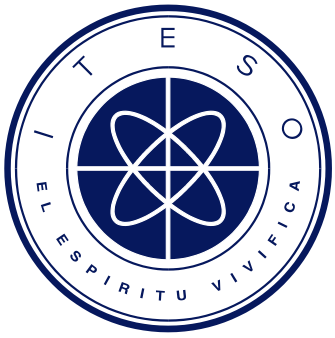
INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE OCCIDENTE

Reconocimiento de validez oficial de estudios de nivel superior según acuerdo secretarial 15018, publicado en el Diario Oficial de la Federación el 29 de noviembre de 1976.

Departamento de Electrónica, Sistemas e Informática

Maestría en Sistemas Computacionales



DESARROLLO DE UNA APLICACIÓN PARA LA REPARACIÓN DE HUECOS (GAPFILLING) DE IMÁGENES SATELITALES LANDSAT 7 ETM+

Trabajo recepcional que para obtener el grado de
Maestro en Sistemas Computacionales

Presenta: Antonio González Velázquez
CVU:783036

Director: Hugo Iván Piza Dávila
Co-director: Guillermo Sánchez Díaz

Tlaquepaque, Jalisco a 26 de noviembre de 2018.

AGRADECIMIENTOS

El autor desea agradecer al CONACYT por el apoyo a este programa académico y haberlo beneficiado con una beca nacional No. 783036, que le permitió cursar esta maestría.

De igual manera desea agradecer al ITESO por brindarle el apoyo económico de la beca de egresado por una segunda ocasión.

También agradecer a familiares, profesores y a amigos por el apoyo en estos dos difíciles años que finalmente han concluido con este trabajo que se espera logre retribuir a lo que se aprendió durante esta jornada.

DEDICATORIA

Para mi Familia.

RESUMEN

En este trabajo se presenta una alternativa para la reparación de imágenes satelitales que presentan daños generados por una falla del sensor del satélite desde 2002, la cual vuelve complejas las tareas de análisis, clasificación, detección, entre otras relacionadas en el ámbito de la geoinformática, geología entre otras.

Se realizó un análisis de las diferentes alternativas reportadas en el estado del arte para reparar las imágenes satelitales que presentan esta falla, encontrando dos principales categorías: algoritmos que usan una imagen de entrada (Input) para corregir la falla mencionada conocida como gaps (huecos o ruido encontrado en las imágenes), las cuales deben tener una fuerte coincidencia con el clima y condiciones presentadas en la imagen a reparar. Por otro lado, existen algoritmos denominados de Inpainting, desarrollados para reparar imágenes en general, sin hacer uso de la segunda imagen Input, usando los pixeles vecinos de los huecos y prediciendo estos valores por diferentes métodos. Sin embargo, estas dos familias de algoritmos presentan diversos inconvenientes desde la precisión en la reparación como en el tiempo de ejecución para aplicar los algoritmos.

Por el estudio realizado, se buscaron algoritmos de tipo Inpainting que presentaran buenas características en procesamiento y soluciones generadas, los cuales no están enfocados a procesar imágenes satelitales. Dentro de estos algoritmos se encontró el desarrollado por Oliveiras [1], el cual usando un procedimiento de convolución de manera iterativa, va mejorando en cada iteración la solución encontrada. Este algoritmo demostró ser capaz de reparar imágenes a partir de una imagen patrón o máscara, la cual funciona como pivote para el manejo de los huecos o gaps. Por medio de un kernel paramétrico (configuración de pixeles vecinos), permitió restaurar imágenes en un tiempo considerablemente corto.

El algoritmo de Oliveiras fue tomado como base, para el desarrollo del algoritmo propuesto en este trabajo de tesis, realizando diferentes adecuaciones en el manejo de la máscara, de los gaps o huecos y realizando programación en GPU, debido a la naturaleza en el tamaño de las imágenes satelitales, en la que se centra este trabajo de tesis. Se realizaron diferentes experimentos, tanto en imágenes satelitales como otras utilizadas en procesamiento de imágenes, para mostrar el desempeño del algoritmo propuesto.

Tabla de contenido.

1. Lista de acrónimos y abreviaturas

ETM		Imágenes de tipo mapa temático mejorado.
USGS		US. Geological Survey
REDD+		Traducido del inglés, Efforts to **r**educe **e**missions from **d**eforestation and forest **d**egradation (Esfuerzos para reducir las emisiones derivadas de la deforestación y la degradación de los bosques)
IDE		Un entonro de desarrollo integrado o entorno de desarrollo interactico, del ingles Integrated Development Environment.

2. Antecedentes

LandSat 7 [1] es un satélite del gobierno de los Estados Unidos de América, puesto en órbita el 15 de abril de 1999, equipado con un sensor llamado Enhanced Thematic Mapper Plus (ETM+). Este sensor incorpora componentes adicionales que lo hace un instrumento más versátil que sus antecesores para el estudio de cambios climáticos, monitoreo de la superficie terrestre y cartografía de grandes áreas.

En el año 2002 el sensor del Landsat 7 (ETM+) presentó una falla, por la cual se comenzaron a generar imágenes satelitales llamadas SLC, por sus siglas en inglés Scan Line Corrector. Este daño origina en todas las imágenes capturadas una pérdida de datos de al menos 22% [14]. Las SLC muestran huecos en forma de líneas paralelas diagonales que cubren la anchura de la imagen. El grosor de cada línea va desde un pixel hasta más de una decena de pixeles (véase Figura 1). Estos huecos serán denominados *gaps* en adelante. Dependiendo del lugar donde se encuentre la falla en la imagen, la pérdida de datos afecta en mayor o menor medida a las tareas de muestreo y clasificación de los elementos de interés de los especialistas.

A raíz de esta falla presentada, se han propuesto diferentes algoritmos que reparan las imágenes rellenando los gaps. Sin embargo, estos algoritmos presentan diferentes limitaciones;

a) Falta de Calidad en la imagen corregida.

   Muchos de estos algoritmos no consiguen corregir las imágenes de manera óptima, dejando zonas borrosas dentro de la imagen; la mayoría de los algoritmos tienen problemas con imágenes que presentan nubes y sombras.

Figura 1 Ejemplo de imagen Landsat 7 sin SLC a la izquierda y con SLC a la derecha. https://glovis.usgs.gov/app

b) No son de Libre acceso.

   Algunos de los algoritmos fueron desarrollados en IDEs o entornos de desarrollo de paga. Lo que bloquea el uso y distribución de los mismos.

c) Necesidad de utilizar imágenes auxiliares.

   Para la mayoría de los algoritmos de gapfilling, es indispensable el uso de imágenes input, las cual es una imagen similar a la que se intenta corregir, pero en un espacio físico y temporal similar.

d) Excesivo tiempo en reparación de la imagen.

   Los algoritmos actuales son tardados debido al gran gasto de memoria que implican, así como de un gran número de comparaciones necesarias para su funcionamiento.

3. Justificación

A pesar de presentar la falla mencionada, las imágenes satelitales producidas por el sensor del Landsat 7 ETM+ siguen siendo utilizadas por expertos en geoinformática. El principal motivo es porque producen bandas de 60 m de largo, mientras que el satélite de características semejantes, el Landsat 8, produce bandas térmicas de 100 m, es decir, tiene menor resolución que Landsat 7. Estas bandas térmicas son utilizadas para realizar estudios de geología que considera los fenómenos térmicos en la superficie terrestre, además, se pueden hacer estudios multi-temporales porque existen bases de datos de imágenes Landsat 7 desde 1999.

LandSat 7 **Enhanced thematic Mapper Plus (ETM+)**	**Bandas**	**Longitud de onda (micrómetros)**	**Resolución (metros)**
	Banda 1- Azul	0.45-0.52	30
	Banda 2- Verde	0.52-0.60	30
	Banda 3- Rojo	0.63-0.69	30
	Banda 4- Infrarrojo cercano (NIR)	0.77-0.90	30
	Banda 5- Onda corta infrarrojo (SWIR) 1	1.55-1.75	30
	Banda 6- Térmico	10.40-12-50	60*(30)
	Banda 7- Onda corta infrarrojo (SWIR) 2	2.09-2.35	30
	Banda 8- pancromático	.52-.90	15
*ETM+ Banda 6 tiene una resolución de 60 metros pero el producto final es resampleado a 30 metros.			

Tabla 1, Descripción de las bandas en el Satélite Landsat 7.

Aunado a este motivo, el tiempo que tarda Landsat 7 en transitar por un mismo punto en el globo terráqueo es de 8 días, mientras que en Landsat 8 es de 16 días, un ciclo de recolección de muestras menor. Por esta razón, Landsat 7 permite a los especialistas un modelado más preciso de las tendencias climáticas.

Las características más relevantes de las imágenes producidas por Landsat 7 y 8 se muestran en las tablas 1 y 2, respectivamente.

LandSat 8	**Bandas**	**Longitud de onda (micrómetros)**	**Resolución (metros)**
Operational Land Imager (OLI)	Banda 1- Ultra Azul	0.435-0.451	30
	Banda 2- Azul	0.452-0.512	30
	Banda 3- Verde	0.533-0.590	30
Thermal Infrared Sensor (TIRS)	Banda 4- Rojo	0.636-0.673	30
	Banda 5- Infrarrojo cercano (NIR)	0.851-0.879	30
	Banda 6- Onda corta infrarrojo (SWIR) 1	1.566-1.651	30
	Banda 7- Onda corta infrarrojo (SWIR) 2	2.107-2.294	30
	Banda 8- Pancromático	0.503-0.676	15
	Banda 9- Nubes grises.	1.363-1.384	30
	Banda 10- Térmico Infrarrojo (TIRS) 1	10.60-11.19	100*(30)
	Banda 11- Térmico Infrarrojo (TIRS) 2	11.50-12-51	100*(30)
*TIRS Bandas tienen una resolución de 100 metros pero el producto final es resampleado a 30 metros.			

Tabla 2, Descripción de las bandas en el Satélite Landsat 8.

4. Estado del Arte

Existen diferentes algoritmos diseñados para reparar las imágenes dañadas con *gaps* del satélite Lansat 7 ETM+. Estos algoritmos se denominan *gapfilling*. Algunos utilizan una imagen de entrada (Input) de la misma zona geográfica que la imagen por reparar, pero en otro momento y con diferente grado de daño. Otros algoritmos de *gapfilling* realizan operaciones de interpolación con los pixeles vecinos de los gaps para rellenarlos, algunos ejemplos de estos algoritmos se pueden encontrar en [15]. Por otro lado, se han desarrollado algoritmos denominados *Inpainting* para restaurar videos, fotografías e imágenes en general con pequeños daños, los cuales no fueron diseñados para realizar tareas de *gapfilling* sobre imágenes satelitales [16].

4.1 Algoritmos Gapfilling

Se han desarrollado diferentes técnicas para rellenar los huecos en las imágenes Landsat 7 ETM +, los cuales han demostrado reparar los *gaps* con diferentes resultados y precisión. Inmediatamente después de la falla de SLC, un número de procedimientos simples fueron sugeridos por la USGS [7], incluyendo un sencillo intercambio de pixeles con los datos disponibles obtenidos de una fecha próxima a la imagen satelital dañada. Un método alternativo, conocido como Local Linear Histogram Matching (LLHM), reemplaza los datos faltantes con observaciones de la distribución de valores en un área local desde otra fuente de datos similares. Este método creado por Scaramuzza, Micijevic y Chander en el 2004 [4], es bastante fácil de implementar porque asume una relación lineal entre la brecha de datos de imagen SLC-off y los datos de la imagen cronológicamente próxima disponibles. Se ha demostrado que es útil en áreas o paisajes relativamente homogéneos (USGS

2004). Sin embargo, el procedimiento no funciona adecuadamente si hay nubes, sombras de las nubes o brillo del sol [5].

En varios escenarios, la solución basada en histogramas no es recomendada debido a su bajo rendimiento en paisajes heterogéneos o nublados [6]; por lo tanto, las creaciones de métodos más complejo fueron necesarios, por ejemplo, en la detección de cambios forestales en Columbia Británica, Wulder et al. (2008) encontraron que al usar una imagen input Landsat 7 ETM+ SLC-off no se identificaron aproximadamente el 35% de los píxeles dañados. Después se implementó un método que rellenaba los *gaps* basado en segmentos, rutina que mejoró considerablemente la precisión de detección de cambios abruptos para los pixeles dañados en las imágenes SLC-off, pero aún así, resultó en que el 12% de los *gaps* (objetos o polígonos) no rellenados [8]. En otro estudio, Bédard [9] empleó una solución basado en histograma para rellenar pixeles del Landsat 7 ETM + SLC-off, encontraron que la precisión de la clasificación de la cobertura terrestre disminuyó aproximadamente 3% en comparación con la precisión que se puede obtener con Landsat 7 ETM + SLC.

Se han desarrollado nuevos algoritmos que utilizan métodos más complejos para el relleno de *gaps* SLC-off basado en estimaciones de pixeles vecinos o geo-estadísticos obtenidas de los datos de la imagen en área que rodea los *gaps*. Por ejemplo, el interpolador de píxeles vecinos similares (NSPI) desarrollado por Chen[5], es un método determinista que supone que pixeles vecinos tienen características espectrales similares. Por lo tanto, estos pixeles vecinos pueden ser seleccionado para interpolar el valor de los pixeles de la imagen. Tal enfoque ha demostrado funcionar bien para el relleno de *gaps* Landsat ETM + SLC-off en paisajes heterogéneos, y en presencia de características pequeñas o estrechas, en comparación con los métodos basados en histogramas. El NSPI también funcionó bien en situaciones donde había un intervalo de tiempo relativamente largo entre

las imágenes, y con cambios espectrales evidente entre los datos SLC-off y los disponibles en la imagen input.

Otro método propuesto es la regresión lineal ponderada (WLR) (Zeng, Shen, y Zhang [10]). Estos enfoques pueden acceder a series de tiempo Landsat multitemporales para proporcionar estimaciones de las imágenes más apropiadas para el relleno de *gaps*. Sin embargo, los resultados obtenidos con tales métodos pueden ser subóptimos si el efecto de la superposición estacionaria intrínseca, no se puede cumplir [11].

La principal limitación que presentan estos algoritmos es que necesitan una imagen sin fallas, cronológica muy parecida a la que se está procesando.

4.2 Algoritmos Inpainting

Inpainting (también conocido como interpolación de imagen o interpolación de video) es un proceso de reconstrucción de partes deterioradas en imágenes y videos. Por ejemplo, en el caso de una pintura valiosa, esta tarea sería llevada por un artista experto en restauración de imágenes. Inpainting se refiere a la aplicación de algoritmos sofisticados para reemplazar piezas pérdidas o dañadas en los datos de la imagen.

Los métodos inpainting pueden ser utilizados en el cine y en la fotografía para eliminar daños como arañazos y manchas de polvo (deterioro) en los datos. También se puede usar para eliminar algún objeto de la imagen o simplemente la eliminación de ojos rojos.

Existen muchas variantes de algoritmos Inpainting. Cada una está en función del tipo de textura de la imagen a restaurar. Por ejemplo, la técnica Texture Synthesis-based image inpainting [17] rellena los gaps mediante muestreo, y copiando los

pixeles vecinos al gap. De acuerdo a la textura de la imagen, mantiene continuidad entre los pixeles del gap y los pixeles de la imagen original.

Una limitación presentada en estos algoritmos es que necesitan crear manualmente una imagen máscara, la cual sirve de pivote o de matriz de datos para localizar los huecos o *gaps* en la imagen original. Además, en varios de los algoritmos el tiempo de ejecución sobre imágenes pequeñas oscila en el orden de los minutos. El tamaño de las imágenes satelitales es un factor a tomar en consideración porque son considerablemente más grandes que las imágenes que se utilizaron para probar este algoritmo.

Otra técnica inpainting ampliamente utilizada es Exemplar and Search-Based image inpainting. Este algoritmo es muy efectivo y es impulsado por el uso de isofotos, propuesto por Criminisi [12]. En este algoritmo se utiliza un mecanismo basado en la prioridad para determinar el orden de llenado del área dañada. Este método funciona muy bien para una gran cantidad de imágenes. La restauración de textura y estructuras es muy buena. La limitación con este método es el manejo de los elementos curvos, los cuales no son resueltos de manera correcta debido a la inadecuada selección de los parches.

El método Fast Semiautomatic Inpainting, propuesto por Oliveras [20], repara la imagen utilizando la región iterativa de la imagen al interior con Kernel de difusión. Este método usa FMM (Fast Marching Method) para la propagación de la información de la imagen. Sin embargo, no es adecuado para imágenes con agujeros de gran tamaño como para imágenes donde la región de los bordes no tienes información.

Los algoritmos diseñados para la restauración de películas no son apropiados para imágenes dañadas ya que normalmente trabajan en regiones relativamente

pequeñas y confían en la existencia de información de los varios cuadros (no solo una imagen). Por otro lado, los algoritmos basados en texturas pueden llenar regiones grandes, pero requieren que el usuario especifique qué textura poner y dónde. Esta es una limitación significativa de estos enfoques, ya que se ha visto en varios ejemplos presentados donde la región de los *gaps* está rodeada de cientos de fondos diferentes, algunos de ellos son estructura y no textura. La técnica de Inpainting de Bertalmio [18]. no requiere intervención por parte del usuario, una vez que se ha seleccionado la región que se va a pintar.

El algoritmo es capaz de llenar simultáneamente regiones rodeadas de diferentes fondos, sin que el usuario especifique "qué poner y donde". No requiere suposiciones sobre la topología de la región a pintar, o sobre la simplicidad de la imagen. El algoritmo está ideado para restaurar en regiones estructuradas (p. ej., regiones que se cruzan a través de límites), aunque no está diseñado para reproducir grandes áreas de textura.

El algoritmo de Bertalmio requiere solamente la imagen que se restaurará y la máscara que delimita la parte que se va a pintar. Como un preprocesamiento, toda la imagen original se somete a difusión anisotrópica suavizadora. El propósito de esto es minimizar la influencia de ruido en la estimación de la dirección de las isofotos que llegan al gap. Después de esto, la imagen ingresa al ciclo de pintar, donde solo los valores dentro del gap son modificados Estos valores cambian de acuerdo a la implementación discreta del procedimiento de pintado. Cada iteración, un paso de difusión anisotrópica es aplicada, se usa una implementación directa de diferencias centrales. Este proceso es repite hasta que se consiga un estado estable. El tiempo promedio de procesamiento requerido para restaurar una imagen de tamaño regular usando el método de Bertalmio oscila en los minutos.

Todo algoritmo de Inpainting fue concebido para restaurar imágenes con áreas dañadas relativamente pequeñas, dando una aproximación de la misma, sin llegar necesariamente a una reconstrucción exacta.

Por lo tanto, un algoritmo Inpainting para que sea razonablemente exitoso, debe reparar imágenes dañadas (o con *gaps*) relativamente pequeñas. A medida que las regiones se vuelven más pequeñas, los modelos más simples pueden ser más efectivos que métodos sofisticados.

Por lo tanto, para un área pequeña a reparar el procedimiento de Inpainting puede ser aproximado por un proceso de difusión isotrópica [21], el cual propaga información de los límites de la imagen al pixel que se está reparando.

El algoritmo Inpainting de Oliveiras [20] consiste en inicializar el pixel en reparación, borrando su información de color y convolucionando repetidamente la región para pintar con un Kernel de difusión. Este proceso se repite mientras la imagen haya tenido algún cambio.

Alternativamente, este algoritmo permite especificar el número de iteraciones a ejecutarse. A medida que el proceso de difusión se itera, la pintura muestra la reparación realizada de forma gradual. Otra observación importante utilizada en el diseño del algoritmo Oliveiras es que el ojo humano puede tolerar cierta cantidad de borrosidad en áreas de alto contraste.

La convolución de una imagen con un núcleo gaussiano (es decir, calcular promedios ponderados de los vecindarios de pixeles) es equivalente a la difusión isotrópica (ecuación de calor lineal). El algoritmo de Oliveiras usa un kernel promedio ponderado que solo considera las contribuciones de los pixeles vecinos (es decir, tiene un peso cero en el centro del kernel).

La creación de la máscara que se utiliza para especificar las regiones que se restaurarán es el paso más lento del proceso de Oliveiras, debido a que requiere la intervención del usuario.

El algoritmo de Oliveiras puede restaurar una imagen en solo unos segundos. Sin embargo, no fue diseñado para restaurar imágenes con máscaras que contengan áreas medianas o grandes. Además, se requiere que el usuario genere la máscara de manera manual.

Algoritmo	**Método**	**Rango de tiempo Promedio**	**Aplicado a ETM+**	**Uso de imagen input**	**Uso de Mascara**	**Uso de GPU**	**Información adicional proporcionada por el usuario.**
GNSPI	Gapfilling	3-5 *M	Si	Si	No	No	Imagen input
NSPI	Gapfilling	3-5 *M	Si	Si	No	No	Imagen input
WLR	Gapfilling	3-5 *M	Si	Si	No	No	Imagen input
LLHM	Gapfilling	1-3 *M	Si	Si	No	No	Imagen input
Criminisi	Inpainting	1 *M	No	No	Si	No	Imagen mascara
Oliveiras	Inpainting	5 **S	No	No	Si	No	Imagen mascara
Bertalmio	Inpainting	1-2 *M	No	No	Si	No	Imagen mascara
Fast Semiautomatic Inpating	Inpainting	10 **S	No	No	Si	No	Imagen mascara
*M=Minutos, **S= Segundos							

Tabla 3, Comparativa algoritmos Gapfilling e Inpainting.

Ninguno de los algoritmos relacionados encontrados en el estado del arte con el problema de restaurar imágenes satelitales Landsat 7, ha sido concebido para procesar las imágenes en forma paralela, usando multi-cores o GPU.

En la tabla 3, se muestran las limitaciones y capacidades de los algoritmos presentados en el estado del arte.

Con base en las limitaciones encontradas, tanto en los algoritmos de tipo inpainting con en los de gapfilling, este trabajo de tesis propone el diseño e implementación de un algoritmo de rellenado de gaps eficiente, aprovechando los núcleos de la GPU. Este algoritmo será del tipo inpainting por la falta de disponibilidad de imágenes de referencia. Por otro lado, se creará la imagen máscara de forma automática para, con esto, no exigir intervención del usuario.

5. Justificación

La meta principal del trabajo propuesto es contribuir a la preservación de los bosques y determinar qué zonas muestras un incremento en la deforestación, además de la continuidad del crecimiento del banco de datos del proyecto satelital LandSat (en este caso del LandSat 7) el cual es determinante debido a programas de cartografía y de geomonitoreo, por mencionar alguno; Reducción de emisiones por la deforestación y degradación de Bosques (REDD+), estos programas tienen como objetivo la reconstrucción y el monitorea de bosques con el paso del tiempo para estimar las pérdidas de las reservas de carbón por la degradación.

"…Sin embargo, las discusiones continuas sobre el cambio climático y el rol de los bosques han llevado a diálogos sobre la reducción de las emisiones derivadas de la deforestación y la degradación forestal, comúnmente conocido como REDD. Como un conjunto de pasos y pautas diseñados para usar incentivos de mercado para mitigar la deforestación y la disminución de la acumulación forestal y los servicios forestales…", REDD tiene como objetivo evitar que los carbonatos almacenados en los bosques sean liberados a la atmósfera. El alcance del concepto original de REDD se limita a la deforestación y la degradación forestal; REDD-plus (REDD +) va más allá e incluye el papel de la conservación, la gestión sostenible de los bosques y la mejora de las reservas forestales de carbono [2].

La agricultura comercial es el motivo más importante de la deforestación en América Latina (alrededor de 2/3 de la superficie total deforestada). En África y Asia tropical, le corresponde aproximadamente 1/3 de la deforestación y tiene una importancia similar a la Agricultura de subsistencia. Por otro lado, la Minería, la infraestructura y la expansión urbana son importantes, pero menos prominente. Las conclusiones sobre los patrones globales de la degradación indican que la extracción de madera y la tala

son actividades que representan más del 70% del total la degradación en América Latina y Asia tropical [22,23].

Dado el escenario presentado arriba, es de gran importancia la implementación de un algoritmo de rellenado de huecos (gap-filling) que corrija el defecto antes señalado por el sensor ETM+ en el satélite LandSat 7, además de que representa una opción más viable y de bajo costo que remplazar el sensor en su totalidad.

6. Hipótesis

El desarrollo de un algoritmo que repare los *gaps* en las imágenes satelitales, de uso libre, efectivo y que el tiempo de ejecución sea considerablemente más bajo que las soluciones que existen actualmente; el uso del método propuesto por Oliveiras, el cual ha sido usado únicamente en restauración de fotografías dañadas, ha demostrado ser más rápido que otros métodos inpating similares, además de ser efectivo en imágenes dañadas parcialmente, además, por el uso de procesamiento en paralelo, esto le otorgará una clara ventaja al reducir el tiempo de procesamiento y en la efectividad del resultado. Dado que el método Oliveiras realiza una operación por cada pixel clasificado como *gap*, el procesamiento paralelo repercutirá en el desempeño de dicho método.

Esto contribuiría en la comunidad especialista en geológica, petrográfica, mineralógica, orogénica y geodinámica aportando una solución no existente en la actualidad, facilitando la exploración continua y contribuyendo a la veracidad del banco de datos de las imágenes geoespaciales.

7. Objetivos

7.1 Objetivo General

Desarrollar una solución factible, de fácil acceso y de código abierto que favorezca la investigación de las imágenes satelitales dañadas, reparándolas para reducir la perdida de información y por consiguiente favorecer el desempeño de los científicos y técnicos especializados en las diferentes ramas de la geología.

7.2 Objetivos Específicos

1. Analizar las soluciones propuestas por contribuciones anteriores para examinar las ventajas y desventajas de cada una para concurrir en la que mejor se adapte al contexto actual.
2. Determinar si las soluciones propuestas son prácticas y materializables y además que no demanden el uso de súper computadores y métodos distribuidos apartados del usuario común o de plataformas de acceso restringido.
3. Desarrollar el algoritmo que mejor satisfaga las necesidades establecidas.
4. Reparar imágenes satelitales ETM+ del Satélite Landsat 7.
5. Comparar resultados con otros métodos propuestos.

8. Novedad científica, tecnológica o aportación.

Los algoritmos que actualmente se encuentran desarrollados y al alcance de los usuarios finales que buscan reparar las imágenes ETM+ del Landsat 7, están desarrollados en lenguajes de programación de poca difusión. Además, los ambientes de ejecución de estos algoritmos tienen costo, por ejemplo: ENVI [24], y ArcGis[25].

Por otro lado, los actuales algoritmos usan una imagen input óptica para remplazar los huecos. Esto representa una pérdida en la precisión ya que la imagen input es susceptible a aparición de nubes o a ruido térmico en el sensor. Además, la distancia cronológica entre las imágenes es tal que ocasionaría una discrepancia en la evolución del problema a tratar; deforestación, erosión, etc.

El algoritmo que probó ser el más eficiente entre aquellos que no necesitan una imagen input es el de Oliveiras. Además, demostró ser bastante eficaz por usar un método de convolución. Se implementaron mejoras considerables a este algoritmo, entre las cuales se destacan: 1) Creación automática de la imagen Máscara (juega un rol importante como pivote o matriz de datos de los huecos de la imagen original), e 2) Implementación de este algoritmo para su ejecución en paralelo (GPU).

9. Marco Teórico/Conceptual

9.1 Imagen Máscara

Una imagen máscara es usada en los algoritmos inpainting como pivote para reconocimiento de la ubicación actual de los *gaps* en la imagen dañada.

En los algoritmos de *gapfilling*, tradicionalmente se crea un arreglo bidimensional (X,Y) donde X es igual al número total de pixeles en el eje X de la imagen dañada. De igual manera Y es igual al número total de pixeles en el eje Y de la imagen dañadas.

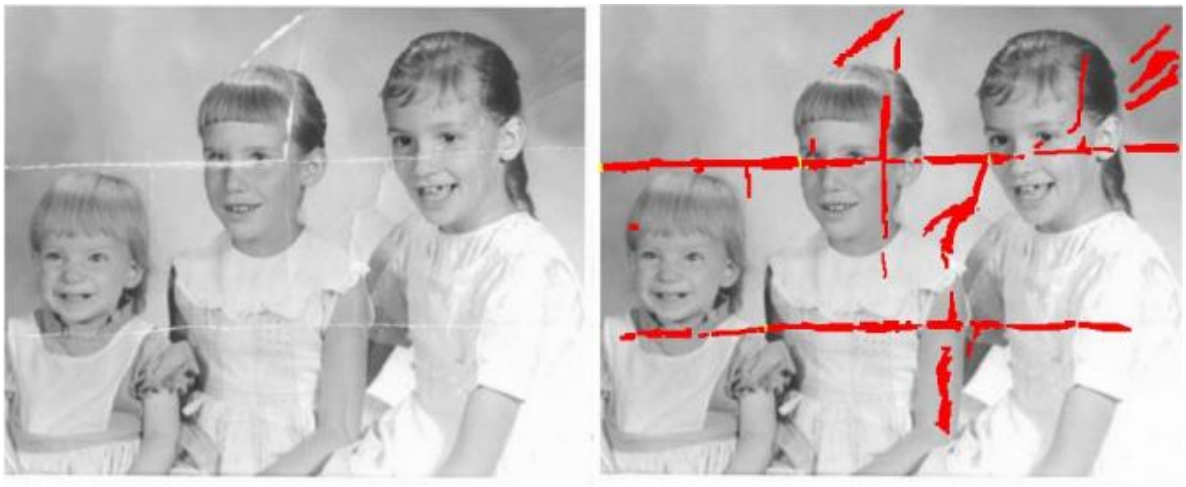


Figura 2, Ejemplo de creación manual de la imagen mascara.

Para los algoritmos de Inpainting, la imagen máscara juega un papel importante, ya que además de ser un pivote de los gaps en la imagen dañada, también contribuye a reducir errores en tiempo de ejecución por falta de memoria RAM cuando estamos lidiando con imágenes muy grandes, ya que la máscara está almacenada en un archivo y se va procesando parcialmente. Esto implica que la imagen máscara tiene que ser creada manualmente por el usuario que conoce

dónde están los gaps a rellenar. Adicionalmente, la lectura de datos en archivo es naturalmente más lenta que la lectura de datos en memoria (arreglo bidimensional).

En el ejemplo mostrado por la figura 2, se puede observar que la imagen máscara es una copia de la imagen dañada que es pintada por un color que no predomine en la imagen original, esto para identificarlo fácilmente. Existen diversos programas de uso libre con el que se puede realizar este trabajo. El tiempo promedio que un usuario tarda en la creación de la imagen mascara es relativamente corto, alrededor de 5 minutos para generar la imagen mascara.

El presente trabajo tiene dentro de sus objetivos facilitar la intervención del usuario al momento de la creación de la imagen mascará, ya que se generará automáticamente por medio de un algoritmo que permite identificar y señalar los *gaps* con un color predeterminado preferentemente no contenido en la imagen original. Para cada pixel contenido en la imagen original dañado donde su valor de color este por debajo de un estimado configurable será marcado como un gap y será rellenado con un color absoluto.

El algoritmo de creación automática de la imagen máscara, promediará el valor de los pixeles en el total de bandas, para considerar si es un gap o no. En el ejemplo de la figura cuatro, se puede apreciar el valor de cuatro pixeles, donde el algoritmo sumará el valor de las tres bandas de cada pixel y lo dividirá por el número de bandas, (en este caso de 3), obteniendo así un valor promedio el cual, si queda por debajo del valor umbral, será considerado gap.

Dado que la mayoría de los valores de los *gaps* producidos por el SLC en el satélite Landsat 7 están por debajo de un umbral de un valor cromático promedio de 30, se utilizó dicho valor por default.

En la figura 4, se puede observar un conjunto de pixeles y sus valores cromáticos, el pixel ubicado en la posición [1,1], tiene un valor cromático promedio de (0,0,51), lo cual es igual a 17, en el caso del algoritmo actual, este sería considerado un gap. Tomando el ejemplo del pixel en la posición [1,3], tiene el calor cromático de (153,0,0), el cual tiene un valor cromático promedio de 51, el cual sobre pasa el valor umbral previamente definido.

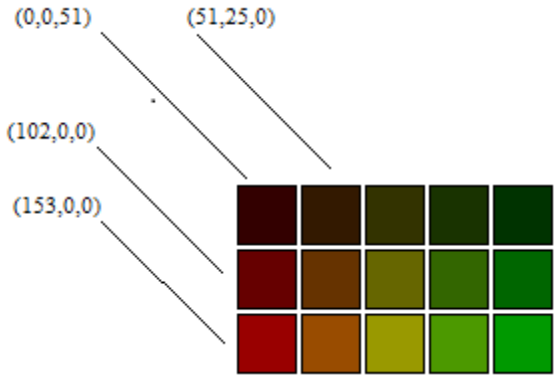


Figura 4, colores y sus valores por banda.

9.2 Oliveiras aplicado a imágenes satelitales Landsat 7 SCL.

Dado el análisis de las diferentes alternativas para solucionar el problema de *gaps* en las imágenes satelitales dañadas en el sensor del Landsat 7 ETM+, se seleccionó el algoritmo de Oliveiras debido a varias de sus cualidades. La primera es que no requiere de una imagen input. La segunda y más importante es que es el algoritmo más rápido de los propuestos; los algoritmos de Inpainting son relativamente rápidos para una imagen de tamaño promedio, pero a diferencia las imágenes satelitales que son de gran tamaño, aumentaría el tiempo de ejecución exponencialmente.

El algoritmo de Oliveiras realiza una convolución por medio de un kernel de difusión. Una vez que se tiene la imagen máscara, el algoritmo de Oliveiras buscará

un pixel gap, realizará un convolución usando los valores promedio de sus pixeles vecinos.

Primero, veamos la definición matemática de la convolución en el domino de tiempo discreto. Podemos ver en la fórmula 1, donde x[n] es señal de entrada, h [n] es nuestro valor del kernel de difusión, y[n] es el valor emitido. Tengamos en cuenta que multiplicamos los términos de x[k] por los valores de 0 a 255 de cada pixel y los sumamos.

$$y[n] = x[n] * h[n] = \sum_{k=-\infty}^{\infty} x[k] \cdot h[n-k]$$

Fórmula 1, definición matemática de la convolución.

El kernel de difusión tiene un tamaño de 3x3. Siempre el centro del kernel tendrá un valor ponderado de cero ya que esta casilla [2,2], está siempre ocupada por un pixel identificado como gap. En la figura 6 se puede observar que la casilla central es claramente un gap, a pesar de que pixeles 7, 8 y 9 también son *gaps*, el valor del gap central será calculado e pesar de que los vecinos puedan o no ser *gaps*. En este caso, los pixeles 1,2 y 3, 4 y 6 tendrán un valor mayor al de los *gaps*, su valor será multiplicado por la matriz del kernel de difusión y sumado lo cual resultará en el valor del pixel central.

a	b	a
b	0	b
a	b	a

c	c	c
c	0	c
c	c	c

Figura 5. ejemplo Kernel de difusión.

Por otro lado, el Kernel de difusión puede ser adaptable con los valores que se estimen, esto dependerá si se decide darle mayor peso a los pixeles que se encuentran en diagonal del pixel central, o se prefiere dar una ponderación equitativa a todas las otras casillas. En la figura 7, se puede observar dos tipos básicos de kernel de difusión, los cuales tiene un valor ponderado de a = 0.073235, b = 0.176765 y c = 0.125. El valor central siempre tendrá el valor de cero, debido a que representa al gap.

Usando el ejemplo de la Figura 5, suponiendo que los valores RGB de los pixeles dentro de la matriz de desplazamiento son los siguientes y que se usará el Kernel de difusión uno, el cual tiene valores alternados de a y b.

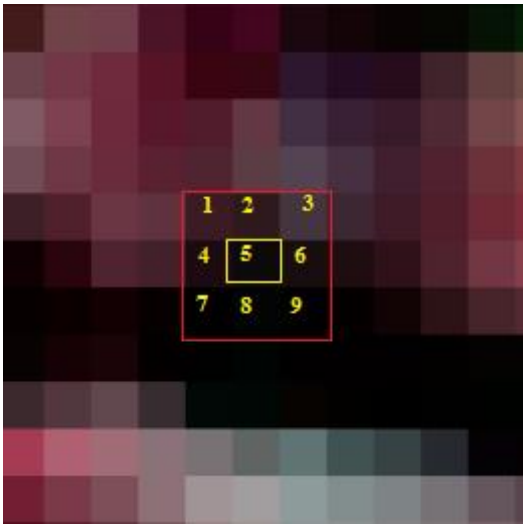


Figura 6, pixeles seleccionados para la convolución.

Después de una primera iteración, el pixel en la posición [2,2] identificado como gap cambió de un valor de 0 a 75. El algoritmo de Oliveira convolucionará todos los pixeles identificados como *gaps*, una vez que acabe, volverá a empezar, pero esta ocasión con los nuevos valores calculados por la convolución, la repetición de este proceso acercará a los *gaps* a un valor más asertivo debido a la convolución difundida por todos los pixeles vecinos. Oliveiras propone un numero de

iteraciones de 200, cuando se tienen secciones de *gaps* muy grandes, pocas iteraciones no logran rellenar la totalidad de datos, esto porque al ser todos sus vecinos *gaps*, su nuevo valor será muy cercano al cero. En la práctica, se verificó experimentalmente que es por esta razón es necesario llegar a las 200 iteraciones.

Posición del pixel	**R**	**G**	**B**	**Valor del Kernel**	**Valor promedio**	**Valor total**
1,1	70	80	50	a=0.073235	(70 *a)+(80*a)+(50*a)	14.647
1,2	70	77	47	b=0.176765	(70 *b)+(77*b)+(47*b)	14.20759
1,3	71	79	48	a=0.073235	(71 *a)+(79*a)+(48*a)	14.50053
2,1	50	20	30	b=0.176765	(50 *b)+(20*b)+(30*b)	7.3235
2,2	0	0	0	0	0	0
2,3	43	20	20	b=0.176765	(43 *b)+(20*b)+(20*b)	6.078505
3,1	13	23	40	a=0.073235	(13 *a)+(23*a)+(40*a)	5.56586
3,2	15	32	37	b=0.176765	(15 *b)+(32*b)+(37*b)	6.15174
3,3	45	30	21	a=0.073235	(45 *a)+(30*a)+(21*a)	7.03056
					Nuevo Valor Gap	**75.50529**

Tabla 4, Rellenado de pixeles por Convolución.

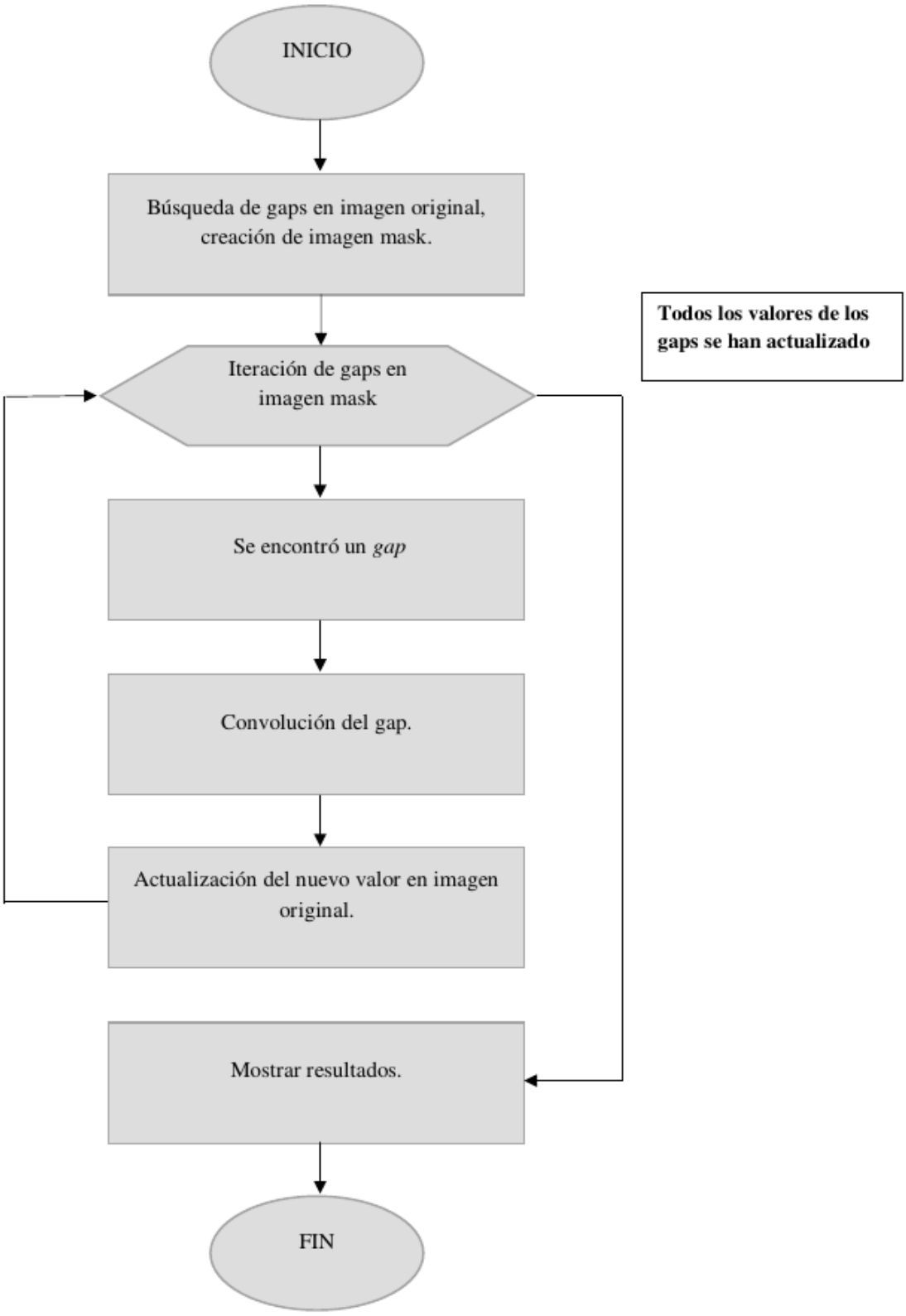


Figura 7, Diagrama de flujo algoritmo Inpainting Oliveiras DirectCompute.

9.3 GPU Inpainting.

En la actualidad el uso de GPU para resolver problemas de programación se ha convertido en una herramienta altamente efectiva para diferentes propósitos, capaz de mejorar el rendimiento de una amplia variedad de aplicaciones paralelas más allá de los gráficos [19]. La arquitectura de los procesadores ha liberado el camino para probar nuevas capacidades del cómputo de la GPU, y proporciona la infraestructura que permite el uso de DirectCompute. Al mismo tiempo, la API de DirectX de Microsoft ha madurado en su interfaz para permitir el uso de gráficos en plataformas Windows, tanto para videojuegos como para aplicaciones graficas tales como edición de fotos y video.

La introducción de DirectCompute permite a los desarrolladores tomar ventaja de las capacidades masivas del cómputo en paralelo de las GPU, sin la necesidad de usar una API de cálculo separada.

DirectCompute usa la funcionalidad del cálculo de la GPU como un shader de cómputo. El shader de cómputo no está conectado específicamente a ninguna etapa de la tubería de gráficos, pero interactúa con las otras etapas a través de recursos gráficos, como renderizar objetivos, búferes y texturas.

A diferencia de un shader de vértices (que se ejecuta una vez para cada vértice de entrada), o un sombreador de píxeles (que se ejecuta una vez para cada píxel), el shader de cómputo no necesita tener un mapeo entre los datos que está procesando y los hilos que están haciendo el procesamiento.

Un hilo puede procesar uno o varios elementos de datos, y la aplicación puede controlar directamente cuántos hilos se utilizan para realizar el cálculo.

El shader de cómputo también permite el acceso a la memoria RAM, en particular la capacidad de realizar escrituras en cualquier ubicación en un búfer (también conocido como escrituras dispersas). La característica principal de DirectCompute es la memoria compartida con los grupos de hilos (a la que se hace referencia de ahora en adelante simplemente como memoria compartida). Esto permite que los grupos de hilos compartan datos, esto puede reducir significativamente los requisitos de ancho de banda.

En conjunto, estas características permiten una estructura de datos y algoritmos más complejos que anteriormente no eran posibles en DirectCompute, y puede mejorar el rendimiento de la aplicación considerablemente. Este es el caso del algoritmo de Inpainting de Oliveiras el cual costa de dos procesos primordiales: a) la generación de la imagen mascara, y b) el rellenado de los pixeles por medio del Kernel de difusión.

```
case FINDGAP:
        {
            int d = (int)(256 * (dot(float4(1, 1, 1, 0), Input[id.xy])
/ 3));
                if (d <= Operation.y)
                    Color = float4(1, 1, 1, 0);
        }
        break;
```

Código 1, DirectxCompute shader que crea la imagen mascara, busca los gaps que tengan un valor inferior al umbral.

En el caso de la creación de la imagen mascará, el shader de Direct Compute busca los *gaps* acordes con un valor de umbral default considerado entre los valores de 20 a 30(ver figura 4) los cuales nos permitirá seleccionar los *gaps* para después ser rellenados por un valor absoluto, de igual manera preseleccionado. En esta tesis se eligió el color de relleno de los *gaps* como el 255 (blanco 255,255,255, ver Código 1) en todas las bandas, esto por ser un color imposible

para un pixel tomado de una imagen satelital de la faz terrestre, con esto se aislarían los *gaps* para ser filtrados una vez que se pase a al procesamiento del kernel de difusión.

```
[numthreads(16,16,1)]
```

Código 2, selección de hilos en el GPU (16x16).

En la Tabla 2, se ha analizado como se lleva a cabo la convolución por cada uno de los pixeles, y como las iteraciones determinaran en gran manera como los pixeles son llevamos de su valor actual, al rellenado por la convolución. DirectCompute a diferencia de un sistema de cómputo lineal; gracias a su ventaja de computo masivo, ejecutará la convolución para un número mayor de pixeles, lo que acorta significativamente el tiempo de ejecución. Para el presente trabajo se optó por procesar 256 hilos a la vez (Codigo 2), esto por ser un numero de bits compatible para la mayoría de los procesadores GPU de la actualidad (Intel, NVidia, AMD), DirectXCompute correrá los 256 hilos a la vez, a diferencia del computo tradicional, mientras el hilo con el ID global 1 puedo aun seguir procesándose, el hilo con el ID 2 podría haber terminado, lo que producirá que ciertos pixeles sean convolucionados con información caduca. Esto se corrige con las iteraciones y el paso de las tuberías de datos de un shader a otro.

```
if (d >= 250){
    for (int j = 0; j < 3; j++)
        for (int i = 0; i < 3; i++)
            Suma+=ImgRead(localid.xy+int2(i,j)) * Kernel[i][j];

    Output[id.xy] = Suma + C;
}
else
    Output[id.xy] =  Input[id.xy];
```

Codigo 3, DirectXCompute de Convolucion Oliveira.

Ya que el paso de FINDGAP a rellenados los *gaps* de nuestra mascara, la convolución empezara a rellenar los *gaps* pero en la imagen original. Una vez que el shader termine con todos los pixeles de la imagen, la imagen resultante será tomada como textura de entrada para el shader siguiente, esto se repetirá las veces que se crea necesario, para este trabajo se ha optado por 200 interacción de la tubería del shader; Oliveiras demostró que después de 200 iteraciones los valores de intercambio en la convolución son insignificante. (ver Código 3).

10. Experimentación

En el presente trabajo se usaron diferentes escenarios para probar el algoritmo de Oliveiras en la plataforma Direct Compute GPU, la experimentación consiste en dos escenarios, el primero son imágenes de Lena dañadas de manera arbitraria usando diferentes anchos para los gaps, los gaps abarcan el ancho y alto de la imagen. Para el segundo escenario fueron usadas imagen del Landsat 7 obtenidas del sitio web Glovis [25] donde las imágenes satelitales presentan gaps en prácticamente toda la imagen; la primera imagen está ubicada en las Bahamas, la segunda en el Amazonas.

Descripción de la Imagen	**Tamaño**	**Daño**	**Ancho Gap**
Lenna 1	512x512	5%	1 Pixeles
Lenna 2	512x512	7%	2 Pixeles
Imagen Satelital GloVis 1	14301x16241	20%	2-22 Pixeles
Imagen Satelital GloVis 2	13901x16021	20%	2-22 Pixeles

Tabla 5, Descripción casos experimentación.

10.1 Escenario Lenna

En la primera prueba la imagen fue dañada de manera deliberada, poniendo atención al ancho de la línea del daño.

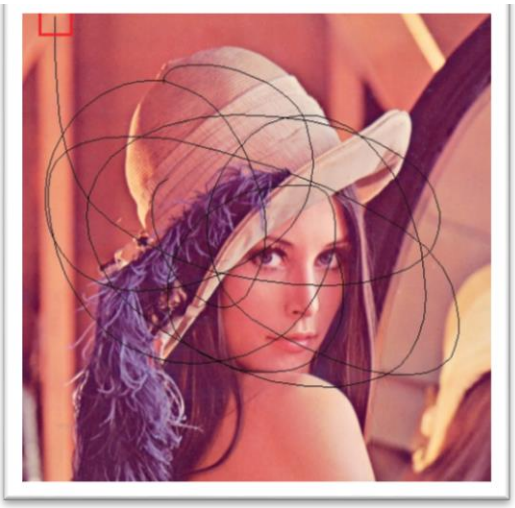


Figura 8a, Lenna, Imagen Dañada.

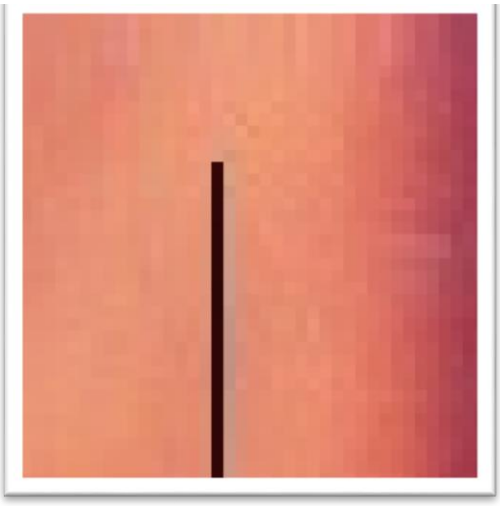
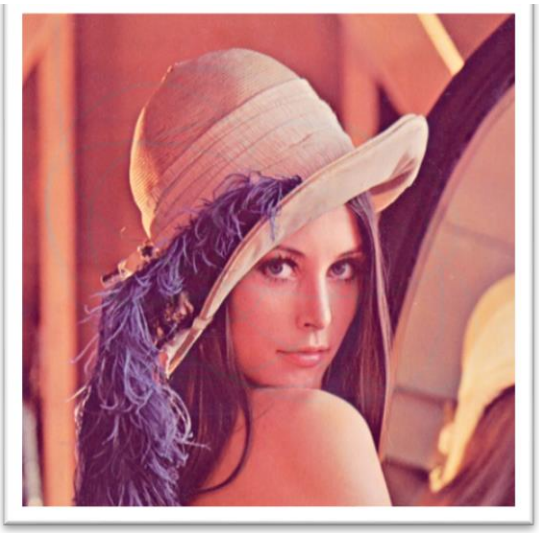


Figura 8b, Ancho de pixel del daño ocasionado.

Realizando un acercamiento al daño creado deliberadamente al pixel, se puede observar el gap con un ancho de solo un pixel.

Después de correr el algoritmo propuesto de Oliveiras con Direct Compute GPU, puede observarse que el daño ocasionado es reparado fácilmente, en un tiempo de 1 segundo aproximadamente. En la figura 8b, se puede observar que el daño ocasionado deliberadamente con un pixel de ancho, generó una pequeña sombra de color gris, lo cual no es suficientemente oscura para ser capturada por el proceso de generación de la imagen mascara, por lo que la imagen resultada es reparada pero la sombra gris permanece allí.



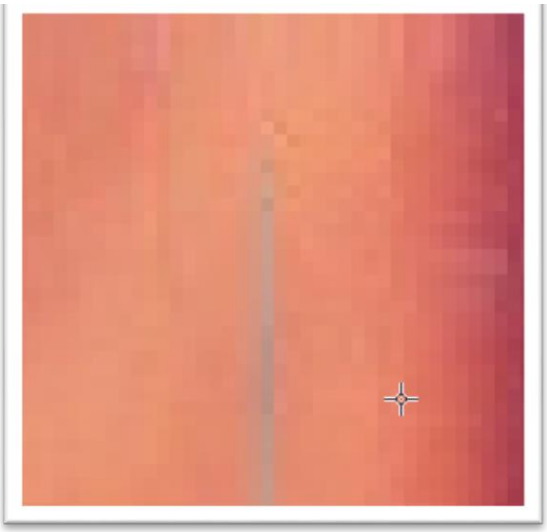


Figura 9a, Lenna, Imagen reparada.

Figura 9b, Lenna, Acercamiento pixel reparado.

En el acercamiento de la figura 9b, el pixel es reparado, pero se mantiene la sombra gris generada por el programa de edición de imágenes.

Después de la primera prueba, se experimentó con un daño más grande, esta vez el daño generado fue de dos pixeles de ancho, el editor de imágenes en el cual se ocasionó el daño esta ocasión no genero el pixel “sombra” que en el escenario pasado.

Figura 10a, Lenna, Imagen dañada, 2 pixeles de ancho. Figura 10b, Lenna, Imagen dañada, 2 pixeles de ancho acercamiento. Figura 10c, Lenna, Imagen reparada.

Se puede observar en la figura 10b, que el daño ocasionado deliberadamente no generó la sombra del escenario anterior.

Después de correr el algoritmo de Oliveiras sobre el escenario con dos pixeles de ancho, se puede observar a simple vista que no existe ningún tipo de vestigio del gap existente.

Se tomó un escenario real de una imagen Landsat 7 dañada. A simple vista se observa el daño del lente en la mayor parte de la imagen. Únicamente en el centro de la imagen la imagen es clara.

10.2 Escenario LandSat 7 SLC

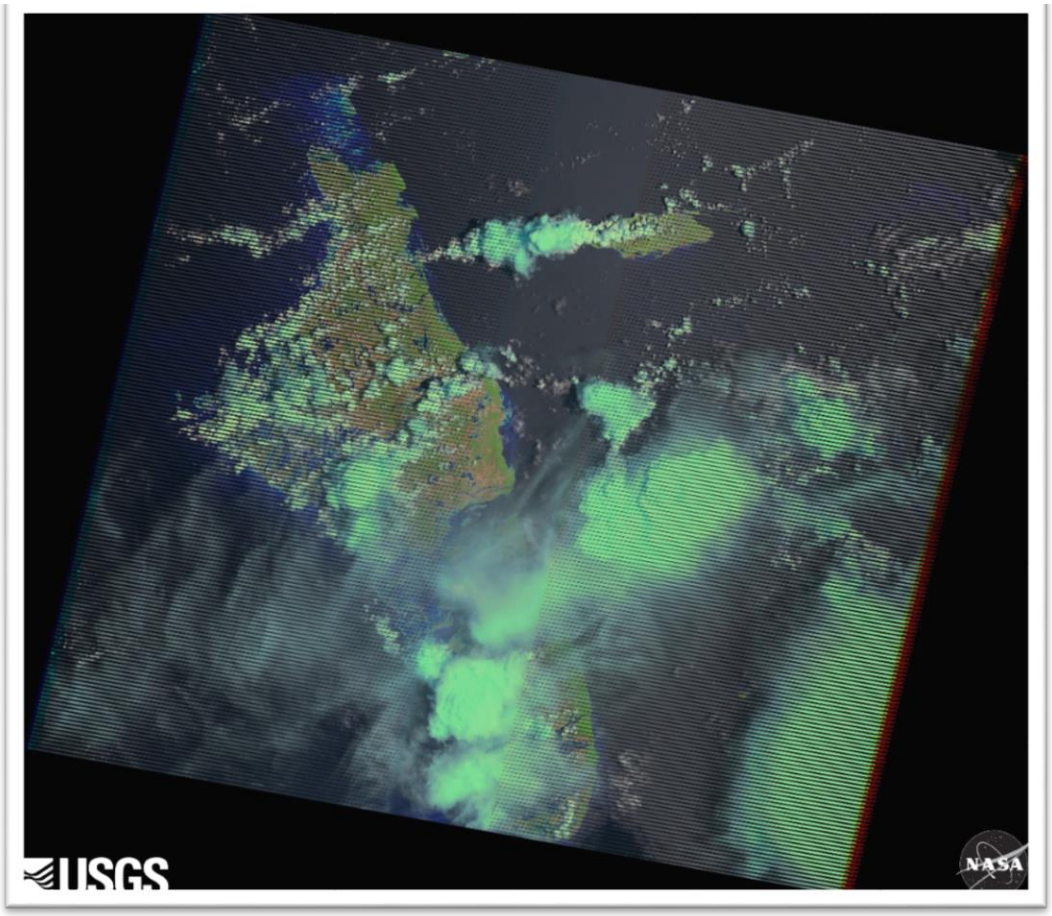


Figura 11, Imagen Landsat 7 dañada. Bahamas. 2011/11/20, Ancho del daño 2 a 22 pixeles, UTM Zona 18, Sensor ETM, Total líneas ancho 14301, total líneas largo 16241. Id imagen LE07_L1TP_013043_20111120_20161205_01_T1

https://glovis.usgs.gov/app

Se tomó la parte superior de la imagen satelital del Landsat 7 como el siguiente escenario donde se encontraron nubes, tierra y mar. Después de ejecutar el algoritmo de Oliveiras Direct Compute. El resultado se puede ver en la figura 11b.

Haciendo una comparativa se puede observar la imagen cuando se le aplicó el algoritmo de relleno de los *gaps*, en las partes donde aparece tierra y agua.

Para el segundo escenario de ejemplo real se tomó como ejemplo una imagen satelital dañada con muchas nubes, esto para ver resolvía el algoritmo un escenario homogéneo.

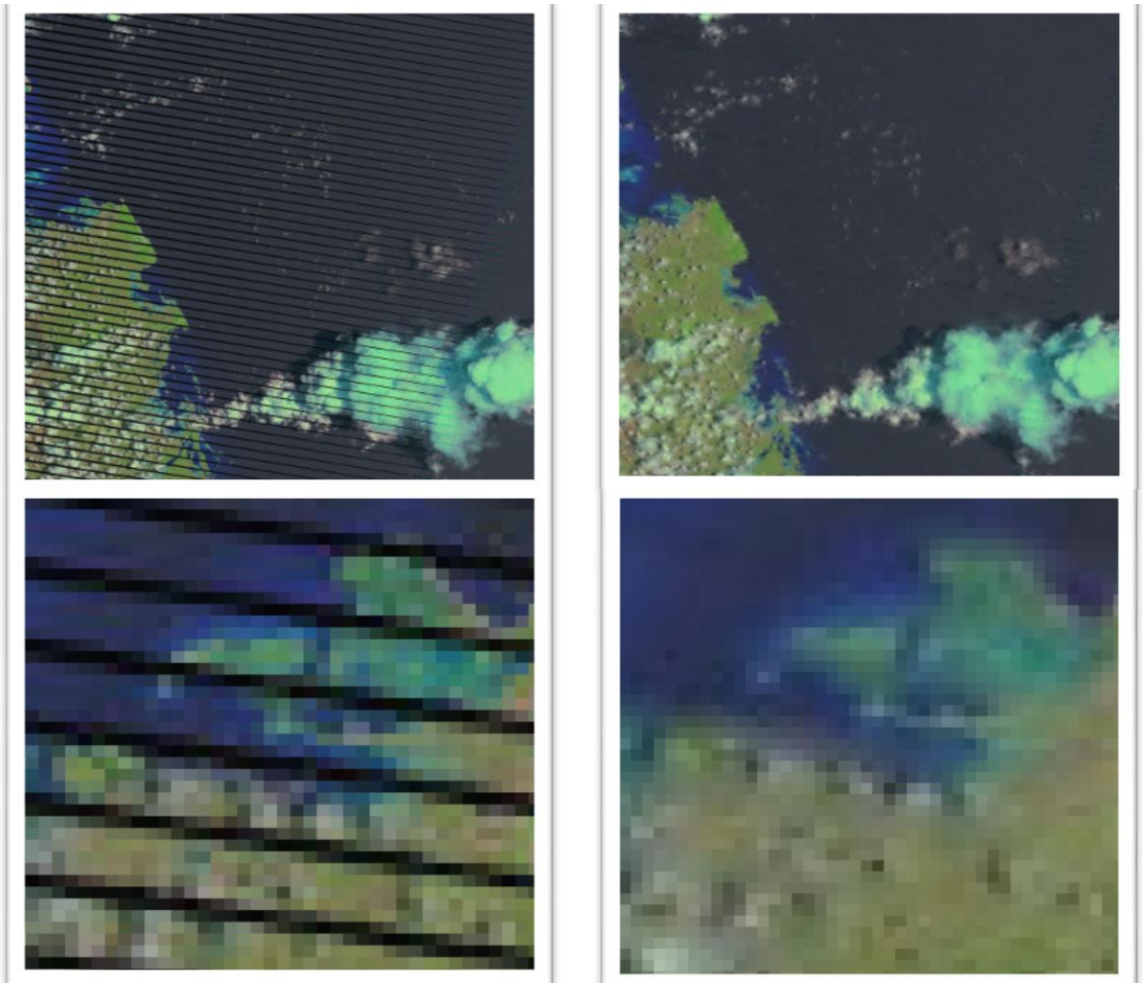


Figura 12a, Imagen Landsat 7 dañada, sección superior. Figura 12b, Imagen Landsat 7 reparada, sección superior. Figura 12c, Acercamiento imagen Landsat 7 dañada. Figura 12d, Acercamiento Imagen Landsat 7 reparada, https://glovis.usgs.gov/app

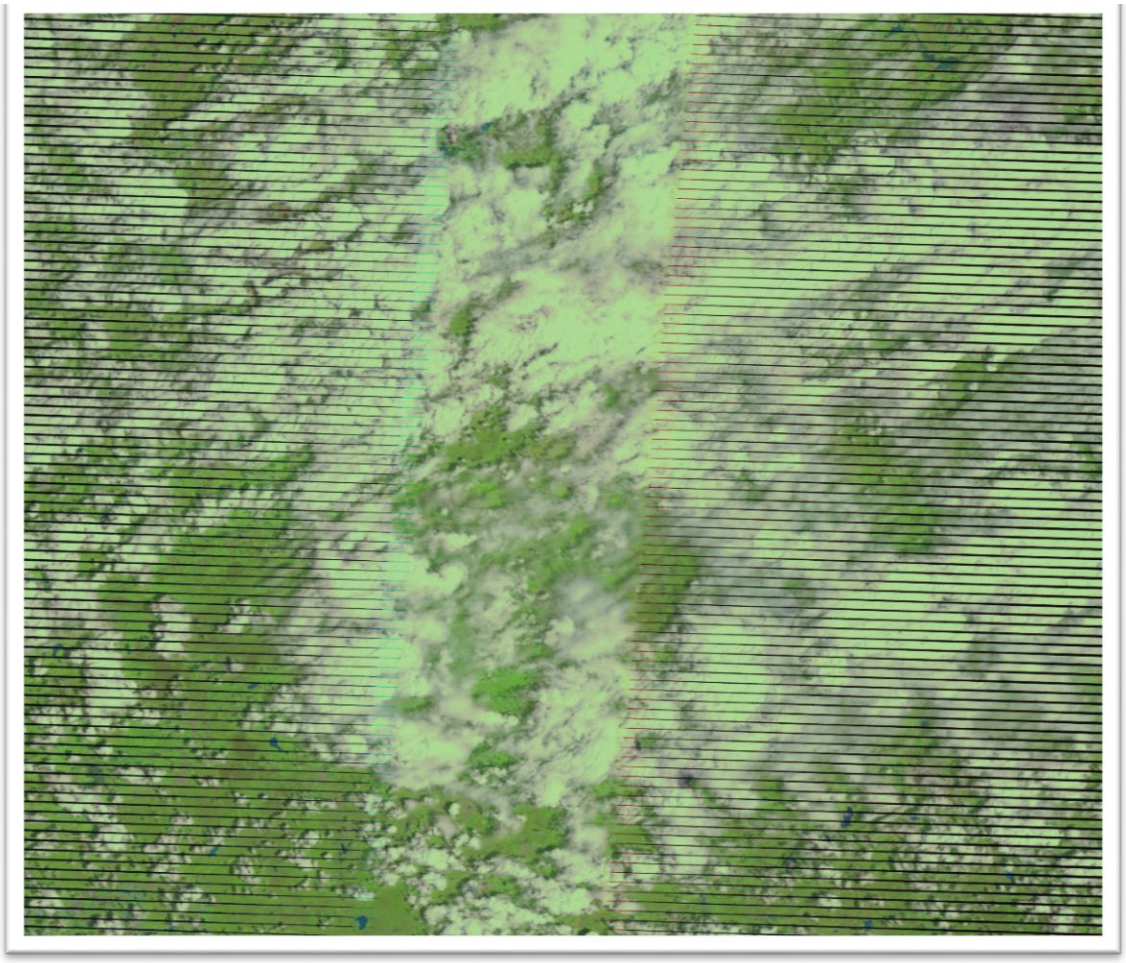


Figura 13, Imagen Landsat 7 dañada. Amazonas. 2016/12/04, Ancho del daño 2 a 22 pixeles, UTM Zona 19, Sensor ETM, Total líneas ancho 13901, total líneas largo 16021. Id imagen LE07_L1TP_002063_20111209_20161204_01_T, https://glovis.usgs.gov/app

Se puede observar que en los cuatro escenarios las imágenes satelitales dañadas fueron reparadas, el criterio para considerar que el proceso fue exitoso es que los *gaps* desaparecieron independiente si la imagen original era homogénea, como el caso de la figura 19, o heterogénea en el caso de la figura 15. En la figura 21, se muestra una forma visual de validar de manera cualitativa la imagen producida como resultado del algoritmo propuesto y la imagen dañada.

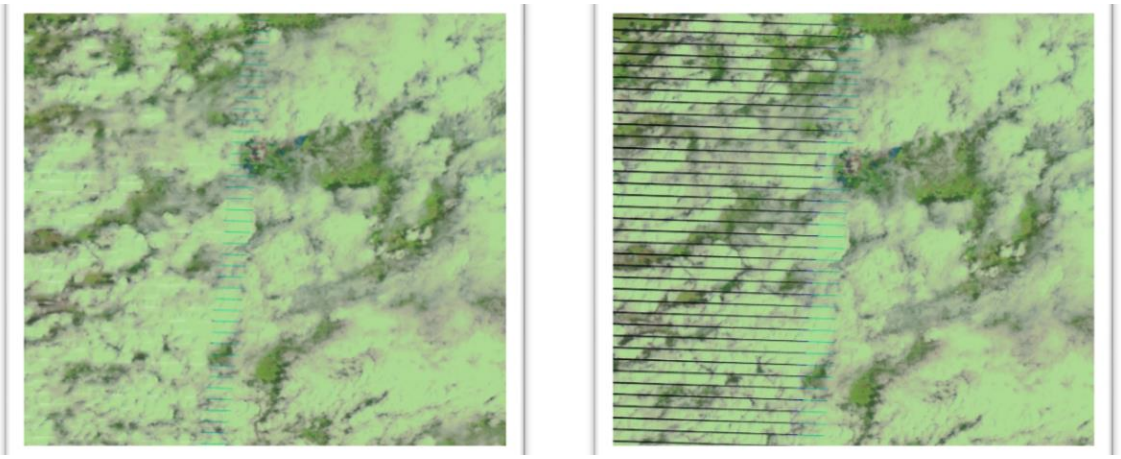


Figura 14a, Imagen Landsat 7 dañada. Figura 14b imagen Landsat 7 reparada.

En los casos donde fue reparada Lenna, se pudo comparar el resultado contra la imagen previa al daño (imagen original) al no existir una imagen ejemplo en las imágenes satelitales, se hace prácticamente imposible conocer con exactitud el índice de reparación del algoritmo.

Para conocer el porcentaje de reparación se usó el valor absoluto de la diferencia del valor del pixel reparado contra el original, después fue sumado y dividido entre el valor total el cual es 255. Se pudo comprobar que los pixeles tiene una proximidad de valor que esta entre el 80 al 90 por ciento en los casos de Lenna.

Caso	**Tiempo Ejecución**	**Porcentaje de reparación, Comparado contra imagen original**
Lenna 1	3 Segundos	80%
Lenna 2	3 Segundos	93%
Imagen Satelital GloVis 1	3 Segundos	NA
Imagen Satelital GloVis 2	3 Segundos	NA

Tabla 6, Porcentaje de reparación

11. Conclusiones

En este trabajo de tesis, se introdujo una modificación del algoritmo de Oliveiras, desarrollado en la plataforma GPU Direct Compute, utilizado para el rellenado de *gaps* en imágenes satelitales Landsat 7 ETM+.

Se verificó de manera experimental que el algoritmo propuesto llevó a cabo de forma eficiente y precisa el rellenado de gaps con cada una de las imágenes probadas.

A diferencia del algoritmo de Oliveiras, en este trabajo se propuso crear la máscara automáticamente.

La implementación del algoritmo utilizando tecnología Direct Compute permitió una reducción en el tiempo de ejecución de 100 a 1 con respecto a su implementación serial utilizando un núcleo del CPU. Por ejemplo, la imagen de prueba de Lenna originalmente se tardaba 120 minutos, mientras que con el algoritmo propuesto el tiempo de ejecución fue de 2 segundos. Este hecho es muy importante, ya que las imágenes satelitales Landsat 7 ETM+, comparativamente son mucho más grandes en sus dimensiones (64 millones de pixeles aproximadamente).

11.2 Limitaciones y trabajo futuro

Este trabajo tiene la limitante de trabajar exclusivamente bajo Windows, utilizando Visual Studio bajo el lenguaje C++. Se pretende llevarlo a otro lenguaje multiplataforma como Java.

Al crearse la máscara de manera automática, en algunas ocasiones, se puede llegar a no considerarse todos los puntos que corresponden a los *gaps* en las imágenes

satelitales. Esto porque se utiliza un umbral para realizar esta selección, y por la naturaleza de los *gaps* en las imágenes Landsat 7, algunos pixeles se pueden llegar a confundir con elementos distintos a los *gaps*. Se trabajará en un futuro en proponer otro mecanismo que elimine esta limitante.

El algoritmo propuesto ya utiliza una serie de valores de sus parámetros por omisión y no se utiliza ninguna interfaz gráfica GUI. En un futuro, se desarrollará una GUI que incorpore el algoritmo propuesto, permitiendo modificar los valores de los parámetros y permitir realizar otro tipo de experimentación.

A pesar de obtener buenos resultados en el rellenado de huecos en las imágenes satelitales, no se realizó una comparación contra otros algoritmos desarrollados previamente en el estado del arte, debido a que finalmente quedo fuera del alcance de este trabajo de tesis. Se trabajará en la implementación de algún algoritmo de gapfilling para poder realizar una comparación numérica, y en base a los resultados obtenidos generar un artículo para enviarlo a congreso o revista.

12. Bibliografía

[1] U.S. Department of the Interior | U.S. Geological Survey, 2018 URL: https://landsat.usgs.gov/landsat-7, Landsat7.

[2] landsat 8 Satellite, 2018, U.S. Department of the Interior | U.S. Geological Survey URL: https://landsat.usgs.gov https://landsat.usgs.gov/landsat-8 USGS , landsat 8 Satellite.

[3] Mi Sun Park, Esther Sekyoung Choi & Yeo-Chang Youn (2013) REDD+ as an international cooperation strategy under the global climate change regime,Forest Science and Technology, 9:4, 213-224, DOI: 10.1080/21580103.2013.846875.

[4] Scaramuzza, P., E. Micijevic, and G. Chander. 2004. "SLC Gap-Filled Products Phase One Methodology." Accessed January 12, 2012.http://landsat.usgs.gov/documents/SLC_Gap_Fill_Methodology.pdf

[5] Chen, J., X. Zhu, J. E. Vogelmann, F. Gao, and S. Jin. 2011. "A Simple and Effective Method for Filling Gaps in Landsat ETM+ SLC-off Images." Remote Sensing of Environment 115 (4): 1053–1064. doi: 10.1016/j.rse.2010.12.010

[6] Zhu, X., D. Liu, and J. Chen. 2012. "A New Geostatistical Approach for Filling Gaps in Landsat ETM+ SLC-off Images." Remote Sensing of Environment 124 (Sep.): 49–60. doi: 10.1016/j. rse.2012.04.019.

[7] Filling the Gaps to use in Scientific Analysys. 2018,U.S. Department of the Interior | U.S. Geological Survey URL: https://landsat.usgs.gov/filling-gaps-use-scientific-analysis, Filling the Gaps to use in Scientific Analysys.

[8] Maxwell, S. K., G. L. Schmidt, and J. C. Storey. 2007. "A Multi-scale Segmentation Approach to Filling Gaps in Landsat ETM+ SLC-off Images." International Journal of Remote Sensing 28 (23): 5339–5356. doi:10.1080/01431160601034902.

[9] Bédard, F., G. Reichert, R. Dobbins, and I. Trépanier. 2008. "Evaluation of Segment-based Gapfilled Landsat ETM+ SLC-off Satellite Data for Land Cover Classification in Southern Saskatchewan, Canada." International Journal of Remote Sensing 29: 2041–2054. doi:10.1080/01431160701281064.

[10] Zeng, C., H. Shen, and L. Zhang. 2013. "Recovering Missing Pixels for Landsat ETM+ SLC-off Imagery Using Multi-Temporal Regression Analysis and a Regularization Method." Remote Sensing of Environment 131 (Apr.): 182–194. doi: 10.1016/j.rse.2012.12.012.

[11] Pringle, M. J., M. Schmidt, and J. S. Muir. 2009. "Geostatistical Interpolation of SLC-off Landsat ETM+ Images." ISPRS Journal of Photogrammetry and Remote Sensing 64 (6): 654–664. doi: 10.1016/j.isprsjprs.2009.06.001.

[12] A. Criminisi, P. Perez, and K. Toyama, 2003, "Object removal by exemplarbasedinpainting," in Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., vol. 2., pp. II-721–II-728.

[13] Jian Sun, Lu Yuan, Jiaya Jian, and Heung-Yeung Shum.2005, Image completion with structure propagation. In Proceedings of ACM Conf. Comp. Graphics .

[14] C. Zhang , W.Li and D.Travis. 2007 Gaps-fill of SLC-off Landsat ETM + satellite image using a geostatistical approach.

[15] Martin Enrique Romero-Sanchez, Raul Ponce-Hernandez, Steven E. Franklinb and and Carlos Arturo Aguirre-Salado Comparison of data gap-filling methods for Landsat ETM+ SLC-off imagery for monitoring forest degradation in a semi-deciduous tropical forest in Mexico.

[16]Pritika Patel, Ankit Prajapati and Shailendra Mishra. 2012, Review of DifferentInpainting Algorithms.
[17]Wei Yao, Ji-xiang Sun, Gang Zou, Shuhua Teng and Gong-jian Wen. 2010. PDE image inpainting with texture synthesis based on damaged region classification. 10.1109/ICACC.2010.5487104.
[18]M. Bertalmío, G. Sapiro, V. Caselles and C. Ballester. Proceedings of SIGGRAPH 2000, New Orleans, USA, July 2000.
[19]NVIDIA, DirectCompute Programming Guide. 2010, PG-05620-001_v3.2.
[20]Oliviera MM, Bowen B, McKenna R, Chang YS. 2001, Fast digital image inpainting. Proceedings of the International Conference on Visualization, Imaging and Image Processing (VIIP '01); pp. 261–266
[21]Vellmer S, Stirnberg R, Edelhoff D, Suter D. 2017, Comparative analysis of isotropic diffusion weighted imaging sequences.
[22]Blaikie, P. 1985, The Political Economy of Soil Erosion in Developing Countries. London: Longman.
[23]Glantz, M. 1977. Desertification: Environmental Degradation in and around Arid Lands. Boulder: Westview Press.
[24]Data One, Data Observation Network for Earth, 2018, https://www.dataone.org/software-tools/envi.
[25]Esri, ArcGis, 2018, https://www.esri.com/es-es/arcgis/products/arcgis-online/pricing.
[26]GloVis,2018, US Department of the interior. https://glovis.usgs.gov/app