



Universidad Austral de Chile

Facultad de Ciencias de la Ingeniería
Escuela de Ingeniería Civil en Informática

EVALUACIÓN Y DESARROLLO DE ALGORITMOS ADAPTATIVOS PARA LA CREACIÓN DE IMÁGENES PANORÁMICAS CASO DE USO: TORRES DEL PAINE

Proyecto para optar al título de
Ingeniero Civil en Informática

PROFESOR PATROCINANTE:
CHRISTIAN LAZO RAMÍREZ
INGENIERO GESTIÓN INFORMÁTICA
DOCTOR EN INGNIERÍA TELEMÁTICA

PROFESOR CO-PATROCINANTE
MIRKO GUEREGAT
INGENIERO CIVIL EN INFORMÁTICA

PROFESOR INFORMANTE
MARÍA ELIANA DE LA MAZA WERNER
INGENIERO CIVIL EN INFORMÁTICA
MAGÍSTER EN INFORMÁTICA EDUCATIVA

RENÉ NICOLÁS FUENTES KEIM
VALDIVIA – CHILE
2021

AGRADECIMIENTOS

El presente trabajo para optar al título de Ingeniero Civil en Informática va destinado a:

Profesora Marianna Villarroel por ser un apoyo fundamental en cada pequeño paso que daba, aconsejarme cuando necesitaba alguna palabra de aliento, por confiar en mí cuando tenía muchas dudas y por siempre darme la oportunidad de auto superarme cuando más necesitaba demostrarme a mí mismo que podía lograrlo.

Profesora María Eliana de la Maza por extender su mano para ayudarme cuando más lo necesitaba, por darse el tiempo para resolver mis dudas y guiarme de la mejor forma en sus asignaturas.

Mis padres, que sin sus esfuerzos y sus constantes luchas día a día nada de esto podría haberlo alcanzado.

Compañeros de universidad que aprendí bastante de cada amistad, en especial a mi amigo Marcelo Arriagada, que estuvo apoyándome durante todo el transcurso de esta etapa, brindándome su conocimiento y su punto de vista distinto al mío, trabajando siempre a la par tanto en las asignaturas como en ámbitos "laborales".

A mi amiga Giannina que tan solo con su presencia me motivaba a seguir adelante en los estudios, exigirme cada vez más y superarme con creces.

Profesores guías, que estuvieron atentos ante cualquier duda que tuviera y mostrarme soluciones ante toda duda existencial que tuviera.

Y para finalizar, quiero agradecerme a mí mismo ... por creer en mí, por nunca renunciar, por el trabajo duro día tras día, por siempre dar más de lo que tengo y dar más de lo que soy.

ÍNDICE

AGRADECIMIENTOS	I
ÍNDICE	I
ÍNDICE DE TABLA.....	III
ÍNDICE DE FIGURAS.....	IV
RESUMEN.....	V
ABSTRACT	VI
1. INTRODUCCIÓN	1
1.1. Motivación.....	1
1.2. Impacto	2
1.3. Objetivos	2
1.3.1. Objetivos generales.....	2
1.3.2. Objetivos específicos	2
2. MARCO TEÓRICO.....	3
2.1 Procesamiento de imagen	7
2.2 Detección de puntos clave	7
2.3. Descriptores de invariantes locales (SIFT, SURF, ORB.)	8
2.3.1 SIFT (Scale Invariant Feature Transform).....	8
2.3.1.1 Detección de espacio extremo a escala.....	9
2.3.1.2 Localización de puntos claves	10
2.3.1.3 Asignación de orientación	11
2.3.1.4 Descriptor de puntos clave.....	11
2.3.1.5 Coincidencia de puntos clave	12
2.3.2 SURF (Speeded Up Robust Features)	12
2.3.3 ORB (Oriented FAST and Rotated BRIEF)	14
2.4. Coincidencia de las características	15
2.5. Estimación de homografía usando RANSAC (Random Sample Consensus) ..	15
2.6. Deformación de la perspectiva	16
2.7. Mezcla de imágenes	17
2.8. Comparación de algoritmos en distintos escenarios.....	17
3. SOLUCIÓN PROPUESTA.....	22
3.1 Descripción de la problemática.....	22
4. TECNOLOGÍAS Y PRUEBAS TEMPRANAS.....	24
4.1 Tecnologías	24
4.2 Pruebas tempranas	25
5. ANÁLISIS Y DISEÑO	31
6. IMPLEMENTACIÓN	33
7. RESULTADOS	38
7.1 Resultados algoritmo adaptativo con Dataset de prueba	38
7.2 Resultados algoritmo adaptativo con Dataset de Torres del Paine	42
8. CONCLUSIONES	48
9. TRABAJOS FUTUROS Y MEJORAS	50
10. REFERENCIAS	51
ANEXOS	55

Anexo A: Tabla de métricas e imágenes panorámicas generadas con algoritmo SIFT versión 1	55
Anexo B: Tabla de métricas e imágenes panorámicas generadas con algoritmo SIFT versión 2	60
Anexo C: Tabla de métricas e imágenes panorámicas generadas con algoritmo con técnica SIFT versión 3	65
Anexo D: Tabla de métricas e imágenes panorámicas generadas con algoritmo ORB	69
Anexo E: Tabla de actividades del proyecto de título y sus descripciones	74
Anexo F: Tabla de secciones del algoritmo adaptativo	75
Anexo G: Tabla de métricas e imágenes panorámicas con algoritmo adaptativo	76
Anexo H: Imagen PCA (Principal Component Analysis)	78

ÍNDICE DE TABLA

Tabla	Página
Tabla 1: Comparación del algoritmo de extracción de características basado en parámetros cuantitativos	18
Tabla 2: Comparación de tres métodos de detectores de características utilizados en la unión de imágenes.....	18
Tabla 3: Resultados estadísticos de la coincidencia de puntos de características.....	19
Tabla 4: Comparación del análisis de rendimiento de los detectores klt, harris y sift para imágenes de interior	19
Tabla 5: Comparación de costo en tiempo medido en milisegundos.....	20
Tabla 6: Imágenes que sufren modificaciones de escala e iluminación, tanto para escenarios internos como externos.....	20
Tabla 7: Comparación de cambios de iluminación, Data representa la repetibilidad y el promedio de repetibilidad	20
Tabla 8: Tabla resumen de las métricas obtenidas por algoritmos testeados.....	30
Tabla 9: Resultados métricas algoritmo adaptativo con tercer Dataset	38
Tabla 10: Resultados métricas algoritmo adaptativo con Dataset Torres del Paine	40
Tabla 11: Métricas algoritmo adaptativo Dataset Torres del Paine	43
Tabla 12: Métricas algoritmo adaptativo una imagen difuminada.....	45
Tabla 13: Métricas algoritmo adaptativo tres imágenes difuminadas.....	47

ÍNDICE DE FIGURAS

Figura	Página
Figura 1: Panorámica de Cincinnati en daguerrotipo.....	3
Figura 2: Panorámica con placas húmedas	4
Figura 3: Panorámica generada evitando movimientos bruscos	5
Figura 4: Panorámica en marte con 1800 millones de píxeles.....	6
Figura 5: Diagrama de etapas de creación de imagen panorámica	7
Figura 6: Detección de puntos clave	8
Figura 7: Diferencia de Gaussiana	9
Figura 8: Máximos y mínimos de las imágenes de diferencia de Gaussiana.....	10
Figura 9: Etapas de selección de puntos claves	11
Figura 10: Emparejamiento de manchas SURF	13
Figura 11: Pirámide de imagen con múltiples escalas	15
Figura 12: Comparación de rotación de imágenes	21
Figura 13: Velocidades vientos año 2019 Puerto Natales.....	23
Figura 14: Dron DJI FC 2103	25
Figura 15: Dataset de pruebas capturadas por el dron DJI.....	26
Figura 16: Zona de captura de imágenes del dron DJI.....	26
Figura 17: Dataset Torres del Paine	27
Figura 18: Gráfico de tiempo de ejecución por algoritmo testeado	28
Figura 19: Gráfico de uso de memoria por algoritmo testado	29
Figura 20: Gráfico uso de disco por algoritmos testeados	29
Figura 21:Imagen panorámica obtenida con método SIFT v1	30
Figura 22: Diagrama algoritmo adaptativo	33
Figura 23: Diagrama main.py	35
Figura 24: Diagrama stiching.py	36
Figura 25: Diagrama rotación.py	37
Figura 26: Panorámica resultante con tercer Dataset de prueba 1014x760px	39
Figura 27: Panorámica resultante con tercer Dataset de prueba 253x190px	39
Figura 28: Gráfico tiempo de ejecución del algoritmo adaptativo.....	40
Figura 29: Gráfico uso de memoria algoritmo adaptativo	41
Figura 30: Gráfico uso de disco algoritmo adaptativo	42
Figura 31: Secuencia de imágenes Torres del Paine	42
Figura 32: Gráfico algoritmo adaptativo Dataset Torres del Paine.....	43
Figura 33: Localización similitudes entre imágenes	44
Figura 34: Panorámica Torres del Paine	44
Figura 35: Imagen no difuminada versus imagen difuminada.....	45
Figura 36: Imagen panorámica con una imagen difuminada	46
Figura 37: Imagen panorámica con tres imágenes difuminadas	46

RESUMEN

Este proyecto de título tiene como objetivo la evaluación y desarrollo de un algoritmo adaptativo para la creación de imágenes panorámicas con caso de uso el parque nacional Torres del Paine.

Se investigó el estado actual en que se encuentra el procesamiento de imágenes, tales como herramientas o tecnologías que apliquen técnicas de *stitching* e investigación de distintas técnicas de esta misma, para así encontrar diferentes casos de bordes que se puedan aplicar a nuestro caso de estudio.

También, se detallan las herramientas y programas ocupados para la creación de este algoritmo adaptativo como también los distintos tipos de Dataset que se usaron para las múltiples pruebas.

El algoritmo adaptativo se desarrolló a través del lenguaje de programación Python junto con la librería OpenCV, en el cual desde que recibe la secuencia de imágenes hasta la salida de la imagen panorámica el algoritmo va midiendo el tiempo de ejecución, uso de memoria y uso del disco, como también evaluando que técnica de stitching es mejor dependiendo el valor del ángulo entregado el cual podía ser SIFT u ORB la técnica aplicarse, y en caso de que la difuminación de la imagen se ocupaba cualquiera de ambas técnicas.

Se efectuaron pruebas con cuatro tipos de Dataset, tres de estos Dataset tomados a partir de un dron que realizó una captura de la misma imagen a la misma hora y el otro Dataset fotografiado por las cámaras del parque nacional Torres del Paine.

Al finalizar, se entregan los análisis de estas pruebas, conclusiones y futuras mejoras para el algoritmo adaptativo.

ABSTRACT

The objective of this degree project is the evaluation and development of an adaptive algorithm for the creation of panoramic images for the Torres del Paine National Park.

We investigated the current state of the art in image processing, such as tools or technologies that apply stitching techniques and research of different stitching techniques, in order to find different edge cases that can be applied to our case study.

Also, the tools and programs used for the creation of this adaptive algorithm are detailed as well as the different types of datasets that were used for the multiple tests.

The adaptive algorithm was developed through the Python programming language together with the OpenCV library, in which from receiving the sequence of images until the output of the panoramic image the algorithm is measuring the execution time, memory usage and disk usage, as well as evaluating which stitching technique is better depending on the value of the delivered angle which could be SIFT or ORB the technique to be applied, and in case of blurring of the image any of both techniques was used.

Four types of Datasets were tested, three of these Datasets were taken from a drone that captured the same image at the same time and the other Dataset was photographed by the cameras of the Torres del Paine National Park.

At the end, the analysis of these tests, conclusions and future improvements for the adaptive algorithm are presented

1. INTRODUCCIÓN

Chile es uno de los países que cuenta con una gran exportación de celulosa a nivel internacional, ya que existen a lo largo del país grandes plantaciones privadas de bosques, como también parques nacionales controlados por CONAF (Corporación Nacional Forestal). Todas ellas cuentan con torres de vigilancia en distintas zonas del parque que permiten monitorear sucesos fuera de los protocolos cotidianos.

En el año 2017, entre las regiones de Coquimbo y Los Lagos hubieron grandes cantidades de incendios forestales; donde fueron afectadas más de 587.000 hectáreas (Chile: Incendios forestales – enero 2017 - Reporte de Situación No. 02 (al 07 de febrero de 2017), 2020). También, entre los años 2011-2012 ocurrió un incendio forestal en uno de los parques nacionales llamado Torres del Paine, que se destaca por su gran cantidad de visitantes/turistas que llegan en distintas fechas del año. Este parque sufrió una pérdida de 17 mil hectáreas (Rivera Venegas, 2016), debido a esto y para evitar la carga severa tanto física como psicológica en el personal de trabajo producto a su constante vigía, se implementó un sistema de monitoreo en este parque, el cual reemplazaría a las personas ubicadas en las torres de vigilancia por cámaras online y que cuentan con energía con paneles solares.

Se colocaron varias cámaras con protocolos ONVIF (Open Network Video Interface Forum), que realizan capturas de imágenes a zonas específicas. Luego, un algoritmo buscaría similitudes entre estas para lograr “unirlas”, generando una vista panorámica de 360° (dos de 180°), esta técnica se le conoce como “*stitching*”¹. Tras la unificación de las imágenes, esta es enviada a una sala de monitoreo del mismo parque, logrando así, mantener informado ante cualquier suceso fuera de lo cotidiano y no forzar al trabajador a pasar una gran cantidad de horas del día aislado en una torre. Este proceso se va repitiendo cada tres minutos.

Debido a las condiciones climáticas o zona horaria existen problemas al momento de capturar las imágenes, ya que tienden a tener distintos bordes por tema de luminosidad, penumbra, medios ambientales, diferencias de ángulos producto del viento, entre otras, ocasionando un conflicto en la lógica del algoritmo y dificultando la creación de estas panorámicas, por lo que se propone buscar o generar algoritmos adaptativos para diferentes escenarios con el tratado de imágenes para cámaras con protocolos ONVIF.

1.1. Motivación

La motivación para realizar este proyecto nace a mediados del año 2019, donde existió una posibilidad de práctica profesional en un área con tratado de imágenes y cámaras de seguridad, por lo que había una atracción por el tema. También, existió interés de realizarlo con énfasis el área forestal, ya que cada verano es más común los incendios

¹ Stitching: costura/unión

forestales, daño a las áreas verdes o cualquier suceso fuera de lo cotidiano, por lo que es considerado como un problema actualmente vigente.

1.2. Impacto

El resultado esperado en esta investigación es lograr un uso correcto en el algoritmo adaptativo, para evitar gastos millonarios en restauración vegetal debido a un incendio forestal, como por ejemplo el incendio del año 2011-2012 mencionado anteriormente.

También, lograr su uso en áreas forestales u otras áreas que necesitasen de tratado de imágenes con filtros especiales.

1.3. Objetivos

1.3.1. Objetivos generales

Lograr una mejora significativa en las condiciones de borde aún no resueltas de uno de los algoritmos estudiados de creación de imágenes panorámicas con técnicas de stitching, con Dataset de imágenes de las cámaras de las Torres del Paine.

1.3.2. Objetivos específicos

1. Evaluar el estado actual en el que se encuentra el procesamiento de imágenes a través de una investigación y revisión sistemática.
2. Estudiar distintas técnicas de algoritmos de stitching.
3. Encontrar condiciones de borde o de mal funcionamiento de los algoritmos estudiados.
4. Evaluar y desarrollar un algoritmo adaptativo que permita generar las imágenes, utilizando los algoritmos evaluados.

2. MARCO TEÓRICO

En una época donde la fotografía no era más que un reciente invento (1826), se logró, unos años más tarde, la creación de fotografías panorámicas (1839), la cual, en esa época, consistía en ocupar varios daguerrotipos (ver Figura 1) (ya que entregaban fotografías en placas) que al unirlas se podía lograr una fotografía panorámica de excelente calidad.



Figura 1: Panorámica de Cincinnati en daguerrotipo²

Este método era muy costoso, por lo que se buscaron diferentes formas para realizar las imágenes panorámicas.

Años después, se optó por ocupar placas húmedas de vidrio (1860) (ver Figura 2), ya que eran menos costosas que los daguerrotipos. Las placas húmedas se unían de forma manual y debían estar muy limpias para lograr una buena imagen panorámica. Luego, estas placas, fueron reemplazadas por varios años, y en la actualidad, por la fotografía instantánea y la digital.

² Fuente:<http://www.fotografia360.org/panoramica-de-cincinnati-1848-en-daguerrotipo/>

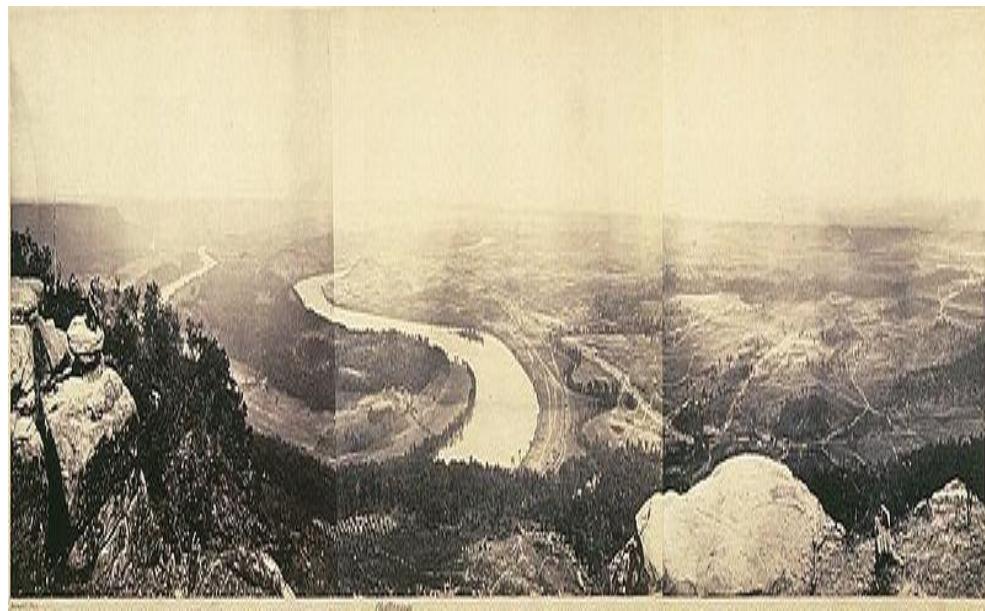


Figura 2: Panorámica con placas húmedas³

Actualmente, los celulares ocupan sistemas de fotografías panorámicas con técnicas de *stitching*, generando imágenes de manera automática en ángulos de 90° hasta 360°, varios han sido los proyectos o investigaciones que tratan de ir mejorando estas técnicas ocupando filtros, mejorando el reconocimiento de píxeles (px) o incluso integrando estas técnicas en tiempo real a objetos pilotados por humanos de forma remota. Algunos de estos vendrían siendo, por ejemplo, la investigación por Araúz Pisón“ (Araúz Pisón, 2016), donde se realiza una imagen panorámica “fusionando” múltiples capturas tomadas por agentes autónomos equipados con una cámara.

Otra investigación relacionada con las imágenes panorámicas fue el de Ruiz Ordinola J, donde se realizan análisis de técnicas para el proceso de *stitching* en imágenes digitales, con la finalidad de obtener una imagen panorámica con menor distorsión (Ruiz Ordinola, 2018).

Otro proyecto, por ejemplo, fue el de Gálvez Ortiz, S, el cual crea un algoritmo para unir secuencias de imágenes, ocupando un rango de 4 píxeles extras para evitar el movimiento brusco de las cámaras (ver Figura 3) (Gálvez Ortiz, 2017).

³ Fuente: http://fido.palermo.edu/servicios_dyc/proyectograduacion/archivos/109.pdf



Figura 3: Panorámica generada evitando movimientos bruscos

Otro proyecto sería realizado por SGO que a través de un dispositivo de realidad virtual que recibe capturas de imágenes permite realizar en tiempo real y de manera rápida un stitching de alta velocidad, generando una panorámica de 360° (SGO, 2020).

También otro en la cual se trata de mejorar filtros de imágenes ya existentes para imágenes con un fondo distinto, buscando la correlación de ciertos objetos (González, Álvarez, Martínez, & Ascencio, 2009).

Fue revelada hace poco una imagen panorámica realizada con técnicas de *stitching* que causó bastante asombro en el mundo, esta imagen fue creada por el robot enviado a Marte llamado “Curiosity”. El robot Curiosity demoró seis horas y media, repartida en cuatro días, y para evitar problemas de luminosidad cada imagen fue tomada a la misma hora durante todo este proceso, logrando así, una imagen de 1.800 millones de píxeles, esta imagen se puede apreciar en la Figura 4 (Infobae, 2020).

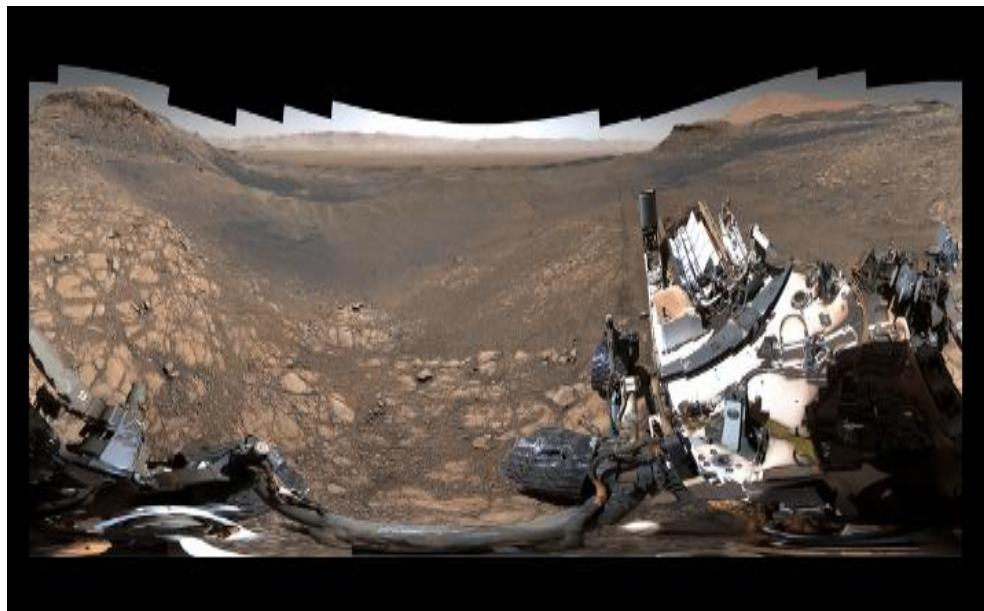


Figura 4: Panorámica en marte con 1800 millones de píxeles⁴

Ahora, se describen a grandes rasgos, los principales conceptos y etapas asociadas al desarrollo de la solución propuesta para la problemática descrita, también, se puede observar estas etapas de manera visual (ver Figura 5).

Entre estas se encuentran:

1. Procesamiento de imagen.
2. Detección de puntos clave.
3. Descriptores de invariantes locales (SIFT, SURF, ORB, entre los más comunes).
4. Coincidencia de las características.
5. Estimación de homografía usando RANSAC.
6. Deformación de la perspectiva.
7. Mezcla de imágenes.

⁴ Fuente: <https://www.infobae.com/america/ciencia-america/2020/03/05/la-nasa-difundio-una-imagen-de-marte-con-1800-millones-de-pixeles>

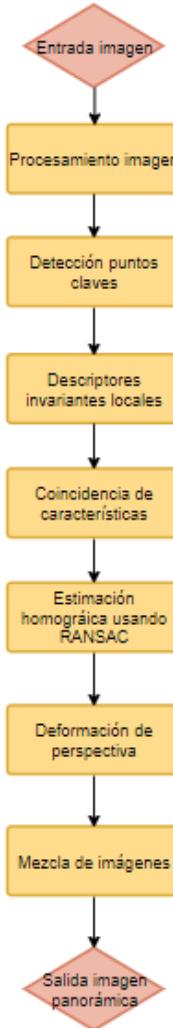


Figura 5: Diagrama de etapas de creación de imagen panorámica

2.1 Procesamiento de imagen

En el procesamiento de imagen se recibe como entrada una imagen digital, la cual es sometida a operaciones que se realizan antes de la extracción, segmentación y coincidencia de las características. Este proceso puede incluir la eliminación de ruido, extracción de bordes, el procesamiento de histogramas, realización de transformaciones de la imagen, filtros para ajuste de brillo, etc.

2.2 Detección de puntos clave

La detección de puntos claves es vital para encontrar de manera automática combinaciones de puntos correspondientes de imágenes adyacentes, calcular la posición inicial relativa

de la cámara, obtener la rotación y translación de la cámara en base a los pares de puntos y lograr un alineamiento entre las imágenes que se están componiendo (ver Figura 6).

Estos puntos de interés o puntos clave, se obtienen a partir de algoritmos de extracción de características, entre los más comunes y con mayor uso se encuentran: SIFT, SURF y ORB, y luego los de menor uso: KLT, PCA-SIFT y HARRIS.



Figura 6: Detección de puntos clave⁵

2.3. Descriptores de invariantes locales (SIFT, SURF, ORB.)

2.3.1 SIFT (Scale Invariant Feature Transform)

El enfoque llamado Transformación de Características de Invariantes de Escala (SIFT) (OpenCV, 2020) fue creado en el año 2004 por D. Lowe, el cual transforma los datos de imágenes de entrada en coordenadas invariantes de escala en relación con las características locales.

El mismo año se llevó a cabo una modificación en el algoritmo SIFT, donde Y. Ke y R. Sukthankar en vez de usar los histogramas suavizados de SIFT, aplica análisis de componentes principales (PCA-SIFT) a los normalizados parches de gradiente, logrando resultados con mayor precisión y una coincidencia más rápida, a partir de una aplicación de recuperación de imágenes (Ke & Sukthankar, 2004).

En 2019, G. Tang, Z. Liu1 y J. Xiong mejoraron el método SIFT combinándolo con el Patrón Binario Local (LBP) debido a su cambio de iluminación, iSIFT mejora la precisión

⁵ Fuente: https://docs.opencv.org/master/da/df5/tutorial_py_sift_intro.html

de la coincidencia de las características de la imagen y la robustez bajo el cambio de condiciones de iluminación comparadas con las del algoritmo SIFT, también, el descriptor iSIFT puede extraer más puntos característicos de la imagen borrosa y la imagen con bordes suaves, así como tener una mayor robustez para la iluminación, la rotación y el escalado (Tang, Liu, & Xiong, 2019).

El algoritmo SIFT tradicional cuenta con las siguientes etapas:

2.3.1.1 Detección de espacio extremo a escala

“La detección de espacio extremo a escala es la primera etapa y consiste en localizar las ubicaciones y escalas que se asignan reiteradamente al mismo objeto desde diferentes vistas. Se buscan características estables a través de la mayor cantidad de posibles escalas de la imagen (ver Figura 7).

El espacio de escala de una imagen se define como una función, $L(x, y, \sigma)$, que se produce a partir de la convolución de un Gaussiano de escala variable, $G(x, y, \sigma)$, con una imagen de entrada, $I(x, y)$:

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \quad (1)$$

donde $*$ es la operación de convolución en x e y . Para detectar eficientemente las ubicaciones de puntos clave estables en el espacio de escala, hemos propuesto (Lowe, 1999) utilizar el espacio de escala extrema en la diferencia de la función gaussiana convivida con la imagen, $D(x, y, \sigma)$, que se puede calcular a partir de la diferencia de dos escalas cercanas separadas por un factor multiplicador constante k “ (Lowe, 2004).

$$D(x, y, \sigma) = (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y) \quad (2)$$

$$D(x, y, \sigma) = L(x, y, k\sigma) - L(x, y, \sigma) \quad (3)$$

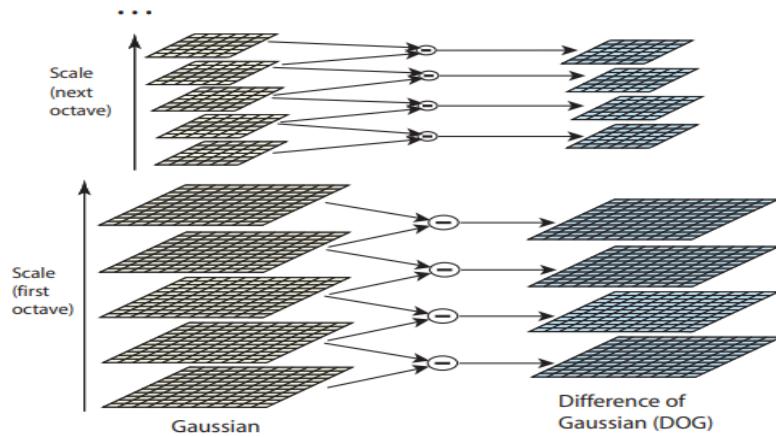


Figura 7: Diferencia de Gaussiana

“Una vez que se encuentra esta diferencia de gaussiana, se buscan imágenes para el extremo local sobre la escala y el espacio. Por ejemplo, un píxel de una imagen se compara con sus 8 vecinos, así como 9 píxeles en la siguiente escala y 9 píxeles en escalas anteriores. Si es un extremo local, es un punto clave potencial. Básicamente significa que el punto clave está mejor representado en esa escala” (OpenCV, 2020).

Se muestra en la siguiente figura (ver Figura 8).

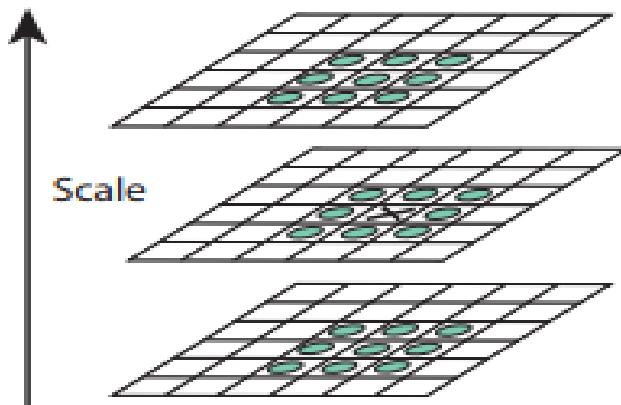


Figura 8: Máximos y mínimos de las imágenes de diferencia de Gaussiana

2.3.1.2 Localización de puntos claves

Se compara un píxel con sus adyacentes para poder tener un candidato a punto clave, al localizarlo se realiza un detallado ajuste a los datos cercanos para la escala, ubicación y proporción de la curvatura, así se logran rechazar otros puntos que no son relevantes, como por ejemplo puntos mal localizados (ver Figura 9).

Luego, se utiliza una matriz Hessiana 2x2 para calcular la curvatura principal y para la eliminación de ciertos puntos en los bordes.

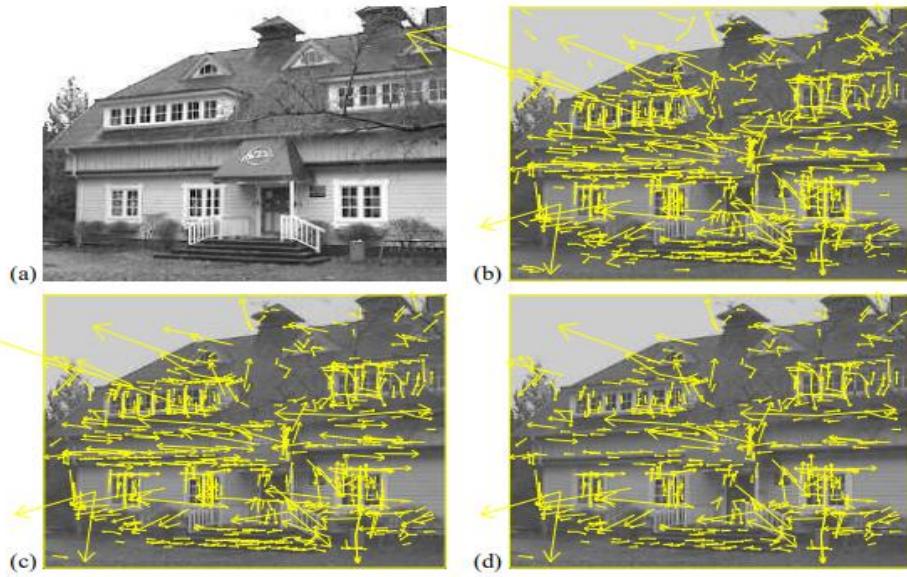


Figura 9: Etapas de selección de puntos claves

En la Figura 9: Etapas de selección de puntos claves, muestra las etapas de la selección de los puntos clave. En la posición superior izquierda (a) se puede observar la imagen original, la cual tiene 233x189 píxeles. En la posición superior derecha (b) se puede ver las 832 posiciones iniciales de los puntos claves, estos corresponden a los máximos y mínimos de la diferencia de la función Gaussiana. Estos puntos se muestran como vectores que indican: escala, orientación y ubicación. En la posición inferior izquierda (c) estos puntos claves se reducen de 832 a 729 tras aplicar un umbral en el contraste mínimo. Por último, en el extremo inferior derecho (d) se pueden observar 536 puntos claves que quedan después de aplicar un umbral adicional sobre la relación de curvaturas principales.

2.3.1.3 Asignación de orientación

“Según las estadísticas de las orientaciones de gradiente que se calculan dentro de la región local centrada en el punto clave, se asignan una o más orientaciones dominantes a cada ubicación del punto clave. Para las ubicaciones con múltiples orientaciones dominantes, habrá múltiples puntos clave construidos en la misma ubicación y escala, pero diferentes orientaciones. Para lograr la invariante de rotación propiedad, la región local centrada en el punto clave se girará en relación con el punto clave orientación dominante antes de que se construya el descriptor local” (D., H., & C., 2011).

2.3.1.4 Descriptor de puntos clave

Este descriptor de puntos clave se crea a partir de un vector. Se selecciona una zona de 16x16 vecina al punto clave, este se divide en 16 matrices de 4x4 con 8 entradas de orientación cada uno, y para cada uno se le crea un histograma de orientación en donde

los datos de entradas de este son almacenados en el vector, dando un total de 128 valores disponibles para cada punto clave. Además, se toman medidas para lograr la robustez contra cambios de rotación, cambios de brillo producto a la saturación de la imagen, etc.

2.3.1.5 Coincidencia de puntos clave

La coincidencia de puntos claves entre ambas imágenes se realiza localizando al vecino más cercano a través de la distancia euclíadiana mínima para el vector descriptor. Aunque a veces puede ocurrir que se encuentren muy cercanas las distancias entre la primera coincidencia con la siguiente provocando errores. Para estos casos, se toman ambas distancias cercanas y si es superior al 0.8 se rechaza, logrando así eliminar contorno del 90% de falsas coincidencias (Huang, Wang, & Li, The Research of Image Mosaic Techniques Based on Optimized SIFT Algorithm, 2019).

2.3.2 SURF (Speeded Up Robust Features)

“El descriptor SURF propuesto se basa en propiedades similares a la del SIFT, con una complejidad aún más reducida. El primer paso consiste en fijar una orientación basada en la información de una región circular alrededor del punto de interés. Luego, construimos una región cuadrada alineada con la orientación seleccionada, y extraer el descriptor de SURF de él. Además, también se propone una versión vertical de nuestro descriptor (USURF) que no es invariable a la rotación de la imagen y por lo tanto más rápido para calcular y más adecuado para las aplicaciones en las que la cámara permanece más o menos horizontal” (Bay, Tuytelaars, & Van Gool, 2006).

“SURF usa respuestas de Wavelet en dirección horizontal y vertical (nuevamente, el uso de imágenes integrales facilita las cosas). Se toma una vecindad de tamaño $20s \times 20s$ alrededor del punto clave donde s es el tamaño. Está dividido en subregiones 4×4 . Para cada subregión, se toman las respuestas de ondículas horizontales y verticales y se forma un vector” (ver Figura 10) (OpenCV, 2021).

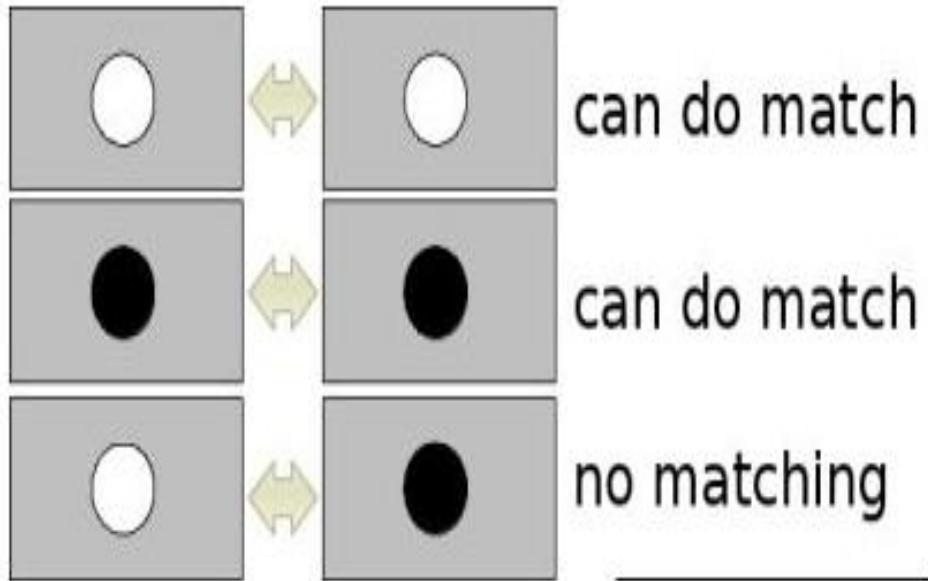


Figura 10: Emparejamiento de manchas SURF⁶

También, una mejora que se menciona y tiene relevancia es el uso del signo Laplaciano (a partir de matriz Hessiana) como punto de interés subyacente. Este signo diferencia en la etapa de emparejamiento manchas que tengan el mismo tipo de contraste, no obstruyendo el rendimiento del cómputo, ya que este signo es calculado durante la detección.

En 2010, se propuso una mejora al algoritmo SURF, la cual consistía en la optimización de las Características Robustas Aceleradas (SURF) y el desarrollo de una nueva función de medición de la similitud basada en las trayectorias generadas a partir de las figuras de Lissajous. “En comparación con el SURF, que tiene una baja tasa de coincidencia de características en algunos casos complejos, el algoritmo SURF actualizado es más robusto y preciso. El algoritmo mejora enormemente la tasa de coincidencia correcta a más del 80%. Además, la capacidad de reconocimiento de la medida de similitud se mejora utilizando una perturbación de la trayectoria estratégica, que es un desplazamiento significativo en la trayectoria inducido por un pequeño error de los parámetros de transformación” (Song & Zhang, 2010).

“SURF ha demostrado ser uno de los detectores y descriptores de características más avanzados, y trata principalmente las imágenes de color como imágenes grises. Sin embargo, el color proporciona información valiosa en las tareas de descripción y reconocimiento de objetos. Es donde en el año 2012 abordaron este problema y añadieron la información sobre el color en el detector y descriptor de puntos de interés de escala y rotación invariable, nombrado CSURF (Colored Speeded Up Robust Features). El CSURF construido es más robusto que el SURF convencional con respecto a las variaciones

⁶Fuente: https://docs.opencv.org/master/d9/d22/tutorial_py_surf_intro.html

de rotación. Además, usando 112 dimensiones para describir no sólo la distribución de las respuestas de las ondículas de Harr sino también la información de color dentro del vecindario del punto de interés. Los resultados de la evaluación demostraron aumentar el promedio de la tasa de coincidencia correcta y robusta contra la rotación” (Fu, Jing, Sun, Lu, & Wang, 2012).

2.3.3 ORB (Oriented FAST and Rotated BRIEF)

Este algoritmo fue presentado por Ethan Rublee, Vincent Rabaud, Kurt Konolige y Gary R. Bradski en su artículo ORB: Una alternativa eficiente a SIFT o SURF en 2011.

ORB es básicamente una fusión del detector de puntos clave FAST y el descriptor BREVE con muchas modificaciones para mejorar el rendimiento (OpenCV, 2021).

En el 2015 se creó ORB-SLAM, “un sistema monocular de localización y cartografía simultáneas (SLAM) basado en características que funciona en tiempo real, en pequeños y grandes entornos interiores y exteriores. El sistema es robusto a un desorden de movimiento severo, permite el cierre y la relocalización de un amplio bucle de base, e incluye una inicialización totalmente automática. Basandonos en los excelentes algoritmos de los últimos años. La supervivencia de la estrategia más adecuada que selecciona los puntos y los fotogramas clave de la reconstrucción conduce a una excelente robustez y genera un mapa compacto y rastreable que sólo crece si el contenido de la escena cambia, permitiendo una operación de por vida. ORB-SLAM logra un rendimiento sin precedentes con respecto a otros enfoques SLAM monoculares de última generación” (Mur-Artal, Montiel, & Tardos, ORB-SLAM: a versatile and accurate monocular SLAM system, 2015).

Dos años más tarde se crea ORB-SLAM2 es “un sistema completo de localización y mapeo simultáneo (SLAM) para camaras monoculares, estéreo y RGB-D, incluyendo la reutilización de mapas, el cierre de bucles y las capacidades de relocalización. El sistema funciona en tiempo real en unidades centrales de procesamiento estándar en una amplia variedad de entornos, desde pequeñas secuencias de mano en interiores, hasta aviones no tripulados que vuelan en entornos industriales y coches que circulan por la ciudad. La evaluación de 29 secuencias públicas populares muestra que el método alcanza una precisión de última generación, estando en la mayoría de los casos la solución más precisa de SLAM” (Mur-Artal & Tardós, Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras, 2017).

“Primero usa FAST y sus variantes son el método de elección para encontrar puntos clave en sistemas en tiempo real que coinciden con las características visuales. Es eficiente y encuentra puntos clave de esquina razonables, aunque debe ser aumentado con esquemas piramidales para la escala, y en nuestro caso, un filtro de esquina Harris para rechazar los bordes y proporcionar una puntuación razonable” (Rublee, Rabaud, Konolige, & Bradski, 2011).

Este método describe las características de la imagen entrante en una cadena binaria, a diferencia de las que las describen como un vector (ver Figura 11).

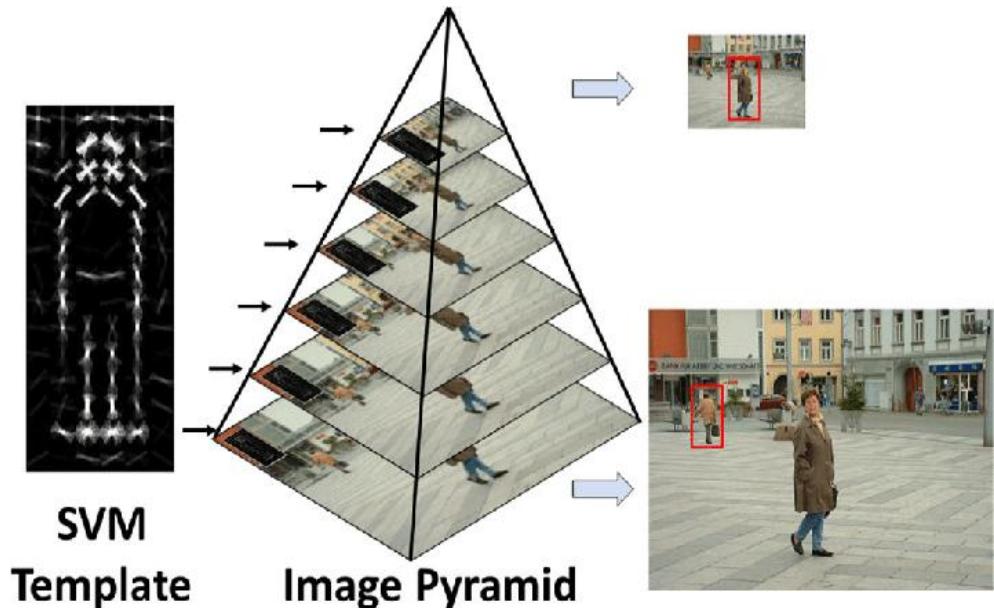


Figura 11: Pirámide de imagen con múltiples escalas

Pirámide de imagen con múltiples escalas. Procesando todas las escalas con la misma plantilla SVM se pueden detectar objetos de diferentes tamaños (Suleiman & Sze, 2016).

2.4. Coincidencia de las características

La coincidencia de las características viene de la mano con RANSAC, ya que hacer coincidir los rasgos de las imágenes es algo sencillo; sin embargo, es necesario los puntos clave y los vectores de características asociados de la primera y la segunda imagen por separado. Se hacen iteraciones sobre los descriptores de las dos imágenes de entrada que se comparan, se calculan las distancias euclidianas y se encuentran las distancias más pequeñas para cada par de descriptores.

2.5. Estimación de homografía usando RANSAC (Random Sample Consensus)

“Es un método iterativo para estimar los parámetros de un modelo matemático a partir de un conjunto de datos observados que contienen valores atípicos. Es un algoritmo no determinista en el sentido de que produce un resultado razonable sólo con una cierta probabilidad, aumentando esta probabilidad a medida que se permiten más iteraciones” (Mistry & Patel, 2016).

El algoritmo básico de RANSAC (Derpanis, 2004) se resume a continuación:

1. Seleccionar aleatoriamente el número mínimo de puntos necesarios para detectar los parámetros del modelo.
2. Resolver para el parámetro del modelo.
3. Determinar cuántos puntos del conjunto de todos los puntos encajan con una tolerancia predefinida.
4. Si la fracción del número de inliers sobre el número total de puntos del conjunto excede un umbral predefinido τ , estimar los parámetros del modelo usando todos los inliers identificados y terminar.
5. De lo contrario, repita los pasos 1 a 4.

Este proceso, se itera reiteradas veces dependiendo del número que se le dé, el modelo se rechaza si se clasifican muy pocos puntos en el conjunto de datos de valores atípicos, estos valores son conocidos como *inliers*. La idea, es verificar y comparar las probabilidades correctas de que los inliers se hayan generado a partir de una coincidencia de imagen correcta o de imagen falsa. Al utilizar RANSAC se obtiene una gran ventaja, la cual es tener un alto grado de precisión, por otro lado, se tiene una desventaja de que no hay límite superior en el tiempo de cálculo de los parámetros, generando que la solución no sea la óptima.

2.6. Deformación de la perspectiva

Existen varios problemas en el análisis de imágenes, sea para registrar una imagen con un mapa, alinear dos o más imágenes o eliminar distorsiones ópticas, es por esto que la deformación de la perspectiva realiza una transformación que mapea todas las posiciones en un plano de la imagen a posiciones en un segundo plano, ya que busca lograr una distorsión suave y que logre una buena coincidencia.

Una imagen puede ser transformada de varias maneras. La deformación pura significa que los puntos son mapeados a puntos sin cambiar los colores.

Hay dos métodos para la generación de una imagen para cualquier tipo de distorsión (Mistry & Patel, 2016) :

1. Mapeo de avance: Se aplica directamente un mapeo dado de las fuentes a las imágenes, en otras palabras, la imagen de origen se mapea sobre una superficie cilíndrica, pero puede dejar agujeros en la imagen de destino ya que algunos píxeles pueden no ser mapeados allí.
2. Mapeo inverso: Un mapeo dado de fuentes a imágenes, la fuente se encuentra a partir de la imagen, esta se utiliza generalmente cuando cada píxel de la imagen de destino se asigna a la imagen de origen.

2.7. Mezcla de imágenes

Tras tener mapeado los píxeles, se debe realizar la mezcla de estos para crear una imagen panorámica. Esta etapa consiste en modificar los niveles grises de la imagen cercana a un límite para tener una transición suave, eliminando la visualización de posibles costuras y teniendo un aspecto más “amigable” de la imagen panorámica.

Algunos métodos de fusión que se utilizan comúnmente (Li & Du, 2010):

1. El método spline multi-resolución es un método de fusión de color porque involucra los problemas estructurales de Costa y Laplace, lo que tiene las desventajas de la gran cantidad de cálculos y el tiempo que consume.
2. El método de promedio ponderado es relativamente simple, pero puede causar fácilmente costuras obvias.
3. La fusión de Poisson es simple y conveniente cuando se seleccionan regiones de fusión. La fusión utiliza el campo de gradiente en el bloque de la imagen de origen como guía, y la diferencia entre la imagen de origen y la imagen de destino en el límite de la fusión se difunde suavemente en el bloque de la imagen fusionada, haciendo el bloque de la imagen fusionada. Consigue un efecto sin problemas cuando se fusiona con la imagen objetivo, y su tono e iluminación son consistentes con el objetivo imagen.

2.8. Comparación de algoritmos en distintos escenarios

Tras conocer cómo funciona el procesamiento de imagen al crear una vista panorámica con técnicas de stitching, además, de cuáles son sus etapas y qué se diferencian cada una de estas, se puede pasar a la siguiente sección en la que analizaremos estas técnicas en distintos escenarios y se logrará ver qué técnica es mejor para que caso.

Usualmente, se realizan pruebas para imágenes capturadas en ambientes cerrados, abiertos, estáticos o realizando alteraciones en las imágenes como, por ejemplo: rotaciones, diferencia de escalas, darle una mayor o menor iluminación de lo normal a la imagen, aplicarle un filtro de difuminación o desenfoque a las imágenes. También, se realizan pruebas en términos de rendimiento computacional y algoritmo como, por ejemplo: uso de memoria, CPU y de disco al momento de ejecutar el código, tiempo de ejecución, números de puntos claves detectados, números de características emparejadas, números de inliers, números de outliers, porcentaje de coincidencia, error en pares encontrados y porcentaje de error en tasa de pares encontrados.

Gracias a la revisión sistemática realizada, se puede comprender que algoritmo funciona mejor o tiene mejor rendimiento dependiendo las condiciones.

Se puede apreciar en la Tabla 1 que los algoritmos más destacados para cada aspecto (escala, rotación, afín, iluminación y desenfoque) son SIFT y PCA – SIFT, también, se evalúa cómo dependiendo de la aplicación se utiliza el algoritmo típico de extracción de características.

Para lograr una técnica excelente de registro de imágenes, SIFT es un candidato idóneo. Como en el caso de la costura de imágenes en tiempo real, los dispositivos de uso como la técnica móvil SURF o FAST serán más adecuados, ya que su complejidad de cálculo es menor y es muy rápida de calcular (Vaidya & Gandhe, The study of preprocessing and postprocessing techniques of image stitching, 2018).

- Comparación distintas técnicas de extracción de características

Tabla 1: Comparación del algoritmo de extracción de características basado en parámetros cuantitativos

Algorithm	Performance Parameters Invariant to -				
	Scale	Rotation	Affine	Illumination	Blur
SIFT	Best	Best	Good	Better	Good
PCA - SIFT	Better	Better	Good	Better	Good
SURF	Normal	Normal	Poor	Normal	Normal
HOG	Normal	Best	Poor	Good	Poor
ORB	Normal	Normal	Poor	Normal	Normal
FAST	Poor	Poor	Poor	Normal	Poor

En la Tabla 2 se puede ver que el algoritmo Hessiano es mejor para la detección de características en imágenes con luz de día, siguiéndolo el método SIFT, aunque, es totalmente distinto en imágenes nocturnas donde, destaca el algoritmo de Esquinas de Harris por mayor cantidad de detección de características (Patil & Gohatre, 2017).

- Para escenarios donde las imágenes eran tomadas de día y de noche:

Tabla 2: Comparación de tres métodos de detectores de características utilizados en la unión de imágenes

Detector Type	Detected Features	Time (Sec)	Image Type
Harris Corner Detector	340	0.6	Daylight Image
Hessian Detector	566	0.45	
SIFT	455	1.3	
Harris Corner Detector	30	0.45	Night Image
Hessian Detector	23	0.38	
SIFT	12	1.34	

En la Tabla 3 se realiza una comparación entre un algoritmo creado en investigación versus el algoritmo SIFT, donde en escenarios tanto internos como externos tuvieron valores casi similares en la detección de características, esta brecha se agrandó en el porcentaje de error donde el algoritmo SIFT obtuvo un mayor valor de error (Huang, Wang, & Li, The Research of Image Mosaic Techniques Based on Optimized SIFT Algorithm, 2019), no obstante, en la Tabla 4 el algoritmo SIFT fue comparado con algoritmos tales como: KLT y HARRIS, donde SIFT alcanzó un valor de tiempo de cómputo muy superior a los otros dos y obtuvo mejores resultados (Vaidya, Gandhe, & Sonawane, Performance Analysis of KLT, Harris and SIFT Feature Detector for Image Stitching, 2016).

- Para escenarios de interior y exterior:

Tabla 3: Resultados estadísticos de la coincidencia de puntos de características

Image	Algorithm	Number of feature points	Point amount matched	Error point pairs	Error Rate (%)
Outdoor scene	SIFT	567	162	38	23.46
	This paper algorithm	594	177	11	6.21
Indoor scene	SIFT	137	52	15	28.85
	This paper algorithm	154	65	6	9.23

- Rendimiento distintos detectores

Tabla 4: Comparación del análisis de rendimiento de los detectores klt, harris y sift para imágenes de interior

Parameters	Number of match features	Number of Inliers	Number of Outliers	Computation Time	Matching Rate (%)
KLT	129	48	80	41.22	37.20
HARRIS	261	171	90	24.08	65.51
SIFT	568	395	173	286.62	69.54

Para la Tabla 5 la cual consistía en objetos estáticos, se puede visualizar que el algoritmo SIFT fue bastante superior versus los algoritmos ORB y SURF en tiempos de ejecución (Wang, Niu, & Yang, 2017). Mientras en la Tabla 6, SIFT obtiene un mayor número de emparejamientos en una cantidad baja de imágenes que sufren cambios de escala, pero no se vuelve viable en una cantidad más elevada de imágenes debido a su tiempo de cómputo, a diferencia de la Tabla 7, donde imágenes que sufren cambios de iluminación SIFT muestra que tiene un porcentaje promedio cercano a PCA-SIFT, pero SURF obtiene el mayor porcentaje promedio (Juan & Gwun, A comparison of sift, pca-sift and surf, 2009).

- Escenarios estáticos

Tabla 5: Comparación de costo en tiempo medido en milisegundos

Test subject	SIFT	SURF	ORB
Cup	2477.472	2344.54	610.526
Test Card	2161.201	2148.54	554.927
Text Stone	1929.79	1918.905	552.988

- Cambios de escala

Tabla 6: Imágenes que sufren modificaciones de escala e iluminación, tanto para escenarios internos como externos

Data	SIFT	PCA-SIFT	SURF
1-2	41	1	10
3-4	35	0	36
5-6	495	19	298
7-8	303	85	418

- Cambios de iluminación

Tabla 7: Comparación de cambios de iluminación, Data representa la repetibilidad y el promedio de repetibilidad

Data	SIFT	PCA-SIFT	SURF
1-2	43%	39%	70%
1-3	32%	34%	49%
1-4	18%	27%	25%
1-5	8%	18%	6%
1-6	2%	9%	5%
Average	21%	25%	31%

Aunque, se puede apreciar que SIFT tiene un tiempo de cómputo alto a diferencia de los otros métodos que se le compararon (ver Figura 12), SIFT muestra una tendencia, la cual es obtener una gran cantidad de detección de características y un alto porcentaje de emparejamiento por lo que se ve viable utilizarlo para el caso de estudio Torres del Paine, ya que hay un tiempo relativamente prudente que son tres minutos en donde se van enviado imágenes panorámicas al personal que monitorea la zona, favoreciendo a la creación de estas, e incluso SIFT tuvo un mejor rendimiento en rotación de imágenes (Juan & Gwun, A comparison of sift, pca-sift and surf, 2009), por lo que ayuda a las imágenes que sufren modificaciones por el movimiento de las cámaras producto de los fuertes vientos producidos en esa zona geográfica.

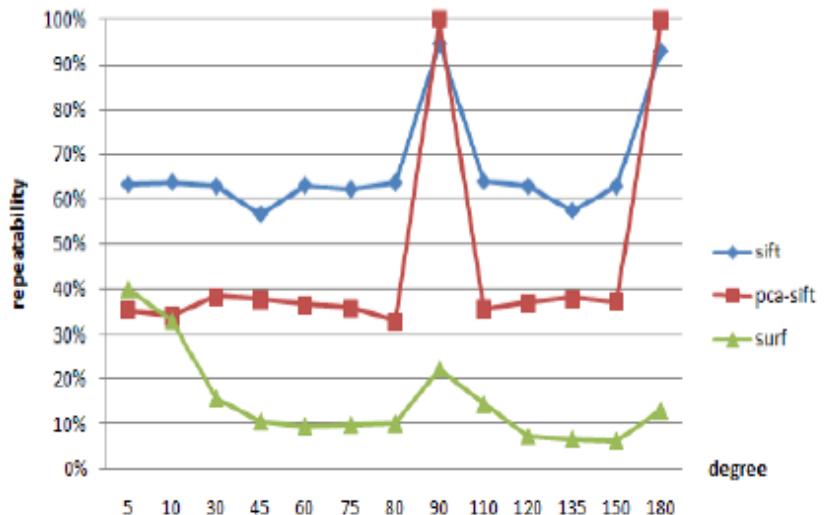


Figura 12: Comparación de rotación de imágenes

3. SOLUCIÓN PROPUESTA

En esta sección se mostrará más en profundidad todo lo referente a la problemática, el por qué se solucionará y el cómo se hará el desarrollo de la solución.

3.1 Descripción de la problemática

En la sección 1 se logró introducir al lector dándole una breve descripción de la problemática, obteniendo así, una idea general de esta.

Actualmente, en el Parque de las Torres del Paine, ubicado en el sur de Chile, siendo más específico en la región de Magallanes y la Antártica Chilena, se implementó un sistema de vigilancia a través de cámaras de vigilancia, la cual realiza múltiples capturas de imágenes y estas son enviadas a una central de monitoreo, que se encuentra en Puerto Natales, que está relativamente cerca del parque. Esta central de monitoreo es administrada por trabajadores del mismo parque y por cada cámara ubicada en el parque hay un computador. Esta recibe las imágenes continuamente y cada tres minutos genera dos imágenes panorámicas, ambas de 180 grados, logrando cubrir la zona en 360 grados.

Esto se realiza las 24 horas para evitar cualquier incidente que ocurra dentro del parque, tales como, por ejemplo: caída de árboles, extravíos de personas, focos tempranos de incendios, desprendimiento de tierras, entre otras.

El problema identificado está asociado directamente a las condiciones climáticas de esta zona, por ejemplo, a través de los datos históricos obtenidos el año 2019, la dirección meteorológica de Chile subdepartamento de climatología y meteorología aplicada entregó un reporte (Dirección Meteorológica de Chile - Servicios Climáticos, 2020) en el cual muestra las velocidades medias máximas de los vientos que eran de 21 nudos, equivalente a 39 kilómetros por hora, incluso una de las velocidades máximas registradas a la fecha 2020 superan los 60 kilómetros por hora (ver Figura 13). Esto provoca dificultad en la captura de imágenes, ya que la estructura donde se encuentran las cámaras posicionadas se ven agitadas por estas fuertes ráfagas de vientos, ocasionando diferencias significativas en los ángulos de capturas de imágenes que entorpecen al algoritmo de stitching para la creación de imágenes panorámicas.

MES	PRESIÓN MEDIA(hpa) AL NIVEL DE		VIENTO DIRECCIÓN DOMINANTE Y VELOCIDAD MEDIA						NUBOSIDAD MEDIA (octavos)			PRECIPITACIÓN (mm)		
	Estación	Mar	12 DD	hrs VM	18 DD	hrs VM	00 DD	hrs VM	12 hrs	18 hrs	00 hrs	TOTAL 24 hrs	MÁX	FECHA
ENERO	993.7	1,001.6	W	18	W	21	W	14	6	6	.	17.9	6.5	29
FEBRERO	999.3	1,007.2	W	19	SW	14	SW	8	6	6	.	52.1	19.6	17
MARZO	1,000.4	1,008.4	NW	11	W	18	W	11	6	6	.	34.7	10.0	16
ABRIL	996.0	1,004.0	W	7	W	15	W	11	.	6
MAYO	996.3	1,004.4	NW	8	W	16	N	5
JUNIO	994.3	1,002.5	NW	10	NW	15	W	10
JULIO	996.0	1,004.2	N	5	NW	15	N	5
AGOSTO	1,001.7	1,010.0	N	9	W	14	NW	11
SEPTIEMBRE	1,001.5	1,009.7	N	5	SW	13	W	15
OCTUBRE	998.8	1,006.8	W	16	W	21	W	11	.	.	.	15.9	4.7	17
NOVIEMBRE	992.6	1,000.6	NW	14	W	19	NW	10	6	6	.	143.7	33.7	18
DICIEMBRE	994.5	1,002.3	W	17	W	17	W	13	6	6	.	53.8	25.9	17
ANUAL	997.1	1,005.1	W	14	W	17	W	13

Figura 13: Velocidades vientos año 2019 Puerto Natales⁷

Otro de los problemas identificados también está asociado a las condiciones climáticas, pero no específicamente de esa zona, si no con los cambios bruscos de luminosidad producto de los rayos de origen solar, dado que en la secuencia de imágenes capturadas hay lapsos en esta donde el haz de luz solar da directamente al lente, ocasionando alteración en la imagen, provocando que se dificulte la detección en puntos clave entre imágenes para realizar un correcto emparejamiento de estas.

Es por esto que se busca solucionar estas problemáticas a partir de varios pasos, los cuales son una investigación para evaluar el estado actual en la que se encuentra el procesamiento de imágenes, ya que esto nos dará la base para saber qué tecnologías, herramientas y distintas técnicas que nos pueden ayudar para lograr un mejor entendimiento del problema. Luego, se estudiarán las distintas técnicas de algoritmos de stitching, ya que como se mencionó en la sección 2, existen distintas formas de crear una imagen panorámica, debido a: orígenes de las imágenes, métodos de cálculo y detección de puntos clave de estas, entre otras mencionadas en la misma sección. Luego, se destinará a buscar y encontrar ciertas condiciones de borde o de mal funcionamiento en los algoritmos evaluados, con sus Dataset originales y de prueba, estos últimos de origen propio. Y para finalizar, se llevará a cabo un algoritmo adaptativo que permita generar las imágenes utilizando los algoritmos evaluados y posteriormente ser evaluado.

⁷ Fuente: <https://climatologia.meteochile.gob.cl/application/publicaciones/anuario/2019>

4. TECNOLOGÍAS Y PRUEBAS TEMPRANAS

En este capítulo se describen las distintas tecnologías que se ocuparon y las pruebas tempranas obtenidas, ambas darán hincapié para la correcta construcción del algoritmo adaptativo de creación de imágenes panorámicas con técnicas de stitching, caso de uso Torres del Paine.

4.1 Tecnologías

1. **Anaconda Navigator:** “Anaconda Navigator es una GUI de escritorio que viene con Anaconda Individual Edition, perteneciente al ecosistema de código abierto, la cual facilita el lanzamiento de aplicaciones y el manejo de paquetes y facilita la gestión de múltiples entornos de datos que pueden ser mantenidos y ejecutados por separado sin interferencias entre ellos, incluso se pueden crear sin usar líneas de comandos” (Anaconda individual edition, 2020).
2. **Python:** “Python es un lenguaje de programación de alto nivel, interpretado y orientado a objetos, con semántica dinámica. La sintaxis simple de Python reduce el costo de mantenimiento del programa, también, soporta módulos y paquetes, lo que fomenta la modularidad del programa y la reutilización del código. El intérprete de Python y la extensa biblioteca estándar están disponibles en forma de código fuente o binaria sin cargo para todas las plataformas principales, y pueden ser distribuidos libremente” (What is Python? Executive Summary, 2020).
3. **OpenCV:** “OpenCV (Open Source Computer Vision Library) es una biblioteca de software de código abierto de visión por ordenador y aprendizaje automático. La biblioteca cuenta con más de 2500 algoritmos optimizados. Estos algoritmos pueden utilizarse para detectar y reconocer rostros, identificar objetos, clasificar acciones humanas en vídeos, unir imágenes para producir una imagen de alta resolución de toda una escena, encontrar imágenes similares en una base de datos de imágenes, seguir los movimientos de los ojos, etc.” (Opencv, 2020).
4. **Visual Studio Code :** “Visual Studio Code es un editor de código libre que facilita la codificación rápida, ya que incluye soporte para depuración, control integrado de Git, resultado de sintaxis, y más. Se usa para codificar cualquier lenguaje de programación, sin cambiar de editor. Es compatible con muchos lenguajes de programación, incluyendo Python, Java, PHP, C++, JavaScript entre los más conocidos” (Learn to code with Visual Studio Code, 2020).
5. **Visual Paradigm :** Visual Paradigm es una poderosa herramienta de diseño y gestión de sistemas informáticos, multiplataforma y fácil de usar. Proporciona a los desarrolladores de software una plataforma de desarrollo de vanguardia para construir aplicaciones de calidad, más rápido, mejor y más barato. Facilita una excelente interoperabilidad con otras herramientas CASE y con la mayoría de los IDEs líderes.

6. **Pix4dMapper** : Pix4Dmapper es un software de procesamiento de imágenes, la cual entrega un mapeo profesional para imágenes tomadas por drones, logrando así optimizar el vuelo y transferir datos de imágenes.
7. **Dron DJI FC 2103**: Un dron es un vehículo no tripulado, el cual puede mantener un vuelo controlado. Se usa usualmente para capturas de imágenes o videos (ver Figura 14).



Figura 14: Dron DJI FC 2103

8. **Cámaras de Torres del Paine**: Cámaras para vigilancia ubicadas en sectores específicos del Parque Nacional Torres del Paine. Estas envían capturas de zonas del parque a la sala de monitoreo.

4.2 Pruebas tempranas

Nuestro Dataset ocupado se compone de imágenes para pruebas tempranas e imágenes para pruebas finales.

En las pruebas tempranas, se tomaron 95 imágenes en la salida Norte Cabo Blanco de Valdivia, Fundo Santa Rosa (ver Figura 15), la cual es propiedad de la Universidad Austral de Chile, estas imágenes se tomaron a una altura de 50 m y fueron tomadas por el dron siguiendo una ruta pre establecida, estas imágenes tienen un formato de 4056x3040 píxeles, en la cual se eligieron 15 imágenes de estas 95.



Figura 15: Dataset de pruebas capturadas por el dron DJI

Se aprovechó el uso de la tecnología pix4dmapper, la cual tenía la ruta que realizó el dron por el fondo para realizar las distintas capturas (ver Figura 16), cada imagen tiene como metadato su latitud, longitud y altura, por lo que era más fácil trazar esta ruta y ver cómo se fueron tomando las imágenes de manera secuencial, esto nos ayudará más adelante a seleccionar mejor las imágenes para realizar un correcto stitching y evitar algún inconveniente.



Figura 16: Zona de captura de imágenes del dron DJI

En pruebas finales, se ocuparon imágenes capturadas desde las Torres del Paine, estas ya son 10 imágenes tomadas por cámaras con paneles solares y una zona al interior del parque y tienen un formato de 480x848 píxeles (ver Figura 17).

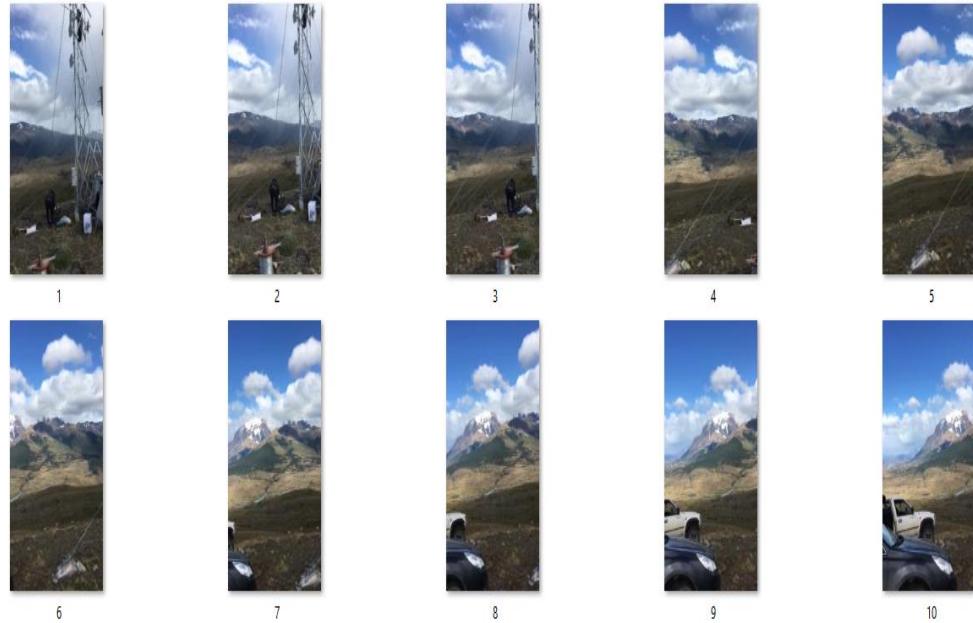


Figura 17: Dataset Torres del Paine

Para las pruebas se ocuparon distintos algoritmos los cuales fueron testeados con las imágenes de prueba, también se obtuvo el valor de distintas métricas aplicadas en el momento de su ejecución, tales como tiempo, uso de disco y memoria.

Las primeras pruebas se realizaron con el primer Dataset proporcionado por el Dron DJI FC 2103 que se mencionó anteriormente, sin embargo, al ser una cantidad de imágenes bastante grande, con mucha cantidad de píxeles y desde distintos puntos referenciales se optó por reducir está a la selección a cinco imágenes para cada testeо de los distintos algoritmos proporcionados.

Como se mencionó en la sección 2.8 se dio prioridad al testeо de algoritmos de stitching basado en SIFT y en ORB, ya que el método SURF se encuentra patentado, dificultando su utilización.

Se realizaron distintas pruebas con tamaños del mismo Dataset, en su totalidad de píxeles, reducido al 25% de su tamaño original y al 6,25%. Se realizaron también mediciones de uso de memoria, disco y tiempo de ejecución.

Los resultados de estas pruebas y las diferencias entre SIFT versión 1, versión y versión 3 y ORB se podrán ver y leer con mejor claridad en los Anexo A: Tabla de métricas e

imágenes panorámicas generadas con algoritmo SIFT versión 1, Anexo B: Tabla de métricas e imágenes panorámicas generadas con algoritmo SIFT versión 2, Anexo C: Tabla de métricas e imágenes panorámicas generadas con algoritmo con técnica SIFT versión 3 y Anexo D: Tabla de métricas e imágenes panorámicas generadas con algoritmo ORB, la cual corresponde a las tablas de cada prueba para los distintos tamaños de los Dataset, sus valores obtenidos y sus imágenes panorámicas resultantes.

A partir de las Figura 18, Figura 19 y Figura 20 se puede apreciar que el algoritmo con técnica SIFT versión 1 (barras naranjas) fue el que más demoró de las tres versiones de SIFT, aunque la brecha que los separa es mínima, al igual que el porcentaje del uso del disco y uso de memoria. Incluso se puede observar en la Tabla 8 que este algoritmo alcanzó valores de 29,73 segundos.

El algoritmo con técnica SIFT versión 2 (barras azules) tuvo valores muy parecidos a su versión 1, obteniendo un valor menor en tiempo de ejecución que la versión 1 pero valores un poco elevados en uso de disco y memoria (0,10% y 0,20% de diferencia).

El algoritmo con técnica SIFT versión 3 (barras verdes) obtuvo valores muy positivos en tiempo de ejecución del algoritmo, este tiempo fue el mejor dentro de los 4 algoritmos (2,991 segundos para el primer caso y 0,45 segundos para el segundo). La diferencia es que obtuvo valores más elevados en uso de memoria y porcentajes idénticos en uso de disco con sus otras versiones de SIFT.

En cambio, se puede observar que la técnica de stitching con ORB (barras grises) es una de las que más tiempo de ejecución tiene a diferencia de las tres versiones distintas de algoritmos de SIFT, lo recompensa con un porcentaje bajo en uso de memoria, pero tiene un alto porcentaje en uso de disco.

Gráfico Tiempo por Algoritmo

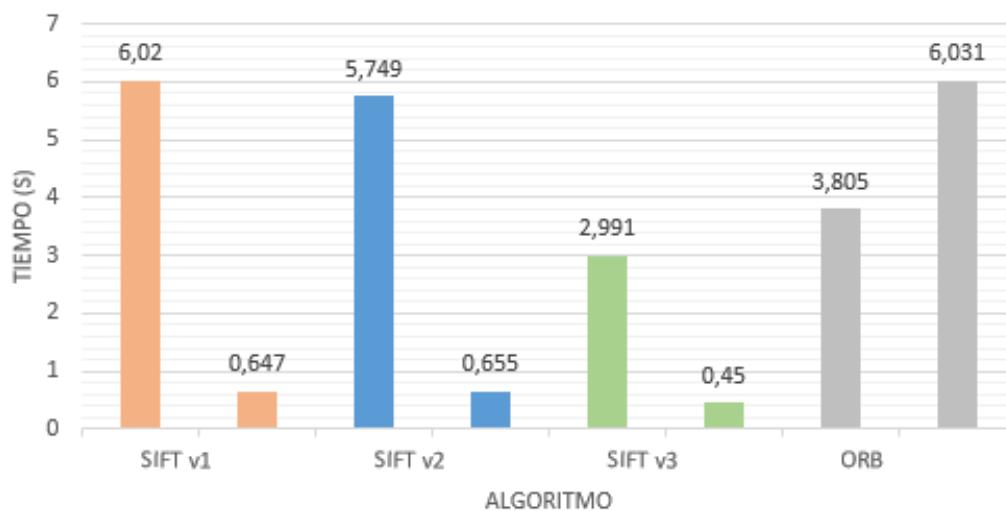


Figura 18: Gráfico de tiempo de ejecución por algoritmo testeado

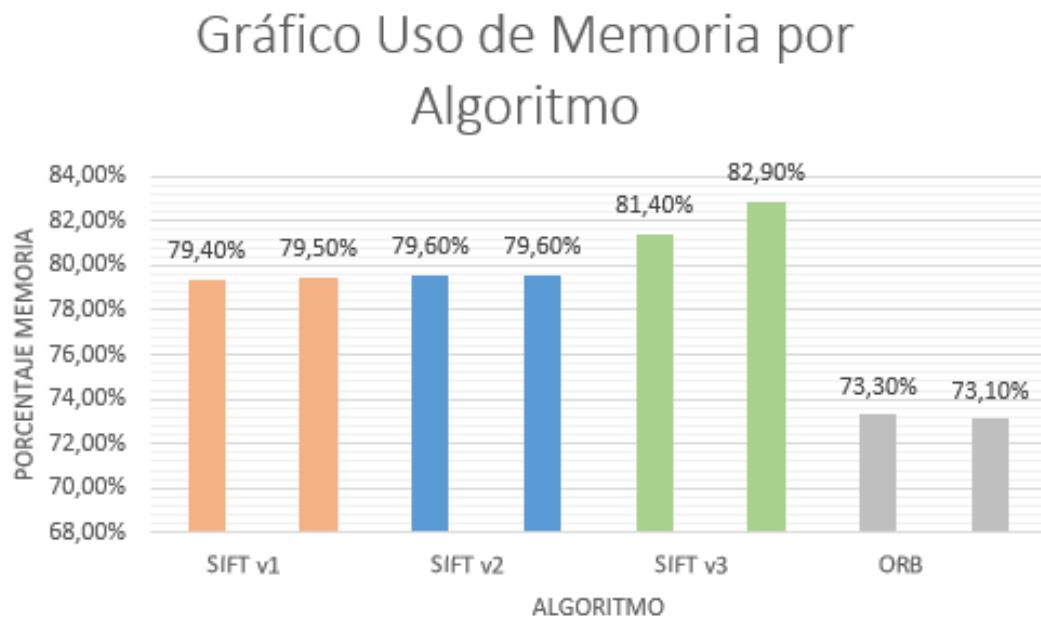


Figura 19: Gráfico de uso de memoria por algoritmo testado

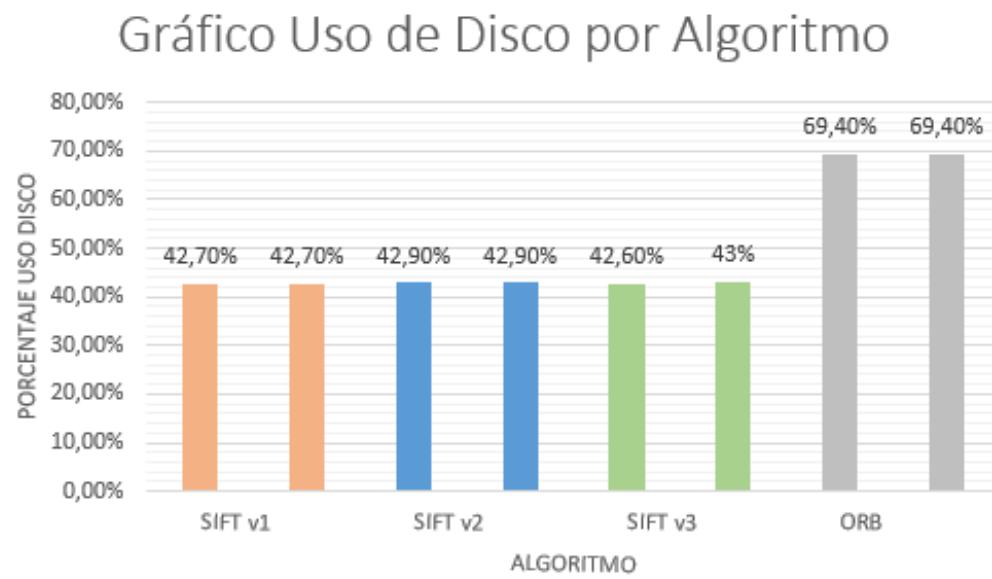


Figura 20: Gráfico uso de disco por algoritmos testeados

Tabla 8: Tabla resumen de las métricas obtenidas por algoritmos testeados

Algoritmo	Tamaño Dataset	Tiempo (segundos)	Uso Memoria	Uso Disco
SIFT v1	4056x3040 px	29,73	80,7 %	42,7 %
	1014x760 px	6,02	79,4%	42,7%
	253x190 px	0,647	79,5%	42,7%
SIFT v2	4056x3040 px	17,912	79,9%	42,8%
	1014x760 px	5,749	79,6%	42,9%
	253x190 px	0,655	79,6%	42,9%
SIFT v3	1014x760 px	2,991	81,4%	42,6%
	253x190 px	0,45	82,9%	43%
ORB	4056x3040 px	16,006	73,2%	69,4%
	1014x760 px	3,805	73,3%	69,4%
	253x190 px	6,031	73,1%	69,4%

También, se puede ver en la Figura 21 como queda finalmente una de las imágenes panorámicas a partir del método SIFT v1 solamente con cinco imágenes seleccionadas. Es por esto que se opta por tomar como referencia el algoritmo SIFT versión 3 con el uso de ORB para equilibrar los valores de bajo tiempo de ejecución, uso de memoria y uso de disco.

Los detalles de las versiones de los algoritmos y más tablas de valores se pueden ver (como se menciona anteriormente) en los Anexo A: Tabla de métricas e imágenes panorámicas generadas con algoritmo SIFT versión 1 Anexo B: Tabla de métricas e imágenes panorámicas generadas con algoritmo SIFT versión 2, Anexo C: Tabla de métricas e imágenes panorámicas generadas con algoritmo con técnica SIFT versión 3 y Anexo D: Tabla de métricas e imágenes panorámicas generadas con algoritmo ORB.

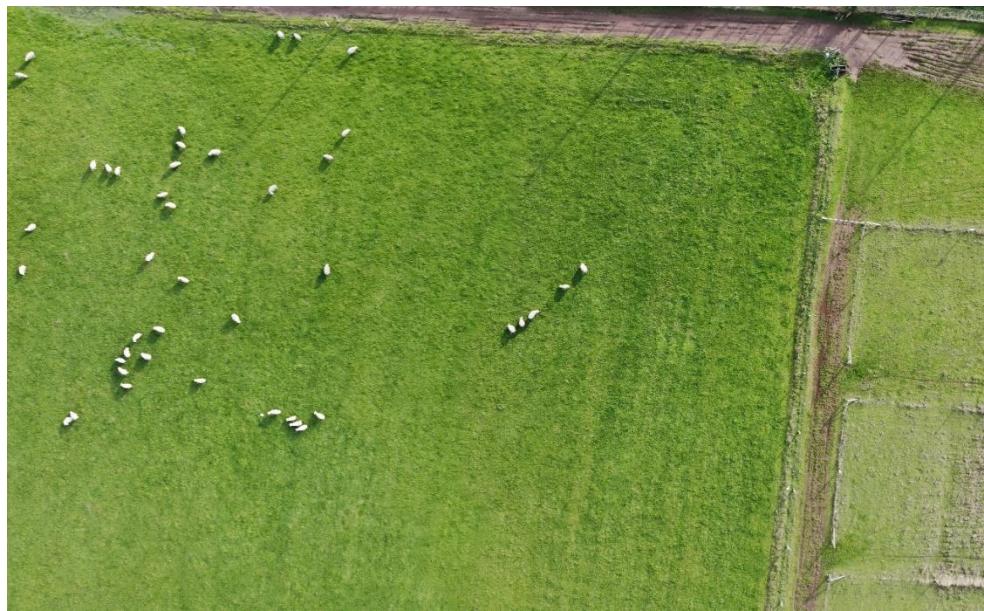


Figura 21:Imagen panorámica obtenida con método SIFT v1

5. ANÁLISIS Y DISEÑO

Los modelos de procesos de software son una representación simplificada de sus procesos. Cada uno de estos modelos representa y ofrece información parcial acerca de dicho proceso. Se pueden ver también como marcos de trabajo o “framework” pero no detalla específicamente las actividades de los procesos.

Estos marcos de procesos son abstracciones que ayudan a comprender mejor estos y se adaptan a cada proyecto para crear procesos de ingeniería de software más específicos.

Entre estos, los más usuales son:

1. Cascada (Waterfall): “La metodología Waterfall es un proceso de desarrollo secuencial de proyectos que suele utilizarse en el desarrollo de software. Esta metodología concibe el trabajo en un conjunto de etapas que deben ejecutarse una tras otra. Su nombre viene dado por las diferentes fases que componen el proyecto, ya que deben colocarse una encima de otra siguiendo un orden concreto y estricto de arriba hacia abajo” (Digital talent agency, 2021).
2. Desarrollo Incremental: “El modelo incremental combina la forma secuencial e iterativo a través de prototipos funcionales, es decir, cada evolución del proyecto se considera un incremento. El primer incremento que sale (primera versión del software), contiene elementos básicos (Core) del proyecto hasta seguir con los siguientes incrementos para mejorar su funcionalidad, priorizando los

requerimientos más importantes. Es importante definir la cantidad de incrementos que se requieren para crear el software” (Sánchez Vázquez, 2018).

3. Ingeniería de software orientada a la reutilización: “La ingeniería de software orientada a la reutilización es un proceso que conlleva al uso recurrente de activos de software en la especificación, análisis, diseño, implementación y pruebas de una aplicación o sistema de software” (Montilva, Arapé, & Colmenares, 2003).

Actualmente, la entrega y el desarrollo rápido son el requerimiento fundamental de los sistemas de software, por lo que se sobrepone la calidad del software y el compromiso con los requerimientos para lograr, con mayor rapidez, las implementaciones que necesita el software. Es por estos entornos cambiantes donde los requerimientos iniciales van variando de forma inevitable es que se proponen los métodos ágiles.

Estos métodos ágiles son métodos de desarrollo incremental, donde los incrementos son mínimos, se crean liberaciones del sistema y cada cierta cantidad de semanas, dos a tres, se les entregan a los clientes para su uso de manera temprana. La idea principal de que los clientes estén involucrados en estos métodos ágiles de manera constante es la retroalimentación temprana y rápida debido a los requerimientos cambiantes, minimizando recursos si es que se hicieran estas instancias ya finalizadas los softwares.

Entre los métodos ágiles más comunes se encuentran:

1. Programación Extrema: “Es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. XP se basa en realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios” (Bautista, 2017).
2. Scrum: “Es proceso de enfoque ágil para desarrollar software Esta metodología centra su atención en las actividades de Gerencia y no especifica prácticas de Ingeniería. Fomenta el surgimiento de equipos autodirigidos cooperativos y aplica inspecciones frecuentes como mecanismo de control. Scrum parte de la base de que los procesos definidos funcionan bien sólo si las entradas están perfectamente definidas y el ruido, ambigüedad o cambio es muy pequeño. Por lo tanto, resulta ideal para proyectos con requerimientos inestables, ya que fomenta el surgimiento de los mismos” (Peralta, 2003).
3. Kanban: “El Kanban (, significa “tarjeta” o “tablero”) se basa en un sistema de producción que dispara trabajo solo cuando existe capacidad para procesarlo. El disparador de trabajo es representado por tarjetas Kanban de las cuales se dispone de una cantidad limitada. Cada tarjeta Kanban acompaña a un ítem de trabajo durante todo el proceso de producción, hasta que este, es empujado fuera del sistema, liberando una tarjeta. Un nuevo ítem de trabajo, solo podrá ser

ingresado/aceptado si se dispone de una tarjeta Kanban libre” (Kniberg, Skarin, de Mary Poppendieck, & Anderson, 2010).

La que más se asocia a este proyecto es el desarrollo incremental, debido a que cada funcionalidad se realiza un análisis, su diseño, su código y su prueba. Al finalizar esto se da paso al siguiente incremento.

Las primeras actividades que darían paso a los requerimientos para la creación de este software se pueden visualizar en las tablas de los Anexo E: Tabla de actividades del proyecto de título y sus descripciones y Anexo F: Tabla de secciones del algoritmo adaptativo.

6. IMPLEMENTACIÓN

La implementación y la realización de pruebas del algoritmo adaptativo parte desde que se ingresan las imágenes tomadas hasta cuando entrega una panorámica.

El diagrama del algoritmo se apreciaría de la siguiente forma (ver Figura 22):

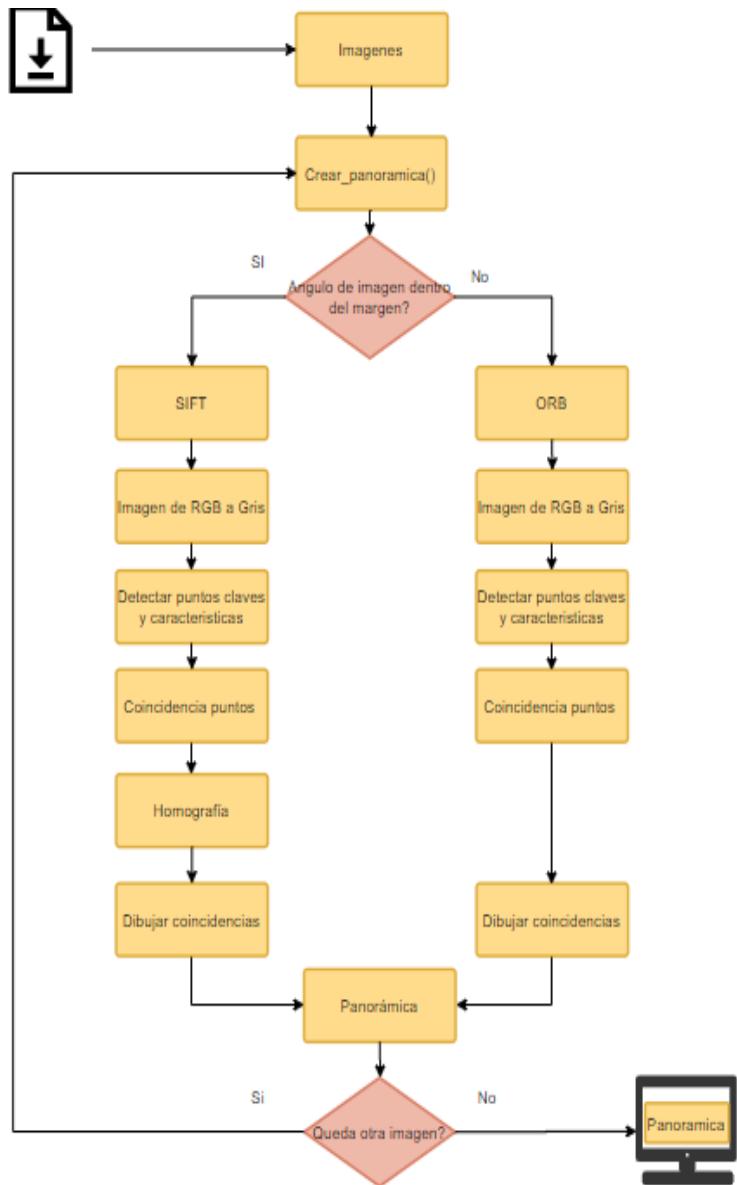


Figura 22: Diagrama algoritmo adaptativo

Al momento de iniciar el programa se le debe dar un parámetro al script de Python, el cual consiste en “-recortar”, donde si se le da el valor “1” se considerará recortar bordes negros de la imagen panorámica final, en caso de ser 0 esta los dejará.

El primer paso vendría siendo la importación de las librerías a utilizar (ver Figura 23).

Para este trabajo de título se consideraron el uso de las siguientes librerías al momento de inicializar el script main.py:

- Cv2: Usado para detección de objetos, análisis y tratamiento de las imágenes.

- Argparse: Sirve para solicitar algún argumento al momento de ejecutar el script, como también para facilitar mensajes de ayuda en caso de errores.
- Numpy : Se usa habitualmente para aplicar funciones matemáticas complejas a matrices y vectores.
- Math: Provee funciones matemáticas.
- Ndimage: Contiene funciones para procesamiento de imágenes.
- Listdir, Isfile e Isdir: Estas tres librerías trabajan en conjunto, ya que contienen funciones que facilitan la interacción con el sistema operativo. Estas librerías se les entrega una ruta específica, la cual hace referencia al o a los archivos a los cuales queremos alcanzar.
- Timeit: Entrega y mide valores de tiempos de ejecución del programa.
- Psutil: Sirve para monitorear recursos de la máquina, ya que entrega información relacionada a : CPU, disco, memoria, entre otras.

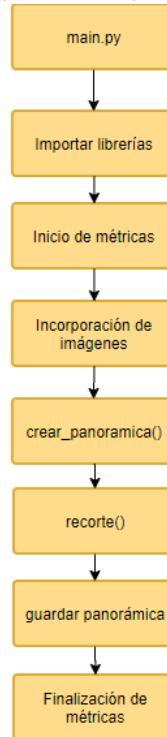


Figura 23: Diagrama main.py

Paso posterior a este, se define la ruta donde deben ir las imágenes y se van almacenando las imágenes en un array. Luego se hace el llamado a la primera función perteneciente a otro script, este se llama stitching.py (ver Figura 24), la cual obtiene como parámetros las últimas dos imágenes del array, ya que se irá creando la panorámica desde la última imagen hasta la primera.

Al llamar a la función crear panorámica, perteneciente a stitching.py, este recibe como parámetros dos imágenes las cuales serán “imagenA” e “imagenB”, se analizará su grado de inclinación con la función “rotate” perteneciente al script rotación.py, esta devolverá un valor booleano que puede ser “True” o “False” dependiendo del resultado obtenido. Estos valores también se les conoce como “1” o “0”

Si devuelve un valor True para ambas imágenes, el algoritmo elegirá el método de stitching SIFT, donde comenzará por detectar los puntos claves y características de ambas imágenes, para este proceso es necesario volver ambas a color gris. Ya teniendo ambas cosas, tanto puntos clave como características se busca la coincidencia de estos puntos ocupando lo mencionado en la 2.4, donde se busca la distancia menor entre los descriptores, la cual sería la distancia euclíadiana. Luego se usa la homografía mencionada en 2.5, para buscar los inliers que se aceptan y cuales se rechazan, los cuales vendrían siendo los datos con valores atípicos, esto gracias al método RANSAC, finalizando se verifica si hubo coincidencia entre las imágenes para posteriormente dibujar las líneas donde ambas imágenes tuvieron puntos de coincidencia.

En cambio sí devuelve al menos un valor False, el algoritmo elegirá el método de stitching ORB, donde realiza pasos similares al SIFT, empezando con cambiar las imágenes a escalas de grises igual, inicializa el detector ORB para encontrar los puntos clave y las características de ambas imágenes, luego a través de Bruteforce, la cual ”consiste en tomar el descriptor de una característica del primer conjunto y se compara con todas las demás características del segundo conjunto mediante un cálculo de distancia, donde devolverá el más cercano, luego, realiza un ordenamiento de estas coincidencias dependiendo la distancia entre ellas para finalizar con dibujar las líneas donde ambas imágenes tuvieron puntos de coincidencia” (Opencv, 2021).

Tras finalizar uno de ambos procesos, ambos entregan como resultado una imagen unida con sus coincidencias, para luego, en caso de existir otra imagen esta se compare y se una con la imagen recién unida. Este proceso se realiza de manera iterativa dependiendo la cantidad de imágenes que se quiera realizar el Stitching.

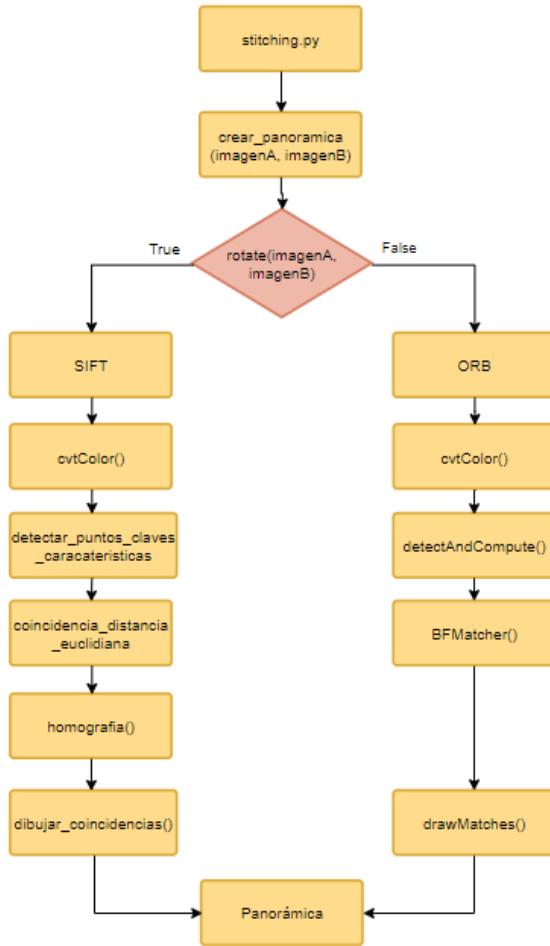


Figura 24: Diagrama stiching.py

En el paso anterior se menciona que la función `rotate` entregará un verdadero o falso dependiendo del ángulo de inclinación de la imagen entregada como parámetro, esto se realiza bajo la función `rotate()` perteneciente a `rotación.py`, donde por medio de dos funciones propias de OpenCV, la cual analiza los objetos en la imagen y otra que obtiene los bordes de la imagen puede obtener el ángulo de esta. Si el ángulo de la imagen está fuera del rango entre -60° y 96° esta entregará un valor “False”, que en términos generales es que no se cumple la condición, en caso contrario entregará un “True”, la cual significa que la imagen si está entre los valores -60° y 96° , cumpliendo así la condición (ver Figura 25).

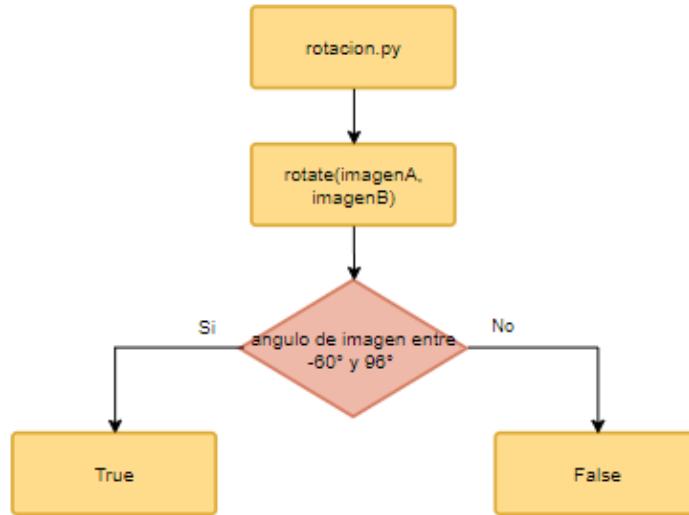


Figura 25: Diagrama rotación.py

7. RESULTADOS

7.1 Resultados algoritmo adaptativo con Dataset de prueba

Para esta etapa se asignó ocupar el mismo Dataset de imágenes que en la sección 5 ya que se buscaba estandarizar y alinear las pruebas que se realizaron anteriormente con las distintas formas de realizar stitching versus con el algoritmo adaptativo.

El Dataset en el cual fue testeado el algoritmo adaptativo tenía dos resoluciones distintas, las cuales fueron: 1014x760px y 253x190px. El tamaño de 4056x3040px se omite por dos razones. La primera es que al tener una gran cantidad de píxeles el algoritmo adaptativo demoraba bastante tiempo más que los 3 minutos límites, que es el tiempo permitido para la realización de las panorámicas en la sala de monitoreo de las Torres del Paine. Y la segunda, es que el tamaño 4056x3040px es bastante comparado a las imágenes que capturan las cámaras en las Torres del Paine que son de 480x848px. Por ambas razones es que se descartó su evaluación y análisis de métricas.

En la Tabla 9 se puede apreciar cómo el algoritmo adaptativo tiene una diferencia significativa de tiempo, en la primera imagen de mayor resolución de 1014x760px tiene un tiempo aproximado de 4,58 segundos, mientras que en la imagen con 253x190px tiene un tiempo estimado de ejecución de 0,19 segundos aproximadamente. Esto se puede deber a que mientras más grande la imagen más definición tendrán ciertos puntos de interés por lo que el algoritmo tiende a buscar más detalles en las imágenes para realizar un correcto stitching. Pero a diferencia del uso de memoria donde la imagen con menor resolución tiene un uso de memoria inferior la cual es 75,8% mientras que la imagen de mayor resolución tiene un uso de memoria de 79,7%. Esto puede deberse a distintos factores tales como estrés del mismo computador, uso de más memoria para detectar ciertos puntos que al tener una resolución baja de imágenes esta tiende a forzar más para encontrar una similitud de mejor manera, entre otras.

Los otros Dataset de imágenes de la zona donde el dron tomó capturas, con sus respectivas imágenes panorámicas creadas y sus métricas obtenidas se encuentran en el Anexo G: Tabla de métricas e imágenes panorámicas con algoritmo adaptativo.

Tabla 9: Resultados métricas algoritmo adaptativo con tercer Dataset

Algoritmo	Tamaño Tercer Dataset	Tiempo (segundos)	Uso Memoria	Uso Disco
Algoritmo Adaptativo	1014x760 px	4,576	75,8%	58,5%
	253x190 px	0,185	79,7%	58,5%

En la Figura 26 y la Figura 27 se pueden ver las panorámicas respectivas creadas a partir del algoritmo adaptativo, se puede apreciar las diferencias notorias en la calidad de los píxeles debido a la resolución de cada una. Recordar que la Figura 26 es la imagen panorámica de 1014x760 píxeles y la Figura 27 es la imagen panorámica de 253x190 píxeles.

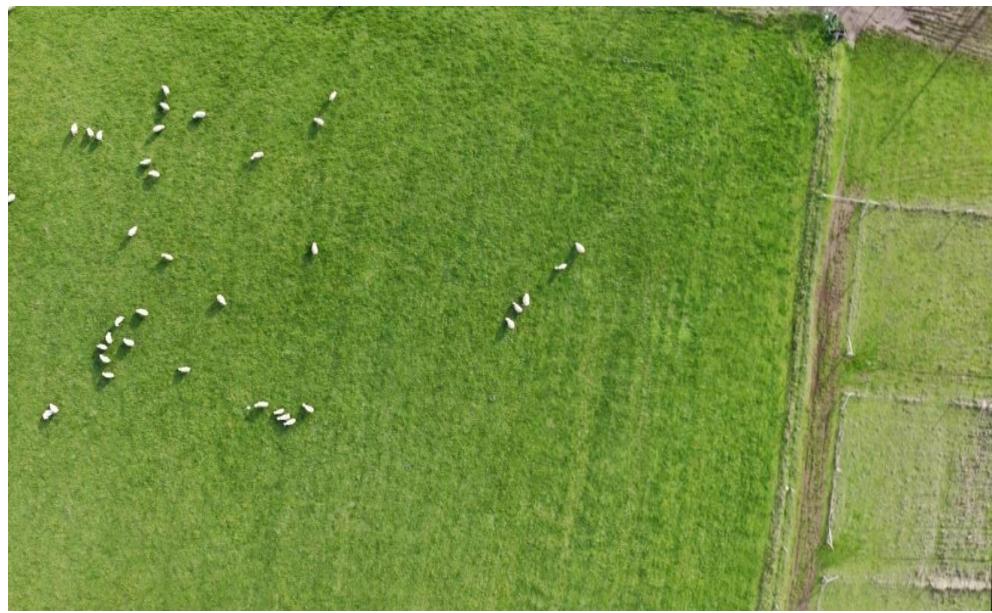


Figura 26: Panorámica resultante con tercer Dataset de prueba 1014x760px

A diferencia de la Figura 26 que tiene una mayor resolución, la Figura 27 se puede visualizar con un poco menos contenido, se puede apreciar que hay zonas que aparecen en la Figura 26 como la parte superior de la imagen o ambos costados de este donde en la Figura 27 no aparecen, esto se debe a que al crear la imagen panorámica estas imágenes al tener una menor definición costó más encontrar similitudes entre sus puntos clave y características por lo que imagen mientras se iba creando se fue desplazando en distintos ángulos, ocasionando que al momento de recortar la imagen recortará ciertos bordes o zonas que pertenecían al bloque que se debía omitir al obtener la imagen panorámica final.



Figura 27: Panorámica resultante con tercer Dataset de prueba 253x190px

En la Tabla 10 se puede apreciar un resumen de datos, la cual tiene las métricas evaluadas con los distintos Dataset de prueba que se usaron para el algoritmo adaptativo, esta fue plasmada en tres gráficos (ver Figura 28).

Tabla 10: Resultados métricas algoritmo adaptativo con Dataset Torres del Paine

Algoritmo	Dataset	Tamaño Dataset	Tiempo (segundos)	Uso Memoria	Uso Disco
Algoritmo Adaptativo	Primer Dataset	1014x760 px	6,752	75,6%	58,5%
		253x190 px	0,201	79,2%	58,5%
	Segundo Dataset	1014x760 px	3,814	79,8%	58,5%
		253x190 px	0,185	79,4%	58,5%
	Tercer Dataset	1014x760 px	4,576	75,8%	58,5%
		253x190 px	0,185	79,7%	58,5%

La Figura 28 corresponde al gráfico de tiempo de ejecución del algoritmo adaptativo en los distintos escenarios del Dataset de prueba, se puede ver una gran variedad de tiempo entre las imágenes con mayor cantidad de píxeles versus las con menor cantidad de píxeles. Los Dataset con imágenes de 1014x760 píxeles bordeaba entre los 3,8 y los 6,7 segundos, con un promedio de 5 segundos, mientras que para las imágenes de 253x190 píxeles, el tiempo de ejecución fue muy semejante entre los tres Dataset, la cual fue en promedio 0.18 segundos.

Gráfico Tiempo Algoritmo Adaptativo

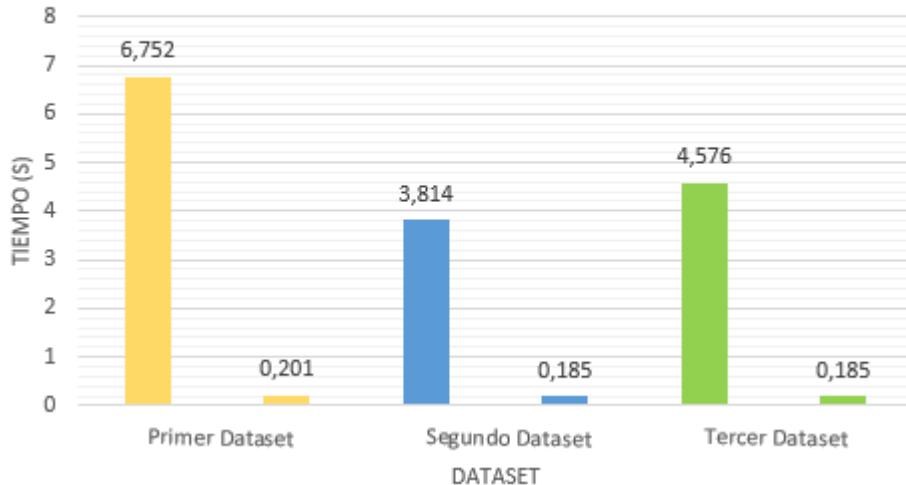


Figura 28: Gráfico tiempo de ejecución del algoritmo adaptativo

En la Figura 29 se puede apreciar el gráfico de uso de memoria mientras se realizaba la ejecución del algoritmo adaptativo. Se puede apreciar cierta tendencia entre las imágenes

con menor cantidad de píxeles las cuales serían 79,20% para el primer Dataset, 79,40% para el segundo Dataset y 79,70% para el tercero, mientras que para las imágenes con mayor cantidad de píxeles hubo una variedad significativa entre los tres tipos de Dataset, hubo un aproximado de 4%, incluso se puede apreciar que en el segundo Dataset ambas barras del gráfico de uso de memoria son casi similares para dos imágenes de tamaños distintos, esto puede deberse a muchos factores, entre estos puede ser que en este caso en particular el algoritmo le costó encontrar puntos y características claves por cómo era la imagen en sí, o también puede deberse a un factor de estrés de la máquina en la cual se hicieron las pruebas, donde el cambio de un test con otro no fue un tiempo prudente.

Gráfico Uso Memoria Algoritmo Adapativo

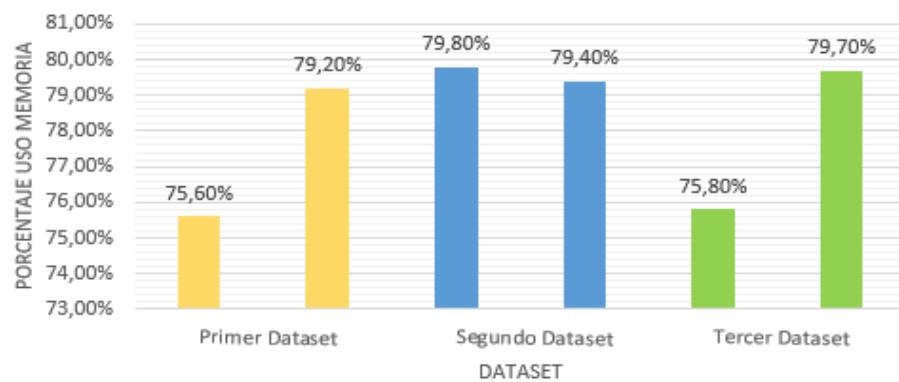


Figura 29: Gráfico uso de memoria algoritmo adaptativo

Finalmente, para la Figura 30, se puede visualizar el uso del disco del algoritmo adaptativo mientras este se ejecutaba, se puede apreciar 6 barras las cuales pertenecen a los tres tipos de Dataset, mencionados anteriormente. Se puede apreciar que el uso del disco se mantuvo de manera constante en cada prueba de esta, incluso variando la cantidad de píxeles entre las imágenes el uso del disco se mantuvo en 58,50%.

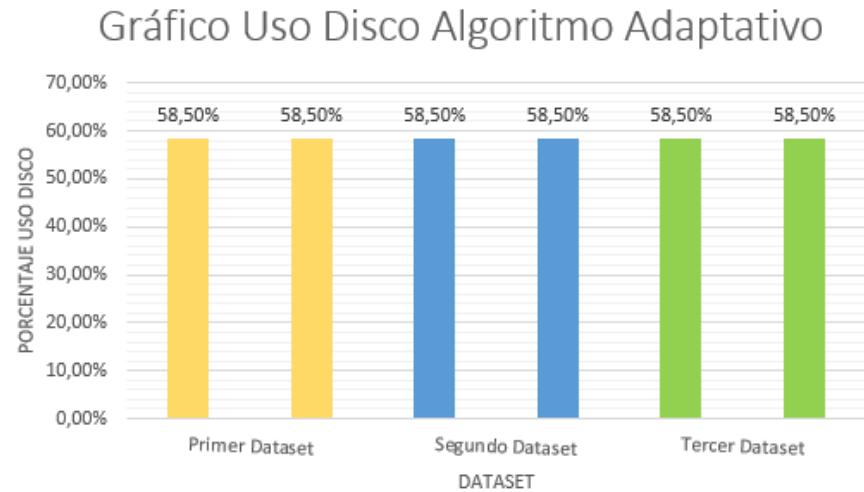


Figura 30: Gráfico uso de disco algoritmo adaptativo

7.2 Resultados algoritmo adaptativo con Dataset de Torres del Paine

Se llevó a cabo seleccionar la misma cantidad de imágenes de prueba para las Torres del Paine como cuando se realizó las pruebas en la sección anterior, indicando que se seleccionaron cinco imágenes. Las cinco imágenes seleccionadas se pueden apreciar en la Figura 31.

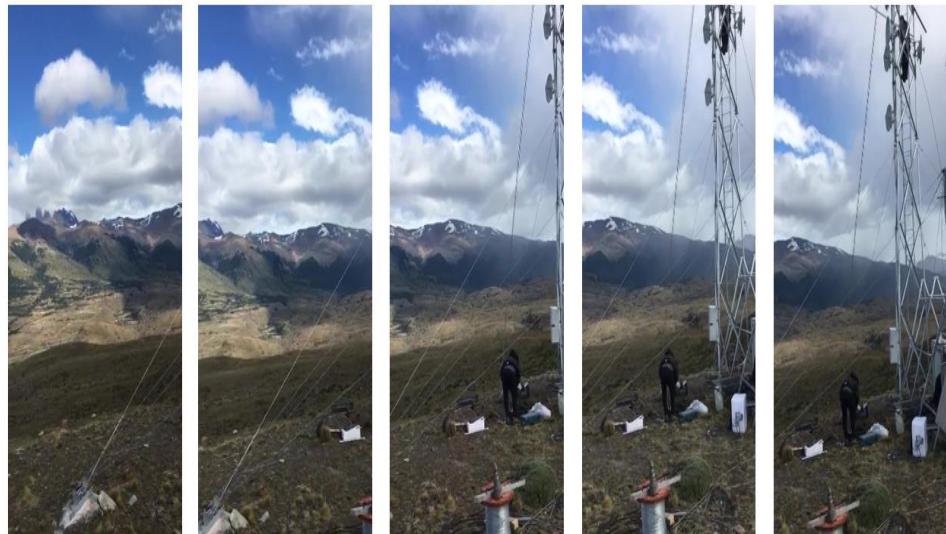


Figura 31: Secuencia de imágenes Torres del Paine

Al llevar estas imágenes al algoritmo adaptativo se puede ver en la Tabla 11 que el algoritmo demoró 1,243 segundos con un uso de memoria de 79% y uso de disco de 58,5%.

Tabla 11: Métricas algoritmo adaptativo Dataset Torres del Paine

Algoritmo	Tamaño Dataset Paine	Tiempo (segundos)	Uso Memoria	Uso Disco
Algoritmo Adaptativo	480x848 px	1,243	79,0%	58,5%

Para poder visualizar de mejor forma estas métricas se llevan a un gráfico (ver Figura 32).

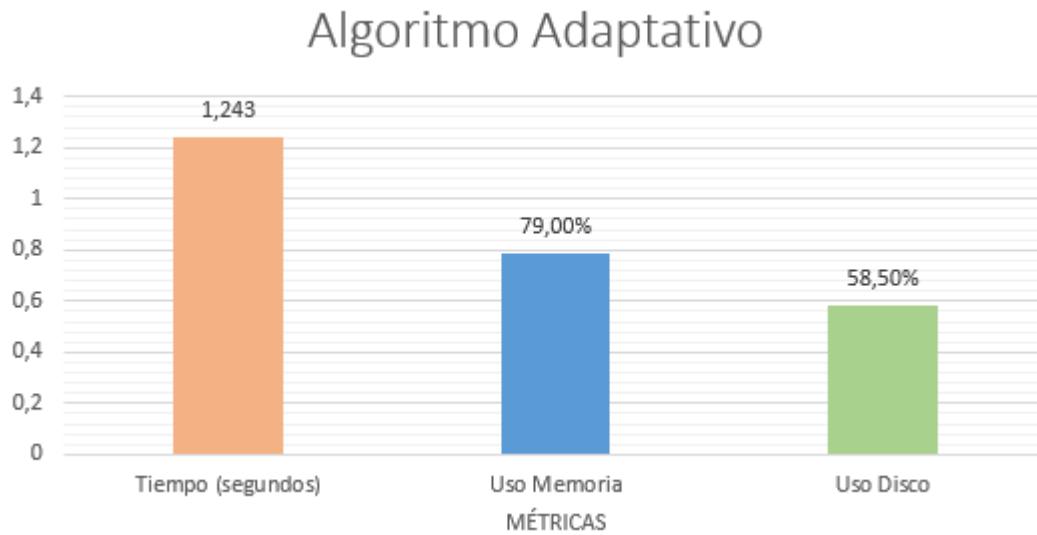


Figura 32: Gráfico algoritmo adaptativo Dataset Torres del Paine

En la Figura 33 se puede apreciar ciertas líneas verdes las cuales nos ayudan a detectar cuáles son los puntos que el algoritmo encontró similitud para realizar stitching entre estas dos imágenes, cabe recalcar que la figura negra entre ambas imágenes es parte del algoritmo y que después es removido de forma automática antes que entregue la panorámica.

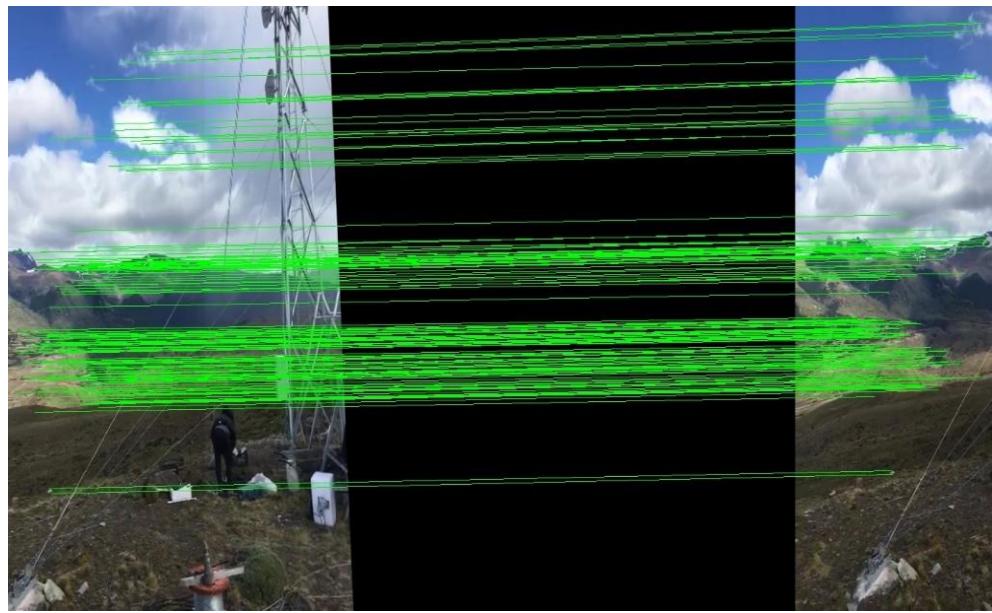


Figura 33: Localización similitudes entre imágenes

Tras finalizar la secuencia de procesos mencionados en la sección 2.1 se puede apreciar la imagen panorámica creada con técnicas de stitching (ver Figura 34).



Figura 34: Panorámica Torres del Paine

También se realizó otra prueba la cual consistía en ver cómo era el comportamiento del algoritmo adaptativo con imágenes difuminadas, dado que como se mencionó en la sección 3.1, la zona del Parque Torres del Paine debido a su particular ambiente tiende a tener problemas con difuminación o iluminación en las imágenes tomadas, aparte de los problemas con los ángulos distintos en las imágenes tomadas producto de los fuertes vientos.

Para el primer caso se observó cómo se comportaba en algoritmo adaptativo con una de las cinco imágenes difuminadas, en este caso se puede observar en la Figura 35 la diferencia entre una imagen no difuminada versus la imagen difuminada.

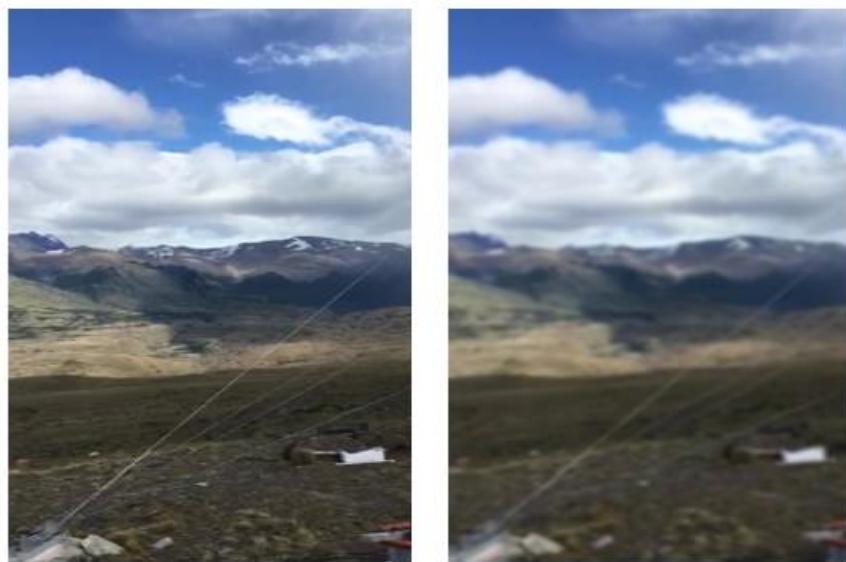


Figura 35: Imagen no difuminada versus imagen difuminada

Las métricas evaluadas fueron las mismas de los casos anteriores, dando un tiempo de ejecución de 1.110 segundos mientras que uso de memoria fue de 75.3% y uso del disco 58.6%, estos valores se pueden apreciar en la Tabla 12.

El algoritmo logró crear una imagen panorámica sin dificultad, la cual vendría siendo la Figura 36.

Tabla 12: Métricas algoritmo adaptativo una imagen difuminada

Algoritmo	Tamaño Dataset	Tiempo (segundos)	Uso Memoria	Uso Disco
Algoritmo Adaptativo	480x848 px	1.110	75,3%	58,6%



Figura 36: Imagen panorámica con una imagen difuminada

Tras haber realizado el stitching se llevó a cabo la difuminación a tres imágenes, las cuales el algoritmo adaptativo logró realizar un stitching de forma correcta (ver Figura 37).



Figura 37: Imagen panorámica con tres imágenes difuminadas

Se puede observar en la Tabla 13 que se logró una reducción de tiempo y de uso de memoria al usar tres imágenes difuminadas a diferencia de solamente una (ver tabla 15), debiéndose a que no hay tanta definición en la imagen, esta reduciría los puntos clave para

realizar similitudes. Mientras que el uso de disco se mantuvo casi igual en la Tabla 12 como en la Tabla 13.

Tabla 13: Métricas algoritmo adaptativo tres imágenes difuminadas

Algoritmo	Tamaño Dataset	Tiempo (segundos)	Uso Memoria	Uso Disco
Algoritmo Adaptativo	480x848 px	0,995	49,8%	58,5%

8. CONCLUSIONES

El objetivo general del proyecto de título, como se dijo en la sección 1.3.1 que era lograr una mejora significativa en las condiciones de borde aún no resueltas de uno de los algoritmos estudiados de creación de imágenes panorámicas con técnicas de stitching, con Dataset de imágenes de las cámaras de las Torres del Paine. Esto se logró ya que se encontró ciertos problemas al momento de efectuar la creación de una imagen panorámica debido a algunas imágenes que tuvieran un cierto grado de rotación al momento de capturarlas, y también, en un porcentaje de difuminación. Esto fue gracias a la adaptabilidad del algoritmo, ya que al momento de tener una falencia un método de stitching ocupaba uno que le favorece.

Mientras que los objetivos específicos mencionados en la sección 1.3.2 se irán analizando y argumentando cada uno de estos.

Para el primer objetivo específico el cual es “Evaluar el estado actual en el que se encuentra el procesamiento de imágenes a través de una investigación y revisión sistemática” se cumplió, ya que se recopilaron ciertos artículos, tecnologías y herramientas que usan actualmente métodos de stitching para la creación de imágenes panorámicas. Estas son mencionadas en la sección 2.

Para el segundo objetivo específico el cual es “Estudiar distintas técnicas de algoritmos de stitching” se logró ya que se describió de forma general todos los pasos que se realizan para la creación de una imagen panorámica con técnicas de stitching y luego se detalló las más usadas que fueron SIFT (ver sección 2.3.1), SURF (ver sección 2.3.2) y ORB (ver sección 2.3.3).

El tercer objetivo específico que es “Encontrar condiciones de borde o de mal funcionamiento de los algoritmos estudiados “se logró ya que para nuestro caso hubo imágenes con mayor resolución (4056x3040px) que, aunque no fuera una condición de borde directamente del método realizar un stitching de estas afectaba al tiempo que ejecución del algoritmo demorando mucho más de tres minutos. También, hubo algoritmos estudiados que directamente arrojaba errores si la imagen no pertenecía a la línea de la secuencia de las imágenes, por ejemplo, si la imagen con la que se quería hacer stitching no pertenecía ni a la izquierda ni a la derecha, si no que pertenecía a una zona arriba de la imagen y producto de esto arrojaba un error en el algoritmo, ya que no se mantenían en el mismo eje “X”, también, que, dependiendo del ángulo de la imagen, si este era muy amplio el algoritmo fallaba. Otra condición de borde era que al tener muchas imágenes adyacentes descartadas de la secuencia tendían a fallar, aunque para el ojo humano era aún posible detectar ciertas similitudes entre las imágenes restantes.

El cuarto objetivo específico hace referencia a “ Evaluar y desarrollar un algoritmo adaptativo que permita generar las imágenes, utilizando los algoritmos evaluados” Este se cumplió de forma parcial, ya que se implementaron solamente dos métodos distintos para realizar stitching, ya que uno de estos métodos que se consideró que podía ser un buen

recurso en este algoritmo adaptativo estaba patentado, el cual era el método SURF, mientras que otra forma que pudo igual ayudar a encontrar mayores puntos clave y características pero que no se logró implementar al algoritmo fue el método PCA (Principal Component Analysis), que aunque no sea un método directamente para la realización de stitching si puede efectuar ciertas mejoras en métodos como en SIFT. Se darán más detalles del PCA en las mejoras a llevarse a cabo a futuro.

Aunque el algoritmo ORB generó unos valores mayores en las métricas obtenidas (ver Anexo D: Tabla de métricas e imágenes panorámicas generadas con algoritmo ORB) esto no sería un mayor problema en este caso de estudio, ya que el tiempo está dentro de los estándares permitidos, pero si se quisiera implementar esta técnica para obtener imágenes panorámicas casi de manera instantánea el uso de PCA probablemente sea muy necesario.

Finalmente, es importante mencionar que el desarrollo de este proyecto permitió aplicar en gran parte variados conocimientos y aprendizajes adquiridos durante el transcurso de estudio de la carrera profesional; principalmente en el ámbito de programación, ingeniería de software, lógica y operaciones matemáticas, esto debido a que el proyecto realizado mezcla e integra diversas tecnologías y técnicas; sin embargo, el tratado de imágenes está cada vez más inculcado en los softwares se debería aplicar con mayor frecuencia en la carrera, debido a que se puede tener un impacto enorme en varias áreas que son totalmente distintas entre ellas.

9. TRABAJOS FUTUROS Y MEJORAS

Tras realizar el algoritmo adaptativo se detectaron varias mejoras durante las pruebas que se podrían implementar para futuros desarrollos o mejoras, para lograr así tener un algoritmo adaptativo que se adapte a la mayor cantidad de condiciones posibles.

- Realizar un stitching a imágenes con tamaño de resoluciones superior a 4056x3040px. Reducir esta imagen a un tamaño predeterminado para lograr obtener toda la información y plasmarla de forma correcta a las imágenes originales sin tener perdidas de información significativa en este proceso.
- Un filtro o un método de stitching que se adapte mejor a un grado de difusión mayor aplicado a las imágenes que la que se ocupó para este proyecto, ya que nada impide que al momento de tomar una captura la imagen se vea casi al 100% difusa.
- Tratar de aplicar PCA para encontrar objetos en las imágenes que tengan una imagen adyacente que no se pueda visualizar o que simplemente se haya “extraviado” en la transmisión de la data. Ya que al ser una secuencia de imágenes partes del objeto detectado tienden a estar en más de una imagen (ver Anexo H: Imagen PCA (Principal Component Analysis)).
- Aplicar un suavizado, algún filtro especial o rotar la imagen al ángulo correspondiente a la imagen adyacente a esta, para que solucione el inconveniente de la línea generada al unirse las imágenes, ya que provoca que se vea como una imagen unida no completamente a la perfección.
- Aplicar más variedad de métodos de stitching enfocados en otras condiciones de bordes encontradas en este caso de uso para cubrir la mayoría de las problemáticas que trae el clima particular de las Torres del Paine.
- Una mejora significativa que se podría optar es tratar de ajustar o implementar el algoritmo adaptativo en un hardware o plataforma web en la nube que cuente con TPU (unidad de procesamiento tensorial) o GPU (unidad de procesamiento gráfico) más avanzada que donde se probó, ya que podría mejorar bastante algunos valores obtenidos en las métricas y el tiempo de ejecución se reduzca bastante.

10. REFERENCIAS

- Anaconda individual edition. (4 de diciembre de 2020). *Accedido el 4 de diciembre, 2020 desde <https://www.anaconda.com/products/individual>.*
- Araúz Pisón, M. T. (2016). *Robot coordination to create collaborative panorama images*. Sevilla.
- Baudisch, P., Tan, D., Steedly, D., Rudolph, E., Uyttendaele, M., Pal, C., & Szeliski, R. (2005). Panoramic viewfinder: providing a real-time preview to help users avoid flaws in panoramic pictures. In *Proceedings of the 17th Australia conference on Computer-Human Interaction: Citizens Online: Considerations for Today and the Future*, (págs. 1-10).
- Bautista, J. (2017). *Programación Extrema (XP)*. La Paz, Bolivia.
- Bay, H., Tuytelaars, T., & Van Gool, L. (2006). SURF: Speeded Up Robust Features. *European conference on computer vision* (págs. 404-417). Berlin, Heidelberg: Springer.
- Brown, M., & Lowe, D. G. (2007). Automatic panoramic image stitching using invariant features. *International journal of computer vision*, 74(1), 59-73.
- Carrión-Ruiz, B., & Lerma García, J. L. (2017). Análisis de componentes principales de imágenes multiespectrales en el ámbito del arte rupestre. *Primer Congreso en Ingeniería Geomática* (págs. 41-47). Editorial Universitat Politècnica de València.
- Chen, C. Y., & Klette, R. (1999). Image stitching—Comparisons and new techniques. In *International conference on computer analysis of images and patterns* (págs. 615-622). Berlin, Heidelberg: Springer.
- Chile: Incendios forestales – enero 2017 - Reporte de Situación No. 02 (al 07 de febrero de 2017). (15 de agosto de 2020). *Accedido el 15 agosto, 2020, desde <https://reliefweb.int/report/chile/chile-incendios-forestales-enero-2017-reportede-situaci-n-no-02-al-07-de-febrero-de>.*
- D., L. S., H., L. J., & C., C. C. (2011). Using gradient features from scale-invariant keypoints on face recognition. *International Journal of Innovative Computing, Information and Control*, 7(4), 1639-1649.
- Derpanis, K. G. (2004). The harris corner detector. *York University*, 2.
- Digital talent agency. (10 de enero de 2021). *Accedido el 10 enero, 2021 , desde https://www.dtagency.tech/cursos/metodologias_gestion_proyectos/tema_1-ModeloWaterfall.pdf.*
- Dirección Meteorológica de Chile - Servicios Climáticos. (15 de diciembre de 2020). *Accedido el 15 de diciembre, 2020, desde <https://climatologia.meteochile.gob.cl/application/publicaciones/anuario/2019> página 89.*
- Flora, G. R., & Zheng, J. Y. (2007). Adjusting route panoramas with condensed image slices. In *Proceedings of the 15th ACM international conference on Multimedia*, (págs. 815-818).
- Fu, J., Jing, X., Sun, S., Lu, Y., & Wang, Y. (2012). C-surf: Colored speeded up robust features. In *International Conference on Trustworthy Computing and Services* (págs. 203-210). Berlin, Heidelberg: Springer.
- Gálvez Ortiz, S. A. (2017). *Desarrollo de un algoritmo de stitching para secuencia de imágenes con amplios movimientos de cámara*.

- González, J. A., Álvarez, J., Martínez, F. E., & Ascencio, J. I. (2009). Algoritmo para optimizar y diseñar un filtro adaptativo de correlacion. *e-Gnosis*, 7, 1-8.
- Huang, Z., Wang, H., & Li, Y. (2019). The Research of Image Mosaic Techniques Based on Optimized SIFT Algorithm. *In Proceedings of the 2019 International Conference on Robotics Systems and Vehicle Technology*, (págs. 80-86).
- Huang, Z., Wang, H., & Li, Y. (2019). The Research of Image Mosaic Techniques Based on Optimized SIFT Algorithm. *In Proceedings of the 2019 International Conference on Robotics Systems and Vehicle Technology*, (págs. 80-86).
- Infobae. (20 de noviembre de 2020). *Accedido el 20 de noviembre, 2020, desde https://www.infobae.com/america/ciencia-america/2020/03/05/la-nasa-difundio-una-imagen-de-marte-con-1800-millones-de-pixels*.
- Juan, L., & Gwun, O. (2009). A comparison of sift, pca-sift and surf. *International Journal of Image Processing*, 3, págs. 143-152.
- Juan, L., & Oubong, G. (2010). SURF applied in panorama image stitching. *In 2010 2nd international conference on image processing theory, tools and applications* (págs. 495-499). IEEE.
- Kawaiji, H., Hatada, K., Yamasaki, T., & Aizawa, K. (2010). Image-based indoor positioning system: fast image matching using omnidirectional panoramic images. *In Proceedings of the 1st ACM international workshop on Multimodal pervasive video analysis*, (págs. 1-4).
- Ke, Y., & Sukthankar, R. (2004). PCA-SIFT: A more distinctive representation for local image descriptors. (*Vol. 2, pp. II-II*). IEEE, (pág. In Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition).
- Kniberg, H., Skarin, M., de Mary Poppendieck, P., & Anderson, D. (10 de enero de 2010). *Kanban y Scrum—obteniendo lo mejor de ambos. Prólogo de Mary Poppendieck & David Anderson*. Estados Unidos: C4Media Inc.
- Learn to code with Visual Studio Code. (4 de diciembre de 2020). *Accedido el 4 de diciembre, 2020, desde https://code.visualstudio.com/learn*.
- Li, J., & Du, J. (2010). Study on panoramic image stitching algorithm. *Second Pacific-Asia Conference on Circuits*. 1, págs. 417-420. IEEE.
- Li, X., Wu, L., Wang, G., Chen, Y., Rong, K., & Lin, J. (2017). Image Stitching and Quality Evaluation Algorithm for Large Size Parts. *In Proceedings of the International Conference on Video and Image Processing*, (págs. 130-134).
- Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International journal of computer visio*, 60(2), 91-110.
- Ma, X., Liu, D., Zhang, J., & Xin, J. (2015). A fast affine-invariant features for image stitching under large viewpoint changes. *Neurocomputing*, 1430-1438.
- Meneghetti, G., Danelljan, M., Felsberg, M., & Nordberg, K. (2015). Image alignment for panorama stitching in sparsely structured environments. *In Scandinavian Conference on Image Analysis*, (págs. 428-439).
- Mistry, S., & Patel, A. (2016). Image stitching using Harris feature detection. *International Research Journal of Engineering and Technology*, 3(4), 2220-6.
- Mohamad, F. S., Alsuhimat, F. M., Mohamed, M. A., Mohamad, M., & Jamal, A. A. (2018). Detection and Feature Extraction for Images Signatures. *International Journal of Engineering & Technology*, 44-48.

- Montilva, J. A., Arapé, N., & Colmenares, J. (2003). Desarrollo de software basado en componentes. *Congreso de Automatización y Control*. . Mérida.
- Mur-Artal, R., & Tardós, J. D. (2017). Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Transactions on Robotics*, 33(5), 1255-1262.
- Mur-Artal, R., Montiel, J. M., & Tardos, J. D. (2015). ORB-SLAM: a versatile and accurate monocular SLAM system. *IEEE transactions on robotics*, 31(5), 1147-1163.
- OpenCV. (25 de septiembre de 2020). *Accedido el 25 de septiembre, 2020, desde https://docs.opencv.org/master/da/df5/tutorial_py_sift_intro.html*.
- Opencv. (4 de diciembre de 2020). *Accedido el 4 de diciembre, 2020, desde https://opencv.org/about/]*.
- OpenCV. (20 de enero de 2021). *Accedido el 20 de enero, 2021, desde https://docs.opencv.org/master/d1/d89/tutorial_py_orb.html*.
- OpenCV. (20 de enero de 2021). *Accedido el 20 de enero. 2021, desde https://docs.opencv.org/master/d9/dd2/tutorial_py_surf_intro.html*.
- Opencv. (10 de enero de 2021). *https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_matcher/py_matcher.html*.
- Ozawa, T., Kitani, K. M., & Koike, H. (2012). Human-centric panoramic imaging stitching. In *Proceedings of the 3rd Augmented Human International Conference*, (págs. 1-6).
- Patil, V. P., & Gohaatre, U. B. (2017). Performance comparison of image stitching methods under different illumination conditions. 4, 249-259.
- Peralta, A. (2003). *Metodología SCRUM*. Uruguay: Universidad ORT Uruguay.
- Petkova, Y., & Nuri, N. (2011). An algorithm for fast image registration and feature matching for the purposes of image stitching. In *Proceedings of the 12th International Conference on Computer Systems and Technologies*, (págs. 288-233).
- Qi, M., Wang, Y., Xu, Y., & Li, Z. (2019). An Efficient Method of PCB Images Stitching. In *Proceedings of the 2nd International Conference on Control and Computer Vision*, (págs. 55-59).
- Rivera Venegas, C. I. (2016). *Análisis espacio temporal de la cobertura vegetacional afectada por incendios en el Parque Nacional Torres del Paine utilizando datos ópticos-microondas pasivas*. Santiago.
- Rublee, E., Rabaud, V., Konolige, K., & Bradski, G. (2011). ORB: An efficient alternative to SIFT or SURF. *International conference on computer vision* (págs. 2564-2571). Ieee.
- Ruiz Ordinola, J. A. (2018). *STITCHING DE IMÁGENES DIGITALES TOMADAS POR UN VEHÍCULO AÉREO NO TRIPULADO (UAV)*.
- Sánchez Vázquez, J. D. (2018). *Aplicación del Modelo Incremental Para el Desarrollo del Sistema de Información Docente*. Texcoco, Estado de México.
- SGO. (20 de noviembre de 2020). *Accedido el 20 de noviembre, 2020, desde https://www.sgo.es/mistika-vr*.

- Shaowen, D., Xiaohu, Z., Jie, W., Hongliang, Z., & Lijun, Z. (2018). The fusion research of UAV image stitching. In *Proceedings of the 2018 10th International Conference on Computer and Automation Engineering*, (págs. 92-96).
- Song, Z. L., & Zhang, J. (2010). Remote sensing image registration based on retrofitted SURF algorithm and trajectories generated from Lissajous figures. *IEEE Geoscience and Remote Sensing Letters*, 7(3), 491-495.
- Suleiman, A., & Sze, V. (2016). An energy-efficient hardware implementation of HOG-based object detection at 1080HD 60 fps with multi-scale support. *Journal of Signal Processing Systems*, 84(3), 325-337.
- Summa, B., Tierny, J., & Pascucci, V. (2012). Panorama weaving: fast and flexible seam processing. *ACM Transactions on Graphics (TOG)*, 1-11.
- Tang, G., Liu, Z., & Xiong, J. (2019). Distinctive image features from illumination and scale invariant keypoints. *Multimedia Tools and Applications*, 78(16), 23415-23442.
- Tchinda, E. N., Kwadjo, D. T., & Bobda, C. (2019). Work-in-Progress: A Distributed Smart Camera Apparatus to Enable Scene Immersion. *International Conference on Hardware/Software Codesign and System Synthesis (CODES+ ISSS)* (págs. 1-2). IEEE.
- Tsuji, S., & Kono, Y. (2010). Generation of roadside panoramic images without obstacles. In *Proceedings of the International Conference on Advanced Visual Interfaces*, (págs. 424-424).
- Vaidya, O. S., & Gandhe, S. T. (2018). The study of preprocessing and postprocessing techniques of image stitching. *International Conference On Advances in Communication and Computing Technology (ICACCT)* (págs. 431-435). IEEE.
- Vaidya, O. S., Gandhe, S. T., & Sonawane, P. B. (2016). Performance Analysis of KLT, Harris and SIFT Feature Detector for Image Stitching. *International Journal of Electrical and Electronics Engineers*, (págs. 536-544).
- Wang, M., Niu, S., & Yang, X. (2017). A novel panoramic image stitching algorithm based on ORB. *International Conference on Applied System Innovation* (págs. 818-821). IEEE.
- What is Python? Executive Summary. (4 de diciembre de 2020). *Accedido el 4 de diciembre, 2020, desde https://www.python.org/doc/essays/blurb/*.
- Xi, J., Teng, P., & Wang, J. (2019). Multi-retinal Images Stitching Based on the Maximum Fusion and Correction Ratio of Gray Average. In *Proceedings of the Third International Symposium on Image Computing and Digital Medicine*, (págs. 60-64).
- Xia, X., Wang, B., Li, J., Song, L., Gong, Y. H., & Deng, B. S. (2020). Real-time Online UAV Images Mosaic with Robustness to Cumulative Errors. *International Conference on Computing, Networks and Internet of Things*, (págs. 174-179).
- Xin, L., Yarong, L., Donghui, L., Liyan, Y., Xiaofei, Y., & Yuchen, W. (2018). Spherical Image Stitching Based on ORB and PROSAC Algorithm. *n Proceedings of the 3rd International Conference on Intelligent Information Processing*, (págs. 160-165).
- Zhu, Z., Lu, J., Wang, M., Zhang, S., Martin, R. R., Liu, H., & Hu, S. M. (2018). A comparative study of algorithms for realtime panoramic video blending. *IEEE Transactions on Image Processing*, 27(6), 2952-2965.

ANEXOS

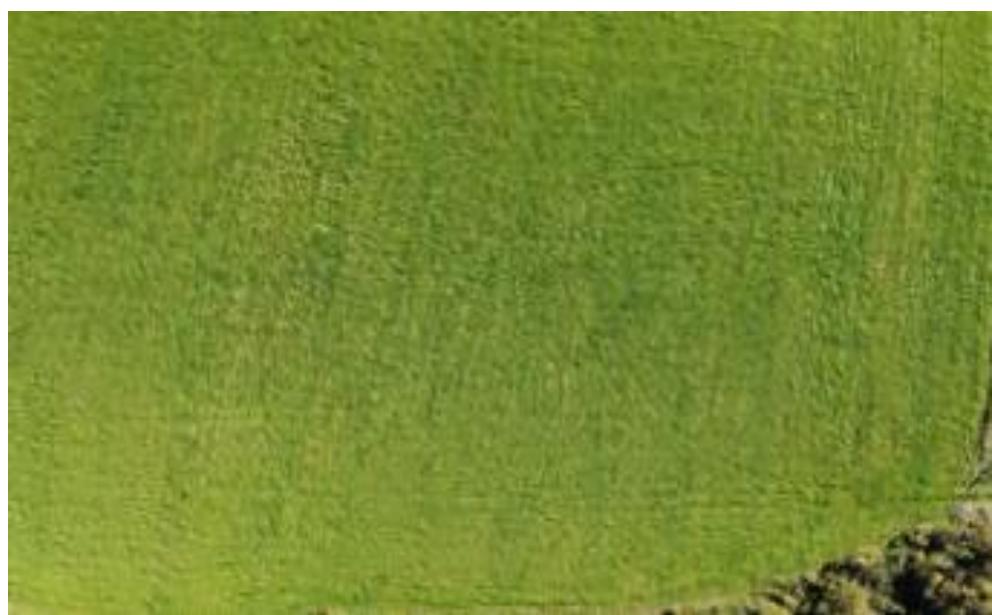
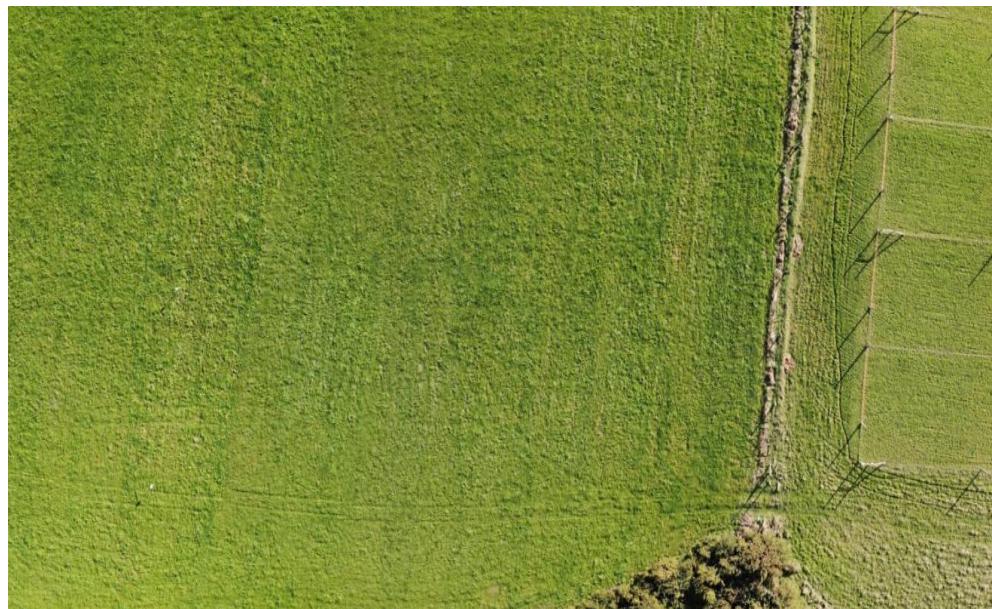
Anexo A: Tabla de métricas e imágenes panorámicas generadas con algoritmo SIFT versión 1

Acá se realizó las pruebas con un algoritmo con técnicas de stitching SIFT (SIFT v1). este algoritmo ocupaba librerías complementarias a openCV la cual se le llama imutils, esta librería lograba generar una imagen panorámica limpia y sin cortes notorios entre imágenes.

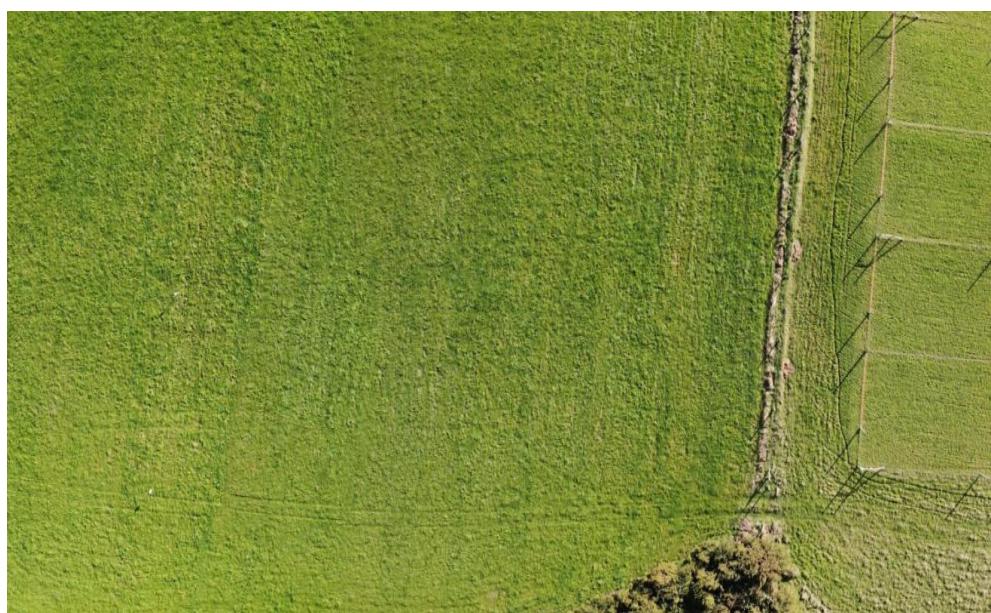
Mientras más se fuera reduciendo los píxeles en las imágenes más se iba reduciendo la totalidad de la imagen panorámica, incluso se puede apreciar zonas “desaparecidas” entre una imagen y otra.

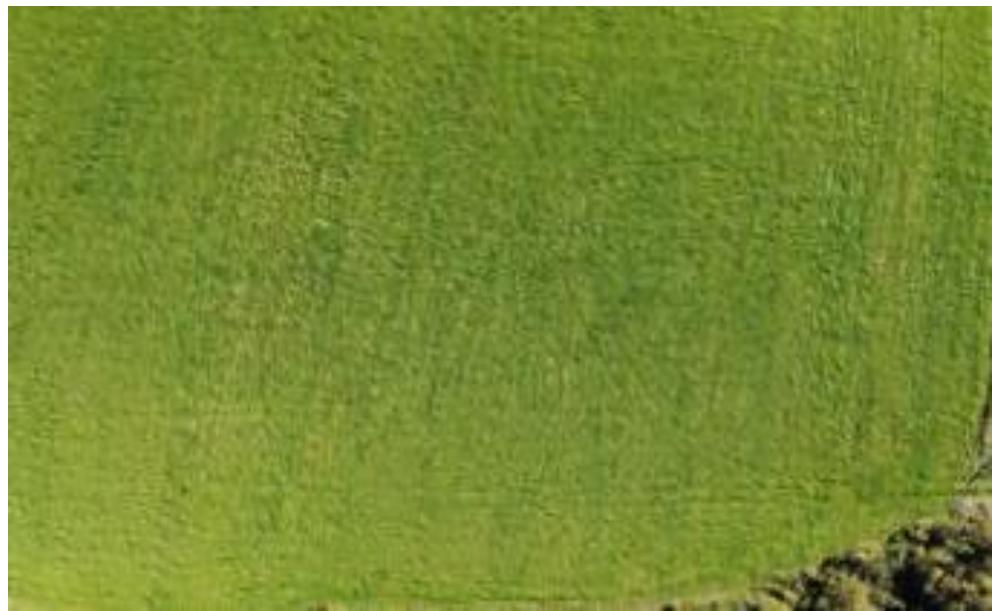
Tamaño Primer Dataset	Tiempo (segundos)	Uso Memoria	Uso Disco
4056x3040 px	21,34	80,3%	42,9%
1014x760 px	2,84	79,2%	42,7%
253x190 px	0,2962	79,0%	42,6%





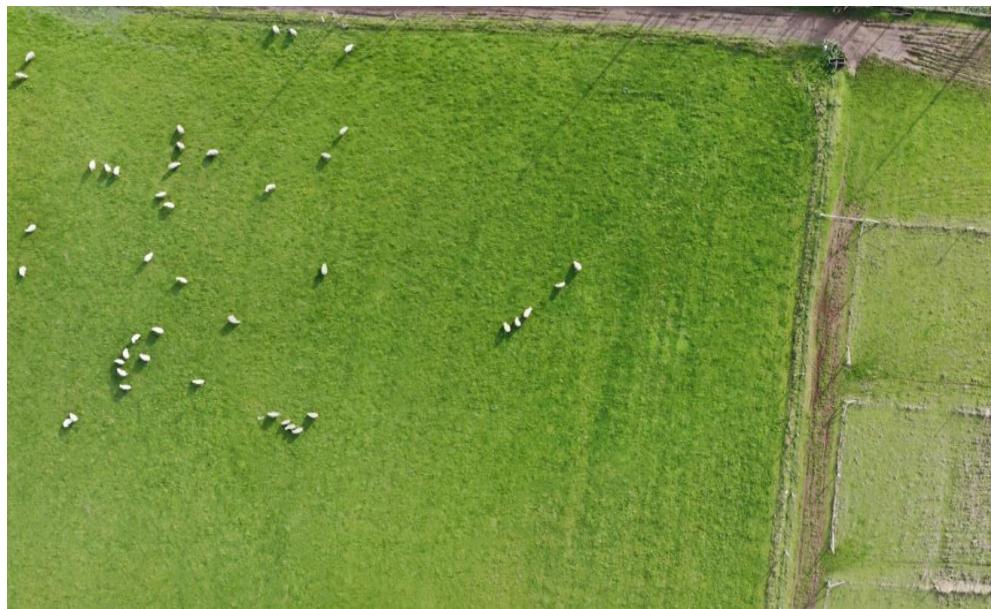
Tamaño Segundo Dataset	Tiempo (segundos)	Uso Memoria	Uso Disco
4056x3040 px	29,73	80,7 %	42,7 %
1014x760 px	6,02	79,4%	42,7%
253x190 px	0,647	79,5%	42,7%





Tamaño Dataset	Tercer	Tiempo (segundos)	Uso Memoria	Uso Disco
4056x3040 px		20,089	81%	43%
1014x760 px		4,838	79,4%	42,7%
253x190 px		0,519	79,5%	42,7%





Anexo B: Tabla de métricas e imágenes panorámicas generadas con algoritmo SIFT versión 2

Se realizaron las pruebas con un algoritmo con técnicas de stitching SIFT (SIFT v2). este algoritmo ocupaba librerías complementarias a openCV(imutils), la diferencia es que este algoritmo solo permitía dos imágenes, por lo que era imposible recibir una secuencia de capturas. Se fueron realizando las pruebas de manera manual y sin recorte, logrando un poco más tediosa las pruebas, pero quedan una imagen aceptable al finalizar.

Ocurre lo mismo que el algoritmo v1, mientras más se fuera reduciendo los píxeles en las imágenes más se iba reduciendo la totalidad de la imagen panorámica. La diferencia es que al quedar imágenes panorámicas con bordes negros estas se pueden remover de mejor forma a como se realizó con el algoritmo anterior, dando la posibilidad de que esta falencia de que se “desaparezca” ciertas zonas de la imagen panorámica creada sean en una menor cantidad.

Tamaño Primer Dataset	Tiempo (segundos)	Uso Memoria	Uso Disco
4056x3040 px	14,582	79,6%	42,7%
1014x760 px	2,579	79,5%	42,9%
253x190 px	0,288	79,7%	42,9%





Tamaño Segundo Dataset	Tiempo (segundos)	Uso Memoria	Uso Disco
4056x3040 px	17,912	79,9%	42,8%
1014x760 px	5,749	79,6%	42,9%
253x190 px	0,655	79,6%	42,9%





Tamaño Tercer Dataset	Tiempo (segundos)	Uso Memoria	Uso Disco
4056x3040 px	16,594	79,7%	42,8%
1014x760 px	4,715	79,6%	42,9%
253x190 px	0,521	79,7%	42,9%





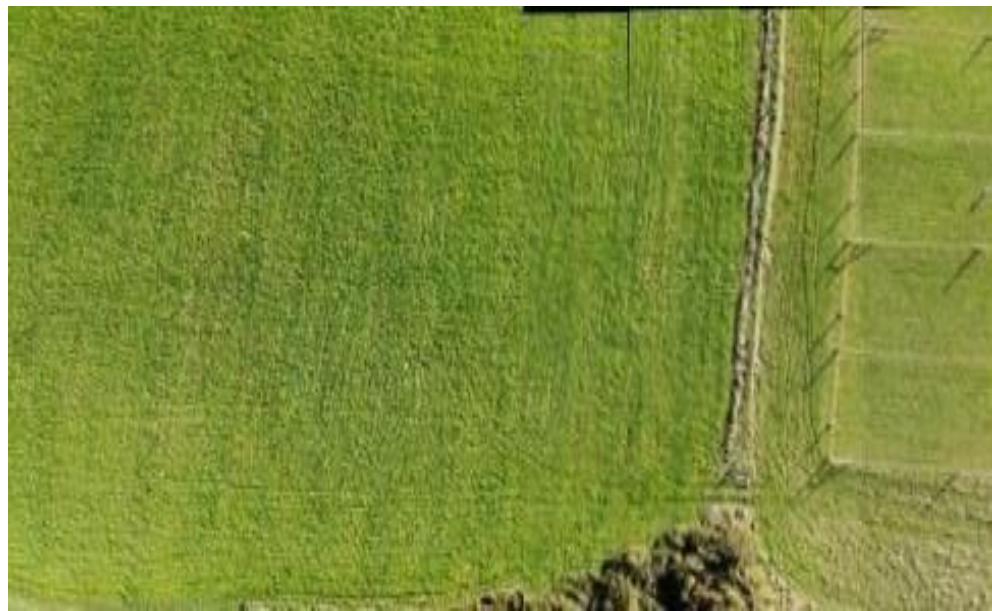
Anexo C: Tabla de métricas e imágenes panorámicas generadas con algoritmo con técnica SIFT versión 3

Se realizaron las pruebas con un algoritmo con técnicas de stitching SIFT (SIFT v3) la diferencia de este algoritmo a los de los Anexos A y B es que no usaba librerías externas, solamente las administradas por openCV.

Esta versión de algoritmo podía recibir secuencia varias imágenes, pero tenía complicaciones con algunas, ya que se podía ver con claridad ciertas líneas de unión entre una y otra imagen, además dejaba las zonas negras al finalizar la creación de la imagen panorámica, ocasionando que cuando las imágenes tenían una leve inclinación la panorámica saliente se podría visualizar algo “quebrada”, teniendo líneas negras entre las imágenes. Y, por último, el algoritmo se detenía cuando las imágenes eran de mucha cantidad de píxeles, es por esto que esas métricas no se obtuvieron para este caso.

Tamaño Primer Dataset	Tiempo (segundos)	Uso Memoria	Uso Disco
1014x760 px	27,625	81,1%	42,6%
253x190 px	0,478	81,3%	43%





Tamaño Segundo Dataset	Tiempo (segundos)	Uso Memoria	Uso Disco
1014x760 px	2,991	81,4%	42,6%
253x190 px	0,45	82,9%	43%





Tamaño Tercer Dataset	Tiempo (segundos)	Uso Memoria	Uso Disco
1014x760 px	2,991	82,9%	42,5%
253x190 px	0,603	86,4%	43,6%



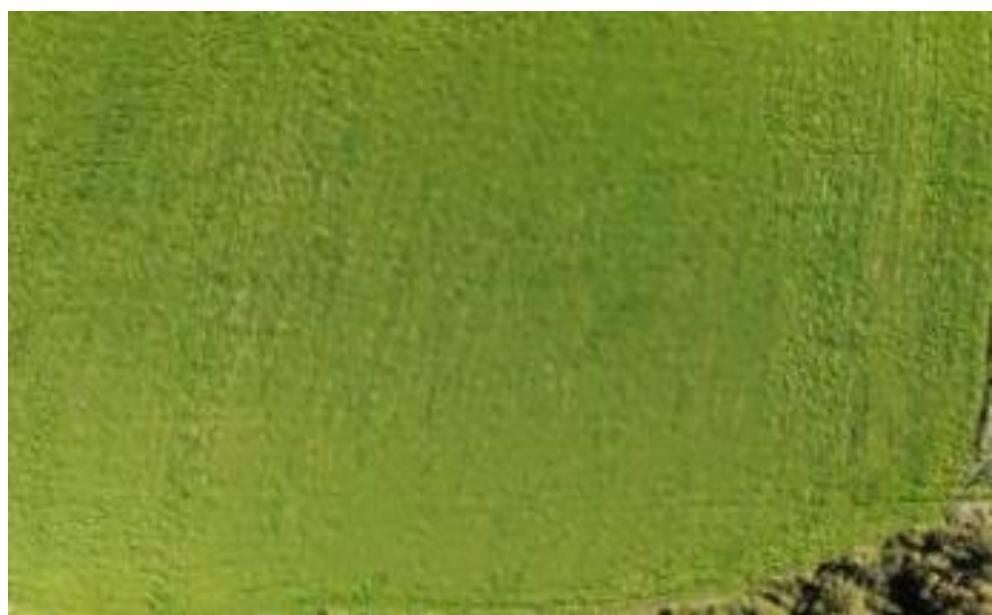
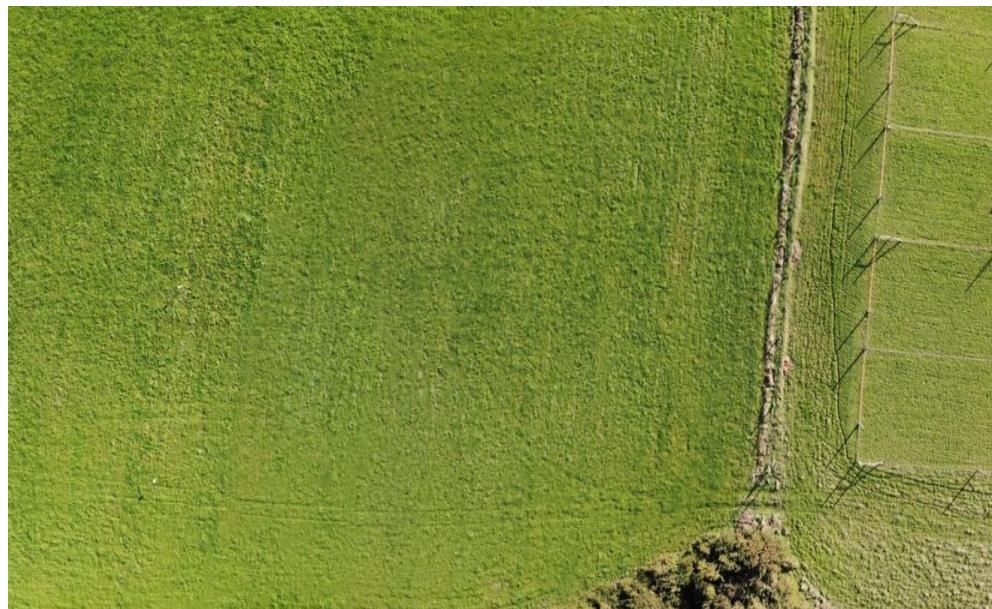


Anexo D: Tabla de métricas e imágenes panorámicas generadas con algoritmo ORB

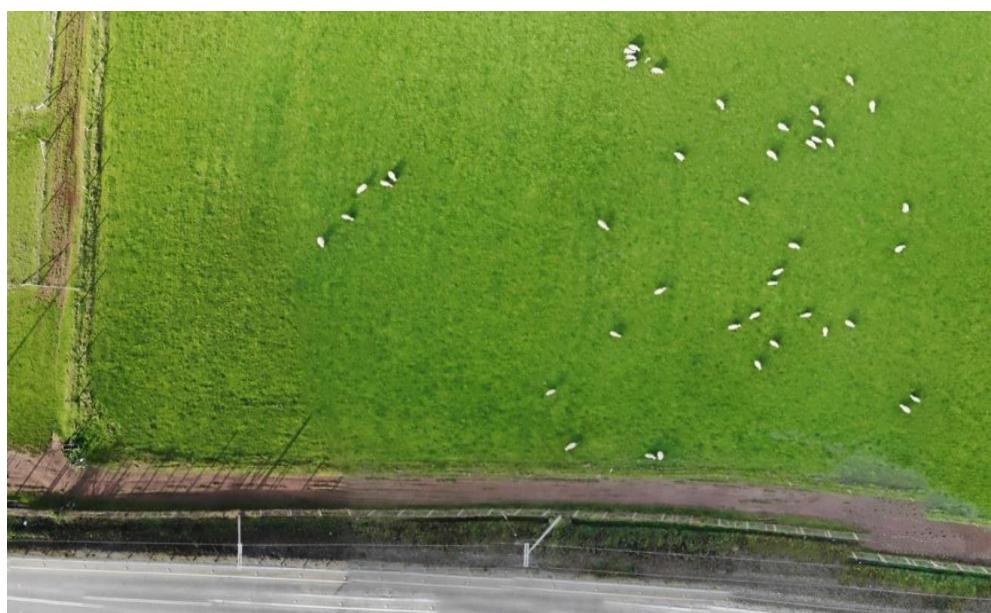
Se realizaron las pruebas con un algoritmo con técnicas de stitching ORB. Aunque su forma de trabajar es un poco distinta a SIFT tiende a tener resultados similares en la creación de imágenes panorámicas, algo que se puede notar que es que tiene cifras mayores a las obtenidas con SIFT, por ejemplo, el tiempo efectuado para imágenes de 4056x3040px dura 46,120 segundos mientras que en SIFT (Anexos A y B) la mayor cifra fue de 29 segundos aproximados.

Tamaño Primer Dataset	Tiempo (segundos)	Uso Memoria	Uso Disco
4056x3040 px	46,120	74,8%	69,6%
1014x760 px	7,796	73,9%	69,4%
253x190 px	1,578	73,3%	69,4%



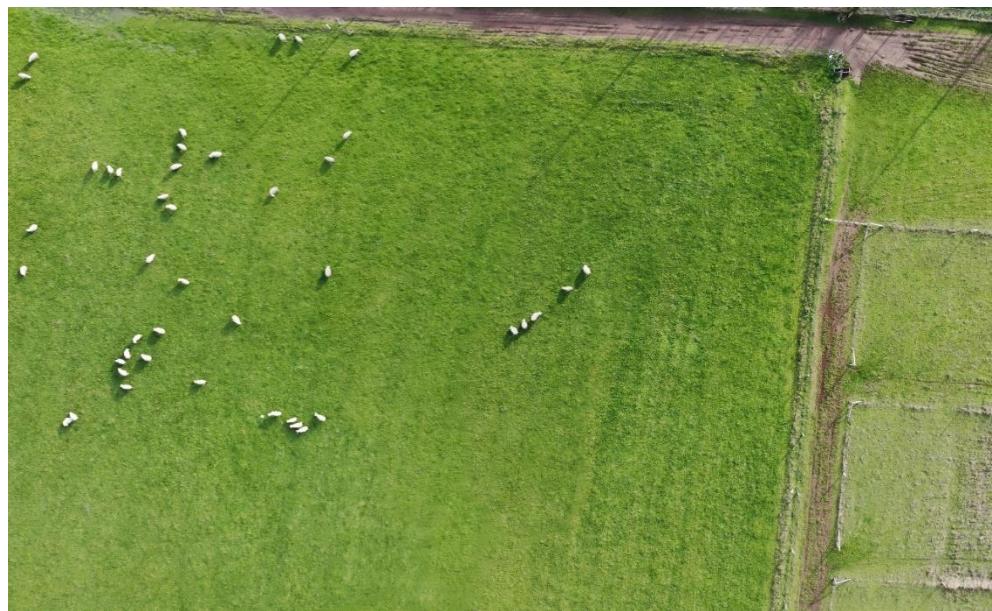


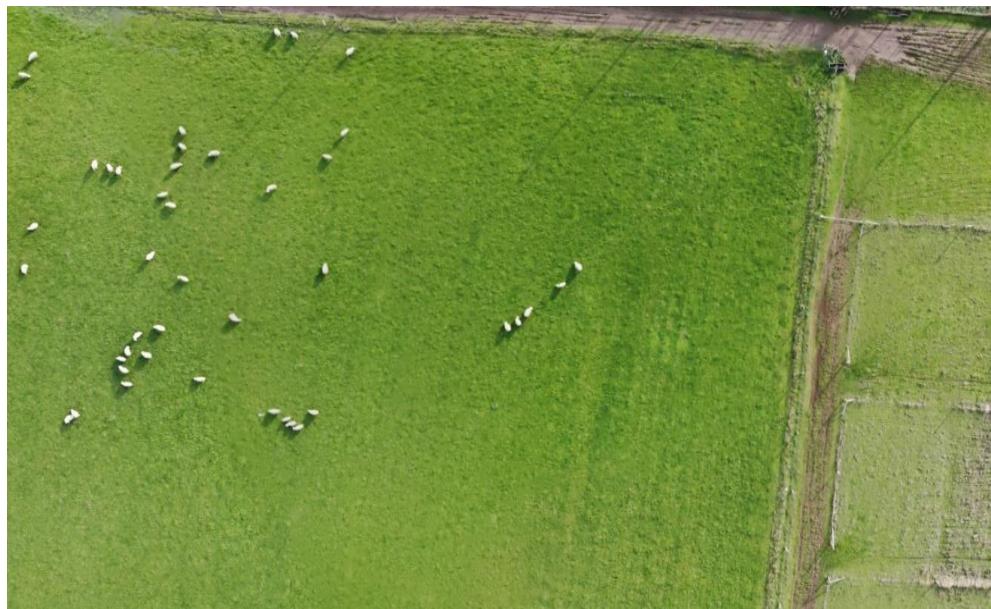
Tamaño Segundo Dataset	Tiempo (segundos)	Uso Memoria	Uso Disco
4056x3040 px	16,006	73,2%	69,4%
1014x760 px	3,805	73,3%	69,4%
253x190 px	6,031	73,1%	69,4%





Tamaño Tercer Dataset	Tiempo (segundos)	Uso Memoria	Uso Disco
4056x3040 px	18,949	73%	69,4%
1014x760 px	8,550	73%	69,5%
253x190 px	5,187	73,3%	69,4%





Anexo E: Tabla de actividades del proyecto de título y sus descripciones

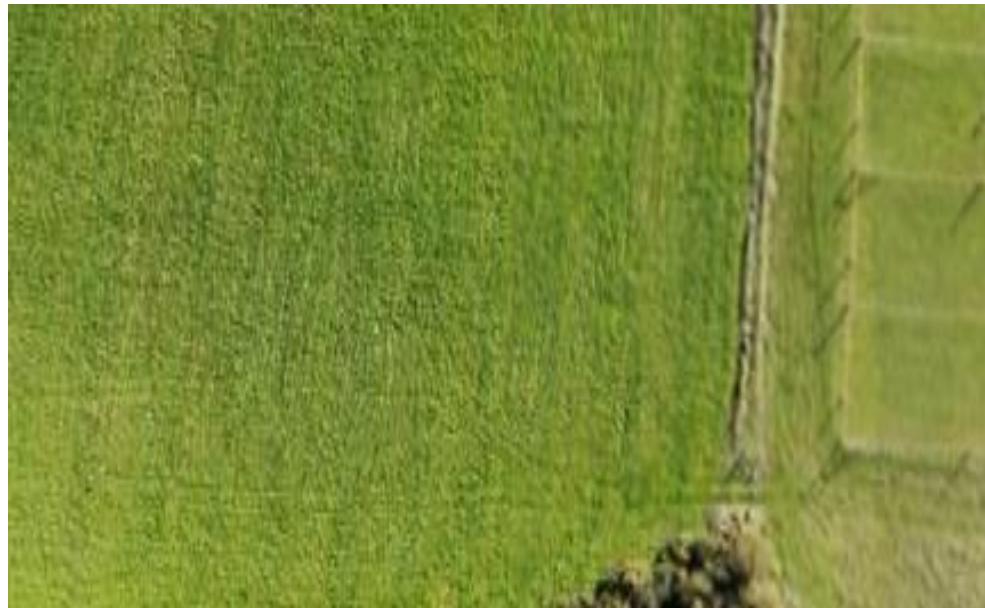
Nombre de la Actividad	Descripción de la actividad
Analizar soluciones y/o herramientas actuales	Investigación sobre las soluciones / herramientas ocupadas para hacer <i>stitching</i> en imágenes.
Analizar distintos tipos de algoritmos de stitching	Investigación de distintas familias de algoritmos de <i>stitching</i>
Analizar su comportamiento y funcionamiento	Investigación de las características de las soluciones / herramientas encontradas y cómo operan
Encontrar y evaluar posibles fallas de los algoritmos estudiados con sus data set distinto	Reconocimiento y valoración de errores de cada algoritmo de <i>stitching</i> investigado con un data set distinto
Especificar Requerimientos de algoritmo adaptativo	Obtener los requisitos del algoritmo adaptativo
Diseñar y desarrollar algoritmo adaptativo	Determinar los componentes significativos del algoritmo y desarrollar el código de acuerdo a los requerimientos
Testear algoritmo adaptativo	Realización de múltiples pruebas para ver y analizar el comportamiento del algoritmo en distintos escenarios.

Anexo F: Tabla de secciones del algoritmo adaptativo

Sección	Descripción
Obtención de imágenes	El algoritmo deberá obtener la ruta donde se encuentran las imágenes para la creación de la imagen panorámica a través de métodos de stitching, debe ser una secuencia de imágenes.
Manejo de imágenes	Las imágenes deben ser almacenadas en un tipo de variable para su correcto manejo de estas. Lo que facilitara al programa para la unificación de cada una de ellas.
Adaptabilidad del código	El algoritmo debe ser capaz de analizar las imágenes y elegir el método de stitching que mejor estime conveniente a partir de la condicional puesta.
Detección de puntos clave y características	El algoritmo deberá poder identificar puntos relevantes en la imagen.
Emparejar puntos	El algoritmo tras tener como referencia los puntos claves de las imágenes deberá poder encontrar similitudes entre estas para lograr un correcto stitching.
Obtener perspectiva de la distorsión	El algoritmo debe ser capaz de generar una distorsión en la perspectiva de la imagen para eliminar posibles distorsiones ópticas.
Dibujar líneas	El algoritmo debe almacenar una imagen donde se pueda ver claramente líneas de colores que indiquen los puntos de similitudes entre ambas imágenes de las cuales se realizó el stitching.
Recortar bordes imagen panorámica	Tras crear la imagen panorámica esta quedará con bordes negros los cuales el algoritmo debe ser capaz de recortar de la imagen final todo recuadro negro que impida dejar una imagen rectangular de fácil comprensión.
Obtención de información del equipo	El algoritmo debe tener líneas de código que faciliten el cálculo de las métricas ocupadas para evaluar la eficiencia del algoritmo mientras se realice la creación de la imagen panorámica. Estos datos en lo posible deben ser almacenados en algún documento para efectos de pruebas.
Obtener imagen panorámica	Finalmente, el algoritmo debe entregar como resultado una imagen panorámica en la cual se vea solo características propias de las imágenes y en su defecto sin dificultad de interpretación de esta misma.

Anexo G: Tabla de métricas e imágenes panorámicas con algoritmo adaptativo

Algoritmo	Tamaño Primer Dataset	Tiempo (segundos)	Uso Memoria	Uso Disco
Algoritmo Adaptativo	1014x760 px	6,752	75,6%	58,5%
	253x190 px	0,201	79,2%	58,5%



Algoritmo	Tamaño Segundo Dataset	Tiempo (segundos)	Uso Memoria	Uso Disco
Algoritmo Adaptativo	1014x760 px	3,814	79,8%	58,5%
	253x190 px	0,185	79,4%	58,5%



Anexo H: Imagen PCA (Principal Component Analysis)

“El análisis de componentes principales (ACP) resulta una técnica muy útil cuando lo que se pretende eliminar la correlación entre los niveles digitales de la imagen digital. Mediante el ACP, se destacan las diferencias entre las bandas de la imagen. Cuando las imágenes de entrada son bandas de radiación reflejada en la región visible del espectro, el empleo del ACP resulta especialmente útil, ya que las bandas de entrada se encuentran muy correlacionadas y es necesario el empleo de algoritmos que reduzcan los datos redundantes” (Carrión-Ruiz & Lerma García, 2017).

Acá se pueden detectar como al ingresar una de las imágenes de las Torres del Paine el algoritmo del PCA detecta los objetos principales de la imagen.

