenes capturadas.

Con estos requerimientos se hace uso de una cámara Web Genius: FaceCam 1000x, que posee las siguientes características técnicas:

- Conexión directa USB 1.0 ó 2.0.
- Sensor de imagen CMOS de píxeles de alta definición 720p
- Resolución de 1MP, 1280 x 720, 640 x 480 pixeles
- Conexión Plug and Play.
- Formato de archivo: AVI/WMV
- Tipo de lente, objetivo de enfoque manual

- Micrófono.
- Dimensiones (A x A x P) 20 x 22 x 60 mm
- Peso: 50 g

3.2.2 Comunicación de la cámara con Matlab.

Esta cámara puede comunicarse de forma directa con el ordenador, mediante conexión USB, el decodificador que emplea Matlab para comunicarse con la cámara es del sistema operativo (Windows en este caso).

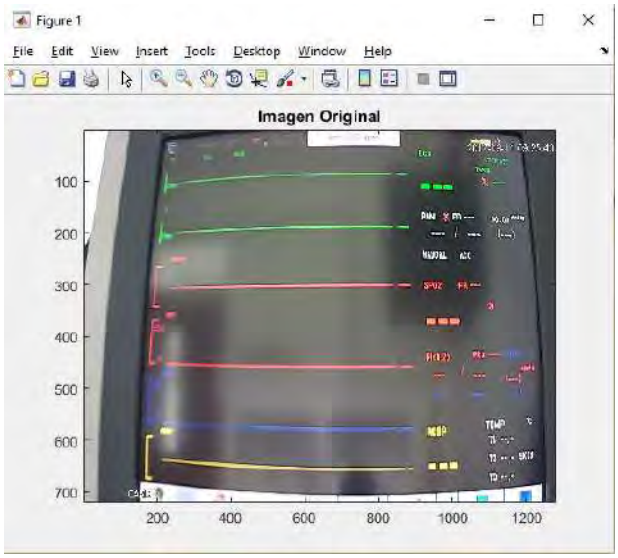


Figura 30: **Fotograma obtenido de la cámara Web.**

Fuente: Elaboración propia (2018).

Como se explicó en el punto 3.1 del diseño de la interface de usuario, el comando "ginput" permite seleccionar una coordenada y guardarla en una variable, esto nos servirá para delimitar las regiones que queramos analizar.

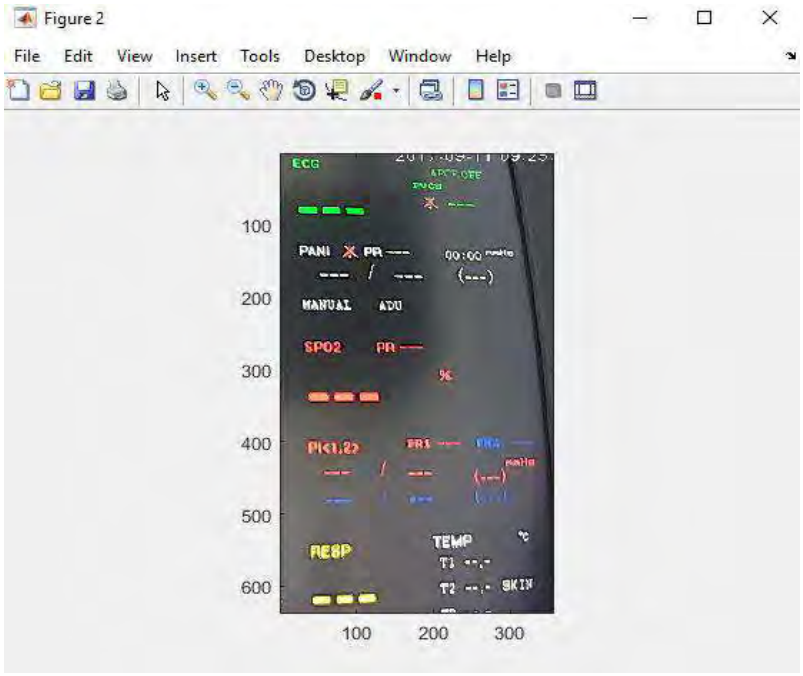


Figura 31: **Recorte de la imagen.**

Fuente: Elaboración propia (2017).

3.3 Diseño de la etapa de reconocimiento de caracteres (OCR).

El reconocimiento óptico de caracteres se divide en dos partes importantes, la primera y más importante, es la creación de la matriz de predicción del sistema que sacará las características de caracteres numéricos y los almacenará en una base de datos, que sirva de referencia para la comparación de nuevos caracteres. En el caso especial del método KNN, lo que se busca es extraer las características más resaltantes de cada número y que estas características sirvan para diferenciarlos unos de otros. Por otra parte, la segunda etapa consiste en el mismo reconocimiento de caracteres extraídos de imágenes de monitores de signos vitales, y poder determinar el carácter correspondiente, para ello también se requerirá procesar la nueva imagen por lo que en ambas etapas se seguirán los siguientes pasos.

La creación de la matriz de predicción se realiza previamente y de ella solo se requerirá la base de datos con las características obtenidas. Para dicho propósito se siguió el siguiente proceso:

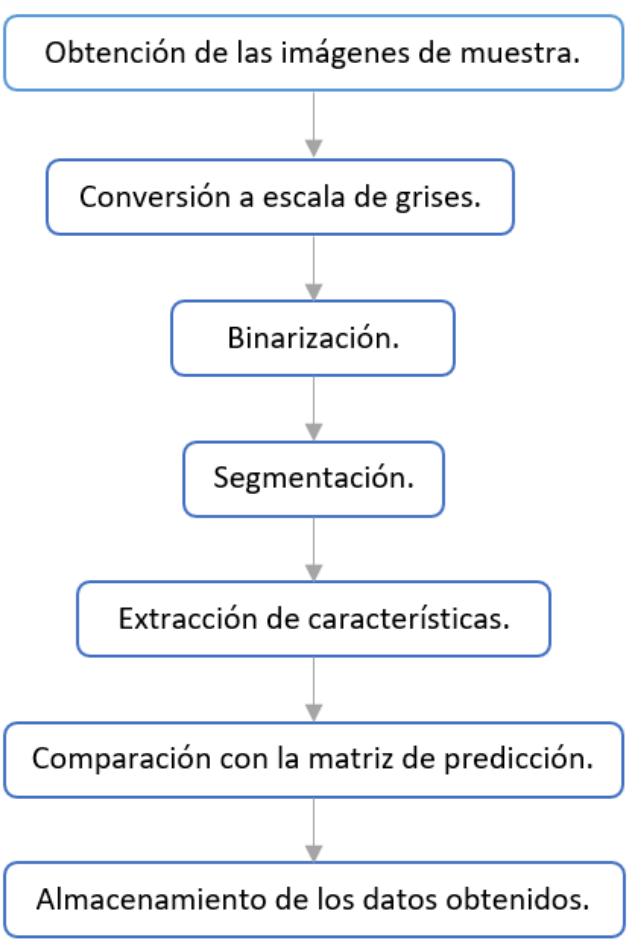


Figura 32: **Diagrama de flujo para la creación de la matriz de predicción.**

Fuente: Elaboración propia (2017).

Como lo indica la figura 32 el primer paso fue obtener imágenes de muestra, para ello se recurrió a la bibliografía referente a los monitores de signos vitales, para este caso, y se vio que la fuente o tipo de letra con las que son mostrados los caracteres varía en tres tipos específicos de fuentes: Arial, Helvética y, Trebuchet. Esto influye mucho en el reconocimiento de caracteres puesto que en algunas fuentes cambian la forma de ciertas letras o números drásticamente. Siguiendo el proceso, se ve como se asemeja al proceso de OCR descrito en el diagrama de flujo, por lo que se explicara con más detalle estos pasos más adelante. Quedando así una base de datos que llamaremos matriz de predicción.

La etapa de reconocimiento de nuevas imágenes se desarrolla inmediatamente después de capturar las imágenes por la cámara y recortar las regiones dadas en la interface de usuario. Así pues, se seguirá el siguiente proceso.

3.3.1 Pre-procesado de la imagen.

Las fotos tomadas por la cámara están sujetas a parámetros externos, como la luz de la habitación, la distancia de la cámara al monitor de señales vitales, y otras cosas que afectan cuánto se puede leer de los números que queremos. Para esto en esta sección se hará un pre-procesado a las imágenes obtenidas por la cámara además de tener condiciones iniciales en los que se deberá encontrar la cámara antes de poder tomar las muestras a ser reconocidas. Entre estas condiciones tenemos la cantidad de iluminación de la habitación en la que se tomara la imagen, la distancia de la cámara al monitor, y por último la posición de la cámara con respecto al monitor. Cumpliéndose en todo momento, que permitan el normal desempeño del equipo y de las personas que manejan el equipo.

Como se describió en el marco teórico Matlab tiene potentes herramientas para el procesado de imágenes para limpiar las imágenes de los brillos y reflejos de luces exteriores al monitor al que se desea extraer los parámetros numéricos, se creó un algoritmo que ajusta el contraste de la imagen luego de haber sido convertida a escala de grises, con esto la imagen queda lista para su binarización. Para ello emplearemos los comandos "RGB2GRAY" e "IMADJUST" siendo el primero para convertir la imagen a escala de grises y el segundo comando para ajustar el contraste de la imagen.

Recordemos que para RGB2GRAY se siguió la siguiente formula:

'0.2989 * R + 0.5870 * G + 0.1140 * B' [*]

Y para el caso de IMADJUST es un proceso de saturación de los valores extremos inferior y superior en un 1% siguiendo la siguiente formula:

'$i'_k = [CDF(i_K)] \times (L-1)$' [*]

[*] Formulas más detalladas en la sección de marco teórico pp. 26-27.

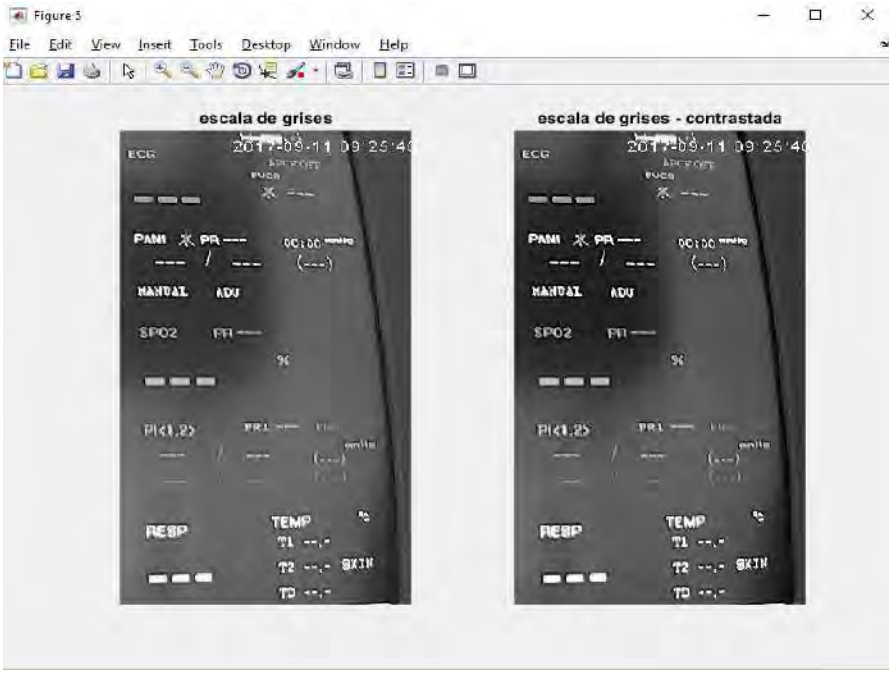


Figura 33: **Conversión a escala de grises y contraste.**

Fuente: Elaboración propia (2017).

3.3.2 Binarización.

El proceso de binarización no es más que volver la imagen en blanco y negro siendo las secciones blancas una matriz de "unos" y las secciones negras de "ceros".

Para esta parte se trabajó con una variación del comando "IMBINARIZE" que permite quitar los brillos de la pantalla y posteriormente binarizar la imagen, el comando calcula el umbral para seleccionar el dato de cada pixel con relación a los pixeles vecinos, usando estadísticas de cada pixel alrededor del punto a evaluar, además este método puede reconocer no solo brillos también lo hace con sombras en fondos blancos, pero estos no son tan útiles en este caso. Dependiendo del caso se emplea el modo directo del comando "IMBINARIZE" que emplea la Umbralización por el método de Otsu, Para ello es necesario recordar que dicho método aplica la siguiente formula:

$$'\sigma_{in}^2(t) = P_0(t).\sigma_0^2(t) + P_1(t).\sigma_1^2(t)' \text{ [*]}$$

[*] Esta ecuación se explica más a detalle en el marco teórico pp. 27-29

Si la imagen presenta brillos o sombras en regiones determinadas de la imagen podemos agregar el comando “ADAPTATIVE”, ya que el comando “IMBINARIZE” trae por defecto el modo “GLOBAL”; Esto quiere decir que se aplicara el método de Otsu por regiones que tengan características similares a su alrededor pero si estas cambian el umbral se calculará de nuevo para esta nueva región (que vendría a ser la región con un brillo o sombra en la imagen)

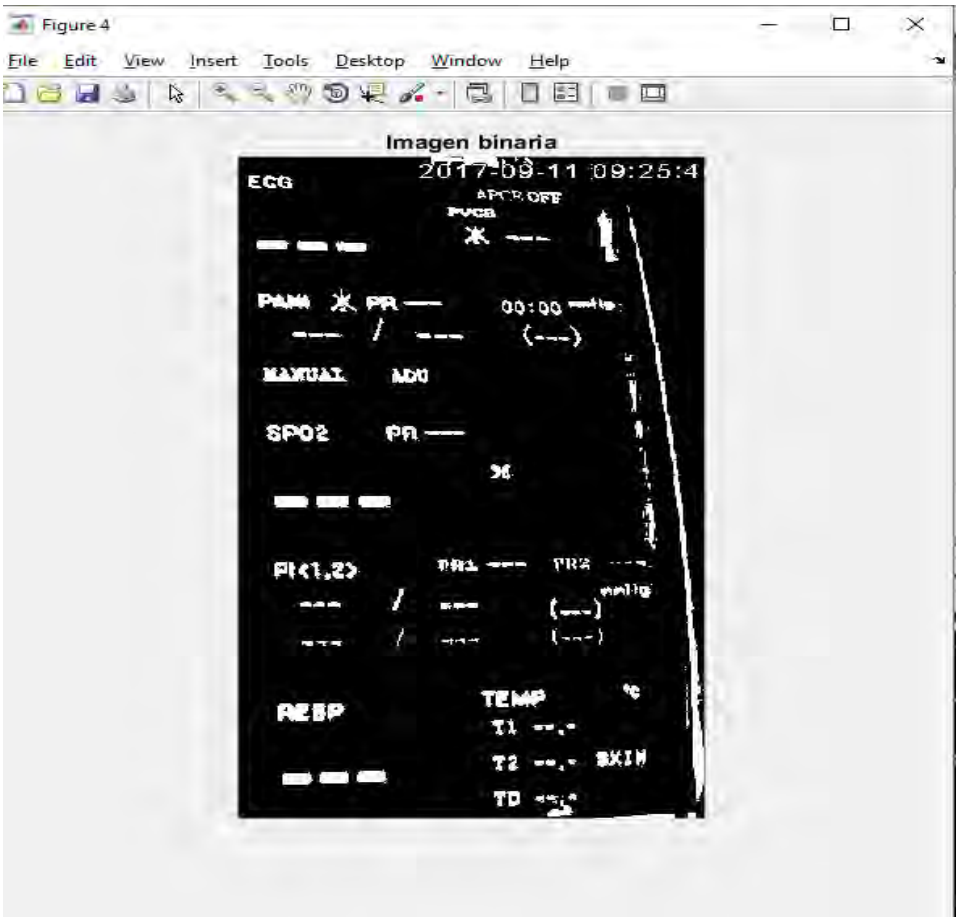


Figura 34: **Imagen binarizada.**

Fuente: Elaboración propia (2017).

3.3.3 Fragmentación o segmentación.

Este proceso abarca la identificación de objetos con ciertos criterios para su identificación como caracteres, en este caso dichos criterios se toman en cuenta, el tamaño de los objetos en pixeles los que darán umbrales que identificaran los objetos a ser reconocidos, para ello se estableció 4 umbrales, 2 para el ancho y 2 para el alto de

los objetos. Para el ancho se consideró el ancho mínimo que puede tener un número, así como el máximo del mismo, de igual manera para el alto. Se tomaron como referencia para todos los casos el tamaño de los parámetros numéricos obtenidos de las capturas por la cámara hacia el monitor desfibrilador del laboratorio de biomédica de la Escuela Profesional de Ingeniería Electrónica.

En el caso de la cámara empleada, esta tiene una resolución 1280 x 720 pixeles, con una distancia promedio de 50 cm. Desde el monitor hasta el lente de la cámara. Esto con la finalidad de tener la cámara cerca del monitor a ser evaluado y aprovechando que el tamaño de la cámara es pequeño, y no interrumpa con el funcionamiento normal del equipo a ser evaluado. Con esta disposición de los componentes los parámetros numéricos tienen un promedio de entre 22 a 45 pixeles de ancho dentro de la imagen captura por la cámara, considerando al más delgado al número "1" y al más ancho el número "5". Por otro lado, el promedio del alto de los caracteres se tiene entre 62 a 66 pixeles, como se puede observar los objetos fuera de estos umbrales no representan ningún parámetro numérico, y solo deja paso a algunos símbolos con estas dimensiones que serán removidos porque sus características difieren bastante a las de un parámetro numérico.

Para este fin Matlab nos permite segmentar la imagen gracias a comandos que leen las propiedades de los objetos dentro de la imagen general, para dicho propósito nos valdremos de las herramientas como la dilatación y erosión, con los que obtendremos una mejor forma a los objetos a ser evaluados. Para la dilatación emplearemos el comando "IMDILATE", para la erosión "IMERODE", y para la aplicación de una erosión seguida de una dilatación el comando "IMOPEN". Para ello se aplicará un factor de erosión, de dilatación y de apertura, denominado "SE" que se desarrolla mediante un elemento de estructuración morfológica que en Matlab viene

siendo el comando “STREL”, como se vio en el marco teórico, este elemento de estructuración puede tener diferentes formas, la que se usó para este caso fue la forma de disco “DISK” para todos los casos tanto en erosión y dilatación.

Para ello lo que es necesario recordar es que el comando ‘Strel’ coge los parámetros de forma y propiedades, y en caso de ‘Disk’ vendría a ser la forma y las propiedades que este toma vienen a ser el R y N con R como el radio del disco, y N las líneas de deformación. En el diseño del sistema solo se uso la propiedad de ‘Radio’. Para mayor detalle ver el marco teórico pp. 29-30.

Esto se hizo para poder segmentar de manera adecuada la imagen, para ello en primera instancia se dilato la imagen para borrar pixeles “huecos” dentro de los caracteres, para posteriormente abrir la imagen lo que termina de eliminar pixeles de “ruido”, que vendrían a ser un pixel diferente en una región dada, ya sea un “Cero” rodeado de una región de “Unos” o viceversa. Todo esto sin hacer más ancha la imagen de lo que ya se ha hecho por la dilatación previa. Y por último se realiza una erosión final para hacer los caracteres más delgados y definidos, además de separar posibles caracteres que se hayan juntado con las operaciones anteriores.

Figura 35: **Arreglos de erosión y dilatación.**

Fuente: Elaboración propia (2017).

Como se observa en la imagen dilatada de la figura 35, los círculos rojos nos muestran que a pesar de no haber valores registrados en dichas regiones aun el algoritmo es capaz de identificar las zonas dándoles prioridad sobre otros caracteres y formas dentro de la imagen, siguiendo con el procedimiento ahora queda el reconocimiento de los objetos dentro de dicha imagen que viene dada por el comando "REGIONPROPS", descrito anteriormente. Dicho comando nos da características variadas según la propiedad que se elija, para este caso se emplea el "BoundingBox" que encierra en rectángulos las regiones cerradas de "Unos", que serán los caracteres segmentados.

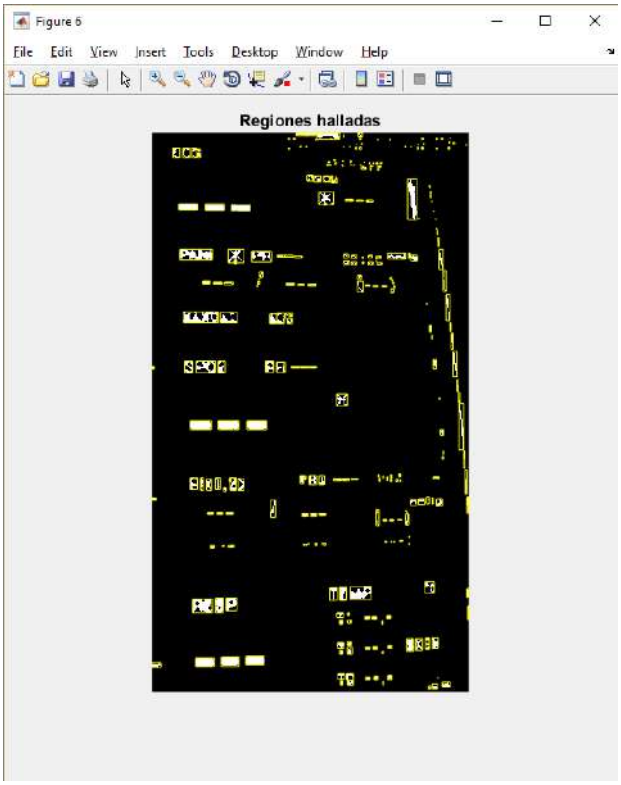


Figura 36: **Reconocimiento de objetos.**

Fuente: Elaboración propia (2017).

Las secciones amarillas indican los objetos encontrados en la imagen luego estos se guardan en una carpeta temporal, para nuestro caso particular emplearemos el monitor desfibrilador del laboratorio de biomédica y los caracteres numéricos encontrados se muestran en la figura 37, de la que se escogerán solo los que se consideran parámetros numéricos, siguiendo lo anterior expuesto se escogieron como umbrales el tamaño de los caracteres:

- 22 pixeles como ancho mínimo.
- 45 pixeles como ancho máximo.
- 26 pixeles como alto mínimo.
- 66 pixeles como alto máximo.

Figura 37: **Reconocimiento y selección de parámetros numéricos.**

Fuente: Elaboración propia. (2017)

3.3.4 Extracción de características.

Este proceso es propio de la creación de la matriz de predicción del sistema, aunque también es necesario en la etapa de reconocimiento. La extracción de características, como lo indica su nombre, busca cuantificar las características inherentes de cada carácter y de esta manera sacar un patrón común en caracteres iguales, y diferenciarlos con estos mismos de caracteres diferentes. (Mohammad, Anarase, Shingote, & Ghanwat, 2014) [23]

Para ello es necesario definir qué características se cuantificarán y formarán parte de la base de datos, para este propósito se nombrarán algunos de las características propias, así como comunes de los caracteres numéricos o caracteres en general.

- Relación del área de la imagen con respecto al área del carácter.
- Razón entre el número de filas y el número de columnas.
- El centroide del carácter en el eje “x” e “y”.
- La distancia del centro al centroide.
- La cantidad de agujeros o huecos del carácter.
- La cantidad de terminaciones del carácter.
- La orientación del carácter.

- La variación estándar de los pixeles con respecto al eje “x” e “y”.

Estas características muchas de ellas se pueden cuantificar, pero esto trae problemas ya que dicha cuantificación ocurre para características muy específicas como lo son el tamaño de los caracteres, así como la fuente en la que estén. Es por esto que no todas las características mencionadas son muy útiles al momento de caracterizar una imagen, puesto que dichos valores solo serán comunes para caracteres iguales del mismo tamaño y fuente. Entre las características que variaran con el tamaño y la fuente están el centroide, la orientación, la variación estándar que son características que al ser cuantificadas dependerán mucho del tamaño. Por otro lado, tenemos el número de terminaciones como una característica invariable con el cambio de fuentes.

Por esto se definirán como características a aquellas que sean invariables o porcentuales para evitar que el cambio de tamaño y fuente malogre el reconocimiento óptimo de caracteres futuros. Para evitar desechar a todas las características anteriores podemos hacer arreglos para que la cuantificación de algunas de dichas características dejen de ser números y sean porcentajes, por ejemplo el centroide en vez de ser expresado como un número exacto puede ser expresado como el porcentaje al que se encuentra el centroide en un determinado eje con respecto al tamaño total de dicho eje, de igual manera para el eje restante. Esto mismo se puede aplicar para la distancia del centro al centroide sabiendo que en porcentaje el centro siempre estará en el 50% de cada eje y conociendo ya el porcentaje de cada centroide la resta de estos nos dará la distancia deseada expresada en porcentaje con respecto al tamaño que posea dicha imagen.

Con respecto a los cambios de fuentes, solo se podría solventar teniendo varias bases de datos, cada una de ellas identificadas por el tipo de fuente de las que fueron

hechas, por ejemplo, si hay monitores que usen fuentes “Arial” tener una base de datos para Arial, si hay otra con fuente “Calibri” tener otra base de datos con fuente hecha para Calibri. Enfatizando las fuentes que tienen cambios notorios en la forma y distribución de los caracteres numéricos.

Tomando en cuenta todos estos detalles aún se tiene pocas características para caracterizar a estos caracteres, siendo hasta el momento unas 8 características, por ello también se vio la manera de caracterizar a la imagen no solo por valores conocidos exactos, y se encontró una solución.

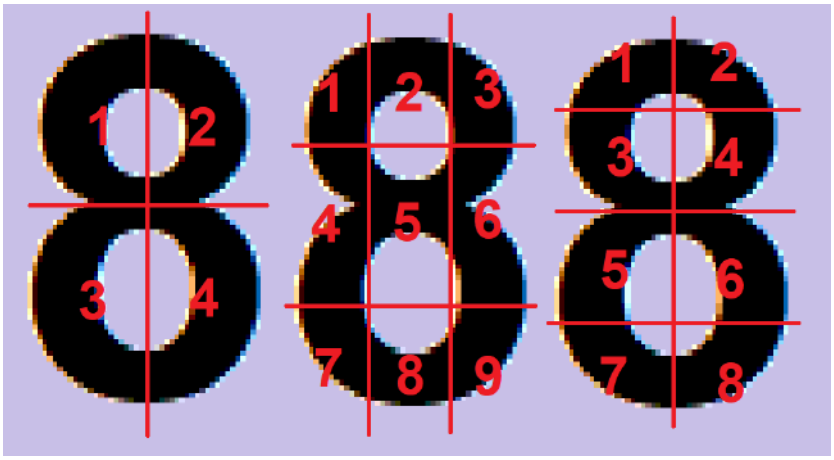


Figura 38: **División en segmentos del carácter numérico.**

Fuente: Elaboración propia (2017).

Al dividirse un carácter en partes más pequeñas se tienen regiones de dicha imagen que son aún más propias de cada carácter así que para caracterizar cada región se empleó la razón de área de la fracción del carácter con respecto al área de la región que está siendo evaluada dando un valor porcentual muy similar entre caracteres iguales, y muy diferenciados entre caracteres diferentes, solo se tuvo que definir cuantas regiones son necesarias para caracterizar correctamente a un carácter. Siendo 4 muy pocos y más de 10, demasiados. Se escogió el de 9 regiones puesto que nos da información de zonas muy específicas y detalladas sin caer en sobreestimación de características y dando las suficientes para caracterizar a la imagen.

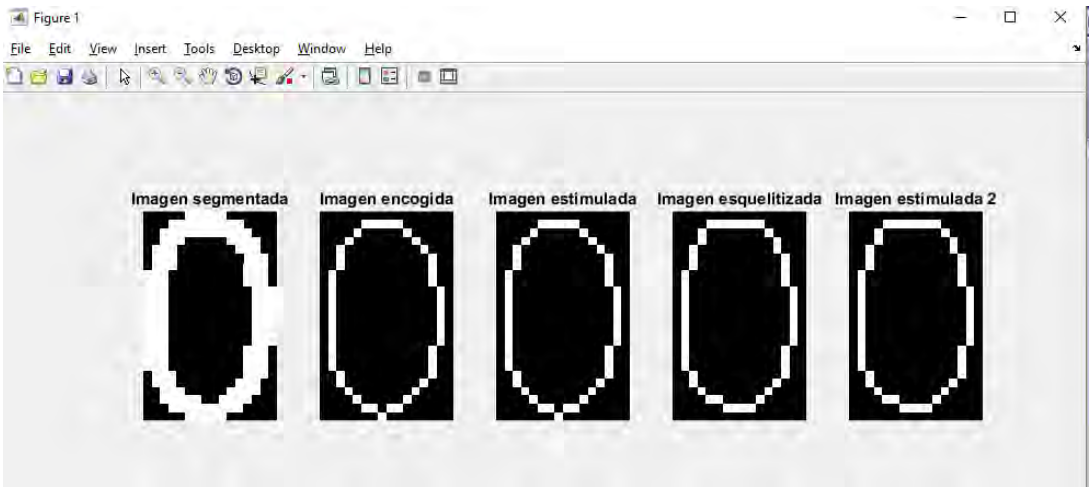


Figura 39: **Tratamiento de parámetros numéricos simétricos.**

Fuente: Elaboración propia (2017).

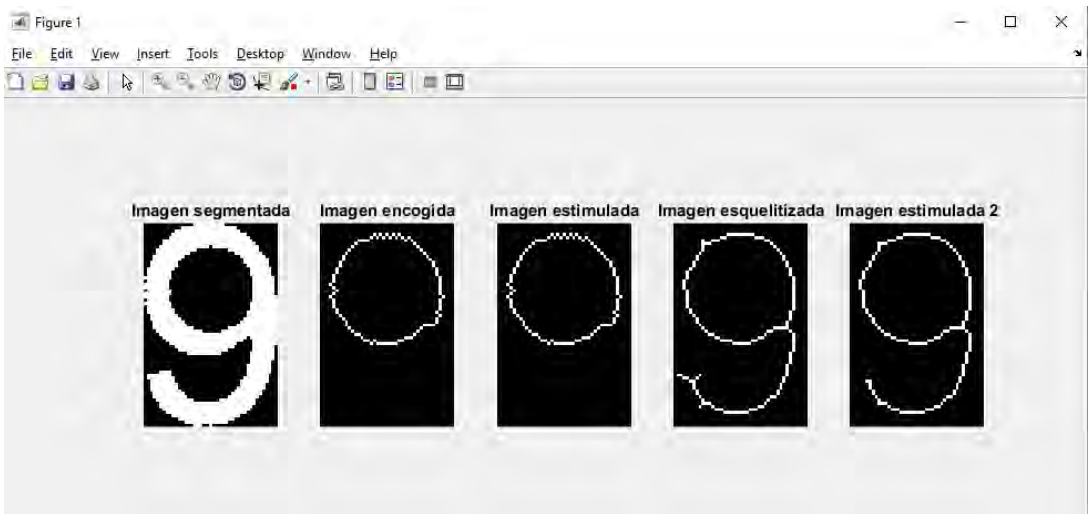


Figura 40: **Visualización de resultados entre caracteres numéricos.**

Fuente: Elaboración propia (2017).

De la figura 39 y 40 podemos observar que tratando correctamente los caracteres numéricos podemos hallar los huecos que posee cada carácter, así como la cantidad de terminaciones de otros ciertos parámetros numéricos. Para ello se recurrió al comando "BWMORPH" que nos permite esqueletizar una imagen, para ello se usa la propiedad "SKEL" y como resultado se tiene la cuarta imagen de las figuras 38 y 98. También nos permite mostrar regiones encerradas o unidas entre otras muy útiles que nos darán como resultado los valores numéricos que se buscaban en un inicio de una manera sencilla y práctica. Por ejemplo para encontrar los agujeros que tenía cada carácter se empleó la

operación “SHRINK” que reduce a un punto un objeto solido si este no tiene ningún agujero o lo transforma en un anillo que contiene al agujero, esto se puede observar en la segunda imagen de las figuras 38 y 39. Por otro lado la estimulación de la imagen es la eliminación de pixeles de las terminaciones que presente una figura esqueletada, y como se puede observar si es una sección cerrada no hará modificaciones sin embargo si esta posee pixeles en las terminaciones se eliminaran los pixeles que el usuario determine, en el caso de las terceras y quintas imágenes de la figura 39 y 40 se empleó el comando “SPUR” con un valor de 3 pixeles a eliminar. Las terminaciones de la imagen en la que se ve un “9” esquelitizado son las esquinas que posee el carácter como bloque y estos pueden variar según el tipo de fuente que se esté utilizando.

Para entender mejor estos comandos, se revisará los algoritmos que van detrás de ellos, para el primer caso, en la operación ‘Skel’ hace una serie de iteraciones que van eliminando pixeles de una imagen binaria siguiendo una serie de condiciones en cada iteración hasta quedar una imagen esqueletizada. Estas condiciones por iteración son las siguientes.

- ‘En la primera subiteración, elimina el píxel p solamente si se cumplen todas las condiciones G_1, G_2 y G_3.’
- ‘En la segunda subiteración, elimina el píxel p solamente si se cumplen todas las condiciones G_1, G_2 y G_3'.’

Condición G_1.

‘$X_H(p) = 1$’ donde

‘$X_H(p) = \sum_{i=1}^{4} b_i$’ con

‘$b_i = \begin{cases} 1, si\ x_{2i-1} = 0\ y\ (x_{2i} = 1\ o\ x_{2i+1} = 1) \\ 0,\ en\ cualquier\ otro\ caso \end{cases}$,

x1, x2 …, x8 son los valores de los 8 entornos de p, empezando por el entorno del este y enumerados en sentido contrario a las agujas del reloj.

Condición G_2.

‘$2 \leq min\{n_1(p), n_2(p)\} \leq 3$’ donde,

‘$n_1(p) = \sum_{k=1}^{4} x_{2k-1} \vee x_{2k}$’ y ‘$n_2(p) = \sum_{k=1}^{4} x_{2k} \vee x_{2k+1}$’

Condición G_3.

‘$(x_2 \vee x_3 \vee \overline{x_8}) \wedge x_1 = 0$’

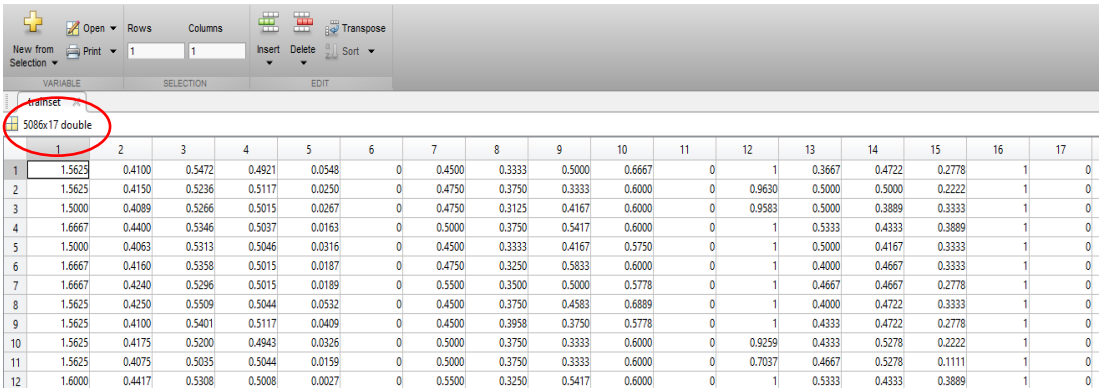
Condición G_3’.

‘$(x_6 \vee x_7 \vee \overline{x_4}) \wedge x_5 = 0$’

Para el comando ‘Shrink’ el proceso es similar solo que la eliminación de pixeles es mas acelerada con cada iteración. Por lo que tienen usos diferentes como se mencionó párrafos atrás.

Para entender mejor el comando ‘spur’ se expuso de mejor manera en el marco teórico p. 35.

Con esto tenemos un total de 17 características a ser cuantificadas y guardadas en una base datos para la matriz de predicción. Por otra parte, para la etapa de reconocimiento si bien es cierto se tendrá que extraer las 17 características de estos nuevos caracteres, no será necesario guardas dichas características.



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	1.5625	0.4100	0.5472	0.4921	0.0548	0	0.4500	0.3333	0.5000	0.6667	0	1	0.3667	0.4722	0.2778	1	0
2	1.5625	0.4150	0.5236	0.5117	0.0250	0	0.4750	0.3750	0.3333	0.6000	0	0.9630	0.5000	0.5000	0.2222	1	0
3	1.5000	0.4089	0.5266	0.5015	0.0267	0	0.4750	0.3125	0.4167	0.6000	0	0.9583	0.5000	0.3889	0.3333	1	0
4	1.6667	0.4400	0.5346	0.5037	0.0163	0	0.5000	0.3750	0.5417	0.6000	0	1	0.5333	0.4333	0.3889	1	0
5	1.5000	0.4063	0.5313	0.5046	0.0316	0	0.4500	0.3333	0.4167	0.5750	0	1	0.5000	0.4167	0.3333	1	0
6	1.6667	0.4160	0.5358	0.5015	0.0187	0	0.4750	0.3250	0.5833	0.6000	0	1	0.4000	0.4667	0.3333	1	0
7	1.6667	0.4240	0.5296	0.5015	0.0189	0	0.5500	0.3500	0.5000	0.5778	0	1	0.4667	0.4667	0.2778	1	0
8	1.5625	0.4250	0.5509	0.5044	0.0532	0	0.4500	0.3750	0.4583	0.6889	0	1	0.4000	0.4722	0.3333	1	0
9	1.5625	0.4100	0.5401	0.5117	0.0409	0	0.4500	0.3958	0.3750	0.5778	0	1	0.4333	0.4722	0.2778	1	0
10	1.5625	0.4175	0.5200	0.4943	0.0326	0	0.5000	0.3750	0.3333	0.6000	0	0.9259	0.4333	0.5278	0.2222	1	0
11	1.5625	0.4075	0.5035	0.5044	0.0159	0	0.5000	0.3750	0.3333	0.6000	0	0.7037	0.4667	0.5278	0.1111	1	0
12	1.6000	0.4417	0.5308	0.5008	0.0027	0	0.5500	0.3250	0.5417	0.6000	0	1	0.5333	0.4333	0.3889	1	0

Figura 41: **Base de datos generada con la caracterización de los caracteres.**

Fuente: Matlab – Elaboración propia (2017).

Como se puede observar la matriz de predicción tuvo como muestras a 5086 caracteres considerando solo a los números, y de cada uno de ellos se sacó 17 características teniendo el siguiente orden

1. Razón nro. Filas sobre nro. Columnas.
2. Razón área de Carácter sobre área de la imagen.
3. Porcentaje centroide X.
4. Porcentaje centroide Y.
5. Porcentaje distancia del centro al centroide.
6. Nro. Euler: Nro. de objetos – nro. De agujeros en los objetos.
7. Las 9 razones de área de letra sobre área de imagen, por bloques en el orden tal y como se muestra en la Figura “37”.
16. Nro. De huecos.
17. Nro. De terminaciones.

3.3.5 Comparación con patrones.

Esta última etapa del OCR no es más que el propósito del reconocimiento y es la comparación de las características obtenidas de nuevas imágenes con la base de datos actual, y determinar el carácter numérico correspondiente, para ello Matlab facilita las herramientas necesarias para cargar la matriz de predicción y aplicar el método KNN, que no es más que buscar los “K” vecinos (Neighbours) más cercanos (Nearest), donde K viene definido por el usuario. Este se logra mediante el comando “FITCKNN”: que permite poner de entradas la matriz de predicción con las características de las letras y sus respectivas “etiquetas” o nombres de cada carácter. Para ello se requiere cargar (“LOAD” en MatLab) la matriz de predicción, y desarrollar el procesamiento de las nuevas imágenes obtenidas para poder extraer sus características, al conseguir dichas características de estas nuevas imágenes se compararán los valores con la base de datos

que tenemos y los K valores más cercanos determinarán el valor de los caracteres numéricos reconocidos.

Para entender mejor este comando lo desglosaremos en partes, para empezar el algoritmo clasificador kNN tiene 3 partes.

Primero como se vio en el capitulo 2.2.7 el algoritmo tiene un grupo de entrenamiento en el que se encuentran las características a los cuales llamamos predictores y sus clasificadores, en la etapa de diseño sacamos que las características del grupo de entrenamiento serian 17, por lo que nuestra matriz de entrenamiento tiene 17 dimensiones, cada una con su clasificador, en palabras sencillas, la letra ‘a’ tiene un vector de 17 dimensiones como características. La letra ‘a’ será nuestro clasificador, y el vector muestra un elemento de nuestra matriz de entrenamiento, así para cada letra, y para cada muestra de una misma letra.

Siguiendo la siguiente sintaxis:

Mdl = fitcknn(Tbl,ResponseVarName);

Siendo ‘Tb1’ nuestro predictor y ResponseVarName nuestro clasificador.

Esta sintaxis se puede modificar para definir la cantidad de K vecinos mas cercanos que queremos que el modelo evalúe y el tipo de métrica de las distancias.

Tenemos el siguiente ejemplo:

Mdl = fitcknn(X,Y,'NumNeighbors',5,'Standardize',1)

Donde:

Clasificador = Y.

Predictor = X, siendo esta una matriz de características.

Numero de Vecinos = 5.

Estandarizado = si (Siendo un valor lógico de ‘0’ o ‘1’)

Métrica de distancia = Euclidiana (Dada por defecto si no se especifica).

Para nuestro caso queda de la siguiente manera:

Model = fitcknn (trainset,classname)

Model.numneighbors = 10;

Como segundo paso, se describió que el valor de K más optimo se define por la proporción de muestras definida por $k(N_i) = cte\sqrt{N_i}$; y $\hat{p}(\frac{X}{w_i})$; [*]

Teniendo como entrenamiento una poco más de 5000 muestras en total y la cantidad por carácter numérico era de aproximadamente 330, entonces p ≈ 15 y k ≈ cte $\sqrt{15}$, siendo k ≈ 3, esto quiere decir que cualquier valor de k por encima de 3 nos dará un acercamiento óptimo al valor que deseamos obtener.

Ahora como último paso queda crear un sistema de clasificación utilizando al 'modelo' como algoritmo de predicción de nuevos datos que vienen dado por nuestros datos de la fase de test.

Para ello, se empleó el comando 'predict'.

Y = predict(model,NewImages);

Pero para que pueda predecir correctamente la entrada de la variable NewImages no puede ser un archivo de imagen, si no que debemos sacar sus 17 caracteristicas, para compararlas con nuestro modelo y asi el algoritmo knn pueda funcionar, por lo que:

Features= getFeatures(Newimages);

Y=predict(model,Features);

Siendo getFeatures un algoritmo de desarrollo propio para poder aplicar el procesamiento digital de imágenes y así extraer sus características mencionadas en el diagrama de flujo de la figura 32.

[*] El cálculo de las distancias y la formula del modelo se detallaron más a detalle en el marco teórico pp. 43-49.

3.4 Diseño del sistema de clasificación de datos.

Hasta este punto ya se entrenó al sistema, y es capaz de reconocer los caracteres correctamente con ligeros errores en la selección de las imágenes a ser caracterizadas, pero este proceso solo reconoce los caracteres de forma individual por ello es necesario implementar un sistema que permita reconocer que caracteres reconocidos son relevantes y que otros no, para ello se seguirá un proceso similar al de eliminación de regiones de la etapa del segmentado.

3.4.1 Reconocimiento de datos.

En esta etapa requeriremos definir los parámetros numéricos a ser reconocidos, es decir para el caso especial de los monitores de funciones vitales, estos suelen tener de entre 4 a 8 parámetros capaces a ser medidos, así que para ello necesitaremos definir que partes del monitor nos revelan dichos valores, para ello se pondran unas condiciones iniciales.

Primero conocer con antelación que parámetros numéricos se guardarán en la base de datos; segundo, su ubicación en el monitor de funciones vitales, y por último el tiempo con el que estos serán censados y así ser reconocidos. Estos datos serán necesarios antes del reconocimiento OCR para poder llevar una base de datos óptima y ordenada.

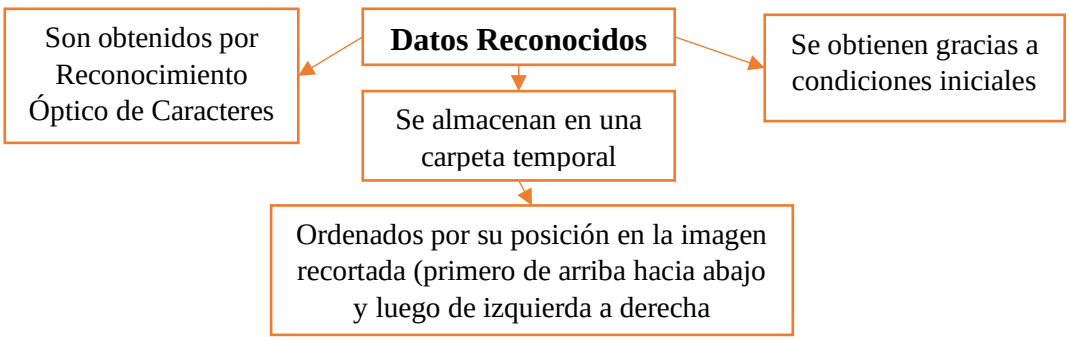


Figura 42: **Diagrama de reconstrucción de los parámetros numéricos.**

Fuente: Elaboración propia (2017).

3.4.2 Esquema de clasificación.

Como se mencionó anteriormente se requieren de condiciones iniciales, de ellas podremos extraer las regiones en las que se encuentran los parámetros numéricos, para este propósito emplearemos la herramienta de Matlab "GINPUT", explicada anteriormente.

"GINPUT" permite guardar en una variable los valores de las coordenadas en los que el cursor haga click, esto será de utilidad puesto que como las condiciones iniciales indican, se debe conocer los parámetros y su ubicación en el monitor, con este comando se podrá guardar las regiones a ser reconocidas. Una vez hecho esto se procede a evaluar las dimensiones de las regiones encontradas en el proceso de segmentación, esto para determinar los límites que delimitan y definen a los caracteres, además de servir para el algoritmo de clasificación.

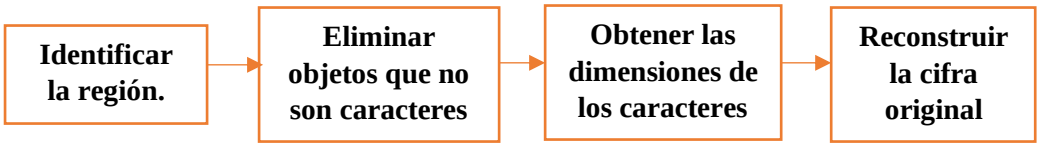


Figura 43: **Diagrama de flujo del proceso de reconstrucción.**

Fuente: Elaboración propia (2017).

Es decir que emplearemos el tamaño de los caracteres para reconstruir la cifra original mostrada en la imagen obtenida por la cámara, para ello se empleara el hecho de que una cifra numérica tienen la misma dimensión de la altura, o al menos esta es prácticamente la misma entre caracteres pertenecientes a una misma cifra numérica, valiéndonos de este principio emplearemos la desviación estándar en las dimensiones de la altura de cada carácter y si este valor es menor o igual a 1 serán consideradas cifras de un mismo número, reconstruyéndola a su forma original.

También es posible utilizar su posición para definir si un parámetro numérico está compuesto por más de una cifra acompañado con el tamaño de los caracteres son un fuerte proceso de reconstrucción del parámetro numérico.

3.5 Diseño del sistema de almacenamiento de datos.

3.5.1 Almacenamiento de los parámetros numéricos.

Ya reconocidos los números como tales queda ser almacenados en archivos de fácil acceso y organización, para ello se guardará los datos en archivos Excel que almacenarán los datos con el nombre de la región y el libro con la unidad de medida del parámetro numérico escogido.

Para ello el nombre de la región, así como la unidad de medida deberán ser datos ingresados por usuario y almacenados en variables para su posterior uso en el sistema de almacenamiento. Esto fue descrito en el diseño de la interface de usuario es allí donde se ha creado dicho archivo Excel.

Todos estos archivos se crearán en una carpeta asignada para la base de datos en el disco duro del ordenador que está siendo empleado.

3.5.2 Almacenamiento de características de los nuevos caracteres.

Así como se guardó las características de la matriz de predicción con todas las características que definen a los números, también se puede salvar la información de las características de los nuevos caracteres reconocidos en nuevas imágenes, esto con el fin de detectar posibles fallos en el reconocimiento de los caracteres e incorporar nuevos datos para la matriz de predicción. Teniendo así un sistema que pueda ir aprendiendo mediante su operación en campo, cabe resaltar que el sistema diseñado solo guarda dichos valores mas no los implementa dentro de la matriz de predicción, por lo que el sistema de aprendizaje solo es una aplicación mencionada, mas no implementada.

3.6 Diseño del proceso experimental – plan de mediciones.

Tras el diseño del trabajo de medición y reconocimiento óptico de caracteres es necesario medir los resultados en diversos escenarios que nos permitan evaluar adecuadamente el rendimiento del sistema y así dar corrección a los errores que se puedan hallar.

Para ello dividiremos los experimentos en: condiciones ideales, alterando el ambiente (iluminación), y variando la configuración de nuestro equipo de captación (tiempo de obturación del lente de nuestra cámara).

En condiciones ideales se pretende medir los resultados del sistema y compararlos con medición directa (toma de datos manual), se definirá como condiciones ideales a que las siguientes condiciones no afecten a la toma de datos del sistema, tales como: buena iluminación del laboratorio, ángulo adecuado de la cámara para que no forme reflejos, y configuración por defecto de la cámara web.

Luego se verá la respuesta del sistema si variamos los niveles de iluminación eliminándola por completo o sobre exponiéndola a mucha luz.

Por último, para ver el desempeño del sistema en otras condiciones se cambiará el tiempo apertura del obturador de la cámara para medir el grado de enfoque y desenfoque cambiando estos parámetros de la cámara.

Para todos los casos se evaluará el desempeño del sistema con captación de una y dos regiones del monitor con un tiempo de muestreo de 5 segundos entre capturas estimando el tiempo total de los experimentos entre 1min 30 seg. A 5 min (18 a 60 muestras). Una vez obtenidos los resultados se estudiará los errores obtenidos y si se puede diseñar una etapa de control de las mismas.

CAPÍTULO 4: RESULTADOS.

4 RESULTADOS.

Este capítulo recopilará los resultados obtenidos por el diseño planteado en el capítulo anterior. Se mostrará el proceso y los resultados de las partes mencionadas por el diagrama de flujo de la figura 21.

Teniendo en consideración las limitaciones planteadas en el capítulo 1, este sería la ficha técnica del proyecto:

Hardware:	Laptop Toshiba Satellite core I5 de 2.6GHz.
	Cámara Web Genius “FaceCam 1000x”.
Software:	MatLab r2017a.
Alcance:	La cámara se ubicó a una distancia de 30cm a 1m del monitor.
Captación:	El sistema capta hasta un máximo de dos regiones simultáneamente.
Tiempo de respuesta:	Intervalo entre capturas 2 seg (1 región) / 4 seg (2 regiones)
Monitores:	Pantallas blancas y caracteres de color.
	Pantallas negras y caracteres de color.
Fuente:	Arial, Helvetica y Trebuchet
Tamaño	Superiores al tamaño de fuente n° 18.
Iluminación	Luz Blanca.
Caracteres:	Numéricos.
Figuras:	No.

4.1. Resultados en una sola región.

El tiempo de prueba fue de 1 minuto 43 segundos, tomando una muestra o fotograma cada 5 segundos, además de que las condiciones para la toma de imágenes de ese instante fueron:

- Hora: 6:20 p.m. cielo nocturno.
- Habitación iluminada con luz blanca.
- Distancia de la cámara al monitor: 40 a 50 cm.
- Cámara fija, sujetada por brazo de apoyo.

Estas condiciones influirán en el brillo de la imagen tomada, así como los reflejos producidos dentro de la pantalla, que deben ser correctamente procesados por el sistema, además de que la posición y la distancia de la cámara con respecto al monitor, influye en el tamaño y la inclinación de los caracteres a ser reconocidos, por lo que se deben tomar en cuenta.

Figura 44: **Interface de usuario menú inicial.**

Fuente: Elaboración propia (2017).

Como se puede observar en la figura 44 el proyecto inicia con una interface de usuario que permite visualizar imágenes provenientes de la cámara, para tener un mejor enfoque de la región de donde se digitalizará los caracteres numéricos, se coloca la cantidad de regiones a reconocer, para este caso se pondrá “1”, y se aprieta el botón “Recortar”. Una

vez determinada la posición a ser reconocida, se recorta la imagen como muestra la figura 45. En el recorte se puede ver como son capturados otros caracteres que no son de interés, los cuales serán removidos automáticamente en el proceso, también se observa que por la ubicación de la cámara y el monitor esta imagen puede estar ligeramente inclinada, pero esto está previsto y no genera mayor repercusión en el reconocimiento del carácter numérico.

Figura 45: **Región recortada de la imagen obtenida.**

Fuente: Elaboración propia (2017).

Figura 46: **Tratamiento de imagen – De forma interna.**

Fuente: Elaboración propia (2017).

Tras el recorte de la imagen obtenida por la cámara, se procede al procesamiento de la imagen, siendo el proceso de binarización el que más problemas presenta, puesto que los diferentes brillos hacen que aparezcan regiones en los caracteres, que no les corresponden, como manchas blancas fuera del carácter o espacios negros dentro del carácter numérico

dificultando su procesamiento, esto debido al brillo de la pantalla y el generado por la iluminación de la habitación en la que se encuentre el monitor a ser examinado siendo más difícil de tratar la iluminación de la luz natural del día. Como se puede observar, en la figura 46, la imagen presenta como una diferencia de intensidad en los bordes del carácter con respecto a su centro esto genera que no se binarice correctamente, para ello se tuvo que ajustar parámetros de binarización para que se obtenga una imagen correctamente binarizada. Estos parámetros están definidos por Matlab y son "el factor de erosión y de dilatación" la buena combinación de estos da como resultado la tercera imagen de la figura 47.

Figura 47: **Tratamiento de la imagen – Erosión y dilatación de la región recortada.**

Fuente: Elaboración propia (2017).

Figura 48: **Interface de Calibración.**

Fuente: Elaboración propia (2017).

El proceso de binarización se realiza de forma interna, pero podemos modificar algunos parámetros como lo son el fondo de la imagen, de donde podemos escoger entre "claros" u "oscuros, la cantidad de erosiones que se realizan a la imagen, todo esto para obtener un correcto binarizado, la figura 48, muestra la interface de calibración, con la

imagen binaria obtenida tras seleccionar el fondo y la cantidad de erosiones a la imagen. Luego de obtener una imagen binarizada correcta, se procede a segmentar la región apretando el botón con dicho nombre, este proceso nos abrirá una carpeta con los caracteres segmentados, cada segmento tiene un nombre numérico del "001" hasta el "n-simo" que se encuentre en la región.

Figura 49: **Caracteres tratados para ser reconocidos.**

Fuente: Elaboración propia (2017).

Por último, la interface permite seleccionar a un segmento correcto, pudiendo elegir cualquiera de los segmentos correctos, esta selección sirve para obtener las dimensiones de un segmento correcto y usarlas posteriormente.

Para terminar el proceso de calibración se presiona el botón de "aplicar cambios" con lo que nos queda la figura 50.

Figura 50: **Interface de región seleccionada.**

Fuente: Elaboración propia (2017).

En esta interface se coloca el nombre de la región que se va a reconocer y las unidades en las que fue medida, también podemos volver a la interface de inicio si se cometió algún error en el recorte o el tratamiento de la imagen, caso contrario podemos dar inicio al proceso de OCR, que inicia el bucle de reconocimiento, y que usará los datos obtenidos y guardados durante los procesos de selección de región y calibración.

Figura 51: **Interface del bucle de reconocimiento.**

Fuente: Elaboración propia (2017).

Una vez iniciado el reconocimiento, se abrirá una interface, figura 51, que irá mostrando las imágenes captadas por la cámara, así como el parámetro reconocido con el nombre que se le haya asignado y sus respectivas unidades.

Los datos recogidos por el bucle, son guardados en el mismo archivo de Excel como se muestra en la figura 52.

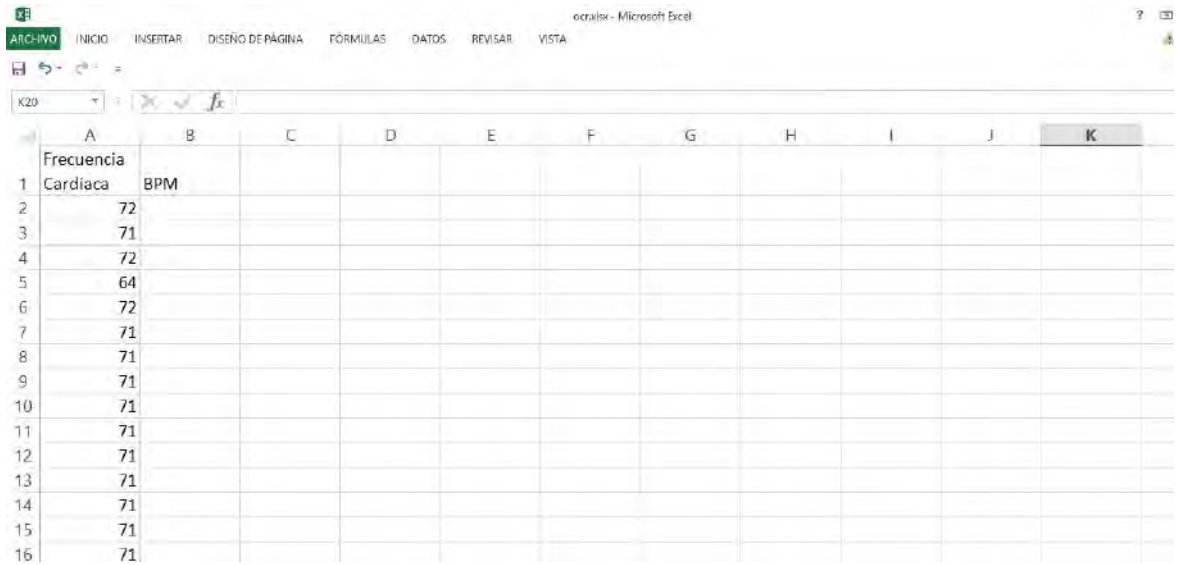


Figura 52: **Acuñado de los datos de las demás imágenes obtenidas hasta su detención.**

Fuente: Elaboración propia (2017).

Los datos observados en la figura 52, son los obtenidos por OCR, es decir el sistema planteado, pero para corroborar dicho proceso se guardó el video y las capturas de la grabación, que servirán para compararlos con los datos obtenidos por el sistema propuesto, obteniendo como resultado la siguiente tabla:

Ritmo cardiaco (HR [Bpm])		Ritmo cardiaco (HR [Bpm])		Ritmo cardiaco (HR [Bpm])	
Persona	**OCR**	**Persona**	**OCR**	**Persona**	**OCR**
119	*119*	97	*97*	71	*71*
120	*120*	107	*107*	71	*71*
118	*118*	97	*97*	71	*71*
119	*119*	97	*97*	71	*71*
120	*120*	118	*118*	71	*71*
97	*97*	102	*102*	71	*71*
89	*89*	87	*87*	71	*71*
76	*76*	87	*87*	71	*71*
97	*97*	77	*77*	71	*71*
97	*97*	73	*73*	71	*71*
119	*119*	72	*72*	71	*71*
120	*120*	71	*71*	71	*71*
106	*106*	71	*71*	71	*71*
107	*107*	65	*65*	71	*71*
120	*120*	64	*64*	71	*71*
119	*119*	64	*64*	71	*71*
118	*118*	72	*72*	71	*71*
118	*118*	72	*72*	71	*71*
98	*88*	72	*72*	71	*71*

89	*89*	71	*71*	71	*71*
82	*82*	71	*71*	58	*58*
82	*82*	71	*71*	58	*58*
97	*97*	71	*71*	58	*58*
82	*82*	64	*64*	53	*53*
82	*82*	64	*64*	64	*64*
97	*97*	64	*64*	64	*64*
107	*107*	64	*84*	58	*58*
107	*107*	71	*71*	58	*58*
106	*106*	71	*71*	64	*64*
107	*107*	71	*71*	64	*64*
119	*119*	72	*72*	71	*71*
89	*89*	72	*72*	72	*72*
89	*89*	72	*72*	72	*72*
97	*97*				

Tabla 4: **Contrastación de los valores obtenidos (1 Región).**

Fuente: Elaboración propia (2018)

Ahora se comparará los valores obtenidos por el sistema con datos tomados por una persona para ver el grado de acierto del sistema. Para ello, se define el error en OCR, donde se tiene la cantidad de aciertos por cada digito sobre la cantidad de dígitos totales, llamando a éste como error de predicción, quedando la siguiente formula:

$$e_{p[\%]} = \frac{n_t - n_a}{n_t} * 100\%$$

Siendo:

n_t: Número total de dígitos.

n_a: Número de aciertos (por cada digito)

Ejemplo: si se tiene como muestra a reconocer a “1234567890” y el sistema OCR detecta “1234587880”. Se obtiene: n_t = 10, n_a = 8 y, por consiguiente, un e_p = 20%

Con los datos de la tabla 5, se tiene que:

$n_t = 221,$ $n_a = 218,$ e_p = 1.357%

Por otro lado, se tiene el error por palabra o cifra que viene dado por la cantidad de muestras erradas sobre las muestras totales, siendo este el error que nos interesa, puesto que

así este errado solo un digito de una cifra, toda la cifra ya es un dato incorrecto. Se tiene entonces la siguiente fórmula:

$e_{c[\%]} = \frac{n_e}{n_m} * 100\%$; Donde

e_c : error por cifra, n_e: nro. de cifras erradas, n_c: nro. muestras totales

Con los datos de la tabla 5, se tiene que:

n_m = 100; n_e = 2; $e_c = 2\%$

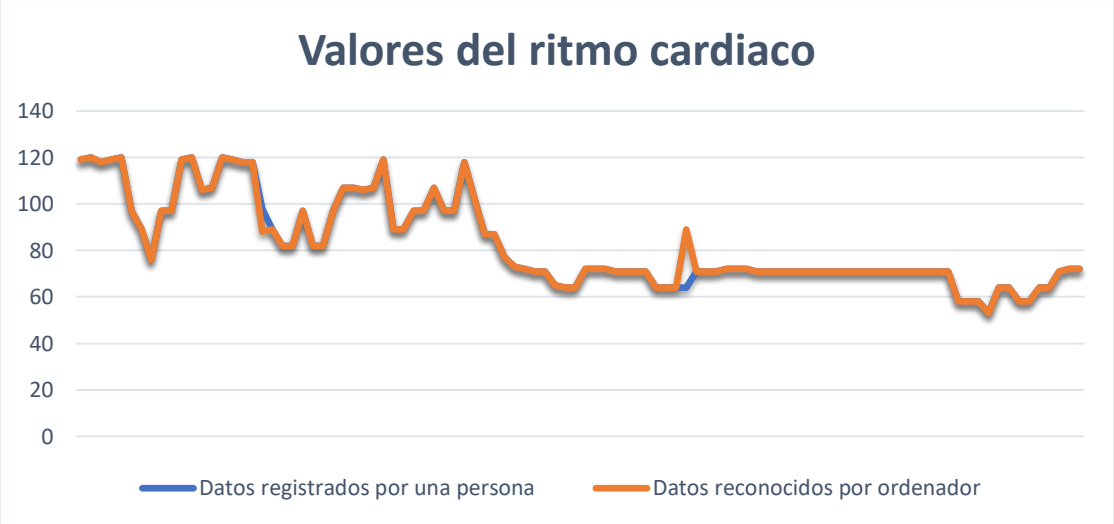


Figura 53: **Valores del Ritmo Cardiaco expresado en BPM tomados cada segundo.**

Fuente: Elaboración propia (2018)

Con estos valores se tiene que cuando se hace un buen arreglo de binarización podemos alcanzar un 98% de acierto en la detección de los caracteres, en la Figura 53 tenemos 100 datos del monitor desfibrilador y el sistema falló en reconocer dos de estos datos. Estando las imágenes fuera de foco o tomadas en el momento del barrido de la generación de una nueva imagen.

4.1.1. Análisis de la precisión del sistema propuesto para una región.

Para determinar la precisión del sistema propuesto, se examinará las razones del por qué el sistema falló en reconocer algunos datos. Para ello, se revisará cada etapa del proceso en cada muestra errónea y se determinará la causante del fallo.

- Muestra 19, dato registrado: 88, dato real: 98.

Figura 54: **Recorte del fotograma de la muestra 19.**

Fuente: Elaboración propia (2018)

En la figura 54, no se observa ningún fallo en particular, así que se verá el proceso de conversión a imagen binaria.

Figura 55: **Imagen binarizada del recorte de la figura 54.**

Fuente: Elaboración propia (2018)

De la figura 55, ya se puede observar cómo la parte inferior del carácter se une por un pixel con la parte central, ahora para ver si esta unión trajo como consecuencia el fallo en el reconocimiento, se examinará el proceso de segmentación de dicha imagen.

Figura 56: **Imagen segmentada del primer término del carácter.**

Fuente: Elaboración propia (2018)

Efectivamente, de la figura 56, se puede ver claramente como la unión de esa zona generó que, al extraer sus características, este sea reconocido como un “ocho”.

- Muestra 61, dato registrado: 84, dato real: 64

Figura 57: **Fotograma de la muestra 61.**

Fuente: Elaboración propia (2018)

Ya desde la captura de la imagen, en la figura 57, se puede observar cómo la región que va a ser reconocida presenta la superposición de dos muestras, por lo que se evaluará cómo afecto esto, al resto del proceso.

Figura 58: **Imagen binarizada del recorte de la figura 57.**

Fuente: Elaboración propia (2018)

Como se preveía, en la figura 58, se obtiene una imagen con un primer término totalmente modificado y un segundo aún reconocible, como ya es previsible la segmentación de esta región no mejorará este resultado, así que se obtiene como valor reconocido un “ocho” para el primer carácter de la región.

4.2. Resultados en dos regiones.

Para este caso se hace un procedimiento muy similar al caso de una sola región, solo que en este se trataran dos regiones distintas cada una con un parámetro a ser reconocido.

Como en el caso anterior también se tomarán imágenes y se procederá a reconocerlas, solo que en esta ocasión se observará como afecta alterar el modo de exposición de la cámara, a través de su obturador, para calibrar la intensidad de luz que entrará a su lente.

Para ello se emplearon tres configuraciones distintas, ajustando la velocidad de obturación de la cámara, siendo estas las condiciones en que se tomaron las imágenes:

- Las imágenes se tomaron con luz diurna cielo soleado.
- Distancia de la cámara al monitor: 50 a 60 cm.
- Cámara fija, sujetada por brazo de apoyo.

4.2.1. Dos regiones con velocidad de obturación de 1/250 seg.

La primera toma de datos fue poniendo la velocidad de obturación en 1/250 segundos con un tiempo de captura de 5 minutos con 2 segundos. Como en el caso anterior también se tomaron las muestras a ser reconocidas en un intervalo de 5 segundos cada muestra, obteniendo 60 muestras.

Figura 59: **Imagen con el obturador en 1/250 seg.**

Fuente: Laboratorio de Biomédica - Elaboración propia (2018).

HR BPM		HR BPM		SpO2 %		SpO2 %	
Persona	OCR	Persona	OCR	Persona	OCR	Persona	OCR
70	*70*	52	*52*	90	*90*	89	*89*
57	*57*	80	*80*	89	*89*	89	*89*

63	*63*	61	*61*	88	*88*	89	*89*
55	*55*	64	*64*	88	*88*	89	*89*
49	*48*	80	*80*	92	*92*	89	*89*
37	*37*	64	*64*	91	*91*	89	*89*
49	*49*	53	*53*	89	*89*	91	*91*
53	*53*	37	*37*	87	*87*	90	*90*
75	*75*	60	*60*	88	*88*	89	*89*
65	*65*	88	*88*	90	*80*	99	*99*
58	*58*	69	*69*	92	*92*	88	*88*
61	*81*	78	*78*	91	*91*	88	*88*
49	*49*	65	*65*	90	*90*	89	*89*
61	*61*	58	*58*	90	*90*	90	*90*
85	*85*	67	*67*	90	*90*	89	*89*
68	*68*	58	*58*	90	*90*	90	*90*
78	*78*	60	*60*	89	*89*	91	*81*
64	*84*	72	*72*	90	*90*	91	*91*
54	*54*	58	*58*	89	*89*	90	*90*
45	*45*	53	*53*	91	*91*	89	*89*
84	*84*	58	*58*	92	*92*	91	*91*
53	*53*	51	*51*	92	*92*	92	*92*
53	*53*	63	*63*	90	*90*	91	*91*
68	*68*	55	*55*	87	*87*	90	*90*
56	*58*	63	*63*	90	*90*	89	*89*
52	*52*	58	*68*	90	*90*	90	*90*
88	*88*	45	*45*	91	*91*	90	*90*
50	*50*	40	*40*	88	*88*	89	*88*
53	*53*	63	*63*	89	*89*	89	*89*
51	*51*	66	*66*	89	*89*	90	*90*

Tabla 5: **Contrastación de los valores obtenidos en dos regiones 1/250seg.**

Fuente: Elaboración propia (2018).

Para este caso se tienen dos parámetros reconocidos en un mismo momento, por lo que se hizo un análisis del error por separado y otro analizando a ambos.

Primero tendremos el error de predicción para el pulso cardiaco HR:

$n_t = 120;$ $n_a = 116;$ e_p = 3.333%

Luego se tiene el caso de la saturación de oxígeno Sp02%:

$n_t = 120;$ $n_a = 117;$ e_p = 2.5%

Pero el error que nos interesa calcular es el de cifra:

Ritmo cardiaco:

n_m = 60; n_e = 4; $e_c = 6.667\%$

Saturación de oxígeno:

n_m = 60; n_e = 3; $e_c = 5\%$

Si consideramos a ambos parámetros:

n_m = 60; n_e = 7; $e_c = 11.667\%$

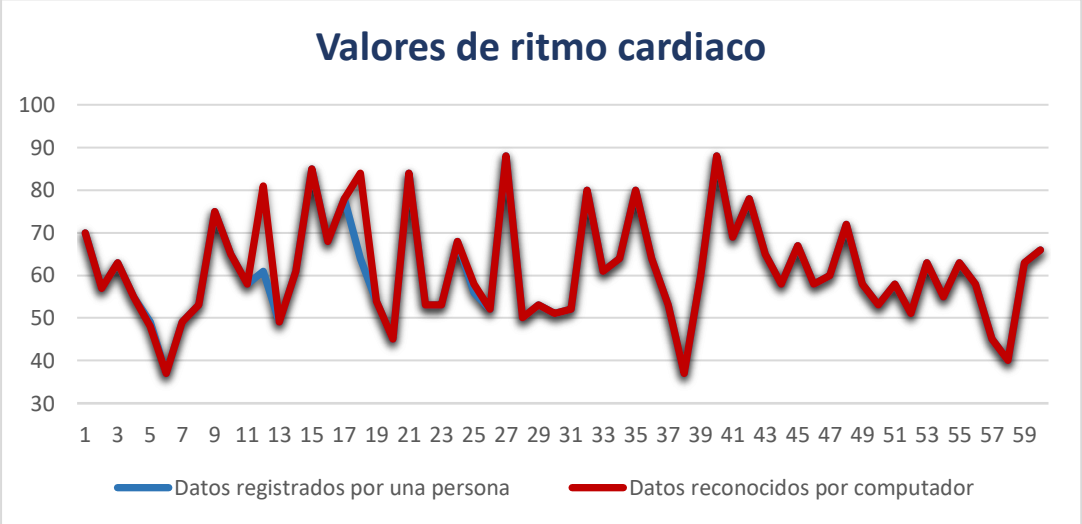


Figura 60: **Valores del ritmo cardiaco tomados cada 5 segundos (1/250).**

Fuente: Elaboración propia (2018).

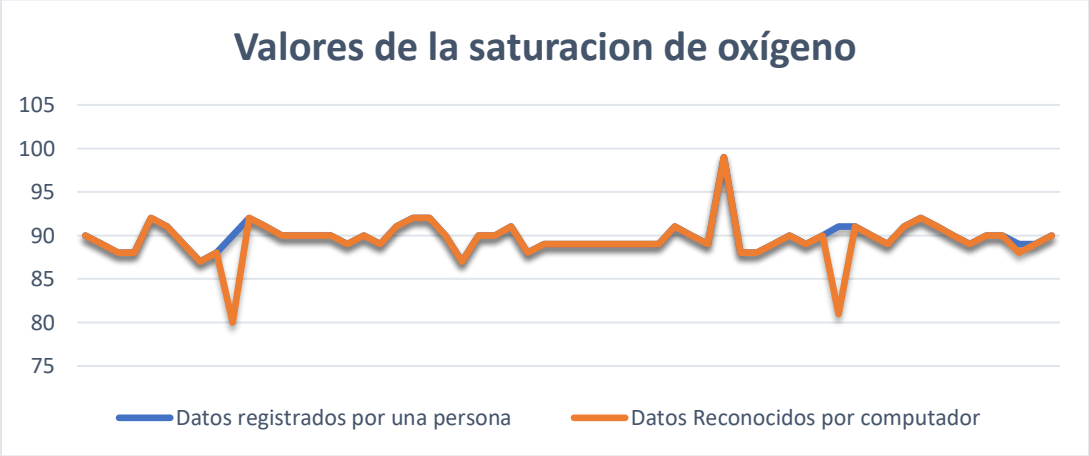


Figura 61: **Valores de SpO2 tomados cada 5 segundos (1/250).**

Fuente: Elaboración propia (2018).

Obteniendo una precisión del 93.33% en el parámetro de ritmo cardiaco y de un 95% en el parámetro de la saturación de oxígeno de un total de 60 muestras. Se observa que considerando ambos parámetros solo se alcanza una precisión del 85% pero se debe

considerar que un dato errado en uno de los parámetros no genera que toda la muestra (fotograma) esté errada, por lo que se considera el error obtenido en cada parámetro.

El aumento del error en imágenes con dos parámetros se debe mayormente a los resultados de la binarización de las regiones puesto que se usan los mismos parámetros de binarización para ambas regiones aun sabiendo que estas poseen distintos colores en los que son representados

4.2.1.1. Análisis de la precisión del sistema propuesto para dos regiones y velocidad de obturación de 1/250 – para el ritmo cardiaco.

Como se estudió en el caso anterior evaluaremos la precisión del sistema para este nuevo caso, donde tendremos que evaluar cada muestra fallida.

- Muestra 5, dato registrado: 48, dato real: 49.

Figura 62: **Fotograma de la región recortada de la muestra 5.**

Fuente: Elaboración propia (2018)

En la figura 62 no se observa ningún fallo por lo que es necesario revisar su tratamiento posterior.

Figura 63: **Imagen binarizada del recorte de la figura 62.**

Fuente: Elaboración propia (2018)

Tras observar la imagen, de la figura 63, se puede ver como el proceso de binarización, juntó por poco la parte inferior izquierda del carácter, siendo el mismo caso de la muestra 19, en el punto 4.1.

- Muestra 12, dato registrado: 81, dato real: 61.

Figura 64: **Fotograma de la región recortada de la muestra 12.**

Fuente: Elaboración propia (2018)

De la figura 64, tampoco podemos sacar ninguna conclusión por lo que examinaremos su procesamiento.

Figura 65: **Imagen binarizada del recorte de la figura 64.**

Fuente: Elaboración propia (2018)

Esta figura también presenta el problema de la muestra anterior, y ya se puede definir como causa de error, al proceso de binarizado cuando este se realiza sobre caracteres numéricos, tales como el “seis” y el “nueve”.

- Muestra 18, dato registrado: 84, dato real: 64

Figura 66: **Fotograma de la región recortada de la muestra 18.**

Fuente: Elaboración propia (2018)

De la figura 66, se puede prever que el problema será la unión de la parte superior del "seis" con su zona central.

Figura 67: **Imagen binarizada del recorte de la figura 66.**

Fuente: Elaboración propia (2018)

Obteniendo este resultado previsto, se denominará como un mismo tipo de error a estos que presentan una unión pequeña en los caracteres como el "seis" y el "nueve", y se llamará error de binarizado.

- Muestra 25, dato registrado: 58, dato real: 56.

Figura 68: **Fotograma de la región recortada de la muestra 25.**

Fuente: Elaboración propia (2018)

Siendo esta muestra el mismo caso de la anterior, ya se contabiliza como error de binarizado. Por lo que se verá su binarizado, en la figura 68, para poder cerciorarse que efectivamente es el mismo error.

Figura 69: **Imagen binarizada del recorte de la figura 68.**

Fuente: Elaboración propia (2018)

4.2.1.2. Análisis de la precisión del sistema propuesto para dos regiones y velocidad de obturación de 1/250 – para la saturación de oxígeno.

Ahora se observará cómo han sido los errores en la segunda región presente en las muestras. Cabe resaltar que como se ha ido viendo, tenemos que en la tabla 5, los errores cometidos vuelven a ocurrir en el carácter "nueve" por lo que se examinará a estos de manera conjunta.

- Muestra 10, dato registrado: 80, dato real: 90.
- Muestra 47, dato registrado: 81, dato real: 91.
- Muestra 58, dato registrado: 88, dato real: 89.

Figura 70: **Fotogramas de los recortes de la muestra 10, 47 y 58.**

Fuente: Elaboración propia (2018)

De la figura 70, aún no se puede deducir si el error es el mismo, pero ya se puede intuir que presentaran el error de binarizado.

Figura 71: **Imagen binarizada de los recortes de la figura 70.**

Fuente: Elaboración propia (2018)

Y de acuerdo a lo esperado las imágenes resultantes, corroboran que la causante de error fue el binarizado.

4.2.2. Dos regiones con velocidad de obturación de 1/500 seg.

La siguiente toma de datos fue poniendo el obturador en 1/500 segundos, y se tomó una captura de datos por un tiempo de 1 minuto 28 segundos, y un muestreo de 5 segundos cada muestra.

Figura 72: **Imagen de la cámara con el obturador en 1/500 seg.**

Fuente: Laboratorio de Biomédica - Elaboración propia (2018).

HR BPM		HR BPM		SpO2 %		SpO2 %	
Persona	OCR	Persona	OCR	Persona	OCR	Persona	OCR
44	--	44	--	92	*92*	93	*93*
50	*50*	47	*47*	92	*92*	91	*91*
64	*64*	49	*49*	91	*91*	88	*88*
84	*84*	67	*67*	89	*89*	88	*88*
78	*78*	58	*58*	88	*88*	90	*90*
64	*64*	42	*42*	90	*90*	91	*91*
61	*61*	42	*42*	92	*92*	90	*90*
81	*81*	42	*42*	91	*91*	89	*89*
79	*79*	49	*49*	91	*81*	89	*89*
64	*64*	54	*54*	90	*90*	90	*90*
57	*57*	46	*46*	89	*89*	90	*90*
67	*67*	67	*67*	88	*88*	90	*90*
61	*61*	64	*64*	85	*85*	90	*90*
50	*50*	49	*49*	85	*85*	90	*90*
44	--			91	*91*		

Tabla 6: **Contrastación de los valores obtenidos en dos regiones 1/500.**

Fuente: Elaboración propia (2018).

Primero tendremos el error de predicción para el pulso cardiaco HR:

$n_t = 58;$ $n_a = 52;$ $e_p = 10.345\%$

Luego se tiene el caso de la saturación de oxígeno Sp02%:

$n_t = 58;$ $n_a = 57;$ $e_p = 1.724\%$

Pero el error que nos interesa calcular es el de cifra:

Ritmo cardiaco:

$n_m = 29;$ $n_e = 3;$ $e_c = 10.345\%$

Saturación de oxígeno:

$n_m = 29;$ $n_e = 1;$ $e_c = 3.448\%$

Si consideramos a ambos parámetros:

$n_m = 60;$ $n_e = 4;$ $e_c = 13.793\%$

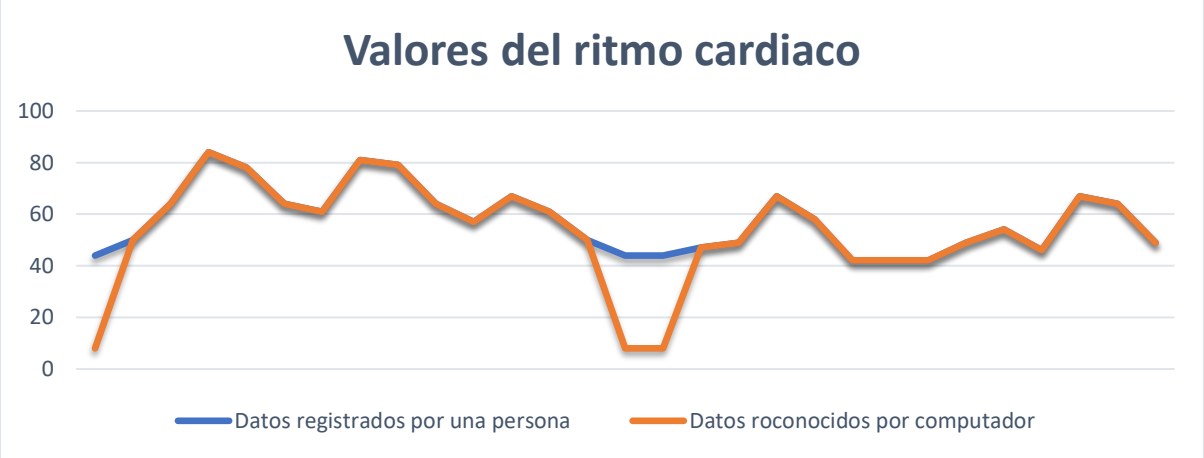


Figura 73: **Valores del ritmo cardiaco tomados cada 5 segundos (1/500).**

Fuente: Elaboración propia (2018).

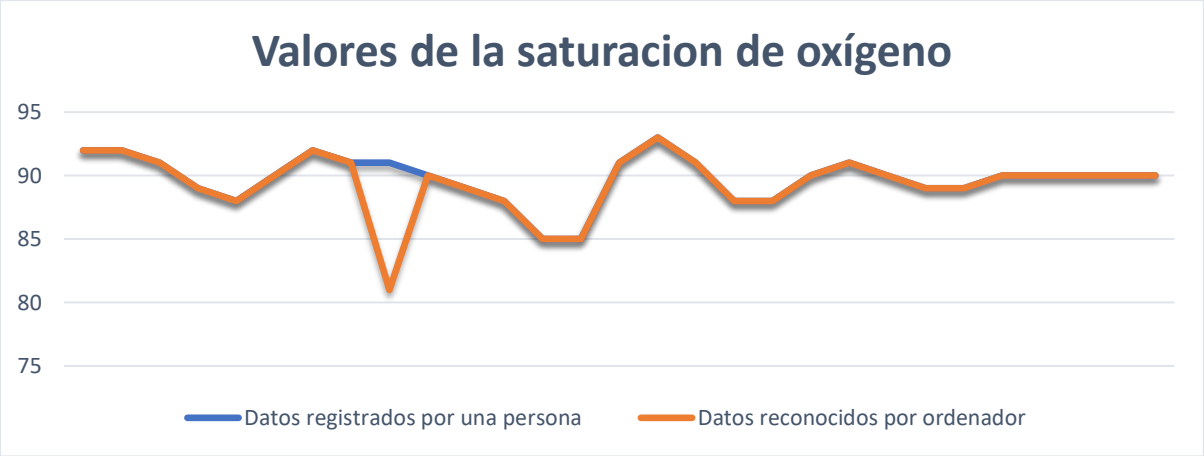


Figura 74: **Valores de SpO2 tomados cada 5 segundos (1/500).**

Fuente: Elaboración propia (2018).

Con esta configuración se alcanzó una precisión del 89.65% en el parámetro del ritmo cardiaco y de un 96.55% en la saturación de oxígeno, de un total de 29 muestras, siendo la causa de error el error de binarizado solo que en este caso algunas imágenes son más nítidas lo que ayuda a reconocer correctamente los parámetros presentes en la pantalla del monitor. Aunque para este caso se tiene un incremento en el error, esto puede deberse a la poca cantidad de muestras recogidas.

4.2.2.1. Análisis de la precisión del sistema propuesto para dos regiones y velocidad de obturación de 1/500 – para el ritmo cardiaco.

Como en el punto anterior se examinará por separado cada región reconocida, además trataremos cada muestra en conjunto ya que las tres ocurren con el mismo patrón.

- Muestra 1, dato registrado: --, dato real: 44.
- Muestra 15, dato registrado: --, dato real: 44.
- Muestra 16, dato registrado: --, dato real: 44.

Figura 75: **Fotogramas de los recortes de la muestra 1, 15 y 16.**

Fuente: Elaboración propia (2018)

De la figura 75, ya podemos intuir que es lo que va suceder con la parte trasera del primer "cuatro" y la parte delantera del segundo, pero para poder cerciorarse de esto, se revisará su binarización.

Figura 76: **Imagen binarizada de los recortes de la figura 75.**

Fuente: Elaboración propia (2018)

Viendo los resultados obtenidos por el sistema propuesto, en la tabla 6, para las muestras 1, 15 y 16, se observa que el sistema no le asigna ningún valor establecido, esto se debe a que se programó el sistema para que pudiera reconocer a

cualquier obstáculo que se atraviese, y con ello no almacenar un valor totalmente diferente del que está presente en la pantalla del monitor. Esto se logra con un algoritmo que reconoce los tamaños de los objetos dentro de la región seleccionada, y si estos no se encuentran dentro los márgenes esperados para un carácter, el sistema desecha este dato y no le asigna un valor, pero si toma la muestra que sirve para contrastar él porque sucedió dicho error. Por lo que, lo descrito anteriormente describe lo ocurrido con las muestras 1, 10 y 15, ya que el algoritmo detectó que dichas muestras como un solo carácter, y estas no corresponden al tamaño de un carácter normal, por consiguiente, no se les asignó ningún valor.

4.2.2.2. Análisis de la precisión del sistema propuesto para dos regiones y velocidad de obturación de 1/500 – para la saturación de oxígeno.

Para esta región solo se tiene una muestra errada, y por lo visto en la tabla 7, esta se encuentra en el carácter "nueve" por lo que se trata del error de binarizado, se verá la figura 77 y 78 para corroborar que este sea el caso.

- Muestra 9, dato registrado: 81, dato real: 91.

Figura 77: **Fotograma del recorte de la muestra 9.**

Fuente: Elaboración propia (2018)

Figura 78: **Imagen binarizada de los recortes de la figura 77.**

Fuente: Elaboración propia (2018)

Efectivamente esta muestra presenta error de binarizado.

4.2.3. Dos regiones con velocidad de obturación de 1/500 seg con aislamiento de luz exterior.

También se tomó muestras de imágenes de tal forma que se vean los colores totalmente diferenciados, para ello se cubrió el monitor y la cámara con una tela (cortina) para que la luz proveniente del exterior no interfiriera con el generado por el monitor. Con esta disposición se tuvo imágenes con los parámetros con casi la misma intensidad en el color, teniendo los siguientes resultados.

Figura 79: **Imagen cubierta, con el obturador en 1/500 seg.**

Fuente: Laboratorio de Biomédica - Elaboración propia (2018).

Como efecto secundario se obtuvo que las imágenes captadas por la cámara sean más brillantes, haciendo a dichas imágenes más gruesas de lo normal y que su color se perciba casi como el mismo, como se observa en la figura 7.

HR BPM		HR BPM		SpO2 %		SpO2 %	
Persona	OCR	Persona	OCR	Persona	OCR	Persona	OCR
40	*40*	56	*56*	91	*91*	96	*96*
43	*43*	64	*64*	90	*90*	92	*92*
47	*47*	86	*86*	89	*89*	91	*91*
60	*60*	92	*92*	90	*90*	90	--
64	*64*	58	*58*	91	*91*	88	*88*
57	*57*	49	--	90	*90*	88	*88*
61	*61*	58	*58*	90	*90*	90	*90*
70	*70*	66	*66*	90	*90*	91	*91*
66	*66*	69	*69*	91	*91*	89	*89*
66	*66*	60	*60*	93	*93*	88	*88*
51	*61*	58	--	95	*95*	89	*89*
86	*86*	64	*64*	96	*96*	90	--
63	*63*	63	*63*	95	*95*	90	--
42	*42*	50	*50*	96	*96*	88	*88*
34	*34*	59	*59*	95	*95*	86	*86*
38	*38*	57	*57*	95	*95*	87	*87*
52	*52*			95	*95*		

Tabla 7: **Contrastación de los valores obtenidos, monitor cubierto - 1/500.**

Fuente: Laboratorio de Biomédica - Elaboración propia (2018).

Primero tendremos el error de predicción para el pulso cardiaco HR:

$n_t = 66;$ $n_a = 62;$ $e_p = 6.061\%$

Luego se tiene el caso de la saturación de oxígeno Sp02%:

$n_t = 66;$ $n_a = 61;$ $e_p = 7.576\%$

Pero el error que nos interesa calcular es el de cifra:

Ritmo cardiaco:

$n_m = 33;$ $n_e = 3;$ $e_c = 9.091\%$

Saturación de oxígeno:

$n_m = 33;$ $n_e = 3;$ $e_c = 9.091\%$

Si consideramos a ambos parámetros:

n_m = 33; n_e = 6; $e_c = 18.182\%$

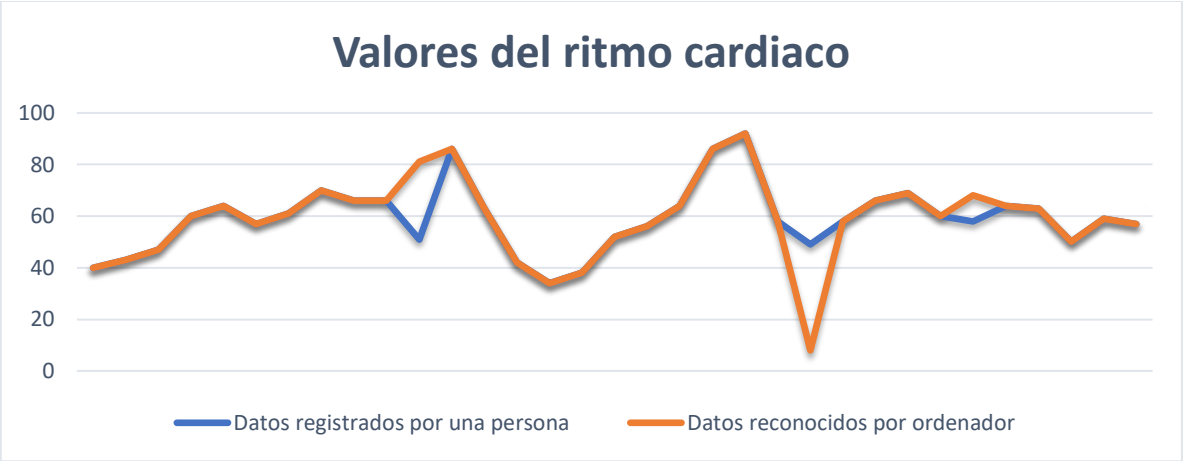


Figura 80: **Valores del ritmo cardiaco tomados cada 5 seg (cubierta 1/500).**

Fuente: Elaboración propia (2018).

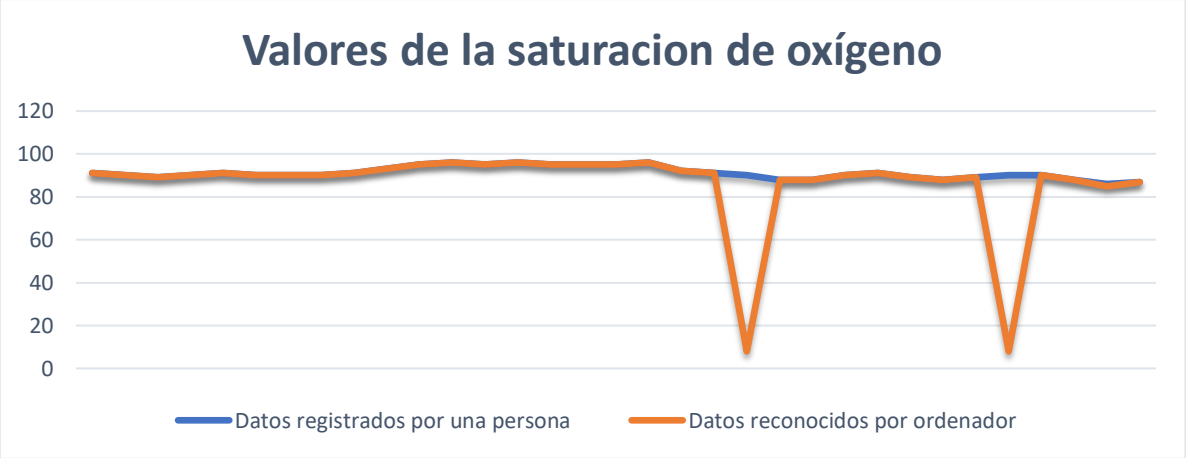


Figura 81: **Valores de SpO2 cardiaco tomados cada 5 seg (cubierta 1/500).**

Fuente: Elaboración propia (2018).

Con esta configuración se alcanzó una precisión del 91% en el parámetro del ritmo cardiaco y de un 91% en la saturación de oxígeno, de un total de 34 muestras, siendo el tratamiento de la imagen (dilatación – erosión) ahora la causa de ocurrencia de error. Teniendo en cuenta que los parámetros del tratamiento de la imagen como lo son el factor de dilatación y el factor de erosión los mismos para todas las muestras; no

es de extrañarse que el error pueda subir cuando las imágenes a tratar se tomen en condiciones diferentes a las que fueron calibradas.

4.2.3.1. Análisis de la precisión del sistema para dos regiones y velocidad de obturación de 1/500 – para el ritmo cardiaco, monitor cubierto.

Tal como se realizó en puntos anteriores se evaluará las muestras que presentan errores y se definirá su causa.

- Muestra 11, dato registrado: 61, dato real: 51.

Figura 82: **Fotograma del recorte de la muestra 11.**

Fuente: Elaboración propia (2018)

De la figura 82, no se puede sacar ninguna conclusión por lo que se evaluará su proceso de binarizado.

Figura 83: **Imagen binarizada de los recortes de la figura 82.**

Fuente: Elaboración propia (2018)

Se ve, que en la figura 83, aún no se percibe una unión como en el caso del error de binarización por lo que se verá si en el proceso de segmentación ocurrió alguna falla.

Figura 84: **Imagen segmentada del primer carácter.**

Fuente: Elaboración propia (2018)

Incluso tras segmentar la imagen no se puede observar un error, por lo que este error, es de reconocimiento.

- Muestra 23, dato registrado: --, dato real: 49.
- Muestra 28, dato registrado: --, dato real: 58.

Figura 85: **Fotogramas de los recortes de la muestra 23 y 28.**

Fuente: Elaboración propia (2018)

De la figura 85, claramente se puede observar que los caracteres se encuentran conectados, y como ya se vio anteriormente este tipo de error, por lo que es necesario clasificarlo, al que se nombrará como error de captura. Por otro lado, se verá los resultados de su binarización para poder observar con mayor claridad como afecto en la toma de decisión del sistema.

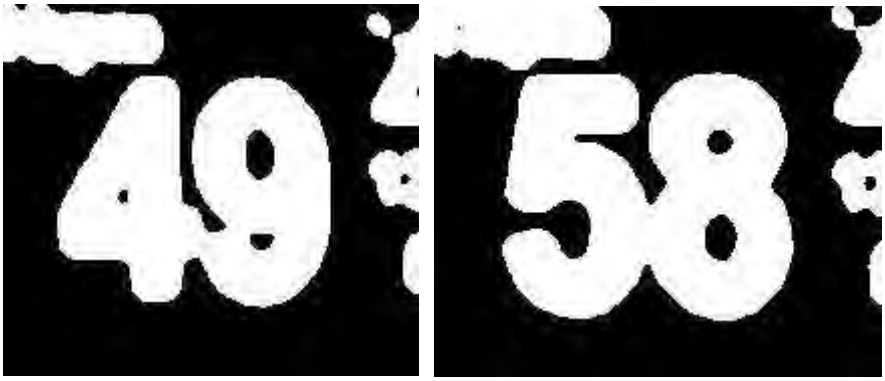


Figura 86: **Imagen binarizada de los recortes de la figura 85.**

Fuente: Elaboración propia (2018)

Tal como fue previsto, en la figura 86, se comprueba que la unión de caracteres, se detectó por el algoritmo de reconocimiento, que rechaza los datos que no corresponden al tamaño de un carácter.

4.2.3.2. Análisis de la precisión del sistema para dos regiones y velocidad de obturación de 1/500 – para la saturación de oxígeno, monitor cubierto.

Para este punto se examinará las tres muestras en conjunto puesto que las tres presentan el mismo tipo de error.

- Muestra 21, dato registrado: --, dato real: 90.
- Muestra 29, dato registrado: --, dato real: 90.
- Muestra 30, dato registrado: --, dato real: 90.

Figura 87: **Fotogramas de los recortes de la muestra 21, 29 y 30.**

Fuente: Elaboración propia (2018)

De la figura 87, como en casos anteriores, se observa el error de captura, teniendo los caracteres claramente unidos, debido a la fuerte intensidad que capta la cámara en esta configuración.

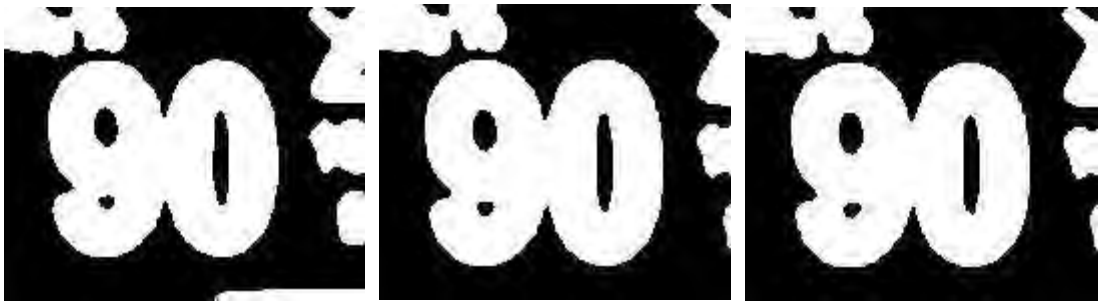


Figura 88: **Imagen binarizada de los recortes de la figura 87.**

Fuente: Elaboración propia (2018)

4.2.4. Dos regiones con velocidad de obturación por defecto con aislamiento de luz exterior.

Por último, se tomó datos con el monitor y la cámara cubiertos por la cortina solo que esta vez se dejó el obturador puesto en su configuración por defecto dando los siguientes resultados.

Figura 89: **Imagen cubierta con la configuración por defecto de la cámara.**

Fuente: Laboratorio de Biomédica - Elaboración propia (2018).

Como se puede observar en la figura 89 la imagen es realmente nítida con los colores bien diferenciados por lo que se podrá observar con claridad la precisión del sistema ante una imagen casi ideal.

Cabe resaltar que en este modo también se dejó de tomar los valores de saturación por un tiempo por lo que se verá también el comportamiento del sistema cuando este no encuentra datos que se consideren parámetros a ser reconocidos.

HR BPM		HR BPM		SpO2 %		SpO2 %	
Persona	OCR	Persona	OCR	Persona	OCR	Persona	OCR
67	*67*	60	*60*	93	*93*	91	*91*
70	*70*	71	*71*	92	*92*	90	*90*
58	*58*	158	*158*	90	*90*	90	*90*
53	*53*	244	*2*	90	*90*	91	*91*
58	*58*	271	*271*	89	*89*	91	*91*
111	*111*	96	*96*	90	*90*	90	*90*
266	*266*	59	*59*	91	*91*	89	*89*
90	*90*	49	*49*	90	*90*	90	*90*
114	*114*	63	*63*	90	*90*	91	*91*
69	*69*	61	*61*	90	*90*	91	*91*
60	*60*	107	*107*	89	*89*	91	*91*
49	*49*	287	*287*	89	*89*	90	*80*
58	*58*	126	*186*	89	*89*	89	*89*
49	*49*	162	*162*	90	*90*	89	*89*
39	*39*	160	*160*	90	*90*	89	*89*
46	*46*	246	*246*	90	*90*	92	*92*
40	*40*	178	*178*	90	*90*	95	*95*
46	*46*	61	*61*	90	*90*	95	*95*
63	*63*	140	*140*	89	*89*	94	*94*
61	*61*	86	*86*	89	*89*	-	
80	*80*	49	*49*	90	*90*	-	
63	*63*	142	*142*	90	*90*	-	
45	*45*	106	*106*	90	*90*	-	
42	*42*	100	*100*	90	*90*	-	
33	*33*	264	*264*	91	*91*	-	
37	*37*			92	*92*		

Tabla 8: **Contrastación de los valores obtenidos, monitor cubierto, parámetros de la cámara por defecto.**

Fuente: Elaboración propia (2018).

Primero tendremos el error de predicción para el pulso cardiaco HR:

$n_t = 120;$ $n_a = 117;$ e_p = 2.5%

Luego se tiene el caso de la saturación de oxígeno Sp02%:

$n_t = 90;$ $n_a = 89;$ e_p = 1.111%

Pero el error que nos interesa calcular es el de cifra:

Ritmo cardiaco:

n_m = 51; n_e = 3; $e_c = 5.882\%$

Saturación de oxígeno:

n_m = 46; n_e = 1; $e_c = 2.174\%$

Si consideramos a ambos parámetros:

n_m = 51; n_e = 4; $e_c = 7.843\%$

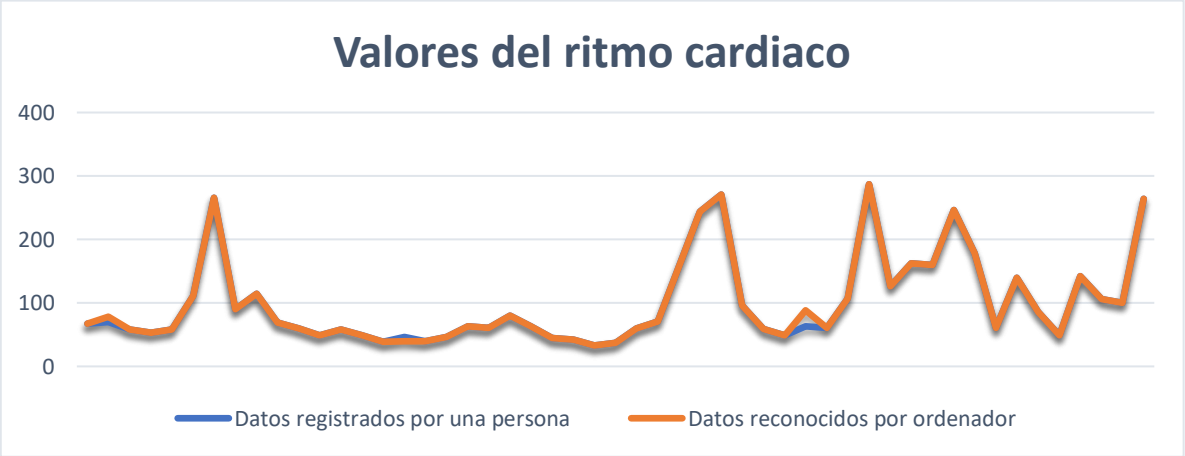


Figura 90: **Valores del ritmo cardiaco tomados cada 5 seg (cubierta).**

Fuente: Elaboración propia (2018).

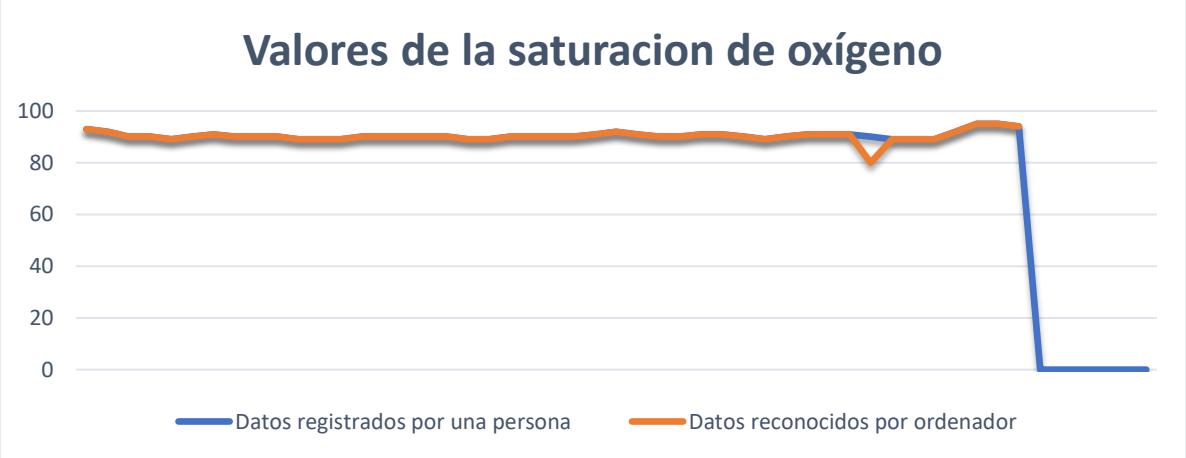


Figura 91: **Valores de SpO2 tomados cada 5 seg (cubierta).**

Fuente Elaboración propia (2018).

Con esta configuración se alcanzó un 96% de precisión para el parámetro de ritmo cardiaco, y de un 98% para la saturación de oxígeno, de un total de 51 muestras, y como se puede observar en la figura 91 hay un tiempo en que la región de saturación de oxígeno deja de registrar datos por lo que el sistema no devuelve valores en dicho momento. Esto como ya se explicó anteriormente, debido al algoritmo de reconocimiento de caracteres válidos.

4.2.4.1. Análisis de la precisión del sistema con el monitor cubierto para el ritmo cardiaco con configuración de la cámara por defecto.

Tal como se estuvo estudiando en puntos anteriores se examinará los errores cometidos con estas condiciones.

- Muestra 30, dato registrado: 2, dato real: 244.

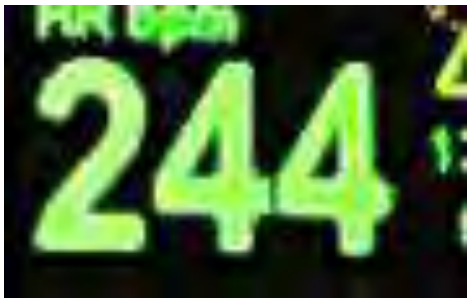


Figura 92: **Fotograma del recorte de la muestra 30.**

Fuente: Elaboración propia (2018)

De la figura 92, vemos un error ya conocido, pero se verá primero su proceso de binarizado para corroborar que es el mismo.

Figura 93: **Imagen binarizada de los recortes de la figura 92.**

Fuente: Elaboración propia (2018)

Como se dedujo, los caracteres del "cuatro" unieron sus extremos formando un solo segmento y fueron reconocidos por el algoritmo de reconocimiento, siendo descartados como carácter valido.

- Muestra 39, dato registrado: 186, dato real: 126.

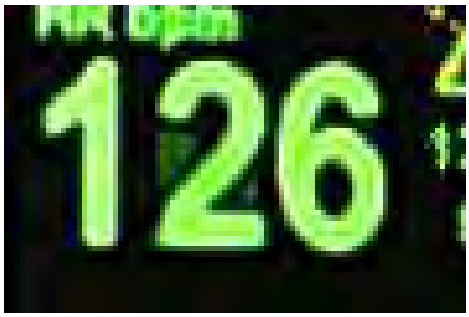


Figura 94: **Fotograma del recorte de la muestra 39.**

Fuente: Elaboración propia (2018)

De la figura 94, se observa qué ocurre un caso similar al de la muestra 61 del punto 4.2.1, y es que la imagen capturada captó una superposición de dos muestras, por lo que se puede considerar también como un error de captura, pero esta sugiere ser muy distinta a la que se vio anteriormente, por lo que la nombraremos como "error por superposición".

Figura 95: **Imagen binarizada de los recortes de la figura 94.**

Fuente: Elaboración propia (2018)

Y tal como en el caso anterior el carácter en que ocurre la superposición cambia totalmente su forma, y el sistema le designa un valor totalmente diferente.

4.2.4.2. Análisis de la precisión del sistema con el monitor cubierto para la saturación de oxígeno con configuración de la cámara por defecto.

Por último, se evaluará la muestra faltante.

- Muestra 38, dato registrado: 80, dato real: 90.

Figura 96: **Fotograma del recorte de la muestra 39.**

Fuente: Elaboración propia (2018)

El recorte de la figura 96 no da indicios de indicar porqué pudo haber fallado el sistema, por lo que se necesita examinar la etapa de procesado de la imagen para descubrir la causa del error.

Figura 97: **Imagen binarizada del recorte de la figura 96.**

Fuente: Elaboración propia (2018)

Examinando con más detenimiento el resultado de la binarización del recorte se observa una pequeña juntura el carácter del número ‘9’, por lo que se produjo un error al momento de binarizar la imagen.

4.3. Clasificación de los errores encontrados en el desarrollo del proyecto.

En esta sección se clasificará los errores encontrados en cada caso del que se tomó datos y, además, se examinará cuánto fue su aporte al error general del sistema.

Error de captura: ocurre cuando la imagen obtenida desde la cámara tiene alguna falla, que impide su correcto tratamiento, se observó que, dentro de este tipo de error, las imágenes con mucha intensidad ocasionan que sus caracteres se junten y se eliminen por causa del algoritmo de reconocimiento, así también ocurrió el caso que en una muestra haya superposición de dos fotogramas que son reconocidos como datos erráticos.

Error de binarización: ocurre cuando al binarizarse la región, se produce uniones en los caracteres en zonas donde no las había, esto genera que la figura resultante sea reconocida de forma errática o si la unión ocurre entre dos cifras, estas sean eliminadas.

Error de comparación: por último, cuando una imagen correctamente tratada obtiene un reconocimiento errático, esta se produjo en la comparación de las características extraídas del digito a tratar con las de la matriz de predicción. Este es propio del método de reconocimiento que se empleó en el sistema.

Para obtener la precisión total del sistema se empleará el error por cifra, ya que el sistema no admite equivocaciones por dígito.

Teniendo así un total de “273” muestras y un total de “22” errores.

Dando como resultado una precisión del 91,94%, que está dentro del margen aceptable por la hipótesis, donde se requiere una precisión mínima del 90%, sabiendo que se tuvo diversos escenarios en los que se evaluó el desenvolvimiento del sistema propuesto.

Ahora se evaluará la contribución de cada error a este resultado, teniendo la siguiente tabla:

	Número de errores	**Porcentaje de contribución**
Error de captura	2	0.733
Error de binarización	10	3.663
Error de segmentación	9	3.297
Error de clasificación	1	0.366

Tabla 9: **Clasificación de errores.**

Fuente: Elaboración propia (2018).

4.4. Detección y corrección de errores.

En el punto anterior se observa que la mayor parte de errores recae en la captura y en la binarización de la región a reconocer, por lo que es necesario implementar un algoritmo de detección y corrección de errores. Para ello se dividirá en dos los tipos de errores: los que se pueden corregir y los que no. Los errores que se puedan corregir aplicarán cambios en la imagen, mientras los que no, requerirán la toma de una nueva muestra.

Dicho algoritmo se aplicará en dos partes, una tras la etapa de conversión a escala de grises, y otra después de la etapa de binarización, detectando fallas de captura como superposición y uniones indebidas en los caracteres, respectivamente.

Para visualizar como actuará la detección de errores se tiene la siguiente figura.

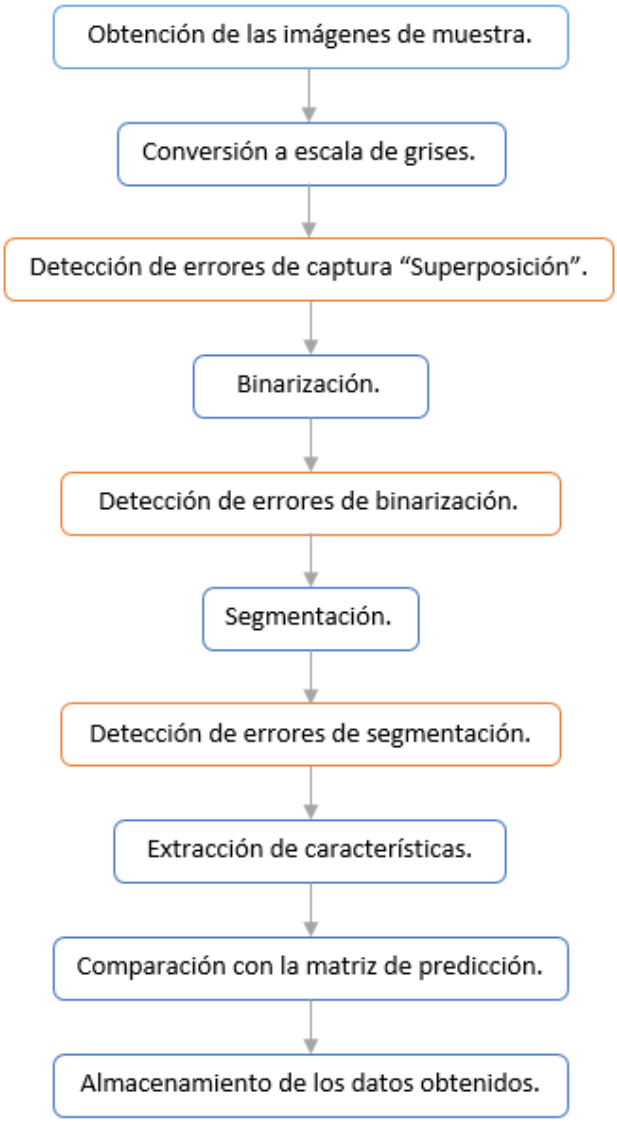


Figura 98: **Diagrama de flujo del proceso con la detección de errores.**

Fuente: Elaboración propia (2018)

4.4.1. Detección de superposición de muestras.

Una vez realizado el binarizado de la región en los casos de superposición, la imagen binarizada resultante exhibe un predominio de una de las muestras y el acople de ciertas partes de la otra, degenerando a ambas muestras, cuando se observa el histograma de una muestra real sin tratar, se ve que todas guardan una similitud entre todas ellas incluyendo a la muestra errada, sin embargo, si se observa el histograma de la imagen a escala de grises, claramente el patrón de similitud se rompe.

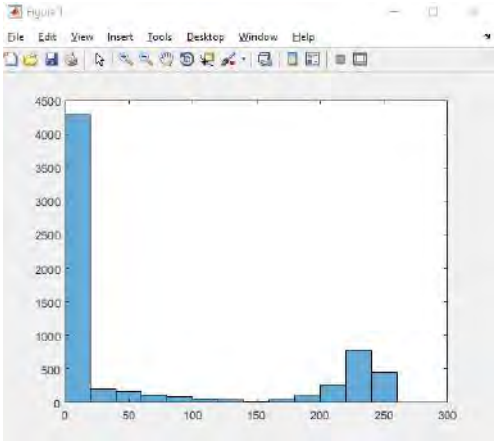


Figura 99: **Histograma de las muestras de una región en escala de grises.**

Fuente: Elaboración propia (2018)

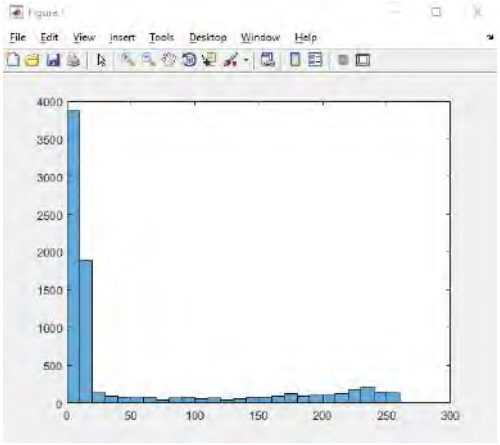


Figura 100: **Histograma de la muestra 61 errada de una región en escala de grises.**

Fuente: Elaboración propia (2018)

Se puede observar cómo empiezan a aparecer tonos que en el patrón normal de las demás muestras no aparecían sobre todo en la zona entre 100 a 200, en las muestras sin fallos de superposición el histograma es prácticamente el mismo sin embargo como se observa en la figura 100 esto varía enormemente en una muestra errada por superposición, para aprovechar esto se aplicó un algoritmo que detecta los "bins" presentes entre 80 a 200 siendo estos los tonos intermedios que se presentan en la conversión en escala de grises de una muestra errada. Si la cantidad de tonos entre el rango mostrado excede las 750 unidades es ya porque hay presencia de muchos tonos

intermedios, lo que implica que la imagen tomada tenga una alta probabilidad de estar errada.

Una vez detectado la falla, se procede a tomar una nueva muestra y comenzar el proceso otra vez, si en esta ocasión la nueva muestra no presenta este fallo, el proceso continuará hasta el siguiente proceso de detección de uniones, que se explica más adelante. Caso contrario vuelve a repetir este procedimiento 2 veces más, si tras esto el proceso no se soluciona salta un aviso: "Ocurrió un error en la cámara, recalibre la cámara."

4.4.2. Detección de uniones indebidas en los caracteres.

Para detectar uniones indeseadas en la binarización, procedemos a ejecutar un algoritmo de búsqueda, para ello emplearemos el comando "norm" de Matlab que nos permite filtrar una matriz, con una condición dada por usuario, su sintaxis es la siguiente:

```
l = norm(hd,pnorm);
```

Dónde: "hd" es la matriz a evaluar, y "pnorm" es la condición con la que filtraremos la matriz.

Para filtrar de manera correcta se tuvo que diseñar la condición de la siguiente forma: nuestra condición es la unión de un extremo del carácter con otra parte de él, por lo que la unión es un vector de pequeño tamaño, se consideró dicho tamaño como un máximo de 2 pixeles, pudiendo tener al vector de las siguientes maneras:

Vector = [0 1 0] para un pixel, o

Vector = [0 1 1 0] para dos pixeles.

Con esto se procedió a estructurar la condición de la siguiente forma:

Teniendo la matriz de la imagen binarizada, se procede a comparar el vector de condición con cada punto de la imagen, dicha comparación no es más que la sustracción de los valores de cada sección de la matriz con mismas dimensiones que posee el vector de comparación y si en resultado de esta comparación es “cero” habremos encontrado una unión no deseada en nuestra imagen.

Una vez ubicada la región se emplea estos comandos del Matlab para corregir dicha unión:

“bwmorph(A,’thin’)” que es una operación morfológica que elimina una capa de pixeles exteriores.

“bwmorph(A,’bridge’)” esta operación morfológica elimina pixeles solitarios en forma de “H”.

“bwmorph(A,’majority’)” que elimina pixeles si no encuentra zona con mayor cantidad de “ceros” o añade pixeles si la zona es mayoritaria de “unos”

Siendo “A” la matriz de la imagen binarizada, esto se explicó con más detalle en la tabla 4 del capítulo 2. Dando como resultado la siguiente imagen.

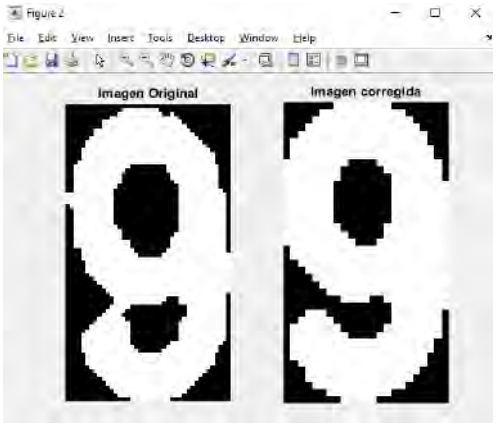


Figura 101: **Comparación de la muestra 19 del punto 4.1.1 antes y después de la corrección de errores**

Fuente: Elaboración propia (2018)

4.4.3. Detección errores de segmentación.

Este error puede deberse a la mala captura de la muestra como también a su mal binarizado, por lo que se tratara como un caso que se detectara tras la etapa de segmentación.

Para detectar este fallo se empleará un algoritmo que use el tamaño de los segmentos guardados en la interface de calibración, con estos valores se definirá que si el alto del segmento está dentro del margen aceptado pero su ancho excede los límites por mucho, este segmento se dividirá en dos partes iguales. Se decidió partir el segmento en partes iguales ya que el ancho promedio de los caracteres sin contar al "uno" es de 23 pixeles, pudiendo subir o bajar por 2 como máximo. No se contó al uno porque no se encontró escenario en el que el "uno" se encuentre unido a otro carácter por los factores descritos al inicio de este punto.

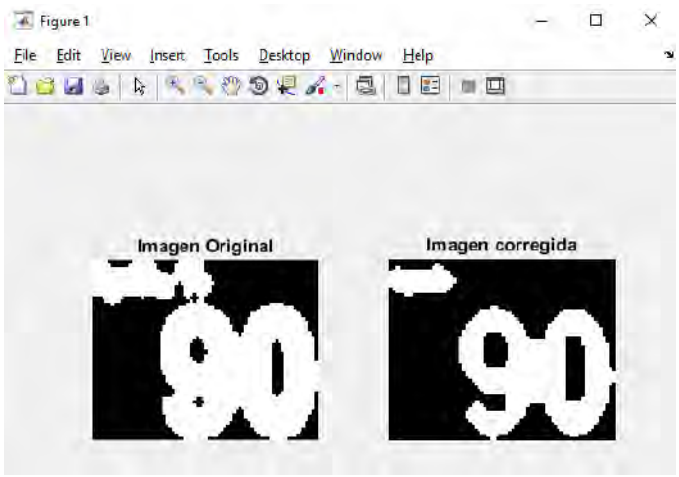


Figura 102: **Muestra 21 del punto 4.2.3.2 tras la corrección de errores de binarización.**
Fuente: Elaboración propia (2018)

Como se puede apreciar la muestra 21 tras pasar por los algoritmos anteriores mejoró notablemente su forma, pero aún mantiene la unión de sus dígitos, por lo que la corrección de errores de este punto se da antes de la segmentación para separar dichos dígitos.

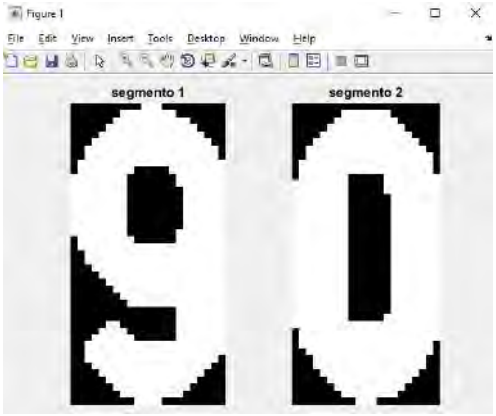


Figura 103: **Segmentación de la muestra 21 tras la corrección de errores de segmentación.**

Fuente: Elaboración propia (2018)

Tras la corrección de errores de segmentación se tienen segmentos claramente definidos y listos para ser procesados.

Con la aplicación del punto 4.4 el sistema trae consigo los siguientes resultados:

- Una sola región

Ritmo cardiaco (HR [Bpm])			Ritmo cardiaco (HR [Bpm])			Ritmo cardiaco (HR [Bpm])		
Muestra	Persona	OCR	Muestra	Persona	OCR	Muestra	Persona	OCR
1	119	*119*	34	97	*97*	67	71	*71*
2	120	*120*	35	97	*97*	68	71	*71*
3	118	*118*	36	107	*107*	69	71	*71*
4	119	*119*	37	97	*97*	70	71	*71*
5	120	*120*	38	97	*97*	71	71	*71*
6	97	*97*	39	118	*118*	72	71	*71*
7	89	*89*	40	102	*102*	73	71	*71*
8	76	*76*	41	87	*87*	74	71	*71*
9	97	*97*	42	87	*87*	75	71	*71*
10	97	*97*	43	77	*77*	76	71	*71*
11	119	*119*	44	73	*73*	77	71	*71*
12	120	*120*	45	72	*72*	78	71	*71*
13	106	*106*	46	71	*71*	79	71	*71*
14	107	*107*	47	71	*71*	80	71	*71*
15	120	*120*	48	65	*65*	81	71	*71*
16	119	*119*	49	64	*64*	82	71	*71*
17	118	*118*	50	64	*64*	83	71	*71*

18	118	*118*	51	72	*72*	84	71	*71*
19	98	*98*	52	72	*72*	85	71	*71*
20	89	*89*	53	72	*72*	86	71	*71*
21	82	*82*	54	71	*71*	87	58	*58*
22	82	*82*	55	71	*71*	88	58	*58*
23	97	*97*	56	71	*71*	89	58	*58*
24	82	*82*	57	71	*71*	90	53	*53*
25	82	*82*	58	64	*64*	91	64	*64*
26	97	*97*	59	64	*64*	92	64	*64*
27	107	*107*	60	64	*64*	93	58	*58*
28	107	*107*	61	71	*71*	94	58	*58*
29	106	*106*	62	71	*71*	95	64	*64*
30	107	*107*	63	71	*71*	96	64	*64*
31	119	*119*	64	72	*72*	97	71	*71*
32	89	*89*	65	72	*72*	98	72	*72*
33	89	*89*	66	72	*72*	99	72	*72*

Tabla 10: **Resultados de una región con detección de errores.**

Fuente: Elaboración propia (2018).

- Dos regiones con velocidad de obturación 1/250 segundos. Ritmo cardiaco.

HR BPM			HR BPM		
Muestra	**Persona**	**OCR**	**Muestra**	**Persona**	**OCR**
1	70	*70*	31	52	*52*
2	57	*57*	32	80	*80*
3	63	*63*	33	61	*61*
4	55	*55*	34	64	*64*
5	49	*49*	35	80	*80*
6	37	*37*	36	64	*64*
7	49	*49*	37	53	*53*
8	53	*53*	38	37	*37*
9	75	*75*	39	60	*60*
10	65	*65*	40	88	*88*
11	58	*58*	41	69	*69*
12	61	*61*	42	78	*78*
13	49	*49*	43	65	*65*
14	61	*61*	44	58	*58*
15	85	*85*	45	67	*67*
16	68	*68*	46	58	*58*
17	78	*78*	47	60	*60*

18	64	*64*	48	72	*72*
19	54	*54*	49	58	*58*
20	45	*45*	50	53	*53*
21	84	*84*	51	58	*58*
22	53	*53*	52	51	*51*
23	53	*53*	53	63	*63*
24	68	*68*	54	55	*55*
25	56	*56*	55	63	*63*
26	52	*52*	56	58	*58*
27	88	*88*	57	45	*45*
28	50	*50*	58	40	*40*
29	53	*53*	59	63	*63*
30	51	*51*	60	66	*66*

Tabla 11: **Resultados del ritmo cardiaco con detección de errores. 1/250.**

Fuente: Elaboración propia (2018).

- Dos regiones con velocidad de obturación 1/250 segundos. Saturación de oxígeno.

SpO2 %			SpO2 %		
Muestra	**Persona**	**OCR**	**Muestra**	**Persona**	**OCR**
1	90	*90*	31	89	*89*
2	89	*89*	32	89	*89*
3	88	*88*	33	89	*89*
4	88	*88*	34	89	*89*
5	92	*92*	35	89	*89*
6	91	*91*	36	89	*89*
7	89	*89*	37	91	*91*
8	87	*87*	38	90	*90*
9	88	*88*	39	89	*89*
10	90	*90*	40	99	*99*
11	92	*92*	41	88	*88*
12	91	*91*	42	88	*88*
13	90	*90*	43	89	*89*
14	90	*90*	44	90	*90*
15	90	*90*	45	89	*89*
16	90	*90*	46	90	*90*
17	89	*89*	47	91	*91*
18	90	*90*	48	91	*91*
19	89	*89*	49	90	*90*
20	91	*91*	50	89	*89*

21	92	*92*	51	91	*91*
22	92	*92*	52	92	*92*
23	90	*90*	53	91	*91*
24	87	*87*	54	90	*90*
25	90	*90*	55	89	*89*
26	90	*90*	56	90	*90*
27	91	*91*	57	90	*90*
28	88	*88*	58	89	*89*
29	89	*89*	59	89	*89*
30	89	*89*	60	90	*90*

Tabla 12: **Resultados de la saturación de oxígeno con detección de errores. 1/250.**

Fuente: Elaboración propia (2018).

- Dos regiones con velocidad de obturación 1/500 segundos. Ritmo cardiaco y saturación de oxígeno.

HR BPM			SpO2 %		
Muestra	**Persona**	**OCR**	**Muestra**	**Persona**	**OCR**
1	44	44	1	92	92
2	50	50	2	92	92
3	64	64	3	91	91
4	84	84	4	89	89
5	78	78	5	88	88
6	64	64	6	90	90
7	61	61	7	92	92
8	81	81	8	91	91
9	79	79	9	91	91
10	64	64	10	90	90
11	57	57	11	89	89
12	67	67	12	88	88
13	61	61	13	85	85
14	50	50	14	85	85
15	44	44	15	91	91
16	44	44	16	93	93
17	47	47	17	91	91
18	49	49	18	88	88
19	67	67	19	88	88
20	58	58	20	90	90
21	42	42	21	91	91
22	42	42	22	90	90

23	42	42	23	89	89
24	49	49	24	89	89
25	54	54	25	90	90
26	46	46	26	90	90
27	67	67	27	90	90
28	64	64	28	90	90
29	49	49	29	90	90

Tabla 13: **Resultados del ritmo cardiaco y SpO2 con detección de errores. 1/500.**

Fuente: Elaboración propia (2018).

- Dos regiones con velocidad de obturación 1/500 segundos. Ritmo cardiaco y saturación de oxígeno, con la pantalla cubierta.

HR BPM			SpO2 %		
Muestra	**Persona**	**OCR**	**Muestra**	**Persona**	**OCR**
1	40	*40*	1	91	*91*
2	43	*43*	2	90	*90*
3	47	*47*	3	89	*89*
4	60	*60*	4	90	*90*
5	64	*64*	5	91	*91*
6	57	*57*	6	90	*90*
7	61	*61*	7	90	*90*
8	70	*70*	8	90	*90*
9	66	*66*	9	91	*91*
10	66	*66*	10	93	*93*
11	51	*61*	11	95	*95*
12	86	*86*	12	96	*96*
13	63	*63*	13	95	*95*
14	42	*42*	14	96	*96*
15	34	*34*	15	95	*95*
16	38	*38*	16	95	*95*
17	52	*52*	17	95	*95*
18	56	*56*	18	96	*96*
19	64	*64*	19	92	*92*
20	86	*86*	20	91	*91*
21	92	*92*	21	90	*90*
22	58	*58*	22	88	*88*
23	49	*49*	23	88	*88*
24	58	*58*	24	90	*90*
25	66	*66*	25	91	*91*

26	69	*69*	26	89	*89*
27	60	*60*	27	88	*88*
28	58	*58*	28	89	*89*
29	64	*64*	29	90	*90*
30	63	*63*	30	90	*90*
31	50	*50*	31	88	*88*
32	59	*59*	32	86	*86*
33	57	*57*	33	87	*87*

Tabla 14: **Resultados del ritmo cardiaco y SpO2 con detección de errores. 1/500 con monitor y cámara cubiertos.**

Fuente: Elaboración propia (2018).

- Dos regiones con el monitor y la cámara cubiertos. Ritmo cardiaco

HR BPM			HR BPM		
Muestra	**Persona**	**OCR**	**Muestra**	**Persona**	**OCR**
1	67	*67*	27	60	*60*
2	70	*70*	28	71	*71*
3	58	*58*	29	158	*158*
4	53	*53*	30	244	*244*
5	58	*58*	31	271	*271*
6	111	*111*	32	96	*96*
7	266	*266*	33	59	*59*
8	90	*90*	34	49	*49*
9	114	*114*	35	63	*63*
10	69	*69*	36	61	*61*
11	60	*60*	37	107	*107*
12	49	*49*	38	287	*287*
13	58	*58*	39	126	---
14	49	*49*	40	162	*162*
15	39	*39*	41	160	*160*
16	46	*46*	42	246	*246*
17	40	*40*	43	178	*178*
18	46	*46*	44	61	*61*
19	63	*63*	45	140	*140*
20	61	*61*	46	86	*86*
21	80	*80*	47	49	*49*
22	63	*63*	48	142	*142*
23	45	*45*	49	106	*106*
24	42	*42*	50	100	*100*

25	33	*33*	51	264	*264*
26	37	*37*			

Tabla 15: **Resultados del ritmo cardiaco con detección de errores. Con el monitor y la cámara cubiertos.**

Fuente: Elaboración propia (2018).

- Dos regiones con el monitor y la cámara cubiertos. Saturación de oxígeno.

SpO2 %			SpO2 %		
Muestra	**Persona**	**OCR**	**Muestra**	**Persona**	**OCR**
1	93	*93*	27	91	*91*
2	92	*92*	28	90	*90*
3	90	*90*	29	90	*90*
4	90	*90*	30	91	*91*
5	89	*89*	31	91	*91*
6	90	*90*	32	90	*90*
7	91	*91*	33	89	*89*
8	90	*90*	34	90	*90*
9	90	*90*	35	91	*91*
10	90	*90*	36	91	*91*
11	89	*89*	37	91	*91*
12	89	*89*	38	90	*90*
13	89	*89*	39	89	*89*
14	90	*90*	40	89	*89*
15	90	*90*	41	89	*89*
16	90	*90*	42	92	*92*
17	90	*90*	43	95	*95*
18	90	*90*	44	95	*95*
19	89	*89*	45	94	*94*
20	89	*89*	46	-	
21	90	*90*	47	-	
22	90	*90*	48	-	
23	90	*90*	49	-	
24	90	*90*	50	-	
25	91	*91*	51	-	
26	92	*92*			

Tabla 16: **Resultados de la saturación de oxígeno con detección de errores. Con el monitor y la cámara cubiertos.**

Fuente: Elaboración propia (2018).

De todas estas tablas se puede observar como la precisión del sistema alcanza un grado de precisión muy alto, pero en cada caso se tomó pocas muestras por lo que se tomaron muestras estáticas para determinar el porcentaje de error de comparación.

Para ello, se dejó al sistema con una imagen fija de un número por un tiempo prolongado para ver con más detalle la precisión del sistema y cuan presente está el error de comparación.

Se hizo la prueba durante la noche (a partir de las 11 pm) en tres ocasiones distintas, las tres por un periodo de 6 horas, la primera con el número fijo de “96”, la segunda con el “90” y la tercera con el número de “44”.

En el primer caso se generó un total de 4302 muestras, con 43 muestras erradas.

En el segundo caso se generó un total de 4298 muestras, con 34 muestras erradas.

Y en el tercer caso se generó un total de 4294 muestras, con 17 muestras erradas.

Para la correcta realización de estas pruebas se tuvo que tener la habitación correctamente iluminada, la cámara fija con un trípode, se debe configurar el monitor para que no entre en estado de reposo, y el trípode de la cámara debe reposar sobre un piso rígido preferiblemente de concreto.

Obteniendo con ello un error de reconocimiento del 0.999% para el primer caso. 0.791% para el segundo caso, y un error de 0.396% para el tercer caso.

Con lo que podemos decir que la precisión general del sistema es del 99% tras la aplicación de la detección y reducción de errores.

CONCLUSIONES

- Se logró desarrollar un sistema que permite captar los parámetros numéricos de un monitor desfibrilador mediante el uso de una webcam ubicada cerca del monitor (30 cm, aprox) que no altera el normal funcionamiento del monitor, digitalizarlos empleando el método de reconocimiento kNN y guardar los datos de frecuencia cardiaca, así como saturación de oxígeno de manera organizada en archivos .xls.
- El método de clasificación de caracteres que se utilizó fue el algoritmo kNN se aplicó dicho método ya que este tiene un bajo coste computacional y no requiere de muchas muestras de entrenamiento, se estudió los errores del sistema en diferentes situaciones.

  Primero se probó el sistema con luz diurna reconociendo una sola región (F.C), con un total de 100 muestras se obtuvo una eficacia del 98%.

  Luego, se probó el sistema reconociendo dos regiones (F.C y SPO2) para este caso se consideraron tres escenarios como: para el primer escenario se captó dos regiones poniendo la velocidad de obturación de la cámara en 1/250'' obteniendo una precisión de 93.3% para la FC y de 95% para el SPO2 de un total de 120 muestras. En el segundo escenario se puso la velocidad de obturación en 1/500'' obteniendo una precisión del 89.7% para la FC y de un 96.6% para el SPO2 de un total de 58 muestras. Por último el tercer escenario se cubrió completamente el monitor para simular un ambiente nocturno escaso de iluminación con una velocidad de obturación de 1/500'' obteniendo una precisión del 91% tanto en la FC como en el SPO2.

  Con esto se pudo ver el desempeño del sistema ante diversas situaciones, viendo que hay un incremento del error a menor iluminación y reduciendo el tiempo de exposición a la luz del obturador de la cámara.

- Por último, se estudió y clasificó los errores cometidos por el sistema, para ello se contrasto los datos obtenidos por el sistema con los obtenidos manualmente, encontrando que un 0.73% son debidos a errores al momento de capturar las imágenes. Un 3.6% son debidos a que no son binarizados correctamente. Un 3.3% son causados a una mala segmentación de los caracteres, y por ultimo un 0.4% por error de clasificación que viene siendo un error propio del método de reconocimiento kNN que no asigno adecuadamente el valor numérico correspondiente. Con esto el sistema logró alcanzar una precisión máxima de 98% y una mínima de 90%.

RECOMENDACIONES

- Si se requiere trabajar con cámaras IP, es recomendable revisar que estas puedan comunicarse con Matlab para ello, dicho software tiene en su página web una relación de las marcas y modelos con los que trabaja adecuadamente. Esto debido mayormente a los codificadores de video que usan dichas cámaras.
- Se recomienda calibrar los parámetros antes de procesar las imágenes de un nuevo monitor ya que, para obtener una imagen binaria adecuada, si se deseara mejorar el proyecto para que sea completamente automático se debería contar con un sistema de calibración de brillo y contraste automático antes del bucle de procesamiento.
- Es recomendable hacer las capturas de las muestras, con luz blanca artificial, puesto que la luz natural (horas diurnas) generan demasiados brillos y reflejos molestos.
- Se recomienda también no calibrar el obturador de la cámara de forma manual, ya que la mayoría de ellas puede auto regular su obturador, dependiendo de la cantidad de luz que haya en la habitación donde se toman las muestras.
- Se recomienda un sistema de detección y reducción de errores como el que se plantea en la parte final del proyecto, que buscó minimizar los errores ocurridos en el sistema, haciendo uso de histogramas para poder identificarlos, así como funciones de discriminación tales como la detección del tamaño de cada segmento o la detección de uniones por pixeles en los caracteres, que descartan la toma de datos incorrectos.

REFERENCIAS

[1] Ayatullah Faruk Mollah, Nabamita Majumder, Subhadip Basu and Mita Nasipuri, 2011. Design of an Optical Character Recognition System for Camerabased Handheld Devices.

[2] N. Vázquez, M. Nakano & H. Pérez-Meana (2002). AUTOMATIC SYSTEM FOR LOCALIZATION AND RECOGNITION OF VEHICLE PLATE NUMBERS.

[3] O. Matei, P.C. Pop, H. V˘alean, 2013. Optical Character Recognition in Real Environments using Neural Networks and k-Nearest Neighbor

[4] Felipe Alberto Cifuentes Ramos, 2016. Clasificación automática de Tweets utilizando K-NN y K-Means como algoritmos de clasificación automática, aplicando TF-IDF y TF-RFL para las ponderaciones

[5] Ana K. Aguilar R. (2006, enero). Monitor de signos vitales [HomePage]. Consultado el día 26 de enero de 2017 de la World Wide Web http://es.slideshare.net/AnnieAguilar/monitor-de-signos-vitales-31603365

[6] Rafael C. González, Richard E. Woods, Steven L. Eddins. Diciembre 2003. Digital Image Processing Using MATLAB. ISBN-10: 0130085197.

[7] Matías Echenique. (2002, noviembre). Consideraciones generales de la teoría cromática [HomePage].
Consultado el día 28 de enero de 2017 de la World Wide Web
https://es.scribd.com/document/460247705/El-color

[8] Nicolás Aguirre Dobernack 2013. Implementación de un Sistema de Detección de Señales de Tráfico Mediante Visión Artificial Basado en FPGA

[9] Jesús Antonio Vega Uribe, Marlon Alfonso Reyes Figueroa. 2006. Transformaciones lineales y no lineales para Espacios de Color en procesamiento digital de imágenes. Revista Internacional de Métodos Numéricos para Cálculo y Diseño en Ingeniería. Vol. 22,3 223-240.

[10] Jahne, B. Digital Image Processing. 2002. 5th and extended edition. ISBN 3-540-67754-2, Germany: Springer.

[11] Petrou, M., P. Bosdogianni. 1999. Image Processing: the fundamentals. ISBN 0-471-9883-4, USA: John Wiley and Sons.

[12] A. Rosenfeld and J. Pfaltz, Oct. 1966. “Sequential operations in digital picture processing,” Journal of the ACM, vol. 13, no. 1, pp. 471–494.

[13] Erik Valdemar Cuevas Jimenez Daniel Zaldivar Navarro. (s.f.) Visión por Computador utilizando MatLAB y el Toolbox de Procesamiento Digital de Imágenes.

[14] Documentation - MATLAB & Simulink - MathWorks América Latina
Consultado el día 30 de enero de 2017 de la World Wide Web
https://la.mathworks.com/help/index.html

[15] Luis Guillermo Moré Rodríguez, diciembre 2017. Mejora Del Contraste De Imágenes A Color Utilizando Un Framework De Optimización Multiobjetivo. Universidad Nacional de Asunción – Paraguay.

[16] Nobuyuki Otsu. 1979. A threshold selection method from grey level histograms. In: IEEE Transactions on Systems, Man, and Cybernetics. New York. S.62–66. ISSN 1083-4419.

[17] Kulturaren Euskal Behatokia, abril-2011. "KULTURA 2.0 - Píldoras de formación: OCR: tecnología para el reconocimiento óptico de caracteres en una imagen."

[18] Carlos J. Sanchez F. & Victor Sandonis C. (s.f.). Reconocimiento óptico de caracteres (ocr)

[19] Blanca Reina Choque García. 2015 "diseño de un autómata para la traducción del idioma chino al español". Universidad Mayor De San Andrés

[20] José R. Hilera González, Juan P. Romero Villaverde, José A. Gutiérrez de Mesa. (s.f.) Sistema De Reconocimiento Óptico De Caracteres (Ocr) Con Redes Neuronales.

[21] Cristina García Cambronero, Irene Gómez Moreno. (s.f). Algoritmos De Aprendizaje: KNN & Kmeans. Universidad Carlos III de Madrid.

[22] Classification Using Nearest Neighbors - MATLAB & Simulink - MathWorks América Latina. Consultado el día 01 de febrero de 2017 de la World Wide Web https://la.mathworks.com/help/stats/classification-using-nearest-neighbors.html

[23] Faisal Mohammad, Jyoti Anarase, Milan Shingote, Pratik Ghanwat, 2014. Optical Character Recognition Implementation Using Pattern Matching.

GLOSARIO

Archivos "*.xls": Son los documentos de salida del programa Microsoft Excel, siendo ahora la extensión "*.xlsx" la más actual y que se mantiene desde el 2007.

Asistentes digitales personales (PDA): El asistente digital personal, también conocido como computadora de bolsillo, organizador personal o agenda electrónica de bolsillo, es una computadora portátil originalmente diseñada como una agenda personal electrónica con un sistema de reconocimiento de escritura que puede usar para dibujar, calendarios, listas de contactos, recordatorios y otras funciones.

Base de datos: es un conjunto de datos que se encuentran en el mismo contexto y se han almacenado de manera organizada para su uso posterior.

Bit: Es el acrónimo de "dígito binario". Un dígito en un sistema de numeración binario se denomina bits.

Bins: Es una forma de agrupación de datos en donde la distribución de estos es casi uniforme, por lo que no tiene una medida fija y depende del atributo que se esté evaluando.

Binarización: Es un método que utiliza bucles o recursividad para realizar un barrido la matriz de imagen digital para reducir la escala de grises a dos valores.

Blocksets: Son paquetes de bloques preestablecidos del paquete de simulación Simulink que viene incluido dentro del software Matlab.

Byte: El mínimo componente de memoria direccionable de una computadora es un conjunto de 8 bits que recibe el tratamiento de una unidad.

Cantidad de frames: La velocidad (tasa) a la que un dispositivo muestra imágenes llamadas cuadros o fotogramas, también conocidas como cuadros o imágenes por segundo.

CCTV (Circuito Cerrado de Televisión): Es una tecnología de videovigilancia que permite la supervisión de una variedad de entornos y actividades. Donde todos sus componentes están enlazados.

Cifra: Se define así a un número o la unión de ellos que indican una cantidad.

Correlación: Es una medida de la similitud entre dos señales.

Cross-Platform: Plataforma digital que puede ser usada por distintos sistemas operativos, así como también dispositivos.

Dígito: Es un número o carácter individual que posee valor numérico.

Dilatar una imagen: Es el proceso en el que se agrandan los bordes de una figura y se llenan los espacios encerrados, cuanto más pequeños sean dichos espacios.

Display de 7 segmentos: Es un método para representar caracteres en dispositivos electrónicos. Está compuesto de siete partes, cada una de las cuales se puede encender o apagar por sí sola. Cada parte tiene la forma de una pequeña línea.

EIA-485: O más común conocida como RS-458, es un estándar de comunicaciones para buses en la capa física del Modelo de Interconexión de Sistemas de Interconexión (OSI). El medio físico de transmisión es un par trenzado que puede alojar 32, 128 o 256 estaciones en un solo par. Tiene una longitud máxima de 1200 metros y opera a 300 a 19 200 bit/s y puede comunicarse en half-duplex (semiduplex) dependiendo de la cantidad de tráfico de cada driver.

Erosionar una imagen: Es un proceso morfológico en el que a través de una razón morfológica la imagen pierde grosor en los bordes haciéndola más delgada o si la razón excede al grosor de la imagen esta desaparece.

Estación de trabajo SUN: Es un computador de alto rendimiento diseñado para tareas científicas o técnicas. Es una computadora en una red de computadoras que permite a los

usuarios acceder a los servidores y otros componentes de la red. SUN es la compañía que comercializa estas estaciones de trabajo con un sistema operativo propio tipo Unix

E-HEALTH: El término se refiere al conjunto de Tecnologías de la Información y la Comunicación (TICs) que se utilizan en el entorno de la salud como herramientas para la prevención, diagnóstico, tratamiento, seguimiento y gestión de la salud, lo que reduce los costos del sistema de salud y mejora su eficacia.

FilledImage: Es una operación de Matlab donde si un objeto tiene una región cerrada, esta llenara el espacio interior dando como resultado una figura sólida.

Fotograma: Es una sola imagen perteneciente a un grupo mayor de imágenes obtenidas por un mismo objeto, sea una cámara, que en su conjunto generan un video o en su defecto una animación.

GUIDE: Es el motor de creación de interfaces de usuario, perteneciente al software de Matlab, que permite crear aplicaciones sencillas de manejar.

IP: En informática, se refiere al protocolo de internet, cuya función principal es la transmisión de datos en origen o destino a través de un protocolo no orientado a conexión que transfiere paquetes conmutados a través de redes físicas previamente enlazadas según la norma de enlace de datos Open Systems Interconnection (OSI).

K-d-trees: Una estructura de datos de particionado del espacio conocida como árbol KD organiza los puntos en un espacio euclídeo de k dimensiones.

La tecnología HTML5: Es la quinta importante revisión del lenguaje fundamental de la World Wide Web, HTML. HTML5 establece dos opciones de sintaxis para HTML: una que se denomina HTML (text/html) tradicional, también conocida como HTML5, y otra que se denomina XHTML5, que debe utilizarse con sintaxis XML (application/xhtml+xml).

Marco táctil de Sencha: Sencha Touch es un marco con código libre que permite a los programadores crear webs que parecen aplicaciones nativas para teléfonos Android e iOS.

Los resultados con HTML5 y CSS3 pueden ser impresionantes, lo que facilita la integración con mapas, la carga de datos con ajax y las animaciones.

Metodo de Otsu: Un conjunto de algoritmos conocidos como métodos de valor umbral tienen como objetivo dividir gráficos rasterizados, es decir, separar los objetos de una imagen que nos interesa del resto. Los métodos de valor umbral se pueden utilizar para determinar qué píxeles conforman los objetos que buscamos y qué píxeles son solo el entorno de estos objetos en situaciones más básicas. Este método es particularmente útil para diferenciar el texto de un documento del fondo de la imagen, como papel amarillento, con manchas y arruguitas, y así garantizar que se obtenga el texto correcto al realizar el reconocimiento óptico de texto (OCR).

ORC: Acrónimo del Ingles de reconocimiento óptico de caracteres.

Roipoly: Es un comando específico de Matlab que crea una herramienta interactiva poligonal asociada a la imagen actual mostrada en una figura, que se emplea en el procesamiento de imágenes.

R-trees: Los árboles R o árboles R son estructuras de datos de tipo árbol similares a los árboles B, pero se utilizan para métodos de acceso espacial, como indexar información multidimensional, como las coordenadas (x, y) de un lugar geográfico.

Sensor de imagen CMOS: es un sensor que detecta la luz basado en tecnología CMOS.

Simulink: Es un entorno de programación visual basado en Matlab. Es un entorno de programación con mayor abstracción que Matlab.

Sistema Plug and Play: es la tecnología o cualquier progreso que permite a un dispositivo informático conectarse a una computadora sin tener que configurar, mediante

jumpers o software específico proporcionado por el fabricante, ni proporcionar parámetros a sus controladores. El sistema operativo de la computadora debe soporte para dicho dispositivo para que sea posible.

Spur: Es el término que usa Matlab para referirse a pixeles que sobresalen de una estructura quedando como protuberancias o salientes que se presentan en una figura sólida.

Toolboxes: Son herramientas prediseñadas para facilitar la programación o diseño de un sistema digital.

UTP: Son las siglas del inglés de "Par Trenzado no blindado" por lo que se refiere a un tipo de cable que suele emplearse en las telecomunicaciones.

Voronoi polygons: Son una construcción geométrica que permite construir una partición del plano euclídeo basada en la distancia euclidiana. Esto es particularmente útil para casos de datos cualitativos. Los puntos se unen trazando las mediatrices de los segmentos de unión. Una serie de polígonos se forman en un espacio bidimensional alrededor de un conjunto de puntos de control a través de las intersecciones de estas mediatrices. Esto hace que el perímetro de los polígonos generados sea equidistante a los puntos cercanos y designe su área de influencia.

Webcam: Cámara digital de uso inmediato y conexión directa al computador.

ANEXOS.

I. Métodos más comunes en el reconocimiento de caracteres.

Entre los principales métodos se encuentran:

K-NN: Debido a su sencillez y a una serie de propiedades estadísticas bien conocidas, este método es muy popular. Estas propiedades estadísticas le brindan un buen comportamiento para abordar una variedad de problemas de clasificación.

Análisis estructural: identifica los caracteres mediante el examen de las formas de sub características de la imagen, los histogramas sub-verticales y horizontales. Su capacidad de reparación de caracteres es ideal para textos de baja calidad y noticias.

Correlación de matrices: convierte cada carácter en un patrón dentro de una matriz y luego compara el patrón con un índice de caracteres conocidos. Su reconocimiento es más fuerte en fuentes monotipo y en las páginas uniformes de una sola columna.

Redes neuronales: Esta estrategia simula la forma en que funciona el sistema neuronal humano; muestrea los píxeles en cada imagen y los combina con un índice conocido de patrones de píxeles de caracteres. La capacidad de reconocer los caracteres a través de la abstracción es ideal para documentos fijos y texto dañado. Las redes neuronales son ideales para tipos específicos de problemas, como procesar datos del mercado de valores o encontrar tendencias en patrones gráficos. En todos estos enfoques, las redes neuronales son más eficientes que otras.

II. Diseño del sistema de red para enlazar varias cámaras.

Esquema de red.

Como se propuso anteriormente en las limitaciones del proyecto esta etapa del proyecto solo se diseñó mas no se implementó. Por lo que solo mostraremos el esquema a seguir para poder desarrollar un servidor basado en varias cámaras y un centro de control.

Figura 104: **Diagrama de red propuesta.**

Fuente: Elaboración propia (2017).

El sistema consta de un servidor dedicado que de acceso a cualquier usuario que se conecte a la red, también se posee un centro de procesamiento que tendrá un papel importante ya que debe ser capaz de aplicar el OCR de varias cámaras en simultaneo, también se prevé un Disco de almacenamiento externo para no saturar el disco del centro de procesamiento, este disco para ahorrar costes puede ser el equipo del usuario. Por último, también se tendrá un enrutador que enlace al usuario con el servidor, el centro de procesamiento y el posible disco de almacenamiento.

El sistema de red empieza con el usuario, seleccionando una cámara a través del servidor, con la cual hará el proceso previo de calibración y elegirá el dispositivo que almacene las imágenes adquiridas por la cámara, también elegirá donde se creará el archivo Excel que se almacenaran los datos. Luego de los pasos previos y se dé inicio al proceso del reconocimiento óptico de caracteres, el centro de procesamiento accederá al disco en el que se están guardando las imágenes obtenidas por la cámara e iniciará el proceso de reconocimiento, cabe resaltar que el centro de procesamiento generará subprocesos por cada cámara de la que se requiera el OCR, para así no saturarse en el procesamiento de varias señales de cámaras en simultaneo. Los datos de los caracteres reconocidos se irán guardando en el disco seleccionado en archivos de Microsoft Excel. El usuario será capaz de detener el proceso en el momento que este lo requiera y el sistema le devolverá los datos reconocidos hasta ese momento.

III. Funcionamiento general del Proyecto.

Como se expuso en el capítulo 3 el proyecto está diseñado para leer las imágenes obtenidas por la webcam. Luego tratarlas y generar un archivo de Excel en el que se almacenen los datos obtenidos de las imágenes. Para ello se sigue el siguiente procedimiento.

Se inicia con la interface de usuario “Inicio” en la que se realizara la configuración inicial de los parámetros a reconocer, explicado más detalladamente en el capítulo 3.

Tras la configuración inicial se abrirá otra ventana en la que podremos calibrar los parámetros de binarización, una vez configurado nos lleva otra interface en donde se podrá ver las regiones capturadas y su conversión a imagen binaria, si esta se encuentra correctamente procesada se procede a inicializar el bucle de reconocimiento de los caracteres. Esto con el botón “Iniciar OCR” que se puede observar en la figura 29, del capítulo 3.

Por último, se procede a esperar que tome la cantidad de muestras que se requieran reconocer o detener el proceso con el botón “Detener” que se observa en la figura 51, del capítulo 4, y tendremos el archivo de Excel creado en la carpeta contenedora del proyecto.

Cabe resaltar que al iniciar el bucle la cámara irá grabando en video todo el proceso, pero no guardará cada imagen que sirven de muestra ni su respectivo procesamiento, ya que esto ralentiza el proceso y ocupa más espacio en el disco duro del ordenador.

Todos los archivos tanto de programación como archivos temporales generados por el proyecto se guardarán en la carpeta contenedora del proyecto esto para evitar problemas de direccionamiento dentro de Matlab.

Para la elaboración se realizaron varios Script, cada uno de ellos con una función en específico, para poder organizarlos a cada Script se le dio un nombre, por lo que en el siguiente anexo se mencionaran cada uno de ellos como título de cada sección de código

IV. Código en Matlab empleado en el sistema.

a. Adquisición de imágenes a través de la webcam con GUIde.

```
function varargout = Inicio(varargin)

gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Inicio_OpeningFcn, ...
                   'gui_OutputFcn',  @Inicio_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before Inicio is made visible.
function Inicio_OpeningFcn(hObject, eventdata, handles, varargin)

clc;
handles.output = hObject;
global vid % definiendo a la variable que contendra los datos de la c mara
axes(handles.axes2); % llamando al objeto en que se visualizar  los datos
obtenidos por la c mara
vid=videoinput('winvideo',1); % se emplea el decodificador de video de windows
vid.FramesPerTrigger = 1; % Cuadros a visualizarse por segundo
vid.TriggerRepeat = Inf; % Repeticion del evento: Hasta una nueva orden
triggerconfig(vid, 'Manual');
hImage=image(zeros(2080,1170,3),'Parent',handles.axes2); % Definiendo el tama o de
"axes2"
preview(vid,hImage); % Obteniendo los datos de la c mara
% Update handles structure
guidata(hObject, handles);

% --- Outputs from this function are returned to the command line.
function varargout = Inicio_OutputFcn(hObject, eventdata, handles)

varargout{1} = handles.output;


function Regiones_Callback(hObject, eventdata, handles)

function Regiones_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
% --- Executes on button press in Recorte.
function Recorte_Callback(hObject, eventdata, handles)

region = str2double(get(handles.Regiones,'string'));
if exist ('Regiones.mat')
    delete ('Regiones.mat');
end
if region < 1
    errordlg('Debe tener al menos una region','Valor Incorrecto');
    region=0;
    save('Regiones.mat','region');
elseif region > 4
    errordlg('Solo se puede escoger hasta un m ximo de 4 regiones','Valor
Incorrecto');
    region=0;
    save('Regiones.mat','region');
elseif region == 3
    errordlg('Seccion no disponible','No Disponible');
elseif region == 4
    errordlg('Seccion no disponible','No Disponible');
else
    save('Regiones.mat','region');
    cutregions
end

% --- Executes on button press in imagen.
function imagen_Callback(hObject, eventdata, handles)

if exist('TEMP/Camera/Calib/' , 'dir')
    rmdir('TEMP/Camera/Calib/' , 's');
end
    mkdir ('TEMP/Camera/Calib/' );
global vid in
capturedimage = getsnapshot(vid); % Captura de un cuadro
imshow(capturedimage); % Mostramos la imagen en "axes2"
imwrite(capturedimage,'TEMP/Camera/Calib/test.png'); % se guarda esta captura para
su calibracion
```

b. Selección de regiones a capturar.

```
function cutregions ()
clc
close all
%% LIMPIAMOS VALORES ANTERIORES
if exist ('coordenadas.mat')
    delete ('coordenadas.mat');
end
if exist('TEMP/Regiones_calib/1_reg/' , 'dir')
    rmdir('TEMP/Regiones_calib/1_reg/' , 's');
end
    mkdir ('TEMP/Regiones_calib/1_reg/' );
if exist('TEMP/Regiones_calib/2_reg/' , 'dir')
    rmdir('TEMP/Regiones_calib/2_reg/' , 's');
end
    mkdir ('TEMP/Regiones_calib/2_reg/' );
if exist('TEMP/Regiones_calib/3_reg/' , 'dir')
    rmdir('TEMP/Regiones_calib/3_reg/' , 's');
end
    mkdir ('TEMP/Regiones_calib/3_reg/' );
if exist('TEMP/Regiones_calib/4_reg/' , 'dir')
    rmdir('TEMP/Regiones_calib/4_reg/' , 's');
end
    mkdir ('TEMP/Regiones_calib/4_reg/' );
%% lectura de la imagen
%test=imread('testSet/img_gotten/5_001.png');
Lista = dir('TEMP/Camera/Calib/*.png');
for nImage=1:length(Lista)
    test = imread(['TEMP/Camera/Calib/' Lista(nImage).name]);
end
figure; image(test);
title('Imagen Original');


%% cargamos valor de regiones
load ('Regiones.mat');

%% lectura de regiones
if region == 1
    r1 = ginput(2); % ginput selecciona las coordenadas de "2" puntos que
seleccionaremos
    % Get the x and y corner coordinates as integers
    xmin1 = min(floor(r1(1)), floor(r1(2)));
    ymin1 = min(floor(r1(3)), floor(r1(4)));
    xmax1 = max(ceil(r1(1)), ceil(r1(2)));
    ymax1 = max(ceil(r1(3)), ceil(r1(4)));
    % Emplearemos las coordenadas que seleccionamos como parametros para recortar
la imagen
    ancho1 = xmax1 - xmin1;
    alto1 = ymax1 - ymin1;
    %Creamos la nueva imagen en base a las coordenadas
    corte1 = test(ymin1:ymax1, xmin1: xmax1,:);
    REGCUT=cat(2, xmin1, ymin1, ancho1, alto1);
    % Guardamos la imagen recortada como archivo temporal
    imwrite(corte1,'TEMP/Regiones_calib/1_reg/corte_1.png');
    % Save coords and close figures
    save('coordenadas.mat','REGCUT');
```

```
    close;
    calib_reg1;
    %binarizacion;
    %region1;
elseif region == 2
    r1 = ginput(2);
    % Get the x and y corner coordinates as integers
    xmin1 = min(floor(r1(1)), floor(r1(2)));
    ymin1 = min(floor(r1(3)), floor(r1(4)));
    xmax1 = max(ceil(r1(1)), ceil(r1(2)));
    ymax1 = max(ceil(r1(3)), ceil(r1(4)));
    ancho1 = xmax1 - xmin1;
    alto1 = ymax1 - ymin1;
    coord_reg1=cat(2, xmin1, ymin1, ancho1, alto1);
    r2 = ginput(2);
    % Get the x and y corner coordinates as integers
    xmin2 = min(floor(r2(1)), floor(r2(2)));
    ymin2 = min(floor(r2(3)), floor(r2(4)));
    xmax2 = max(ceil(r2(1)), ceil(r2(2)));
    ymax2 = max(ceil(r2(3)), ceil(r2(4)));
    ancho2 = xmax2 - xmin2;
    alto2 = ymax2 - ymin2;
    coord_reg2=cat(2, xmin2, ymin2, ancho2, alto2);
    %Creamos la nueva imagen en base a las coordenadas
    corte1 = test(ymin1:ymax1, xmin1: xmax1,:);
    corte2 = test(ymin2:ymax2, xmin2: xmax2,:);
    REGCUT = cat (1, coord_reg1, coord_reg2);
    imwrite(corte1,'TEMP/Regiones_calib/1_reg/corte_1.png');
    imwrite(corte2,'TEMP/Regiones_calib/2_reg/corte_2.png');
    % Save coords and close figures
    save('coordenadas.mat','REGCUT');
    close;
    binarizacion;
    region2;
elseif region == 3
    r1 = ginput(2);
    % Get the x and y corner coordinates as integers
    xmin1 = min(floor(r1(1)), floor(r1(2)));
    ymin1 = min(floor(r1(3)), floor(r1(4)));
    xmax1 = max(ceil(r1(1)), ceil(r1(2)));
    ymax1 = max(ceil(r1(3)), ceil(r1(4)));
    ancho1 = xmax1 - xmin1;
    alto1 = ymax1 - ymin1;
    coord_reg1=cat(2, xmin1, ymin1, ancho1, alto1);
    r2 = ginput(2);
    % Get the x and y corner coordinates as integers
    xmin2 = min(floor(r2(1)), floor(r2(2)));
    ymin2 = min(floor(r2(3)), floor(r2(4)));
    xmax2 = max(ceil(r2(1)), ceil(r2(2)));
    ymax2 = max(ceil(r2(3)), ceil(r2(4)));
    ancho2 = xmax2 - xmin2;
    alto2 = ymax2 - ymin2;
    coord_reg2=cat(2, xmin2, ymin2, ancho2, alto2);
    REGCUT = cat (1, coord_reg1, coord_reg2);
    r3 = ginput(2);
    % Get the x and y corner coordinates as integers
    xmin3 = min(floor(r3(1)), floor(r3(2)));
    ymin3 = min(floor(r3(3)), floor(r3(4)));
    xmax3 = max(ceil(r3(1)), ceil(r3(2)));
```

```
    ymax3 = max(ceil(r3(3)), ceil(r3(4)));
    ancho3 = xmax3 - xmin3;
    alto3 = ymax3 - ymin3;
    coord_reg3=cat(2, xmin3, ymin3, ancho3, alto3);
    %Creamos la nueva imagen en base a las coordenadas
    corte1 = test(ymin1:ymax1, xmin1: xmax1,:);
    corte2 = test(ymin2:ymax2, xmin2: xmax2,:);
    corte3 = test(ymin3:ymax3, xmin3: xmax3,:);
    REGCUT = cat (1, coord_reg1, coord_reg2, coord_reg3);
    imwrite(corte1,'TEMP/Regiones_calib/1_reg/corte_1.png');
    imwrite(corte2,'TEMP/Regiones_calib/2_reg/corte_2.png');
    imwrite(corte3,'TEMP/Regiones_calib/3_reg/corte_3.png');
    % Save coords and close figures
    save('coordenadas.mat','REGCUT');
    close;
elseif region == 4
    r1 = ginput(2);
    % Get the x and y corner coordinates as integers
    xmin1 = min(floor(r1(1)), floor(r1(2)));
    ymin1 = min(floor(r1(3)), floor(r1(4)));
    xmax1 = max(ceil(r1(1)), ceil(r1(2)));
    ymax1 = max(ceil(r1(3)), ceil(r1(4)));
    ancho1 = xmax1 - xmin1;
    alto1 = ymax1 - ymin1;
    coord_reg1=cat(2, xmin1, ymin1, ancho1, alto1);
    r2 = ginput(2);
    % Get the x and y corner coordinates as integers
    xmin2 = min(floor(r2(1)), floor(r2(2)));
    ymin2 = min(floor(r2(3)), floor(r2(4)));
    xmax2 = max(ceil(r2(1)), ceil(r2(2)));
    ymax2 = max(ceil(r2(3)), ceil(r2(4)));
    ancho2 = xmax2 - xmin2;
    alto2 = ymax2 - ymin2;
    coord_reg2=cat(2, xmin2, ymin2, ancho2, alto2);
    REGCUT = cat (1, coord_reg1, coord_reg2);
    r3 = ginput(2);
    % Get the x and y corner coordinates as integers
    xmin3 = min(floor(r3(1)), floor(r3(2)));
    ymin3 = min(floor(r3(3)), floor(r3(4)));
    xmax3 = max(ceil(r3(1)), ceil(r3(2)));
    ymax3 = max(ceil(r3(3)), ceil(r3(4)));
    ancho3 = xmax3 - xmin3;
    alto3 = ymax3 - ymin3;
    coord_reg3=cat(2, xmin3, ymin3, ancho3, alto3);
    REGCUT = cat (1, coord_reg1, coord_reg2, coord_reg3);
    r4 = ginput(2);
    % Get the x and y corner coordinates as integers
    xmin4 = min(floor(r4(1)), floor(r4(2)));
    ymin4 = min(floor(r4(3)), floor(r4(4)));
    xmax4 = max(ceil(r4(1)), ceil(r4(2)));
    ymax4 = max(ceil(r4(3)), ceil(r4(4)));
    ancho4 = xmax4 - xmin4;
    alto4 = ymax4 - ymin4;
    coord_reg4=cat(2, xmin4, ymin4, ancho4, alto4);
    %Creamos la nueva imagen en base a las coordenadas
    corte1 = test(ymin1:ymax1, xmin1: xmax1,:);
    corte2 = test(ymin2:ymax2, xmin2: xmax2,:);
    corte3 = test(ymin3:ymax3, xmin3: xmax3,:);
    corte4 = test(ymin4:ymax4, xmin4: xmax4,:);
```

```
    REGCUT = cat (1, coord_reg1, coord_reg2, coord_reg3, coord_reg4);
    imwrite(corte1,'TEMP/Regiones_calib/1_reg/corte_1.png');
    imwrite(corte2,'TEMP/Regiones_calib/2_reg/corte_2.png');
    imwrite(corte3,'TEMP/Regiones_calib/3_reg/corte_3.png');
    imwrite(corte4,'TEMP/Regiones_calib/4_reg/corte_4.png');
    % Save coords and close figures
    save('coordenadas.mat','REGCUT');
    close;
end
```

c. Binarización.

```
function binarizacion()
close all
clc
%% leemos la imagen recortada
load('Regiones.mat');
if region == 1
    Lista = dir('TEMP/Regiones_calib/1_reg/*.png');
    for nImage=1:length(Lista)
        reg1 = imread(['TEMP/Regiones_calib/1_reg/' Lista(nImage).name]);
    end
    % si la imagen es RGB, se convierte a escala de grises
    if ndims(reg1) == 3
        reg1 = rgb2gray(reg1);
    end
    grayImage = imadjust(reg1);
    %load('fondo.mat');
    %if A == 'Claro'
     %   binaryRever = imbinarize(grayImage);
      %  binaryImage1 = not(binaryRever);
    %elseif A == 'Oscuro'
        binaryImage1 = imbinarize(grayImage);
    %end
    % tratamos la imagen
    cFactor=1;
    se = strel('disk',cFactor);
    cFactor2=1;
    se2 = strel('disk',cFactor2);
    closeImage = imclose(binaryImage1,se);
    %load(
    dilatedImage = imerode(closeImage,se2);
    imwrite(dilatedImage,'TEMP/Regiones_calib/bin1/imag_bin.png');
    % Etiquetado de las regiones hallados
    [regions, numObj] = bwlabel(dilatedImage);
    % Se obtiene el Bounding Box de las regiones conectadas
    bBox = regionprops(regions, 'BoundingBox');
    % Se llena la estructura de regiones encontradas
    for k = 1 : numObj
        %se obtiene la regi n de imagen segmentada
        segments1(k).image = regions( ...
            ceil(bBox(k).BoundingBox(1,2)) : floor(bBox(k).BoundingBox(1,2)) +
floor(bBox(k).BoundingBox(1,4)), ...
            ceil(bBox(k).BoundingBox(1,1)) : floor(bBox(k).BoundingBox(1,1)) +
floor(bBox(k).BoundingBox(1,3)));
        %se obtiene el centro
        segments1(k).center = ceil(size(segments1(k).image)/2);
        segments1(k).bBox = bBox(k).BoundingBox;
        %se obtiene el tama o
        segments1(k).size = size(segments1(k).image);
    end
    % se crea el folder en donde se guardar n los segmentos
    if exist('TEMP/Regiones_calib/segment_1/' , 'dir')
        rmdir('TEMP/Regiones_calib/segment_1/' , 's');
    end
    mkdir ('TEMP/Regiones_calib/segment_1/' );
    % 3. Guardar los segmentos
    numObj = length(segments1);
```

```
    for k = 1 : numObj
        imwrite(segments1(k).image,['TEMP/Regiones_calib/segment_1/' ...
                num2str(k,'%03d') '.png']);
    end
    % Delimitar las medidas de los segmentos
    imageList = dir('TEMP/Regiones_calib/segment_1/*.png');
    segment_size=[];
    for nImage=1:length(imageList)
        %se obtiene las imagenes de la carpeta
        currentImage = imread(['TEMP/Regiones_calib/segment_1/' ...
                       imageList(nImage).name]);
        [h,l] = size(currentImage);
        vector_size=cat(2, h, l);
        segment_size=cat(1, segment_size, vector_size);
    end
    save('segment_size.mat','segment_size');
    winopen('C:/Users/harold/Desktop/OCR-
THESIS/TEMP/Regiones_calib/segment_1');
    open('segment_size.mat');
elseif region == 2
    % Region 1
    Lista = dir('TEMP/Regiones_calib/1_reg/*.png');
    for nImage=1:length(Lista)
        reg1 = imread(['TEMP/Regiones_calib/1_reg/' Lista(nImage).name]);
    end
    % si la imagen es RGB, se convierte a escala de grises
    if ndims(reg1) == 3
        reg1 = rgb2gray(reg1);
    end
    grayImage = imadjust(reg1);
    binaryImage1 = imbinarize(grayImage);
    % tratamos la imagen
    cFactor=2;
    se = strel('disk',cFactor);
    cFactor2=3;
    se2 = strel('disk',cFactor2);
    closeImage = imclose(binaryImage1,se);
    dilatedImage = imerode(closeImage,se2);
    imwrite(dilatedImage,'TEMP/Regiones_calib/bin1/imag_bin.png');
    % Etiquetado de las regiones hallados
    [regions, numObj] = bwlabel(dilatedImage);
    % Se obtiene el Bounding Box de las regiones conectadas
    bBox = regionprops(regions, 'BoundingBox');
    % Se llena la estructura de regiones encontradas
    for k = 1 : numObj
        %se obtiene la regi n de imagen segmentada
        segments1(k).image = regions( ...
            ceil(bBox(k).BoundingBox(1,2)) : floor(bBox(k).BoundingBox(1,2)) +
floor(bBox(k).BoundingBox(1,4)), ...
            ceil(bBox(k).BoundingBox(1,1)) : floor(bBox(k).BoundingBox(1,1)) +
floor(bBox(k).BoundingBox(1,3)));
        %se obtiene el centro
        segments1(k).center = ceil(size(segments1(k).image)/2);
        segments1(k).bBox = bBox(k).BoundingBox;
        %se obtiene el tama o
        segments1(k).size = size(segments1(k).image);
    end
    % se crea el folder en donde se guardar n los segmentos
    if exist('TEMP/Regiones_calib/segment_1/' , 'dir')
```

```
        rmdir('TEMP/Regiones_calib/segment_1/' , 's');
    end
    mkdir ('TEMP/Regiones_calib/segment_1/' );
    % 3. Guardar los segmentos
    numObj = length(segments1);
    for k = 1 : numObj
        imwrite(segments1(k).image,['TEMP/Regiones_calib/segment_1/' ...
                num2str(k,'%03d') '.png']);
    end
    % Delimitar las medidas de los segmentos
    imageList = dir('TEMP/Regiones_calib/segment_1/*.png');

    segment_size=[];
    for nImage=1:length(imageList)
        %se obtiene las imagenes de la carpeta
        currentImage = imread(['TEMP/Regiones_calib/segment_1/' ...
                        imageList(nImage).name]);
        [h,l] = size(currentImage);
        vector_size=cat(2, h, l);
        segment_size=cat(1, segment_size, vector_size);
    end
    save('segment_size.mat','segment_size');

    %% Region 2
    Lista = dir('TEMP/Regiones_calib/2_reg/*.png');
    for nImage=1:length(Lista)
        reg2 = imread(['TEMP/Regiones_calib/2_reg/' Lista(nImage).name]);
    end
    % si la imagen es RGB, se convierte a escala de grises
    if ndims(reg2) == 3
        reg2 = rgb2gray(reg2);
    end
    grayImage = imadjust(reg2);
    binaryImage2 = imbinarize(grayImage);
    % tratamos la imagen
    cFactor=2;
    se = strel('disk',cFactor);
    cFactor2=3;
    se2 = strel('disk',cFactor2);
    closeImage2 = imclose(binaryImage2,se);
    dilatedImage2 = imerode(closeImage2,se2);
    imwrite(dilatedImage2,'TEMP/Regiones_calib/bin2/imag_bin.png');
    % Etiquetado de las regiones hallados
    [regions2, numObj] = bwlabel(dilatedImage2);
    % Se obtiene el Bounding Box de las regiones conectadas
    bBox = regionprops(regions2, 'BoundingBox');
    % Se llena la estructura de regiones encontradas
    for k = 1 : numObj
        %se obtiene la regi n de imagen segmentada
        segments1(k).image = regions2( ...
            ceil(bBox(k).BoundingBox(1,2)) : floor(bBox(k).BoundingBox(1,2)) +
floor(bBox(k).BoundingBox(1,4)), ...
            ceil(bBox(k).BoundingBox(1,1)) : floor(bBox(k).BoundingBox(1,1)) +
floor(bBox(k).BoundingBox(1,3)));
        %se obtiene el centro
        segments1(k).center = ceil(size(segments1(k).image)/2);
        segments1(k).bBox = bBox(k).BoundingBox;
        %se obtiene el tama o
        segments1(k).size = size(segments1(k).image);
```

```
    end
    % se crea el folder en donde se guardar n los segmentos
    if exist('TEMP/Regiones_calib/segment_2/' , 'dir')
        rmdir('TEMP/Regiones_calib/segment_2/' , 's');
    end
    mkdir ('TEMP/Regiones_calib/segment_2/' );
    % 3. Guardar los segmentos
    numObj = length(segments1);
    for k = 1 : numObj
        imwrite(segments1(k).image,['TEMP/Regiones_calib/segment_2/' ...
                num2str(k,'%03d') '.png']);
    end
    % Delimitar las medidas de los segmentos
    imageList2 = dir('TEMP/Regiones_calib/segment_2/*.png');
    segment_size2=[];
    for nImage=1:length(imageList2)
        %se obtiene las imagenes de la carpeta
        currentImage2 = imread(['TEMP/Regiones_calib/segment_2/' ...
                        imageList2(nImage).name]);
        [h2,l2] = size(currentImage2);
        vector_size=cat(2, h2, l2);
        segment_size2=cat(1, segment_size2, vector_size);
    end
    save('segment_size2.mat','segment_size2');
    winopen('C:/Users/harold/Desktop/OCR-
THESIS/TEMP/Regiones_calib/segment_1');
    winopen('C:/Users/harold/Desktop/OCR-
THESIS/TEMP/Regiones_calib/segment_2');
    open('segment_size.mat');
    open('segment_size2.mat');
end
```

d. Extracción de características.

```
function [features] = getFeatures(img, DEBUG)
     %% Valores por default
    if nargin<2,
        DEBUG=0;
    end;
    if nargin<1,
        img = im2bw(imread('trainingSet/clases/3/segment_331.png'));
    end;
    %casting
    img = double(img);
    %contador de caracteristicas
    cont = 1;
    [nFilas, nCols] = size(img);
    center = [ceil(nFilas/2), ceil(nCols/2)];

    shrink = bwmorph(img,'shrink',Inf);
    spur = bwmorph(shrink,'spur',Inf);
    skel = bwmorph (img,'skel',Inf);
    spur_2 = bwmorph(skel,'spur',4);

    %% razón nFilas/nCols
    features(cont) = nFilas/nCols;
    cont = cont + 1;

    %% razon de area letra - imagen
    bin = imbinarize (img);
    features(cont) = sum(sum(bin))/(nFilas*nCols);
    cont = cont + 1;

    %% centroide X, centroide Y
    [y, x] = find( spur_2 );
    centroid = [mean(x) mean(y)];
    features(cont) = centroid(1)/nCols;
    cont = cont + 1;
    features(cont) = centroid(2)/nFilas;
    cont = cont + 1;

    %% distancia del centro al centroide
    features(cont) = sqrt( ...
        ((center(1,2) - centroid(1))/nCols)^2 + ...
        ((center(1,1) - centroid(2))/nFilas)^2);
    cont = cont + 1;

    %%  numero de euler: e=#deObjetos - #deAgujerosEnLosObjetos
    feature_euler  = regionprops(spur, 'EulerNumber');
    features(cont) = feature_euler(1).EulerNumber;
    cont = cont + 1;

    %% procesar por bloque
    nBlocks = 3;
    sizeBlockFila = nFilas/nBlocks;
    sizeBlockCol = nCols/nBlocks;
    filasProcesadas = 0;
    for f = 1:sizeBlockFila+1:nFilas
        colsProcesadas = 0;
        for c = 1:sizeBlockCol+1:nCols
```

```
        finFila = round(f + sizeBlockFila -1);
        finCol = round(c + sizeBlockCol - 1);
        if finFila>nFilas
            finFila = nFilas;
        end
        if finCol>nCols
            finCol = nCols;
        end
        sizeFila = finFila - f;
        sizeCol = finCol - c;
        imgBlock = imcrop(bin,[round(c) round(f) round(sizeCol) ...
                    round(sizeFila)]);
        [m,n] = size(imgBlock);
        %atributo razon caracter - bloque
        features(cont) = sum(sum(imgBlock))/(m*n);
        cont = cont + 1;
        colsProcesadas = colsProcesadas +1;
        %figure; imshow(imgBlock);
    end
    %asegurar que las cols procesadas sean nBloques
    for i = colsProcesadas:(nBlocks-1)
        features(cont) = 0;
        cont = cont + 1;
    end
    filasProcesadas = filasProcesadas +1;
end
%asegurar que las cols procesadas sean nBloques
for i = filasProcesadas:(nBlocks-1)
    features(cont) = 0;
    cont = cont + 1;
end

%% numero de huecos
filled = imfill(img, 'holes');
holes = filled & ~img;
bigholes = bwareaopen(holes, 30);
stats = regionprops(bigholes, 'Area');
nHoles = length(find([stats.Area]>10));
features(cont) = nHoles;
cont = cont + 1;

%debug mode
if DEBUG==1
    figure; imshow(filled); title('All holes filled')
    figure; imshow(holes); title('Hole pixels identified')
    figure; imshow(bigholes); title('Only the big holes')
end

%% numero de end points
    endPoints = bwmorph(spur_2,'endpoints');
    branchpoints = bwmorph(spur_2,'branchpoints');
    se = strel('disk',round(nFilas/12));
    endPoints_dilated = imdilate(endPoints,se);
    branchpoints_dilated = imdilate(branchpoints,se);
    stats = regionprops(endPoints_dilated, 'Area');
    nEndPoints = length(find([stats.Area]>5));

    features(cont) = nEndPoints;
    cont = cont + 1;
```

```
    %debug mode
    if DEBUG==1
        figure; imshow(spur); title('spur')
        figure; imshow(endPoints); title('end points')
        figure; imshow(branchpoints); title('branch points')
        figure; imshow(endPoints_dilated); title('endPoints dilated')
        figure; imshow(branchpoints_dilated); title('branch points dilated')
    end
end
```

e. Extracción de características para imágenes de entrenamiento.

```
function getSegmentsFromTrainingSet(nTrainingSet, folderName)
%% debug mode
    DEBUG = 0;

%% 1. Adquisición de la imagen para entremamiento
    grayImageOriginal = imread(['trainingSet/' num2str(nTrainingSet) '.jpg']);
    % si la imagen es RGB, se convierte a escala de grises
    if ndims(grayImageOriginal) == 3
        grayImageOriginal = rgb2gray(grayImageOriginal);
    end
    grayImage = imadjust(grayImageOriginal);

    %debug mode
    if DEBUG==1
        %mostrar imagen
        figure; imshow(grayImage);
        title('Imagen en escala de grises - contrastada');
    end

%% 2. Obtener los segmentos de la imagen
    segments = getSegments(grayImage,0);

%% 3. Guardar los segmentos
    numObj = length(segments);
    for k = 1 : numObj
        j=k+550;
        imwrite(segments(k).image,['trainingSet/' folderName '/'
num2str(nTrainingSet) '/' num2str(nTrainingSet) '_' num2str(j,'%03d') '.png']);
    end
end
```

f. Extracción de características de imágenes nuevas.

```
function createDataSet()
clc
close all
    %%Se obtiene la lista de clases
    folderName ='arialSegmented';
    dirList = dir(['trainingSet/' folderName]);
    cont = 1; trainset = []; className = [];
    for ndir=1:length(dirList)
        if ~(strcmp(dirList(ndir).name, '.') || strcmp(dirList(ndir).name,
'..'))
            if dirList(ndir).isdir == 1,
                %% se obtiene la lista de imagenes de cada lista
                imageList = dir(['trainingSet/' folderName '/'
dirList(ndir).name '/*.png']);
                disp(['Procesando clase: ' dirList(ndir).name]);
                for nImage=1:length(imageList)
                    currentImage = imread(['trainingSet/' folderName '/'
dirList(ndir).name '/' imageList(nImage).name]);
                    % nombre de la clase
                    className(cont,1) = dirList(ndir).name;
                    ax=char(className);
                    % se obtienen los atributos de la imagen actual
                    trainset = cat(1, trainset, getFeatures(currentImage,0));
                    cont = cont + 1;
                end
            end
        end
    end
    %% save dataset
    save('trainset.mat','trainset');
    save('className.mat','ax');
```

g. Etiquetado de los parámetros de cada región (1 región) GUIde.

```
function varargout = region1(varargin)

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @region1_OpeningFcn, ...
                   'gui_OutputFcn',  @region1_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before region1 is made visible.
function region1_OpeningFcn(hObject, eventdata, handles, varargin)
axes(handles.Original)
Lista = dir('TEMP/Regiones_calib/1_reg/*.png');
for nImage=1:length(Lista)
    reg1 = imread(['TEMP/Regiones_calib/1_reg/' Lista(nImage).name]);
end
image(reg1);
axis off

axes(handles.Binaria)
ListaBin = dir('TEMP/Regiones_calib/bin1/*.png');
    for nImage=1:length(ListaBin)
    bin1 = imread(['TEMP/Regiones_calib/bin1/' ListaBin(nImage).name]);
    end
imshow(bin1);
axis off
% Choose default command line output for region1
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% --- Outputs from this function are returned to the command line.
function varargout = region1_OutputFcn(hObject, eventdata, handles)

% Get default command line output from handles structure
varargout{1} = handles.output;

function Nombre_Callback(hObject, eventdata, handles)

% --- Executes during object creation, after setting all properties.
function Nombre_CreateFcn(hObject, eventdata, handles)
```

```
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function Unidades_Callback(hObject, eventdata, handles)

% --- Executes during object creation, after setting all properties.
function Unidades_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in binarizar.
function binarizar_Callback(hObject, eventdata, handles)

% --- Executes on button press in volver.
function volver_Callback(hObject, eventdata, handles)
close
Inicio

% --- Executes on button press in OCR.
function OCR_Callback(hObject, eventdata, handles)

%% limpiamos valores iniciales
if exist('TEMP/Camera/OCR/' , 'dir')
    rmdir('TEMP/Camera/OCR/' , 's');
end
    mkdir ('TEMP/Camera/OCR/' );
if exist ('stado.mat')
    delete ('stado.mat');
end
if exist ('tiempo.mat')
    delete ('tiempo.mat');
end
if exist ('OCR_DATA.mat')
    delete ('OCR_DATA.mat');
end
if exist ('nombre.mat')
    delete ('nombre.mat');
end
if exist ('unidades.mat')
    delete ('unidades.mat');
end
%% Establecemos condiciones iniciales
stado=0;
nombre = get(handles.Nombre,'string');
unidades = get(handles.Unidades,'string');
save('nombre.mat','nombre');
save('unidades.mat','unidades');
save('stado.mat','stado');
close
vid=videoinput('winvideo',1);
vid.FramesPerTrigger = 1;
vid.TriggerRepeat = Inf;
triggerconfig(vid, 'Manual');
OCR_DATA=[];
```

```
tiempo=[];
for i=1:70
    load('stado.mat');
    if stado == 1
       break
    end
    capturedimage = getsnapshot(vid);
    imwrite(capturedimage,['TEMP/Camera/OCR/' num2str(i,'%03d') '.png']);
    OCR_1
    if exist ('ocr.mat')
        load('ocr.mat');
        OCR_DATA=cat(1,OCR_DATA,parametro);
        save('OCR_DATA.mat','OCR_DATA');
    end
    if exist ('t.mat')
        load('t.mat');
        tiempo=cat(1,tiempo,t);
        save('tiempo.mat','tiempo');
    end
    pause(2)
    if i == 70
        hor='Hora';
        minu='Minuto';
        seg='Segundo';
        load('nombre.mat');
        filename=([nombre '.xlsx']);
        load('OCR_DATA.mat');
        load('tiempo.mat');
        xlswrite(filename,nombre,1,'A1');
        load('unidades.mat');
        xlswrite(filename,unidades,1,'B1');
        xlswrite(filename,OCR_DATA,1,'A2');
        xlswrite(filename,hor,2,'A1');
        xlswrite(filename,minu,2,'B1');
        xlswrite(filename,seg,2,'C1');
        xlswrite(filename,tiempo,2,'A2');
        winopen(filename);
    end
end
```

h. Generación y organización de los parámetros (1 región) GUIde.

```
function varargout = OCRreg1(varargin)

gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @OCRreg1_OpeningFcn, ...
                   'gui_OutputFcn',  @OCRreg1_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before OCRreg1 is made visible.
function OCRreg1_OpeningFcn(hObject, eventdata, handles, varargin)

Lista = dir('TEMP/Camera/OCR/*.png');
for nImage=1:length(Lista)
    I = imread(['TEMP/Camera/OCR/' Lista(nImage).name]);
end
image(I);
axis off
load('nombre.mat');
disp(nombre)

    set(handles.titulo,'String',nombre);

load('unidades.mat');

    set(handles.UNIT,'String',unidades);

load('stado.mat');
if exist ('ocr.mat')
    load('ocr.mat');
    set(handles.OCR1,'String',parametro);
else
    set(handles.OCR1,'String','No hay parametro');
end
% Choose default command line output for OCRreg1
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% --- Outputs from this function are returned to the command line.
function varargout = OCRreg1_OutputFcn(hObject, eventdata, handles)

varargout{1} = handles.output;
```

```
% --- Executes on button press in Fin.
function Fin_Callback(hObject, eventdata, handles)

hor='Hora';
minu='Minuto';
seg='Segundo';
load('nombre.mat');
filename=([nombre '.xlsx']);
load('OCR_DATA.mat');
load('tiempo.mat');
%xlswrite(filename,A,sheet,xlRange)
xlswrite(filename,nombre,1,'A1');
load('unidades.mat');
xlswrite(filename,unidades,1,'B1');
xlswrite(filename,OCR_DATA,1,'A2');
xlswrite(filename,hor,2,'A1');
xlswrite(filename,minu,2,'B1');
xlswrite(filename,seg,2,'C1');
xlswrite(filename,tiempo,2,'A2');
winopen(filename);
stado=1;
save('stado.mat','stado');
```

i. OCR para 1 región.

```
function OCR_1 ()

%% Limpiamos valores iniciales
if exist('TEMP/Procesado_imag_OCR/seg1/' , 'dir')
        rmdir('TEMP/Procesado_imag_OCR/seg1/' , 's');
end
    mkdir ('TEMP/Procesado_imag_OCR/seg1/' );
if exist ('ocr.mat')
    delete ('ocr.mat');
end
if exist ('t.mat')
    delete ('t.mat');
end
%% Obtenemos imagen nueva
Lista = dir('TEMP/Camera/OCR/*.png');
for nImage=1:length(Lista)
    I = imread(['TEMP/Camera/OCR/' Lista(nImage).name]);
end
load('coordenadas.mat')
xmin = REGCUT(1,1);
ymin = REGCUT(1,2);
anchura = REGCUT(1,3);
altura = REGCUT(1,4);
corte=imcrop(I,[xmin ymin anchura altura]);
%% binarizamos
    if ndims(corte) == 3
        corte = rgb2gray(corte);
    end
    grayImage = imadjust(corte);
    load('fondo.mat');
    if A == 'Oscuro'
        binaryImage1 = imbinarize(grayImage);
    elseif A == 'Claros'
        binaryRever = imbinarize(grayImage);
        binaryImage1 = not(binaryRever);
    end
    %binaryImage1 = imbinarize(grayImage);
    % tratamos la imagen
    closeFactor=1;
    se = strel('disk',closeFactor);
    closeImage = imclose(binaryImage1,se);
    % Etiquetado de las regiones hallados
    [regions, numObj] = bwlabel(closeImage);
    % Se obtiene el Bounding Box de las regiones conectadas
    bBox = regionprops(regions, 'BoundingBox');
    % Se llena la estructura de regiones encontradas
    for k = 1 : numObj
        %se obtiene la regi n de imagen segmentada
        segments1(k).image = regions( ...
            ceil(bBox(k).BoundingBox(1,2)) : floor(bBox(k).BoundingBox(1,2)) +
floor(bBox(k).BoundingBox(1,4)), ...
            ceil(bBox(k).BoundingBox(1,1)) : floor(bBox(k).BoundingBox(1,1)) +
floor(bBox(k).BoundingBox(1,3)));
        %se obtiene el centro
        segments1(k).center = ceil(size(segments1(k).image)/2);
        segments1(k).bBox = bBox(k).BoundingBox;
```

```
        %se obtiene el tama o
        segments1(k).size = size(segments1(k).image);
    end
%% Segmentamos
    % Guardar los segmentos
    numObj = length(segments1);
    for k = 1 : numObj
        imwrite(segments1(k).image,['TEMP/Procesado_imag_OCR/seg1/' ...
                num2str(k,'%03d') '.png']);
    end
%% LIMPIAMOS
%Se obtiene la lista de clases
dirList = dir('TEMP/Procesado_imag_OCR/seg1/');
    % se obtiene la lista de imagenes de cada clase
    imageList = dir('TEMP/Procesado_imag_OCR/seg1/*.png');
    %se establecen umbrales
    load('segment_size.mat');
    load('segmento.mat');
    alto_min = (segment_size(segmento,1))-5;
    ancho_min = (segment_size(segmento,2))-10;
    alto_max = (segment_size(segmento,1))+5;
    ancho_max = (segment_size(segmento,2))+10;
    %se eliminan imagenes basura
    for nImage=1:length(imageList)
        %se obtiene la imagen actual
        currentImage = imread(['TEMP/Procesado_imag_OCR/seg1/'...
                               imageList(nImage).name]);
        [h1,l1] = size(currentImage);
        ancho_current = l1;
        altura_current = h1;
        % se elimina si est  afuera de los l mites
        if ((altura_current < alto_min) || (altura_current > alto_max))
           delete(['TEMP/Procesado_imag_OCR/seg1/' imageList(nImage).name]);
        elseif ((ancho_current < ancho_min) || (ancho_current > ancho_max))
           delete(['TEMP/Procesado_imag_OCR/seg1/' imageList(nImage).name]);
        end
    end
%% RECONOCEMOS LOS PARAMETROS
% CARGAMOS LA BASE DE ENTRENAMIENTO
load('trainset.mat');
load('className.mat');
% Aplicamos el algoritomo
model = fitcknn(trainset,ax);
model.NumNeighbors = 13;
% leemos los segmentos
Lista = dir('TEMP/Procesado_imag_OCR/seg1/*.png');
%filename = 'ocr.xlsx';
altura1 = [];
n = length(Lista);
numero = [];
for nImage=1:length(Lista)
    %se obtiene la imagen actual
    Imagen = imread(['TEMP/Procesado_imag_OCR/seg1/' Lista(nImage).name]);
    [h,l] = size(Imagen);
    altura1 = cat (1,altura1,h);
    label = predictionGivenImage(model,Imagen);
    dec = str2double(label)*(10^(n-1));
    numero = cat (1,numero,dec);
    n = n-1;
```

```
    %disp(['caracter reconocido: ' label]);
end
c1=clock;
c=fix(c1);
hor=c(1,4);
minu=c(1,5);
seg=c(1,6);
t=cat(2,hor,minu,seg);
limite = std(altura1);
if limite <= 2
    parametro = sum(numero);
    %disp(parametro);
    save('ocr.mat','parametro');
    save('t.mat','t');
    OCRreg1;
else
    disp('los caracteres son muy diferentes');
    OCRreg1;
end

if exist('TEMP/Camera/OCR/' , 'dir')
        rmdir('TEMP/Camera/OCR/' , 's');
end
    mkdir ('TEMP/Camera/OCR/' );
end

function predictionGivenClass(classExpected, model)
    %% Predicci n de una Instancia
    imgTest = imread(['TEMP/Procesado_imag_OCR/seg1/' num2str(classExpected)
'.png']);
    label = predictionGivenImage(model,imgTest);
    disp(['expected:' num2str(classExpected) ' - predicted:' num2str(label)]);
end

function [label] = predictionGivenImage(model,imgTest)
    instance4test = getFeatures(imgTest);
    label = predict(model,instance4test);
    end
```

j. Etiquetado de los parámetros de cada región (2 regiones) GUIde.

```
function varargout = region2(varargin)

gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @region2_OpeningFcn, ...
                   'gui_OutputFcn',  @region2_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% --- Executes just before region2 is made visible.
function region2_OpeningFcn(hObject, eventdata, handles, varargin)

% REGION 1
axes(handles.rec1)
Lista = dir('TEMP/Regiones_calib/1_reg/*.png');
for nImage=1:length(Lista)
    reg1 = imread(['TEMP/Regiones_calib/1_reg/' Lista(nImage).name]);
end
image(reg1);
axis off

axes(handles.bin1)
ListaBin = dir('TEMP/Regiones_calib/bin1/*.png');
    for nImage=1:length(ListaBin)
    bin1 = imread(['TEMP/Regiones_calib/bin1/' ListaBin(nImage).name]);
    end
imshow(bin1);
axis off
% REGION 2
axes(handles.rec2)
Lista = dir('TEMP/Regiones_calib/2_reg/*.png');
for nImage=1:length(Lista)
    reg2 = imread(['TEMP/Regiones_calib/2_reg/' Lista(nImage).name]);
end
image(reg2);
axis off

axes(handles.bin2)
ListaBin = dir('TEMP/Regiones_calib/bin2/*.png');
    for nImage=1:length(ListaBin)
    bin2 = imread(['TEMP/Regiones_calib/bin2/' ListaBin(nImage).name]);
    end
imshow(bin2);
axis off
% Choose default command line output for region2
handles.output = hObject;
```

```
% Update handles structure
guidata(hObject, handles);

% --- Outputs from this function are returned to the command line.
function varargout = region2_OutputFcn(hObject, eventdata, handles)

% Get default command line output from handles structure
varargout{1} = handles.output;

function nombre1_Callback(hObject, eventdata, handles)

% --- Executes during object creation, after setting all properties.
function nombre1_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function unit1_Callback(hObject, eventdata, handles)

% --- Executes during object creation, after setting all properties.
function unit1_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function nombre2_Callback(hObject, eventdata, handles)

function nombre2_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in Volver.
function Volver_Callback(hObject, eventdata, handles)

close
Inicio

% --- Executes on button press in OCR.
function OCR_Callback(hObject, eventdata, handles)

if exist('TEMP/Camera/OCR/' , 'dir')
        rmdir('TEMP/Camera/OCR/' , 's');
end
    mkdir ('TEMP/Camera/OCR/' );
if exist ('stado.mat')
    delete ('stado.mat');
end
if exist ('tiempo.mat')
    delete ('tiempo.mat');
end
if exist ('OCR_DATA1.mat')
    delete ('OCR_DATA1.mat');
```

```
end
if exist ('OCR_DATA2.mat')
    delete ('OCR_DATA2.mat');
end
stado=0;
nombre1 = get(handles.nombre1,'string');
unidad1 = get(handles.unit1,'string');
nombre2 = get(handles.nombre2,'string');
unidad2 = get(handles.unit2,'string');
save('nombre1.mat','nombre1');
save('unidades1.mat','unidad1');
save('nombre2.mat','nombre2');
save('unidades2.mat','unidad2');
save('stado.mat','stado');
close
vid=videoinput('winvideo',1);
vid.FramesPerTrigger = 1;
vid.TriggerRepeat = Inf;
triggerconfig(vid, 'Manual');
OCR_DATA1=[];
OCR_DATA2=[];
tiempo=[];
for i=1:70
    load('stado.mat');
    if stado == 1
       break
    end
    capturedimage = getsnapshot(vid);
    imwrite(capturedimage,['TEMP/Camera/OCR/' num2str(i,'%03d') '.png']);
    OCR_2
    if exist ('t.mat')
        load('t.mat');
        tiempo=cat(1,tiempo,t);
        save('tiempo.mat','tiempo');
        if exist ('ocr1.mat')
            load('ocr1.mat');
            OCR_DATA1=cat(1,OCR_DATA1,parametro1);
            save('OCR_DATA1.mat','OCR_DATA1');
        else
            parametro1=0;
            OCR_DATA1=cat(1,OCR_DATA1,parametro1);
            save('OCR_DATA1.mat','OCR_DATA1');
        end
        if exist ('ocr2.mat')
            load('ocr2.mat');
            OCR_DATA2=cat(1,OCR_DATA2,parametro2);
            save('OCR_DATA2.mat','OCR_DATA2');
        else
            parametro2=0;
            OCR_DATA2=cat(1,OCR_DATA2,parametro2);
            save('OCR_DATA2.mat','OCR_DATA2');
        end
    end
    pause(2)
end

function unit2_Callback(hObject, eventdata, handles)

% --- Executes during object creation, after setting all properties.
```

```
function unit2_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

k. Generación y organización de los parámetros (2 regiones) GUIde.

```
function varargout = OCRreg2(varargin)
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @OCRreg2_OpeningFcn, ...
                   'gui_OutputFcn',  @OCRreg2_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end

% --- Executes just before OCRreg2 is made visible.
function OCRreg2_OpeningFcn(hObject, eventdata, handles, varargin)
Lista = dir('TEMP/Camera/OCR/*.png');
for nImage=1:length(Lista)
    I = imread(['TEMP/Camera/OCR/' Lista(nImage).name]);
end
image(I);
axis off
load('nombre1.mat');
load('unidades1.mat');
load('nombre2.mat');
load('unidades2.mat');
load('stado.mat');
set(handles.name1,'String',nombre1);
set(handles.unit1,'String',unidad1);
set(handles.name2,'String',nombre2);
set(handles.unit2,'String',unidad2);
if exist ('ocr1.mat')
    load('ocr1.mat');
    set(handles.ocr1,'String',parametro1);
else
    set(handles.ocr1,'String','Invalid Data');
end
if exist ('ocr2.mat')
    load('ocr2.mat');
    set(handles.ocr2,'String',parametro2);
else
    set(handles.ocr2,'String','Invalid Data');
end
% Choose default command line output for OCRreg2
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% --- Outputs from this function are returned to the command line.
function varargout = OCRreg2_OutputFcn(hObject, eventdata, handles)
```

```
% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes on button press in fin.
function fin_Callback(hObject, eventdata, handles)
load('nombre1.mat');
load('nombre2.mat');
titulo=strcat(nombre1,' and ',nombre2);
filename=([titulo '.xlsx']);
load('OCR_DATA1.mat');
load('OCR_DATA2.mat');
load('unidades1.mat');
load('tiempo.mat');
xlswrite(filename,OCR_DATA1,1);
xlswrite(filename,OCR_DATA2,2);
xlswrite(filename,tiempo,3);
winopen(filename);
stado=1;
save('stado.mat','stado');
```

l. OCR para 2 regiones.

```
function OCR_2 ()

%% Limpiamos valores iniciales
if exist('TEMP/Procesado_imag_OCR/seg1/' , 'dir')
        rmdir('TEMP/Procesado_imag_OCR/seg1/' , 's');
end
    mkdir ('TEMP/Procesado_imag_OCR/seg1/' );
if exist('TEMP/Procesado_imag_OCR/seg2/' , 'dir')
        rmdir('TEMP/Procesado_imag_OCR/seg2/' , 's');
end
    mkdir ('TEMP/Procesado_imag_OCR/seg2/' );
if exist ('ocr1.mat')
    delete ('ocr1.mat');
end
if exist ('ocr2.mat')
    delete ('ocr2.mat');
end
if exist ('t.mat')
    delete ('t.mat');
end
%% Obtenemos imagen nueva
Lista = dir('TEMP/Camera/OCR/*.png');
for nImage=1:length(Lista)
    I = imread(['TEMP/Camera/OCR/' Lista(nImage).name]);
end
%I = imread('TEMP/Camera/OCR/48_024.png'); %modificar
%% Recortamos con los valores de "coordenadas" REGIÓN 1
load('coordenadas.mat')
x1 = REGCUT(1,1);
y1 = REGCUT(1,2);
anchura1 = REGCUT(1,3);
altura1 = REGCUT(1,4);
corte1=imcrop(I,[x1 y1 anchura1 altura1]);
%% binarizamos
    if ndims(corte1) == 3
        corte1 = rgb2gray(corte1);
    end
    grayImage = imadjust(corte1);
    binaryImage1 = imbinarize(grayImage);
    % tratamos la imagen
    closeFactor=1;
    se = strel('disk',closeFactor);
    closeImage = imclose(binaryImage1,se);
    % Etiquetado de las regiones hallados
    [regions, numObj] = bwlabel(closeImage);
    % Se obtiene el Bounding Box de las regiones conectadas
    bBox = regionprops(regions, 'BoundingBox');
    % Se llena la estructura de regiones encontradas
    for k = 1 : numObj
        %se obtiene la región de imagen segmentada
        segments1(k).image = regions( ...
            ceil(bBox(k).BoundingBox(1,2)) : floor(bBox(k).BoundingBox(1,2)) +
floor(bBox(k).BoundingBox(1,4)), ...
            ceil(bBox(k).BoundingBox(1,1)) : floor(bBox(k).BoundingBox(1,1)) +
floor(bBox(k).BoundingBox(1,3)));
        %se obtiene el centro
```

```
        segments1(k).center = ceil(size(segments1(k).image)/2);
        segments1(k).bBox = bBox(k).BoundingBox;
        %se obtiene el tamaño
        segments1(k).size = size(segments1(k).image);
    end
%% Segmentamos
    % Guardar los segmentos
    numObj = length(segments1);
    for k = 1 : numObj
        imwrite(segments1(k).image,['TEMP/Procesado_imag_OCR/seg1/' ...
                num2str(k,'%03d') '.png']);
    end
%% LIMPIAMOS
%Se obtiene la lista de clases
dirList = dir('TEMP/Procesado_imag_OCR/seg1/');
    % se obtiene la lista de imagenes de cada clase
    imageList = dir('TEMP/Procesado_imag_OCR/seg1/*.png');
    %se establecen umbrales
    alto_min = 49;
    ancho_min = 19;
    alto_max = 79;
    ancho_max = 45;
    %se eliminan imagenes basura
    for nImage=1:length(imageList)
        %se obtiene la imagen actual
        currentImage = imread(['TEMP/Procesado_imag_OCR/seg1/'...
                               imageList(nImage).name]);
        [h1,l1] = size(currentImage);
        ancho_current = l1;
        altura_current = h1;
        % se elimina si está afuera de los límites
        if ((altura_current < alto_min) || (altura_current > alto_max))
           delete(['TEMP/Procesado_imag_OCR/seg1/' imageList(nImage).name]);
        elseif ((ancho_current < ancho_min) || (ancho_current > ancho_max))
           delete(['TEMP/Procesado_imag_OCR/seg1/' imageList(nImage).name]);
        end
    end
%% RECONOCEMOS LOS PARAMETROS
% CARGAMOS LA BASE DE ENTRENAMIENTO
load('trainset.mat');
load('className.mat');
% Aplicamos el algoritomo
model = fitcknn(trainset,ax);
model.NumNeighbors = 13;
% leemos los segmentos
Lista = dir('TEMP/Procesado_imag_OCR/seg1/*.png');
%filename = 'ocr.xlsx';
altura1 = [];
n = length(Lista);
numero = [];
for nImage=1:length(Lista)
    %se obtiene la imagen actual
    Imagen = imread(['TEMP/Procesado_imag_OCR/seg1/' Lista(nImage).name]);
    [h,l] = size(Imagen);
    altura1 = cat (1,altura1,h);
    label = predictionGivenImage(model,Imagen);
    dec = str2double(label)*(10^(n-1));
    numero = cat (1,numero,dec);
    n = n-1;
```

```
    %disp(['caracter reconocido: ' label]);
end

    %% Recortamos con los valores de "coordenadas" REGION 2
load('coordenadas.mat')
x2 = REGCUT(2,1);
y2 = REGCUT(2,2);
anchura2 = REGCUT(2,3);
altura2 = REGCUT(2,4);
corte2=imcrop(I,[x2 y2 anchura2 altura2]);
%% binarizamos
    if ndims(corte2) == 3
        corte2 = rgb2gray(corte2);
    end
    grayImage2 = imadjust(corte2);
    binaryImage2 = imbinarize(grayImage2);
    % tratamos la imagen
    closeFactor=1;
    se = strel('disk',closeFactor);
    closeImage2 = imclose(binaryImage2,se);
    % Etiquetado de las regiones hallados
    [regions, numObj] = bwlabel(closeImage2);
    % Se obtiene el Bounding Box de las regiones conectadas
    bBox = regionprops(regions, 'BoundingBox');
    % Se llena la estructura de regiones encontradas
    for k = 1 : numObj
        %se obtiene la región de imagen segmentada
        segments1(k).image = regions( ...
            ceil(bBox(k).BoundingBox(1,2)) : floor(bBox(k).BoundingBox(1,2)) +
floor(bBox(k).BoundingBox(1,4)), ...
            ceil(bBox(k).BoundingBox(1,1)) : floor(bBox(k).BoundingBox(1,1)) +
floor(bBox(k).BoundingBox(1,3)));
        %se obtiene el centro
        segments1(k).center = ceil(size(segments1(k).image)/2);
        segments1(k).bBox = bBox(k).BoundingBox;
        %se obtiene el tamaño
        segments1(k).size = size(segments1(k).image);
    end
%% Segmentamos
    % Guardar los segmentos
    numObj = length(segments1);
    for k = 1 : numObj
        imwrite(segments1(k).image,['TEMP/Procesado_imag_OCR/seg2/' ...
                num2str(k,'%03d') '.png']);
    end
%% LIMPIAMOS
%Se obtiene la lista de clases
dirList = dir('TEMP/Procesado_imag_OCR/seg2/');
    % se obtiene la lista de imagenes de cada clase
    imageList = dir('TEMP/Procesado_imag_OCR/seg2/*.png');
    %se establecen umbrales
    alto_min = 53;
    ancho_min = 18;
    alto_max = 76;
    ancho_max = 45;
    %se eliminan imagenes basura
    for nImage=1:length(imageList)
        %se obtiene la imagen actual
        currentImage = imread(['TEMP/Procesado_imag_OCR/seg2/'...
```

```
                                imageList(nImage).name]);
        [h1,l1] = size(currentImage);
        ancho_current = l1;
        altura_current = h1;
        % se elimina si está afuera de los límites
        if ((altura_current < alto_min) || (altura_current > alto_max))
           delete(['TEMP/Procesado_imag_OCR/seg2/' imageList(nImage).name]);
        elseif ((ancho_current < ancho_min) || (ancho_current > ancho_max))
           delete(['TEMP/Procesado_imag_OCR/seg2/' imageList(nImage).name]);
        end
    end
%% RECONOCEMOS LOS PARAMETROS
% CARGAMOS LA BASE DE ENTRENAMIENTO
load('trainset.mat');
load('className.mat');
% Aplicamos el algoritomo
model = fitcknn(trainset,ax);
model.NumNeighbors = 13;
% leemos los segmentos
Lista = dir('TEMP/Procesado_imag_OCR/seg2/*.png');
%filename = 'ocr.xlsx';
altura2 = [];
n = length(Lista);
numero2 = [];
for nImage=1:length(Lista)
    %se obtiene la imagen actual
    Imagen = imread(['TEMP/Procesado_imag_OCR/seg2/' Lista(nImage).name]);
    [h2,l] = size(Imagen);
    altura2 = cat (1,altura2,h2);
    label = predictionGivenImage(model,Imagen);
    dec2 = str2double(label)*(10^(n-1));
    numero2 = cat (1,numero2,dec2);
    n = n-1;
    %disp(['caracter reconocido: ' label]);
end
%% Iniciamos la toma del tiempo
c1=clock;
c=fix(c1);
hor=c(1,4);
minu=c(1,5);
seg=c(1,6);
t=cat(2,hor,minu,seg);
%% verificamos que sea un dato correcto
limite = std(altura1);
if limite <= 2
    parametro1 = sum(numero);
    %disp(parametro);
    save('ocr1.mat','parametro1');
    %OCRreg2;
else
    disp('los caracteres R1 son muy diferentes');
end
%% Verificamos que se este captando un valor correcto
limite2 = std(altura2);
if limite2 <= 2
    parametro2 = sum(numero2);
    %disp(parametro);
    save('ocr2.mat','parametro2');
else
```

```
    disp('los caracteres R2 son muy diferentes');
end
if (limite <= 2 || limite2 <= 2)
    save('t.mat','t');
    OCRreg2;
else
    OCRreg2;
end
if exist('TEMP/Camera/OCR/' , 'dir')
        rmdir('TEMP/Camera/OCR/' , 's');
end
    mkdir ('TEMP/Camera/OCR/' );
end


function predictionGivenClass(classExpected, model)
    %% Predicción de una Instancia
    imgTest = imread(['TEMP/Procesado_imag_OCR/seg1/' num2str(classExpected)
'.png']);
    label = predictionGivenImage(model,imgTest);
    disp(['expected:' num2str(classExpected) ' - predicted:' num2str(label)]);
end

function [label] = predictionGivenImage(model,imgTest)
    instance4test = getFeatures(imgTest);
    label = predict(model,instance4test);
    end
```

m. Calibración de regiones.

```
function varargout = calib_reg1(varargin)

gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @calib_reg1_OpeningFcn, ...
                   'gui_OutputFcn',  @calib_reg1_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end

% --- Executes just before calib_reg1 is made visible.
function calib_reg1_OpeningFcn(hObject, eventdata, handles, varargin)

axes(handles.axes1) % obtenemos el objeto en que mostraremos el recorte
% leemos la ubicaion de la imagen
Lista = dir('TEMP/Regiones_calib/1_reg/*.png');
for nImage=1:length(Lista)
    reg1 = imread(['TEMP/Regiones_calib/1_reg/' Lista(nImage).name]);
end
image(reg1);
axis off
nIter=1;
save('nIter.mat','nIter');
% Choose default command line output for calib_reg1
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% --- Outputs from this function are returned to the command line.
function varargout = calib_reg1_OutputFcn(hObject, eventdata, handles)

% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes when selected object is changed in uibuttongroup1.
function uibuttongroup1_SelectionChangedFcn(hObject, eventdata, handles)

A=get(hObject,'String'); % obtener el valor de fondo seleccionado
save('fondo.mat','A'); % guardamos este valor para el bucle
switch A
    case 'Claros' % si el valor seleccionado es "claros"
        %% leemos la imagen recortada
        load('Regiones.mat');
        if region == 1
            Lista = dir('TEMP/Regiones_calib/1_reg/*.png');
            for nImage=1:length(Lista)
```

```
                reg1 = imread(['TEMP/Regiones_calib/1_reg/'
Lista(nImage).name]);
            end
            % si la imagen es RGB, se convierte a escala de grises
            if ndims(reg1) == 3
                reg1 = rgb2gray(reg1);
            end
            grayImage = imadjust(reg1); % conversion a escala de grises
            notbin1 = imbinarize(grayImage); % binarizacion
            binaryImage1 = not(notbin1); % inversion de la binarizacion
            cFactor=1; % factor de dilatacion y erosion
            se = strel('disk',cFactor); % operacion morfologica
            cFactor2=1; % factor de erosion
            se2 = strel('disk',cFactor2);
            closeImage = imclose(binaryImage1,se); % dilatacion seguida de una
erosion con el mismo factor
            load('nIter.mat'); % leemos el valor selecionado de iteraciones
            for n = 1:nIter
                 dilatedImage = imerode(closeImage,se2); % erosion de la
imagen
            end
            axes(handles.axes2); % seleccion del objeto en el que se
visualizara el resultado
            imshow(dilatedImage); % mostramos la imagen binaria resultante
            axis off %quitamos la cuadricula
            imwrite(dilatedImage,'TEMP/Regiones_calib/bin1/imag_bin.png'); %
guardamos esta imagen para ser usada posteriormente
        end
    case 'Oscuro' % para el caso "oscuros se realiza lo mismo
        %% leemos la imagen recortada
        load('Regiones.mat');
        if region == 1
            Lista = dir('TEMP/Regiones_calib/1_reg/*.png');
            for nImage=1:length(Lista)
                reg1 = imread(['TEMP/Regiones_calib/1_reg/'
Lista(nImage).name]);
            end
            % si la imagen es RGB, se convierte a escala de grises
            if ndims(reg1) == 3
                reg1 = rgb2gray(reg1);
            end
            grayImage = imadjust(reg1);
            % solo que aca no se niega la binarizacion puesto que el fondo
            % ya es oscuro
            binaryImage1 = imbinarize(grayImage);
            cFactor=1;
            se = strel('disk',cFactor);
            cFactor2=1;
            se2 = strel('disk',cFactor2);
            closeImage = imclose(binaryImage1,se);
            load('nIter.mat');
            for n = 1:nIter
                 dilatedImage = imerode(closeImage,se2);
            end
            axes(handles.axes2);
            imshow(dilatedImage);
            axis off
            imwrite(dilatedImage,'TEMP/Regiones_calib/bin1/imag_bin.png');
        end
```

```
end


% --- Executes on button press in cambios.
function cambios_Callback(hObject, eventdata, handles)

value=get(handles.popupmenu2,'Value');
medida=get(handles.popupmenu2,'String');
if strcmp(medida{value},'001')
    segmento=1;
elseif strcmp(medida{value},'002')
    segmento=2;
elseif strcmp(medida{value},'003')
    segmento=3;
elseif strcmp(medida{value},'004')
    segmento=4;
elseif strcmp(medida{value},'005')
    segmento=5;
elseif strcmp(medida{value},'006')
    segmento=6;
elseif strcmp(medida{value},'007')
    segmento=7;
elseif strcmp(medida{value},'008')
    segmento=8;
elseif strcmp(medida{value},'009')
    segmento=9;
elseif strcmp(medida{value},'010')
    segmento=10;
elseif strcmp(medida{value},'011')
    segmento=11;
end
save('segmento.mat','segmento');
close;
region1;


% --- Executes on selection change in popupmenu2.
function popupmenu2_Callback(hObject, eventdata, handles)

% --- Executes during object creation, after setting all properties.
function popupmenu2_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in segmentar.
function segmentar_Callback(hObject, eventdata, handles)

    %leemos la imagen binaria
    dilatedImage = imread('TEMP/Regiones_calib/bin1/imag_bin.png');
    % Etiquetado de las regiones hallados
    [regions, numObj] = bwlabel(dilatedImage);
    % Se obtiene el Bounding Box de las regiones conectadas
    bBox = regionprops(regions, 'BoundingBox');
    % Se llena la estructura de regiones encontradas
    for k = 1 : numObj
        %se obtiene la regi n de imagen segmentada
```

```
        segments1(k).image = regions( ...
            ceil(bBox(k).BoundingBox(1,2)) : floor(bBox(k).BoundingBox(1,2)) +
floor(bBox(k).BoundingBox(1,4)), ...
            ceil(bBox(k).BoundingBox(1,1)) : floor(bBox(k).BoundingBox(1,1)) +
floor(bBox(k).BoundingBox(1,3)));
        %se obtiene el centro
        segments1(k).center = ceil(size(segments1(k).image)/2);
        segments1(k).bBox = bBox(k).BoundingBox;
        %se obtiene el tama o
        segments1(k).size = size(segments1(k).image);
    end
    % se crea el folder en donde se guardar n los segmentos
    if exist('TEMP/Regiones_calib/segment_1/' , 'dir')
        rmdir('TEMP/Regiones_calib/segment_1/' , 's');
    end
    mkdir ('TEMP/Regiones_calib/segment_1/' );
    % 3. Guardar los segmentos
    numObj = length(segments1);
    for k = 1 : numObj
        imwrite(segments1(k).image,['TEMP/Regiones_calib/segment_1/' ...
                num2str(k,'%03d') '.png']);
    end
    % Guardamos el tama o de los segmentos encontrados
    imageList = dir('TEMP/Regiones_calib/segment_1/*.png');
    segment_size=[];
    for nImage=1:length(imageList)
        %se obtiene las imagenes de la carpeta
        currentImage = imread(['TEMP/Regiones_calib/segment_1/' ...
                       imageList(nImage).name]);
        [h,l] = size(currentImage);
        vector_size=cat(2, h, l);
        % segment_zise es el archivo en que se guarda los tama os
        segment_size=cat(1, segment_size, vector_size);
    end
    save('segment_size.mat','segment_size');
    %se abre la carpeta contenedora para elegir un caracter
    winopen('C:/Users/harold/Desktop/OCR-
THESIS/TEMP/Regiones_calib/segment_1');

% --- Executes on button press in radiobutton4.
function radiobutton4_Callback(hObject, eventdata, handles)

% --- Executes on button press in radiobutton3.
function radiobutton3_Callback(hObject, eventdata, handles)

function uibuttongroup2_SelectionChangedFcn(hObject, eventdata, handles)

B=get(hObject,'String');
nIter=str2double(B);
switch B
    case '1'
        save('nIter.mat','nIter')
    case '2'
        save('nIter.mat','nIter')
    case '3'
        save('nIter.mat','nIter')
    case '4'
        save('nIter.mat','nIter')
end
```

n. Comprobación de estado.

```
%% iniciamos bucle
stado=0;
t = 1;
OCR_1
OCRreg1
while stado == 0
    load('stado.mat');

    disp(t);
    disp('programa en funcionamiento');
    t=t+1;
end
```