Diseño de la solución

Daniel Hernández Ferrándiz - Diego Silva López

Marzo 2020

1 Introducción

En esta fase del proyecto, se ha buscado una solución efectiva para el problema dado, detección de árboles en imágenes aéreas.

1.1 Funcionamiento de test

A continuación, se va a mostrar los resultados obtenidos en los test.

(a) Máscara de color. (b) Resultado obtenido.

Figure 1: Resultados tests.

A la izquierda de esta imagen es la máscara que se ha obtenido al filtrar la imagen inicial dejando pasar los píxeles que se están dentro de un rango. El rango deja pasar todos aquellos píxeles que son del color de los árboles, ciertos tonos de verde.

A la derecha de esta imagen se ha combinado la máscara anterior con la imagen inicial. Los píxeles iluminados en la máscara permiten resaltar las zonas en las que hay árboles marcados en verde.

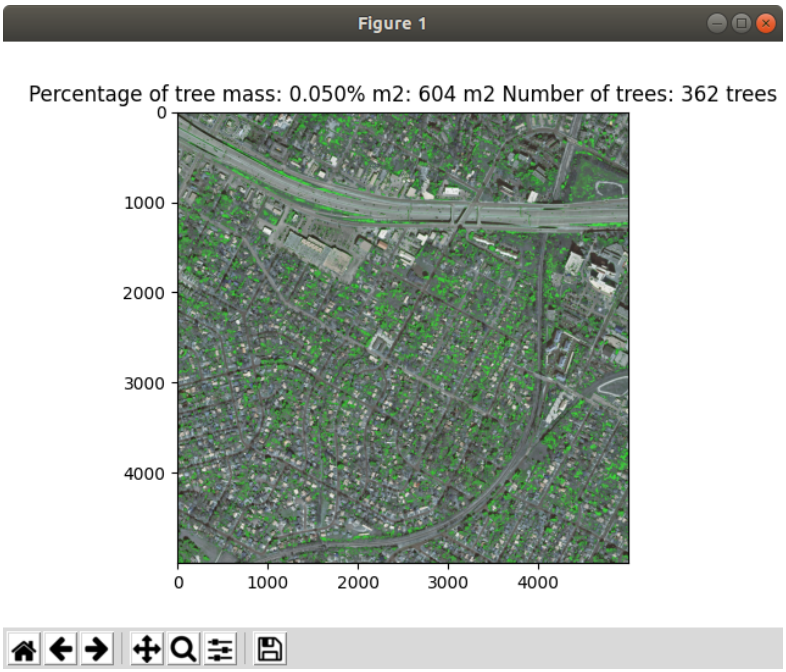

Figure 2: Resultado de la aplicación.

Por último, en esta figura podemos ver la imagen anterior, pero con la información que se pedía en un principio. Número, porcentaje y metros cuadrados de árboles en la imagen. Se puede apreciar que el porcentaje es un valor muy pequeño y esto es debido a que la imagen de entrada tiene alta resolución, 5000x5000, por lo que, el número de píxeles de árboles es muy pequeño en comparación del número total de píxeles.

1.2 Enlace al repositorio GitHub

Este es el enlace al repositorio GitHub: https://github.com/dsilvalo28/AIVA-DAIA

2 Diseño de la solución

En esta sección se va a explicar el funcionamiento de la aplicación mediante diagramas UML de clases y una descripción del funcionamiento de cada clase.

2.1 Application

Esta clase permite la comunicación entre el usuario y el programa, funciona como interfaz. En ella el usuario podrá realizar diferentes procedimientos. Lo primero y principal, el usuario podrá decirle al sistema la zona geográfica que se desea

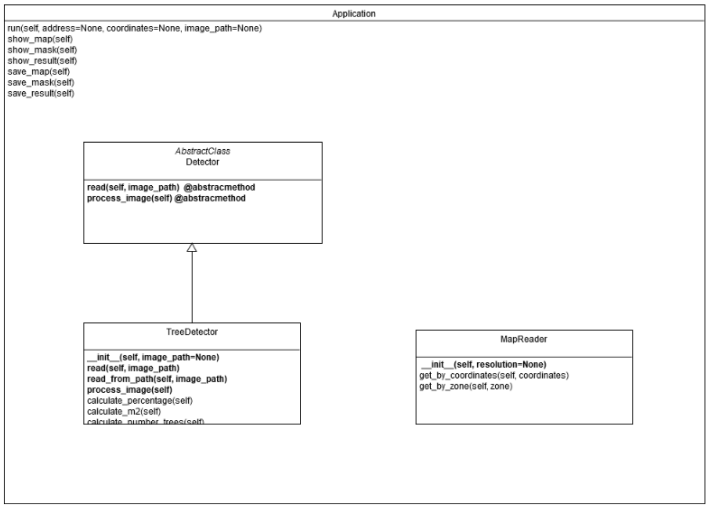

Figure 3: Diagrama de clases de la aplicación.

estudiar mediante el método run introduciendo una dirección, las coordenadas geográficas o la propia imagen que desea estudiar. Podrá visualizar el mapa de la zona que desea analizar con el método show_map, si lo desea también podrá visualizar la máscara de color que ha generado la aplicación mediante el método show_mask y podrá visualizar el resutado gracias al método show_result. Cada una de estas imágenes, el mapa, la máscara de color y el resultado final, podrán almacenarse en el sistema con sus respectivos métodos save.

2.2 Detector

La clase detector es una clase abstracta que funciona como interfaz para el resto de las distintas clases que hereden de ella. Por el momento solo hay una clase que herede de ella, TreeDetector. Se ha considerado conveniente crear una clase padre por si en un futuro se desea añadir más funcionalidad a la aplicación, por ejemplo, con nuevas tareas de detección de diversos objetos o patrones.
Esta clase solo tiene dos métodos, read y process_image. Ambos métodos son abstractos, por lo que cada clase que herede de esta se verá obligado a implementarlos. Al método read se le pasa la ruta de una imagen y él se encarga de leerla y guardarla en memoria. El método process_image se encarga de aplicar los algoritmos necesarios a la imagen para obtener los datos deseados. Dependiendo de la clase que implemente este método los algoritmos cambiaran.

2.3 TreeDetector

Es la clase principal de la aplicación. Hereda de la clase Detector y por consiguiente tiene implementados los métodos read y process_image. El método read como se ha explicado antes solamente carga la imagen en memoria. El método process_image se encarga de aplicar el algoritmo de detección de árboles a la imagen. A parte de estos dos métodos, la clase implementa otros distintos. Calculate_percentage, encargado de obtener qué porcentaje de la imagen está formada por árboles. Cálculate_m2, encargado de calcular cuántos metros cuadrados hay de árboles en las imágenes. Calculate_numer_ tree, cuya tarea es obtener el número de árboles que hay en la imagen. Este método se ayuda de los dos anteriores para poder calcular el dato con mayor precisión.

2.4 MapReader

Esta clase es la encargada de generar las imágenes de entrenamiento para la aplicación. Es necesaria esta clase ya que no se cuenta con la cantidad de imágenes necesarias para entrenar el algoritmo. La clase se conecta a la API de Google para obtener imágenes aéreas dadas las coordenadas de una zona o su dirección, Maps Static. Los principales métodos de la clase son get_by_coordinates y get_by_zone. Get_by_zone, se encarga de obtener las coordenadas de una zona dada una dirección mediante la API de Google Geocode. Una vez con estas coordenadas hace uso de get_by_coordinates, que es el método encargado de generar las imágenes en un formato válido para la aplicación.

2.5 Diagrama de secuencia

En este apartado se va a explicar el diagrama de secuencia de la aplicación.

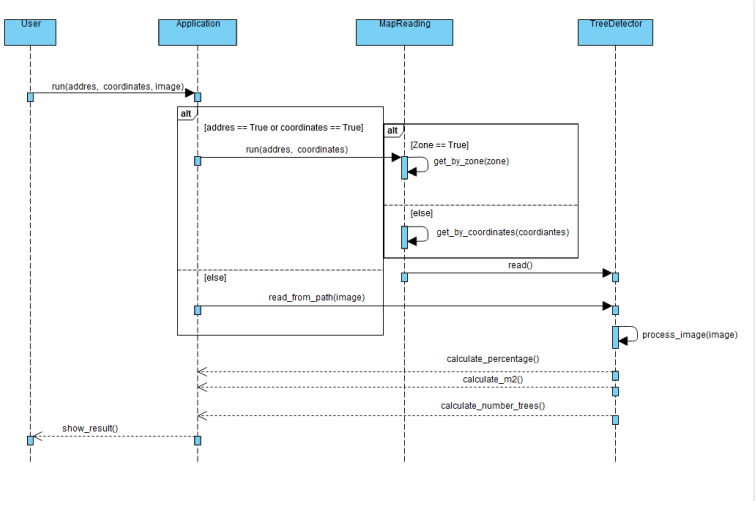

Figure 4: Diagrama de secuencia de la aplicación.

Como se puede observar está formado por 4 principales agentes: el usuario y las clases Application, MapReading y TreeDetector. El flujo lo comienza el usuario introduciendo uno de los 3 parámetros posibles, dirección, coordenadas o imagen. En el caso de que se haya introducido uno de los dos primeros parámetros, la aplicación hará uso de la clase MapReading para generar la imagen de la zona que se desea analizar. En caso contrario, la imagen introducida en la aplicación pasará directamente a la clase TreeDetector donde será analizada.
Una vez en la clase MapReading se comprobará si la información que ha llegado es una dirección o una coordenada. Esto es debido a que las coordenadas pueden ser rastreadas directamente en la API de Google y las direcciones necesitan un tratamiento previo para transformarlas a coordenadas. Las imágenes obtenidas se pasan a la clase TreeDetector.
En la clase TreeDetector se ejecuta el algoritmo de detección y se devuelve la información obtenida a la clase Application. Esta información es el número de árboles, los metros cuadrados y el porcentaje de árboles en la imagen.
Finalmente, esta información le llega al usuario a través de la clase Application.

2.6 Algoritmo de detección

Este algoritmo está implementado en la clase TreeDetector. Dada una imagen el

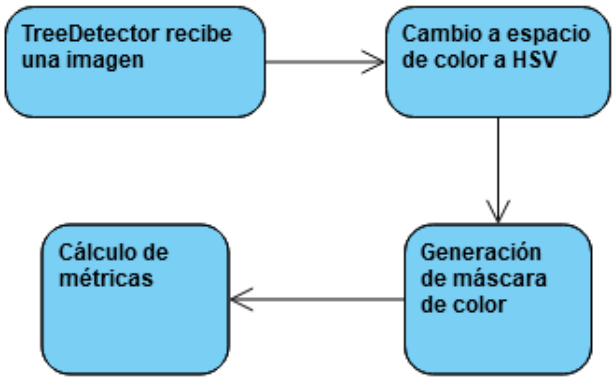

Figure 5: Diagrama de acción algoritmo detección.

primer paso es transformarla al espacio de color HSV. Una vez con la imagen en el espacio de color deseado, se crea una máscara de color para filtrar los colores más parecidos al tono de color de los árboloes. La máscara no es más un filtro que solamente deja pasar los píxeles que tengan un valor entre un intervalo.
Con la máscara generada se calculan las siguientes medidas. Porcentaje de árboles en una imagen, se divide el número de píxeles que tienen valor positivo

en la máscara entre el número total de píxeles de la imagen. Metros cuadrados de árboles, previamente se ha estimado cuantos píxeles equivalen a un metro cuadrado en la imagen, por lo que sabiendo cuantos píxeles tienen valor positivo en la máscara se puede calcular los metros cuadrados ocupados por árboles. Número de árboles, al igual de que la medida anterior se ha estimado cuantos píxeles ocupa un árbol, por lo que siguiendo el procedimiento anterior podemos obtener el número de árboles en la imagen.

2.7 Algoritmo generador de imágenes

Este algoritmo está implementado en la clase MapReading. Dadas unas coordenadas o una dirección el algoritmo es capaz de obtener una imagen para enviarla a TreeDetector. Si recibe una dirección lo primero que tiene que hacer es transformar esa dirección a coordenadas geográficas. Esto se consigue conectándose a la API Geocode Google que permite esta transformación. Una vez que el sistema tiene las coordenadas, se conecta a la API Maps Static Google para descargarse la imagen de la zona deseada. Debido a que las imágenes que se utilizarán en un futuro tienen una resolución de 5000x5000 píxeles y que las imágenes obtenidas de la API tienen una resolución máxima de 640x640, es necesario descargarse un total de 100 imágenes de 500x500 para generar una imagen con la misma resolución que la deseada. Para ello se toman las coordenadas dadas como puntos centrales de la imagen y se expande esta zona hasta abarcar la resolución requerida.
Una vez obtenidas las imágenes se unen respetando las coordenadas hasta generar una imagen final de 5000x5000. Esta imagen es la que se envía al algoritmo de detección para que trabaje con ella.

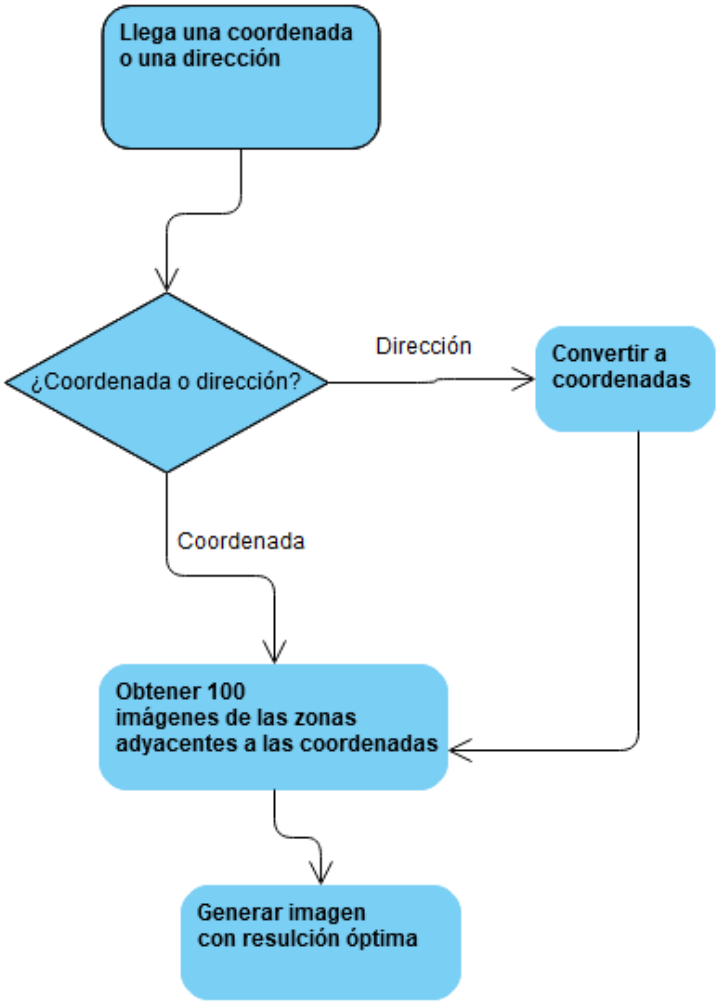

Figure 6: Diagrama de acción algoritmo generador de imágenes.