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

IDENTIFICACIÓN DE ESPECIES VEGETALES UTILIZANDO DISPOSITIVOS MÓVILES

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL EN COMPUTACIÓN

SEBASTIÁN VALENZUELA PÉREZ

PROFESOR GUÍA:
PABLO BARCELÓ BAEZA

MIEMBROS DE LA COMISIÓN:
ROMAIN ROBBES
JUAN ÁLVAREZ RUBIO

SANTIAGO DE CHILE
ABRIL, 2013

Resumen

Las plantas juegan un rol determinante para la vida humana (y del planeta en general), sin embargo la mayoría de las personas hoy en día saben poco de las especies vegetales que las rodean: si se puede usar como alimento, medicina, si tiene alguna significancia especial para algún grupo humano, etc. Tal conocimiento forma parte del saber humano, sólo que no se ha masificado.

Por otra parte, actualmente somos testigos de cómo las tecnologías móviles son accedidas por un público más amplio y de cómo se vuelven más potentes, siendo muy común la disponibilidad de un teléfono, conexión a internet, sistema de posicionamiento, etc. Entre estos dispositivos destacan los que usan los sistemas operativos iOS y Android, que se han posicionado por las facilidades que presentan para el desarrollo de aplicaciones como la documentación en línea, el soporte de amplias comunidades, confiabilidad y estabilidad.

Las tecnologías de reconocimiento de imágenes se han introducido con fuerza en las últimas décadas tratando de extraer información semántica de las imágenes: Reconocimiento facial, clasificación de imágenes en medicina, astronomía, entre otras. Dada la naturaleza humana, siendo la visión el sentido más importante, las tecnologías de este tipo constituyen un área de muchísimo interés para la evolución de la tecnología.

Con el afán de formar conciencia sobre las especies vegetales que se tienen al alcance, nace la idea de implementar una guía de campo en la que sea fácil hacer búsquedas, tan fácil como hacer una observación de la planta en cuestión, esta observación corresponde a la toma de una fotografía a partir de la cual se entregue el nombre de la especie. Una aplicación que hace esta tarea es leafsnap para el sistema operativo iOS, en esta memoria se presenta el desarrollo de un prototipo que implementa una alternativa para la identificación de especies vegetales por reconocimiento de imágenes para el sistema operativo android.

El desarrollo de esta memoria apunta ampliar el impacto de leafsnap, considerando un área geográfica distinta, usuarios de dispositivos android. Además, de contar un servicio web que realice las búsquedas para la identificación de especies, que no requiera la interacción de una aplicación específica sino que sea accesible por cualquiera que desee usarlo.

A los pueblos originarios, en especial a los que resisten por nuestra sobrevivencia como especie en el Amazonas.

Tabla de contenido

Índice de figuras

1 Introducción

En cualquier lugar del mundo las personas están rodeadas de especies vegetales, a las que se les da diversos usos, desde ornamentales hasta en la producción de alimentos y medicinas. Lamentablemente la mayoría de las personas sabe muy poco acerca de las plantas que le rodean, muchas veces ni siquiera se conoce el nombre de los árboles que se encuentra en algún parque por el que se camina todos los días o incluso en el propio jardín.

Todo ese conocimiento, sin embargo, está disponible en la web, basta con ingresar el nombre de una especie en cualquier buscador para encontrarse con descripciones, usos y fotografías de la planta en cuestión. El problema entonces es cómo saber el nombre de esa especie (común o científico), para esto es común el uso de guías de campo, pero estas guías suelen ser demasiado amplias y específicas a algún sector geográfico (en especial las impresas) por lo que es muy costoso utilizarlas para un usuario común sin mayores conocimientos en botánica.

Por otra parte el uso de dispositivos móviles con conexión a internet ha aumentado fuertemente en los últimos años, especialmente los teléfonos inteligentes y los tablets, los que mayoritariamente cuentan con cámara y sistema de posicionamiento, entre muchas otras características. Aprovechando esto han surgido sendas iniciativas para generar una guía en que la búsqueda se hace utilizando una imagen tomada con uno de estos dispositivos, la que luego se envía a un servidor en el que se procesa la imagen para identificar la especie en cuestión. La más exitosa de estas iniciativas se llama leafsnap[1], proyecto desarrollado en conjunto por Columbia University, University of Maryland y Smithsonian Institution. La aplicación está actualmente disponible para el sistema operativo iOS para dispositivos Apple. Esta aplicación, aunque es gratuita, no es de código abierto[1] y el reconocimiento de imágenes en el servidor no cuenta con una interfaz o api de acceso público hasta el momento.

Esta memoria tiene como finalidad entregar una posible implementación de una guía de campo electrónica, disponible para el sistema operativo android, que permita al usuario la identificación de especies vegetales a través de una muestra fotográfica tomada por el mismo usuario.

[1] Aunque los autores de leafsnap han decidido publicar el código, hasta el momento no se ha hecho.

1.1 Motivación y justificación

Las plantas son parte fundamental del ecosistema planetario, ellas producen el oxígeno que respiramos, alimentos, medicina, fibras textiles, papel, etc. Por lo que saber más sobre ellas se hace una necesidad indispensable para el humano que pretende entrar a una era de sustentabilidad, y no sólo para unos pocos expertos, sino que este conocimiento debiese masificarse.

La utilización de guías de campo son una gran ayuda en el ámbito de la educación, ya sea formal o por afición personal, la experiencia directa es una de las formas más eficaces de aprendizaje. Específicamente en el campo de la botánica contar con una herramienta que identifique automáticamente especies basándose en una acción tan simple como tomar una fotografía con un aparato móvil, se convierte en una inmensa ayuda, tanto para el aficionado que quiere saber un poco más de lo que le rodea como del profesional que necesita aprender sobre muchísimas especies, lo que se hace tedioso buscando por otros índices como el nombre o la distribución geográfica.

Por otra parte esta aplicación podría tener aplicaciones en turismo, por ejemplo un hotel podría ofrecer el servicio de esta aplicación con una base de datos restringida a las especies que se pueden encontrar en el entorno del mismo.

Aunque ya existe una aplicación con este objetivo disponible para usuarios del sistema operativo iOS, aún se le adeuda a los usuarios de android (otro de los sistemas operativos para dispositivos móviles más populares hoy) una aplicación que cumpla con esa función, además de generar una base de datos con especies locales que eventualmente podría crecer para abarcar especies de todo Chile o todo el mundo, por lo que esta memoria se centrará precisamente en desarrollar un sistema similar a leafsnap pero enfocándose en el sistema operativo android para su utilización.

Se pretende, además, que a diferencia de leafsnap, la implementación permita un fácil acceso a la base de datos, su manejo y al servicio de identificación de especies, desde otros sistemas, no necesariamente la aplicación para android incluida en este desarrollo, se incluye también un servicio que permite agregar fácilmente una muestra de una especie determinada, haciendo de la construcción de la base de datos un proceso colaborativo.

1.2 Objetivos

1.2.1 Objetivo General

Desarrollar una aplicación para el sistema operativo android y un servicio accesible vía web que permitan identificar especies vegetales, realizando una búsqueda por similitud de características, extraídas usando algoritmos de reconocimiento de imágenes sobre la consulta que es una fotografía tomada por el usuario con la aplicación.

1.2.2 Objetivos específicos

- Generar un modelo de datos que permita almacenar una descripción matemática de especies vegetales.
- Implementar un algoritmo de reconocimiento de imágenes que permita encontrar, eficazmente, la imagen más parecida a la de búsqueda en una base de datos de imágenes.
- Disponer de una interfaz o API pública que ofrezca el servicio de reconocimiento de especies a partir de una imagen.
- Desarrollar una aplicación para android que permita al usuario capturar una fotografía a partir de la cual se identifique y se despliegue la información sobre la especie.
- Realizar un análisis comparativo entre distintas posibilidades de implementación, tomando en cuenta algoritmos, arquitectura de hardware y software y modelo de datos.

2 Antecedentes

A continuación se explicarán los sustentos teóricos para el desarrollo de la aplicación, también se explicarán los principales algoritmos utilizados y se revisará a Leafsnap, aplicación para iOS que sirve de modelo para esta memoria.

2.1 Aplicaciones para dispositivos móviles (Apps).

Desde hace algún tiempo el uso de dispositivos móviles de alta capacidad se ha masificado, en especial los que cuentan con los sistemas operativos iOS y Android, para los que existen una gran variedad de aplicaciones desarrolladas por los mas diversos desarrolladores y para una amplia gama de propósitos, desde mapas, navegación por la web, juegos, redes sociales y un largo etcétera. Estas plataformas constituyen un ambiente ideal para construir sistemas de información, pues presentan una buena cantidad y variedad de recursos como cámara, GPS, acelerómetro, una capacidad de procesamiento y almacenamiento que crece con los modelos que se pelean por dominar el mercado, haciendo que los equipos de mas bajo costo sean muy poderosos.

Desarrollar una aplicación para dispositivos móviles (app), entonces, es una buena forma de entregar un servicio de información a una gran cantidad de usuarios, pues sólo hace falta una sencilla instalación de ésta para accederla y usarla, sin necesitar equipamiento extra. En este contexto, el sistema desarrollado en esta memoria es ideal para reemplazar guías de campo en una expedición para la identificación de especies, además de poner esta información a disposición de un público que de otra forma no la accedería.

2.2 Reconocimiento de especies vegetales[3]

El reconocimiento de una especie vegetal se puede hacer tomando en cuenta una variedad de criterios. Son distintivos de cada especie el tallo o tronco, la corteza, la ramificación, las flores, frutos, semillas, hojas. Basándose en un criterio de usabilidad se puede descartar utilizar las flores, los frutos y las semillas, ya que suelen encontrarse por períodos muy limitados durante el año. La corteza puede ser difícil de capturar y se restringe a los árboles, la ramificación es difícil de capturar. Las hojas, por otra parte, tienen una mayor presencia que flores, frutos y semillas, y en muchos casos es posible encontrar hojas en la planta toda su vida, además esta parte se encuentra en la mayoría de las especies vegetales.

La hoja de una planta puede diferenciarse por la nervadura, la textura o la forma de la hoja. La nervadura y la textura de las hojas pueden tener variaciones pequeñísimas, apenas perceptibles, entre distintas especies, por lo que una mejor opción es utilizar la forma de la hoja. Otra característica que se descarta usar es el color, pues la hoja puede variar mucho su color dependiendo de la época del año o de las condiciones de crecimiento de la planta (disponibilidad de agua, nutrientes, luz, dióxido de carbono, etc.).

Dadas las razones expuestas anteriormente, el criterio que se utilizará para reconocer las especies vegetales es la forma de las hojas.

2.3 Reconocimiento de imágenes[2]

Como consecuencia de la sección anterior, una de las tareas más importantes es la extracción de una descripción matemática de la forma de la hoja para poder hacer comparaciones entre las muestras que constituyen la base de datos y las que envía el usuario. Por lo tanto este proyecto se centró en desarrollar los módulos necesarios para llevar a cabo las operaciones de segmentación de la imagen, extracción de características y búsqueda por cercanía en la base de datos.

2.3.1 Segmentación de la imagen

Para extraer las características de forma de la hoja es necesario primero identificar su borde en la imagen, lo que se logra diferenciando que área de la imagen es parte de la hoja y cual no. Es decir, se empieza por sustraer el fondo. Para facilitar esta tarea se exige hacer la búsqueda con imágenes con un fondo de color claro (idealmente blanco), para tener un buen contraste. La sustracción puede ser efectuada con un criterio umbral o “thresholding”, lo que consiste en clasificar cada pixel por su brillo, suponiendo que si un pixel tiene un brillo sobre cierto umbral entonces pertenece al fondo de la imagen, de otra forma es parte de la hoja. Por la facilidad de implementación de este método, fue escogido para el desarrollo, sin embargo, fue prontamente descartado por su poca efectividad. Un mejor apronte se hizo con el método de Maximización de la Esperanza (EM).

2.3.1.1 Segmentación usando EM[8].

Los colores de una imagen pueden ser descritos en una variedad de formas o sistemas coordenados, como las bandas RGB, matiz-saturación-iluminación (HSL) o matiz-saturación-valor(HSV) entre otros. Para el caso de segmentación de imágenes que contienen hojas, la saturación y el valor de la imagen han resultado ser útiles y suficientes.

El valor corresponde a la máxima componente del valor.

$$V = \max(R, G, B)$$

La saturación corresponde a que tan colorido es el pixel.

$$S = \begin{cases} 0 & c = 0 \\ \frac{c}{v} & c \neq 0 \end{cases}$$

donde

$$c = \max(R, G, B) - \min(R, G, B)$$

Las imágenes con las que se trabaja tienen básicamente dos grupos de pixeles, la hoja y el fondo, si graficamos los pixeles en las coordenadas de saturación y valor, se pueden distinguir dos cúmulos de pixeles, tal como se aprecia en la Figura 2.1, los que siguen distribuciones gaussianas, entonces la distribución de probabilidad para un pixel en este espacio es:

$$p(x|\theta) = \sum_{k=1}^{2} \frac{1}{2} p(x|\mu_k, \Sigma)$$

donde $p(x|\mu_k, \Sigma)$ es una distribución gaussiana, con media μ_k y covarianza $\sum$, se le ha dado un peso de ½ a cada una, esta aproximación funciona bastante bien, pues, luego de recortar la imagen acotándola al contenido relevante ("cropping"), la hoja utiliza aproximadamente la mitad del espacio en la imagen, esto se cumple al menos en las muestras utilizadas. Los parámetros μ_k se calculan encontrando el máximo en las regiones donde se sabe es más probable encontrar los cúmulos, esta región ha sido indicada manualmente.

Sobre esto se aplica el algoritmo de EM[9], que consiste en encontrar estimadores de máxima verosimilitud de los parámetros de un modelo estadístico, para ello se alternan dos pasos: calcular la probabilidad de un pixel y actualizar los parámetros del modelo usando esa probabilidad, utilizando la simplificación de que la covarianza es igual para ambas distribuciones y es la identidad, la probabilidad de que un pixel x se encuentre en una u otra zona, denotada por $z \in \{1,2\}$, se puede calcular como

$p(z = 1|x) = \frac{1}{1+\exp(\beta_0+\beta^T x)}$ y $p(z = 2|x) = 1 - p(z = 1|x)$, donde

$$\beta \equiv (\mu_2 - \mu_1)$$

$$\beta_0 \equiv -\frac{1}{2}(\mu_2^T \mu_2 - \mu_1^T \mu_1)$$

y los parámetros son actualizados usando

$$\mu_1^{(t+1)} = \frac{\sum_{i=1}^{n} p_i^{(t)}(z=1|x_i) * x_i}{\sum_{i=1}^{n} p_i^{(t)}(z=1|x_i)} \text{ y } \mu_2^{(t+1)} = \frac{\sum_{i=1}^{n} p_i^{(t)}(z=2|x_i) * x_i}{\sum_{i=1}^{n} p_i^{(t)}(z=2|x_i)}$$

estos parámetros se inicializan cerca del centro esperado de cada una de las gaussianas que componen el modelo.

(a) (b) (c)

Figura 2.1: (a)Imagen original, (b) Imagen segmentada, (c) Distribución de pixeles en el espacio saturación (eje horizontal) – valor(eje vertical), en azul los que corresponden al fondo y en verde a la hoja.

2.3.2 Detección de bordes

El borde de la hoja en una fotografía puede describirse como una curva cerrada, los pixeles que componen esa curva son los que estando dentro de la región (que corresponde a la hoja) tienen vecinos que están fuera de ella.

La conectividad de la curva puede definirse de dos formas, analizando los vecinos de arriba, abajo y a los lados o además los que están en diagonal, ambos tienen igual validez, ya que si hay conectividad en el sentido de cuatro direcciones también la hay en el ocho y viceversa, esto se diagrama en la figura 2.2.

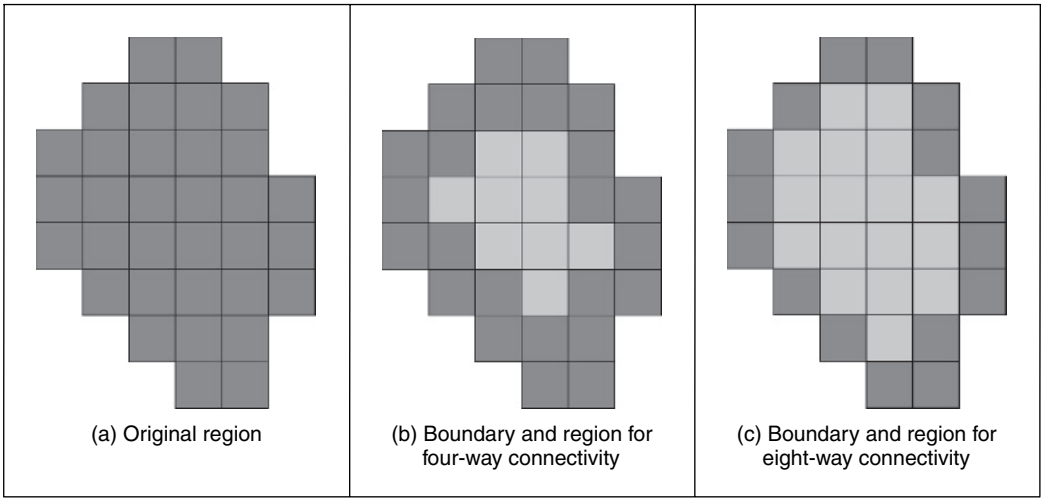

Figura 2.2: Detección de bordes (imagen original en [2])

Entonces, para detectar el borde primero es necesario encontrar un pixel que esté en el borde y luego proceder con una de las reglas para detectar el resto.

2.3.3 Descriptores elípticos de Fourier

La descripción de la forma de la hoja se puede hacer con una función que describa el borde como una curva o una que describa el área de la hoja. Para describir la curva, uno de los algoritmos más populares es el de descriptores elípticos de Fourier(EFD). Para este proyecto se optó por los (EFD) en primera instancia por la facilidad de su implementación, pero también porque mostraron ser una representación suficientemente buena para que la búsqueda entregase resultados coherentes. Por otra parte explorar e implementar otras funciones como los momentos de Zernik[2] (descripción del área) se consideró que el desarrollo podría tomar demasiado tiempo en relación a la mejoría teórica en los resultados (basándose en la evidencia presentada en[4]). Otras funciones pueden presentar mejores estadísticas de reconocimiento, pero el costo computacional es demasiado elevado [4].

Los EFD tienen la ventaja de ser invariantes a la escala, rotación y posición de la curva, por lo que se ajusta muy bien a las necesidades de la aplicación. Así sólo es importante tomar la foto completamente de frente a la hoja, pues de otra forma aparecerían deformaciones difíciles de detectar y manipular.

El borde de la hoja puede ser descrito como una función paramétrica en un plano complejo

$$c(t) = x(t) + jy(t)$$

La expansión de Fourier de esta función en su forma trigonométrica, expresada matricialmente queda como

$$\begin{bmatrix} x(t) \\ y(t) \end{bmatrix} = \frac{1}{2}\begin{bmatrix} a_{x0} \\ a_{y0} \end{bmatrix} + \sum_{k=1}^{\infty} \begin{bmatrix} a_{xk} & b_{xk} \\ a_{yk} & b_{yk} \end{bmatrix} \begin{bmatrix} \cos(k\omega t) \\ \sin(k\omega t) \end{bmatrix}$$

donde

$$a_{xk} = \frac{2}{m}\sum_{i=1}^{m} x_i \cos(k\omega i\tau) \qquad b_{xk} = \frac{2}{m}\sum_{i=1}^{m} x_i \sin(k\omega i\tau)$$

$$a_{yk} = \frac{2}{m}\sum_{i=1}^{m} y_i \cos(k\omega i\tau) \qquad b_{yk} = \frac{2}{m}\sum_{i=1}^{m} y_i \sin(k\omega i\tau)$$

donde x_i, y_i son los valores que toman las funciones $x(t), y(t)$ en un punto de muestreo i y $\tau = \frac{2\pi}{m}$. Se definen los coeficientes elípticos de Fourier, después de la introducción de invariancia como

$$d_k = \frac{\sqrt{a_{xk}^2 + a_{yk}^2}}{\sqrt{a_{x1}^2 + a_{y1}^2}} + \frac{\sqrt{b_{xk}^2 + b_{yk}^2}}{\sqrt{b_{x1}^2 + b_{y1}^2}}$$

El desarrollo completo de estas ecuaciones puede ser revisado en [2].

En la Figura 2.3 se pueden ver ejemplos de la reconstrucción, a partir de los descriptores, del borde de algunas hojas.

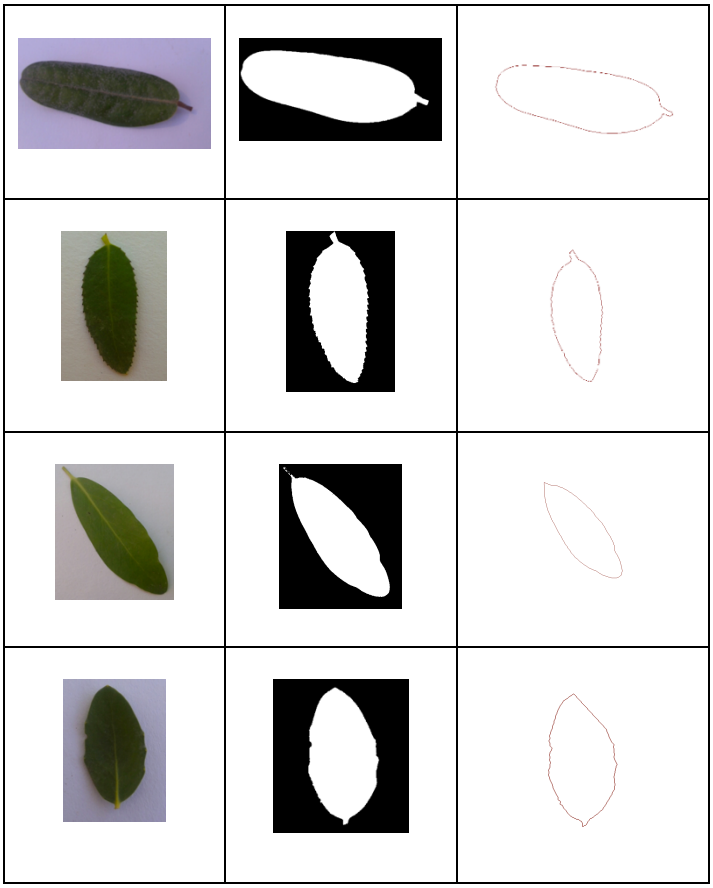

Figura 2.3: Ejemplos de reconstrucción con EFD, primera columna imagen original, segunda columna imagen segmentada, tercera columna contorno reconstruido con EFD.

2.4 Búsqueda

Lo que se obtiene del procesamiento de la imagen, explicado en la sección 2.3, es un vector que contiene los parámetros de la expansión de Fourier, tales parámetros se llaman descriptores. Para saber qué tan parecidas son dos curvas basta comparar los descriptores en un espacio euclidiano de *n* dimensiones, donde *n* es la cantidad de descriptores que se está usando. La hoja más parecida en una base de datos a una de consulta, entonces, es cuyos descriptores están mas cerca en el espacio recién descrito.

Para hacer la búsqueda se consideran los vecinos más cercanos a la imagen de búsqueda, ordenándolos por cercanía. En la base de datos se encuentran varias muestras por cada especie, por lo que el resultado de la búsqueda entregará un conjunto en el cual se puede repetir la especie, basándose en esto se utiliza una función de agregación que asigna peso según la cercanía a la búsqueda y que asigna un puntaje a cada especie, las especies son entregadas en el orden que determine el puntaje, pero la identificación final queda a cargo del usuario.

2.5 Desarrollo para el sistema operativo Android

Las aplicaciones en Android se deben programar en el lenguaje de programación Java – junto a otros recursos de configuración – y se compilan con el Android SDK[2] con el fin de generar un único archivo (de extensión .apk) para ser instalado en los dispositivos. Una vez instalado, la aplicación vivirá en su propia sandbox[3] por motivos de seguridad y ejecución de procesos en Android. Adicionalmente, y de manera opcional, se pueden utilizar componentes nativas de Linux que no son accesibles desde la API[4] de Android. Estas rutinas deben ser escritas en los lenguajes C o C++ y son compiladas con el Android NDK[5] para ser utilizadas desde la aplicación escrita en Java mediante JNI[6].

Toda aplicación debe contener un único archivo de configuración – de nombre *AndroidManifest.xml* – ubicado en el directorio raíz del desarrollo. En este archivo se declaran los elementos a utilizar por la aplicación, la forma en que se relacionan y los permisos necesarios, además de otras configuraciones. Existen dos elementos importantes en esta configuración, primero la clase a instanciar para mantener un estado global de la aplicación, segundo las clases a instanciar de las diferentes componentes de aplicación que constituyen la aplicación.

Existen cuatro tipos de componentes de aplicación para desarrollar una aplicación en Android, cada una con diferentes propósitos y ciclos de vida que definen cómo se crean y destruyen. Nos centraremos en dos, las cuales son de mayor interés desde el punto de vista de un usuario final; los *Activities* y *Services*. La primera representa un proceso con interfaz de usuario desplegada en el dispositivo y la segunda un proceso sin interfaz de usuario que se ejecuta paralelamente a otras del tipo anterior. De esta manera, una aplicación puede tener más de una componente de aplicación y estas componentes pueden iniciar otras mediante diferentes mecanismos, sin embargo al momento de ejecutar la aplicación comenzará con la ejecución de solo una componente configurada en el archivo de configuración de la aplicación.

A veces las componentes de aplicación requieren compartir contenido entre ellas, para esto existen al menos dos formas, una es por medio de solicitudes de acciones mediante *Intents* (además estas solicitudes poseen la capacidad de iniciar componentes de aplicación), la otra forma es manteniendo actualizado el

[2] **S**oftware **D**evelopment **K**it
[3] Mecanismo para ejecutar programas con seguridad y de manera separada de los otros.
[4] **A**pplication **P**rogramming **I**nterface
[5] **N**ative **D**evelopment **K**it
[6] **J**ava **N**ative **I**nterface

estado global de la aplicación por medio de la instancia de la clase configurada en el archivo de configuración de la aplicación.

2.6 Trabajos relacionados

A continuación se describe qué es leafsnap, aplicación que es la referencia más directa en el desarrollo de esta memoria.

2.6.1 Leafsnap

Leafsnap es la primera aplicación para móviles que aborda el problema de identificación de plantas utilizando sólo una fotografía tomada desde un dispositivo móvil. Su versión actual incluye especies presentes en el noreste de Norteamérica y está disponible para el sistema operativo iOS.

La aplicación primero clasifica la imagen como válida, para ello detecta si lo que se le envía corresponde a una hoja sobre un fondo sólido de color claro, lo que en caso de ser negativo se le indica al usuario tal falla y además se dan las instrucciones para tomar correctamente la fotografía; por el contrario si la imagen corresponde a una hoja, se prosigue a segmentar la foto, separando el fondo de la hoja, luego se extrae la característica de la forma de la hoja, que luego se contrasta con la base de datos y que por un criterio de cercanía por distancia euclidiana se determinan las posibles respuestas, la identificación final, entonces, queda a cargo del usuario.

Así, y dadas las características de esta aplicación, se utilizó como una guía para el desarrollo del prototipo presentado en esta memoria. La idea es extender esta solución a usuarios del sistema operativo Android, además de cubrir un área geográfica distinta.

3 Diseño

En este capítulo se explican las distintas soluciones que se analizaron y qué pruebas se realizaron.

3.1 Arquitectura de hardware

Se analizaron tres arquitecturas como soluciones posibles cada una de las cuales se describe a continuación

- Arquitectura 1: El dispositivo móvil se utiliza sólo para obtener las imágenes y desplegar los resultados, el procesamiento y almacenamiento se hace en servidores. Esta solución queda ilustrada en el diagrama de distribución de la Figura 3.1.

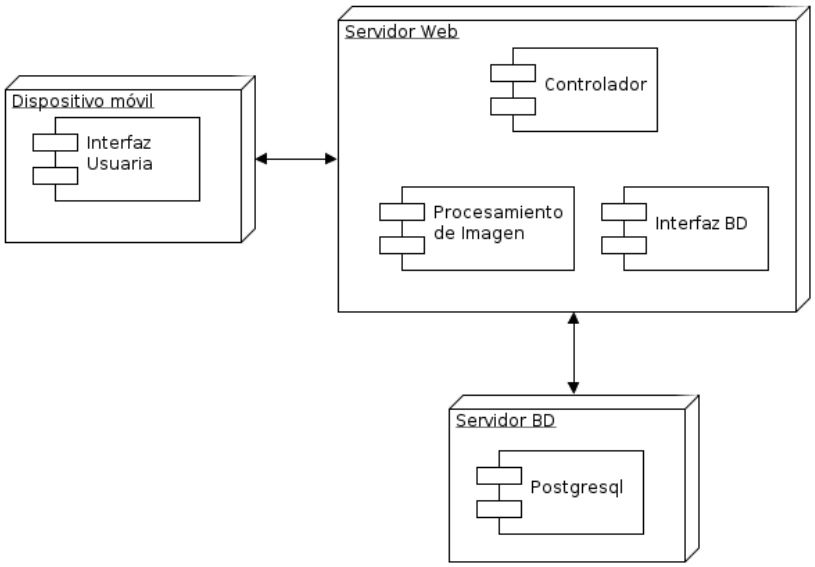

Figura 3.1: Arquitectura de despliegue, propuesta 1.

  En este caso se puede disponer de una API REST para construir otras aplicaciones sobre el servicio. Sin embargo, requiere que se envíe una imagen a través de la red, lo que en muchos casos podría tardar muchísimo afectando el uso de la aplicación o causando problemas en el servidor que demora mucho por cada petición por lo que es más susceptible a sobrecargas.
- Arquitectura 2: la segunda propuesta es que el mismo dispositivo se encargue de todo el procesamiento, incluida la base de datos. En la figura 3.2 se puede ver el diagrama de distribución para esta arquitectura.

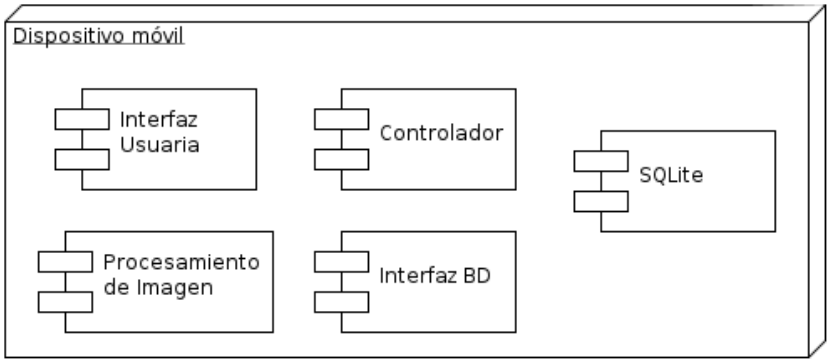

Figura 3.2: Arquitectura de despliegue, propuesta 2

Aquí no se depende de la disponibilidad de acceso a internet, por lo que se tiene la independencia de utilizar la aplicación en parques o lugares alejados de las ciudades donde no existe cobertura de redes para celulares (3G, EDGE, etc). Para algunos dispositivos el procesamiento de la imagen y la búsqueda en la base de datos podría ser muy demandante, pero los aparatos que aparecen en el mercado son cada vez más poderosos, por lo que este no se espera sea un gran problema. Finalmente la utilización de SQLite, que es la base de datos que utilizan los dispositivos móviles, en vez de Postgresql, mas poderosa y que se utilizaría en un servidor, trae ciertas limitantes y desarrollos extra.

- Arquitectura 3: Finalmente se puede optar por una solución híbrida y poner sólo la base de datos y su interfaz en un servidor como se muestra en la Figura 3.3: Arquitectura de despliegue, propuesta 3..

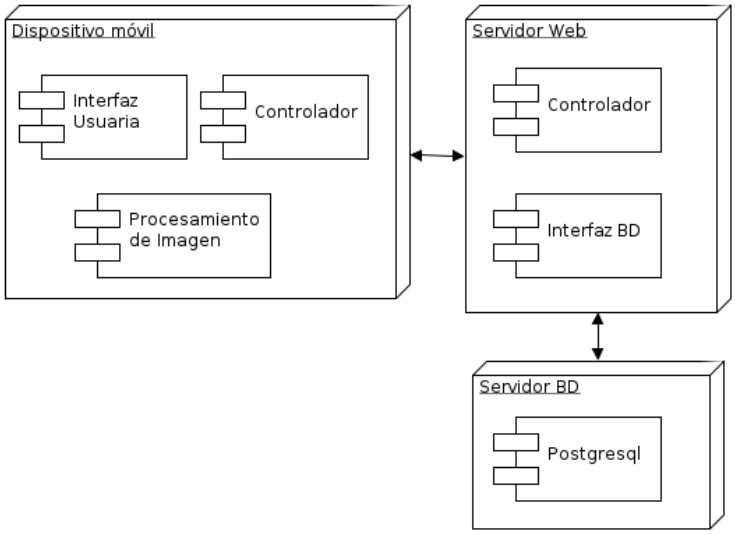

Figura 3.3: Arquitectura de despliegue, propuesta 3.

En este caso los datos que deben ser transmitidos por la red son de mucho menor tamaño, ya que sólo se necesita enviar los descriptores de la imagen para su búsqueda en la base de datos y no la imagen entera. Además se cuenta con una mejor base de datos que en la arquitectura 2.

Finalmente se decidió por la arquitectura 1, pues:

1. De las tres es la de más fácil implementación, pues no necesita la instalación de paquetes especiales para el procesamiento de imágenes en el dispositivo móvil, como en las otras dos alternativas, además hay más experiencia en este tipo de desarrollo (servicios web) por parte del autor.
2. Si se considera que la base de datos puede estar en constante crecimiento, para la opción 2, esto requeriría estar descargando actualizaciones constantemente, además se puede volver demasiado grande y las búsquedas demasiado complejas para estar contenidas en un dispositivo móvil, al menos los que se conocen hoy.
3. La base de datos puede quedar disponible para su utilización por otras aplicaciones y así tener un carácter más colaborativo.
4. La arquitectura 3, aunque puede ser una evolución de la 1, significa un gran costo de implementación, pues para manipular imágenes se utilizan bibliotecas especializadas en Python y en C, las que se pueden instalar en los dispositivos pero requieren de la compilación y empaquetamiento para el sistema operativo Android. Aunque por supuesto, esta opción queda como un posible desarrollo futuro.
5. El procesamiento de imágenes, dependiendo del tamaño de ellas puede ser bastante costoso, por lo que hacerlo en el dispositivo móvil podría resultar en un consumo de recursos muy alto y podría volver el proceso muy lento.

3.2 Arquitectura de software

La aplicación cuenta con 4 módulos principales: Interfaz usuaria, controlador(es), procesador de imágenes, interfaz de base de datos. A continuación se describe cada uno.

3.2.1 Interfaz usuaria

La función de este módulo es tomar la fotografía, enviarla al módulo de controlador y recibir y desplegar la información correspondiente al resultado de la búsqueda. En resumen se encarga de la relación del usuario con la aplicación. La usabilidad de la interfaz no se estudió mayormente, ya que esto requiere un

estudio más amplio en esta área lo que no fue posible por priorizar las otras partes del sistema. Sólo se consultó a cinco personas, quienes coincidieron en que el uso de la aplicación es bastante fácil e intuitivo, sin embargo, lo reducido del grupo y la no formalización y estandarización de estas pruebas, hacen que no se pueda concluir que la interfaz sea la más adecuada para una versión final.

La aplicación se inicia con una vista de la cámara del aparato con la opción de toma una foto (Figura 3.4).

Figura 3.4: Previsualización de la cámara

Cuando se ha tomado, la foto se muestra en la pantalla, dando la opción de enviarla para realizar una búsqueda o tomar una nueva foto si la que se tomo no satisface al usuario (Figura 3.5).

Figura 3.5: Fotografía tomada

Luego de enviar la fotografía, se despliegan los resultados de la búsqueda, que se ha fijado en dos (ver Figura 3.6).

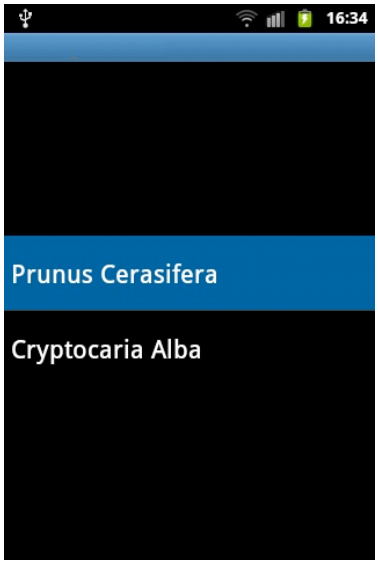

Figura 3.6: Lista de resultados

Al seleccionar una especie en la pantalla de resultados se muestra la imagen (de alta resolución) del servicio y la opción de ver la entrada de la especie en Wikipedia (en el navegador del dispositivo)(Figura 3.7).

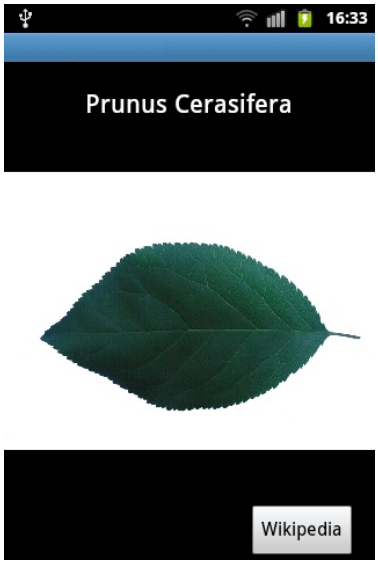

Figura 3.7: Imagen de muestra de la especie

3.2.2 Procesador de imágenes

Se encarga de extraer la información necesaria para describir la forma de la hoja de manera que se pueda comparar matemáticamente. Para ello se siguen los siguientes pasos:

1. Convertir la imagen a binaria[7], diferenciando la hoja sobre el fondo
2. Identificar el borde de la hoja y extraer los descriptores

3.2.3 Interfaz de base de datos

Su función es intermediar entre el controlador y la base de datos que contiene información de las especies vegetales, en este módulo se implementan las consultas en el lenguaje de la base de datos (SQL) y responde con datos en el formato adecuado al lenguaje del controlador.

3.2.4 Controlador(es)

Como se vio en los diagramas de la sección 3.1, según la arquitectura puede haber 1 ó 2 controladores. En general, los controladores son los encargados del flujo de la aplicación. En este caso se encargan de recibir la información –

[7] Una imagen binaria tiene sólo pixeles blancos y negros

imagen- desde la interfaz usuaria, obtener los descriptores de la imagen utilizando el procesador de imágenes, solicitar y recibir los resultados de una búsqueda en la base de datos con la interfaz de base de datos y finalmente enviar la información de vuelta a la interfaz, para el caso de 2 controladores estas tareas se reparten y se agrega la de comunicarse entre ambos controladores.

3.3 Modelo de datos

Se trabajará con los siguientes datos por cada planta: nombre, nombre común, imagen de una hoja de la planta, dirección de la entrada en Wikipedia (con el fin de entregarle más datos al usuario) y los descriptores que identifican el borde de la hoja. Todos estos datos son únicos a cada planta, excepto por los descriptores, que se toman desde varias muestras. Un modelo entidad relación se muestra en la Figura 3.8

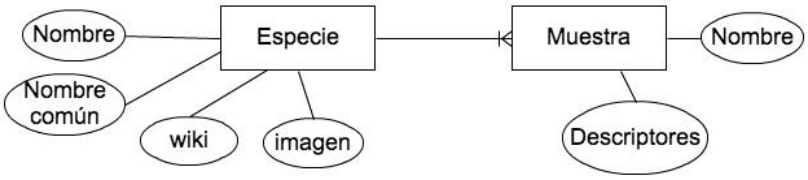

Figura 3.8: Diagrama ER del modelo de datos

4 Desarrollo

4.1 Implementación

Se pueden identificar dos etapas en el desarrollo: el servicio web y la aplicación para dispositivos móviles.

4.1.1 Servicio web

Este desarrollo se llevó a cabo utilizando el lenguaje de programación Python 2.7, la herramienta de control de versiones git[8] con la plataforma Github[9]. El servicio se compone de tres módulos: Plants-Web, Plants-Libs y Plants-Dal. En todos los casos se utilizó la herramienta easy_install para empaquetamiento e instalación de cada uno

4.1.1.1 Plants-Libs

En este módulo se implementan las funciones para el tratamiento de las imagines y la extracción de los descriptores. Para esta manipulación se utilizaron las bibliotecas PIL (Python Image Library) y OpenCV (originalmente escrita en C++, pero que dispone de un adaptador para Python).

- Substracción del fondo: Se pide que la fotografía se tomada con un fondo blanco, para que así se pueda aplicar un criterio de substracción de fondo mas bien sencillo, en este caso, en una primera iteración, es simplemente por el brillo de los pixeles, considerando que un pixel que cercano al blanco es parte del fondo y el resto es parte de la hoja. De esto se reemplaza cada pixel del fondo con un pixel negro y cada pixel de la hoja con uno blanco. En el Código 4.1 se muestra la implementación de esta tarea, aunque se pretendía usar el algoritmo EM finalmente se desistió pues la reestimación de los parámetros tarda demasiado tiempo, pues requiere recalcular las probabilidades para cada pixel en cada iteración, por lo que finalmente se fijaron los parámetros tomando como media de cada una de las gaussianas el punto de máximo del histograma de saturación-valor de cada "cúmulo", lo que resultó entregar segmentaciones precisas, cercanas a los bordes reales.

[8] http://git-scm.com/
[9] http://www.github.com

```
pixels = self.image.load()
#observed data points to these means for dsitributions of saturation and
value
beta = (0.5-0.16,0.19-0.7)
beta0 = -0.5 * (0.25 + (0.19*0.19)-(0.16*0.16)-0.49)
scale = 1000.0
sat_threshold = 300
val_threshold = 300
hist_sat = [0]*1001
hist_val = [0]*1001
for x in range(self.image.size[0]):
    for y in range(self.image.size[1]):
        v = float(max(pixels[x, y]))
        saturation = (float(max(pixels[x, y]) - min(pixels[x, y]))/v,
                      v/255.0)
        hist_sat[int(saturation[0]*int(scale))] +=1
        hist_val[int(saturation[1]*int(scale))] +=1
        mu1 = (float(hist_sat[:sat_threshold].index(\
               max(hist_sat[:sat_threshold])))/scale,
               float(hist_val[val_threshold:].index(\
               max(hist_val[val_threshold:])) + val_threshold)/scale)

         mu2 = (float(hist_sat[sat_threshold:].index(\
                max(hist_sat[sat_threshold:]))+sat_threshold)/scale,
                float(hist_val[:val_threshold].index(\
                max(hist_val[:val_threshold]))))/scale)
         beta = (mu2[0] - mu1[0],mu2[1]-mu1[1])
         beta0 = -0.5 * (pow(mu2[0],2) + pow(mu2[1],2) -\
                  pow(mu1[0],2)-pow(mu1[1],2))

for x in range(self.image.size[0]):
    for y in range(self.image.size[1]):
        v = float(max(pixels[x,y]))
        saturation = (float(max(pixels[x,y])-min(pixels[x,y]))/v,
                      v/255.0)
        pz1 = 1/(1+exp(beta0 + beta[0]*saturation[0] +\
              beta[1]*saturation[1]))
        if pz1 > 0.5:
            pixels[x,y] = (0,0,0)
        else:
            pixels[x,y] = (255,255,255)
#debugging
#self.image.show()
return self.image
```

Código 4.1: segmentación de la imagen (Python)

- Detección del borde: en un principio se quiso implementar directamente el

criterio de ocho direcciones para detectar un borde continuo, sin embargo, las pruebas sobre cada intento de corrección de esta función fallaron innumerables veces. Finalmente se dio con que esta funcionalidad se encontraba implementada en la biblioteca OpenCV, que entrega una muy buena aproximación del borde.

- Descriptores elípticos de Fourier: esta implementación es directa del algoritmo descrito en [2], donde se presenta el código para Matlab.

4.1.1.2 Plants-Dal (Data Access layer) y base de datos

Este módulo es la interfaz de interacción con la base de datos, se encarga de agregar registros y hacer las búsquedas en la base de datos. La base de datos que se utilizó en este desarrollo es Postgresql 9.0, y para facilitar la comunicación con ella se usó la biblioteca psycopg2.

La base de datos, en un principio, tenía sólo una tabla que se detalla como sigue:

Tabla Leaves:

- name: varchar, el nombre científico de la planta.
- common_name: varchar, nombre común de la planta.
- photo: text, dirección del archivo en un servidor externo (Amazon S3).
- wiki: text, url de la entrada en Wikipedia de la planta.
- descriptors: double precision[], vector que contiene los descriptores de la imagen.

Fue necesario además explicitar en la base de datos la función de distancia con la que se deben comparar las entradas, tal implementación se puede ver en el **Error! Reference source not found.** y corresponde a la distancia euclidiana entre vectores de dimensiones indeterminadas (requiere que los vectores tengan las mismas dimensiones).

```
CREATE OR REPLACE FUNCTION vdistance(double precision[], double
precision[])
  RETURNS double precision AS
$BODY$
DECLARE
      ret_sum double precision;
      sum1 double precision;
BEGIN
      ret_sum := 0;
      for i in 1..array_length($1,1) LOOP
            sum1:= $1[i]-$2[i];
            ret_sum=ret_sum+pow(sum1,2);
      END LOOP;
      return sqrt(ret_sum);
END
```

Código 4.2: Distancia euclidiana entre vectores en postgresql

Las pruebas mostraron que esta implementación fue insuficiente, fue necesario tomar más de una imagen como referencia, tal número no es fijo y puede ir creciendo al acumularse datos. Es por eso que la base de datos pasó a tener dos tablas como se detallan a continuación:

Tabla Leaves:

- name: varchar, el nombre científico de la planta.
- common_name: varchar, nombre común de la planta.
- photo: text, dirección del archivo en un servidor externo (Amazon S3).
- wiki: text, url de la entrada en Wikipedia de la planta.

Tabla Samples:

- name: varchar, el nombre científico de la planta, corresponde a algún nombre de la tabla “Leaves”
- descriptors: double precision[], vector que contiene los descriptores de la imagen de muestra.

La búsqueda por imagen en la base de datos se hace comparando la imagen con los valores en la tabla de muestras, se toman las diez más cercanas y se agrupan por el nombre de la especie (ver Código 4.3), el resultado es un conjunto de especies posibles, de las cuales el usuario hará la identificación final.

```
SELECT distinct Leaves.name as name
FROM Leaves, (SELECT name, vdistance(descriptors ,
            cast(%s as double precision[])) as distance
  FROM descriptors
  ORDER BY distance ASC
  LIMIT 10) as descr
WHERE Leaves.name = descr.name;
```

Código 4.3: Consulta en SQL para la búsqueda de las muestras más cercanas.

4.1.1.3 Plants-Web

En este módulo se encuentran las funciones de comunicación vía el protocolo http, y que opera con una API REST para sus llamadas. Se utilizó el framework Bottle 0.9, que facilita el manejo de peticiones y respuestas del servicio. Se implementaron las siguientes peticiones:

- POST: /plants/add
  Se agrega la entrada que se solicita en la base de datos, para ello se exige una imagen conteniendo la hoja de la planta en cuestión, la que será subida a un servicio, en este caso el servicio "S3" de Amazon, desde donde será servida estáticamente. Si el nombre de la planta no estaba en la base de datos, entonces se añade el registro, al contrario si estaba el servidor produce un mensaje adecuado.
- POST: /plants/add_sample
  Se agrega una imagen de referencia a la base de datos, se debe especificar a qué especie corresponde.
- POST: /plants/search
  Se sube una imagen, la cual será procesada y contrastada con la base de datos, entregando los datos correspondientes a la(s) entrada(s) que resulte(n) más cercana(s) a esa imagen. Los resultados se entregan en el formato JSON

4.1.2 Aplicación para dispositivos móviles android

En un principio se pretendió desarrollar la aplicación móvil en una plataforma que permitiera compilarla tanto para android como para iOS, tal como Titanium[7]. Sin embargo, al poco andar se cambió la decisión por la alternativa de desarrollar la aplicación directamente para android, usando sus herramientas de desarrollo. Esto se debió a que el aprender a usar debidamente una herramienta de desarrollo multiplataforma tiene una mayor dificultad y el

desarrollo tomaría más tiempo en comparación a las herramientas de desarrollo para android. Además y pese a lo anterior, la aplicación del dispositivo es más bien sencilla por lo que no debiese tomar mucho tiempo la reimplementación en otra plataforma, ya sea en iOS (Xcode) o Titanium u otra en la que eventualmente se quiera trabajar.

Este desarrollo se realiza en el lenguaje de programación Java, utilizando el IDE Eclipse y el plugin ADT (Android development tools) para eclipse. Se utilizó el equipo Samsung Galaxy Ace S5830M, el que cuenta con la versión 2.3.6 del sistema operativo android instalada.

Cada una de las interfaces requiere de un *layout*, el cual se describe en un archivo *xml*, el que se facilita su creación mediante una herramienta gráfica integrada en el ADT, esa interfaz cuenta con una clase en java, la cual es del tipo (hereda) *Activity*, aquí se definen las acciones que se toman con cada interacción del usuario con la interfaz. La declaración como *Activity* se debe hacer el *AndroidManifest*.

Una buena parte del desarrollo se enfocó en hacer funcionar correctamente la cámara del dispositivo, pues utilizando la clase del controlador de la cámara de forma simple, se obtuvo una imagen rotada y distorsionada, este problema es sólo de despliegue en el dispositivo, pues al tomar la fotografía esta se guarda adecuadamente. La solución para la distorsión de la imagen fue adecuar el aspecto de la cámara al espacio asignado para la previsualización en la aplicación. Además se debió ajustar los parámetros de enfoque, utilización de flash y la resolución de la fotografía a tomar, mientras mejor resolución mejor debiesen ser los resultados aunque también es mas costoso enviar la fotografía a través de la red.

4.1.3 Control de versiones

Para el control de versiones se utilizó la herramienta git, para cada módulo del sistema se creó un repositorio público en la plataforma github, a los que se puede acceder en: https://github.com/sebavp/Plantas-Libs, https://github.com/sebavp/Plants-Dal, https://github.com/sebavp/Plants-Web, https://github.com/sebavp/Plants-Front.

4.2 Poblamiento de la base de datos.

Para esta fase se tomaron fotografías de alta resolución (4000x3000 pixeles aproximadamente), utilizando para ello una cámara fotográfica “Pentax Optio S12”. Las muestras fueron tomadas desde el jardín Mapulemu en el Parque Metropolitano de Santiago, donde cada ejemplar se encuentra identificado con

una señalética donde se indica el nombre común, el nombre científico, la familia y su estado de conservación, un ejemplo de esto se pude ver en la Figura 4.1.

Figura 4.1: Ejemplo de señalética identificando una especie, Jardín Mapulemu

Las imágenes obtenidas fueron luego procesadas por el mismo sistema a utilizar para la identificación, salvo que los datos obtenidos fueron guardados en la base de datos para ser la referencia con la que se contrastarán las muestras.

4.3 Pruebas

Las pruebas tienen como objeto corroborar la utilidad del software, es decir, que se entregue la especie a la que corresponde la hoja consultada dentro de la lista con la que la aplicación responde. En cada ronda de pruebas se toman 3 muestras de cada una de las 15 especies de las que se tienen registros y se registra en qué lugar de la lista aparece la especie por la que se está consultando, o si no aparece.

La siguiente fase fueron las pruebas, básicamente se necesita medir que tan útil es la aplicación, o sea, que si se fotografía una especie determinada, ésta esté entre las respuestas que la aplicación entrega como posibles identificaciones.

Para llevar a cabo las pruebas se intentó simular un escenario lo más parecido posible al que se esperaría que un usuario final enfrentaría al utilizar la aplicación, se instaló el sistema en un servidor proveído por “Amazon web services (AWS)”, se instaló la aplicación en un dispositivo “Samsung Galaxy Ace

GT-S5830” y se procedió a probar la aplicación tomando muestras desde el jardín Mapulemu, mismo lugar donde se tomaron las muestras que se incluyeron en la base de datos, se hicieron tres rondas de pruebas, entre las cuales se implementaron mejoras en el sistema.

La primera ronda de pruebas dio como resultado que la aplicación no es capaz de identificar especies, las respuestas parecen ser más bien azarosas. De las 45 búsquedas efectuadas sólo 2 entregaron como resultado la especie consultada. Evidentemente la implementación era deficiente y no contemplaba trabajar con imágenes reales de búsqueda, pues hasta este punto las pruebas piloto habían sido hechas con imágenes referenciales de alta calidad, no comparables con los problemas de luminosidad y definición de las tomadas por un equipo como el que se utilizó para las pruebas.

Las primeras sospechas sobre dónde se produce la falla es en el módulo de fragmentación de la imagen, o sea, en la separación de la hoja y el fondo de la imagen. Al analizar esta posibilidad se dio con que el simple criterio de umbral no es suficiente, pues las variaciones de luz, sombra y reflejos que puede presentar la imagen distan mucho de las obtenidas inicialmente en condiciones de “laboratorio”. Se hizo necesario, entonces, buscar una alternativa que permitiera hacer una correcta fragmentación, tras una nueva revisión bibliográfica se dio con el algoritmo de Maximización de la esperanza (ver sección 2.3.1). La implementación de dicho algoritmo resultó ser mucho mas eficaz , dando como resultado contornos más precisos, cercanos a los reales y esto se ve reflejado en el aumento de los aciertos de la búsqueda en la segunda ronda de pruebas (ver Tabla 4.1), en la Figura 4.2 se puede ver la comparación entre los resultados de ambos algoritmos para varias muestras.

Con la fragmentación mejorada, se dio paso a una segunda ronda de pruebas, las que siguieron entregando resultados negativos. En esta ocasión, de las 45 búsquedas sólo 11 entregaron la especie consultada entre los resultados. Al revisar la implementación del algoritmo de extracción de los EFD (sección 2.3.3), se concluyó que ése es correcto y que la reconstrucción de la forma de la hoja a partir de los descriptores es la esperada, como se muestra en la Figura 2.3. Se especuló entonces que tener sólo una imagen de referencia es demasiado poco, y que un mejor apronte sería tener un conjunto de referencias (en la práctica se usaron entre 3 y 5, pero mientras más grande el conjunto se suponen mejores resultados) para cada especie (ver sección 2.4).

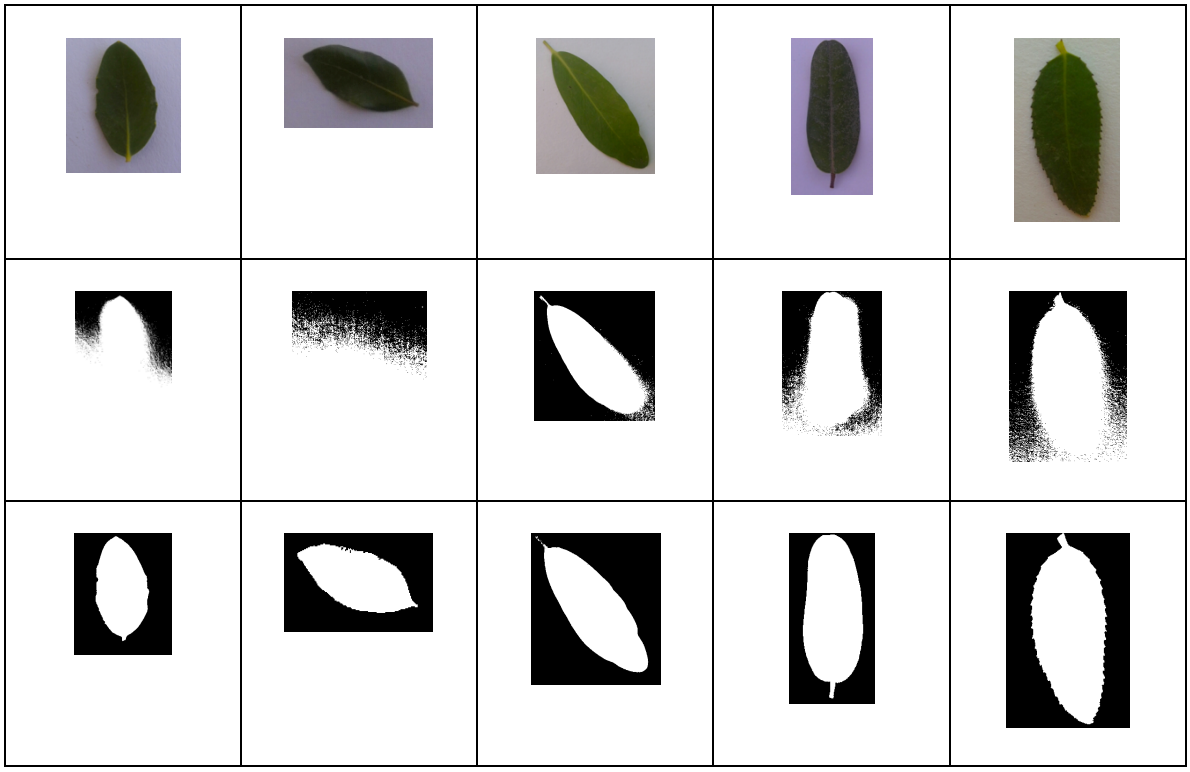

Figura 4.2: Comparación entre resultados de distintas implementaciones de segmentación

Finalmente, con las mejoras implementadas, se dio paso a una tercera ronda de pruebas. La mejora es notable los resultados se resumen en la Tabla 4.2, donde se puede ver que más del 85% de las consultas fueron respondidas correctamente, aunque en muy pocas ocasiones (3) la especie consultada aparece como primer resultado.

En la Tabla 4.1 se muestran los resultados de las pruebas en detalle, divididos en tres rondas, cada ronda consistió en tres búsquedas con cada una de las especies como muestra, entre cada una de las rondas se implementó una de las mejoras descritas en esta sección, es por eso que se observan mejoras también en la precisión de las búsquedas.

	1	2	3	4	5	6	7	8	9
Escallonia Rubra	-	-	-	-	-	-	3	2	3
Maytenus Boaria	-	-	-	-	2	-	2	2	-
Cryptocaria Alba	-	-	-	-	-	-	3	3	3
Escallonia Ilinita	-	-	-	-	-	-	-	3	4
Quillja Saponaria	-	-	-	-	-	3	1	4	2
Schinus Latifolius	-	2	-	-	-	-	1	2	4
Colliguaya Odorifera	-	-	-	3	-	1	2	3	2
Prunus Cerasifera	-	-	1	-	-	-	3	2	4
Pitavia Punctata	-	-	-	1	1	-	2	3	-
Myrceugenia Planipes	-	-	-	-	1	4	2	3	3
Tropaeolum Cilatum	-	-	-	-	-	-	-	2	3
Drimys Winteri	-	-	-	4	-	-	3	2	-
Aextoxicon Punctatum	-	-	-	-	-	3	2	-	3
Otholobium Glandulosum	-	-	-	-	3	-	3	2	3
Platanus Orientalis	-	-	-	-	-	-	1	3	3

Tabla 4.1: Resultados de las pruebas en detalle
(Cada fila es la especie consultada, cada columna es el número de la muestra, del 1 al 3 corresponde a la 1ª ronda, del 4 al 6 a la 2ª y del 7 al 9 a la 3ª, en cada celda se muestra la posición en que aparece la especie consultada en la lista de resultados o un guión (-) en caso de no aparecer.)

1º	2º	3º	4º	Total aciertos	Sin acertar	Total búsquedas	Porcentaje aciertos (%)
3	14	18	4	39	6	45	86,6

Tabla 4.2: Resumen del resultado de la tercera ronda de pruebas.

4.4 Prototipo

Se ha implementado un prototipo de aplicación para el sistema operativo Android. La aplicación cuenta con una base de datos de 15 especies, la mayoría nativas de Chile, y otras que han sido introducidas y que tienen bastante presencia en la zona central del país.

La aplicación empieza por mostrar la cámara al usuario para tomar una foto, el usuario ahora tiene la opción de enviar la foto para la identificación o tomar una nueva, esta última opción devuelve la aplicación a su estado inicial. Por otra parte si se envía la foto, ésta se procesa en el servidor, primero se separa el fondo del objeto, en este caso la hoja, a la que después se le extrae la característica de su forma mediante el algoritmo conocido como “descriptores elípticos de Fourier”. Se ha obtenido entonces un vector que se contrasta con los vectores almacenados en la base de datos, mediante una simple comparación con distancia euclidiana se determina las 2 especies más cercanas, estos resultados se devuelven al usuario a quien se le presentan los nombres, las fotos y los links a la entrada en Wikipedia de la especie, la identificación final, entonces, queda a cargo del usuario.

A diferencia de Leafsnap, la aplicación que sirvió de guía para este desarrollo, no se implementó un módulo que discrimine las fotos que efectivamente contienen una hoja de las que no, pues tal desarrollo es bastante costoso y no se contó con los recursos para emprender tal tarea, que de todas formas queda como un desarrollo futuro para una versión definitiva de la aplicación.

5 Conclusiones

5.1 Resultados

En esta memoria se logró, en primer lugar, la implementación de algoritmos capaces de segmentar una imagen, distinguiendo un objeto fotografiado sobre un fondo de color claro, más o menos uniforme; extraer las características de forma del objeto como una descripción matemática; y finalmente hacer una búsqueda por cercanía en una base de datos de objetos del mismo tipo, en este caso hojas de plantas.

Estas implementaciones fueron integradas en un servicio web, donde se da la posibilidad de hacer búsquedas en la base de datos a partir de una imagen que se le envía al servidor, este servicio implementa el estándar HTTP, por lo que puede ser reutilizado por cualquier aplicación o usuario.

Por otra parte, se creó una aplicación para dispositivos móviles, que utilizan el sistema operativo android, que interactúa con el servicio web, enviando fotografías tomadas con el mismo dispositivo al servidor y desplegando los resultados de la búsqueda, para que el usuario pueda hacer la identificación final de la especie que se ha fotografiado entre las opciones que la aplicación ha entregado.

También se compararon distintas formas de implementar el sistema, se discutió sobre qué arquitectura de software es la más adecuada, concluyendo que para esta etapa tal arquitectura es la presentada en (), en la que todo el procesamiento de imágenes y operaciones en base de datos se hacen en un servidor web y una aplicación móvil es utilizada como interfaz usuaria para el servicio; se diseñó un modelo de datos que permite agregar distintas muestras para la misma entidad(especie), haciendo que la búsqueda sea más precisa.

El sistema desarrollado fue capaz de entregar resultados correctos en la amplia mayoría de los casos: 86,6 % de las pruebas entregaron la especie consultada entre los resultados de búsqueda, teniendo en cuenta eso sí lo limitada de la cobertura de la base de datos, se hace necesario hacer pruebas con una mayor cantidad de muestras, aunque la evolución de los resultados conforme se fueron implementando mejoras en el sistema fueron bastante claros y dan pie a concluir que se logró el objetivo de desarrollar un servicio capaz de identificar especies con la sola fotografía de una hoja.

Finalmente el sistema desarrollado constituye un prototipo que sirve como guía para la producción de una versión más acabada del mismo apto para la utilización por parte del público en general. Este prototipo ha servido para mostrar que el reconocimiento de especies vegetales basándose en fotografías tomadas por dispositivos móviles es posible y además factible de implementar, dejando en claro, eso sí, que el sistema de reconocimiento de imágenes más poderoso conocido es por lejos el ojo humano.

5.2 Dificultades presentadas

En esta memoria los desafíos técnicos, para el autor, se concentraron en el mundo de la visión por computador, específicamente en el reconocimiento de imágenes, campo en el que no se tenía real experiencia, por lo que fue necesario estudiar conceptos desde una fase muy básica y el reestudio de conceptos matemáticos que no habían sido aplicados en bastante tiempo. Fue necesario también aprender a hacer desarrollo para android lo que terminó requiriendo más tiempo del presupuestado.

Por otra parte también se debió enfrentar el déficit de recursos, no se tuvo acceso libre a un equipo corriendo android, sino hasta bien avanzado el desarrollo, siendo esto fundamental para avanzar a buen ritmo en la implementación de esa parte de la memoria.

5.3 Desarrollos futuros

Para que el prototipo presentado en esta memoria llegue a ser una aplicación útil para los usuarios es necesario completar una serie de pasos, que por falta de recursos, tiempo o complejidad no se alcanzaron a cubrir en este desarrollo:

1. Interfaz de usuario, aunque la interfaz de usuario es natural y se asemeja a la de leafsnap, es necesario hacer pruebas mas formales, con un grupo de usuarios más amplio y que además incorpore un diseño visual más atractivo.
2. Ampliación de la base de datos, al final de este desarrollo, la base de datos cuenta con 15 especies distintas, conjunto que resultó ser de bastante utilidad dados los objetivos planteados, a pesar de lo reducido, de las cuales hay entre 2 y 5 muestras. Esta cantidad es claramente insuficiente pues se estima que en Chile hay más de 5000 especies distintas de plantas (y esto no se considera tan diverso comparado con otros lugares del mundo). Además el aumento de muestras se traduce en que la identificación sea más precisa.

3. Ligado al punto 2 también está el encontrar una fórmula que permita que la construcción de la base de datos se haga de forma colaborativa, encontrando un grupo de usuarios que tenga la responsabilidad de hacer una correcta identificación de especies que se agreguen a la base de datos, ya sea como una especie nueva o como muestra de una ya presente.
4. Llevar la base de datos a un formato entendible por otras máquinas, conectándola a una red semántica ("linked data"), haciendo posible una mayor disponibilidad de los datos recolectados para su uso en otros contextos.
5. Performance, el tiempo que demora cada búsqueda puede ser mejorado, tomando en cuenta que en condiciones de conectividad limitada, el envío de una imagen de alta resolución puede tardar demasiado, además de consumir los saldos de datos, que es la forma en que las compañías proveedoras de servicio de internet móvil tarifican el uso, se pueden hacer optimizaciones en la transferencia de datos desde el dispositivo al servidor y viceversa, por ejemplo comprimiendo las imágenes, dejándolas de un tamaño aún adecuado para la identificación (el que tendría que ser encontrado), también se puede optimizar la segmentación y la extracción de características de la imagen. Por último se pueden explorar formas de optimizar la búsqueda en la base de datos, ya sea dándole otra estructura o utilizando otros motores de bases de datos, no necesariamente relacionales (MongoDB, CouchDB, BigData, Neo4j, etc.).
6. El sistema desarrollado en esta memoria puede ser fácilmente adaptado para que pueda reconocer otros objetos, distintos a las hojas, basándose en su silueta, como por ejemplo vehículos, animales, instrumentos musicales, herramientas, etc.

Referencias

[1] www.leafsnap.com

[2] M. Nixon, A. S. Aguado: “Feature extraction and image processing”, 2º edición, pp. 77-81, 184-185, 285-311 Elsevier, Londres, 2008.

[3] J. S. Cope, D. P. A. Corney, J. Y. Clark, P. Remagnino, P. Wilkin: "Plant species identification using digital morphometrics: A review", Expert Systems with Applications, 2012, Vol. 39, Issue 8, pp. 7562-7573.

[4] C Direkoglu, M Nixon: “Shape classification via image-based multiscale description“, Pattern Recognition, 2011, Vol. 44, Issue 9, pp. 2134–2146.

[5] Y. Tao, K. Yi, C. Sheng, P. Kalnis: “Quality and efficiency in high dimensional nearest neighbor search”, Proceedings of the 2009 ACM SIGMOD International Conference on Management of data, 2009, pp. 563-576.

[6] www.postgresql.org

[7] www.appcelerator.com

[8] N. Kumar, P. Belhumeur, A. Biswas, D. Jacobs, W. Kress, I. Lopez and J. Soares: “Leafsnap: A Computer Vision System for Automatic Plant Species Identification”, 12th European Conference on Computer Vision, Florence, Italy, October 7-13, 2012, Proceedings, Part II, pp. 502-516.

[9] McLachlan, Geoffrey J.; Krishnan, Thriyambakam; NG, See Ket (2004): “The EM Algorithm”, Papers / Humboldt-Universität Berlin, Center for Applied Statistics and Economics (CASE), No. 2004,24, http://hdl.handle.net/10419/22198

Anexo A: Especies incluidas en la base de datos

 Figura: Cryptocaria alba (Peumo)	 Figura: Schinus latifolius (Molle)	 Figura: Quillaja saponaria (Quillay)
 Figura: Escallonia illinita (Barraco)	 Figura: Myrtus communis (Arrayán)	 Figura: Colliguaja odorifera (Colliguay)
 Figura: Escallonia rubra (Siete camisas)	 Figura: Myrecugenia Planiples (Patagua Valdiviana)	 Figura: Drimys winteri (Canelo)
 Figura: Maytenus boaria (Maitén)	 Figura: Tropaeolum ciliatum (Pajarito)	 Figura: Aextoxicon punctatum (Olivillo)
 Figura: Prunus cerasifera (Cerezo)	 Figura: Pitavia punctata (Pitao)	 Figura: Platanus orientalis (Plátano oriental)