UNIVERSIDAD POLITÉCNICA DE CARTAGENA

MÁSTER UNIVERSITARIO EN INGENIERÍA DE TELECOMUNICACIÓN

Trabajo Fin de Máster

DESARROLLO E IMPLEMENTACIÓN DE UN MÉTODO BASADO EN PROCESADO DE IMAGEN PARA LA SEGMENTACIÓN E IDENTIFICACIÓN DE DESPERFECTOS EN BOTELLAS EN UNA LÍNEA DE PRODUCCIÓN

Francisco Javier Martínez García

21 de febrero de 2023

Director: ***Dr. D. Rafael Verdú Monedero***,
Codirector: ***Dr. D. Rubén Martínez Sandoval***

Agradecimientos

Este Trabajo Fin de Máster se ha realizado en el marco del PROGRAMA ENCUENTRO TF, según el artículo 6 del Decreto 381/2023, de 16 de noviembre, (BORM 269 de 30 de noviembre de 2023) para actuaciones de consolidación e internacionalización del Campus de Excelencia Internacional "Mare Nostrum 37/38", para el curso 2023/2024.

CAMPUS DE EXCELENCIA INTERNACIONAL

CAMPUS MARE NOSTRUM

Universidad Politécnica de Cartagena

MIEMBRO DE EUT+ EUROPEAN UNIVERSITY OF TECHNOLOGY

Índice

CAPÍTULO 1

Introducción

En el contexto de la producción industrial, donde la eficiencia y la calidad son elementos cruciales, la aplicación de tecnologías avanzadas desempeña un papel fundamental. Este Trabajo de Fin de Máster trata de dar una solución específica dentro del ámbito de la Visión Artificial, centrándose en el control de calidad del etiquetado de las botellas de Licor 43 [1], un producto manufacturado por el Grupo Zamora [2]. Este proyecto se enmarca en la colaboración con Biyectiva Technology [3], una empresa especializada en soluciones de visión con sede en Cartagena, Murcia.

En el presente trabajo se pretende detectar defectos en el etiquetado de botellas. Para ello, se propone una metodología basada en el registro de imágenes y técnicas de morfología matemática. La idea fundamental es alinear las imágenes de producción con imágenes de referencia de productos sin desperfectos, permitiendo así la comparación y evaluación de posibles defectos en tiempo real, permitiendo el rechazo de botellas en las que se detecten estos desperfectos en el etiquetado.

La estructura de este Trabajo de Fin de Máster se organiza en capítulos para abordar de manera sistemática los aspectos fundamentales del problema y la solución propuesta. Inicialmente, se establece el contexto y la problemática específica que motiva la investigación, lo cual nos permite saber de qué partimos y a dónde vamos. A continuación, se explora el marco teórico que respalda la metodología propuesta, dividiéndolo en las dos secciones principales que intervienen

en este proyecto: registro de imagen y segmentación mediante morfología matemática. Esto da una visión general de los conocimientos necesarios para entender el proyecto.

El desarrollo de la solución se presenta detalladamente en el Capítulo 3, donde se describen las diferentes etapas del proceso, como la segmentación de la etiqueta y el registro rígido y deformable. Los resultados obtenidos, incluyendo la elección de parámetros óptimos y el análisis de las diferentes etapas, se exponen en el Capítulo 4. Finalmente, el trabajo concluye con las lecciones aprendidas y las perspectivas futuras en el Capítulo 5.

CAPÍTULO 2

Contexto y marco teórico

Este Trabajo Fin de Máster, el cual se enmarca en el Programa Encuentro TF [4], trata de dar solución a una parte de un proyecto a cargo de Biyectiva Technology, una empresa dedicada al campo de Visión Artificial, situada en Cartagena, Murcia. Este proyecto trata de utilizar un sistema 360, para el control de calidad del etiquetado de las botellas de Licor 43, producto de la empresa Grupo Zamora, en la línea de producción. Esto incluye la detección de la posición o presencia de las etiqueta, o lo que concierne a este trabajo, que es la detección de desperfectos, como rasgaduras producidas por golpes y roces entre las botellas. Debido a la geometría de las botellas, la cual es de tronco de cono invertido, estos golpeos afectan sobre todo a la etiqueta superior de la cara frontal de la botella. En la Figura 2.1, podemos ver la cara frontal de la botella, siendo la etiqueta redonda, es decir, la que contiene los dígitos *43* y letras *LICOR CUARENTA Y TRES*, nuestra etiqueta de interés.

Un sistema 360 consta de cuatro cámaras y una fotocélula, la cual da la señal de disparo a las cámaras cada vez que pasa una botella, además de un expulsor que permitirá descartar las botellas defectuosas de la línea de producción. Cada una de las cuatro cámaras toma una imagen y el sistema se encarga de encontrar en cuál de las cuatro imágenes se encuentra la etiqueta a analizar. Una vez la encuentra, la empareja con una de las imágenes de una base de datos de la que dispone el sistema. Esta base de datos se ha creado tomando capturas a un producto, el cual se considera sin desperfectos, haciéndolo girar en pasos de un grado, sobre el eje de la posición de disparo de la fotocélula. Al hacer el emparejamiento, obtenemos las dos imágenes que necesitamos

Figura 2.1: Recorte frontal de una botella de Licor 43.

utilizar. La primera imagen es la captada por las cámaras en el momento de la producción, la cual presenta el producto candidato a ser descartado, la cual llamaremos *query*; la segunda es la imagen de la base de datos, en la cual el producto presenta el mismo ángulo de rotación que el producto que se encuentra en la *query* y, por tanto, al compararlas se podrá comprobar si el producto que ha pasado presenta desperfectos en las etiquetas. La imagen de la base de datos que se obtiene del emparejamiento con la query, la se denominará *train*. Estos nombres son muy comúnmente utilizados en problemas de *Machine Learning* o Aprendizaje Máquina, siendo *train* el elemento entrenado y *query*, el elemento consulta [5]. Una vez obtenida tanto la imagen *query* como su correspondiente imagen *train*, es en este punto donde comienza el desarrollo de la solución del software de este trabajo.

La idea base de este proyecto es registrar la imagen *query* con la imagen *train* para alinear las etiquetas de cada imagen y después hacer una comparación, comprobando si la imagen obtenida en producción (*query*) es similar a la imagen obtenida fuera de producción con una botella sin desperfectos (*train*). El registro se hará en dos fases. En la primera, al considerarse un desplazamiento afín, se realizará un registro afín. En esta primera fase se utilizará la morfología matemática para preparar las imágenes para el registro. La segunda fase es realizar un registro deformable que

permitirá elevar el nivel de similitud entre la imagen de referencia y la registrada en la primera etapa, corrigiendo pequeños errores que puedan quedar tras esta.

2.1. Registro de imagen

El registro de imagen se define como el proceso de superponer dos o más imágenes provenientes de varios equipos y sensores de imagen tomados a diferentes tiempos y ángulos, o del mismo escenario para alinear geométricamente las imágenes para el análisis [6]. El procedimento del registro de imagen se basa en la detección y alineamiento de características o *features* de la imagen capturada (*query*) y la imagen de referencia (*train*); estimación de parámetros de funciones de mapeo; y el remuestreo y transformación de las imágenes. Estos métodos son únicos para cada problema debido a las singularidades de cada uno, como las distorsiones geométricas de las imágenes; el ruido, la naturaleza de las características de los datos y el nivel de precisión requerido.

Para nuestro caso, el registro de imagen se basará en la búsqueda de características comunes en la etiqueta de interés en ambas imágenes, desarrollar un método de emparejamiento de características y a partir de estas estimar la matriz de transformación con la que se alineará la imagen *query* con la imagen *train* basándose en los desplazamientos rígidos de las características respectivamente. Estos desplazamientos resultan de que el sistema está en producción, lo que provoca que las botellas no vayan todas alineadas en la misma posición, sino que sufren pequeños desplazamientos sobre la cinta. Además, los posibles desfases diferentes del disparador de las cámaras provocan que las botellas presenten cierta incertidumbre en la posición en las imágenes. En un entorno teórico todas las imágenes se tomarían al mismo instante y en la misma posición por lo que no habría que poner de manifiesto el registro de imagen.

Matemáticamente, se puede definir el problema del registro en imagen como:

$$T \circ \phi(x) = T(\phi(x)) = T_\phi(x) \approx R(x), \tag{2.1}$$

donde $\phi(x)$ es la transformación que se va a realizar, $R(x)$ es el conjunto de datos de referencia (de la imagen *train*), y $T(x)$ es el conjunto de datos captados (de la imagen *query*).

2.1.1. Métodos de búsqueda de características

Hay diversos métodos para encontrar características en las imágenes. Estos suelen basarse en primeras y segundas derivadas para analizar cambios en la intensidad y realizar detección de patrones. Los más comunes son: SIFT [7, 8], SURF [9], ORB [10] o el detector de esquinas de Harris [11, 12].

El método SIFT (*Scale-Invariant Feature Transform*) [7, 8] destaca por su capacidad para identificar y describir características invariables ante cambios de escala y orientación. Este método emplea un espacio de escala para localizar puntos clave y utiliza histogramas de gradientes para describir los alrededores.

SURF (*Speeded-Up Robust Features*) [9], una evolución de SIFT, se centra en mejorar la eficiencia computacional manteniendo la robustez del primero. Utiliza filtros tipo caja para calcular respuestas de características, mejorando así la velocidad de cálculo. SURF comparte la invariancia de escala y la rotación característica de SIFT.

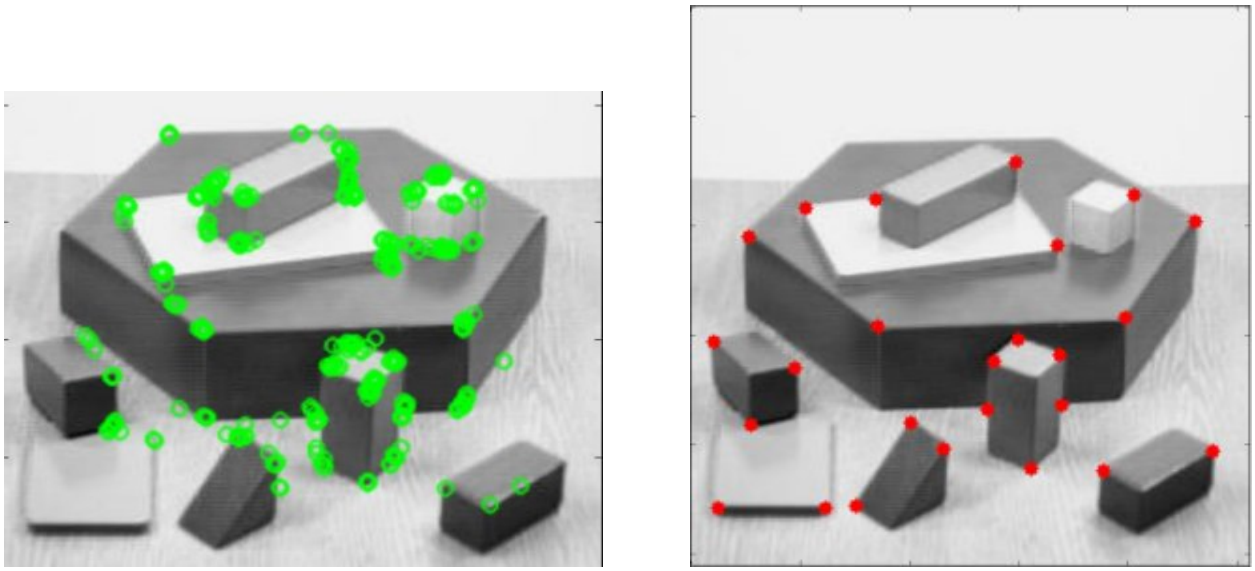
(a) SIFT. **(b)** SURF.

Figura 2.2: Métodos para la obtención de características.

ORB (*Oriented FAST and Rotated BRIEF*) [10] combina el detector FAST [13] y el descriptor BRIEF [14] para lograr una detección y descripción de características eficiente. Fue diseñado con un enfoque en la rapidez y la eficiencia, siendo adecuado para aplicaciones en tiempo real, como el emparejamiento y el reconocimiento de objetos.

Por último, el detector de esquinas de Harris [11, 12] se basa en la identificación de cambios significativos en la intensidad de una imagen para localizar esquinas. Evalúa la variación de la función de autocorrelación local para encontrar esquinas. Si bien es sensible a cambios en la intensidad en dos direcciones, su aplicación principal es la detección de esquinas en imágenes.

Como podemos ver en la Figura 2.3b, el resultado más simple lo obtenemos para el detector de Harris, lo cual para nuestro problema es lo más óptimo, ya que además de ser muy rápido computacionalmente, la solución la obtenemos en las esquinas de los objetos de la imagen, lo cual es más preciso de emparejar que los puntos característicos obtenidos por los demás métodos, que se basan en ”zonas” características, como sobre todo podemos ver en las Figuras 2.2a y 2.2b.



(a) ORB. **(b)** Detector de esquinas de Harris.

Figura 2.3: Métodos para la obtención de características.

2.1.2. Registro afín

El registro afín comprende transformaciones que utilizan puntos característicos extraídos de las imágenes para estimar los parámetros del modelo de mapeado. Estos modelos consideran rotación, traslación y escalado, la transformación afín, y la proyección de perspectiva.

El modelo de rotación, traslación y escalado es el más sencillo. El campo vectorial de desplazamientos (u,v) solo tiene cuatro parámetros:

$$u = s(x\cos(\phi) - y\sin(\phi)) + t_x, \tag{2.2}$$
$$v = s(x\sin(\phi) + y\cos(\phi)) + t_y, \tag{2.3}$$

donde s es el factor de escalado, ϕ es el ángulo de rotación y t_x y t_y son los desplazamientos.

Por otro lado, el modelo de transformación afín es un poco más complejo. Tiene seis parámetros:

$$u = a_0 + a_1x + a_2y, \tag{2.4}$$
$$v = b_0 + b_1x + b_2y. \tag{2.5}$$

Por último, el modelo de proyección en perspectiva es el más complejo. Tiene ocho parámetros:

$$u = \frac{a_0 + a_1 x + a_2 y}{1 + c_1 x + c_2 y}, \tag{2.6}$$

$$v = \frac{b_0 + b_1 x + b_2 y}{1 + c_1 x + c_2 y}. \tag{2.7}$$

En la Figura 2.4 se puede ver como, según el modelo es más complejo, el resultado de la transformación deforma más la imagen original. Aunque, independientemente del modelo, todos los puntos se deforman de forma similar en su entorno más cercano, es decir, todos tienen el mismo patrón de deformación.

(a) Rotación, traslación y escalado. **(b)** Transformación afín. **(c)** Proyección en perspectiva.

Figura 2.4: Diferentes modelos de prerregistro aplicados a una misma imagen.

2.1.3. Registro deformable

Una vez realizado el ajuste grueso con el prerregistro recurrimos al registro deformable para conseguir un ajuste fino [15, 16]. En este tipo de registros los píxeles pueden tener más libertad de movimiento, incluso en su entorno más cercano. El registro deformable es muy conveniente si la solución del problema debe tener un nivel de calidad suficientemente alto, como es en este caso. En la Figura 2.5 vemos cómo puede afectar el registro deformable sobre una imagen, realizando una deformación arbitraria.

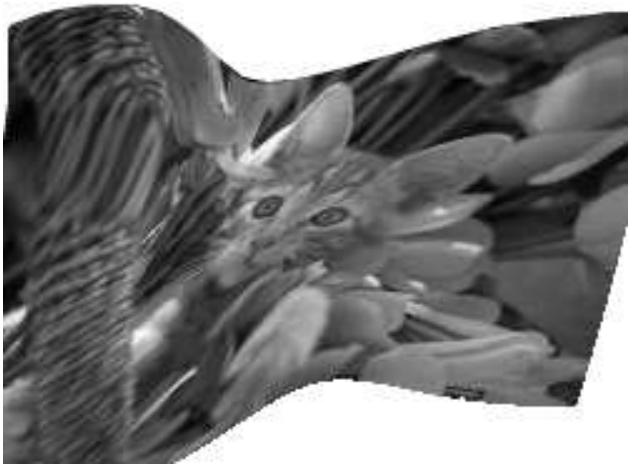
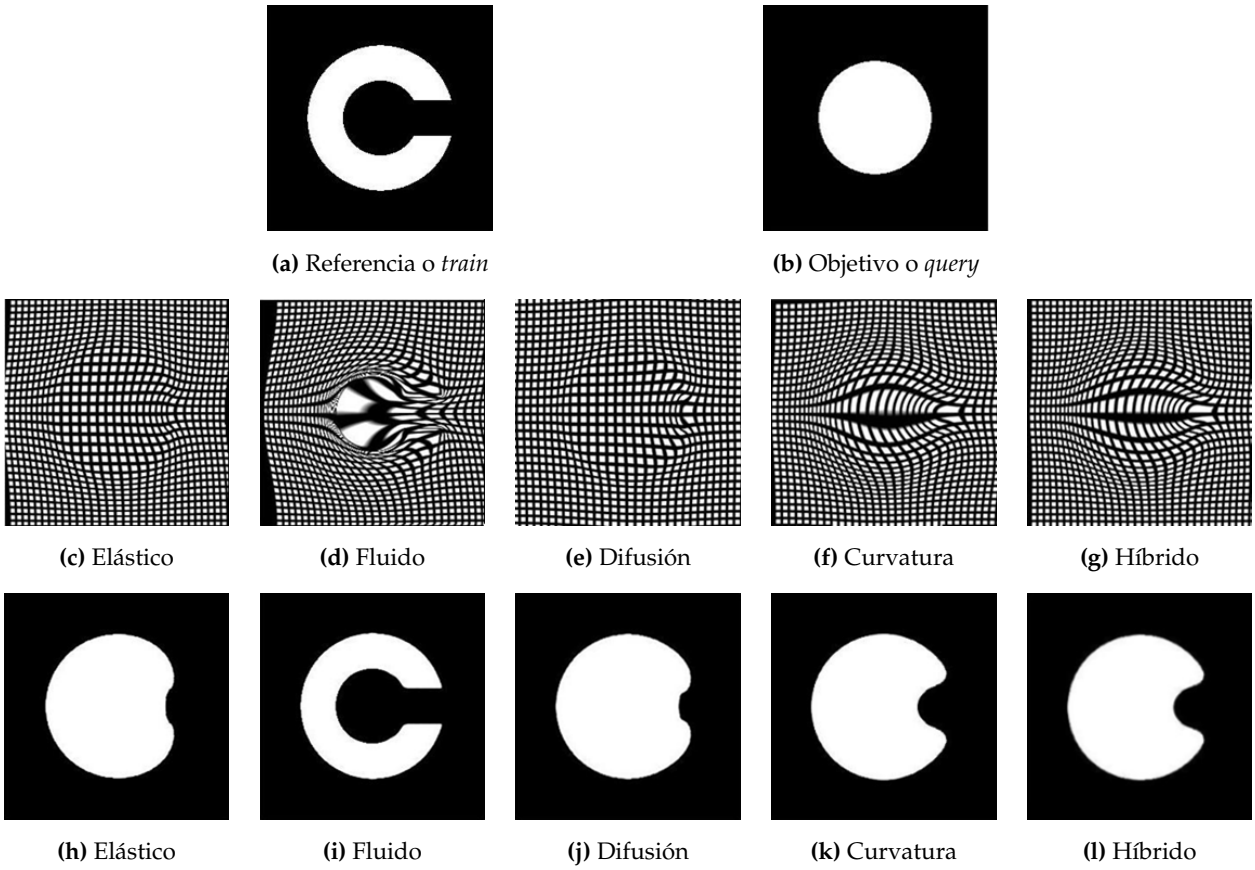


Figura 2.5: Aplicación de una deformación arbitraria a una imagen.

Hay varios tipos de registro deformable [17]: el elástico, que considera el conjunto de datos como un cuerpo elástico; el fluido, que considera el conjunto de datos como un cuerpo visco-elástico; por difusión, el cual minimiza la energía de las derivadas de primer orden, permite las transformaciones de tipo rígido y penaliza las oscilaciones en el campo de desplazamiento; por curvatura, que minimiza la energía de las derivadas de segundo orden, permite las transformaciones afines lineales y permite las oscilaciones en el campo de desplazamiento; y por último el híbrido [18, 19], el cual minimiza la energía de las derivadas fraccionaria de primer (difusión) y segundo orden (curvatura) y es regularizador con características intermedias y ajustables de forma gradual.

En la Figura 2.6 podemos ver la imagen objetivo (subfigura (b)), la cual queremos registrar contra la imagen referencia (subfigura (a)). De la subfigura (h) a la (l), vemos la imagen resultado de aplicar cada modelo, y de la subfigura (c) a la (g), la rejilla que se obtendría al aplicar esa misma deformación a una cuadrícula regular.



(a) Referencia o *train* **(b)** Objetivo o *query*

(c) Elástico **(d)** Fluido **(e)** Difusión **(f)** Curvatura **(g)** Híbrido

(h) Elástico **(i)** Fluido **(j)** Difusión **(k)** Curvatura **(l)** Híbrido

Figura 2.6: Comparación de los diferentes modelos de registro deformable.

2.2. Segmentación y morfología matemática

La segmentación trata de detectar y separar diferentes elementos semánticos de una imagen [12]. Para ello se utilizan técnicas como detección de bordes, uso de gradiente, o lo que usaremos en este proyecto, la morfología matemática.

La morfología matemática [20] permite segmentar la imagen aprovechando las diferentes formas que esta presenta. Existen diferentes operaciones morfológicas, las cuales vamos a describir en esta sección. Todas ellas utilizan un núcleo muy simple llamado *elemento estructurante*. Un elemento estructurante es un pequeño conjunto o subimagen. Suele estar compuesto por píxeles binarios, cuyo valor puede ser o 0 o 255, y puede tener diferentes formas y diferentes centros en función del problema que se vaya a resolver.

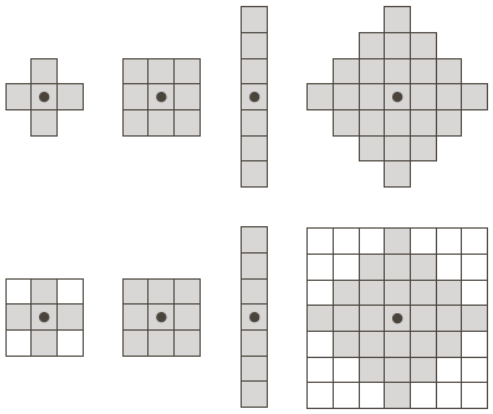


Figura 2.7: Diferentes elementos estructurantes en forma de elemento o de array 2D (imagen).

2.2.1. Erosión y dilatación

La erosión junto con la dilatación son las operaciones morfológicas más simples. Utilizan el elemento estructurante para hacer los elementos de la imagen más pequeños o más grandes, respectivamente, solo teniendo en cuenta la forma del elemento estructurante.

Dados los conjuntos A y B en Z^2, se define la erosión de dos formas:

$$A \ominus B = \{z \mid (B)_Z \subseteq A\}; \tag{2.8}$$

$$A \ominus B = \{z \mid (B)_Z \cap A^c = \emptyset\}. \tag{2.9}$$

En imagen podemos definir estas ecuaciones de forma simple como que se recorre la imagen con el elemento estructurante, y comprueba si todo el elemento estructurante "cabe" o no en los elementos de la imagen, para cada posición, o píxel. El píxel que se está evaluando, es decir, el píxel en el que se encuentra el centro del elemento estructurante en esa iteración, tomará el valor 255 (para imágenes binarias), si el elemento estructurante está completamente contenido en el elemento, si no, ese punto tomará un valor de 0. Así, esta operación afecta únicamente a los contornos de los elementos.

Por otro lado, su operación opuesta, la dilatación se define con las siguientes ecuaciones:

$$A \oplus B = \{z \mid (\hat{B})_z \cap A \neq \emptyset\}, \tag{2.10}$$

$$A \oplus B = \{z \mid [(\hat{B})_z \cap A] \subseteq A\}. \tag{2.11}$$

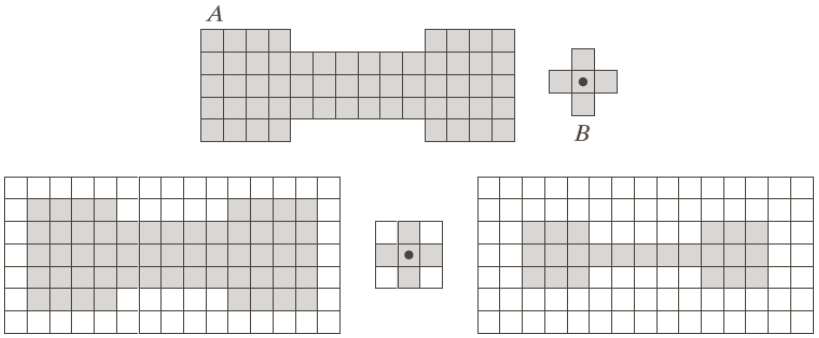


Figura 2.8: Ejemplo de erosión con elemento estructurante en forma de cruz (+).

La dilatación se puede entender como el proceso contrario a la erosión. Se recorre también la imagen con el elemento estructurante, pero en este caso, si el centro del elemento estructurante coincide con algún elemento de la imagen, es decir, el punto a evaluar no es nulo, todos los puntos sobre los que queda el elemento estructurante, da igual si son nulos o no, en el resultado final tendrán valor no nulo. Por tanto la dilatación también afectará sobre todo a los contornos, extendiéndolos o *dilatándolos*.

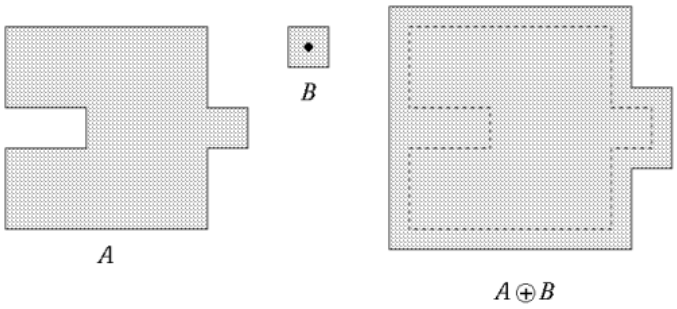


Figura 2.9: Ejemplo de dilatación con elemento estructurante cuadrado.

2.2.2. Apertura y cierre

La apertura y el cierre son operaciones morfológicas un poco más complejas que la erosión y la dilatación, ya que no afecta a los contornos indiferentemente de la forma.

La apertura de un conjunto A con un elemento estructurante B se define como:

$$A \circ B = (A \ominus B) \oplus B. \tag{2.12}$$

Es decir, erosión seguida de dilatación. Aunque también tiene una interpretación geométrica:

$$A \circ B = \cup\{(B)_z | (B)_z \subseteq A\}. \tag{2.13}$$

Se puede explicar de forma simple como que al recorrer la imagen con el elemento estructurante, en los puntos donde el elemento estructurante cabe completamente en los elementos de la imagen, el píxel resultado tiene el mismo valor que el píxel origen, pero en los puntos en los que el elemento estructurante no puede entrar al recorrer el elemento de la imagen por su interior, porque dejaría de estar completamente contenido, se eliminan. Además, si el elemento de la imagen estuviera contenido en el elemento estructurante, porque fuera más pequeño, se eliminaría por completo.

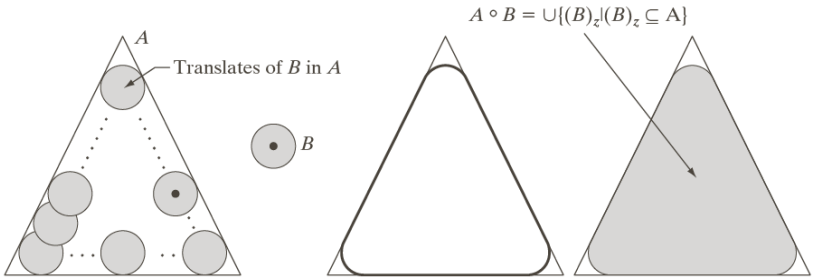


Figura 2.10: Ejemplo de apertura de un triángulo con un elemento estructurante circular.

La apertura sirve para agrandar el espacio entre zonas estrechas, suavizar contornos o eliminar elementos no deseados como ruido.

Por otro lado, el cierre es la operación dual o inversa a la apertura. Se puede definir como:

$$A \bullet B = (A \oplus B) \ominus B. \tag{2.14}$$

Esto es, dilatación seguida de erosión. Al igual que la apertura, el cierre tiene una interpretación geométrica:

$$A \bullet B = \{z|(\hat{B})_z \cap A \neq \emptyset\}. \tag{2.15}$$

En este caso, el elemento estructurante se tiene en cuenta al recorrer el contorno de los elementos de la imagen pero por el exterior. En el momento en el que el elemento estructurante no quede totalmente excluído del elemento, ya que parte de él quedaría solapado con el elemento, se "rellena" la zona que queda entre ambos elementos. Por lo que el cierre "cierra" las zonas estrechas que sean más pequeñas que el elemento estructurante.

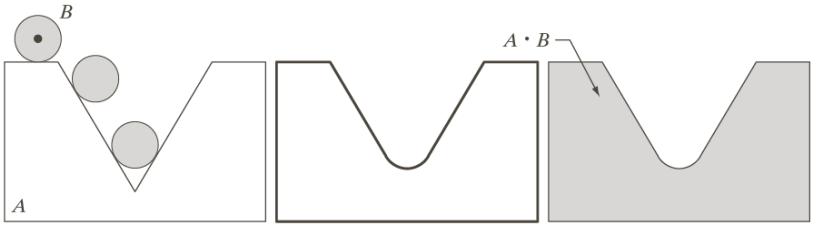


Figura 2.11: Ejemplo de cierre de una zona abierta triangular con un elemento estructurante circular.

2.2.3. Reconstrucción morfológica

Otra de las operaciones que se utilizan en este proyecto es la reconstrucción morfológica. Esta es una operación más compleja, debido a que involucra dos imágenes y un elemento estructurante. El *marcador* (F) es la imagen de partida de la transformación; la *máscara* (G) es la imagen que restringe la transformación; y por último, el elemento estructurante (B) es el que define la conectividad. Hay varios tipos de reconstrucción morfológica: dilatación y erosión geodésica; o la reconstrucción morfológica por erosión y dilatación.

En este proyecto usamos la reconstrucción morfológica por dilatación, la cual se define como:

$$R_G^D(F) = D_G^{(k)}(F), \tag{2.16}$$

hasta que se alcanza una situación estable con un k:

$$D_G^k(F) = D_G^{(k+1)}(F). \tag{2.17}$$

En simples palabras, en cada iteración, se dilata el marcador actual con el elemento estructurante y se restringe a la máscara, hasta que el resultado de una iteración es el mismo de la iteración anterior.

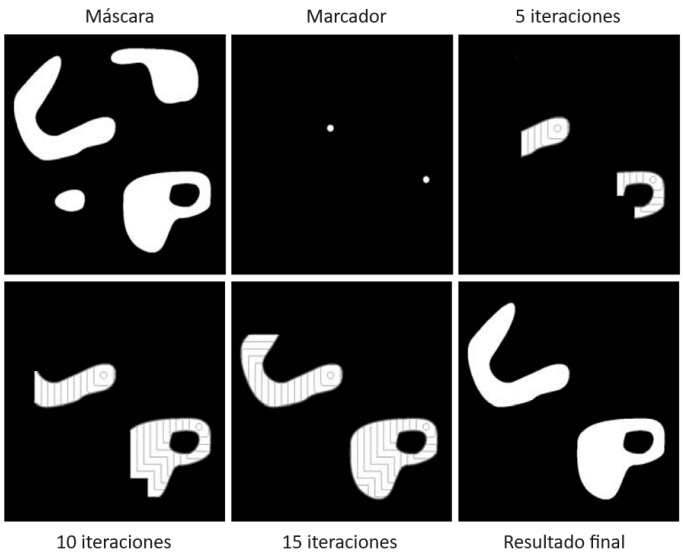


Figura 2.12: Ejemplo del desarrollo de una reconstrucción morfológica.

Gracias a la reconstrucción morfológica podemos recuperar la forma inicial de elementos deseados, los cuales están presentes en el marcador inicial, y excluir los no deseados.

2.2.4. Morfología en niveles de gris

Hasta ahora hemos definido la morfología para imágenes binarias, con valores o 0 o 1 (255), pero lo más común es usar la morfología sobre imágenes en niveles de grises. Podemos extrapolar lo que sabemos de operaciones morfológicas hasta ahora cambiando *contornos* por *niveles de intensidad*.

En cuanto a erosión y dilatación, los niveles que tomará el resultado de estas dependerá del entorno. Al erosionar, el contorno erosionado tomará el valor de los píxeles cercanos al contorno del objeto del fondo cercano; mientras que al dilatar, el contorno dilatado tomará el valor de los píxeles cercano pertenecientes al objeto.

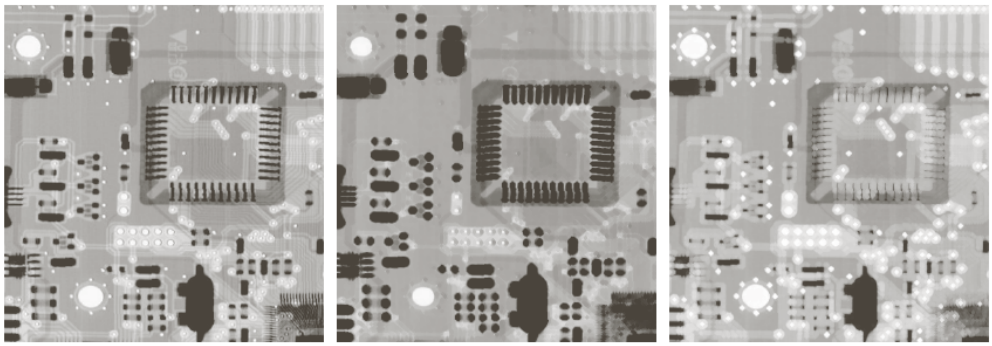


Figura 2.13: Imagen original en niveles de gris seguida de su erosión y dilatación, en ese orden, con un elemento estructurante circular de radio dos.

Podemos ver en la Figura 2.13 que, en niveles de gris, la erosión agranda los elementos oscuros y la dilatación agranda los elementos brillantes.

Por otro lado, la apertura y el cierre en niveles de gris, eliminan picos brillantes u oscuros respectivamente, más pequeños que el elemento estructurante.

Podemos ver en la Figura 2.14 cómo afectan la apertura y el cierre con un elemento estructurante plano sobre el corte longitudinal de una imagen, esto es el perfil de intensidad de los píxeles a lo largo de una fila de la imagen.

Como podemos ver en la Figura 2.15, tanto la apertura como el cierre no afectan a todos los contornos indiferentemente, sino que solo afectan a las zonas brillantes u oscuras, respectivamente, donde no cabe el elemento estructurante. En la apertura vemos que ninguna de la zonas o elementos oscuros han sido afectados, y en el cierre ninguno de los elementos brillantes.

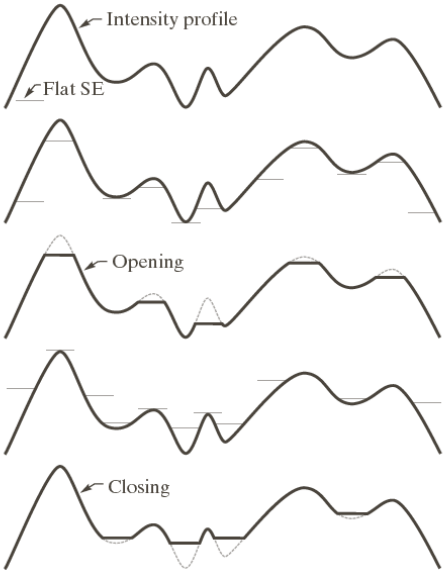


Figura 2.14: Apertura y cierre sobre una señal unidimensional.

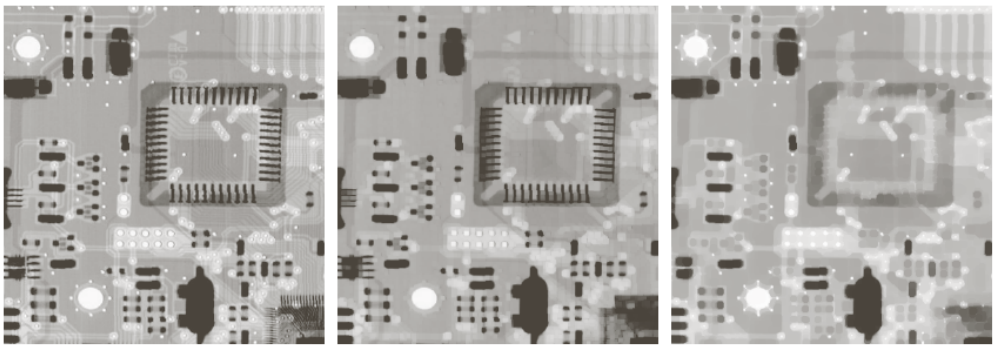


Figura 2.15: Imagen original en niveles de gris, seguida de una apertura con un elemento estructurante circular de radio dos y un cierre de radio cinco (independientes).

CAPÍTULO 3

Diseño y desarrollo de la solución

El diseño de la solución software consta de tres etapas diferenciadas. La primera consiste en obtener una máscara mediante segmentación, utilizando operaciones morfológicas tanto de la imagen *query*, como de la imagen *train*. Esta máscara permitirá delimitar la etiqueta que es propensa a tener golpes y por tanto rasgaduras.

La siguiente etapa es el prerregistro de la imagen *query* contra la imagen *train* mediante emparejamiento de puntos clave o *key points* de ambas imágenes. Estos puntos los encontraremos utilizando el método de la detección de esquinas de Harris y posteriormente pasarán por un método de emparejamiento el cual toma unos puntos que se consideran correctos y a partir de sus diferencias en x e y, compararlas punto a punto entre todos los puntos encontrados, buscando la mayor similitud por distancia en píxeles, ya que las variaciones entre una imagen y otra, al estar emparejadas en ángulo de rotación, se consideran un desplazamiento rígido, o casi-rígido.

Por último, vamos a realizar el ajuste fino mediante registro deformable, debido a que este desplazamiento entre las dos imágenes no es estrictamente rígido, por pequeñas inclinaciones, la deformación producida por la óptica, o fallos de resolución de la base de datos al emparejar las imágenes. Al realizar el registro deformable podemos corregir las imperfecciones del prerregistro, las cuales son del tamaño de unos pocos píxeles. Este tipo de registro es muy lento en comparación con el rígido, de ahí que se use primero el rígido, para dejar lo más similares las imágenes y así al usar el deformable, se necesitará menos iteraciones y reducirá el tiempo de cómputo de esta

etapa. El tiempo que tarde el algoritmo en devolver si la etiqueta presenta algún desperfecto es crucial en un entorno industrial, en el que la velocidad de producción es muy alta, y por tanto, la velocidad a la que el sistema decide debe serlo también.

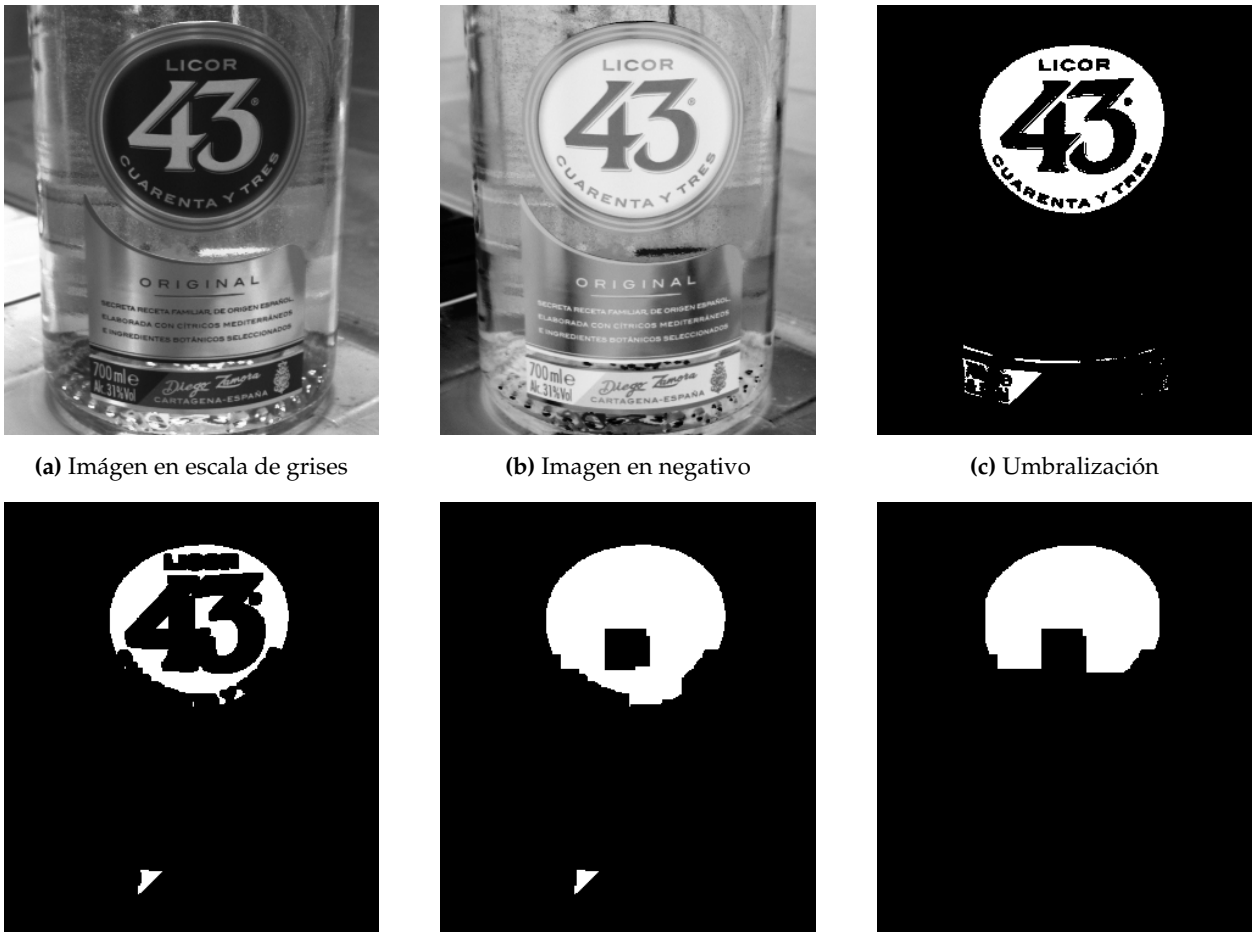
3.1. Segmentación de la etiqueta

Partimos de que la etiqueta de interés está formada por un círculo negro, donde se encuentran los dígitos *"43"* y las letras *"CUARENTA Y TRES"*, y este círculo a su vez está circunscrito en un borde con circunferencias de tonos dorados. Entonces, para segmentar la etiqueta, vamos a aprovechar ese fondo negro. Primero convertimos la imagen a escala de grises y obtenemos también la imagen negativo de esta.

Ahora el fondo negro pasa a ser blanco, y por tanto, aplicamos un umbral por el que pasará esta parte, además de algunas zonas de la imagen que no son deseadas, como ruido por reflejos o partes de otras etiquetas que también presentan zonas oscuras en la imagen original. El siguiente paso es quedarnos solo con el fondo negro, que ahora es blanco, de la etiqueta superior. Para ello, primero erosionamos la imagen umbralizada, después le aplicamos un cierre, y por último una apertura. Al erosionar primero, reducimos el tamaño del ruido, lo que nos permitirá realizar el cierre sin que se una ruido a la zona de interés, y además podremos aplicar una apertura más pequeña, asegurándonos de que no se pierde la etiqueta superior cuando la botella se encuentra en ciertos ángulos en los que esta etiqueta se ve parcialmente, y por tanto, tiene menos área. Con el cierre nos aseguramos de que el fondo blanco de la etiqueta superior no se elimine al aplicar la apertura tras haber erosionado, ya que los números y letras de dentro no pasan el umbral (pasan a ser negras) y esto provoca que la apertura pueda eliminarla. Este proceso se muestra en la Figura 3.1.

Una vez tenemos un único componente, el cuál es una parte del fondo blanco de la etiqueta, realizamos una reconstrucción, usando el resultado de la apertura como marcador (véase Figura 3.1f), y el resultado de la umbralización (Figura 3.1c) como máscara. De esta forma obtendremos de la imagen umbralizada, solo la parte de la etiqueta que se necesita (véase Figura 3.2a). Cerramos para rellenar los píxeles donde se encuentran loa números y las letras, y de esta forma obtenemos la máscara inicial, como vemos en la Figura 3.2b.

La máscara inicial, debido a la composición de la etiqueta, no ocupa todo el fondo negro, ya que el borde es rojo y al hacer la umbralización del negativo no lo contiene. Al tener la máscara inicial, podemos aprovecharla para dilatarla un poco y filtrar la imagen (véase Figura 3.3a). Así podemos aplicar un umbral más bajo sin que se cuele todo el ruido del fondo, el cual no permite segmentar bien la etiqueta, sobre todo en los ángulos en los que la etiqueta queda parcialmente visible, tocando la zona negra el fondo de la imagen que también es oscuro. Tras aplicar el umbral



(a) Imágen en escala de grises **(b)** Imagen en negativo **(c)** Umbralización

(d) Erosión **(e)** Cierre **(f)** Apertura

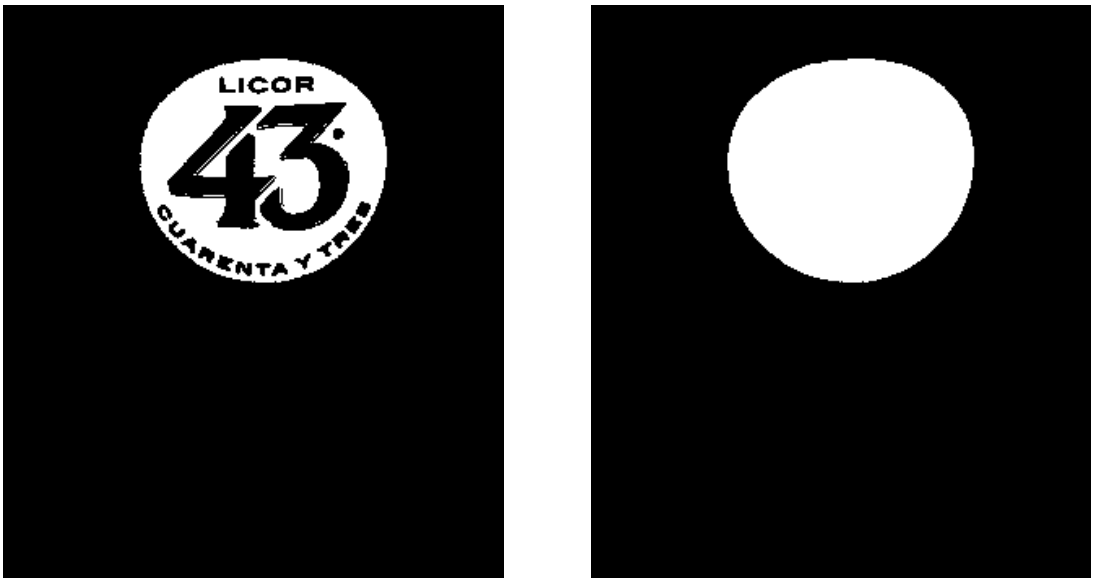
Figura 3.1: Primeros pasos del proceso de extracción de la máscara para la etiqueta de análisis.

(Figura 3.3b), aplicamos una pequeña apertura por cualquier pequeño ruido que se pueda colar (Figura 3.3c), y un cierre para cerrar los dígitos como anteriormente (Figura 3.3d). Así obtenemos la máscara final, la cual ocupa toda la zona negra, incluyendo el borde rojo.

En la Figura 3.3e vemos la comparación entre la máscara inicial y la máscara final. Comprobamos que efectivamente la máscara final contiene el borde rojo que se ha comentado que no pasaba con la máscara inicial, obteniendo una segmentación completa de la etiqueta objetivo.

3.2. Registro rígido

Una vez tenemos la máscara en la cual se van a buscar los puntos que se van a emparejar de ambas imágenes, vamos a encontrar los diferentes elementos de la etiqueta. Como ya sabemos, en la etiqueta están presentes números y letras. Debido a la diferencia en forma de estos, vamos



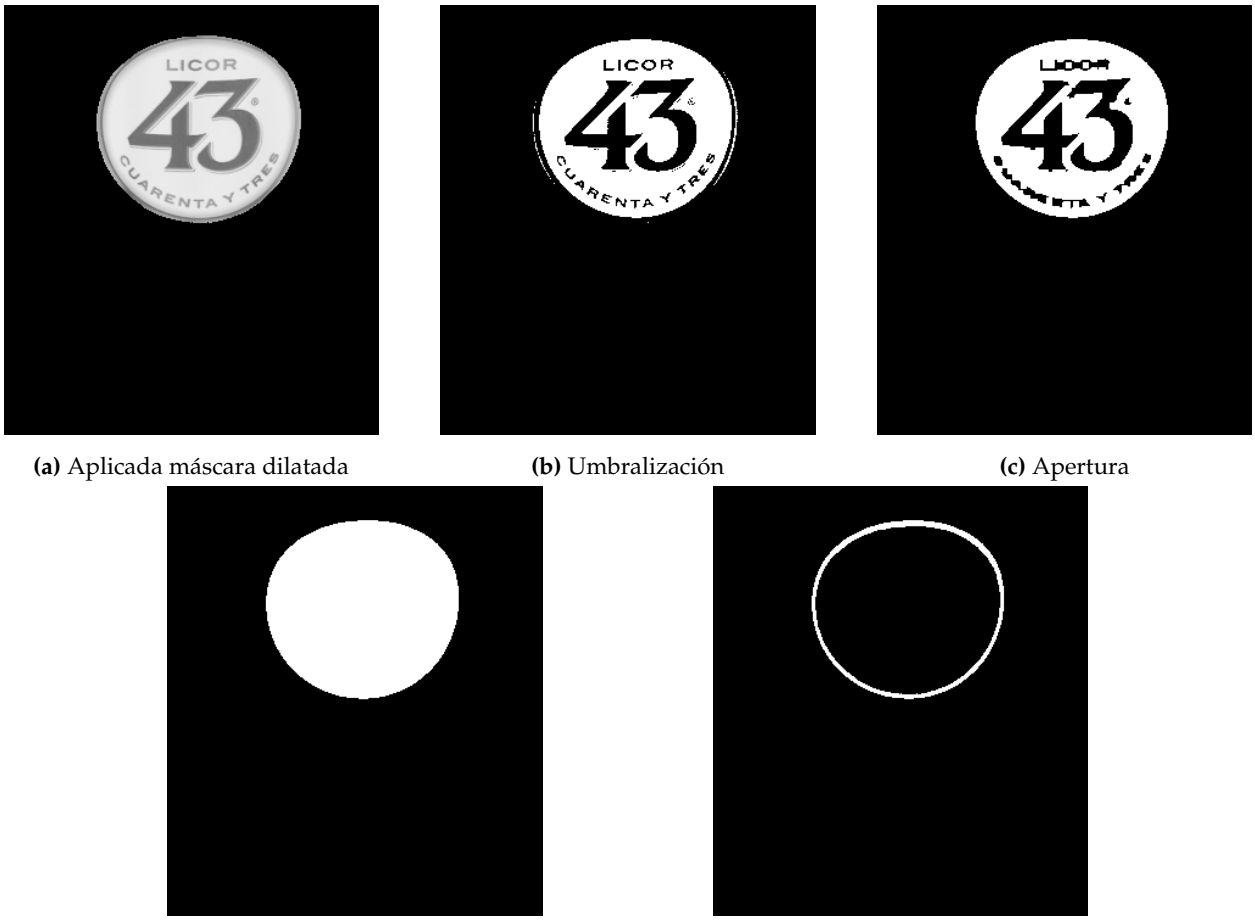
(a) Reconstrucción morfológica. **(b)** Máscara inicial.

Figura 3.2: Obtención de la máscara inicial.

a buscar los puntos por separado, por tanto, el primer paso será dividir los números de las letras, y más a fondo, las letras de *"LICOR"* de las de *"CUARENTA Y TRES"*. Para ello recurriremos nuevamente a la morfología matemática. Para segmentar los números, aplicamos una apertura en la cual solo caben las letras, y por tanto se eliminan (Figura 3.4a). Una vez tenemos los números, para obtener las letras es tan simple como restarle a la imagen original los números, y por tanto nos quedan las letras (Figura 3.4b). Por último, para segmentar la palabra *"LICOR"*, hacemos un cierre a las letras y como esta queda arriba, nos quedamos el componente que se encuentre más arriba en la imagen (Figura 3.4c).

En las tres subfiguras de la segunda fila de la Figura 3.4 vemos los resultados finales de la segmentación de cada uno de los elementos de la etiqueta: los números *"43"* (Figura 3.4d), las letras *"CUARENTA Y TRES"* (Figura 3.4e)y por último el componente cerrado correspondiente a las letras *"LICOR"* (3.4f). Este último se deja en forma de componente cerrado debido a que se va a utilizar más adelante a la hora del emparejamiento, a diferencia de los otros elementos que solo se utilizan para buscar los puntos característicos.

Una vez obtenidos los diferentes elementos de la etiqueta, aplicaremos el método de detección de esquinas de Harris a cada uno de ellos utilizando la función de OpenCV *goodFeaturesToTrack*, la cual puede tomar cuatro parámetros. La imagen a la que se le aplica la detección, en nuestro caso cada elemento de la etiqueta. El número de puntos a detectar, si aumenta, el número de puntos detectados aumentará. Un umbral de calidad que hará que de todos los puntos detectados, solo devuelva los que pasen el umbral. Por último la separación mínima en píxeles permitida entre punto y punto detectado.



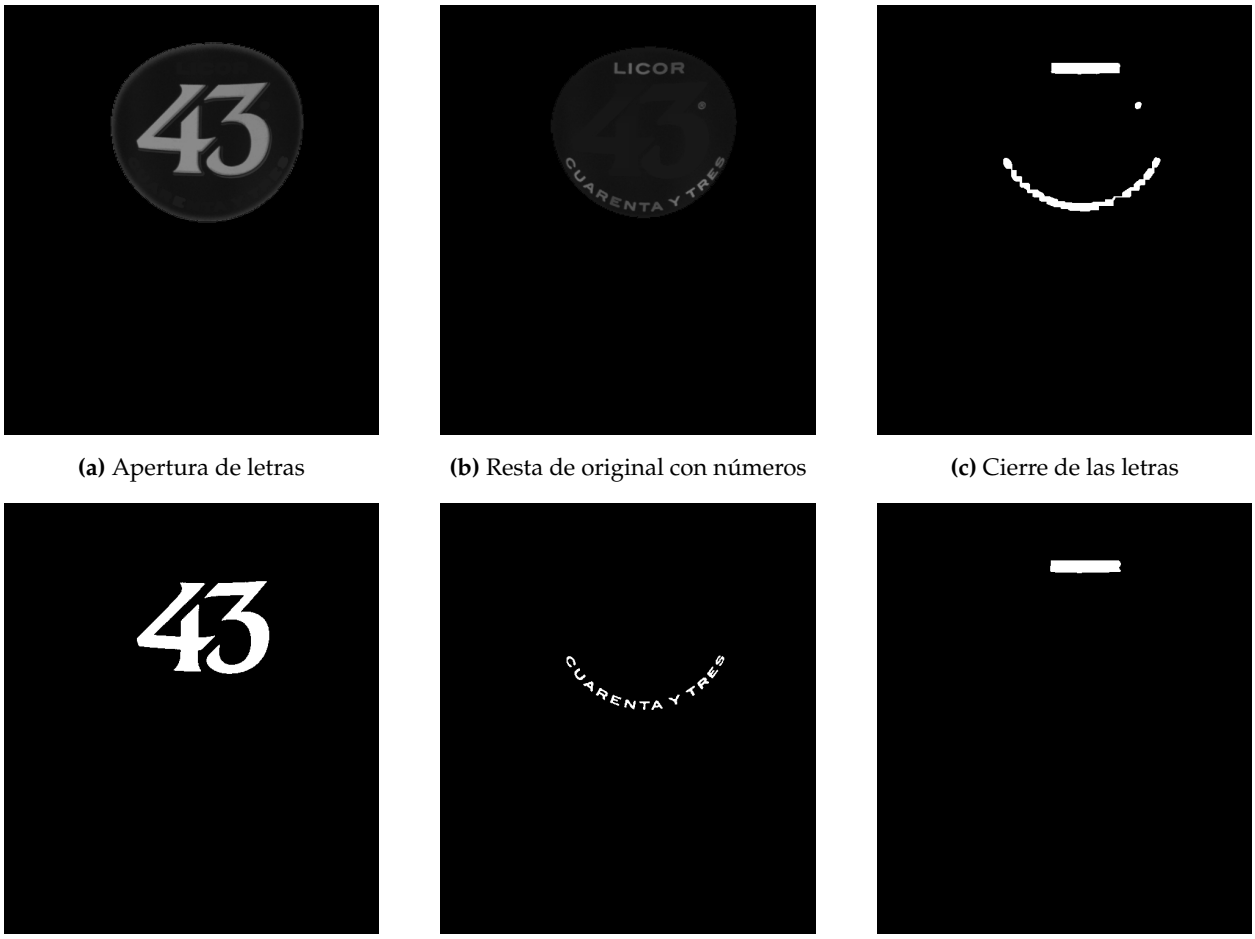
(a) Aplicada máscara dilatada **(b)** Umbralización **(c)** Apertura

(d) Cierre **(e)** Máscara inicial vs máscara final

Figura 3.3: Obtención de la máscara final.

Como hemos comentado anteriormente, para cada elemento se le pasará a la función diferentes parámetros, en función de su forma. Para los números, al ser grandes y tener pocas esquinas, el número de puntos detectables no será necesariamente alto, aunque el umbral sí, ya que son pocos, pero son muy significativos, y así forzamos a que no se detecten puntos en zonas de alta curvatura donde no hay esquinas reales. Para las letras, al ser mucho más pequeñas y tener gran "densidad de esquinas", el número de puntos detectables deberá ser mayor, y el umbral mayor incluso que para los números. Esto evitará que se emparejen puntos muy cercanos de forma incorrecta, es decir, que los punto emparejados provengan de una esquina o letra de la etiqueta diferente en cada imagen. Para la palabra *"LICOR"*, al ser más grande que *"CUARENTA Y TRES"* y tener menos esquinas, pondremos un número de puntos detectables bajo, y un umbral de esquinas no necesariamente alto.

Lo que obtenemos a la salida de esta función es una lista de puntos, es decir, una lista con las coordenadas de cada punto detectado. Por tanto, tendremos una lista por cada componente de la etiqueta (números, letras y *LICOR*). En total son seis listas, tres de la imagen *query* y tres



(a) Apertura de letras **(b)** Resta de original con números **(c)** Cierre de las letras

(d) Números **(e)** Letras **(f)** Licor

Figura 3.4: Segmentación de los tres elementos clave para realizar el registro rígido.

de la imagen *train*, las cuales vamos a concatenar para tener una única lista para cada imagen. El problema ahora es ordenar estas listas haciendo que cada posición posea el punto correspondiente de ambas imágenes para aplicarlas al registro. Esto es el emparejamiento que íbamos buscando.

El emparejamiento se realizará basándose en las diferencias en la coordenada x y en la coordenada y que hay entre los puntos detectados de ambas imágenes. Para ello, tomamos como referencia los cuatro puntos que obtenemos del contorno del elemento *LICOR* (véase Figura 3.4f), ya que estos puntos se pueden emparejar de forma automática, al conformar las cuatro esquinas de un rectángulo. Estas esquinas se pueden clasificar y ordenar fácilmente a partir de sus coordenadas.

Una vez ordenados los puntos, obtenemos las diferencias dx y dy para estos cuatro puntos entre ambas imágenes. Para ello simplemente restamos punto a punto, con su correspondiente en ambas imágenes, en cada coordenada. Entonces realizamos el emparejamiento de los puntos por elemento, es decir, realizamos tres emparejamientos. De esta forma ahorramos en tiempo de

(a) Imagen *train*. **(b)** Imagen *query*.

Figura 3.5: Puntos detectados en la imagen *train* y la imagen *query* con el método de Harris.

cómputo al tener que recorrer listas más cortas.

El emparejamiento se basa en recorrer ambas listas de puntos, la de la imagen *train* y la de la imagen *query*, y ver la diferencia entre un punto de una lista con todos los puntos de la otra lista. Las diferencias obtenidas en cada iteración del bucle se comparan con la media de las diferencias obtenidas para el elemento *LICOR*, las cuales se consideran correctas. Como las botellas pueden solo sufrir un desplazamiento rígido en el espacio, al estar emparejadas en ángulo de rotación, las diferencias entre un punto y su correspondiente de la otra imagen, deben tener valores muy similares a las de la palabra *LICOR*.

Cuando acaba la comprobación de un punto respecto a todos los de la otra imagen, se añade a dos listas nuevas el punto que se ha analizado, y el punto que presenta mayor similitud al desplazamiento que se toma como referencia. Si ningún punto pasa el umbral, consideramos que ese punto se ha detectado en una imagen, pero en la otra no y por tanto no tiene correspondencia y no se añade a ninguna lista. Así nos aseguramos que cada punto que se tendrá en cuenta tendrá un punto emparejado, y que de los puntos que sí tienen correspondencia, se empareja el más probable a cumplir con el desplazamiento que hay entre una imagen y otra.

Podemos ver en la Figura 3.6 los puntos emparejados de ambas imágenes. Los puntos de la imagen *train* son los verdes, y los de la imagen *query* los puntos rojos. En este ejemplo se puede ver que el desplazamiento de la botella que ha pasado por la cinta en producción es en dirección noroeste, respecto de la botella de la base de datos.

(a) *Train* **(b)** *Query*

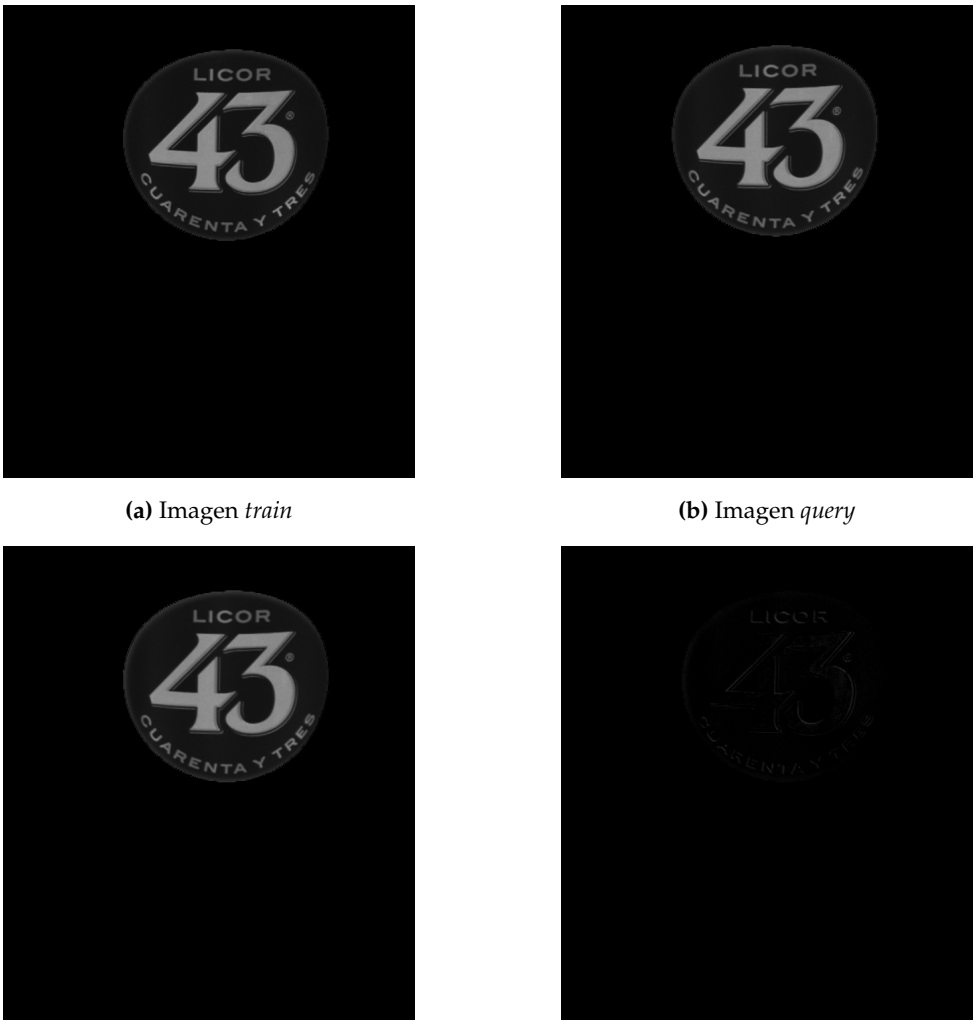
Figura 3.6: Puntos emparejados en la imagen *train* y la imagen *query*.

Estas listas de puntos emparejados se pasan como argumento a la función de OpenCV *estimateAffine2D*, la cual devuelve la matriz de transformación que se le aplicará a la imagen *query* para registrarla contra la imagen *train*. Para hacer el registro usamos la función *warpAffine*, también de OpenCV, la cual requiere: la imagen que se va a registrar, la matriz de transformación, y el tamaño de la imagen resultado.

Como podemos ver en las imágenes (c) y (d) de la Figura 3.7, la imagen *query* ha quedado registrada contra la imagen *train*, tal que al hacer la resta de ambas queda prácticamente negra la imagen resultado. Aunque pueden verse algunos contornos de los elementos, debido a que el desplazamiento no es estrictamente rígido y los vectores de desplazamiento para cada punto son diferentes, haciendo que se desvíen por influencia de los demás. Para corregir este defecto y conseguir un resultado perfecto píxel a píxel, recurrimos al siguiente paso: el registro deformable. Es en este momento cuando podríamos detectar si una etiqueta está correcta o presenta algún desperfecto, ya que al comparar la imagen registrada con la de la base de datos, veríamos las diferencias que presentara.

3.3. Registro deformable

El registro deformable permite, en cada zona de la imagen, aplicar una deformación distinta, sujeta a ciertas restricciones. En nuestro caso utilizaremos un modelo creado en el Departamento de Tecnologías de la Información y las Comunicaciones de la UPCT. Este modelo permite aplicar



(a) Imagen *train*

(b) Imagen *query*

(c) Imagen registrada

(d) Comparación *train* con registrada

Figura 3.7: Registro de la imagen *query* con la imagen *train*.

una deformación por difusión (σ=1) o por curvatura (σ=2). En la formulación del registro deformable aparece un término que mide la similitud entre las dos imágenes a registrar; y un término que mide la suavidad o regulación del campo vectorial de la deformación.

La medida de similitud o *Similarity Measure*, el cual se refiere a la métrica utilizada para medir la similitud entre las imágenes, el cual puede tomar los valores: SSD (Sum of Squared Differences) [12], mide la diferencia cuadrática entre los píxeles de las imágenes; IM (Mutual Information) [21], mide la información mutua entre las imágenes, indicando cuánta información de una imagen se puede obtener a partir de la otra; y CR (Correlation Ratio) [22, 23], mide la relación entre las intensidades de píxeles en las imágenes. Por último, el parámetro α controla la influencia del término de regularización en el modelo frente al parecido entre las imágenes registradas, lo cual permite ajustar cuánto de suave puede ser el modelo, es decir, un α bajo dará resultado vectores más diferentes, y un α alto, hace más suave la deformación, es decir, los vectores son más

parecidos.

Este modelo presenta una interfaz que permite realizar el análisis, aunque en la implementación de este, no será necesaria.

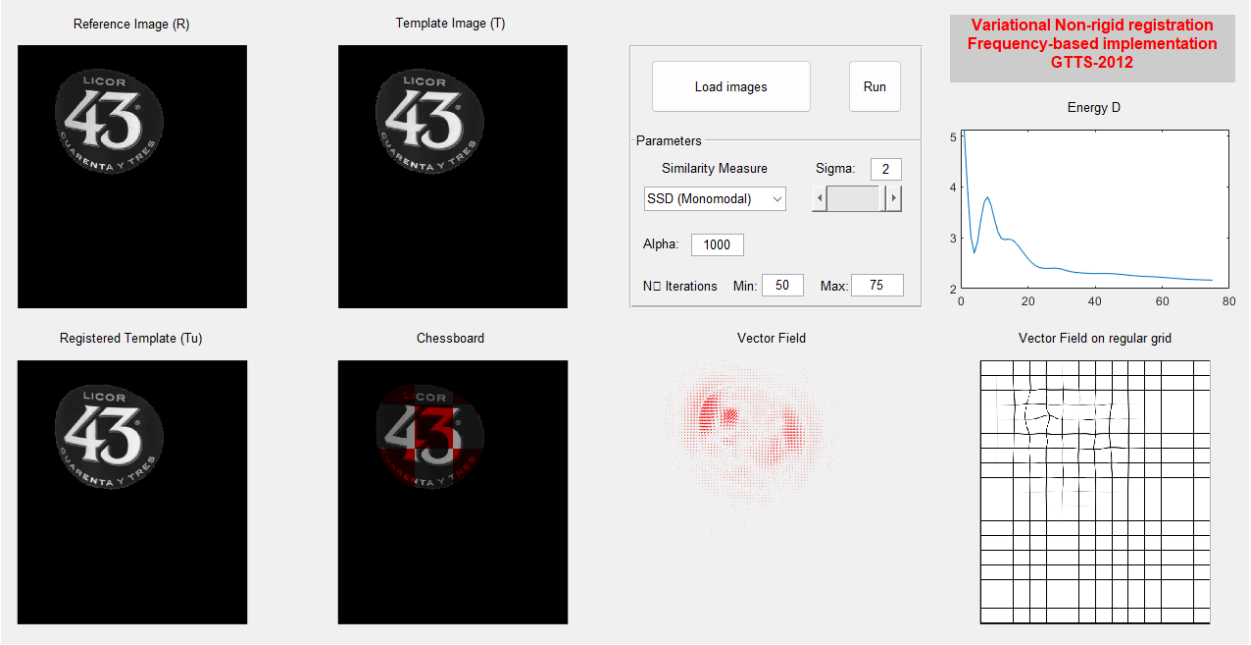


Figura 3.8: Ejemplo de uso de la interfaz del registro deformable con un σ=2, α=1000, SSD como medida de similitud y un rango de iteraciones entre 50 y 75.

Esta interfaz permite ajustar el rango de iteraciones mínimo y máximo que se llevarán a cabo a la hora de buscar la convergencia del modelo, asegurando sobre todo con el umbral de iteraciones mínimo que no se quede el modelo estancado en un mínimo local, como podemos ver en la Figura 3.8 entorno a la iteración número 5. Además, podemos ver en esta interfaz, como quedaría una imagen superpuesta a la otra en forma de tablero de ajedrez (*Chessboard*), el campo de vectores (*Vector Field*) que se genera con la transformación, y una rejilla cuadricular (*Vector Field on regular grid*) que se va deformando siguiendo la misma transformación que la imagen objetivo.

CAPÍTULO 4

Resultados

En este capítulo, veremos el análisis seguido para la elección de los parámetros óptimos del registro deformable para nuestro caso de estudio y los resultados obtenidos en términos de mejora de PSNR en las diferentes etapas del proceso.

4.1. Elección de los parámetros óptimos para el registro deformable

Se ha realizado un análisis variando los parámetros del modelo del registro deformable y comparando los niveles de PSNR obtenidos al registrar una imagen obtenida del registro rígido, contra la imagen *train* asociada.

En la Tabla 4.1, se muestran los diferentes resultados obtenidos, siendo el mostrado en la última fila el que consideramos más aceptable ya que tenemos un bajo número de iteraciones, lo cual nos hace ganar en rapidez, y el parámetro α es lo suficientemente alto para que el registro tenga carácter suave, sin llegar a ser tan alto que no dejaría posibilidad de aplicar el modelo deformable. Preferimos que tenga algo de carácter rígido para que el desplazamiento de los vectores siga teniendo sentido con la naturaleza del problema (un desplazamiento en el tiempo del objeto, visto desde la misma cámara). Por tanto, utilizaremos un registro deformable con un $\sigma = 2$ (registro

σ	Medida de similitud	α	$n_{iteraciones}$	Valor de similitud
1	SSD	100	200	47,17 dB
1	SSD	500	200	45,62 dB
1	CR	100	200	99,84 %
1	CR	500	200	99,77 %
2	SSD	1000	100	45,12 dB
2	SSD	1000	100	44,88 dB
2	SSD	4000	100	44,71 dB
2	SSD	4000	50	44,51 dB
2	SSD	1000	50	44,92 dB
2	SSD	2000	50	44,72 dB

Tabla 4.1: Resultados obtenidos al probar diferentes configuraciones de los parámetros del registro deformable.

por curvatura), una medida de similitud SSD, un $\alpha = 2000$, y un número de iteraciones igual a 50.

4.2. Resultados obtenidos en las diferentes etapas del proceso

En este apartado, mostraremos los resultados obtenidos para tres casos diferentes, en tres etapas del proceso: el punto inicial, en el que no se ha realizado ningún registro; tras el registro afín; y por último, tras el registro deformable. En ellos se compara la imagen de esa etapa con la imagen de referencia (*train*).

En la Figura 4.1 se representan las diferentes imágenes obtenidas resultado de restar cada imagen de cada etapa con la imagen *train*. Vemos cómo tras cada etapa la diferencia entre las imágenes va disminuyendo. Para poder medir esta mejora, vamos a realizar un análisis numérico comparando la PSNR o Relación Señal-Ruido de Pico. Esta se define como:

$$PSNR = 20 \cdot \log_{10}\left(\frac{1}{\sqrt{MSE}}\right), \tag{4.1}$$

donde MSE es el error cuadrático medio entre la imagen que queremos comparar con la imagen de referencia:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}\left(Imagen_{original} - Imagen_{transformada}\right)^2. \tag{4.2}$$

Con la PSNR se puede medir el nivel de parecido de una imagen con la referencia. Cuanto mayor sea el valor de la PSNR más se parecerá a la imagen original. En nuestro caso, cuanto mayor sea la PSNR, más se parecerá la imagen que queríamos registrar (*query*) con la imagen de referencia de la base de datos (*train*).

(a) **(b)** **(c)**

(d) **(e)** **(f)**

(g) **(h)** **(i)**

Figura 4.1: Conjunto de imágenes diferencia para tres casos a lo largo de las etapas del procesado. Cada fila corresponde a cada uno de los casos. La primera columna muestra la diferencia en el punto inicial, la segunda, tras realizar el registro afín, y tercera, tras realizar el registro deformable, para cada caso.

Como podemos ver en la Figura 4.2, para la primera imagen como para la tercera, obtenemos tras el registro rígido un nivel de PSRN un poco inferior a 40 dB y tras el registro deformable, unos valores superiores a 40 dB. Por otro lado, la segunda imagen presenta un reflejo en la zona central de la etiqueta. Esto es debido a que las condiciones de iluminación de la cabina por donde pasan las botellas no son estáticas, si no que dependen de lo juntas o separadas que vayan las botellas en la línea. Este efecto provoca que la PSNR baje, pero si nos fijamos en los bordes, vemos que se ajustan bastante bien, que es lo que íbamos buscando con el registro deformable.

En la práctica, se suele considerar que un valor de PSNR mayor de aproximadamente 30 dB indica una calidad de imagen bastante buena y que las diferencias son difíciles de percibir para la mayoría de las personas. Valores más altos, como 40 dB o más, generalmente se asocian con una calidad excelente y diferencias prácticamente imperceptibles. Por tanto podríamos decir que tras realizar el registro rígido, para la mayoría de las imágenes obtendríamos una calidad bastante buena y no sería necesario aplicar el registro deformable. Pero teniendo en cuenta que gran parte de la imagen la hacemos nula al aplicar la máscara y la PSNR se obtiene haciendo una media con el número total de píxeles, el resultado que obtenemos no es del todo fiable.

Vamos ahora a realizar otro análisis pero reduciendo la imagen resultante de la resta solo al entorno donde se encuentran la etiqueta. Como era de esperar, la PSNR ha bajado, al ser el error más representativo para la cantidad de píxeles totales. En la Figura 4.3 vemos que, para el caso 1 y 3, seguimos obteniendo los valores más altos, entorno a 35 dB tras el registro deformable, lo cual es considerablemente bueno. Por otro lado, para el segundo caso, como hemos dicho antes, la PSNR resulta más baja debido al reflejo, y ahora, sin aplicar el registro deformable no llegaríamos a ese mínimo subjetivo de calidad de 30 dB.

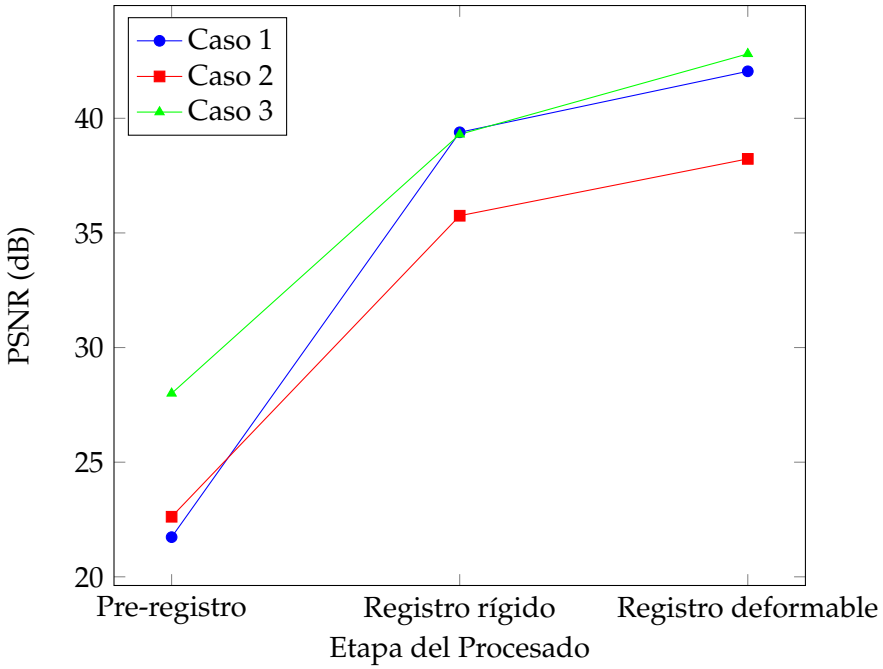


Figura 4.2: PSNR en función de la etapa del procesado.

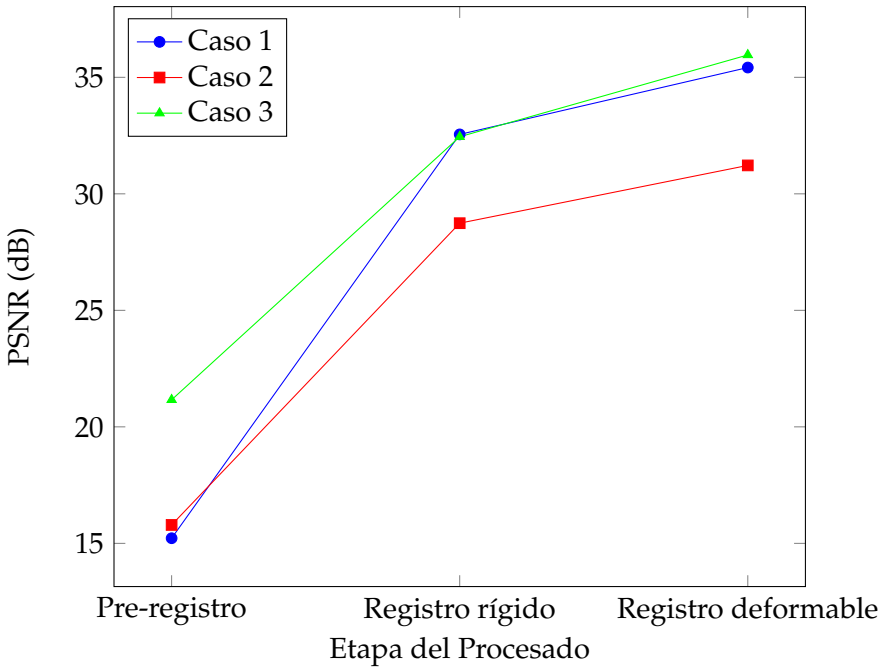


Figura 4.3: PSNR en función de la etapa del procesado recortando las imágenes a la etiqueta.

CAPÍTULO 5

Conclusiones

Esta investigación ha abordado el diseño de una solución para el control de calidad del etiquetado en las botellas de Licor 43, producto de Grupo Zamora, en colaboración con Biyectiva Technology. Esta solución hace uso de técnicas tradicionales de procesado de imagen, como son la morfología matemática o el registro de imágenes, tanto afín como deformable, con el propósito de alinear eficazmente las imágenes de producción con las de referencia y así poder compararlas.

Los resultados obtenidos, claramente evidenciados por mejoras sustanciales en la relación señal a ruido de pico (PSNR), subrayan la eficacia y versatilidad del uso de técnicas de registro, y sobre todo del registro deformable, debido a su capacidad de dar un resultado excelente ante imágenes en diversas condiciones, como variaciones en la iluminación y la presencia de reflejos. Al examinar detalladamente la región de interés, que en nuestro caso es la etiqueta superior del producto, se ha confirmado que, a pesar de la disminución en el PSNR, los valores obtenidos después del registro deformable mantienen niveles aceptables.

Cabe destacar que esta solución se ha diseñado de manera específica para abordar los desafíos únicos asociados con el control de calidad del etiquetado de las botellas de Licor 43. Debido a esto, se han considerado parámetros específicos en el desarrollo. En morfología, como el tamaño de los elementos estructurantes o la operaciones utilizadas; y en los registros, tanto en el afín, al definir la forma y parámetros para encontrar características en la etiqueta, como en el deformable, al elegir los parámetros adecuados para el modelo utilizado.

En el contexto de futuras investigaciones, se propone realizar análisis más detallados que involucren etiquetas con rasgaduras reales, obtenidas directamente de entornos de producción. Este enfoque específico busca incorporar la variabilidad y complejidad inherentes a las condiciones industriales reales, permitiendo una evaluación más precisa del rendimiento del sistema propuesto. La utilización de etiquetas con rasgaduras reales facilitará la identificación de patrones y características específicas asociadas a estos desperfectos, lo que contribuirá a definir un umbral de rechazo. La implementación de este umbral de rechazo será esencial para la toma de decisiones automatizada en el proceso de control de calidad, mejorando la eficiencia del sistema. Esta estrategia de mejora continua es necesaria para la implementación de soluciones en entornos industriales, como es nuestro caso, garantizando que se mantenga actualizada y eficiente.

Bibliografía

[1] Licor43. https://www.licor43.com/.

[2] Grupo Zamora. https://www.zamoracompany.com/.

[3] Biyectiva. https://www.biyectiva.com/.

[4] Programa Encuentro TF, curso 2023/2024. www.campusmarenostrum.es.

[5] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th International Conference on Machine Learning*, pages 129–136, 2007.

[6] Barbara Zitová and Jan Flusser. Image registration methods: a survey. *Image and Vision Computing*, 21(11):977–1000, 2003.

[7] David G Lowe. Object recognition from local scale-invariant features. In *Proceedings of the seventh IEEE International Conference on Computer Vision*, volume 2, pages 1150–1157, 1999.

[8] David G Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60:91–110, 2004.

[9] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (SURF). *Computer Vision and Image Understanding*, 110(3):346–359, 2008.

[10] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. ORB: An efficient alterna-

tive to SIFT or SURF. In *2011 International Conference on Computer Vision*, pages 2564–2571, 2011.

[11] Chris Harris and Mike Stephens. A combined corner and edge detector. plessey research roke manor. *United Kingdom*, 1988.

[12] Rafael C. González and Richard E. Woods. *Digital Image Processing*. 3rd edition, 2008.

[13] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. In *Computer Vision–ECCV 2006: 9th European Conference on Computer Vision*, pages 430–443, 2006.

[14] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. Brief: Binary robust independent elementary features. In *Computer Vision–ECCV 2010: 11th European Conference on Computer Vision*, pages 778–792, 2010.

[15] Jorge Larrey-Ruiz, Rafael Verdú-Monedero, and Juan Morales-Sánchez. A Fourier domain framework for variational image registration. *Journal of Mathematical Imaging and Vision*, 32(1):57–72, 2008.

[16] Rafael VerdÃº-Monedero, Jorge Larrey-Ruiz, and Juan Morales-Sanchez. Frequency implementation of the Euler-Lagrange equations for variational image registration. *IEEE Signal Processing Letters*, 15:321–324, 2008.

[17] Jan Modersitzki. *Numerical methods for image registration*. 2003.

[18] Jorge Larrey-Ruiz, Juan Morales-Sánchez, and Rafael Verdú-Monedero. Generalized regularization term for non-parametric multimodal image registration. *Signal processing*, 87(11):2837–2842, 2007.

[19] Rafael Verdu-Monedero, Jorge Larrey-Ruiz, Juan Morales-Sánchez, and José Luis Sancho-Gómez. Fractional regularization term for variational image registration. *Mathematical Problems in Engineering*, 2009, 2009.

[20] Pierre Soille. *Morphological Image Analysis: Principles and Applications*. Springer Science & Business Media, 2013.

[21] Frederik Maes, Dirk Vandermeulen, and Paul Suetens. Medical image registration using mutual information. *Proceedings of the IEEE*, 91(10):1699–1722, 2003.

[22] Alexis Roche, Grégoire Malandain, Xavier Pennec, and Nicholas Ayache. The correlation

ratio as a new similarity measure for multimodal image registration. In *Medical Image Computing and Computer-Assisted Intervention – MICCAI: First International Conference*, pages 1115–1124, 1998.

[23] Alexis Roche, Grégoire Malandain, Nicholas Ayache, and Xavier Pennec. *Multimodal image registration by maximization of the correlation ratio*. PhD thesis, INRIA, 1998.