UNIVERSIDAD DEL BIO-BIO
Facultad de Ciencias Empresariales

Escuela de Ingeniería de Ejecución en Computación e Informática

Evaluación de técnicas de extracción de características para detección automática de anomalías en redes de tuberías de alcantarillado a partir de imágenes de video.

MEMORIA PARA OPTAR A TÍTULO DE INGENIERO DE EJECUCIÓN EN COMPUTACIÓN E INFORMÁTICA

Alumnos: Jerson A. Martínez Salgado
Nicolás I. Alarcón Salazar

Profesor Guía: Pedro G. Campos Soto
Profesor Informante: Roberto E. Mercado Cuevas

CONCEPCIÓN, 2019
4 de Septiembre

Índice General

Índice de tablas

Índice de Imágenes

Resumen

Este Proyecto de Título tiene como objetivo evaluar algunas técnicas de extracción de características de imágenes y comparar los resultados al utilizarlas para la clasificación de imágenes de tuberías de alcantarillado con o sin anomalías, buscando identificar automáticamente anomalías en las redes de alcantarillado, como por ejemplo grietas, desacople de unión de tuberías y roturas, uniones domiciliarias penetrantes, incrustaciones y grasas.

Para lo anterior, se estudiaron propuestas descritas en la literatura del área, implementándose algunas de ellas en base a las descripciones publicadas. Posteriormente, se aplicaron estas técnicas sobre un conjunto de imágenes reales. Con las características obtenidas, se entrenaron diferentes clasificadores comúnmente utilizados, disponibles en la suite de Aprendizaje Automático Weka. Los mejores resultados se obtuvieron con los algoritmos Radom Forest, Perceptrón Multi capas, J48 y Bagging.

En el desarrollo de este proyecto se enfrentaron dificultades propias del trabajo con imágenes reales, en este caso obtenidas de videos de CCTV o ITV. Por ejemplo, presentaban importantes variaciones en la calidad de la imagen, lo que se conoce como “ruido”. Para trabajar adecuadamente con estas imágenes, se utilizó una biblioteca de procesamiento de imágenes llamada OpenCV.

Uno de los aprendizajes relevantes de este proyecto es la importancia de la extracción de características mediante la forma. En particular, la técnica de detección de bordes resultó útil para identificar anomalías como grietas o desacople de unión de tuberías y roturas.

Por último, se realiza una experimentación de los métodos de detección de bordes y las características analizadas, por medio de métricas, logrando un desempeño de 91% de accuracy en la detección de bordes de Sobel mediante Random Forest, posicionándolo como el mejor método en la experimentación.

1. Introducción

1.1 Motivación

Las grandes y extensas redes de alcantarillado que recorren nuestro país, a través de calles y ciudades, tienen un propósito, que es llevar nuestros residuos a las plantas de tratamientos de aguas servidas (Rosas Coronado, 2008). Estas tuberías son propensas a tener fallos o defectos (Flórez Casillas, 2007), en cuyo caso no logran cumplir su principal función. Muchos de estos fallos son causados, entre otros motivos, por el tiempo de vida de éstas, creando grietas, rupturas, pérdida de pendiente, deformaciones (colectores de polímeros) y UD (uniones domiciliarias) penetrantes. Otra causa de estos fallos es por componentes no residuales (tubos, palos, etc.), que son desechados a estas tuberías, las cuales crean incrustaciones, obstrucciones y acumulación de grasa. Por esta razón están sujetas a múltiples inspecciones mediante CCTV (circuitos cerrados de televisión) (Rizzo, 2010) o ITV (inspección televisiva), para analizar el estado en que se encuentran estos colectores.

Las empresas que se encargan de monitorear estas extensas redes y de realizar limpiezas sobre éstas, generan alrededor de 170 inspecciones de CCTV o ITV por semana, lo cual crea una gran cantidad de videos de los colectores analizados, generando un trabajo arduo para los analistas que revisan constantemente los videos para poder dar su reporte de fallas con respecto a cada inspección.

La no respuesta rápida ante las obstrucciones o defectos graves de las redes de alcantarillado pueden crear problemas mayores, tales como rebalses, liberación de gases de alcantarillas y malos olores, pudiendo provocar enfermedades como la hepatitis y otras consecuencias como náuseas, dolores de cabeza, plagas e incapacidad de habitar un domicilio o lugar público temporalmente (Ramos et al., 2016). Por ende, es importante su mantención y una respuesta ágil ante los defectos de nuestras redes de alcantarillados, ya sea en lugares públicos o domiciliario.

Considerando lo anterior, este Proyecto de Título tiene como fin, avanzar en la etapa de extracción de características, que pueda permitir a futuro la creación de un sistema automatizado para localizar anomalías en colectores, gracias por ejemplo al aprendizaje de una Red Neuronal Artificial (Nilsson, 1997) para así facilitar y disminuir el tiempo de respuesta, previniendo las posibles discontinuidades del servicio en domicilios o vías públicas. Como se comentó previamente, el trabajo de inspección es constante y agotador por parte de los analistas. En este sentido, este proyecto permitirá aportar al futuro desarrollo de sistemas que apoyen a los inspectores de tuberías, minimizando posibles errores humanos productos del cansancio.

1.2 Objetivos

1.2.1 Objetivo General

Comparar el rendimiento de diferentes técnicas de extracción de características en la tarea de detección automática de anomalías, mediante algoritmos de Aprendizaje Automático, en redes de alcantarillado a partir de imágenes de video.

1.2.2 Objetivos Específicos

- Analizar trabajos previos para identificar características y algoritmos de mejor rendimiento en la tarea seleccionada.
- Generar un dataset de imágenes extraídas desde videos con referencias a la detección de anomalías en redes de alcantarillado.
- Implementar rutinas para la obtención de características seleccionadas desde imágenes, seleccionadas a partir de la literatura analizada.
- Comparar el desempeño de las características obtenidas en la tarea de detección de anomalías, al ser utilizadas por diferentes algoritmos de Aprendizaje Automático.

1.3 Descripción de la metodología utilizada

- **Generar un dataset de imágenes de colectores de tuberías con defectos.**

  Este proceso es necesario para el entrenamiento de modelos, y las pruebas del proceso de extracción de características.

  Para este Proyecto de Título, se propuso una meta de alrededor de 200 imágenes para formar un dataset apropiado. Se obtuvieron 142 imágenes de tuberías en buen estado y 63 de tuberías con defectos.

  En estas imágenes se pueden encontrar diferentes tipos de defectos, tales como: roturas, desacople (de unión de tuberías) y grietas. En el caso de las imágenes en buen estado se consideraron tuberías sin problemas, como por ejemplo unión de tuberías (sin desacoplamiento), y caminos de tuberías sin problema alguno.

- **Analizar trabajos previos para identificar características y algoritmos de mejor rendimiento.**

  Para entender el problema y tener una base en este proyecto, se analizaron diferentes artículos publicados en revistas y conferencias científicas relacionadas con el tema, para identificar diferentes métodos de extracción de características y características utilizadas en el procesamiento de este tipo de imágenes.

- **Implementar rutinas para la obtención de características seleccionadas a partir de la literatura analizada.**

  Para esto, se utilizó el lenguaje de programación Java. Se construyeron algoritmos de extracción de características de acuerdo a las técnicas identificadas en la bibliografía encontrada, utilizando en algunos casos algoritmos implementados en la biblioteca OpenCV[1].

[1] Open Source Computer Vision Library, disponible en: https://opencv.org/

- **Comparar el desempeño de las características obtenidas sobre diferentes algoritmos de Aprendizaje Automático.**

  Se realizaron experimentos para medir los resultados obtenidos al utilizar las técnicas de extracción de características seleccionadas en el entrenamiento de modelos de Aprendizaje Automático, utilizando diferentes métricas. Lo anterior, con el fin de evaluar el desempeño en la tarea de identificar defectos de manera automatizada.

  Métricas utilizadas (Skymind, 2019):

  - Accuracy
  - Precision
  - Recall
  - F1-Score

1.4 Herramientas utilizadas

1.4.1 Software Utilizado

- **IntelIJ 19.1.3** (JetBrains s.r.o, 2000)**:** Entorno de desarrollo integrado (IDE), para el lenguaje de programación Java. En este IDE se desarrollaron los prototipos de este Proyecto de Título.

1.4.2 Bibliotecas utilizadas

- **OpenCv v.4.1.0** (OpenCvTeam, 1999)**:** Biblioteca de Computer Vision, en el cual se realizó la gran parte del procesamiento de imágenes (Imagen binaria, operaciones morfológicas), antes del proceso de extracción de características.
- **Weka v.3.8.3** (U. Waikato, 2011)**:** Es una biblioteca de algoritmos de Aprendizaje Automático, que ayuda a crear modelos con diferentes métodos de aprendizaje. Recibe un archivo en específico como entrada (.arff), con los datos de extracción de características. Está implementada en el lenguaje de programación Java.
- **JavaFx v.11.0.2** (Oracle, 2011)**:** Biblioteca desarrollada por Oracle, otorga nuevas implementaciones gráficas, tanto para aplicaciones web y escritorio, en este proyecto de título se utilizó para la interfaz gráfica del programa.

1.4.3 Herramientas externas

- **LabelImg 1.1.0** (Tzutalin, 2017)**:** Programa que ayuda a definir la ROI (*region of interest* o región de interés) en una imagen, dejando sus coordenadas en un archivo de tipo XML.
- **Wondershare Filmora9:** Programa utilizado para estandarizar las imágenes a un mismo tamaño, quitando letras y números que entregan los videos de CCTV o ITV, para así, trabajar solo con imágenes.

1.4.4 Lenguajes utilizados

- **Java 8** (Oracle, 2014)**:** Lenguaje de programación utilizado para la implementación de las rutinas requeridas en este Proyecto de Título.

1.5 Estructura del Informe

En el Capítulo 1 se explica la motivación de este Proyecto de Título, los objetivos planteados, la metodología con la que se abordaron estos objetivos y las herramientas que se utilizaron. En el Capítulo 2 se explican conceptos generales y las técnicas básicas de preprocesamiento y extracción de características de imágenes utilizados. En el Capítulo 3 se presentan trabajos relacionados, que fueron utilizados como base para definir las características a utilizar y el proceso diseñado. En el Capítulo 4 se describen las características de imágenes que se consideraron en este Proyecto de Título y se detalla su funcionamiento y fórmulas de cálculo. En el Capítulo 5 se describe la implementación que se realizó para 3 algoritmos estudiados, MSED, Top-Hat y Canny, explicando los operadores utilizados y sus diagramas respectivos. En el Capítulo 6 se reportan los resultados obtenidos al utilizar diferentes algoritmos de clasificación provistos por Weka entrenados con las características estudiadas.

Por último, en el Capítulo 6 se cierra a este Proyecto de Título presentando las principales conclusiones de este trabajo.

2. Marco Conceptual

En este capítulo se presentan las principales definiciones o conceptos fundamentales relacionados con este Proyecto de Título. Entre los conceptos generales se encuentran *Machine Learning* (Aprendizaje Automático), *Computer Vision* (Visión por Computador), *Pattern Recognition* (Reconocimiento de Patrones) y algunos términos utilizados en el área de aplicación de este proyecto, los alcantarillados. En el caso de reconocimiento de tuberías, se describen las etapas fundamentales y técnicas de procesamiento que utilizaron.

2.1 Conceptos Generales

2.1.1 Machine Learning

La mayoría de las máquinas o software complejos utilizan algoritmos, que se podrían definir como metodologías definidas para resolver un problema específico. En algunos casos, el algoritmo tiene una capacidad de adaptación, lo que permite que el algoritmo se adapte a diferentes condiciones.

En contraste, en el Aprendizaje Automático (*Machine Learning*) no busca desarrollar algoritmos específicos, sino más bien, generar estructuras básicas para procesar datos y producir modelos a partir de los datos (Parker, 2017).

El campo del aprendizaje automático surgió de la Estadística tradicional y de la Inteligencia Artificial, de los esfuerzos de corporaciones como Google, Microsoft, Facebook, Amazon, entre otras, El Aprendizaje Automático se ha convertido en uno de los temas más interesantes de la Ciencia Computacional en la última década. Esto unido a la recolección de grandes volúmenes de datos, ha proporcionado una oportunidad para revitalizar los enfoques estadísticos y computacionales, para generar modelos útiles a partir de datos. (Huddleston & Brown, 2018).

Tipos de Machine Learning

El aprendizaje automático se clasifica en términos generales como aprendizaje supervisado, no supervisado y reforzado, en cada uno se pueden encontrar diferentes tareas, como muestra la Figura 1.

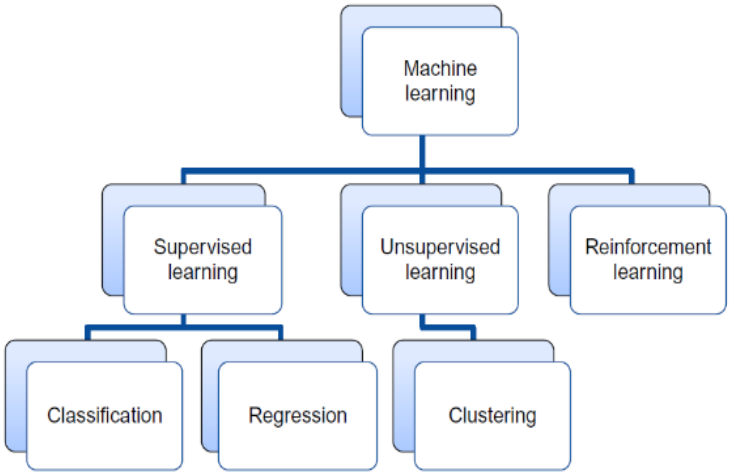


Figura 1: Tipos de *Machine Learning*. Fuente: (Gangadhar Shobha, 2018)

Aprendizaje Supervisado

El aprendizaje supervisado es un modelo de aprendizaje creado para hacer predicciones, dada una instancia de entrada. Generalmente, un algoritmo de aprendizaje supervisado toma un conjunto de datos de entrada y resultados conocidos (por ejemplo, un conjunto de imágenes y etiquetas sobre su contenido) para aprender un modelo de regresión o clasificación.

- **Clasificación:** La tarea de clasificación predice respuestas discretas, por ejemplo, si una imagen contiene o no contiene una tubería., Se recomienda si los datos se pueden clasificar, etiquetar o separar en grupos o clases específicos. Los modelos de clasificación asignan instancias (por ejemplo, imágenes) según los datos de entrada (atributos o información de la imagen) a categorías. Algunas aplicaciones populares o principales de clasificación incluyen la calificación crediticia del banco, imágenes médicas y reconocimiento de voz. También se utilizan algoritmos de clasificación para el reconocimiento de escritura a mano, para reconocer letras y números, para verificar si un correo electrónico es genuino o no es spam, o incluso para detectar si un tumor es benigno o maligno.

- **Regresión:** Las técnicas de regresión predicen respuestas continuas. Una regresión lineal intenta modelar la relación entre dos variables ajustando la ecuación lineal a los datos observados. Por ejemplo, se recopilan datos sobre cuán felices son las personas después de haber dormido, y la cantidad de horas que durmieron. En este conjunto de datos, el sueño y las personas felices son las variables. Por análisis de regresión, se puede relacionar estas variables, y realizar predicciones, respecto de cuán feliz es una persona dependiendo de cuántas horas durmió.

Aprendizaje No Supervisado

En el aprendizaje supervisado, el objetivo es aprender la asignación de cada instancia, descrita por sus atributos o valores de entrada, a una salida cuyos valores correctos son conocidos para una gran cantidad de casos. En el aprendizaje no supervisado, el objetivo es encontrar regularidades en los datos de entrada de manera de detectar patrones que se produzcan con mayor frecuencia que otros. Un ejemplo de esto es la agrupación (clustering).

- **Clustering:** En una agrupación de elementos similares, en base a sus atributos descriptivos. Por ejemplo, en la agrupación de documentos, el objetivo es agrupar documentos. No existe una clasificación conocida, y sólo se agrupan en base a qué tan similares pueden ser sus contenidos. Por lo general, cualquier documento se representa como una "bolsa de palabras", es decir, un léxico predefinido de N palabras, por ejemplo, documentos que coincidan en un mayor número de palabras serán asignados al mismo grupo.

Aprendizaje Reforzado

El aprendizaje por refuerzo implica interactuar con el entorno circundante. El aprendizaje por refuerzo aborda el problema de cómo un agente autónomo que percibe y actúa en su entorno, puede aprender a elegir acciones óptimas para lograr sus objetivos. El comportamiento de un agente se recompensa en función de las acciones que tomó, considera las consecuencias de sus

acciones y buscar dar pasos óptimos. Un ejemplo de esto se puede observar en sistemas que juega al ajedrez con humanos, y aprenden qué movimientos realizar al acumular el conocimiento sobre los resultados de múltiples jugadas.

2.1.2 Pattern Recognition

En la vida cotidiana, constantemente se utilizan los sentidos, como la vista, el tacto, el olfato etc. y través de estos sentidos es posible, por ejemplo, categorizar o reconocer ciertos objetos, animales o personas. El concepto de reconocimiento de patrones es muy similar a lo que ocurre al hacer uso de estos sentidos, pero para los seres humanos realizar estas acciones es algo trivial, sin embargo, implementarlo artificialmente, se convierte en una tarea muy compleja (Casasent, 1981).

Según lo dicho anterior podemos entender un poco de lo que se trata el reconocimiento de patrones, pero ¿qué es un patrón?, según el autor del libro "Pattern recognition: Human and mechanical" es posible decir que es una entidad a la que se le puede dar un nombre y que está representada por un conjunto de propiedades medidas y las relaciones entre ellas (vector de características) (Watanabe & Satosi, 1985).

Normalmente se encontraban 2 grandes enfoques de reconocimiento de patrones, los cuales son los estadísticos y sintácticos, pero la idea de automatizar esta tarea generó otro enfoque, con la incorporación de *Machine Learning*, con implementación de "algoritmos de caja negra" (Taylor, 2010).

El reconocimiento de patrones se ha desarrollado en varias áreas, algunos ejemplos son (Casasent, 1981):

- Bioinformática.
- La clasificación de documentos.
- El análisis de imágenes.
- La minería de datos.
- La automatización industrial.
- El reconocimiento biométrico.

- La detección remota.
- El análisis de textos escritos a mano.
- El diagnóstico médico.
- El reconocimiento de voz.

La similitud entre todas estas aplicaciones descritas anteriormente es en su enfoque de solución. En éstas se deben extraer características, para después analizarlas y clasificarlas. Esto quiere decir que, por ejemplo, para clasificar de diferente forma a 2 objetos, es necesario encontrar diferencias, descripciones diferentes, las cuales pertenecen a modelos (patrones) diferentes, que son expresados de forma matemática.

El objetivo general en el reconocimiento y clasificación de patrones es determinar a qué categoría pertenecen estos modelos (Navarrete, 2019).

El proceso general del reconocimiento de patrones se divide en 3 etapas que se deben desarrollar, esto se puede visualizar en la Figura 2.

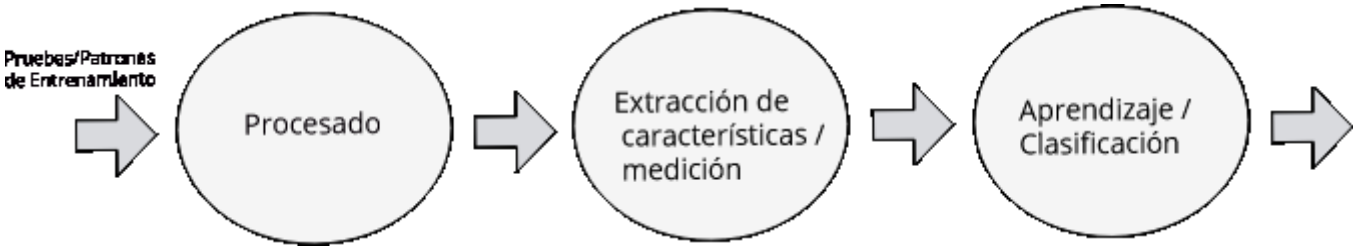


Figura 2: Pasos para el reconocimiento de patrones de forma general. Fuente:(Taylor, 2010).

Para entender mejor este concepto, se presenta a continuación un ejemplo del libro “Pattern Recognition”:

Una empresa de embalaje de pescado, quiere automatizar el proceso de clasificación de los peces entrantes en una cinta transportadora de acuerdo con las especies. Como proyecto piloto, sugieren separar la lubina del salmón. (Duda, Hart, & Stork, 2000)

Para realizar ese proceso, el autor comenta que se tomaron fotografías para una detección óptica, dejando en evidencia las diferencias físicas que poseen estas especies (Salmon y lubina), como son: longitud, ancho, forma y posición de la boca. Esto representa claramente las características que se pueden utilizar o extraer para una detección automática. Sin embargo, también observa que, al utilizar cámaras digitales, las fotografías presentan ruido o variaciones en las imágenes, como: variación en la iluminación, en la posición del pescado en el transportador y algunos errores debido al uso de la cámara. Esto indica que las imágenes deben ser procesadas adecuadamente, requiriéndose por ejemplo segmentación y técnicas de preprocesamiento (descritas en Sección 2.2.1), para contar una imagen limpia para la fase de extracción de características, que finalmente puedan ser aprovechadas por un clasificador y así decidir si corresponde a un salmón o una lubina.

Todo esto permite determinar los pasos necesarios para realizar un procedimiento de clasificación satisfactorio. La Figura 3 resume los pasos del ejemplo comentado del libro "Pattern Recognition".

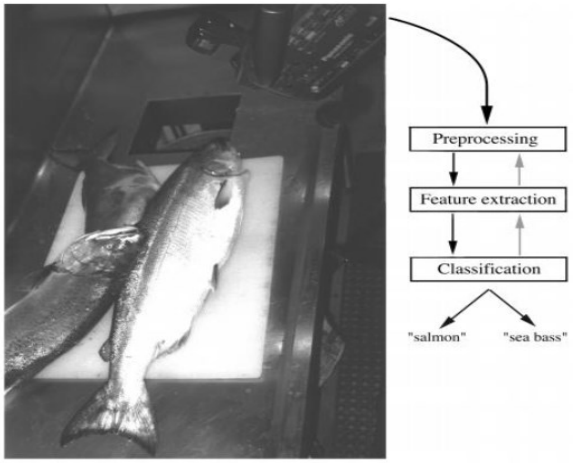


Figura 3: Proceso de clasificación de pescados. Fuente: (Duda et al., 2000)

2.1.3 Computer Vision

Con origen en el campo de la Inteligencia Artificial, la Visión Computacional se ocupa del estudio de la capacidad de las máquinas para realizar tareas visuales.

En este caso, los datos de entrada corresponden a imágenes o secuencias de video, capturadas a través de cámaras.

Estas tareas van desde la eliminación y filtrado o la segmentación de imagen / video (es decir, la operación de agrupar píxeles de imagen en entidades semánticamente uniformes) y, finalmente, la detección o reconocimiento de alto nivel, como por ejemplo, reconocer objetos o caras en imágenes (Liu, 2014).

En pocas palabras *Computer Vision* "es *la transformación de datos de una cámara de fotos o video en una decisión o una nueva representación"* (Bradski & Kaehler, 2008).

El viejo proverbio chino, "Una imagen vale más que mil palabras", resume mejor los desafíos y oportunidades de este campo. Si bien es beneficioso tener una gran cantidad de información disponible en una imagen, no es una tarea fácil analizar la información, imponerle una estructura y razonar sobre ella de manera eficiente, aunque estas tareas pueden ser simples e intuitivas para los seres humanos.

Por lo tanto, el objetivo de *Computer Vision* es emular la visión humana utilizando imágenes digitales a través de tres componentes principales de procesamiento, ejecutados uno tras otro (Haio, 2012):

- Adquisición de imágenes.
- Procesamiento de imagen.
- Análisis y comprensión de la imagen.

Lo anterior se puede ver resumido en la Figura 4.

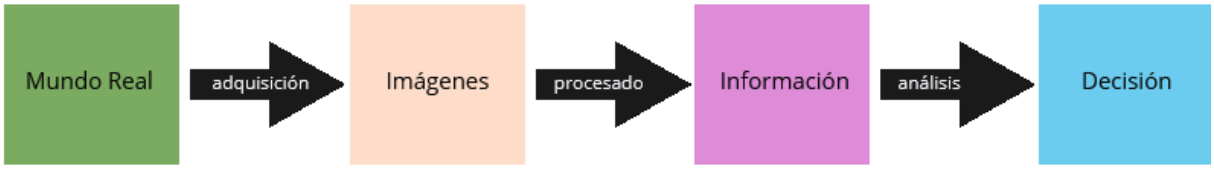


Figura 4: Pasos de *Computer Vision*. Fuente: (Haio, 2012)

Adquisición de imagen

La adquisición de imágenes es el proceso de traducir el mundo analógico que nos rodea en datos binarios compuestos de ceros y unos, interpretados como imágenes digitales (Haio, 2012). En este caso las imágenes son extraídas de videos de CCTV o ITV de la empresa Inggepro.

Procesamiento de imágenes

El segundo componente de *Computer Vision* es el procesamiento de imágenes. Los algoritmos se aplican a los datos binarios adquiridos en el primer paso para inferir información de la imagen. Este tipo de información se caracteriza por bordes de imagen, entidades de puntos o segmentos, por ejemplo. Son todos los elementos geométricos básicos que construyen objetos en imágenes (Haio, 2012).

Este segundo paso generalmente involucra algoritmos y técnicas avanzadas de matemática aplicada.

Los algoritmos de procesamiento de imágenes:

- Detección de bordes.
- Segmentación.
- Clasificación.
- Detección de características y coincidencia.

Análisis de imagen y comprensión

El último paso del proceso de *Computer Vision* es el análisis de los datos, lo que permitirá la toma de decisiones, utilizando tanto los datos de imagen como la información calculada del procesamiento de imágenes en los pasos anteriores (Haio, 2012).

Ejemplos de análisis de imágenes son:

- Mapeo de escenas 3D.
- Reconocimiento de objetos.
- Seguimiento de objetos.

2.2 Reconocimiento de tuberías

El proceso establecido en este Proyecto de Título, se basa en tres pasos, los cuales son, pre-procesamiento de imágenes, extracción de características y creación de modelo para el reconocimiento de anomalías en las redes de alcantarillado, de manera similar a lo planteado por Navarrete (Navarrete, 2019) en la detección y clasificación de hojas de árboles nativos de la región del Bio Bío, el que se puede ver reflejado en el diagrama de la Figura 5:

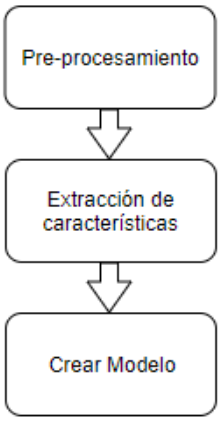


Figura 5: Principales etapas a considerar para identificación de anomalías en tuberías de alcantarillado. Fuente: (Navarrete, 2019)

A continuación, se describen brevemente estas etapas. El detalle de cada una de las técnicas empleadas en cada etapa se presenta más adelante en este documento:

- **Pre-procesamiento:** Esta es la primera etapa de nuestra metodología, la cual consiste en limpiar las imágenes obtenidas desde los videos de inspección televisiva, ya que estas presentan mucho ruido. En primera instancia se recortan las imágenes y, además, se consideran algunos procesos de segmentación binaria y morfológica para su limpieza y una correcta extracción de características. Entre estos se encuentran Thresholding, específicamente el Método de Otsu, y en el caso de la segmentación morfológica, se encuentran erosión, dilatación, apertura morfológica, etc.

- **Extracción de características:** Esta etapa se extraen características o atributos que describen la información contenida en la imagen, como el área, eje mayor, menor, excentricidad, entre otras estas características se pueden centrar en textura o la forma como los contornos, perímetro etc. Esta etapa es posterior al pre-procesamiento, pues gracias a las técnicas aplicadas en dicha etapa previa, se cuenta con imágenes que permiten una mejor extracción de las características mencionadas.

- **Crear modelo:** Esta última etapa, consiste en la creación de un modelo que contenga los patrones que permitan la clasificación de imágenes de tuberías con y sin defectos. Para esto, se utilizan algoritmos de Machine Learning provistos por el framework WEKA, los algoritmos escogidos serán los que presentan mejores resultados en este api.

2.2.1 Técnicas de Pre-procesamiento.

Imagen

Una imagen digital es información de una imagen en forma digital. Una imagen digital se puede filtrar para eliminar el ruido y obtener una mejora de calidad. También se puede transformar para extraer características para el reconocimiento de patrones. La imagen puede comprimirse para su almacenamiento y recuperación, así como transmitirse a través de una red informática o un sistema de comunicación (Poldrack et al., 2011).

Una imagen digital consta de píxeles, la posición de cada píxel se especifica en términos de un índice para el número de columnas y otro para el número de filas. La Figura 6 muestra que el píxel p (2, 8) tiene un valor de 86 en una imagen de escala de grises y está ubicado en la segunda fila, octava columna.

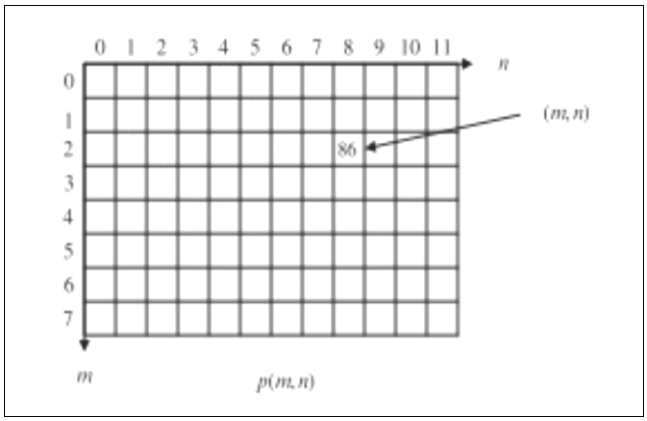


Figura 6: Matriz representativa de una imagen en escala de grises. Fuente: (Poldrack et al., 2011)

En la ecuación (1) se expresa el pixel de la Figura 6:

$$\boldsymbol{p(2,8) = 86} \qquad \textbf{(1)}$$

En este proyecto, las imágenes son fundamentales, pues como se señaló anteriormente, de estas se puede obtener información valiosa, por ejemplo, características que permitan el entrenamiento de modelos de clasificación. Por ende, se recolectaron alrededor de 200 imágenes.

Imagen de Tono gris

Las imágenes en tono gris, solo tienen 1 canal de información, donde cada pixel tiene un valor y este representa su intensidad. El valor 0 representa el color negro y el valor 255 corresponde al color blanco, y los valores intermedios son la intensidad de escala de grises. La Figura 7 corresponde a un ejemplo de una imagen en escala de grises.

Figura 7: Tubería con incrustación en escala de grises. Fuente: (Elaboración propia)

Por otra parte, a partir de una imagen de escala de grises es posible graficar cuantas veces se encuentra cada valor de intensidad mediante un Histograma. La Figura 8 muestra el histograma de intensidades para la imagen de la Figura 7.

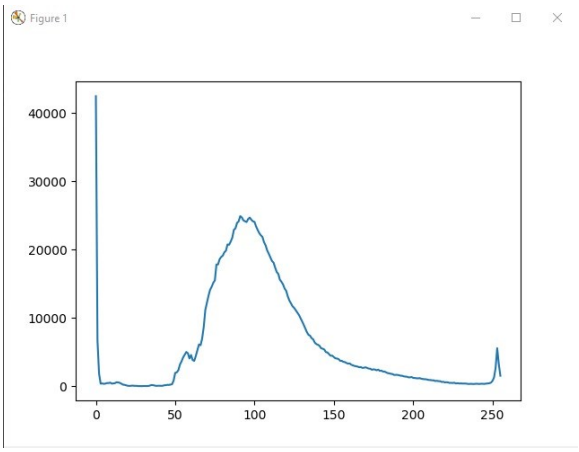


Figura 8: Histograma de una imagen en escala de grises Fuente: (Elaboración propia).

Imagen RGB de 24 bits

La sigla RGB, significa Red (Rojo), Green (verde), Blue (Azul). Estas imágenes poseen 3 canales de información (uno para cada color), a diferencia de las imágenes en tono gris, que solo poseen un canal. La figura 9 muestra que cada pixel tiene tres canales o intensidades.

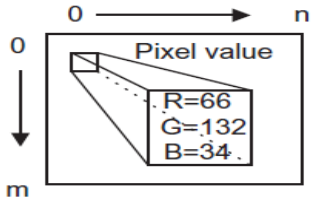


Figura 9: RGB 3 Canales Fuente: (Poldrack et al., 2011)

Esto quiere decir que por cada posición se encuentran 3 valores. Esta información se puede representar en 3 imágenes, cada una con valores de rango de 0 a 255 y unidas crean la imagen RGB.

Estas imágenes son de 24 bits, pues cada canal tiene 8 bits. Al contabilizar todas las opciones de codificación según el número de bits disponible, es posible determinar el número de colores diferentes que puede representar una imagen digital. La ecuación 2 describe la cantidad total de colores:

$$2^{24} = 16.777216\ x\ \ 10^{6} \quad (2)$$

Es decir, es posible representar más de 16 millones de diferentes tipos de colores en este tipo de imagen, a si también se pueden separar la intensidad de cada canal (Red, Green, Blue), En la Figura 10 se muestra un ejemplo:

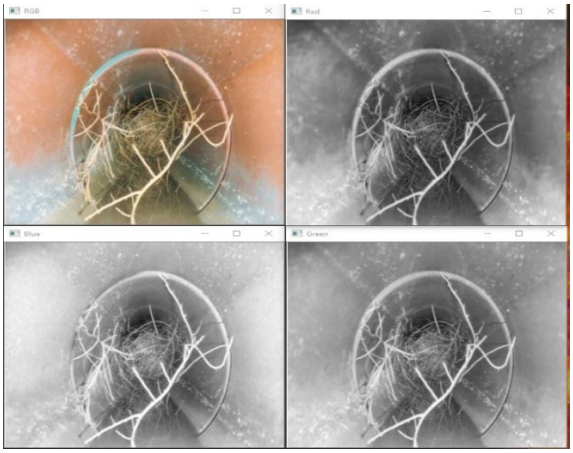


Figura 10: Representación RGB de 24 bits en sus respectivos canales de intensidad, tubería con raíces. Fuente: (Elaboración Propia)

Thresholding

En el proceso de *thresholding* se define un valor umbral y se seleccionan los píxeles sobre dicho umbral, dejando los pixeles restantes como "fondo" de la imagen, descartándolos para procesos posteriores. Una imagen de este tipo se muestra como una imagen binaria (de dos niveles) utilizando blanco y negro u otros colores para distinguir las regiones. No hay una convención estándar sobre cuáles son los rasgos de interés, si los blancos o los negros, así que la elección varía en cada caso (Aguirre Dobernack, 2013).

La selección del valor umbral, denotado como "t", se basa en algún criterio. Si el valor umbral se determina únicamente a partir del nivel de gris de cada píxel, entonces el método de umbral es dependiente del punto. Si se determina a partir de alguna propiedad local (por ejemplo, la distribución del nivel de gris local) en la vecindad de cada píxel, entonces el método de umbral

es dependiente de la región. Una técnica de umbral global, es una que determina un valor umbral única para toda la imagen, mientras que una técnica de *thresholding* local es aquella que divide una imagen dada en subimágenes y determina un umbral para cada una de estas subimágenes (Khooa et al., 1997).

Para entender mejor este concepto la ecuación (3) nos muestra los valores que se obtienen en una imagen binaria al realizar el proceso de *thresholding*, donde existen 2 valores según la condición del umbral.

$$p(x,y) = \begin{cases} 0 & si\ el\ p(x,y) < umbal \\ 1 & si\ el\ p(x,y) >= umbral \end{cases} \quad (3)$$

El problema al definir un umbral único para toda la imagen, es que se puede perder parte de la imagen, lo que se ve reflejado en la Figura 11, en el cual la grieta pierde su forma y no es posible hacer una detección de bordes exitosa.

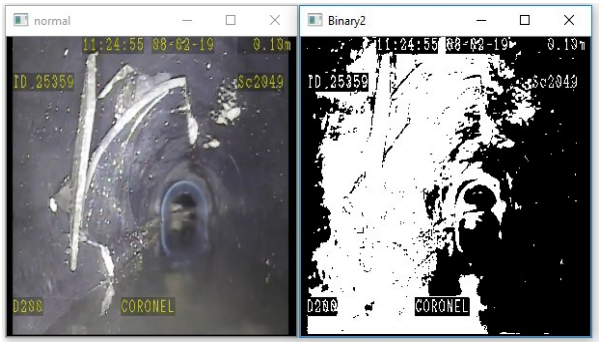


Figura 11: Tubería con rotura, umbral de *thresholding* manual no favorable. Fuente: (Elaboración propia).

El siguiente ejemplo, mostrado en la Figura 12, se utiliza un umbral correcto, lo que permite extraer datos de la imagen binara de forma satisfactoria. Esto puede ser acompañado de algunos

procesos morfológicos para limpiar ciertos ruidos y poder dejar la región de interés (ROI) más accesible para la extracción de características.

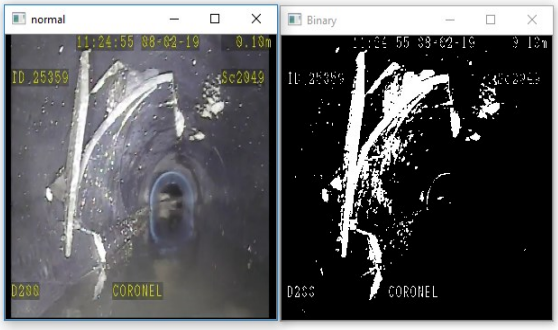


Figura 12: Tubería con rotura, umbral de *thresholding* favorable. Fuente: (Elaboración propia)

Pero, aun así, al definir de manera manual el umbral para una imagen, no implica que sea satisfactorio a todas las imágenes, porque cada imagen necesita un umbral diferente para tener una visualización correcta. Por esta razón existen métodos automáticos, como el Método de Otsu.

Thresholding mediante Método de Otsu

El método de segmentación de imágenes de Otsu, desarrollado por Nobuyuki Otsu, consiste en la selección de un umbral óptimo que maximiza la varianza entre clases mediante una búsqueda exhaustiva (Smith et al., 1979). Inicialmente este algoritmo particiona una imagen de N pixeles según los niveles de gris. Existen L posibilidades de niveles diferentes dentro de una imagen, donde la probabilidad de ocurrencia del nivel de gris se representa en la ecuación (4):

$$P_i = \frac{f_i}{N} \tag{4}$$

donde f_i es la frecuencia de repetición del nivel de gris i-ésimo con i=1, 2, …, L.

La umbralización de 2 niveles en el método de Otsu es un caso particular, ya que consiste en dividir los pixeles en dos clases C_1 y C_2, con niveles de gris $[1, 2, \dots, \mathrm{t}]$ y $[\mathrm{t}+1, \mathrm{t}+2, \dots, \mathrm{L}]$ respectivamente donde las distribuciones de probabilidad de ambas clases se representan en la ecuación (5):

$$C_1: \frac{p_1}{\omega_{1(t)}}, \dots., \frac{p_t}{\omega_{1(t)}} \quad C_2: \frac{p_{t+1}}{\omega_{2(t)}}, \dots., \frac{p_{t+2}}{\omega_{2(t)}}, \dots., \frac{p_L}{\omega_{2(t)}} \tag{5}$$

Donde. p_i es la probabilidad de que un nivel de gris ocurra, t es el umbral que maximiza la varianza., $\omega_1(t) = \sum_{i=1}^{t} p_i$ y $\omega_2(t) = \sum_{i=t+1}^{L} p_1$

En la imagen de la Figura 13 se muestra el resultado de haber aplicado el algoritmo de Otsu de 2 niveles donde $t = 85$

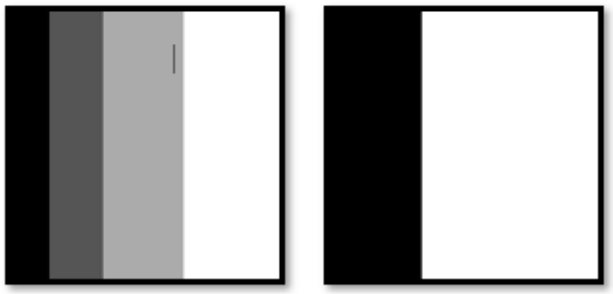


Figura. 13: Resultado de aplicar el método de Otsu, con un umbral de 85. Fuente: (Gonzalez, 2002)

2.2.1.1 Operaciones Morfológicas

Las operaciones morfológicas son operaciones simples basadas en la forma de la imagen, normalmente se realiza en imágenes binarias.

En este proyecto se utilizó la biblioteca OpenCV, la cual entrega muy buenas herramientas de segmentación morfológica y técnicas de procesamiento de imágenes.

Estas herramientas necesitan siempre como entrada 2 parámetros, una es la imagen original, la segunda se llama elemento de estructuración o Kernel, que decide la naturaleza de la operación. Dos operadores morfológicos básicos son la erosión y la dilatación, además de variantes como Apertura, Cierre, Gradiente, etc. (OpenCvTeam, 1999).

Kernel o elemento estructurante

Los procesos morfológicos están sujetos a un elemento estructurante o kernel (núcleo), el cual tiene diferentes formas y tamaños. Esta estructura está compuesta de una matriz, que contiene 1's y 0's, entre las más comunes podemos encontrar las siguientes.

- Cuadrada.
- Elipse.
- Rombo.
- Diamante.

Erosión

La idea básica de la erosión, es similar a la erosión del suelo, erosiona los límites del objeto en primer plano.

El kernel se desliza a través de la imagen, un píxel en la imagen original (1 o 0) se considerará 1 solo si todos los píxeles debajo del kernel son 1, de lo contrario se erosionará (se pondrá a cero). En otras palabras, lo que sucede es que todos los píxeles cerca del límite se descartarán dependiendo del tamaño del kernel. Por lo tanto, el grosor o el tamaño del objeto en primer plano disminuye o simplemente la región blanca disminuye en la imagen. Es útil para eliminar pequeños ruidos blancos (OpenCvTeam, 1999). En la Figura 14 se muestra un ejemplo de la operación morfológica de erosión.

Figura 14: Representación visual del operador de erosión, imagen de la izquierda es la original y la imagen de la derecha es el resultante. Fuente: (OpenCvTeam, 1999)

Dilatación

Es justo lo contrario de la erosión. Aquí, un píxel es considerado "1" si al menos un píxel debajo del núcleo es "1". Por lo tanto, aumenta la región blanca en la imagen o el tamaño del objeto en primer plano aumenta. Normalmente, en casos como la eliminación de ruido, la erosión es seguida por la dilatación. Esto, ya que la erosión elimina los ruidos blancos, pero también encoge los objetos. Por tanto, al dilatar, el área de los objetos aumenta. Dado que el ruido se ha eliminado, no volverá a aparecer. También es útil para unir partes rotas de un objeto (OpenCvTeam, 1999). La Figura 15 muestra un ejemplo de la operación morfológica de dilatación.

Figura 15: Representación visual del operador de dilatación, imagen de la izquierda es la original y la imagen de la derecha es el resultante. Fuente: (OpenCvTeam, 1999)

Apertura Morfológica

Este proceso consiste en realizar una erosión seguida de una dilatación (OpenCvTeam, 1999). Este proceso es necesario cuando se requiere eliminar ciertos ruidos, que quedan después del proceso de transformar una imagen a color en binaria. Por ende, para no perder el formato de los objetos, se aplica una erosión, que consiste en quitar capas de pixeles, pero el objeto o área de interés se ve afectado. Por lo que a continuación se aplica una dilatación, para dejar el área de interés en su tamaño normal, logrando eliminar el ruido fuera del objeto de interés o ROI.
En la Figura 16 se muestra un ejemplo del operador de Apertura morfológica.

Figura 16: Representación visual del operador de Apertura Morfológica, imagen de la izquierda es la original y la imagen de la derecha es el resultante. Fuente: (OpenCvTeam, 1999)

Cerrado Morfológico

Contrario al proceso de apertura, este proceso realiza una dilatación seguida de una erosión. Es muy eficaz para eliminar ciertos ruidos en el objeto de interés (OpenCvTeam, 1999). Este proceso utiliza la dilatación para eliminar ciertas imperfecciones del objeto de interés, pero agrega capas de pixeles al objeto, por lo que se torna más grande. Para evitar esto, se aplica una erosión, para dejar el objeto o ROI en su tamaño original sin las imperfecciones originales (ruido interior). En la Figura 17 se muestra un ejemplo de la operación de cerrado morfológico.

Figura 17: Representación visual del operador de Cerrado Morfológica, imagen de la izquierda es la original y la imagen de la derecha es el resultante. Fuente: (OpenCvTeam, 1999)

Filtro Gaussiano

La función del filtro Gaussiano es crear una imagen suavizada (smoothing), la cual crea un desenfoque en la imagen que se le entrega, este filtro quita ruido en la imagen, para realizar el suavizado este filtro busca el pixel con más peso mediante la ecuación de distribución gaussiana. En la ecuación (6), x corresponde a los valores de los píxeles de la imagen en escala de grises, σ corresponde a la distribución normal.

$$G(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2}{2\sigma^2}} \tag{6}$$

al tener el pixel con mayor peso sus vecinos disminuyen sus valores a distancia de este (OpenCvTeam, 1999), para entender mejor este concepto se visualizara en el siguiente ejemplo (Ver Figura 18), donde el pixel de mayor peso se encuentra en el medio.

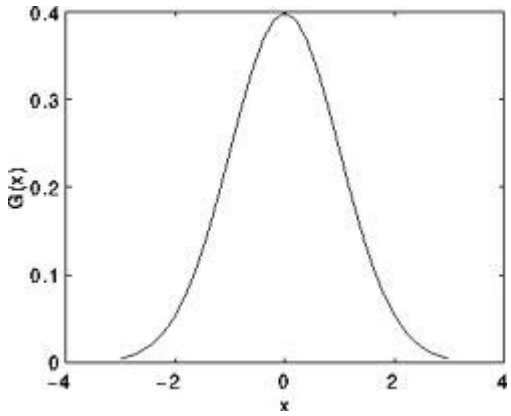


Figura 18: Grafica del funcionamiento del filtro Gaussiano Fuente: (OpenCvTeam, 1999)

En la Figura 19 se visualizará un ejemplo de una imagen aplicando el filtro gaussiano.

Figura 19: Imagen normal a la izquierda e imagen resultante con el filtro Gaussiano a la derecha. Fuente: (OpenCvTeam, 1999)

2.2.2 Extracción de características.

En esta sección se describen los métodos de extracción de características considerados en este proyecto.

Forma

La forma es una característica fundamental, ya que por medio de ésta se obtienen diversas características de las anomalías. El contorno es base para obtener otras características del objeto de interés, como por ejemplo el área, eje mayor, eje menor, excentricidad, color medio, intensidad, ratio de ejes y el perímetro. Estas características se especifican más adelante.

Sobel

Sobel trabaja con una aproximación a un derivado de una imagen, creando 2 imágenes. La primera con la dirección del eje x, y la segunda con la dirección del eje y. Para lograr esto se utiliza una matriz de kernel de 3 por 3, una para cada dirección x e y. El gradiente para la dirección x tiene números negativos en el lado izquierdo y números positivos en el lado derecho. De manera similar, el gradiente para la dirección y tiene números negativos en la parte inferior y números positivos en la parte superior, para más detalles ver Figura 20.

-1	0	+1
-2	0	+2
-1	0	+1

Gx

+1	+2	+1
0	0	0
-1	-2	-1

Gy

Figura 20: Matriz de 3x3 del eje x e y del operador Sobel Fuente: (R. Fisher, S. Perkins, 2003)

Se realiza una convolución por cada pixel con las matrices correspondientes al eje x e y, multiplicando el valor del pixel por cada lugar de la matriz y luego sumando cada uno y remplazando ese resultado en la posición del pixel, si el valor es diferente de 0 se logra obtener un borde, a continuación, se muestra un ejemplo en la Figura 21.

100	100	200	200
100	100	200	200
100	100	200	200
100	100	200	200

-1	0	1
-2	0	2
-1	0	1

-100
-200
-100
200
400
+200
=400

Figura 21: Matriz de 3x3 del eje x e y del operador Sobel Fuente: (R. Fisher, S. Perkins, 2003)

Detector de bordes de Canny

El algoritmo detector de borde de Canny es uno de los más famosos y ocupados actualmente en el área de *Computer Vision*. Este método busca los cambios bruscos de intensidad y su principal función es encontrar los límites o bordes de un objeto (Ding & Goshtasby, 2000).

Este método consta de 4 pasos (OpenCvTeam, 1999):

- **Eliminar Ruido:** Este método es muy susceptible al ruido de la imagen, por lo tanto, el preprocesamiento de la imagen es fundamental para su implementación, un ejemplo de limpieza sería el filtro gaussiano.
- **Obtención del gradiente:** Realiza un filtro de Sobel en la imagen, con dirección horizontal y vertical, para la obtención de la primera derivada, teniendo estos 2 resultados (imágenes), se puede lograr obtener el gradiente mediante la ecuación (7) y su dirección mediante la ecuación (8):

$$G = \sqrt{G_Z^2 + G_Y^2} \tag{7}$$

$$\theta = tan^{-1}(\frac{G_y}{G_z}) \tag{8}$$

- **Supresión no máxima:** Después de obtener la magnitud y dirección del gradiente, se realiza un escaneo completo de la imagen para eliminar los píxeles no deseados que pueden no constituir el borde. Para esto, en cada píxel se verifica si es un máximo local en su vecindario en la dirección del gradiente, Para demostrar esto se puede visualizar en la Figura 22.

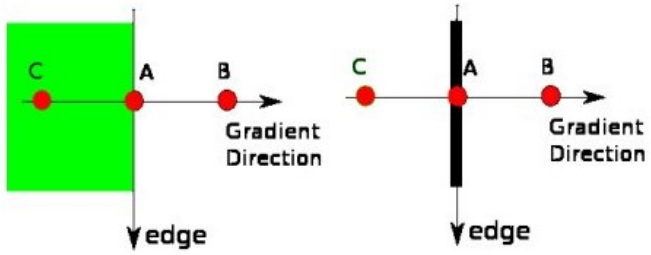


Figura 22: Representación de cómo eliminar pixeles no deseados. Fuente: (OpenCvTeam, 1999)

En la Figura 22 el punto A está en el borde (en dirección vertical). La dirección del gradiente es normal al borde. Los puntos B y C están en direcciones de gradiente. Por

lo tanto, el punto A se comprueba con el punto B y C para ver si forma un máximo local. Si es así, se considera para la siguiente etapa, de lo contrario, se suprime (se pone cero). En resumen, el resultado que obtienes es una imagen binaria con "bordes finos".

- **Histéresis de umbral:** Esta etapa decide cuáles bordes detectados en el paso anterior son realmente bordes y cuáles no. Para esto, se necesitan dos valores de umbral, minVal y maxVal. Cualquier borde con un gradiente de intensidad mayor que maxVal seguramente será un borde y los que están por debajo del minVal seguramente no serán bordes, por lo que se descartan. Los que se encuentran entre estos dos umbrales se clasifican como bordes o no según su conectividad. Si están conectados a píxeles de "borde seguro", se consideran parte de los bordes. De lo contrario, también se descartan. En la Figura 23 se muestra un ejemplo de los dicho anterior.

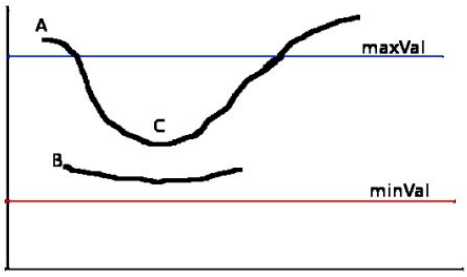


Figura 23: Representación de los bordes considerados Fuente: (OpenCvTeam, 1999)

En la Figura 23 el borde A está por encima de maxVal, por lo que se considera como "borde seguro", aunque el borde C está por debajo de maxVal, está conectado al borde A, por lo que también se considera un borde válido y se obtiene la curva completa. El borde B, aunque está por encima del valor mínimo y está en la misma región que el borde C, no está conectado a ningún "borde seguro", por lo que se descarta. Por lo tanto, es muy importante seleccionar valores adecuados de manera manual para minVal y maxVal. para obtener el resultado correcto.

2.2.3 Algoritmos para entrenamiento de clasificadores.

A continuación, se describen los algoritmos de *Machine Learning* para entrenamiento de clasificadores considerados en este Proyecto de Título, los que incluyen Random Forest, Multilayer Perceptron y J48.

Algoritmo J48

El algoritmo J48 (llamado así en Weka) es una implementación del algoritmo C4.5, desarrollado por Ross Quinlan, para la construcción de árboles de decisión. Es un algoritmo de aprendizaje supervisado, requiere un conjunto de ejemplos de entrenamiento y cada ejemplo puede verse como un par: objeto de entrada y un valor de salida deseado (clase). El algoritmo analiza el conjunto de entrenamiento y crea un clasificador que debe poder clasificar correctamente tanto el entrenamiento como ejemplos de prueba. Un ejemplo de prueba es un objeto de entrada y el algoritmo debe predecir un valor de salida (el ejemplo debe asignarse a una clase) (U. OBerta de catalunya, 2010). Un ejemplo claro de esto se puede ver en la Figura 24 donde se puede observar las posibles decisiones de panorama que se pueden tomar en un día cualquiera (en este caso jugar o no jugar) dependiente de las condiciones climáticas existentes.

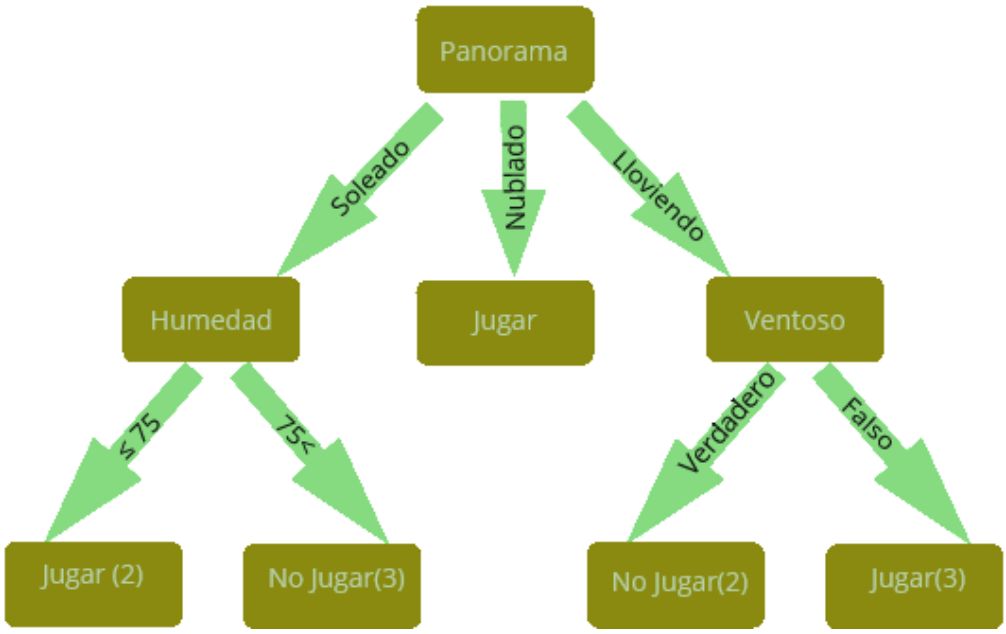


Figura 24: Árbol de decisión. Fuente: (Elaboración propia)

Random Forest

Para entender mejor este modelo de aprendizaje, es necesario entender qué es un árbol de decisión o "Decision tree". Este es un tipo de árbol, que identifica la variable más significativa y el valor que proporciona los mejores conjuntos homogéneos de población (Gonzales, 2010). Para más información de cómo funciona un árbol de decisión ver el algoritmo de entrenamiento anterior J48.

Random forest es una combinación de árboles, en que los atributos seleccionados para cada árbol dependen de los valores de un vector aleatorio probado independientemente y con la misma distribución para todos los árboles (Cutler, Cutler, & Stevens, 2012).

Este modelo de entrenamiento es un método versátil de aprendizaje automático capaz de realizar tanto tareas de regresión como de clasificación. También permite realizar reducción dimensional, trata valores perdidos, valores atípicos, entre otros. Es un tipo de método de aprendizaje por conjuntos, donde un grupo de modelos débiles se combinan para formar un modelo poderoso. En otras palabras, Random Forest es un conjunto de múltiples árboles de decisión, en el cual predomina el voto mayoritario. Una gran desventaja que posee este algoritmo de clasificación, es que se utiliza mayoritariamente como caja negra, ya que se tiene poco control sobre lo que hace el modelo.

Algunas ventajas de Random Forest son (Gonzales, 2010):

- Resuelve problemas de regresión y clasificación.
- Maneja grandes cantidades de datos con alta dimensionalidad.
- Es muy efectivo para estimar datos faltantes y mantiene la precisión cuando falta una gran proporción de los datos.

Multilayer Perceptron

Para entender mejor este concepto primero se describe el perceptrón simple y sus limitaciones. Según el libro "Neuronal Networks and Learning Machines" de Simon Haykin describe al perceptrón como la forma más simple de una red neuronal para la clasificación de patrones, el

cual son linealmente separable, en otras palabras patrones que se encuentran en lados opuestos en un hiperplano (Haykin, 2014).

El perceptrón simple puede recibir elementos externos o también puede recibir como entrada las salidas de otros perceptrones (Alpaydin, 1965). Para poder entender el funcionamiento se puede visualizar en la Figura 25, en el cual recibe como entrada (input) los elementos X_1, X_2, …, X_m, mientras que los elementos W_{K1}, W_{K2}, …, W_{Km} representan los pesos asignadas a los elementos que se recibieron en la entrada, b_k es el peso que viene desde una entrada adicional, la unidad extra de bias o sesgo, u_k es la salida del combinador lineal debido a las señales de entrada, mientras que φ_k pertenece al valor de salida del perceptrón los cuales se expresan en valores entre 0 y 1 (sigmoide) o solo valores de 0 o 1 (Heavyside).

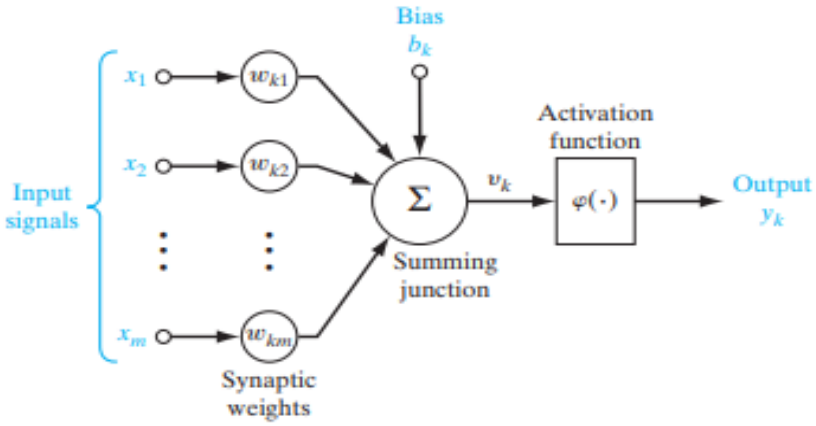


Figura 25: Arquitectura del Perceptrón Simple. Fuente: (Haykin, 2014)

En una función booleana la entradas son binarias (0 y 1) y los valores de salida corresponden a 1 si la función de salida es verdadera y 0 si la función de salida es falsa (Alpaydin, 1965), en otras palabras el perceptrón tiene la capacidad de separar 2 clases siempre y cuando sean linealmente separables, un ejemplos es la función AND se muestra a continuación en la Figura 26, donde el perceptrón puede separar los resultados con una línea, dejando los resultados con el valor 0 (r) en el lado derecho y los resultados con el valor 1 (r) en el lado izquierdo.

x_1	x_2	r
0	0	0
0	1	0
1	0	0
1	1	1

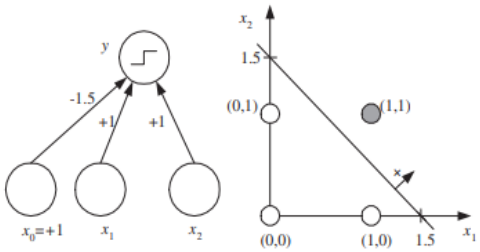


Figura 26: Separación lineal del perceptrón en la función AND. Fuente: (Haykin, 2014)

En la Figura 27 se muestra la limitación que posee el perceptrón en el caso no lineal como es la función XOR, en el cual es incapaz de clasificar en 2 grupos esta función.

x_1	x_2	r
0	0	0
0	1	1
1	0	1
1	1	0

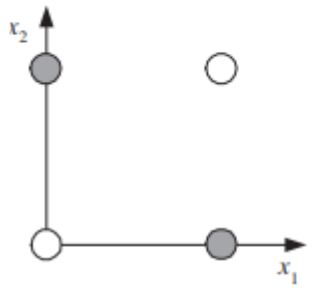


Figura 27: Limitación no lineal del perceptrón en la función XOR. Fuente: (Haykin, 2014)

Los perceptrones multicapa (MLP - *Multi Layer Perceptron*) son una clase de Red Neuronal Artificial (ANN - *Artificial Neural Network*) ampliamente utilizada para el modelado no lineal. Su mayor ventaja es que no se requiere conocimiento a priori de la forma funcional específica.

En la Figura 28 podemos observar la estructura que posee el perceptrón multicapa, en el cual, a diferencia del perceptrón simple, este posee 3 capas, uno de entrada, uno de salida y uno escondido.

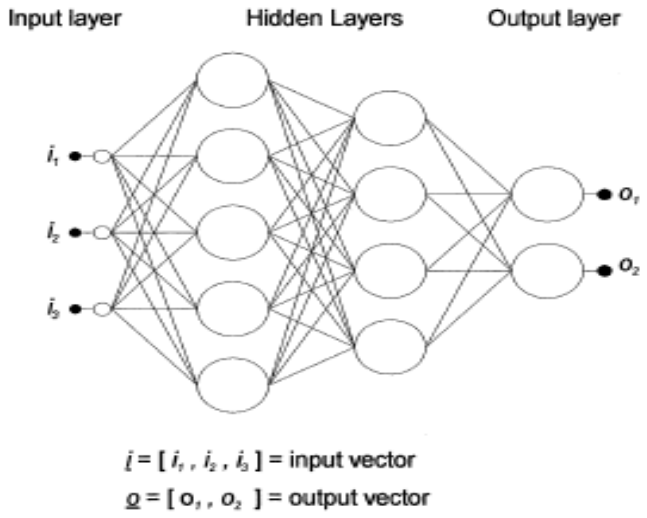


Figura 28: Estructura del Multi Layer Perceptrón. Fuente: (Gardner & Dorling, 1998)

La limitación que posee el perceptrón simple, no es el caso para el perceptrón multicapa, este puede ser usado tanto como para clasificación o regresión, por lo tanto, puede realizar la función no lineal. En la Figura 29 vemos como el perceptrón multicapa da solución al problema de la función XOR, donde recibe los elementos externos x_1, x_2 y los valores bias o sesgo x_0, se calculan los pesos, y los resultados activan la "función de activación" de la sigmoide y estas activan la capa oculta z_1, z_2, en el cual reciben nuevamente otra fuente de valor bias o sesgo z_0, se vuelven a calcular los pesos llegando a la señal de salida y.

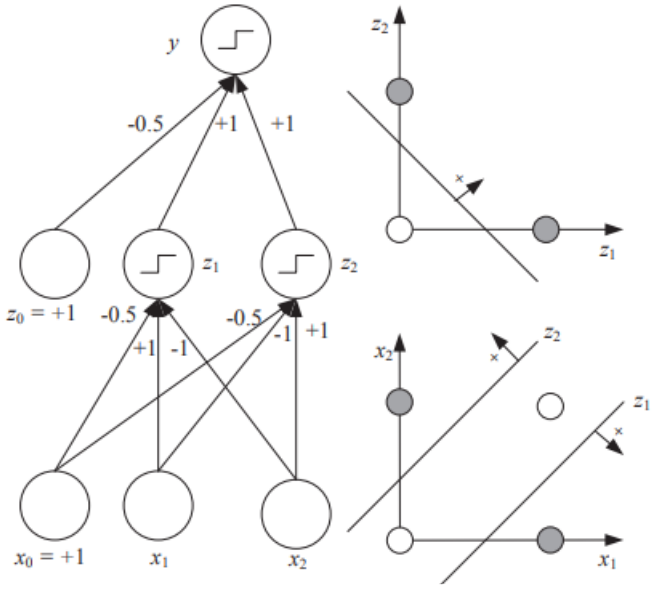


Figura 29: Solución de la función XOR con el Perceptrón multicapa. Fuente:(Alpaydin, 1965)

Bagging

También conocido como Agregación Bootstrap, se define como un meta-algoritmo diseñado para conseguir combinaciones de modelos a partir de una familia inicial, provocando una disminución de la varianza y evitando el sobreajuste (F. Cappirini, 2009). El sobreajuste ocurre cuando un modelo se ajusta demasiado a los datos de entrenamiento.

La técnica consiste en lo siguiente (F. Cappirini, 2009):

- Dado un conjunto de entrenamiento, D, de tamaño N, se generan M nuevos conjuntos de entrenamiento, D_i, de tamaño N', tomando al azar elementos de D de manera uniforme y con reemplazo. Por lo tanto, algunos elementos del conjunto original pueden aparecer repetidos en los nuevos conjuntos generados.
- Si N'=n, entonces para valores de N suficientemente grandes, se espera que cada D_i tenga una fracción de $(1 - 1/e) \approx 63.2\%$ elementos únicos de D, y el resto son duplicados.

- A partir de estos M nuevos conjuntos de entrenamiento se construyen M nuevos modelos de aprendizaje, y la respuesta final de la combinación se consigue por medio de votación de las M respuestas (en caso de buscar una clasificación) o por la media de ellas (en caso de buscar una regresión).

Bagging tiende a producir mejoras en los casos de modelos individuales inestables (como es el caso de las redes neuronales o los árboles de decisión).

Métricas

El desempeño de los modelos de clasificación se mide a través de métricas que evalúan la capacidad de estimar correctamente la clase de cada instancia. La mayoría de métricas se basan en clasificación binaria (dos clases), considerando como valores de clase Si (positivo) o No (negativo), y toman como valores básicos el conteo de predicciones correctas e incorrectas, como sigue:

- Positivos verdaderos (TP): estos son los valores positivos pronosticados correctamente, lo que significa que el valor de la clase real y el valor de la clase pronosticada coinciden.
- Verdaderos negativos (TN): estos son los valores negativos predichos correctamente, lo que significa que el modelo pronostica correctamente que la instancia evaluada no corresponde a la clase positiva.
- Falsos positivos (FP): casos en que el modelo pronostica incorrectamente la clase positiva.
- Falsos negativos (FN): casos en que el modelo pronostica incorrectamente la clase negativa.

A continuación, se detallan las métricas empleadas en este Proyecto de Título (Exilio solutions, 2016).

- **Precision:** En la ecuación (9) se muestra la relación entre las observaciones positivas predichas correctamente y el total de observaciones positivas:

$$\frac{TP}{TP\ +\ FP} \tag{9}$$

- **Recall:** Es la proporción de observaciones positivas predichas correctamente con respecto a todas las observaciones en la clase real. En la ecuación (10) se muestra la fórmula del Recall:

$$\frac{TP}{TP\ +\ FN} \tag{10}$$

- **F1-score:** Es la media armónica entre *precision* y *recall*. Por lo tanto, este puntaje tiene en cuenta tanto los falsos positivos como los falsos negativos En la ecuación (11) se muestra la fórmula del F1-score:

$$2 \cdot \frac{Precision \cdot\ Recall}{Precision\ +\ Recall} \tag{11}$$

- **Accuracy:** Mide la frecuencia con que el modelo acierta en las predicciones que realiza. En la ecuación (12) se muestra la fórmula del *accuracy*:

$$\frac{TP\ +\ TN}{TP\ +\ TN\ + FN + FP} \tag{12}$$

3. Trabajos relacionados

En este capítulo se describen trabajos relacionados que han abordado la detección de anomalías como grietas, roturas y desacoplamiento en las uniones de las tuberías. Cabe destacar que anomalías como incrustaciones y unión penetrante no son consideradas en estos trabajos.

3.1 Morphological segmentation based on edge detection for sewer pipe defects on CCTV images

Lo descrito a continuación corresponde a resumen propuesto por los autores originales en su respectiva publicación (Su, Yang, Wu, & Lin, 2011).

Esta investigación se basa en las técnicas de procesamiento de imágenes e inteligencia artificial para el desarrollo de un sistema de diagnóstico, en base a morfologías segmentadas de imágenes de CCTV (circuitos cerrados de televisión). Se proponen 2 soluciones a esta problemática, la primera se llama "Morphological Segmentation based on Edge Detection" (MSED) y la segunda, Top-Hat, estos métodos deben detectar las deformidades, grietas y las uniones defectuosas.

Algunos ejemplos de estas anomalías se muestran en la Figura 30 Las imágenes en la primera fila corresponden a grietas, las imágenes en la segunda fila corresponden a roturas, las imágenes en la tercera fila corresponden a uniones defectuosas con grandes roturas y las imágenes en la cuarta fila corresponden a deformidades.

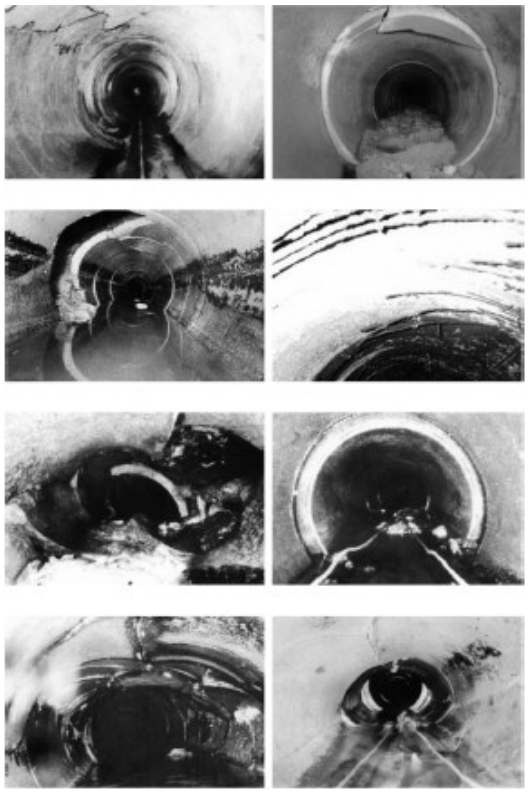


Figura 30: Muestra de anomalías utilizadas en el artículo. Fuente: (Su et al., 2011)

3.1.1 Diagrama de las técnicas de MSED y Top-Hat.

La Figura 31 muestra un diagrama esquemático de estos métodos. En el caso del MSED, se utiliza una detección de bordes por Sobel, Prewitt y Roberts, el siguiente paso es determinar el mejor método, en este caso es Sobel según lo señalado por (Su et al., 2011), terminando con una segmentación basada en los bordes. En el caso del Top-Hat se inicia con operaciones morfológicas, el siguiente paso es restarle el resultado de esta operación a la imagen, y al final se aplica un proceso de segmentación de imagen binaria por el método de Otsu, resultando en una imagen de bordes.

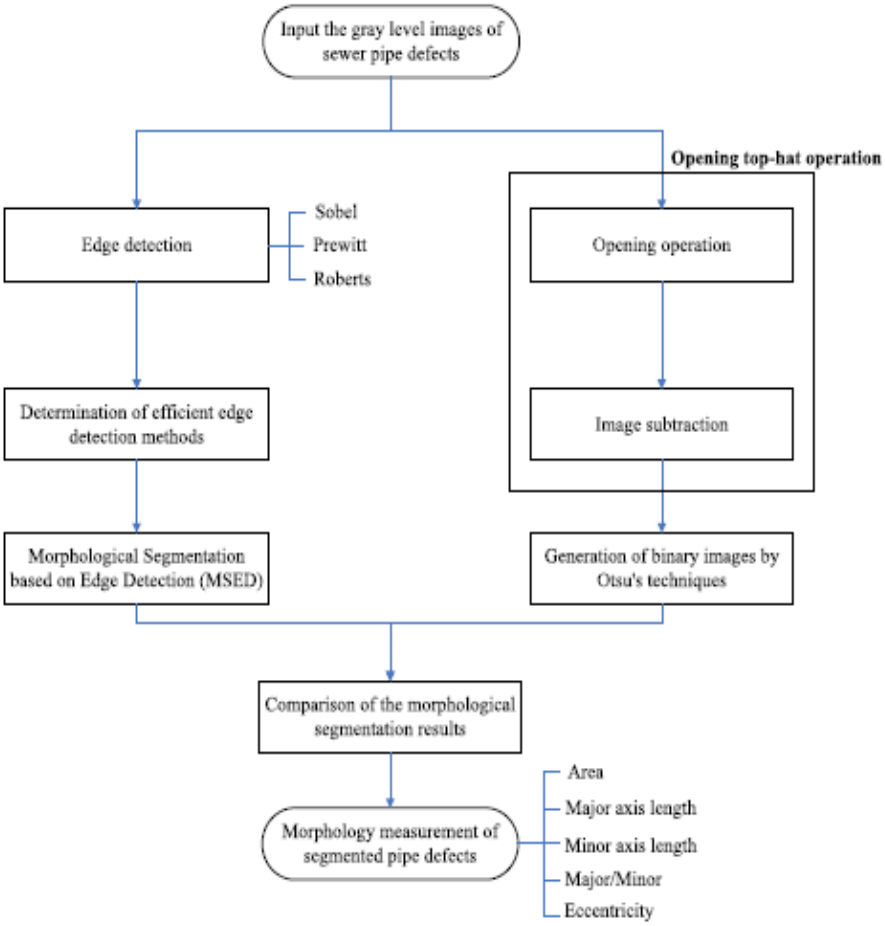


Figura 31: Pasos descriptivos del método por MSED y Top-Hat. Fuente: (Su et al., 2011)

3.1.2 Top-Hat

En primer lugar, se explica el método de Top-Hat, describiendo cuáles son los elementos estructurantes aplicados en el kernel, y como llegar al Top-Hat por medio de los operadores básicos de procesamiento de imágenes, de dilatación y erosión.

Opening Operation, image subtraction

Las partes claras y oscuras de la imagen pueden ser remodeladas o transformadas de varias maneras bajo el control de un elemento estructurante que puede considerarse como un parámetro para la operación morfológica (Sinha & Fieguth, 2006). En la Figura 32 se visualizan los elementos estructurantes ocupados en el artículo (Su et al., 2011).

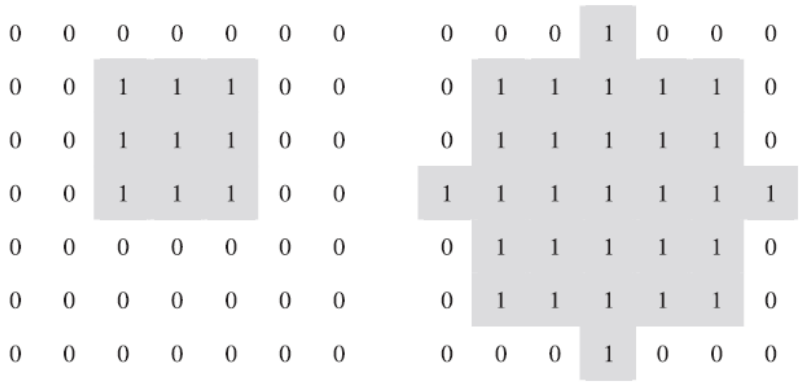


Figura 32: elementos estructurantes, cuadrado y diamante. Fuente: (Su et al., 2011)

La dilatación y la erosión son las dos operaciones morfológicas básicas (Bradski & Kaehler, 2008) que se ocupan en esta operación, para explicarlo mejor se mostrara una demostración de cómo llegar al Top-Hat a través de los métodos básicos de las operaciones morfológicas (erosión y dilatación).

Los conjuntos A y B se definen para representar una imagen que consta de píxeles p (x, y) y un elemento de estructuración, respectivamente:

$A = \{(x,y)|\, p(x,y)\}$

$B = \{(x,y)|(x,y)\ en\ elemento\ estructurantes\}$

La dilatación de A por B, denominada $A\theta B$, es la unión de todos los píxeles en A rodeados por la forma de B y definidos como (para más información de la dilatación revisar la sección 2.2.1.1):

$A\ \theta\ B = \{a\ + b\ |$ para todo a $\epsilon\ A\ y\ b\ \epsilon\ B\}$

De manera similar, la erosión de A por B, denotada como AφB, elimina todos los píxeles dentro de una distancia B desde el borde A y se define como (para más información de la erosión revisar la sección 2.2.1.1):

A φ B = {a | b + a ϵ A para cada b ϵ B}

Entonces la operación de apertura morfológica se puede definir de la siguiente manera (para más información de la apertura morfológica revisar la sección 2.2.1.1):

A ° B = (A φ B) θ B

El efecto que crea la operación de apertura morfológica es eliminar regiones de la imagen que están ligeramente relacionados con el elemento estructurante, al tiempo que se conservan regiones de imagen más grande que el elemento estructurante (Sinha & Fieguth, 2006). Relacionado con la operación de apertura podemos definir la apertura de sombrero de copa (Top-Hat) como:

$OTH_{A^{\circ}B} = A - (\text{A}\ \varphi\ \text{B})\ \theta\ B$

Tanto la dilatación como la erosión necesitan un elemento estructurante, como poder realizar de manera satisfactoria las operaciones morfológicas, los elementos ocupados en este proceso son de la del cuadrado y diamante, sabemos que estos elementos pueden tener diferente forma o tamaño, en énfasis a la vecindad de 1's, que se encargan de detectar los bordes (Su et al., 2011), dicho lo anterior esto fue necesario para la elaboración de la operación de sombrero de copa.

Procesamiento binario a través de la técnica de Otsu.

La técnica de Otsu, es una técnica de imagen binaria de manera automatizado, se encarga de crear este tipo imagen después del resultado del sombrero de copa (Top Hat), resaltando en gran medida los bordes detectados para más información y cómo funciona el método de Otsu revisar sección 2.2.1

3.1.3 MSED (Morphological Segmentation based on Edge Detection)

En segundo lugar, se explica el método MSED, detallando las principales etapas que considera.

Edge detection

La detección de bordes es un enfoque más común para detectar discontinuidades significativas en una imagen en escala de grises, ya que estas se ven enfrentadas sobre la diferencia de luz. Para llevar a cabo esto se utiliza la primera y segunda derivada del perfil de nivel de grises.

A continuación, se mostrarán algunas matrices de los operadores de bordes, las cuales corresponden a Sobel, Prewitt y Roberts, Las matrices de la figura 33 corresponde a Sobel. La Figura 34 corresponde a Prewitt y la Figura 35 corresponde a Roberts. Estas matrices poseen valores negativos y positivos, los que representan diferencia de luminosidad en la imagen. Estas matrices son utilizadas para detectar bordes de manera horizontal o vertical en una imagen. De acuerdo a lo sugerido en el (Su et al., 2011).

-1	-2	-1
0	0	0
1	2	1

-1	0	1
-2	0	2
-1	0	1

Figura 33: Matrices representativas a Sobel para una detección de bordes. Fuente: (Su et al., 2011)

-1	-1	-1
0	0	0
1	1	1

-1	0	1
-1	0	1
-1	0	1

Figura 34: Matrices representativas a Prewitt para una detección de bordes. Fuente: (Su et al., 2011)

1	0
0	-1

0	1
-1	0

Figura 35: Matrices representativas a Roberts para una detección de bordes. Fuente: (Su et al., 2011)

Ejemplo de funcionamiento de MSED

A continuación, se muestra un ejemplo de cómo opera esta metodología de segmentación MSED sobre un cuadrado.

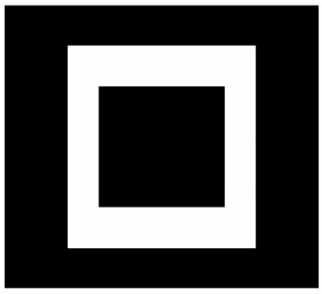


Figura 36: Ejemplo de funcionamiento de método MSED. Fuente: (Su et al., 2011)

Al inicio del cuadrado (Figura 36) se interpreta el valor, ya sea 255 o 0 del pixel P(1 , 1) de la imagen binaria completa, si el valor es 255, entonces consideramos la segmentación inicial, de lo contrario pertenece a un fondo oscuro en la imagen original, el procedimiento empieza al hallar un valor de 255 (borde) se le proporciona una distancia d ($d = 4$), si la dirección del pixel p(1 , 1 + d) es un pixel con valor de 255 (borde) todos los valores que se encuentren en este intervalo desde el pixel p(1 , 1) hasta el pixel p(1, 1 + d) serán asignados con el valor de 255, de lo contrario, los valores que se encuentran en este intervalo se le mantendrán sus valores respectivos (0). Este algoritmo repite el proceso por columnas y filas, ósea en ambas direcciones, y llenando el objeto de interés, en la Figura 37 se visualiza como el objeto de interés (cuadrado) se va llenado según el funcionamiento del MSED.

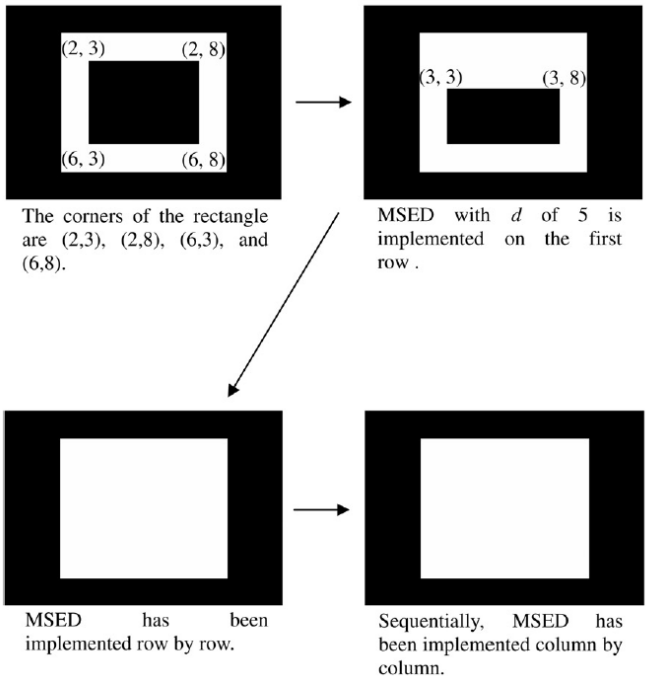


Figura 37: Ejemplo de funcionamiento de método MSED sobre un cuadrado. Fuente: (Su et al., 2011)

3.1.4 Extracción de características

Las características utilizadas en estos dos métodos son las mismas, las cuales son eje menor, eje mayor, excentricidad, ratios de ejes y el área (descritas en el Capítulo 4). Gran parte de estas características son operadores que trabajan con los bordes. Estas características se pueden obtener mediante rutinas provistas por la biblioteca OpenCV (OpenCvTeam, 1999) y el lenguaje Matlab. Los autores del artículo utilizaron esta última opción.

4. Características de imagen para clasificación.

En este capítulo se dan a conocer las diferentes características utilizadas en este proyecto de título, se explicaran su funcionamiento y ecuación o función que se ha utilizado, estas características fueron escogidas por medio de los siguientes artículos (Su et al., 2011), (Navarrete, 2019), (Guo, Soibelman, & Garrett, 2009)

- **Área:** Corresponde al área de los contornos del objeto de interés en una imagen. En este trabajo, se obtiene mediante una función de la biblioteca de OpenCV *contourArea(contorno)* (Su et al., 2011), en la Figura 38 se puede ver los contornos de la grieta donde la función de OpenCv las recibe y devuelve el área.

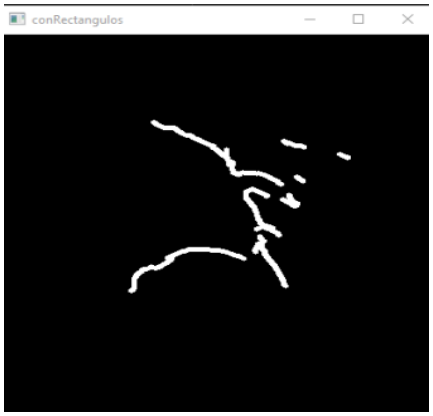


Figura 38: Grieta de una tubería imagen procesada Fuente: (Elaboración propia)

- **Ejes en una elipse:** Corresponde a la longitud del largo de los ejes en una elipse que rodea el objeto de interés, en donde se pueden encontrar el eje mayor y eje menor, estas características son utilizadas por (Su et al., 2011), en la Figura 39 se puede visualizar los ejes en el primer cuadrante, los valores que reciben los ejes son positivos.

Figura 39: Representación de los ejes y focos de una elipse. Fuente: (Varsity LLC, 2008)

- **Ratio de ejes:** Esta característica es muy simple, corresponde a una división entre los dos ejes obtenidos de la elipse, ver ecuación (13) (Su et al., 2011).

$$\text{Ratio de ejes} = {Eje\ Mayor}/{Eje\ Menor} \tag{13}$$

- **Excentricidad:** es una relación de la distancia entre los dos ejes (Su et al., 2011). El valor de la excentricidad se encuentra en el intervalo de 0 a 1, para extraer esta característica se utiliza la ecuación (14)

$$\in = \sqrt{\frac{a^2 - b^2}{a^2}} = \sqrt{1 - \frac{b^2}{a^2}} \tag{14}$$

En la ecuación (14) a es el valor del eje mayor, y b es el valor del eje menor.

- **Perímetro**: Corresponde al perímetro del objeto detectado, en este caso es el máximo del borde detectado en el objeto, esto se obtiene mediante la siguiente función $arcLength()$ (OpenCvTeam, 1999) aplicada al contorno (Navarrete, 2019).

- **Intensidad media:** Esta característica corresponde al nivel de intensidad de todos los valores de los pixeles, para realizar la media, se hace un promedio de todas estas

intensidades y el resultante se guarda como característica utilizadas por (Guo et al., 2009), en la ecuación (15), k representa la cantidad de pixeles de intensidad del roi, i representa la intensidad del pixel.

$$\text{Intensidad Media} = \frac{\sum_{n=0}^{k} i_n}{k} \tag{15}$$

- **Color medio:** Esta característica se calcula a partir de nivel de rojo, verde y azul (RGB) en la imagen. El proceso es muy similar al utilizado para determinar la intensidad media, se trabaja los niveles por separado y se calcula un promedio de cada uno por todos los pixeles encontrados en el área (Guo et al., 2009), en la ecuación (16), k representa la cantidad de pixeles que contiene del roi, C representa el canal de color del pixel, este proceso se repite 3 veces, para los 3 canales de una imagen RGB, tanto para Red, Green y Blue.

$$\text{Color Medio} = \frac{\sum_{n=0}^{k} C_n}{k} \tag{16}$$

5. Implementación

En este capítulo se explicará cómo se abordaron la metodología de detección de bordes estudiadas en los artículos (Sobel (MSED), Canny, Top-Hat) estudiados, además se darán a conocer las diferentes funciones utilizadas en la parte de preprocesamiento y extracción de características.

5.1 Diagrama esquemático de los pasos de implementación

En la Figura 40 se muestran los principales pasos de la implementación realizada en nuestro proyecto de título, cada etapa se explicará, en las siguientes secciones

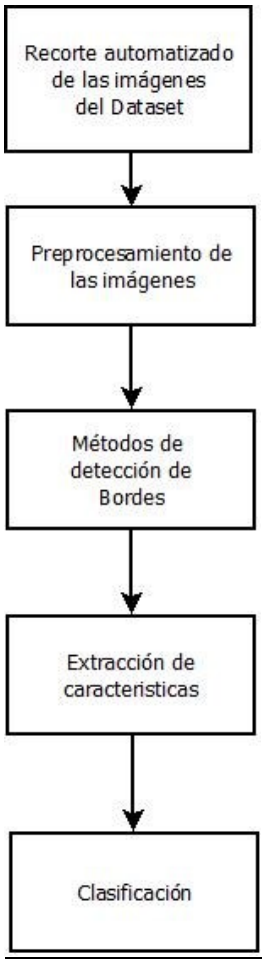


Figura. 40: Diagrama General de la implementación. Fuente: (Elaboración Propia)

5.2 Recorte Automatizado de las imágenes del dataset

Las imágenes fueron estandarizadas (dejándolas del mismo tamaño) mediante el programa Wondershare Filmora9. Luego Analizando todas las imágenes obtenidas nos dimos cuenta de ciertas cantidades de ruidos generados por la cámara de ITV O CCTV por ende el primer paso es quitar caracteres relativos a fecha, hora, ubicación y otros datos que son sobrepuestos a las imágenes por estas cámaras. Esto, para efectos de la detección de anomalías en las tuberías, representa ruido que puede entorpecer en gran medida la extracción de características. Por lo anterior, se decidió quitar las regiones de las imágenes en las que se ubican estos caracteres, mediante el programa Labelimg, con el cual se seleccionaba un área de interés de forma general para todas las imágenes, ya que estas etiquetas (caracteres agregados por el dispositivo de grabación) se encuentran en la misma posición, para cortar la imagen de manera automatizada. En la Figura 41 se muestra un ejemplo de lo que se decide cortar de la imagen. En la Figura 42, se puede visualizar la imagen ya cortada.

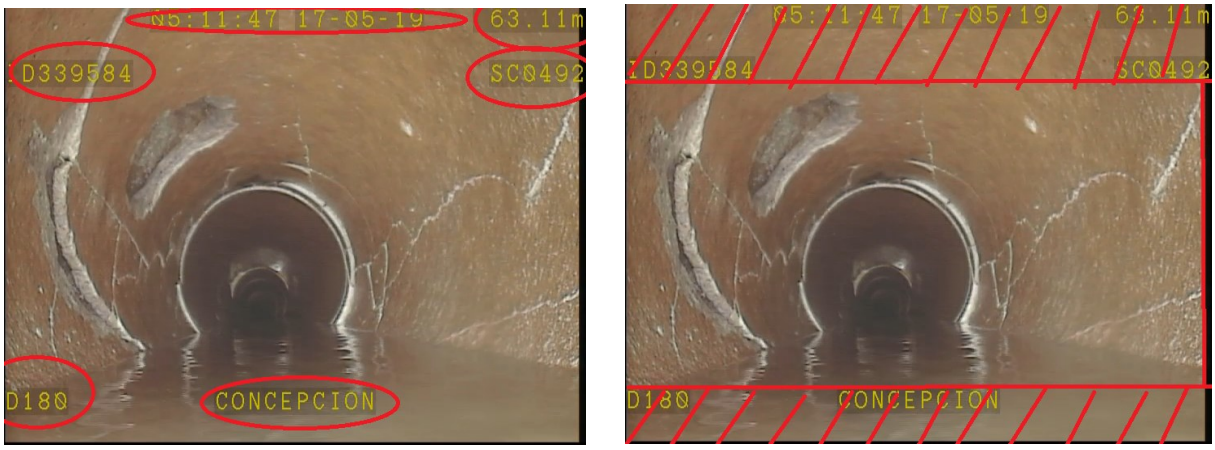


Figura. 41: Imágenes con el ruido generado por las cámaras de CCTV o ITV. Fuente: (elaboración Propia)

Figura 42: Imagen resultante al eliminar regiones con caracteres. Fuente: (Elaboración propia).

Una vez que se eliminan las regiones con caracteres de la imagen, está lista para pasar por la etapa de preprocesamiento y extracción de características, de cada metodología estudiada. Esta primera etapa de recorte automatizado se aplica a todas las imágenes utilizadas.

5.3 Preprocesamiento de las imágenes

En esta sección se darán a conocer los métodos utilizados para la limpieza de las imágenes del dataset, utilizando operadores morfológicos de la biblioteca de OpenCv.

Imagen en escala de grises

Para esta transformación existe una formula estándar, la cual se aplica a cada pixel de una imagen para transformarla a escala de grises, según muestra la ecuación (17). (para más información de la imagen en escala de grises ver sección 2.2.1.1 de operaciones morfológicas)

$$imagen\ gris = 0.2989 * R + 0.5870 * G + 0.1140 * B \quad (17)$$

La función *cvtColor* de la biblioteca de OpenCV permite realizar esta transformación, de la siguiente forma:

$cvtColor(src, dst, code, dstCn)$

Parámetros:

- **src:** Imagen de entrada.
- **dst:** Nombre de la variable que contendrá la imagen de salida.
- **code:** Código para la conversión de espacio de color.
- **dstCn:** Número de canales en la imagen de salida. Si no se especifica, se deriva de forma automática desde src y code.

suavizado de imagen

Esta función aplica un filtro Gaussiano para suavizar la imagen eliminando ciertos ruidos (para más información del filtro gaussiano ver sección 2.2.1.1 de operaciones morfológicas).

$$gaussianBlur(src, dst, ksize, sigmaX, sigmaY, borderType)$$

Parámetros:

- **src:** Imagen de entrada.
- **dst:** Nombre de la variable que contendrá la imagen de salida.
- **ksize:** Tamaño del kernel Gaussiano a utilizar.
- **sigmaX:** Desviación estándar del kernel Gaussiano en la dirección de X.
- **sigmaY:** Desviación estándar del kernel Gaussiano en la dirección de Y. Existe el caso si sigmaY es 0, tomará el valor que contiene sigmaX, en otras palabras, serán similares y si ambos son 0, sigmaX es calculado a partir del ancho de ksize, mientras que sigmaY se calcula a partir de la altura de Ksize.
- **borderType:** Método de extrapolación de pixel.

Disminuir el ruido

Las imágenes recolectadas poseen una muy baja resolución, por ende, poseen una gran cantidad de ruido, y no solamente por la resolución, sino también por cambios de color u otros factores, y que pueden afectar de manera importante la capacidad para detectar defectos en las tuberías

de alcantarillado. Por esta situación, se aplica la apertura morfológica con doble iteración para sacar la gran cantidad de ruido (para más información de la apertura morfológica ver sección 2.2.1.1 de operaciones morfológicas).

$$morphologyEx(src, op, kernel, dst, anchor, iterations, borderType, borderValue)$$

Parámetros:

- **src:** Imagen de entrada.
- **op:** Tipo de operación morfológica, en este caso es de tipo MORPH_OPEN (apertura morfológica), para quitar los ruidos pequeños.
- **kernel:** Elemento estructurante.
- **dst:** Nombre de la variable que contendrá la imagen resultante
- **anchor:** Posición del punto de anclaje donde el kernel es negativo, este argumento puede estar vacío.
- **iterations:** Número de veces que la operación es realizada, en este caso es realizada 2 veces.
- **borderType:** Método de extrapolación de pixel.
- **borderValue:** Valor del borde en caso de un poseer un borde constante.

5.4 Métodos de detección de bordes

En esta etapa, la imagen se encuentra con mucho menos ruido, lo que facilita detectar los bordes de las anomalías correspondientes. Se darán a conocer los 3 métodos de detección de bordes utilizados en nuestro proyecto de título, además algunos operadores morfológicos para unir bordes y la utilización de imagen binaria en unos de los métodos.

5.4.1 Detección de bordes con Canny

En esta sección se hablará del método de detección de bordes de Canny y algunos operadores morfológicos básicos utilizados (para más información sobre el método de detección de bordes de Canny ver sección 2.2.2).

Detección de bordes de Canny

El método de detección de bordes de Canny se utilizó por medio de la biblioteca de OpenCv, la función es la siguiente:

$$canny(src, edges, threshold1, threshold2, apertureSize, Le2grandient)$$

Parámetros:

- **src:** Imagen de entrada.
- **edges:** imagen resultante con los bordes detectados.
- **threshold1:** Primer umbral para el procedimiento de histéresis.
- **threshold2:** Segundo umbral para el procedimiento de histéresis.
- **apertureSize:** Tamaño de la apertura del operador Sobel.
- **l2grandient:** Se define la magnitud del gradiente de la imagen a calcular.

Incremento de los bordes de interés

En esta operación, se identifican los pixeles que pertenecen a los bordes, y se les agregan más capas de pixeles por medio de una dilatación morfológica, para juntar las grietas o roturas que deberían estar juntas (para más información sobre la dilatación revisar la sección 2.2.1.1)

$$dilate(src, dst, kernel, anchor, iterations, borderType, borderValue)$$

Parámetros:

- **src:** Imagen de entrada.
- **dst:** Nombre de la variable que contendrá la imagen resultante

- **kernel:** Elemento estructurante.
- **anchor:** Posición del punto de anclaje donde el kernel es negativo, este argumento puede estar vacío.
- **iterations:** Número de veces que la operación es realizada, en este caso es realizada 1 veces.
- **borderType:** Método de extrapolación de pixel.
- **borderValue:** Valor del borde en caso de un poseer un borde constante.

Eliminar ruido interior

Se utiliza la operación de cerrado morfológico para eliminar el ruido que quedan al unir los bordes, la función del cerrado morfológico se muestra a continuación (para más información del cerrado morfológico ver la sección 2.2.1.1):

$$morphologyEx(src, op, kernel, dst, anchor, iterations, borderType, borderValue)$$

Parámetros:

- **src:** Imagen de entrada.
- **op:** Tipo de operación morfológica, en este caso es de tipo MORPH_CLOSE (cerrado morfológico), para quitar los ruidos pequeños del objeto de interés.
- **kernel:** Elemento estructurante.
- **dst:** Nombre de la variable que contendrá la imagen resultante
- **anchor:** Posición del punto de anclaje donde el kernel es negativo, este argumento puede estar vacío.
- **iterations:** Número de veces que la operación es realizada, en este caso es realizada 1vez.
- **borderType:** Método de extrapolación de pixel.
- **borderValue:** Valor del borde en caso de un poseer un borde constante.

5.4.2 Detección de bordes con Sobel (Msed)

En esta sección se hablará del método de detección de bordes de Sobel según el artículo (Su et al., 2011), algunos operadores morfológicos básicos utilizados y uso de la imagen binaria (para más información de Sobel ver sección 2.2.2)

Sobel eje x eje y

Se realiza la operación de Sobel en la imagen con dirección en el eje x y con dirección en el eje y, por lo tanto, se realiza 2 veces para tener ambas direcciones.

$$Sobel(src, dst, ddepth, kernelX, kernelY, anchor, delta, borderType)$$

Parámetros:

- **src:** Imagen de entrada.
- **dst:** Imagen resultante, esta posee el mismo tamaño y los mismos canales de la que recibió por src.
- **ddepth:** Profundidad de la imagen de destino.
- **kernelX:** Coeficiente para filtrar por fila.
- **kernelY:** Coeficiente para filtrar por columna.
- **anchor:** Posición del anclaje dentro del núcleo. El valor predeterminado es que el núcleo se encuentra en el centro.
- **delta:** Valor agregado a los resultados filtrados antes de almacenarlos.
- **borderType:** Método de extrapolación de píxeles

Juntar las imágenes de Sobel en dirección "eje x" y "eje y"

El siguiente paso es juntar estas imágenes mediante el siguiente operador.

$$addWeighted(src1, alpha, src2, gamma, dst, dtype)$$

Parámetros:

- **src1:** Primera matriz de entrada (resultado al usar el operador Sobel eje x).
- **alfa:** Peso de los primeros elementos de la matriz.
- **src2:** Segunda matriz de entrada del mismo tamaño y numero de canal que src1 (resultado al usar operador Sobel eje y).
- **beta:** Peso de los elementos de la segunda matriz.
- **dst:** Matriz de salida que tiene el mismo tamaño y canal que las otras 2 matriz de entradas (src1, src2).
- **dtype:** Profundidad es opcional en la matriz de salida; cuando ambas matrices de entrada tienen el mismo tamaño, dtype se puede establecer en -1, que será equivalente a src1.depth().

Elaboración de imagen binaria

Se crea la imagen binaria para resaltar los bordes. En este caso se utilizó una imagen binaria con un umbral de valor 100.

$$threshold(src, dst, thresh, maxval, type)$$

Parámetros:

src: Matriz de entrada (imagen), pueden ser de múltiples canales, 8-bit o 32-bit.

dst: Matriz de salida (imagen) del mismo tipo, tamaño y número de canales que la imagen entrante (src1),

thresh: Valor del umbral.

maxval: Valor máximo para usar con los tipos de threshold.

type: Tipo de threshold en este caso THRESH_BINARY.

Incremento de los bordes de interés

En esta operación, se identifican los pixeles que pertenecen a los bordes, y se les agregan más capas de pixeles por medio de una dilatación morfológica, para juntar las grietas o roturas que

deberían estar juntas, esta operación se realiza con una iteración doble (para más información sobre la dilatación revisar la sección 2.2.1.1)

$$dilate(src, dst, kernel, anchor, iterations, borderType, borderValue)$$

Parámetros:

- **src:** Imagen de entrada.
- **dst:** Nombre de la variable que contendrá la imagen resultante
- **kernel:** Elemento estructurante.
- **anchor:** Posición del punto de anclaje donde el kernel es negativo, este argumento puede estar vacío.
- **iterations:** Número de veces que la operación es realizada, en este caso es realizada 2 veces.
- **borderType:** Método de extrapolación de pixel.
- **borderValue:** Valor del borde en caso de un poseer un borde constante.

Eliminar ruido interior

Se utiliza la operación de cerrado morfológico para eliminar el ruido que quedan al unir los bordes, la función del cerrado morfológico se muestra a continuación (para más información del cerrado morfológico ver la sección 2.2.1.1):

$$morphologyEx(src, op, kernel, dst, anchor, iterations, borderType, borderValue)$$

Parámetros:

- **src:** Imagen de entrada.
- **op:** Tipo de operación morfológica, en este caso es de tipo MORPH_CLOSE (cerrado morfológico), para quitar los ruidos pequeños del objeto de interés.
- **kernel:** Elemento estructurante.
- **dst:** Nombre de la variable que contendrá la imagen resultante

- **anchor:** Posición del punto de anclaje donde el kernel es negativo, este argumento puede estar vacío.
- **iterations:** Número de veces que la operación es realizada, en este caso es realizada 1 sola vez.
- **borderType:** Método de extrapolación de pixel.
- **borderValue:** Valor del borde en caso de un poseer un borde constante.

5.4.3 Detección de bordes con Top-Hat

En esta sección se dará a conocer el otro método mencionado en el artículo estudiado (Su et al., 2011), apodado Top-Hat (gorro de copa) y un proceso de segmentación de imagen binaria de Otsu.

Top-Hat

Para hacer uso del Top-Hat, se utilizó un operador de OpenCv, el cual se detalla a continuación (para más información del funcionamiento del Top-Hat ver sección 3.1.2)

$$morphologyEx(src, op, kernel, dst, anchor, iterations, borderType, borderValue)$$

Parámetros:

- **src:** Imagen de entrada.
- **op:** Tipo de operación morfológica, en este caso es de tipo MORPH_TOPHAT (Top-Hat).
- **kernel:** Elemento estructurante.
- **dst:** Nombre de la variable que contendrá la imagen resultante
- **anchor:** Posición del punto de anclaje donde el kernel es negativo, este argumento puede estar vacío.
- **iterations:** Número de veces que la operación es realizada, en este caso es realizada 1 vez.
- **borderType:** Método de extrapolación de pixel.

- **borderValue:** Valor del borde en caso de un poseer un borde constante.

Imagen binaria con el método de Otsu

El siguiente paso es utilizar el método Otsu para crear la imagen binaria con los bordes encontrados al utilizar la detección de bordes del Top-Hat.

$$threshold(src, dst, thresh, maxval, type)$$

Parámetros:

- **src:** Matriz de entrada (imagen), pueden ser de múltiples canales, 8-bit o 32-bit.
- **dst:** Matriz de salida (imagen) del mismo tipo, tamaño y número de canales que la imagen entrante (src1),
- **thresh:** Valor del umbral.
- **maxval:** Valor máximo para usar con los tipos de threshold.
- **type:** Tipo de threshold en este caso THRESH_OTSU.

5.5 Extracción de características

Las características seleccionadas para la extracción, se basan en la forma. Por lo tanto, el primer paso para extraer características, es buscar los bordes en la imagen resultante de los detectores de bordes.

5.5.1 Diagrama de extracción de características

En la Figura 43 se puede visualizar el proceso de extracción de características de forma general, de los artículos estudiados (Su et al., 2011) (Navarrete, 2019)(Guo et al., 2009) las cuales se explicarán como obtenerlas en la siguiente sección.

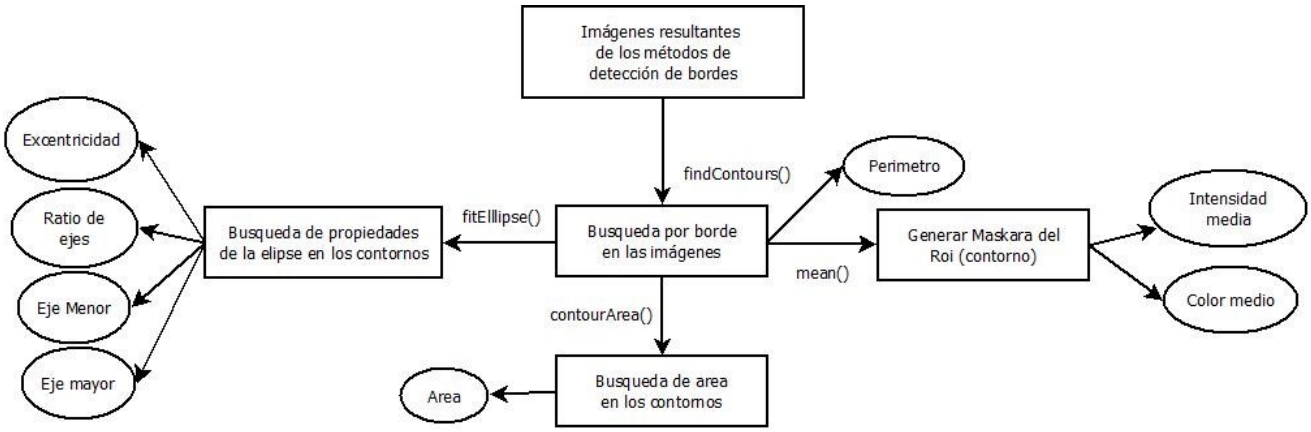


Figura 43: Diagrama esquemático de las características extraídas. Fuente: (Elaboración propia)

5.5.2 Implementación de extracción de características

Búsqueda de bordes

Este método es primordial, porque a través de éste se puede obtener todas las características mencionadas en el diagrama de la Figura 43, a continuación, se visualizará la función ocupada y sus parámetros.

$$findContours(image, mode, method, contours, hierachy, offset)$$

Parámetros:

- **image:** Imagen binaria de entrada.
- **mode:** El modo en el que se van a extraer los contornos.
    - **Cv2.CV_RETR_EXTERNAL:** Extrae los contornos más externos e la imagen.
    - **Cv2.RETR_LIST:** Extrae todos los contornos presentes en la imagen, sin establecer relaciones jerárquicas.

  - **Cv2.RETR_CCOMP:** Extrae todos los contornos en la imagen y los organiza en una jerarquía de 2 niveles, en el primer nivel superior se ubican los contornos externos (bordes) de los objetos, mientras que en el segundo nivel están los contornos internos de cada objeto.
  - **Cv2.RETR_TREE:** Extrae todos los contornos de la imagen y los organiza en una jerarquía familiar completa.
- **method:** Método de aproximación de contornos.
  - **Cv2.CHAIN_APPROX_NONE:** Se guardan todos los puntos que forman un contorno.
  - **Cv2.CHAIN_APROX_SIMPLE:** Comprime los segmentos verticales, horizontales y diagonales de un contorno, guardando solo los puntos de inicio y fin de tal segmento.
  - **Cv2.CHAIN_APPROX_TC89_L1, CV_CHAIN_APPROX_TC89_KCOS:** Aplica una de las formas del algoritmo de aproximación de cadenas de Teh-Chain.
- **contours:** Variable sobre la cual se guardan los contornos encontrados.
- **hierarchy:** Vector de salida opcional, contiene información adicional respecto a los contornos, como el número de estos encontrados.
- **offset:** Offset opcional usado para desplazar cada punto que forma un contorno.

Área de la anomalía

Esta área es calculada usando los contornos, mediante un operador entregado por la librería de "OpenCv", la cual se mostrará a continuación.

$$contourArea(contour, oriented)$$

Parámetros:

- contour: Vector de puntos en 2D.

- oriented: Si su valor es “true” permite que la función u operador, entregue la información respecto a la orientación de un contorno, por defecto su valor es false.

Características de una Elipse

Para sustraer el eje menor, eje mayor, ratio y su excentricidad es necesario, encontrar los factores de una elipse en las anomalías, para esto se utiliza una función de OpenCv, que se mostrara a continuación

$$fitEllipse(contours)$$

Parámetros:

- Contornos: Entrada de puntos en 2D

Esta función entrega 2 valores, las cuales corresponden al eje mayor y eje menor, para seleccionar los ejes de manera correcta se utiliza los métodos de java de Math con los abs y los argumentos de “max” y.” min”, dejando el eje menor con lo que entrega el valor mínimo y el eje mayor con el valor máximo, para el caso del ratio solo se divide el eje mayor por el eje menor y por último la excentricidad se aplica una fórmula matemática mencionada en el capítulo 4, utilizando los ejes de la elipse (menor y mayor).

Maskara

La makara es esencial para la extracción de características del color medio e intensidad media, continuación se visualizará la función necesaria para adquirir estas características mencionadas anteriormente.

$$mean(img, mask)$$

Parámetros:

- **img**: Imagen original (RGB) o imagen en escala de grises (intensidad).

- **mask**: Objeto detectado (ROI).

Para realizar este procedimiento, se necesita la maskara, la cual es necesario crear una matriz con valores 0's (imagen binaria en negro) del mismo tamaño de la original, la cual se dibuja sobre ella los contornos de interés que se extrae del algoritmo de detección de bordes, sobre estos contornos se aplica una operación morfológica básica que adhiere capas de pixeles, llamada dilatación y a través de esto se realiza una media del valor de los pixeles, para obtener ya sea la intensidad o el color medio (sobrepone en la imagen original o en la imagen de escala de grises).

$$dilate(src, dst, kernel, anchor, iterations, borderType, borderValue)$$

Parámetros:

- **src:** Imagen de entrada.
- **dst:** Nombre de la variable que contendrá la imagen resultante
- **kernel:** Elemento estructurante.
- **anchor:** Posición del punto de anclaje donde el kernel es negativo, este argumento puede estar vacío.
- **iterations:** Número de veces que la operación es realizada, en este caso es realizada 1vez.
- **borderType:** Método de extrapolación de pixel.
- **borderValue:** Valor del borde en caso de un poseer un borde constante.

5.6 Clasificación

Ya teniendo todas las características extraídas, se realiza el entramiento para la clasificación mediante 4 algoritmos de Machine Learning, mediante un api llamada Weka, y trabajar estos métodos como caja negra. Para más información de los algoritmos de entrenamientos escogidos ver la sección 2.2.3

Para el entrenamiento, se utilizó la metodología validación cruzada (cross validation), con 10 folds. Esto consiste en dividir el dataset en 10 subconjuntos, para realizar 10 repeticiones de entrenamiento y prueba. En cada repetición, se deja uno de los subconjuntos como conjunto de prueba (con el cual se contrastan las clasificaciones hechas por el modelo), y las restantes 9 son utilizadas para entrenar el modelo.

Para más detalles de los resultados obtenidos se realizará una experimentación que se verá en el capítulo 6.

6. Experimentación

En este capítulo se detallan los experimentos realizados con el fin de evaluar los algoritmos implementados. Los resultados se presentan en diferentes tablas, diferenciadas según el algoritmo de aprendizaje y características utilizadas. Por último, se realiza una discusión a partir de los resultados obtenidos para cada escenario utilizado.

6.1. Datos y algoritmos utilizados.

El dataset está compuesto de imágenes de tuberías en buen estado y otras con anomalías, de tipo rotura, grieta y desacople, obtenidas de videos de CCTV, contiene 142 imágenes de tuberías en buen estado y 63 con anomalías.

El desbalanceo del dataset de imágenes se debe a que se trabajó a nivel de objeto (detectados dentro de una imagen) y en las imágenes con anomalías se ha detectado una mayor cantidad de objetos con respecto a las imágenes sin anomalía, por ende, en la cantidad total de objetos detectados se igualan. Cabe señalar que la cantidad de objetos detectados varia con respecto al algoritmo de detección de bordes utilizado.

Descripción de imágenes:

- Tamaño de 720 x 540 pixeles.
- Formato PNG.

Algoritmos de entrenamiento de clasificadores:

- Random Forest.
- Bagging.
- J48.
- Multilayer Perceptron.

Algoritmos de detección de bordes.

- Canny.
- MSED(Sobel).

El algoritmo de detección de bordes Top-Hat no mostró resultados coherentes con las imágenes del dataset utilizado en pruebas iniciales, probablemente por el hecho de poseer una resolución precaria y dado que este algoritmo es demasiando susceptible al ruido. Por ello, fue descartado para la etapa de experimentación.

A continuación, se muestran algunas imágenes obtenidas al aplicar algoritmo, que permiten observar la baja calidad de sus resultados.

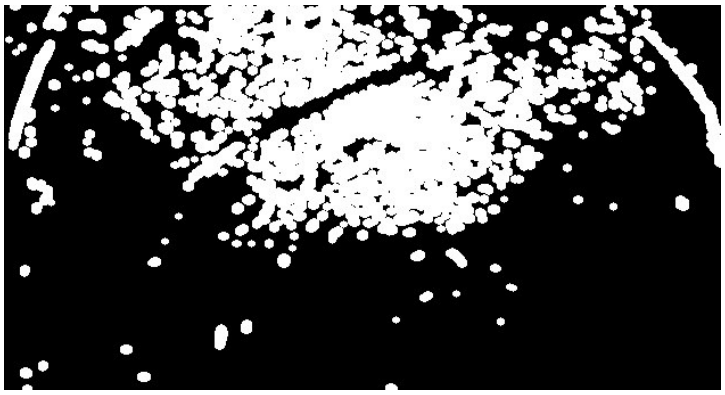


Figura 44: Top-Hat detección de bordes. Fuente: (Elaboración propia).

Figura 45: Top-Hat detección de bordes. Fuente: (Elaboración propia).

6.2. Caso de prueba

Para la evaluación, Se utilizan diferentes características para medir el desempeño de clasificación y comparar resultados de los 2 algoritmos de extracción de características utilizados. Para evaluar el desempeño se utilizarán las siguientes métricas de rendimiento.

- Accuracy.
- Precision.
- Recall.
- F1-Scorre.

6.3. Resultados

A continuación, se detallan los resultados obtenidos con las diferentes combinaciones de características, algoritmos de entrenamiento de clasificadores, y algoritmos.

6.3.1 Características base

Se evalúan en primer lugar los resultados obtenidos al utilizar las características base, que corresponden a las características utilizadas en el trabajo (Su et al., 2011) obtenidas con alguno de los 2 algoritmos de extracción de características implementados, el método de detección de bordes Sobel y Canny. Estas características son:

- Eje mayor.
- Eje menor.
- Excentricidad.
- Ratios de ejes.
- Área.

En las siguientes tablas, se muestran los resultados obtenidos atreves del api de Weka de los clasificadores seleccionados, con las características mencionadas en cada ítem respectivamente.

Extracción de características basado en la detección de bordes de Canny

Algoritmos	Precision	Recall	F1-Score	Accuracy(%)
Multilayer Perceptron	**0.835**	**0.802**	**0.797**	**0.80172**
Random Forest	0.792	0.787	0.787	0.78735
J48	**0.841**	**0.802**	**0.796**	**0.80172**
Bagging	0.799	0.790	0.789	0.79023

Tabla 1: Métricas de desempeño en clasificación al emplear las características seleccionadas, obtenidas con el método de extracción de características basado en la detección de bordes de Canny.

Extracción de características basado en la detección de bordes de Sobel

Algoritmos	Precision	Recall	F1-Score	Accuracy(%)
Multilayer Perceptron	0.861	0.856	0.855	0.85632
Random Forest	0.862	0.859	0.858	0.85919
J48	0.879	0.868	0.866	0.86781
Bagging	**0.886**	**0.876**	**0.875**	**0.87643**

Tabla 2: Métricas de desempeño en clasificación al emplear las características seleccionadas, obtenidas con el método Sobel.

En la tabla 1 y 2 se puede observar que con las mismas características (Eje mayor, Eje menor, Excentricidad, Ratio de ejes y Área), el método de detección de bordes de Sobel supera a el método de Canny en su *accuracy.* Sobel obtiene con su mejor algoritmo de clasificación (Bagging) para estas características un 87% de *accuracy,* comparados con el 80% de *accuracy* que se obtiene con el método de Canny con sus clasificadores más puntuados (Random Forest y Multilayer Perceptron).

6.3.2 Perímetro

Posteriormente, evaluó el resultado de añadir la característica "perímetro" a los 2 algoritmos de extracción de características (se mantienen las otras características, 6 en total). Las características evaluadas en este caso son:

- Eje mayor.
- Eje menor.
- Excentricidad.
- Ratios de ejes.
- Área.
- Perímetro

Extracción de características basado en la detección de bordes de Canny

Algoritmos	Precision	Recall	F1-Score	Accuracy
Multilayer Perceptron	**0.837**	**0.805**	**0.800**	**0.80459**
Random Forest	0.750	0.790	0.789	0.79023
J48	0.836	0.799	0.793	0.79885
Bagging	0.776	0.764	0.762	0.76436

Tabla 3: Métricas de desempeño en clasificación al emplear las características seleccionadas, obtenidas con el método Canny.

Extracción de características basado en la detección de bordes de Sobel

Algoritmos	Precision	Recall	F1-Score	Accuracy(%)
Multilayer Perceptron	0.867	0.859	0.857	0.85919
Random Forest	0.872	0.868	0.867	0.86781
J48	**0.889**	**0.879**	**0.874**	**0.87643**
Bagging	0.870	0.865	0.864	0.86494

Tabla 4: Métricas de desempeño en clasificación al emplear las características seleccionadas, obtenidas con el método Sobel.

En las tablas 3 y 4 se puede observar algo similar a los resultados de las tablas 1 y 2 donde el método de detección de borde de Sobel supera el método de Canny en su *accuracy, los* porcentajes que favorecen a Sobel son iguales al de las tablas anteriores (87% de *accuracy*). Canny por su parte pierde *accuracy* en el algoritmo de clasificación j48 (un 1% aproximadamente). Por lo cual podemos inferir que, al añadir la característica de perímetro, (de Objeto) no ayudó al desempeño de ambos métodos.

6.3.3 Color Medio

Se les agregará el color medio a los 2 algoritmos de extracción de características (se mantienen las otras características, 7 en total), para medir su desempeño con los diferentes algoritmos de Machine Learning

- Eje mayor.
- Eje menor.
- Excentricidad.
- Ratios de ejes.
- Área.
- Perímetro
- Color Medio.

Extracción de características basado en la detección de bordes de Canny

Algoritmos	Precision	Recall	F1-Score	Accuracy(%)
Perceptrón Multilayer	**0.879**	**0.879**	**0.879**	**0.87931**
Random Forest	**0.884**	**0.879**	**0.879**	**0.87931**
J48	0.853	0.853	0.853	0.85344
Bagging	0.866	0.866	0.865	0.86494

Tabla 5: Métricas de desempeño en clasificación al emplear las características seleccionadas, obtenidas con el método Canny.

Extracción de características basado en la detección de bordes de Sobel

Algoritmos	Precision	Recall	F1-Score	Accuracy(%)
Multilayer Perceptron	0.889	0.904	0.897	0.88793
Random Forest	**0.903**	**0.899**	**0.899**	**0.89942**
J48	0.888	0.888	0.888	0.88793
Bagging	**0.904**	**0.899**	**0.899**	**0.89942**

Tabla 6: Métricas de desempeño en clasificación al emplear las características seleccionadas, obtenidas con el método Sobel.

Observando los resultados de las tablas 5 y 6, se ve claramente al añadir la característica de color medio en ambos métodos de detección de borde Sobel y Canny, sus métricas (*precisión, recall, f1-score* y *accuracy*) se ven afectadas positivamente. El método de Sobel, vuelve a superar el Método de Canny en su *accuracy*, pero esta vez los algoritmos de clasificación con mejores resultados fueron Random Forest y Bagging con un 89% de *accuracy*. Canny por su parte los algoritmos que obtuvieron mejores resultados fueron Perceptrón Multilayer y Random Forest con un 87% de *accuracy*.

6.3.4 Intensidad Media

Se les agregará la intensidad media a los 2 algoritmos de extracción de características (se mantienen las otras características, 8 en total), para medir su desempeño con los diferentes algoritmos de Machine Learning.

- Eje mayor.
- Eje menor.
- Excentricidad.
- Ratios de ejes.
- Área.
- Perímetro
- Color Medio.
- Intensidad Media

Extracción de características basado en la detección de bordes de Canny

Algoritmos	Precision	Recall	F1-Score	Accuracy(%)
Multilayer Perceptron	0.868	0.868	0.868	0.86781
Random Forest	**0.890**	**0.885**	**0.885**	**0.88505**
J48	0.856	0.856	0.856	0.85632
Bagging	0.878	0.874	0.873	0.87356

Tabla 7: Métricas de desempeño en clasificación al emplear las características seleccionadas, obtenidas con el método Canny.

Extracción de características basado en la detección de bordes de Sobel

Algoritmos	Precision	Recall	F1-Score	Accuracy(%)
Multilayer Perceptron	0.892	0.882	0.887	0.87931
Random Forest	**0.919**	**0.914**	**0.913**	**0.91379**
J48	0.885	0.885	0.885	0.88505
Bagging	0.883	0.879	0.878	0.87931

Tabla 8: Métricas de desempeño en clasificación al emplear las características seleccionadas, obtenidas con el método Sobel.

En las tablas 7 y 8 en las que se agregó la última característica estudiada, la intensidad media, se pude visualizar que el método de detección de borde de Sobel supero a el método de Canny. Esta vez el algoritmo de clasificación Random Forest fue el que destaco en ambos métodos de detección de borde obteniendo un 91% de *accuracy* por parte de Sobel, y un 88% en Canny, se puede apreciar que, al añadir esta nueva característica, (intensidad media) solo el algoritmo de clasificación Random Forest se ve beneficiado.

6.3.5. Gráficos resultantes de cada algoritmo de clasificación.

En esta sección se muestran gráficos que permiten visualizar el desempeño de cada algoritmo de entrenamiento de clasificadores con respectos a las características estudiadas, en el "eje y" de cada grafico tiene un rango de *accuracy* de 0.75 (75%) hasta 0.95 (95%).

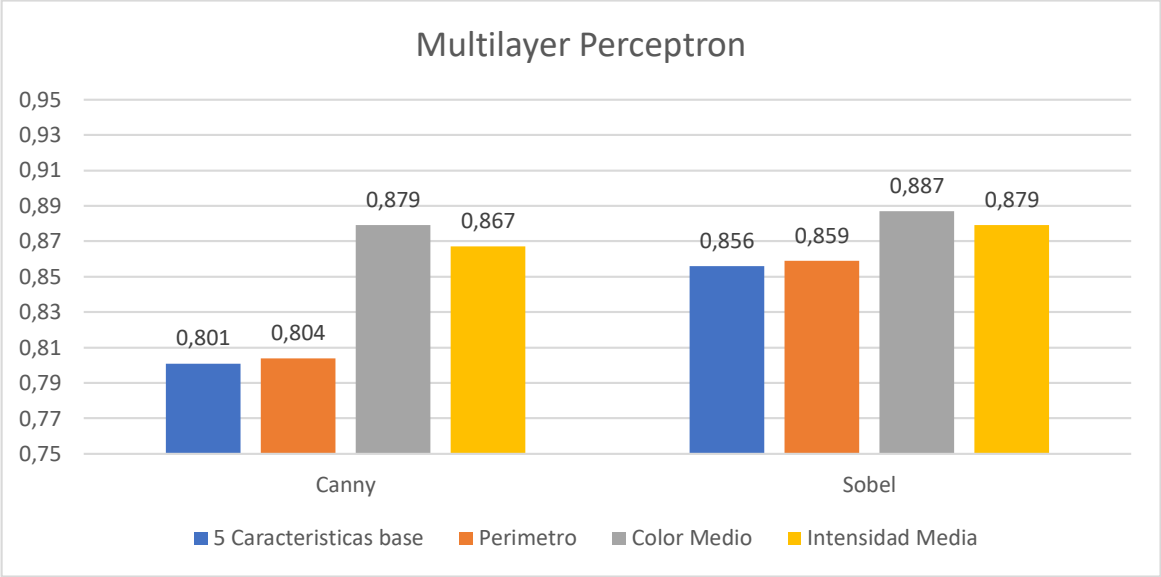


Figura 46: Desempeño (*accuracy*) de Multilayer Perceptron con respecto a características utilizadas y método de extracción. Fuente: (Elaboración propia)

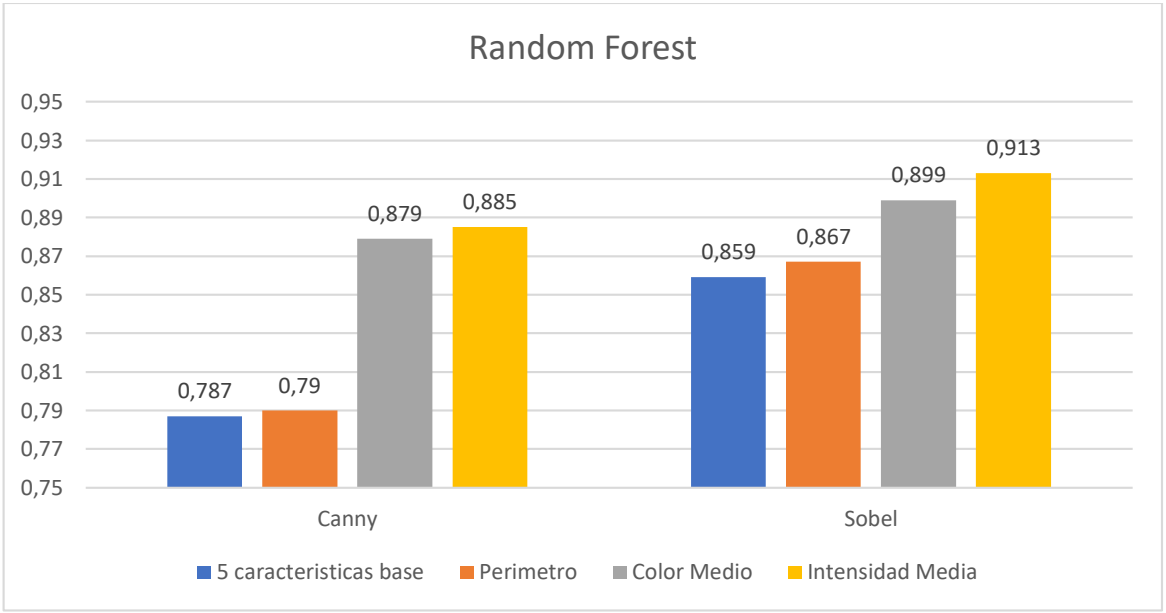


Figura 47: Desempeño (*accuracy*) de Random Forest con respecto a características utilizadas y método de extracción. Fuente: (Elaboración propia)

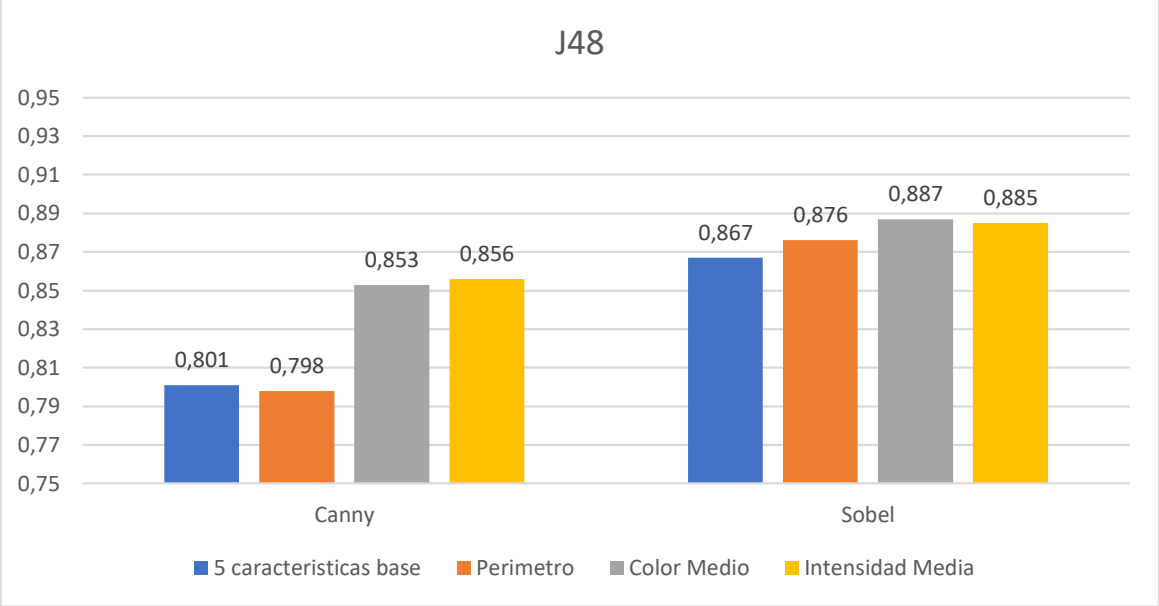


Figura 48: Desempeño (*accuracy*) de J48 con respecto a características utilizadas y método de extracción. Fuente: (Elaboración propia)

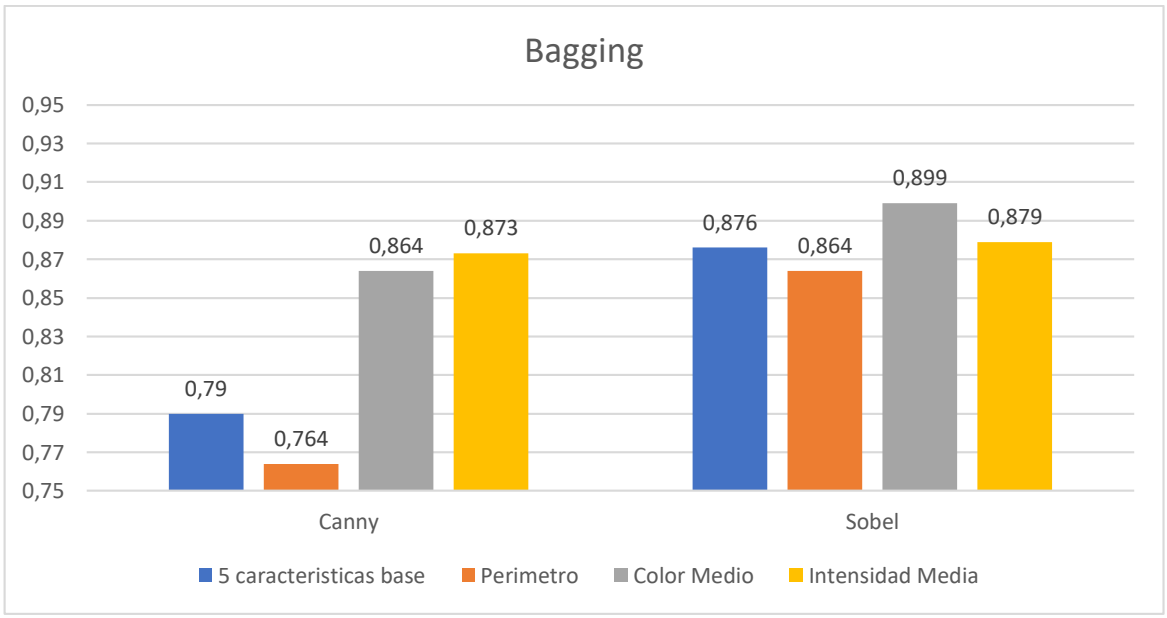


Figura 49: Desempeño (*accuracy*) de Bagging con respecto a características utilizadas y método de extracción. Fuente: (Elaboración propia)

6.4. Discusión

Gracias a los resultados obtenidos de los diferentes casos propuestos en la experimentación (ver sección 6.2) podemos inferir que el mejor método de detección de bordes, es Sobel, superando a Canny en los 4 casos propuestos. Cabe destacar que el desempeño de ambos métodos de detección de bordes se encuentra en un rango similar de *accuracy* con Random Forest (ver Figura 47), el cual es el mejor algoritmo de clasificación para estos 2 métodos, Sobel obtuvo un 91% de *accuracy* y Canny 88% (diferencia aproximada de 3%).

Las características con más significancia en ambos métodos, fueron: la excentricidad, eje menor, eje mayor, ratios de ejes y el área de (Su et al., 2011) y el color medio de (Guo et al., 2009). Sin embargo, las características menos influyentes son el perímetro de (Navarrete, 2019) y la intensidad media de (Guo et al., 2009).

También es posible identificar cuáles son las características que permiten un mejor rendimiento para cada algoritmo de aprendizaje, las que se detallan a continuación:

Perceptron multilayer

- **Canny:** Área, Eje mayor, Eje menor, Ratio de ejes, Excentricidad, Perímetro, Color medio.
- **Sobel:** Área, Eje mayor, Eje menor, Ratio de ejes, Excentricidad, Perímetro, Color medio.

Random Forest

- **Canny:** Área, Eje mayor, Eje menor, Ratio de ejes, Excentricidad, Color medio, intensidad media, Perímetro.
- **Sobel:** Área, Eje mayor, Eje menor, Ratio de ejes, Excentricidad, Perímetro, Color medio, Intensidad media.

J48

- **Canny:** Área, Eje mayor, Eje menor, Ratio de ejes, Excentricidad, Color medio, intensidad media.
- **Sobel:** Área, Eje mayor, Eje menor, Ratio de ejes, Excentricidad, Perímetro, Color medio.

Bagging

- **Canny:** Área, Eje mayor, Eje menor, Ratio de ejes, Excentricidad, Color medio, intensidad media.
- **Sobel:** Área, Eje mayor, Eje menor, Ratio de ejes, Excentricidad, Color medio.

7. Conclusiones

En este Proyecto de Título se estudió y midió el desempeño de diferentes técnicas de extracción de características, con el objetivo de clasificar anomalías en redes de alcantarillado.

Para lo anterior, en primer lugar, se realizó un estudio de material académico, para conocer los conceptos relacionados y seleccionar técnicas útiles para esta tarea, descritos en los Capítulos 2 y 3.

En segundo lugar, fue necesario recolectar las imágenes para experimentación. Se analizaron más de 100 videos de CCTV e ITV, otorgados por la empresa Inggepro, que facilitó el acceso a estos videos. Fue posible recolectar 200 imágenes, que, entre otras características propias de datos reales, incluyen una gran cantidad de ruido.

La extracción de características se basó en la forma, haciendo uso de métodos de detección de bordes. La implementación de estos métodos no estuvo exenta de complicaciones, ya que no se encuentran disponibles librearías de código, sino solo la descripción de algoritmos, en artículos científicos. Además, el gran nivel de ruido en las imágenes también añadió complejidad, pero el uso de operaciones morfológicas facilitó la limpieza de ruido, para lograr una extracción de características de forma eficaz.

La implementación de algoritmos de Machine Learning fue mecho menos compleja, ya que se hizo uso del API Weka, que provee de diferentes implementaciones de algoritmos de Machine Learning, así como de rutinas para la ejecución de experimentos y cálculo de métricas.

Del análisis de los resultados, es posible concluir que el mejor método de detección de bordes es Sobel, con el que se alcanzó un *accuracy* de 91%. En el caso del método Top-Hat, no fue posible entrenar clasificadores con buenos resultados, ya que este método es muy susceptible al ruido de las imágenes, y no se pudo obtener resultados relevantes.

La característica más significativa en base a la forma, fue el color medio, que permitió un aumento significativo en el a*ccuracy* obtenido con ambos métodos (Canny y Sobel).

Según lo visto, en este proyecto se cumple de manera satisfactoria los objetivos planteados al inicio del mismo. Mediante este informe, se entrega información que puede ser de utilidad para posteriores trabajos relacionado al tema.

Referencias

Aguirre Dobernack, N. (2013). *Implementación de un Sistema de Detección de Señales de Tráfico Mediante Visión Artificial Basado en FPGA*. 31. Retrieved from http://bibing.us.es/proyectos/abreproy/12112/fichero/Documento_por_capitulos%252F3_Capítulo_3.pdf

Alpaydin, E. (1965). Introducction to Machine Leraning. In *Introducction to Machine Leraning* (Vol. 111). https://doi.org/10.1192/bjp.111.479.1009-a

Bradski, G., & Kaehler, A. (2008). *Learning OpenCV, Computer Vision with OpenCV Library*.

Casasent, D. (1981). Pattern recognition: a review. *IEEE Spectrum*, *vol.18*, *no*(September), 28–33.

Cutler, A., Cutler, D. R., & Stevens, J. R. (2012). Random forests. *Ensemble Machine Learning: Methods and Applications*, 157–175. https://doi.org/10.1007/9781441993267_5

Ding, L., & Goshtasby, A. (2000). Computer Science and Engineering Department Wright State University. *Pattern Recognition*, *34*(34), 721–725.

Duda, R. O., Hart, P. E., & Stork, D. G. (2000). *Pattern Classification by Richard O. Duda, David G. Stork, Peter E.Hart .pdf* (p. 738). p. 738.

Exilio solutions. (2016). Métricas. Retrieved from https://blog.exsilio.com/all/accuracy-precision-recall-f1-score-interpretation-of-performance-measures/

F. Cappirini. (2009). F. Caparrini. Retrieved from http://www.cs.us.es/~fsancho/?e=106

Flórez Casillas, V. (2007). *Patología y errores de diseño más frecuentes en conducciones de agua*.

Gangadhar Shobha, S. R. (2018). Machine learning 2. In *Informs Analytics Body of Knowledge* (1st ed., Vol. 38). https://doi.org/10.1002/9781119505914.ch7

Gardner, M. W., & Dorling, S. R. (1998). Artificial neural networks (the multilayer perceptron) - a review of applications in the atmospheric sciences. *Atmospheric Environment*, *32*(14–15), 2627–2636. https://doi.org/10.1016/S1352-2310(97)00447-0

Gonzales, L. (2010). Ligdi-Gonzales-Machine Learning. Retrieved from https://ligdigonzalez.com/aprendizaje-supervisado-random-forest-classification/

Gonzalez, R. C. (2002). *Digital_Image_Processing_2ndEd.pdf*.

Guo, W., Soibelman, L., & Garrett, J. H. (2009). Visual Pattern Recognition Supporting Defect Reporting and Condition Assessment of Wastewater Collection Systems. *Journal of Computing in Civil Engineering*, *23*(3), 160–169. https://doi.org/10.1061/(asce)0887-

3801(2009)23:3(160)

Haio. (2012). Hayo. Retrieved from Cco website: https://www.mendeley.com/library/#

Haykin, S. (2014). Intriguing properties of neural networks, Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, Rob Fergus. In *arXiv preprint*. https://doi.org/978-0131471399

Huddleston, S. H., & Brown, G. G. (2018). Machine learning. In *Informs Analytics Body of Knowledge*. https://doi.org/10.1002/9781119505914.ch7

Inggepro Ltda. (2019). Retrieved April 17, 2019, from http://www.inggepro.cl/

JetBrains s.r.o. (2000). IntelliJ IDEA: The Java IDE for Professional Developers by JetBrains. Retrieved June 12, 2019, from https://www.jetbrains.com/idea/

Khooa, V. S., Dearnaley, D. P., Finnigan, D. J., Padhani, A., Tanner, S. F., & Leach, M. O. (1997). Computer aids for decision-making in diagnostic radiology—a literature review. *Radiotherapy and Oncology*, *42*(1), 1–15. https://doi.org/10.1016/0734-189X(88)90022-9

Liu, H. (2014). Introduction to Computer Vision on Mobile Devices. *Facial Detection and Recognition on Mobile Devices*, 1–9. https://doi.org/10.1016/b978-0-12-417045-2.00001-x

Navarrete, C. (2019). *Estudio y comparación de algoritmos para el reconocimiento visual y clasificación de hojas de árboles nativos de la región del Biobío*. Universidad del Bío-Bío.

Nilsson, N. J. (1997). *Introduction to Machine Learning*.

OpenCvTeam. (1999). OpenCV. Retrieved June 12, 2019, from https://opencv.org/

Oracle. (2011). JavaFx.

Oracle. (2014). Java Platform Standard Edition 8 Documentation. Retrieved June 12, 2019, from https://docs.oracle.com/javase/8/docs/

Parker, M. (2017). Introduction to Machine Learning. In *Digital Signal Processing 101*. https://doi.org/10.1016/b978-0-12-811453-7.00026-3

Poldrack, R. A., Nichols, T., Mumford, J., Poldrack, R. A., Nichols, T., & Mumford, J. (2011). Image processing basics. In *Handbook of Functional MRI Data Analysis*. https://doi.org/10.1017/cbo9780511895029.003

R. Fisher, S. Perkins, A. W. and E. W. (2003). Feature Detectors - Sobel Edge Detector. Retrieved July 7, 2019, from http://homepages.inf.ed.ac.uk/rbf/HIPR2/sobel.htm

Ramos, A. M., Gally, M., Szapiro, G., Itzcovich, T., Carabajal, M., & Levin, L. (2016). In vitro growth and cell wall degrading enzyme production by Argentinean isolates of

Macrophomina phaseolina, the causative agent of charcoal rot in corn. *Revista Argentina de Microbiología*, *48*(4), 267–273. https://doi.org/10.1016/j.ram.2016.06.002

Rizzo, P. (2010). Water and Wastewater Pipe Nondestructive Evaluation and Health Monitoring: A Review. *Advances in Civil Engineering*, *2010*, 1–13. https://doi.org/10.1155/2010/818597

Rosas Coronado, D. A. (2008). *Generación De Indicadores Para La Planta De Tratamiento De Aguas Servidas De Osorno* (Universidad Austral de Chile). Retrieved from http://cybertesis.uach.cl/tesis/uach/2008/bmfcir789g/doc/bmfcir789g.pdf

Sinha, S. K., & Fieguth, P. W. (2006). Automated detection of cracks in buried concrete pipe images. *Automation in Construction*, *15*(1), 58–72. https://doi.org/10.1016/j.autcon.2005.02.006

Skymind. (2019). Artificial Intelligence Wiki. Retrieved April 21, 2019, from https://skymind.ai/wiki/accuracy-precision-recall-f1

Smith, P., Reid, D. B., Environment, C., Palo, L., Alto, P., & Smith, P. L. (1979). Otsu1972_Threshold. *Pdfs.Semanticscholar.Org*, *20*(1), 62–66. https://doi.org/10.1109/TSMC.1979.4310076

Su, T. C., Yang, M. Der, Wu, T. C., & Lin, J. Y. (2011). Morphological segmentation based on edge detection for sewer pipe defects on CCTV images. *Expert Systems with Applications*, *38*(10), 13094–13114. https://doi.org/10.1016/j.eswa.2011.04.116

Taylor, W. K. (2010). Pattern Recognition. *Electronics and Power*, *24*(9), 679. https://doi.org/10.1049/ep.1978.0378

Tzutalin. (2017). LabelImg. Retrieved June 12, 2019, from https://tzutalin.github.io/labelImg/

U. OBerta de catalunya. (2010). j48. Retrieved from http://data-mining.business-intelligence.uoc.edu/home/j48-decision-tree?fbclid=IwAR36d-zOSVzyuwzLqOuRaiJjxiE2jn0Z4DZ6JD9sx4VzOnZ19iyLczNTREc

U. Waikato. (2011). Weka 3 - Data Mining with Open Source Machine Learning Software in Java. Retrieved June 12, 2019, from https://www.cs.waikato.ac.nz/ml/weka/

Varsity LLC. (2008). Elipse. Retrieved July 21, 2019, from https://www.varsitytutors.com/hotmath/hotmath_help/spanish/topics/ellipse

Watanabe, S., & Satosi. (1985). *Pattern recognition : human and mechanical*. Retrieved from https://dl.acm.org/citation.cfm?id=4394

Anexo

Estudio de factibilidad

En esta sección se mostrarán la factibilidad técnica, factibilidad operacional y factibilidad operativa.

Factibilidad Técnica

Software utilizado

IntelIJ 19.1.3 (JetBrains s.r.o, 2000)**:** Entorno de desarrollo integrado (IDE), para el lenguaje de programación Java, en este IDE se han desarrollado los prototipos de nuestro proyecto de título.

Requerimientos de hardware

- 1 GB de RAM mínimo.
- 2 GB de espacio libre + 1 GB para cache.
- Pantalla comuna resolución mínima de 1024x768.

Sistema operativo

- Windows.
- Linux.
- Mac.

Weka 3.8.3 (U. Waikato, 2011)**:** Es una Interfaz de programación de aplicación (API) que ayuda a crear modelos con diferentes métodos de aprendizajes, recibe un archivo en específico como entrada (.rff), con los datos de extracción de características, para el lenguaje de programación Java.

Requerimientos de hardware

- 512 mb de RAM mínimo.
- 300 mb libres.
- Pantalla comuna resolución mínima de 1024x768.

Sistema operativo

- Windows.
- Linux.
- Mac.

Wondershare Filmora9: Programa para estandarizar las imágenes en un mismo tamaño, fotogramas, se pueden editar videos en este programa.

Requerimientos de hardware

- Intel i3 o superior, con multinúcleos, 2Ghz o superior.
- RAM de al menos 4GB RAM físicos (8 GB requeridos para videos HD y 4k).
- Intel HD Graphics 5000 o posterior; NVIDIA GeForce GTX 700 o posterior; AMD Radeon R5 o posterior. 2 GB de vRAM (se requieren 4 GB para videos HD y 4K).
- Al menos 10GB de espacio libre en el disco duro para la instalación (Disco de estado sólido SSD recomendado para editar videos HD y 4K).

Sistema operativo

- Windows 7/8/10.

Bibliotecas utilizadas

OpenCv v.4.1.0 (OpenCvTeam, 1999)**:** Biblioteca de *Computer Vision*, en el cual se realizó la gran parte del procesamiento de imágenes (Imagen binaria, operaciones morfológicas), antes del proceso de extracción de características.

JavaFx v.11.0.2(Oracle, 2011)**:** Biblioteca otorgada por Oracle, gratis, en este proyecto se utilizó para la interfaz gráfica.

Esta biblioteca fueron las únicas utilizadas para este proyecto de título, para el preprocesamiento de imágenes, detección de bordes, imágenes binarias y procesos morfológicos, en el caso de OpenCv y Javafx para la interfaz gráfica.

Lenguaje utilizado

Java 8 (Oracle, 2014)**:** Lenguaje de programación en lo que se realizaron los prototipos de este proyecto de títulos.

Se puede descargar el jdk respectivo desde la página oficial de Oracle.

Hardware utilizado

Se cuenta con un equipo computacional propio para el desarrollo, ejecución y prueba del proyecto realizado.

Nicolás Alarcón Salazar

Equipo: Computador portátil Lenovo Legion Y520 y cuenta con las siguientes características:

- 8GB RAM.
- Disco Duro 1 Tera.
- SSD 128GB.
- Windows 10, Linux Pop! Os.
- Tarjeta Nvidia Geforce GTX 1060 3GB.
- Procesador Intel core i5-7300HQ con 2.5 GHz hasta 3.5 GHz.

Jerson Martínez Salgado

Equipo: Computador Escritorio y cuenta con las siguientes características:

- 16 GB RAM.
- Cuenta con 2 SSD, una 512GB y otra de 128 GB.
- Disco Duro de 1Tera.
- Windows 10, Linux Mint.
- Tarjeta Nvidia Geforce RTX 2070 8GB.
- Ryzen 5 2600x.

Conclusión factibilidad técnica

El software utilizado se puede descargar desde la página oficial de Intelij, posee 2 versiones uno de pago y uno gratuito, además, antes de iniciar este proyecto de título, ya se contaba con el hardware requerido y estos dispositivos cumplen con los requisitos de software que se exige. Se concluye que este proyecto de título y su desarrollo es factible técnicamente.

Factibilidad Económica

Recursos a ocupar

Recursos que serían necesario para el completo desarrollo del proyecto de título.

- Computador portátil Lenovo Legion Y520.
- Computador Escritorio.
- Horas hombre programador.

Obs: Cabe destacar que ya se tenía estos dispositivos antes de iniciar este proyecto de título.

Evaluación Económica

- Computador Portatil Lenovo Legion Y520 $599.990.

Costos en software y licencia

- **IntelIJ 19.1.3:** Puede ser descargado de forma gratuita por medio de la página oficial de JetBrain https://www.jetbrains.com/
- **Opencv:** Es una librería totalmente gratuita Open-Source, especializada en computer visión, la cual se puede obtener de su página oficial. https://opencv.org/releases/
- **Wondershare Filmora9:** Programa que se utilizó para estandarizar imágenes (mismo tamaño), posee una versión gratuita la cual es suficiente https://filmora.wondershare.net/filmora-video-editor.html
- **Weka:** Programa utilizado para trabajar con diferentes Machine Learning, como caja negra, este programa es totalmente gratis. Se puede descargar de su página oficial https://www.cs.waikato.ac.nz/ml/weka/

- **JavaFx v.11.0.2:** Biblioteca totalmente gratis utilizado para la interfaz gráfica se puede descargar desde la siguiente página: https://openjfx.io
- **ImgLabel:** Programa para ver dimensión de una imagen para después ser estandarizada, este programa es totalmente gratis, se puede descargar desde su repositorio oficial en github https://github.com/tzutalin/labelImg

Costos en horas hombres (HH)	
HH en desarrollo de la aplicación	160 (media jornada x día) x2
Total, de HH	360
Valor HH ingeniero civil informático	$6.800 x Hora
(Conexión Ingenieros, 2017) (calculados por 20 días hábiles por mes y 8 horas de trabajo al día)	
Costo Total de horas hombres en peso	$2.448.000
Horas Hombre	$2.448.000
Costos de hardware	$599.999
Costos en software y licencias	$0
MONTO TOTAl	$3.047.999

Tabla 9 costos factibilidad económica

Conclusión factibilidad económica

Se considera el caso hipotético de un desarrollo pagado, y se establece un costo de $3.047.999, correspondiendo en mayor medida al costo de esfuerzo (HH contratadas). Dado que este trabajo se realizó con fines académicos, el esfuerzo correspondiente es parte del Proyecto de Título, por lo cual este costo se descarta. Además, el hardware con el cual se trabajó ya se tenía en uso antes de empezar. Por tanto, considerado como trabajo académico, el proyecto es factible económicamente.

Factibilidad Operativa

En este proyecto de título pretende mejorar la extracción de características y las técnicas que se utilizan en base a la detección de bordes y demostrar que algoritmos de Machine Learning, logran un mayor factor de exactitud y precisión, para esto se demuestra a través de métricas, en caso de los algoritmos se deja las rutinas de códigos y las bibliotecas utilizadas, dicho todo lo anterior se concluye que es este proyecto es factible operativamente.

Configuración para los algoritmos

En esta sección se mostrarán las configuraciones utilizadas en las rutinas de códigos implementadas en este proyecto de título.

Configuración utilizada en Canny

Imagen en escala de grises:

$cvtColor(image, gray, COLOR\_BGR2GRAY)$

Imagen suavizada (elimina poco de ruido):

$GaussianBlur(gray, image, new\ Size(5,5), 0)$

Aplicación de apertura, para eliminar ruido exterior:

$getStructuringElement(MORPH\_ELLIPSE, new\ Size(2*4-1, 2*4-1))$

$morphologyEx(image, image, MORPH\_OPEN, kernel, new\ Point(), 2)$

Canny:

$Canny(image, image, 50, 250);$

dilatación:

$getStructuringElement(MORPH\_ELLIPSE, new\ Size(5,5))$

$dilate(image, image, dilateElement)$

Apertura cerrada para eliminar ruidos dentro del objeto:

$morphologyEx(image, image, MORPH\_CLOSE, dilateElement)$

Búsqueda de contornos:

$findContours(image, contorno, new\ MAT(), RETR\_EXTERNAL, CHAIN\_APPROX\_SIMPLE)$

Características de intensidad y color:

$Mat\ mask\ =\ new\ Mat(new\ Size(img.cols(), img.rows()), CvType.CV\_8UC1)$

$mask\_setTo(new\ Scalar(0.0))$

$drawContours(mask, contorno, -1, new\ Scalar(255), 2)$

$dilate(mask, image, dilateElement)$

$double\ [] intensity\ =\ Core.mean(gray, mask).val$

$double\ [] color\ =\ Core.mean(img, mask).val$

Características de la elipse:

$findContours(image, contornos, new\ Mat(), RETR\_EXTERNAL, CHAIN\_APPROX\_SIMPLE)$

$RotatedRect\ mimax\ =\ fitEllipse(new\ MatOfPoint2f(x.toArray()))$

$Rect\ g\ =\ mimax.boundingRect()$

Area:

$double\ area\ =\ contourArea(x)$

Configuración utilizada en Sobel

Imagen en escala de grises:

$cvtColor(image, gray, COLOR\_BGR2GRAY)$

Imagen suavizada (elimina poco de ruido):

$GaussianBlur(gray, image, new\ Size(5, 5), 0)$

Aplicación de apertura, para eliminar ruido exterior:

$getStructuringElement(MORPH\_ELLIPSE, new\ Size(2 * 4 - 1, 2 * 4 - 1))$

$morphologyEx(image, image, MORPH\_OPEN, kernel, new\ Point(), 2)$

Sobel:

$Mat\ grad\_x\ =\ new\ Mat()$

$Mat\ grad\_y\ =\ new\ Mat()$

$Mat\ abs\_grad\_x\ =\ new\ Mat()$

$Mat\ abs\_grad\_Y\ =\ new\ Mat()$

$Sobel(image, grad\_x, Cvtype.CV_16S, 1,0,3,1,0)$

$Sobel(image, grad\_y, Cvtype.CV\_16S, 0, 1, 3, 1, 0)$

$convertScaleAbs(grad\_x, abs\_grad\_x)$

$convertScaleAbs(grad\_y, abs\_grad\_y)$

$addWeighted(abs\_grad\_x, 0.5, abs\_grad\_y, 0.5, 1, result)$

Imagen Binaria:

$threshold(result, result, 100, 255, THRESH\_BINARY)$

Dilatación:

$dilate(result, result, kernel, new\ Point(), 2)$

Características de la elipse:

$findContours(image, contornos, new\ Mat(), RETR\_EXTERNAL, CHAIN\_APPROX\_SIMPLE)$

$RotatedRect\ mimax\ =\ fitEllipse(new\ MatOfPoint2f(x.toArray()))$

$Rect\ g\ =\ mimax.boundingRect()$

Area:

$double\ area\ =\ contourArea(x)$

Configuración utilizada en Top-Hat

Imagen en escala de grises:

$cvtColor(image, gray, COLOR\_BGR2GRAY)$

Imagen suavizada (elimina poco de ruido):

$GaussianBlur(gray, image, new\ Size(5,5), 0)$

Aplicación de gorro de copa Top-Hat:

$getStructuringElement(MORPH\_ELLIPSE, new\ Size(2*4-1, 2*4-1))$

$morphologyEx(image, image, MORPH\_TOPHAT, kernel)$

Imagen Binaria

$threshold(result, result, 100, 255, THRESH\_OTZU)$

Características de la elipse:

$findContours(image, contornos, new\ Mat(), RETR\_EXTERNAL, CHAIN\_APPROX\_SIMPLE)$

$RotatedRect\ mimax\ =\ fitEllipse(new\ MatOfPoint2f(x.toArray()))$

$Rect\ g\ =\ mimax.boundingRect()$

Se aplica valor absoluto para que el eje menor siempre sea el valor mínimo y los negativos se pasen al primer cuadrante con valores positivos

Area:

$double\ area\ =\ contourArea(x)$

Métricas Completas

En esta sección se detalla las Métricas completas obtenidas por el API de Weka

características base (excentricidad, área, ratios de ejes, excentricidad, eje mayor y menor)

Canny

Perceptron Multilayer

```
=== Confusion Matrix ===

   a   b   <-- classified as
 166   7 |   a = DaÃ±o
  62 113 |   b = Sin DaÃ±o
```

```
Correctly Classified Instances         279               80.1724 %
Incorrectly Classified Instances        69               19.8276 %
```

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	0,960	0,354	0,728	0,960	0,828	0,637	0,840	0,807	DaÃ±o
	0,646	0,040	0,942	0,646	0,766	0,637	0,840	0,864	Sin DaÃ±o
Weighted Avg.	0,802	0,196	0,835	0,802	0,797	0,637	0,840	0,836	

Random Forest

```
=== Confusion Matrix ===

   a   b   <-- classified as
 147  26 |   a = DaÃ±o
  48 127 |   b = Sin DaÃ±o
```

```
Correctly Classified Instances         274               78.7356 %
Incorrectly Classified Instances        74               21.2644 %
```

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	0,850	0,274	0,754	0,850	0,799	0,580	0,840	0,785	DaÃ±o
	0,726	0,150	0,830	0,726	0,774	0,580	0,840	0,863	Sin DaÃ±o
Weighted Avg.	0,787	0,212	0,792	0,787	0,787	0,580	0,840	0,824	

J48

```
=== Confusion Matrix ===

   a   b   <-- classified as
 168   5 |   a = DaÃ±o
  64 111 |   b = Sin DaÃ±o
```

Correctly Classified Instances	279	80.1724 %
Incorrectly Classified Instances	69	19.8276 %

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	0,971	0,366	0,724	0,971	0,830	0,642	0,766	0,673	DaÃ±o
	0,634	0,029	0,957	0,634	0,763	0,642	0,766	0,796	Sin DaÃ±o
Weighted Avg.	0,802	0,196	0,841	0,802	0,796	0,642	0,766	0,734	

Bagging

```
=== Confusion Matrix ===

   a   b   <-- classified as
 151  22 |   a = DaÃ±o
  51 124 |   b = Sin DaÃ±o
```

Correctly Classified Instances	275	79.023 %
Incorrectly Classified Instances	73	20.977 %

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	0,873	0,291	0,748	0,873	0,805	0,589	0,844	0,795	DaÃ±o
	0,709	0,127	0,849	0,709	0,773	0,589	0,844	0,863	Sin DaÃ±o
Weighted Avg.	0,790	0,209	0,799	0,790	0,789	0,589	0,844	0,829	

Sobel

Perceptron Multilayer

```
=== Confusion Matrix ===

   a   b   <-- classified as
 173  14 |   a = DaÃ±o
  36 125 |   b = Sin DaÃ±o
```

Correctly Classified Instances	298	85.6322 %
Incorrectly Classified Instances	50	14.3678 %

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	0,925	0,224	0,828	0,925	0,874	0,714	0,911	0,890	DaÃ±o
	0,776	0,075	0,899	0,776	0,833	0,714	0,911	0,902	Sin DaÃ±o
Weighted Avg.	0,856	0,155	0,861	0,856	0,855	0,714	0,911	0,896	

Random Forest

```
=== Confusion Matrix ===

   a   b   <-- classified as
 172  15 |   a = DaÃ±o
  34 127 |   b = Sin DaÃ±o
```

Correctly Classified Instances	299	85.9195 %
Incorrectly Classified Instances	49	14.0805 %

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	0,920	0,211	0,835	0,920	0,875	0,719	0,887	0,849	DaÃ±o
	0,789	0,080	0,894	0,789	0,838	0,719	0,887	0,885	Sin DaÃ±o
Weighted Avg.	0,859	0,151	0,862	0,859	0,858	0,719	0,887	0,866	

J48

```
=== Confusion Matrix ===

   a   b   <-- classified as
 180   7 |   a = DaÃ±o
  39 122 |   b = Sin DaÃ±o
```

```
Correctly Classified Instances         302               86.7816 %
Incorrectly Classified Instances        46               13.2184 %
```

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	0,963	0,242	0,822	0,963	0,887	0,744	0,835	0,774	DaÃ±o
	0,758	0,037	0,946	0,758	0,841	0,744	0,835	0,846	Sin DaÃ±o
Weighted Avg.	0,868	0,147	0,879	0,868	0,866	0,744	0,835	0,807	

Bagging

```
=== Confusion Matrix ===

   a   b   <-- classified as
 180   7 |   a = DaÃ±o
  36 125 |   b = Sin DaÃ±o
```

```
Correctly Classified Instances         305               87.6437 %
Incorrectly Classified Instances        43               12.3563 %
```

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	0,963	0,224	0,833	0,963	0,893	0,759	0,899	0,870	DaÃ±o
	0,776	0,037	0,947	0,776	0,853	0,759	0,899	0,894	Sin DaÃ±o
Weighted Avg.	0,876	0,137	0,886	0,876	0,875	0,759	0,899	0,881	

Perímetro

Canny

Multilayer Perceptron

```
=== Confusion Matrix ===

   a   b   <-- classified as
 166   7 |   a = DaÃ±o
  61 114 |   b = Sin DaÃ±o
```

```
Correctly Classified Instances         280               80.4598 %
Incorrectly Classified Instances        68               19.5402 %
```

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	0,960	0,349	0,731	0,960	0,830	0,641	0,849	0,812	DaÃ±o
	0,651	0,040	0,942	0,651	0,770	0,641	0,849	0,871	Sin DaÃ±o
Weighted Avg.	0,805	0,194	0,837	0,805	0,800	0,641	0,849	0,842	

Random Forest

```
=== Confusion Matrix ===

   a   b   <-- classified as
 150  23 |   a = DaÃ±o
  50 125 |   b = Sin DaÃ±o
```

```
Correctly Classified Instances         275               79.023  %
Incorrectly Classified Instances        73               20.977  %
```

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	0,867	0,286	0,750	0,867	0,804	0,588	0,840	0,771	DaÃ±o
	0,714	0,133	0,845	0,714	0,774	0,588	0,840	0,865	Sin DaÃ±o
Weighted Avg.	0,790	0,209	0,798	0,790	0,789	0,588	0,840	0,818	

J48

```
=== Confusion Matrix ===

   a   b   <-- classified as
 167   6 |   a = DaÃ±o
  64 111 |   b = Sin DaÃ±o
```

```
Correctly Classified Instances         278               79.8851 %
Incorrectly Classified Instances        70               20.1149 %
```

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	0,965	0,366	0,723	0,965	0,827	0,635	0,775	0,681	DaÃ±o
	0,634	0,035	0,949	0,634	0,760	0,635	0,775	0,814	Sin DaÃ±o
Weighted Avg.	0,799	0,199	0,836	0,799	0,793	0,635	0,775	0,748	

Bagging

```
=== Confusion Matrix ===

   a   b   <-- classified as
 150  23 |   a = DaÃ±o
  59 116 |   b = Sin DaÃ±o
```

```
Correctly Classified Instances         266               76.4368 %
Incorrectly Classified Instances        82               23.5632 %
```

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	0,867	0,337	0,718	0,867	0,785	0,541	0,826	0,775	DaÃ±o
	0,663	0,133	0,835	0,663	0,739	0,541	0,826	0,854	Sin DaÃ±o
Weighted Avg.	0,764	0,234	0,776	0,764	0,762	0,541	0,826	0,815	

Sobel

Multilayer Perceptron

```
=== Confusion Matrix ===

   a   b   <-- classified as
 176  11 |   a = DaÃ±o
  38 123 |   b = Sin DaÃ±o
```

Correctly Classified Instances	299	85.9195 %
Incorrectly Classified Instances	49	14.0805 %

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	0,941	0,236	0,822	0,941	0,878	0,723	0,914	0,910	DaÃ±o
	0,764	0,059	0,918	0,764	0,834	0,723	0,914	0,901	Sin DaÃ±o
Weighted Avg.	0,859	0,154	0,867	0,859	0,857	0,723	0,914	0,906	

Random Forest

```
=== Confusion Matrix ===

   a   b   <-- classified as
 174  13 |   a = DaÃ±o
  33 128 |   b = Sin DaÃ±o
```

Correctly Classified Instances	302	86.7816 %
Incorrectly Classified Instances	46	13.2184 %

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	0,930	0,205	0,841	0,930	0,883	0,737	0,890	0,839	DaÃ±o
	0,795	0,070	0,908	0,795	0,848	0,737	0,890	0,886	Sin DaÃ±o
Weighted Avg.	0,868	0,142	0,872	0,868	0,867	0,737	0,890	0,860	

J48

```
=== Confusion Matrix ===

   a   b   <-- classified as
 182   5 |   a = DaÃ±o
  38 123 |   b = Sin DaÃ±o
```

Correctly Classified Instances	305	87.6437 %
Incorrectly Classified Instances	43	12.3563 %

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	0,973	0,236	0,827	0,973	0,894	0,762	0,834	0,776	DaÃ±o
	0,764	0,027	0,961	0,764	0,851	0,762	0,834	0,838	Sin DaÃ±o
Weighted Avg.	0,876	0,139	0,889	0,876	0,874	0,762	0,834	0,805	

Bagging

```
=== Confusion Matrix ===

   a   b   <-- classified as
 175  12 |   a = DaÃ±o
  35 126 |   b = Sin DaÃ±o
```

Correctly Classified Instances	301	86.4943 %
Incorrectly Classified Instances	47	13.5057 %

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	0,936	0,217	0,833	0,936	0,882	0,732	0,896	0,869	DaÃ±o
	0,783	0,064	0,913	0,783	0,843	0,732	0,896	0,892	Sin DaÃ±o
Weighted Avg.	0,865	0,147	0,870	0,865	0,864	0,732	0,896	0,880	

Color Medio

Canny

Multilayer Perceptron

```
=== Confusion Matrix ===

   a   b   <-- classified as
 152  21 |   a = DaÃ±o
  21 154 |   b = Sin DaÃ±o

Correctly Classified Instances         306               87.931  %
Incorrectly Classified Instances        42               12.069  %


                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0,879    0,120    0,879      0,879    0,879      0,759    0,910     0,894     DaÃ±o
                 0,880    0,121    0,880      0,880    0,880      0,759    0,910     0,887     Sin DaÃ±o
Weighted Avg.    0,879    0,121    0,879      0,879    0,879      0,759    0,910     0,890
```

Random Forest

```
=== Confusion Matrix ===

   a   b   <-- classified as
 162  11 |   a = DaÃ±o
  31 144 |   b = Sin DaÃ±o


Correctly Classified Instances         306               87.931  %
Incorrectly Classified Instances        42               12.069  %


                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0,936    0,177    0,839      0,936    0,885      0,764    0,933     0,915     DaÃ±o
                 0,823    0,064    0,929      0,823    0,873      0,764    0,933     0,924     Sin DaÃ±o
Weighted Avg.    0,879    0,120    0,884      0,879    0,879      0,764    0,933     0,920
```

J48

```
=== Confusion Matrix ===

   a   b   <-- classified as
 148  25 |   a = DaÃ±o
  26 149 |   b = Sin DaÃ±o
```

Correctly Classified Instances	297	85.3448 %
Incorrectly Classified Instances	51	14.6552 %

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	0,855	0,149	0,851	0,855	0,853	0,707	0,905	0,904	DaÃ±o
	0,851	0,145	0,856	0,851	0,854	0,707	0,905	0,843	Sin DaÃ±o
Weighted Avg.	0,853	0,147	0,853	0,853	0,853	0,707	0,905	0,873	

Bagging

```
=== Confusion Matrix ===

   a   b   <-- classified as
 154  19 |   a = DaÃ±o
  28 147 |   b = Sin DaÃ±o
```

Correctly Classified Instances	301	86.4943 %
Incorrectly Classified Instances	47	13.5057 %

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	0,890	0,160	0,846	0,890	0,868	0,731	0,922	0,904	DaÃ±o
	0,840	0,110	0,886	0,840	0,862	0,731	0,922	0,909	Sin DaÃ±o
Weighted Avg.	0,865	0,135	0,866	0,865	0,865	0,731	0,922	0,907	

Sobel

Multilayer Perceptron

```
=== Confusion Matrix ===

   a   b   <-- classified as
 169  18 |   a = DaÃ±o
  21 140 |   b = Sin DaÃ±o
```

Correctly Classified Instances	309	88.7931 %
Incorrectly Classified Instances	39	11.2069 %

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	0,904	0,130	0,889	0,904	0,897	0,774	0,940	0,938	DaÃ±o
	0,870	0,096	0,886	0,870	0,878	0,774	0,940	0,918	Sin DaÃ±o
Weighted Avg.	0,888	0,115	0,888	0,888	0,888	0,774	0,940	0,929	

Random Forest

```
=== Confusion Matrix ===

   a   b   <-- classified as
 178   9 |   a = DaÃ±o
  26 135 |   b = Sin DaÃ±o
```

Correctly Classified Instances	313	89.9425 %
Incorrectly Classified Instances	35	10.0575 %

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	0,952	0,161	0,873	0,952	0,910	0,800	0,926	0,900	DaÃ±o
	0,839	0,048	0,938	0,839	0,885	0,800	0,926	0,904	Sin DaÃ±o
Weighted Avg.	0,899	0,109	0,903	0,899	0,899	0,800	0,926	0,902	

J48

```
=== Confusion Matrix ===

   a   b   <-- classified as
 171  16 |   a = DaÃ±o
  23 138 |   b = Sin DaÃ±o
```

Correctly Classified Instances	309	88.7931 %
Incorrectly Classified Instances	39	11.2069 %

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	0,914	0,143	0,881	0,914	0,898	0,775	0,909	0,905	DaÃ±o
	0,857	0,086	0,896	0,857	0,876	0,775	0,909	0,841	Sin DaÃ±o
Weighted Avg.	0,888	0,116	0,888	0,888	0,888	0,775	0,909	0,875	

Bagging

```
=== Confusion Matrix ===

   a   b   <-- classified as
 179   8 |   a = DaÃ±o
  27 134 |   b = Sin DaÃ±o
```

Correctly Classified Instances	313	89.9425 %
Incorrectly Classified Instances	35	10.0575 %

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	0,957	0,168	0,869	0,957	0,911	0,801	0,912	0,867	DaÃ±o
	0,832	0,043	0,944	0,832	0,884	0,801	0,912	0,901	Sin DaÃ±o
Weighted Avg.	0,899	0,110	0,904	0,899	0,899	0,801	0,912	0,883	

Intensidad Media

Canny

Multilayer Perceptron

```
=== Confusion Matrix ===

   a   b   <-- classified as
 153  20 |   a = DaÃ±o
  26 149 |   b = Sin DaÃ±o
```

Correctly Classified Instances	302	86.7816 %
Incorrectly Classified Instances	46	13.2184 %

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	0,884	0,149	0,855	0,884	0,869	0,736	0,913	0,906	DaÃ±o
	0,851	0,116	0,882	0,851	0,866	0,736	0,913	0,890	Sin DaÃ±o
Weighted Avg.	0,868	0,132	0,868	0,868	0,868	0,736	0,913	0,898	

Random Forest

```
=== Confusion Matrix ===

   a   b   <-- classified as
 163  10 |   a = DaÃ±o
  30 145 |   b = Sin DaÃ±o
```

Correctly Classified Instances	308	88.5057 %
Incorrectly Classified Instances	40	11.4943 %

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	0,942	0,171	0,845	0,942	0,891	0,775	0,935	0,908	DaÃ±o
	0,829	0,058	0,935	0,829	0,879	0,775	0,935	0,929	Sin DaÃ±o
Weighted Avg.	0,885	0,114	0,890	0,885	0,885	0,775	0,935	0,919	

J48

```
=== Confusion Matrix ===

   a   b   <-- classified as
 144  29 |   a = DaÃ±o
  21 154 |   b = Sin DaÃ±o
```

Correctly Classified Instances	298	85.6322 %
Incorrectly Classified Instances	50	14.3678 %

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	0,832	0,120	0,873	0,832	0,852	0,713	0,876	0,880	DaÃ±o
	0,880	0,168	0,842	0,880	0,860	0,713	0,876	0,797	Sin DaÃ±o
Weighted Avg.	0,856	0,144	0,857	0,856	0,856	0,713	0,876	0,838	

Bagging

```
=== Confusion Matrix ===

   a   b   <-- classified as
 160  13 |   a = DaÃ±o
  31 144 |   b = Sin DaÃ±o
```

Correctly Classified Instances	304	87.3563 %
Incorrectly Classified Instances	44	12.6437 %

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	0,925	0,177	0,838	0,925	0,879	0,751	0,932	0,904	DaÃ±o
	0,823	0,075	0,917	0,823	0,867	0,751	0,932	0,927	Sin DaÃ±o
Weighted Avg.	0,874	0,126	0,878	0,874	0,873	0,751	0,932	0,916	

Sobel

Multilayer Perceptron

```
=== Confusion Matrix ===

   a   b   <-- classified as
 165  22 |   a = DaÃ±o
  20 141 |   b = Sin DaÃ±o
```

```
Correctly Classified Instances         306               87.931  %
Incorrectly Classified Instances        42               12.069  %
```

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	0,882	0,124	0,892	0,882	0,887	0,758	0,935	0,938	DaÃ±o
	0,876	0,118	0,865	0,876	0,870	0,758	0,935	0,904	Sin DaÃ±o
Weighted Avg.	0,879	0,121	0,879	0,879	0,879	0,758	0,935	0,922	

Random Forest

```
=== Confusion Matrix ===

   a   b   <-- classified as
 182   5 |   a = DaÃ±o
  25 136 |   b = Sin DaÃ±o
```

```
Correctly Classified Instances         318               91.3793 %
Incorrectly Classified Instances        30                8.6207 %
```

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	0,973	0,155	0,879	0,973	0,924	0,831	0,927	0,917	DaÃ±o
	0,845	0,027	0,965	0,845	0,901	0,831	0,927	0,900	Sin DaÃ±o
Weighted Avg.	0,914	0,096	0,919	0,914	0,913	0,831	0,927	0,909	

J48

```
=== Confusion Matrix ===

   a   b   <-- classified as
 167  20 |   a = DaÃ±o
  20 141 |   b = Sin DaÃ±o
```

```
Correctly Classified Instances         308               88.5057 %
Incorrectly Classified Instances        40               11.4943 %
```

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	0,893	0,124	0,893	0,893	0,893	0,769	0,890	0,881	DaÃ±o
	0,876	0,107	0,876	0,876	0,876	0,769	0,890	0,819	Sin DaÃ±o
Weighted Avg.	0,885	0,116	0,885	0,885	0,885	0,769	0,890	0,852	

Bagging

```
=== Confusion Matrix ===

   a   b   <-- classified as
 176  11 |   a = DaÃ±o
  31 130 |   b = Sin DaÃ±o
```

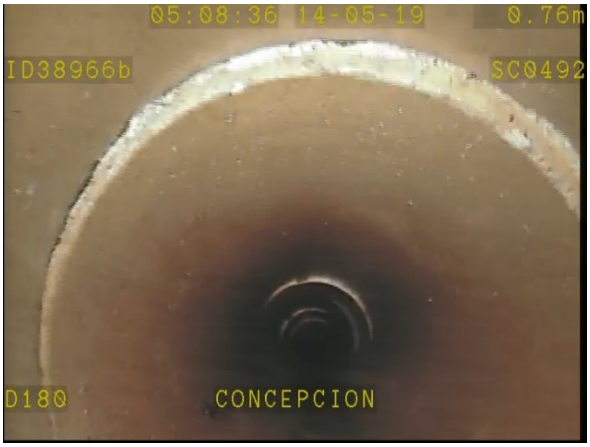
```
Correctly Classified Instances         306               87.931  %
Incorrectly Classified Instances        42               12.069  %
```

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	0,941	0,193	0,850	0,941	0,893	0,760	0,926	0,931	DaÃ±o
	0,807	0,059	0,922	0,807	0,861	0,760	0,926	0,909	Sin DaÃ±o
Weighted Avg.	0,879	0,131	0,883	0,879	0,878	0,760	0,926	0,921	

Muestras de imágenes utilizadas en el dataset

Imágenes con anomalía.

Se mostrarán imágenes con defectos de tipo grieta, desacople y roturas en las tuberías



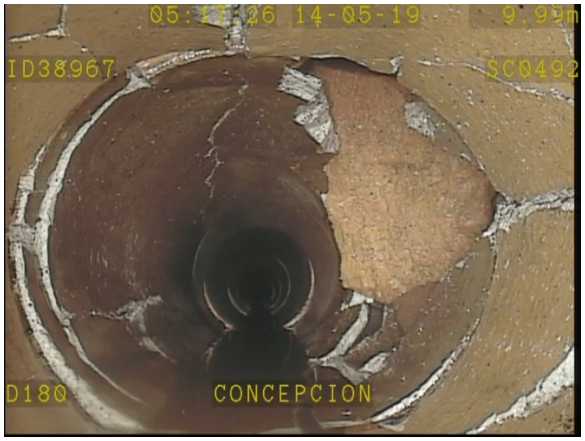
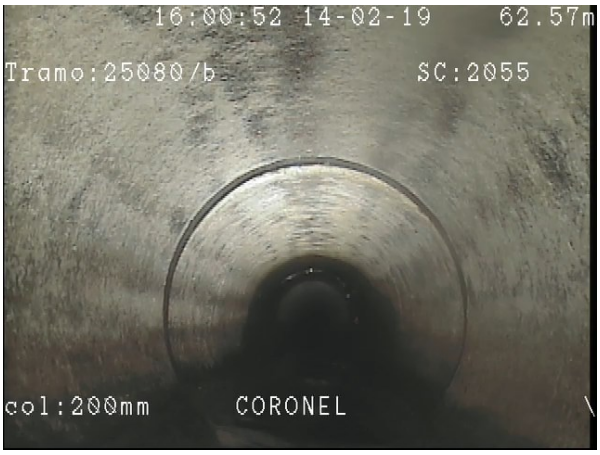
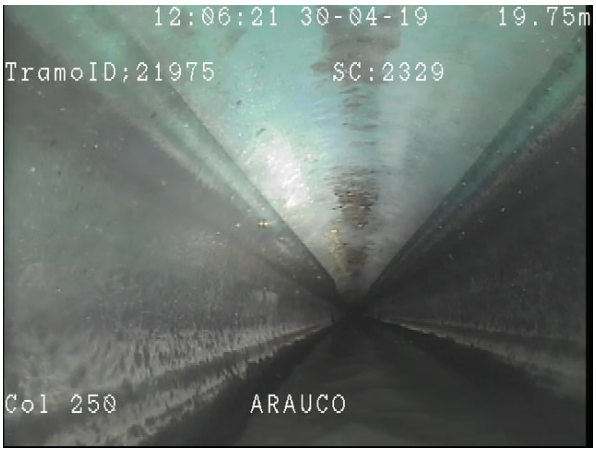


Figura 50: Imágenes con roturas, fisuras y desacople Fuente: (Inggepro Ltda.,2019)

Imágenes en buen estado.

Se presentarán imágenes de tuberías en buen estado que se utilizaron en la elaboración del dataset.





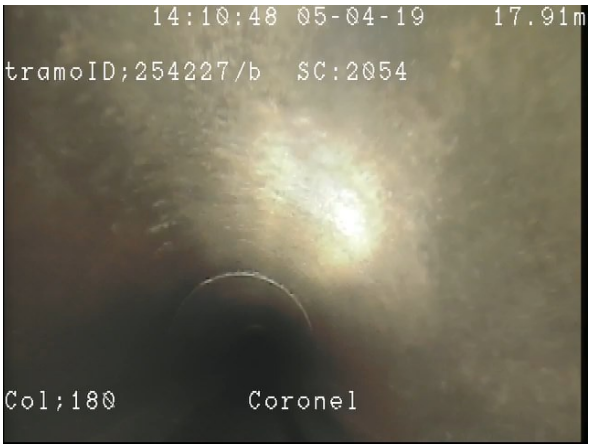


Figura 51: Imágenes en buen estado de tuberías Fuente: (Inggepro Ltda., 2019)

Muestras del preprocesamiento de las técnicas utilizadas.

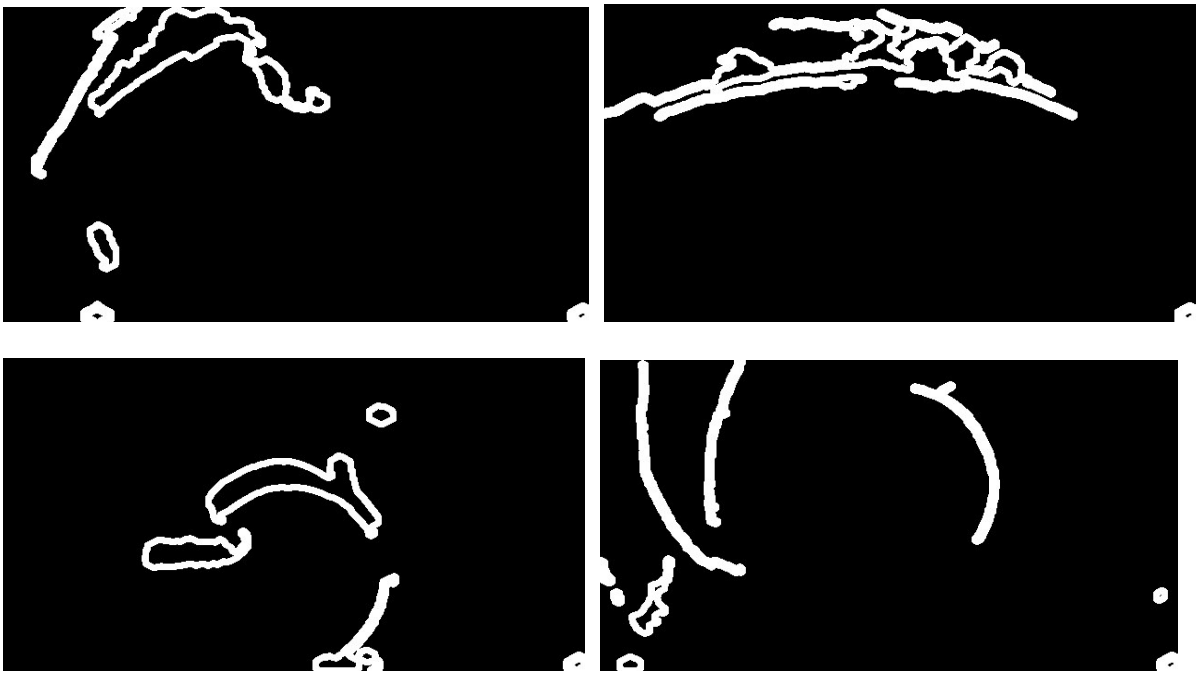


Figura 52: Procesamiento de las técnicas utilizadas. Fuente: (Elaboración Propia)

Muestras del resultado de las técnicas utilizadas.

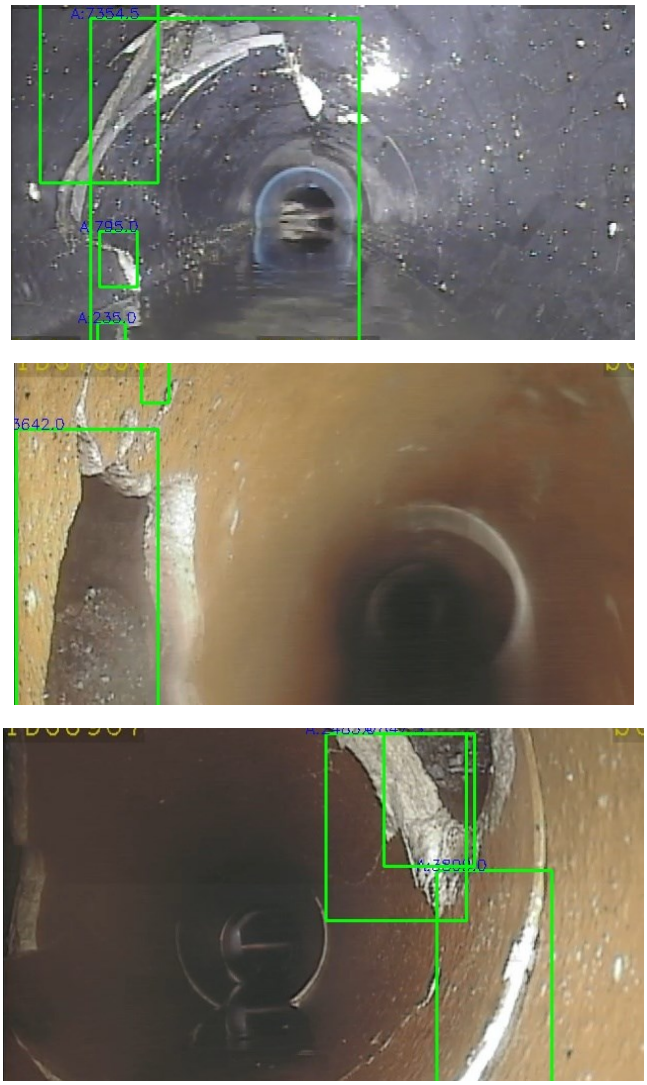


Figura 53: Imágenes resultantes al detectar el roi : (Elaboración Propia)