UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf
Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

BuscaPic, detección de picos mediante realidad aumentada y el servicio Goggle Maps Elevation API

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Sebastián Vicente Ciscar Breitzler

Tutor: José Vicente Soler Bayona

Curso 2015/2016

Resumen

El proyecto a realizar trata de introducir el servicio de Goggle Maps Elevation para mejorar la detección de picos o montañas utilizando realidad aumentada. Se pretende mejorar la fiabilidad de los sensores de un dispositivo Android, especialmente el compás magnético y a la vez detectar la superposición de picos a partir de los valores de altura obtenidos mediante el servicio. Para todo ello se utilizará los datos obtenidos a partir de los sensores del dispositivo (GPS, compás, acelerómetros) y se accederá a la información pública del Instituto Geográfico Nacional (vértices geodésicos).

Palabras clave: realidad aumentada, detección, Google Maps Elevation, Android, altura.

Resum

El projecte a realitzar tracta d'introduir el servici de Goggle Maps Elevation per a millorar la detecció de pics o muntanyes utilitzant realitat augmentada. Es pretén millorar la fiabilitat dels sensors d'un dispositiu Android, especialment el compàs magnètic i al mateix temps detectar la superposició de pics a partir dels valors d'altura obtinguts per mitjà del servici. Per a tot això s'utilitzarà les dades obtingudes mitjançant dels sensors del dispositiu (GPS, compàs, acceleròmetres) i s'accedirà a la informació pública de l'Instituto Geogràfico Nacional (vèrtexs geodèsics)

Paraules clau: realitat augmentada, detecció, Google Maps Elevation, Android, altura.

Abstract.

The project to be carried out is introducing the service of Goggle Maps Elevation to improve the detection of peaks or mountains using augmented reality. Aims to improve the reliability of a device Android, especially the magnetic compass and sensors simultaneously detect the superposition of peaks from the height values obtained through the service. For all data obtained from sensors of the device will be used (GPS, compass, accelerometer) and access to public information of the national geographical Institute (geodetic vertices).

Key words: augmented reality, detection, Google Maps Elevation, Android, height.

Tabla de contenidos

Tabla de ilustraciones

1. Introducción.

1.1. Motivación

El motivo de este proyecto viene dado por la curiosidad innata en el montañero por conocer las cotas de altura frente a él. Poder llegar a un lugar donde vislumbrar el horizonte y poder saber la altura y el nombre de cualquier cota frente a él. En la actualidad disponemos de potentes dispositivos que llevamos con nosotros casi en cualquier momento, por lo que el uso de estos dispositivos para dar respuesta a estas cuestiones es obvio.

Para facilitar el uso y hacerlo más eficiente pensamos en utilizar la realidad aumentada, esta tecnología está viviendo un resurgir debido a la potencia de los dispositivos.

Tras realizar una búsqueda de aplicaciones ya realizadas con un propósito similar, encontramos que no hay ninguna que se adapte a nuestras expectativas, puesto que deseamos conocer la cota de cualquier punto alto recortado en el horizonte, y todas hacen uso de datos almacenados con anterioridad para indicar las alturas de las cumbres.

Además, debido a esta colección de datos limitada, no es posible conocer la altura de cumbres de menor importancia, pero que, debido a nuestra situación geográfica respecto a una cumbre más importante situada a espaldas de esta, no son tenidas en cuenta por las aplicaciones, arrojando incluso errores significativos en los datos, puesto que el accidente geográfico señalado no es el correspondiente.

Por lo que, esta aplicación va dirigida al público en general, amante de la montaña que desee conocer las cotas de altura de cualquier accidente geográfico frente a él.

1.2. Objetivos

El objetivo principal para este proyecto es la creación de una primera versión de aplicación móvil desarrollada para dispositivos con sistema operativo Android.

Esta aplicación va dirigida al público en general y no solamente a montañeros experimentados.

Como objetivos englobados dentro del principal tenemos los siguientes:

- Obtención de la altura del accidente geográfico enfocado, mostrando sobre impresa la altura en la imagen en tiempo real del pico.

- Consulta de la toponimia asociada al lugar, en caso de estar disponible, mostrado sobreimpreso en tiempo real.

- Distancia geográfica.
- Distancia en ruta.
- Gráfico de altimetría.
- Ajuste del compás magnético del dispositivo, por medio del tratamiento de imágenes.

1.3. Estructura de la memoria.

Esta memoria está dividida en los siguientes capítulos, a continuación, describiremos brevemente cada uno de ellos.

- **Introducción**: en este apartado se presenta un breve resumen del proyecto, seguida de la motivación que suscita la realización del proyecto, los objetivos marcados para su realización y finalmente este mismo apartado con la estructura interna de la memoria.

- **Estado de la cuestión:** en este capítulo analizamos el estado de la tecnología actual al respecto del problema que nos atañe. También realizamos un pequeño recorrido por las aplicaciones, con un funcionamiento similar, existentes.

- **Análisis:** se describe el estudio realizado al problema para encontrar una solución satisfactoria. En él podemos encontrar el diagrama que representa el funcionamiento de la aplicación, el análisis de requisitos y los diagramas de casos de uso y el de clases. También podemos ver el análisis de riesgos realizado para este proyecto.

- **Diseño:** en esta parte de la memoria presentamos los algoritmos y cálculos más importantes, estos serán realmente la parte más importante del diseño de la solución para el proyecto. Está incluida también el prototipado de la interfaz gráfica de usuario.

- **Implementación:** aquí desgranamos como ha sido la construcción de la aplicación, pasando a describir cada iteración que forma la codificación de la herramienta.

- **Conclusión:** como punto final, el estado final del proyecto y las acciones aprendidas.

2. Estado de la cuestión.

2.1. Tecnologías utilizadas

2.1.1. Android

El sistema operativo Android es una plataforma para dispositivos móviles basada en Linux. Ha sido desarrollado por Google y la Open Handset Alliance (OHA).

La plataforma Android incluye un sistema operativo basado en Linux, una interfaz gráfica de usuario, un navegador web y aplicaciones de usuario que pueden ser descargadas. Este sistema operativo está diseñado para poder ser utilizado en la gran mayoría de dispositivos móviles.

Android soporta multitud de estándares en cuanto a dispositivos móviles se refiere, como Bluetooth, Wifi, SMS, Pantalla táctil, GPS, compás, acelerómetros, etc...

2.1.2. GPS

El GPS (Servicio de posicionamiento global), es un sistema propiedad de los Estados Unidos de América que proporciona, la posición, el desplazamiento y el tiempo.

Su funcionamiento se basa en una red de 24 o más satélites, que orbitan alrededor de la tierra cubriendo toda la superficie terráquea. Son necesarios como mínimo 4 de estos satélites para poder calcular, por medio de la triangulación la posición del dispositivo.

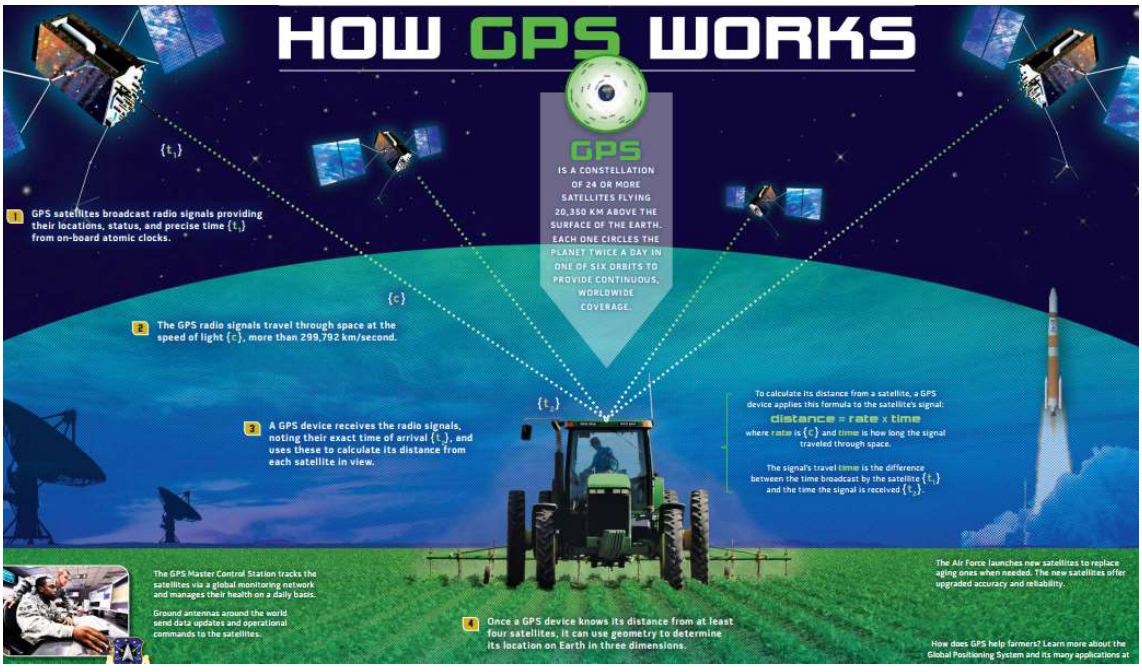


Ilustración 1. Funcionamiento del sistema GPS. http://www.gps.gov/multimedia/poster/poster-web.pdf

2.1.3. Compás Android

La plataforma Android dispone de dos sensores que permiten conocer la posición del dispositivo, estos son: el geomagnético y el acelerómetro.

Estos sensores devuelven unas colecciones de datos haciendo referencia a la posición tridimensional del dispositivo.

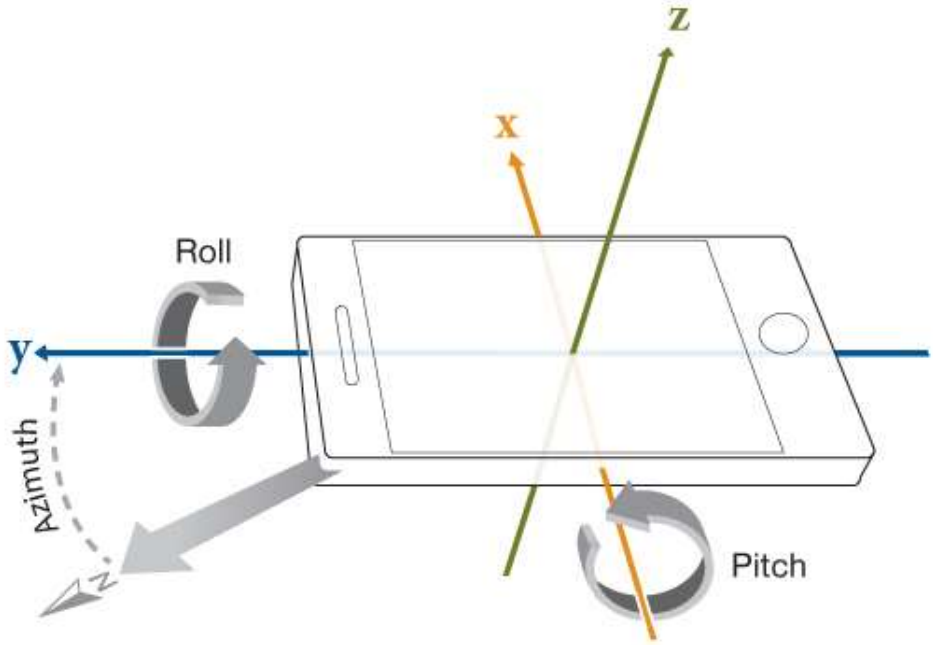


Ilustración 2. Dimensiones de los ángulos de los sensores. https://www.mathworks.com/matlabcentral/mlcdownloads/downloads/submissions/40876/versions/8/screenshot.jpg

2.1.4. OpenCV

La librería OpenCV es un conjunto de herramientas de código abierto, está dirigida al tratamiento de imágenes por computador. Está distribuida bajo la licencia BSD, por lo que la hace muy interesante a la hora de realizar desarrollos de aplicaciones comerciales.

Los algoritmos contenidos en esta librería permiten el reconocimiento de rostros, detección de figuras, etc.... todo realizado sobre video o imagen estática. Estos algoritmos están codificados en diferentes lenguajes de programación, como son C++, C, Python, Java o Mathlab, y es soportado por diferentes sistemas operativos como, Windows, Mac OS o Android.

2.1.5. Github

Debido a la nula experiencia en este tipo de desarrollos, optamos por utilizar un gestor de versiones, el cual nos ofrece una cierta tranquilidad frente a la propagación de errores, estos errores pueden y serán producidos por las sucesivas versiones en las que está

dividido el proyecto. Por lo tanto, optamos por Github, el cual basándose en la herramienta de control de versiones de forma local Git, nos ofrece también un repositorio en la nube, donde poder gestionar las versiones que iremos produciendo, y además, disponer de una copia de seguridad en todo momento del estado del proyecto.

2.2. Aplicaciones existentes.

En el mercado podemos encontrar aplicaciones con funcionalidades muy similares, basándose estas en el posicionamiento GPS del dispositivo y el valor del compás magnético para realizar una consulta a su base de datos. Finalmente, el uso de la realidad aumentada en muchas de ellas forma parte de la interfaz de usuario, y no permite la interacción con ella.

2.2.1. Criterios de elección de aplicaciones antecedentes.

Para este proyecto buscamos y analizamos aplicaciones similares, los criterios seguidos para su elección son los siguientes:

- Aplicación disponible para sistemas Android.
- Disponible en Play Store
- Instalación gratuita o con compras integradas.
- Uso de la realidad aumentada.

2.2.2. Aplicaciones elegidas para su estudio.

Existen un buen número de herramientas con la misma funcionalidad principal, por lo que nos centramos en las siguientes:

2.2.2.1. Puntos geodésicos.

Ilustración 3. Logo Puntos geodésicos.
https://play.google.com/store/apps/details?id=com.simple_and_basic.geodesic_points&hl=es

Esta aplicación se basa en la herramienta Layar de realidad aumentada, la herramienta Layar aporta una superficie de realidad aumentada donde poder insertar las capas que desee el desarrollador, de este modo en la capa que añaden se encuentran las alturas y los nombres de los picos que se encuentran en la dirección hacia donde apunta el dispositivo.

Su funcionamiento no es demasiado correcto, puesto que no dan a conocer la ubicación exacta del pico, ni se muestra cerca del pico los datos relacionados con él. Esta aplicación recibe los datos de una base de datos de picos reducida, al menos en la zona donde se realizan las pruebas, por lo que no permite conocer la altura de cualquier lugar. No tiene sistema de disparo, por lo que los datos que se muestran no pueden ser elegidos por el usuario.

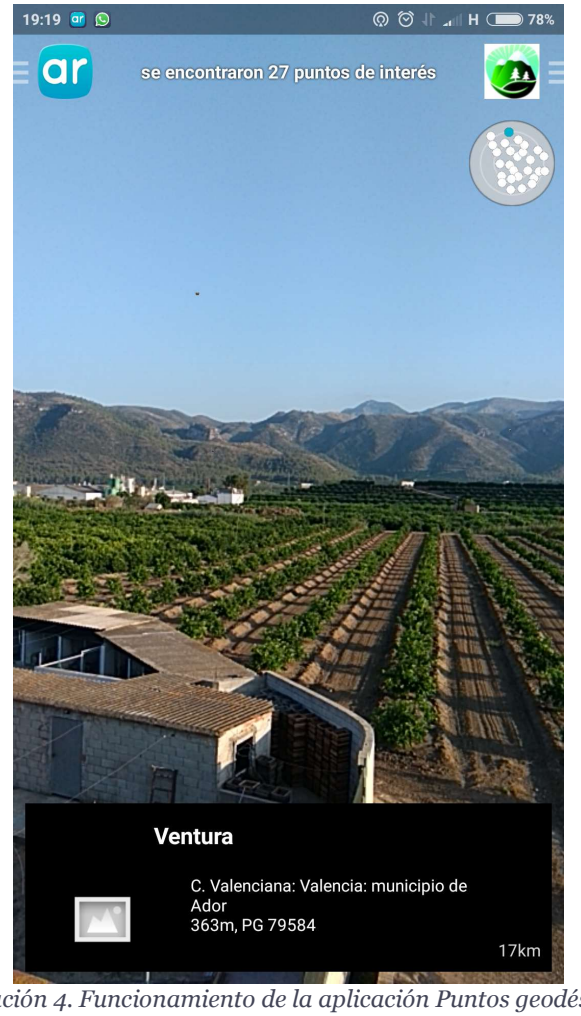


Ilustración 4. Funcionamiento de la aplicación Puntos geodésicos.

2.2.2.2. PeakAr.

Ilustración 5. Logo de PeakAr.
https://lh6.ggpht.com/MMlx2ktnAzN7F4_6TyF8Lj8Su7WrmSNi8NS7AxZlDhqMBSOCzTqrsyfcv6skYvPx15AG=w300

Esta aplicación la encontramos en el repositorio de aplicaciones Google Play, está desarrollada por Salzburg Research Forschungsgesellschaft mbH de Austria. Está basada en realidad aumentada y recibe los datos de los nombres de los picos por medio de consulta a base de datos, aunque como en el caso anterior en el lugar de las pruebas tiene una colección de datos limitada.

Ilustración 6. Funcionamiento de la aplicación PeakAr.

No es necesaria conexión de datos para el funcionamiento de la aplicación, lo que supone un punto a favor. Como puntos desfavorables encontramos la ausencia de sistema de disparo, por lo que siempre muestra los picos que están a la vista, tanto si son visibles por parte del usuario como si están escondidos tras otro pico menor, su base de datos (en el lugar de las pruebas) es pobre y su error a la hora de situar el pico resulta considerable, como podemos ver en la imagen de posicionamiento del pico.

2.2.2.3. World Summits.

Ilustración 7. Logo de World Summits.
https://lh5.ggpht.com/EMS9NkUIsUY_7qLVLYIFPz8WD8Uj7rJSv6cEi4jcFAmH8Vm65llNsZQSS8L-62qH_6Q=w300-rw

Esta herramienta está desarrollada por Ramlabs y se encuentra disponible en GooglePlay con descarga gratuita. Es una aplicación muy completa, que además de disponer del servicio de identificación de picos y alturas, ofrece la posibilidad de grabar las rutas para después compartirlas en su página web. También dispone de buscador de puntos de interés en la montaña, como pueden ser refugios, rutas y vivacs.

Su base de datos es mundial y en la zona de pruebas bastante completa, dispone de disparador por lo que suponemos un consumo de recursos más eficiente. Como puntos negativos podemos observar la publicidad que se encuentra situada dificultando la lectura de la distancia hasta el pico, y el no ofrecer la altura de cualquier lugar.

2.2.2.4. Find the Hill.

Ilustración 9. Logo de Find the Hill.
https://lh3.ggpht.com/kmibhp5xYUgpr3Ozb4GMfIlixGCfoJjWAGX6BrO378w7QJ09a5IkPgQRA_q0HftL9nK=w300-rw

Esta herramienta, desarrollada por Starbug software development, a nuestro parecer es la que mejor funcionamiento tiene. Encontramos una interfaz limpia, con ausencia de multitud de opciones que es de agradecer. Esta aplicación se basa solamente en realizar bien su cometido que es identificar los picos y sus cotas. Para ello utiliza un sistema de consulta sobre base de datos y muestra sobre pantalla por medio de la realidad aumentada, la información relevante de los picos que se encuentran situados hacia donde se orienta la cámara.

Es interesante el uso de los filtros para poder desgranar entre toda la información de picos que nos muestra, puesto que tiene un amplio catálogo de referencias. No dispone de disparador, pero identifica correctamente la posición de los picos. Como punto negativo al igual que las demás aplicaciones estudiadas, puede conducir a error puesto que indica la altura y nombre de picos que se pueden encontrar ocultos tras algún monte de menor altura.

3. Análisis.

3.1. Descripción de la solución propuesta

Vamos a describir paso a paso como resuelve el propósito del proyecto la aplicación diseñada:

En primer lugar, mostramos al usuario las imágenes que está capturando la cámara del dispositivo, para de este modo, poder enfocar al pico del cual desea conocer la altura. Para este menester, utilizamos la librería OpenCV, la cual nos brinda unas funciones ya implementadas por las cuales capturamos la salida de la cámara, y tras ajustar el tamaño y la posición, mostramos por pantalla el resultado.

Acto seguido, definimos un marco más pequeño dentro de la imagen principal, donde el usuario situará la imagen del pico. Este paso lo realizamos al reducir el espacio de trabajo de reconocimiento de formas, y de este modo, conseguir un ahorro de recursos en el dispositivo. Realmente todo el análisis de imagen será realizado sobre este submarco de la pantalla principal, quedando de forma transparente para el usuario este funcionamiento.

Una vez el sistema muestra las imágenes, lanzamos el algoritmo de detección de bordes para poder analizar el contorno y así, tras aplicar el algoritmo de detección que nos indica que la cámara esta fija en el punto más alto visible, lanzar los algoritmos de consulta.

Una vez lanzados los algoritmos, consultamos las alturas al servicio de Google Elevation API, mediante el algoritmo de detección de altura máxima visible, obtenemos la altura y las coordenadas del punto más alto visible y pasamos a realizar la consulta al Instituto Geográfico Nacional para comprobar si existe un nombre asociado a estas coordenadas.

De forma paralela a la consulta de nombres, mostramos por pantalla la distancia a la que se encuentra el punto más alto, calculamos la distancia en ruta por medio de cálculos trigonométricos básicos y mostramos el perfil de esta altimetría.

3.2. Diagrama de flujo

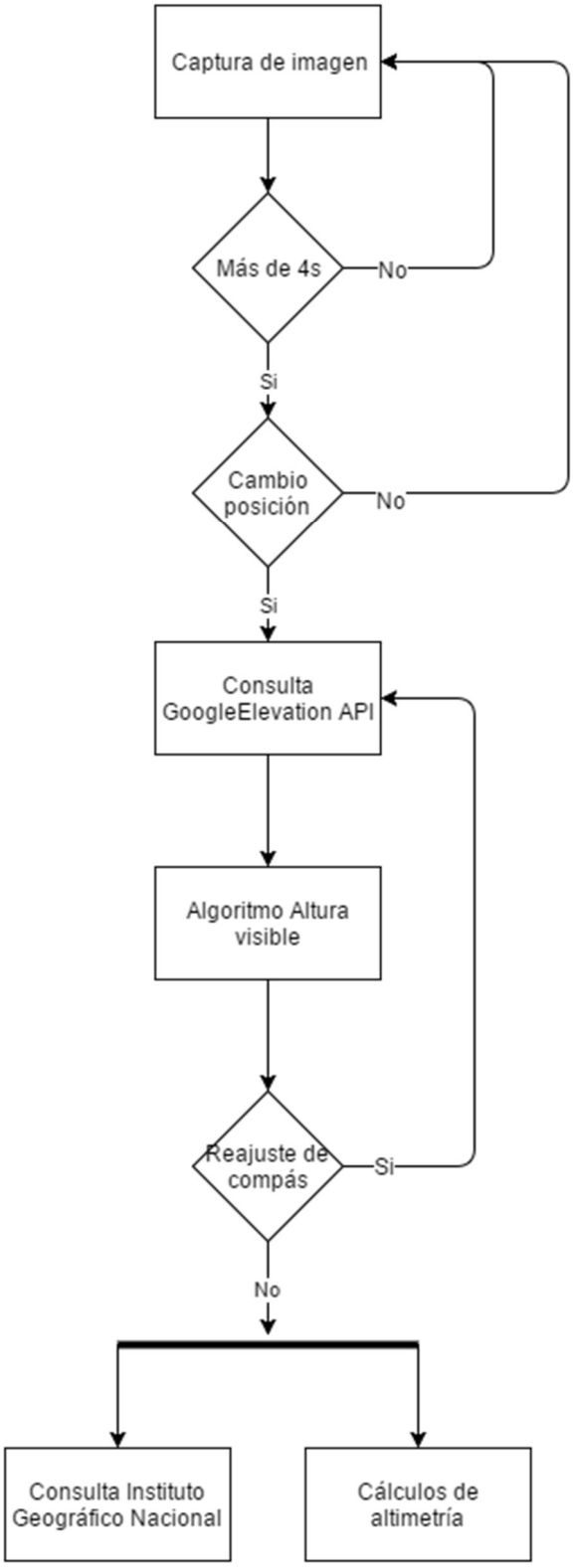


Ilustración 10. Diagrama de flujo de la aplicación.

3.3. Análisis de requisitos.

Vamos a desgranar los objetivos previamente indicados, para ir sacando a la luz los requisitos para este proyecto.

3.3.1. Requisitos funcionales.

Obtención de la altura del accidente geográfico enfocado, mostrando sobre impresa la altura en la imagen en tiempo real del pico.

- Consultar con el servicio Google Elevation API para obtener la altura asociada al punto en cuestión.

- Analizar la imagen para mostrar sobre impresa la altura.

Consulta de la toponimia asociada al lugar, en caso de estar disponible, mostrado sobreimpreso en tiempo real.

- Obtener el nombre del pico, si estuviera disponible, desde la página web del Instituto Geográfico Nacional.

- Si lo hubiese, mostrar por pantalla el nombre del pico.

Distancia geográfica.

- Calcular la distancia en metros desde la ubicación del dispositivo, hasta el pico, medida en línea recta hasta la intersección con el plano perpendicular en dicho punto.

- Mostrar la distancia geográfica.

Distancia en ruta.

- Calcular la distancia en ruta desde la ubicación del dispositivo hasta el pico.

- Mostrar la distancia en ruta en metros.

Gráfico de altimetría.

- Mostrar el gráfico de altimetría asociado a la distancia en ruta hasta el pico.

Ajuste del compás magnético.

- Cálculo del error del compás magnético al determinar el ángulo de la cámara respecto el Norte magnético.

Realidad aumentada.

- Uso de la realidad aumentada para disparar las consultas a los servicios.
- Cálculo de la posición exacta del pico para realizar el ajuste del compás.

3.3.2. Requisitos no funcionales.

Vamos a ver ahora los requisitos no funcionales asociados al proyecto.

- Rendimiento: La herramienta debe realizar su tarea de una forma fluida.
- Optimización: El consumo de recursos, debe ser mínimo, en cualquier caso.
- Integración: La aplicación debe ser ejecutada sobre dispositivos con sistemas operativos Android.
- Mantenibilidad: El diseño y la implementación deben de dar facilidades a su mantenimiento y mejora.

3.4. Casos de uso.

Los casos de uso que se extraen de la especificación de requisitos indicada anteriormente son los siguientes:

- **Consultar altura:** Consultar con el servicio de Google Elevation API las altimetrías.
- **Obtener coordenadas:** Captura de coordenadas GPS del dispositivo.
- **Obtener ángulo:** Obtener el ángulo respecto el norte.

- **Consultar nombre:** Consulta a la web del Instituto Geográfico Nacional para obtener el nombre del pico.

- **Cálculo distancia geográfica:** Calcular la distancia geográfica en metros desde el usuario hasta las coordenadas del punto más alto visible.

- **Cálculo de distancia en ruta:** Calcular la distancia en ruta en línea recta que hay desde el usuario hasta el punto más alto visible.

- **Calcular altimetría:** Preparación de los datos para mostrar la gráfica de altimetría.

- **Cálculo desviación del compás:** Calcular si existe alguna altura superior con desplazamientos de 1º a ambos lados de la cota máxima.

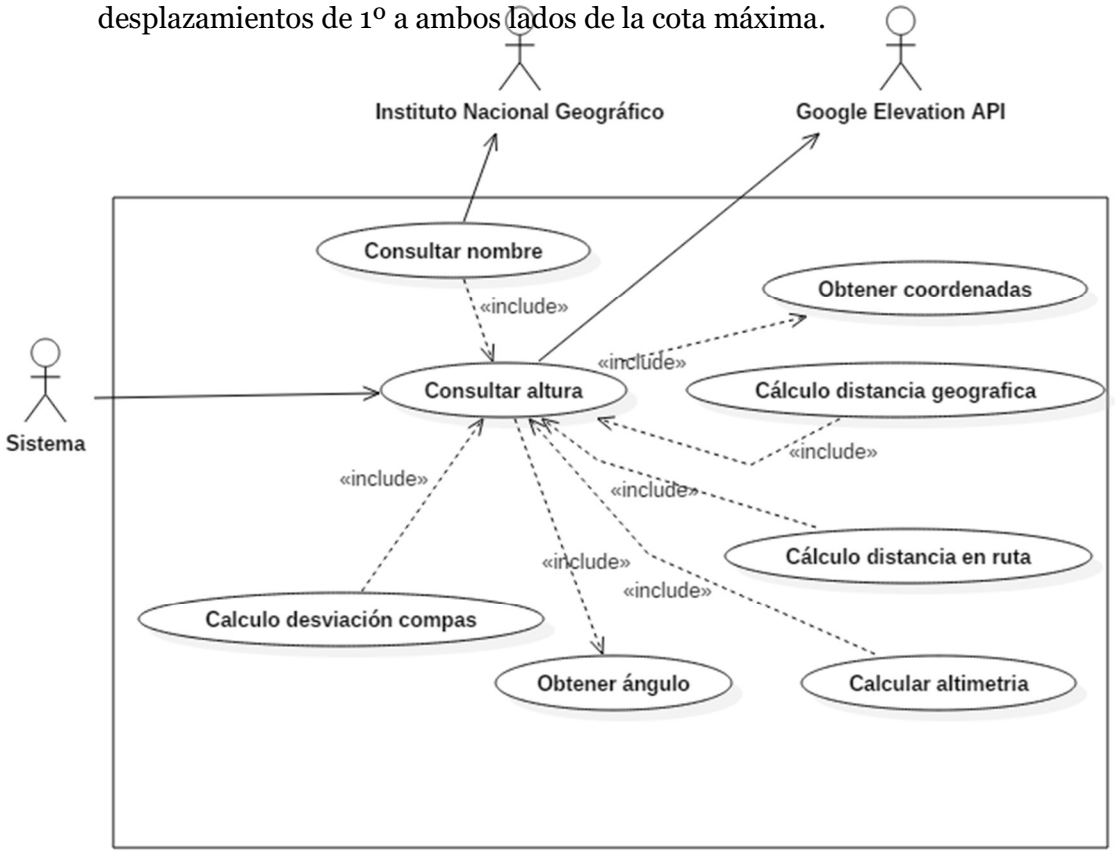


Ilustración 11. Casos de uso.

3.5. Diagrama de clases

El diagrama de clases que describe las entidades del proyecto es el que sigue:

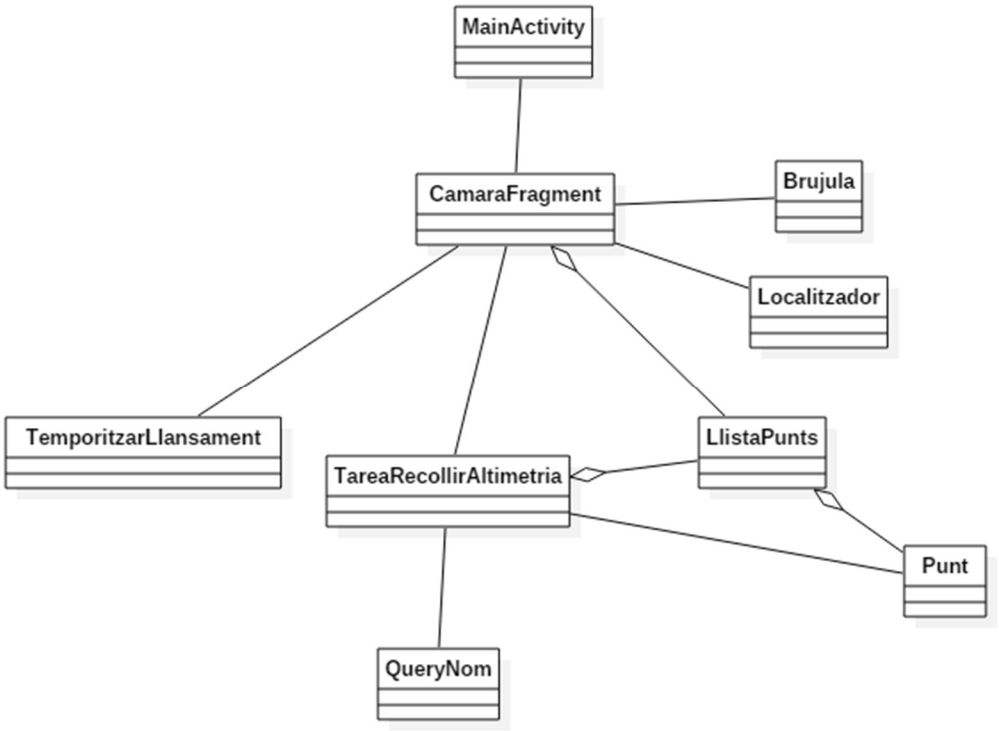


Ilustración 12. Diagrama de clases.

3.6. Riesgos

Los riesgos encontrados han sido una multitud, debida a la poca experiencia en los campos, lo cual nos ha llevado a realizar numerosos estudios en todos los problemas enfrentados.

3.6.1. Mostrar la imagen de la cámara.

Nos encontramos ante la dificultad de mostrar la imagen capturada por la cámara del dispositivo en tiempo real, decidimos asignarle un riesgo Alto. Para el caso de la toma de instantáneas encontramos mucha y diversa información, pero para el caso de mostrar la imagen sin ningún tratamiento fue más difícil, esta parte del desarrollo finalmente fue más laboriosa que complicada por la poca información, puesto que, para poder mostrar la imagen, en el formato y forma elegido, tuvimos que utilizar el método de prueba y error.

Los métodos utilizados devolvían una imagen distorsionada en cuanto a relación de aspecto y respecto al ángulo de visión. Tras varias pruebas, conseguimos intuir, para

después confirmar el funcionamiento de las funciones y así poder ofrecer una visión más real.

3.6.2. Capturar la posición GPS.

En este caso identificamos el riesgo como bajo, puesto que había abundante información sobre la captura de la posición por medio del sensor GPS del dispositivo. Definitivamente así fue, puesto que esta funcionalidad fue implementada con rapidez.

3.6.3. Obtención del ángulo respecto al norte.

Este paso había sido etiquetado como de riesgo bajo, habíamos encontrado información al respecto, y parecía muy accesible, nada más lejos de la realidad. Hemos encontrado gran cantidad de problemas con el ángulo devuelto por las funciones que obtienen el valor a partir del sensor, hemos tenido que contrarrestar el ángulo Pitch del dispositivo, puesto que interferiría en la lectura del ángulo Azimut, también hemos encontrado dificultad debido a las interferencias del lugar de desarrollo.

3.6.4. Cálculo del segmento a consultar.

Este cálculo, asignado a un riesgo Medio, fue un poco complicado de salvar, puesto que, aunque no existe abundante información, sí que pudimos encontrar suficiente información de cómo realizar el cálculo, el problema fue la multitud de algoritmos para realizar el cálculo de la posición final, y el poder acertar con las unidades necesarias para las variables representadas en los cálculos.

3.6.5. Consulta de altimetría.

Esta función ha sido de las más sencillas, puesto que contábamos experiencia en el uso de servicios externos y en el tratamiento de resultados. Si bien cabe comentar que la única dificultad fue el conseguir afinar las consultas con el número de muestras a pedir al servicio. Por lo tanto, le asignamos un riesgo Bajo y acertamos.

3.6.6. Identificar el contorno de la imagen

En este caso el riesgo era Alto debido a la falta de experiencia en la cuestión, incluso haciendo acopio de información, esta parte necesita de un afinado de las funciones, que hacen necesaria la técnica de prueba y error para conseguir los resultados esperados.

3.6.7. Consulta al Instituto Geográfico Nacional.

Debido a la naturaleza de la página, puesto que no es un servicio web, hay que analizar el código HTML devuelto para poder extraer el resultado, gracias a experiencias anteriores con este tipo de algoritmos le asignamos un riesgo bajo.

3.6.8. Transformación de coordenadas

En este punto creíamos que no presentaría dificultad y asignamos un riesgo Bajo, sin embargo, la falta de documentación, pues en todos los lugares visitados para su consulta se trataban de herramientas online de transformación de coordenadas, ha sido su mayor dificultad. Existe una diferencia entre los formatos de los datos devueltos por el servicio de Google Elevation API y los necesarios para el uso de la página web del Instituto Geográfico Nacional.

4. Diseño.

4.1. Descripción de los cálculos y algoritmos de la aplicación.

Debido a la naturaleza de la aplicación, hemos utilizado algoritmos ya implementados en librerías, otros que hemos adaptado al lenguaje de programación Java y otros que son el núcleo propio de la aplicación.

4.1.1. Algoritmo Canny.

Este algoritmo se basa en tres criterios:

- La detección de bordes, evitando el borrado de bordes importantes y el proporcionar falsos bordes.
- La localización, este criterio establece que la distancia entre la posición detectada y la real debe ser mínima.
- Una respuesta, se deben integrar en una única respuesta todas las respuestas sobre un único borde.

Una vez introducidos los criterios utilizados, veamos ahora los pasos que describe el algoritmo. Primeramente, se aplica un desenfoque gaussiano a la imagen, con la intención de suavizar la imagen y así poder desechar cualquier ruido existente en el original.

Acto seguido se obtiene el gradiente, en el cual, por medio del uso de la primera derivada, obtenemos todos los píxeles donde hay un cambio de intensidad. Ahora cabe adelgazar estos contornos detectados para dar el mínimo grosor, para después aplicar la histéresis de umbral, en este proceso se analiza punto por punto tomando la orientación de los puntos, para ver si está dentro del mínimo y el máximo definidos en el umbral.

Por último, queda cerrar los contornos que pudieran haber quedado abierto a causa del ruido.

4.1.2. Algoritmo lanzamiento consulta.

Para poder lanzar las consultas en el momento óptimo, esto es, cuando el punto más alto visible del pico está situado en el centro de la imagen, y no hay ninguno cerca más alto, utilizamos el siguiente algoritmo.

Calculamos el punto más alto de cada contorno devuelto por la función Canny, aplicamos unos márgenes por el posible movimiento de la mano, si tras 3 segundos, todavía sigue en los márgenes, se aceptan los parámetros y se lanza la consulta.

4.1.3. Cálculo de coordenadas destino.

En este cálculo, obtenemos las coordenadas de destino, dada una posición, un ángulo y un desplazamiento. Utilizamos este cálculo para poder obtener el segmento de consulta de altimetría.

$$\varphi_2 = \operatorname{asin}(\sin \varphi_1 \cdot \cos \delta + \cos \varphi_1 \cdot \sin \delta \cdot \cos \theta)$$

$$\lambda_2 = \lambda_1 + \operatorname{atan2}(\sin \theta \cdot \sin \delta \cdot \cos \varphi_1, \cos \delta - \sin \varphi_1 \cdot \sin \varphi_2)$$

Donde φ es la latitud, λ es la longitud, δ es la distancia angular sobre el radio de la tierra y θ es el ángulo en sentido de las agujas del reloj desde el Norte.

4.1.4. Algoritmo del punto más alto visible.

Este algoritmo nos servirá para obtener el punto sobre el que trabajará la aplicación, descartando las cotas más altas, pero no visibles desde la posición del usuario. El funcionamiento realmente es muy simple, una vez tenemos la colección de datos devuelta por Google Elevation Api, obtenemos el punto más alto de la colección, esta sería en el mejor de los casos el punto más alto visible. Calculamos el ángulo entre la visión del usuario y esta altura, y con esto sabremos a partir de que altura se pierde la visión.

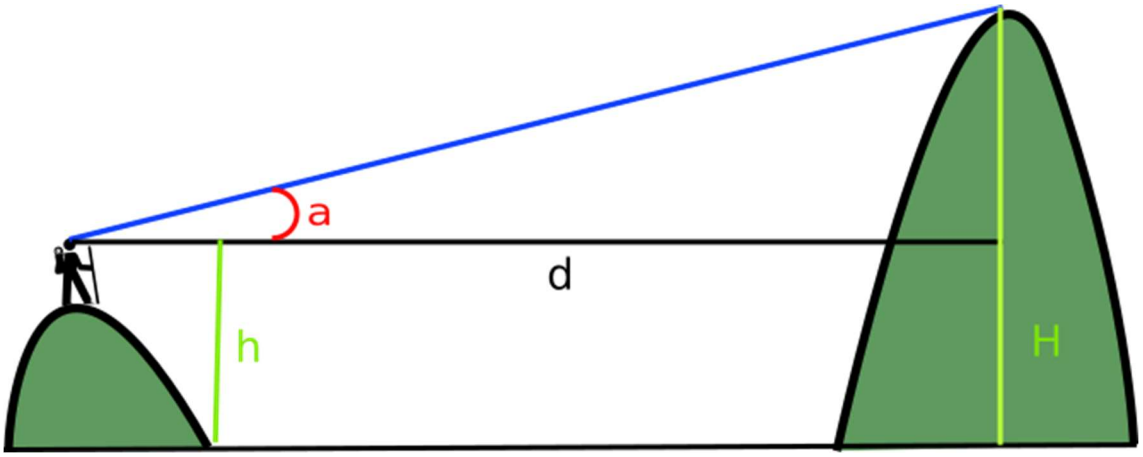


Ilustración 13. Cálculo del ángulo visible respecto a una altura.

Por lo que recorremos todos los puntos en orden decreciente hasta encontrar un punto que sobrepase esta altura, este será nuestro nuevo punto más alto visible.

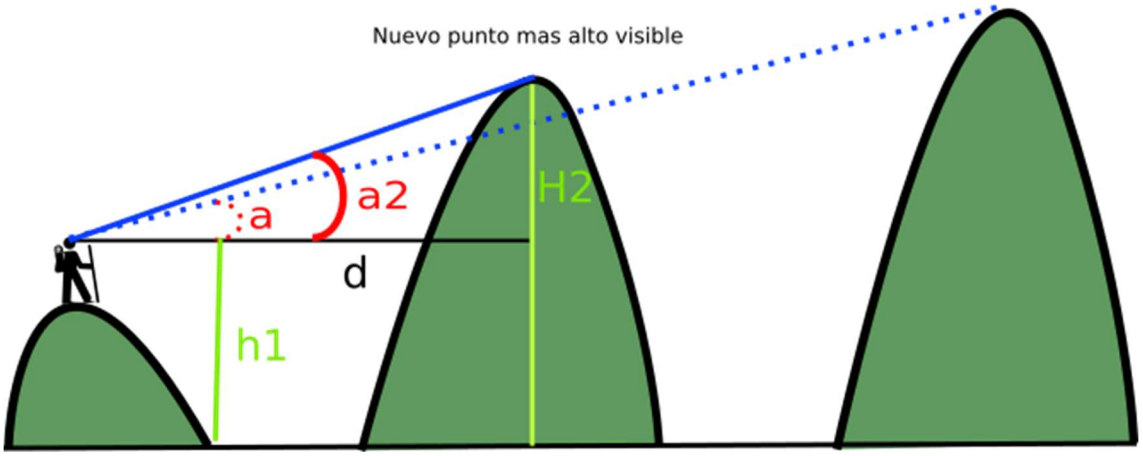


Ilustración 14. Cálculo del nuevo ángulo visible.

Mostramos ahora el diagrama de flujo de este algoritmo.

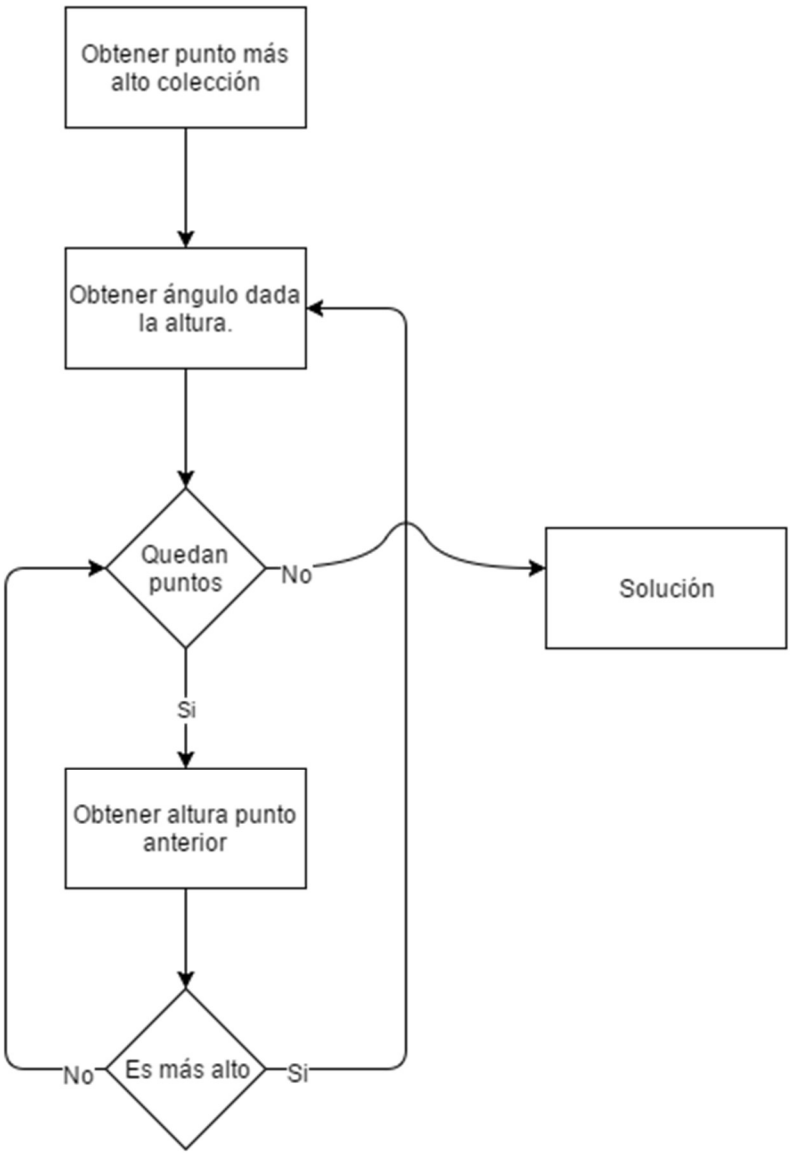


Ilustración 15. Diagrama de flujo del algoritmo Altura Visible

4.1.5. Algoritmo de ajuste de compás.

Para poder ajustar el ángulo que devuelve el compás magnético, realizamos consultas aumentando y disminuyendo el valor del ángulo para encontrar, dentro de un margen, el valor de altura más alto posible.

Empezamos realizando la consulta de altimetrías a tres segmentos diferentes, estos segmentos son los siguientes, el primero corresponde a restar un grado al ángulo original, el segundo corresponde con el ángulo original y en el tercero sumamos un grado al original. Comparamos los resultados, y si el valor más alto es el ángulo original, damos por terminado el algoritmo, en caso contrario aumentamos o disminuimos en 2 grados el ángulo que ha devuelto el valor máximo. Así sucesivamente hasta 4 iteraciones, que sería el margen establecido.

Cabe destacar el aumento de 2 grados para no repetir consultas innecesarias, y el uso de una variable para conocer si venimos de aumentar el ángulo y el resultado nos indica que hay que disminuirlo de nuevo, esto nos indica el resultado óptimo y también terminaría el algoritmo.

4.2. Prototipo de la interfaz gráfica de usuario.

Para esta versión de la herramienta solo contamos con la información mostrada por pantalla en tiempo real, por lo que diseñamos una interfaz que dé cabida a toda la información resultante del análisis de altimetría.

Decidimos mostrar impresa sobre las imágenes de la cámara, la altura, el nombre si lo hubiere y el gráfico de altimetría. Dejando para la parte baja de la interfaz los datos más dinámicos como son la posición GPS, el ángulo del compás magnético y las distancias al objetivo.

Ilustración 16. Prototipo de interfaz gráfica de usuario.

5. Implementación.

Pasamos ahora a detallar el proceso llevado a cabo para la implementación de la aplicación. Hemos creído interesante el enumerar los riesgos encontrados durante el análisis del proyecto, para acto seguido presentar las soluciones propuestas y el resultado de estas.

También indicaremos las sucesivas versiones que hemos ido generando en el transcurso de la implementación del proyecto, puesto que encontramos que es una buena metodología de trabajo.

5.1. Versiones

Hemos realizado un desarrollo basado en versiones iterativas, en las cuales en cada iteración se entregaba una versión de la aplicación en la que se añadía una nueva función, la cual servía para el propósito final de la herramienta, esto es debido a la poca, por no decir nula experiencia del desarrollador en el campo de los dispositivos Android y de la Realidad Aumentada. Por lo que se creyó recomendable realizar un desarrollo “paso a paso” consiguiendo las metas marcadas y dando solución a los riesgos identificados durante el análisis.

5.1.1. Versión 0.1

En esta versión mostramos la captura de la cámara en tiempo real, lo hacemos en el formato elegido que es el de mantener el dispositivo en horizontal. Hemos conseguido con el método setMaxFrameSize() de la clase CameraBridgeViewBase, tener el aspecto deseado para la aplicación, puesto que teníamos la cámara girada 90º y solo empleaba una parte de la pantalla.

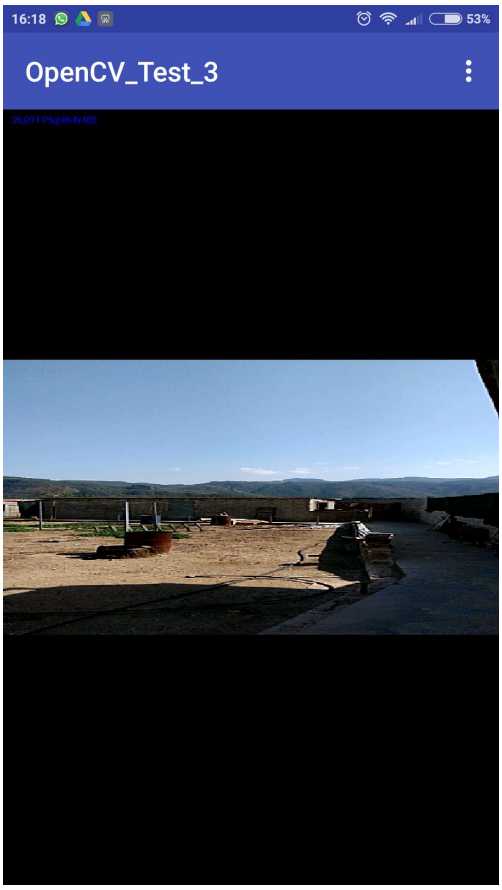


Ilustración 17. Primera captura de la cámara.

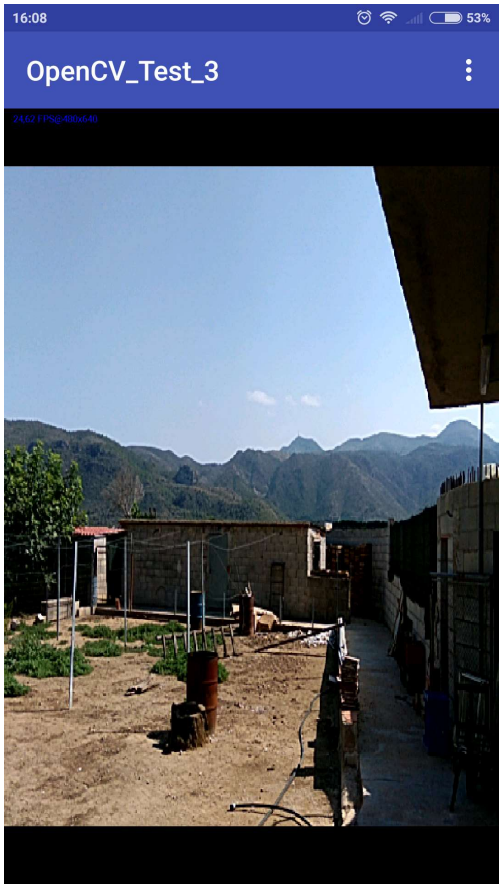


Ilustración 18. Ajuste de dimensiones.

5.1.2. Versión 0.1.2

En esta versión manteniendo la visión de la cámara, añadimos la localización GPS y mostramos su resultado por pantalla. Podemos observar como abajo a la derecha sobre un fondo rojo aparece la latitud y la longitud en coordenadas geográficas decimales. Estos datos los obtenemos de la clase Location por medio de los métodos getLatitude() y getLongitude().

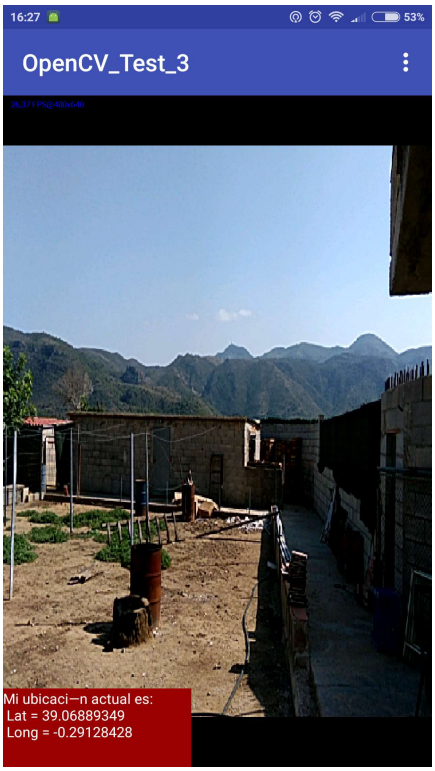


Ilustración 17. Ya se puede observar la posición GPS.

5.1.3. Versión 0.2

En esta iteración le sumamos el compás magnético, en este caso, hay que acceder a los sensores Sensor.TYPE_ACCELEROMETER y Sensor.TYPE_MAGNETIC_FIELD. Tras aplicar el método SensorManager.getOrientation() a la matriz devuelta por SensorManager.getRotatitonMatrix(), utilizamos el valor de la posición 0, que es el correspondiente al Azimut.

Aquí todavía existe el error referente a la inclinación del dispositivo, arrojando diferentes valores dependiendo de si el dispositivo está orientado hacia abajo o hacia arriba.

Ilustración 19. Orientado hacia abajo.

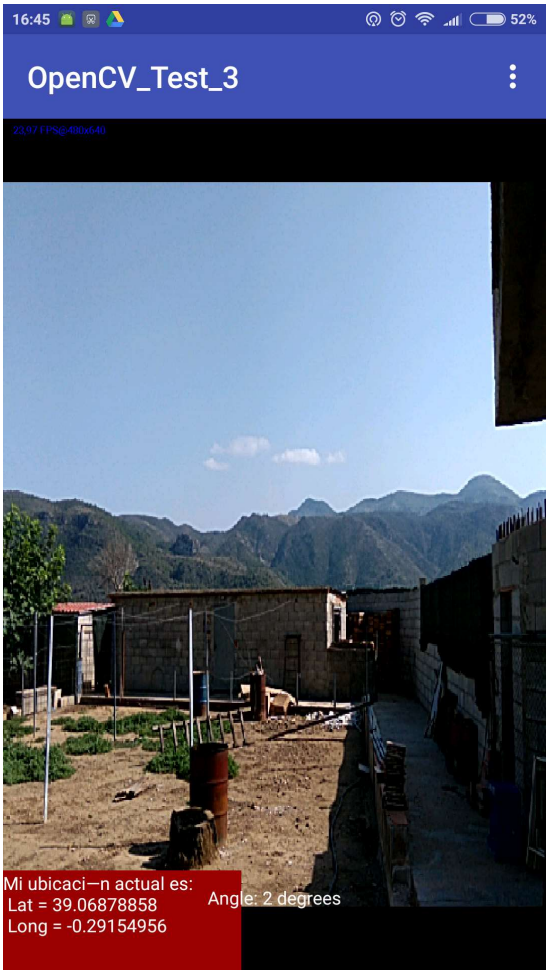
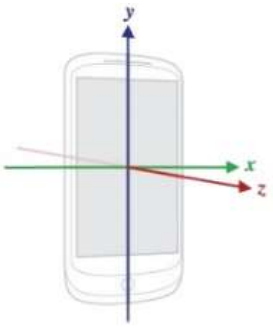


Ilustración 18. Orientado hacia arriba.

Podemos observar como en la primera instantánea el ángulo capturado es 151 grados, y como sin variar prácticamente el ángulo, pero orientando el dispositivo hacia arriba el ángulo pasa a ser de 2 grados.

Mediante el uso de la función SensorManager.remapCoordinateSystem() para transformar las coordenadas obtenidas de acuerdo a la matriz de rotación, obtenemos el ajuste del ángulo respecto el norte independientemente de la orientación vertical del dispositivo.



values[0]: Azimuth, angle between the magnetic north direction and the y-axis, around the z-axis (0 to 359). 0=North, 90=East, 180=South, 270=West
values[1]: Pitch, rotation around x-axis (-180 to 180), with positive values when the z-axis moves toward the y-axis.
values[2]: Roll, rotation around y-axis (-90 to 90), with positive values when the x-axis moves toward the z-axis.

Ilustración 20. Valores de la colección de valores de getOrienteation(). http://image.slidesharecdn.com/gettingstartedwithandroid-devandtestperspective-110613072000-phpapp02/95/getting-started-with-android-dev-and-test-perspective-47-728.jpg?cb=1308890551

5.1.4. Versión 0.3

En esta iteración definimos el segmento que consultaremos a Google Elevation API, dada la posición del dispositivo y el ángulo referente al Norte, dibujaremos una línea imaginaria de 25Km y calcularemos las coordenadas finales del segmento.

La elección de esta medida es la óptima respecto al error introducido por el servicio de altimetrías, puesto que a mayor distancia mayor margen de error. Observamos que un pico a 25Km es lo suficientemente pequeño como para que un usuario no pueda enfocarlo con garantías.

5.1.5. Versión 0.4

Estamos ya preparados para realizar la consulta al servicio de Google Elevation API. Primeramente, es necesario registrarse como usuarios del sistema, el cual tiene una cota de uso para los usuarios gratuitos de 250 consultas diarias, esta es una de las razones por las que intentaremos realizar las menores consultas accidentales por parte del usuario. Nos proporcionan en su documentación una estructura de Query para poder realizar las consultas, en ella elegiremos las coordenadas iniciales del segmento, las coordenadas finales, el número de muestras a tomar a distancias equitativas en el segmento y el formato de la respuesta.

Elegiremos siempre el máximo de muestras posibles que está fijado en 500, esto es para tener la máxima exactitud, y el formato JSON que trataremos y guardaremos en una colección para su posterior análisis.

5.1.6. Versión 0.5

Una vez tenemos los datos de altimetría podemos pasar a calcular el punto más alto de la colección. Cabe destacar el formato de los datos devuelto por la consulta, cada dato de altimetría es una estructura formada por los siguientes datos:

- Latitud.
- Longitud.
- Resolución.
- Altitud.

La colección viene dada por orden de distancia de menor a mayor desde el inicio al fin del segmento. Por lo que resulta fácil recorrer la colección para determinar el punto más alto.

5.1.7. Versión 0.6

Vamos a calcular ahora el punto más alto visible por el usuario. Tomando como primer punto el punto más alto de la colección, recorreremos en orden inverso la colección haciendo uso del algoritmo del punto más alto visible. De este modo al finalizar el recorrido, habremos obtenido el punto más alto visible y tendremos además sus coordenadas y su distancia hasta el usuario.

5.1.8. Versión 0.7

En esta iteración, vamos a capturar el contorno del pico a analizar, así como a dibujar los márgenes que servirán para el disparador.

Empezamos definiendo un pequeño marco que será donde se ejecutará el análisis de contorno, de este modo evitamos el uso de recursos innecesarios puesto que solamente nos interesa la parte superior del pico, y la parte inferior está ocupada por la gráfica de altimetría.

Una vez tenemos definido el marco, aplicamos el algoritmo de Canny a este segmento de pantalla para poder realizar la captura de contornos. Dibujamos también un marco de color rojo, que serán los márgenes que delimitarán la zona de disparo de la consulta.

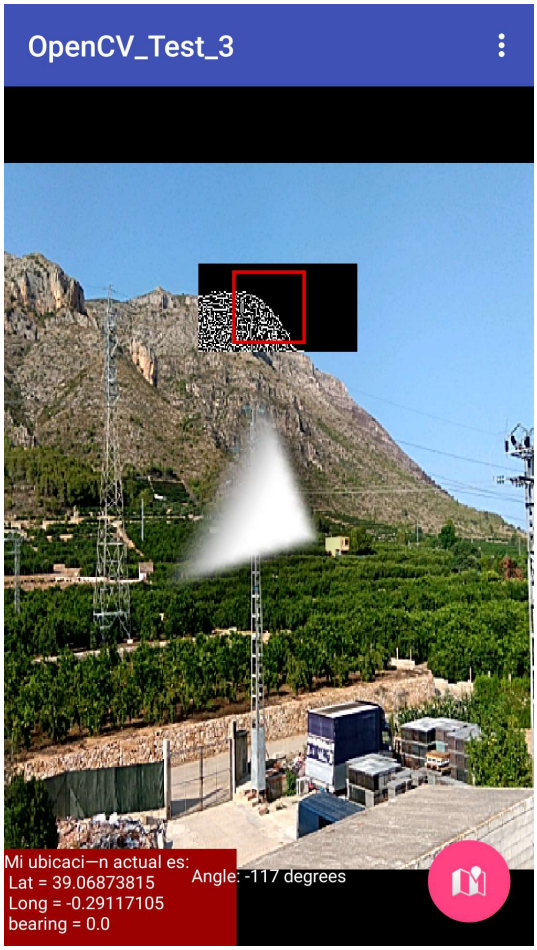


Ilustración 21. Captura del funcionamiento del algoritmo de Canny.

5.1.9. Versión 0.8

En esta iteración mostramos por pantalla la altura relacionada con el punto más alto visible. Esto es posible ya que en estos momentos no es el sistema el que lanza la consulta, sino el propio usuario por medio de un botón. De este modo se puede probar la aplicación aunque no se encuentre enfocada la cumbre de una montaña. Todavía no se puede calibrar el compás magnético, pero si conocer la altura de cualquier montaña frente al usuario, aunque no sea la cota más alta visible. En este momento estudiamos la posibilidad de dejar el botón para añadir la funcionalidad.

5.1.10. Versión 0.9

En esta versión mejoramos el uso de la brújula pues presenta valores negativos. Añadimos un condicional, donde a valores negativos los convierte en positivos según el valor arrojado por la función.

5.1.11. Versión 0.10

Es en esta iteración cuando comenzamos a tratar los contornos de la imagen para poder conocer si el punto más alto de la montaña está dentro de los márgenes. Primeramente, gracias al algoritmo de Canny tenemos los contornos dibujados, de este modo la función findContours() de la clase Imgproc nos devuelve una colección con los puntos que forman los contornos detectados por el algoritmo de Canny.

Ilustración 22. Puntos más altos de los contornos detectados.

Para conseguir el punto más alto de cada contorno, simplemente recorremos los puntos pertenecientes al contorno y para el punto más alto dibujamos un pequeño cuadrado rojo. Para el más alto de todos, dibujamos un cuadrado de color verde. De este modo podemos comprobar a simple vista el funcionamiento de la aplicación.

5.1.12. Versión 0.11

En esta versión implementamos un sistema para temporizar el lanzamiento de la consulta a Google Elevation API. Debido a la restricción en la cuota de uso, y que el sistema usa los datos del plan de datos del usuario, debemos evitar en lo posible lanzamientos de consulta erróneos. Por medio de este temporizador, nos aseguramos de que el usuario es consciente y ha pasado al menos 3 segundos enfocando al pico al que quiere acceder a la información.

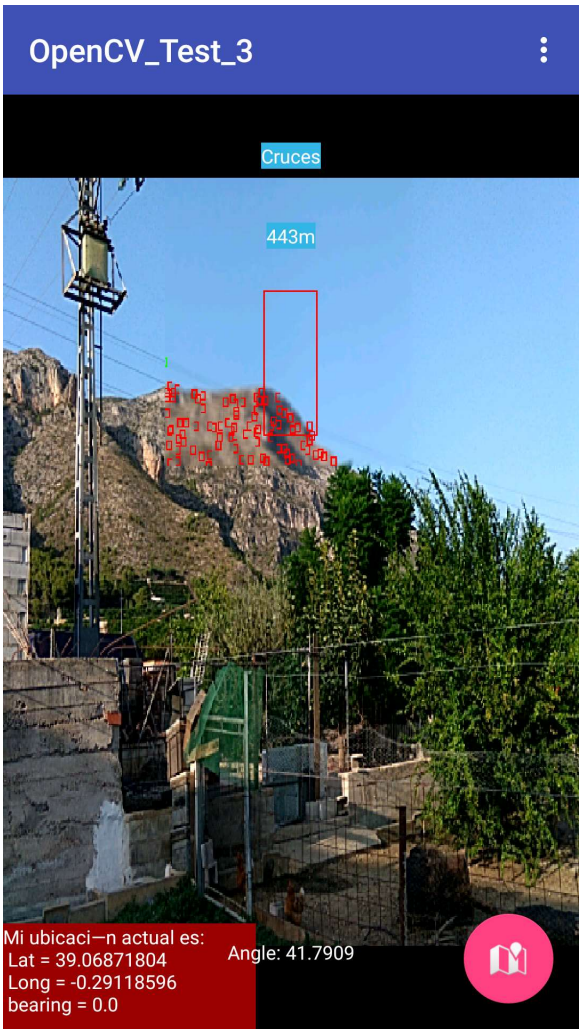


Ilustración 23. Se puede observar el nombre del pico.

5.1.13. Versión 0.12

Llegamos a la iteración donde realizamos la consulta al Instituto Geográfico Nacional para conocer si existe un nombre asociado al pico. Al conocer la posición geográfica del punto más alto visible, preparamos el formulario de consulta del Instituto Geográfico Nacional para conocer si existe un vértice geodésico en esa posición. Para poder extraer el resultado no podemos hacer uso de herramientas ya implementadas, puesto que el resultado es código HTML y habrá que recorrerlo para poder conseguir el nombre en caso que lo hubiese.

5.1.14. Versión 0.13

En esta iteración refactorizamos el código implementado, creando clases y métodos. La mayor parte de la documentación asociada a los problemas que deseamos solucionar tiene el código englobado en la actividad principal, hemos realizado la implementación siguiendo los mismos esquemas que nos muestran estos documentos. Debido a esto se hace necesario, una vez comprobado su funcionamiento, el realizar un trabajo de mantenimiento para refactorizar el código y así conseguir un código más mantenible.

5.1.15. Versión 0.14

Esta es la última iteración y vamos a mostrar la interfaz final. Ahora es el momento de montar la interfaz diseñada en el prototipo, como es norma en Android, la capa de la vista está separada de la lógica de negocio, por lo que solamente modificaremos los archivos que contienen las plantillas con los elementos visuales. Además, añadiremos la funcionalidad de mostrar el gráfico de altimetría desde el usuario hasta el pico analizado.

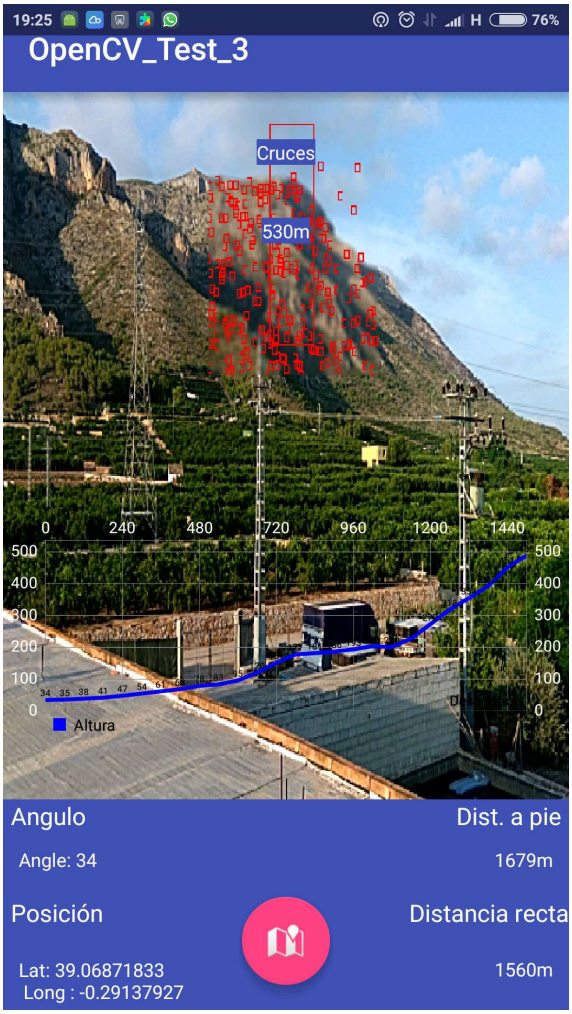


Ilustración 24. Interfaz con toda la información en pantalla.

6. Conclusiones.

Encontramos buenas herramientas, aunque no consiguen (tampoco lo persiguen) la finalidad de este proyecto de abordar la altura de cualquier pico, por lo que en estas aplicaciones solo podemos acceder a las cotas almacenadas en sus bases de datos, si bien cabe remarcar la aplicación FindTheHill, que cuenta con una gran base de datos de alturas y toponimias. Con este proyecto esperamos llenar el nicho existente en la determinación de cualquier altura, aunque no exista en una base de datos. Hemos utilizado la librería OpenCV la cual nos parece una herramienta ideal para el procesamiento de imágenes tanto en imagen estática como en video. Consideramos que, gracias a esta herramienta, y a su documentación, el uso de la realidad aumentada puede estar al alcance de los desarrolladores noveles, que de otro modo no podrían llevar a cabo sus proyectos. Cabe destacar el carácter gratuito de esta librería, y el enfoque gratuito de este proyecto.

Dedicatoria y agradecimientos.

Ha llegado el momento de agradecer a los que han hecho posible este proyecto, la ayuda y la paciencia de mi tutor, y a mi pareja por entender mi cabezonería en el tema del proyecto y sacrificar las merecidas vacaciones para la realización del mismo.

Lo dedico, como no, a mis padres por estar siempre apoyándome en mi formación académica.

Bibliografía

[1] Jesús Tomas. *Localización – Diploma de Especialización en desarrollo de aplicaciones para Android. Universidad Politécnica de Valencia.* [Disponible] http://www.androidcurso.com/index.php/tutoriales-android/41-unidad-7-seguridad-y-posicionamiento/284-localizacion [Último acceso: 19-09-2016]

[2] Referencias para el desarrollo de Android. [Disponible] https://developer.android.com/index.html [Último acceso: 22-08-2016]

[3] David Sachs, GoogleTechTalks. *Sensor fusión on Android devices: A revolution in motion processing*. [Disponible] https://www.youtube.com/watch?v=C7JQ7Rpwn2k [Último acceso: 15-09-2016]

[4] *Official U.S. Government information about the Global Positioning System (GPS) and related topics*. [Disponible] http://www.gps.gov [Último acceso: 24-08-2016]

[5] OpenCV (Open Source Computer Vision). [Disponible] http://opencv.org/ [Último acceso: 29-08-2016]

[6] F. J. Canny. A computional approach to edge detection. IEEE Transactions on Pattern Analysis and Machine Intelligence. ISSN: 0162-8828 Páginas: 679-698

[7] Chris Veness. *Calculate distance, bearing and more between Latitude/Longitude points*. [Disponible] http://www.movable-type.co.uk/scripts/latlong.html [Último acceso: 29-07-2016]

[8] *Canny Edge Detection*. [Disponible] http://docs.opencv.org/trunk/da/d22/tutorial_py_canny.html [Último acceso: 25-08-2016]

[9] Sami Salkosuo. *Coordinate conversions made easy*. [Disponible] http://www.ibm.com/developerworks/library/j-coordconvert/ [Último acceso: 15-09-2016]

[10] Philip Jahoda. *Librería MPAndroidChart* [Disponible] https://github.com/PhilJay/MPAndroidChart [Último acceso: 22-08-2016]

[11] An*droid Open Source Project*. [Disponible]
http://source.android.com/
[Último acceso: 29-09-2016]