ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA MECÁNICA

“ANÁLISIS DE FALLA EN ELEMENTOS DE ACERO SOMETIDOS A CARGAS CÍCLICAS MEDIANTE PROCESAMIENTO DE IMÁGENES EN SOFTWARE LIBRE”

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO MECÁNICO

ARIEL VINICIO DÍAZ SAAVEDRA
ariel.diaz@epn.edu.ec

ERICK GABRIEL POVEDA LLERENA
erick.poveda@epn.edu.ec

DIRECTOR: ING. CARLOS WIME DÍAZ, M.Sc.
carlos.diaz@epn.edu.ec

CODIRECTOR: ING. VICTOR HUGO HIDALGO, D.Sc.
victor.hidalgo@epn.edu.ec

QUITO, MARZO 2022

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por los señores **ARIEL VINICIO DÍAZ SAAVEDRA** y **ERICK GABRIEL POVEDA LLERENA**, bajo mi supervisión.

_____	_____
Ing. Carlos Wime Díaz, M.Sc.	Ing. Víctor Hugo Hidalgo, D.Sc.
DIRECTOR DEL PROYECTO	**CODIRECTOR DEL PROYECTO**

DECLARACIÓN

Nosotros, **ARIEL VINICIO DIAZ SAAVEDRA Y ERICK GABRIEL POVEDA LLERENA**, declaramos bajo juramento que el trabajo aquí descrito es de nuestra autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que hemos consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedemos nuestros derechos de propiedad intelectual correspondiente a este trabajo, a la Escuela Politécnica Nacional, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normativa institucional vigente.

_____	_____
Ariel Vinicio Díaz Saavedra	Erick Gabriel Poveda Llerena

DEDICATORIA

Esta tesis está dedicada:

A mi madre Cristina quien con su amor, ternura y valentía me ha permitido hoy llegar a cumplir una meta más. Por enseñarme que sin importar lo fuerte que nos golpee la vida, es importante seguir adelante y enfrentarse a todo sin temer. A la memoria de mi padre Roberto.

A mi hermanita Camila que con amor y paciencia siempre me ha apoyado y enseñado a ser una mejor persona. Mi abuelita Susana quien con su cariño incondicional me alentó en cada paso y nuevo reto que he enfrentado.

Mi tía Crucita y mi tío Marco por haberme brindado su apoyo y cariño sin importar las circunstancias. Por tantas risas y buenos momentos junto a ellos. A mis primos Christian y Julián por siempre estar a mi lado y crecer conmigo. A toda mi familia por su ayuda y consejos. A mi novia Brenda por haberme brindando su amor y paciencia en este largo viaje.

Ariel Díaz

DEDICATORIA

La siguiente obra va dedicada a dos de las personas más importantes en mi vida: a mi abuelita y a mi madre, las cuales fueron pilares fundamentales para que yo me realizara como persona. Aprendiendo día a día de sus consejos y de sus regaños que con mucho cariño lo hacían para que siguiera el camino que comencé y que debía terminar.

A mi tío Carlos, que con sus historias y su apoyo desde muy temprana edad me oriento a escoger la carrera a la cual le tengo mucho amor.

A mi padre que me enseño que hay que luchar, que nada viene fácil, y aunque no esté en cada momento, siempre, me estará sosteniendo.

A mi hermana que siempre estuvo en los momentos más complicados y siempre tuvo un abrazo reconfortante.

Erick Poveda

AGRADECIMIENTO

Mi profundo agradecimiento a la Escuela Politécnica Nacional y a la gloriosa Facultad de Ingeniería Mecánica, por el apoyo incondicional brindado a la comunidad estudiantil y por haberme formado como una persona pulcra y honesta.

De igual manera, agradezco a Carlos Díaz y Víctor Hidalgo por habernos apoyado a lo largo de nuestra formación educativa. Además, por sus valiosos consejos que nos han ayudado a convertirnos en profesionales y personas de calidad.

A mi madre, a mi hermana y a toda mi familia por su apoyo y consejos.

A mi amigo Erick por tantas noches de desvelo y horas de esfuerzo en la realización de este trabajo. Por su paciencia y perseverancia.

Finalmente, a mi prima Sol por sus buenos consejos.

Ariel Díaz

AGRADECIMIENTO

Agradezco los consejos emitidos por mis padres y tíos quienes lograron enforcarme a seguir la gran carrera de Ingeniería Mecánica y más aún por haberme dirigido a una de las mejores instituciones académicas del país que sin saber se convertiría en mi alma máter.

Agradezco a la ciudad que me recibió y me permitió tener las más grandes experiencias en mi vida. La ciudad que me encanto con sus maravillosos paisajes, historia y en donde se encuentra la institución de la cual estaré eternamente feliz de haber pertenecido la poderosa Escuela Politécnica Nacional, que a más de ser la institución de la cual aprendí los caminos de la ciencia, hoy puedo decir que se convirtió en un hogar más en donde conocí a personas que se volvieron amigos en este camino para convertirme no solo en un ingeniero sino en una mejor persona. Entre aquellos amigos, hermanos debo mencionar a mi compañero del presente proyecto Ariel, siempre estaré agradecido por la paciencia, el apoyo para continuar escribiendo cada palabra de la presente investigación.

Uno de los momentos que siempre llevare en mi mente fue aquel instante cuando llegué a la Facultad de los "guayperos", a la Facultad de Ingeniería Mecánica. En ese momento me encontré como un pequeño que entraba por primera vez a la escuela, pero tenía grandes sueños y aspiraciones.

Viviré agradecido con cada uno de los docentes que con su trabajo permanente lograron despertar en mi la curiosidad por siempre saber más para superarme a mí mismo. Llevare siempre en el corazón los recuerdos de aquellas clases de Ciencia de Materiales en donde el Ing. Carlos Díaz dictaba clases magistrales acerca del comportamiento de los aceros. Fueron sucesos que marcaron mi camino y lo seguirán haciendo pues ahora sé en qué se enfocarán mis próximos estudios.

Sin más que decir agradezco a la vida por ponerme en una de las mejores universidades del país y en la mejor Facultad del mundo. Por última vez quiero elevar mi grito para decir: "Tres Rases, un Chispum y un Carajo por Mecánica".

Erick Poveda

ÍNDICE

RESUMEN

El presente trabajo de investigación tiene por objetivo el análisis de la superficie de fractura de elementos mecánicos de acero sometidos a cargas cíclicas durante altos ciclos de servicio, es decir, fallas por fatiga. Mediante el desarrollo de un software para procesamiento de imágenes en Python. En el cual se presentan diversas herramientas para poder identificar, resaltar y añadir información con respecto a la apariencia superficial de la fractura. El software implementado se divide en dos módulos principales. Iniciando por el preprocesamiento en donde se modifica a la imagen hasta que posea las características visuales necesarias para su correcto análisis. Seguido, se realiza el procesamiento en el cual se muestran funciones para detección y marcación de parámetros superficiales característicos de las fallas por fatiga. Finalmente, se ejecuta un cuestionario final. Este permite añadir información adicional sobre el proceso de falla, mediante preguntas puntuales y la comparación visual de los resultados obtenidos en los dos módulos anteriores con imágenes adicionales que presentan marcas específicas de distintas condiciones de funcionamiento. Los resultados corresponden a los criterios identificados a partir de la información visual del software y el cuestionario. Estos criterios deben ser interpretados para poder establecer un diagnóstico, en el cual a partir de la información obtenida del software se pueda identificar una posible causa de falla.

Palabras clave: Bordes, Falla, Fatiga, Imagen, Procesamiento, Python.

ABSTRACT

The objective of this research work is to analyze the fracture surface of mechanical steel elements subjected to cyclic loads during high service cycles, that is, fatigue failures. By developing software for image processing in Python. In which various tools are presented to be able to identify, highlight and add information regarding the superficial appearance of the fracture. The implemented software is divided into two main modules. Starting with the preprocessing where the image is modified until it has the necessary visual characteristics for its correct analysis. Next, the processing is carried out in which functions are presented for the detection and marking of surface parameters characteristic of fatigue failures. Finally, a final questionnaire is presented. This allows adding additional information about the failure process, through specific questions and the visual comparison of the results obtained in the two previous modules with additional images that present specific marks of operating conditions. The results correspond to the criteria identified from the visual information of the software and the questionnaire. These criteria must be interpreted to establish a diagnosis, in which, based on the information obtained from the software, a possible cause of failure can be identified.

Keywords: Edges, Failure, Fatigue, Processing, Python.

“ANÁLISIS DE FALLA EN ELEMENTOS DE ACERO SOMETIDOS A CARGAS CÍCLICAS MEDIANTE PROCESAMIENTO DE IMÁGENES EN SOFTWARE LIBRE”

INTRODUCCIÓN

Desde los cimientos de la ingeniería, una de las principales dudas siempre ha sido poder identificar acertadamente el tiempo de vida y las condiciones de servicio que puede soportar un elemento antes de fallar. A lo largo de los años se ha determinado que parte de la ingeniería debe enfocarse en prevenir daños y evitar que se repitan constantemente. La mayoría de las fallas estructurales han ocurrido a lo largo del siglo XX. Estos eventos desafortunados han ocasionado una revolución en: la inspección, técnicas, prácticas, procesos de almacenamiento y de fabricación que han modificado considerablemente los criterios de falla (Campbell, 2012).

Entre la gran cantidad de fallas que existen, hay un tipo que se diferencia de las demás, ya que no presenta ninguna señal o advertencia antes de que suceda. Las fallas por fatiga son ocasionadas debido a la aplicación de cargas cíclicas durante la vida de servicio de un elemento. Son consideradas de alto riesgo, debido a que se producen de manera súbita, a diferencia de otras que muestran comportamientos característicos previos a la falla. El estudio superficial de un elemento que ha sido sometido a este tipo de cargas permite identificar criterios adicionales con respecto al proceso de falla. Lo que hace posible identificar condiciones inadecuadas tanto en el diseño y operación, para evitar que se repitan.

Para que las marcas características de este tipo de falla puedan ser claramente identificadas es necesario ciertas condiciones. Bajas cargas cíclicas permiten asegurar una larga vida de servicio, lo que ocasiona una marcación de rasgos superficiales definida. El análisis de estos parámetros permite identificar condiciones de servicio fundamentales en un análisis de falla. De manera que es posible reducir el campo de análisis en un estudio más específico para identificar posibles causas de falla.

La utilización de un software, como herramienta auxiliar, es una técnica vanguardista que permite proporcionar información de una forma más eficaz. De manera que las limitaciones humanas pueden ser superadas generando resultados confiables. Actualmente, todas las áreas de la ingeniería poseen aplicaciones que sirven de soporte en diversos procesos productivos. En este caso en particular, el procesamiento de imágenes mediante software libre representa una alternativa poderosa. La posibilidad

de alterar las variables presentes en una imagen permite afinar resultados en función de la capacidad visual del usuario.

Pregunta de Investigación

¿Es posible ejecutar un análisis de falla mediante la utilización de un software para procesamiento de imagen, como herramienta auxiliar?

Objetivo general

Analizar la falla en elementos de acero sometidos a elevadas cargas cíclicas y bajos esfuerzos mediante análisis fotográfico y procesamiento de imágenes en software libre.

Objetivos específicos

- Realizar una revisión del estado del arte sobre la inspección fractográfica mediante procesamiento de imágenes.
- Determinar la metodología más apropiada para el procesamiento de imágenes aplicado a falla.
- Desarrollar un código de programación basado en la metodología seleccionada.
- Evaluar el código generado con diferentes muestras fracturadas y analizar el grado de similitud para efectos de validación.

1. MARCO TEÓRICO

1.1 Análisis de falla

El término “Falla” es un concepto ligado a la confiabilidad de cualquier tipo de estructura. Se define como la culminación de la capacidad que tiene un determinado elemento o producto para efectuar sus funciones según lo establecido (IEC, 1990)(Rausand & Ølen, 1996). Dado el caso en el que se produzca, puede generar grandes pérdidas humanas y económicas. A partir de esto se puede considerar que su estudio es un proceso complejo, debido a que se busca identificar soluciones e incluso generar nuevas alternativas que permitan asegurar la obtención de elementos, componentes y equipos más confiables (Díaz Cortés et al., 2018).

1.1.1 Modos y efectos de falla

Es importante, considerar términos que permitan abarcar la mayor cantidad de información con respecto a una falla. El análisis de modos y efectos de falla (FMEA) es una de las metodologías más importantes para poder entender y comprender el origen de su formación. Los distintos modos y efectos de falla pueden ser clasificados, según la pérdida su función:

1. **Pérdida total de la función:** Se considera un elemento que no puede volver a ejecutar su función original. Por ejemplo, un vehículo que presente un eje fracturado prácticamente se vuelve obsoleto. No es posible volver a retomar el funcionamiento normal del automotor (Tovar S., 1999).
2. **Pérdida parcial de la función:** Cuando un sistema posee elementos que, si bien pueden operar bajo las condiciones de servicio, no cumplen con la función principal para la cual fueron creados. Por ejemplo, cuando un vehículo presenta pérdida de tolerancia entre el cilindro y los anillos del pistón en un motor de combustión interna, se genera fricción y corrosión. El automotor puede operar, pero no cumple adecuadamente su función principal, ya que se generan pérdidas de potencia y un elevado consumo de recursos (combustible y aceite) (Tovar S., 1999).
3. **Función errónea:** El daño en un determinado elemento ha ocasionado una condición de inseguridad, la cual pone en duda su confiabilidad para ejecutar sus funciones. Por ejemplo, el caso de un automotor que presente un sistema de frenos desgastado, su funcionamiento se ve comprometido, ya que es necesario una pronta revisión o reparación (Tovar S., 1999).

La Figura 1.1 muestra una clasificación de los modos y efectos de falla desde un enfoque general. En este caso se considera la frecuencia e impacto que poseen, de manera que pueden ser orientados a un resultado catastrófico o leve. Tomando en consideración la naturaleza del proyecto de titulación, se prioriza el análisis de los elementos que han perdido totalmente su función y se producen de manera repentina, es decir, fallas catastróficas.

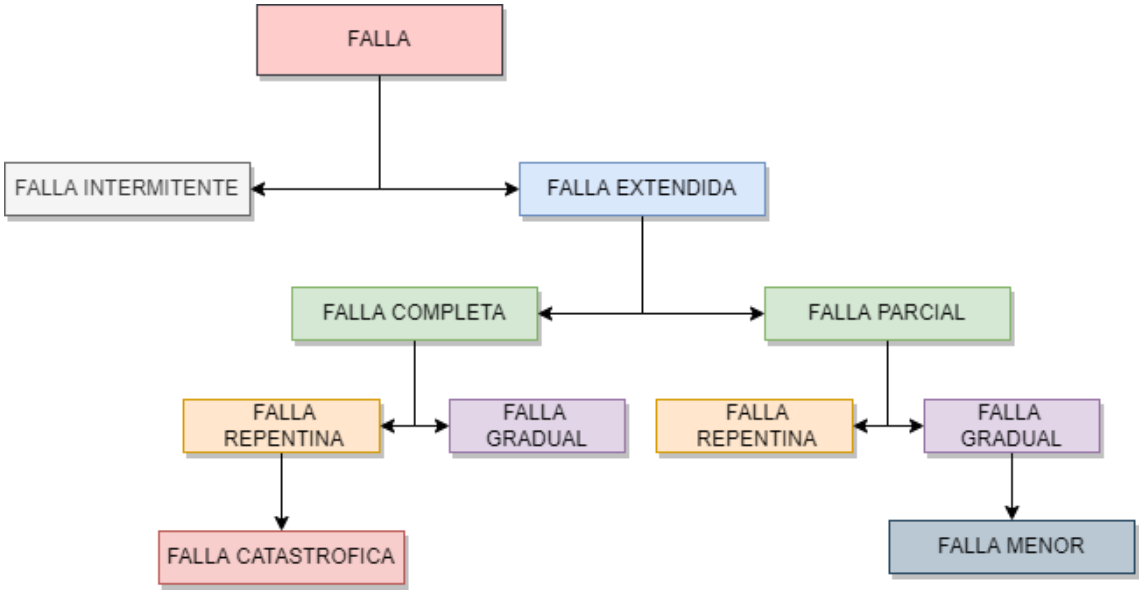


Figura 1.1. Clasificación de fallas.
(Rausand & Ølen, 1996)

1.1.2 Etapas del análisis de falla

La ejecución del análisis de los distintos modos y efectos de falla representa un proceso complejo, pero necesario en varios ámbitos de la industria. Desde un enfoque general, a continuación, se presentan las etapas principales para ejecutar este análisis, que serán más o menos minuciosas en función del caso de estudio.

- **Revisión Documental:** la falla de un elemento es el resultado de diferentes condiciones. Para poder identificar claramente el motivo o causa que originó este fenómeno, es necesario generar una base de información sólida previa a la falla. Al establecer una investigación minuciosa de los antecedentes de un elemento es posible encontrar certificados, informes de pruebas efectuadas, prototipos de diseño, planos, entre otros. Este aspecto puede ser considerado como el más sensible, ya que se establece el historial previo a la falla.(Díaz Cortés et al., 2018)
- **Condiciones de operación y servicio:** las condiciones previas a la falla son tan importantes como las condiciones de servicio que originaron la misma. En esta

se efectúa la identificación de parámetros de operación a los cuales estaba siendo sometido determinado elemento. Los más comunes son: presión, temperatura, velocidad, procedimientos de mantenimiento previos, condiciones del medio (ambientales), etc. Además, se puede incluir testimonios de los principales actores que estuvieron presentes durante la falla (Díaz Cortés et al., 2018).

- **Ensayos y pruebas:** el empleo de técnicas estandarizadas permite generar información adicional, la cual no es posible obtener a simple vista con una inspección visual. Generalmente, se llevan a cabo ensayos no destructivos y mecánicos. La aplicación de los ensayos no destructivos (END) permite analizar un elemento de manera que su integridad no se vea afectada ni deteriorada. Los ensayos mecánicos son efectuados en probetas bajo las condiciones de servicio similares previamente identificadas. Es importante someter al elemento a un análisis tanto microscópico como macroscópico para identificar condiciones superficiales específicas. La utilización de herramientas auxiliares o software es de vital importancia en esta etapa, ya que permite obtener una mejor apreciación de las marcas características presentes en los objetos de estudio. Otorga mayor exactitud y sencillez al proceso (Díaz Cortés et al., 2018).
- **Conclusiones y recomendaciones:** después de haber recolectado la información previa a la falla, efectuar las pruebas y ensayos correspondientes y analizar los datos obtenidos, es necesario realizar un informe. Este documento presenta los aspectos más importantes de cada etapa del proceso. Se identifican estándares de comparación, los cuales permiten encaminar la investigación hacia una causa de falla.

Otro aspecto importante de esta etapa es la formulación de recomendaciones. El objetivo principal del análisis de falla es poder evitar que dichos eventos vuelvan a ocurrir. Las recomendaciones muestran una síntesis de toda la información obtenida, que busca proponer alternativas y temperaturas de funcionamiento, diseño, condiciones de servicio, magnitudes de carga, entre otras. Para evitar que en un futuro cercano se vuelvan a cometer los mismos errores y las consecuencias sean aún más severas (Díaz Cortés et al., 2018).

1.1.3 Tipos de fallas

Al hablar de un análisis de falla se puede establecer una semejanza con un “sistema forense”. Partiendo de un elemento que ha fallado se puede obtener criterios de vital

importancia para encontrar la relación existente entre el diseño, fabricación e historial de operación con el fin de su vida útil (Díaz Cortés et al., 2018). Para poder estudiar de una manera más detallada, los distintos tipos de falla han sido divididos de la siguiente forma:

1.1.3.1 Falla por diseño

En la década de los 50, con la fabricación de los primeros aviones comerciales (Jets) Figura 1.2, se produjeron una serie de fatídicos eventos ocasionados por fallas durante el vuelo de varias aeronaves. La principal causa de estos accidentes fue la presencia de altos esfuerzos producidos en las ventanas rectangulares de los aviones, debido a su diseño con aristas agudas. La presencia de estos esfuerzos ayuda a la proliferación de grietas (Díaz Cortés et al., 2018).

Figura 1.2. Fabricación de aviones comerciales en los años 50.
(Cash, 1950)

1.1.3.2 Falla por selección errónea de materiales

Para este caso se puede considerar: la mala selección del material en función de la aplicación y la presencia de un material defectuoso. Los materiales empleados en la fabricación deben estar libres de impurezas y defectos (Inclusiones, porosidades, entre otras) para poder asegurar la calidad del producto final. El empleo de materiales inadecuados puede ocasionar desastres. Tal es el caso de los aviones de combate F-111 de la Fuerza Aérea Americana Figura 1.3. Las uniones de las alas al fuselaje eran demasiado frágiles, esto produjo una gran cantidad de accidentes a lo largo de la década de los sesenta (Díaz Cortés et al., 2018).

Figura 1.3. Modelo F-111 Aardvark.
(General Dinamycs, 2021)

1.1.3.3 Falla por factor humano

Cuando un elemento es sometido a condiciones de servicio inadecuadas o no se respetan los protocolos establecidos, pueden ocasionarse fallos que ponen en riesgo la integridad del equipo y del personal. Generalmente, este tipo de eventos se producen cuando los operadores desactivan los sistemas de seguridad para poder trabajar bajo condiciones no estipuladas en la programación. Tal es el caso de la plataforma petrolífera Piper Alpha, la cual debido a fallas humanas genero un desastre de grandes proporciones Figura 1.4 (Díaz Cortés et al., 2018).

Figura 1.4. Desastre en la plataforma petrolífera Piper Alpha en 1988.
(Macleod Fiona & Richardson Stephen, 2018)

1.1.3.4 Falla por tratamiento térmico inadecuado

Al aplicar un tratamiento térmico (TT) es importante considerar que, durante el proceso se pueden generar microgrietas y fracturas, estos defectos pueden aparecer inmediatamente o después de un periodo corto de servicio. En algunos casos estos procedimientos ocasionan una microestructura no uniforme en uniones soldadas, tal como en los aviones de combate F-111 mencionados anteriormente (Díaz Cortés et al., 2018).

1.1.3.5 Falla por fabricación

Los elementos obtenidos por procesos de fundición, por lo general, presentan en su estructura porosidades como se muestra en la Figura 1.5. La aglomeración de porosidades en zonas de elevados esfuerzos puede producir concentradores, lo que ayuda a la formación de grietas que ocasionan la fractura del elemento. De manera similar, la presencia de escoria, inclusiones y agrietamiento en soldaduras puede producir la rotura de la unión (Díaz Cortés et al., 2018).

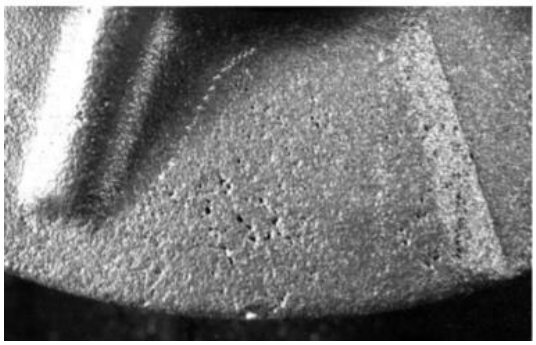


Figura 1.5. Porosidad subsuperficial en el soporte de eje de acero de un carro minero.
(D.C. Williams & R.L. Naro, 2019)

1.1.3.6 Falla por montaje erróneo

Generalmente, cuando se habla de errores de montaje se hace referencia al ensamblaje original de un equipo y procesos de mantenimiento inadecuados. Es muy común generar un desalineamiento de las partes móviles, montajes ambiguos y extracción accidental de elementos auxiliares, cuando se llevan a cabo labores de mantenimiento como se puede observar en la Figura 1.6. En algunos casos las fallas se deben a una mala preparación de la zona en donde el equipo va a ser instalado como, por ejemplo: superficies irregulares, conexiones eléctricas defectuosas, entre otras (Tovar S., 1999).

Figura 1.6. Montaje incorrecto de la válvula del motor de un automóvil.
(Rheinmentall, 2021)

1.1.3.7 Falla por fatiga

La presencia de esfuerzos cíclicos en el material permite la formación, de grietas, ver la Figura 1.7. Este tipo de fallas ocurren con la aplicación de cargas menores a las necesarias para ocasionar deformación en el material a un elevado número de cíclicos de servicio (Díaz Cortés et al., 2018). La fatiga es responsable aproximadamente del 80% (Posiblemente el 90%) de todas las fallas mecánicas (ASM International. Handbook, 1998). El presente proyecto de titulación pretende enfocarse en este tipo de fallas debido a su elevada presencia en la industria. Más adelante se presenta una revisión exhaustiva.

Figura 1.7. Falla por fatiga de un eje AISI 8620.
(McCall & French, 1977)

1.1.3.8 Falla por desgaste

El desgaste puede ser definido como el daño a una superficie sólida, debido a la pérdida de material ocasionada por las condiciones de servicio a las cuales está sometido, como se puede observar en la Figura 1.8. Es un proceso que depende de una gran cantidad de factores como, por ejemplo: dureza, lubricación, tiempo de servicio, magnitud de las cargas, entre otras. Además, puede ser considerado como abrasivo por la pérdida progresiva del material y adhesivo con deformación superficial (Díaz Cortés et al., 2018).

Figura 1.8. Desgaste de la superficie del diente de un engrane.
(Miró, 2021)

1.1.3.9 Falla por corrosión

La corrosión es el deterioro gradual de un material, debido a una reacción química con el medio ambiente, ver la Figura 1.9. La presencia de condiciones ambientales agresivas, bacterias, entre otros, son factores que proliferan la formación de este fenómeno (Díaz Cortés et al., 2018). Para evitar la corrosión es necesario la utilización de protecciones o recubrimiento dependiendo del caso.

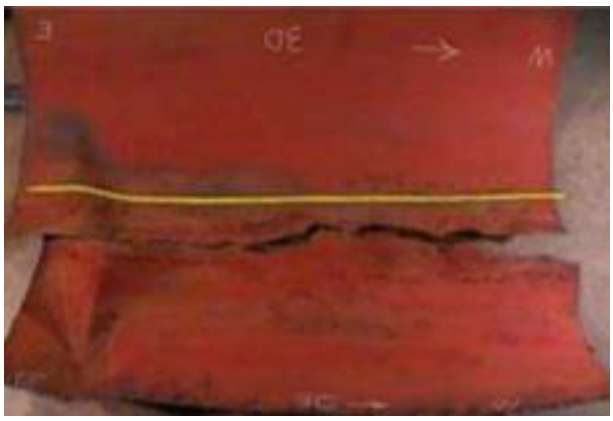


Figura 1.9. Rotura de una tubería de transmisión de gas debido a un proceso corrosivo.
(Rumiche & Idacochea, n.d.)

1.2 Fallas por fatiga

En un análisis de elementos que han fallado por cargas cíclicas, se puede identificar que los esfuerzos máximos eran menores que la resistencia última del material y en algunos casos menor que la resistencia a la fluencia. A partir de estas consideraciones se establece el concepto de falla por fatiga. La principal característica de este tipo de fallas es que los esfuerzos son bajos y se aplican durante un elevado número de veces, es decir, debido a la aplicación de cargas cíclicas. Esto ocasiona una condición acumulativa de estrés en los elementos (Budynass G., 2014).

Generalmente, los elementos que fallan debido a cargas estáticas presentan una deformación elevada, porque el esfuerzo sobrepasó el límite elástico del material. La detección temprana de este evento permite reemplazar dicho elemento antes de que suceda la fractura. En el caso de una falla por fatiga no se puede observar ningún tipo de advertencia en el elemento. Es repentina y devastadora (Budynass G., 2014).

A raíz del peligro que este tipo de fallas representan en cualquier ámbito de la industria, el presente proyecto analiza elementos que han fallado debido a la presencia de cargas cíclicas durante elevados ciclos de servicio, es decir, fallas por fatiga.

1.2.1 Aceros

El acero es un material que a través de los años ha contado con múltiples transformaciones. Su composición ha sido modificada para obtener diferentes propiedades. Principalmente , constan de hierro y carbono (Maldonado, 1996).

Los aceros, llamados al simple carbono, contienen cantidades pequeñas de otros elementos como, por ejemplo: Mn, Si, S y P. Por otra parte, los aceros aleados son aquellos que contienen cantidades o porcentajes específicos de: Ni, Cr, Mb, Vn y Tg. Todos estos elementos de aleación son empleados para proporcionar distintas propiedades al acero.

El carbono es un elemento indispensable en la composición del acero, de tal forma que, en función de su porcentaje dentro de la composición química puede tener la siguiente clasificación:

- Acero de bajo carbono
- Acero de medio carbono
- Acero de alto carbono
- Acero de herramientas

En la Figura 1.10 se observa el diagrama de equilibrio Fe-C en donde se puede identificar la solubilidad del carbono en el hierro y las estructuras que se forman según el porcentaje en peso de carbono disuelto en hierro. Los aceros de bajo carbono contienen cantidades de carbono menores al 0.3% en peso. Los aceros de medio contenido de carbono poseen del 0.3 al 0.6 %. Los aceros que contienen por encima de 0.6 % se denominan aceros de alto contenido de carbono y aquellos que tienen un porcentaje por encima de 0.77% se denominan aceros de herramientas. El acero con porcentajes de entre 1.3 a 2% no es muy común, sin embargo, las composiciones que son mayores al 2% de carbono son conocidas comúnmente como fundiciones. De esta

forma para el presente proyecto se restringe el campo de aplicación a elementos de acero de bajo contenido de carbono.

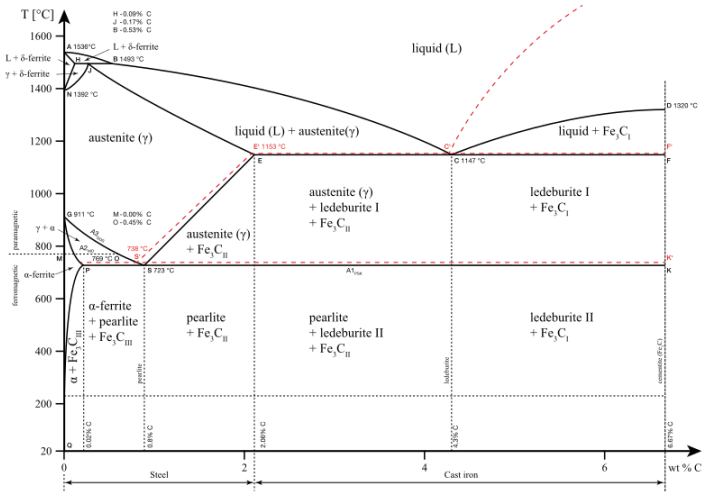


Figura 1.10. Diagrama de equilibrio Fe-C
(Rumiche & Idacochea, n.d.)

1.2.2 Cargas cíclicas

Las cargas cíclicas son aquellas que varían en función del tiempo, debido a esto representan un riesgo potencial en cualquier aplicación. Generalmente, cuando se diseñan elementos se considera las cargas estáticas para la realización de pruebas y ensayos. Si bien, estas se aproximan bastante a las condiciones de funcionamiento real de los elementos, se deja de lado los esfuerzos cíclicos (Budynass G., 2014).

Este proyecto de titulación se enfoca en las fallas producidas por la fatiga, debido a esto el análisis se reduce a elementos que han sido sometidos a cargas cíclicas. Estas cargas son las responsables de generar las marcas características que se describen más adelante y que servirán de base para la obtención de criterios en el análisis de falla.

1.2.3 Altos ciclos y bajos esfuerzos de servicio

Una vez que han sido definidas las cargas a considerarse en este proyecto, es necesario establecer adecuadamente el rango de vida. No se ha establecido de manera formal categorías para las fallas por fatiga, pero generalmente se utilizan tres categorías. La Figura 1.11 muestra los rangos de falla por fatiga, es importante mencionar que no se indica la fatiga de muy bajos ciclos, debido a que se produce en un tiempo muy corto. A continuación, se describe cada una:

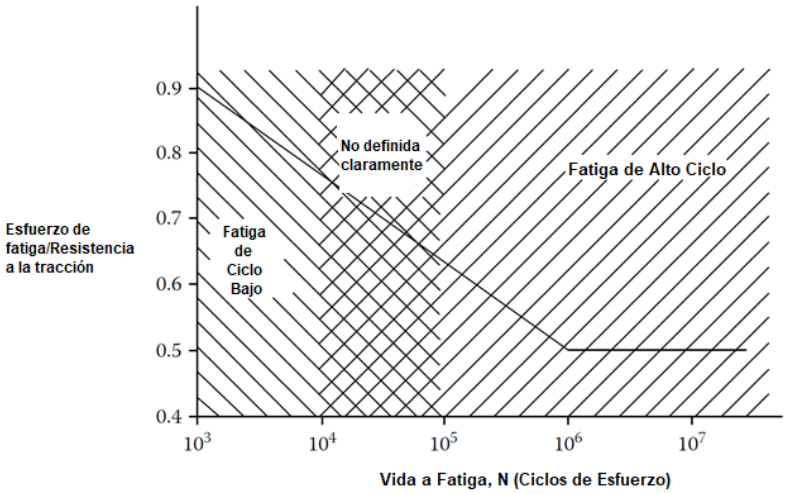


Figura 1.11. Gráfico S vs N que presenta los rangos de fallas por fatiga de ciclo alto y bajo (VLCF estaría situado a la izquierda del gráfico)
(Sachs, 2016)

1. Fatiga de altos ciclos: es necesario que el elemento haya estado sometido a más de 10 000 ciclos de esfuerzo. Representan el 90% de las fallas por fatiga. A diferencia de las otras categorías, son las más comunes. Generalmente, los elementos que fallan en esta categoría requieren, millones de ciclos desde que se aplica la fuerza inicial.
2. Fatiga de ciclo bajo: es cuando al elemento le toma menos de 10 000 y más de 25 ciclos de esfuerzo para fallar. No son comunes en equipos industriales. Ocupan esfuerzos más altos que las fallas de altos ciclos.
3. Fatiga de ciclo muy bajo (VLCF): a diferencia de la fatiga de ciclo alto y bajo, presenta deformaciones en materiales dúctiles, esto debido a que toma menos de 10 ciclos de esfuerzo hasta la falla final. Son más comunes que las de ciclo bajo. En la Figura 1.11 la fatiga de ciclo muy bajo se localizaría en el lado izquierdo (Sachs, 2016).

Este proyecto se enfoca en elementos sometidos a fatiga de altos ciclos. Debido al riesgo que presentan en la industria. Los procesos de fatiga ocurren con cargas menores a las necesarias para generar deformación y altos ciclos de servicios. De lo contrario el elemento no puede estar en servicio por demasiado tiempo, ya que la violencia de las cargas evitaría un proceso largo de falla y se transforma en un proceso de fatiga de muy bajo ciclo. Esto no significa que en procesos de fatiga cortos no se generan marcas, pero estas no van a ser de una calidad visual aceptable (Sachs, 2016).

1.3 Fractografía en la ingeniería

En el año 1944, el ingeniero y físico Carl A. Zapffe acuñó el término de fractografía después de haber descubierto un método para poder lograr acercamientos mayores en las superficies irregulares de una fractura. Esta rama de la mecánica permite analizar características propias de un proceso de fractura, de manera que se pueda identificar y relacionar ciertos aspectos de la superficie con las causas y mecanismos básicos de falla (ASM International. Handbook, 1998).

El campo de aplicación de esta ciencia es demasiado extenso, debido a esto es necesario definir varios términos con el objetivo de abarcar temas más específicos de análisis. La microfractografía y macrofractografía se utilizan para poder discernir entre un aumento visual bajo y alto.

1.3.1 Microfractografía

Permite aumentos mayores (100X) de manera que se puede obtener una visualización mucho más detallada de la topografía de la superficie de falla. A inicios del siglo XX los análisis microscópicos de los metales se limitaron únicamente a muestras pulidas. Si bien, era una novedad para la época, no representaba un análisis aplicable a todo tipo de materiales. Zapfee y Clogg efectuaron estudios significativos de las marcas características de las fracturas, mediante la utilización de un microscopio óptico. A partir de los trabajos publicados por los autores la microfractografía empieza a ser aplicada de manera sistemática. La Figura 1.12. muestra ejemplos de los trabajos realizados por Zapfee y Clogg (ASM International. Handbook, 1998).

Figura 1.12. Facturas por clivaje en muestras de impacto (Temperatura Ambiente).
(ASM International. Handbook, 1998)

Los avances tecnológicos permiten obtener distintas herramientas con las cuales los análisis fractográfico pueden ser más precisos. El microscopio electrónico de transición (TEM) ha permitido desarrollar enormemente esta rama de la mecánica, ya que emplea aumentos mayores y con mejor resolución.

En la actualidad se ha desarrollado la microscopía electrónica de barrido (SEM) que permite observar de manera directa la superficie de fractura con mejor resolución y mayor profundidad que la obtenida con el microscópico electrónico de transición. La utilización de ambas técnicas es sumamente importante ya que, no se puede anteponer una por sobre otra, son complementarias (Acuña., n.d.).

1.3.2 Macrofractografía

Son el conjunto de técnicas efectuadas como primer paso en el proceso de análisis de falla de un elemento fracturado. Permite el análisis de las características superficiales a partir de una microscopía óptica de bajos aumentos. En el siglo XVII el físico francés René Antpine Ferchault de Réamur presento importantes hallazgos con respecto a superficies de fractura del acero y el hierro. Este hecho científico marcó el punto de partida para el desarrollo de conocimiento a partir de una inspección visual. Investigaciones posteriores permitieron obtener características particulares en función del análisis topográfico de las superficies. En la Figura 1.13 se muestran imágenes de las investigaciones realizadas por R.A.F de Réamur (ASM International. Handbook, 1998).

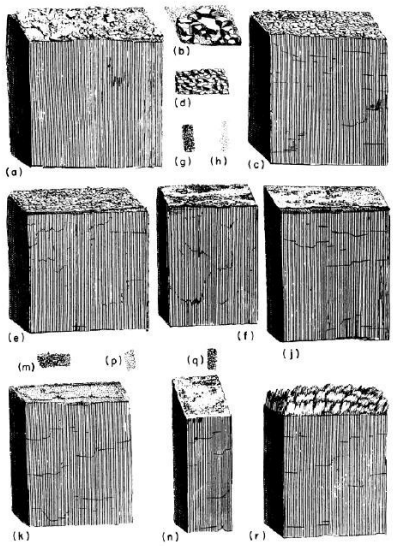


Figura 1.13. Boceto de R.A.F. de Réamour que describe las categorías de fracturas en hierro y acero.
(ASM International. Handbook, 1998)

La fractografía óptica se consolida como el método por excelencia de la macrofractografía. El análisis minucioso de los parámetros sobresalientes identificados en la superficie permite determinar posibles causas de falla (Acuña., n.d.).

1.3.3 Contribución de las técnicas fractográficas en el análisis de falla

El objetivo de la fractografía es determinar las causas de rotura de un componente e identificar las acciones pertinentes para que no se repita, por medio de un rediseño o selección de diferente material. Una de las contribuciones más importantes es poder reconocer el origen de formación y propagación de la grieta de fractura. Esta técnica es considerada como una herramienta auxiliar en un proceso de análisis de falla.

La mayoría de las causas de rotura de elementos se dan por esfuerzo excesivo, sin embargo, hay diversos factores a considerar como: la disminución del área efectiva debido a procesos de fatiga o corrosión. La superficie de falla puede proporcionar los siguientes parámetros:

- Mecanismo de propagación de la fisura.
- Fractotenacidad del material.
- Configuración de las cargas.
- Origen de la fractura.
- Medio químico al cual estuvo expuesto.

Mientras, más parámetros se puedan identificar en la superficie de falla, la efectividad del análisis fractográfico será mayor. Los pasos que se tienen dentro de este proceso son los siguientes:

Tabla 1.1. Proceso de análisis fractográfico

Proceso de análisis fractográfico	
1. Observación visual	Observación de características macroscópicas.
2. Inspección bajo lupa estereoscópica	A partir de una macrografía de bajo aumento, es posible confirmar las observaciones anteriormente realizadas.
3. Microscopía de barrido o de transmisión	Análisis microscópico de la superficie de fractura.

(Fuente: Propia)

1.4 Marcas características de fallas por fatiga

Las marcas fractográficas presentes en una falla por fatiga son: marcas de playa, de progresión o nivel, marcas de trinquete y estrías de fatiga. De estas marcas las dos primeras pueden ser observadas en una macrografía óptica, sin embargo, las estrías son identificables en una micrografía, por lo cual quedan fuera del rango de análisis del proyecto.

1.4.1 Marcas de playa (Beach Marks)

Las marcas de playa son el rasgo macroscópico más prominente e importante de una falla por fatiga. Aparecen alrededor de un punto en común que corresponde a la zona de iniciación de la grieta, como se puede apreciar en la Figura 1.14. Un espaciamiento delgado entre las sucesivas marcas de playa indica un proceso de propagación lento. Al contrario, un espaciamiento ancho indica que el proceso de propagación fue rápido y violento. La ausencia de estas marcas no es un indicador de la ausencia de un proceso de fatiga (Campbell, 2012).

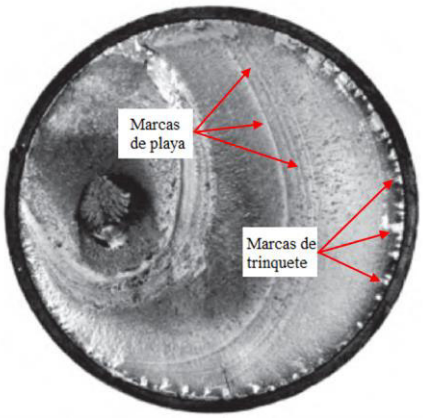


Figura 1.14. Marcas de playa y trinquete en un eje de acero 1050 sometido a flexión rotatoria. (Campbell, 2012)

Las marcas de playa se originan principalmente debido a:

1. **Por deformación plástica microscópica**: la deformación en la punta de avance de la grieta durante los periodos de descanso o cuando el esfuerzo cíclico no es lo suficientemente alto como para hacer progresar la grieta por fatiga como se puede apreciar en la Figura 1.15. Durante el periodo de reposo esta región se va endureciendo debido al trabajo en frío que se realiza a lo largo de la punta de la grieta. Luego, debido al esfuerzo por fatiga, esta zona actúa

como una barrera temporal, sin embargo, la misma se rompe y se origina el proceso normal de fatiga en donde se produce la formación de estrías continuas.

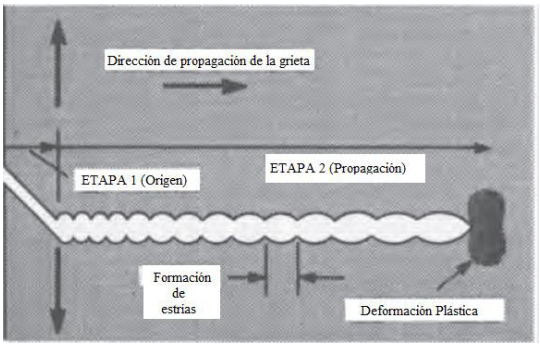


Figura 1.15. Bosquejo de la sección trasversal ampliado de la etapa 1 y 2 de una falla por fatiga
(Campbell, 2012)

2. **Diferencias en el tiempo de corrosión en la propagación de la fisura:** el área cercana al origen u orígenes está expuesta al medio ambiente durante más tiempo que cualquier otra parte de la grieta por fatiga. A veces estas variaciones en la corrosión forman anillos o líneas concéntricas que provienen del origen. Estas líneas, al igual que el cambio de contraste, ayudan a identificar con mayor facilidad el número y la ubicación de los orígenes de la fractura por fatiga.
3. **Grandes cambios de magnitud o frecuencia de carga:** el más común es un aumento considerable de la carga, que esencialmente forma estrías gigantes (marcas de playa) que pueden ser visibles a nivel macroscópico. Además, ciertos tipos de elementos están sometidos a fluctuaciones elevadas de las cargas aplicadas, que son visibles en la superficie de fractura. (Wulpi, 1985).

1.4.2 Marcas de trinquete (Ratchet Marks)

El término marcas de nivel o de trinquete se emplea para describir características macroscópicas que permiten identificar fracturas por fatiga. El número total de marcas de trinquete presentes en una superficie es igual o similar al número total de orígenes de fractura. La presencia de estas marcas suele ser un indicativo de concentradores de esfuerzo y altas cargas de servicio. Estas marcas son perpendiculares a la superficie de origen, por tanto, en piezas circulares aparecen como formas radiales que apuntan hacia el centro, como se muestran en la Figura 1.14 (Wulpi, 1985).

Se forman cuando varios orígenes de fatiga son adyacentes entre sí, cada uno de estos comienza a propagarse en su propia grieta. No son orígenes como tal, simplemente separan dos fracturas por fatiga adyacentes. Conforme estas grietas aumentan su

profundidad, tienden a crecer juntas de manera que ocasionan una sola fractura por fatiga que posee numerosos orígenes (Wulpi, 1985).

1.4.3 Estrías (Striations)

En la inspección a elevados aumentos, por medio de microscopias, las características predominantes que se encuentran son parches de marcas paralelas finamente espaciadas, conocidas como estrías. Las estrías de fatiga son orientadas perpendicularmente a la dirección macroscópica de propagación de la grieta, como se puede apreciar en la Figura 1.16. En general se reconocen dos tipos de estrías: dúctiles y quebradizas.

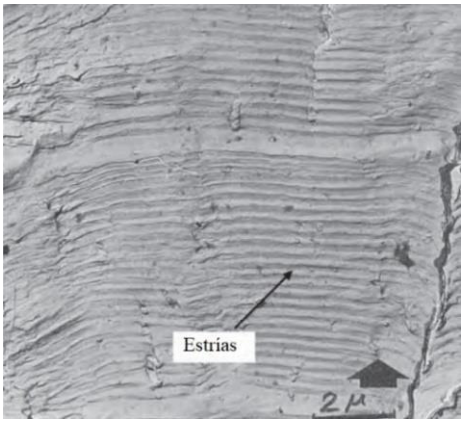


Figura 1.16. Micrografía electrónica de barrido que indica las estrías por fatiga.
(Campbell, 2012)

1.5 Proceso de falla por fatiga

Las etapas que componen este proceso de falla son complejas, esto se debe a que son influenciadas por varios factores. Para poder comprender de mejor manera cada uno de los fenómenos presentes, se han considerado tres etapas como se muestra en la Figura 1.17.

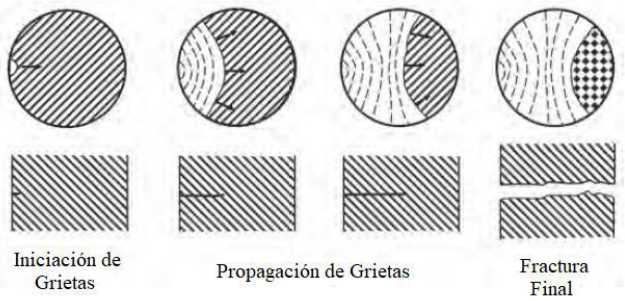


Figura 1.17. Proceso de una Falla por Fatiga.
(Campbell, 2012)

1.5.1 Etapa 1: Iniciación de microgrietas.

Las microgrietas son pequeñas aberturas poco separadas que son ocasionadas por fatiga y se propagan en zonas que presentan elevados valores de deformación plástica cíclica. Este comportamiento se debe a que, la mayoría de los materiales empleados en ingeniería no son perfectos por la presencia de ciertos defectos. Las imperfecciones de estos materiales generan concentradores de esfuerzos que ocasionan zonas con alta deformación. Esta etapa no puede ser observada a simple vista debido a que la propagación de las grietas es muy baja y no presenta marcas.

Debido a que las grietas nacen a partir de defectos presentes en el material, es necesario identificar los motivos por los cuales aparecen, se muestran a continuación:

- Cambios bruscos en la sección transversal como, por ejemplo: cuñas, orificios, entre otros. Estos aspectos considerados durante el diseño del elemento se convierten en concentradores de esfuerzo.
- Contacto entre elementos que giran o se deslizan entre sí como, por ejemplo: cojinetes, engranes, rodamientos, etc. La presión constante a la que están sometidos puede desarrollar esfuerzos por contacto, lo que resulta en la formación de picaduras o astillas en la superficie, después de elevados ciclos de servicio.
- Presencia de marcas, estampados y labrados no previstos en el diseño de los elementos. Generalmente, son el resultado de errores durante el proceso de fabricación.
- Discontinuidades ocasionadas debido a procesos adicionales de fabricación como, por ejemplo: forjado, laminado, estirado, etc.

1.5.2 Etapa 2: Propagación de microgrietas.

Las microgrietas formadas en la etapa anterior se convierten en macrogrietas. La longitud de la microgrieta se vuelve lo suficientemente grande para propagarse en la dirección normal al esfuerzo aplicado. Debido a este cambio en su longitud se forman superficies paralelas en forma de mesetas, las cuales están separadas por crestas longitudinales (Marcas de playa o Beach Marks) como se muestra en la Figura 1.18. (Campbell, 2012).

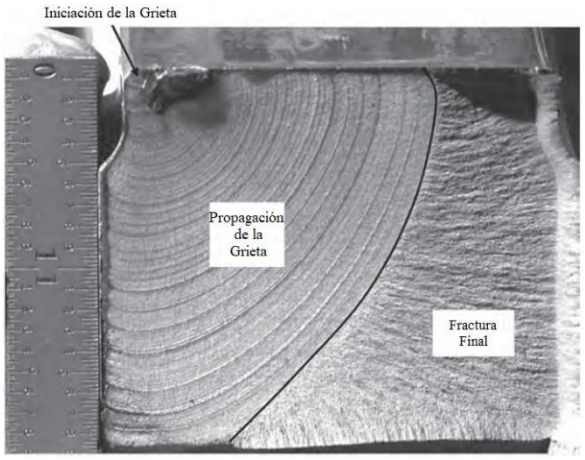


Figura 1.18. Propagación de una grieta por fatiga en un elemento de acero de alta resistencia. (Campbell, 2012)

En esta etapa se producen estrías finas (estrías de fatiga) las cuales solo pueden ser observadas a partir de elevados aumentos. Durante el crecimiento de la grieta se genera un patrón marcado de estrías, en donde cada una de estas representa un ciclo de fatiga. La formación de estrías no es indispensable para que pueda ocurrir un proceso de falla por fatiga. Los procesos de fatiga a bajos esfuerzos y elevados ciclos de servicio generan superficies de fractura de grano fino y pulido. En cambio, en procesos de elevados esfuerzos y bajos ciclos de servicio la superficie es fibrosa y rugosa (Campbell, 2012).

1.5.3 Etapa 3: Fractura final.

Esta es la etapa final del proceso de falla. La grieta generada anteriormente se vuelve más larga que la sección trasversal restante, de manera que esta no puede tolerar la carga aplicada como se muestra en la Figura 1.17. Esta etapa es la más fácil de comprender debido a que técnicamente no se habla de un proceso de fatiga, ya que se produce una fractura rápida y súbita que puede ser frágil, dúctil o una combinación de ambas. Generalmente, se genera cuando la grieta alcanzó el tamaño crítico.

La superficie de fractura final es fibrosa, similar a las superficies de una fractura por impacto. Es importante considerar algunos aspectos del área de fractura final como: tamaño, forma y ubicación. Estos parámetros permiten hallar relaciones importantes entre cargas aplicadas. Finalmente, en la Figura 1.19 se puede apreciar un eje que presenta una fractura por fatiga, donde el origen está marcado por la flecha (Campbell, 2012)(Wulpi, 1985).

Figura 1.19. Superficie de una fractura por fatiga de un eje de acero.
(Campbell, 2012)

1.6 Áreas de fractura

La fractura presenta ciertas zonas relevantes que poseen información del proceso de falla. En este caso se considera el área de fatiga y el área de fractura final. A continuación, se describe cada una de estas y sus características:

1.6.1 Área de fatiga

Es el área que posee todas las marcas características producidas durante un proceso de fatiga. Las marcas de playa y de trinquete se localizan en esta zona y su aspecto visual depende de la naturaleza de las cargas de servicio. Del aspecto superficial de esta zona es posible distinguir cierta información relevante que sirve para llevar a cabo el proceso de análisis de falla. Se puede apreciar la localización de esta área en la parte superior de la Figura 1.20.

Cuando el proceso de crecimiento de grietas es lento, esta zona presenta una superficie lisa (No existen marcas características). Un indicio de la edad de la grieta es la suavidad de esta área. Básicamente se puede decir que, mientras más lisa sea esta superficie, el proceso de crecimiento es lento y la grieta más vieja (Sachs, 2016).

Ahora, considerando el caso en el que la carga cíclica no sea constante mientras la grieta crece, la velocidad de crecimiento y la apariencia superficial se verán afectadas. Este proceso da origen a las marcas de playa. Se describen como un calendario que

muestra los cambios en la carga durante la formación y crecimiento de la grieta. Mientras mayor rugosidad y marcaciones presente una superficie, la vida de la grieta estuvo sometida a muchos cambios en la carga (Sachs, 2016).

1.6.2 Área de fractura final

Es el área o zona instantánea de fractura. Cuando la carga aplicada sobre el elemento es mayor a la resistencia restante, esta se fractura dando el origen a esta área. En la mayoría de los casos es una fractura frágil con una superficie rugosa de apariencia cristalina. Esto se debe a que se produce de una manera súbita sin dar oportunidad a generar un proceso de deformación.

La información que aporta esta área permite relacionar su tamaño con las cargas aplicadas en el elemento al momento de la fractura final. Si el área de fractura final es grande, la carga a la cual estuvo sometida era elevada. Por otro lado, si el área de fractura final es pequeña, la carga final era relativamente ligera (Sachs, 2016). Se puede apreciar la localización de esta área en la parte inferior de la Figura 1.20.

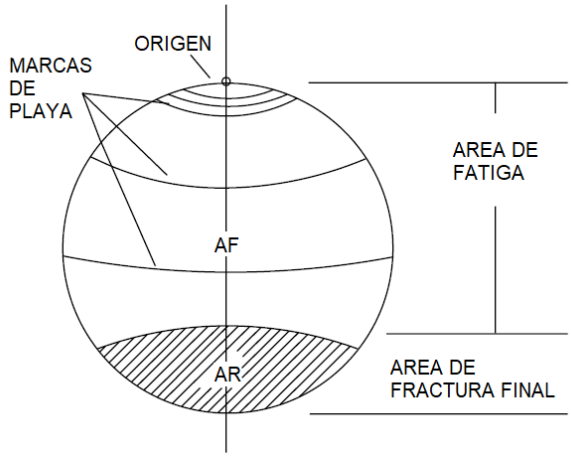


Figura 1.20. Área de fatiga y de fractura final.
(Sachs, 2016)

1.7 Influencia de las cargas en las fallas por fatiga

Existen cuatro tipos de cargas que pueden ocasionar fallas por fatiga: sobrecarga, flexión plana, flexión rotatoria y torsión. Para este proyecto se consideran únicamente los casos de flexión plana y giratoria, ya que son los más comunes. Una falla por flexión plana es característica de elementos que se encuentran expuestos a cargas cíclicas de tensión compresión como los resortes. Este tipo de falla se caracteriza por tener una superficie perpendicular al eje de acción del elemento, poseer zonas de falla y progresión definidas. Además, posee marcas de playa concéntricas y simétricas al origen.

La falla por flexión rotatoria o giratoria muestra marcas de progresión no simétricas con respecto al origen. Además, poseen un ángulo de desface entre la bisectriz del área final y el origen, que permite determinar el sentido de giro del elemento. Este tipo de carga se observa en elementos como ejes de cajas reductoras o con cargas de correas. En la Figura 1.21 se muestra como varia el origen con respecto a la bisectriz generada por la el área de fractura final.

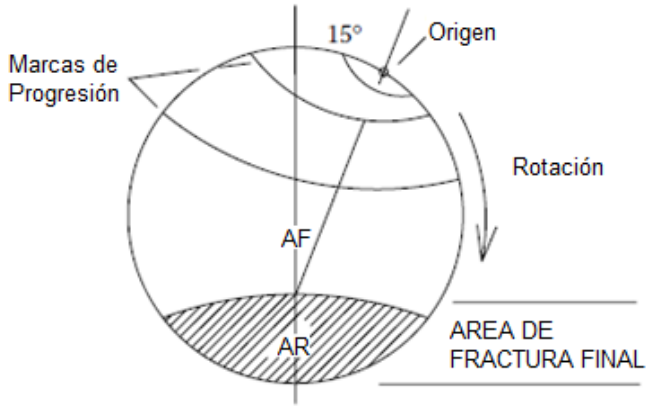


Figura 1.21. Zona de fractura de falla por flexión rotatoria.
(Sachs, 2016)

2 METODOLOGÍA

2.1 Software libre

En la actualidad, existe una gran variedad de lenguajes de programación que permiten desarrollar algoritmos para la resolución de problemas en ingeniería. Sin embargo, es importante definir la aplicación y alcance que cada uno de estos lenguajes posee. Uno de los aspectos fundamentales a la hora de seleccionar un lenguaje, es la restricción que presenta el costo de utilización. Los elevados costos que representan el pago de licencias reducen las posibilidades para el desarrollo de software con fines educativos. Con esto, no solo el proceso de implementación se ve restringido, sino también la capacidad de difusión.

Considerando la naturaleza académica educativa de este proyecto de titulación es importante acoger en su totalidad la opción que representa el empleo de software libre. Esta alternativa permite un desarrollo exento de costos elevados. Además, puede ser fácilmente difundido entre la comunidad educativa. Brindando así un acceso total a la información desarrollada, que puede ser aprovechada por estudiantes, investigadores y docentes.

2.1.1 Python

Entre los diferentes lenguajes de programación que poseen la etiqueta de software libre está Python, el cual es un lenguaje de programación interpretado de alto nivel. Presenta una sintaxis sencilla, además, es muy potente para varias aplicaciones de ingeniería (Challenger et al., 2014). Es importante mencionar que en la actualidad es considerado como un lenguaje de vanguardia, ya que cada vez es más utilizado. Por estas razones ha sido seleccionado para ser aplicado en este proyecto de titulación.

Para aprovechar al máximo las características de este lenguaje es necesario utilizar ciertas bibliotecas y módulos. En este caso, se utiliza dos bibliotecas: NumPy y Open CV. NumPy es utilizada para poder emplear una amplia variedad de operaciones con matrices. Permite manipular y crear matrices con datos numéricos (*NumPy User Guide*, 2012). Open CV posee algoritmos de visión artificial. Se utiliza para poder procesar imágenes y mostrar resultados al usuario de una manera sencilla (*OpenCV: Introduction*, 2021). Además, es necesario emplear el módulo Math que permite el acceso a funciones matemáticas definidas previamente (*Math-Funciones Matemáticas*, 2021).

2.2 Obtención de fotografías

La calidad que debe poseer una fotografía para poder ser procesada mediante el software planteado debe ser alta, es decir, que sea posible visualizar las diferentes marcas de la superficie de falla sin que exista brillo metálico. Este fenómeno reduce la visibilidad en la imagen. Para evitar esto es necesario establecer procedimientos que permitan obtener los mejores resultados. Estas consideraciones deben ser aplicadas en función del objeto de estudio. A continuación, se presentan algunas indicaciones generales.

2.2.1 Preparación de muestras

A partir de la selección previa del elemento es necesario efectuar un protocolo de limpieza. Para esto se requiere una franela, ya que permite retirar cualquier tipo de grasa presente. No se debe presionar demasiado la superficie del elemento, para evitar que se generen rayas o marcas adicionales. El empleo de técnicas de limpieza suaves permite conservar la integridad de la superficie de fractura. La Figura 2.1. indica el proceso de limpieza con franela de una muestra para procesamiento.

Figura 2.1. Proceso de limpieza (Preparación de muestras).
(Fuente: Propia)

No es recomendable, en ninguna circunstancia, realizar operaciones que incluyan: ataque químico, desbaste, lijado y pulido, debido a que estas son abrasivas y de ser aplicadas destruirían por completo los parámetros superficiales utilizados en este proyecto. Una vez realizada la toma de fotografías es necesario conservar el elemento para comparar resultados posteriormente. Después de haber ejecutado este procedimiento el elemento puede ser sometido a cualquier tipo de proceso abrasivo si así se requiere.

2.2.2 Toma de fotografías

Es importante tener en cuenta ciertos aspectos al momento de tomar la fotografía. Se debe evitar alumbrar directamente la superficie del objeto, ya que es posible generar reflejos. Para obtener una iluminación adecuada, se recomienda utilizar cartulinas negras. Estas permiten reflejar la luz delicadamente. Al momento de generar la fotografía se debe utilizar el modo "Macro" de la cámara. Este permite obtener imágenes grandes de objetos de tamaño reducido.

2.3 Elaboración del software

La metodología para la realización del software empleado se resume en la Figura 2.2. El software parte de una imagen cargada por el usuario, la cual debe cumplir con los parámetros mencionados en la metodología. Seguido de esto se efectúan las funciones definidas para el preprocesamiento y procesamiento de imagen. Finalmente, se realiza el cuestionario final y se muestran los resultados obtenidos.

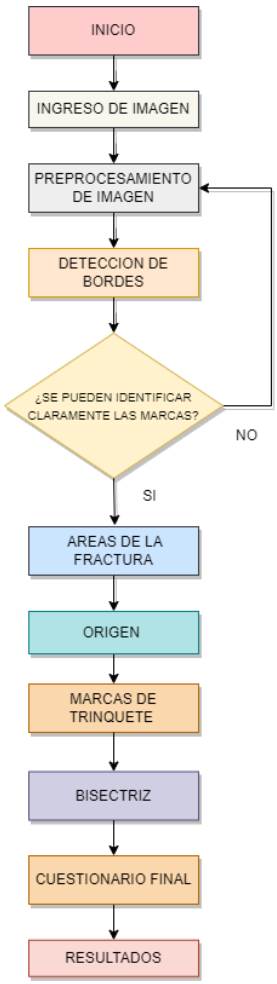


Figura 2.2. Metodología empleada en el desarrollo del software
(Fuente: Propia)

2.3.1 Preprocesamiento de imágenes

La Figura 2.3. presenta las etapas definidas para el preprocesamiento de imágenes. El objetivo de este conjunto de funciones es preparar y optimizar la imagen cargada por el usuario. Al definir adecuadamente los parámetros, se puede orientar los resultados a las necesidades visuales del estudio o del usuario

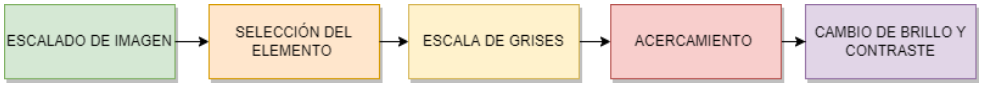


Figura 2.3. Etapas del preprocesamiento de imagen
(Fuente: Propia)

2.3.1.1 Escalado de imagen

Esa función parte de la imagen cargada por el usuario. El método seleccionado como se muestra en la Figura 2.4, se basa en el cálculo de una dimensión de la imagen a partir de otra fija, con el objetivo de evitar distorsiones y conservar la relación de aspecto. En este caso se considera como valor fijo la altura de la imagen en 690 pixeles y el ancho se determina automáticamente. El tamaño máximo que puede llegar a tomar una imagen en el software es de 920 x 690 pixeles, ya que este es el tamaño del visor principal del software. Finalmente, esta función imprime una copia de la imagen en formato png, llamada “Imagen Estandar.png”.

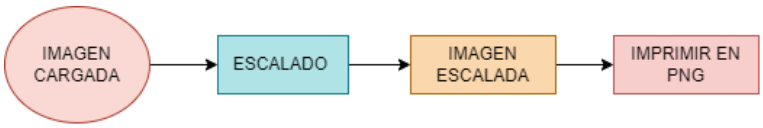


Figura 2.4. Escalado de Imagen
(Fuente: Propia)

2.3.1.2 Selección de elemento

Permite identificar el tipo de sección transversal que más le convenga al usuario dependiendo del caso y posteriormente, retirar el fondo que se encuentre por fuera de la sección marcada. Se han planteado dos secciones: circular y rectangular. La Figura 2.5. muestra el diagrama del proceso.

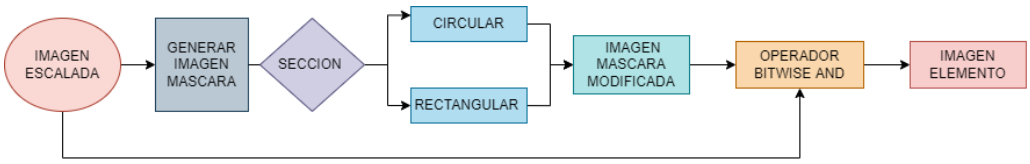


Figura 2.5. Selección de elemento
(Fuente: Propia)

La selección inicia con la imagen escalada. Se genera una imagen máscara de las mismas dimensiones en color negro. A continuación, el usuario debe identificar el tipo de sección. A partir de esta información, la imagen máscara se modifica. Todo lo que se encuentre por fuera de la figura será de color negro y la zona interior será de color blanco. La Figura 2.6. muestra las máscaras generadas en función de la sección escogida.

Figura 2.6. Máscaras generadas en función de la sección seleccionada (Imagen máscara modificada).
(Fuente: Propia)

Mediante la imagen máscara modificada se realiza una intersección utilizando los operadores bitwise con la imagen escalada. En este caso la intersección se realiza con el operador "And". La Tabla 2.1. muestra una analogía de la tabla de verdad del operador "And" en Open CV. El valor 0 corresponde al color negro y 255 corresponde al color blanco o una combinación.

Tabla 2.1. Tabla de verdad "And".

Imagen Escalada	Imagen Mascara Modificada	Imagen Elemento	Imagen Elemento
0	0	0	Fondo
0	255	0	Fondo
255	0	0	Fondo
255	255	255	Elemento

(Fuente Propia)

Al aplicar la tabla de verdad del operador "And" los pixeles, en función de su valor, pueden ser considerados como parte del fondo (Color Negro) o del elemento (Color) como se puede apreciar en la Figura 2.7. Finalmente, después de clasificar los pixeles se genera la imagen elemento.

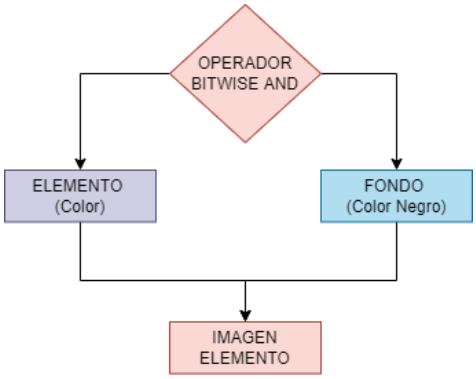


Figura 2.7. Intersección con el operador bitwise and.
(Fuente: Propia)

2.3.1.3 Escala de grises

La escala de grises permite transformar una imagen modelo BGR a una imagen gris, la cual presenta únicamente un canal que posee valores de intensidad que varían entre 0 y 255. Donde el 0 representa el color negro y 255 el color blanco. Esta operación se emplea con frecuencia en el procesamiento de imágenes porque permite simplificar la complejidad del análisis, mediante la reducción de los canales de color. Al realizar esta operación la intensidad del píxel se define por un único valor, esto permite facilitar la detección, discriminación e identificación de parámetros relevantes (Viera, 2017).

La función definida en el software utiliza la librería Open CV la cual, a partir de la imagen obtenida de la etapa anterior en BGR, permite obtener una imagen del mismo tamaño en escala de grises. Además, imprime la imagen obtenida en formato png con el nombre de "Imagen Escala de Grises.png". La Figura 2.8 muestra el proceso para ejecutar la transformación.

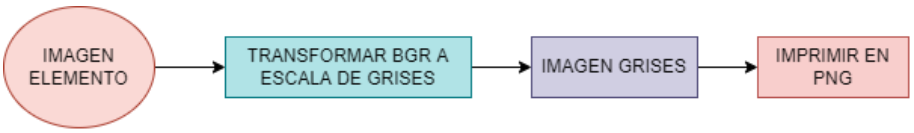


Figura 2.8. Escala de grises.
(Fuente: Propia)

2.3.1.4 Acercamiento

Debido a la naturaleza visual de este proyecto es necesario efectuar un acercamiento para discernir de mejor manera los detalles que presenta la superficie. La Figura 2.11 indica el procedimiento ejecutado para realizar el acercamiento de imagen. El proceso inicia con la imagen grises. Seguido se marca la zona de interés. La metodología para escalar este fragmento seleccionado de imagen es el mismo utilizado en la función

“Escalado de Imagen”. Finalmente, con este fragmento se genera la imagen acercamiento.

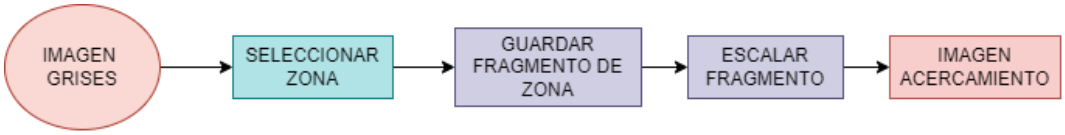


Figura 2.9. Acercamiento.
(Fuente: Propia)

2.3.1.5 Brillo y contraste

Al analizar una superficie es necesario controlar dos variables: brillo y contraste: el primero permite aumentar la luminosidad de la imagen, es decir aclarar u opacar colores, y el segundo aumenta la definición de un determinado objeto (*Función de Brillo y Contraste—Ayuda | ArcGIS for Desktop*, n.d.).

Al modificar estos parámetros es posible mejorar la calidad de detección de bordes. Además, mejora el aspecto visual para la realización del análisis visual. La biblioteca Open CV posee ciertas funciones que permiten alterar el brillo y contraste mediante la modificación de sus parámetros de entrada. Básicamente, consiste en variar el peso que puede tomar cada una de estas imágenes, con el objetivo de retocar la sensibilidad de la imagen resultante. La Figura 2.10. muestra el procedimiento para cambiar el brillo y contraste de una imagen.

Figura 2.10. Cambio de contraste y brillo.
(Fuente: Propia)

Se toma la imagen acercamiento y se genera otra imagen de iguales dimensiones en color negro, para posteriormente juntarlas. A continuación, se añaden distintos valores en la función, que permiten alterar el peso de cada imagen. Para el brillo los valores van desde -127 a 127. Esto se debe a que se considera un intervalo intermedio, es decir, de -127 a 0 se extrae brillo de la imagen y de 0 a 127 se añade brillo. De manera que se obtienen los 256 valores de intensidad como se indica en la función “Escala de Grises”. El contraste varía entre 0 y 100%. En este caso, se ha definido esta escala porcentual ya que es más sencillo cuantificar la variación del brillo. La alteración de este parámetro indica que tan grande es la diferencia entre tonos claros y obscuros. Finalmente, se genera una imagen mejorada.

2.3.2 Detección de bordes

Para poder identificar y resaltar las marcas superficiales se han establecido dos métodos. Estos presentan la misma funcionalidad, pero tienen diferente sensibilidad para detectar rasgos superficiales. El usuario puede escoger entre estos dos métodos dependiendo del caso. Es importante mencionar que cada método posee un rango de valores entre 0 a 255, conforme al intervalo de intensidad que se pueden encontrar en una imagen digital. La Figura 2.11. muestra las etapas de cada método.

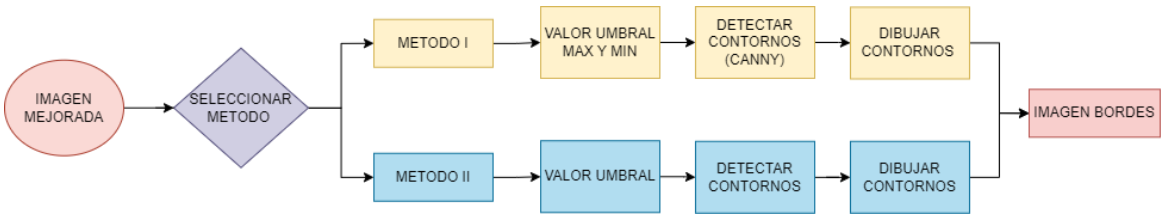


Figura 2.11. Detección de bordes (Método I y II).
(Fuente: Propia)

2.3.2.1 Método I

El proyecto de titulación se enfoca en la detección de marcas que presentan una separación reducida, por lo cual es necesario emplear funciones que enmarquen delicadamente las zonas de interés. Por lo cual se emplea la función “Canny ” que utiliza el método de “Histéresis” para generar una marcación nítida.

La histéresis es un método umbral empleado comúnmente para la segmentación de imágenes digitales. En este caso se realiza a partir de dos umbrales (Tmáx y Tmín). Estos dos valores permiten establecer un rango de análisis, para separar o rechazar los bordes a partir del valor de intensidad que poseen los pixeles en la imagen (Hz et al., 2007).

Finalmente, estos bordes son colocados en la imagen mejorada de manera que se obtiene una nueva imagen bordes. La Figura 2.15. muestra un diagrama con los criterios respectivos para la generación de bordes.

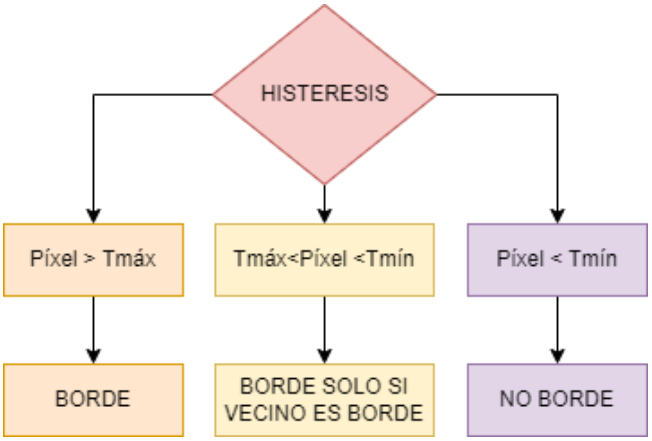


Figura 2.15. Histéresis.
(Fuente: Propia)

- Valores de intensidad de píxel mayores al Tmáx se consideran automáticamente como bordes. Se pintan de color dorado.
- Valores de intensidad de píxel entre Tmáx y Tmín solo se consideran como bordes cuando a su alrededor existe un píxel vecino que sea borde de lo contrario no, este intervalo es una zona de incertidumbre.
- Valores de intensidad de píxel menores a Tmín no se consideran bordes en ninguna circunstancia.

2.3.2.2 Método II

Emplea distintas funciones de la biblioteca Open CV, las cuales en conjunto permiten detectar bordes mediante la "Umbralización simple". En este método, a diferencia del método anterior, el usuario debe introducir únicamente un valor umbral. La Figura 2.16. muestra los criterios para la identificación de bordes.

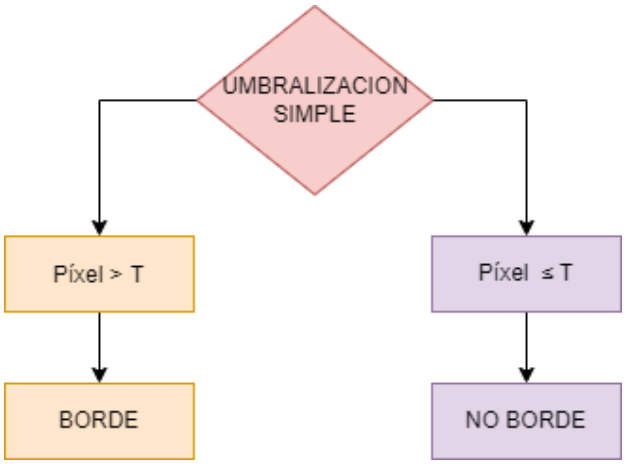


Figura 2.16. Umbralización simple.
Fuente: Propia

- Valores de intensidad de píxel mayores al umbral T se consideran como un borde y se pintan de color verde.
- Valores de intensidad de píxel menores al umbral T no muestran ningún cambio.

El proceso inicia con la imagen mejorada. Después, se debe introducir el valor umbral. A partir de este punto se realizan las mismas operaciones que en el método I. Se imprimen los bordes detectados en color verde sobre la imagen mejorada obteniendo así una nueva imagen bordes. Finalmente, se genera una copia del resultado en formato png, llamada “Imagen con Bordes.png”.

2.3.3 Áreas de la fractura

Antes de poder seleccionar cualquier área de interés, es necesario considerar el caso en el cual el elemento analizado sea un eje hueco. Para esto se realiza una selección controlada del agujero central del eje. De manera que, al seleccionar las áreas, este espacio no se considere en los cálculos correspondientes. El funcionamiento de selección es el mismo utilizado para la marcación de las áreas de fractura, el cual se detalla en los siguientes párrafos. En la Figura 2.18 se muestra las etapas que componen este procedimiento.

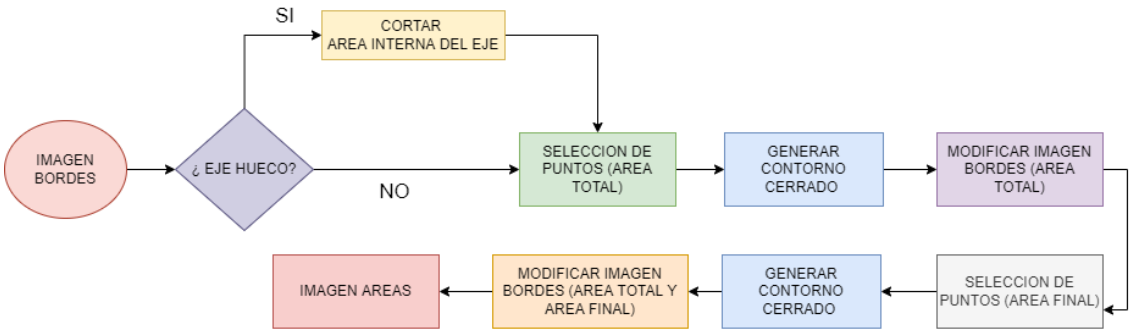


Figura 2.18. Etapas de la selección del área total y de fractura final.
(Fuente: Propia)

Para seleccionar las áreas es necesario tomar la imagen borde. Seguido se generan puntos que marcan un contorno cerrado. Empleando funciones de OpenCV se calcula el área del contorno mediante una matriz de puntos. Para mostrar los resultados es necesario buscar el contorno y posteriormente sobreponerlo en la imagen bordes modificándola. Este procedimiento es utilizado para la selección de ambas áreas y para retirar el agujero de un eje hueco de ser necesario. La imagen áreas presenta los contornos seleccionados. En la Figura 2.19 se muestra la selección de las distintas áreas de fractura. El porcentaje de las áreas permite clasificar la magnitud de la carga

final aplicada. En este caso, si el porcentaje de área final es menor a 50% se considera una carga baja y si supera este porcentaje se establece una carga elevada.

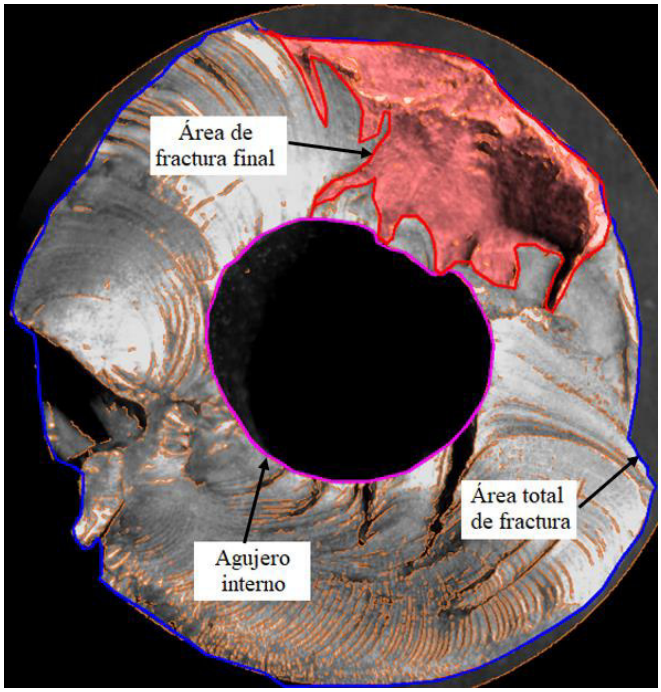


Figura 2.19. Selección de las distintas áreas de fractura.
(Fuente: Propia)

2.3.4 Marcas de trinquete

El proceso inicia con la imagen áreas, tal y como se muestra en la Figura 2.20. Después de la selección, se genera una línea de color amarillo a lo largo de la marca de trinquete identificada. Finalmente se crea la imagen trinquete.

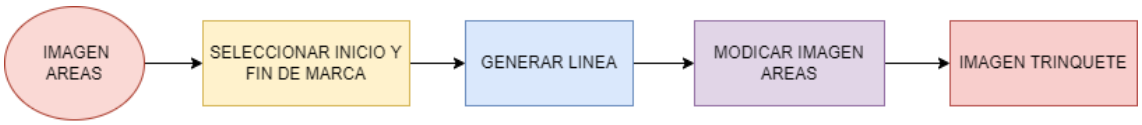


Figura 2.20. Etapas de la selección de marcas de trinquete.
(Fuente: Propia)

2.3.5 Selección de origen

La Figura 2.21 muestra las etapas establecidas para la selección del origen de fractura. En este caso se define un modelo geométrico que permite obtener una línea de progresión en la cual se encuentra el origen, este será seleccionado por el usuario según su criterio. Este proceso parte de la imagen trinquete y mediante la modificación de esta se obtiene la imagen origen.

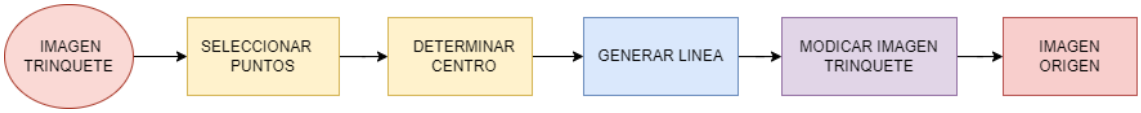


Figura 2.21. Etapas de la selección de origen.
(Fuente: Propia)

2.3.6 Bisectriz

Una bisectriz se define como una recta que divide un ángulo o zona en dos partes iguales. Esta función permite obtener una bisectriz, que nace del área de fractura final a partir de dos puntos seleccionados por el usuario. Toma como parámetro de entrada la imagen origen. Los resultados se muestran en la imagen bisectriz. El ángulo que se forma entre la bisectriz y la línea de origen permite establecer el tipo de carga. Es decir, si el ángulo es menor a 10º se considera que la carga principal de falla es flexión plana. Si es mayor a 10º es flexión rotatoria. Es importante mencionar que este criterio únicamente se cumple cuando existe un solo origen. Esto se debe a que la existencia de varios orígenes lo que implica la presencia de concentradores de esfuerzo y cargas de flexión plana y rotatoria. La Figura 2.22 muestra las etapas realizadas para generar la bisectriz.

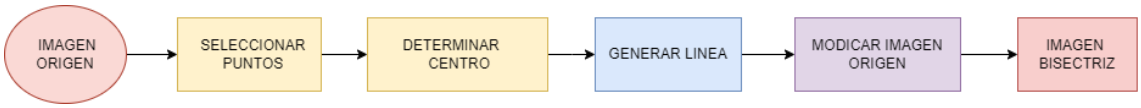


Figura 2.22. Etapas de la Bisectriz.
(Fuente: Propia).

Finalmente, para más información con respecto a cada una de las funciones mencionadas en la metodología revisar el manual de usuario. En este se detalla su funcionamiento y los principios utilizados para su creación.

2.4 Cuestionario final

Una vez que las funciones han sido ejecutadas, es necesario establecer un cuestionario que permita al usuario introducir más información necesaria. Mediante siete preguntas claves, se solicita información con respecto a parámetros superficiales específicos visibles en la fractura. En función de estos es posible otorgar resultados más relevantes en cuanto al análisis superficial de la fractura. La Tabla 2.2 muestra las preguntas que componen el cuestionario. En esta se puede observar la imagen empleada y su función.

Tabla 2.2. Preguntas del cuestionario final.

No	Pregunta	Función del programa	Resultado
1	¿La cara de la superficie de la fractura es perpendicular al eje de acción del elemento?	Acercamiento	Imagen Acercamiento
2	¿Existe deformación en el área de fractura?	Detección de bordes	Imagen Bordes
3	¿Existen marcas de trinquete?	Detección de bordes	Imagen Bordes
4	¿Existen marcas de progresión?	Detección de bordes	Imagen Bordes
5	¿El área de fatiga rodea al área de fractura final?	Áreas de la fractura	Imagen Áreas
6	¿El área de fractura final es redonda u ovalada?	Áreas de la fractura	Imagen Áreas
7	¿La bisectriz que se forma en la zona de fractura final apunta al origen?	Bisectriz	Imagen Bisectriz

(Fuente: Propia)

A continuación, se presentan los criterios utilizados para cada una de las preguntas:

- **Pregunta #1:** La relación que existe entre la superficie de fractura y su eje de acción ya sea perpendicular o posea un ángulo determinado, permite distinguir entre dos tipos de esfuerzo a los que el elemento pudo haber sido sometido. Esfuerzo de flexión o torsión. Los esfuerzos de torsión generan fracturas con ciertos ángulos, a diferencia de los esfuerzos de flexión, que dejan como resultado una superficie plana medianamente regular.
- **Pregunta #2:** La deformación presente en la superficie de fractura otorga información con respecto a la categoría de falla. La fatiga de muy bajos ciclos se caracteriza por presentar deformación superficial, debido a que el proceso se efectúa bajo cargas elevadas. Al contrario, la fatiga de altos ciclos se lleva a cabo con cargas bajas, de manera que la superficie de fractura se vuelve lisa.
- **Pregunta #3:** La presencia de marcas de trinquete está ligada a elevadas tensiones de servicio y concentradores de esfuerzo. Para más información revisar la sección “Marcas de Trinquete” en el marco teórico.
- **Pregunta #4:** La identificación de marcas de playa o marcas de progresión es un claro indicativo de que la carga cíclica aplicada ha variado a lo largo de la vida del elemento. De lo contrario, no se generan este tipo de marcas.

- **Pregunta #5:** La ubicación del área de fractura final con respecto a la superficie de fractura total del elemento permite determinar el tipo de carga principal que ocasionó la falla, flexión plana o la flexión rotatoria. En caso de que la zona de fatiga rodee a la zona de fractura final se considera una carga de flexión rotatoria como principal responsable de la falla del elemento, de no ser así se considera una carga de flexión plana.
- **Pregunta #6:** La forma de la fractura final permite determinar la carga secundaria a la que fue sometido el elemento. De esta forma si la zona de la fractura final posee una forma redonda, indica que no estuvo involucrada una carga de flexión plana. Sin embargo, si tiene una forma ovalada o alargada se determina que existe una carga de flexión plana implicada.
- **Pregunta #7:** La dirección de la bisectriz que se forma en la zona de fractura final permite determinar si existe o no la presencia de flexión rotatoria. Si la bisectriz apunta en la misma dirección que la línea de referencia del origen, no existe una flexión rotatoria implicada. Mientras que si la bisectriz apunta en otra dirección se establece la presencia de una flexión rotatoria.

2.5 Interfaz gráfica

La interacción del usuario con el software implementado se logra a partir de una interfaz gráfica, como se muestra en la Figura 2.23. Esta permite reducir la complejidad de uso, de manera que pueda ser ejecutada por cualquier persona. En este caso se utiliza PyQT5 de la biblioteca gráfica QT para Python. Este binding permite crear interfaces de manera sencilla, debido a que posee un diseñador con varios elementos preestablecidos. Es posible colocar distintos botones y posteriormente vincular las funciones programadas.

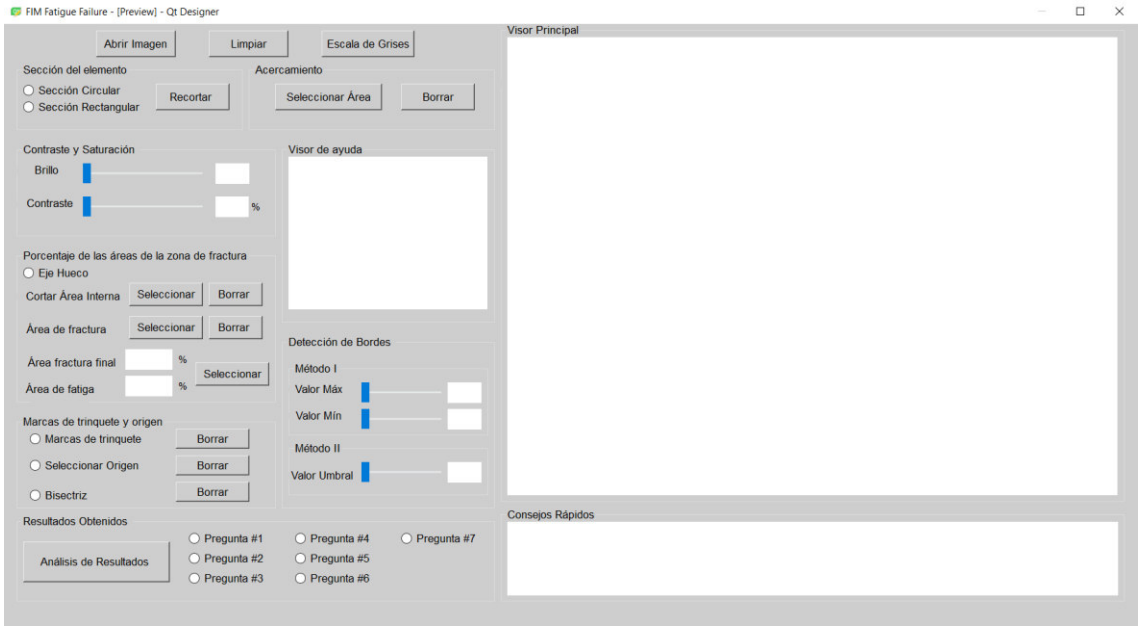


Figura 2.23. Interfaz gráfica implementada.
(Fuente: Propia)

La interfaz ha sido dividida en 3 secciones. En primer lugar, se genera una ventana introductoria, como se muestra en la Figura 2.24. En esta se presenta el nombre del software y de los autores. Además, se muestran las restricciones de uso.

Figura 2.24. Ventana de Bienvenida.
(Fuente: Propia)

La Figura 2.25 muestra la primera sección de la interfaz. En esta se presentan todas las funciones que corresponden al preprocesamiento de imagen.

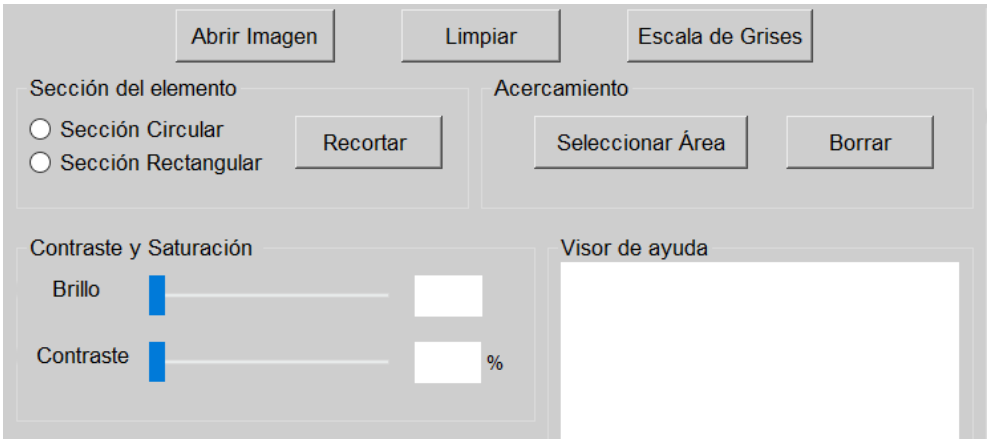


Figura 2.25. Primera Sección.
(Fuente: Propia)

A continuación, se analiza la segunda sección que se muestra en la Figura 2.26. En esta se presentan todas las funciones que comprenden el procesamiento de imagen.

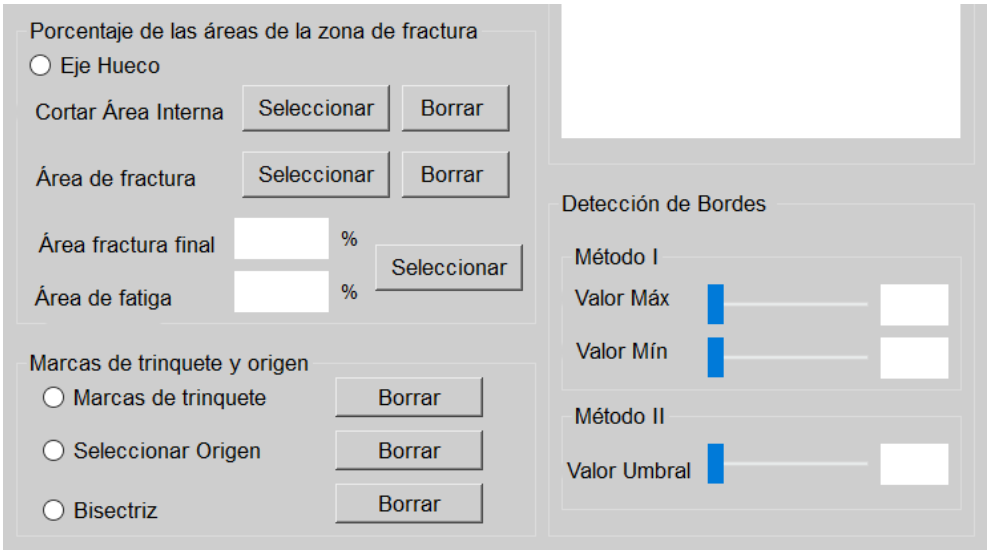


Figura 2.26. Segunda Sección.
(Fuente: Propia)

Finalmente, la tercera sección corresponde al cuestionario final y análisis de resultados, como se puede apreciar en la Figura 2.27. En esta, se presentan todas las preguntas mencionadas en la metodología.

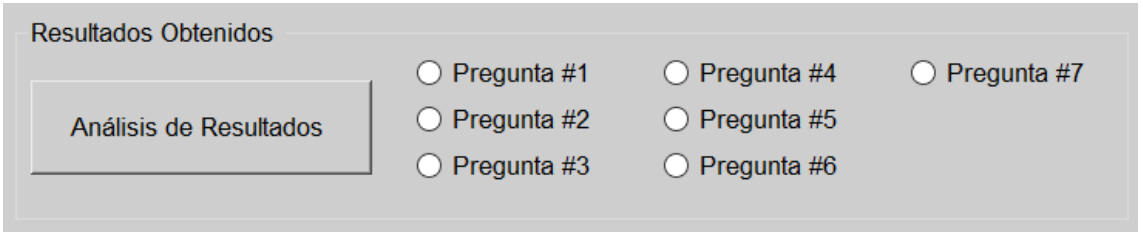


Figura 2.27. Tercera Sección.
(Fuente: Propia)

La Figura 2.28 indica la ventana generada para cada una de las siete preguntas correspondientes al cuestionario final, ver Tabla 2.2. Además, se muestran dos imágenes adicionales, que sirven de base para poder comparar visualmente los resultados obtenidos. Se incluye un visor de texto que presenta la descripción de la pregunta y sus opciones.

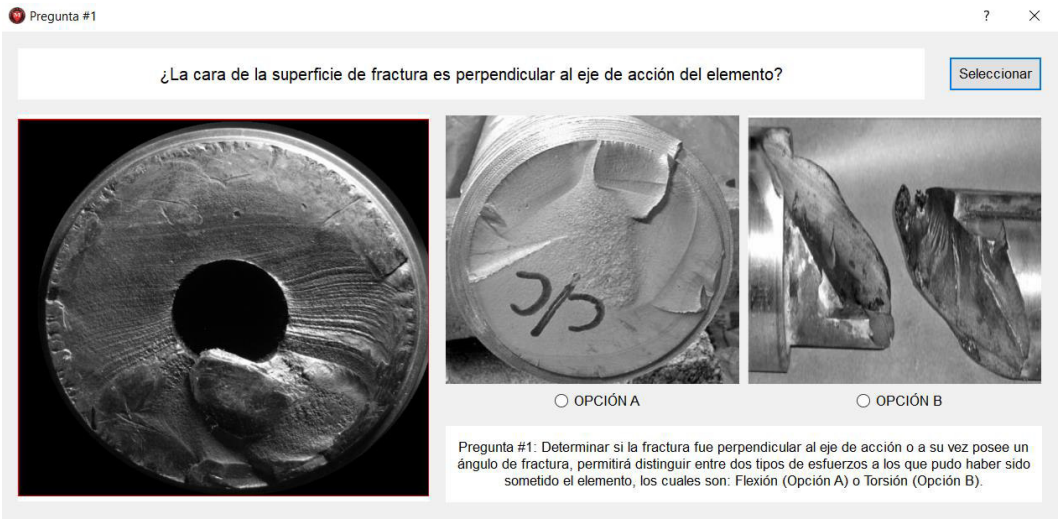


Figura 2.28. Ventana auxiliar de la Pregunta #1.
(Fuente: Propia)

Una vez que el cuestionario ha sido finalizado, se despliega la ventana de resultados obtenidos, como se puede observar en la Figura 2.29. Esta presenta la imagen final del software y la información recopilada a partir de las selecciones ejecutadas por el usuario en el cuestionario.

Figura 2.29. Ventana de Resultados Obtenidos.
(Fuente: Propia)

2.5.1 Consideraciones

- Todas las imágenes obtenidas en la metodología se presentan en el visor principal. La Figura 2.30 muestra el visor principal, que contiene una imagen cargada por el usuario.

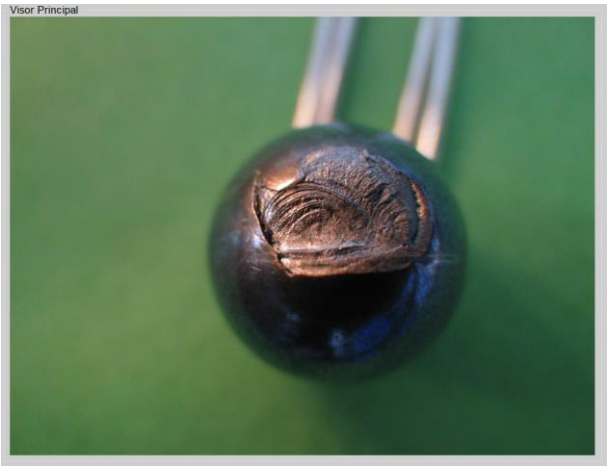


Figura 2.30. Visor Principal.
(Fuente: Propia)

- Los botones se bloquean y desbloquean automáticamente conforme el usuario avance en el software. La Figura 2.31 muestra la apariencia de los botones bloqueados y desbloqueados.

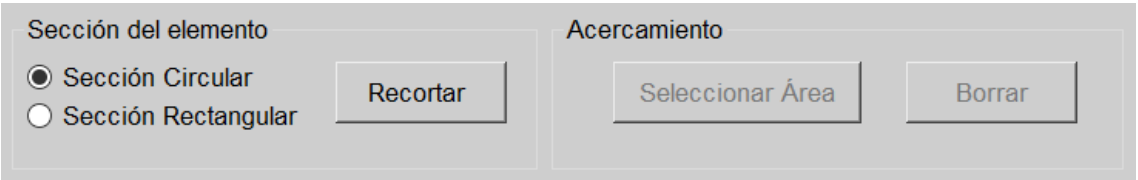


Figura 2.31. Bloqueo y desbloqueo de botones.
(Fuente: Propia)

- El “Visor de Ayuda” presenta imágenes que contienen indicaciones gráficas automáticamente, para ejecutar las diferentes funciones presentes en el software. La Figura 2.32 muestra al visor cargado con una imagen de apoyo para ejecutar la función “Selección de Elemento”.

Figura 2.32. Visor de ayuda.
(Fuente: Propia)

- El visor “Consejos Rápidos” presenta información relevante con respecto a cada una de las funciones. Conforme el usuario haga clic en los botones este texto cambiará automáticamente. La Figura 2.33 muestra el visor de texto con la descripción de la función “Escala de Grises”.

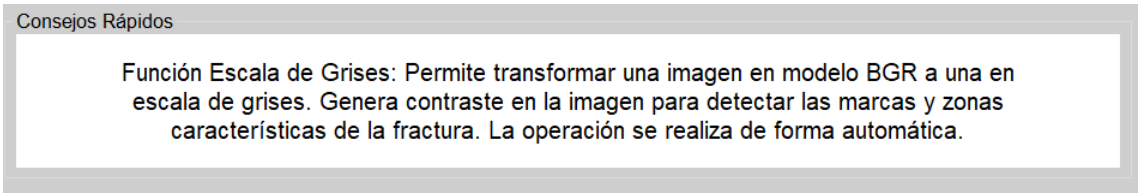


Figura 2.33. Visor de Consejos Rápidos.
(Fuente: Propia)

2.6 Metodología para obtención de error

Se puede identificar dos tipos de errores. El primero, que es imposible de analizar en el presente proyecto, es el error que se obtiene por la distorsión generada, debido a la proyección de la superficie de falla en un plano. Hay que entender que la superficie de falla no siempre posee una forma regular. Posee diferentes deformaciones a diversos planos de acción, sin embargo, en una fotografía tomada de forma perpendicular a la zona de fractura se observa como la proyección. Esto ocasiona pérdida de información.

El segundo error es el ocasionado por el cálculo de los porcentajes de las diferentes áreas de fractura. Debido a la metodología empleada, se calcula el área del contorno por medio de momentos los cuales son estimados a partir de la intensidad de los pixeles que se encuentran en la región. Esta forma de cálculo hace que la incidencia de la intensidad de los pixeles pueda ocasionar fallos.

Para determinar este error es necesario obtener un valor referencial que pueda ser comparado con el obtenido por medio del software. AutoCAD, permite el cálculo de una superficie dentro de un contorno por medio de un método distinto al utilizado en este proyecto. Esta operación se realiza mediante una integración, por esta razón, es más preciso. Las unidades están mm^2.

El proceso inicia con la imagen escalada correspondiente al eje hueco analizado en este documento. Esta imagen posee una regla numerada, que permite brindar una escala de referencia. La Figura 2.34 muestra la fotografía del eje junto con la regla numerada.

Figura 2.34. Imagen del elemento analizado empleado para la validación.
(Fuente: Propia)

Seguido de esto se procede a escalar la imagen áreas, obtenida mediante el software. Para esto se considera como referencia una medida nominal de la Figura 2.34. Esta corresponde al diámetro del eje. En la Figura 2.35 (a) se observa la medida con la cual fue escalada la imagen áreas y la Figura 2.35 (b) la imagen áreas escalada.

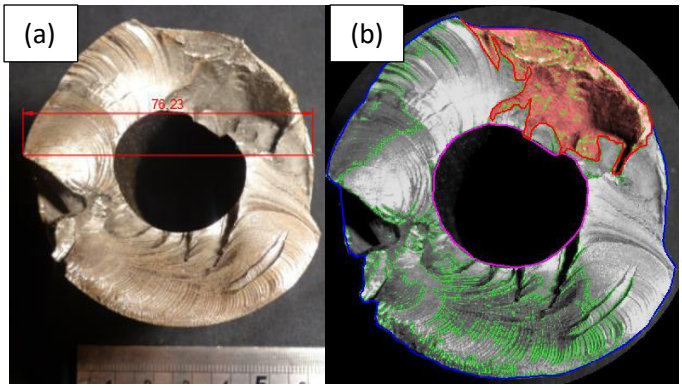


Figura 2.35. (a) Imagen con la cota de referencia (b) Imagen áreas escalada.
(Fuente: Propia)

Después, con la ayuda de una polilínea, se generan tres perímetros que delimitan las diferentes zonas de análisis. En este caso el Perímetro 1 indica el contorno del elemento, el Perímetro 2 delimita la zona interna del eje hueco y el Perímetro 3 será el contorno del área de fractura final. A partir de estas zonas marcadas se obtienen las áreas correspondientes. En la Figura 2.36 se muestra la selección de los tres contornos establecidos.

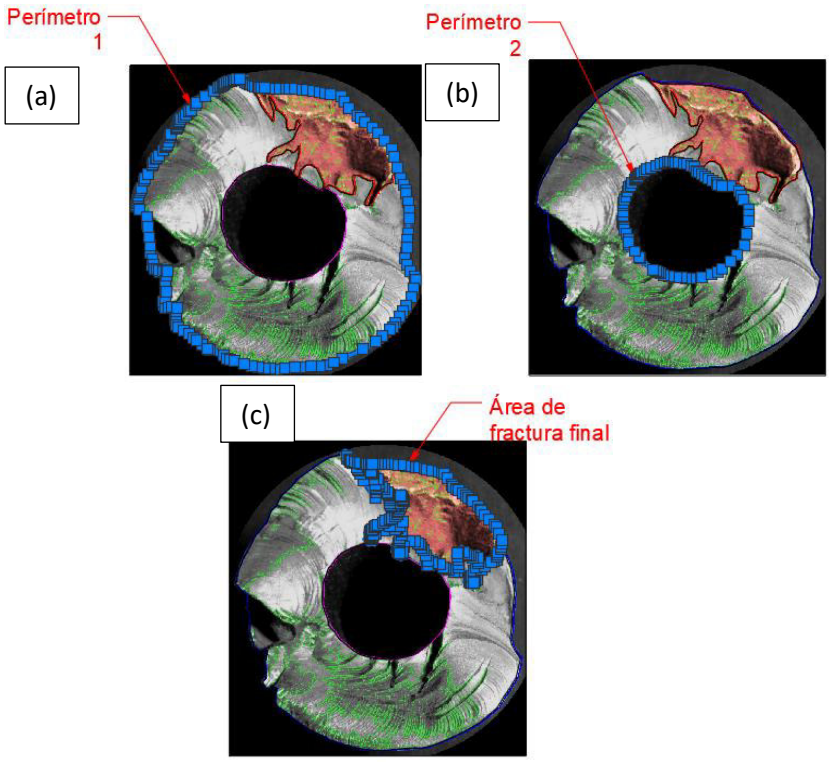


Figura 2.36. Selección de áreas de estudio. (a) Selección del perímetro 1. (b) Selección del perímetro 2. (c) Selección del área de fractura final o perímetro 3.
(Fuente: Propia)

Una vez terminada la selección, se emplea la herramienta de “Cálculo de áreas” para obtener el área de los diferentes contornos en AutoCAD. Luego se procede a comparar

los valores obtenidos para determinar los porcentajes de error mediante la Ecuación 2.1 y 2.2. Cabe mencionar que este proceso fue realizado para ambos métodos de detección de bordes presentes en el software.

$$\%\ del\ área\ de\ fractura\ final = \frac{Área\ de\ fractura\ final}{Área\ de\ fractura\ total} * 100\% \qquad (2.1.)$$

$$\%\ del\ área\ de\ fatiga = \frac{Área\ de\ fractura\ total - \ Area\ de\ final}{Area\ de\ fractura\ total} * 100\% \qquad (2.2.)$$

3 RESULTADOS Y DISCUSIÓN

3.1. Resultados

Los resultados que se presentan a continuación han sido obtenidos mediante el software desarrollado. Para mostrar esta información se considera la fractura de un elemento mecánico que se ajusta a los parámetros analizados en el marco teórico. En este caso se emplea un eje hueco de acero de bajo carbono. Para más información con respecto a los resultados y funciones revisar la metodología.

Los primeros resultados, se obtienen de las funciones que conforman el preprocesamiento de imagen. La Figura 3.1 muestra la imagen escalada correspondiente al elemento.

Figura 3.1. Resultados de la Función Escalado de Imagen. (Imagen Escalada)
(Fuente: Propia)

A continuación, se identifica y selecciona la sección transversal del elemento. En este caso, se elige la sección circular como se puede observar en la Figura 3.2 (a). La calidad de la selección depende del usuario. La Figura 3.2 (b) presenta el resultado final, en el cual el fondo ha sido removido y se torna de color negro.

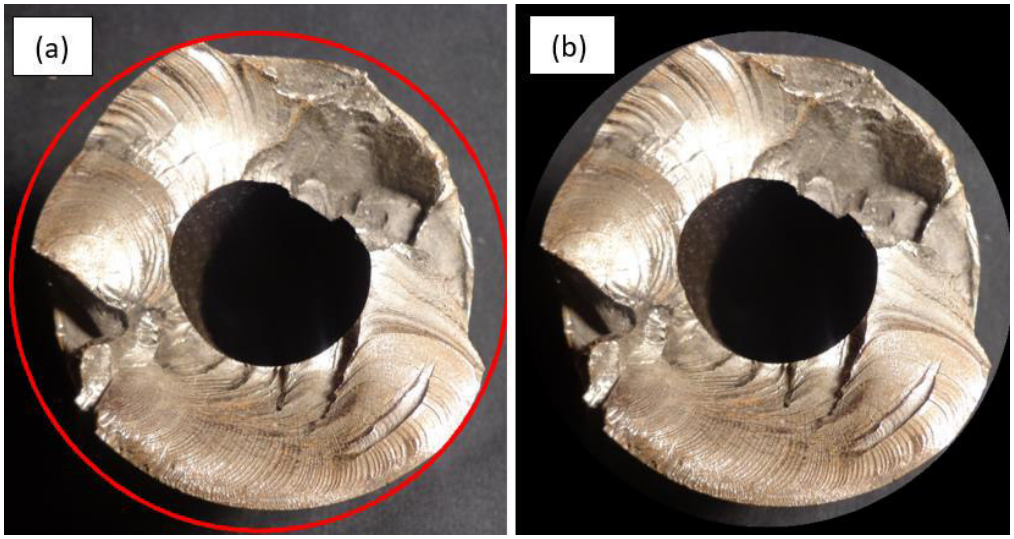


Figura 3.2. Resultado Función Selección de Elemento (Imagen Elemento). (a) selección de elemento. (b) imagen sin fondo.
(Fuente: Propia)

Una vez que ha sido seleccionada el área de interés, se procede a transformar la imagen de formato BGR a escala de grises. La Figura 3.3 muestra la conversión del elemento. En este caso se aplica una conversión a grises estándar. Mediante el uso de la función acercamiento se generan una ampliación de la zona seleccionada. La Figura 3.4 presenta el acercamiento en el eje hueco.

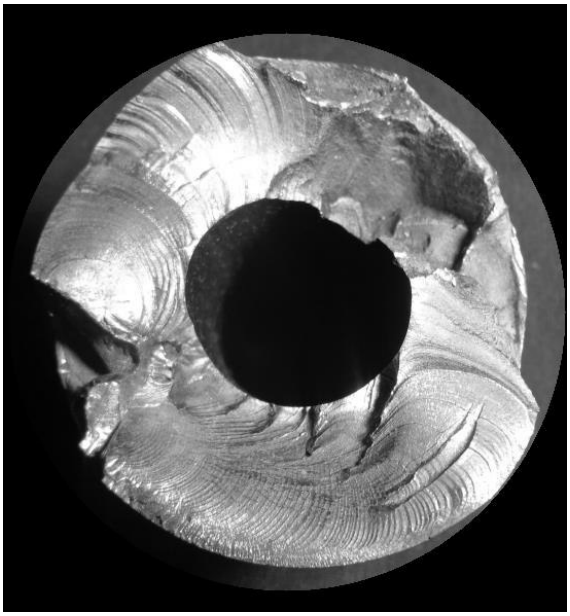


Figura 3.3. Resultados Función Selección Escala de Grises. (Imagen Grises)
(Fuente: Propia)

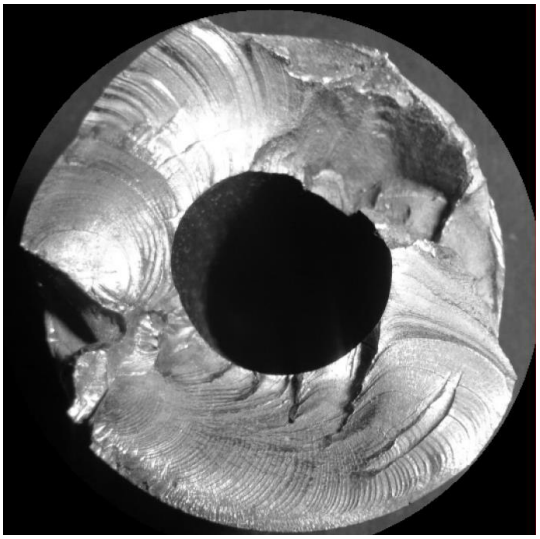
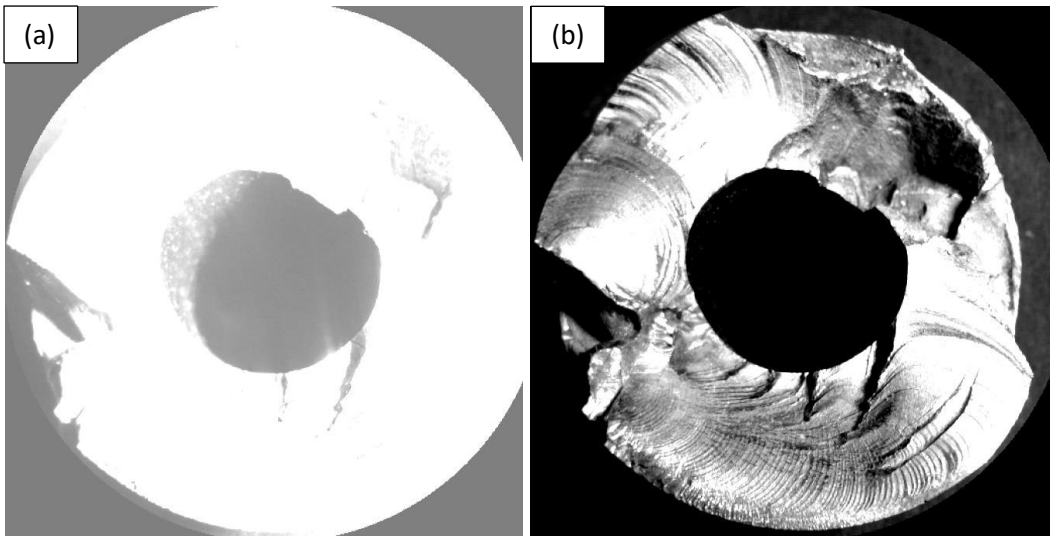


Figura 3.4. Resultados de la Función Acercamiento. (Imagen Acercamiento)
(Fuente: Propia)

Finalmente, se efectúa la última función correspondiente al preprocesamiento, el ajuste de brillo y contraste. La Figura 3.5 corresponde a los resultados obtenidos para el eje hueco. En la Figura 3.5 (a) se muestra una imagen con el 100% de contraste y un valor de brillo de 127. La Figura 3.5 (b) presenta el caso en el que existe un 100% de contraste y un valor de brillo de -127. En la Figura 3.5 (c) se muestra una imagen con el 70% de contraste y un valor de brillo igual a 0. Finalmente, se muestra en la Figura 3.5 (d) el resultado correspondiente para el caso en el que los parámetros de brillo y contraste poseen una combinación equilibrada. Brillo -30 y contraste al 50%.



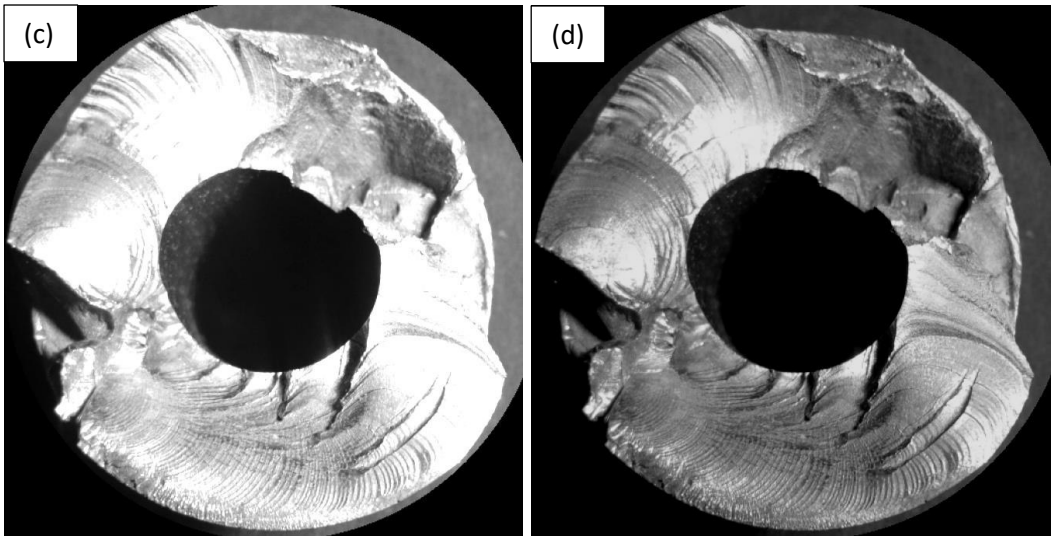
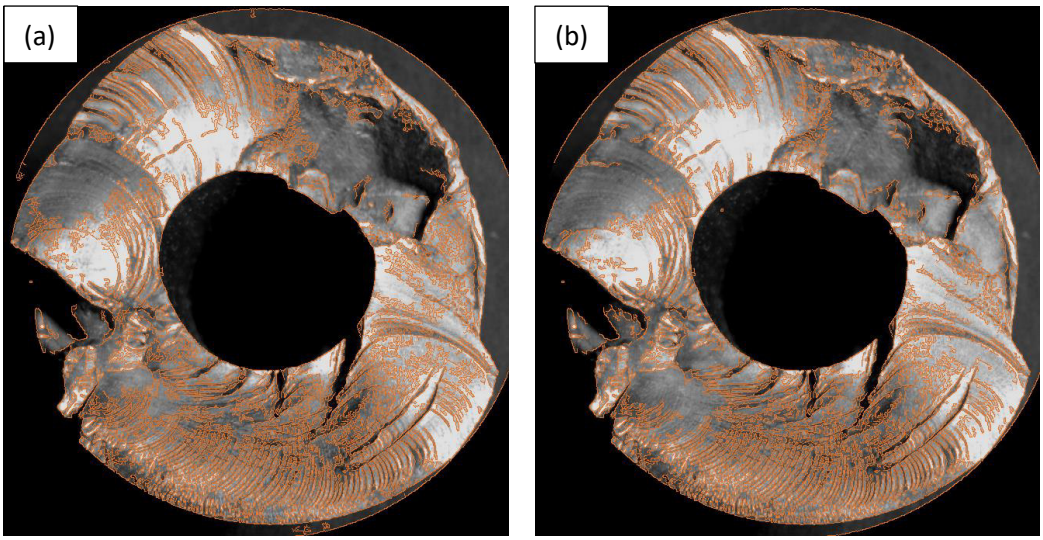


Figura 3.5. Resultados de la Función Brillo y Contraste para el Eje hueco. a) Configuración: brillo = 127, contraste = 100%. b) Configuración: brillo = -127, contraste = 100%. c) Configuración: brillo = 0, contraste = 70. d) Configuración: brillo = -30, contraste = 50%. (Imagen Mejorada)
(Fuente: Propia)

Con el preprocesamiento de imagen terminado, es posible continuar con la detección de bordes. Para esto se emplean las condiciones de brillo y contraste presentadas en las Figura 3.5 (d), debido a su aspecto visual homogéneo. En la Figura 3.6 se puede observar varias imágenes con una marcación de bordes de color dorado, que presentan diferentes valores de umbral máximo y mínimo. Estos resultados corresponden al método I de la detección de bordes. La Figura 3.7 presenta al eje hueco con distintos valores umbral y una marcación de color verde, correspondiente a los resultados del método II.



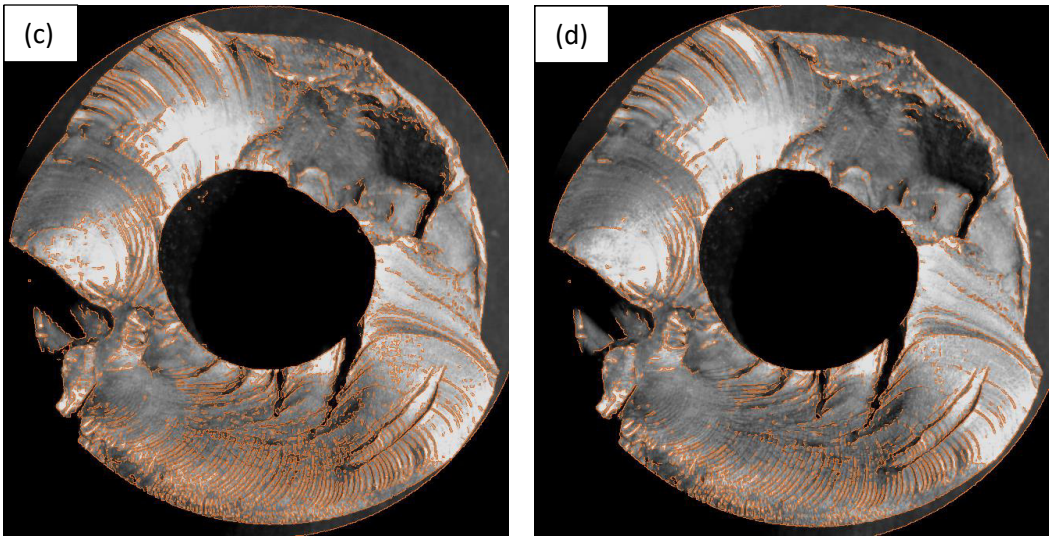


Figura 3.6. Resultados de la Función Detección de Bordes (Método I). a) Valores umbral: Tmáx = 255, Tmín = 0. b) Valores umbral: Tmáx = 200, Tmín = 50. c) Valores umbral: Tmáx = 150, Tmín = 100. d) Valores umbral: Tmáx = 210, Tmín = 140. (Imagen Bordes)
(Fuente: Propia)

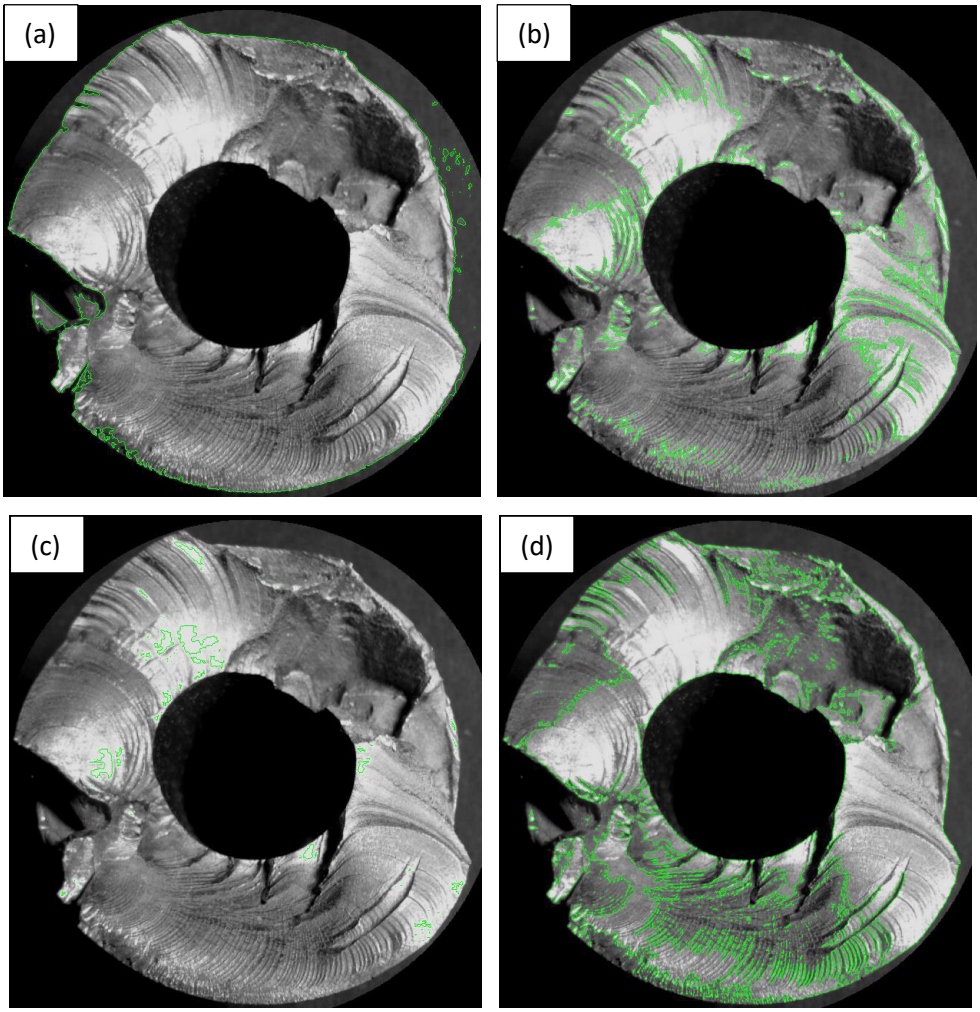


Figura 3.7. Resultados de la Función Detección de Bordes (Método II). a) T= 70 b) T= 175 c) T= 224 d) T= 120. (Imagen Bordes)
(Fuente: Propia)

La marcación de áreas se realiza según el procedimiento presentado en el capítulo anterior. Además, se considera la marcación de bordes de la Figura 3.6 (d) para el método I, la Figura 3.7 (d) para el método II en las funciones posteriores.

La Figura 3.8 (a) y 3.9 (a) presentan la selección del área interna en color rosado. En la Figura 3.8 (b) y 3.9 (b) se puede observar la selección del área total de fractura en color azul. Finalmente, en la Figura 3.8 (c) y 3.9 (c) se muestra el área de fractura final en color rojo.

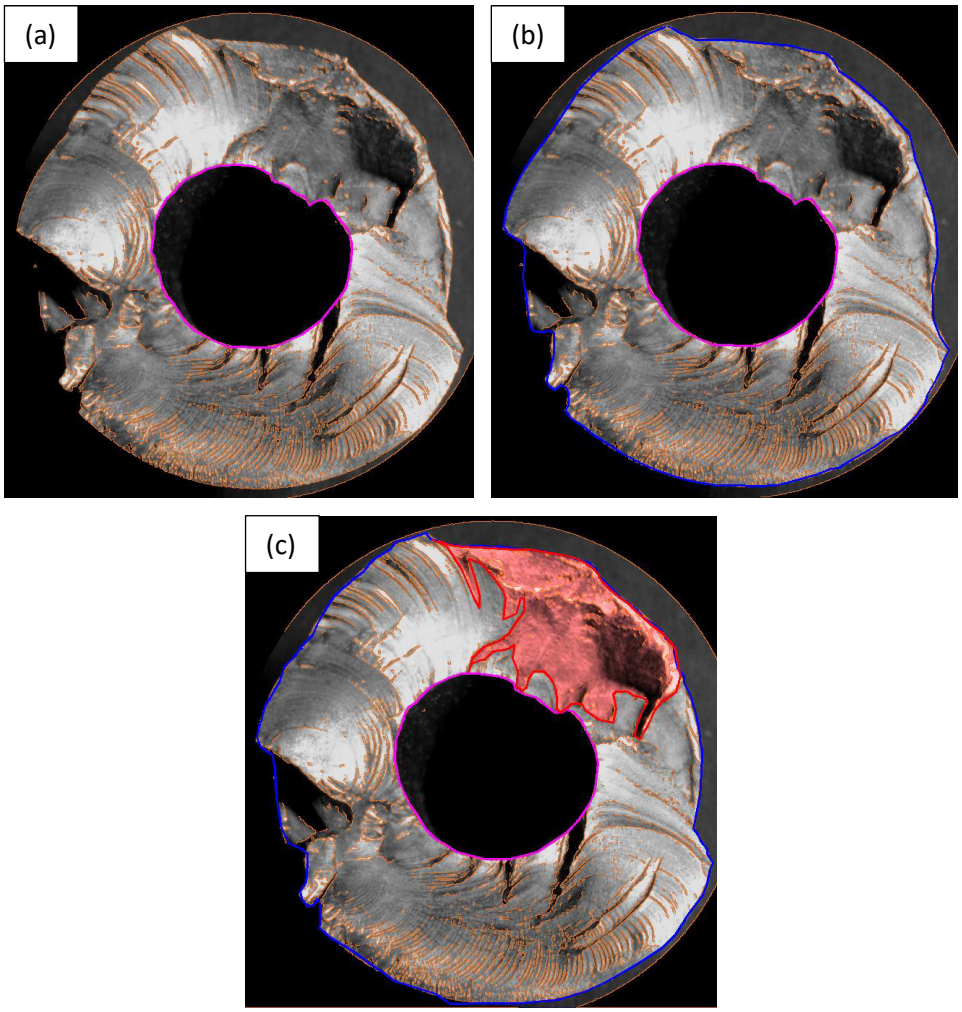


Figura 3.8. Resultados de la Función Áreas de Fractura (Método I). a) Área interna b) Área total de Fractura c) Área de Fractura Final. (Imagen Áreas)
(Fuente: Propia)

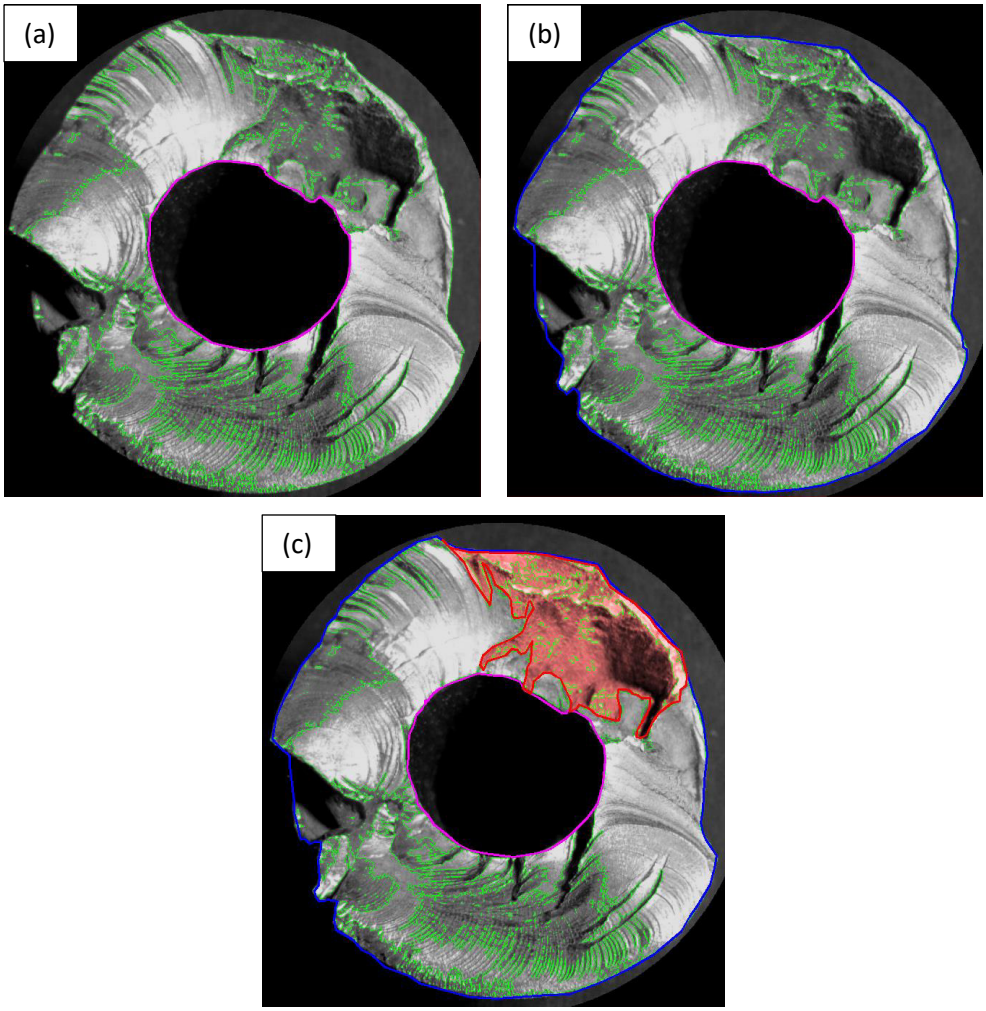


Figura 3.9. Resultados de la Función Áreas de Fractura (Método II). a) Área interna b) Área total de Fractura c) Área de Fractura Final. (Imagen Áreas)
(Fuente: Propia)

La Tabla 3.1 presenta los porcentajes correspondientes a las áreas de fractura obtenidos para el método I de detección de bordes. La Tabla 3.2 muestra los resultados para el método II. Estos valores han sido obtenidos en función de la selección realizada.

Tabla 3.1. Porcentajes de las áreas de fractura (Método I).

Área	Porcentaje
Fatiga	86.76
Fractura final	13.24

(Fuente: Propia)

Tabla 3.2. Porcentajes de las áreas de fractura (Método II).

Área	Porcentaje
Fatiga	86.56
Fractura final	13.44

(Fuente: Propia)

La Figura 3.10 (a) y 3.11 (a) muestran la marca de trinquete en color amarrillo. A continuación, la Figura 3.10 (b) y 3.11 (b) indican la progresión y punto de origen de la fractura en color morado. Puede existir más de un origen en un solo elemento. Finalmente, la Figura 3.10 (c) y 3.11 (c) presentan la bisectriz correspondiente en color naranja.

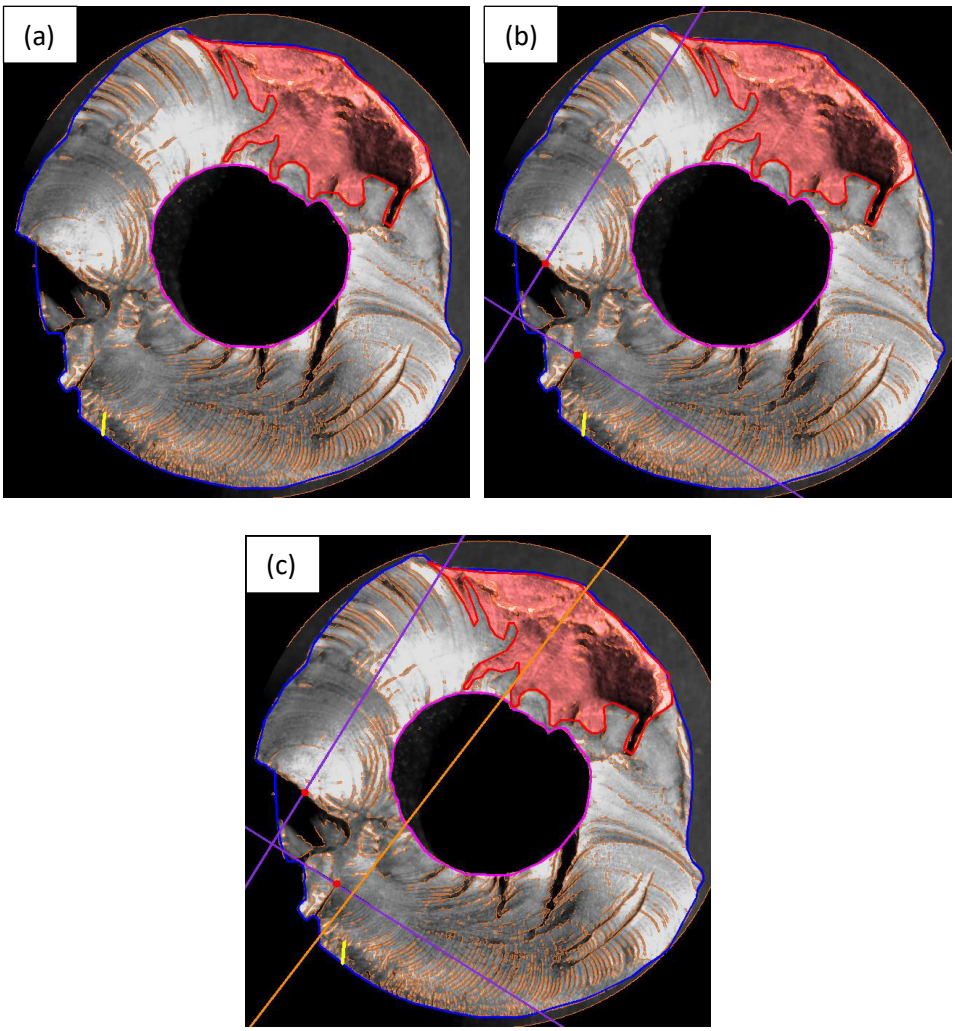


Figura 3.10. Resultados del Método I de: a) Función Marcas de trinquete b) Función Seleccionar Origen c) Función Bisectriz (Imagen Trinquete, Origen y Bisectriz).
(Fuente: Propia)

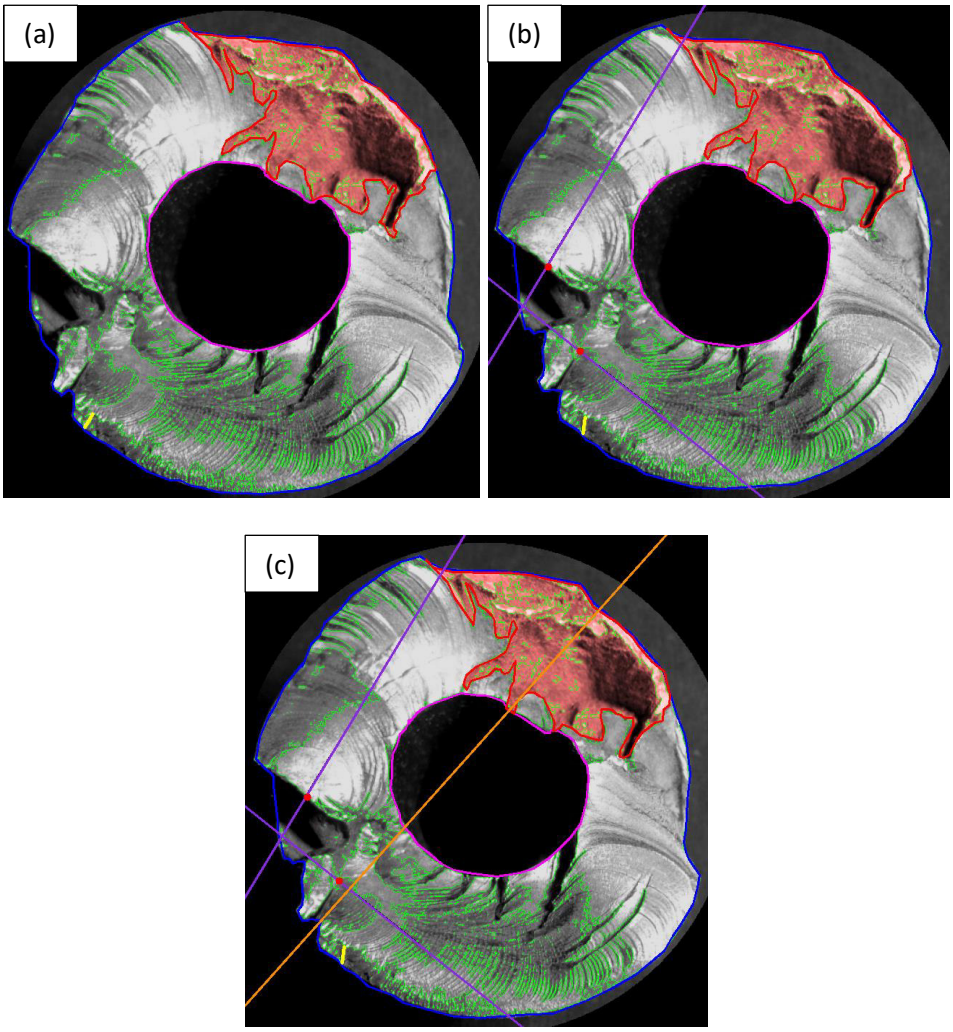


Figura 3.11. Resultados del Método II de: a) Función Marcas de trinquete b) Función Seleccionar Origen c) Función Bisectriz (Imagen Trinquete, Origen y Bisectriz).
(Fuente: Propia)

A partir de los resultados obtenidos en el preprocesamiento y procesamiento de las imágenes, se efectúa el cuestionario final. La Tabla 3.3 muestra las respuestas del cuestionario final para la imagen obtenida mediante el método I de detección de bordes. Las capturas correspondientes para cada pregunta del método I se presentan en el Anexo II. Además, en el Anexo III se muestra una captura del informe obtenido. La Figura 3.12 y 3.13 muestran la ventana auxiliar con los resultados del cuestionario para ambos métodos.

Tabla 3.3. Respuesta del cuestionario final (Método I).

Pregunta	Respuesta Seleccionada
Pregunta #1	Opción A
Pregunta #2	Opción B
Pregunta #3	Opción A

Pregunta #4	Opción A
Pregunta #5	Opción B
Pregunta #6	Opción A
Pregunta #7	Opción B

(Fuente: Propia)

Figura 3.12. Resultados del cuestionario para el método I.
(Fuente: Propia)

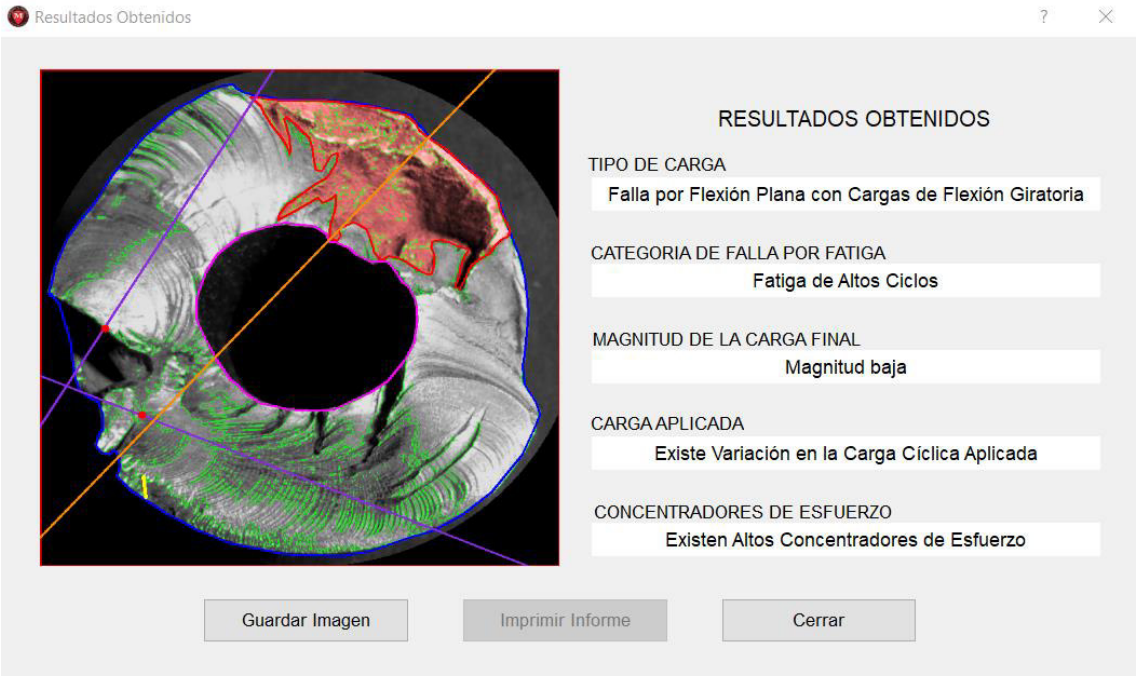


Figura 3.13. Resultados del cuestionario para el método II.
(Fuente: Propia)

3.1.1 Diagnóstico de falla

En función de los criterios anteriormente mencionados y la información obtenida se puede justificar que el elemento analizado ha fallado debido a un proceso de fatiga. A continuación, se presentan los criterios correspondientes:

Figura 3.15. Caso de Estudio
(Fuente: Propia)

- La presencia de marcas de progresión indica que la carga cíclica aplicada era variable. Además, presentan buena definición, esto es un indicativo de que el elemento estuvo en servicio durante un alto número de ciclos. La progresión de estas marcas coincide con los orígenes identificados.
- El porcentaje del área de fractura final obtenida muestra que, el elemento estuvo sometido a cargas cíclicas bajas. Debido a la existencia de varios orígenes no es posible utilizar el criterio de la bisectriz especificado en la metodología.
- La presencia de varios orígenes indica que el elemento estuvo sometido a una combinación de cargas de flexión plana y rotatoria, sumado a la presencia de concentradores de esfuerzos.

3.1.2 Porcentajes de error

Para establecer los porcentajes de error se aplicó el procedimiento descrito en la metodología conforme a la comparación de los resultados del programa AutoCAD con los datos obtenidos por el software desarrollado. En la Tabla 3.4 se presentan los resultados del programa AutoCAD. En la Tabla 3.5 y 3.6 se muestra la tabulación de datos y porcentajes de error, para los dos métodos de detección de borde.

Tabla 3.4. Áreas de los contornos seleccionados en el programa AutoCAD

	Método I	Método II
Zonas	Área [mm2]	Área [mm2]
Perímetro 1	4221.28	4218.96
Perímetro 2	738.36	732.36
Perímetro 3	564.44	556.61

(Fuente: Propia)

Tabla 3.5. Porcentaje de error obtenido entre AutoCAD y el software desarrollado para el método I de detección de bordes.

Áreas	Área [mm2]	Porcentaje AutoCAD	Porcentaje Software	Error
Área de fatiga	3339.92	84.04	86.76	3.24
Área de fractura final	634.49	15.96	13.24	17.06
Área de fractura total	3974.41			

(Fuente: Propia)

Tabla 3.6. Porcentaje de error obtenido entre AutoCAD y el software desarrollado para el método II de detección de bordes.

Áreas	Área [mm2]	Porcentaje AutoCAD	Porcentaje Software	Error
Área de fatiga	3348.24	83.79	86.56	3.30
Área de fractura final	647.57	16.21	13.44	17.06
Área de fractura total	3995.81			

(Fuente: Propia)

3.2. Discusión

3.1.3 Preprocesamiento de imágenes

El escalado automático mostrado en la Figura 3.1 permite generar una imagen de dimensiones estándar. Al estandarizar las variables de la imagen de entrada es posible uniformizar el proceso, de manera que la relación de aspecto no se vea afectada. Los resultados posteriores tendrán las mismas dimensiones.

Para seleccionar el área del elemento es importante marcar dos puntos opuestos que rodeen acertadamente la superficie de fractura como se puede evidenciar en la Figura 3.2 (a). La utilización de imágenes con demasiada información digital poco relevante genera un mayor gasto de recursos computacionales entorpeciendo el desarrollo del software. Para evitar esta situación es necesario retirar el fondo de la imagen, como se muestra en la Figura 3.2 (b).

La naturaleza de este proyecto requiere analizar imágenes en las cuales el contraste entre diferentes zonas sea elevado. Considerando el trabajo de Aníbal Silva “Desarrollo del código de programación para procesamiento de imágenes aplicada en fundiciones

nodulares" y el ASM Handbook Vol12 Fractografia, es preferible analizar una imagen en escala de grises que en un modelo BGR, debido al contraste generado. La Figura 3.3 permite observar la conversión realizada. Las zonas más obscuras presentes en las imágenes indican deformaciones en la superficie de falla, es decir, permiten identificar de mejor manera las marcas características de la fractura.

Para la correcta aplicación de este software es necesario brindar al usuario todas las herramientas visuales para una correcta interpretación. Debido a esto se aplica un acercamiento como el mostrado en la Figura 3.4. El aumento de la zona de fractura en las imágenes permite observar de mejor manera los parámetros analizados.

La Figura 3.5 muestra diferentes imágenes, con valores de brillo y contraste distintos. Los parámetros fueron modificados con el objetivo de que puedan otorgar una mayor diferenciación entre los tonos claros y obscuros de las imágenes. De manera que la identificación de bordes sea más eficiente. En la Figura 3.5 (a) se puede observar que la combinación empleada resulta en una imagen con poca información visual relevante. La combinación utilizada en la Figura 3.5 (b) permite apreciar una mayor distinción de la marcación. Si bien existe más información, esta no puede observarse debido a que las marcas se pierden en las zonas claras y oscuras. La Figura 3.5 (c) indica zonas demasiado claras, que dificultan la identificación. Finalmente, la Figura 3.5. (d) es la que presenta un equilibrio de ambas variables, ya que se pueden ver con claridad las marcas.

3.1.4 Procesamiento de imágenes

La Figura 3.6 muestra los resultados de la detección de bordes con el método I. La Figura 3.6 (a) y 3.6 (b) muestran valores umbrales alejados entre sí. Esto ocasiona que el intervalo de marcación sea demasiado grande, de manera que se detecta contaminación indeseable (Ruido). Conforme estos umbrales van reduciendo su diferencia, se puede apreciar una menor cantidad de ruido, como en la Figura 3.6 (c). Para detectar bordes de calidad, es necesario mantener el valor umbral máximo cercano a 255, y el valor mínimo en un punto medio para evitar contaminación. La combinación ideal dependerá de que tan definidas se encuentren las zonas y marcas de interés para los dos métodos.

A continuación, en la Figura 3.7 se muestran los resultados para el método II. En este caso mientras mayor sea el valor umbral, menos ruido va a detectar. Pero al ser tan restrictivo, elimina ciertos bordes con una intensidad intermedia. La Figura 3.7 (a)

corresponde al umbral más bajo en el cual se pueden detectar bordes. Por esta razón presenta una marcación débil. En cambio, la Figura 3.7 (c) corresponde al máximo valor umbral que detecta bordes. No existe una marcación abundante debido a que en la imagen existen pocos valores de intensidad de esa magnitud. En la Figura 3.7 (b) se muestra un umbral intermedio.

Al comparar los dos métodos utilizados, se puede identificar ciertas ventajas de uno sobre otro. La sensibilidad de detección permite identificar más o menos bordes. Este parámetro es distinto en cada método. Considerando el método I, la presencia de una zona de incertidumbre incrementa la holgura del intervalo de detección. Es decir, permite detectar una mayor cantidad de bordes en la imagen debido a las aproximaciones que se realizan en esta zona intermedia.

El método II no cuenta con un intervalo de aproximación. El valor umbral actúa como una barrera que no permite generar ningún borde por debajo de este. Por esta razón, se obtiene una cantidad de bordes menor. La Tabla 3.7 muestra la cantidad de bordes identificados para cada método. Se considera la Figura 3.6 (d) y 3.7 (d) para el método I y II respectivamente, ya que estas imágenes presentan la marcación de bordes más definida.

Tabla 3.7. Bordes detectados por método.

Función	Bordes detectados
Método I	1203
Método II	591

(Fuente: Propia)

A partir de estos resultados se puede identificar la diferencia existente entre el método I y II. El método I detecta más bordes, por ende, es el más sensible. Es importante mencionar que estos valores varían en función de los umbrales y la configuración de brillo y contraste seleccionada, pero la tendencia se mantiene.

Otro aspecto importante de considerar a la hora de seleccionar el método para detectar bordes es la velocidad de aplicación. La Tabla 3.8 presenta el tiempo que demora el software en implementar cada método, para la Figura 3.6 (d) y 3.7 (d).

Tabla 3.8. Tiempo de implementación por método.

Función	Tiempo [s]
Método I	0.515
Método II	0.154

(Fuente: Propia)

Los resultados muestran que la implementación del método II es más corta que la del método I. Esto se debe a que no presenta cálculos adicionales, a diferencia del método I que debe considerar la aproximación de bordes en la zona de incertidumbre.

La Figura 3.8 muestra la selección de las distintas áreas de fractura seleccionadas. Esta operación se realiza con el objetivo de obtener el porcentaje del área de fractura final y relacionarlo con los criterios presentes en el cuestionario final.

En la Tabla 3.1 se muestran los porcentajes de las áreas de fractura para el método I de detección de bordes. Se obtiene un 13,24% para el área de fractura final con respecto al área total, lo que indica que la carga final aplicada al elemento fue de magnitud baja. La Tabla 3.2 indica los porcentajes para el método II. Se tiene un valor de 13,44%. Similar al obtenido con el método I. De esta forma es posible identificar que independientemente del método, el error en la obtención de los porcentajes depende en absoluto del usuario y su capacidad de identificar las diferentes áreas de la fractura. Para esto se debe establecer un rango de aceptación para el error, en donde se condicione una correcta selección de dichas áreas.

La Tabla 3.9 muestra los errores obtenidos variando el rango de precisión en un milímetro para cada área de análisis. Estos resultados indican que, si el usuario varía su selección, ya sea hacia dentro o fuera del contorno en un milímetro, el error obtenido para el área de fractura final puede variar entre un 14% a un 15% mientras que para el área de fractura total varia de entre 4% a 5%.

Tabla 3.9. Porcentajes de error de selección de áreas de análisis.

Áreas	Porcentaje de error	
	Método I	Método II
Área de fractura final (1 mm dentro del contorno)	14.05%	14.05%
Área de fractura final (1 mm fuera del contorno)	14.98%	14.35%
Área de fractura total (1 mm dentro del contorno)	4.61%	4.58%
Área de fractura total (1 mm afuera del contorno)	4.61%	4.58%

(Fuente: Propia)

Finalmente, en la Figura 3.10 (a) es posible identificar las marcas de trinquete localizadas en la parte inferior izquierda del eje. Para más información revisar el marco teórico. La Figura 3.10 (b) contiene la línea que indica el origen y la progresión de la

fisura. De manera que se puede obtener el desplazamiento de las marcas de progresión, lo cual facilita la identificación del origen. En la Figura 3.10 (c) se presenta la bisectriz de la zona de fractura final. El ángulo mencionado en la metodología se calcula automáticamente y se vincula con los resultados obtenidos una vez resuelto el cuestionario por el usuario.

3.1.5 Análisis de diagnóstico

Debido a la flexión rotatoria en el elemento se puede argumentar que era parte de un reductor de velocidades o estaba acoplado a poleas para transmisión de movimiento. Además, su integridad se vio afectada por la presencia de concentradores de esfuerzo, debido a esto presenta varios orígenes. La variación en la carga cíclica permitió un desplazamiento lento de la grieta, razón por la cual se produce una fatiga por altos ciclos de servicio. Se descarta sobrecarga, ya que el porcentaje del área de fractura final es menor al 50% del área total de fractura.

En el párrafo anterior se describen las conclusiones obtenidas a partir de los criterios que entrega el software desarrollado. Como se puede observar, la información permite orientar el análisis de falla a ciertas áreas. En este caso en particular, se identifican problemas en el mantenimiento y diseño del elemento. Como se menciona en el marco teórico existe una gran cantidad da razones por las cuales se puede generar una falla. La utilización de este software como herramienta auxiliar permite reducir considerablemente el campo de análisis, mediante la identificación de marcas y zonas de la fractura.

3.2 Validación

En la Tabla 3.5 y 3.6 se presentan los porcentajes de error que pertenecen al cálculo de área de fatiga y de fractura final. Se observa que para el método I en el área de fatiga se obtiene un error de 3.24% y en el área de fractura final un error de 17.06%. Estos valores de error son bajos para el procesamiento realizado. De igual forma para el método II se presentan errores de 3.30% y de 17.06% a para el área de fatiga y de fractura final respectivamente.

A partir de los resultados obtenidos, se puede identificar que en el caso de ejecutar operaciones adicionales como lo son la adición y sustracción de áreas se genera un error mayor. Esto se debe principalmente a la forma de interacción del programa con la obtención de áreas. Al aplicar cualquiera de las operaciones mencionadas sobre una

misma área el error absoluto del proceso aumenta de forma drástica. Debido a esto el error que se presenta para un solo contorno, como el del área de fractura final, es menor. En función de los resultados obtenidos se puede identificar que el error será mínimo cuando exista un único contorno. De igual forma la diferencia de los porcentajes de error entre ambos métodos de detección de bordes son mínimos como se muestran en la Tabla 3.5 y 3.6. Por esta razón, sin importar el método, es posible alcanzar porcentajes de áreas válidos.

En el trabajo de Diana Puga “Desarrollo de un código de programación en lenguaje Python para el estudio de zonas forestales” se menciona que un error del 20% para este tipo de aplicaciones es aceptable (Puga, 2019). En función de los distintos valores de error obtenidos, se considera que la metodología empleada para la selección de áreas en el software es válida, debido a que estos no sobrepasan el límite establecido por la bibliografía.

4 CONCLUSIONES

- Se desarrolló un software en lenguaje de programación Python acompañado de una interfaz de usuario para el análisis de falla en elementos de acero sometidos a elevados ciclos de servicio y bajos esfuerzos, acorde a la metodología establecida a partir de la revisión bibliográfica correspondiente.
- La utilización de este software como herramienta auxiliar permite obtener criterios y resultados visuales que facilitan el análisis de falla. Esto se debe a que el campo de estudio se ve reducido debido a la interpretación de la información obtenida.
- La escala de grises permite intensificar el contraste presente en las imágenes. De manera que es más sencillo identificar y diferenciar rasgos superficiales. Además, de representar ahorro en recursos computacionales.
- Los valores establecidos para la configuración de brillo y contraste y detección de bordes dependen en su totalidad de la imagen. La adecuada selección de estos parámetros permite mejorar el aspecto visual, para identificar y obtener rasgos superficiales característicos.
- Las estrías por fatiga no pueden ser consideradas en este estudio, debido a su naturaleza microscópica. El análisis de estas marcas permite obtener información más precisa en cuanto al tiempo de servicio del elemento.
- La interfaz gráfica desarrollada es amigable con el usuario. Posee visores de ayuda que presentan información visual o escrita relevante, sobre cada una de las funciones implementadas.
- El software puede ser difundido de manera sencilla a través de la comunidad educativa, debido a que ha sido desarrollado en software libre. Puede ser instalado en cualquier computador.
- A partir del error obtenido al comparar los valores de áreas del programa AutoCAD con los del software desarrollado, se puede observar que se encuentra dentro del límite aceptable según la bibliografía. Esto otorga validez a la metodología empleada.
- La selección de las áreas de fractura es un proceso que posee alta variabilidad debido a la creación del contorno por el usuario. El rango de porcentaje establecido para el error tiene la holgura necesaria para mitigar esta variación.
- Los criterios obtenidos como resultado del software corresponden a las principales características que se pueden analizar en la superficie de

fractura. Estas proveen información necesaria para determinar la posible causa de falla del elemento.

- El software desarrollado ha sido acoplado con funciones de ayuda, que permite mejorar la experiencia y facilitar el uso. De manera que puede ser utilizado como una herramienta didáctica para la enseñanza, tanto para profesores como estudiantes.

5 RECOMENDACIONES

- Para el correcto uso del software implementado es necesario revisar el "MANUAL DE USUARIO DEL SOFTWARE FATIGUE FAILURE". En este documento se presenta el funcionamiento de cada módulo.
- Generar una base de datos de fotografías después de haber realizado el procesamiento correspondiente. Esto permite identificar patrones de falla adicionales, que pueden ser recurrentes en elementos similares.
- Analizar el impacto económico, tecnológico y productivo que representa la implementación de este tipo de software para procesamiento de imágenes en industrias. Al identificar otras ventajas, es posible impulsar el desarrollo y optimización de esta metodología.
- Implementar funciones y criterios adicionales para diversos materiales utilizados en la industria. Esto permite abarcar un mayor campo de estudio.
- Investigar acerca de nuevos métodos para la detección de bordes. Es posible implementar nuevas alternativas para la detección, con el objetivo de aumentar el rango de calidad fotográfica permisible en el software.
- Considerar la revisión de otros tipos de cargas a las que puede estar sometido el elemento. Al considerar estos conceptos es posible vincular criterios adicionales en el software incrementando su versatilidad.
- Vincular el estudio microscópico de estrías generadas por fatiga. Estas marcas superficiales otorgan información adicional en cuanto a la propagación de la fractura.

6 REFERENCIAS BIBLIOGRÁFICAS

[1] Acuña., M. I. R. J. (n.d.). FRACTOGRAFÍA APLICACIONES AL ANÁLISIS DE FALLAS. *CNA - ARGENTINA*.

[2] ASM International. Handbook. (1998). ASM Handbook: Volume 12: Fractography. *Metals Handbook*, *12*, 857. http://books.google.com.hk/books?id=eC-Zt1J4oCgC

[3] Budynass G., R. . (2014). Diseño en Ingeniería Mecánica de Shigley. In *Igarss 2014* (octava, Issue 1). Mc Graw Hill.

[4] Campbell, F. C. (2012). Fatigue and Fracture: Understanding the Basics. *Journal of Chemical Information and Modeling*, *53*(9), 699.

[5] Cash, J. A. (1950). *Antigua Fábrica de aeronaves*. https://cutt.ly/oTtbQ3F

[6] Challenger, I., Díaz, Y., & García, R. B. (2014). El lenguaje de programación Python/The programming language Python. *Redalyc*, *XX*, 1–13. https://www.redalyc.org/pdf/1815/181531232001.pdf

[7] D.C. Williams, & R.L. Naro. (2019). *Eliminación de defectos de microporosidades sub-superficiales en aceros de aleación agregando ferroselenio.* https://www.asi-alloys.com/pdf/SS_2019_Fall-SPA_ASI.pdf

[8] Díaz Cortés, C. L., Yazo Salgado, O. E., & Ayala Pineda, D. A. (2018). *Análisis de falla en elementos mecánicos de sujeción* [Fundación Universitaria Los LIbertadores]. https://repository.libertadores.edu.co/bitstream/handle/11371/1723/diaz_camila_2018.pdf?sequence=1&isAllowed=y

[9] *Función de brillo y contraste—Ayuda | ArcGIS for Desktop*. (n.d.). Retrieved March 6, 2022, from https://desktop.arcgis.com/es/arcmap/10.3/manage-data/raster-and-images/contrast-and-brightness-function.htm

[10] General Dinamycs. (2021). *General Dynamics F-111 AARDVARK*. https://misistemasolar.com/general-dynamics-f-111-aardvark/

[11] Hz, S., Vijayarani, S., & Vinupriya, M. M. (2007). Performance Analysis of Canny and Sobel Edge Detection Algorithms in Image Mining Related papers Performance Analysis of Canny and Sobel Edge Detection Algorithms in Image Mining. *International Journal of Innovative Research in Computer and Communication Engineering (An ISO*, *3297*. www.ijircce.com

[12] IEC. (1990). *IEC 60050 - Vocabulario electrotécnico internacional - Detalles para el número de IEV 192-03-01: "falla."* https://www.electropedia.org/iev/iev.nsf/display?openform&ievref=192-03-01

[13] Macleod Fiona, & Richardson Stephen. (2018). *Piper Alpha: The Disaster in*

Detail - Features - The Chemical Engineer. https://www.thechemicalengineer.com/features/piper-alpha-the-disaster-in-detail/

[14] Maldonado, J. (1996). *Aceros y sus Aplicaciones* [Universidad Autónoma de Nuevo León]. http://eprints.uanl.mx/421/1/1020118272.PDF

[15] *math-Funciones matemáticas*. (2021). https://docs.python.org/es/3/library/math.html#

[16] McCall, J., & French, P. M. (1977). *Metallography in Failure Analysis*.

[17] Miró, G. (2021). *La influencia de la lubricación en los elementos de desgaste de maquinaria industrial*. https://blog.atten2.com/elementos-de-desgaste-de-maquinaria-industrial

[18] *NumPy User Guide*. (2012). 1–93.

[19] *OpenCV: Introduction*. (2022). https://docs.opencv.org/4.x/d1/dfb/intro.html

[20] Puga, D. S. (2019). *Desarrollo de un código de programación en lenguaje Python para el estudio de zonas forestales* [Escuela Politécnica Nacional]. http://bibdigital.epn.edu.ec/handle/15000/20184

[21] Rausand, M., & Ølen, K. (1996). The basic concepts of failure analysis. *Reliability Engineering and System Safety*, *53*(1). https://doi.org/10.1016/0951-8320(96)00010-5

[22] Rheinmentall. (2021). *Daños de válvulas y sus causas · Technipedia · Motorservice*. https://www.ms-motorservice.com/es/tecnipedia/post/danos-de-valvulas-y-sus-causas/

[23] Rumiche, F., & Idacochea, E. (n.d.). *Estudios de Caso de Fallas y Accidentes en Gasoductos y Oleoductos*. *2005*, 11. https://cutt.ly/qTtZTir

[24] Sachs, N. W. (2016). Practical Plant Failure Analysis. In *Practical Plant Failure Analysis*. CRC Press. https://doi.org/10.1201/9781420020007

[25] Tovar S., G. (1999). Análisis de falla de componentes de ingeniería. *Revista de Ingeniería*, *9*. https://doi.org/10.16924/revinge.9.10

[26] Viera, G. (2017). Procesamiento de imágenes usando OpenCV aplicado en Raspberry Pi para la clasificación del cacao. *Thesis*, 136. https://pirhua.udep.edu.pe/bitstream/handle/11042/2916/IME_218.pdf?sequence=1&isAllowed=y

[27] Wulpi, D. J. (1985). Understanding How Components Fail. In *Understanding How Compon Fail*. ASTM. https://doi.org/10.1016/0026-0800(86)90014-5

7 ANEXOS

ANEXO I.

FLUJOGRAMA DE SELECCIÓN DE CARGA

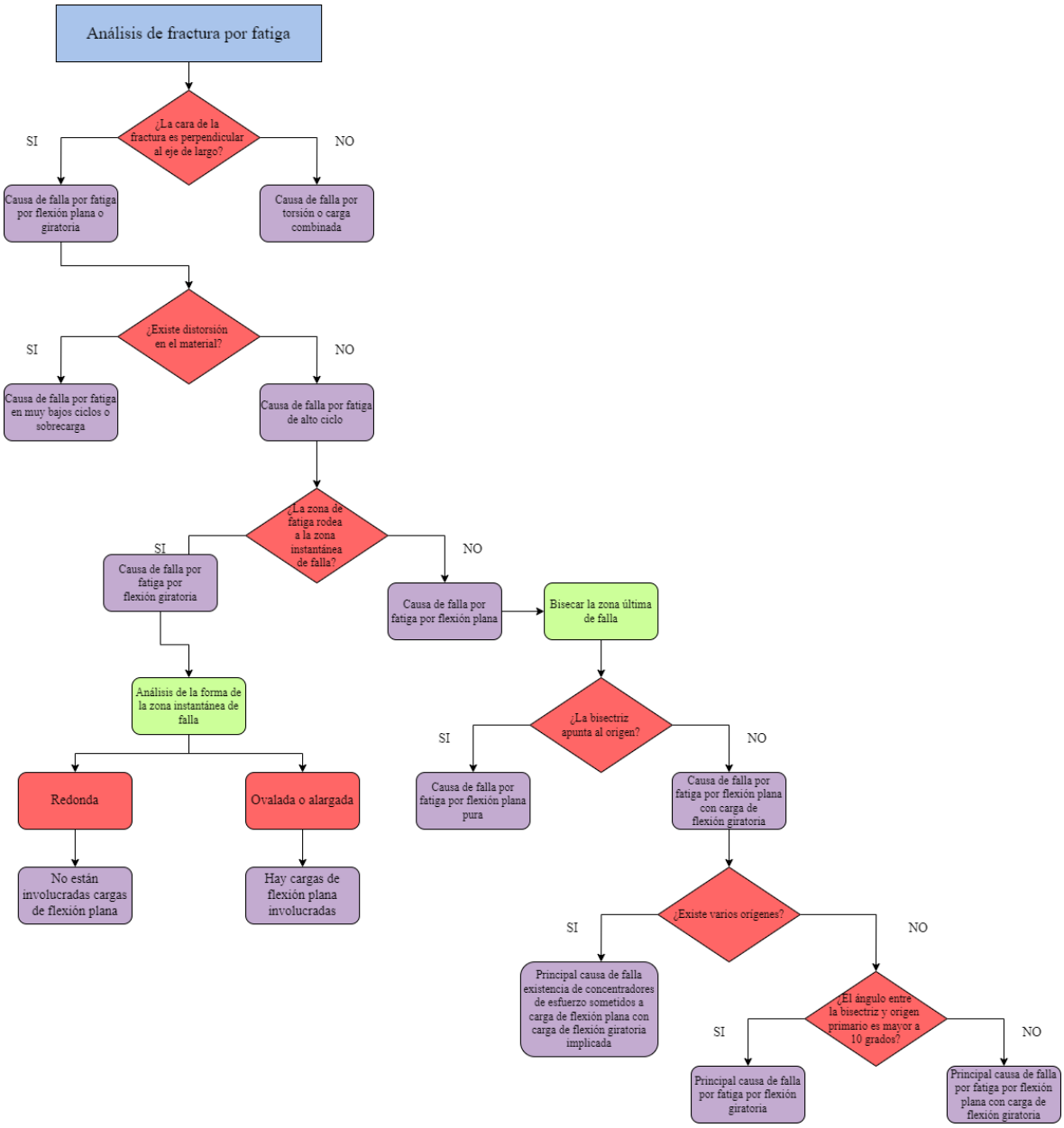


Figura 1. Flujograma de selección de cargas actuantes en elementos analizados con las preguntas que se realizan en el cuestionario.
(Fuente: Propia)

ANEXO II.

PREGUNTAS DEL CUESTIONARIO

Figura 1. Pregunta #1.
(Fuente: Propia)

Figura 2. Pregunta #2.
(Fuente: Propia)

Figura3. Pregunta #3.
(Fuente: Propia)

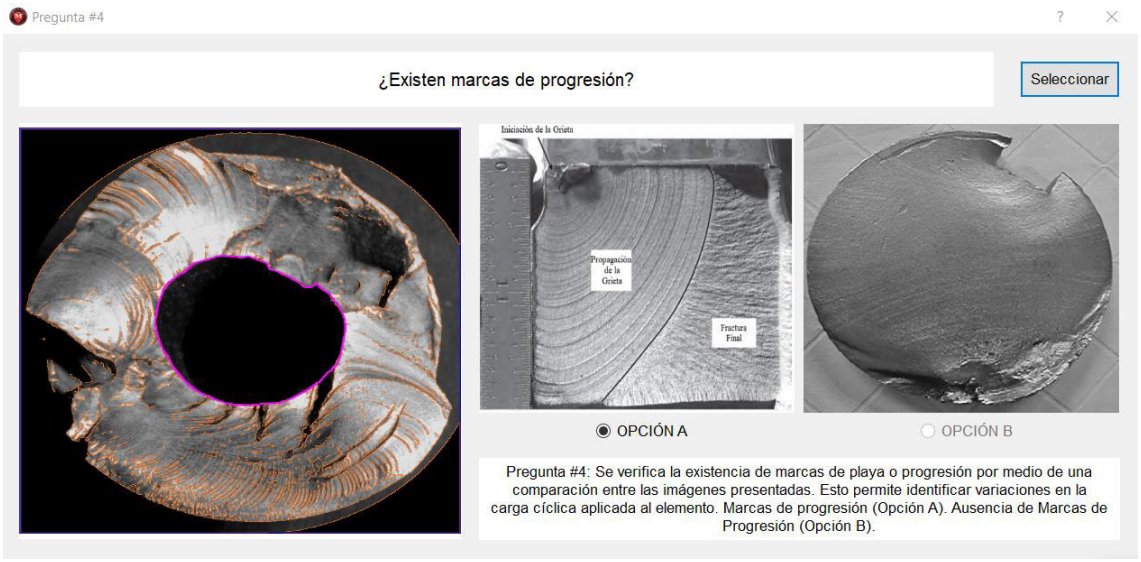


Figura 4. Pregunta #4.
(Fuente: Propia)

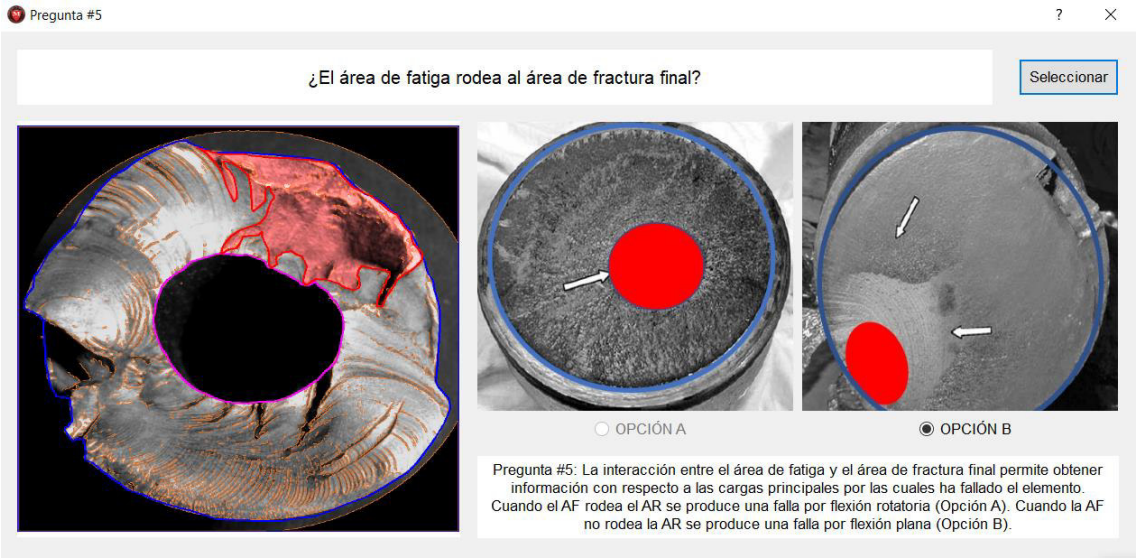


Figura 5. Pregunta #5.
(Fuente: Propia)

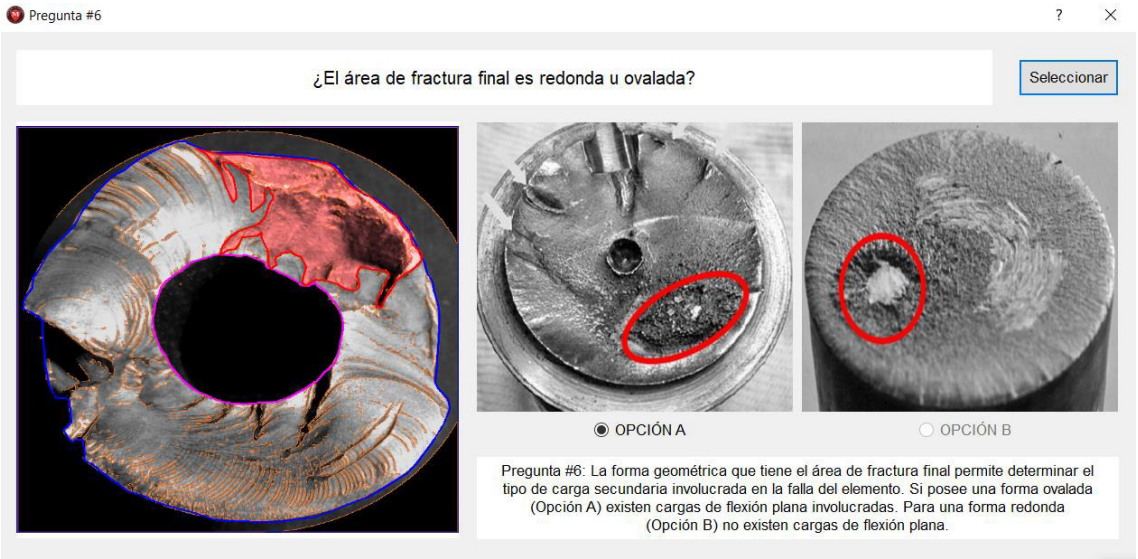


Figura 6. Pregunta #6.
(Fuente: Propia)

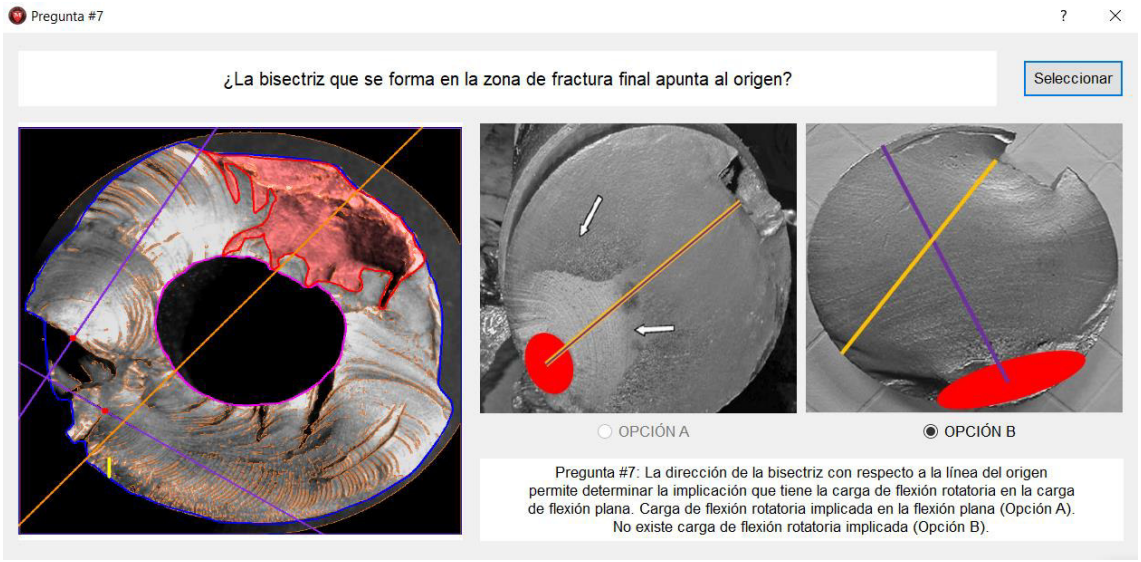


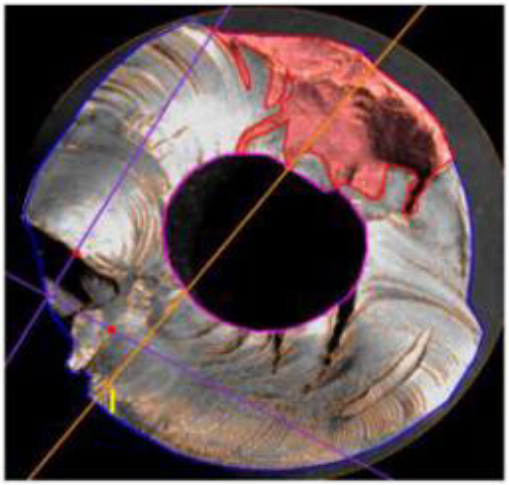
Figura 7. Pregunta #7.
(Fuente: Propia)

ANEXO III.

INFORME GENERADO

INFORME DEL SOFTWARE FIM FATIGUE FAILURE

PARAMETRO	RESULTADO
-TIPO DE CARGA:	Falla por Flexión Plana con Cargas de Flexión Giratoria
-CATEGORIA DE FALLA POR FATIGA:	Fatiga de Altos Ciclos
-MAGNITUD DE CARGA FINAL:	Magnitud baja
-CARGA APLICADA:	Existe Variación en la Carga Cíclica Aplicada
-CONCENTRADORES DE ESFUERZOS:	Existen Altos Concentradores de Esfuerzo
-VALOR DE BRILLO:	-30
-VALOR DE CONTRASTE:	50
-METODO DE DETECCION DE BORDES:	Método I
-PORCENTAJE AREA DE FRACTURA FINAL:	13.24
-PORCENTAJE AREA DE FATIGA:	86.76



ANEXO IV.

MANUAL DE USUARIO DEL SOFTWARE FATIGUE FAILURE

Introducción

El software para procesamiento de imágenes ha sido desarrollado como herramienta auxiliar en el análisis de falla. Mediante diferentes funciones es posible modificar las condiciones visuales de fotografías. Esto se realiza para poder detectar marcas características de las fallas por fatiga e identificar criterios para ejecutar el análisis respectivo.

Ha sido diseñado en el lenguaje de programación Python. Debido a que es muy potente para aplicaciones de ingeniería y posee una sintaxis amigable con el usuario. Además, es un software libre, lo que facilitad su desarrollo y difusión.

Requerimientos técnicos

Considerar los siguientes requerimientos antes de iniciar el software:

- Resolución de pantalla 1920 x 1080.
- Sistema operativo Windows 10.

Inicio del software

Para poder iniciar adecuadamente el software es necesario considerar lo siguiente:

- Cerrar todas las ventanas abiertas del escritorio.
- Hacer doble clic en el documento ejecutable.
- Hacer clic en “Show Details” y considerar las restricciones. Seguido de esto dar clic en aceptar, como se muestra en la Figura 2.24.

Uso y funcionamiento del software

Sección I

Este proceso inicia al hacer clic en el botón “Abrir Imagen”, este permite cargar una imagen desde el escritorio. En caso de ser necesario es posible introducir otra imagen, presionar

el botón “Limpiar”. Seguido de esto es necesario seleccionar la sección del elemento. Para una sección circular, con los dos puntos seleccionados (A y B) se forma una línea ($\overline{AB}$) y, mediante operaciones matemáticas, se obtiene el centro (Cx, Cy) y la longitud (d) de esta. La longitud se divide en dos para hallar así el radio (R) del círculo que delimite la zona de análisis. Una vez que han sido obtenidos estos parámetros se genera el círculo de radio R en el centro (Cx, Cy) como se muestra en la Figura 1.

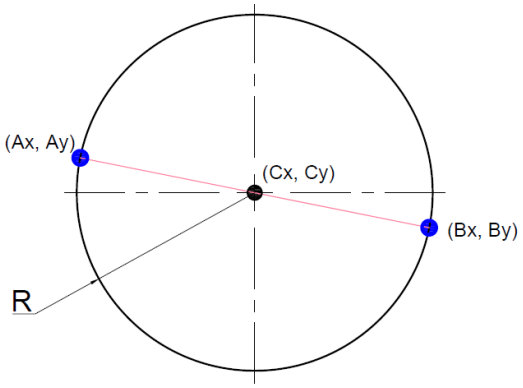


Figura 1. Sección circular.
(Fuente: Propia)

En caso de ser una sección rectangular el usuario debe hacer clic izquierdo con el ratón en dos vértices opuestos. A partir de los vértices seleccionados (A y B) se obtiene las coordenadas de los otros dos vértices restantes para formar el rectángulo (A, B, C y D). El procedimiento se muestra en la Figura 2.

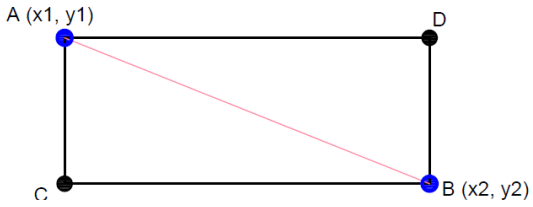


Figura 2. Sección rectangular.
(Fuente: Propia)

Al finalizar la selección se debe presionar el botón Esc. A continuación, es necesario hacer clic en el botón “Recortar” para retirar el fondo de la imagen. Seguido se efectúa la conversión automáticamente del modelo BGR a escala de grises con el botón “Escala de Grises”.

Para seleccionar la zona de acercamiento es necesario generar un rectángulo en la imagen escala de grises, a partir de dos clics izquierdos en los vértices opuestos (A y B) de manera similar a la función “Selección de elemento”. La Figura 3 muestra el proceso de selección

de la zona de interés. A partir de la marcación de estos vértices se obtiene el rectángulo completo. A continuación, se escala el fragmento o zona delimitada. Al finalizar se debe pulsar el botón Esc. En caso de requerir otro encuadre, se debe presionar el botón “Borrar” y repetir la operación.

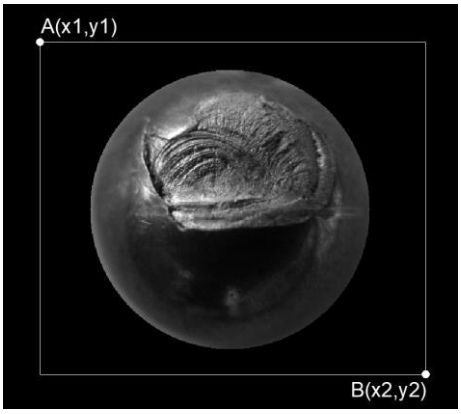


Figura 3. Selección de zona de interés para acercamiento.
(Fuente: Propia)

Finalmente, para alterar los valores de brillo y contraste es necesario utilizar los botones deslizables que se presentan en la Figura 2.23. Este proceso es iterable, ya que el usuario tiene la libertad de alterar los parámetros de manera continua. Junto a los botones se presenta un contador, que muestra los valores que toma cada variable.

Sección II

La detección de bordes se aplica de manera similar al cambio de brillo y contraste. Mediante botones deslizables se introducen las variables en la función. La imagen obtenida se puede visualizar en el visor principal. El proceso es iterativo, de manera que puede ser modificado conforme el usuario lo desee.

Seguido de esto se realiza la selección de áreas de fractura. Al seleccionar cualquier área, aparece una ventana en la cual el usuario puede ejecutar el procedimiento detallado en la función “Áreas de la fractura” y posteriormente cerrarla con el botón Esc. En primer lugar, es necesario considerar si el elemento es un eje hueco, de ser el caso se debe seleccionar “Eje Hueco” y delimitar el área interna. Si se considera un eje sólido, es posible saltarse este paso y seleccionar directamente el área de fractura y el área de fractura final. En los contadores se presentan los porcentajes de las áreas correspondientes. Es posible borrar la selección de áreas y repetir la operación.

Para resaltar las marcas de trinquete, se debe hacer clic en el botón correspondiente. Después, se genera una ventana en la cual el usuario puede identificar estas marcas. A continuación, se deben marcar los puntos correspondientes al inicio y fin, como se muestra en la Figura 4. Al finalizar la selección, apretar el botón Esc.

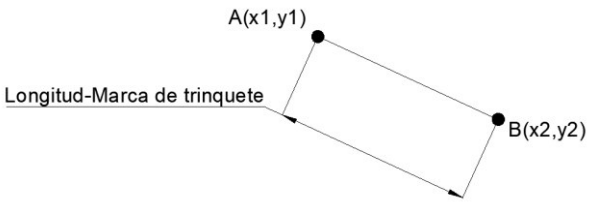


Figura 4. Selección de puntos (Marcas de trinquetes).
(Fuente: Propia)

La selección del origen inicia al apretar el botón correspondiente. A continuación, se presenta la metodología empleada para la selección del origen:

1. El usuario ingresa dos puntos (A y B) mediante el clic en la ventana generada, los cuales marcan el inicio y el fin de la primera marca de playa presente en la zona de fatiga.
2. A continuación, se obtiene la ecuación de la recta ($\overline{AB}$), su pendiente y centro C. Seguido de esto se determina la ecuación de una recta perpendicular a la recta ($\overline{AB}$), la cual indica la progresión de origen.
3. Por último, se selecciona por medio de un clic derecho, la ubicación del origen de la fractura. Este punto (Punto O) debe ser colocado únicamente a lo largo de la línea de progresión del origen. De esta forma, se obtiene la imagen origen que contiene los parámetros mencionados.

La Figura 5 muestra un esquema del proceso ejecutado en esta función. Al final de la selección apretar el botón Esc.

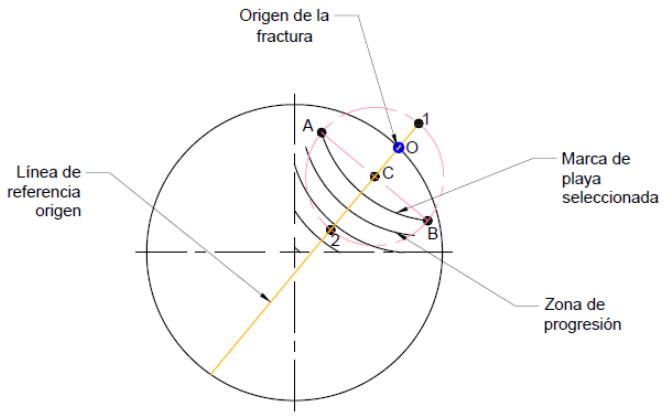


Figura 5. Método de determinación del origen de la fractura.
(Fuente: Propia)

Finalmente, como último paso se debe seleccionar la bisectriz del área de fractura final. El procedimiento inicia mediante la introducción de puntos por parte del usuario (A y B) mediante el clic izquierdo del ratón en los dos extremos del área de fractura final, como se muestra en la Figura 6. A partir de esto se determina la ecuación de la recta ($\overline{AB}$) y se obtiene el punto medio o centro C. Mediante el cálculo del centro, es posible generar una recta perpendicular a la recta generada originalmente. Para esto es necesario invertir la pendiente de la recta ($\overline{AB}$), obtenido así la bisectriz ($\overline{CD}$). Seguido de esto se obtiene el ángulo generado entre la línea del origen y la bisectriz. Al final de la selección apretar el botón Esc.

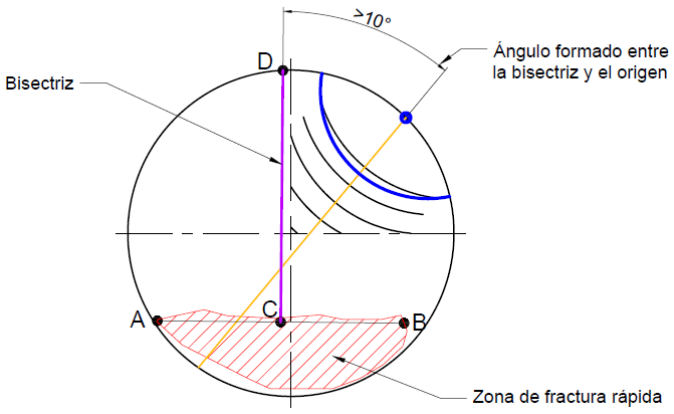


Figura 6. Generación de la Bisectriz.
(Fuente: Propia)

El botón “Borrar” que se encuentra situado a un lado de cada función permite eliminar la selección realizada y repetirla. En caso de no identificar cualquiera de estas marcas, solo presionar Esc para continuar.

Sección III

A continuación, es necesario responder las siete preguntas del cuestionario como se presenta en la Figura 2.27. Para esto se debe hacer clic sobre la “Pregunta #1”. Seguido se genera una ventana auxiliar, como se muestra en la Figura 2.28. Es necesario hacer clic en el botón “Seleccionar” y escoger una de las dos opciones en función de la similitud visual. Al término de cada pregunta presionar Esc para continuar.

Por último, el botón “Análisis de Resultados” presente en la Figura 2.27, permite generar una segunda ventana auxiliar que muestra los criterios obtenidos. Esta ventana posee tres botones principales, como se puede apreciar en la Figura 2.29. El primero crea un archivo en formato png de la imagen obtenida al final del procesamiento, llamado “Imagen Final.png”. El segundo botón genera un reporte llamado “Informe.docx”, como se puede apreciar en el Anexo III. Este contiene la imagen obtenida al final del procesamiento de imágenes y los criterios respectivos presentados en la ventana de resultados. El último botón permite cerrar el software. Para desbloquear el segundo y tercer botón es necesario hacer clic en “Guardar imagen” y luego en “Informe”.

Recomendaciones

- Respetar la secuencia de operación establecida.
- En caso de que el software no responda, cerrar e ingresar de nuevo.
- Cerrar el archivo “Informe.docx” después de usarlo, ya que puede generar conflicto al sobrescribirlo.
- Seleccionar siempre una opción en el cuestionario final, para evitar problemas en el desarrollo del procedimiento.

ANEXO V.

CODIGO DEL SOFTWARE FATIGUE FAILURE

```
"""ESCUELA POLITÉCNICA NACIONAL
FACULTAD DE INGENIERÍA MECÁNICA

    FIM FATIGUE FAILURE

Autores: Ariel Díaz y Erick Poveda"""
import sys
import math
from PyQt5 import QtCore, QtGui,QtWidgets
from PyQt5.QtWidgets import QApplication,QMainWindow, QMessageBox
from PyQt5 import uic
import cv2
import numpy as np
from PyQt5.QtCore import Qt
from PyQt5.QtGui import QImage, QPalette,QColor,QPixmap, QIcon
from PyQt5.QtWidgets import QFileDialog
from PyQt5.QtGui import QFont
import imutils
import copy
from numpy.polynomial import Polynomial as p
import os
from Preguntas import Ui_Dialog
from Resultados import Ui_Form
from docx import Document
from docx.shared import Cm
from PIL import Image
import time

img_point=[]
img_point2=[]
img_point3=[]
img_point4=[]
img_point5=[]
```

```
img_point6=[]
img_point7=[]
img_point8=[]
img_point9=[]
img_point10=[]
img_point11=[]
img_point12=[]
contador=0
cont=0
contadorang=0
listain=[]
lista=[]
lista2=[]
lista3=[]
lista4=[]
areas=np.zeros((3,1))
contl = 0
respuestas_a=np.zeros((1,8))
respuestas_b=np.zeros((1,8))
contr = 0

# In[FIM FATIGUE FAILURE]:
# Clase heredad de QMainQindow(Constructor de ventanas)

class Interfaz(QMainWindow):
    #Método constructor de la clase
    def __init__(self):
        #Inicar el objeto QMainWindow
        QMainWindow.__init__(self)#__init__ propiedad privada solo es accesib en el interior
de la clase
        #cARGAR LA CONFIGURACION DEL ARCHIVO .ui EN EL OBJETO
        uic.loadUi("Interfaz.ui", self)

        #Nombre del software
        self.setWindowTitle("FIM FATIGUE FAILURE")
        self.setWindowIcon(QIcon("Logo.png"))
```

```
        #Tamaño mínimo de la ventana
        self.setMinimumSize(1700,900)#Establece el tamañano minio de la ventana
        self.setMaximumSize(1700,900)#Establece el tamaño maximo de la ventana
        #self.setGeometry(0, 0, 500, 500)

        self.pushButton.setStyleSheet("QPushButton:hover {border: 3px solid black; background-color: gray; color: black;}")
        self.pushButton_4.setStyleSheet("QPushButton:hover {border: 3px solid black; background-color: gray; color: black;}")
        self.pushButton_4.setEnabled(False)
        self.pushButton_2.setStyleSheet("QPushButton:hover {border: 3px solid black; background-color: gray; color: black;}")
        self.pushButton_8.setStyleSheet("QPushButton:hover {border: 3px solid black; background-color: gray; color: black;}")
        self.radioButton.setStyleSheet("QPushButton:hover {border: 3px solid black; background-color: gray; color: black;}")
        self.radioButton_2.setStyleSheet("QPushButton:hover {border: 3px solid black; background-color: gray; color: black;}")
        self.radioButton_6.setStyleSheet("QPushButton:hover {border: 3px solid black; background-color: gray; color: black;}")
        self.radioButton_7.setStyleSheet("QPushButton:hover {border: 3px solid black; background-color: gray; color: black;}")
        self.radioButton_8.setStyleSheet("QPushButton:hover {border: 3px solid black; background-color: gray; color: black;}")
        self.radioButton_9.setStyleSheet("QPushButton:hover {border: 3px solid black; background-color: gray; color: black;}")
        self.radioButton_11.setStyleSheet("QPushButton:hover {border: 3px solid black; background-color: gray; color: black;}")
        self.radioButton_12.setStyleSheet("QPushButton:hover {border: 3px solid black; background-color: gray; color: black;}")
        self.radioButton_13.setStyleSheet("QPushButton:hover {border: 3px solid black; background-color: gray; color: black;}")
        self.radioButton_14.setStyleSheet("QPushButton:hover {border: 3px solid black; background-color: gray; color: black;}")
        self.radioButton_15.setStyleSheet("QPushButton:hover {border: 3px solid black; background-color: gray; color: black;}")
```

```
        self.radioButton_17.setStyleSheet("QPushButton:hover {border: 3px solid black; background-color: gray; color: black;}")
        self.radioButton_9.setStyleSheet("QPushButton:hover {border: 3px solid black; background-color: gray; color: black;}")
        self.pushButton_8.setStyleSheet("QPushButton:hover {border: 3px solid black; background-color: gray; color: black;}")
        self.pushButton_18.setStyleSheet("QPushButton:hover {border: 3px solid black; background-color: gray; color: black;}")
        self.pushButton_19.setStyleSheet("QPushButton:hover {border: 3px solid black; background-color: gray; color: black;}")
        self.pushButton_23.setStyleSheet("QPushButton:hover {border: 3px solid black; background-color: gray; color: black;}")
        self.pushButton_28.setStyleSheet("QPushButton:hover {border: 3px solid black; background-color: gray; color: black;}")
        self.pushButton_29.setStyleSheet("QPushButton:hover {border: 3px solid black; background-color: gray; color: black;}")
        self.pushButton_22.setStyleSheet("QPushButton:hover {border: 3px solid black; background-color: gray; color: black;}")
        self.pushButton_24.setStyleSheet("QPushButton:hover {border: 3px solid black; background-color: gray; color: black;}")
        self.pushButton_25.setStyleSheet("QPushButton:hover {border: 3px solid black; background-color: gray; color: black;}")
        self.pushButton.clicked.connect(self.abrir)
        #self.radioButton_6.clicked.connect(self.origen_proge)
        self.radioButton.clicked.connect(self.seleccioncirculo)
        self.radioButton_2.clicked.connect(self.seleccionrectangulo)
        self.pushButton_4.clicked.connect(self.grises)
        self.pushButton_8.clicked.connect(self.mascara)
        self.pushButton_18.clicked.connect(self.area)
        self.pushButton_19.clicked.connect(self.area2)
        self.label.setPixmap(QtGui.QPixmap(""))
        self.horizontalSlider_2.setMinimum(0)
        self.horizontalSlider_2.setMaximum(100)
        self.horizontalSlider_2.setSingleStep(1)
        self.horizontalSlider_2.setValue(50)
        self.horizontalSlider_2.valueChanged.connect(self.contrasatu)
```

```
self.pushButton_24.clicked.connect(self.acercar)
self.horizontalSlider.setMinimum(-127)
self.horizontalSlider.setMaximum(127)
self.horizontalSlider.setSingleStep(1)
self.horizontalSlider.setValue(50)
self.horizontalSlider.valueChanged.connect(self.contrasatu)
self.horizontalSlider_6.setMinimum(0)
self.horizontalSlider_6.setMaximum(255)
self.horizontalSlider_6.setSingleStep(1)
self.horizontalSlider_6.setValue(50)
self.horizontalSlider_6.valueChanged.connect(self.canny)
self.horizontalSlider_5.setMinimum(0)
self.horizontalSlider_5.setMaximum(255)
self.horizontalSlider_5.setSingleStep(1)
self.horizontalSlider_5.setValue(50)
self.horizontalSlider_5.valueChanged.connect(self.canny)
self.horizontalSlider_8.setMinimum(0)
self.horizontalSlider_8.setMaximum(255)
self.horizontalSlider_8.setSingleStep(1)
self.horizontalSlider_8.setValue(50)
self.horizontalSlider_8.valueChanged.connect(self.hold)
#self.pushButton_2.clicked.connect(self.limpiar)
self.pushButton_22.clicked.connect(self.borrar_zoom)
self.radioButton_7.clicked.connect(self.origen)
self.radioButton_8.clicked.connect(self.marcas_trinquetes)
self.radioButton_6.clicked.connect(self.bisectriz)
self.pushButton_26.clicked.connect(self.borrar_bisectriz)
self.pushButton_16.clicked.connect(self.borrar_origen)
self.pushButton_23.clicked.connect(self.borrar_areas)
self.radioButton_9.clicked.connect(self.pregunta_1)
self.radioButton_11.clicked.connect(self.pregunta_2)
self.radioButton_12.clicked.connect(self.pregunta_3)
self.radioButton_13.clicked.connect(self.pregunta_4)
self.radioButton_16.clicked.connect(self.pregunta_5)
self.radioButton_15.clicked.connect(self.pregunta_6)
self.radioButton_17.clicked.connect(self.pregunta_7)
```

```
self.pushButton_25.clicked.connect(self.resultados)
self.pushButton_28.clicked.connect(self.area_interna)
self.pushButton_29.clicked.connect(self.borra_interna)
self.radioButton_10.clicked.connect(self.eje_hueco)

#Bloqueo de Botones
self.pushButton_4.setEnabled(False)
self.pushButton_8.setEnabled(False)
self.pushButton_18.setEnabled(False)
self.pushButton_23.setEnabled(False)
self.pushButton_19.setEnabled(False)
self.pushButton_9.setEnabled(False)
self.pushButton_16.setEnabled(False)
self.pushButton_26.setEnabled(False)
self.radioButton_6.setEnabled(False)
self.radioButton_7.setEnabled(False)
self.radioButton_8.setEnabled(False)
self.radioButton.setEnabled(False)
self.radioButton_2.setEnabled(False)
self.groupBox.setEnabled(False)
self.groupBox_2.setEnabled(False)
self.groupBox_3.setEnabled(False)
self.groupBox_4.setEnabled(False)
self.groupBox_7.setEnabled(False)
self.groupBox_8.setEnabled(False)
self.radioButton_9.setEnabled(False)
self.radioButton_11.setEnabled(False)
self.radioButton_12.setEnabled(False)
self.radioButton_13.setEnabled(False)
self.radioButton_14.setEnabled(False)
self.radioButton_15.setEnabled(False)
self.radioButton_17.setEnabled(False)
self.radioButton_16.setEnabled(False)
self.pushButton_25.setEnabled(False)
self.pushButton_28.setEnabled(False)
self.pushButton_29.setEnabled(False)
```

```
        self.pushButton_2.setEnabled(False)


# In["Abrir Imagen"]:

    def abrir(self):
        global abrir, im
        adv= QMessageBox(self)
        adv.setWindowIcon(QIcon("logo.png"))
        adv.setWindowTitle("Indicaciones Iniciales")
        color= QPalette()

        panel = QPalette.Background
        color.setColor(panel, QColor("white"))
        adv.setPalette(color)
        f= self.font()
        f.setPointSize(11)
        adv.setFont(f)
        adv.setIconPixmap(QPixmap("Logo.png").scaled(150, 150, Qt.KeepAspectRatio))
        adv.setText(" <b>FIM FATIGUE FAILURE</b>    ")
        adv.setInformativeText("Software creado por Ariel Díaz y Erick Poveda\nEstudiantes
de la FIM-EPN")
        adv.setDetailedText("FIM FATIGUE FAILURE presenta las siguientes
restricciones:\n"
        "- Elementos de Acero.\n"
        "- Altos ciclos de servicio y cargas cíclicas.\n"
        "- Elementos de sección transversal circular o rectangular.\n"
        "- No se analiza estrías de fatiga")
        botonAceptar = adv.addButton("Aceptar", QMessageBox.YesRole)
        adv.setDefaultButton(botonAceptar)

        adv.exec_()

        resultado=str("Función Escalado de Imagen: Permite cargar y escalar la fotografía
correspondiente al objeto\n"
```

```
        "de estudio. El formato puede ser PNG o JPEG. Seleccione la imagen y pulse “Aceptar”. ")
        self.label_2.setFont(QFont("Arial",12))
        self.label_2.setAlignment(QtCore.Qt.AlignCenter)
        self.label_2.setText(resultado)
        self.groupBox_2.setEnabled(True)
        self.radioButton.setEnabled(True)
        self.radioButton_2.setEnabled(True)
        self.pushButton_2.setEnabled(True)
        self.filename=QFileDialog.getOpenFileName(filter = "im (*.*)")[0]
        self.im=cv2.imread(self.filename)
        abrir=imutils.resize(self.im,height=690, inter=cv2.INTER_AREA)
        w=abrir.shape[1]
        if w>920:
            d=(920,690)
            abrir=cv2.resize(abrir,d)
        self.label.resize(w,690)
        if w>920:
            self.label.resize(920,690)
        cv2.imwrite("Imagen Estandar.png",abrir)
        self.tra=cv2.cvtColor(abrir,cv2.COLOR_BGR2RGB)
        im=QImage(self.tra, self.tra.shape[1],self.tra.shape[0],self.tra.strides[0], QImage.Format_RGB888)
        self.label.setPixmap(QtGui.QPixmap.fromImage(im))

# In["Recortar"]:
    def seleccioncirculo(self):
        global mascara_seleccionar, im, maskab
        resultado=str("Función Selección de Elemento (Sección Circular): Permite generar un círculo alrededor del\n"
        "elemento. Para esto necesario que haga clic en dos puntos opuestos del perímetro del objeto\n"
        "de estudio, como se muestra en el Visor de Ayuda. Al terminar la selección presione\n"
        "ESC, seguido del botón recortar.")
        self.label_2.setFont(QFont("Arial",12))
```

```
        self.label_2.setAlignment(QtCore.Qt.AlignCenter)
        self.label_2.setText(resultado)
        visor=cv2.imread("ayuda_seleccion.png")
        dim=(300,230)
        imagen_visor=cv2.resize(visor,dim)
        i=QImage(imagen_visor,
imagen_visor.shape[1],imagen_visor.shape[0],imagen_visor.strides[0],
QImage.Format_RGB888)
        self.label_3.setPixmap(QtGui.QPixmap.fromImage(i))
        self.pushButton_8.setEnabled(True)
        def dibujando(event,x,y,flags,program):
            global cont, drawing, img_point
            if event == cv2.EVENT_LBUTTONDOWN:
                drawing = True
                img_point.append((x,y))
                cont=cont+1

                p1=img_point[0]
                x1=p1[0]
                y1=p1[1]

                if cont == 2:

                    q1=img_point[1]
                    x2=q1[0]
                    y2=q1[1]

                    cx=int((x2+x1)/2)
                    cy=int((y2+y1)/2)

                    d=int(math.sqrt(((x2-x1)**2)+((y2-y1)**2)))
                    radio=int(d/2)



                    cv2.circle(imagen,(cx,cy),radio,(0,0,255),3)
                    cv2.circle(maska,(cx,cy),radio,(255,255,255),-1)
```

```
                cv2.circle(maskab,(cx,cy),radio,(250,250,250),-1)
                drawing=[]
                img_point=[]
                cont=0

        imagen=cv2.imread("Imagen Estandar.png")
        cv2.namedWindow("imagen", cv2.WINDOW_AUTOSIZE)
        cv2.setMouseCallback("imagen",dibujando)
        h, w, c = imagen.shape
        maska=np.zeros(shape=(h, w), dtype = "uint8")
        maskab=np.zeros(shape=(h, w), dtype = "uint8")
        while True:
            cv2.imshow("imagen", imagen)
            k= cv2.waitKey(1) & 0xFF
            if k==27:
                break
        cv2.destroyAllWindows()
        mascara_seleccionar=maskab
        self.tra=cv2.cvtColor(imagen,cv2.COLOR_BGR2RGB)
        self.im=imutils.resize(self.tra,height=690, inter=cv2.INTER_AREA)
        im=QImage(self.im,                    self.im.shape[1],self.im.shape[0],self.im.strides[0],
QImage.Format_RGB888)
        self.label.setPixmap(QtGui.QPixmap.fromImage(im))
        self.pushButton_8.setEnabled(True)

    def seleccionrectangulo(self):
        global mascara_seleccionar
        resultado=str("Función Selección de Elemento (Sección Rectangular): Permite
generar un rectángulo alrededor\n"
        "del elemento. Para esto necesario que haga clic en dos vértices opuestos del
perímetro del\n"
        "objeto de estudio, como se muestra en el Visor de Ayuda. Al terminar la selección
presione\n"
        "ESC, seguido del botón recortar.")
        self.label_2.setFont(QFont("Arial",12))
        self.label_2.setAlignment(QtCore.Qt.AlignCenter)
```

```
        self.label_2.setText(resultado)
        visor=cv2.imread("ayuda_seleccionr.png")
        dim=(300,230)
        imagen_visor=cv2.resize(visor,dim)
        i=QImage(imagen_visor,
imagen_visor.shape[1],imagen_visor.shape[0],imagen_visor.strides[0],
QImage.Format_RGB888)
        self.label_3.setPixmap(QtGui.QPixmap.fromImage(i))
        def dibujando(event,x,y,flags,program):
            global cont, drawing, img_point
            if event == cv2.EVENT_LBUTTONDOWN:
                drawing = True
                img_point.append((x,y))
                cont=cont+1



                p1=img_point[0]
                x1=p1[0]
                y1=p1[1]

                if cont == 2:

                    q1=img_point[1]
                    x2=q1[0]
                    y2=q1[1]

                    cv2.rectangle(imagen,(x1,y1),(x2,y2),(0,0,255),3)
                    cv2.rectangle(mask,(x1,y1),(x2,y2),(250,250,250),-1)

                    drawing=[]
                    img_point=[]
                    cont=0

        imagen=cv2.imread("Imagen Estandar.png")
        cv2.namedWindow("imagen", cv2.WINDOW_AUTOSIZE)
        cv2.setMouseCallback("imagen",dibujando)
```

```
        h, w, c = imagen.shape
        mask=np.zeros(shape=(h, w), dtype = "uint8")
        while True:
            cv2.imshow("imagen", imagen)
            k= cv2.waitKey(1) & 0xFF
            if k==27:
                break
        cv2.destroyAllWindows()
        mascara_seleccionar=mask
        self.tra=cv2.cvtColor(imagen,cv2.COLOR_BGR2RGB)
        self.im=imutils.resize(self.tra,height=690, inter=cv2.INTER_AREA)
        im=QImage(self.im,                    self.im.shape[1],self.im.shape[0],self.im.strides[0],
QImage.Format_RGB888)
        self.label.setPixmap(QtGui.QPixmap.fromImage(im))
        self.pushButton_8.setEnabled(True)

    def mascara(self):
        global imagen_recortada
        resultado=str("Función Selección de Elemento:Recorta el fondo que se encuentra por
fuera del área seleccionada.")
        self.label_2.setFont(QFont("Arial",12))
        self.label_2.setAlignment(QtCore.Qt.AlignCenter)
        self.label_2.setText(resultado)
        imagen=copy.copy(abrir)
        mascara = copy.copy(mascara_seleccionar)
        self.nm=cv2.bitwise_and(imagen, imagen, mask = mascara)
        self.ima=cv2.cvtColor(self.nm,cv2.COLOR_BGR2RGB)
        imagen_recortada=imutils.resize(self.ima,height=690, inter=cv2.INTER_AREA)
        im=QImage(imagen_recortada,
imagen_recortada.shape[1],imagen_recortada.shape[0],imagen_recortada.strides[0],
QImage.Format_RGB888)
        self.label.setPixmap(QtGui.QPixmap.fromImage(im))
        self.pushButton_4.setEnabled(True)

# In["Escala de Grises"]:
    def grises(self):
```

```
        global imagen_grises
        self.groupBox_2.setEnabled(False)
        self.pushButton.setEnabled(False)
        self.pushButton_2.setEnabled(False)
        resultado=str("Función Escala de Grises: Permite transformar una imagen en modelo
BGR a una en\n"
        "escala de grises. Genera contraste en la imagen para detectar las marcas y zonas\n"
        "características de la fractura. La operación se realiza de forma automática.")
        self.label_2.setFont(QFont("Arial",12))
        self.label_2.setAlignment(QtCore.Qt.AlignCenter)
        self.label_2.setText(resultado)
        imagen= imagen_recortada
        self.negro=cv2.cvtColor(imagen,cv2.COLOR_BGR2GRAY)
        self.im_1=cv2.cvtColor(self.negro,cv2.COLOR_BGR2RGB)
        imagen_grises=imutils.resize(self.im_1,height=690, inter=cv2.INTER_AREA)
        self.tra=QImage(imagen_grises,
imagen_grises.shape[1],imagen_grises.shape[0],imagen_grises.strides[0],
QImage.Format_RGB888)
        cv2.imwrite("Imagen Escala de Grises.png",imagen_grises)
        self.label.setPixmap(QtGui.QPixmap.fromImage(self.tra))
        self.groupBox_3.setEnabled(True)

# In["Zoom"]:
    def acercar(self):
        resultado=str("Función Acercamiento: Realiza un acercamiento en la zona
seleccionada. Es necesario que el\n"
        "usuario señale dos vértices opuestos de un rectángulo sobre el elemento, para
identificar\n"
        "la zona de acercamiento, como se muestra en el Visor de Ayuda. Al terminar
presionar Esc. ")
        self.label_2.setFont(QFont("Arial",12))
        self.label_2.setAlignment(QtCore.Qt.AlignCenter)
        self.label_2.setText(resultado)
        self.pushButton_4.setEnabled(False)
        global imagen_zoom, imagen,imagen_sele
        def dibujando(event,x,y,flags,program):
```

```
            global cont, drawing, img_point2
            if event == cv2.EVENT_LBUTTONDOWN:
                drawing = True
                img_point2.append((x,y))
                cont = cont+1
                if cont>=2:
                    for i in range (1,len(img_point2)):
                        x1 = img_point2[i-1][0]
                        y1 = img_point2[i-1][1]
                        x2 = img_point2[i][0]
                        y2 = img_point2[i][1]

                        cv2.rectangle(imagen,(x1,y1),(x2,y2),(0,0,255),2)
                        drawing=[]
                        cont=0
        visor=cv2.imread("ayuda_acerca.png")
        dim=(300,230)
        imagen_visor=cv2.resize(visor,dim)
        imagen_visor=cv2.cvtColor(imagen_visor,cv2.COLOR_BGR2RGB)
        i=QImage(imagen_visor,
imagen_visor.shape[1],imagen_visor.shape[0],imagen_visor.strides[0],
QImage.Format_RGB888)
        self.label_3.setPixmap(QtGui.QPixmap.fromImage(i))
        imagen = copy.copy(imagen_grises)
        cv2.namedWindow("imagen", cv2.WINDOW_NORMAL)
        cv2.setMouseCallback("imagen",dibujando)

        while True:
            cv2.imshow("imagen", imagen)
            k= cv2.waitKey(1) & 0xFF
            if k==27:
                break
        cv2.destroyAllWindows()

        x11 = img_point2[0][0]
        y11 = img_point2[0][1]
```

```
        x22 = img_point2[1][0]
        y22 = img_point2[1][1]

        im1=imagen[y11:y22, x11:x22]#(y1,y2),(x1,x2)
        self.groupBox.setEnabled(True)
        self.tra=cv2.cvtColor(im1,cv2.COLOR_BGR2RGB)
        imagen_zoom=imutils.resize(self.tra,height=690)
        w=imagen_zoom.shape[1]
        w=int(w)
        if w>920:
            d=(920,690)
            imagen_zoom=cv2.resize(imagen_zoom,d)
        self.label.resize(w,690)
        if w>920:
            self.label.resize(920,690)
        self.label.resize(920,690)
        im=QImage(imagen_zoom,
imagen_zoom.shape[1],imagen_zoom.shape[0],imagen_zoom.strides[0],
QImage.Format_RGB888)
        self.label.setPixmap(QtGui.QPixmap.fromImage(im))

    def borrar_zoom(self,imagen_zoom):
        resultado=str("Pulse Seleccionar Área y repita el procedimiento.")
        self.label_2.setFont(QFont("Arial",12))
        self.label_2.setAlignment(QtCore.Qt.AlignCenter)
        self.label_2.setText(resultado)
        a=self.pushButton_22.isChecked()
        if a==False:
            del imagen_zoom
            del img_point2[:]

# In["Contraste y Saturación"]:
    def contrasatu(self):
        global imagen_contrasa, brillo,contraste
        resultado=str("Función Brillo y Contraste: Permite alterar el brillo y contraste de la
Imagen.\n"
```

```
        "Para esto, es necesario deslizar los botones correspondientes. Los cambios efectuados se\n"
        "muestran automáticamente en el Visor Principal.")
        self.label_2.setFont(QFont("Arial",12))
        self.label_2.setAlignment(QtCore.Qt.AlignCenter)
        self.label_2.setText(resultado)
        self.im=imagen_zoom
        value=self.horizontalSlider.value()
        value_1=self.horizontalSlider_2.value()
        brillo=str(value)
        contraste=str(value_1)
        self.groupBox_3.setEnabled(False)
        self.label_4.setText(str(value))
        self.label_5.setText(str(value_1))
        self.al=int(value_1)*2/100
        self.el=int(value)
        negro=np.zeros(self.im.shape,self.im.dtype)
        self.con=cv2.addWeighted(self.im,self.al,negro,0,self.el)
        self.im_1=cv2.cvtColor(self.con,cv2.COLOR_BGR2RGB)
        imagen_contrasa=imutils.resize(self.im_1,height=690, inter=cv2.INTER_AREA)
        im=QImage(imagen_contrasa,
imagen_contrasa.shape[1],imagen_contrasa.shape[0],imagen_contrasa.strides[0],
QImage.Format_RGB888)
        self.label.setPixmap(QtGui.QPixmap.fromImage(im))
        self.groupBox_7.setEnabled(True)



# In["Detección de Bordes"]:
    #Método I
    def canny(self):
        tiempoin=time.time()
        global imagen_bordes, metodo
        resultado=str("Función Detección de Bordes (Método I): Es posible detectar los rasgos superficiales más\n"
```

```
        "importantes de la imagen a partir de valores umbrales máximos y mínimo. Para esto, es\n"
        "necesario deslizar los botones correspondientes. Los cambios efectuados se muestran\n"
        "automáticamente en el Visor Principal.")
        metodo=str("Método I")
        self.label_2.setFont(QFont("Arial",12))
        self.label_2.setAlignment(QtCore.Qt.AlignCenter)
        self.label_2.setText(resultado)
        self.groupBox_8.setEnabled(True)
        self.radioButton_10.setEnabled(True)
        self.pushButton_18.setEnabled(True)
        self.i=copy.copy(imagen_contrasa)
        self.im=cv2.cvtColor(self.i,cv2.COLOR_BGR2GRAY)
        value=self.horizontalSlider_5.value()
        value_1=self.horizontalSlider_6.value()
        self.label_10.setText(str(value))
        self.label_11.setText(str(value_1))
        self.ci=cv2.Canny(self.im,value_1,value)
        contorno, jerarquia = cv2.findContours(self.ci,cv2.RETR_CCOMP, cv2.CHAIN_APPROX_SIMPLE )
        for i in range (0,len(contorno)):
            cv2.drawContours(self.i ,contorno, i,(231, 113, 35),1)
        self.im_1=self.i
        imagen_bordes=imutils.resize(self.im_1,height=690, inter=cv2.INTER_AREA)
        cv2.imwrite("Imagen con Bordes.png",imagen_bordes)
        im=QImage(imagen_bordes,
imagen_bordes.shape[1],imagen_bordes.shape[0],imagen_bordes.strides[0],
QImage.Format_RGB888)
        self.label.setPixmap(QtGui.QPixmap.fromImage(im))
        tiempofin=time.time()
    #Método II
    def hold(self):
        tiempoin=time.time()
        global imagen_bordes, metodo
```

```
        resultado=str("Función Detección de Bordes (Método II): Es posible detectar los rasgos superficiales más\n"
        "importantes de la imagen a partir de un valor umbral. Para esto, es necesario deslizar los\n"
        "botones correspondientes. Los cambios efectuados se muestran automáticamente en el\n"
        "Visor Principal.")
        metodo=str("Método II")
        self.label_2.setFont(QFont("Arial",12))
        self.label_2.setAlignment(QtCore.Qt.AlignCenter)
        self.label_2.setText(resultado)
        self.groupBox_8.setEnabled(True)
        self.radioButton_10.setEnabled(True)
        self.pushButton_18.setEnabled(True)
        self.im=copy.copy(imagen_contrasa)
        umbral=self.horizontalSlider_8.value()
        self.label_13.setText(str(umbral))
        bi=cv2.cvtColor(self.im,cv2.COLOR_BGR2GRAY)
        _,th=cv2.threshold(bi,umbral,255,cv2.THRESH_BINARY)
        #_,th=cv2.adaptiveThreshold(bi,255,cv2.ADAPTIVE_THRESH_MEAN_C,
cv2.THRESH_BINARY,11,2)

(contornos,_)=cv2.findContours(th,cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_SIMPL
E)
        for i in range (len(contornos)):
            cv2.drawContours(self.im,contornos, i,(0,255,0),1)
        self.ima=cv2.cvtColor(self.im,cv2.COLOR_BGR2RGB)
        imagen_bordes=imutils.resize(self.ima,height=690, inter=cv2.INTER_AREA)
        cv2.imwrite("Imagen con Bordes.png",imagen_bordes)
        im=QImage(imagen_bordes,
imagen_bordes.shape[1],imagen_bordes.shape[0],imagen_bordes.strides[0],
QImage.Format_RGB888)
        self.label.setPixmap(QtGui.QPixmap.fromImage(im))


# In["Áreas"]:
```

```
    def eje_hueco(self):
        resultado=str("Función Áreas de Fractura (Área Interna): Permite seleccionar y recortar el área interna de\n"
        "un eje hueco. Para esto es necesario pulsar el botón Seleccionar y generar una polilínea alrededor\n"
        "del área correspondiente, como se muestra en el Visor de Ayuda. Al finalizar\n"
        "debe presionar Esc.")
        self.label_2.setFont(QFont("Arial",12))
        self.label_2.setAlignment(QtCore.Qt.AlignCenter)
        self.label_2.setText(resultado)
        self.pushButton_28.setEnabled(True)
        self.pushButton_29.setEnabled(True)
        self.groupBox_7.setEnabled(False)
        self.groupBox.setEnabled(False)
    def area_interna(self):
        visor=cv2.imread("ayuda_area.png")
        dim=(300,230)
        imagen_visor=cv2.resize(visor,dim)
        i=QImage(imagen_visor,
imagen_visor.shape[1],imagen_visor.shape[0],imagen_visor.strides[0],
QImage.Format_RGB888)
        self.label_3.setPixmap(QtGui.QPixmap.fromImage(i))
        resultado=str("Función Áreas de Fractura (Área Interna): Permite seleccionar y recortar el área interna de\n"
        "un eje hueco. Para esto es necesario pulsar el botón Seleccionar y generar una polilínea alrededor\n"
        "del área correspondiente, como se muestra en el Visor de Ayuda. Al finalizar\n"
        "debe presionar Esc.")
        self.label_2.setFont(QFont("Arial",12))
        self.label_2.setAlignment(QtCore.Qt.AlignCenter)
        self.label_2.setText(resultado)
        global mascara_area_in, area_in, areainterna, imagen_bordes
        self.pushButton_29.setEnabled(True)
        self.pushButton_18.setEnabled(True)
        self.label_2.setText(resultado)
```

```
def dibujando(event,x,y,flags,program):
    global cont, drawing, img_point12
    if event == cv2.EVENT_LBUTTONDOWN:
        drawing = True
        img_point12.append((x,y))
        cont = cont+1
        if cont>=2:
            for i in range (1,len(img_point12)):
                x1 = img_point12[i-1][0]
                y1 = img_point12[i-1][1]
                x2 = img_point12[i][0]
                y2 = img_point12[i][1]
                cv2.line(imagen, (x1,y1), (x2,y2), (255,0,255), 2)
                cont=0

imagen=cv2.imread("Imagen con Bordes.png")
cv2.namedWindow("imagen", cv2.WINDOW_NORMAL)
cv2.setMouseCallback("imagen",dibujando)
while True:
    cv2.imshow("imagen", imagen)
    k= cv2.waitKey(1) & 0xFF
    if k==27:
        break
cv2.destroyAllWindows()
ln=len(img_point12)
m0 = np.zeros((ln,1,2), dtype=np.int32)

for i in range (0,ln):
    for j in range (0,2):
        m0[i,0,j]=img_point12[i][j]

boton=self.pushButton_28.isChecked()
if boton==False:
    listain.append(m0)
    mascara_area_in= np.zeros_like(imagen)
    cv2.drawContours(mascara_area_in,listain, 0,(255,0,255),-1)
```

```
            areainterna=int( cv2.contourArea(listain[0]))
            areainterna=round(areainterna,2)
            areas[0]=areainterna
            mascara_area_in[mascara_area_in==255]=imagen[mascara_area_in==255]
            self.con=cv2.bitwise_or(imagen, mascara_area_in)
            imagen_bordes=imutils.resize(self.con,height=690, inter=cv2.INTER_AREA)
            im=QImage(imagen_bordes,
imagen_bordes.shape[1],imagen_bordes.shape[0],imagen_bordes.strides[0],
QImage.Format_RGB888)
            self.label.setPixmap(QtGui.QPixmap.fromImage(im))

    def borra_interna(self):
        resultado=str("Pulse Seleccionar y repita el procedimiento.")
        self.label_2.setFont(QFont("Arial",12))
        self.label_2.setAlignment(QtCore.Qt.AlignCenter)
        self.label_2.setText(resultado)
        global mascara_area_in,areainterna,listain,img_point12
        a=self.pushButton_23.isChecked()
        if a==False:
            del img_point12[:]
            del listain[:]
            del areainterna
            del mascara_area_in

    def area(self):
        visor=cv2.imread("ayuda_area.png")
        dim=(300,230)
        imagen_visor=cv2.resize(visor,dim)
        i=QImage(imagen_visor,
imagen_visor.shape[1],imagen_visor.shape[0],imagen_visor.strides[0],
QImage.Format_RGB888)
        self.label_3.setPixmap(QtGui.QPixmap.fromImage(i))
        resultado=str("Función Áreas de Fractura (Área Total): Permite seleccionar el área
total de fractura. Para\n"
        "esto es necesario pulsar el botón Seleccionar y generar una polilínea alrededor del
área\n"
```

```
"correspondiente, como se muestra en el Visor de Ayuda. Al finalizar debe presionar Esc.")
        self.label_2.setFont(QFont("Arial",12))
        self.label_2.setAlignment(QtCore.Qt.AlignCenter)
        self.label_2.setText(resultado)
        self.groupBox_7.setEnabled(False)
        self.groupBox.setEnabled(False)
        self.pushButton_29.setEnabled(False)
        global mascara_area_1, area_total
        self.pushButton_19.setEnabled(True)
        def dibujando(event,x,y,flags,program):
            global cont, drawing, img_point4
            if event == cv2.EVENT_LBUTTONDOWN:
                drawing = True
                img_point4.append((x,y))
                cont = cont+1
                if cont>=2:
                    for i in range (1,len(img_point4)):
                        x1 = img_point4[i-1][0]
                        y1 = img_point4[i-1][1]
                        x2 = img_point4[i][0]
                        y2 = img_point4[i][1]
                        cv2.line(imagen, (x1,y1), (x2,y2), (0, 0, 255), 2)
                        cont=0
        imagen=copy.copy(imagen_bordes)
        cv2.namedWindow("imagen", cv2.WINDOW_NORMAL)
        cv2.setMouseCallback("imagen",dibujando)
        while True:
            cv2.imshow("imagen", imagen)
            k= cv2.waitKey(1) & 0xFF
            if k==27:
                break
        cv2.destroyAllWindows()
        ln=len(img_point4)
        m1 = np.zeros((ln,1,2), dtype=np.int32)
```

```
        for i in range (0,ln):
            for j in range (0,2):
                m1[i,0,j]=img_point4[i][j]

        lista.append(m1)
        mascara_area_1= np.zeros_like(imagen)
        cv2.drawContours(mascara_area_1,lista, 0,(0,0,255),-1)
        areatotal=int( cv2.contourArea(lista[0]))
        areatotal=round(areatotal,2)
        areas[0][0]=areatotal
        mascara_area_1[mascara_area_1==255]=imagen[mascara_area_1==255]
        self.con=cv2.bitwise_or(imagen, mascara_area_1)
        area_total=imutils.resize(self.con,height=690, inter=cv2.INTER_AREA)
        im=QImage(area_total,  area_total.shape[1],area_total.shape[0],area_total.strides[0],
QImage.Format_RGB888)
        self.label.setPixmap(QtGui.QPixmap.fromImage(im))

    def area2(self):
        resultado=str("Función Áreas de Fractura (Área de Fractura Final): Permite
seleccionar el área de fractura\n"
        "final. Para esto es necesario pulsar el botón Seleccionar y generar una polilínea
alrededor\n"
        "del área correspondiente, como se muestra en el Visor de Ayuda. Al finalizar debe\n"
        "presionar Esc. ")
        self.label_2.setFont(QFont("Arial",12))
        self.label_2.setAlignment(QtCore.Qt.AlignCenter)
        self.label_2.setText(resultado)
        global resultadosub,resultadofinal
        global area_2, out, out_1, mask_2, areafinal
        self.groupBox_4.setEnabled(True)
        self.radioButton_8.setEnabled(True)
        self.pushButton_9.setEnabled(True)
        def dibujando(event,x,y,flags,program):
            global cont, drawing, img_point5
            if event == cv2.EVENT_LBUTTONDOWN:
                drawing = True
```

```
        img_point5.append((x,y))
        cont = cont+1
        if cont>=2:
            for i in range (1,len(img_point5)):
                x1 = img_point5[i-1][0]
                y1 = img_point5[i-1][1]
                x2 = img_point5[i][0]
                y2 = img_point5[i][1]
                cv2.line(imagen, (x1,y1), (x2,y2), (255,0,0),2)
imagen=copy.copy(area_total)
cv2.namedWindow("imagen", cv2.WINDOW_NORMAL)
cv2.setMouseCallback("imagen",dibujando)
while True:
    cv2.imshow("imagen", imagen)
    k= cv2.waitKey(1) & 0xFF
    if k==27:
        break
cv2.destroyAllWindows()

ln=len(img_point5)
m2 = np.zeros((ln,1,2), dtype=np.int32)

for i in range (0,ln):
    for j in range (0,2):
        m2[i,0,j]=img_point5[i][j]

lista2.append(m2)
areasel=int( cv2.contourArea(lista2[0]))
areasel=round(areasel,2)
areatotal=areas[0][0]-areas[2][0]
areafinal=((areasel)/areatotal)*100
areafinal=round(areafinal,2)
areafrac=((areatotal-areasel)/areatotal)*100
areafrac=round(areafrac,2)
resultadosub=str(areafinal)
resultadofinal=str(areafrac)
```

```
        areatotal=str(areatotal)
        self.label_6.setFont(QFont("Arial",10))
        self.label_6.setAlignment(QtCore.Qt.AlignCenter)
        self.label_6.setText(resultadosub)
        self.label_7.setFont(QFont("Arial",10))
        self.label_7.setAlignment(QtCore.Qt.AlignCenter)
        self.label_7.setText(resultadofinal)
        mask_2=np.zeros_like(imagen)
        out=np.zeros_like(imagen)
        out_1=np.zeros_like(imagen)
        cv2.drawContours(mask_2,lista2, 0,(255,0,0),-1)
        out_1[mask_2==255]=imagen[mask_2==255]
        self.con=cv2.add(imagen, out_1)
        #self.con=imagen
        #self.tra=cv2.cvtColor(self.con,cv2.COLOR_BGR2RGB)
        area_2=imutils.resize(self.con,height=690, inter=cv2.INTER_AREA)
        im=QImage(area_2,                area_2.shape[1],area_2.shape[0],area_2.strides[0],
QImage.Format_RGB888)
        self.label.setPixmap(QtGui.QPixmap.fromImage(im))
        self.pushButton_23.setEnabled(True)

    def borrar_areas(self):
        resultado=str("Pulse Seleccionar y repita el procedimiento.")
        self.label_2.setFont(QFont("Arial",12))
        self.label_2.setAlignment(QtCore.Qt.AlignCenter)
        self.label_2.setText(resultado)
        global area_2,mascara_area_1,area_total,mask
        a=self.pushButton_23.isChecked()
        if a==False:
            del img_point4[:]
            del img_point5[:]
            del area_2
            del mascara_area_1
            del area_total
            del lista2[:]
            del lista[:]
```

```
# In["Marcas de trinquete y Origen"]:
    def marcas_trinquetes(self):
        resultado=str("Función Marcas de Trinquete: Permite subrayar las marcas de trinquete presentes en la\n"
        "superficie. Para esto es necesario marcar con dos clics el inicio y el final de la marca\n"
        "respectivamente, como se muestra en el Visor de Ayuda. Al finalizar debe presionar Esc.")
        self.label_2.setFont(QFont("Arial",12))
        self.label_2.setAlignment(QtCore.Qt.AlignCenter)
        self.label_2.setText(resultado)
        self.radioButton_7.setEnabled(True)
        self.pushButton_16.setEnabled(True)
        self.groupBox_8.setEnabled(False)
        visor=cv2.imread("ayuda_trinquete.png")
        dim=(300,230)
        imagen_visor=cv2.resize(visor,dim)
        i=QImage(imagen_visor,
imagen_visor.shape[1],imagen_visor.shape[0],imagen_visor.strides[0],
QImage.Format_RGB888)
        self.label_3.setPixmap(QtGui.QPixmap.fromImage(i))
        global imagen_trinquete
        def dibujando(event,x,y,flags,program):
            global cont, drawing, img_point8
            if event == cv2.EVENT_LBUTTONDOWN:
                drawing = True
                img_point8.append((x,y))
                cont = cont+1
                if cont>=2:
                    for i in range (1,len(img_point8)):
                        x1 = img_point8[i-1][0]
                        y1 = img_point8[i-1][1]
                        x2 = img_point8[i][0]
                        y2 = img_point8[i][1]
```

```
                cv2.line(imagen, (x1, y1), (x2, y2), (255, 255, 0), 3)
                cont = 0
                img_point8 = []

        imagen=copy.copy(area_2)
        cv2.namedWindow("imagen", cv2.WINDOW_NORMAL)
        cv2.setMouseCallback("imagen",dibujando)

        while True:
            cv2.imshow("imagen", imagen)
            k= cv2.waitKey(1) & 0xFF
            if k==27:
                break
        cv2.destroyAllWindows()
        self.tra=imagen
        imagen_trinquete=imutils.resize(self.tra,height=690, inter=cv2.INTER_AREA)
        im=QImage(imagen_trinquete,
imagen_trinquete.shape[1],imagen_trinquete.shape[0],imagen_trinquete.strides[0],
QImage.Format_RGB888)
        self.label.setPixmap(QtGui.QPixmap.fromImage(im))

    def borrar_trinquete(self,imagen_trinquete):
        resultado=str("Pulse Seleccionar y repita el procedimiento")
        self.label_2.setFont(QFont("Arial",12))
        self.label_2.setAlignment(QtCore.Qt.AlignCenter)
        self.label_2.setText(resultado)
        a=self.pushButton_9.isChecked()
        if a==False:
            del imagen_trinquete
            del img_point8[:]

    def origen(self):
        visor=cv2.imread("ayuda_origen.png")
        dim=(300,230)
        imagen_visor=cv2.resize(visor,dim)
```

```
        i=QImage(imagen_visor,
imagen_visor.shape[1],imagen_visor.shape[0],imagen_visor.strides[0],
QImage.Format_RGB888)
        self.label_3.setPixmap(QtGui.QPixmap.fromImage(i))
        resultado=str("Función Selección de Origen: Permite generar una línea de progresión
y el origen mediante\n"
        "un punto. Para esto es necesario marcar con dos clics el inicio y el final de la primera
marca de\n"
        "playa. Después, debe generar un punto mediante un clic sobre la línea generada para
indicar\n"
        "el origen, como se muestra en el Visor de Ayuda. Al finalizar debe presionar Esc.")
        self.label_2.setFont(QFont("Arial",12))
        self.label_2.setAlignment(QtCore.Qt.AlignCenter)
        self.label_2.setText(resultado)
        self.radioButton_6.setEnabled(True)
        self.pushButton_26.setEnabled(True)
        self.radioButton_8.setEnabled(False)
        self.pushButton_9.setEnabled(False)
        global imagen_origen
        def direccion(event,x,y,flags,program):
            global contl, contr, drawing, img_point9, ny, ny1, nx, nx1, nr,mc,contadorang
            if event == cv2.EVENT_LBUTTONDOWN:
                drawing = True
                img_point9.append((x,y))
                contl = contl+1
                contadorang=contadorang+1
                if contl == 2:
                    for i in range (1, len(img_point9)):

                        #linea origen
                        x1 = img_point9[i-1][0]
                        y1 = -img_point9[i-1][1]
                        x2 = img_point9[i][0]
                        y2 = -img_point9[i][1]

                        cx=(x2+x1)/2
```

```
cy=(y2+y1)/2
d=math.sqrt(((x2-x1)**2)+((y2-y1)**2))
r=(d/2)

m=(y2-y1)/(x2-x1)
m_in=-1/m

a=(-m_in*cx)+cy
b=a-cy
#i*x^2+j*x+b
i=1+(m_in**2)
j=(-2*cx)+(2*m_in*b)
k=-(r**2)+(cx**2)+(b**2)

polinomio=p([k, j, i])
raices=polinomio.roots()

nr = int(math.sqrt((r**2)+(r**2)))
nx = int(raices[0])
ny = -int(m_in*nx+a)

nx1 = int(raices[1])
ny1 = -int(m_in*nx1+a)

mc = (ny1-ny)/(nx1-nx)

#centros
cv2.line(imagen, (nx,ny), (nx1, ny1), (0,0,0), 2)
h, w, c = imagen.shape
yx = 0
yx1 = h
xl1 = int((yx-ny)/mc) + nx
xl2 = int((yx1-ny)/mc) + nx
cv2.line(imagen, (xl1, yx), (xl2, yx1), (138,43,226), 2)
contl=0
del img_point9[:]
```

```
            if event == cv2.EVENT_RBUTTONDOWN:
                drawing = True
                img_point11.append((x,y))
                contr = contr+1

                x1r = img_point11[0][0]
                y1r = -img_point11[0][1]

                cv2.circle(imagen, (x1r, -y1r), 5, (255,0,0), -1)
                cv2.circle(imagen, (x1r, -y1r), 5, (255,0,0), -1)
                contr=0
                del img_point11[:]

        imagen = copy.copy(imagen_trinquete)
        cv2.namedWindow("imagen", cv2.WINDOW_NORMAL)
        cv2.setMouseCallback("imagen",direccion)

        while True:
            cv2.imshow("imagen", imagen)
            k= cv2.waitKey(1) & 0xFF
            if k==27:
                break
        cv2.destroyAllWindows()
        self.tra=imagen
        imagen_origen=imutils.resize(self.tra,height=690, inter=cv2.INTER_AREA)
        im=QImage(imagen_origen,
imagen_origen.shape[1],imagen_origen.shape[0],imagen_origen.strides[0],
QImage.Format_RGB888)
        self.label.setPixmap(QtGui.QPixmap.fromImage(im))


    def borrar_origen(self,imagen_origen):
        resultado=str("Pulse Seleccionar y repita el procedimiento")
        self.label_2.setFont(QFont("Arial",12))
        self.label_2.setAlignment(QtCore.Qt.AlignCenter)
```

```
        self.label_2.setText(resultado)
        b=self.pushButton_16.isChecked()
        if b==False:
            del imagen_origen
            del img_point9[:]
            del img_point11[:]


    def bisectriz(self):
        visor=cv2.imread("ayuda_bisectriz.png")
        dim=(300,230)
        imagen_visor=cv2.resize(visor,dim)
        i=QImage(imagen_visor,
imagen_visor.shape[1],imagen_visor.shape[0],imagen_visor.strides[0],
QImage.Format_RGB888)
        self.label_3.setPixmap(QtGui.QPixmap.fromImage(i))
        resultado=str("Función Selección de Bisectriz: Genera una bisectriz que divide en dos el área de Fractura\n"
        "Final. Para esto es necesario pulsar el botón Seleccionar y hacer clic en dos puntos opuestos, de\n"
        "manera horizontal, en el perímetro del área seleccionada, como se muestra en el Visor de Ayuda.\n"
        "Al finalizar debe presionar Esc.")
        self.label_2.setFont(QFont("Arial",12))
        self.label_2.setAlignment(QtCore.Qt.AlignCenter)
        self.label_2.setText(resultado)
        global imagen_bisectriz, thetadeg
        self.groupBox_11.setEnabled(True)
        self.radioButton_9.setEnabled(True)
        self.radioButton_7.setEnabled(False)
        self.pushButton_16.setEnabled(False)
        def dibujando(event,x,y,flags,program):
            global cont, drawing, img_point10

            if event == cv2.EVENT_LBUTTONDOWN:
                global thetadeg
```

```
        drawing = True
        img_point10.append((x,y))
        cont = cont+1
        if cont>=2:
            for i in range (1,len(img_point10)):
                x1 = -img_point10[i-1][0]
                y1 = -img_point10[i-1][1]
                x2 = -img_point10[i][0]
                y2 = -img_point10[i][1]

                cx = (x1 + x2)/2
                cy = (y1 + y2)/2

                mr1 = (y2 - y1) / (x2 - x1)
                mr2 = -1/mr1

                h, w, c = imagen.shape
                xnew1 = 0
                xnew2 = w

                ynew1 = -int((mr2 * (-xnew1 - cx)) + cy)
                ynew2 = -int((mr2 * (-xnew2 - cx)) + cy)

                cv2.line(imagen, (xnew1, ynew1), (xnew2, ynew2), (251,140,0), 2)

                tetha = math.atan ((mc - mr2)/(1 + (mc*mr2)))
                thetadeg = (tetha*360)/(2*math.pi)

                if thetadeg > 0:
                    thetadeg=thetadeg

                if thetadeg < 0:
                    thetadeg = thetadeg + 180


imagen=copy.copy(imagen_origen)
```

```
        cv2.namedWindow("imagen", cv2.WINDOW_NORMAL)
        cv2.setMouseCallback("imagen",dibujando)

        while True:
            cv2.imshow("imagen", imagen)
            k= cv2.waitKey(1) & 0xFF
            if k==27:
                break
        cv2.destroyAllWindows()
        self.tra=imagen
        imagen_bisectriz=imutils.resize(self.tra,height=690, inter=cv2.INTER_AREA)
        im=QImage(imagen_bisectriz,
imagen_bisectriz.shape[1],imagen_bisectriz.shape[0],imagen_bisectriz.strides[0],
QImage.Format_RGB888)
        self.label.setPixmap(QtGui.QPixmap.fromImage(im))

    def borrar_bisectriz(self,imagen_bisectriz):
        resultado=str("Pulse Seleccionar y repita el procedimiento")
        self.label_2.setFont(QFont("Arial",12))
        self.label_2.setAlignment(QtCore.Qt.AlignCenter)
        self.label_2.setText(resultado)
        c=self.pushButton_26.isChecked()
        if c==False:
            del imagen_bisectriz
            del img_point10[:]

    def imagenes(self):
            ######### Opción A #########
        dim_a=(350,320)
        opa=cv2.resize(pregunta_a,dim_a)
        opcion_a=cv2.cvtColor(opa,cv2.COLOR_BGR2RGB)
        im_a=QImage(opcion_a,    opcion_a.shape[1],opcion_a.shape[0],opcion_a.strides[0],
QImage.Format_RGB888)
        self.ui.opcionadialog.setPixmap(QtGui.QPixmap.fromImage(im_a))

            ######### Opción B #########
```

```
        dim_b=(350,320)
        opb=cv2.resize(pregunta_b,dim_b)
        opcion_b=cv2.cvtColor(opb,cv2.COLOR_BGR2RGB)
        im_b=QImage(opcion_b,    opcion_b.shape[1],opcion_b.shape[0],opcion_b.strides[0],
QImage.Format_RGB888)
        self.ui.opcionbdialog.setPixmap(QtGui.QPixmap.fromImage(im_b))



    def a(self):
        global respuesta, respuestas_a
        respuesta=self.ui.radioButtondialog.isChecked()
        if respuesta==True:
            self.ui.radioButton2dialog.setEnabled(False)
            numero=contador
            respuestas_a[0,numero-1]=1

    def b(self):
        global respuesta, respuestas_b
        respuesta=self.ui.radioButton2dialog.isChecked()
        if respuesta==True:
            self.ui.radioButtondialog.setEnabled(False)

    def pregunta_1(self):
        resultado=str("Cuestionario Final: Permite obtener información adicional sobre el
proceso de falla. Para esto\n"
        "se debe hacer clic en el botón Seleccionar de la ventana generada. Seguido, escoger
la opción\n"
        "correspondiente. Finalmente, presionar el botón Esc. ")
        self.label_2.setFont(QFont("Arial",12))
        self.label_2.setAlignment(QtCore.Qt.AlignCenter)
        self.label_2.setText(resultado)
        self.groupBox_4.setEnabled(False)
        global pregunta_a, pregunta_b, contador
        ####### VENTANA-PREGUNTA #1
        self.ventana=QtWidgets.QDialog()
```

```
        self.ui=Ui_Dialog()
        self.ui.setupUi(self.ventana)
        self.ventana.show()
        contador=1
        self.radioButton_11.setEnabled(True)
        self.radioButton_9.setEnabled(False)
        self.ventana.setWindowTitle("Pregunta #1")
        self.ventana.setWindowIcon(QIcon("Logo.png"))
        pregunta_a=cv2.imread("pregunta_1_1.jpeg")
        pregunta_b=cv2.imread("pregunta_1_2.jpeg")
        self.ui.selecciondialog.clicked.connect(self.imagenes)
        self.ui.radioButtondialog.clicked.connect(self.a)
        self.ui.radioButton2dialog.clicked.connect(self.b)
        pregunta=str("\n¿La cara de la superficie de fractura es perpendicular al eje de acción
del elemento?\n ")
        self.ui.pregunta.setFont(QFont("Arial",12))
        self.ui.pregunta.setAlignment(QtCore.Qt.AlignCenter)
        self.ui.pregunta.setText(pregunta)
        informacion=str("Pregunta #1: Determinar si la fractura fue perpendicular al eje de
acción o a su vez posee un\n"
        "ángulo de fractura, permitirá distinguir entre dos tipos de esfuerzos a los que pudo
haber sido \n"
        "sometido el elemento, los cuales son: Flexión (Opción A) o Torsión (Opción B).")
        self.ui.info.setFont(QFont("Arial",10))
        self.ui.info.setAlignment(QtCore.Qt.AlignCenter)
        self.ui.info.setText(informacion)
        abrir_a=copy.copy(imagen_zoom)
        dim_a=(490,450)
        opcion_a=cv2.resize(abrir_a,dim_a)
        im_a=QImage(opcion_a,    opcion_a.shape[1],opcion_a.shape[0],opcion_a.strides[0],
QImage.Format_RGB888)
        self.ui.resultadodialog.setPixmap(QtGui.QPixmap.fromImage(im_a))

    def pregunta_2(self):
        ####### VENTANA-PREGUNTA #2
```

```
        resultado=str("Cuestionario Final: Permite obtener información adicional sobre el proceso de falla. Para esto\n"
        "se debe hacer clic en el botón Seleccionar de la ventana generada. Seguido, escoger la opción\n"
        "correspondiente. Finalmente, presionar el botón Esc. ")
        self.label_2.setFont(QFont("Arial",12))
        self.label_2.setAlignment(QtCore.Qt.AlignCenter)
        self.label_2.setText(resultado)
        global pregunta_a, pregunta_b,contador
        contador=2
        self.radioButton_12.setEnabled(True)
        self.radioButton_11.setEnabled(False)
        self.ventana=QtWidgets.QDialog()
        self.ui=Ui_Dialog()
        self.ui.setupUi(self.ventana)
        self.ventana.show()
        self.ventana.setWindowTitle("Pregunta #2")
        self.ventana.setWindowIcon(QIcon("Logo.png"))
        pregunta_a=cv2.imread("pregunta_2_1.jpeg")
        pregunta_b=cv2.imread("pregunta_2_2.jpeg")
        self.ui.selecciondialog.clicked.connect(self.imagenes)
        self.ui.radioButtondialog.clicked.connect(self.a)
        self.ui.radioButton2dialog.clicked.connect(self.b)
        pregunta=str("\n¿Existe deformación en el área de fractura?\n ")
        self.ui.pregunta.setFont(QFont("Arial",12))
        self.ui.pregunta.setAlignment(QtCore.Qt.AlignCenter)
        self.ui.pregunta.setText(pregunta)
        informacion=str("Pregunta #2: Determina si existe deformación plástica en el elemento visto a forma de distorsión\n"
        "en su geometría. Permite identificar la categoría de falla por fatiga entre dos tipos: \n"
        "Fatiga de Muy Bajos Ciclos (Opción A) y Fatiga de Altos Ciclos (Opción B).")
        self.ui.info.setFont(QFont("Arial",10))
        self.ui.info.setAlignment(QtCore.Qt.AlignCenter)
        self.ui.info.setText(informacion)
        abrir_a=copy.copy(imagen_bordes)
        dim_a=(490,450)
```

```
        opcion_a=cv2.resize(abrir_a,dim_a)
        im_a=QImage(opcion_a,    opcion_a.shape[1],opcion_a.shape[0],opcion_a.strides[0],
QImage.Format_RGB888)
        self.ui.resultadodialog.setPixmap(QtGui.QPixmap.fromImage(im_a))

    def pregunta_3(self):
        ####### VENTANA-PREGUNTA #3
        resultado=str("Cuestionario Final: Permite obtener información adicional sobre el
proceso de falla. Para esto\n"
        "se debe hacer clic en el botón Seleccionar de la ventana generada. Seguido, escoger
la opción\n"
        "correspondiente. Finalmente, presionar el botón Esc. ")
        self.label_2.setFont(QFont("Arial",12))
        self.label_2.setAlignment(QtCore.Qt.AlignCenter)
        self.label_2.setText(resultado)
        global pregunta_a, pregunta_b, contador
        contador=3
        self.radioButton_13.setEnabled(True)
        self.radioButton_12.setEnabled(False)
        self.ventana=QtWidgets.QDialog()
        self.ui=Ui_Dialog()
        self.ui.setupUi(self.ventana)
        self.ventana.show()
        self.ventana.setWindowTitle("Pregunta #3")
        self.ventana.setWindowIcon(QIcon("Logo.png"))
        pregunta_a=cv2.imread("pregunta_3_1.jpeg")
        pregunta_b=cv2.imread("pregunta_3_2.jpeg")
        self.ui.selecciondialog.clicked.connect(self.imagenes)
        self.ui.radioButtondialog.clicked.connect(self.a)
        self.ui.radioButton2dialog.clicked.connect(self.b)
        pregunta=str("\n¿Existen marcas de trinquete?\n ")
        self.ui.pregunta.setFont(QFont("Arial",12))
        self.ui.pregunta.setAlignment(QtCore.Qt.AlignCenter)
        self.ui.pregunta.setText(pregunta)
        informacion=str("Pregunta #3: Permite identificar la existencia de las marcas de
trinquete basándose en las\n"
```

```
        "imágenes presentadas. Estas marcas están ligadas a la presencia de concentradores\n"
        "de esfuerzos dentro del elemento. Marcas de Trinquete (Opción A). Ausencia de\n"
        "Marcas de Trinquete (Opción B).")
        self.ui.info.setFont(QFont("Arial",10))
        self.ui.info.setAlignment(QtCore.Qt.AlignCenter)
        self.ui.info.setText(informacion)
        abrir_a=copy.copy(imagen_trinquete)
        dim_a=(490,450)
        opcion_a=cv2.resize(abrir_a,dim_a)
        im_a=QImage(opcion_a, opcion_a.shape[1],opcion_a.shape[0],opcion_a.strides[0], QImage.Format_RGB888)
        self.ui.resultadodialog.setPixmap(QtGui.QPixmap.fromImage(im_a))

    def pregunta_4(self):
        resultado=str("Cuestionario Final: Permite obtener información adicional sobre el proceso de falla. Para esto\n"
        "se debe hacer clic en el botón Seleccionar de la ventana generada. Seguido, escoger la opción\n"
        "correspondiente. Finalmente, presionar el botón Esc. ")
        self.label_2.setFont(QFont("Arial",12))
        self.label_2.setAlignment(QtCore.Qt.AlignCenter)
        self.label_2.setText(resultado)
        ####### VENTANA-PREGUNTA #4
        global pregunta_a, pregunta_b,contador
        contador=4
        self.radioButton_16.setEnabled(True)
        self.radioButton_13.setEnabled(False)
        self.ventana=QtWidgets.QDialog()
        self.ui=Ui_Dialog()
        self.ui.setupUi(self.ventana)
        self.ventana.show()
        self.ventana.setWindowTitle("Pregunta #4")
        self.ventana.setWindowIcon(QIcon("Logo.png"))
        pregunta_a=cv2.imread("pregunta_4_1.jpeg")
        pregunta_b=cv2.imread("pregunta_4_2.jpeg")
```

```
        self.ui.selecciondialog.clicked.connect(self.imagenes)
        self.ui.radioButtondialog.clicked.connect(self.a)
        self.ui.radioButton2dialog.clicked.connect(self.b)
        pregunta=str("\n¿Existen marcas de progresión?\n ")
        self.ui.pregunta.setFont(QFont("Arial",12))
        self.ui.pregunta.setAlignment(QtCore.Qt.AlignCenter)
        self.ui.pregunta.setText(pregunta)
        informacion=str("Pregunta #4: Se verifica la existencia de marcas de playa o
progresión por medio de una \n"
        "comparación entre las imágenes presentadas. Esto permite identificar variaciones en
la\n"
        "carga cíclica aplicada al elemento. Marcas de progresión (Opción A). Ausencia de
Marcas de\n"
        "Progresión (Opción B).")
        self.ui.info.setFont(QFont("Arial",10))
        self.ui.info.setAlignment(QtCore.Qt.AlignCenter)
        self.ui.info.setText(informacion)
        abrir_a=copy.copy(imagen_bordes)
        dim_a=(490,450)
        opcion_a=cv2.resize(abrir_a,dim_a)
        im_a=QImage(opcion_a,    opcion_a.shape[1],opcion_a.shape[0],opcion_a.strides[0],
QImage.Format_RGB888)
        self.ui.resultadodialog.setPixmap(QtGui.QPixmap.fromImage(im_a))

    def pregunta_5(self):
        resultado=str("Cuestionario Final: Permite obtener información adicional sobre el
proceso de falla. Para esto\n"
        "se debe hacer clic en el botón Seleccionar de la ventana generada. Seguido, escoger
la opción\n"
        "correspondiente. Finalmente, presionar el botón Esc. ")
        self.label_2.setFont(QFont("Arial",12))
        self.label_2.setAlignment(QtCore.Qt.AlignCenter)
        self.label_2.setText(resultado)
        ####### VENTANA-PREGUNTA #5
        global pregunta_a, pregunta_b,contador
        contador=5
```

```
        self.radioButton_15.setEnabled(True)
        self.radioButton_16.setEnabled(False)
        self.ventana=QtWidgets.QDialog()
        self.ui=Ui_Dialog()
        self.ui.setupUi(self.ventana)
        self.ventana.show()
        self.ventana.setWindowTitle("Pregunta #5")
        self.ventana.setWindowIcon(QIcon("Logo.png"))
        pregunta_a=cv2.imread("pregunta_5_1.jpeg")
        pregunta_b=cv2.imread("pregunta_5_2.jpeg")
        self.ui.selecciondialog.clicked.connect(self.imagenes)
        self.ui.radioButtondialog.clicked.connect(self.a)
        self.ui.radioButton2dialog.clicked.connect(self.b)
        pregunta=str("\n¿El área de fatiga rodea al área de fractura final?\n ")
        self.ui.pregunta.setFont(QFont("Arial",12))
        self.ui.pregunta.setAlignment(QtCore.Qt.AlignCenter)
        self.ui.pregunta.setText(pregunta)
        informacion=str("Pregunta #5: La interacción entre el área de fatiga y el área de
fractura final permite obtener\n"
        "información con respecto a las cargas principales por las cuales ha fallado el
elemento.\n"
        "Cuando el AF rodea el AR se produce una falla por flexión rotatoria (Opción A).
Cuando la AF\n"
        "no rodea la AR se produce una falla por flexión plana (Opción B).")
        self.ui.info.setFont(QFont("Arial",10))
        self.ui.info.setAlignment(QtCore.Qt.AlignCenter)
        self.ui.info.setText(informacion)
        abrir_a=copy.copy(area_2)
        dim_a=(490,450)
        opcion_a=cv2.resize(abrir_a,dim_a)
        im_a=QImage(opcion_a,  opcion_a.shape[1],opcion_a.shape[0],opcion_a.strides[0],
QImage.Format_RGB888)
        self.ui.resultadodialog.setPixmap(QtGui.QPixmap.fromImage(im_a))
    def pregunta_6(self):
        ####### VENTANA-PREGUNTA #6
```

```
        resultado=str("Cuestionario Final: Permite obtener información adicional sobre el proceso de falla. Para esto\n"
        "se debe hacer clic en el botón Seleccionar de la ventana generada. Seguido, escoger la opción\n"
        "correspondiente. Finalmente, presionar el botón Esc. ")
        self.label_2.setFont(QFont("Arial",12))
        self.label_2.setAlignment(QtCore.Qt.AlignCenter)
        self.label_2.setText(resultado)
        global pregunta_a, pregunta_b, contador
        contador=6
        self.radioButton_17.setEnabled(True)
        self.radioButton_15.setEnabled(False)
        self.ventana=QtWidgets.QDialog()
        self.ui=Ui_Dialog()
        self.ui.setupUi(self.ventana)
        self.ventana.show()
        self.ventana.setWindowTitle("Pregunta #6")
        self.ventana.setWindowIcon(QIcon("Logo.png"))
        pregunta_a=cv2.imread("pregunta_6_1.jpeg")
        pregunta_b=cv2.imread("pregunta_6_2.jpeg")
        self.ui.selecciondialog.clicked.connect(self.imagenes)
        self.ui.radioButtondialog.clicked.connect(self.a)
        self.ui.radioButton2dialog.clicked.connect(self.b)
        pregunta=str("\n¿El área de fractura final es redonda u ovalada?\n ")
        self.ui.pregunta.setFont(QFont("Arial",12))
        self.ui.pregunta.setAlignment(QtCore.Qt.AlignCenter)
        self.ui.pregunta.setText(pregunta)
        informacion=str("Pregunta #6: La forma geométrica que tiene el área de fractura final permite determinar el\n"
        "tipo de carga secundaria involucrada en la falla del elemento. Si posee una forma ovalada\n"
        "(Opción A) existen cargas de flexión plana involucradas. Para una forma redonda\n"
        "(Opción B) no existen cargas de flexión plana.")
        self.ui.info.setFont(QFont("Arial",10))
        self.ui.info.setAlignment(QtCore.Qt.AlignCenter)
        self.ui.info.setText(informacion)
```

```
        self.pushButton_14.setEnabled(False)
        abrir_a=copy.copy(area_2)
        dim_a=(490,450)
        opcion_a=cv2.resize(abrir_a,dim_a)
        im_a=QImage(opcion_a,  opcion_a.shape[1],opcion_a.shape[0],opcion_a.strides[0],
QImage.Format_RGB888)
        self.ui.resultadodialog.setPixmap(QtGui.QPixmap.fromImage(im_a))

    def pregunta_7(self):
        resultado=str("Cuestionario Final: Permite obtener información adicional sobre el
proceso de falla. Para esto\n"
        "se debe hacer clic en el botón Seleccionar de la ventana generada. Seguido, escoger
la opción\n"
        "correspondiente. Por último, presionar Esc y hacer clic en el botón Análisis de
Resultados ")
        self.label_2.setFont(QFont("Arial",12))
        self.label_2.setAlignment(QtCore.Qt.AlignCenter)
        self.label_2.setText(resultado)
        ####### VENTANA-PREGUNTA #7
        global pregunta_a, pregunta_b,contador
        contador=7
        self.pushButton_25.setEnabled(True)
        self.radioButton_17.setEnabled(False)
        self.ventana=QtWidgets.QDialog()
        self.ui=Ui_Dialog()
        self.ui.setupUi(self.ventana)
        self.ventana.show()
        self.ventana.setWindowTitle("Pregunta #7")
        self.ventana.setWindowIcon(QIcon("Logo.png"))
        pregunta_a=cv2.imread("pregunta_7_1.jpeg")
        pregunta_b=cv2.imread("pregunta_7_2.jpeg")
        self.ui.selecciondialog.clicked.connect(self.imagenes)
        self.ui.radioButtondialog.clicked.connect(self.a)
        self.ui.radioButton2dialog.clicked.connect(self.b)
        pregunta=str("\n¿La bisectriz que se forma en la zona de fractura final apunta al
origen?\n ")
```

```
        self.ui.pregunta.setFont(QFont("Arial",12))
        self.ui.pregunta.setAlignment(QtCore.Qt.AlignCenter)
        self.ui.pregunta.setText(pregunta)
        informacion=str("Pregunta #7: La dirección de la bisectriz con respecto a la línea del
origen\n"
        "permite determinar la implicación que tiene la carga de flexión rotatoria en la carga
\n"
        "de flexión plana. Carga de flexión rotatoria implicada en la flexión plana (Opción A).\n"
        "No existe carga de flexión rotatoria implicada (Opción B).")
        self.ui.info.setFont(QFont("Arial",10))
        self.ui.info.setAlignment(QtCore.Qt.AlignCenter)
        self.ui.info.setText(informacion)
        self.pushButton_15.setEnabled(False)
        abrir_a=copy.copy(imagen_bisectriz)
        dim_a=(490,450)
        opcion_a=cv2.resize(abrir_a,dim_a)
        im_a=QImage(opcion_a,    opcion_a.shape[1],opcion_a.shape[0],opcion_a.strides[0],
QImage.Format_RGB888)
        self.ui.resultadodialog.setPixmap(QtGui.QPixmap.fromImage(im_a))


    def imagen_final(self):
        global final
        self.ui.imagenfinalwidget_2.setEnabled(True)
        final=cv2.imwrite("Imagen Final.png",imagen_bisectriz)


    def resultados(self):
        self.groupBox_11.setEnabled(False)
        global thetadeg
        global contadorang
        self.forma=QtWidgets.QDialog()
        self.ui=Ui_Form()
        self.ui.setupUi(self.forma)
        self.forma.show()
        self.forma.setWindowTitle("Resultados Obtenidos")
        self.forma.setWindowIcon(QIcon("Logo.png"))
```

```
        imagen_final=copy.copy(imagen_bisectriz)
        self.ui.imagenfinalwidget_2.setEnabled(False)
        self.ui.cerrarwidget.setEnabled(False)
        dim_final=(481,460)
        ima_final=cv2.resize(imagen_final,dim_final)
        im_f=QImage(ima_final,      ima_final.shape[1],ima_final.shape[0],ima_final.strides[0],
QImage.Format_RGB888)
        self.ui.resultadoswidget.setPixmap(QtGui.QPixmap.fromImage(im_f))
        self.ui.imagenfinalwidget.clicked.connect(self.imagen_final)
        self.ui.imagenfinalwidget_2.clicked.connect(self.informe)
        self.ui.cerrarwidget.clicked.connect(self.cerrar)
        global respuestas_a, a, b,c,d,e
        if respuestas_a[0][0]==1:
            if respuestas_a[0][1]==0:
                respuesta_carga=str("Fatiga de Altos Ciclos")
                a=str("Fatiga de Altos Ciclos")
                self.ui.categoriawidget.setFont(QFont("Arial",11))
                self.ui.categoriawidget.setAlignment(QtCore.Qt.AlignCenter)
                self.ui.categoriawidget.setText(respuesta_carga)
                if respuestas_a[0][4]==0:
                    if respuestas_a[0][6]==1:
                        respuesta_carga=str("Falla por Flexión Plana Pura")
                        b=str("Falla por Flexión Plana Pura")
                        self.ui.cargawidget.setFont(QFont("Arial",11))
                        self.ui.cargawidget.setAlignment(QtCore.Qt.AlignCenter)
                        self.ui.cargawidget.setText(respuesta_carga)
                    if respuestas_a[0][6]==0:
                        if contadorang==2:
                            if thetadeg>10:
                                respuesta_carga=str("Falla por Flexión Giratoria")
                                b=str("Falla por Flexión Giratoria")
                                self.ui.cargawidget.setFont(QFont("Arial",11))
                                self.ui.cargawidget.setAlignment(QtCore.Qt.AlignCenter)
                                self.ui.cargawidget.setText(respuesta_carga)
                            else:
```

```
                    respuesta_carga=str("Falla por Flexión Plana con Cargas de Flexión
Giratoria")
                    b=str("Falla por Flexión Plana con Cargas de Flexión Giratoria")
                    self.ui.cargawidget.setFont(QFont("Arial",11))
                    self.ui.cargawidget.setAlignment(QtCore.Qt.AlignCenter)
                    self.ui.cargawidget.setText(respuesta_carga)
                if contadorang>2:
                    respuesta_carga=str("Falla por Flexión Plana con Cargas de Flexión
Giratoria")
                    b=str("Falla por Flexión Plana con Cargas de Flexión Giratoria")
                    self.ui.cargawidget.setFont(QFont("Arial",11))
                    self.ui.cargawidget.setAlignment(QtCore.Qt.AlignCenter)
                    self.ui.cargawidget.setText(respuesta_carga)
            if respuestas_a[0][4]==1:
                if respuestas_a[0][5]==1:
                    respuesta_carga=str("Falla por Flexión Giratoria sin Cargas de Flexión
Plana")
                    b=("Falla por Flexión Giratoria sin Cargas de Flexión Plana")
                    self.ui.cargawidget.setFont(QFont("Arial",11))
                    self.ui.cargawidget.setAlignment(QtCore.Qt.AlignCenter)
                    self.ui.cargawidget.setText(respuesta_carga)

                else:
                    respuesta_carga=str("Falla por Flexión Giratoria con Cargas de Flexión
Plana.")
                    b=str("Falla por Flexión Giratoria con Cargas de Flexión Plana.")
                    self.ui.cargawidget.setFont(QFont("Arial",11))
                    self.ui.cargawidget.setAlignment(QtCore.Qt.AlignCenter)
                    self.ui.cargawidget.setText(respuesta_carga)
            else:
                respuesta_carga=str("Fatiga de Muy Bajos Ciclos")
                a=str("Fatiga de Muy Bajos Ciclos")
                b=str("Fatiga de Muy Bajos Ciclosa")
                self.ui.categoriawidget.setFont(QFont("Arial",11))
                self.ui.categoriawidget.setAlignment(QtCore.Qt.AlignCenter)
                self.ui.categoriawidget.setText(respuesta_carga)
```

```
        self.ui.categoriawidget.setFont(QFont("Arial",11))
        self.ui.categoriawidget.setAlignment(QtCore.Qt.AlignCenter)
        self.ui.categoriawidget.setText(b)
else:
    respuesta_carga=str("Carga de torsión o Carga Combinada")
    b=str("Carga de torsión o Carga Combinada")
    a=str("No Aplica")
    self.ui.cargawidget.setFont(QFont("Arial",11))
    self.ui.cargawidget.setAlignment(QtCore.Qt.AlignCenter)
    self.ui.cargawidget.setText(respuesta_carga)
    self.ui.categoriawidget.setFont(QFont("Arial",11))
    self.ui.categoriawidget.setAlignment(QtCore.Qt.AlignCenter)
    self.ui.categoriawidget.setText(a)

if respuestas_a[0][2]==1:
    respuesta_carga=str("Existen Altos Concentradores de Esfuerzo")
    e=str("Existen Altos Concentradores de Esfuerzo")
    self.ui.concentradoreswidget.setFont(QFont("Arial",11))
    self.ui.concentradoreswidget.setAlignment(QtCore.Qt.AlignCenter)
    self.ui.concentradoreswidget.setText(respuesta_carga)
else:
    respuesta_carga=str("No Existen Concentradores de Esfuerzo")
    e=str("No Existen Concentradores de Esfuerzo")
    self.ui.concentradoreswidget.setFont(QFont("Arial",11))
    self.ui.concentradoreswidget.setAlignment(QtCore.Qt.AlignCenter)
    self.ui.concentradoreswidget.setText(respuesta_carga)
if respuestas_a[0][3]==1:
    respuesta_carga=str("Existe Variación en la Carga Cíclica Aplicada")
    d=str("Existe Variación en la Carga Cíclica Aplicada")
    self.ui.aplicadawidget.setFont(QFont("Arial",11))
    self.ui.aplicadawidget.setAlignment(QtCore.Qt.AlignCenter)
    self.ui.aplicadawidget.setText(respuesta_carga)
else:
    respuesta_carga=str("No Existe Variación en la Carga Cíclica Aplicada")
    d=str("No Existe Variación en la Carga Cíclica Aplicada")
    self.ui.aplicadawidget.setFont(QFont("Arial",11))
```

```
            self.ui.aplicadawidget.setAlignment(QtCore.Qt.AlignCenter)
            self.ui.aplicadawidget.setText(respuesta_carga)

        if areafinal>50:
            respuesta_carga=str("Magnitud alta")
            c=str("Magnitud alta")
            self.ui.magnitudwidget.setFont(QFont("Arial",11))
            self.ui.magnitudwidget.setAlignment(QtCore.Qt.AlignCenter)
            self.ui.magnitudwidget.setText(respuesta_carga)
        else:
            respuesta_carga=str("Magnitud baja")
            c=str("Magnitud baja")
            self.ui.magnitudwidget.setFont(QFont("Arial",11))
            self.ui.magnitudwidget.setAlignment(QtCore.Qt.AlignCenter)
            self.ui.magnitudwidget.setText(respuesta_carga)

    def informe(self):
        self.ui.cerrarwidget.setEnabled(True)
        document = Document()
        document.add_heading("INFORME DEL SOFTWARE FIM FATIGUE FAILURE",
level=3)
        data = (('-TIPO DE CARGA:', b), ('-CATEGORIA DE FALLA POR FATIGA:', a),('-
MAGNITUD DE CARGA FINAL:', c),('-CARGA APLICADA:', d),
        ('-CONCENTRADORES DE ESFUERZOS:', e),('-VALOR DE BRILLO:', brillo ),('-
VALOR DE CONTRASTE:', contraste ),
        ('-METODO DE DETECCION DE BORDES:',metodo ),('-PORCENTAJE AREA DE
FRACTURA FINAL:', resultadosub),('-PORCENTAJE AREA DE FATIGA:',resultadofinal))
        table = document.add_table(rows=1, cols=2)
        table.rows[0].cells[0].text = 'PARAMETRO'
        table.rows[0].cells[1].text = 'RESULTADO'
        for prod, numbr in data:
            row_cells = table.add_row().cells
            row_cells[0].text = prod
            row_cells[1].text = str(numbr)
        document.add_picture("Imagen Final.png", width=Cm(5))
        document.save('Informe.docx')
```

```
    def cerrar(self):
        self.ui.cerrarwidget.setEnabled(True)
        self.close()
        self.forma=QtWidgets.QDialog()
        self.ui=Ui_Form()
        self.close()

#Instancia para inicar una aplicación
app=QApplication(sys.argv)
    #Crear un objeto de la clase
_ventana=Interfaz()
    #Mostrar la ventana
_ventana.show()
    #Ejecutar a aplicación
app.exec_()
```

ANEXO VI.

CODIGO DE LA VENTANA AUXILIAR I

```
from PyQt5 import QtCore, QtGui, QtWidgets
class Ui_Dialog(object):
    def setupUi(self, Dialog):
        Dialog.setObjectName("Dialog")
        Dialog.resize(1262, 586)
        self.opcionadialog = QtWidgets.QLabel(Dialog)
        self.opcionadialog.setGeometry(QtCore.QRect(530, 100, 350, 320))
        self.opcionadialog.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.opcionadialog.setText("")
        self.opcionadialog.setObjectName("opcionadialog")
        self.info = QtWidgets.QLabel(Dialog)
        self.info.setGeometry(QtCore.QRect(530, 470, 711, 91))
        self.info.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.info.setText("")
        self.info.setObjectName("info")
        self.pregunta = QtWidgets.QLabel(Dialog)
        self.pregunta.setGeometry(QtCore.QRect(20, 20, 1081, 61))
        self.pregunta.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.pregunta.setText("")
        self.pregunta.setObjectName("pregunta")
        self.selecciondialog = QtWidgets.QPushButton(Dialog)
        self.selecciondialog.setGeometry(QtCore.QRect(1130, 30, 111, 41))
        font = QtGui.QFont()
        font.setFamily("Arial")
        font.setPointSize(10)
        self.selecciondialog.setFont(font)
        self.selecciondialog.setObjectName("selecciondialog")
        self.opcionbdialog = QtWidgets.QLabel(Dialog)
        self.opcionbdialog.setGeometry(QtCore.QRect(890, 100, 350, 320))
        self.opcionbdialog.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.opcionbdialog.setText("")
        self.opcionbdialog.setObjectName("opcionbdialog")
        self.radioButtondialog = QtWidgets.QRadioButton(Dialog)
```

```
        self.radioButtondialog.setGeometry(QtCore.QRect(660, 430, 111, 20))
        font = QtGui.QFont()
        font.setFamily("Arial")
        font.setPointSize(10)
        self.radioButtondialog.setFont(font)
        self.radioButtondialog.setObjectName("radioButtondialog")
        self.radioButton2dialog = QtWidgets.QRadioButton(Dialog)
        self.radioButton2dialog.setGeometry(QtCore.QRect(1020, 430, 111, 20))
        font = QtGui.QFont()
        font.setFamily("Arial")
        font.setPointSize(10)
        self.radioButton2dialog.setFont(font)
        self.radioButton2dialog.setObjectName("radioButton2dialog")
        self.resultadodialog = QtWidgets.QLabel(Dialog)
        self.resultadodialog.setGeometry(QtCore.QRect(20, 99, 491, 461))
        self.resultadodialog.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.resultadodialog.setText("")
        self.resultadodialog.setObjectName("resultadodialog")

        self.retranslateUi(Dialog)
        QtCore.QMetaObject.connectSlotsByName(Dialog)

    def retranslateUi(self, Dialog):
        _translate = QtCore.QCoreApplication.translate
        Dialog.setWindowTitle(_translate("Dialog", "Dialog"))
        self.selecciondialog.setText(_translate("Dialog", "Seleccionar"))
        self.radioButtondialog.setText(_translate("Dialog", "OPCIÓN A"))
        self.radioButton2dialog.setText(_translate("Dialog", "OPCIÓN B"))
if __name__ == "__main__":
    import sys
    app = QtWidgets.QApplication(sys.argv)
    Dialog = QtWidgets.QDialog()
    ui = Ui_Dialog()
    ui.setupUi(Dialog)
    Dialog.show()
    sys.exit(app.exec_()
```

ANEXO VII.

CODIGO DE LA VENTANA AUXILIAR II

```
from PyQt5 import QtCore, QtGui, QtWidgets

class Ui_Form(object):
    def setupUi(self, Form):
        Form.setObjectName("Form")
        Form.resize(1056, 601)
        self.resultadoswidget = QtWidgets.QLabel(Form)
        self.resultadoswidget.setGeometry(QtCore.QRect(39, 30, 481, 460))
        self.resultadoswidget.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.resultadoswidget.setText("")
        self.resultadoswidget.setObjectName("resultadoswidget")
        self.pushButton_27 = QtWidgets.QPushButton(Form)
        self.pushButton_27.setGeometry(QtCore.QRect(540, 110, 141, 16))
        font = QtGui.QFont()
        font.setFamily("Arial")
        font.setPointSize(10)
        self.pushButton_27.setFont(font)
        self.pushButton_27.setStyleSheet("border-radius:20px;")
        self.pushButton_27.setObjectName("pushButton_27")
        self.pushButton_29 = QtWidgets.QPushButton(Form)
        self.pushButton_29.setGeometry(QtCore.QRect(660, 60, 261, 31))
        font = QtGui.QFont()
        font.setFamily("Arial")
        font.setPointSize(12)
        font.setUnderline(False)
        self.pushButton_29.setFont(font)
        self.pushButton_29.setStyleSheet("border-radius:20px;")
        self.pushButton_29.setObjectName("pushButton_29")
        self.pushButton_31 = QtWidgets.QPushButton(Form)
        self.pushButton_31.setGeometry(QtCore.QRect(530, 190, 311, 20))
        font = QtGui.QFont()
```

```
font.setFamily("Arial")
font.setPointSize(10)
self.pushButton_31.setFont(font)
self.pushButton_31.setStyleSheet("border-radius:20px;")
self.pushButton_31.setObjectName("pushButton_31")
self.pushButton_33 = QtWidgets.QPushButton(Form)
self.pushButton_33.setGeometry(QtCore.QRect(550, 350, 141, 16))
font = QtGui.QFont()
font.setFamily("Arial")
font.setPointSize(10)
self.pushButton_33.setFont(font)
self.pushButton_33.setStyleSheet("border-radius:20px;")
self.pushButton_33.setObjectName("pushButton_33")
self.pushButton_34 = QtWidgets.QPushButton(Form)
self.pushButton_34.setGeometry(QtCore.QRect(520, 270, 311, 20))
font = QtGui.QFont()
font.setFamily("Arial")
font.setPointSize(10)
self.pushButton_34.setFont(font)
self.pushButton_34.setStyleSheet("border-radius:20px;")
self.pushButton_34.setObjectName("pushButton_34")
self.pushButton_42 = QtWidgets.QPushButton(Form)
self.pushButton_42.setGeometry(QtCore.QRect(540, 430, 301, 20))
font = QtGui.QFont()
font.setFamily("Arial")
font.setPointSize(10)
self.pushButton_42.setFont(font)
self.pushButton_42.setStyleSheet("border-radius:20px;")
self.pushButton_42.setObjectName("pushButton_42")
self.imagenfinalwidget = QtWidgets.QPushButton(Form)
self.imagenfinalwidget.setGeometry(QtCore.QRect(190, 520, 191, 41))
font = QtGui.QFont()
font.setFamily("Arial")
font.setPointSize(10)
self.imagenfinalwidget.setFont(font)
self.imagenfinalwidget.setObjectName("imagenfinalwidget")
```

```
        self.cerrarwidget = QtWidgets.QPushButton(Form)
        self.cerrarwidget.setGeometry(QtCore.QRect(670, 520, 181, 41))
        font = QtGui.QFont()
        font.setFamily("Arial")
        font.setPointSize(10)
        self.cerrarwidget.setFont(font)
        self.cerrarwidget.setObjectName("cerrarwidget")
        self.cargawidget = QtWidgets.QLabel(Form)
        self.cargawidget.setGeometry(QtCore.QRect(550, 130, 471, 31))
        font = QtGui.QFont()
        font.setFamily("Arial")
        font.setPointSize(10)
        self.cargawidget.setFont(font)
        self.cargawidget.setLayoutDirection(QtCore.Qt.LeftToRight)
        self.cargawidget.setStyleSheet("background-color: rgb(0, 0, 0);\n"
"background-color: rgb(255, 255, 255);")
        self.cargawidget.setText("")
        self.cargawidget.setAlignment(QtCore.Qt.AlignCenter)
        self.cargawidget.setObjectName("cargawidget")
        self.categoriawidget = QtWidgets.QLabel(Form)
        self.categoriawidget.setGeometry(QtCore.QRect(550, 210, 471, 31))
        font = QtGui.QFont()
        font.setFamily("Arial")
        font.setPointSize(10)
        self.categoriawidget.setFont(font)
        self.categoriawidget.setLayoutDirection(QtCore.Qt.LeftToRight)
        self.categoriawidget.setStyleSheet("background-color: rgb(0, 0, 0);\n"
"background-color: rgb(255, 255, 255);")
        self.categoriawidget.setText("")
        self.categoriawidget.setAlignment(QtCore.Qt.AlignCenter)
        self.categoriawidget.setObjectName("categoriawidget")
        self.magnitudwidget = QtWidgets.QLabel(Form)
        self.magnitudwidget.setGeometry(QtCore.QRect(550, 290, 471, 31))
        font = QtGui.QFont()
        font.setFamily("Arial")
        font.setPointSize(10)
```

```
        self.magnitudwidget.setFont(font)
        self.magnitudwidget.setLayoutDirection(QtCore.Qt.LeftToRight)
        self.magnitudwidget.setStyleSheet("background-color: rgb(0, 0, 0);\n"
"background-color: rgb(255, 255, 255);")
        self.magnitudwidget.setText("")
        self.magnitudwidget.setAlignment(QtCore.Qt.AlignCenter)
        self.magnitudwidget.setObjectName("magnitudwidget")
        self.aplicadawidget = QtWidgets.QLabel(Form)
        self.aplicadawidget.setGeometry(QtCore.QRect(550, 370, 471, 31))
        font = QtGui.QFont()
        font.setFamily("Arial")
        font.setPointSize(10)
        self.aplicadawidget.setFont(font)
        self.aplicadawidget.setLayoutDirection(QtCore.Qt.LeftToRight)
        self.aplicadawidget.setStyleSheet("background-color: rgb(0, 0, 0);\n"
"background-color: rgb(255, 255, 255);")
        self.aplicadawidget.setText("")
        self.aplicadawidget.setAlignment(QtCore.Qt.AlignCenter)
        self.aplicadawidget.setObjectName("aplicadawidget")
        self.concentradoreswidget = QtWidgets.QLabel(Form)
        self.concentradoreswidget.setGeometry(QtCore.QRect(550, 450, 471, 31))
        font = QtGui.QFont()
        font.setFamily("Arial")
        font.setPointSize(10)
        self.concentradoreswidget.setFont(font)
        self.concentradoreswidget.setLayoutDirection(QtCore.Qt.LeftToRight)
        self.concentradoreswidget.setStyleSheet("background-color: rgb(0, 0, 0);\n"
"background-color: rgb(255, 255, 255);")
        self.concentradoreswidget.setText("")
        self.concentradoreswidget.setAlignment(QtCore.Qt.AlignCenter)
        self.concentradoreswidget.setObjectName("concentradoreswidget")
        self.imagenfinalwidget_2 = QtWidgets.QPushButton(Form)
        self.imagenfinalwidget_2.setGeometry(QtCore.QRect(430, 520, 191, 41))
        font = QtGui.QFont()
        font.setFamily("Arial")
        font.setPointSize(10)
```

```
        self.imagenfinalwidget_2.setFont(font)
        self.imagenfinalwidget_2.setObjectName("imagenfinalwidget_2")

        self.retranslateUi(Form)
        QtCore.QMetaObject.connectSlotsByName(Form)

    def retranslateUi(self, Form):
        _translate = QtCore.QCoreApplication.translate
        Form.setWindowTitle(_translate("Form", "Form"))
        self.pushButton_27.setText(_translate("Form", "TIPO DE CARGA"))
        self.pushButton_29.setText(_translate("Form", " RESULTADOS OBTENIDOS"))
            self.pushButton_31.setText(_translate("Form", "CATEGORIA DE FALLA POR
FATIGA"))
        self.pushButton_33.setText(_translate("Form", "CARGA APLICADA"))
        self.pushButton_34.setText(_translate("Form", "MAGNITUD DE LA CARGA FINAL"))
                self.pushButton_42.setText(_translate("Form", "CONCENTRADORES DE
ESFUERZO"))
        self.imagenfinalwidget.setText(_translate("Form", "Guardar Imagen"))
        self.cerrarwidget.setText(_translate("Form", "Cerrar"))
        self.imagenfinalwidget_2.setText(_translate("Form", "Imprimir Informe"))

if __name__ == "__main__":
    import sys
    app = QtWidgets.QApplication(sys.argv)
    Form = QtWidgets.QWidget()
    ui = Ui_Form()
    ui.setupUi(Form)
    Form.show()
    sys.exit(app.exec_())
```

ANEXO VIII.

CODIGO DE LA INTERFAZ GRAFICA

```
from PyQt5 import QtCore, QtGui, QtWidgets

class Ui_MainWindow(object):
    def setupUi(self, MainWindow):
        MainWindow.setObjectName("MainWindow")
        MainWindow.resize(1713, 911)
        MainWindow.setCursor(QtGui.QCursor(QtCore.Qt.ArrowCursor))
        MainWindow.setStyleSheet("background-color: rgb(189, 0, 0);\n"
"background-color: rgb(206, 206, 206);")
        self.centralwidget = QtWidgets.QWidget(MainWindow)
        self.centralwidget.setObjectName("centralwidget")
        self.pushButton = QtWidgets.QPushButton(self.centralwidget)
        self.pushButton.setGeometry(QtCore.QRect(140, 10, 120, 40))
        font = QtGui.QFont()
        font.setFamily("Arial")
        font.setPointSize(10)
        self.pushButton.setFont(font)
        self.pushButton.setStyleSheet("")
        self.pushButton.setObjectName("pushButton")
        self.pushButton_2 = QtWidgets.QPushButton(self.centralwidget)
        self.pushButton_2.setGeometry(QtCore.QRect(310, 10, 120, 40))
        font = QtGui.QFont()
        font.setFamily("Arial")
        font.setPointSize(10)
        self.pushButton_2.setFont(font)
        self.pushButton_2.setStyleSheet("")
        self.pushButton_2.setObjectName("pushButton_2")
        self.pushButton_4 = QtWidgets.QPushButton(self.centralwidget)
        self.pushButton_4.setGeometry(QtCore.QRect(480, 10, 140, 40))
        font = QtGui.QFont()
        font.setFamily("Arial")
```

```
        font.setPointSize(10)
        self.pushButton_4.setFont(font)
        self.pushButton_4.setObjectName("pushButton_4")
        self.groupBox = QtWidgets.QGroupBox(self.centralwidget)
        self.groupBox.setGeometry(QtCore.QRect(20, 180, 391, 141))
        font = QtGui.QFont()
        font.setFamily("Arial")
        font.setPointSize(10)
        self.groupBox.setFont(font)
        self.groupBox.setStyleSheet("background-color: rgb(255, 255, 255);\n"
"background-color: rgb(206, 206, 206);")
        self.groupBox.setObjectName("groupBox")
        self.pushButton_7 = QtWidgets.QPushButton(self.groupBox)
        self.pushButton_7.setGeometry(QtCore.QRect(0, 70, 101, 41))
        font = QtGui.QFont()
        font.setFamily("Arial")
        font.setPointSize(10)
        self.pushButton_7.setFont(font)
        self.pushButton_7.setStyleSheet("border-radius:20px;")
        self.pushButton_7.setObjectName("pushButton_7")
        self.horizontalSlider_2 = QtWidgets.QSlider(self.groupBox)
        self.horizontalSlider_2.setGeometry(QtCore.QRect(100, 80, 181, 31))
        self.horizontalSlider_2.setOrientation(QtCore.Qt.Horizontal)
        self.horizontalSlider_2.setObjectName("horizontalSlider_2")
        self.horizontalSlider = QtWidgets.QSlider(self.groupBox)
        self.horizontalSlider.setGeometry(QtCore.QRect(100, 30, 181, 31))
        self.horizontalSlider.setStyleSheet("border-bottom-color: rgb(255, 255, 0);")
        self.horizontalSlider.setOrientation(QtCore.Qt.Horizontal)
        self.horizontalSlider.setObjectName("horizontalSlider")
        self.label_4 = QtWidgets.QLabel(self.groupBox)
        self.label_4.setGeometry(QtCore.QRect(300, 30, 51, 31))
        font = QtGui.QFont()
        font.setFamily("Arial")
        font.setPointSize(10)
        self.label_4.setFont(font)
        self.label_4.setLayoutDirection(QtCore.Qt.LeftToRight)
```

```
        self.label_4.setStyleSheet("background-color: rgb(0, 0, 0);\n"
"background-color: rgb(255, 255, 255);")
        self.label_4.setText("")
        self.label_4.setAlignment(QtCore.Qt.AlignCenter)
        self.label_4.setObjectName("label_4")
        self.label_5 = QtWidgets.QLabel(self.groupBox)
        self.label_5.setGeometry(QtCore.QRect(300, 80, 51, 31))
        font = QtGui.QFont()
        font.setFamily("Arial")
        font.setPointSize(10)
        self.label_5.setFont(font)
        self.label_5.setStyleSheet("background-color: rgb(0, 0, 0);\n"
"background-color: rgb(255, 255, 255);")
        self.label_5.setText("")
        self.label_5.setAlignment(QtCore.Qt.AlignCenter)
        self.label_5.setObjectName("label_5")
        self.pushButton_10 = QtWidgets.QPushButton(self.groupBox)
        self.pushButton_10.setGeometry(QtCore.QRect(0, 20, 91, 41))
        font = QtGui.QFont()
        font.setFamily("Arial")
        font.setPointSize(10)
        self.pushButton_10.setFont(font)
        self.pushButton_10.setStyleSheet("border-radius:20px;")
        self.pushButton_10.setObjectName("pushButton_10")
        self.pushButton_31 = QtWidgets.QPushButton(self.groupBox)
        self.pushButton_31.setGeometry(QtCore.QRect(350, 80, 21, 31))
        font = QtGui.QFont()
        font.setFamily("Arial")
        font.setPointSize(9)
        self.pushButton_31.setFont(font)
        self.pushButton_31.setStyleSheet("border-radius:20px;")
        self.pushButton_31.setObjectName("pushButton_31")
        self.groupBox_2 = QtWidgets.QGroupBox(self.centralwidget)
        self.groupBox_2.setGeometry(QtCore.QRect(20, 60, 341, 101))
        font = QtGui.QFont()
        font.setFamily("Arial")
```

```
        font.setPointSize(10)
        self.groupBox_2.setFont(font)
        self.groupBox_2.setStyleSheet("background-color: rgb(206, 206, 206);\n"
"")
        self.groupBox_2.setObjectName("groupBox_2")
        self.radioButton = QtWidgets.QRadioButton(self.groupBox_2)
        self.radioButton.setGeometry(QtCore.QRect(10, 30, 161, 21))
        font = QtGui.QFont()
        font.setFamily("Arial")
        font.setPointSize(10)
        self.radioButton.setFont(font)
        self.radioButton.setObjectName("radioButton")
        self.radioButton_2 = QtWidgets.QRadioButton(self.groupBox_2)
        self.radioButton_2.setGeometry(QtCore.QRect(10, 50, 191, 31))
        font = QtGui.QFont()
        font.setFamily("Arial")
        font.setPointSize(10)
        self.radioButton_2.setFont(font)
        self.radioButton_2.setObjectName("radioButton_2")
        self.pushButton_8 = QtWidgets.QPushButton(self.groupBox_2)
        self.pushButton_8.setGeometry(QtCore.QRect(210, 30, 111, 40))
        font = QtGui.QFont()
        font.setFamily("Arial")
        font.setPointSize(10)
        self.pushButton_8.setFont(font)
        self.pushButton_8.setObjectName("pushButton_8")
        self.groupBox_7 = QtWidgets.QGroupBox(self.centralwidget)
        self.groupBox_7.setGeometry(QtCore.QRect(420, 470, 321, 261))
        font = QtGui.QFont()
        font.setFamily("Arial")
        font.setPointSize(10)
        self.groupBox_7.setFont(font)
        self.groupBox_7.setStyleSheet("background-color: rgb(255, 255, 255);\n"
"background-color: rgb(206, 206, 206);")
        self.groupBox_7.setObjectName("groupBox_7")
        self.groupBox_6 = QtWidgets.QGroupBox(self.groupBox_7)
```

```
        self.groupBox_6.setGeometry(QtCore.QRect(10, 40, 301, 111))
        font = QtGui.QFont()
        font.setFamily("Arial")
        font.setPointSize(10)
        self.groupBox_6.setFont(font)
        self.groupBox_6.setStyleSheet("background-color: rgb(255, 255, 255);\n"
"background-color: rgb(206, 206, 206);")
        self.groupBox_6.setObjectName("groupBox_6")
        self.pushButton_13 = QtWidgets.QPushButton(self.groupBox_6)
        self.pushButton_13.setGeometry(QtCore.QRect(0, 60, 91, 41))
        font = QtGui.QFont()
        font.setFamily("Arial")
        font.setPointSize(10)
        self.pushButton_13.setFont(font)
        self.pushButton_13.setStyleSheet("border-radius:20px;")
        self.pushButton_13.setObjectName("pushButton_13")
        self.horizontalSlider_5 = QtWidgets.QSlider(self.groupBox_6)
        self.horizontalSlider_5.setGeometry(QtCore.QRect(110, 30, 121, 31))
        self.horizontalSlider_5.setOrientation(QtCore.Qt.Horizontal)
        self.horizontalSlider_5.setObjectName("horizontalSlider_5")
        self.horizontalSlider_6 = QtWidgets.QSlider(self.groupBox_6)
        self.horizontalSlider_6.setGeometry(QtCore.QRect(110, 70, 121, 31))
        self.horizontalSlider_6.setStyleSheet("border-bottom-color: rgb(255, 255, 0);")
        self.horizontalSlider_6.setOrientation(QtCore.Qt.Horizontal)
        self.horizontalSlider_6.setObjectName("horizontalSlider_6")
        self.label_10 = QtWidgets.QLabel(self.groupBox_6)
        self.label_10.setGeometry(QtCore.QRect(240, 30, 51, 31))
        font = QtGui.QFont()
        font.setFamily("Arial")
        font.setPointSize(10)
        self.label_10.setFont(font)
        self.label_10.setLayoutDirection(QtCore.Qt.LeftToRight)
        self.label_10.setStyleSheet("background-color: rgb(0, 0, 0);\n"
"background-color: rgb(255, 255, 255);")
        self.label_10.setText("")
        self.label_10.setAlignment(QtCore.Qt.AlignCenter)
```

```
        self.label_10.setObjectName("label_10")
        self.label_11 = QtWidgets.QLabel(self.groupBox_6)
        self.label_11.setGeometry(QtCore.QRect(240, 70, 51, 31))
        font = QtGui.QFont()
        font.setFamily("Arial")
        font.setPointSize(10)
        self.label_11.setFont(font)
        self.label_11.setStyleSheet("background-color: rgb(0, 0, 0);\n"
"background-color: rgb(255, 255, 255);")
        self.label_11.setText("")
        self.label_11.setAlignment(QtCore.Qt.AlignCenter)
        self.label_11.setObjectName("label_11")
        self.pushButton_14 = QtWidgets.QPushButton(self.groupBox_6)
        self.pushButton_14.setGeometry(QtCore.QRect(0, 20, 91, 41))
        font = QtGui.QFont()
        font.setFamily("Arial")
        font.setPointSize(10)
        self.pushButton_14.setFont(font)
        self.pushButton_14.setStyleSheet("border-radius:20px;")
        self.pushButton_14.setObjectName("pushButton_14")
        self.groupBox_5 = QtWidgets.QGroupBox(self.groupBox_7)
        self.groupBox_5.setGeometry(QtCore.QRect(10, 160, 301, 81))
        font = QtGui.QFont()
        font.setFamily("Arial")
        font.setPointSize(10)
        self.groupBox_5.setFont(font)
        self.groupBox_5.setStyleSheet("\n"
"background-color: rgb(255, 255, 255);\n"
"background-color: rgb(206, 206, 206);")
        self.groupBox_5.setObjectName("groupBox_5")
        self.horizontalSlider_8 = QtWidgets.QSlider(self.groupBox_5)
        self.horizontalSlider_8.setGeometry(QtCore.QRect(110, 30, 121, 31))
        self.horizontalSlider_8.setStyleSheet("border-bottom-color: rgb(255, 255, 0);")
        self.horizontalSlider_8.setOrientation(QtCore.Qt.Horizontal)
        self.horizontalSlider_8.setObjectName("horizontalSlider_8")
        self.pushButton_21 = QtWidgets.QPushButton(self.groupBox_5)
```

```
        self.pushButton_21.setGeometry(QtCore.QRect(0, 30, 101, 41))
        font = QtGui.QFont()
        font.setFamily("Arial")
        font.setPointSize(10)
        self.pushButton_21.setFont(font)
        self.pushButton_21.setStyleSheet("border-radius:20px;")
        self.pushButton_21.setObjectName("pushButton_21")
        self.label_13 = QtWidgets.QLabel(self.groupBox_5)
        self.label_13.setGeometry(QtCore.QRect(240, 30, 51, 31))
        font = QtGui.QFont()
        font.setFamily("Arial")
        font.setPointSize(10)
        self.label_13.setFont(font)
        self.label_13.setLayoutDirection(QtCore.Qt.LeftToRight)
        self.label_13.setStyleSheet("background-color: rgb(0, 0, 0);\n"
"background-color: rgb(255, 255, 255);")
        self.label_13.setText("")
        self.label_13.setAlignment(QtCore.Qt.AlignCenter)
        self.label_13.setObjectName("label_13")
        self.groupBox_3 = QtWidgets.QGroupBox(self.centralwidget)
        self.groupBox_3.setGeometry(QtCore.QRect(370, 60, 371, 101))
        font = QtGui.QFont()
        font.setFamily("Arial")
        font.setPointSize(10)
        self.groupBox_3.setFont(font)
        self.groupBox_3.setStyleSheet("background-color: rgb(255, 255, 255);\n"
"background-color: rgb(206, 206, 206);")
        self.groupBox_3.setObjectName("groupBox_3")
        self.pushButton_22 = QtWidgets.QPushButton(self.groupBox_3)
        self.pushButton_22.setGeometry(QtCore.QRect(230, 30, 111, 40))
        font = QtGui.QFont()
        font.setFamily("Arial")
        font.setPointSize(10)
        self.pushButton_22.setFont(font)
        self.pushButton_22.setObjectName("pushButton_22")
        self.pushButton_24 = QtWidgets.QPushButton(self.groupBox_3)
```

```
        self.pushButton_24.setGeometry(QtCore.QRect(40, 30, 161, 40))
        font = QtGui.QFont()
        font.setFamily("Arial")
        font.setPointSize(10)
        self.pushButton_24.setFont(font)
        self.pushButton_24.setObjectName("pushButton_24")
        self.groupBox_4 = QtWidgets.QGroupBox(self.centralwidget)
        self.groupBox_4.setGeometry(QtCore.QRect(20, 590, 391, 141))
        font = QtGui.QFont()
        font.setFamily("Arial")
        font.setPointSize(10)
        self.groupBox_4.setFont(font)
        self.groupBox_4.setStyleSheet("background-color: rgb(206, 206, 206);\n"
"")
        self.groupBox_4.setObjectName("groupBox_4")
        self.radioButton_6 = QtWidgets.QRadioButton(self.groupBox_4)
        self.radioButton_6.setGeometry(QtCore.QRect(20, 110, 141, 21))
        font = QtGui.QFont()
        font.setFamily("Arial")
        font.setPointSize(10)
        self.radioButton_6.setFont(font)
        self.radioButton_6.setObjectName("radioButton_6")
        self.radioButton_7 = QtWidgets.QRadioButton(self.groupBox_4)
        self.radioButton_7.setGeometry(QtCore.QRect(20, 60, 171, 31))
        font = QtGui.QFont()
        font.setFamily("Arial")
        font.setPointSize(10)
        self.radioButton_7.setFont(font)
        self.radioButton_7.setObjectName("radioButton_7")
        self.pushButton_9 = QtWidgets.QPushButton(self.groupBox_4)
        self.pushButton_9.setGeometry(QtCore.QRect(240, 20, 111, 30))
        font = QtGui.QFont()
        font.setFamily("Arial")
        font.setPointSize(10)
        self.pushButton_9.setFont(font)
        self.pushButton_9.setObjectName("pushButton_9")
```

```
        self.radioButton_8 = QtWidgets.QRadioButton(self.groupBox_4)
        self.radioButton_8.setGeometry(QtCore.QRect(20, 20, 171, 31))
        font = QtGui.QFont()
        font.setFamily("Arial")
        font.setPointSize(10)
        self.radioButton_8.setFont(font)
        self.radioButton_8.setObjectName("radioButton_8")
        self.pushButton_16 = QtWidgets.QPushButton(self.groupBox_4)
        self.pushButton_16.setGeometry(QtCore.QRect(240, 60, 111, 30))
        font = QtGui.QFont()
        font.setFamily("Arial")
        font.setPointSize(10)
        self.pushButton_16.setFont(font)
        self.pushButton_16.setObjectName("pushButton_16")
        self.pushButton_26 = QtWidgets.QPushButton(self.groupBox_4)
        self.pushButton_26.setGeometry(QtCore.QRect(240, 100, 111, 30))
        font = QtGui.QFont()
        font.setFamily("Arial")
        font.setPointSize(10)
        self.pushButton_26.setFont(font)
        self.pushButton_26.setObjectName("pushButton_26")
        self.groupBox_8 = QtWidgets.QGroupBox(self.centralwidget)
        self.groupBox_8.setGeometry(QtCore.QRect(20, 340, 391, 231))
        font = QtGui.QFont()
        font.setFamily("Arial")
        font.setPointSize(10)
        self.groupBox_8.setFont(font)
        self.groupBox_8.setStyleSheet("background-color: rgb(255, 255, 255);\n"
"background-color: rgb(206, 206, 206);")
        self.groupBox_8.setObjectName("groupBox_8")
        self.pushButton_18 = QtWidgets.QPushButton(self.groupBox_8)
        self.pushButton_18.setGeometry(QtCore.QRect(170, 100, 111, 35))
        font = QtGui.QFont()
        font.setFamily("Arial")
        font.setPointSize(10)
        self.pushButton_18.setFont(font)
```

```
        self.pushButton_18.setFocusPolicy(QtCore.Qt.TabFocus)
        self.pushButton_18.setObjectName("pushButton_18")
        self.label_7 = QtWidgets.QLabel(self.groupBox_8)
        self.label_7.setGeometry(QtCore.QRect(164, 190, 71, 31))
        font = QtGui.QFont()
        font.setFamily("Arial")
        font.setPointSize(10)
        self.label_7.setFont(font)
        self.label_7.setStyleSheet("background-color: rgb(0, 0, 0);\n"
"background-color: rgb(255, 255, 255);")
        self.label_7.setText("")
        self.label_7.setAlignment(QtCore.Qt.AlignCenter)
        self.label_7.setObjectName("label_7")
        self.pushButton_15 = QtWidgets.QPushButton(self.groupBox_8)
        self.pushButton_15.setGeometry(QtCore.QRect(240, 150, 21, 31))
        font = QtGui.QFont()
        font.setFamily("Arial")
        font.setPointSize(9)
        self.pushButton_15.setFont(font)
        self.pushButton_15.setStyleSheet("border-radius:20px;")
        self.pushButton_15.setObjectName("pushButton_15")
        self.label_6 = QtWidgets.QLabel(self.groupBox_8)
        self.label_6.setGeometry(QtCore.QRect(164, 150, 71, 31))
        font = QtGui.QFont()
        font.setFamily("Arial")
        font.setPointSize(10)
        self.label_6.setFont(font)
        self.label_6.setLayoutDirection(QtCore.Qt.LeftToRight)
        self.label_6.setStyleSheet("background-color: rgb(0, 0, 0);\n"
"background-color: rgb(255, 255, 255);")
        self.label_6.setText("")
        self.label_6.setAlignment(QtCore.Qt.AlignCenter)
        self.label_6.setObjectName("label_6")
        self.pushButton_11 = QtWidgets.QPushButton(self.groupBox_8)
        self.pushButton_11.setGeometry(QtCore.QRect(10, 150, 151, 41))
        font = QtGui.QFont()
```

```
font.setFamily("Arial")
font.setPointSize(10)
self.pushButton_11.setFont(font)
self.pushButton_11.setStyleSheet("border-radius:20px;")
self.pushButton_11.setObjectName("pushButton_11")
self.pushButton_17 = QtWidgets.QPushButton(self.groupBox_8)
self.pushButton_17.setGeometry(QtCore.QRect(240, 190, 21, 31))
font = QtGui.QFont()
font.setFamily("Arial")
font.setPointSize(9)
self.pushButton_17.setFont(font)
self.pushButton_17.setStyleSheet("border-radius:20px;")
self.pushButton_17.setObjectName("pushButton_17")
self.pushButton_19 = QtWidgets.QPushButton(self.groupBox_8)
self.pushButton_19.setGeometry(QtCore.QRect(270, 170, 111, 35))
font = QtGui.QFont()
font.setFamily("Arial")
font.setPointSize(10)
self.pushButton_19.setFont(font)
self.pushButton_19.setFocusPolicy(QtCore.Qt.TabFocus)
self.pushButton_19.setObjectName("pushButton_19")
self.pushButton_20 = QtWidgets.QPushButton(self.groupBox_8)
self.pushButton_20.setGeometry(QtCore.QRect(10, 110, 131, 20))
font = QtGui.QFont()
font.setFamily("Arial")
font.setPointSize(10)
self.pushButton_20.setFont(font)
self.pushButton_20.setStyleSheet("border-radius:20px;")
self.pushButton_20.setObjectName("pushButton_20")
self.pushButton_23 = QtWidgets.QPushButton(self.groupBox_8)
self.pushButton_23.setGeometry(QtCore.QRect(290, 100, 81, 35))
font = QtGui.QFont()
font.setFamily("Arial")
font.setPointSize(10)
self.pushButton_23.setFont(font)
self.pushButton_23.setFocusPolicy(QtCore.Qt.TabFocus)
```

```
self.pushButton_23.setObjectName("pushButton_23")
self.pushButton_12 = QtWidgets.QPushButton(self.groupBox_8)
self.pushButton_12.setGeometry(QtCore.QRect(10, 190, 111, 41))
font = QtGui.QFont()
font.setFamily("Arial")
font.setPointSize(10)
self.pushButton_12.setFont(font)
self.pushButton_12.setStyleSheet("border-radius:20px;")
self.pushButton_12.setObjectName("pushButton_12")
self.pushButton_27 = QtWidgets.QPushButton(self.groupBox_8)
self.pushButton_27.setGeometry(QtCore.QRect(0, 50, 171, 41))
font = QtGui.QFont()
font.setFamily("Arial")
font.setPointSize(10)
self.pushButton_27.setFont(font)
self.pushButton_27.setStyleSheet("border-radius:20px;")
self.pushButton_27.setObjectName("pushButton_27")
self.pushButton_28 = QtWidgets.QPushButton(self.groupBox_8)
self.pushButton_28.setGeometry(QtCore.QRect(170, 50, 111, 35))
font = QtGui.QFont()
font.setFamily("Arial")
font.setPointSize(10)
self.pushButton_28.setFont(font)
self.pushButton_28.setFocusPolicy(QtCore.Qt.TabFocus)
self.pushButton_28.setObjectName("pushButton_28")
self.pushButton_29 = QtWidgets.QPushButton(self.groupBox_8)
self.pushButton_29.setGeometry(QtCore.QRect(290, 50, 81, 35))
font = QtGui.QFont()
font.setFamily("Arial")
font.setPointSize(10)
self.pushButton_29.setFont(font)
self.pushButton_29.setFocusPolicy(QtCore.Qt.TabFocus)
self.pushButton_29.setObjectName("pushButton_29")
self.radioButton_10 = QtWidgets.QRadioButton(self.groupBox_8)
self.radioButton_10.setGeometry(QtCore.QRect(10, 20, 141, 31))
font = QtGui.QFont()
```

```
font.setFamily("Arial")
font.setPointSize(10)
self.radioButton_10.setFont(font)
self.radioButton_10.setObjectName("radioButton_10")
self.groupBox_9 = QtWidgets.QGroupBox(self.centralwidget)
self.groupBox_9.setGeometry(QtCore.QRect(750, 730, 941, 141))
font = QtGui.QFont()
font.setFamily("Arial")
font.setPointSize(10)
self.groupBox_9.setFont(font)
self.groupBox_9.setObjectName("groupBox_9")
self.label_2 = QtWidgets.QLabel(self.groupBox_9)
self.label_2.setGeometry(QtCore.QRect(10, 20, 921, 111))
font = QtGui.QFont()
font.setFamily("Arial")
self.label_2.setFont(font)
self.label_2.setStyleSheet("background-color: rgb(255, 255, 255);")
self.label_2.setText("")
self.label_2.setObjectName("label_2")
self.groupBox_10 = QtWidgets.QGroupBox(self.centralwidget)
self.groupBox_10.setGeometry(QtCore.QRect(420, 180, 321, 271))
font = QtGui.QFont()
font.setFamily("Arial")
font.setPointSize(10)
self.groupBox_10.setFont(font)
self.groupBox_10.setObjectName("groupBox_10")
self.label_3 = QtWidgets.QLabel(self.groupBox_10)
self.label_3.setGeometry(QtCore.QRect(10, 20, 300, 230))
self.label_3.setStyleSheet("background-color: rgb(255, 255, 255);")
self.label_3.setText("")
self.label_3.setObjectName("label_3")
self.groupBox_11 = QtWidgets.QGroupBox(self.centralwidget)
self.groupBox_11.setGeometry(QtCore.QRect(20, 740, 721, 131))
font = QtGui.QFont()
font.setFamily("Arial")
font.setPointSize(10)
```

```
self.groupBox_11.setFont(font)
self.groupBox_11.setObjectName("groupBox_11")
self.pushButton_25 = QtWidgets.QPushButton(self.groupBox_11)
self.pushButton_25.setGeometry(QtCore.QRect(10, 40, 221, 61))
font = QtGui.QFont()
font.setFamily("Arial")
font.setPointSize(10)
self.pushButton_25.setFont(font)
self.pushButton_25.setObjectName("pushButton_25")
self.radioButton_9 = QtWidgets.QRadioButton(self.groupBox_11)
self.radioButton_9.setGeometry(QtCore.QRect(260, 20, 171, 31))
font = QtGui.QFont()
font.setFamily("Arial")
font.setPointSize(10)
self.radioButton_9.setFont(font)
self.radioButton_9.setObjectName("radioButton_9")
self.radioButton_12 = QtWidgets.QRadioButton(self.groupBox_11)
self.radioButton_12.setGeometry(QtCore.QRect(260, 80, 121, 31))
font = QtGui.QFont()
font.setFamily("Arial")
font.setPointSize(10)
self.radioButton_12.setFont(font)
self.radioButton_12.setObjectName("radioButton_12")
self.radioButton_13 = QtWidgets.QRadioButton(self.groupBox_11)
self.radioButton_13.setGeometry(QtCore.QRect(420, 20, 171, 31))
font = QtGui.QFont()
font.setFamily("Arial")
font.setPointSize(10)
self.radioButton_13.setFont(font)
self.radioButton_13.setObjectName("radioButton_13")
self.radioButton_11 = QtWidgets.QRadioButton(self.groupBox_11)
self.radioButton_11.setGeometry(QtCore.QRect(260, 50, 121, 31))
font = QtGui.QFont()
font.setFamily("Arial")
font.setPointSize(10)
self.radioButton_11.setFont(font)
```

```
        self.radioButton_11.setObjectName("radioButton_11")
        self.radioButton_15 = QtWidgets.QRadioButton(self.groupBox_11)
        self.radioButton_15.setGeometry(QtCore.QRect(420, 80, 171, 31))
        font = QtGui.QFont()
        font.setFamily("Arial")
        font.setPointSize(10)
        self.radioButton_15.setFont(font)
        self.radioButton_15.setObjectName("radioButton_15")
        self.radioButton_17 = QtWidgets.QRadioButton(self.groupBox_11)
        self.radioButton_17.setGeometry(QtCore.QRect(580, 20, 131, 31))
        font = QtGui.QFont()
        font.setFamily("Arial")
        font.setPointSize(10)
        self.radioButton_17.setFont(font)
        self.radioButton_17.setObjectName("radioButton_17")
        self.radioButton_16 = QtWidgets.QRadioButton(self.groupBox_11)
        self.radioButton_16.setGeometry(QtCore.QRect(420, 50, 171, 31))
        font = QtGui.QFont()
        font.setFamily("Arial")
        font.setPointSize(10)
        self.radioButton_16.setFont(font)
        self.radioButton_16.setObjectName("radioButton_16")
        self.groupBox_12 = QtWidgets.QGroupBox(self.centralwidget)
        self.groupBox_12.setGeometry(QtCore.QRect(750, 0, 941, 721))
        font = QtGui.QFont()
        font.setFamily("Arial")
        font.setPointSize(10)
        self.groupBox_12.setFont(font)
        self.groupBox_12.setStyleSheet("background-color: rgb(255, 255, 255);\n"
"background-color: rgb(206, 206, 206);")
        self.groupBox_12.setObjectName("groupBox_12")
        self.pushButton_30 = QtWidgets.QPushButton(self.groupBox_12)
        self.pushButton_30.setGeometry(QtCore.QRect(0, 70, 101, 41))
        font = QtGui.QFont()
        font.setFamily("Arial")
        font.setPointSize(10)
```

```
        self.pushButton_30.setFont(font)
        self.pushButton_30.setStyleSheet("border-radius:20px;")
        self.pushButton_30.setText("")
        self.pushButton_30.setObjectName("pushButton_30")
        self.label = QtWidgets.QLabel(self.groupBox_12)
        self.label.setGeometry(QtCore.QRect(10, 20, 920, 690))
        font = QtGui.QFont()
        font.setFamily("Niagara Solid")
        self.label.setFont(font)
        self.label.setStyleSheet("background-color: rgb(255, 255, 255);\n"
"background-color: rgb(255, 255, 255);")
        self.label.setText("")
        self.label.setPixmap(QtGui.QPixmap("Zoom.png"))
        self.label.setScaledContents(False)
        self.label.setObjectName("label")
        self.radioButton_14 = QtWidgets.QRadioButton(self.groupBox_12)
        self.radioButton_14.setGeometry(QtCore.QRect(330, 760, 191, 31))
        font = QtGui.QFont()
        font.setFamily("Arial")
        font.setPointSize(10)
        self.radioButton_14.setFont(font)
        self.radioButton_14.setObjectName("radioButton_14")
        self.groupBox_7.raise_()
        self.pushButton.raise_()
        self.pushButton_2.raise_()
        self.pushButton_4.raise_()
        self.groupBox.raise_()
        self.groupBox_2.raise_()
        self.groupBox_3.raise_()
        self.groupBox_4.raise_()
        self.groupBox_8.raise_()
        self.groupBox_9.raise_()
        self.groupBox_10.raise_()
        self.groupBox_11.raise_()
        self.groupBox_12.raise_()
        MainWindow.setCentralWidget(self.centralwidget)
```

```
        self.statusbar = QtWidgets.QStatusBar(MainWindow)
        self.statusbar.setObjectName("statusbar")
        MainWindow.setStatusBar(self.statusbar)
        self.actionAbrir = QtWidgets.QAction(MainWindow)
        self.actionAbrir.setObjectName("actionAbrir")
        self.actionSalir = QtWidgets.QAction(MainWindow)
        self.actionSalir.setObjectName("actionSalir")

        self.retranslateUi(MainWindow)
        self.pushButton_23.clicked.connect(self.label.clear)
        self.pushButton_22.clicked.connect(self.label.clear)
        self.pushButton_2.clicked.connect(self.label.clear)
        self.pushButton_4.clicked.connect(self.label_3.clear)
        self.pushButton_9.clicked.connect(self.label.clear)
        self.horizontalSlider_2.actionTriggered['int'].connect(self.label_3.clear)
        self.horizontalSlider.actionTriggered['int'].connect(self.label_3.clear)
        self.pushButton_16.clicked.connect(self.label.clear)
        self.pushButton_26.clicked.connect(self.label.clear)
        self.pushButton_29.clicked.connect(self.label.clear)
        QtCore.QMetaObject.connectSlotsByName(MainWindow)

    def retranslateUi(self, MainWindow):
        _translate = QtCore.QCoreApplication.translate
        MainWindow.setWindowTitle(_translate("MainWindow", "FIM Fatigue Failure"))
        self.pushButton.setText(_translate("MainWindow", "Abrir Imagen"))
        self.pushButton_2.setText(_translate("MainWindow", "Limpiar"))
        self.pushButton_4.setText(_translate("MainWindow", "Escala de Grises"))
        self.groupBox.setTitle(_translate("MainWindow", "Contraste y Saturación "))
        self.pushButton_7.setText(_translate("MainWindow", "Contraste"))
        self.pushButton_10.setText(_translate("MainWindow", "Brillo"))
        self.pushButton_31.setText(_translate("MainWindow", "%"))
        self.groupBox_2.setTitle(_translate("MainWindow", "Sección del elemento"))
        self.radioButton.setText(_translate("MainWindow", "Sección Circular "))
        self.radioButton_2.setText(_translate("MainWindow", "Sección Rectangular"))
        self.pushButton_8.setText(_translate("MainWindow", "Recortar "))
        self.groupBox_7.setTitle(_translate("MainWindow", "Detección de Bordes "))
```

```
        self.groupBox_6.setTitle(_translate("MainWindow", "Método I"))
        self.pushButton_13.setText(_translate("MainWindow", "Valor Mín"))
        self.pushButton_14.setText(_translate("MainWindow", "Valor Máx"))
        self.groupBox_5.setTitle(_translate("MainWindow", "Método II"))
        self.pushButton_21.setText(_translate("MainWindow", "Valor Umbral"))
        self.groupBox_3.setTitle(_translate("MainWindow", "Acercamiento"))
        self.pushButton_22.setText(_translate("MainWindow", "Borrar"))
        self.pushButton_24.setText(_translate("MainWindow", "Seleccionar Área"))
        self.groupBox_4.setTitle(_translate("MainWindow", "Marcas de trinquete y origen"))
        self.radioButton_6.setText(_translate("MainWindow", "Bisectriz "))
        self.radioButton_7.setText(_translate("MainWindow", "Seleccionar Origen"))
        self.pushButton_9.setText(_translate("MainWindow", "Borrar"))
        self.radioButton_8.setText(_translate("MainWindow", "Marcas de trinquete"))
        self.pushButton_16.setText(_translate("MainWindow", "Borrar"))
        self.pushButton_26.setText(_translate("MainWindow", "Borrar"))
          self.groupBox_8.setTitle(_translate("MainWindow", "Porcentaje de las áreas de la zona de fractura"))
        self.pushButton_18.setText(_translate("MainWindow", "Seleccionar"))
        self.pushButton_15.setText(_translate("MainWindow", "%"))
        self.pushButton_11.setText(_translate("MainWindow", "Área fractura final "))
        self.pushButton_17.setText(_translate("MainWindow", "%"))
        self.pushButton_19.setText(_translate("MainWindow", "Seleccionar"))
        self.pushButton_20.setText(_translate("MainWindow", "Área de fractura"))
        self.pushButton_23.setText(_translate("MainWindow", "Borrar "))
        self.pushButton_12.setText(_translate("MainWindow", "Área de fatiga"))
        self.pushButton_27.setText(_translate("MainWindow", "Cortar Área Interna"))
        self.pushButton_28.setText(_translate("MainWindow", "Seleccionar"))
        self.pushButton_29.setText(_translate("MainWindow", "Borrar "))
        self.radioButton_10.setText(_translate("MainWindow", "Eje Hueco"))
        self.groupBox_9.setTitle(_translate("MainWindow", "Consejos Rápidos"))
        self.groupBox_10.setTitle(_translate("MainWindow", "Visor de ayuda"))
        self.groupBox_11.setTitle(_translate("MainWindow", "Resultados Obtenidos "))
        self.pushButton_25.setText(_translate("MainWindow", "Análisis de Resultados "))
        self.radioButton_9.setText(_translate("MainWindow", "Pregunta #1"))
        self.radioButton_12.setText(_translate("MainWindow", "Pregunta #3"))
        self.radioButton_13.setText(_translate("MainWindow", "Pregunta #4"))
```

```
        self.radioButton_11.setText(_translate("MainWindow", "Pregunta #2"))
        self.radioButton_15.setText(_translate("MainWindow", "Pregunta #6"))
        self.radioButton_17.setText(_translate("MainWindow", "Pregunta #7"))
        self.radioButton_16.setText(_translate("MainWindow", "Pregunta #5"))
        self.groupBox_12.setTitle(_translate("MainWindow", "Visor Principal"))
        self.radioButton_14.setText(_translate("MainWindow", "Pregunta #5"))
        self.actionAbrir.setText(_translate("MainWindow", "Abrir"))
        self.actionSalir.setText(_translate("MainWindow", "Salir"))

if __name__ == "__main__":
    import sys
    app = QtWidgets.QApplication(sys.argv)
    MainWindow = QtWidgets.QMainWindow()
    ui = Ui_MainWindow()
    ui.setupUi(MainWindow)
    MainWindow.show()
    sys.exit(app.exec_())
```