UNIVERSIDAD TÉCNICA DE AMBATO

FACULTAD DE INGENIERÍA EN SISTEMAS, ELECTRÓNICA E INDUSTRIAL

MAESTRÍA EN MATEMÁTICA APLICADA

Tema: MODELO MATEMÁTICO BASADO EN TÉCNICAS DE DETECCIÓN DE BORDES Y PROPAGACIÓN DE TEXTURAS PARA RESTAURAR IMÁGENES FOTOGRÁFICAS DOCUMENTALES RGB.

Trabajo de Investigación, previo a la obtención del Grado Académico de Magíster en Matemática Aplicada

Autor: Ing. Carlos Rodrigo Jordán Bolaños

Director: Ing. Carlos Gordón PhD.

Ambato-Ecuador

2021

APROBACIÓN DEL TRABAJO DE TITULACIÓN

A la Unidad Académica de Titulación de la Facultad de Ingeniería en Sistemas Electrónica e Industrial.

El Tribunal receptor del Trabajo de Investigación, presidido por la Ingeniera Elsa Pilar Urrutia Urrutia Magister, e integrado por los señores: Ing. Fabián Rodrigo Salazar Escobar, Dr. y el Ing. Víctor Santiago Manzano Villafuerte, Mg. designados por la Unidad Académica de Titulación de la Universidad Técnica de Ambato, para receptar el Trabajo de Investigación con el tema: MODELO MATEMÁTICO BASADO EN TÉCNICAS DE DETECCIÓN DE BORDES Y PROPAGACIÓN DE TEXTURAS PARA RESTAURAR IMÁGENES FOTOGRÁFICAS DOCUMENTALES RGB, elaborado y presentado por el señor Ingeniero Carlos Rodrigo Jordán Bolaños, para optar por el Grado Académico de Magíster en Matemática Aplicada; una vez escuchada la defensa oral del Trabajo de Investigación; el Tribunal aprueba y remite el trabajo para uso y custodia en las bibliotecas de la UTA.

Ing. Elsa Pilar Urrutia Urrutia Mg.

Presidente y Miembro del Tribunal de Defensa

Ing. Fabián Rodrigo Salazar Escobar, Dr.

Miembro del Tribunal de Defensa

Ing. Víctor Santiago Manzano Villafuerte, Mg.

Miembro del Tribunal de Defensa

AUTORÍA DEL TRABAJO DE INVESTIGACIÓN

La responsabilidad de las opiniones, comentarios y críticas emitidas en el Trabajo de Investigación, presentado con el tema: "MODELO MATEMÁTICO BASADO EN TÉCNICAS DE DETECCIÓN DE BORDES Y PROPAGACIÓN DE TEXTURAS PARA RESTAURAR IMÁGENES FOTOGRÁFICAS DOCUMENTALES RGB", le corresponde exclusivamente a: Ingeniero Carlos Rodrigo Jordán Bolaños, Autor bajo la Dirección del Ing. Carlos Gordón PhD. Director del Trabajo de Investigación; y el patrimonio intelectual a la Universidad Técnica de Ambato.

Ing. Carlos Rodrigo Jordán Bolaños

C.C. 1803428141

AUTOR

Ing. Carlos Gordón PhD

C.C. 1803405495

DIRECTOR

DERECHOS DE AUTOR

Autorizo a la Universidad Técnica de Ambato, para que el Trabajo de Investigación, sirva como un documento disponible para su lectura, consulta y procesos de investigación, según las normas de la Institución.

Cedo los Derechos de mi trabajo, con fines de difusión pública, además apruebo la reproducción de este, dentro de las regulaciones de la Universidad.

Ing. Carlos Rodrigo Jordán Bolaños

C.C. 1803428141

ÍNDICE GENERAL DEL CONTENIDO

CONTENIDO **Pág.**

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

AGRADECIMIENTO

Quiero en primer lugar agradecer a Dios por darme la fuerza y por ser ese padre que siempre me ha cuidado y que nunca me dejó desmayar.

A mis padres, que son el pilar de mi éxito, sin el apoyo de ellos jamás hubiese llegado al lugar donde hoy me encuentro, por confiar y creer en nuestras expectativas, por los consejos, valores y principios que nos han inculcado.

A toda mi familia en general, en especial a mi esposa Gabriela por su cariño y apoyo, quien siempre ha estado junto a mí pese a las adversidades e inconvenientes que se han presentado. Agradezco sus enseñanzas de perseverancia y por cada paso que hemos dado juntos.

A mis hijas Nebraska y Briana por ser el motor que me permite salir adelante y cumplir mis sueños.

Al Dr. Carlos Gordon por su tutoría, consejos y aportes que han sido necesarios para que este trabajo de investigación sea un aporte para la sociedad.

Carlos Rodrigo Jordán Bolaños

DEDICATORIA

El presente trabajo investigativo quiero dedicarle a Dios, a mis padres, a mi esposa e hijas, porque gracias a ellos y a su apoyo incondicional se pudo cumplir y llegar a la meta.

También es importante dedicar este trabajo a mis hermanos y sobrinos por siempre darme ánimo y el apoyo moral a lo largo de esta etapa de mi vida.

A mi Director de Tesis quien ha permitido que este trabajo se realice con éxito, gracias por abrirnos las puertas y compartir sus conocimientos.

Carlos Rodrigo Jordán Bolaños

UNIVERSIDAD TÉCNICA DE AMBATO

FACULTAD DE INGENIERÍA EN SISTEMAS, ELECTRÓNICA E INDUSTRIAL

MAESTRÍA EN MATEMÁTICA APLICADA

TEMA: MODELO MATEMÁTICO BASADO EN TÉCNICAS DE DETECCIÓN DE BORDES Y PROPAGACIÓN DE TEXTURAS PARA RESTAURAR IMÁGENES FOTOGRÁFICAS DOCUMENTALES RGB.

AUTOR: Ingeniero Carlos Rodrigo Jordán Bolaños.

DIRECTOR: Ingeniero Carlos Gordón PhD.

LÍNEA DE INVESTIGACIÓN:

- Modelamiento matemático.

FECHA: Doce de enero de 2021

RESUMEN EJECUTIVO

La restauración de imágenes fotográficas documentales RGB, mediante un modelo matemático basado en dos técnicas fundamentales propagación de textura y detección de bordes buscar dar prioridad equilibrada a las dos técnicas, es decir, ninguna debe primar sobre la otra, utilizando la notación para restauración de imágenes: I = Imagen, ϕ = Región fuente. Es la región conocida de la imagen, Ω = Región objetivo. Es la región a restaurar y $\delta\Omega$ = Frontera. Las imágenes fotográficas documentales RGB (color) se centran en las personas y en los grupos sociales, para mostrar aspectos de su vida cotidiana y nacen con la intención de plasmar la realidad en una imagen fija.

La detección de bordes es una técnica que facilita la identificación de los objetos de la imagen y por lo tanto su reconocimiento. Si el propósito de los algoritmos de detección de bordes es obtener como resultado una imagen donde se resalten los pixeles de aquellos puntos de la imagen original en donde se presentan cambios bruscos de intensidad. A este

respecto vale la pena resaltar entre otras, las técnicas relacionadas con cruces por cero (“Zero crossing”) y de curvas de nivel (“Level set”).

Por otra parte, la propagación de texturas es una técnica que consiste en construir a partir de una pequeña imagen de muestra, una gran imagen que conserve la estructura de la muestra. Parte importante del modelo que se propone en este trabajo, consiste en el llenado de la imagen por pixeles, en primer lugar, se debe contar con que el área a restaurar está bien definida para que las texturas vecinas mantengan el balance y armonía en el resultado final.

El modelo matemático propuesto está diseñado para obtener de la imagen original, primero el gradiente de la imagen seguido del mapa de extracción que no es más que la silueta del objeto a extraer; luego se crea el mapa energético de la imagen para proceder con la eliminación de costuras. Costura tras costura serán eliminadas hasta crear una imagen resultante similar en tamaño y características de la original. Finalmente se hará la restauración considerando los bordes y texturas de la imagen original, con ello se valorará si la imagen fotográfica documental restaurada es visualmente satisfactoria. El criterio de similitud para hacer la comparación entre la imagen original y resultante tiene que ser de tipo matemático, es decir, un algoritmo comparará pixel por pixel las dos imágenes mostrando en pantalla cuan eficiente es el algoritmo.

Bajo el contexto anterior se dice que la restauración de imágenes a través de modelos matemáticos busca perfeccionar la modificación de manera inteligente, es decir, que al finalizar el proceso de restauración la imagen sea visualmente agradable al observador en donde la sustracción de elementos o la modificación del tamaño no dañe la armonía de colores, formas y texturas.

Descriptores: Seam Carving, propagación de texturas, detección de bordes, restauración de imágenes, imágenes documentales, RGB, modelo matemático, gradiente, función de energía, eficiencia.

UNIVERSIDAD TÉCNICA DE AMBATO

FACULTAD DE INGENIERÍA EN SISTEMAS, ELECTRÓNICA E INDUSTRIAL

MAESTRÍA EN MATEMÁTICA APLICADA

TEMA: MODELO MATEMÁTICO BASADO EN TÉCNICAS DE DETECCIÓN DE BORDES Y PROPAGACIÓN DE TEXTURAS PARA RESTAURAR IMÁGENES FOTOGRÁFICAS DOCUMENTALES RGB.

AUTOR: Ingeniero Carlos Rodrigo Jordán Bolaños.

DIRECTOR: Dr. Carlos Diego Gordón Gallegos.

LÍNEA DE INVESTIGACIÓN:

- Modelamiento matemático.

FECHA: Doce de enero de 2021

EXECUTIVE SUMMARY

The restoration of RGB documentary photographic images, using a mathematical model based on two fundamental techniques texture propagation and edge detection seek to give balanced priority to the two techniques, that is, neither should prevail over the other, using the notation for image restoration: I = Image, ϕ = source region. It is the known region of the image, Ω = target region. It is the region to restore and $\delta\Omega$ = border. RGB (color) documentary photographic images focus on people and social groups, to show aspects of their daily lives and are born with the intention of capturing reality in a still image.

Edge detection is a technique that facilitates the identification of image objects and therefore their recognition. If the purpose of edge detection algorithms is to obtain as a result an image where the pixels of those points of the original image where sudden changes in intensity are presented are highlighted. In this respect it is worth highlighting,

among others, the techniques related to zero crossing ("Zero crossing") and level set ("level set").

On the other hand, texture propagation is a technique that consists of building from a small sample image, a large image that preserves the structure of the sample. An important part of the model that is proposed in this work, consists in filling the image by pixels, first of all, you must count on the area to be restored is well defined so that the neighboring textures maintain balance and harmony in the final result.

The proposed mathematical model is designed to obtain from the original image, first the gradient of the image followed by the extraction map which is nothing more than the silhouette of the object to be extracted; then the energy map of the image is created to proceed with the removal of seams. Seam after seam will be removed until you create a resulting image similar in size and characteristics to the original. Finally the restoration will be done considering the edges and textures of the original image, with this it will be evaluated if the restored documentary photographic image is visually satisfactory. The similarity criterion to make the comparison between the original and resulting image has to be of mathematical type, that is, an algorithm will compare pixel by pixel the two images showing on screen how efficient the algorithm is.

Under the above context it is said that the restoration of images through mathematical modelling seeks to refine the amendment in an intelligent way, that is to say, that at the end of the restore process, the image is visually pleasing to the observer, where the subtraction of elements or the size modification does not harm the harmony of colors, shapes and textures.

Descriptors: Seam Carving, texture propagation, edge detection, image restoration, documentary images, RGB, mathematical model, gradient, energy function, efficiency.

CAPÍTULO I

EL PROBLEMA DE INVESTIGACIÓN

1.1. Introducción

Las imágenes fotográficas documentales RGB (color) se centran en las personas y en los grupos sociales, para mostrar aspectos de su vida cotidiana y nacen con la intención de plasmar la realidad en una imagen fija. La restauración de este tipo de imágenes, mediante un modelo matemático basado en dos técnicas fundamentales: propagación de texturas y detección de bordes, buscan dar prioridad equilibrada a las dos técnicas. Es decir, ninguna debe primar sobre la otra.

En primera instancia, la detección de bordes es una técnica que facilita la identificación de los objetos de la imagen y por lo tanto su reconocimiento. En sí, el propósito de los algoritmos de detección de bordes es obtener como resultado una imagen donde se resalten los pixeles de aquellos puntos de la imagen original en donde se presentan cambios bruscos de intensidad. En consonancia, es necesario resaltar las técnicas relacionadas con cruces por cero ("Zero crossing") y de curvas de nivel ("Level set") como mecanismos matemáticos para obtener dichos cambios.

Por otra parte, la propagación de texturas es una técnica ampliamente utilizada a nivel de software de procesamiento gráfico con la finalidad de complementar estructuras desenfocadas y regiones restantes o inexistentes. Su base se centra en la selección de una textura que contenga suficiente información sintetizada y propagarla a las regiones restantes con el afán de complementar parte de la información faltante; de esta forma se obtendría una imagen final con una calidad visual superior a su símil en procesamiento. [1]

Bajo el contexto anterior se dice que la restauración de imágenes a través de modelos matemáticos busca perfeccionar la modificación de manera inteligente, es decir, que al finalizar el proceso de restauración la imagen sea visualmente agradable al observador en donde la sustracción de elementos o la modificación del tamaño no dañe la armonía de

colores, formas y texturas. Sin embargo, existe muy poca información y/o documentación sobre los mecanismos empleados a nivel de software en distribuciones que permiten manipular imágenes de forma automática.

Los seres humanos están rodeados de imágenes y la mayoría de información que recibimos, son recolectadas en formatos de tipo mezcla de colores o en blanco y negro. En los primeros días del arte y la fotografía la restauración de obras se hacía manualmente por los artistas y con el pasar de los años el mismo proceso se fue imitando para convertirse en una técnica digital y semiautomática.

La restauración de imágenes tiene cabida en varios campos de aplicación:

Se puede destacar el área de la medicina en donde las imágenes sufren degradaciones por el ruido que inyectan los equipos médicos, haciendo que al final el diagnóstico no sea tan fiable. Por citar varios ejemplos claros de ello, se tienen las imágenes de tomodensitometría (rayos X), tomografía computarizada belicoidal (HTC), la resonancia magnética (MRI), el ultrasonido, entre otros. Las cuales requieren un proceso de restauración posterior a la adquisición con el objetivo de reducir el tiempo de exposición del paciente a corrientes de radiación. [2]

Otro campo que se ha visto beneficiado con este proceso es la astronomía, el objetivo es corregir efectos de degradación causados por polvo galáctico o distorsión de la luz en las fotografías tomadas por telescopios. En este campo, la restauración es aplicada para recuperar detalles que se perdieron en la adquisición de imágenes para lo cual suelen ser sometidas a una serie de procesos previos a la reconstrucción ya que al ser captadas en el espacio se debe analizar factores como desenfoque, movimiento y degrado atmosférico. [3]

En la actualidad la restauración de imágenes ha tenido un gran crecimiento debido al uso de modelos matemáticos y computacionales, estos métodos están orientados a mejorar la calidad de la imagen para facilitar la interpretación humana. La restauración ideal de una imagen solo se obtiene a través de un proceso matemático e implica analizar el grado de afectación para compensar o eliminar distorsiones. Para compensar distorsiones se tiene

en cuenta que en muchos casos las imágenes no son afectadas en su totalidad y algunas solo presentan pérdida de información en regiones específicas, también se puede dar el caso que simplemente se requiere eliminar algún objeto o modificar el tamaño para construir la imagen deseada.

Aunque hoy en día se cuente con instrumentos o aplicaciones cada vez mejores y con mayor precisión para definir la calidad de una imagen no está exentos de errores que impidan obtener a detalle la información. Ya sea para dar un diagnóstico más acertado en el campo de la medicina u obtener imágenes espaciales que contribuyan al avance de la ciencia; la importancia de restaurar imágenes es conseguir una imagen verdadera, recuperando detalles que se perdieron durante la etapa de adquisición y permita demostrar algo que existe y que es visible.

De continuar empleando algoritmos que modifiquen el tamaño de imágenes de una manera poco inteligente, la información importante de la imagen como sombras, texturas o bordes no son cuidados a detalle y por ende tienden a perder calidad visual para el espectador.

Otro aspecto a considerar es que durante el proceso de modificación el tiempo de procesamiento computacional y la demanda de recursos en cuanto a memoria de intercambio de la máquina para aplicar determinado algoritmo, influye en la obtención pronta de resultados de las imágenes sometidas a restauración.

Bajo este contexto, una imagen de mala calidad por lo general suele ser desechada; sin embargo, con el uso de modelos matemáticos ligeros convertidos en algoritmos computacionales, aparte de ser rápidos y concisos en el proceso de restauración pueden generar resultados aceptables para el observador.

1.2. Justificación

La restauración de imágenes es sin duda un campo de especial interés tanto desde el punto de vista matemático como comercial. En las industrias del cine y la publicidad es frecuente la necesidad de restaurar imágenes que por el paso del tiempo y/o falta de cuidado han llegado a deteriorarse y perdido el valor histórico que representan.

Algo similar pasa en el campo del arte, en el cual es frecuente la necesidad de restaurar determinadas obras. En este sentido existe una discusión abierta entre quienes consideran que una obra de valor artístico se debe restaurar y quienes consideran que solo se debe conservar.

La problemática de restaurar una imagen puede solucionarse utilizando uno o varios modelos matemáticos convertidos en algoritmos computacionales, desarrollados en un entorno de programación de acceso global. Para lo cual se requiere en primera instancia implementar el algoritmo que permita procesar las imágenes fotográficas documentales RGB, tomando en cuenta las técnicas del procesamiento de imágenes basados primordialmente en detección de bordes y propagación de texturas.

En tal virtud, para solventar el problema se ha planteado **UN MODELO MATEMÁTICO BASADO EN TÉCNICAS DE DETECCIÓN DE BORDES Y PROPAGACIÓN DE TEXTURAS PARA RESTAURAR IMÁGENES FOTOGRÁFICAS DOCUMENTALES RGB** implementado a nivel de software.

El mismo que ofrece la versatilidad para reconstruir secciones pérdidas o deterioradas y suprimir objetos de imágenes fotográficas documentales RGB; pese a que esta labor ya es conocida desde los inicios de los años 2000, desde el punto de vista técnico en el mundo de las imágenes este concepto sigue cambiado, por lo que la investigación presentada a permitido corroborar la funcionalidad de un modelo matemático computarizado que complementa dos principios de restauración de imágenes desarrollado en una plataforma de programación de libre acceso.

1.3. Objetivos

1.3.1. Objetivo General

Desarrollar un modelo matemático basado en las técnicas de procesamiento digital de imágenes, detección de bordes y propagación de texturas que permitan restaurar imágenes fotográficas documentales RGB.

1.3.2. Objetivos Específicos

- Realizar el estado del arte relacionado a las técnicas de detección de bordes y propagación de texturas que permitan restaurar imágenes fotográficas documentales RGB.
- Diseñar un algoritmo computacional del modelo matemático propuesto que permita la restauración de imágenes fotográficas documentales RGB.
- Implementar el algoritmo computacional del modelo matemático propuesto que permita la restauración de imágenes fotográficas documentales RGB.
- Realizar pruebas y análisis de restauración en imágenes fotográficas documentales RGB, mediante el algoritmo propuesto.

CAPÍTULO II

ANTECEDENTES INVESTIGATIVOS

2.1. Estado del arte

El objetivo de esta sección es presentar un análisis documental sobre los conceptos más preponderantes dentro del campo de la reconstrucción digital de imágenes; de esta manera, se detalla a continuación una recopilación teórica de varios autores que sustentan el desarrollo de este trabajo:

En la investigación de Shai, A. y Ariel, S (2007) [4] desarrollada en Estados Unidos se menciona que, para el cambio de tamaño apropiado de las imágenes no solo se debe usar restricciones geométricas sino también el contenido de la imagen. En la investigación se demuestra como un simple método llamado Seam Carving (Tallado de Costuras) admite el cambio de tamaño de una imagen según el contenido y se aplica tanto para reducir o ampliar la imagen. Se considera que una costura o seam es el camino o la ruta más óptima de 8 pixeles, conectados ya sea de arriba hacia abajo o de derecha a izquierda en una imagen en donde la optimización se obtiene a través de la función de energía de la imagen. El método de Seam Carving inserta progresivamente costuras en una dirección o ambas para cambiar el aspecto de la imagen y su principal objetivo es que el orden o ubicación de las costuras protejan el contenido de la imagen y visualmente sea agradable al observador.

Jonghoon, S etc al. (2016) en [5] realizado en Seúl, presenta un novedoso algoritmo rápido y preciso para el trazado de contorno de imágenes. Este algoritmo clasifica por tipos el pixel de contorno, luego traza el siguiente contorno en base al contorno anterior; por ende, puede clasificar el pixel como línea recta, esquina interior, esquina exterior y esquina interior-exterior además puede extraer pixeles de un contorno específico. El algoritmo también puede trazar pixeles de contorno rápidamente porque determina la ruta mínima local y es capaz de comprimir datos de pixeles de contorno empleando puntos representativos y de las esquinas interior-exteriores para restaurar con precisión el

contorno de una imagen. Lo novedoso del algoritmo es el rendimiento y tiempo de procesamiento y a diferencia de algoritmos convencionales puede proporcionar los datos comprimidos de los pixeles de contorno para restaurar con precisión la imagen incluyendo la esquina interior-exterior. En el procesamiento de imágenes la detección de bordes es de gran importancia y utilidad porque facilita tareas como el reconocimiento de objetos y segmentación de regiones.

Rubintein, M etc al. (2008) en [6] publicado en Seúl, Corea, presentan un estudio para modificar el tamaño de los videos y al igual que para imágenes el método Seam Carving puede ser empleado, pero con variantes que mejora el proceso. El método consiste en eliminar costuras 2D de volúmenes de espacio-tiempo 3D de manera que, el método de programación dinámica empleado para costuras 1D de imágenes 2D es reemplazado por Seam Craving con cortes gráficos adecuados para volúmenes 3D. En la nueva formulación las costuras están dadas por un corte mínimo en el gráfico para luego construir un gráfico de manera que el corte resultante sea una costura valida. También se muestra un novedoso criterio energético para mejorar la calidad visual de las imágenes y videos y consiste en eliminar las costuras que introducen la menor cantidad de energía en el resultado reorientado. El original Seam Carving elimina las costuras con menor cantidad de energía ignorando la energía que este operador introduce en la imágenes y videos restaurados, en este documento se muestra como codificar el método Seam Carving y la programación dinámica para cortes de gráficos en imágenes mostrando los diversos resultados.

Zhang, Dengyong, et al. (2017) en [7] proponen en su investigación un algoritmo para detectar si una imagen es original o fue alterada basado en el método de Seam Carving. El tallado de costuras o seam carving es una técnica para corregir o modificar imágenes, al emplear esta técnica se eliminan algunas costuras de la imagen original alterando las entropías espaciales y espectrales (SSE por sus siglas en ingles Spatial and Spectral Entropies). El algoritmo combina la función de energía basada en el patrón binario local para obtener características que permiten detectar la alteración de la imagen. También presenta un enfoque basado en imágenes de bajo escalado, es decir, si una imagen presenta pérdida de información se concluye que atravesó por una reconstrucción.

Mahdi, H et al. (2019) en [8] publicado en Irán, presenta en su investigación el método de Seam Carving mejorado y que conserva las sombras. Los algoritmos que reducen el tamaño de las imágenes basados en el método de Seam Carving dependen en gran medida de la cantidad del mapeo de energía extraído de la imagen, sin embargo, hasta el momento los algoritmos existentes no han tomado en cuenta las sombras dentro de la imagen. Las sombras dentro de una imagen implican información importante y ayudan a una mejor compresión del contenido, este hecho hace que la conservación de las sombras debe ser mantenida en el proceso de reducción de tamaño. El algoritmo propuesto en este estudio fue diseñado con el propósito de extraer las sombras de manera eficiente además se propone un mapa de prominencia para resaltar objetos sobresalientas en las imágenes y otro mapa de degrado que combinados muestra como las imágenes resultantes guardan sombras dentro de ellas y estructuras importantes.

2.2. Deep Learning

Deep learning o aprendizaje profundo es una técnica mejorada en donde los sistemas tienen que ser capaces de aprender de la experiencia para adquirir por sí mismos su propio conocimiento. Al ser un aprendizaje automático no es necesario la supervisión de operador humano que especifique los conocimientos que necesita el sistema. En el campo de reconocimiento de imágenes su funcionamiento es complejo, pero permite reconocerlas y clasificarlas cualitativamente de forma eficaz.

Para una computadora es difícil entender el significado de los datos de entrada sensorial sin procesar, por ejemplo, una imagen representada con valores de píxeles en donde la función de mapeo de un conjunto de píxeles es muy complicada aprender o evaluar si se aborda directamente. El aprendizaje profundo resuelve esta dificultad rompiendo el complicado mapeo en una serie de mapeos simples anidados, cada uno descrito por una capa diferente del modelo.

La entrada se presenta en la capa visible, llamada así porque contiene las variables que podemos observar. Luego, una serie de capas ocultas extrae cada vez más características abstractas de la imagen. Estas capas se denominan "ocultas" porque sus valores no se dan en los datos; en cambio, el modelo debe determinar qué conceptos son útiles para explicar

las relaciones en los datos observados. Las imágenes aquí son visualizaciones del tipo de característica representada por cada unidad oculta. Dados los píxeles, la primera capa puede identificar fácilmente los bordes comparando el brillo de los píxeles vecinos. Dado que descripción de los bordes de la primera capa oculta, la segunda capa oculta puede buscar fácilmente esquinas y contornos extendidos, que son reconocibles como conjuntos de bordes. La descripción de la imagen de la segunda capa oculta en términos de esquinas y contornos, la tercera capa oculta puede detectar partes enteras de objetos específicos, encontrando colecciones específicas de contornos y esquinas. Finalmente, esta descripción de la imagen se puede utilizar para reconocer los objetos presentes en la misma. [9]

2.3. Marching Cubes

Uno de los algoritmos más difundidos para la extracción de iso-superficies es el algoritmo Maching Cubes (MC), este algoritmo genera una malla poligonal a partir de la extracción de iso-superficies de un conjunto volumétrico de datos. El principio del algoritmo MC es subdividir el espacio en una cadena de cubos pequeños conocidos como vóxeles (imagen segmentada), es decir, utiliza un iso-valor para extraer y renderizar una iso-superficie desde contornos segmentados, como se observa en la figura 1 cada punto de la malla es un vértice de un cubo y cada cubo está definido por el mallado del volumen. [10]

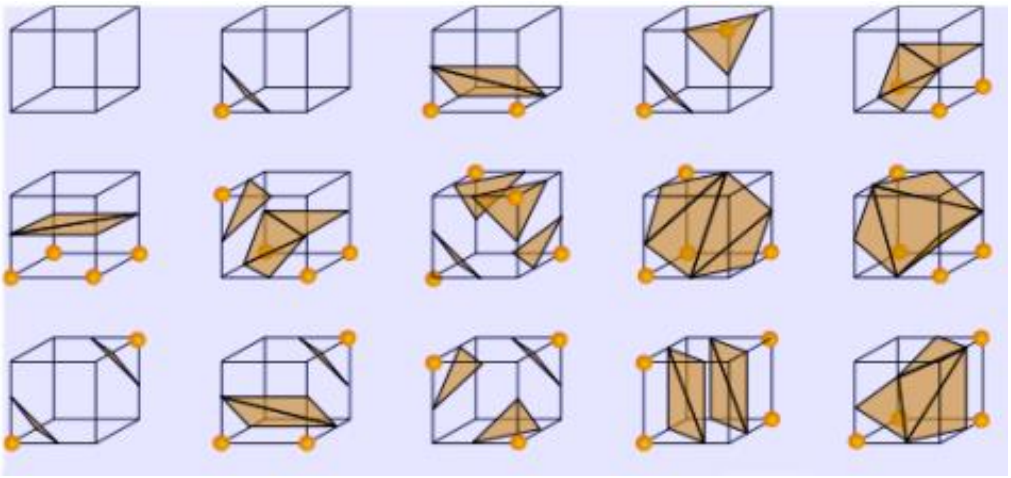


Figura 1. Las 15 combinaciones del algoritmo Marching Cubes.
Fuente: [10]

Cada uno de los ocho vértices de un cubo pueden o no marcarse, por lo tanto son posibles $2^8 = 256$ combinaciones pero este número se reduce a 128 asumiendo que dos

configuraciones son iguales si se invierten los puntos marcados y no marcados y las normales de los triángulos generados, por lo tanto se simplifica a 15 combinaciones finales mostradas en la figura anterior. EL algoritmo produce un conjunto de polígonos triangulares que se toma como datos de entrada, cada arista de la celda puede tener una intersección con la iso-superficie y se produce cuando el valor de umbral queda acotado en la parte superior e inferior por los valores en los vértices de la arista como muestra la figura 2.

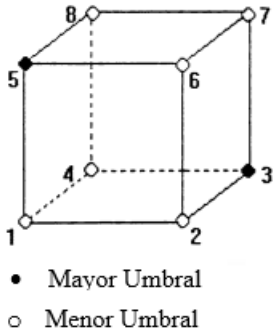


Figura 2. Numeración de vértices.
Fuente: [10]

En el caso de que el valor de un vértice este por encima de la iso-superficie y el valor de un vértice adyacente este por debajo se interpola linealmente la posición donde la iso-superficie corta el borde, los puntos de intersección se calculan por interpolación línea. Por ejemplo, V1 y V2 son los vértices de un borde de corte y Val1 y Val2 son los valores escalares de cada vértice, el punto de intersección P está dado por la ecuación 1:

$$P = V_1 + (isovalor - Val_1)\frac{V_2 - V_1}{Val_2 - Val_1} \quad (1)$$

2.4. Método Seam Carving

El método Seam Carving conocido también como tallado de costura, es un algoritmo de redimensionamiento de imágenes que procura preservar su contenido. El método consiste en reducir el tamaño de una imagen, sin alterar la calidad, para lo cual emplea una costura que es una ruta óptima de 8 pixeles conectados de arriba hacia abajo o de izquierda a derecha en una única imagen, donde la ruta está definida por una función de energía. [11]

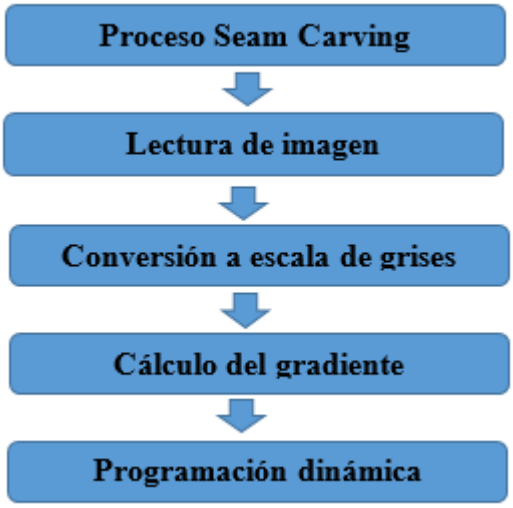


Figura 3. Proceso del método Seam Carving.
Elaborado por: El Investigador.

2.4.1. Método del gradiente

Una forma de determinar la función de energía en una imagen es a través del método del gradiente y consiste en derivar el mapa de energía de la imagen original para obtener las regiones más destacadas de la imagen. Para aplicar el método Seam Carving primero se debe calcular el valor de energía para cada pixel de la imagen, el método más básico de la función de energía consiste en asignar a cada pixel un valor $e(i,j)$, la fórmula del gradiente se puede ver en la ecuación 2.

$$e(I) = \left|\frac{\partial}{\partial x} I\right| + \left|\frac{\partial}{\partial y} I\right| \tag{2}$$

Ahora, la ecuación debe ser interpretada como la función de energía de la imagen I y es igual a la suma en valor absoluto de la derivada parcial en el eje X y la derivada parcial en el eje Y, obtenido como resultado el valor energético de ese pixel. [11]

Formalmente, una imagen I puede ser n×m donde la costura vertical puede ser definida por la ecuación 3.

$$s^x = \{s_i^x\}_{i=1}^n = \{(x(i), i)\}_{i=1}^n, s.t.\ \forall i, |\, x(i) - x(i-1)| \leq 1 \tag{3}$$

Donde x es un mapeo $x: [1, \ldots., n] \rightarrow [1, \ldots. m]$. La costura vertical es un camino conectado de 8 pixeles en la imagen de arriba hacia debajo de manera que contengan

exactamente uno y solo un pixel en cada fila de la imagen. De igual manera ocurre con la costura horizontal, que viene a ser un camino de derecha a izquierda y contiene exactamente un pixel en cada columna de la imagen. Si y es mapeado se tiene y: $[1, \ldots., m] \rightarrow\ [1, \ldots. n]$ donde la costura horizontal está dada por la ecuación 4.

$$s^y = \{s_j^y\}_{j=1}^m = \{(j, y(j))\}_{j=1}^m, s.t.\ \forall j, |\ y(j) - y(j-1)| \leq 1 \qquad (4)$$

El efecto de eliminar una costura es desplazar los pixeles hacia arriba o hacia la izquierda para compensar la ruta que falta, al tallar consecutivamente y de manera repetitiva se orienta la imagen a un nuevo tamaño. Encontrar la costura óptima de una imagen es un ejercicio simple de programación dinámica, el primer paso es recorrer la imagen desde la segunda fila hasta la última fila y calcular la energía mínima acumulada M para todas las costuras posibles conectadas en cada entrada (i, j) y está determinada por la ecuación 5.

$$M(i,j) = e(i,j) + \min\big(M(i-1, j-1), M(i-1, j), M(i-1, j+1)\big) \qquad (5)$$

Al culminar el proceso, el valor mínimo de la última fila en M indica el final de la costura mínima conectada, el segundo paso es retroceder desde esta entrada mínima en M para encontrar el camino óptimo de la costura. La definición de M para costuras horizontales es similar. A manera de ejemplo se muestra una imagen con su mapa de energía que nada más es la representación de la magnitud del gradiente. [12]

a) b)

Figura 4. a) Imagen de entrada al algoritmo Seam Carving; b) Mapa energético de la imagen
Fuente: [12]

Ahora, a partir del mapa de energía de la imagen se puede generar un conjunto de costuras que abarcan la imagen de forma horizontal o vertical. Las costuras de la imagen 5 en color rojo son las series que se puede eliminar sin que la imagen se vea afectada.

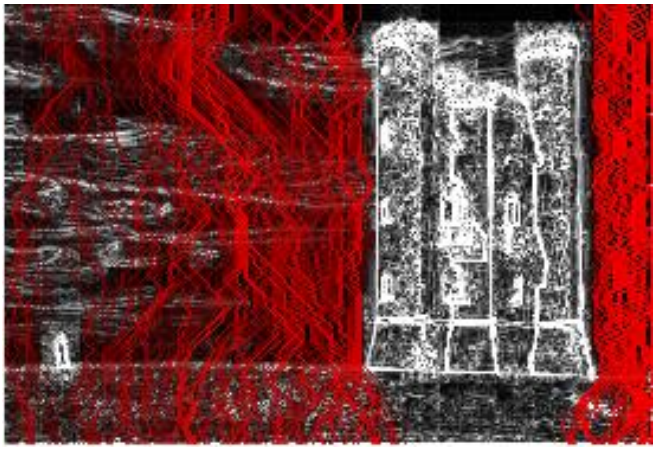
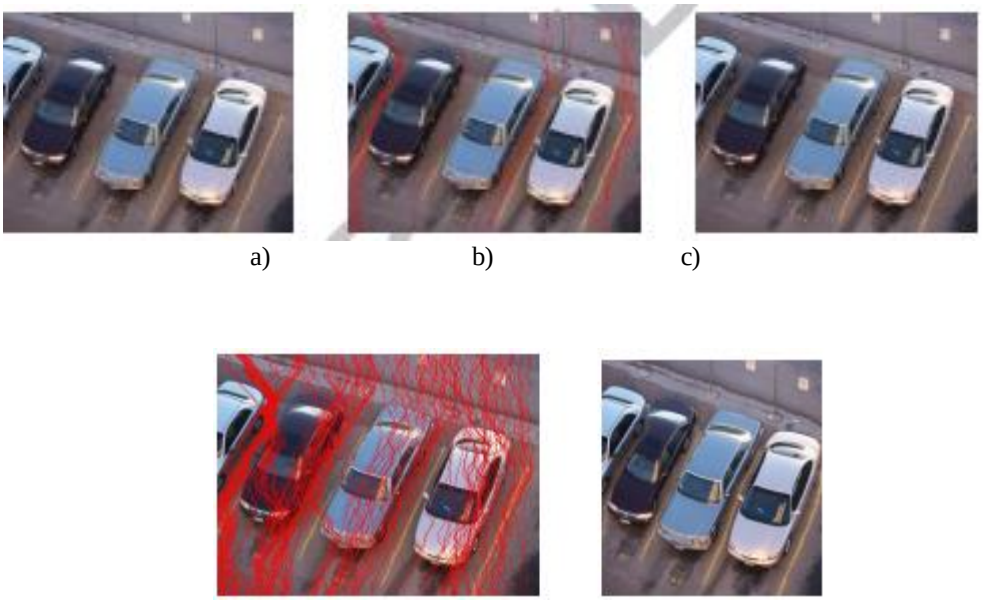


Figura 5. Costuras a partir del mapa energético.
Fuente: [12]

Las costuras con energía baja de la imagen anterior al ser retiradas reducen el tamaño de la imagen resultante, siendo el resultado deseado por el investigador.

Figura 6. Imagen restaurada.
Fuente: [12]

Cuando se retira una costura o seam de una imagen todos los pixeles se desplazan para rellenar los elementos faltantes, la figura 7 muestra el resultado de aplicar este algoritmo en una imagen. [13]



a) b) c)

d) e)

Figura 7. Uso de Seam Carving. a) Imagen original; b) imagen orinal con 3% de costuras verticales; c) imagen grabada al 3%; d) imagen orinal con 30% de costuras verticales; e) imagen grabada al 30%.
Fuente: [13]

Este algoritmo también puede ser empleado en video, la idea principal es la misma, cambiar el tamaño del contenido eliminando las costuras con menor cantidad de energía tomando cada fotograma de video como una imagen y cambiar el tamaño de manera independiente. De esta manera el contenido menos importante del video se elimina y deja las características más importantes a mayor escala creando una especie de zoom. Un aspecto a tomar en cuenta cuando se aplica Seam Carving en videos son las RIO (Regiones de Interés) y se usan para establecer las rutas de visualización en dispositivos en los que el tamaño de la pantalla es más pequeño que el video.

En un video se busca un operador de cambio (secuencia de fotogramas donde la cámara dispara continuamente) para ser aplicado por separado en cada cuadro del video y encontrar las regiones que son de baja importancia en todos los cuadros del mismo. Este proceso se realiza calculando la función de energía de cada imagen de manera independiente para tomar el valor de energía más alto de cada pixel ubicado. Las costuras capturadas a través de este método son llamadas costuras estáticas porque no cambian a lo largo de los cuadros del video, por lo tanto, dada una secuencia de video $\{I_t\}_{t=1}^{N}$ se

puede pasar del espacio $L_1 - normal$ al espacio-temporal $L_1 - normal$ por medio de la ecuación 6.

$$E_{espacial}(i,j) = max_{t=1}^{N}\left\{\left|\frac{\partial}{\partial x}I_t(i,j)\right| + \left|\frac{\partial}{\partial y}I_t(i,j)\right|\right\}$$

$$E_{temporal}(i,j) = max_{t=1}^{N}\left\{\left|\frac{\partial}{\partial t}I_t(i,j)\right|\right\}$$

$$E_{global}(i,j) = \alpha \cdot E_{espacial} + (1-\propto)E_{temporal} \quad (6)$$

Esta medida puede verse con la proyección máxima del espacio $L_1 - normal$ en 2D, donde α € [0,1] sirviendo como un parámetro de balance en la distribución espacial y temporal. En la práctica es bueno sesgar la energía hacia la importancia temporal tomando α=0.3. La figura 8 muestra un ejemplo del mapa de energía global y la eliminación de costuras estáticas en videos. [6]

Figura 8. Eliminación de costuras estáticas para un video de golf. La función de energía se muestra empleando mapeo de color violeta (bajo) a rojo (alto) la siguiente secuencia de imágenes muestra el resultado de emplear el algoritmo.
Fuente: [6]

2.4.2. Programación dinámica para encontrar la costura de menor energía

El cálculo de la costura de menor energía consiste en obtener la energía de cada píxel para obtener la ruta de energía mínima. Para encontrar la ruta, se debe buscar la costura de arriba abajo, el valor del píxel se sumará cada tres píxeles más cercanos por encima de él, y luego se determina el valor mínimo. [14]

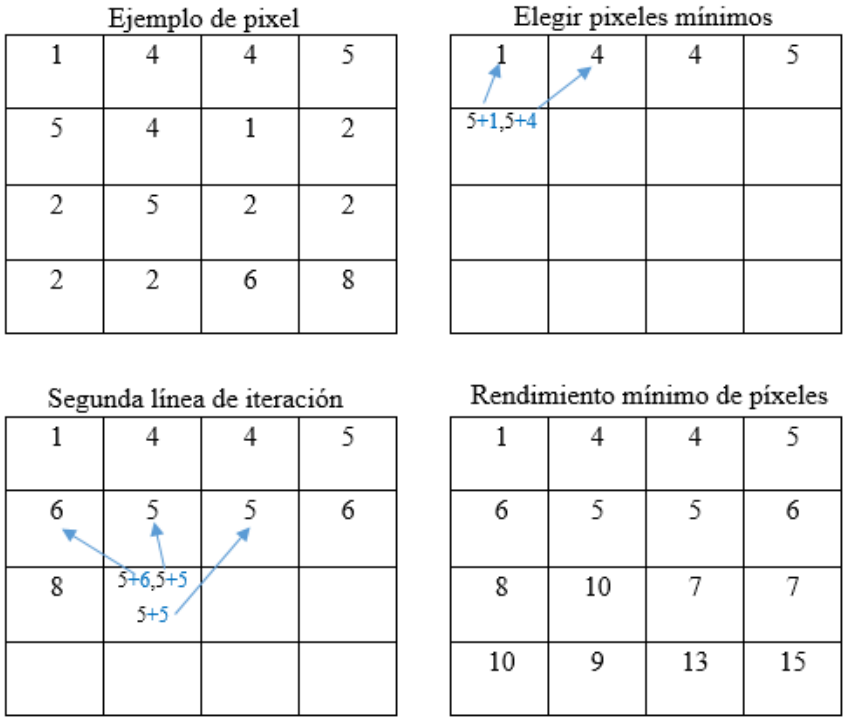


Figura 9. Asignación de valores mínimos
Elaborado por: El Investigador.

Después de obtener el rendimiento mínimo de pixeles se lleva a cabo la formación de la ruta. El píxel con el valor mínimo en la fila inferior será el punto de partida para seleccionar la ruta óptima. Luego, la siguiente ruta se selecciona en función del valor mínimo de los tres píxeles vecinos por encima de ella.

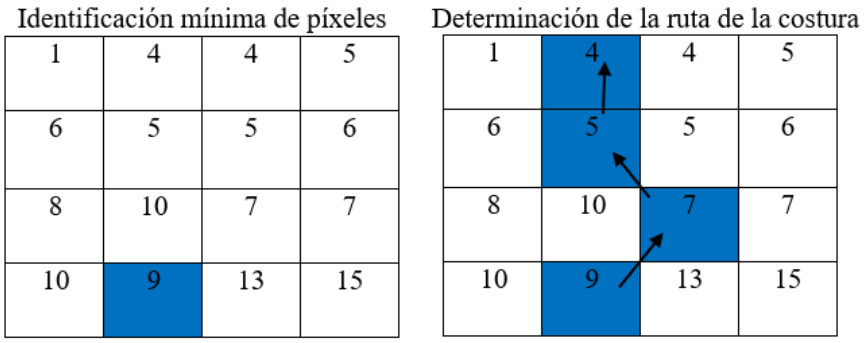


Figura 10. Ruta de la costura de menor energía.
Elaborado por: El Investigador.

La tabla 1 muestra diferentes técnicas para restaurar imágenes, en dicha tabla compara cada técnica mostrando sus ventajas y desventajas con la finalidad de escoger la técnica que mejor se ajuste a las necesidades del investigador.

Tabla 1. Comparación de técnicas para restaurar imágenes.

Técnica	**Ventajas**	**Desventajas**
Deep Learning	Identifican objetos en una imagen bajo modelos entrenados y con ese patrón remueven solo determinados objetos, es decir, usan el modelo entrenado para resolver un problema y luego resuelve por sí mismo problemas similares. No requiere la intervención humana para seleccionar objetos a remover. Modelos pre entrenados para restaurar numerosas imágenes en menor tiempo bajo un patrón aprendido.	No disponibilidad de grandes cantidades de datos para entrenar a los sistemas o modelos. No disponer de hardware y software realmente potentes en los computadores para ejecutar los modelos. Se debe emplear gran cantidad de tiempo en la etapa de entrenamiento y en el proceso de los cálculos matemáticos.
Matching Cubes	Algoritmo de gran utilidad en el campo de la medicina, empleado para construir superficies tridimensionales a partir de imágenes en 2D obtenidas por tomografías o resonancias magnéticas.	Durante la reconstrucción se genera superficies con huecos que no representan fielmente las superficies que se intenta modelar. Al ser un algoritmo diseñado para 3D el tiempo de

		procesamiento y demanda de recursos computacionales es más alto.
Seam Carving	Basa la restauración de imágenes de acuerdo al contenido significante de pixeles. Modelo matemático diseñado para mantener el tamaño original de la imagen a través de inpainting. Respeta el contenido más importante de la imagen a través de la detección de bordes y propagación de texturas, lo que significa tener éxito en la remoción de objetos y mantener el tamaño original.	Selección manual de los objetos a eliminar. La selección de objetos es una tarea lenta especialmente si hay demasiadas imágenes a restaurar. Si el objeto seleccionado no tiene forma concreta, por ejemplo, la silueta de un vehículo, casa, persona, montaña, etc. la probabilidad de fracaso del algoritmo es alta.

Elaborado por: El Investigador.

2.5. Técnicas de reconstrucción de imágenes

Existen gran variedad de algoritmos para la reconstrucción de imágenes o *inpainting* basados en la síntesis de texturas, extensión de estructuras y en la correspondencia de colores. Sin embargo, se clasifican principalmente en las siguientes categorías [15], [16]:

2.5.1. Reconstrucción de imágenes basadas en texturas

La detección de texturas es una técnica empleada para segmentar regiones con propiedades de textura significativas. Esta técnica de reconstrucción de imágenes reemplaza el área señalada con muestras de los pixeles cercanos, sin tener en cuenta los

bordes y filos por lo que su nivel de rendimiento es bajo y no funciona para la reconstrucción de todo tipo de imágenes. Se define como textura a la existencia de regularidad en una región continua de la imagen manifestándose de forma repetitiva o como patrón.

Existen diversos algoritmos que permiten la segmentación de imágenes y pueden dividir una imagen en regiones uniformes de color y textura; entre algunas técnicas se puede mencionar los campos aleatorios de Markov, dimensión fractal, polígonos de Voronoi, filtros de Gabor y descomposición de Wavelet. [17]

El método de Voronoi consiste en subdividir una superficie plana de forma regular y debe cumplir dos características; que no se superpongan figuras y que no queden espacios vacíos. De esta manera el teselado puede ser muy simple con polígonos regulares como cuadrados o triángulos o más complejo formando un teselado irregular. La figura 11 muestra ejemplos de teselados semirregulares. [18]

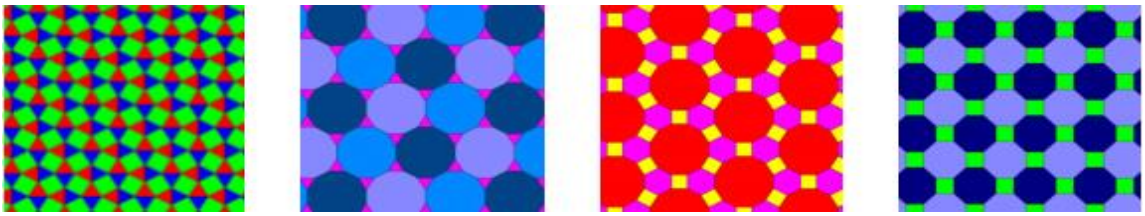
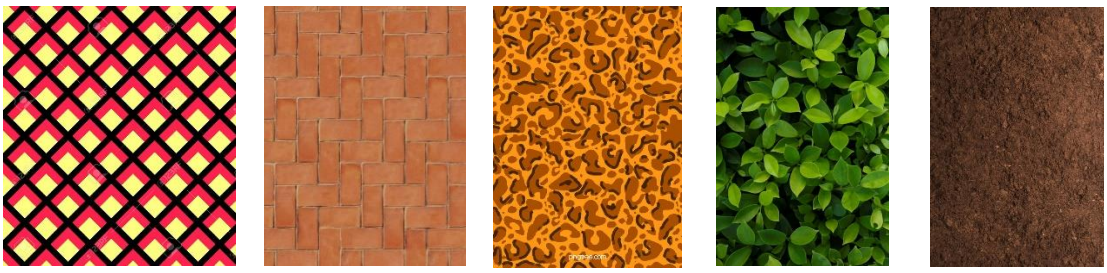


Figura 11. Teselados semirregulares.
Fuente: [18]

La figura 12 muestra una posible clasificación de los tipos de texturas como regular, semirregular, irregular, semiestocástica y estocástica.



a) b) c) d) e)

Figura 12. a) regular, b) semirregular, c) irregular, d) semiestocástica, e) estocástica.
Elaborado por: El Investigador.

2.5.2. Reconstrucción de imágenes basadas en ejemplares

Está técnica de reconstrucción de imágenes utiliza una combinación de la síntesis de texturas y expansión de estructuras para reemplazar los pixeles de la región señalada. Para ello se realiza el cómputo de prioridades con la finalidad de determinar el orden del llenado de la región. Funciona bien para gran cantidad de imágenes, pero tiene problemas en el llenado de imágenes con regiones curvas.

2.5.3. Reconstrucción de imágenes basadas en Ecuaciones diferenciales parciales

Las técnicas de reconstrucción de imágenes basadas en Ecuaciones diferenciales parciales (PDE) predicen la información de imagen señalada mediante la extrapolación de píxeles vecinos utilizando ciertos criterios como la recuperación de bordes o principios variacionales. Para ello se basa en el supuesto de que la región a pintar es coherente con las regiones vecinas a través de la preservación de la topología. Las técnicas basadas en PDE son eficientes en la reconstrucción de regiones pequeñas con bordes delgados, pero deja un efecto borroso en las grandes.

2.5.4. Aproximaciones mixtas

Cada una de las técnicas presentadas en los puntos anteriores reconstruye imágenes utilizando enfoques diferentes, por lo que cada una tiene fortalezas y defectos. Debido a esto, las aproximaciones mixtas buscan combinar las fortalezas de las técnicas basadas en síntesis de textura y PDE para le reconstrucción de regiones. Primero se descompone la imagen en una región de textura y estructura para luego llenarlas por síntesis de textura y algoritmos de propagación de bordes.

2.6.Técnicas de Filtrado

Este tipo de métodos permite modificar de forma selectiva una imagen, es decir se puede corregir su iluminación, mejorar su claridad, destacar elementos o eliminarlos. En una imagen digital se puede usar dos técnicas de filtrado: *i) dominio del espacio,* en donde se trabaja de forma directa sobre la matriz de píxeles de cada imagen; o, *ii) dominio de la frecuencia,* aquí se lleva a cabo la transformada de Fourier de la imagen a modificarse.

Debido al enfoque de este estudio, los filtros correspondientes al dominio del espacio son de especial interés, pudiéndose describir de forma general como muestra la ecuación 7 [19]:

$$ND'_{i,j} = \frac{ND_{i-1,j-1} + ND_{i,j-1} + ND_{i+1,j-1} + ND_{i-1,j} + ND_{i,j} + ND_{i+1,j} + ND_{i-1,j+1} + ND_{i-1,j\mp1} + ND_{i-1,j+1}}{9} \quad (7)$$

En la ecuación 1, *i, j* representan la fila y columna de cada pixel respectivamente, *NDi,j*, establecen su nivel digital y, *ND'i,j* el nivel digital después de realizar el filtrado. Esta misma ecuación puede representarse mediante tablas, con valores por defecto como se muestra en la tabla 2.

Tabla 2. Filtro de espacio.

1	1	1
1	1	1
1	1	1

DIV=9

Elaborado por: El Investigador.

A continuación, se detallan los filtros más usados de acuerdo al dominio del espacio:

2.6.1. Filtros de Paso Bajo

El objetivo de este tipo de filtros es eliminar la mayor cantidad posible de ruido.

2.6.2. Filtro de la media

Fija al pixel central la media de todos los píxeles de la imagen. La matriz para este caso está compuesta por unos, además de que su divisor es el número total de elementos de la matriz. Esto se lo puede apreciar a través de la ecuación 8 [19]:

$$u_{media}(x,y) = \frac{\Sigma_{(x',y')\in V(x,y)} \; I(x',y')}{card(V(x',y'))} \quad (8)$$

Donde, $u_{media}(x, y)$ representa el promedio de los pixeles vecinos del pixel central escogido, $V(x', y')$ se refiere a la ventana de pixeles escogidos para realizar el promedio, $I(x', y')$ se refiere a la posición de cada pixel vecino y $card(V(x', y'))$ se interpreta como la cardinalidad de la ventana seleccionada.

2.6.3. Filtro de media ponderada

En este caso, no todos los elementos de la matriz corresponden a unos, se escoge un elemento y se le asigna un peso o ponderación más elevada que al resto de componentes. El elemento central es por lo general seleccionado para poseer una ponderación más alta. A continuación, se detalla en la fórmula 9 de filtro de media ponderada [19]:

$$I(x, y) = \frac{I(x', y') * P(x\prime, y\prime)}{\sum_{(x', y') \in P} P(x', y')} \quad (9)$$

Donde, $I(x, y)$, representa la matriz de píxeles filtrada, $I(x', y')$ se refiere a los píxeles de la matriz original, $P(x', y')$, es la matriz de ponderaciones, donde por lo general $0 \leq P(x', y') \leq 1$.

2.6.4. Filtro de la mediana

Generalmente se aplica después de un filtro de media, esto debido a que el valor final del píxel corresponde a un valor real y no a un valor promedio. Este tipo de filtro es complejo de calcular debido a que se debe realizar un proceso de ordenamiento de píxeles. Este filtro se basa en reemplazar el pixel central de la matriz original por la mediana de la matriz ordenada. Así se demuestra en la ecuación 10 [19]:

$$[3\ 5\ 8\ 6\ 9\ 2\ 4\ 0\ 9\ ] = \{3,5,8,6,9,2,4,0,9\}$$

$$[0\ 2\ 3\ 4\ 5\ 6\ 8\ 9\ 9\ ] = \{0,2,3,4,\ 5,\ 6,8,9,9\ \} \quad (10)$$

Donde la ecuación para un pixel de 3*3 sería:

$$[3\ 5\ 8\ 6\ 5\ 2\ 4\ 0\ 9\ ] = nuevo\ valor\ de\ atributo\ del\ pixel\ central$$

Entonces el filtro de la media es;

$$I'(n,m) = ord\ asc\ (I(n,m))$$

$$I\left(\frac{n+1}{2},\frac{m+1}{2}\right) = I'\left(\frac{n+1}{2},\frac{m+1}{2}\right) \tag{11}$$

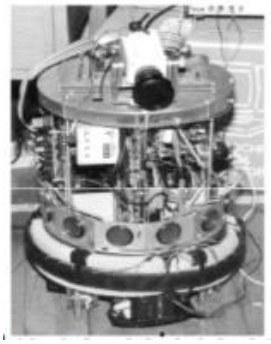
En la ecuación planteada, se puede apreciar que $I'(n,m)$ representa la matriz original ordenada ascendentemente, $I\left(\frac{n+1}{2},\frac{m+1}{2}\right)$, establece el pixel central o mediana de la matriz original, y; $I'\left(\frac{n+1}{2},\frac{m+1}{2}\right)$, representa el pixel central o mediana de la matriz ordenada de forma ascendente.

2.6.5. Filtros gaussianos

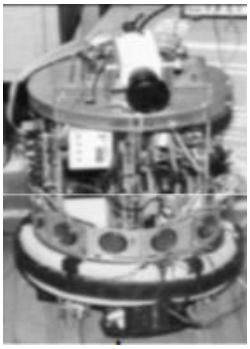
El máximo valor aparece en el píxel central y va disminuyendo hacia los extremos de acuerdo al valor del parámetro s (desviación típica). Para este tipo de filtros, se obtiene como resultados valores entre cero y uno. La distribución gaussiana empleada es de tipo bivariante. La ecuación 12 muestra la fórmula del filtro gaussiano, en donde σ^2 representa la varianza, es la siguiente [19]:

$$G\ (x,y) = \frac{e^{-\frac{x^2+y^2}{2\sigma^2}}}{2\pi\sigma^2} \tag{12}$$

Dentro de las características de este filtro se puede mencionar que disminuye la nitidez, pierde los detalles y además produce un suavizado altamente uniforme. Este tipo de características dependen del valor de la desviación estándar σ como se muestra a continuación:



a) Imagen Original



b) Filtro Gaussiano con $\sigma = 1$

c) Filtro Gaussiano con $\sigma = 2$

Figura 13. Aplicación de Filtros Gaussianos.
Fuente: [20]

2.6.6. Filtros de Paso alto

En este tipo de filtros, se elimina la media, es decir el elemento principal utilizado en los filtros de paso bajo. El objetivo es distinguir las zonas de mayor mutabilidad dentro de una imagen. Los filtros más importantes de esta clasificación son:

- Sustracción de la media.
- Filtros basados en las derivadas.

2.6.7. Filtros direccionales

Como su nombre lo indica, este filtro permite seguir una dirección específica trazada en el espacio, además de resaltar la diferencia existente entre cada píxel a lo largo de la trayectoria.

2.6.8. Filtros de detección de bordes

El filtro u operador de Sobel es el más utilizado en esta clasificación. Esta técnica permite detectar los bordes verticales y horizontales, de forma separada, de una imagen a escala de grises [21]. En el caso que se trabaje con una imagen en RGB, es necesario transformarla a un formato adecuada para su modificación. El filtro de Sobel está basado en gradientes que se aplica en imágenes en escala de grises y tiene plantillas de detección de bordes 3×3 en dirección dos direcciones que detectan bordes horizontal y vertical respectivamente como muestra la ecuación 13.

$$h_x = \begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix} \qquad h_{y=} \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix} \tag{13}$$

Al emplear el operador de Sobel se usan las plantillas de operador horizontal y vertical para realizar la convolución de cada pixel de la imagen como muestra las ecuaciones 14 y 15. [22]

$$h_x(x,y) = h(x+1,y-1) + 2h(x+1,y) + h(x+1,y+1) - h(x-1,y-1) - 2h(x-1,y) - h(x-1,y+1) \tag{14}$$

$$h_y(x,y) = h(x-1,y+1) + 2h(x,y+1) + h(x+1,y+1) - h(x-1,y+1) - 2h(x,y-1,) - h(x+1,y-1) \tag{15}$$

Desde el punto de vista matemático el operador de Sobel emplea mascaras para aplicar la convolución a la imagen original y calcular las derivadas en el eje horizontal y vertical. Para cada pixel de una imagen los resultados de los gradientes pueden ser combinados para calcular la magnitud total con la ecuación 16. [23]

$$G = \sqrt{G_x^2 + G_y^2} \tag{16}$$

La figura 14 muestra el resultado final es una imagen con bordes resaltados, líneas negras y vestigios del color original aplicando el filtro de Sobel.

a) Imagen Original

b) Aplicación de filtro Sobel

Figura 14. Ejemplo de aplicación de filtro Sobel.
Fuente: [24]

2.6.9. Comparación de las técnicas de filtrado

En la tabla 3, se presenta una comparación general de todas las técnicas de filtrado antes mencionadas [25].

Tabla 3. Comparación filtros.

Técnica	Ventajas	Desventajas
Filtros de paso bajo	Permiten suavizar la imagen al eliminar el ruido presente en ella. Permite resaltar zonas que se encuentren en una escala específica.	Solo se puede utilizar en imágenes con gran cantidad de ruido.
Filtros de paso alto	Permite resaltar los contrastes ya que suaviza las bajas frecuencias.	Son difíciles de calcular ya que requieren la aplicación de derivadas.
Filtros direccionales	Permite identificar cambios de los niveles de intensidad de los pixeles en una imagen. Permite aislar estructuras que siguen una dirección en una imagen.	Se pueden producir falsos bordes a causa de la existencia de ruido en la imagen.
Filtros de detección de bordes	Permite identificar cambios de los niveles de intensidad de los pixeles en una imagen. Permite aislar características en las regiones de una imagen.	Se pueden producir falsos bordes a causa de la existencia de ruido en la imagen.

Elaborado por: El Investigador

2.6.10. Comparación de los filtros de paso bajo

En la tabla que se muestra a continuación, se exponen las ventajas y desventajas de los filtros de paso bajo, los cuales son de principal interés para esta investigación [19], [26]:

Tabla 4. Comparación filtros paso bajo.

Filtro	**Ventajas**	**Desventajas**
Filtro de la media	La gran ventaja de este filtro es que es simple de aplicar.	Este filtro es sensible a cambios locales. Puede generar intensidades de gris que no existían.
Filtro de la mediana	El valor del último pixel es un valor real. Reduce el efecto borroso en la imagen	La aplicación de este filtro es más compleja ya que requiere que se ordenen los valores y determinar su valor central.
Filtro gaussiano	Permite realizar dos convoluciones unidimensionales, una en sentido horizontal y otra en vertical. Permite tener un suavizado uniforme.	Tiene una pobre respuesta a los ruidos de sal y pimienta.

Elaborado por: El Investigador

En base a los cuadros comparativos de ventajas y desventajas de técnicas de filtrado y de filtros presentados en las tablas 3 y 4 correspondientemente, se determina que para la aplicación de este proyecto se van a utilizar los filtros de paso bajo ya que permiten reducir el ruido de la imagen, dentro de este específicamente el filtro gaussiano. Adicionalmente

se va a utilizar el filtro de detección de bordes con lo que se va a poder identificar y extraer regiones de una imagen.

2.7. Imágenes

2.7.1. Imagen analógica

Este tipo de imágenes, hacen referencia a ilustraciones captadas por una cámara o cualquier dispositivo óptico sin la necesidad de ser procesadas o almacenadas en un ordenador. Las imágenes analógicas pueden ser a blanco y negro, a escala de grises o a colores [21].

2.7.2. Imagen digital

La conversión de una imagen analógica en un formato adecuado para su manipulación por ordenador, se conoce como imagen digital. Este tipo de imágenes son representadas bidimensionalmente (eje "x" e "y") a partir de una matriz numérica expresada, comúnmente, en lenguaje binario. Matemáticamente, una imagen puede ser expresada a través de una función f (x, y), cuyos valores de intensidad deben ser discretos y finitos. Un punto se representa mediante el índice de las filas y las columnas y un valor que identifica el color, al cual se denomina pixel. El píxel ("picture element" o "elemento de imagen") se lo puede definir como la unidad más pequeña que posee una imagen digital [27].

Para entender estos conceptos de mejor manera, se puede considerar una imagen cualquiera y dividirla a través de una cuadrícula uniforme. Cada parte de esta cuadrícula representa un píxel, es decir una imagen digital se la puede describir a través de una matriz de píxeles. En el caso de que se trabaje con una imagen en escala de grises o a blanco y negro, cada elemento de la matriz de píxeles guarda la intensidad de brillo correspondiente a esa ubicación, ver figura 15 [28] .

Por otro lado, si se trabaja con una imagen a color, una matriz no es suficiente, debido a esto se hace necesario el uso de tres matrices independientes monocromáticas, que representan la intensidad de cada color detallado en el modelo RGB, así lo demuestra la figura 14 [29].

$$\begin{bmatrix} P_{1,1} & P_{1,2} & \dots & P_{1,n} \\ P_{2,1} & P_{2,2} & \dots & P_{2,n} \\ P_{3,1} & P_{3,2} & \dots & P_{3,n} \\ \vdots & \vdots & \ddots & \vdots \\ P_{m-2,1} & P_{m-2,2} & \dots & P_{m-2,n} \\ P_{m-1,1} & P_{m-1,2} & \dots & P_{m-1,n} \\ P_{m,1} & P_{m,2} & \dots & P_{m,n} \end{bmatrix}$$

Figura 15. Matriz de pixeles imagen blanco y negro.
Fuente: Elaborado por: El Investigador.

$$\begin{bmatrix} R_{1,1} & R_{1,2} & \dots & R_{1,n} \\ R_{2,1} & R_{2,2} & \dots & R_{2,n} \\ R_{3,1} & R_{3,2} & \dots & R_{3,n} \\ \vdots & \vdots & \ddots & \vdots \\ R_{m-2,1} & R_{m-2,2} & \dots & R_{m-2,n} \\ R_{m-1,1} & R_{m-1,2} & \dots & R_{m-1,n} \\ R_{m,1} & R_{m,2} & \dots & R_{m,n} \end{bmatrix} \begin{bmatrix} G_{1,1} & G_{1,2} & \dots & G_{1,n} \\ G_{2,1} & G_{2,2} & \dots & G_{2,n} \\ G_{3,1} & G_{3,2} & \dots & G_{3,n} \\ \vdots & \vdots & \ddots & \vdots \\ G_{m-2,1} & G_{m-2,2} & \dots & G_{m-2,n} \\ G_{m-1,1} & G_{m-1,2} & \dots & G_{m-1,n} \\ G_{m,1} & G_{m,2} & \dots & G_{m,n} \end{bmatrix} \begin{bmatrix} B_{1,1} & B_{1,2} & \dots & B_{1,n} \\ B_{2,1} & B_{2,2} & \dots & B_{2,n} \\ B_{3,1} & B_{3,2} & \dots & B_{3,n} \\ \vdots & \vdots & \ddots & \vdots \\ B_{m-2,1} & B_{m-2,2} & \dots & B_{m-2,n} \\ B_{m-1,1} & B_{m-1,2} & \dots & B_{m-1,n} \\ B_{m,1} & B_{m,2} & \dots & B_{m,n} \end{bmatrix}$$

Figura 16. Matriz RGB imagen a color.
Elaborado por: El Investigador.

2.8. Modelo de color RGB

Una imagen digital en color está formada por un número finito de píxeles, donde cada uno tiene una localidad y un valor particular. En una imagen a color el valor de cada pixel se encuentra codificado de acuerdo al modelo de color.

Los modelos de color más comunes son: RGB, CMYK, HDB, RYB, NCS, HSI. Este tipo de modelos, representan un método numérico diferente de descripción de los colores, sin embargo, debido a que el presente trabajo de investigación se basa en el estudio y procesamiento de imágenes mediante la modificación de píxeles se profundizará en el modelo más empleado en este campo, RGB [30], [31].

El modelo RGB está compuesto por los tres colores luz primarios que son: Rojo, Verde y Azul, cuyas siglas en inglés corresponden a Red, Green y Blue. Cada elemento de la imagen puede tomar valores del 0 al 255, donde 0 significa que el color no interviene. Este modelo permite representar el valor de color de un píxel mediante una tripleta de valores o matriz de niveles de intensidad, donde cada valor corresponde a la intensidad de cada uno de sus componentes (R, G, B). Para representar un color se combinan por adición ciertas cantidades de los tres colores, ver figura 17 [28], [32].

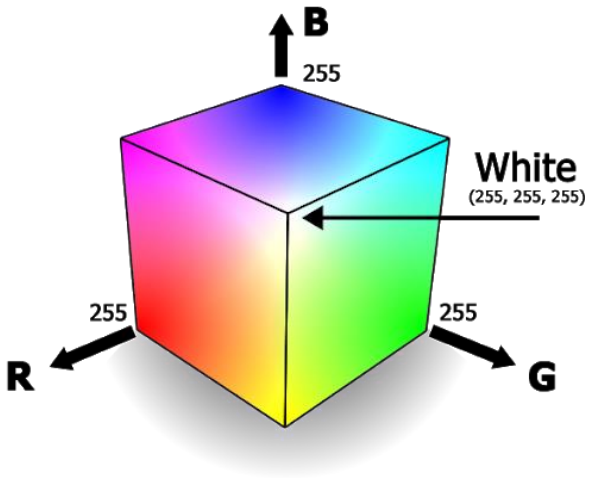


Figura 17. Representación RGB.
Elaborado por: El Investigador.

2.8.1. Procesamiento de imágenes

El procesamiento de imágenes es un método mediante el cual se realiza operaciones con una imagen digital con la finalidad de obtener una imagen mejorada o de extraer información de ella. El procesamiento de imágenes incluye principalmente las siguientes fases [27], ver figura 18:

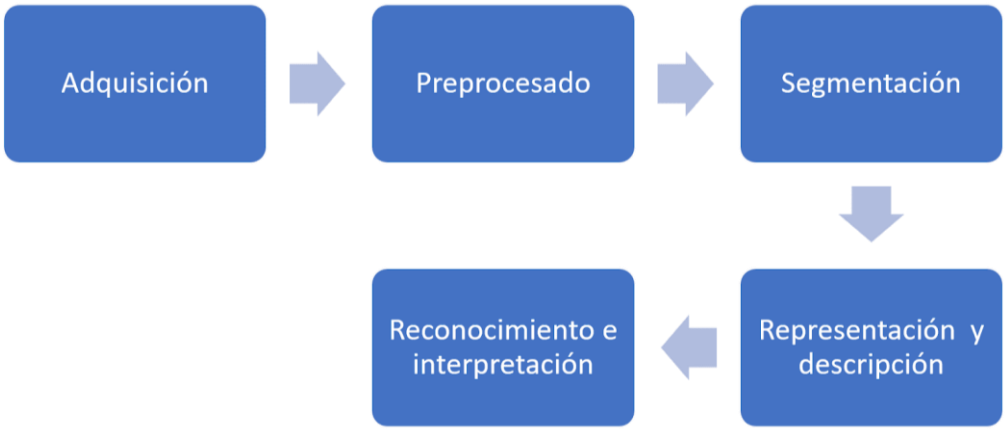


Figura 18. Fases del procesamiento de imágenes.
Elaborado por: El Investigador.

- Adquisición: En esta fase se obtiene una imagen para ello generalmente se utilizan dispositivos de digitalización tales como cámaras.
- Pre-procesado: Consiste en la mejora de la imagen en características tales como el contraste, suavizado, reducción de ruido y el realce de los detalles con la finalidad de que la imagen se va a procesar tenga mejores resultados,
- Segmentación: En esta fase se realiza la división de la imagen en imágenes más pequeñas a fin de poder separar los objetos relevantes o representativos.
- Representación y descripción: En esta fase se busca obtener las características principales y representar los píxeles de las diferentes imágenes obtenidas de la segmentación de tal forma que sea fácil diferenciar una de otra y se facilite su posterior procesamiento.
- Reconocimiento e interpretación: En esta fase se identifican los objetos en base a las diferencias en cuanto a características obtenidas en la fase anterior y se les asigna un significado a los objetos reconocidos.

2.9. Reconstrucción de imágenes

La reconstrucción de imágenes es el proceso de reparar partes deterioradas o dañadas de una imagen [33]. Este concepto generalmente es referenciado mediante el término inglés *inpainting*. Este término ha sido utilizado desde el renacimiento para referirse a la restauración de obras de arte y posteriormente fue adaptada al área digital. En el año 2002 Marcelo Beltamio introdujo el término *digital inpainting* para referirse al proceso automático de reconstrucción de imágenes sin más intervención humana que la limitación del área a ser reconstruida [34].

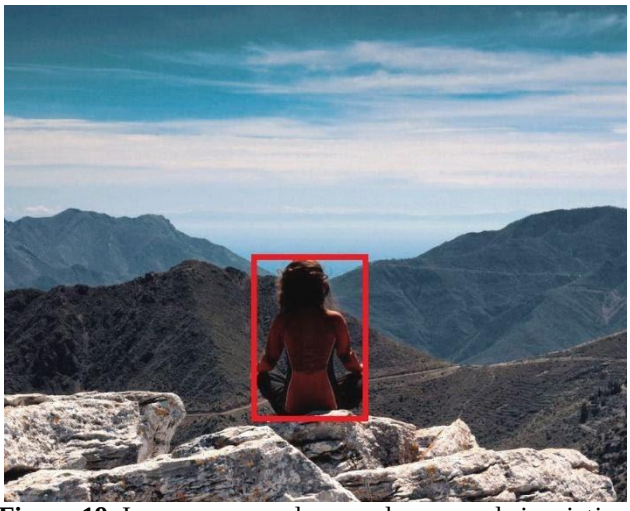


Figura 19. Imagen preparada para el proceso de inpainting.
Fuente: [34]

En el área de computación la reconstrucción de imágenes se refiere a la aplicación de algoritmos con la finalidad de reemplazar o sustituir las partes deterioradas o dañadas una imagen digital. En la actualidad esta técnica también se utiliza para eliminar o remover objetos no deseados de una imagen digital [35].

La reconstrucción de imágenes tiene varias áreas de aplicación, entre ellas se encuentran la cinematografía, medicina, cartografía, arquitectura, astronomía, arte, etc. En estas áreas se aplica la reconstrucción de imágenes para reemplazar objetos, mejorar la calidad de las imágenes e incluso introducir objetos para adaptar las imágenes a ideologías.

2.10. Hipótesis

El modelo matemático basado en técnicas de detección de bordes y propagación de texturas incide en la restauración de imágenes fotográficas documentales RGB.

2.11. Señalamiento de variables

Variable Independiente: Modelo matemático basado en técnicas de detección de bordes y propagación de texturas.

Variable Dependiente: Restauración de imágenes fotográficas documentales RGB.

CAPÍTULO III

METODOLOGÍA

3.1. Ubicación

El presente trabajo de investigación fue realizado en la ciudad de Ambato capital de la provincia de Tungurahua, es una de las capitales más importantes de la región centro del Ecuador. Esta cuidad está considerada con un nivel socio económico medio según el Instituto Nacional de Estadísticas y Censos (INEC) en 2010, luego de realizar una encuesta a una muestra de 9.744 viviendas distribuidas en cinco ciudades del país como Quito, Guayaquil, Cuenca, Machala y Ambato. [36]

3.2.Equipos y materiales

Los equipos y materiales utilizados en el proyecto de investigación se describen a continuación:

Tabla 5. Equipos y materiales.

Equipos/materiales	Costo (USD)
Equipos, software y servicio técnico	500
Materiales y suministros	200
Material bibliográfico y fotocopias	200
Transporte	200
Gastos varios	200
Consulta con expertos	500
Total	1800

Elaborado por: El Investigador.

3.3.Tipo de investigación

3.3.1. Investigación Exploratoria

La investigación fue de tipo exploratoria porque la literatura revela documentos poco estudiados con ideas remotamente vinculadas con el tema a desarrollar, es decir, se tiene pocos antecedentes sobre el tema en cuestión.

3.3.2. Investigación Correlacional

La investigación fue de tipo correlacional porque buscó determinar la influencia del modelo matemático basado en técnicas de detección de bordes y propagación de texturas sobre las imágenes procesadas a través del mismo.

3.3.3. Investigación Explicativa

La investigación fue de tipo explicativa porque define el algoritmo, proceso y técnicas empleadas en la restauración de imágenes fotográficas documentales RGB identificando las causas y consecuencias de los problemas detectados en el procesamiento de las imágenes.

3.4. Prueba de hipótesis

El modelo matemático basado en técnicas de detección de bordes y propagación de texturas incide en la restauración de imágenes fotográficas documentales RGB.

3.5.Población y muestra

3.5.1. Población

Para el desarrollo del presente proyecto de investigación se tomará como población todos los tipos de imágenes fotográficas siendo estas el objeto de estudio en la restauración de imágenes. Al hablar de tipos de fotografías no se refiere a géneros fotográficos sino más bien hace referencias a la manera de catalogar, enmarcar cada imagen por una técnica concreta. Dentro de esta clasificación se tiene imágenes fotográficas de tipo artística, retratos, publicitaria, documental, periodística y de moda.

3.5.2. Muestra

La muestra de la presente investigación encaja en el muestreo de tipo no probabilístico o dirigido, lo que quiere decir que no todos los miembros de la población serán seleccionados, en este tipo de muestreo no se puede calcular estadísticamente el tamaño de la muestra y la elección recae en el investigador para seleccionar los datos. Una de las técnicas del muestreo no probabilístico es el muestreo por conveniencia y consiste en seleccionar las unidades del muestreo más convenientes para el investigador además de ser más accesibles, fáciles de medir y cooperativas; este tipo de muestreo se emplea en estudios exploratorios. [37] [38]

Para esta investigación se toma como muestra las imágenes documentales, este tipo de imágenes no solo se limitan a los humanos también pueden tratar sobre animales o naturaleza, inclusive estas fotografías contienen más detalle y calidad visual convirtiéndose en las más propicias para evaluar los resultados obtenidos al aplicar el modelo matemático propuesto.

3.6. Recolección de información

La galería de imágenes es la información base para aplicar el modelo a matemático y fue recolectada de Internet, estos cuadros mostrarán y evidenciarán el potencial del algoritmo a través de un antes y un después en la restauración de cada una de las imágenes.

Tabla 6. Recolección de la Información.

Preguntas Básicas	Explicación
¿Para qué?	Para alcanzar los objetivos de la Investigación.
¿De qué personas u objetos?	Imágenes fotográficas documentales RGB.
¿Sobre qué aspectos?	Detección de bordes. Propagación de texturas.
¿Quién, Quiénes?	Investigador

¿Cuándo?	6 meses
¿Dónde?	Ambato
¿Cuántas veces?	En base a prueba y error.
¿Qué técnicas de recolección?	Recopilación de imágenes resultantes.
¿Con qué?	Modelo matemático.
¿En qué situación?	Horas dedicadas a la investigación.

Elaborado por: El Investigador.

3.7. Procesamiento de la información y análisis estadístico

El procesamiento de la información y análisis de los resultados se realiza de forma matemática con el objetivo de interpretar la información obtenida al aplicar el algoritmo matemático para restaurar las imágenes, es así como se plantea el cálculo de la eficiencia del modelo matemático propuesto.

3.8.Variables respuesta o resultados alcanzados

Dentro de los resultados alcanzados se pretende:

- Analizar los resultados de acuerdo a los objetivos planteados.
- Interpretar los resultados teniendo como base el marco teórico expuesto en el Capítulo II.
- Determinar la calidad de la imagen resultante una vez procesada por el algoritmo matemático propuesto.
- Comprobar la hipótesis planteada.
- Establecer conclusiones y recomendaciones acorde a los resultados esperados por el investigador.

CAPÍTULO IV

ANÁLISIS E INTERPRETACIÓN DE RESULTADOS

4.1. Modelo matemático propuesto

Con base al marco teórico detallado en el Capítulo II se procede a explicar el modelo matemático diseñado para la restauración de imágenes documentales. En primera instancia se detalla el código fuente del algoritmo, funciones empleadas y métodos aplicados para procesar las imágenes, luego se aplica el modelo matemático a una serie de imágenes para observar los resultados y finalmente se compara el procesamiento, calidad y precisión de la imagen original con la imagen procesada para obtener algunas observaciones del modelo matemático propuesto. De forma sencilla el algoritmo funciona de la siguiente manera:

- Asigna un valor de energía a cada pixel de la imagen.
- Encuentra una ruta de 8 pixeles conectados que contienen la menor energía.
- Elimina todos los pixeles de la ruta encontrada.
- Vuelve a repetir los tres primeros pasos hasta eliminar el objeto seleccionado.

4.1.1. Modelamiento basado en Seam Carving

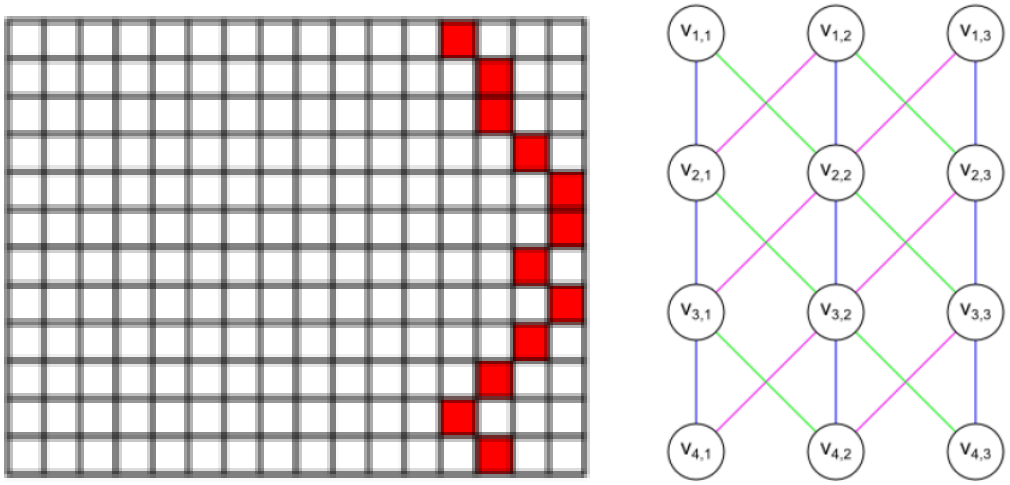
Basándose en la característica principal del modelo expuesto, el tallado de costuras a emplearse se centra en la obtención de un arreglo de valores obtenidos en el barrido vertical de la imagen RGB concatenada (convergencia de los tres colores). De tal manera que pueda obtenerse una costura de menor energía (pixel o pixeles de menor valor entre 0 – 255) y generar la supresión del mismo, sin alterar el tamaño ancho (W) y alto (H) de la imagen en proceso.

4.1.1.1.Proceso de obtención del vector de menor costo o costura mínima

El barrido vertical para obtención de la costura presentado en la figura 20a, obedece al procesamiento matemático a nivel matricial de la matriz RGB concatenada de dimensiones HxW de tal manera que se cumpla con la siguiente ecuación:

$$MatrizCostura(a,b) = \begin{cases} Si\ [valorRGB(h,w) \leq 10 => valorRGB(h,w) \\ Si\ [valorRGB(h,w) > 10 => saltar\ valor \end{cases} \quad (17)$$

De tal manera que, para representar la selección de la costura ideal de la imagen, el proceso (ver figura 20b) obedece al escogimiento de valores cercanos al pixel central bajo análisis (V2,2) en barrido de izquierda a derecha y de arriba hacia abajo.



(a) Generación de costura en imagen RGB (b) Barrido de selección de pixeles de menor valor concatenado

Figura 20. Generación de costura en imagen RGB
Fuente: [39]

Por lo tanto, el modelo matemático que se ajusta a la selección del vector de menor valor costura mínima se puede representar por la siguiente ecuación:

$$P = (V1, j1(V1, j1, V2, j2), V2, j2, \dots, (V[h-1], j[h-1]), Vh, jh)Vh, jh) \quad (18)$$

4.1.1.2.Proceso de enmascaramiento

Una vez obtenida la matriz de eliminación o costura de menor energía, el proceso de supresión ha de modificarse con el afán de precautelar la eliminación parcial o total de objetos que contengan información de mayor relevancia en la imagen. Para el caso en estudio, el proceso de generación de la máscara de protección tomando como punto de partida el modelo matemático que utiliza Seam Sarving para el redimensionamiento de imágenes.

Como punto de partida, la imagen de valores RGB concatenados y que visualmente pueden llegar a lucir como una imagen a escala de grises; es operada aplicando un proceso de derivadas parciales para obtener la convolución con operador de Sobel de tal forma que:

$$Cv(i,j) = \left|\frac{\partial I}{\partial x}\ (i,j)\right| + \left|\frac{\partial I}{\partial i}\ (i,j)\right| \tag{19}$$

Donde:

I representa la imagen RGB concatenada

La ventaja del procesamiento planteado previamente, es que contribuye a la obtención de la función que al aplicarse tanto en los ejes x(alto) e y(ancho) genera una aproximación del gradiente de la imagen bajo análisis que visualmente retorna los cambios representativos de la misma (detección del borde de un objeto particular). Es decir, si se aplicase la operación sobre una imagen 3x3 con valores concatenados RGB como se presenta en el siguiente ejemplo:

$$\frac{\partial I}{\partial x} = I * \begin{vmatrix} \frac{1}{8} & 0 & -\frac{1}{8} \\ \frac{1}{4} & 0 & -\frac{1}{4} \\ \frac{1}{8} & 0 & -\frac{1}{8} \end{vmatrix} \quad \frac{\partial I}{\partial y} = I * \begin{vmatrix} 1/8 & 1/4 & 1/8 \\ 0 & 0 & 0 \\ -1/8 & -1/4 & -1/8 \end{vmatrix}$$

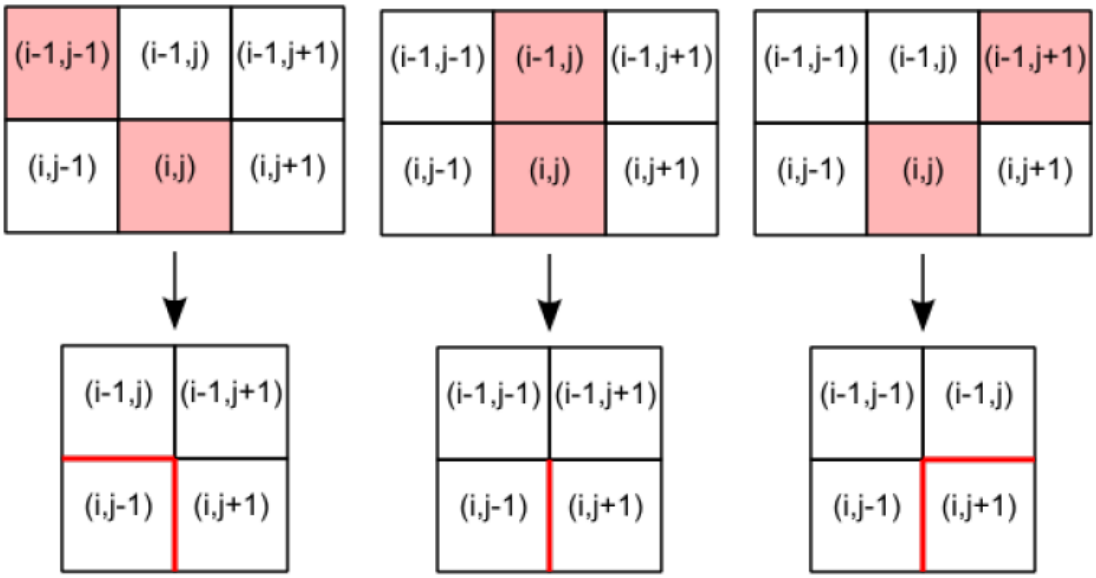
Los resultados obtenidos en el ejemplo planteado, retornaría un valor de la derivada parcial absoluta cuyo valor máximo seria de máximo 2M y valor mínimo –2M. Y, en consecuencia, se puede establecer que:

$$C'v(i,j) = \begin{Bmatrix} -2M, si\ pixel(i,j)\ debe\ ser\ borrada \\ 2M, si\ pixel(i,j)\ debe\ ser\ protegida \\ Cv(i,j)\ restante\ se\ conserva \end{Bmatrix} \tag{20}$$

4.1.1.3.Proceso de remoción de costuras

Al generar el barrido de la imagen para obtener una matriz de valores mínimos o de información irrelevante, en conjunto con la matriz convolucionada de protección que toma como base primordial objetos encerrados en puntos de inflexión o bordes. Seam Carving

genera la remoción de la costura, sustrayendo de la matriz original la matriz de valores mínimos considerando la matriz convolucionada como mascara protectora de información relevante. De esta manera se eliminan los valores de la matriz original y se unen sus vecinos en ausencia del valor suprimido como se muestra en la siguiente figura:



(a) Costura Izquierda (b) Costura central (c) Costura derecha

Figura 21. Proceso de eliminación de costuras y unión de vecinos en barrido vertical
Fuente: [39]

Para el estudio propuesto, se han modificado los procesos primarios del modelo Seam Carving tomando el mismo modelamiento matemático y ha sido modificado para la aplicación sobre una estructura u objeto predeterminado. De esta forma, la generación de la costura es determinada por un enmascaramiento manual cuyo proceso de convolución establece el segmento de la imagen a suprimirse y continuando con el proceso tradicional de supresión de costuras. Finalmente, se ha generado el redimensionamiento de la imagen con el afán de mantener la relación original HxW de la imagen RGB a través de un proceso inverso de supresión de costuras (inverso de la figura 39), utilizando el mismo proceso de generación del vector de menor costo (costura mínima); sin embargo, para el caso de redimensionar en lugar de eliminar dichos pixeles, se introducen valores similares a los de sus vecinos.

4.1.2. Lenguaje de programación

Python es un lenguaje de altísimo nivel orientado a objetos, dinámico y fuerte. Posee una sintaxis clara y concisa que favorece la legibilidad del condigo-fuente, haciendo de este software más productivo. Además, incluye estructuras de alto nivel como: listas, diccionarios, data/hora y soporta programación modular y funcional. Python es un software de código abierto (General Public License [GPL]) inclusive puede ser incorporado en productos propietarios, al ser empleado como lenguaje principal en el desenvolvimiento de sistemas es muy utilizado como lenguaje script en varios softwares permitiendo automatizar tareas y adicionar nuevas funcionalidades como: LibreOffice, PostgreSQL y Blender. [40]

El modelo matemático fue desarrollado específicamente en Python 3, el archivo como tal se puede ejecutar en sistemas Linux, Unix y Windows puesto que el sistema cuenta con el control multiplataforma. Se debe tener en cuenta que Python 3 es incompatible con Python 2, en este último existen bibliotecas antiguas que están en desuso y no compilan en Python3, sin embargo, existen mecanismos para mudar una biblioteca de una versión a otra.

Figura 22. Software Python.
Fuente: [40]

En el área de procesamiento de imágenes existen diferentes scripts, librerías como numpy se emplean para operaciones complejas y manejo numérico, matplotlib permite desarrollar gráficos de calidad y figuras interactivas, scikit-image la cual permite manejar operaciones sencillas y complejas para el tratamiento de imágenes.

4.2. Virtualenv

Es una herramienta para crear entornos de Python aislados, es decir, se crea con directorios de instalación propios que no comparte bibliotecas con otros entornos y tampoco accede a bibliotecas globalmente instaladas. La ventaja de esta herramienta tan simple y útil es tener entornos de trabajo separados, disponer de distintas versiones del interprete con los módulos necesarios para trabajar sin que otros ambientes influyan o afectan en el proceso. [42]

4.2.1. Paquetes de procesamiento matemático

Opencv_python: Es una librería de software abierto y de código libre, esta API fue creada para ser utilizada en el lenguaje de programación Python y puede ser instalada en diferentes sistemas operativos como Windows, MAC OS y Linux. Esta librería se complementa con NumPy que es una librería especializada en operaciones numéricas con una sintaxis similar a MATLAB. Las estructuras tipo matriz de OpencCV se transforman en matrices NumPy lo que permite que la integración con otras librerías como SciPy y Matplotlib muy usadas en el tratamiento de imágenes sea más fácil. [43]

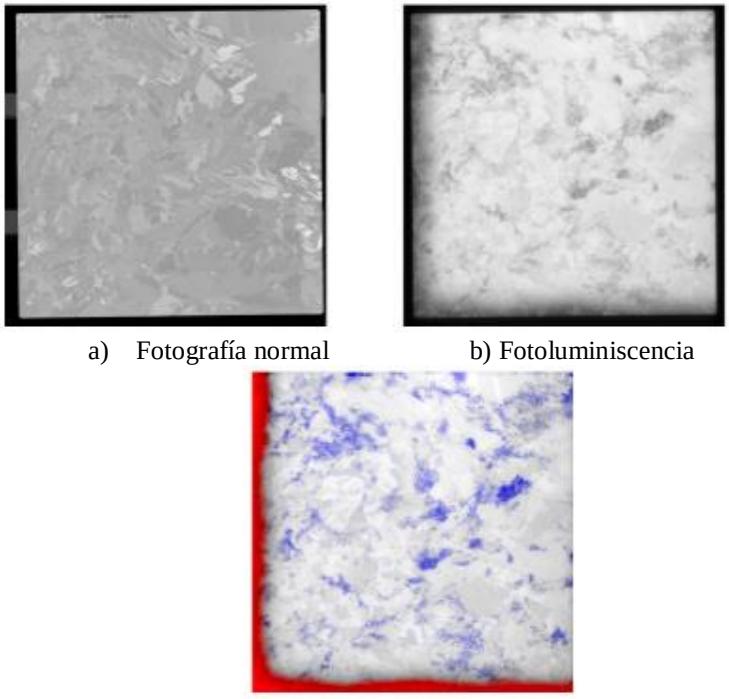
Tensorflow: Librearía ampliamente utilizada para Machine Learning y Deep Learning, esta herramienta es OpenSource desarrollada por Google Brain y liberada en 2015 enfocada principalmente a Python. Tensor Flow trabaja con grafos computacionales, un grafo es un conjunto de operaciones que representan el modelo a estudiar, los nodos (conjunto de grafos) representan diagramas de operaciones matemáticas y los bordes del grafo son el conjunto de datos multidimensionales llamados tensores (elementos que dan nombre a esta librería). Los bordes indican los datos entrantes o salientes de un nodo.

Un tensor es un concepto matemático complejo que engloba objetos geométricos y describen relaciones entre tensores geométricos, escalares y otros tensores, se puede definir como un arreglo de n dimensiones, así cada nodo del grafo tomará como entrada 0, 1 o más tensores y dará como resultado a la salida un tensor que será pasado a otros nodos. [44]

Numpy: Abreviatura de Numerical Python, es un paquete empleado para el procesamiento numérico en Python y ofrece algunas funciones para el procesamiento de imágenes, es decir, puede convertir las imágenes en matrices y arreglos. Entre los recursos que tiene esta librería se tiene: funciones para efectuar procesamientos entre arrays, herramientas para leer y grabar datos basados en arrays, operaciones de algebra lineal, transformadas de Fourier, generación de números aleatorios, una API C madura que permite que lenguajes de más bajo nivel C, C++ o Fortran acceden a estructuras de datos y recursos de procesamiento de numpy además posee funcionalidades semejantes a Matlab. [44]

Scikit-image: Es una colección de algoritmos desarrollada para el procesamiento de imágenes, trabaja con arrays de numpy y también hace uso de algunas utilidades gráficas de matplotlib; esta librería proporciona una API bien documentada en el lenguaje de programación Python y está desarrollada por un equipo activo e internacional de colaboradores, se encuentra disponible de forma gratuita y sin restricciones. Uno de los objetivos principales de scikit-image o skimage es facilitar la experiencia del usuario especialmente de aquellos que ya están familiarizados con las herramientas de Python, para procesar una imagen basta con cargar la imagen desde un disco o se puede usar una imagen de prueba de scikit-image, luego se aplica uno o más filtro de imagen y se obtienen los resultados de manera rápida.

El paquete está disponible para instalar en todas las plataformas como BSD, GNU / Linux, OS X, Windows, A continuación, se expone un ejemplo de aplicación de esta librería. En la fabricación de celdas solares las obleas deben ser de buena calidad, empleando scikit-image se puede extraer características de las obleas de silicio que no son visibles en condiciones de visualización estándar. La figura 23a muestra una imagen óptica de la oblea de silicio, la figura 23b muestra la misma imagen oblea usando fotoluminiscencia (PL) y la figura 23c muestra los defectos e impurezas mediante la aplicación de scikit-imagen.



a) Fotografía normal b) Fotoluminiscencia

c) Imagen procesada

Figura 23. Uso de scikit-image para estudiar impurezas de obleas de silicio. (a) Una imagen de oblea de silicio antes de ser procesada para celdas solares. (b) Una imagen PL de la misma oblea con regiones oscuras que representa un impacto negativo en la eficiencia de las celdas solares. (c) Imagen procesada, el color azul representa los defectos en el crecimiento de los cristales y el color rojo indica las impurezas.
Fuente: [40]

Scikit-imagen procesa la imagen aplicando una transformación Hough (transform.hough_line) para encontrar los bordes de la oblea, luego emplea un filtro pasa banda con una diferencia gaussiana (filter.gaussian_filter) para detectar los defectos cristalinos de la oblea. De esta manera los fabricantes de celdas solares pueden detectar y rechazar obleas de baja calidad para aumentar las celdas solares con alta eficiencia de conversión solar. [40]

4.3. Importaciones necesarias para el modelo matemático

El código planteado para el modelo matemático requiere las siguientes importaciones para el correcto funcionamiento. [41]

- **import os:** módulo de SO para interactuar con el sistema operativo del computador.
- **import numpy:** librería para trabajar con matrices.

- **import glob:** modulo para buscar una lista de archivos en el sistema de archivos con nombres que coinciden con un patrón.
- **import jit:** librería de numpy, hace que el código de Python compile más rápido.
- **import ndimage:** librería de scipy y sirve para el procesamiento y análisis de imágenes.
- **matplotlib:** es una librería de trazado 2D.
- **from PIL import Image:** instancia que sirve para recuperar el tamaño de la imagen.

```
import os
import numpy as np
from cv2 import *
from glob import glob as files
from pyfiglet import Figlet
from numba import jit
from scipy import ndimage as ndi
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
from PIL import Image
```

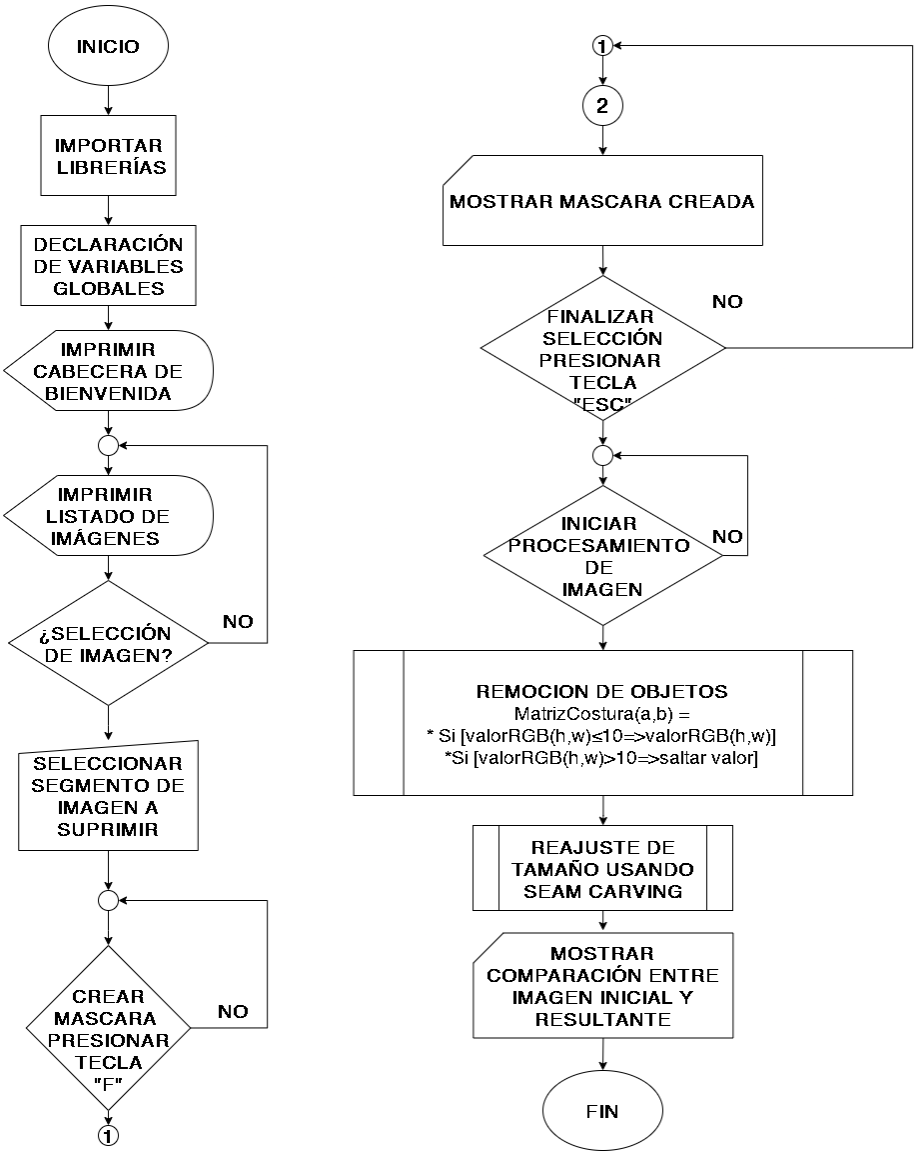
Figura 24. Importación de librerías
Elaborado por: El Investigador.

4.4. Estructura matemática del modelo

El modelo matemático cuenta en su código fuente con una serie de métodos necesarios e indispensables para ejecutar el programa en su totalidad. A continuación, se listas las funciones empleadas en el algoritmo del modelo.

4.4.1. Definir estructura del funcionamiento en base al diagrama de flujo

El siguiente diagrama de flujo muestra brevemente el funcionamiento del modelo matemático.


INICIO
IMPORTAR LIBRERÍAS
DECLARACIÓN DE VARIABLES GLOBALES
IMPRIMIR CABECERA DE BIENVENIDA
IMPRIMIR LISTADO DE IMÁGENES
¿SELECCIÓN DE IMAGEN?
NO
SELECCIONAR SEGMENTO DE IMAGEN A SUPRIMIR
CREAR MASCARA PRESIONAR TECLA "F"
NO
1
1
2
MOSTRAR MASCARA CREADA
FINALIZAR SELECCIÓN PRESIONAR TECLA "ESC"
NO
INICIAR PROCESAMIENTO DE IMAGEN
NO
REMOCION DE OBJETOS
MatrizCostura(a,b) =
* Si [valorRGB(h,w)≤10=>valorRGB(h,w)]
*Si [valorRGB(h,w)>10=>saltar valor]
REAJUSTE DE TAMAÑO USANDO SEAM CARVING
MOSTRAR COMPARACIÓN ENTRE IMAGEN INICIAL Y RESULTANTE
FIN

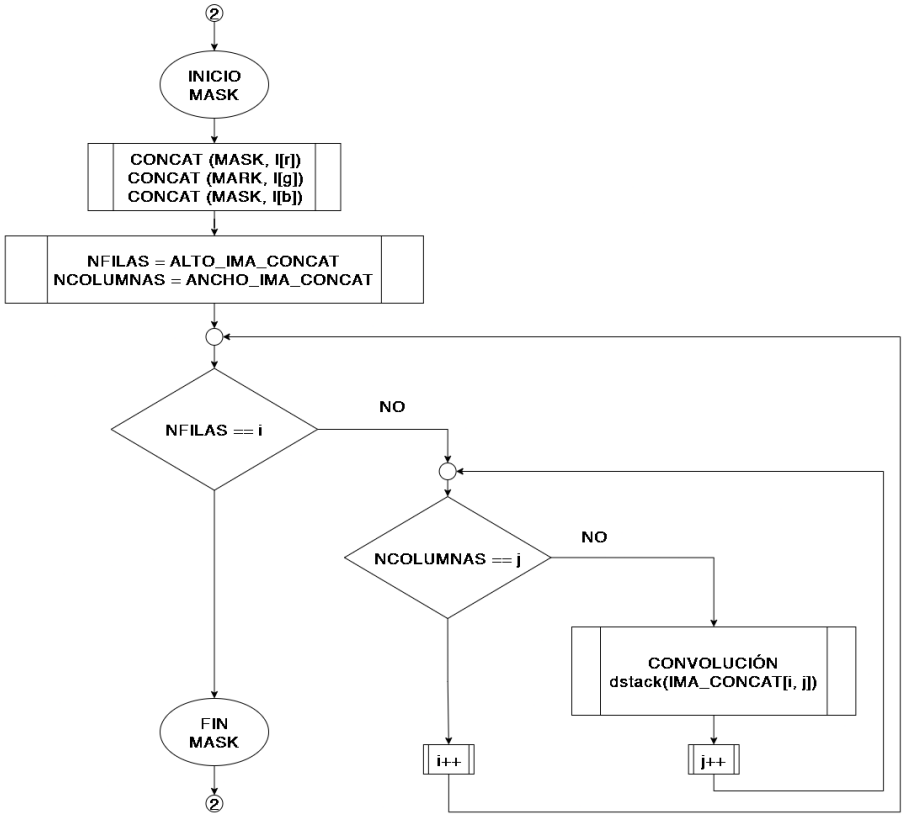


Figura 25. Diagrama de flujo del algoritmo.
Elaborado por: El Investigador.

4.5 Detección de Bordes

En la técnica de detección se debe especificar el tipo de imagen que se va a procesar porque el modelado para tratar imágenes en escala de grises y RGB es distinto, en este caso las imágenes a ser restauradas son RGB y el proceso es más complejo. Por ejemplo, cuando una imagen está en escala de grises la energía de un pixel es más pequeña puesto que sus vecinos tienen colores similares, pero en una imagen RGB la energía de un pixel es más grande porque sus vecinos tienen colores muy diferentes.

Cuando se aplica el tallado de costuras el hecho de eliminar pixeles implica analizar que el pixel actual está tocando con pixeles que antes no estaban en esa posición y da lugar a nuevos bordes formados al conectar con otras costuras.

En los siguientes aparatados se muestra el proceso para calcular la energía de un pixel, las líneas de código mostradas en la imagen 24 detalla el proceso para extraer las dimensiones de la imagen a través de un arreglo de matrices, luego crea una ventana que será la máscara protectora en donde se procesará la imagen original con el fin obtener el mismo tamaño al finalizar el proceso de restauración.

```
# Metodo de deteccion de bordes
text_box = np.zeros((sizeBlank, 2*image.shape[1] + sizeBlank, 3)) + 1.
empty = np.zeros(image.shape)
blank = np.zeros((image.shape[0], sizeBlank, 3)) + 1
namedWindow("Deep Object Removal", WINDOW_NORMAL)
setMouseCallback('Deep Object Removal', mouse_callback)
createTrackbar('Pen Size', 'Deep Object Removal', 1, 50, lambda x: x)
filtered_image = empty
```

Figura 26. Método detección de bordes.
Elaborado por: El Investigador.

A continuación, se detalla paso por paso cómo funciona el método de detección de bordes:

- En primer lugar, se genera una Matriz de 1"s para dividir las imágenes a mostrar, la línea de código que ejecuta esta acción es:

  *text_box = np.zeros((sizeBlank, 2*image.shape[1] + sizeBlank, 3)) + 1.*

- Luego, se genera una matriz de 0's de iguales dimensiones de la imagen a procesar con la función *empty*.
- Se genera una matriz de 1's para generar la máscara de imagen, la línea de código que ejecuta esta acción es:

  blank = np.zeros((image.shape[0], sizeBlank, 3)) + 1

- Se declara el nombre de la ventana para la creación de la máscara a través de *namedWindow("Deep Object Removal", WINDOW_NORMAL).*
- Se adjunta la detección del movimiento de mouse en la ventana para dibujar por medio de *setMouseCallback('Deep Object Removal', mouse_callback).*
- Se establece una barra para elegir el tamaño del pincel con *createTrackbar('Pen Size', 'Deep Object Removal', 1, 50, lambda x: x).*
- Finalmente, se inicializa la máscara con *filtered_image = empty.*

4.6. Propagación de texturas

En la reconstrucción de imágenes se puede emplear algunos métodos para llegar a obtener la imagen ideal, el algoritmo propuesto busca mantener la misma dimensión de la imagen para lo cual, se retira el objeto no deseado en la imagen y se inserta en su lugar y de manera matemática muestras de pixeles cercanos, por ende, las texturas vecinas mantendrán un balance y armonía en el resultado de la imagen final de tal manera que no sea perceptible que en ese lugar existió el objeto borrado.

La propagación de texturas se lleva a cabo con el método de enmascaramiento que se detalla en los siguientes apartados, el primer paso para aplicar la propagación de texturas es en primer lugar seleccionar y retirar el objeto a eliminar seguido de la inspección de texturas vecinas, las líneas de condigo que se muestran en la imagen 25 muestra el método creado para realizar este proceso.

```
# Metodo para remover objetos de la imagen
def object_removal(im, rmask, mask=None, vis=True, horizontal_removal=False):
    im = im.astype(np.float64) # Se convierte la matriz de imagen de enteros a flotantes
    rmask = rmask.astype(np.float64) # Se convierte la matriz de mascara de enteros a flotantes
    if mask is not None:
        mask = mask.astype(np.float64)
    output = im # Se pasan los valores de la imagen original a de salida

    h, w = im.shape[:2] # Se optiene las dimenciones de imagen

    if horizontal_removal: # Ciclo para procesar imagen si se rota
    # Se inicia cliclo para el procesado
    while len(np.where(rmask > MASK_THRESHOLD)[0]) > 0:
    # Se consulta la dimencion de la imagen para hacer el relleno de imagen
    num_add = (h if horizontal_removal else w) - output.shape[1]
    # Se realiza la insercion de contenido
    output, mask = seams_insertion(output, num_add, mask, vis, rot=horizontal_removal)
    if horizontal_removal:
        output = rotate_image(output, False)
    # Se retorna la imagen ya procesada
    return output
```

Figura 27. Inserción de texturas.
Elaborado por: El Investigador.

4.7. Función de Seam Carving

El método Sean Carving conocido también como tallado de costura, es un algoritmo de redimensionamiento de imágenes que procura preservar su contenido. Este método consiste en reducir el tamaño de una imagen, sin alterar la calidad, para lo cual emplea una costura que es una ruta óptima de 8 pixeles conectados de arriba hacia abajo o de izquierda a derecha en una única imagen, donde la ruta está definida por una función de energía de la imagen.

Para cambiar el tamaño de las imágenes según su contenido se debe encontrar una costura, camino o serie de menor relevancia en la imagen para eliminarlo reduciendo las dimensiones sin alterarla visualmente, también se puede emplear para mejorar el contenido de una imagen. Este método también puede ser empleado de forma manual, es decir, el usuario puede seleccionar manualmente el objeto o las áreas que desea retirar de la imagen.

Ahora, una imagen puede ser modificada a través de un recorte a escala, pero existen desventajas asociadas a este proceso en donde las dos partes al ser unidas no encajarán por completo, siempre abra una línea visible en la imagen que no coincida visualmente con la armonía del cuadro. A manera de ejemplo se muestra la imagen original de un surfista y a continuación se visualiza la misma imagen recortada con un programa existen en el computador. [45]

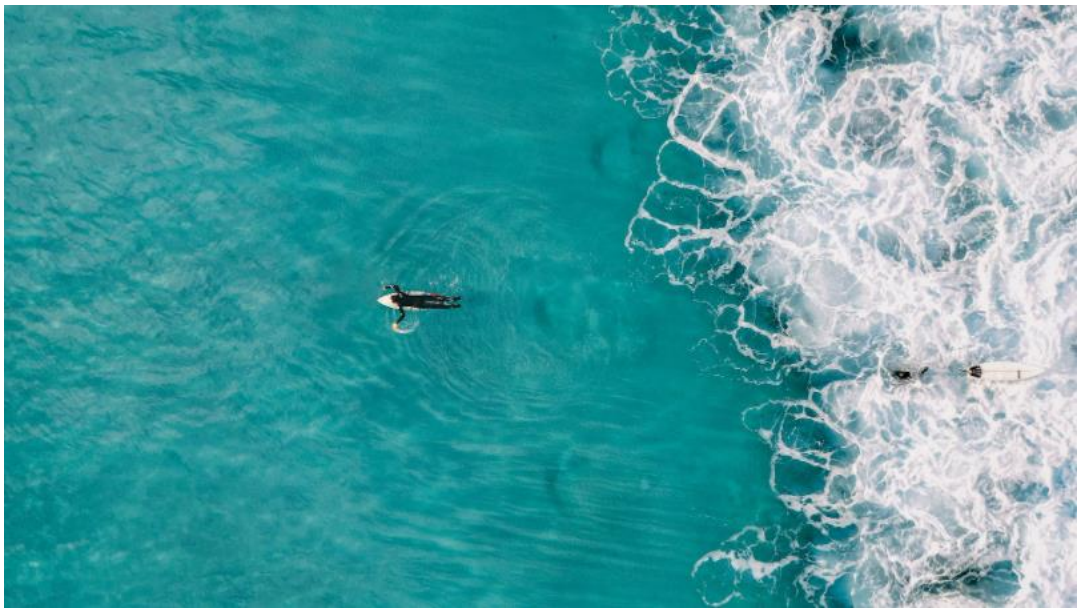


Figura 28. Imagen original de un surfista.
Fuente: [45]

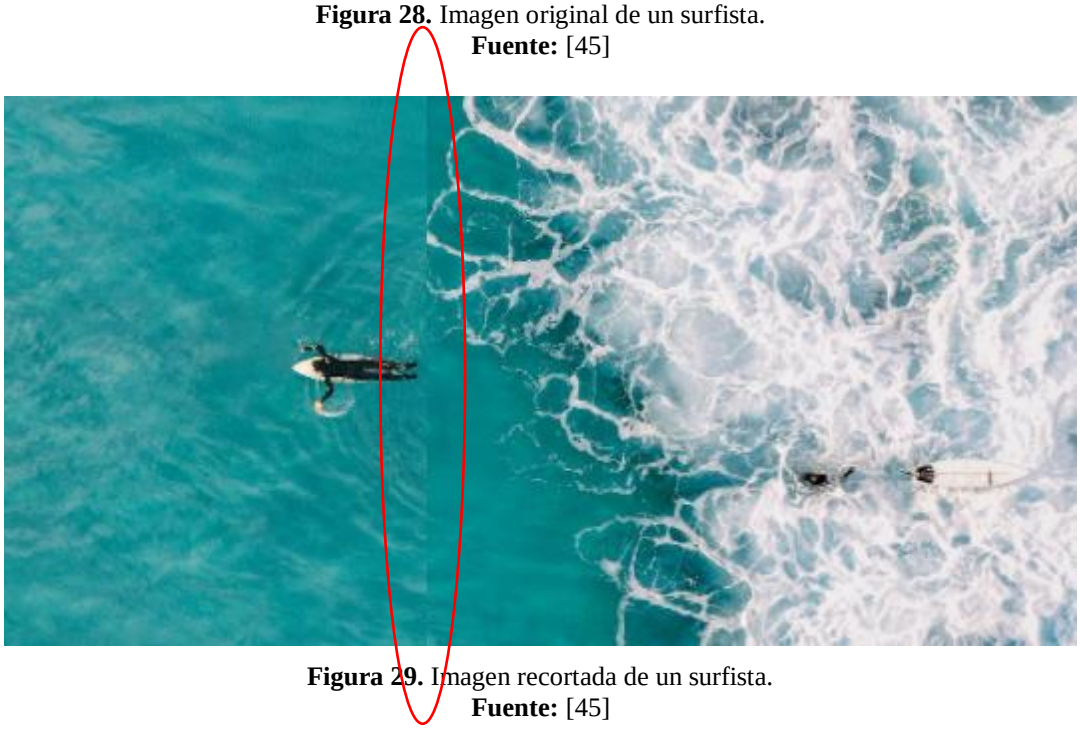


Figura 29. Imagen recortada de un surfista.
Fuente: [45]

En esta última imagen se puede observar una línea visible en la imagen que la hace visualmente poco aceptable. El método Seam Carving pretende reducir el tamaño de una imagen de manera inteligente, volviendo a la imagen del surfista, se puede observar en la figura 30 como el método detecta la costura de menor energía que en este caso pasa por el centro de la imagen donde el agua está más tranquila.

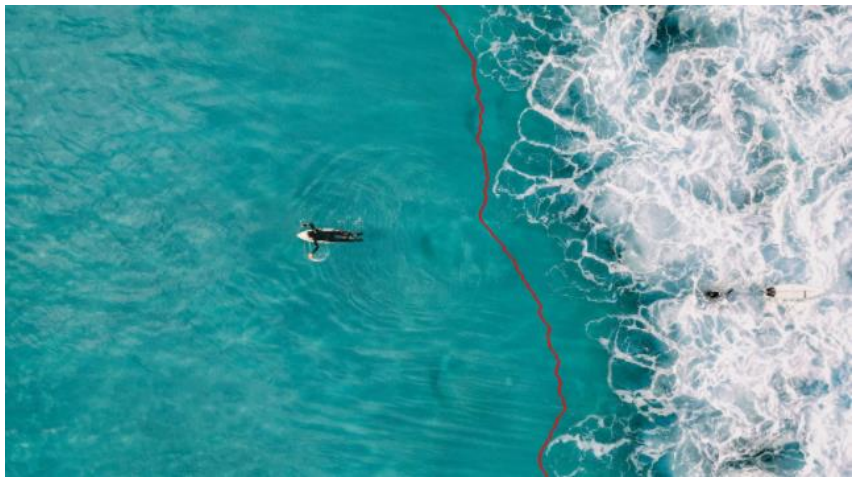


Figura 30. Costura vertical de la imagen.
Fuente: [45]

Una vez identificada la costura de menor energía se procede a eliminar dichos pixeles para reducir el tamaño de la imagen y visualmente sea aceptable, este proceso se puede hacer hasta que el usuario quede satisfecho con el resultado.

Figura 31. Imagen reconstruida del surfista.
Fuente: [45]

Para obtener los resultados anteriores en el código fuente del modelo matemático se aplicó las siguientes líneas de comando.

```
########################################
# Funciones de Ayuda SEAM CARVING
########################################

@jit
def add_seam(im, seam_idx):
    h, w = im.shape[:2]
    output = np.zeros((h, w + 1, 3))
    for row in range(h):
        col = seam_idx[row]
        for ch in range(3):
            if col == 0:
            else:

    return output

@jit
def add_seam_grayscale(im, seam_idx):

@jit
def remove_seam(im, boolmask):

@jit
def remove_seam_grayscale(im, boolmask):

@jit
def get_minimum_seam(im, mask=None, remove_mask=None):
```

Figura 32. Método Seam Carving.
Elaborado por: El Investigador.

A continuación, se describe la función de cada objeto en el código fuente:

- *add_seam* es la herramienta para redimensionar imágenes a una dimensión mayor o menor utilizando el tallado de costuras, utiliza una combinación de la energía de gradiente para determinar la costura menos importante.
- *add_seam_grayscale* convierte la imagen con canales RGB en una imagen con un solo canal de escala de grises.
- *remove_seam* modifica la imagen y en su lugar devuelve un corte W-1(ancho)×h×3 (alto) para eliminar la costura.
- *remove_seam_grayscale* elimina la imagen en escala de grises.
- *get_minimum_seam* obtiene la energía mínima acumulada.

4.7.1. Función de energía

Para obtener la función de energía en Python se emplea *backward_energy* y *forward_energy*. *backward_energy* es un concepto que define los pixeles adyacentes antes de la eliminación de la costura mientras que *forward_energy* predice qué píxeles estarán adyacentes después de la eliminación de una costura y la utiliza para sugerir la mejor costura a eliminar.

El denominado *forward_energy* considera la energía de una imagen después de quitar una costura, en lugar de la energía actual de la imagen. Esta sencilla modificación del algoritmo de tallado de costuras original da como resultado un cambio de tamaño de imagen más natural considerando el contenido. Además, este tipo de energía conserva mejor las líneas rectas y cambia el tamaño de diferentes partes de la imagen de manera uniforme

```
########################################
# Funciones de Energia
########################################

def backward_energy(im):
    #obtiene la convolucion de la imagen en cada eje para decolorar la imagen COLOR > B/N
    xgrad = ndi.convolve1d(im, np.array([1, 0, -1]), axis=1, mode='wrap')
    ygrad = ndi.convolve1d(im, np.array([1, 0, -1]), axis=0, mode='wrap')
    #generar una imagen B/N
    grad_mag = np.sqrt(np.sum(xgrad**2, axis=2) + np.sum(ygrad**2, axis=2))

    return grad_mag

@jit
def forward_energy(im):
    #obneter las dimensiones de la imagen bajo procesamiento
    h, w = im.shape[:2]
    im = cv2.cvtColor(im.astype(np.uint8), cv2.COLOR_BGR2GRAY).astype(np.float64)

    #creacion de matriz nula de para procesamiento de energia
    energy = np.zeros((h, w))
    m = np.zeros((h, w))

    # Aplanar la imagen bajo procesamiento
    # Procesado de aplanado consiste en mover la imagen(matriz dimensional) de ancho(w) x alto(h)
    #   en un vector unidimensional con el afán de reducir el tiempo de procesamiento
    #   para ello se utiliza el proceso NP.ROLL para rotar la imager eje por eje U, L y una complementaria
    #   R de apoyo al procesamiento.
    U = np.roll(im, 1, axis=0)
    L = np.roll(im, 1, axis=1)
    R = np.roll(im, -1, axis=1)
```

Figura 33. Función de energía.
Elaborado por: El Investigador.

Aplicando estos conceptos al algoritmo se tiene el mapa energético de la imagen para eliminar la costura de menor energía calculada a través del método del gradiente.

Figura 34. Función de energía.
Elaborado por: El Investigador.

4.7.2. Máscara protectora

El enmascaramiento protector proporciona una máscara binaria al algoritmo de eliminación, las áreas blancas de la máscara, como se muestra en la imagen 35, pueden recibir grandes valores de energía positiva o negativa en el mapa de energía que disuade fuertemente en construcción de la costura óptima. El programa mostrará en pantalla, la imagen original con el contorno marcado en color naranja del objeto a ser eliminado y adjunto se muestra la máscara protectora con la silueta del objeto seleccionado (área blanca) por el mouse.

Figura 35. Máscara protectora.
Elaborado por: El Investigador.

Dentro del código el comando que permite crear el enmascaramiento es *mask* y representa el camino a la máscara protectora, debe ser binaria y tener el mismo tamaño que la imagen de entrada. La creación de la máscara protectora procura que el algoritmo aplique la detección de bordes y propagación de texturas con la finalidad de rellenar el vacío de los pixeles con menor energía que serán eliminados y la imagen mantenga su dimensión original, textura inicial y no pierda detalle en los bordes.

```
# Metodo para crear mascara de imagen
def masking(image):
    # Se buscan todos los puntos de color azul
    mask = (np.array(image[:,:,0]) == 0.0)
    # Se buscan todos los puntos de color verde
    mask = mask & (np.array(image[:,:,1]) == 0.49)
    # Se buscan todos los puntos de color rojo
    mask = mask & (np.array(image[:,:,2]) == 1.0)
    # Se concatenan todos los puntos encontrados
    mask = np.dstack([mask, mask, mask]);
    #Se invierte la matriz concatenada de 0's a 1's
    return (False ^ mask) * (np.array(empty) + 1)
```

Figura 36. Enmascaramiento.
Elaborado por: El Investigador.

4.7.2. Selección de objetos

Dentro del código se adicionó la selección de objetos a través del mouse para lo cual se debe reconocer el movimiento del mismo partiendo desde el pulso que se da al botón izquierdo del mouse seguido del movimiento realizado para trazar las líneas al dibujar el contorno del objeto a ser removido, posteriormente se detecta cuando el botón izquierdo ha sido dejado de presionar

```
#  Metodo para deteccion del movimiento del mouse
def mouse_callback(mouse_event, x, y, flags, parameters):
    # Se llama a las variables globales
    global _x, _y, isDrawn

    if mouse_event == EVENT_LBUTTONDOWN: # Se detecta el pulso del boton izquierdo del mouse
        isDrawn = True
        _x, _y = x, y

    elif mouse_event == EVENT_MOUSEMOVE: # Se Detecta los movimientos del mouse para dibujar lineas
        if isDrawn:
            line(image, (_x, _y), (x, y), (0.0, 0.49, 1.0), stroke_size) # Se dibuja la linea de color Naranja
            _x, _y = x, y

    elif mouse_event == EVENT_LBUTTONUP: # Se detecta el soltar el boton izquierdo del mouse
        isDrawn = False
        line(image, (_x, _y), (x, y), (0.0, 0.49, 1.0), stroke_size)
```

Figura 37. Movimiento del mouse.
Elaborado por: El Investigador.

4.7.3. Costura de menor energía

El primer paso para encontrar la costura óptima es recorrer la imagen de izquierda a derecha o de arriba hacia abajo y calcular la energía mínima acumulada para todas las posibles costuras conectadas. La energía mínima de cada pixel en una imagen es determinada por el mapa de energía y determina qué costura eliminar, el valor de una costura es la suma de los valores de energía a lo largo de los píxeles en la costura. La imagen 36 muestra la costura encontrada a través del mapa de energía, la misma que será eliminada para obtener la restauración esperada en la imagen resultante.

Figura 38. Costura de menor energía.
Elaborado por: El Investigador.

4.7.4. Eliminación de objetos

Una técnica empleada para la eliminación de objetos es seleccionar las regiones, al eliminar repetidamente las costuras hasta que la máscara de eliminación este vacía se eliminara el objeto seleccionado en la imagen, para lo cual se emplea una máscara protectora en donde se ejecutara el proceso matemático. Al ejecutar la eliminación de costuras verticales se reducirá y ancho de la imagen, pero el método de la máscara protectora insertará pixel para mantener la misma dimensión de la imagen original.

```python
# Metodo para remover objetos de la imagen
def object_removal(im, rmask, mask=None, vis=True, horizontal_removal=False):
    im = im.astype(np.float64) # Se convierte la matriz de imagen de enteros a flotantes
    rmask = rmask.astype(np.float64) # Se convierte la matriz de mascara de enteros a flotantes
    if mask is not None:
        mask = mask.astype(np.float64)
    output = im # Se pasan los valores de la imagen original a de salida

    h, w = im.shape[:2] # Se optiene las dimenciones de imagen

    if horizontal_removal: # Ciclo para procesar imagen si se rota
        output = rotate_image(output, True)
        rmask = rotate_image(rmask, True)
        if mask is not None:
            mask = rotate_image(mask, True)
    # Se inicia cliclo para el procesado
    while len(np.where(rmask > MASK_THRESHOLD)[0]) > 0:
        seam_idx, boolmask = get_minimum_seam(output, mask, rmask) # Se llama al metodo de minimum_seam
        if vis: # Si se encuentra habilitado se muestra el procesado de remover objeto
            visualize(output, boolmask, rotate=horizontal_removal)  #Se llama el metodo para visualizar
        output = remove_seam(output, boolmask) # Se llama al metodo remove_seam
        rmask = remove_seam_grayscale(rmask, boolmask)
        if mask is not None: # Si se tiene mascara se procesara en escala de grises
            mask = remove_seam_grayscale(mask, boolmask)
```

Figura 39. Eliminación de costuras.
Elaborado por: El Investigador.

A continuación, se describe la función de cada objeto en el código fuente:

- *vis* si está presente, muestra una ventana mientras se ejecuta el algoritmo que muestra las costuras a medida que se eliminan.
- *rmask* el camino a la máscara de eliminación siendo las áreas en color naranja las regiones que se eliminarán.

La eliminación de costuras una tras otra arrojará como resultado la imagen esperada como se puede observar en las siguientes figuras

Figura 40. Eliminación de los pixeles de menor energía
Elaborado por: El Investigador.

Figura 41. Proceso de eliminación de seam vertical.
Elaborado por: El Investigador.

Figura 42. Eliminación total de la región seleccionada.
Elaborado por: El Investigador.

Figura 43. Eliminación de costuras y reajuste de bordes y texturas en la imagen.
Elaborado por: El Investigador.

4.7.5. Resultado final

La imagen 44a corresponde a una imagen documental de una persona en las montañas, en esta imagen se requiere eliminar a la persona para obtener solo los detalles del paisaje, por lo tanto, se selecciona la silueta de la persona para proceder con la restauración. El resultado final será solo el paisaje, sin el objeto seleccionado obteniendo así el resultado esperado y aceptable para el investigador como muestra la imagen 44b.

a) b)

Figura 44. Restauración aplicando el modelo matemático.
Elaborado por: El Investigador.

4.7.6. Limitaciones

Una de las limitaciones de este modelo matemático es que al eliminar de forma excesiva la costura hará que los objetos, rostros humanos o demás detalles que conforman la imagen sufran notables cambios y la imagen resultante presente distorsiones notables y poco satisfactorias al observador. La siguiente secuencia de imágenes muestra un intento fallido del modelo matemático aplicado.

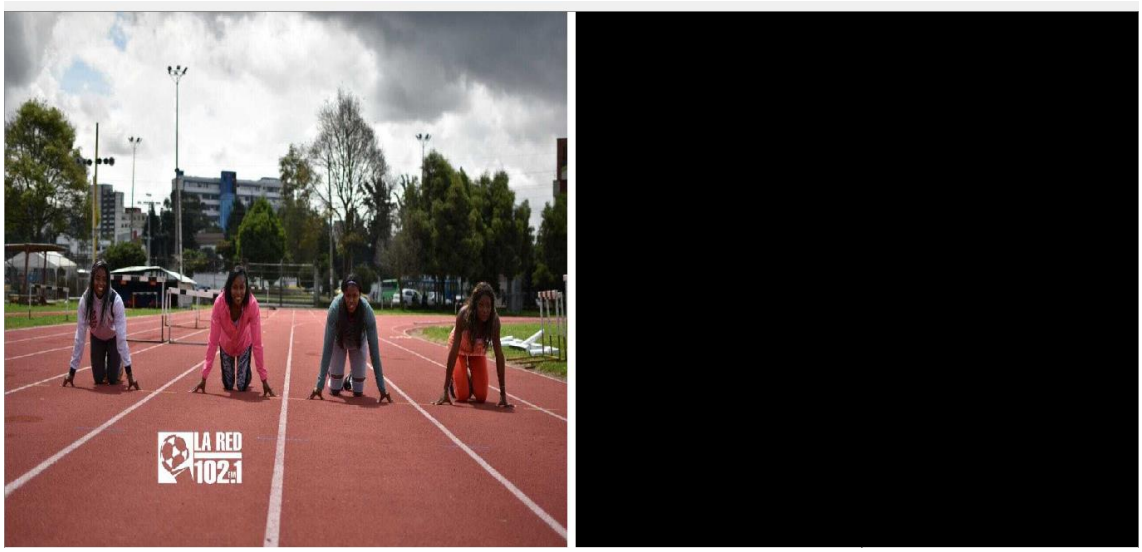


Figura 45. Secuencia 1-Función de energía.
Elaborado por: El Investigador.

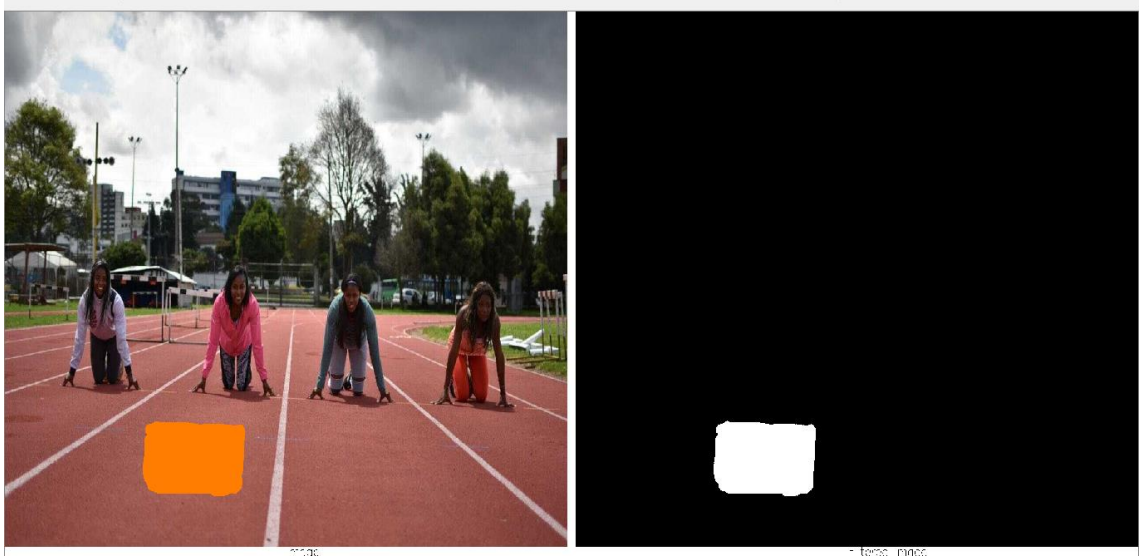
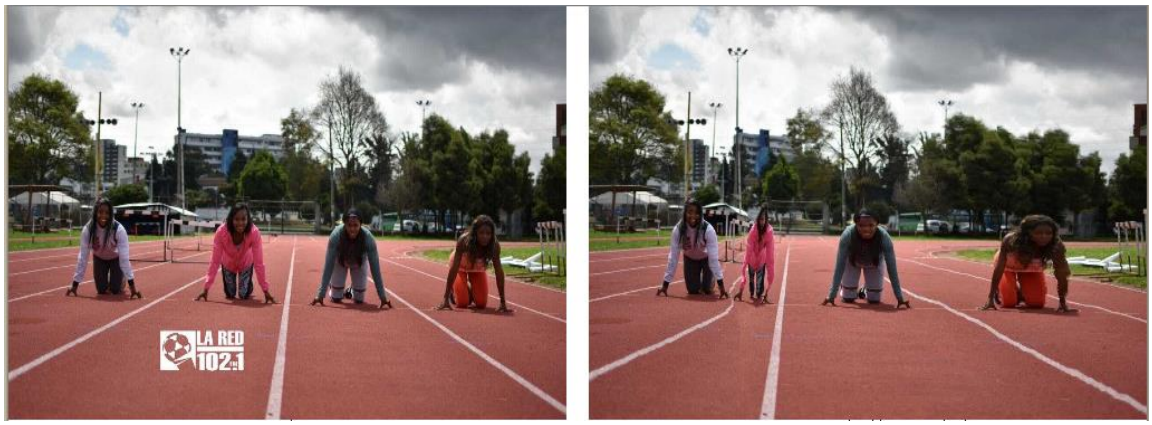


Figura 46. Secuencia 2-Máscara protectora.
Elaborado por: El Investigador.

Figura 47. Secuencia 3-Costura vertical.
Elaborado por: El Investigador.

Figura 48. Secuencia 4-Eliminación de costura.
Elaborado por: El Investigador.

Figura 49. Secuencia 5-Eliminación de costura.
Elaborado por: El Investigador.



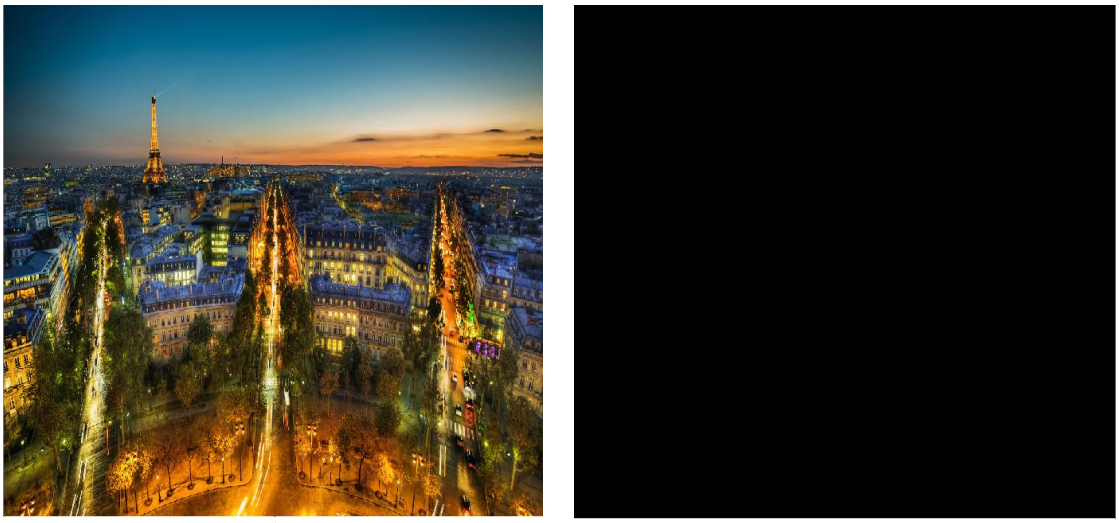
a) b)

Figura 50. Resultado final. a) Imagen original. b) Imagen resultante.
Elaborado por: El Investigador.

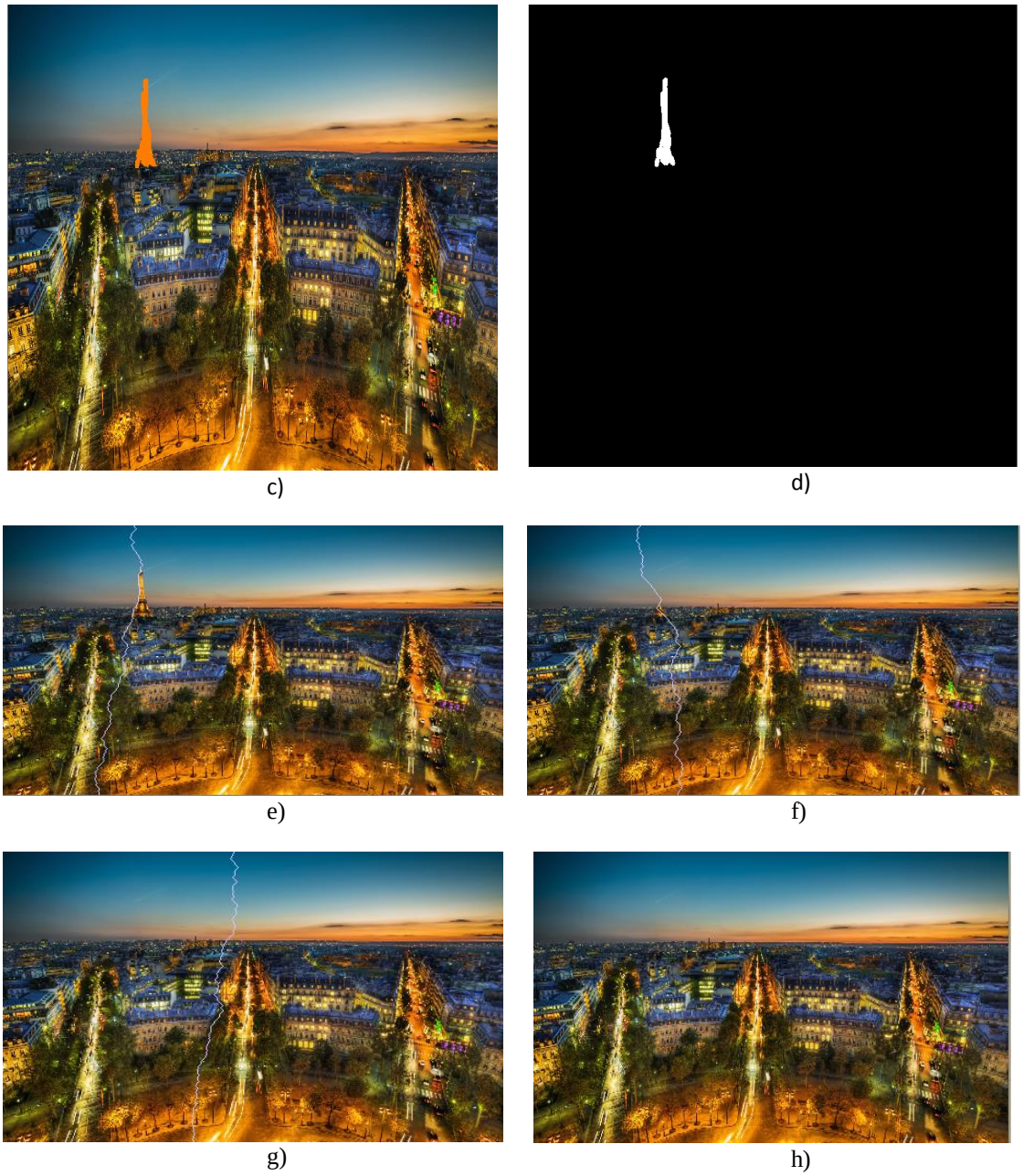
4.8. Análisis

4.8.1. Imágenes restauradas a través de modelo matemático propuesto

La siguiente galería de imágenes muestra el funcionamiento del modelo matemático, donde se indica paso a paso la ejecución del algoritmo para pasar de una imagen que requiere restauración a obtener otra que sea visiblemente aceptable para el observador.



a) b)



c) d)

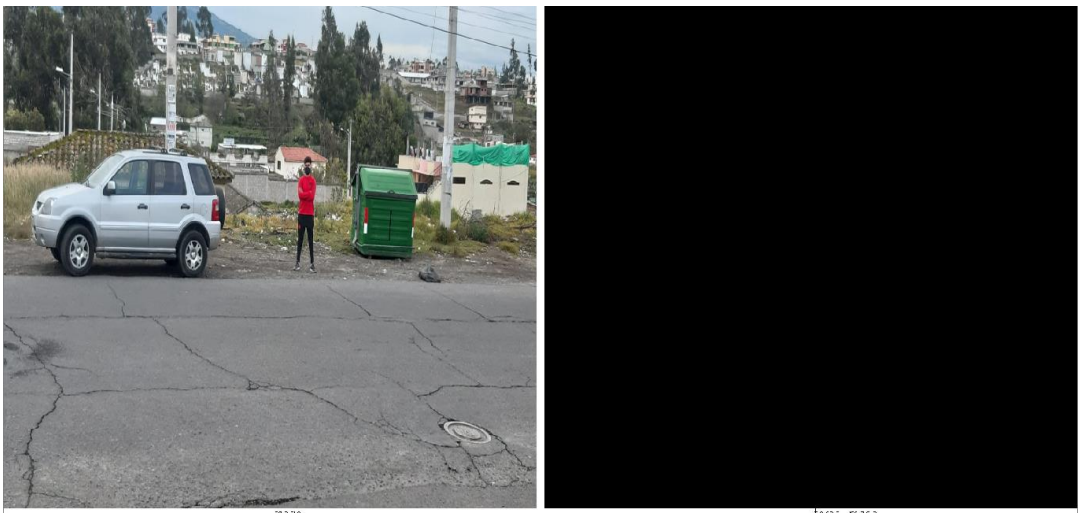
e) f)

g) h)

Figura 51. a) Imagen original. b) Iteración. c) Iteración. d) Iteración. e) Iteración f) Iteración. g) Iteración. h) Imagen resultante.
Elaborado por: El Investigador.

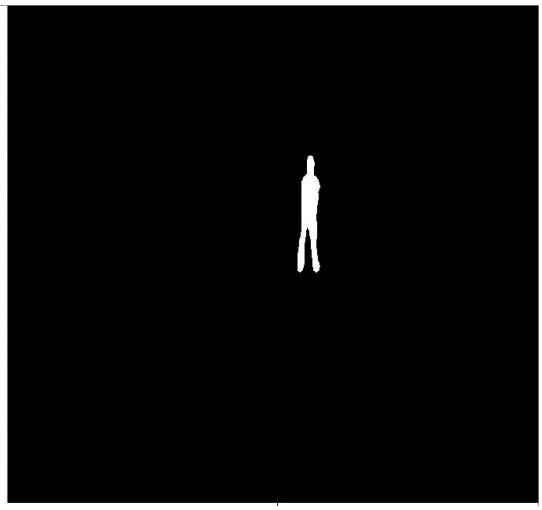
La siguiente galería de imágenes muestra más resultados del modelo matemático en aplicado en otro tipo de imágenes documentales que muestran la vida diaria de las personas en diferentes locaciones que es una de las características de este tipo de imágenes.

La primera imagen escogida es la fotografía de una persona en una zona urbana de la ciudad de Ambato, dentro de la imagen existen objetos sobresalientes como el vehículo, un contenedor de basura y una persona. El objeto a remover es la persona para lo cual el primer paso es obtener la energía de los pixeles de la imagen conocida como máscara protectora (figura 52b), con el puntero del mouse se traza líneas de color naranja para este ejemplo a manera de capa sobre el objeto a remover (figura 52c), dando como resultado que sobre la máscara de protección se cree una silueta en color blanco del objeto seleccionada (figura 52d). En este caso la silueta viene a representar los pixeles que se les otorgó menor energía con la finalidad del crear el mapa de energía de la imagen original, una vez que el algoritmo sabe que pixeles debe quitar y cuales debe proteger se procede con la creación de las costuras verticales (figura 52e), al trazar el camino de las costuras a eliminar el algoritmo procesara las *n* iteraciones de eliminación de costuras (figura 52f, 52g) hasta remover por completo el objeto seleccionada. Internamente el modelo matemático calcula la energía de los pixeles para concatenar los nuevos pixeles vecinos considerando la detección de bordes y propagación de texturas para así obtener la imagen restaurada (figura 52h).

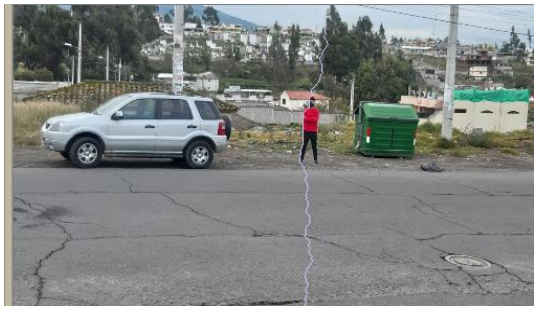


a) b)

c)

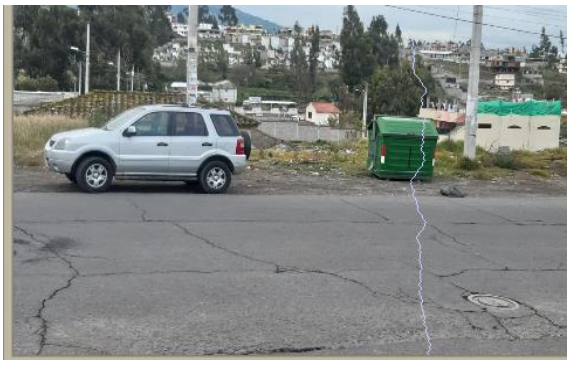


d)

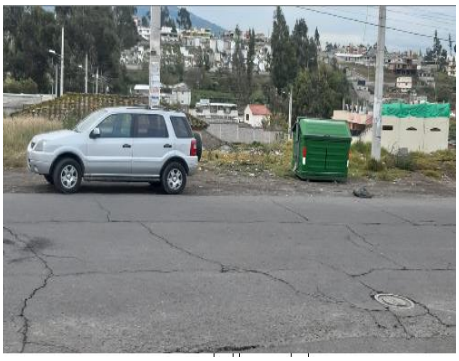


e)

f)



g)



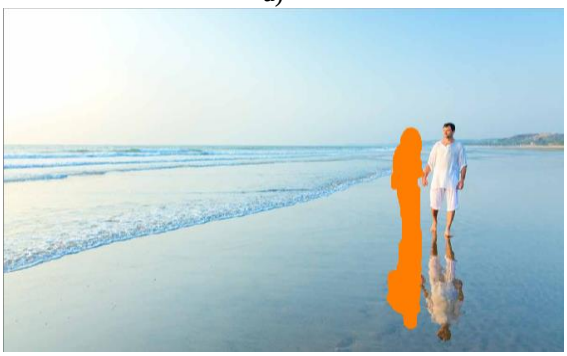
h)

Figura 52. a) Imagen original. b) Iteración. c) Iteración. d) Iteración. e) Iteración f) Iteración. g) Iteración. h) Imagen resultante.
Elaborado por: El Investigador.

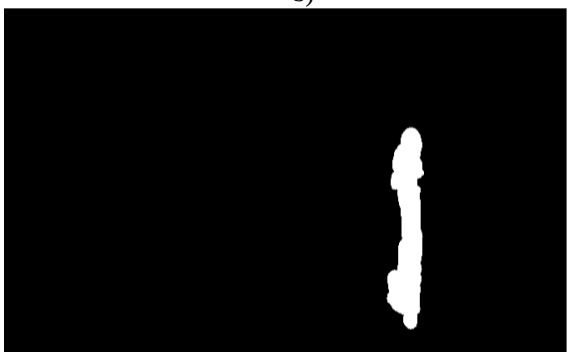
La figura 53 adopta el mismo proceso, pero el número de iteraciones desde el hallazgo de la costura vertical (figura 53e), hasta la imagen restaurada (figura 53i), es de 3 procesos por eliminación de costuras, lo que implica que el algoritmo debe procesar mayor información debido a la calidad de la imagen, mayores detalles y dimensión de la misma.

a)

b)



c)



d)

e)

f)

g)

h)

i)

Figura 53. a) Imagen original. b) Iteración. c) Iteración. d) Iteración. e) Iteración f) Iteración. g) Iteración. h) Iteración. i) Imagen resultante.
Elaborado por: El Investigador.

La figura 54 muestra el algoritmo matemático sujeto a los pasos empleados en el procesamiento de imágenes. El primer paso es la adquisición de datos a través de la imagen original o imagen de entrada, luego se define el procesado de la imagen en este paso se da lugar al mapa del gradiente de la imagen conocido también como máscara de protección que no es nada más que el cálculo de los niveles de energía de la imagen. Además, en el procesado se debe especificar el objeto a remover de la imagen original con la finalidad de formar el mapa energético que es la combinación de la máscara de protección con el mapa de extracción. Luego el algoritmo entra en la fase de representación de los datos, selecciona las costuras con menor energía para ser eliminadas y a la vez va rellenando la región faltante cuidando así los bordes y texturas de la imagen. Finalmente, la fase de interpretación de resultados es el último proceso en donde se compara si la imagen inicial como la resultante tiene el tamaño adecuado y características adecuadas para acorde a la programación del investigador para dar por terminado el procesamiento matemático de la imagen.

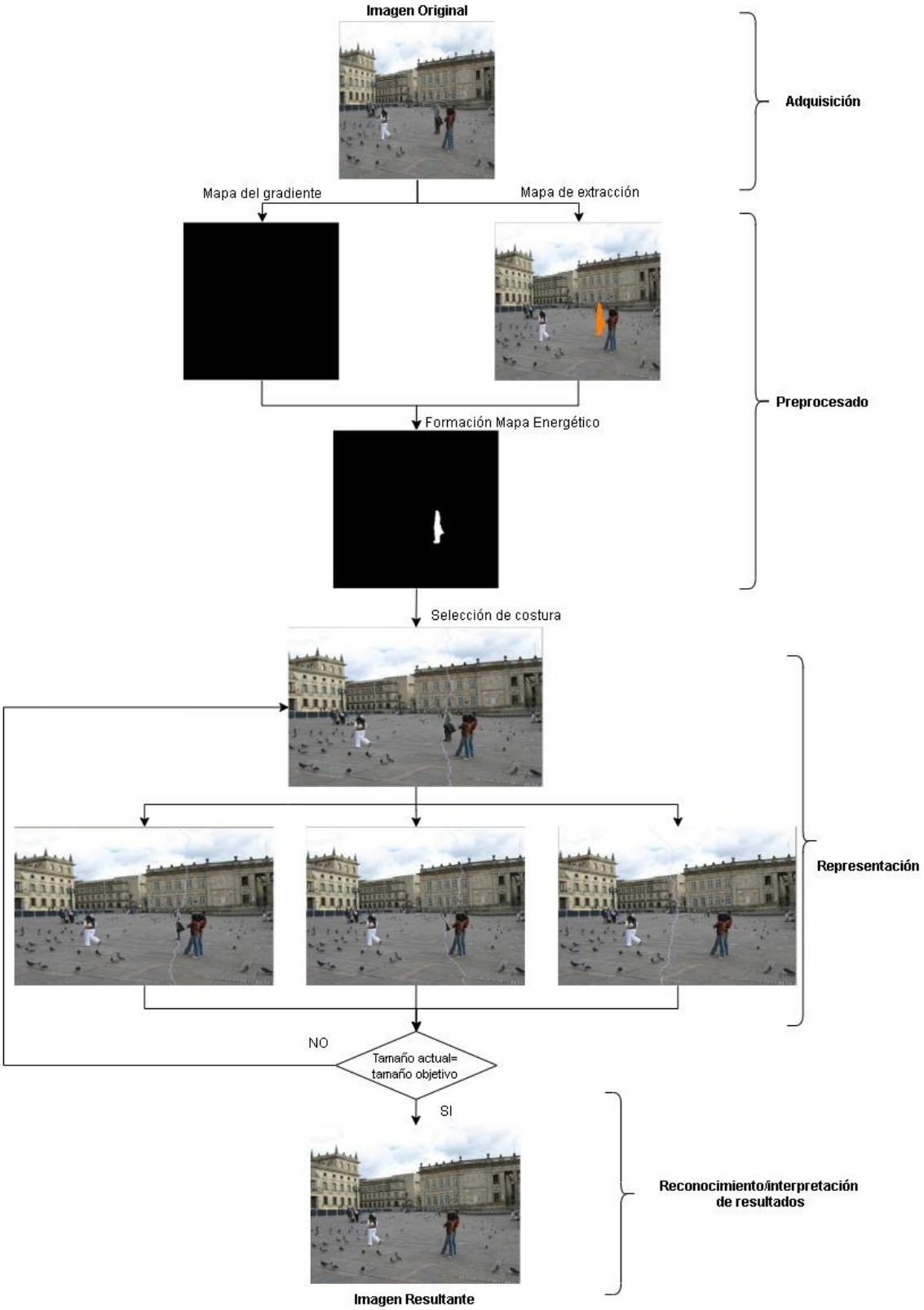


Figura 54. Procesamiento matemático aplicado al modelo matemático
Elaborado por: El Investigador.

4.9. Resultados

Luego de exponer los resultados de las imágenes sometidas al algoritmo matemático en el apartado anterior se procede a demostrar la eficiencia del software desarrollado. Para este proceso se diseñó un algoritmo que compara la imagen original con la imagen resultante comparando pixel a pixel las similitudes y mostrando en pantalla los resultados de la eficiencia. La figura 55 muestra el diagrama de flujo de la eficiencia del modelo matemático.

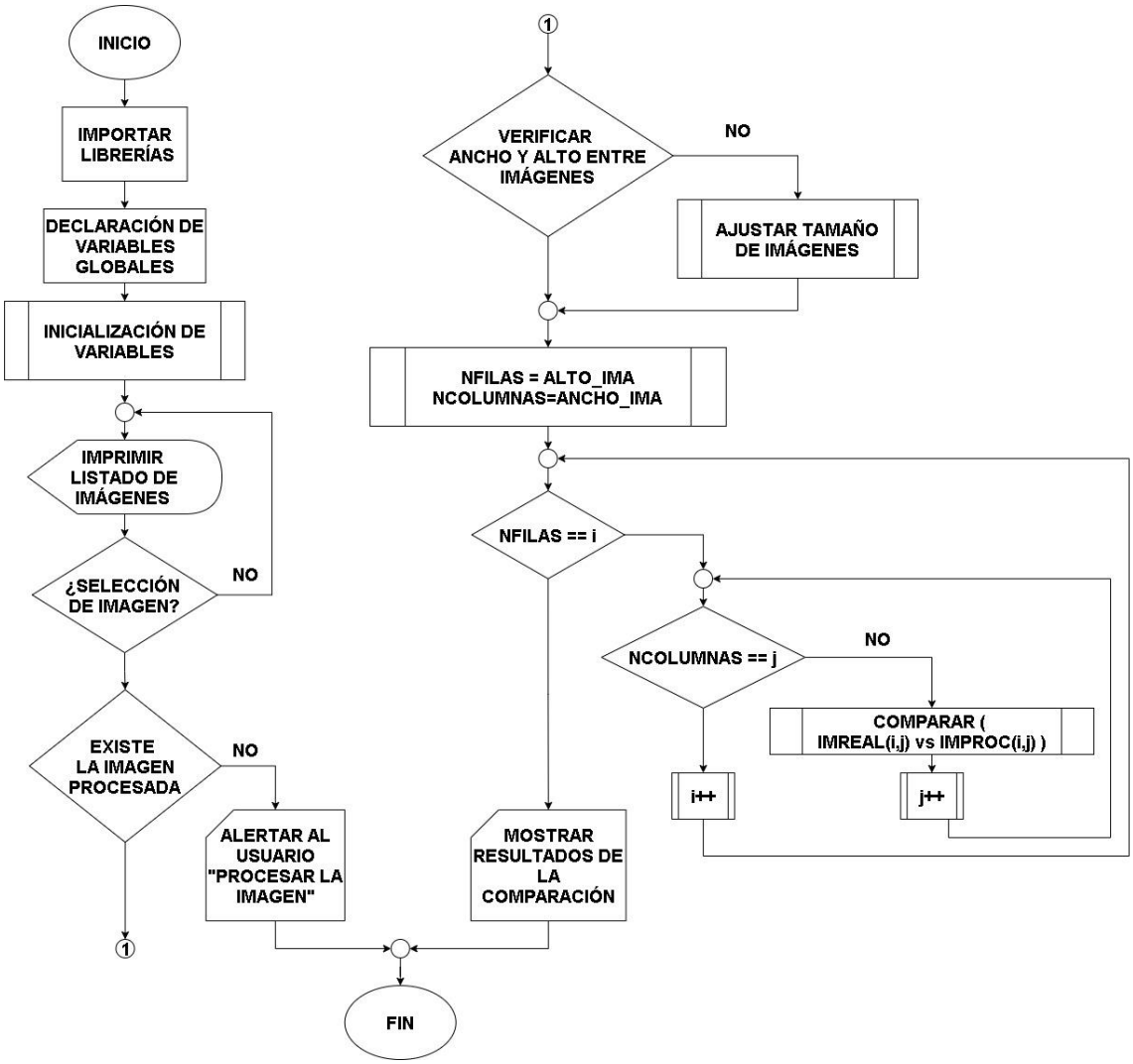


Figura 55.Cálculo de la eficiencia del modelo matemático
Elaborado por: El Investigador.

Y, la figura 56 muestra las líneas de código para la calcular la eficiencia, este software crea un nuevo entorno y muestra la imagen real con la procesada, luego evalúa pixel por pixel si son iguales, por ejemplo, una imagen de 10x10 pixeles se tiene en total 100 pixeles que se debe comparar, al final del proceso se muestra cuanta de variación existe entre la imagen original y la procesada.

```
#############################################################################################
#creacion de funcion de comparacion que se utilizara mas adelante
def comparar(imagen1, imagen2, x_val, y_val):
    efideal = x_val*y_val #cantidad de evaluaciones esperadas
    efespera = 0 #incializar variable
    print("PROCESANDO")
    for i in range(x_val):
        for j in range(y_val):
        bar.update(i*100/x_val)
    bar.finish()
    efespera = cad.double2String(efespera) #procesar valor obtenido para convertirlo en cadena

    print("eficiencia ideal(N pixeles totales) = ",efideal)
    print("eficiencia resultante(N pixeles iguales entre ambas imagenes) = ",efespera)
    eficiencia = round(efespera * 100 / efideal,2)
    mensaje = "Nivel de eficiencia del algoritmo " + str(eficiencia) + " %"
    print(mensaje) #imprime mensaje en consola

    #ctypes.windll.user32.MessageBoxW(0, mensaje, "EFICIENCIA", 1) #despliega un cuadro de mensaje

    f, axarr = plt.subplots(1,2,figsize=(11,4)) #crea matriz para mostrar comparativa
    axarr[0].axis('off') #desactivar ejes numerados
    axarr[0].set_title("Imagen Original") #crear titulos
    axarr[0].imshow(im) #cargar imagen visualmente
    axarr[1].axis('off')
    axarr[1].set_title("Imagen Procesada")
    axarr[1].imshow(im_proc)
    plt.figtext(0.5,0.1, mensaje, ha="center", va="top", fontsize=14, color="r")
    plt.show() #mostrar resultados
```

Figura 56. Eficiencia del modelo matemático
Elaborado por: El Investigador.

La eficiencia se basa en los cálculos se muestran a continuación:

Donde la imagen original está representada por la matriz $f = [m \times n]$.

$$f = \begin{bmatrix} f_{11} & f_{12} & f_{1n} \\ f_{21} & f_{22} & f_{2n} \\ f_{m1} & f_{m2} & f_{mn} \end{bmatrix}$$

Y la imagen resultante esta representa por la matriz $g = [m \times n]$.

$$g = \begin{bmatrix} g_{11} & g_{12} & g_{1n} \\ g_{21} & g_{22} & g_{2n} \\ g_{m1} & g_{m2} & g_{mn} \end{bmatrix}$$

Entonces;

$$y = \sum_{i=1}^{m \times n} [f(m,n) = g(m,n)]$$

$$\forall\, z \in \{f(0,0) = g(0,0), \dots\dots\dots, f(m,n) = g(m,n)\}$$

$$f(z) = \begin{cases} 1\,, Si\ f(m,n) = g(m,n) \\ 0\,, Si\ f(m,n) \neq g(m,n) \end{cases}$$

Por lo tanto, la eficiencia es igual a la ecuación 17.

$$eficiencia = \frac{y \times 100}{m \times n} \tag{21}$$

La figura 57 muestra la eficiencia del modelo matemático en donde la imagen original en comparación con la imagen procesada demuestra que el software es eficiente en un 74.96 %. Además, la calidad de la imagen es muy buena considerando que las texturas de Los Alpes son regulares y los bordes no presentan cambios bruscos de intensidad resaltando los pixeles de la imagen obteniendo un resultado similar a la original.

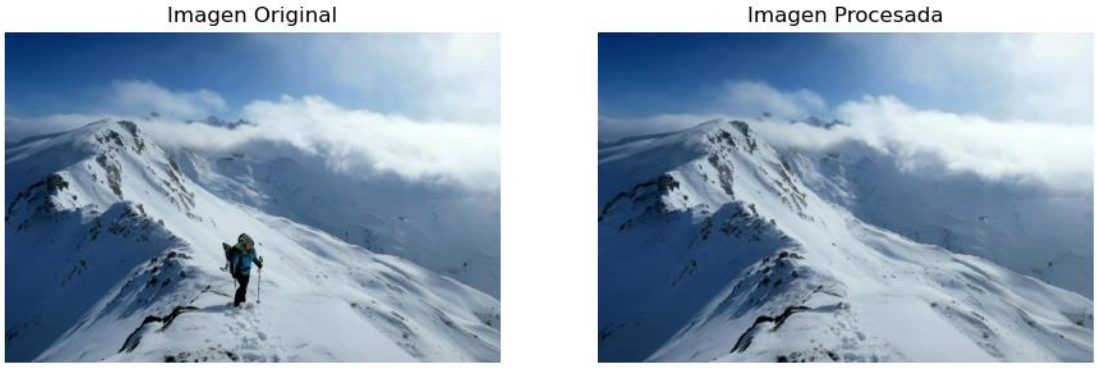


Figura 57. Eficiencia y calidad del modelo matemático (Los Alpes).
Elaborado por: El Investigador.

La figura 58 muestra la eficiencia del modelo matemático en donde la imagen original en comparación con la imagen procesada demuestra que el software es eficiente en un 74.56 %. La calidad de la imagen también es muy buena considerando que las texturas de la laguna son regulares por lo tanto los bordes no presentan cambios bruscos de intensidad resaltando los pixeles de la imagen obteniendo un resultado similar a la original.

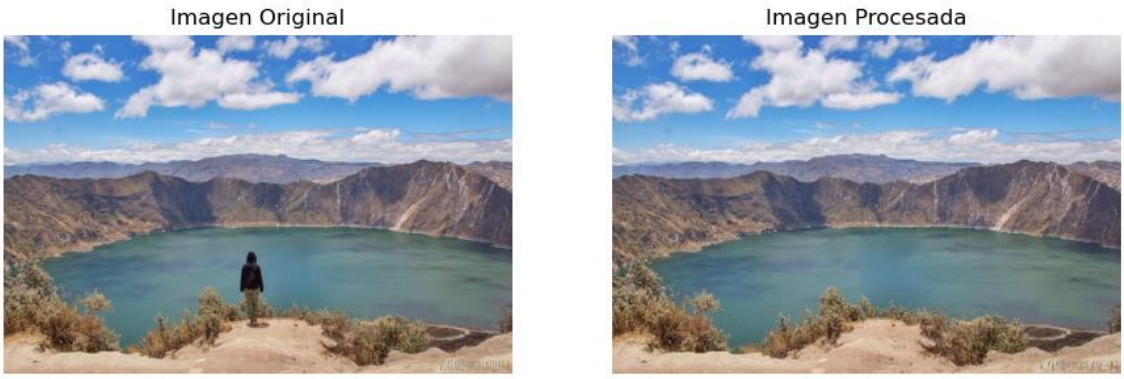


Figura 58. Eficiencia y calidad del modelo matemático (Laguna Quilotoa).
Elaborado por: El Investigador.

La figura 59 muestra la eficiencia del modelo matemático en donde la imagen original en comparación con la imagen procesada demuestra que el software es eficiente en un 69.82 %. Además, la calidad de la imagen es buena considerando que las texturas son semirregulares y los bordes si presentan cambios bruscos de intensidad en los pixeles de la imagen obteniendo un resultado visiblemente aceptable en relación a la original.

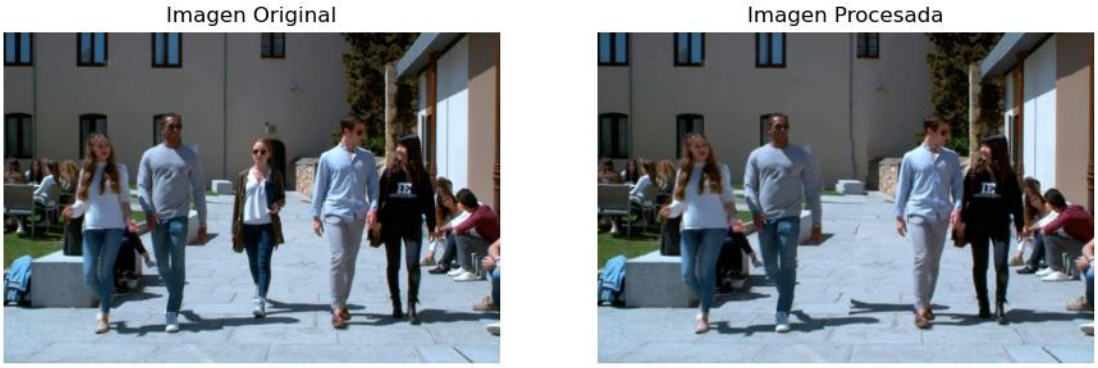


Figura 59. Eficiencia y calidad del modelo matemático (Campus Universitario).
Elaborado por: El Investigador.

Por último, se muestra la imagen 60 de calidad regular en donde la eficiencia del algoritmo es de 47.78 %. Estos resultados se obtienen debido a que las texturas son irregulares y el cambio brusco de bordes hace que los pixeles que rellenan la imagen luego de remover el objeto no concatenen adecuadamente con los demás pixeles.

Figura 60. Eficiencia y calidad del modelo matemático (Paisaje Ambato).
Elaborado por: El Investigador.

Como resultado se obtiene una imagen visiblemente distorsionada, un objeto que sufrió una modificación notable es el automóvil como se observa en la imagen 60b seguido de los cables eléctricos donde el seno que forma en la figura 60a se ve alterado en la imagen resultante.

CAPITULO V

CONCLUSIONES Y RECOMENDACIONES

5.1. Conclusiones

- La bibliografía estudiada infiere que el método Seam Carving conocido también como tallado de costura, es un algoritmo de redimensionamiento de imágenes que procura preservar su calidad y contenido, para lo cual emplea una costura que es una ruta óptima de 8 pixeles conectados de arriba hacia abajo o de izquierda a derecha, utilizando el mismo concepto del tallado de costura y una adaptación del mismo, debido que aplicamos a un punto en específico de la imagen, logre eliminar objetos de imágenes documentales RGB.

- Los algoritmos computacionales implementados basándose en el tallado de costuras y procesos matemáticos matriciales han demostrado que la madurez de la tecnología de procesamiento de imágenes RGB está en la capacidad de reducir el tiempo de procesamiento, escalado, supresión de objetos o redimensionamiento a escasos segundos con una eficiencia en sus resultados del 75% o superior para imágenes documentales en las que las tonalidades sean constantes. Sin embargo, para imágenes con cambios abruptos en su composición los algoritmos han permitido obtener resultados visualmente óptimos, llegando a alcanzar valores que rodean el 45 o 50% de eficiencia del modelo.

- Durante el desarrollo de los algoritmos de procesamiento de imágenes se concluyó que utilizar varios mecanismos matemáticos computacionales para repotenciar algoritmos diseñados para propósitos específicos permite generar nuevas herramientas de libre acceso y con las mismas prestaciones que algoritmos propietarios incluidos en software comercial. Como es el caso de nuestro trabajo de investigación, Python ha simplificado la tarea de procesar matrices y vectores de gran tamaño a un máximo de cinco líneas de código.

- De los resultados obtenidos en la etapa de pruebas, se concluyó que el tiempo de ejecución y respuesta de salida de los algoritmos implementados están

directamente relacionados y matemáticamente son directamente proporcionales. Debido a la cantidad de información que contienen, el análisis computacional a nivel de pixeles RGB requiere gran capacidad de procesamiento para que visualmente la restauración sea exitosa; sin embargo, pese a no existir algoritmos ideales ni modelamiento completamente eficaz para restaurar por completo una imagen, se obtiene resultados óptimos y aceptables en donde los detalles y diferencias entre la imagen real y resultante sean imperceptibles al ojo humano.

- En este trabajo, se investigó una técnica de eliminación de elementos de imágenes consciente del contenido popular, el tallado de costuras, en términos de detección de bordes y propagación de texturas, evalúa las imágenes de forma semántica. Asigna un valor de importancia a cada píxel y elimina la ruta de píxel con la menor importancia acumulativa (energía) del objeto seleccionado en la imagen documental, posteriormente se inserta en su lugar y de manera matemática muestras de pixeles cercanos, por ende, las texturas vecinas mantendrán un balance y armonía en el resultado de la imagen final, por lo que la eliminación se produce principalmente con poco o ningún deterioro visual.

5.2. Recomendaciones

- Tanto en el campo de la matemática aplicada, así como a nivel de software existen infinitas soluciones efectivas que permitan procesar y obtener un resultado al momento de resolver determinados cuestionamientos relacionados con la restauración de imágenes documentales RGB. Sin embargo, se recomienda perfeccionar la modificación de manera inteligente, es decir, que al finalizar el proceso la imagen sea visualmente agradable al observador en donde la sustracción de elementos o la modificación del tamaño no dañe la armonía de colores, formas y texturas.

- El modelamiento matemático implementado en el campo computacional y orientado a solventar un problema en específico como es el caso del tallado de costuras, requiere del entrenamiento y perfeccionamiento a nivel de software e inclusive después de varias décadas de desarrollo aún se continua con procesos evolutivos por lo que es recomendable utilizar un modelo con la madurez tecnológica suficiente para adaptarlo como solución a los problemas planteados.

- Durante el desarrollo de los algoritmos computacionales para la evaluación de resultados del modelo matemático y trabajos futuros relacionados al tallado de costuras, es recomendable seleccionar un único tipo de imagen documental para obtener resultados óptimos durante la etapa de desarrollo. Debido que el eliminar sectores de imágenes y reemplazarlos por texturas próximas al objeto eliminado puede llegar a deformar visualmente la imagen resultante, es por ello que se aconseja que, en la imagen seleccionada, la información a suprimir no ocupe más de 20% del total de la información documental contenida para obtener los mejores resultados.

- Para el procesamiento y restauración de imágenes documentales mediante el algoritmo implementado, pese a que la solución a nivel de software es multiplataforma y con el afán de no limitar futuras investigaciones con un tipo específico de ordenador. Se recomienda el uso de uno de medianas prestaciones

debido a que la rapidez de obtención de resultados está directamente relacionada con la velocidad de procesamiento del mismo.

- Existen numerosas posibilidades de extensión de este trabajo. Una de ellas es extender esta aproximación hacia otros dominios, el primero de ellos el redimensionamiento en vídeo. Esto implicaría pasar de seams exclusivamente espaciales a seams espacio-temporales. Otra sería compaginar adecuadamente la eliminación de costuras verticales y horizontales simultáneamente, pudiendo tener un orden que priorizase los seams de forma entremezclada para optimizar los resultados.

Bibliografía

[1] L. Shutao y Z. Ming, «Image inpainting with salient structure completion and texture propagation,» *Elsevier*, vol. 32, nº 9, pp. 1256-1266, 9 July 2011.

[2] M. Á. Santiago Cabello, «Restauración de imágenes con desensibilización de estimaciones,» Madrid, 2011.

[3] Unimedios, «Imágenes se restaurarían con método matemático,» 2020.

[4] S. Avidan y A. Shamir, «Seam Carving for Content-Aware Image Resizing,» *ACM Transactions on Graphics*, vol. 26, nº 3, p. 10, 2007.

[5] J. Seo, S. Chae, J. Shim, D. Kim, C. Cheong y T.-D. Han 2, «Algoritmo de seguimiento de contorno rápido basado en un método de seguimiento de píxeles para sensores de imagen,» *Sensores*, vol. 16, nº 3, p. 353, 2016.

[6] M. Rubinstein, A. Shamir y S. Avidan, «Improved Seam Carving for Video Retargeting. ACM Transactions on Graphics,» *ACM Trans. Graph.*, vol. 27, nº 3, pp. 1-9, 2008.

[7] D. Zhang, T. Yin, G. Yang, M. Xia, L. Li y X. Sund, «Detecting image seam carving with low scaling ratio using multi-scale spatial and spectral entropies,» *Journal of Visual Conmucation and Image Representation* , vol. 48, pp. 281-291, 2017.

[8] M. Hashemzadeh, B. Asheghi y N. Farajzadeh, «Content-aware image resizing: An improved and shadow-preserving seam carving method,» *Signal Processing*, vol. 155, pp. 233-246, 2019.

[9] Y. G. I. C. A. Bengio, «Deep Learning,» 03 Octubre 2015. [En línea]. Available: https://d1wqtxts1xzle7.cloudfront.net/53631590/Deep_Learning_Bengio_2015-10-03.pdf?1498183571=&response-content-disposition=inline%3B+filename%3DDeep_Learning.pdf&Expires=1609596967

&Signature=fOVSmU-UFEsmt6-R2bJmlJ70qd17IGsmPdWUlJUcVqq6waDvSiOfdWYcf03DNat. [Último acceso: 29 Diciembre 2020].

[10] M. Gómez Mora y R. Rodríguez Rodríguez, «Modelo tridimensional de objetos heterogéneos a partir de imágenes médicas: Herramientas y algoritmos,» 2018.

[11] H. J. Tse, «Seam Carving,» 2018.

[12] A. Rosebrock, «PyImageSearch.,» 23 Enero 2017. [En línea]. Available: https://www.pyimagesearch.com/2017/01/23/seam-carving-with-opencv-python-and-scikit-image/. [Último acceso: 21 Octubre 2020].

[13] D. Zhang, T. Yin, G. Yang, M. Xia, L. Li y X. Sun, «La detección de la costura imagen Talla con baja relación de escala usando multi-escala espacial y espectral entropías,» *Revista de comunicación visual y representación de imágenes,* vol. 48, pp. 281-291, 2017.

[14] M. Wahyudi, H. Purwadi y A. B. W. Putra, «Pengaruh Implementasi Seam Carving Pada Citra Berdasarkan Ciri Tekstur Menggunakan GLCM,» *Edu Komputika Journal,* vol. 7, nº 1, pp. 1-7, 2020.

[15] A. Sai Hareesh y V. Chandrasekaran, «Exemplar-based color image inpainting: a fractional gradient function approach,» *Pattern Analysis and Applications,* vol. 17, nº 2, pp. 389--399, 2014.

[16] P. Patel, A. Prajapati y S. Mishra, «Review ofDifferentInpainting Algorithms,» *International Journal of Computer Applications,* vol. 59, nº 18, pp. 30-34, 2012.

[17] M. Del Fresno, M. Vénere y A. Clausse, «Detección de texturas en imágenes digitales usando el modelo de Lattice Boltzmann,» *Asociación Argentina de Mecánica Computaconal,* vol. XXIX, pp. 6195-6203, 2010.

[18] B. C. Nelson, «Repositorio Universidad Nacional de Colombia,» 2011. [En línea]. Available: https://repositorio.unal.edu.co/bitstream/handle/unal/8383/8300062011_Parte1.pdf?sequence=1&isAllowed=y. [Último acceso: 20 Noviembre 2020].

[19] J. Gómez Rojas, L. L. Camargo Ariza y B. Medina Delgado, «Técnicas de filtrado,» de *Telecomunicación Analógica: Principios de simulación y tratamiento de señal*, Primera ed., Santa Marta, Unimagdalena, 2017, pp. 51-66.

[20] «Filtros,» [En línea]. Available: http://alojamientos.us.es/gtocoma/pid/tema3-1.pdf. [Último acceso: 02 Septiembre 2020].

[21] «Introducción a las imágenes digitales».

[22] G. Portillo Ramírez y H. Cruz Sárez, «Construyendo un algoritmo para el reconocimiento automático de placas,» 2020.

[23] R. Millon, E. Frati y E. Rucci, «Implementación de Filtro de Detección de Bordes Sobel en SoC usando Síntesis de Alto Nivel,» 2020.

[24] «Filtros para detectar bordes,» Noviembre 2002. [En línea]. Available: https://docs.gimp.org/es/gimp-filter-edge-sobel.html. [Último acceso: 20 Agosto 2020].

[25] J. P. Graffigna, «Técnicas de Realce de Imágenes,» 2016.

[26] E. Ortiz Rangel, M. Mejía-Lavalle y H. Sossa, «Filtrado de ruido Gaussiano mediante redes neuronales pulso-acopladas,» *Computación y Sistemas*, vol. 21, nº 2, pp. 381-395, 2017.

[27] J. E. Pisoya, «Sistema de visión artificial para apoyar e la identificación de plagas y enfermedades del cultivo de sandía en el distrito de Ferrenafe,» Chiclayo, 2019.

[28] J. Martínez Matías, J. L. Hernández Hernández y M. HernándezMario Hernández, «Búsqueda del mejor espacio de color para el reconocimiento de frutas utilizando

visión artificial,» de *Congreso Internacional de computación México - Colombia*, Tercera ed., FABBECOR.ONG , 2018, pp. 125- 131.

[29] A. Aroa Gutiérrez , «Representación Morfométrica de Grabados y Petroglifos: Nueveas Tecnologías y Procesos en el Tratamiento Digital de Imagenes RGB,» Madrid , 2017.

[30] J. L. Vázquez Noguera, «Framework de ordenamientolexicogr´afico adaptativo de colores RGB utilizando par´ametros est´adisticos de los histogramas de cada componente de color,» San Lorenzo , 2018.

[31] J. C. Ambriz Polo, R. Alejo Eleuterio y E. López González, «Implementación de visión por computadora a un móvil autónomo basado en Raspberry PI,» *Universidad&Ciencia,* vol. 6, pp. 326-337, 2017.

[32] A. Villegas, D. Gómez y F. Moreno, «Dispositivos electrónicos para reproducir el color en odontología. Revisión de literatura,» *Acta Odontológica Venezolana,* vol. 54, nº 1, 2016.

[33] raj, «raj (2020). inpainting (https://www.mathworks.com/matlabcentral/fileexchange/50366-inpainting), MATLAB Central File Exchange. Recuperado17 de octubre de 2020.,» MATLAB Central File Exchange, 2020. [En línea]. Available: https://www.mathworks.com/matlabcentral/fileexchange/50366-inpainting. [Último acceso: 17 Octubre 2020].

[34] S. E. Auchterberge, «Implementación y análisis de un algoritmo para detección de manipulaciones en imágenes digitales,» 2014.

[35] M. J. Pinilla Bustamante, «Desarrollo de técnicas de inpainting y demosaicing basados en el modelamiento disperso.,» 2019.

[36] I. N. d. E. y. C. (INEC), «Ecuador en cifras,» 2010. [En línea]. Available: https://www.ecuadorencifras.gob.ec/documentos/web-

inec/Estadisticas_Sociales/Encuesta_Estratificacion_Nivel_Socioeconomico/111220_NSE_Presentacion.pdf. [Último acceso: 20 Noviembre 2020].

[37] Á. Fernández Nogales, Investigación y Técnicas de Mercado, Segunda ed., Madrid : Esic, 2004, pp. 154-155.

[38] N. K. Malhotra, Investigación de mercados: un enfoque aplicado, Cuarta ed., Naucalpan de Juárez: Pearson Educación, 2004, p. 94.

[39] L. E. Borges, «Python para desenvolvedores: aborda Python 3.3,» pp. 15-16, 2014.

[40] S. Van der Walt, J. L. Schönberger , J. Nunez Iglesias , F. Boulogne, J. D. Warner, N. Yager, E. Gouillart y T. Yu, «scikit-image: procesamiento de imágenes en Python,» *PeerJ,* vol. 2, p. e453, 2014.

[41] P. S. Foundation, «Python Software Foundation virtualeny 20.0.26,» 2020.

[42] A. Herrera Castro, «Plug-in de procesado visual (OpenCV) en OpenDomo OS Máster Universitario en Software libre,» 2015.

[43] J. C. Suaréz Sánchez , L. Colín Rivas, A. Mejía González y J. C. Ambríz Polo, «Una aproximación al diagnostico de enfermedades de la piel por medio de aprendizaje profundo,» *Aristas: Investigación Básica y Aplicada,* vol. 6, nº 12, pp. 13-16, 2018.

[44] W. Mckinney, Python para Análise de Datos: Tratamiento de datos con Pandas, Numby e Ipython, Segunda ed., São Paulo: Novatec, 2019.

[45] A. Das, «AvikDas,» 29 Julio 2019. [En línea]. Available: https://avikdas.com/2019/07/29/improved-seam-carving-with-forward-energy.html. [Último acceso: 21 Octubre 2020].

ANEXOS

Anexo 1: Código fuente en Python.

```
import os
import numpy as np
from cv2 import *
from glob import glob as files
from pyfiglet import Figlet
from numba import jit
from scipy import ndimage as ndi
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
from PIL import Image

_x, _y = -1, -1 # Variables iniciales para graficar

f = Figlet(font='slant') # Variables para graficar frase inicial en
consola
header = f.renderText('SEWING CARVING V2') # Texto a Graficar
sizeBlank, image_no, isDrawn, stroke_size = 20, 0, False, 3 # Variables
iniciales para ventanas
font = FONT_ITALIC # Tipo de fuente para graficar
clear = lambda: os.system('clear')  # Metodo para limpiar consola
SEAM_COLOR = np.array([255, 200, 200])     # Color para la visualizacion
del procesado seam
SHOULD_DOWNSIZE = True                     # Si es True se habilita el
procesamiento rapido
DOWNSIZE_WIDTH = 500                       # tamano de escalado si
SHOULD_DOWNSIZE es True
ENERGY_MASK_CONST = 100000.0
MASK_THRESHOLD = 10                        # minimo de intensidad de
pixel para la imagen procesada
USE_FORWARD_ENERGY = True

########################################
# CODIGO UTILITARIO
########################################
# Metodo para visualizar el procesado en tiempo real
def visualize(im, boolmask=None, rotate=False):
    vis = im.astype(np.uint8)
    if boolmask is not None:
        vis[np.where(boolmask == False)] = SEAM_COLOR
    if rotate:
        vis = rotate_image(vis, False)
    cv2.imshow("visualization", vis)
    cv2.waitKey(1)
    return vis
# Metodo para realizar el reescalado y acelerar el procesesado
def resize(image, width):
    dim = None
    h, w = image.shape[:2]
    dim = (width, int(h * width / float(w)))
    return cv2.resize(image, dim)
# Metodo para  rotar imagen de ser necesario
```

```
def rotate_image(image, clockwise):
    k = 1 if clockwise else 3
    return np.rot90(image, k)

#######################################
# Funciones de Energia
#######################################

def backward_energy(im):
      #obtiene la convolucion de la imagen en cada eje para decolorar
la imagen COLOR > B/N
    xgrad = ndi.convolve1d(im, np.array([1, 0, -1]), axis=1,
mode='wrap')
    ygrad = ndi.convolve1d(im, np.array([1, 0, -1]), axis=0,
mode='wrap')
    #generar una imagen B/N
    grad_mag = np.sqrt(np.sum(xgrad**2, axis=2) + np.sum(ygrad**2,
axis=2))

    return grad_mag

@jit
def forward_energy(im):
      #obneter las dimensiones de la imagen bajo procesamiento
    h, w = im.shape[:2]
    im = cv2.cvtColor(im.astype(np.uint8),
cv2.COLOR_BGR2GRAY).astype(np.float64)

      #creacion de matriz nula de para procesamiento de energia
    energy = np.zeros((h, w))
    m = np.zeros((h, w))

      # Aplanar la imagen bajo procesamiento
      # Procesado de aplanado consiste en mover la imagen(matriz
dimensional) de ancho(w) x alto(h)
      #     en un vector unidimensional con el afán de reducir el
tiempo de procesamiento
      #     para ello se utiliza el proceso NP.ROLL para rotar la
imager eje por eje U, L y una complementaria
      #     R de apoyo al procesamiento.
    U = np.roll(im, 1, axis=0)
    L = np.roll(im, 1, axis=1)
    R = np.roll(im, -1, axis=1)

      # Se obtiene el valor absoluto de los nuevos vectores siendo
    cU = np.abs(R - L)
    cL = np.abs(U - L) + cU
    cR = np.abs(U - R) + cU

      # EVALUAR PUNTO POR PUNTO LOS VALORES DE LA IMAGEN TRATADA COMO
VECTOR UNIDIMENCIONAL
    for i in range(1, h):
        mU = m[i-1]
        mL = np.roll(mU, 1)
        mR = np.roll(mU, -1)
```

```
        mULR = np.array([mU, mL, mR])
        cULR = np.array([cU[i], cL[i], cR[i]])
        mULR += cULR

        argmins = np.argmin(mULR, axis=0)
        m[i] = np.choose(argmins, mULR)
        energy[i] = np.choose(argmins, cULR)

    return energy


########################################
# Funciones de Ayuda SEAM
########################################

@jit
def add_seam(im, seam_idx):
    h, w = im.shape[:2]
    output = np.zeros((h, w + 1, 3))
    for row in range(h):
        col = seam_idx[row]
        for ch in range(3):
            if col == 0:
                p = np.average(im[row, col: col + 2, ch])
                output[row, col, ch] = im[row, col, ch]
                output[row, col + 1, ch] = p
                output[row, col + 1:, ch] = im[row, col:, ch]
            else:
                p = np.average(im[row, col - 1: col + 1, ch])
                output[row, : col, ch] = im[row, : col, ch]
                output[row, col, ch] = p
                output[row, col + 1:, ch] = im[row, col:, ch]

    return output

@jit
def add_seam_grayscale(im, seam_idx):
    h, w = im.shape[:2]
    output = np.zeros((h, w + 1))
    for row in range(h):
        col = seam_idx[row]
        if col == 0:
            p = np.average(im[row, col: col + 2])
            output[row, col] = im[row, col]
            output[row, col + 1] = p
            output[row, col + 1:] = im[row, col:]
        else:
            p = np.average(im[row, col - 1: col + 1])
            output[row, : col] = im[row, : col]
            output[row, col] = p
            output[row, col + 1:] = im[row, col:]

    return output

@jit
def remove_seam(im, boolmask):
```

```
    h, w = im.shape[:2]
    boolmask3c = np.stack([boolmask] * 3, axis=2)
    return im[boolmask3c].reshape((h, w - 1, 3))

@jit
def remove_seam_grayscale(im, boolmask):
    h, w = im.shape[:2]
    return im[boolmask].reshape((h, w - 1))

@jit
def get_minimum_seam(im, mask=None, remove_mask=None):
    h, w = im.shape[:2]
    energyfn = forward_energy if USE_FORWARD_ENERGY else
backward_energy
    M = energyfn(im)

    if mask is not None:
        M[np.where(mask > MASK_THRESHOLD)] = ENERGY_MASK_CONST

    # dar prioridad a la máscara de eliminación sobre la máscara
protectora mediante el uso de un valor negativo más grande
    if remove_mask is not None:
        M[np.where(remove_mask > MASK_THRESHOLD)] = -ENERGY_MASK_CONST
* 100

    backtrack = np.zeros_like(M, dtype=np.int)

    # rellenar matriz DP
    for i in range(1, h):
        for j in range(0, w):
            if j == 0:
                idx = np.argmin(M[i - 1, j:j + 2])
                backtrack[i, j] = idx + j
                min_energy = M[i-1, idx + j]
            else:
                idx = np.argmin(M[i - 1, j - 1:j + 2])
                backtrack[i, j] = idx + j - 1
                min_energy = M[i - 1, idx + j - 1]

            M[i, j] += min_energy

    # retroceso para encontrar el camino
    seam_idx = []
    boolmask = np.ones((h, w), dtype=np.bool)
    j = np.argmin(M[-1])
    for i in range(h-1, -1, -1):
        boolmask[i, j] = False
        seam_idx.append(j)
        j = backtrack[i, j]

    seam_idx.reverse()
    return np.array(seam_idx), boolmask
```

```
########################################
# Algoritmo Principal
########################################
# Metodo principal para remover tanto en eje x como eje y
def seams_removal(im, num_remove, mask=None, vis=False, rot=False):
    for _ in range(num_remove):
        seam_idx, boolmask = get_minimum_seam(im, mask)
        if vis:
            visualize(im, boolmask, rotate=rot)
        im = remove_seam(im, boolmask)
        if mask is not None:
            mask = remove_seam_grayscale(mask, boolmask)
    return im, mask

# Metodo principal para insertar tanto en eje x como eje y
def seams_insertion(im, num_add, mask=None, vis=False, rot=False):
    seams_record = []
    temp_im = im.copy()
    temp_mask = mask.copy() if mask is not None else None

    for _ in range(num_add):
        seam_idx, boolmask = get_minimum_seam(temp_im, temp_mask)
        if vis:
            visualize(temp_im, boolmask, rotate=rot)

        seams_record.append(seam_idx)
        temp_im = remove_seam(temp_im, boolmask)
        if temp_mask is not None:
            temp_mask = remove_seam_grayscale(temp_mask, boolmask)

    seams_record.reverse()

    for _ in range(num_add):
        seam = seams_record.pop()
        im = add_seam(im, seam)
        if vis:
            visualize(im, rotate=rot)
        if mask is not None:
            mask = add_seam_grayscale(mask, seam)

        #actualizar los índices de costura restantes
        for remaining_seam in seams_record:
            remaining_seam[np.where(remaining_seam >= seam)] += 2

    return im, mask
```

```
########################################
# FUNCIONES DE CONTROL PRINCIPAL
########################################
# Metodo principal para remover tanto en eje x como eje y
def seam_carve(im, dy, dx, mask=None, vis=False):
    im = im.astype(np.float64)
    h, w = im.shape[:2]
    assert h + dy > 0 and w + dx > 0 and dy <= h and dx <= w

    if mask is not None:
        mask = mask.astype(np.float64)

    output = im

    if dx < 0:
        output, mask = seams_removal(output, -dx, mask, vis)

    elif dx > 0:
        output, mask = seams_insertion(output, dx, mask, vis)

    if dy < 0:
        output = rotate_image(output, True)
        if mask is not None:
            mask = rotate_image(mask, True)
        output, mask = seams_removal(output, -dy, mask, vis, rot=True)
        output = rotate_image(output, False)

    elif dy > 0:
        output = rotate_image(output, True)
        if mask is not None:
            mask = rotate_image(mask, True)
        output, mask = seams_insertion(output, dy, mask, vis, rot=True)
        output = rotate_image(output, False)

    return output
# Metodo para remover objetos de la imagen
def object_removal(im, rmask, mask=None, vis=True,
horizontal_removal=False):
    im = im.astype(np.float64) # Se convierte la matriz de imagen de
enteros a flotantes
    rmask = rmask.astype(np.float64) # Se convierte la matriz de
mascara de enteros a flotantes
    if mask is not None:
        mask = mask.astype(np.float64)
    output = im # Se pasan los valores de la imagen original a de
salida

    h, w = im.shape[:2] # Se optiene las dimenciones de imagen

    if horizontal_removal: # Ciclo para procesar imagen si se rota
        output = rotate_image(output, True)
        rmask = rotate_image(rmask, True)
        if mask is not None:
            mask = rotate_image(mask, True)
    # Se inicia cliclo para el procesado
    while len(np.where(rmask > MASK_THRESHOLD)[0]) > 0:
```

```
        seam_idx, boolmask = get_minimum_seam(output, mask, rmask) # Se
llama al metodo de minimum_seam los parametros de salida y macara
        if vis: # Si se encuentra habilitado se muestra el procesado de
remover objeto
            visualize(output, boolmask, rotate=horizontal_removal)  #
Se llama el metodo para visualizar el procesado
        output = remove_seam(output, boolmask) # Se llama al metodo
remove_seam
        rmask = remove_seam_grayscale(rmask, boolmask)
        if mask is not None: # Si se tiene mascara se procesara en
escala de grises
            mask = remove_seam_grayscale(mask, boolmask)
    # Se consulta la dimencion de la imagen para hacer el relleno de
imagen
    num_add = (h if horizontal_removal else w) - output.shape[1]
    # Se realiza la insercion de contenido
    output, mask = seams_insertion(output, num_add, mask, vis,
rot=horizontal_removal)
    if horizontal_removal:
        output = rotate_image(output, False)
    # Se retorna la imagen ya procesada
    return output
# Metodo para crear mascara de imagen
def masking(image):
      # Se buscan todos los puntos de color azul
    mask = (np.array(image[:,:,0]) == 0.0)
    # Se buscan todos los puntos de color verde
    mask = mask & (np.array(image[:,:,1]) == 0.49)
    # Se buscan todos los puntos de color rojo
    mask = mask & (np.array(image[:,:,2]) == 1.0)
    # Se concatenan todos los puntos encontrados
    mask = np.dstack([mask, mask, mask]);
    #Se invierte la matriz concatenada de 0's a 1's
    return (False ^ mask) * (np.array(empty) + 1)

#  Metodo para deteccion del movimiento del mouse
def mouse_callback(mouse_event, x, y, flags, parameters):
    # Se llama a las variables globales
    global _x, _y, isDrawn

    if mouse_event == EVENT_LBUTTONDOWN: # Se detecta el pulso del
boton izquierdo del mouse
        isDrawn = True
        _x, _y = x, y

    elif mouse_event == EVENT_MOUSEMOVE: # Se Detecta los movimientos
del mouse para dibujar lineas
        if isDrawn:
            line(image, (_x, _y), (x, y), (0.0, 0.49, 1.0),
stroke_size) # Se dibuja la linea de color Naranja
            _x, _y = x, y

    elif mouse_event == EVENT_LBUTTONUP: # Se detecta el soltar el
boton izquierdo del mouse
        isDrawn = False
        line(image, (_x, _y), (x, y), (0.0, 0.49, 1.0), stroke_size)
```

```
# Inicio de Programa
#clear()
# Mensaje de Inicio de Consola
#print("\033[95m" + header + "\033[0m")
#print("Lista de Imagenes a Procesar:\n")


#clear()
os.system('cls' if os.name == 'nt' else 'clear')
print("\r\n")
#print("\033[95m" + header + "\033[0m")
print(header)
print("Autor: Nombre Apellido\n")
print("UNIVERSIDAD TÃ‰CNICA DE AMBATO\n")
print("Lista de Imagenes a Procesar:\n")

# Se indexa los directorios de las imagenes alojadas en in_images
images_files = [] #Vector para almacenar la ubicacion de la imagenes
images_index = 0 #Index
images_files.extend(sorted(files(os.path.join('in_images', '*.jpg'))) )
#Se Busca todos los archivos de formato JPG y se agregan las
ubicaciones al vector

split_var = "\\" #Variable para diferenciar las ubicaciones en Windows
y Linux
# Ciclo para remplazar la variable de la ubicacion
for img_path in images_files:
    if split_var not in img_path: #Si la condicion no cumple se
remplazara la variable
        split_var = '/'

    image_name = img_path.split(split_var) #Se divide la ubicacion para
extraer el nombre de la imagen
    print("\033[92m    [%s] - %s\033[0m" % (images_index,
image_name[1])) # Se muestra el nombre de la imagen
    images_index += 1 #Se incremena el index de las imagenes

while(True): #Condicion para presentar todas las imagenes disponibles
para procesar
    images_selection = int(input('\nElija una de las imagenes antes
listadas:')) # Variable para el seleccion de la imagen por teclado
    if images_selection < len(images_files):#Verifica si el numero
ingresado se encuentra entre las imagenes disponibles
        images_path = imread(images_files[images_selection]) / 255.
#Selecciona la imagen del vector de imagenes
        image = images_path #Se almacena la imagen para ser procesada
        break
    else:
        print("\033[91mOpcion Incorrecta - Intentalo
nuevamente\033[0m") #Se Muestra un mensaje si la imagen no es
disponible y reinicia el ciclo

text_box = np.zeros((sizeBlank, 2*image.shape[1] + sizeBlank, 3)) + 1.
#Se genera una Matriz de 1"s para dividir las imagenes a mostrar
```

```
empty = np.zeros(image.shape) #Se genera una matriz de 0's de iguales
dimenciones que la imagen a procesar
blank = np.zeros((image.shape[0], sizeBlank, 3)) + 1 #Se genera una
matriz de 1's para generar la mascara de imagen

namedWindow("Deep Object Removal", WINDOW_NORMAL) #Se declara el nombre
de la ventana para la creacion de la mascara
setMouseCallback('Deep Object Removal', mouse_callback) #Se adjunta la
deteccion del movimiento de mouse en la ventana para dibujar

createTrackbar('Pen Size', 'Deep Object Removal', 1, 50, lambda x: x)
#Se establece una barra para elegir el tamano de pincel

filtered_image = empty #Se inicializa la mascara

while(True):

    sub_window = np.hstack((image, blank, filtered_image[:,:,[2, 1,
0]])) #Se concatenan la imagen original el espacio de separacion y la
mascara
    window = np.vstack((sub_window, text_box)) #Se agrega las imagenes
concatenadas a la ventana a mostrar
    imshow('Deep Object Removal', window)  #Se hace el llamado a la
ventada
    putText(text_box, 'Image', (int(0.5 * image.shape[1]), 15), font,
0.8,(0, 0, 0), 1) #Se agrega el pie de pagina en la imagen Original
    putText(text_box, 'Filtered Image', (int(1.5 * image.shape[1]),
15), font, 0.8,(0, 0, 0), 1) #Se agrega el pie de pagina en la mascara

    key_pressed = waitKey(1) & 0xFF #Se espera el teclear una tecla

    if key_pressed == 27: # Si se teclea ESC el programa se detiene
      break

    elif key_pressed == 102: # Si se teclea F se se crea la mascara

        input_image_masked = masking(image) #Se llama el metodo masking
enviando el parametro imagen
        input_image_masked = input_image_masked[:,:,[2, 1, 0]]
        shape = np.array(input_image_masked).shape #Se extraen las
dimensiones de la imagen procesada

        filtered_image = input_image_masked #Se asigna la mascara a una
variable separada
        out_mask = input_image_masked[:,:,[2, 1, 0]] * 255 #Se asigna
la mascara en la variable de salida para escribir el archivo de mascara
        imwrite('mask_temp.png', out_mask) #Se escribe el archivo de
mascara

    stroke_size = getTrackbarPos('Pen Size', 'Deep Object Removal') #Se
agrega el la barra para seleccion del tamano de pincel

destroyAllWindows() #Se cierran todas las ventanas abiertas

im = imread(images_files[images_selection]) #Se lee la imagen original
assert im is not None
```

```
rmask = imread("mask_temp.png", 0) #Se lee la imagen de mascara
USE_FORWARD_ENERGY = not False #Se establece la visualizacion del
procesado

h, w = im.shape[:2] #Se Estrae las dimenciones de la imagen
if SHOULD_DOWNSIZE and w > DOWNSIZE_WIDTH: # Se compara las dimenciones
para bajar escalar tamano de imagen
    im = resize(im, width=DOWNSIZE_WIDTH) # Se llama al metodo para
escalar
    if rmask is not None: # Se verifica si no se creo la mascara
        rmask = resize(rmask, width=DOWNSIZE_WIDTH) # Se escala la
mascara

print("\n\033[94m****--La Mascara se creo correctamente--
****\n\033[0m")

while(True):
    process_image = input('Iniciar el procesado de la Imagen: (Y/n)')
or 'Y' #Se consulta si se inicia el procesado de procesado

    if process_image in ('Y','y','Yes','yes'): # Se compara respuesta
ingresada por teclado
        assert rmask is not None
        output = object_removal(im, rmask, None, True, False) # Se
llama al metodo para remover el objeto, se envia la imagen original y
mascara
        out_name = images_files[images_selection].split(split_var)[1] #
Se consulta la ubicacion del archivo original para extraer el nombre de
archivo
        img_origin = Image.open(images_files[images_selection]) # Se
abre la imagen original
        os.chdir("out_images") # Se cambia al directorio de imagenes de
salida
        imwrite(out_name, output) # Se escribe la nueva imagen
procesada

        text_box = np.zeros((sizeBlank, 2*im.shape[1] + sizeBlank, 3))
+ 1. #Se genera una Matriz de 1"s para dividir las imagenes a mostrar
        blank = np.zeros((im.shape[0], sizeBlank, 3)) + 1 #Se genera
una matriz de 1's para generar la mascara de imagen
        im = im / 255
        output =  output / 255
        while(True):

            sub_window = np.hstack((im, blank, output)) #Se concatenan
la imagen original el espacio de separacion y la mascara
            window = np.vstack((sub_window, text_box)) #Se agrega las
imagenes concatenadas a la ventana a mostrar
            imshow('Deep Object Removal', window)  #Se hace el llamado
a la ventada
            putText(text_box, 'Image', (int(0.5 * im.shape[1]), 15),
font, 0.8,(0, 0, 0), 1) #Se agrega el pie de pagina en la imagen
Original
            putText(text_box, 'Filtered Image', (int(1.5 *
im.shape[1]), 15), font, 0.8,(0, 0, 0), 1) #Se agrega el pie de pagina
en la mascara
```

```
            key_pressed = waitKey(1) & 0xFF #Se espera el teclear una
tecla

            if key_pressed == 27: # Si se teclea ESC el programa se
detiene
                break

            stroke_size = getTrackbarPos('Pen Size', 'Deep Object
Removal') #Se agrega el la barra para seleccion del tamano de pincel

        destroyAllWindows() #Se cierran todas las ventanas abiertas

        clear() # Se limpia la consola
        print(header) # Se muestra mensaje de inicio
        print("\nLa imagen se almaceno en la carpeta out_images:") # Se
muestra ubicacion de la imagen procesada
        print("\n\033[94m\033[1m    %s\033[0m" % out_name)
        print("\n\033[94mEl proceso se cumplio correctamente\033[0m")
        break # Termina el programa
    else:
        print("No se ha procesado la Imagen Seleccionada")
        break
```

Anexo 2: Código fuente de la eficiencia del software.

```
import os #librerias de sistema
import sys #librerias de sistema
import ctypes  #librerias de sistema
import numpy as np
from PIL import Image
import relation as cad
import progressbar
from time import sleep
from glob import glob as files
import matplotlib.pyplot as plt


#definicion de variables globales
#####creacion de barra de progreso para ayuda visual
bar = progressbar.ProgressBar(maxval=100, \
    widgets=[progressbar.Bar('=', '[', ']'), ' ',
progressbar.Percentage()])
bar.start() #inicializacion


#########################################################################
#creacion de funcion de comparacion que se utilizara mas adelante
def comparar(imagen1, imagen2, x_val, y_val):
    efideal = x_val*y_val #cantidad de evaluaciones esperadas
    efespera = 0 #incializar variable
    print("PROCESANDO")
    for i in range(x_val):
        for j in range(y_val):
            c = np.isclose(imagen1[i,j], imagen2[i,j], atol=5)
            if(c.all()): #si verdadero
```

```
                efespera = efespera + 1
            bar.update(i*100/x_val)
        bar.finish()

        print("eficiencia ideal(N pixeles totales) = ",efideal)
        print("eficiencia resultante(N pixeles iguales entre ambas
imagenes) = ",efespera)

        #obtener porcentaje del valor procesado
        eficiencia = round(efespera * 100 / efideal,2)
        mensaje = "Nivel de eficiencia del algoritmo " + str(eficiencia) +
" %"
        print(mensaje) #imprime mensaje en consola

        #crea matriz para
        f, axarr = plt.subplots(1,2,figsize=(11,4)) mostrar comparativa
        axarr[0].axis('off') #desactivar ejes numerados
        axarr[0].set_title("Imagen Original") #crear titulos
        axarr[0].imshow(im) #cargar imagen visualmente
        axarr[1].axis('off')
        axarr[1].set_title("Imagen Procesada")
        axarr[1].imshow(im_proc)
        plt.figtext(0.5,0.1, mensaje, ha="center", va="top", fontsize=14,
color="r") #cargar mensaje
        plt.show() #mostrar resultados


########################################################################
#funcion para control de errores
def fin():

print("============================================================")
        print("La imagen a comparar no existe en la carpeta de imagenes
procesadas OUT_IMAGES")
        print("      POR FAVOR PROCESARLA PRIMERO!                    ")

print("============================================================")
        sys.exit(1)


########################################################################
#                           INICIO DEL ALGORTIMO
########################################################################

print("Lista de Imagenes para comparar:\n")

# Se indexa los directorios de las imagenes alojadas en in_images
images_files = [] #Vector para almacenar la ubicacion de la imagenes
images_index = 0 #Index
images_files.extend(sorted(files(os.path.join('in_images', '*.jpg'))) )
#Se Busca todos los archivos de formato JPG y se agregan las
ubicaciones al vector

split_var = "\\" #Variable para diferenciar las ubicaciones en Windows
y Linux
# Ciclo para remplazar la variable de la ubicacion
```

```
for img_path in images_files:
    if split_var not in img_path: #Si la condicion no cumple se
remplazara la variable
        split_var = '/'

    image_name = img_path.split(split_var) #Se divide la ubicacion para
extraer el nombre de la imagen
    print(">> [%s] - %s" % (images_index, image_name[1])) # Se muestra
el nombre de la imagen
    images_index += 1 #Se incremena el index de las imagenes

while(True): #Condicion para presentar todas las imagenes disponibles
para procesar
    images_selection = int(input('\nElija una de las imagenes antes
listadas:')) # Variable para el seleccion de la imagen por teclado
    if images_selection < len(images_files):#Verifica si el numero
ingresado se encuentra entre las imagenes disponibles
        dir_in = images_files[images_selection] #OBTENER EL PATH DE LA
IMAGEN A PROCESAR
        dir_out = dir_in.replace("in_","out_") #CREAR PATH DE LA IMAGEN
        #print("seleccion entrada:",dir_in) #AYUDA VISUAL
        #print("seleccion salida:",dir_out) #AYUDA VISUAL

        if os.path.isfile(dir_out): #verificar la existencia del
archivo a trabajar
            print ("SELECCION VALIDA")
        else:
            print ("No existe el archivo: ", dir_out)
            fin()

        im = Image.open(dir_in) #abrimos imagen a procesar. Soporta
casi cualquier extension
        im_proc = Image.open(dir_out) #abrimos imagen a procesar.
Soporta casi cualquier extension
        break
    else:
        print("Opcion Incorrecta - Intentalo nuevamente") #Se Muestra
un mensaje si la imagen no es disponible y reinicia el ciclo


#evalua tamaño de imagenes
real = im.size
proc = im_proc.size

#corrige tamaño de ser necesario
if(real != proc): #verificar tamaños entre ambas imagenes
    im_proc = im_proc.resize((real))
    #crea archivo temporal
    im_proc.save('temporal.jpg') #almacenamos archivo temporal
    im_proc = Image.open('temporal.jpg') #recargar imagen corregida
para procesamiento
else:
    print("la imagenes coinciden en tamaño")

#verificar que el tamaño sea igual para poder comparar matrices
print ("Tamaño de la imagen real: ", im.size)  #ayuda visual
```

```
print ("Tamaño de la imagen proc: ", im_proc.size) #ayuda visual
tam1 = im.size
tam2 = im_proc.size
total1 = tam1[0] * tam1[1]
total2 = tam2[0] * tam2[1]
print("total de pixeles a evaluar en 1: ",total1) #ayuda visual
print("total de pixeles a evaluar en 2: ",total2) #ayuda visual

#creamos objetos con informacion de imagenes
pix = im.load() # cargamos la informacion de la imagen
pix_proc = im_proc.load()

#invocar funcion con argumentos "imagenes a procesar"
comparar(pix,pix_proc,tam1[0],tam1[1])
```