UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

DETECCIÓN DE PATRONES REPETITIVOS EN OBJETOS ARQUEOLÓGICOS TEXTURIZADOS

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERA CIVIL EN COMPUTACIÓN

Genesis Alexandra Moraga Azócar

PROFESOR GUÍA:
Iván Sipiran Mendoza

MIEMBROS DE LA COMISIÓN:
Patricio Inostroza Fajardin
Eduardo Graells Garrido

SANTIAGO DE CHILE
2023

DETECCIÓN DE PATRONES REPETITIVOS EN OBJETOS ARQUEOLÓGICOS TEXTURIZADOS

En la actualidad, se han logrado avances significativos en el estudio de patrones en imágenes mediante el uso de diversos algoritmos, tanto clásicos como más avanzados basados en el aprendizaje profundo (*deep learning*). No obstante, hasta ahora, estos estudios no han abordado específicamente el campo de la arqueología digital, una disciplina que se dedica a la digitalización de diferentes artefactos arqueológicos, con el objetivo de que personas de todo el mundo puedan acceder a ellos, ya sea con propósitos científicos o no. Además, esta área de investigación permite una mejor conservación de dichos objetos, los cuales son muy susceptibles a desgaste por la manipulación humana, así como a riesgos de fracturas o fisuras. Ante esta problemática, contar con modelos que sean capaces de identificar patrones en objetos arqueológicos representa un gran avance en esta disciplina. Por lo tanto, en el marco de esta investigación, se llevó a cabo una comparativa de distintos algoritmos con la capacidad de realizar dicha tarea.

En base a lo expuesto, se trabajó con un conjunto de datos que consta de 82 imágenes de artefactos arqueológicos, acompañadas de las imágenes correspondientes a sus patrones, así como un archivo JSON que contiene las coordenadas donde se encuentran dichos patrones en la imagen principal. Se llevaron a cabo un total de 16 experimentos utilizando 4 algoritmos distintos: Template Matching, SIFT, ORB y LoFTR. Para cada algoritmo, se aplicaron 4 variaciones (sin correcciones, filtrado, redimensionado y filtrado con redimensionado), siendo cada variación una modificación específica del algoritmo original.

Con el fin de realizar una comparativa, se utilizó la métrica MAP (Mean Average Precision). Los resultados obtenidos revelaron que el modelo más efectivo en todas las variaciones fue LoFTR, obteniendo los dos mejores resultados de MAP, con un 51 % y un 45 %, respectivamente. En segundo lugar, se ubicó SIFT con un resultado de MAP del 32 %. Es probable que el rendimiento superior de LoFTR se deba a su uso de redes neuronales complejas. No obstante, es importante destacar que los competidores no se quedaron rezagados, teniendo en cuenta que son algoritmos clásicos.

Finalmente, los resultados de esta investigación logran alcanzar el objetivo planteado, que consistía en brindar un aporte a la comunidad científica en el campo de la arqueología digital, específicamente en la selección adecuada de herramientas para detectar patrones en objetos arqueológicos.

Dedicado a todos los que me han acompañado en este camino,
es especial a mi familia, novio y amigos.

Tabla de Contenido

Índice de Tablas

Índice de Ilustraciones

Capítulo 1

Introducción

1.1. Contexto y problema

En las últimas décadas, han ocurrido avances significativos en el campo de la visión computacional y el aprendizaje automático. En particular, el enfoque en el aprendizaje profundo ha llevado a mejoras notables en el *hardware*, como las GPU, lo que ha impulsado y potenciado estas tecnologías. Asimismo, se ha producido un gran progreso en los algoritmos utilizados en visión computacional, como los mencionados en [1]. Todos estos avances fomentan el desarrollo de investigaciones, como la presente, que permiten estudiar el funcionamiento de estas tecnologías en imágenes de objetos diversos, con características específicas como resolución, forma e iluminación.

Estos avances tecnológicos también han facilitado el almacenamiento y difusión de información en diversas áreas, lo que ha impulsado el crecimiento de la arqueología digital. Esta disciplina se basa en el análisis de objetos históricos utilizando herramientas tecnológicas. Gracias a la digitalización y al impacto de la globalización, cada vez más personas tienen acceso a estos objetos arqueológicos. Además, la digitalización contribuye a reducir los riesgos asociados con la manipulación física, como posibles daños, fracturas y deterioro, al evitar la necesidad de manipulación manual directa.

La identificación de patrones en imágenes ha sido objeto de estudio por parte de diversos investigadores en los últimos tiempos, con el propósito de mejorar las técnicas y herramientas utilizadas en este campo. Sin embargo, hasta el momento, estos esfuerzos no se han centrado en mayor medida en el ámbito de la arqueología digital. En este sentido, en [2] se plantea un enfoque particular en el análisis tridimensional de objetos cerámicos para la detección de patrones repetitivos, donde el objetivo es aprovechar los beneficios que la digitalización puede brindar a la arqueología, como se mencionó anteriormente.

En el contexto de la arqueología digital, surge un problema relacionado con las herramientas específicas utilizadas en este campo. Aunque existen herramientas disponibles, no se ha establecido con certeza cuáles son las más adecuadas para abordar los desafíos asociados a la detección de patrones. Es importante destacar que esta tarea implica enfrentarse a dificultades como variaciones en el tamaño, así como transformaciones de traslación, rotación y simetría de los patrones. Además, se debe tener en cuenta que los objetos arqueológicos fotografiados pueden presentar desgastes o fisuras debido a su antigüedad, y las imágenes cap-

turadas pueden tener baja claridad. Estos factores agregan un nivel de complejidad adicional al proceso de detección de patrones en la arqueología digital.

Con base en el problema planteado, resulta de suma importancia el avance en esta disciplina, no solo por el progreso que se logrará en el campo, al poder identificar con mayor precisión los patrones repetitivos que permitirán documentar y analizar la información contenida en los objetos arqueológicos, sino también por el relevante paso que se dará en apoyo a los propios profesionales dedicados a esta materia, es decir, los arqueólogos. Este último aspecto se debe principalmente a que la utilización de herramientas digitales contribuye de manera significativa en la conservación y registro de piezas arqueológicas, al reducir la manipulación de estos objetos que, debido a sus estructuras y antigüedad, resultan sumamente frágiles. Asimismo, el análisis de patrones tiene el potencial de proponer nuevas hipótesis relacionadas con el cómo y el porqué de la creación de estos objetos, lo que brindaría a los profesionales del área una perspectiva renovada. En definitiva, el avance en esta disciplina no solo favorecerá el campo de estudio, sino que también ofrecerá un valioso respaldo a los arqueólogos al mejorar la conservación, el registro y la comprensión de estos objetos arqueológicos.

1.2. Objetivos

1.2.1. Objetivo general

Se busca comparar qué forma es más efectiva, en términos de exactitud en la localización, para detectar patrones repetitivos en imágenes de objetos arqueológicos. En particular, se comparará si se obtienen mejores resultados con el uso de métodos clásicos de búsqueda de patrones visuales o con modelos de *deep learning*.

1.2.2. Objetivos específicos

1. Definir él o las técnicas en que se basará la implementación con descriptores clásicos a estudiar, las que deben ser eficientes en la detección de patrones en imágenes similares a las presentes en el conjunto de datos con el que se trabajará, es decir, que sean adecuados para el problema.

2. Implementar el o los algoritmos que usen descriptores clásicos de los mencionados en el punto anterior, basándose preliminarmente en [3], [4], [5], [6] y [7].

3. Definir claramente él o los modelos de *deep learning* que se usarán en la búsqueda de patrones, que de igual forma deben ser adecuados para esta tarea.

4. Realizar experimentos sobre el conjunto de datos usando las técnicas mencionadas en todos los puntos anteriores.

5. Comparar los resultados obtenidos en los experimentos realizados.

1.3. Evaluación

Para evaluar esta investigación se escogerán dos dimensiones, que son a nivel de algoritmo y a nivel de aporte a otras investigaciones. Respecto a la primera, será relevante poder llegar a una conclusión válida y coherente con los resultados. Por lo tanto, se propone utilizar métricas para poder realizar comparaciones entre modelos y así lograr obtener información valiosa para ser discutida. Específicamente, se empleará la métrica MAP (Mean Average Precision). En cuanto a la segunda dimensión, se propone lograr que el conocimiento generado, es decir, los resultados y conclusiones obtenidas, puedan llegar a estar a libre disposición para ser utilizado en otras investigaciones.

1.4. Solución y resultados

Con el fin de abordar el problema planteado y alcanzar los objetivos establecidos, se consideraron diversas opciones para la detección de patrones en las imágenes de vasijas, hasta llegar a la selección de los algoritmos Template Matching, ORB, SIFT y LoFTR. Esta elección se fundamenta en diferentes razones para cada uno de ellos. Template Matching fue seleccionado debido a su simplicidad al comparar píxeles de forma directa. Por su parte, ORB y SIFT fueron elegidos por sus resultados prometedores documentados en la literatura especializada. En cuanto a LoFTR, se optó por este modelo por ser una opción actual que emplea redes neuronales y, a su vez, ofrece facilidad de uso al estar completamente implementado en una biblioteca de Python llamada Kornia.

Posteriormente, se llevaron a cabo pruebas preliminares con el objetivo de comprender de manera más detallada el funcionamiento de los distintos métodos, así como observar visualmente los resultados que proporcionaban en casos particulares. Esto permitió identificar las dificultades que enfrentaban y los casos en los que parecían funcionar mejor. Todo este análisis previo sentó las bases para la solución final, la cual consta de los siguientes pasos: preprocesamiento de los datos de las imágenes, ejecución de los algoritmos con variaciones para fines comparativos, diseño y ejecución de diferentes experimentos, y finalmente, la implementación de una métrica que posibilitara una evaluación cuantitativa de la solución obtenida.

Gracias a todo lo mencionado anteriormente, fue posible obtener resultados comparables entre sí, lo que permitió determinar cuáles métodos se destacan en diferentes condiciones. Específicamente, se pudo observar cuantitativamente, mediante el uso de la métrica seleccionada, que el método LoFTR proporciona excelentes resultados. En particular, su promedio máximo de MAP al aplicarlo a todas las imágenes del conjunto de datos fue de aproximadamente un 51 %. Sin embargo, es importante destacar que los resultados de las variaciones de estos algoritmos presentaron un aspecto interesante: al realizar pequeñas modificaciones, los modelos clásicos que detectan características locales, como SIFT, lograron mejorar sus resultados, aunque en ninguna variación superaron a LoFTR.

Capítulo 2

Estado del arte

En el contexto de esta investigación, se ha observado que otros investigadores han dedicado tiempo a explorar diferentes técnicas para abordar la detección de patrones en diversos tipos de imágenes. Por ejemplo, en el trabajo de [5] y [6], se ha enfocado en encontrar patrones en imágenes de grandes estructuras urbanas utilizando enfoques clásicos. Han logrado identificar repeticiones de ventanas, puertas, diseños en fachadas y otros elementos. Por otro lado, en [5], se utiliza una base de datos de patrones para detectar patrones en imágenes considerando posibles transformaciones como traslaciones. Además, en [6], se emplea un método basado en características para extraer hipótesis de repetición y simetría de imágenes rectificadas, obteniendo límites de regiones de repetición que se utilizan para maximizar las simetrías.

En una línea similar, los autores de [3] han abordado la detección de patrones repetitivos mediante la identificación de simetrías de traslación. Para lograr esto, el algoritmo requiere de patrones iniciales que serán buscados de manera iterativa y procesados posteriormente. Este enfoque demuestra ser efectivo incluso en la detección de patrones curvados y distorsionados. Por ejemplo, es capaz de detectar pequeñas figuras repetitivas en pisos de cerámica, ventanas de edificios, mallas de cercas o tejados de casas. Estos resultados destacan la capacidad del algoritmo para identificar patrones en diferentes contextos y formas geométricas.

En otro enfoque, el autor de [8] propone el método SIFT (Scale-Invariant Feature Transform), el cual se dedica a detectar y describir características locales de una imagen. Este método es conocido por abordar los desafíos relacionados con la escala y rotación, lo que le permite encontrar puntos clave significativos en la imagen. SIFT es particularmente efectivo en la búsqueda de coincidencias de imágenes y la detección de objetos. Además, debido a su relevancia y utilidad, ha sido incorporado en la popular librería de visión computacional OpenCV, poniéndolo al alcance de la comunidad científica. La implementación de SIFT consta de cuatro etapas: detección de posibles puntos clave en diferentes escalas y ubicaciones mediante una función de diferencias gaussianas, localización precisa de estos puntos clave, asignación de orientaciones a los puntos clave mediante el cálculo de histogramas de gradientes orientados, y finalmente, la construcción de descriptores que contienen información sobre los puntos clave.

Otro algoritmo relevante en OpenCV es el descriptor ORB (Oriented FAST and Rotated BRIEF), presentado en [4]. ORB es similar a SIFT en el sentido de que también detecta y describe características locales en una imagen, manteniendo la invariancia a la escala y

rotación. Sin embargo, ORB se destaca por ser más rápido y eficiente, ya que es una mejora del algoritmo FAST (Features from Accelerated Segment Test) y el descriptor BRIEF (Binary Robust Independent Elementary Features). Combina la velocidad de FAST con la robustez de BRIEF, lo que lo convierte en una opción atractiva en aplicaciones que requieren procesamiento en tiempo real. Aunque SIFT ha demostrado mejores resultados en diversos escenarios [7], ORB no se queda rezagado y ofrece una alternativa sólida y eficiente.

Otro algoritmo disponible en la misma librería es Template Matching, presentado en [9]. Este método realiza una búsqueda de una imagen predefinida, llamada plantilla (*template*), sobre otra imagen y devuelve las ubicaciones donde se encuentran coincidencias. Para lograr esto, desplaza la plantilla píxel a píxel sobre toda la imagen de entrada, calculando una medida de similitud entre la plantilla y la sección correspondiente de la imagen en cada posición. La medida de similitud se puede calcular utilizando seis métodos diferentes, que se describen en la tabla 2.1. Template Matching es un método de nivel superior en visión computacional, lo que significa que su implementación no es muy compleja. Debido a esto, tiene limitaciones al ser utilizado en imágenes muy grandes debido a la cantidad de comparaciones que se deben realizar. Además, no aborda directamente el problema de la escala y la rotación, ya que compara la plantilla directamente con la imagen sin considerar transformaciones geométricas.

Tabla 2.1: Métodos que se pueden utilizar con Template Matching

Nombre	Descripción
TM_SQDIFF	Calcula la diferencia de cuadrados entre las imágenes, por lo que a mayor cercanía a 0 es mejor la coincidencia.
TM_CCORR	Usa la correlación entre las imágenes. A mayor valor mejor coincidencias.
TM_CCOEFF	Usa la correlación, pero a los valores de ambas imágenes le resta el promedio de su imagen respectiva.
TM_SQDIFF_NORMED	Usa la diferencia de cuadrados normalizada.
TM_CCORR_NORMED	Usa correlación, pero normalizada.
TM_CCOEFF_NORMED	Similar a TM_CCOEFF, pero normalizado.

En el ámbito específico de la arqueología, en el estudio realizado en [10], se aborda el reconocimiento de patrones y palabras en documentos históricos. Para lograr esto, se emplean vectores de descriptores agregados localmente (VLAD) y vectores de Fisher (FV). Estos métodos se basan en la utilización de parches extraídos de la imagen mediante un proceso de muestreo. Luego, para cada parche, se utiliza el algoritmo SIFT. Si bien los autores logran detectar alrededor del 60 % de los patrones, consideran que existe margen de mejora en este aspecto.

En contraste, en el estudio realizado por los autores de [11], se emplea un enfoque diferente utilizando redes convolucionales pre-entrenadas para detectar patrones en imágenes

de documentos históricos. Específicamente, utilizan RetinaNet como modelo de detección para realizar las consultas. Si bien obtienen buenos resultados en la ubicación de patrones, se observa que surgen dificultades al utilizar un gran número de páginas del documento en la consulta. Es notable destacar que estas dos investigaciones mencionadas se encuentran estrechamente relacionadas con la arqueología digital, al abordar el análisis de documentos históricos como parte de su estudio.

En el estudio presentado en [12], se utilizan redes neuronales residuales (ResNet) que, a pesar de ser muy profundas, se ha demostrado que son más fáciles de entrenar en comparación con otras arquitecturas. Los autores resaltan la importancia de la profundidad en el reconocimiento visual y exploran redes con hasta 152 capas, las cuales logran obtener buenos resultados en esta área. Es relevante destacar que ResNet ya está implementada en el *framework* PyTorch, ofreciendo cinco versiones con diferentes cantidades de capas: 8, 34, 50, 101 y 152. Además, PyTorch proporciona la opción de utilizar modelos pre-entrenados en el conjunto de datos de ImageNet, lo cual facilita su aplicación en diversas tareas.

Por otra parte, es importante mencionar la existencia de una biblioteca reciente en el campo de visión artificial y el aprendizaje profundo llamada Kornia [11]. Esta biblioteca, desarrollada en PyTorch, ofrece diversas funcionalidades como filtrado de imágenes, detección de bordes, extracción de descriptores de características, entre otras aplicaciones. Una característica destacada de Kornia es su soporte para operaciones en GPU, lo que permite un procesamiento más rápido.

Un aspecto interesante de Kornia es la inclusión de la arquitectura de red neuronal llamada LoFTR (Local Feature Transformer) [13], basada en transformers y, en particular, en la autoatención local. Esta arquitectura permite la detección y el emparejamiento preciso y eficiente de características locales en imágenes. El método LoFTR consta de cuatro componentes principales: una red neuronal convolucional para extraer características, un módulo LoFTR que utiliza capas de atención local y atención cruzada para procesar las características, un módulo Coarse-to-Fine que realiza el emparejamiento de características y, finalmente, un módulo de correspondencia. Además, es posible utilizar el modelo LoFTR pre-entrenado con imágenes de entornos exteriores o interiores gracias a los conjuntos de datos MegaDepth y ScanNet, respectivamente. Estas opciones permiten adaptar el modelo a diferentes escenarios y mejorar su rendimiento en tareas específicas.

Cada uno de los investigadores mencionados previamente ha realizado avances significativos desde diversas perspectivas, pero no se han especializado en la arqueología digital con figuras 3D. Si bien los autores de [11] y [10] trabajaron en esta área, no lo amplían al reconocimiento en objetos 3D. Por lo tanto, aún no se cuenta con información sobre las herramientas más efectivas ni los mejores métodos para abordar la detección de patrones en objetos arqueológicos.

Por último, es importante destacar el trabajo fundamental realizado en [2], el cual proporciona una base sólida para la presente investigación al publicar un conjunto de datos de imágenes compuesto por 82 ejemplos de vasijas arqueológicas. Estas imágenes representan una proyección plana de los objetos 3D y contienen imágenes de los patrones presentes en ellos, junto con etiquetas que proporcionan información sobre los patrones repetitivos pre-

sentes. En la figura 2.2, se muestra un ejemplo de la representación plana de una vasija, y en la figura 2.1, se observa la imagen correspondiente al patrón detectado en dicha vasija. Estas imágenes fueron obtenidas del conjunto de datos compartido en [2].

Figura 2.1: Imagen del patrón de una vasija contenida en el conjunto de datos creado en [2]

Figura 2.2: Imagen de una vasija aplanada contenida en el conjunto de datos mostrados en [2]

Además, en [2] se aborda otro tema relevante, donde se presentan y explican posibles métricas que se ajustan a la tarea de detección de patrones. En particular, se detallan las implementaciones de Mean Average Precision (MAP) y Average Nearest-Neighbor Distance (ANND). La métrica MAP se utiliza cuando las detecciones proporcionan un área como resultado, mientras que ANND se emplea cuando el resultado es un punto en el espacio. Estas métricas proporcionan una manera cuantitativa de evaluar la calidad de los resultados obtenidos en la detección de patrones.

Capítulo 3

Problema

La arqueología digital es una disciplina que utiliza tecnologías avanzadas de procesamiento de imágenes, análisis de datos y modelado 3D para el estudio de objetos arqueológicos. Esta disciplina ha surgido como una forma complementaria y mejorada de las técnicas tradicionales de la arqueología, convirtiéndose en una herramienta importante para la investigación y conservación de los objetos arqueológicos. Por ejemplo, el modelado 3D simple de estos objetos permite su conservación al evitar la manipulación constante, al mismo tiempo que contribuye a la divulgación y acceso a los mismos para arqueólogos e investigadores de diferentes partes del mundo.

Sin embargo, uno de los mayores desafíos que enfrenta la arqueología digital es la selección de las herramientas adecuadas para abordar los desafíos de la detección de patrones en imágenes arqueológicas. Los objetos arqueológicos son muy variados en términos de tamaño, forma, textura y material, lo que hace que su análisis sea complejo. Además, los objetos pueden tener desgaste o fisuras debido a su antigüedad y las imágenes pueden tener problemas de resolución y calidad, lo que dificulta aún más el análisis.

Para detectar patrones en estas imágenes arqueológicas, se requiere el uso de herramientas de procesamiento de imágenes que sean capaces de manejar la variabilidad de los objetos arqueológicos y proporcionar resultados precisos y útiles para los arqueólogos. Esto implica la necesidad de técnicas avanzadas de análisis de imágenes, como la detección de bordes, la segmentación y el reconocimiento de formas.

No obstante, las herramientas de procesamiento de imágenes no son perfectas y pueden presentar limitaciones en su capacidad para detectar patrones complejos o para lidiar con la variabilidad de los objetos arqueológicos. Por lo tanto, puede ser necesario combinar varias herramientas y técnicas para lograr los mejores resultados en un proyecto de arqueología digital. Además, es necesario saber qué limitaciones presentan y en qué se destacan para escoger los mejores algoritmos según la necesidad requerida.

En la figura 3.1, se observa una vasija arqueológica, la que ejemplifica los obstáculos que hacen más complejo el análisis de patrones: posee fisuras, desgaste y falta de pequeños fragmentos. Además, cuenta con patrones que se encuentran en diferentes tamaños en la imagen original. A pesar de esto, se tiene la tarea de lograr identificar los patrones contenidos en la imagen aproximándose lo más posible a las anotaciones de estos.

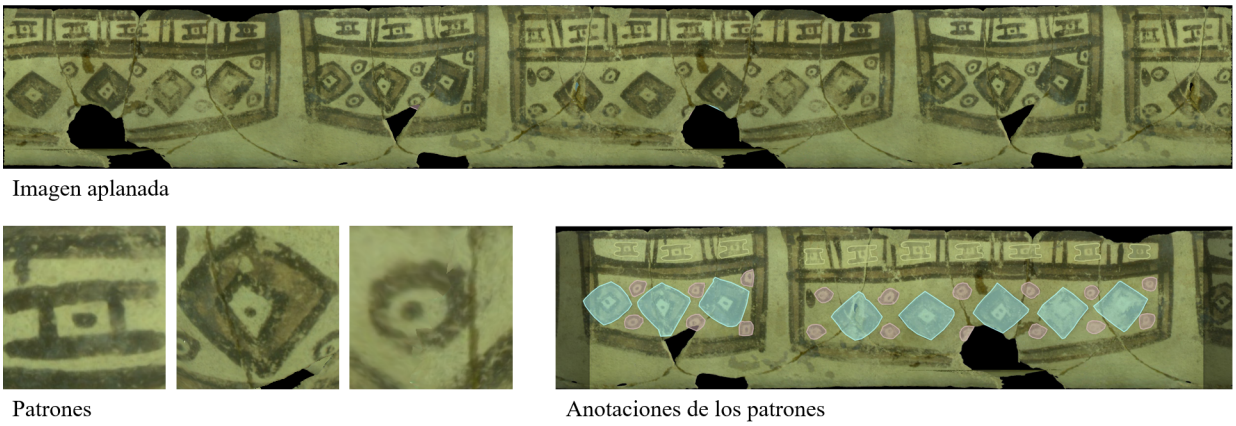

Figura 3.1: Imagen 202 del conjunto de datos en [2], con sus respectivos patrones y anotaciones.

Debido a todo lo anterior, la selección de las herramientas adecuadas para abordar los desafíos de la detección de patrones en imágenes arqueológicas es un problema importante en el campo de la arqueología digital. Se requiere de técnicas avanzadas de análisis de imágenes y modelado 3D para manejar la variabilidad de los objetos arqueológicos y proporcionar resultados precisos y útiles para los arqueólogos. A pesar de las limitaciones de las herramientas de procesamiento de imágenes, la arqueología digital sigue siendo una disciplina en evolución que promete avances significativos en el estudio y la conservación del patrimonio cultural.

Capítulo 4

Propuesta de solución

Para llevar a cabo la tarea de comparar la efectividad de algoritmos clásicos y modelos de aprendizaje profundo en la detección de patrones repetidos en una figura, es necesario realizar pruebas preliminares y evaluar si estos algoritmos pueden adaptarse al problema de detección y proporcionar información precisa sobre la ubicación de los patrones detectados en una imagen.

Con respecto a la primera área, el uso de descriptores clásicos, hay una gran variedad de métodos documentados para encontrar patrones en imágenes, por lo que en un principio es necesario revisar con más profundidad cuál se ajusta mejor el problema, pero teniendo como base los mencionados en [3], [4], [5], [6] y [7] que detectan patrones repetitivos en diferentes casos como edificios, tapices, estructuras arquitectónicas, etc.

En relación con el segundo tema, el aprendizaje profundo, existen dos alternativas principales a considerar. En primer lugar, se puede optar por utilizar una red neuronal artificial pre-entrenada, como ResNet, la cual ha demostrado obtener buenos resultados en el procesamiento de imágenes. Por otro lado, se encuentra el modelo LoFTR, una red diseñada específicamente para la detección de características locales y el emparejamiento entre dos imágenes.

Para llevar a cabo los experimentos, se utilizará un conjunto de datos que consta de 82 modelos 3D de diversas vasijas antiguas pintadas [2]. Estos modelos incluyen imágenes en las que la superficie de las figuras se encuentra aplanada, las cuales serán utilizadas como entrada en las pruebas de detección de patrones. Además, se cuenta con anotaciones precisas de los patrones de estas figuras, las que servirán como referencia para evaluar la exactitud de los resultados obtenidos.

En el ámbito tecnológico, toda la programación se basará en algoritmos implementados en el lenguaje Python, utilizando diversas bibliotecas según sea necesario. El uso de este lenguaje se debe a su versatilidad en diferentes áreas y a la amplia documentación y contenido disponible en la web. De esta forma, se podrán hacer experimentos, mejorar o especializar los algoritmos en el mismo lenguaje, más en concreto, mientras sea posible, se utilizará Google Colab, para facilitar el uso de distintas librerías de Python que ya tiene instaladas, además de aprovechar la facilidad que entrega en el manejo de archivos, en este caso el conjunto de imágenes, al conectarse con Google Drive. Siguiendo esta línea, en una primera instancia de

prueba se usará la librería OpenCV que ya posee algoritmos para hacer coincidir imágenes y el framework PyTorch ya sea al usar la librería Kornia o directamente para trabajar con redes neuronales como ResNet que cuenta con la opción de usarla pre-entrenada.

Por otra parte, para realizar la evaluación de las diferentes técnicas se tendrá una salida que se comparará con el resultado correcto. Con respecto al tipo de salida que entregarán los experimentos, se cuenta con tres formas:

- Posición absoluta del patrón, un punto en el espacio.
- Posición de la figura completa del patrón, encerrado en un cuerpo estándar como un rectángulo.
- Posición de la figura completa del patrón, con la verdadera forma del cuerpo.

De las formas antes mencionadas, se espera lograr que los algoritmos y modelos implementados entreguen resultados de la primera y segunda forma, puntos en el espacio y áreas como rectángulos en el espacio, dejando abierta la posibilidad de encontrar la forma de la figura.

Siguiendo lo anterior, para comparar la salida con la posición completa del patrón se cuenta con la métrica Mean Average Precision (MAP), que se define como el promedio del Average Precision de cada clase de patrón, que a su vez corresponde a una aproximación del área bajo la curva de la gráfica de las métricas de Recall y Precision. Además, al igual que Recall y Precision, MAP entrega resultados entre 0 y 1, donde cero indica un resultado deficiente y uno es el mejor posible.

De forma matemática, se tienen las siguientes definiciones de las métricas antes mencionadas:

$$R_k = \frac{\sum_{i=1}^{k} TP}{\sum_{i=1}^{n} TP + FN} \tag{4.1}$$

$$P_k = \frac{\sum_{i=1}^{k} TP}{\sum_{i=1}^{k} TP + FP} \tag{4.2}$$

$$AP = \sum_{i=1}^{n} (R_i - R_{i-1}) * P_i \tag{4.3}$$

$$MAP = \frac{\sum_{i=1}^{m} AP_i}{m} \tag{4.4}$$

Donde R, P y AP, indican Recall, Precicion y Average Precisión, respectivamente. Luego, se tiene que TP, FP y FN significan verdaderos positivos, falsos positivos y falsos negativos, en el mismo orden. Además, se cuenta con que m es el número de imágenes, n es el total de clasificaciones como TP, FP o FN, y k indica un índice de posición al ir obteniendo estos resultados.

Para poder llevar a cabo todo lo anterior, se propone la carta gantt de la figura 4.1, que contiene la planificación para implementar lo planteado en un periodo de quince semanas. Además, en esta carta gantt se usan los términos CDA y DLA que significan algoritmos con descriptores clásicos y algoritmos con aprendizaje profundo, respectivamente.

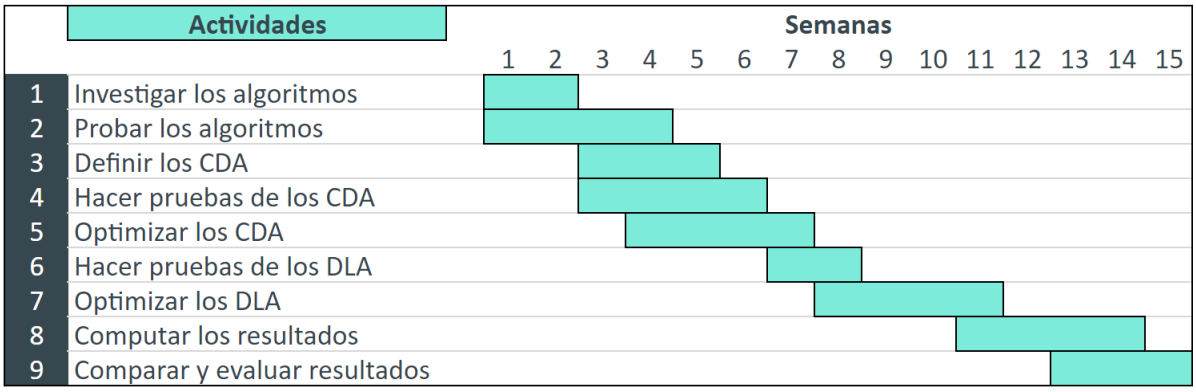

Figura 4.1: Carta gantt.

Capítulo 5

Desarrollo de solución

Se parte con varios algoritmos a evaluar, pero se eligen los siguientes: TemplateMatching, ORB, SIFT y LoFTR, donde solamente el último usa redes neuronales. Cabe destacar que la elección de estos modelos será explicada en la sección de algoritmos. Además, a medida que se avanzó fue necesario reordenar la información del conjunto de datos a usar para facilitar su uso en los diferentes casos. Por otra parte, se crearon funciones generales que ayudaron a obtener resultados con los diferentes algoritmos escogidos.

En la figura 5.1, se muestra un diagrama que ayuda a comprender cómo es el proceso de la solución implementada por el que se llega desde los archivos JSON e imágenes del dataset hasta lograr procesar detecciones en los experimentos que luego son evaluadas. En más detalle se tienen los siguientes pasos:

1. Preprocesamiento: Se leen los archivos JSON con la información de cada imagen para obtener los datos a ocupar y traspasar esta información a un solo archivo JSON general.

2. Cómputo de los experimentos: A partir del archivo JSON general, se ejecutan los diversos experimentos que aplican un algoritmo específico con una configuración determinada a todas las imágenes del conjunto de datos. Cada experimento genera un archivo JSON que contiene los resultados obtenidos.

3. Cálculo de métricas: Se procede a leer los resultados previos y se calculan las métricas correspondientes para su posterior evaluación.

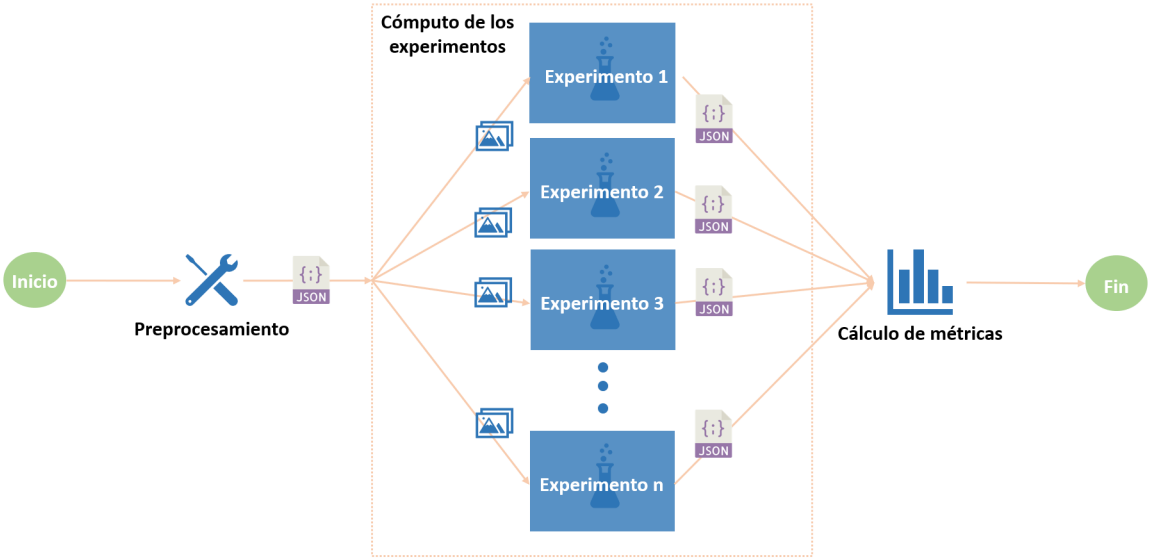

Figura 5.1: Diagrama del flujo de la solución implementada.

En el contexto tecnológico, todas las tareas fueron llevadas a cabo en Google Colab, utilizando Google Drive como gestor de archivos. El desarrollo se realizó utilizando el lenguaje de programación Python 3 y se hicieron uso de los siguientes paquetes y librerías: OpenCV, Kornia, Shapely, NumPy, Pandas, Json y Matplotlib.

A continuación, se explican en más detalle las diferentes partes de la solución.

5.1. Preprocesamiento de los datos

Se disponía de un conjunto de datos que consistía en 82 imágenes de vasijas arqueológicas. Para el propósito de esta investigación, se utilizaron las imágenes aplanadas de las vasijas, así como las imágenes de sus patrones. Además, se contaba con archivos JSON correspondientes a cada vasija, los cuales contenían anotaciones de las posiciones de los patrones detectados en forma de conjuntos de puntos en la imagen. Estos archivos JSON también contenían otros datos relevantes que no fueron utilizados en esta investigación. En la figura 5.2, se puede ver un diagrama resumido de un ejemplo de estos JSON, en el que se muestran principalmente los datos usados, en particular al final del diagrama se tienen los *polygonPts* donde están guardados los puntos de los polígonos que representan las ocurrencias reales de un patrón.

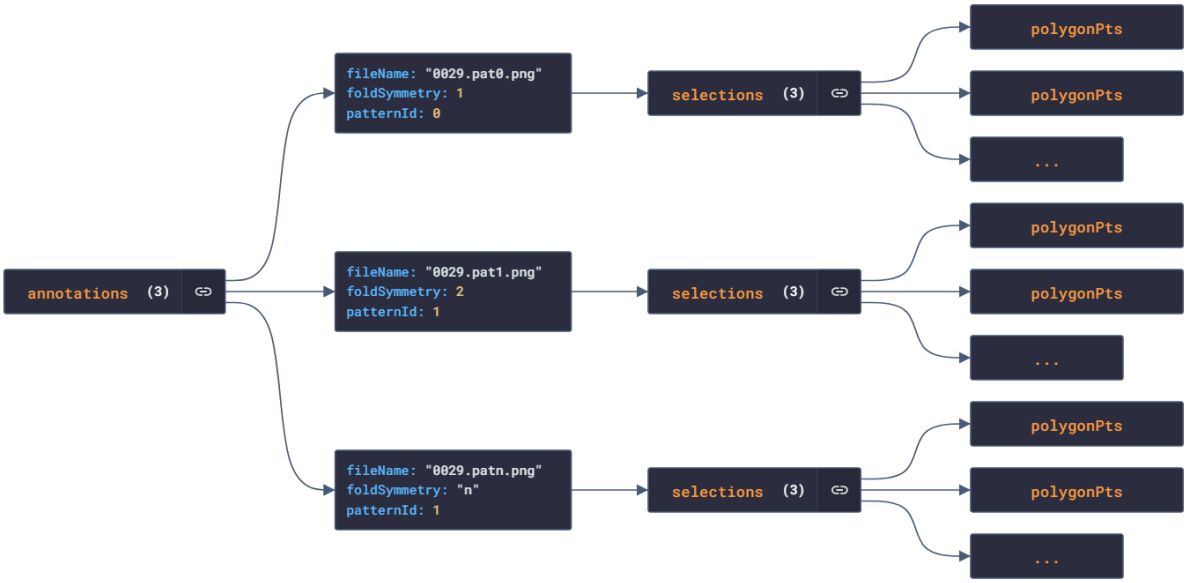

Figura 5.2: Resumen del formato de un JSON que se encuentra en el dataset.

A partir de lo anterior, a medida que se avanzó en los experimentos, fue necesario que estos puntos que identificaban la posición del patrón se transformaran en un rectángulo, el menor posible que contuviera todos los puntos, esto debido a que los algoritmos entregaban un rectángulo como resultado de una detección, no un contorno de forma del objeto. De esta forma se podía tener una comparación más acorde entre lo obtenido y lo esperado.

Además, fue requerido calcular un dato importante correspondiente al promedio de la altura y el ancho de cada patrón. Esto fue necesario debido a que se exploró la posibilidad de modificar el tamaño de la imagen de consulta, es decir, la imagen del patrón que se buscaba, en función del tamaño promedio real de los patrones. Esta consideración surgió a raíz del conocimiento de que el método de coincidencia de plantillas (Template Matching) no es robusto frente a cambios de escala en comparación con otros métodos. Por lo tanto, al escalar la imagen de consulta según el promedio de tamaño de los patrones, se buscó mejorar la capacidad de detección de patrones en imágenes a diferentes escalas.

Por otra parte, con el objetivo de agilizar la lectura de la información, se consolidó la información de los 82 archivos JSON en un solo archivo. Este archivo único contenía todos los datos necesarios, como los promedios mencionados anteriormente, así como los rectángulos que encerraban los patrones detectados y otra información adicional, como la numeración de las imágenes y los patrones. Este enfoque permitió acceder de manera más eficiente a los datos necesarios para ejecutar los algoritmos y calcular las evaluaciones correspondientes. En la figura 5.3, se muestra un diagrama que ilustra la estructura y los contenidos del archivo único creado en este proceso.

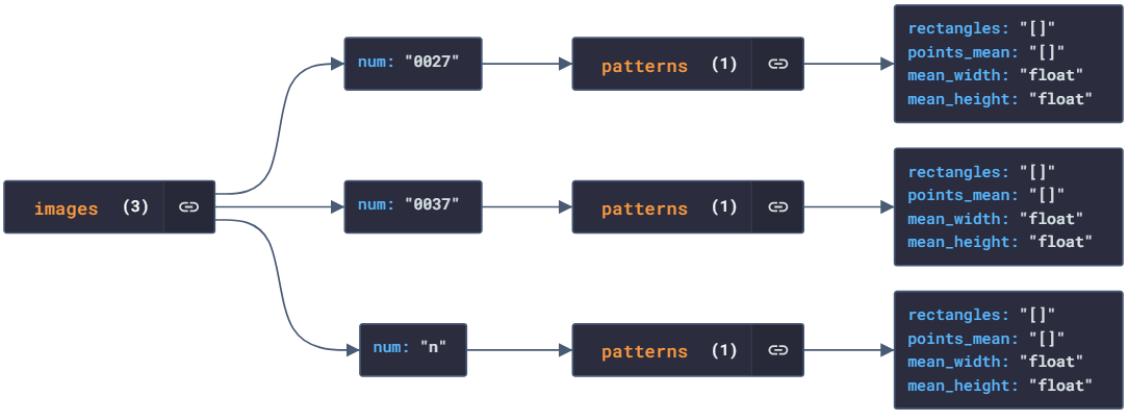

Figura 5.3: Resumen del formato del JSON general creado.

5.2. Funciones generales

A medida que se implementaban los diferentes algoritmos y se realizaban pruebas con las imágenes, surgieron algunos problemas generales. Uno de estos problemas consistía en obtener detecciones superpuestas, es decir, rectángulos casi idénticos que se solapaban entre sí. Para abordar este problema, fue necesario desarrollar una función que filtrara estas detecciones y conservara únicamente aquellas con mayor confianza. En particular, se estableció un criterio de filtrado basado en el solapamiento de los rectángulos. Si se encontraba que una detección con un valor de confianza inferior que se superponía en más del 50 % de su longitud o anchura con otra detección de mayor confianza, entonces se descartaba.

Además del filtro previamente mencionado, se implementó un filtro adicional para asociar las detecciones con las ocurrencias de los patrones. Esto fue necesario debido a que la métrica utilizada requería la identificación exclusiva de pares únicos, seleccionando el mejor emparejamiento entre todas las combinaciones posibles.

5.3. Algoritmos con descriptores clásicos

En diversos estudios [4, 6, 7, 14], se han utilizado distintos algoritmos para abordar la coincidencia de imágenes. Entre estos algoritmos, se ha tomado la decisión de utilizar Template Matching, ORB y SIFT. La elección del primero se debe a su simplicidad y capacidad para realizar emparejamientos de manera sencilla, lo cual lo convierte en un punto de partida idóneo gracias a su baja complejidad y facilidad de uso. Por otro lado, tanto ORB como SIFT presentan implementaciones más complejas que han demostrado obtener mejores resultados en investigaciones previas [1, 4, 7].

5.3.1. Template Matching

Para llevar a cabo la detección de patrones repetitivos utilizando el algoritmo Template Matching, se requiere una imagen de consulta conocida como *template*, y la imagen general de la vasija aplanada. El objetivo es buscar este *template* dentro de la imagen general y encontrar sus posibles ubicaciones.

Para realizar esta tarea, se utiliza una función provista por la biblioteca OpenCV llamada `matchTemplate`. Esta función toma como entrada ambas imágenes en formato de escala de grises y, mediante el método `cv2.TM_CCOEFF_NORMED`, realiza un proceso de emparejamiento. El resultado es una matriz que contiene las correlaciones entre el *template* y las diferentes regiones de la imagen general. Cada posición en la matriz corresponde a la esquina superior izquierda de un rectángulo que representa una posible detección del patrón.

Posteriormente, se procede a ordenar las detecciones de acuerdo a las correlaciones obtenidas, de mayor a menor. Luego, se aplica un filtro utilizando un umbral fijo de 0,5. Este paso es necesario debido a que el algoritmo de Template Matching realiza una búsqueda exhaustiva, generando todas las posibles ubicaciones del *template* desplazándolo píxel a píxel dentro de la imagen. El filtro permite seleccionar únicamente las detecciones que superen el umbral establecido, descartando aquellas con una similitud inferior. Con este proceso, se obtienen las detecciones más relevantes y prometedoras que corresponden a la presencia del patrón buscado en la imagen general.

A continuación, si es necesario, se procede a aplicar un filtro adicional a las detecciones obtenidas, utilizando la función mencionada en la sección anterior que evita la superposición excesiva de los rectángulos detectados. Este filtro tiene como objetivo refinar aún más los resultados, eliminando aquellas detecciones que presenten un alto grado de solapamiento. De esta forma, se obtienen las detecciones finales correspondientes a una imagen específica y al patrón buscado.

5.3.2. ORB

El algoritmo ORB utiliza emparejamientos de puntos clave entre dos imágenes. Para comenzar, se crea una instancia de tipo ORB utilizando la función `cv2.ORB_create()`, la cual se encarga de encontrar los puntos clave al recibir un par de imágenes en escala de grises. Luego, se utiliza un objeto de tipo `cv2.BFMatcher` para realizar el emparejamiento de estos puntos clave. Una vez obtenidos los emparejamientos, se ordenan de menor a mayor distancia. De esta forma, si se desea filtrar las superposiciones, se seleccionan únicamente las mejores parejas de puntos.

Posteriormente, para convertir estos puntos clave en un rectángulo que encierre el patrón buscado, se posiciona cada punto clave de la imagen buscada sobre la imagen completa de la vasija. De esta manera, se crea un rectángulo del tamaño del patrón buscado alrededor de cada punto clave. Este proceso se repite para cada punto clave, obteniendo así un conjunto de detecciones en forma rectangular.

5.3.3. SIFT

El algoritmo SIFT se utiliza para identificar puntos clave o características en una imagen que son invariantes a la escala, rotación e iluminación. El proceso de uso de SIFT es similar al de ORB. Primero, se crea un objeto SIFT utilizando la función `cv2.SIFT_create()` de la biblioteca OpenCV. Este objeto extrae los puntos clave y los descriptores de una imagen. A continuación, se crea una instancia de `cv2.BFMatcher()`, que se encarga de emparejar los descriptores de dos imágenes. Los emparejamientos obtenidos pueden ser ordenados para

seleccionar los mejores en caso de aplicar algún filtro.

Con los emparejamientos y las posiciones de los puntos clave, es posible determinar la correspondencia entre los puntos de una imagen y los de la otra. Esta información permite obtener la posición de cada punto clave en una imagen con respecto a otra.

Finalmente, al igual que con ORB, para encerrar el patrón buscado en un área rectangular, se aplica el mismo enfoque: se posiciona cada punto clave sobre la imagen completa de la vasija y se crea un rectángulo que abarque el tamaño del patrón buscado alrededor de cada punto clave.

5.4. Algoritmos con aprendizaje profundo

Inicialmente se consideraron dos opciones para abordar el problema de detección de patrones con aprendizaje profundo: utilizar una arquitectura de red neuronal pre-entrenada en PyTorch y/o utilizar LoFTR. Sin embargo, se optó por utilizar exclusivamente LoFTR debido a que esta red ha sido específicamente desarrollada para abordar este tipo de problemas de correspondencias entre imágenes y, al mismo tiempo, no requiere modificaciones adicionales ni el uso de Transfer Learning para adaptarse al problema planteado.

5.4.1. LoFTR

LoFTR es una red neuronal que permite encontrar coincidencias entre dos imágenes. Para utilizar esta red, se crea un objeto *matcher* llamado `KF.LoFTR()`, el cual utiliza por defecto la versión pre-entrenada para escenas al aire libre (*outdoor*). A este *matcher* se le proporciona un diccionario que contiene las dos imágenes a evaluar, representadas como tensores en escala de grises.

Al aplicar LoFTR, se obtienen matrices que contienen los puntos clave de cada imagen emparejados, así como la confianza asociada a cada coincidencia. Estos valores de confianza son utilizados para ordenar las coincidencias en función de su nivel de confiabilidad. De esta manera, es posible obtener las coincidencias más relevantes entre las dos imágenes analizadas.

En una variante del enfoque utilizado por los demás algoritmos, se lleva a cabo una doble pasada en las imágenes. Después de aplicar los pasos previamente descritos, se eliminan las detecciones correspondientes al patrón buscado en la imagen aplanada de la vasija. Luego, se realiza una segunda pasada en esta imagen modificada con el objetivo de identificar detecciones adicionales que no se limiten únicamente a la ocurrencia exacta del patrón buscado. De esta manera, se evita descartar otras posibles ocurrencias del mismo patrón y se amplía el alcance de detección en el análisis.

En la etapa final del proceso, los puntos clave obtenidos en ambas pasadas se combinan en una sola lista. Posteriormente, al igual que en los otros métodos mencionados previamente, se aplica un filtro opcional a los puntos clave. Luego de este filtro, se procede a generar un rectángulo para cada punto de emparejamiento encontrado, de manera que se obtiene un conjunto de detecciones representadas por rectángulos delimitadores.

5.5. Computo de resultados

Para aplicar los algoritmos a todo el conjunto de datos, se desarrolló una función específica que lee el archivo JSON que contiene la información previamente mencionada durante el preprocesamiento de los datos. Esta función recorre la numeración de las imágenes para acceder a las respectivas carpetas y obtener las imágenes aplanadas correspondientes. Posteriormente, invoca la función específica para ejecutar el algoritmo seleccionado en cada imagen. Los resultados obtenidos se guardan en un archivo JSON ubicado en Google Drive. Esto permite, en etapas posteriores, desde otro cuaderno de Google Colab, realizar evaluaciones para determinar qué tan efectivamente se lograron detectar los patrones en las imágenes.

5.6. Implementación de métricas

Basándose en la explicación y propuestas de métricas presentadas en el trabajo de referencia [2], se logró implementar el cálculo de la métrica Mean Average Precision (MAP) desde cero utilizando Python en el entorno de Google Colab. La implementación se adaptó específicamente al problema en cuestión, aprovechando las detecciones generadas por los algoritmos previamente mencionados. Para llevar a cabo estos cálculos, fue fundamental utilizar la biblioteca NumPy, ya que le permitió al sistema trabajar eficientemente con matrices y ordenar los índices de las detecciones según su calidad. Además, se utilizó el paquete Shapely, el cual proporcionó funcionalidades para trabajar con polígonos y realizar cálculos automáticos de intersección y unión de áreas entre ellos. A continuación, se proporcionará una explicación más detallada de la implementación realizada.

5.6.1. MAP: Mean Average Precision

Para implementar esta métrica, primero se lee el archivo JSON con los resultados de las detecciones y el archivo que contiene las recurrencias reales. Luego, se recorre esta información de imagen en imagen, y dentro de esto patrón a patrón. En este punto, se generan todos los pares posibles (d_i, o_i) entre las detecciones encontradas d_i y los patrones verdaderos o_i. Luego, con estos pares, se pueden obtener las áreas de cada uno al acceder al polígono de la detección junto al de la ocurrencia, así se crean con Polygon de Shapely estos objetos y se calcula fácilmente el IoU (Intersection Over Union) de este par. A continuación, se ordenan según el resultado de IoU de mayor a menor, para luego hacer un filtro para que los pares (d_i, o_i) no se repitan, recorriendo según el orden anterior, de esta manera solo se seleccionen los que presenten mayor valor de IoU.

Con los valores de IoU ya calculados, se marcan los pares con IoU$>$0.5 como verdaderos positivos, en caso contrario como falsos positivos. Además, si hay algún patrón verdadero o_i que no tiene un par d_i, se marca como falso negativo. Con estos resultados, se puede calcular Recall y Precision por cada patrón en imagen, lo que permite a su vez obtener el AP (Average Precision), para luego calcular el MAP como el promedio del AP de todos los patrones de una imagen. Cabe mencionar que se interpolaron a once valores los resultados de Recall y Precision, para hacer el cálculo del área bajo la curva (AP) más sencillo.

5.7. Experimentos

5.7.1. Experimentos preliminares

Al comenzar a usar los algoritmos se fueron probando con algunas imágenes del dataset, para captar cómo funcionaban, mostrando de forma visual los resultados obtenidos.

Un primer acercamiento con Template Matching, mostró que detectaba muy bien el patrón exacto que se estaba buscando, pero no tanto sus repeticiones. De esta forma, se puede observar en la figura 5.4, que se marcan muchas detecciones sobre la misma ocurrencia del patrón. En la imagen son dos ocurrencias las más destacadas, que corresponden a la misma sección de la vasija debido a que se repite parte del objeto al aplanarlo. Dado lo anterior, se decidió hacer la función nombrada en la sección 5.2 que filtra las detecciones sobrepuestas. En la figura 5.5 se pueden ver las detecciones filtradas en la misma imagen de la figura antes nombrada.

Figura 5.4: Imagen 27 con las detecciones de Templete Matching sin variaciones.

Figura 5.5: Imagen 27 con las detecciones de Templete Matching al aplicarles el filtro.

También, continuando con el algoritmo anterior, se observó que no detecta los patrones cuando el tamaño de la imagen template era muy diferente al que se encontraba en la imagen aplanada, por esto se decidió experimentar preliminarmente cambiando el tamaño de la imagen con el patrón a buscar. En la figura 5.6 se puede observar al probar con diferentes tamaños, el primero sin redimensionar y el segundo al cambiar el tamaño a uno similar al de las recurrencias del patrón en la imagen de la vasija.

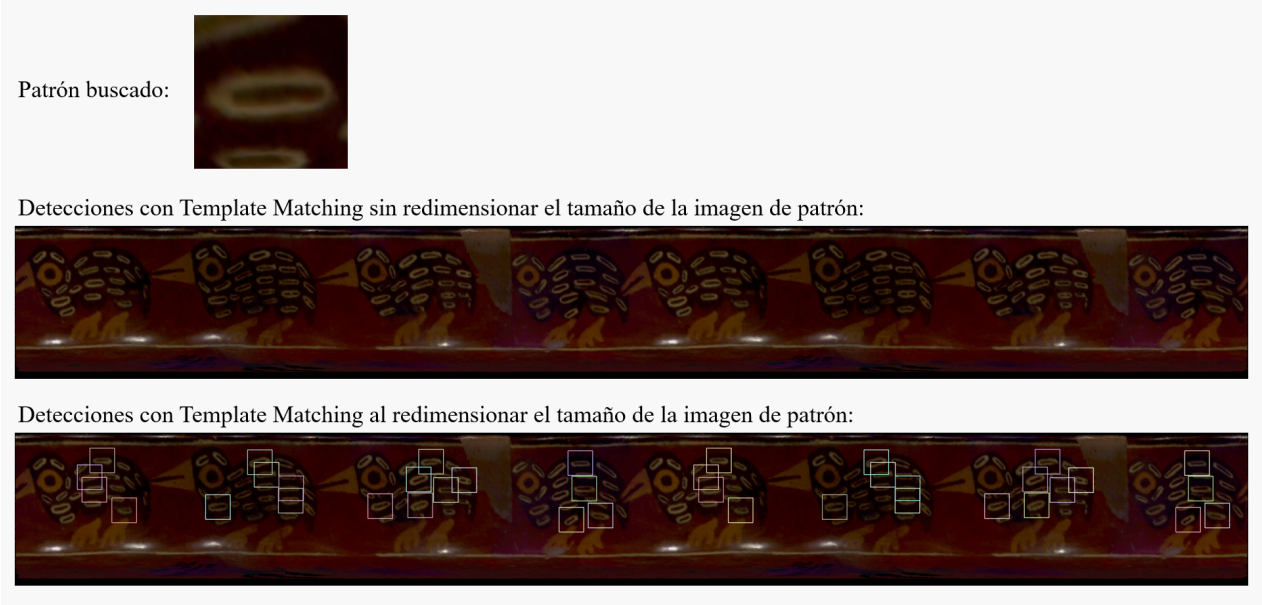

Figura 5.6: Imagen 64 con las detecciones de Templete Matching puro y con redimensión del patrón.

Con ORB y SIFT de igual forma se hicieron pruebas con algunas imágenes, pero en estos casos se observó que los puntos claves emparejados están más esparcidos a través de la imagen.

Por otra parte, con LoFTR se observaron las mismas complicaciones anteriores que con el primer algoritmo, pero aún más marcadas ya que en la figura 5.7 se muestra que los puntos claves emparejados son prácticamente todos en la sección de dónde se obtiene el patrón. Debido a este experimento se decide hacer dos pasadas con este algoritmo, una con la imagen general de la vasija y la segunda con los primeros puntos detectados borrados de la imagen. En la figura 5.8 se puede ver el resultado de la segunda pasada por la imagen alterada.

Figura 5.7: Imagen 31 con los puntos claves emparejados con LoFTR.

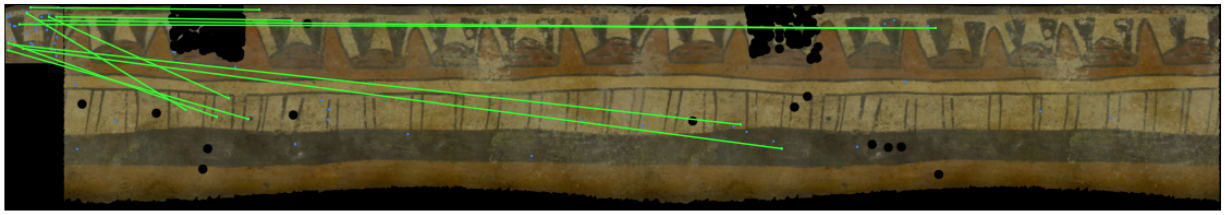

Figura 5.8: Imagen 31 con los puntos claves emparejados con LoFTR, luego de la eliminación de los primeros puntos detectados

5.7.2. Experimentos finales

Todos los experimentos preliminares dieron las directrices para definir los experimentos finales que serían computados a través de todo el conjunto de imágenes y de los que se obtendrían evaluaciones cuantitativas para poder comparar. Es así que se decide hacer los experimentos por algoritmos, ya sea con o sin filtro de las detecciones y con o sin redimensionar el tamaño del patrón. Con estas combinaciones, se obtienen los experimentos finales de la tabla 5.1, donde filtrado indica que todas las detecciones obtenidas por el algoritmo son pasadas por un filtro que las reduce y, redimensionado significa que la imagen del patrón a buscar se le cambia su tamaño a uno más acorde a los patrones reales encontrados en la imagen de la vasija completa.

Tabla 5.1: Tabla de los experimentos definidos.

Nombre	**Descripción**
TM	Template Matching sin variaciones
TM-F	Template Matching filtrado
TM-R	Template Matching redimensionado
TM-FR	Template Matching con filtrado y con redimensionado
S	SIFT sin variaciones
S-F	SIFT filtrado
S-R	SIFT redimensionado
S-FR	SIFT con filtrado y con redimensionado
O	ORB sin variaciones
O-F	ORB filtrado
O-R	ORB redimensionado
O-FR	ORB con filtrado y con redimensionado
L	LoFTR sin variaciones
L-F	LoFTR con filtro
L-R	LoFTR con redimensión
L-FR	LoFTR con filtro y redimensión

Capítulo 6

Validación de resultados

6.1. Resultados

Al llevar a cabo todos los experimentos nombrados en la sección anterior, se obtuvieron valores de MAP por cada imagen, por lo que para poder visualizar de forma general esto, se promediaron estos resultados y se calcularon otras estadísticas como desviación estándar, mínimos y máximos. En la tabla 6.1 se aprecian estos resultados. Además, se cuenta con que para todos los experimentos el mínimo valor de MAP es cero.

Tabla 6.1: Tabla con estadísticas de los resultados de MAP en los experimentos realizados.

Experimento	**Promedio**	**Desv. Estándar**	**Mínimo**	**Máximo**
TM	0,054	0,183	0,000	0,932
TM-F	0,042	0,136	0,000	0,800
TM-R	0,192	0,298	0,000	0,995
TM-FR	0,216	0,299	0,000	1,000
S	0,204	0,303	0,000	0,986
S-F	0,170	0,276	0,000	0,933
S-R	0,328	0,336	0,000	1,000
S-FR	0,254	0,266	0,000	0,967
O	0,171	0,272	0,000	0,950
O-F	0,153	0,257	0,000	0,950
O-R	0,245	0,322	0,000	1,000
O-FR	0,201	0,276	0,000	0,917
L	0,318	0,407	0,000	1,000
L-F	0,244	0,348	0,000	1,000
L-R	0,517	0,390	0,000	1,000
L-FR	0,454	0,349	0,000	1,000

6.2. Análisis de resultados

En la tabla 6.2, se tienen los resultados por algoritmo, de esta forma podemos comparar entre ellos. Así, se puede concluir que el mejor resultado lo obtiene LoFTR con redimensión

(L-R), seguido por el mismo algoritmo pero con filtro y redimensión (L-FR). Además, otro que destacó y se mantuvo cerca de los anteriores fue SIFT con redimensionamiento (S-R).

Algo que puede parecer anormal es que SIFT, ORB y LoFTR mejoren bastante con la redimensión del patrón, cuando estos son métodos fuertes contra la escala y rotación. Sin embargo, estos resultados tienen sentido ya que la implementación aquí expuesta entrega como resultado de la detección un rectángulo con el tamaño del patrón buscado, por lo que, si no se hace una redimensión de su tamaño, es muy posible que no pase el umbral de IoU al comparar este rectángulo con el que encierra la ocurrencia real del patrón que puede tener un tamaño muy diferente. Es por esto mismo, que el hacer esta redimensión de tamaño mejora los resultados de todos los algoritmos. Más concretamente, Template Matching es el único que presenta su mejor resultado al aplicarle la variación de redimensión en conjunto con el filtro, lo que concuerda con el hecho de que se decidió probar filtrar los resultados, ya que este algoritmo detectaba el mismo patrón repetidas veces.

Tabla 6.2: Tabla comparativa de resultado del promedio de MAP separada por algoritmo.

Algoritmo	**Template Matching**	**ORB**	**SIFT**	**LoFTR**
Sin modificaciones	0,054	0,171	0,204	0,318
Con filtro	0,042	0,153	0,170	0,244
Con redimensión	0,192	0,245	0,328	0,517
Con filtro y redimensión	0,216	0,201	0,254	0,454
Valores máximos	0,216	0,245	0,328	0,517

Cabe destacar que, viendo los promedios máximos de MAP por algoritmos, LoFTR sobrepasa en gran cantidad a los demás. Además, Template Matching y ORB dan un resultado muy similar en este caso, lo que da a entender que según las modificaciones que se hagan tanto Template Matching y ORB pueden entregar resultados similares. No obstante, LoFTR es el que mejor se comporta incluso al no modificar el algoritmo, lo que puede explicarse con que sus detecciones se concentran principalmente en la parte exacta de dónde se obtiene el patrón, es decir, si se busca exactamente un mismo objeto es muy bueno detectándolo, pero si estos son similares más no idénticos, como es el caso de estos patrones hechos a mano, se le presenta un poco más de dificultad. Pese a esto LoFTR gana en todas las variaciones de los algoritmos ya que tiene a su favor la complejidad de las redes neuronales junto a su pre-entrenamiento.

Por otro lado, en la figura 6.1, se puede observar de forma gráfica cómo se comportan los diferentes algoritmos en algunas imágenes específicas, en este caso con la variación que usa el filtro y la redimensión del patrón. En particular, en la última imagen Template Matching se destaca, ya que detecta muy bien el patrón pequeño de la imagen, cuando los demás algoritmos presentan grandes dificultades. También, ORB y SIFT presentan buenos resultados en la primera y segunda imagen. Aquí se puede apreciar que, si bien LoFTR gana en la tabla de promedios, estos algoritmos clásicos no se quedan atrás y pueden presentar muy buenos resultados en ciertas imágenes específicas, sin embargo, esto no es representativo en términos generales.

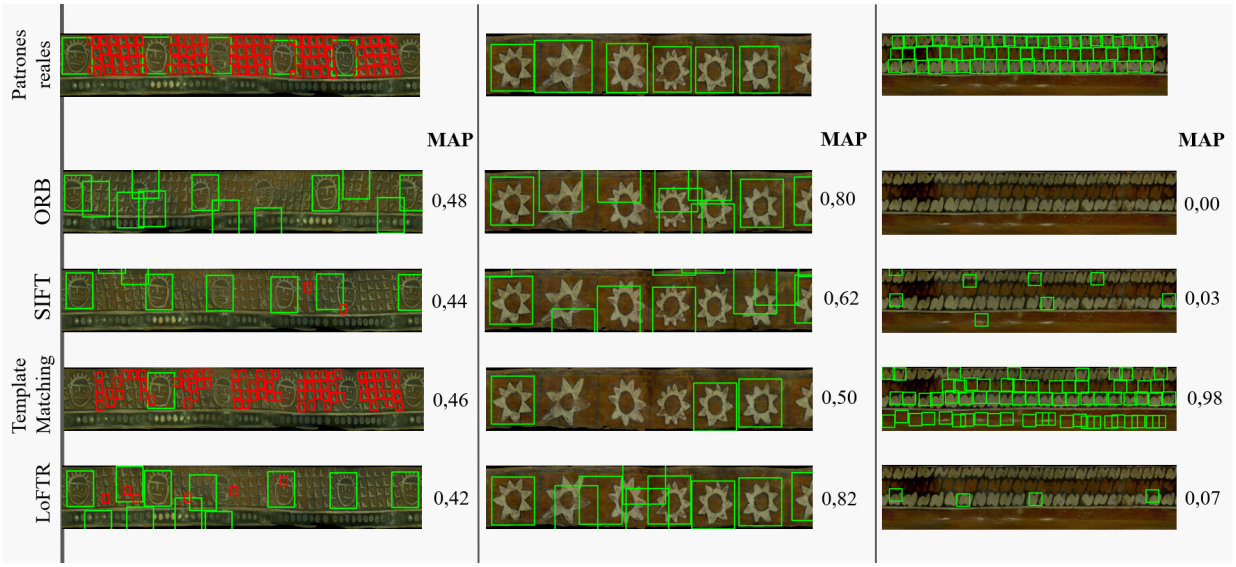

Figura 6.1: Detecciones y promedio MAP obtenido con la imagen 36, 39 y 97, para los diferentes algoritmos con la variación de filtro y redimensión

Capítulo 7

Discusión

Como se observó en la sección anterior, ningún resultado de MAP fue muy sobresaliente, más detalladamente, solo uno sobrepasó el cincuenta por ciento en promedio con esta métrica. Sin embargo, en esta investigación no se tiene como meta alcanzar precisiones extraordinarias, sino más bien, comparar los diferentes métodos aquí planteados, para así poder saber cuál funciona mejor en la detección de patrones en objetos arqueológicos. Es por esto que el tener resultados que se puedan comparar entre los diferentes algoritmos y sus variaciones, ayuda a cumplir esta meta.

Por otra parte, a lo largo del desarrollo de esta investigación, se tuvieron diferentes problemas, uno de ellos ocurrió al querer usar SIFT con la versión por defecto de Google Colab, que no permitía usarlo por temas de licencia. De hecho, en primera instancia se quería usar solo SIFT en lugar de ORB, no obstante, por este problema se fueron haciendo pruebas con ORB, lo que conllevó a que se incluyera en el desarrollo de esta investigación. Una primera solución fue usar la versión 4.4.0.40 de OpenCV, pero más adelante Colab actualizó la versión por defecto a 4.6.0 por lo que se terminó usando esta. Finalmente, pese a que ocurrió este problema, se pudo solucionar, y dado el avance ya realizado, se continuó trabajando en Colab.

Otra complicación, fue con el uso de LoFTR ya que en promedio le tomaba dos minutos analizar una de las 82 imágenes. El problema con esto se debe a que Google Colab limita el tiempo de uso de sus recursos, haciendo casi imposible pasar por 82 imágenes en un mismo bloque de código. Esto se solucionó, calculando las detecciones de diez en diez imágenes, guardando esta información, que luego se une en un solo JSON. Cabe destacar, que la implementación que tiene Kornia de LoFTR solo permite recibir dos imágenes, por lo que no fue posible aprovechar el paralelismo del que normalmente se disfruta al usar modelos de aprendizaje profundo ni fue necesario requerir de una máquina con más capacidad de *hardware* como CPU debido a que el algoritmo procesaba secuencialmente las imágenes.

Capítulo 8

Conclusiones

La principal meta de esta investigación es aportar a definir qué algoritmos funcionan mejor en el mundo de la arqueología digital, para ayudar a los científicos de esta área a tomar decisiones sobre qué herramientas usar en la detección de patrones. En esta línea, se tiene que esta investigación aporta una semilla en esta área mezclada con el campo de la visión computacional, ya que se logra llegar a comparar los métodos usados en esta investigación, lo que permite entender cuáles funcionan mejor en sus diferentes variaciones. En específico, se tiene que LoFTR obtiene los dos mejores resultados de promedio de MAP de los distintos experimentos, con un valor de 0,517 y 0,454 seguido por SIFT con 0,328. Si bien están lejanos en este caso, hay imágenes específicas que dan mejores resultados con algoritmos clásicos, pero esto no es representativo. Dentro de lo clásico, SIFT se destaca como es de esperar. Además, Template Matching obtiene su mejor promedio de MAP 0,216 en un experimento que logró superar a ORB, esto no se compara con los resultados de SIFT o LoFTR, pero al ser un algoritmo sencillo en comparación al resto no deja de sorprender.

Si se deseara continuar con esta investigación, sería muy interesante realizar pruebas con un modelo de red ya sea Resnet, o algo más avanzado con Transformers, trabajando directamente con ella en PyTorch, ya que así se podría realizar Transfer Learning, como se tenía pensado en la propuesta inicial de la solución, junto al hecho de que se aprovecharía el paralelismo para más rapidez en cómputo. Por lo tanto, se tendrían más casos con el que comparar, además de LoFRT, en el ámbito del aprendizaje profundo.

Otro aspecto que no se desarrolló fue la detección de la forma exacta del patrón, ya que los modelos usados no entregaban información tan detallada, sin embargo, se puede dejar como una propuesta futura el investigar algoritmos que entreguen este tipo de detección y evaluar cómo responden a los objetos arqueológicos usados durante este trabajo.

Bibliografía

[1] Tareen, S. A. K. y Saleem, Z., "A comparative analysis of sift, surf, kaze, akaze, orb, and brisk," en 2018 International conference on computing, mathematics and engineering technologies (iCoMET), pp. 1–10, IEEE, 2018.

[2] Lengauer, S., Sipiran, I., Preiner, R., Schreck, T., y Bustos, B., "A benchmark dataset for repetitive pattern recognition on textured 3d surfaces," en Computer Graphics Forum, vol. 40, pp. 1–8, Wiley Online Library, 2021.

[3] Cai, Y. y Baciu, G., "Translation symmetry detection: A repetitive pattern analysis approach," en Proceedings of the IEEE conference on computer vision and pattern recognition workshops, pp. 223–228, 2013.

[4] Rublee, E., Rabaud, V., Konolige, K., y Bradski, G., "Orb: An efficient alternative to sift or surf," en 2011 International conference on computer vision, pp. 2564–2571, Ieee, 2011.

[5] Schindler, G., Krishnamurthy, P., Lublinerman, R., Liu, Y., y Dellaert, F., "Detecting and matching repeated patterns for automatic geo-tagging in urban environments," en 2008 IEEE Conference on Computer Vision and Pattern Recognition, pp. 1–7, IEEE, 2008.

[6] Wu, C., Frahm, J.-M., y Pollefeys, M., "Detecting large repetitive structures with salient boundaries," en European conference on computer vision, pp. 142–155, Springer, 2010.

[7] Karami, E., Prasad, S., y Shehata, M., "Image matching using sift, surf, brief and orb: performance comparison for distorted images," arXiv preprint arXiv:1710.02726, 2017.

[8] Lowe, D. G., "Distinctive image features from scale-invariant keypoints," International journal of computer vision, vol. 60, no. 2, pp. 91–110, 2004.

[9] Marengoni, M. y Stringhini, D., "High level computer vision using opencv," en 2011 24th SIBGRAPI Conference on Graphics, Patterns, and Images Tutorials, pp. 11–24, IEEE, 2011.

[10] En, S., Petitjean, C., Nicolas, S., y Heutte, L., "A scalable pattern spotting system for historical documents," Pattern Recognition, vol. 54, pp. 149–161, 2016.

[11] Úbeda, I., Saavedra, J. M., Nicolas, S., Petitjean, C., y Heutte, L., "Pattern spotting in historical documents using convolutional models," en Proceedings of the 5th International Workshop on Historical Document Imaging and Processing, pp. 60–65, 2019.

[12] He, K., Zhang, X., Ren, S., y Sun, J., "Deep residual learning for image recognition," en Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 770–778, 2016.

[13] Sun, J., Shen, Z., Wang, Y., Bao, H., y Zhou, X., "Loftr: Detector-free local feature

matching with transformers," en Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 8922–8931, 2021.

[14] Lowe, D. G., "Object recognition from local scale-invariant features," en Proceedings of the seventh IEEE international conference on computer vision, vol. 2, pp. 1150–1157, Ieee, 1999.