Sistema de Reconocimiento de Setas (FindMush)

Guillermo Manjón Mateo
Ingeniería Informática

David Isern Alarcón

9 de junio de 2015

FICHA DEL TRABAJO FINAL

Título del trabajo:	Sistema de Reconocimiento de Setas
Nombre del autor:	Guillermo Manjón Mateo
Nombre del consultor:	David Isern Alarcón
Fecha de entrega (mm/aaaa):	06/2015
Área del Trabajo Final:	Inteligencia Artificial
Titulación:	*Ingeniería Informática*

Resumen del Trabajo (máximo 250 palabras):

El trabajo consiste en un Sistema de Reconocimiento de Setas basado en una aplicación Android, una aplicación de escritorio y un servidor.

El servidor contiene un clasificador, que se va alimentando con nuevas muestras desde la aplicación de escritorio, y clasifica las muestras que le envía la aplicación Android.

Para que la imagen de una seta sea procesable la transformamos primero a números cuantificables y comparables. Primero se segmenta la seta usando uso de la librería OpenCV y el algoritmo GrabCut, que utiliza un cuadrado en el que se enmarca la imagen a segmentar. Este cuadrado lo proporciona el usuario, y es la única interacción necesaria con él para la segmentación de la seta.

Una vez se tiene la seta segmentada se divide mediante diferentes algoritmos en sus 3 partes principales (sombrero, pie y escamas), se parametrizan transofrmando imagenes a números (dimensiones, colores y estadísticas), y estos números son los que se envían al servidor, ya sea para alimentar el clasificador (desde la aplicación de escritorio) o para clasificar una muestra (aplicación Android).

Abstract (in English, 250 words or less):

The project consists of a Mushroom Recognition System based on an Android app, a desktop app and a server.

The server has a classificator, which is fed with new samples coming from the desktop app, and classifies the samples send by the Android app.

For a mushroom image to be procesable we convert it to cuantifiable and comparable numbers. Firstly, we segmentate the mushroom using the OpenCV library and GrabCut algorithm, which utilizes a rectangle in which to frame the image to be segmentated. This rectangle is provided by the user, and is the only user interaction needed for the segmentation.

Once we have the mushroom segmentated, we divide it by means of different algorithms in its main three parts (cap, stem and scales), we parameterize it translating images into numbers (dimensions, colors and statistics), and those numbers are the ones to be sent to the server, whether it is for feed the classificator (desktop app) or for classify a sample (Android app).

Palabras clave (entre 4 y 8):

seta, clasificador, segmentación, weka, opencv, android

Índice

Table of Contents

Capítulo 1.Introducción....1

1.1.Resumen....1

1.2.Motivación....1

1.3.Objetivos....1

1.4.Tareas....2

1.5.Planificación....3

Capítulo 2.Tecnologías, Modelo de Datos y Segmentación....4

2.1.Presentación de las tecnologías que se usarán para los diferentes subsistemas....4

2.1.1.Aplicación móvil....4

2.1.2.Servidor....4

2.1.3.Aplicación de escritorio....4

2.2.Especificación del dominio reducido que contemplará el PFC. 5

2.3.Definición del Modelo de Datos....7

2.3.1.Definir que partes de la seta se van a medir....7

2.3.2.Definir las estructuras y datos que identificarán las diferentes partes de la seta (parametrización)....7

2.3.3.Definir una estructura de BBDD que se ajuste a los datos parametrizados....8

2.4.Sistema de Segmentación de la seta....9

2.4.1.Pre-procesado: Preparar la imagen para ser segmentada eficaz y eficientemente....9

2.4.1.1.BasicLutFilter....9

2.4.1.2.BinaryFilter....10

2.4.1.3.BlurFilter....11

2.4.1.4.ContrastFilter....11

2.4.1.5.GrayscaleFilter....12

2.4.1.6.HistogramBasedLutFilter....13

2.4.1.7.ResizeFilter....13

2.4.2.Segmentación de la seta con respecto al resto de la imagen....14

2.4.3.Segmentación de las diferentes partes de la seta....15

2.4.3.1.Pre-cálculos iniciales....15

2.4.3.2.Segmentación del sombrero....18

2.4.3.3.Segmentación del pie....19

2.4.3.4.Segmentación de las escamas....20

2.4.3.5.Código fuente de la segmentación....21

Capítulo 3.Parametrización, clasificación, Android app y servidor....22

3.1.Sistema de parametrización de la seta....22

3.1.1.Implementación de las parametrizaciones de las diferentes de partes de la seta 22

3.1.2.Integración con el Sistema de Segmentación....22

3.2.Sistema de Clasificación de la seta....23
3.2.1.Presentar los posibles algoritmos de clasificación....23
3.2.2.Elegir razonadamente uno o varios de ellos....23
3.2.3.Implementar el algoritmo de clasificación....23
3.3.Implementación de la App de Android....24
3.3.1.Desarrollo de las pantallas....24
3.3.1.1.Pantalla principal....24
3.3.1.2.Pantalla de selección de la imagen....24
3.3.1.3.Pantalla de selección del rectángulo....24
3.3.1.4.Pantalla de progresso del proceso....24
3.3.1.5.Pantalla de resultados....24
3.3.2.Integración con el Sistema de Segmentación....25
3.4.Implementación del Servidor....25
3.4.1. Desarrollo de los servicios web vía REST....25
3.4.2. Integración con la BBDD....25
3.4.3. Integración con el Sistema de Clasificación....25

Capítulo 4.Resultados....26

Capítulo 5.Conclusiones....30

Bibliografía....31

Anexo A: Código fuente....32

Anexo B: Script de Base de Datos....34

Anexo C: Manual de Instalación....36
Server....36
Desktop App....37
Android App....37

Anexo D: Manual de Usuario....38
Desktop App....38
Android App....42

Índice de ilustraciones

Ilustración 1: Amanita Muscaria....5
Ilustración 2: Amanita Caesarea....5
Ilustración 3: Imagen propicia....6
Ilustración 4: Imagen no propicia....6
Ilustración 5: Partes de la seta....7
Ilustración 6: Basic LUT Filter....10
Ilustración 7: BinaryFilter....10
Ilustración 8: Blur Filter....11
Ilustración 9: Contrast Filter....12
Ilustración 10: Grayscale Filter....12
Ilustración 11: Histogram Based LUT Filter....13
Ilustración 12: Resize Filter....14
Ilustración 13: Recuadro segmentacion....15
Ilustración 14: Whole Mushroom Segmentated....15
Ilustración 15: Puntos cardinales....16
Ilustración 16: Contorno imagen segmentada....17
Ilustración 17: Cuadrado segmentación sombrero....19
Ilustración 18: Sombrero segmentado....19
Ilustración 19: Cuadrado segmentación pie....20
Ilustración 20: Pie segmentado....20
Ilustración 21: Segmentacion escamas: original....21
Ilustración 22: Segmentacion escamas: Grayscale Filter....21
Ilustración 23: Segmentacion escamas: Contrast Filter....21
Ilustración 24: Segmentacion escamas: Binary Filter....21
Ilustración 25: Segmentacion escamas: resultado....21
Ilustración 26: AC - muestra 1....26
Ilustración 27: AC - muestra 2....26
Ilustración 28: AC - muestra 3....26
Ilustración 29: AC - muestra 5....26
Ilustración 30: AC - muestra 6....26
Ilustración 31: AC - muestra 4....26
Ilustración 32: AM - muestra 1....26
Ilustración 33: AM – muestra 3....26
Ilustración 34: AM - muestra 2....26
Ilustración 35: AM - muestra 4....27
Ilustración 36: AM - muestra 5....27
Ilustración 37: AM - muestra 6....27
Ilustración 38: AC - muestra 9....28
Ilustración 39: AC - muestra 8....28
Ilustración 40: AC - muestra 7....28
Ilustración 41: AC - muestra 11....28
Ilustración 42: AC - muestra 12....28
Ilustración 43: AC - muestra 10....28
Ilustración 44: AM - muestra 8....28
Ilustración 45: AM - muestra 9....28
Ilustración 46: AM - muestra 7....28
Ilustración 47: AM - muestra 11....29

Ilustración 48: AM - muestra 10....29
Ilustración 49: AM - muestra 12....29
Ilustración 50: Desktop App - Main....38
Ilustración 51: Desktop App - Set server IP....39
Ilustración 52: Desktop App - Confirmar recuadro....40
Ilustración 53: Desktop App - Resultado segmentación....41
Ilustración 54: Android App - Main....42
Ilustración 55: Android App - Seleccionar recuadro....43
Ilustración 56: Android App - Confirmar selección....43
Ilustración 57: Android App - Progreso segmentación....44
Ilustración 58: Android App - Límite sombrero izquierda....45
Ilustración 59: Android App - Límite sombrero derecho....45
Ilustración 60: Android App - Parametrizando....45

Capítulo 1. Introducción

1.1. Resumen

Este proyecto pretende elaborar un sistema de reconocimiento de setas con una estructura escalable que permita ir añadiendo diferentes tipos de setas para su identificación con poco esfuerzo.

El sistema consistirá de los siguientes componentes:

1. **Android App**: Una aplicación Android que porcesará la foto de una seta para su clasificación. La propia App llevará a cabo la segmentación, parametrizará la seta, y enviará los datos a un servidor para su consulta.

2. **Server**: Un servidor que constará de:
    a. Un base de datos que contendrá todas las muestras de setas conocidas que vayamos insertando.
    b. Un algoritmo de clasificación que se inicializará con diferentes muestras conocidas y a la que se le pasará cada seta recibida por la Android App para que intente clasificar correctamente.

3. **Desktop App**: Un aplicación de escritorio que servirá para introducir las setas de muestra parametrizadas en la base de datos. Mediante petición al servidor. Estos datos serán los que luego se usarán para inicializar el clasificador.

1.2. Motivación

La motivación no es más que generar una aplicación móvil con utilidad real y extensible en el futuro. Poder identificar setas con un porcentaje de acierto elevado y hacerlo escalable para poder añadir más y más setas en el futuro.

No existe actualmente en el mercado ninguna aplicación similar que pueda indicar la especie de una seta a partir de una foto.

1.3. Objetivos

Uno de los objetivos de este proyecto es establecer las bases para una idea propia original que consiste en una aplicación móvil de reconocimiento de setas a través de una fotografía.

El objetivo final de este proyecto va más allá del alcance de este proyecto, y es realizar un sistema de reconocimiento de setas que pueda ayudar a cualquier a identificar una seta que se encuentre, ya sea casualmente o habiendo ido expresamente al monte a recogerlas.

1.4. Tareas

Las tareas que se llevarán a cabo durante este proyecto son las siguientes:

1. Presentación de las tecnologías que se usarán para los diferentes subsistemas
2. Especificación del dominio reducido que contemplará el PFC.
3. Definición del Modelo de Datos
   - 3.1. Definir que partes de la seta se van a medir
   - 3.2. Definir las estructuras y datos que identificarán las diferentes partes de la seta (parametrización)
   - 3.3. Definir una estructura de BBDD que se ajuste a los datos parametrizados.
4. Sistema de Segmentación de la seta
   - 4.1. Pre-procesado: Preparar la imagen para ser segmentada eficaz y eficientemente.
   - 4.2. Segmentación de la seta con respecto al resto de la imagen.
   - 4.3. Segmentación de las diferentes partes de la seta
   - 4.4. Pruebas unitarias del Sistema de segmentación
5. Sistema de Parametrización de la seta
   - 5.1. Implementación de las parametrizaciones de las diferentes de partes de la seta
   - 5.2. Integración con el Sistema de Segmentación
   - 5.3. Pruebas unitarias del Sistema de Parametrización
6. Sistema de Clasificación de la seta
   - 6.1. Presentar los posibles algoritmos de clasificación
   - 6.2. Elegir razonadamente uno o varios de ellos
   - 6.3. Implementar el algoritmo de clasificación
   - 6.4. Pruebas unitarias del algoritmo de clasificación
7. Implementación de la App de Android
   - 7.1. Desarrollo de las pantallas
   - 7.2. Integración con el Sistema de Segmentación
8. Implementación del Servidor
   - 8.1. Desarrollo de los servicios web vía REST
   - 8.2. Integración con la BBDD
   - 8.3. Integración con el Sistema de Clasificación
9. Implementación de la Tool para añadir muestras
   - 9.1. Definición de una interfaz muy sencilla.
   - 9.2. Integración con el Sistema de Segmentación
   - 9.3. Integración con el Sistema de Parametrización
   - 9.4. Integración con la BBDD
10. Exposición de los siguientes pasos a seguir para continuar con el proyecto
11. Repaso y elaboración final de la memoria

12. Elaboración de la presentación del PFC

1.5. Planificación

Las fechas claves durante el PFC, y las tareas que se planificarán como completadas en cada una de esas fechas son las siguientes:

Entrega PAC2:	8 abril 2015	Tareas desde la 1 a la 4
Entrega PAC3:	6 mayo 2015	Tareas desde la 5 a la 8
Entrega final del PFC:	3 junio 2015	Tareas desde la 9 a la 12

Capítulo 2. Tecnologías, Modelo de Datos y Segmentación

2.1. Presentación de las tecnologías que se usarán para los diferentes subsistemas

Como ya se ha comentado en un punto anterior, el producto a entregar constará de los siguientes sistemas:

2.1.1. Aplicación móvil

La aplicación móvil se implementará únicamente para sistemas Android, por lo que se realizará usando Android API y la librería OpenCV para Android.

2.1.2. Servidor

El servidor tiene toda la base de conocimiento para decidir en qué categoría se clasificará la seta, por lo que constará de dos partes:

- Servicios Web: con los que se comunicarán tanto la aplicación móvil como la aplicación de escritorio. Se usará para ello una arquitectura REST implementada en Java 6 y corriendo sobre un servidor JBoss 7.1.1

- Base de Datos: Debido al corto alcance y reducido dominio del presente proyecto, la base de datos se implementará con SQLite, por a su sencillez de uso, configuración y ejecución.

2.1.3. Aplicación de escritorio

La aplicación de escritorio se llevará a cabo en Java 6 y se usará la librería Swing, ya que los requerimientos de prestaciones y disponibilidad de elementos gráficos son en general reducidos.

2.2. Especificación del dominio reducido que contemplará el PFC.

El presente PFC contemplará la distinción únicamente entre dos setas.

La Amanita Muscaria:

Ilustración 1: Amanita Muscaria

Y la Amanita Caesarea:

Ilustración 2: Amanita Caesarea

Como se comentó en el Resumen, el sistema será escalable y añadir nuevas setas para su reconocimiento supone muy poco esfuerzo. Por lo que si se quieren reconocer más setas

solamente habrá que darlas de alta en base de datos , los **`enum`** de las aplicaciones y añadir sus setas de muestra.

Debido a la complejidad de la segmentación de todas las diferentes formas que puede tomar una seta, y al amplio modelo de arquitectura que abarca el proyecto (sistemas móviles, escritorio y servidor) la lógica de segmentación se ve reducida a funcionar correctamente con setas cuya forma esté bien definida y el fondo no tenga unos colores demasiado similares al de la seta. Por ejemplo, el sistema siempre segmentará bien la siguiente imagen:

Ilustración 3: Imagen propicia

Sin embargo esta otra es muy probable que de un error durante la segmentación:

Ilustración 4: Imagen no propicia

Se puede observar como la volva cubre casi la totalidad del pie, por lo que las partes que queremos diferenciar están poco o mal definidas. La volva ni siquiera pretendemos reconocerla, por lo que de esta imagen solo se pueden esperar malos resultados.

La imagen ha de tener un sombrero destacado y un pie bien definido.

2.3. Definición del Modelo de Datos

2.3.1. Definir que partes de la seta se van a medir

Para distinguir una seta de otra utilizaremos las partes más destacadas y características de cualquier seta.

- Sombrero (Cap): La parte superior con forma de sobrero.
- Pie o tallo (Stem): La parte que une el sombrero con la tierra.
- Escamas (Scales): Las pintas blancas o de otro color que llevan algunas setas sobre el sobrero.

Otras partes típicas como las láminas (o himenio), el anillo o la volva se han descartado por la dificultad de su identificación y segmentación dentro de la imagen.

En la siguiente imagen se pueden observar todas estas partes:

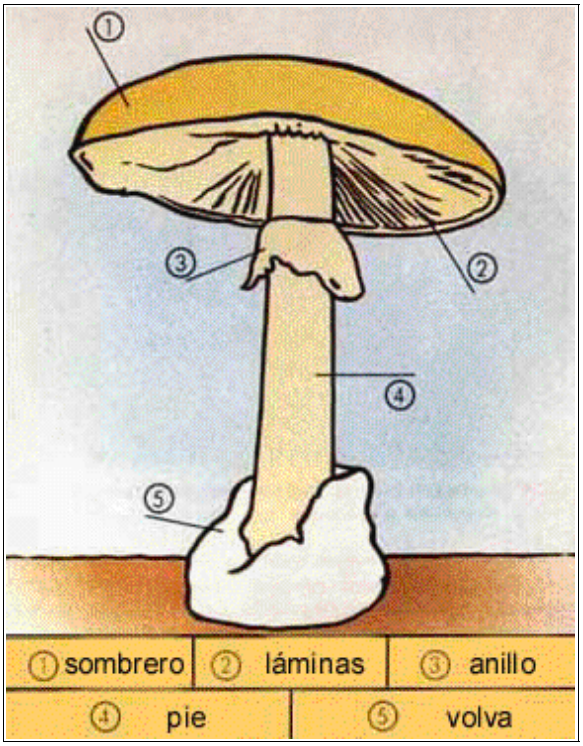


Ilustración 5: Partes de la seta

2.3.2. Definir las estructuras y datos que identificarán las diferentes partes de la seta (parametrización)

Todos las medidas espaciales (altura, anchura y tamaño -area-) se medirán proporcionalmente en base al tamaño total de la seta, de esta forma, las medidas serán invariantes, no cambian en función del tamaño de la imagen.
Para las diferentes partes de la seta que vamos a identificar se medirán los siguientes parámetros:

- Cap
  - Size: el tamaño del sombrero
  - Color: el color medio del sombrero
  - Width: el ancho del sombrero (es normal que esto sea muchas veces el 100%)
  - Height: la altura del sombrero

- Stem
  - Size: el tamaño del pie.
  - Color: el color medio del pie
  - Width: el ancho del pie
  - Height: la altura del pie

- Scales
  - Number: número de escamas detectadas por el sistema de segmentación que no sean puntos individuales.
  - Color: el color de las escamas. En el dominio que abarcaremos consideraremos como escamas aquellas cuyo color sea más claro que el resto del sombrero.
  - Biggest size: el tamaño de la escama más grande
  - Smallest size: el amaño de la escama más pequeña
  - Average size: tamaño medio de las escamas
  - Standard deviation: Desviación típica de las escamas.
  - Size: tamaño sumado de todas las escamas

2.3.3. Definir una estructura de BBDD que se ajuste a los datos parametrizados.

La función de la BBDD es alojar muestras parametrizadas de setas. Siendo esta la única información que necesitamos guardar las tablas que crearemos serán una para cada componente de la seta (CAP, STEM y SCALES) otra principal con los datos globales de la muestra MUSHROOM_SAMPLE con cuyo id se relacionan las 3 anteriores, y MUSHROOM_TYPE, que contiene las setas que el sistema puede reconocer con información de sus características principales para mostrar al usuario.

Se crea también una vista con todos las las tablas de SAMPLE ya relacionadas para poder hacer *queries* sencillas que obtengas todos los datos de las muestras en la misma fila.

Puede verse el script de creación de la base de datos en el Anexo B.

2.4. Sistema de Segmentación de la seta

Para la segmentación de imágenes se usará la librería OpenCV debido a que existe versión para Java y para Android, que son las dos plataformas en las que vamos a necesitar tratamiento de imágenes. La versión que usaremos será la 2.49.

2.4.1. Pre-procesado: Preparar la imagen para ser segmentada eficaz y eficientemente.

Para la imagen inicial únicamente haremos un redimensionado, ya que los algoritmos de segmentación utilizados mejoran considerablemente su rendimiento. Por lo tanto reduciremos el ancho a 500px cuando sea mayor, y el otro lado proporcionalmente. Este tamaño es suficiente para una buena segmentación y acelera el proceso de forma exponencial.

Otros pre-procesados son necesarios para segmentaciones de las diferentes partes. Para ello se ha creado una infraestructura de Filtros en Java de fácil aplicación para su facilidad de uso durante el desarrollo. Todos implementados por supuesto vía OpenCV. En el código de las aplicaciones se encuentra en el package **`filters.`**

Sobre todo estos filtros están siendo de utilidad para preparar el sombrero ya segmentado para la detección de las escamas. El principal problema es la diferencia de contraste , que hace que un brillo blanco en la imagen se interprete como imagen blanca, generando una escama muchas veces grande y que encapsula a otras escamas. Este problema se ha resuelta aumentando el *threshold* para el filtro binario, de esta forma se perderán escamas que no sean suficientemente claras, pero en una seta con escamas siguen detectándose la mayoría de ellas y el número de falsos positivos se reduce enormemente.

Los filtros implementados hasta ahora son los siguientes:

2.4.1.1. BasicLutFilter

Realiza un filtro basado en una tabla de correspondencias (Look-Up-Table: LUT). Se parametriza en base al numero de contenedores que se usarán en el rango 0-255. Por ejemplo, si se le pasa como parámetro 3 contenedores, modificará los colores en base a la siguiente tabla:

Antes	Después
0 - 85	0
86 - 170	86
171 - 255	171

Con esto se consigue básicamente reducir el número de colores sin perder la información básica de la imagen.

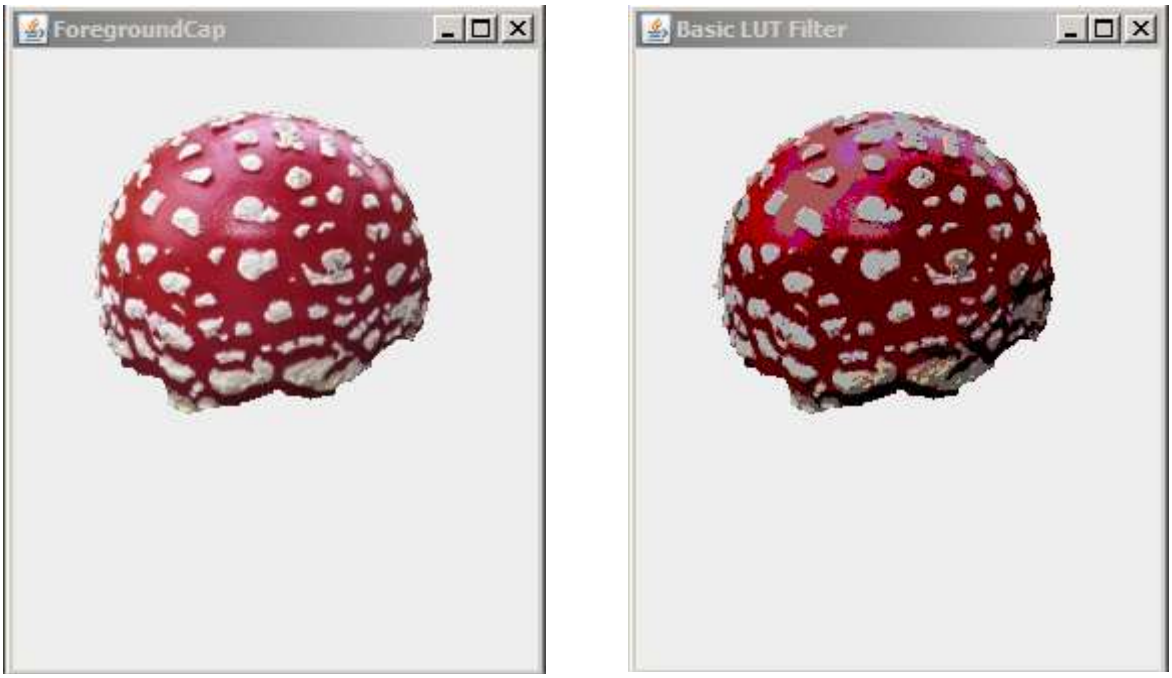


Ilustración 6: Basic LUT Filter

2.4.1.2. BinaryFilter

Genera una imagen binaria a partir de la imagen original. Es decir, la imagen resultante contendrá únicamente dos colores, el blanco y el negro.

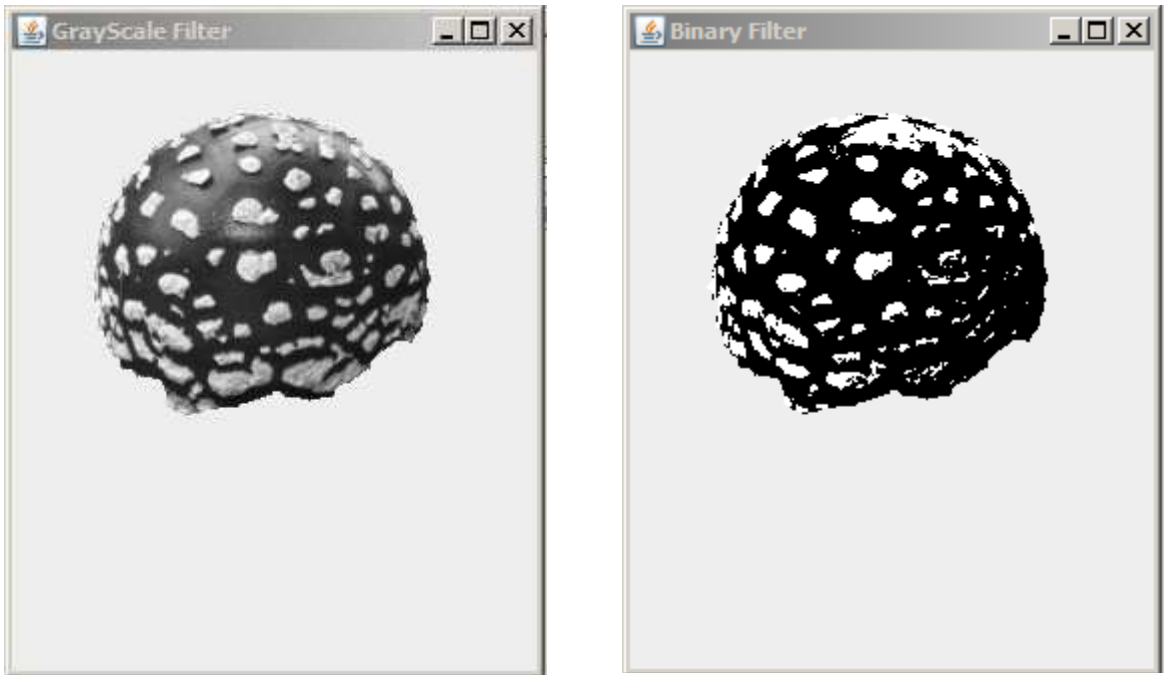


Ilustración 7: BinaryFilter

2.4.1.3. BlurFilter

Filtro que genera una imagen borrosa en base a una matriz de aplicación. Con este filtro se consiguen eliminar pequeña imperfecciones y darle un aspecto más suave que generará contornos más uniformes.

Por ejemplo, para una matriz de 3x3, el Blur Filter daría el siguiente resultado:

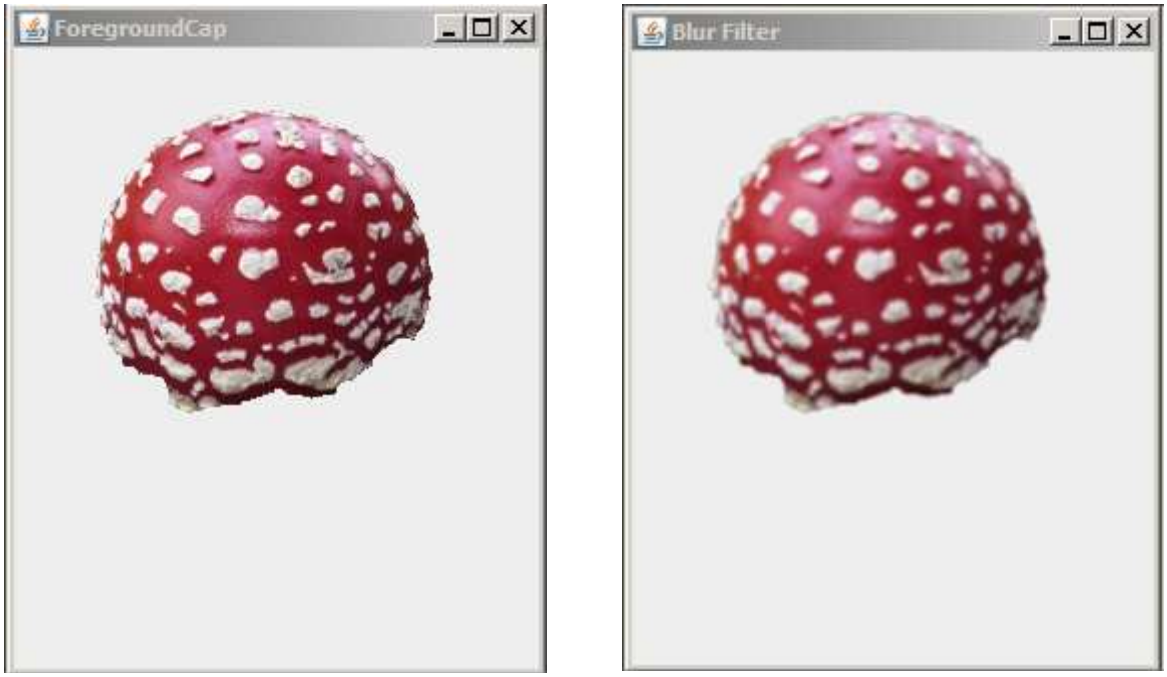
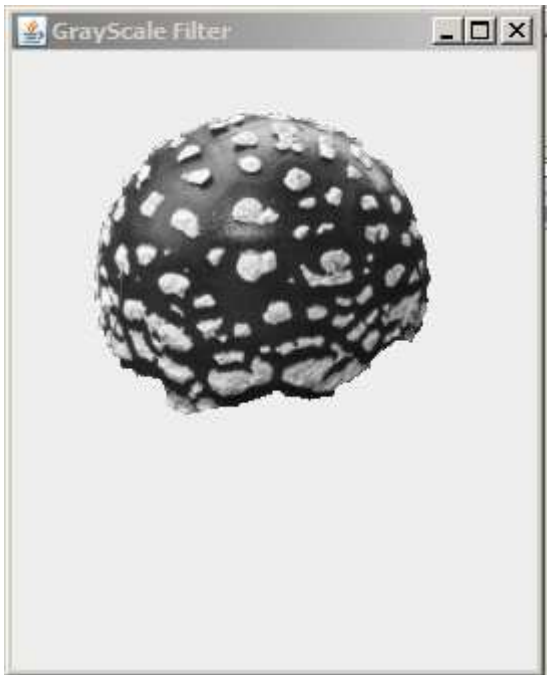


Ilustración 8: Blur Filter

2.4.1.4. ContrastFilter

Aumenta el contraste de la imagen. Aumentar el contraste nos ayuda a detectar mejor los límites entre áreas que se pretenden diferenciar.



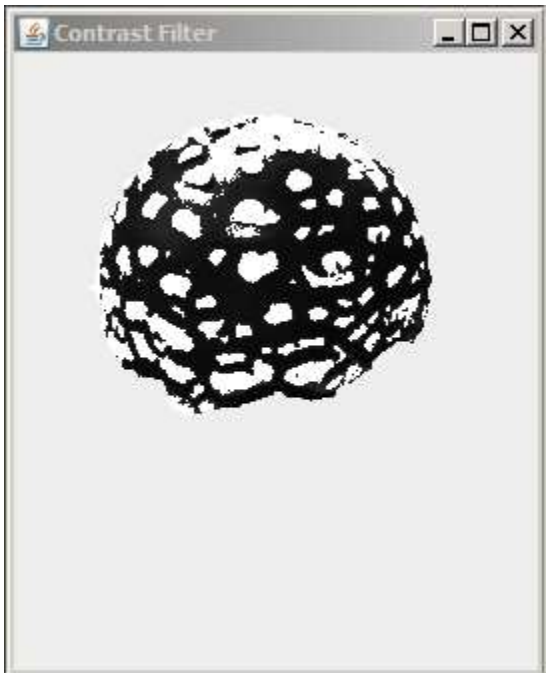


Ilustración 9: Contrast Filter

2.4.1.5. GrayscaleFilter

Convierte una imagen en color a escala de grises.

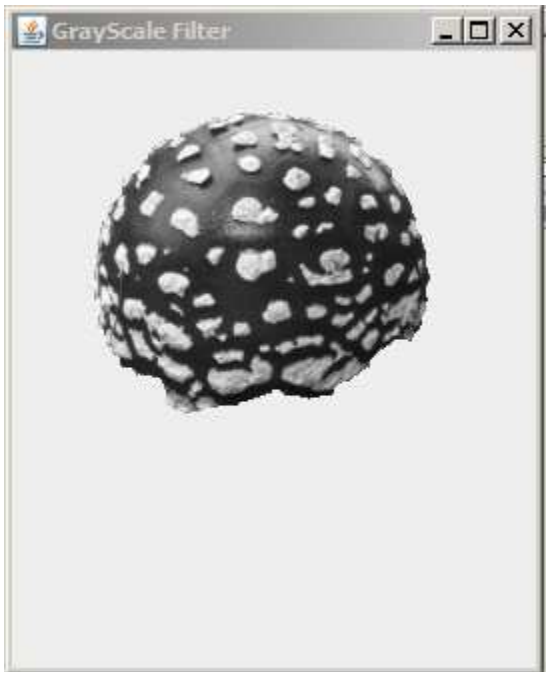


Ilustración 10: Grayscale Filter

2.4.1.6. HistogramBasedLutFilter

Similar al BasicLutFilter, pero en vez de utilizar rangos del mismo tamaño estos se calculan en base a la información proporcionada en el histograma.

Primero se calculan los colores más abundantes en la imagen, después se genera una tabla LUT que hace corresponder un color con su color abundante más cercano, y finalmente se aplica dicha tabla a la imagen.

La generación de esta tabla basada en el histograma no se ha encontrado en la librería y ha tenido que ser generada desde cero. No la aplicación de la tabla, sino su generación.

Por ejemplo, para una reducción de 4 colores máximos en cada canal, el filtro daría el siguiente resultado:

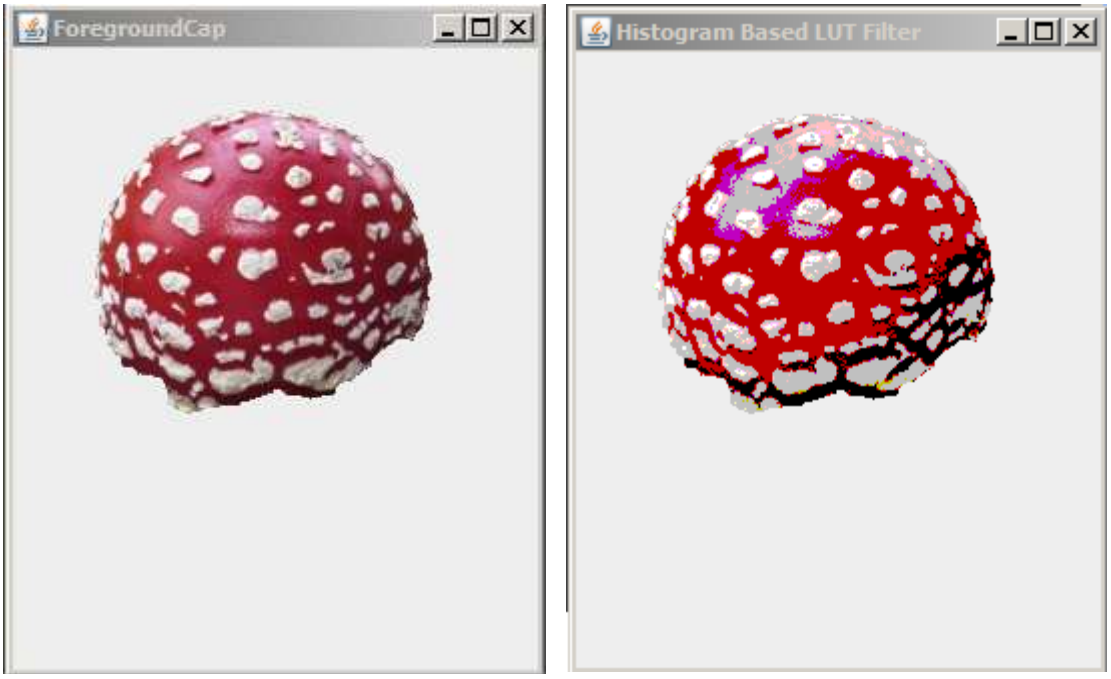


Ilustración 11: Histogram Based LUT Filter

2.4.1.7. ResizeFilter

Redimensiona la imagen original al tamaño deseado. Otra forma de reducir colores es reducir la imagen a por ejemplo 10x10 y volverla a aumentar. Se está estudiando aún la eficacia de este proceso como ayuda para la segmentación.

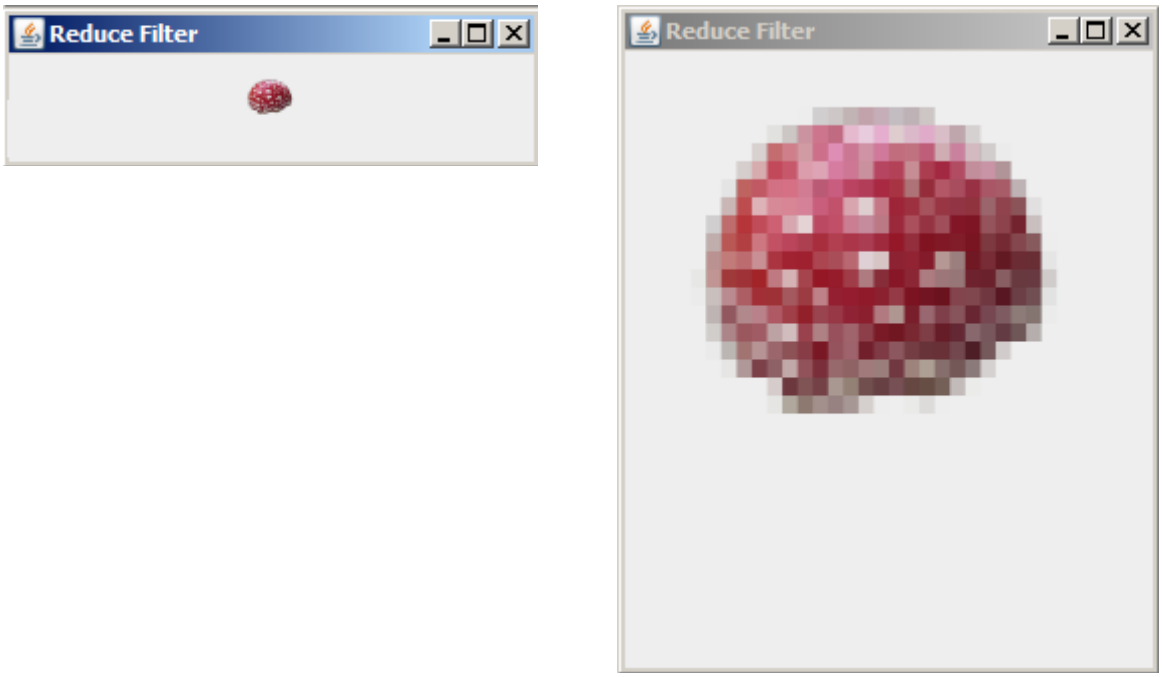


Ilustración 12: Resize Filter

2.4.2. Segmentación de la seta con respecto al resto de la imagen.

Para la segmentación principal de la seta se ha utilizado el algoritmo GrabCut, disponible en la librería OpenCV.

Este algoritmo se basa en encuadrar la imagen que se quiere segmentar un un sencillo recuadro dentro de toda la imagen. El algoritmo marca los pixeles fuera del cuadro como background seguro, y analiza los de dentro para marcarlos como background si son similares a los de fuera del cuadro, o foreground si son distintos. Más internamente utiliza valores intermedios como backgrounds probables y foregrounds probables, pero dicha explicación queda fuera del alcance de este PFC.

De esta forma, al usuario del aplicativo móvil se le abrirá únicamente la aplicación que maneja la cámara con un recuadro en el que encuadrar la seta a identíficar.

Para la aplicación de escritorio, y para mostrar el proceso de segmentación en esta entrega, se abrirá la imagen de la seta en una ventana y el usuario dibujará el rectángulo sobre la imagen deslizando el ratón.

Ilustración 13: Recuadro segmentacion

Ilustración 14: Whole Mushroom Segmentated

2.4.3. Segmentación de las diferentes partes de la seta

2.4.3.1. Pre-cálculos iniciales

Para segmentar las diferentes partes de la seta (sombrero, pie y escamas) se ha seguido el siguiente proceso.

1. Hallar los puntos cardinales. Primero buscamos los límites de la imagen segmentada por arriba, abajo, izquierda y derecha (a partir de ahora les llamaremos top, bottom, left y right). El proceso es trivial por lo que se obvia su explicación (en la imagen marcados en verde):
2.

Ilustración 15: Puntos cardinales

3. Para los diferentes cálculos necesitamos un contorno bien definida, como una figura siempre cerrada y única. El algoritmo GrabCut tiene el problema de que devuelve los pixeles que creen que forman parte la imagen principal dentro del recuadro, esto provoca que en ocasiones la imagen principal venga acompañada de pequeños píxeles o grupos de píxeles sueltos que implican diferentes geometrías segmentadas.

   Para solucionar este problema usaremos otro método de la librería OpenCV: *`findContours`*. Como su propio nombre indica busca contornos en una imagen. Nosotros le pasamos la imagen segmentada y nos devuelve una lista con todos los contornos encontrados, por lo que nosotros solo habremos de coger el más grande de ellos y de esta forma obtenemos el contorno de nuestra seta segmentada como único y cerrado. En la siguiente imagen se puede ver el contorno en amarillo:

Ilustración 16: Contorno imagen segmentada

4. El siguiente paso consiste en hallar los dos puntos donde el sombrero limita con el pie. Para ello usaremos el siguiente algoritmo:

    1. Para el punto izquierdo, trazamos una linea que una los puntos left y bottom.

    2. Recorremos todos los puntos del contorno entre esos dos puntos

    3. Calculamos la distancia entre cada punto del contorno y la recta.

    4. El punto con distancia mayor a la recta y que se encuentre a la derecha de la recta (es decir, con la recta *bottom - current* con pendiente mayor que la recta *bottom - left*) será el que usaremos de límite entre el sombrero y el pie de la seta.

    5. Para el punto derecho repetimos los mismos pasos simétricamente.

    En la siguiente imagen podemos ver en rojo las rectas y en verde, aparte de los puntos cardinales, los puntos límite entre sombrero y pie (a partir de ahora innerLeftCapLimit e innerRightCapLimmit) calculados con el mencionado algoritmo.

5. Con los datos que tenemos ya podemos segmentar el sombrero y el pie.

2.4.3.2. Segmentación del sombrero

Para el sobrero usaremos otra vez el algoritmo GrabCut partiendo del rectángulo original, pero esta vez le cambiaremos el límite inferior a dicho rectángulo.

Para calcular el nuevo límite inferior recorreremos lo puntos entre left e innerLeftCapLimit, y entre innerRightCapLimmit y right. El punto más bajo de todo ese recorrido será el que usemos como límite inferior del nuevo rectángulo que pasaremos a GrabCut para segmentar el sombrero:

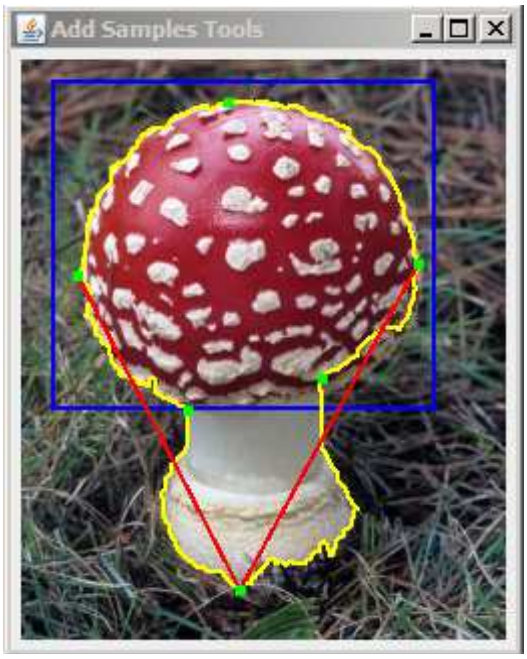


Ilustración 17: Cuadrado segmentación sombrero

Ilustración 18: Sombrero segmentado

2.4.3.3. Segmentación del pie

Para segmentar el pie utilizaremos el mismo concepto. Pero esta vez, para calcular el lateral superior del rectángulo para GrabCut necesitamos emplear otro método:

1. Primero obtenemos el contorno de solamente el sombrero

2. Buscaremos el punto más elevado de dicho contorno entre los puntos innerLeftCapLimmit y innerRightCapLimit.

   Aquí debemos tener cuidado ya que el contorno del sombrero no tiene por qué coincidir exactamente con el de la seta, esos puntos pueden no encontrarse entre el contorno del sombrero. Por lo tanto usaremos la vertical de ambos puntos cara calcular el punto de corte con el contorno del sombrero.

   Una vez tenemos el tramo del contorno del sombrero que limita con el pie podemos hallar su punto más elevado.

3. Buscaremos el resto de puntos cardinales del pie, cuyo cálculo es trivial.

4. Una vez tenemos los 4 puntos cardinales formamos el rectángulo correspondiente para pasarle al algoritmo GrabCut:

Ilustración 19: Cuadrado segmentación pie

Ilustración 20: Pie segmentado

2.4.3.4. Segmentación de las escamas

Para la segmentación de las escamas utilizaremos otro algoritmo de la librería del OpenCV: findContours. Grosso modo, este algoritmo lo que hace es buscar contornos en una imagen binaria contornos. Por lo tanto, la calidad de la segmentación de las escamas dependerá en gran medida de la capacidad para generar una imagen binaria de calidad, lo cual en estos momentos no se ha conseguido, pero se sigue trabajando en la combinación de filtros idónea para tal fin.

La cadena de filtros actualmente es la siguiente:

Grayscale > Contrast > Binary

Siendo el resultado el siguiente:

Ilustración 21: Segmentacion escamas: original

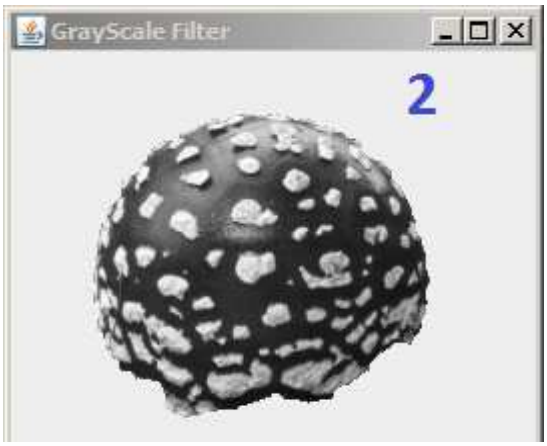


Ilustración 22: Segmentacion escamas: Grayscale Filter

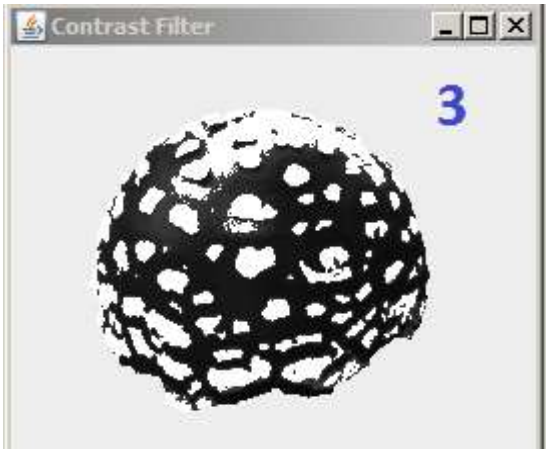


Ilustración 23: Segmentacion escamas: Contrast Filter

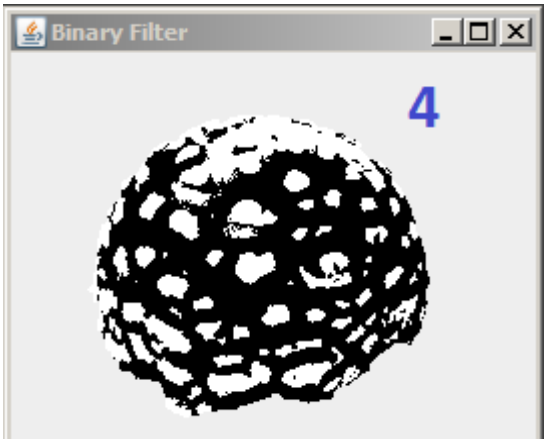


Ilustración 24: Segmentacion escamas: Binary Filter

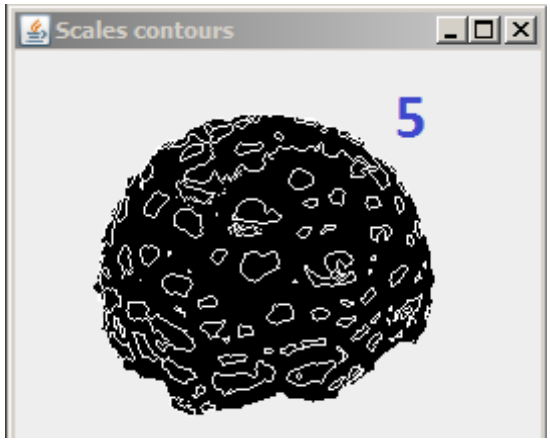


Ilustración 25: Segmentacion escamas: resultado

2.4.3.5. Código fuente de la segmentación

Capítulo 3. Parametrización, clasificación, Android app y servidor.

3.1. Sistema de parametrización de la seta

La parametrización de la seta consiste en coger todas esas partes de la seta que ya tenemos segmentadas y poder asignarle un valor cuantificable representativa que pueda ser procesado por un algoritmo, en esta ocasión se trata de pasar imágenes a parámetros numéricos.

Las partes que se iban a parametrizar ya se han comentado en puntos anteriores, aquí especificaremos como se lleva a cabo para cada uno de los tipos de parámetros.

3.1.1. Implementación de las parametrizaciones de las diferentes de partes de la seta

Altura: La altura de la seta, del sombero y del tallo se mediará en proporción a toda la seta, es decir, nunca mediremos distancias, sino porcentajes; no parametrizaremos el tallo como de 120px de alto, sino como un 40% del alto de toda la seta. De esta forma conseguiremos unas medidas independientes del tamaño de la foto original.

Anchura: Exactamente lo mismo.

Size: Lo mismo pero en superficie. Y para contar la superficie para calcular los porcentajes simplemente usamos la cantidad de píxeles que contiene.

AspectRatio: Guardamos también una cifra que nos identifica la relación entre la altura y la anchura de cada una de las partes.

Color: Para medir el color de una parte de la seta llevamos a cabo una media de los colores de todos los píxeles de su imagen. Para ello usamos la funcion **`Core.mean`** de OpenCV pasándole como máscara el canal alfa de la imagen.

3.1.2. Integración con el Sistema de Segmentación

Tanto a nivel de la aplicación de escritorio como de la aplicación Android, se ha creado clases de model diferentes para cada ámbito. Por lo que tenemos

```
edu.uoc.gmanjon.findmushs.parameterization.model.Mushroom
y
edu.uoc.gmanjon.findmushs.segmentation.model.Mushroom
```

Tanto a nivel de segmentación como de parametriación las partes son las mismas (seta, sombrero, tallo y escamas), pero la información que contiene cada uno de ellos en cada ámbito no tiene nada que ver, por lo que se ha separado.

Tanto el proceso de segmentación como el de parametrización están encapsulados dentro de MushroomSegmentator y Parameterizator respectivamente, por lo que gracias a dicho encapsulamiento la integración es trivial, ya que MushroomSegmentator de vuelve un Mushroom (del modelo de segmentación), que es justo el parámetro que acepta Parameterizator. A partir de este y en el método `parameterize()` se devuelve el Mushroom ahora ya sí del modelo de parametrización.

3.2. Sistema de Clasificación de la seta

3.2.1. Presentar los posibles algoritmos de clasificación

Al tratarse de un problema de "¿En que grupo de cosas meto esta nueva?" los dos tipos de algoritmos que barajaba era de Clusterización y Clasificación. No parece que tenga sentido contemplar ningún otro tipo de algoritmo más.

3.2.2. Elegir razonadamente uno o varios de ellos

Pero la duda se disipa rápidamente a poco que analicemos la situación.

Disponemos de una muestra de clases ya conocidas (todas las que metemos a través de la aplicación de escritorio), y queremos con ellas generar un modelo clasificatorio que nos infiera la clase de nuevas instancias (las enviadas desde la aplicación móvil).

Lo que acabo de describir es claramente un clasificador. Sabemos las clases de las muestras de antemano (todas las que metemos a través de la aplicación de escritorio), por lo que la clusterización aquí no tiene sentido sentido.

3.2.3. Implementar el algoritmo de clasificación

El proceso de clasificación se lleva a cabo en el Servidor, y para ello usaremos la API de WEKA. El propio programa ya distribuye un weka.jar para poder acceder a todo su potencial desde cualquier aplicación Java.

Entre las diferentes implementaciones de un Clasificador que tiene WEKA hemos elegido el IBk (vecino más cercano), que genera una árbol de decisión para posteriormente clasificar las setas de los usuarios.
El proceso por nuestra parte es bien sencillo, ya que ahora toda la carga del algoritmo cae sobre Weka.

Cargamos los datos de la base de datos, con cada seta generamos un instancia con todos los datos en cadena y la clase al final de todo. Se acumulan todas las instancia en un objeto **Instances** y estas se le pasan al método `buildClassifier()`. Y así ya tenemos nuestro clasificador.

Ahora cada vez que queramos clasificar una seta la convertimos en una instancia y llamamos al método **classifyInstance** de nuestro clasificador. Dicho método nos devolverá la clase.

3.3. Implementación de la App de Android

3.3.1. Desarrollo de las pantallas

En la aplicación Android tendremos 5 pantallas:

3.3.1.1. Pantalla principal

En esta pantalla:

- Indicamos la IP doned está instalado el servidor
- Y después del texto "Identify mushroo from..." encontramos dos botones grandes "...fromCamera" y "...fromGaleley" que hablan por si solos.

3.3.1.2. Pantalla de selección de la imagen

Simplemente se abre la ventana de Android de archivos recientes filtrando por archivos de tipo imagen.

En caso de que se haya seleccionado **from Camera** es se abrirá y la foto tomada será la que se envía la siguiente pantalla.

3.3.1.3. Pantalla de selección del rectángulo

En ambas pantallas anteriores, con el resultado (fotos tomada y de galería) se envía a esta pantalla, en donde el usuario, con un dedo, marca un recuadro empezando por la esquina superior izquierda en la que ha de encuadrar la seta.

3.3.1.4. Pantalla de progresso del proceso

Esta pantalla contiene 5 secciones. Las cuatro primeras son contenedores donde se van depositando las diferentes partes segmentadas a medida que se van obteniendo. La última seccion es una progressBar que indica el progreso de la parametrización.

3.3.1.5. Pantalla de resultados

En esta pantalla simplemente se muestra una tabla con los datos básicos de la seta que ha identificado.

3.3.2. Integración con el Sistema de Segmentación

Para el sistema de segmentación de Android se ha utilizado el mismo que para la aplicación de escriotrio. Se ha transportado el código tal cual ya que los objetos de OpenCV para java y para Android son prácticamente los mismo. Solamente ha habido que hacerle algunos retoques como adaptarlo a usar **Bitmap** en vez de **BufferedImage**.

3.4. Implementación del Servidor

3.4.1. Desarrollo de los servicios web vía REST

Gracias a las anotaciones introducidas en el lenguaje a partir de la versión 5, la implementación de unos servicios REST es mucho más fácil y menos susceptible a errores.

Tendremos desplegados dos métodos: addSample y classify. Ambos reciben un parámetro Mushroom del modelo de parametrización, que usarán para añadir muestra y para clasificar respectivamente. Por ejemplo, si el servidor está desplegado en local podremos acceder a ambos métodos a través de las URL:

```
http://localhost:8080/FindMushServer/ws/addSample
http://localhost:8080/FindMushServer/ws/classify
```

Ambos reciben el Mushroom en formato de JSON, pero es ya la propia infraestructura de Web Services quien se encaga de los procesos de marshal y unmarshal.

3.4.2. Integración con la BBDD

Para la BBDD se ha creado una clase Singleton, **DDBBManager**, que se ocupa de todas las **SELECT** y todos los **INSERT** necesarios. El mapeado entre objetos y tablas es también bastante trivial, por lo que no se entra en detalle.

3.4.3. Integración con el Sistema de Clasificación

El sistema de clasificación , al igual que el resto, se encuentra encapsulado en su propia clase, y solamente tiene una clase **ClassificationFunctions** como helper.

Capítulo 4. Resultados

Para alimentar el classificador se han usado las siguientes imágenes:

Amanitas Caesareas:

Ilustración 26: AC - muestra 1

Ilustración 27: AC - muestra 2

Ilustración 28: AC - muestra 3

Ilustración 31: AC - muestra 4

Ilustración 29: AC - muestra 5

Ilustración 30: AC - muestra 6

Amanitas Muscarias:

Ilustración 32: AM - muestra 1

Ilustración 34: AM - muestra 2

Ilustración 33: AM – muestra 3

Ilustración 35: AM - muestra 4

Ilustración 36: AM - muestra 5

Ilustración 37: AM - muestra 6

Con estas muestras es con las que se encuentra en la actualidad la base de datos.

Para probar el clasificador primero hemos pasado las mismas muestras con las que hemos alimentado el clasificador. Eso sí, esta vez segmentadas y parametrizadas desde la Android App. En principio se mantuvo la reduccion de imagenes a 300px de ancho, pero hubo algunas segmentaciones (marcadas con asterisco) que fallaron, estas se volvieron a probar reduciendo a 500px las imagenes, aunque el tiempo procesamiento fuese algo mayor.
Estos ha sido los resultados.

Amanitas Caeareas		
	Segmentacion	Clasificación
Muestra 1*	OK a 500px	OK
Muestra 2	OK	OK
Muestra 3*	OK	OK
Muestra 4	OK a 500px	OK
Muestra 5	OK	OK
Muestra 6	OK	OK

Amanitas Muscaria		
	Segmentacion	Clasificación
Muestra 1*	OK	OK
Muestra 2	OK	OK
Muestra 3*	OK*	ERROR
Muestra 4	OK	OK
Muestra 5	OK	OK
Muestra 6	ERROR	

Total, las clasificaciones correctas de muestras con las que se alimentó el clasificador han sido del 83%. Solo fallaron 2, una por error de segmentación, y la otra por error en la clasificación.

Pasando ahora a muestras totalmente nuevas, utilizaremos las siguientes:

Amanitas Caesareas:

Ilustración 40: AC - muestra 7

Ilustración 39: AC - muestra 8

Ilustración 38: AC - muestra 9

Ilustración 43: AC - muestra 10

Ilustración 41: AC - muestra 11

Ilustración 42: AC - muestra 12

Amanistas Muscarias:

Ilustración 46: AM - muestra 7

Ilustración 44: AM - muestra 8

Ilustración 45: AM - muestra 9

Ilustración 47: AM - muestra 11

Ilustración 48: AM - muestra 12

Ilustración 49: AM - muestra 10

Esta vez se decide dejar directamente la reducción en caso de ser necesaria a 500px. Y los resultados obtenidos han sido los siguientes:

Amanitas Caeareas		
	Segmentacion	Clasificación
Muestra 7	A la segunda ajustando	OK
Muestra 8	Mejorable	OK
Muestra 9	OK si no cogemos volva	ERROR
Muestra 10	OK	OK
Muestra 11	OK	OK
Muestra 12	OK	OK

Amanitas Muscaria		
	Segmentacion	Clasificación
Muestra 7	OK	ERROR
Muestra 8	OK	OK
Muestra 9	OK	OK
Muestra 10	OK	ERROR
Muestra 11	OK	OK
Muestra 12	OK	OK

De nuevo obtenemos el 83% de aciertos.

Capítulo 5. Conclusiones

Toda la infraestructura montada es muy potente. Se consigue alcanzar un amplio sector de usuarios gracias a la aplicación Android. Se elimina del cliente la base de datos que presumiblemente crecerá de forma considerada, para añadir más setas solo hay que darlas de alta en el sistema y cualquiera con la aplicación de escritorio instalada puede añadir muestras sin necesidad de conocimientos informáticos.

A la vista de los resultados obtenidos se pueden sacar dos conclusiones:

1. El proceso de segmentación requiere de un continuo refinamiento, segmenta bien siempre que partamos de formas bastante uniformes y a poco similar que sea el fondo la calidad de la segmentación decae.

   Habrá que plantearse si merece la pena incluir más interacción del usuario para mejorar la calidad de la segmentación.

2. En la segunda tanda de clasificaciones, las Amanitas Muscarias que fallaron fueron la que tenía el sombrero algo mas naranja, y la que tenía la forma menos habitual entre las introducidas para alimentar al clasificador.

   El color parece una parte imporatnte, ya se había detectado antes y por ello se desactivaron los parámetros de tamaño (devido también a las más que evidentes similitudes de morfología entre ambas setas). Es probable que aumentando el muestreo que alimenta al clasificador con muchos más tipos diferentes de ambas setas el porcentaje de acierto aumente.

Finalmente quisiera comentar que este sistema eminentemente evolutivo, no se pueden alcanzar muy buenos resultados en sus primeras versiones, al contrario, necesita ir refinándose y evolucionar poco a poco, con más datos, con pequeñas mejoras en la segmentación y la clasificación que permita ir añadir nuevas formas de seta y mejorar la clasificación de las formas que ya se clasifican. Aún así, los resultados obtenidos son prometedores.

Bibliografía

[1] Android API Javadoc - http://developer.android.com/reference/packages.html

[2] OpenCV API Reference - http://docs.opencv.org/modules/refman.html

[3] Java API Javadoc - http://docs.oracle.com/javase/6/docs/api/

[4] JEE API Javadoc - http://docs.oracle.com/javaee/6/api/

[5] Jboss AS 7.1 Getting Started Guide - https://docs.jboss.org/author/display/AS71/Getting+Started+Guide

[6] Google Images - https://images.google.com/

Anexo A: Código fuente

Por lo tanto tenemos 3 proyectos principales, cada uno con varios subsistemas que están agrupados en diferentes paquetes. De los cuales los principales son los siguientes:

Desktop App

edu.uoc.gmanjon.findmush.addsamplestool

Contiene los elementos que conforman la aplicación principal. La clase Main es empieza la aplicación, y la clase AddSample se ejecuta cada vez que se queire añadir una nueva muestra.

edu.uoc.gmanjon.findmush.segmentation

Segmentacion de la seta para añadir Samples

edu.uoc.gmanjon.findmush.segmentation.model

Modelo de datos usado en la segmentación de la seta, contiene basicamente las diferentes partes de la seta con sus objetos de imágenes de OpenCV, puntos importantes (extremos norte, sur, este, y oeste), etc..

edu.uoc.gmanjon.findmush.parameterzition

Parameterización de la seta para añadir Samples

edu.uoc.gmanjon.findmush.segmentation.model

Modelo de datos usado en la parametrización de la seta, los nombres de las clases son los mismos que en el modelo de la segmentación, pero en este caso tendremos como atricutos de cada clase todo aquello que vamos a medir y a usar en el clasificador, tanto en las muestras para construirlo como para clasificar.

edu.uoc.gmanjon.findmush.rest

Comunicación con el servidor via servicios REST para añadir Samples. La comunicación se hacevia objetos JSON.

edu.uoc.gmanjon.findmush.filters

Contiene las implementaciones de los filtros usados para preprocesar las iamgenes.

Android App

edu.uoc.gmanjon.findmush.android.segmentation

Actua exactamente igual en Desktop App, pero aquí todo está adaptado para trabajar con Bitmap (forma en que trata Android la imágenes) en vez de BufferedImage (estandar de J2SE).

edu.uoc.gmanjon.findmush.android.parameterzition

Copia exacta de lo que tenemos en Desktop App, con la diferencia que para Android está adaptado para ejecutarse como **`AsyncTask`**.

edu.uoc.gmanjon.findmush.android.rest

Comunicaciópn con el servidor via servicios REST para clasificar la seta. La comunicación se hacevia objetos JSON.

edu.uoc.gmanjon.findmush.android.activities

Pantallas principales de la applicación

edu.uoc.gmanjon.findmush.android.views

Pantallas secundarias de la applicación

edu.uoc.gmanjon.findmush.filters

Copia de los filtros de la Desktop App con ligeras adaptaciones para trabajar en Android.

edu.uoc.gmanjon.findmush.listeners

Contiene los listeners usados en la aplicación para que esta actue en la medida de lo posible bajo el paradigma de la programación dirigida por eventos que tan incrustada está en los sitemas Android.

Server

edu.uoc.gmanjon.findmush.server.classification

Clasificación de la seta mediante la API de Weka. Se programa mediante un componente EJB Singleton

edu.uoc.gmanjon.findmush.server.ddbb

Singleton de comunicación con la base datos. Se programa mediante un componente EJB Singleton

edu.uoc.gmanjon.findmush.server.ws

Servicios REST sobre los que hacen peticiones los otros componentes. Desktop App para añadir muestras, y Android App para clasificar muestras. Usan los otros dos componentes del Server, Classificator y BBDDManager mediante inyección de EJB.

edu.uoc.gmanjon.findmush.server.model

Copia exacta del modelo de parametrizaciones que tenemos tanto en Desktop App como Android App. Hay que tener en cuenta que la comunicación se hace transformacion el objeto Mushroom del model de parametrización a JSON, se envía mediante servicio REST (tanto Desktop como Android) y el servidor parse el JSON String a un POJO que ha de tener las mismas propiedades.

Anexo B: Script de Base de Datos

```
DROP TABLE MUSHROOM_TYPE;
DROP TABLE MUSHROOM_SAMPLE;
DROP TABLE CAP;
DROP TABLE STEM;
DROP TABLE SCALES;
DROP VIEW MUSHROOM_FULL_SAMPLE;

CREATE TABLE MUSHROOM_TYPE (
   ID INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
   GENUS          TEXT NOT NULL,
   SPECIE         TEXT NOT NULL,
   CAP_INFO       TEXT NOT NULL,
   STEM_INFO      TEXT NOT NULL,
   EDIBLE_INFO    TEXT NOT NULL,
   LINK           TEXT NOT NULL
);

INSERT INTO MUSHROOM_TYPE (GENUS, SPECIE, CAP_INFO, STEM_INFO, EDIBLE_INFO,
LINK) VALUES
('Amanita', 'Muscaria',
 'De 6 a 20 cm. de diámetro. Ovoide, hemisférico y luego aplanado, húmedo un
poco viscoso, con el borde ligeramente estriado. cutícula separable de color
rojo anaranjado o escarlata, recubierta de numerosas verrugas piramidales
blancas o algo amarillentas, que son restos de la frágil volva.',
 'Cilíndrico, separable, lleno y luego hueco, engrosado en la base en un
bulbo ovoide adornado de círculos concéntricos y verrugas, restos del velo
general o volva, de color blanco harinoso.',
 'MUY TÓXICA, alucinógena. No confundir con Amanita caesarea.',
 'http://www.setasysitios.com/setas-y-sitios/setas/setas-toxicas/amanita-
muscaria-foto-y-ficha-tecnica'),
('Amanita','Caesarea',
'De 8 a 20 cm. de diámetro. Primero ovoide ( sale de un huevo) luego convexo
y al final plano con el borde estriado. Cutícula fácil de separar, lisa
untuosa, de un llamativo anaranjado vivo, a veces con fragmentos membranosos
de la volva blanca. ',
'Cilíndrico, grueso, bulboso, hueco y relleno de una sustancia lanosa, de
color amarillo. Separable, con anillo amplio, frágil, membranoso, estriado
del mismo color que el pie. En su base una amplia volva blanca, gruesa,
membranosa y elástica.',
'Comestible excelente y muy buscado desde la época de los Césares romanos',
'http://www.setasysitios.com/setas-y-sitios/setas/setas-comestibles/amanita-
caesarea---foto-y-ficha-tecnica');

CREATE TABLE MUSHROOM_SAMPLE (
   ID INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
   TYPE        INT NOT NULL,
   ASPECT_RATIO    DECIMAL(10,2)      NOT NULL,
   DATE            TIMESTAMP DEFAULT (STRFTIME('%S', 'NOW')),
   FOREIGN KEY (TYPE) REFERENCES MUSHROOM_TYPE(ID)
);

CREATE TABLE CAP (
   ID INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
   SAMPLE_ID INT             NOT NULL,
   SIZE      DECIMAL(10,2)   NOT NULL,
   COLOR     CHAR(6)         NOT NULL,
```

```
   WIDTH     DECIMAL(10,2)            NOT NULL,
   HEIGHT    DECIMAL(10,2)            NOT NULL,
   ASPECT_RATIO    DECIMAL(10,2)      NOT NULL,
   FOREIGN KEY (SAMPLE_ID) REFERENCES MUSHROOM_SAMPLE(ID)
);

CREATE TABLE STEM (
   ID INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
   SAMPLE_ID INT                 NOT NULL,
   SIZE      DECIMAL(10,2)       NOT NULL,
   COLOR     CHAR(6)             NOT NULL,
   WIDTH     DECIMAL(10,2)       NOT NULL,
   HEIGHT    DECIMAL(10,2)       NOT NULL,
   ASPECT_RATIO    DECIMAL(10,2) NOT NULL,
   FOREIGN KEY (SAMPLE_ID) REFERENCES MUSHROOM_SAMPLE(ID)
);

CREATE TABLE SCALES(
   ID INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
   SAMPLE_ID INT                NOT NULL,
   NUM_OF_SCALES INT            NOT NULL,
   SIZE      DECIMAL(10,2)      NOT NULL,
   BIGGEST      DECIMAL(10,2)   NOT NULL,
   SMALLEST      DECIMAL(10,2)  NOT NULL,
   AVERAGE      DECIMAL(10,2)   NOT NULL,
   STANDARD_DEV   DECIMAL(10,2) NOT NULL,
   COLOR     CHAR(6)         NOT NULL,
   FOREIGN KEY (SAMPLE_ID) REFERENCES MUSHROOM_SAMPLE(ID)
);

CREATE VIEW MUSHROOM_FULL_SAMPLE AS
    SELECT
            mush.TYPE AS MUSH_TYPE,
            mush.ASPECT_RATIO AS MUSH_ASPECT_RATIO,
            cap.SIZE AS CAP_SIZE,
            cap.COLOR AS CAP_COLOR,
            cap.WIDTH AS CAP_WIDTH,
            cap.HEIGHT AS CAP_HEIGHT,
            cap.ASPECT_RATIO AS CAP_ASPECT_RATIO,

            stem.SIZE AS STEM_SIZE,
            stem.COLOR AS STEM_COLOR,
            stem.WIDTH AS STEM_WIDTH,
            stem.HEIGHT AS STEM_HEIGHT,
            stem.ASPECT_RATIO AS STEM_ASPECT_RATIO,

            scales.NUM_OF_SCALES AS NUM_OF_SCALES,
            scales.SIZE AS SCALES_SIZE,
            scales.COLOR AS SCALES_COLOR,
            scales.BIGGEST AS SCALES_BIGGEST,
            scales.SMALLEST AS SCALES_SMALLEST,
            scales.AVERAGE AS SCALES_AVERAGE,
            scales.STANDARD_DEV AS SCALES_STD_DEV
    FROM MUSHROOM_SAMPLE mush, CAP cap, STEM stem, SCALES scales
    WHERE cap.SAMPLE_ID = mush.ID AND
          stem.SAMPLE_ID = mush.ID AND
          scales.SAMPLE_ID = mush.ID;
```

Anexo C: Manual de Instalación

IMPORTANTE: Los dispositivos donde estén instalados los diferentes modulos (Dekstop, Android y Server) han de estar en la misma red para que puedan tener comunicación directa. Esto no sucede si el movil está conectado mediante la red de datos del operador (3G, 4G, etc...), por lo que habrá que conectar el telefono a la WiFi de la red donde esté instalado el servidor.

Tanto para el Server como para Desktop App se definien las carpetas de trabajo de FindMush. En ella se escribirán los logs en la carpeta **%FINDMUSH_HOME %\logs**. Ambas apliaciones escriben en all.log, y además cada una tiene su **server.log** y **desktop.log**. Los archivos de log se van rotando cada día.

En este anexo usaremos de ejemplo la carpeta de trabajo **C:\findmush**

Server

El servidor ha de instalarse en un servidor de aplicaciones Jboss 7.1.1 que se puede descargar de la siguiente URL:

http://jbossas.jboss.org/downloads/

no requiere de ninguna configuración ya que todas las librerías necesarias se encuentran dentro de la applicación y el Jboss funciona correctamente con su configuración de serie.

Para instalar el Server en el JBoss solo tenemos que copiar el archivo **`FindMushServer.war`** que se encuentra en la carpeta raíz del proyecto al subdirectiorio del JBoss **`...\jboss-as-7.1.1\standalone\deployments\FindMushServer.war`**.

Copiar la base de datos que encontraremos en la carpeta raíz del proyecto en la carpeta de trabajo:

```
C:\findmush\findmush.db
```

Una vez hecho esto arrancamos el JBoss: vamos a la carpeta **`...\jboss-as-7.1.1\bin`** y ejecutamos el siguiente comando:

```
standalone.bat -b 0.0.0.0 -DFINDMUSH_HOME=C:\findmush
```

(**`-b 0.0.0.0`** es para que acepte conexiones más allá de **`localhost`**, necesario para la Android App). Podemos comprobar si ha quedado bien instalado si al abrir la siguiente direccion en el navegador:

http://localhost:8080/FindMushServer/ws/findmush

el navegador escribe por pantalla únicamente **`{"FINDMUSH_HOME": "C:\findmush"}`**, es decir, un objeto con la carpeta de trabajo.

Desktop App

El único requisito es tener la versión de java 6 en la variable de entorno PATH. Para arrancar la aplicación de escritorio únicamente hay que ejecutar el archivo **`runDesktopApp.bat`** que se encuentra en la carpeta raíz del proyecto.

Android App

Para instalar la Android App has de tener habilitada la posibilidad de instalar aplicaciones de fuentes desconocidas, habitualmente en la ruta:

Configuración > Seguridad > Fuentes desconocidas

Copia el archivo **`androidApp-release.apk`** que encontrarás en la carpeta raíz del proyecto en tu teléfono Android. Después llega hasta donde lo hayas guardado mediante el explroador de archivos de Android y haz click sobre el mismo. La aplicación debería instalarse sin problemas y poder empezar a usarla.

Anexo D: Manual de Usuario

Desktop App

La ventana principal que se abre es la siguiente:

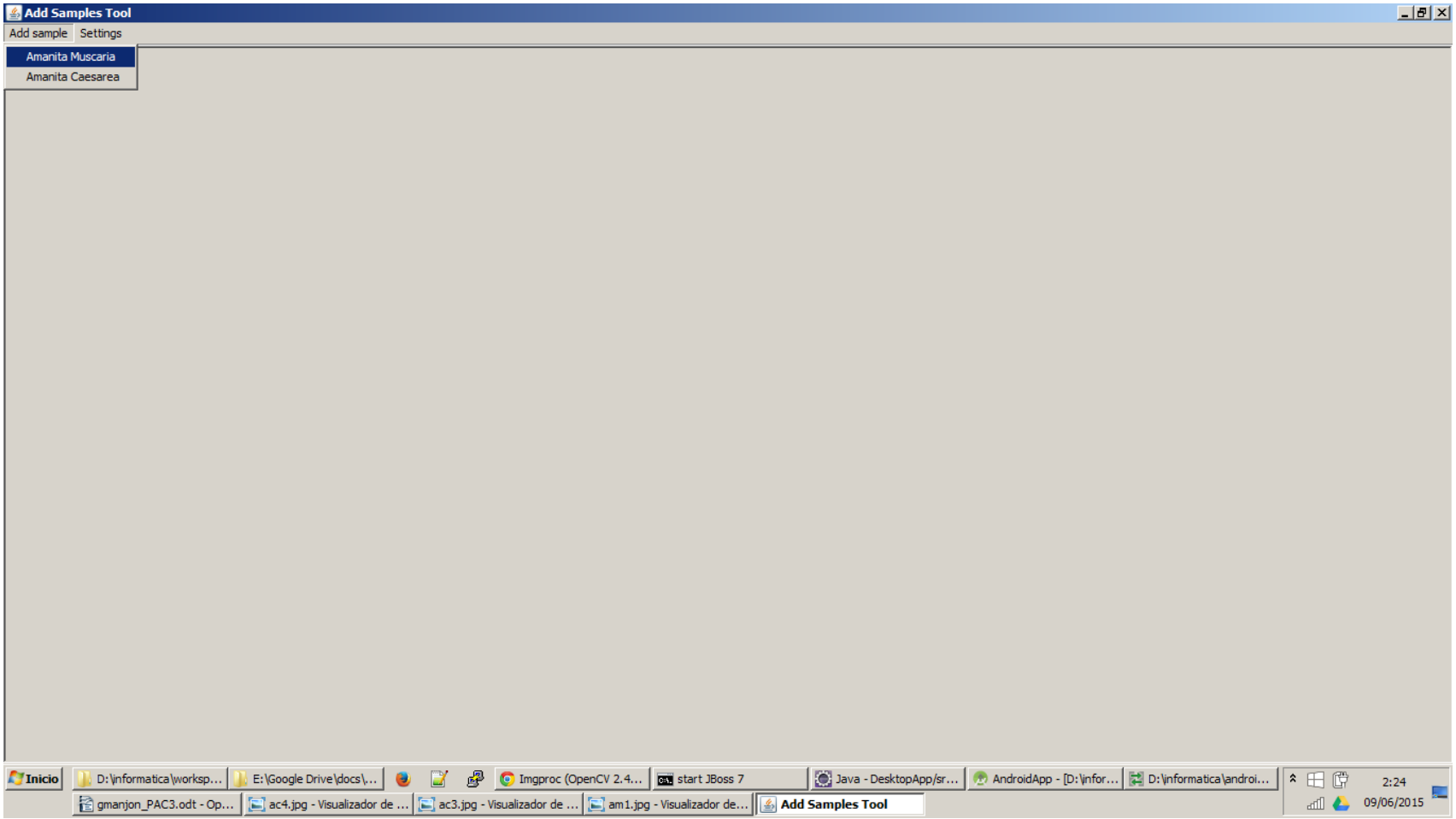


Ilustración 50: Desktop App - Main

Contiene dos menus:

- Add Sample
  - Amanita Muscaria
  - Amanita Caesarea
- Settings
  - Set server IP

En Settings deberemos introducir la dirección IP donde está isntalado el servidor, el valor por defecto es localhost. Cuando se selecciona se abre una ventana como la siguiente para indicar la IP:

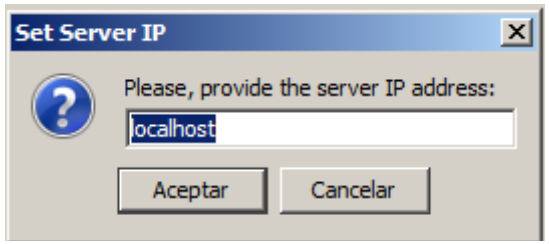


Ilustración 51: Desktop App - Set server IP

En el menu Add Sample añadimos nuevas muestras de la seta seleccionada. Al seleccionar el tipo de seta se no abre una ventana para seleccionar el archivo imagen que queremos utilizar para la muestra, se carga en la pantalla y el programa nos pide que dibujemos un recuadro en el que enmarcar la imagen, empezando por la esquina superior izquierda.

Una vez dibujado el recuadro nos pedirá confirmación por si no nos ha quedado a nuestro gusto. Se mantiene el anterior recuadro como referencia, pero se borrará nada más comenzar el nuevo recuadro.

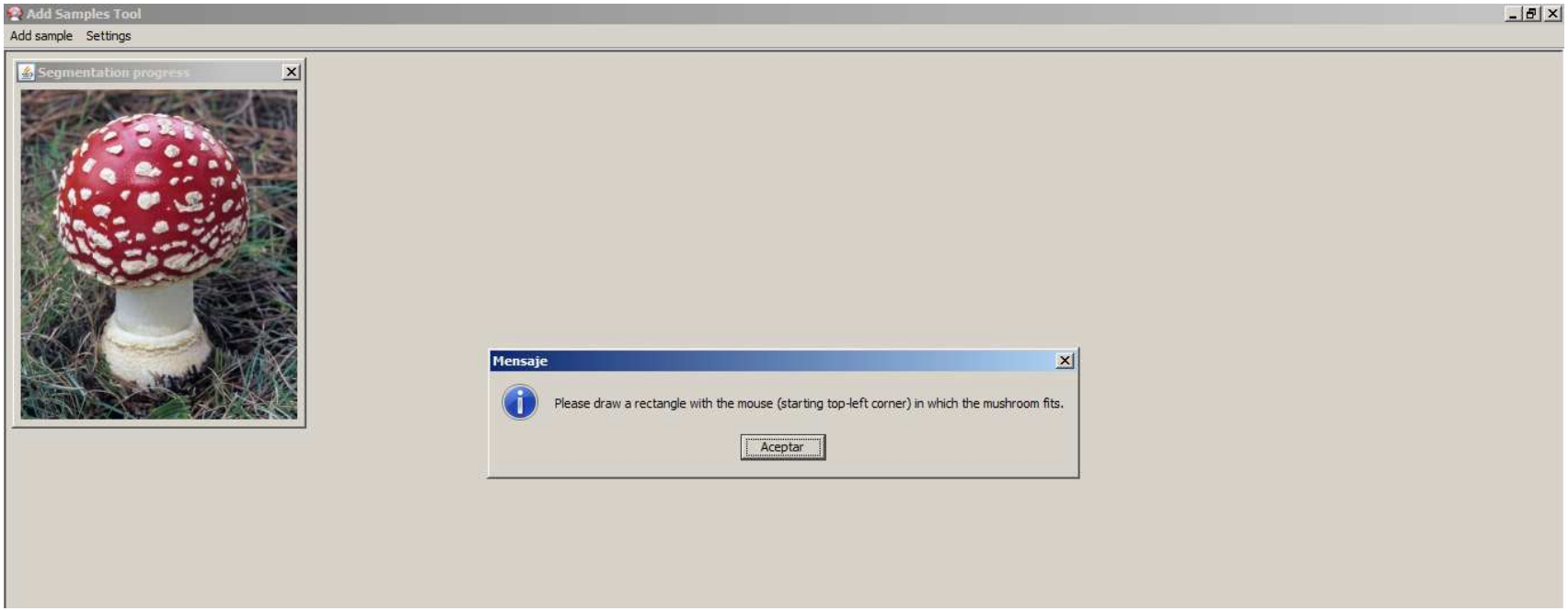


Ilustración 52: Desktop App - Confirmar recuadro

Una vez confirmado el cuadro la aplicación procederá a la segmentación de la seta mostrando por pantalla la evolución de la misma y mostrando los diferentes pasos calculados a medida que vaya avanzado, generando finalmente una ristra de ventanas como las de la siguiente imagen:

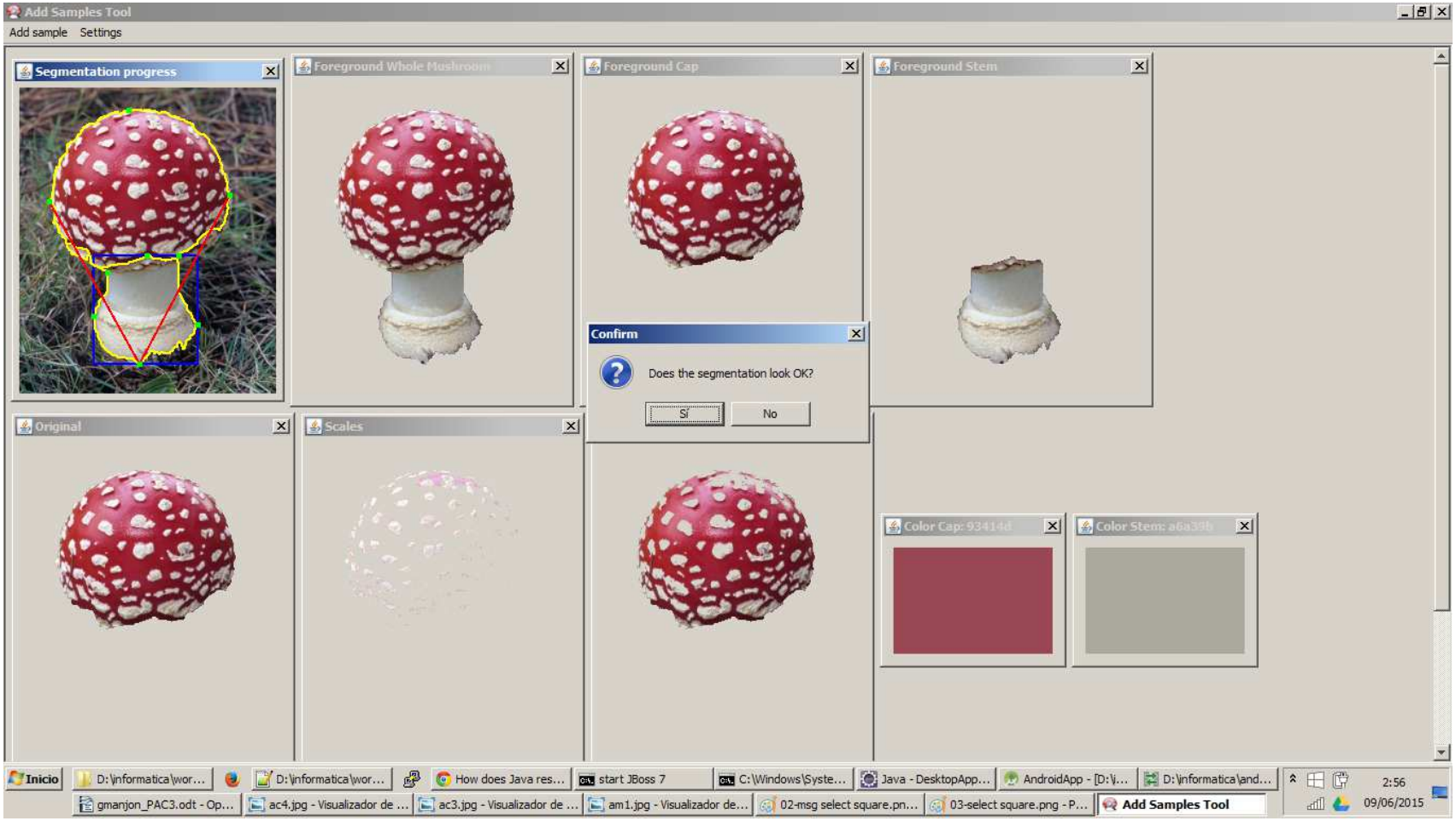


Ilustración 53: Desktop App - Resultado segmentación

Y preguntándonos si la segmentación se ve correctamente antes de enviarla al servidor para insertarla en BBDD y ser usada en el clasificador.

En caso de comunicacion correcta con el servidor nos muestra el mensaje de OK, en caso contraro muestra una ventana de error con la descripción del error.

Android App

La pantalla inicial de la aplicación Android es la siguiente:

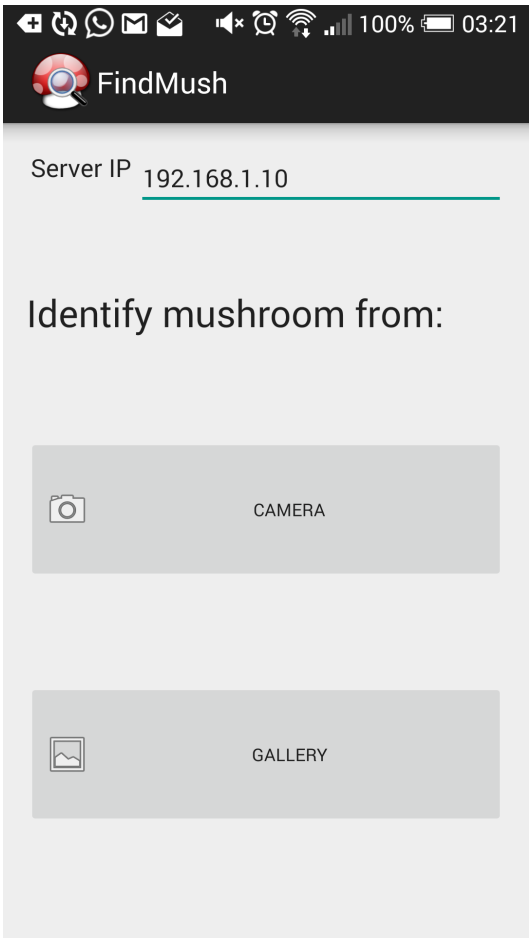


Ilustración 54: Android App - Main

En ella tenemos 3 elementos:

- Server IP: Aquí seleccionaremos la IP donde está instalado el servidor
- Identify mushroom from... y 2 botones:
  - CAMERA: Para abrir la cámara y sacar una foto a la seta a seleccionar
  - GALLERY: Si quieres identificar una seta a partir de una foto ya existente.

En cualquiera de los dos casos nos abrirá la imagen seleccionada y nos pedirá, al igual en la Desktop App, que seleccionemos un recuadro comenzando por la esquina superior izquierda.

Una vez seleccionado el recuadro nos pedirá conirmación por si el recuadro no ha quedado como esperábamos. Se mantiene el anterior recuadro como referencia, pero se borrará nada más comenzar el nuevo recuadro.

Ilustración 55: Android App - Seleccionar recuadro

Ilustración 56: Android App - Confirmar selección

Una vez confirmado el recuadro comienza el proceso de segmentación. La siguiente pantalla que se abre irá indicando lso progresos mostrando diferentes imagenes. La pantalla base antes de cualquiera actualización de la pantalla por parte del segmentador es la siguiente:

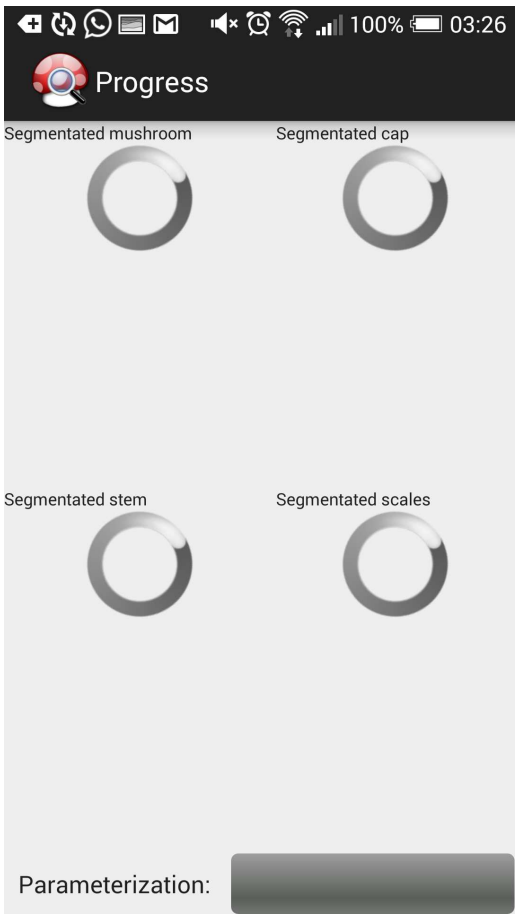


Ilustración 57: Android App - Progreso segmentación

En el primer recuadro aparecerá una miniatura de la imagen y se verá el proceso de como se van seleccionado los limites entre el sombrero y el pie mediante el algoritmo de la sección 2.4.3.1.4. Primero el izquierdo y después el derecho:

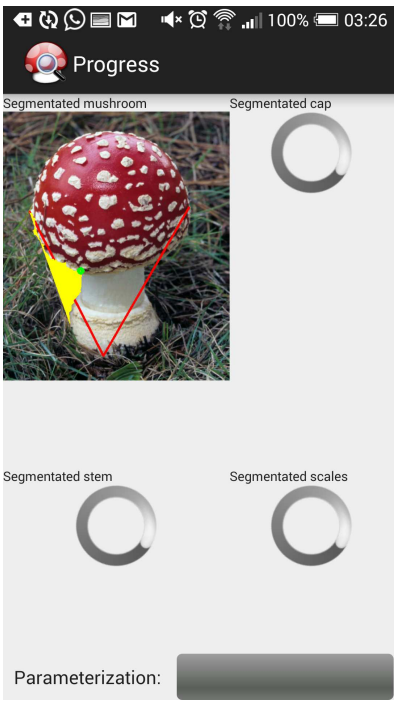


Ilustración 58: Android App - Límite sombrero izquierda

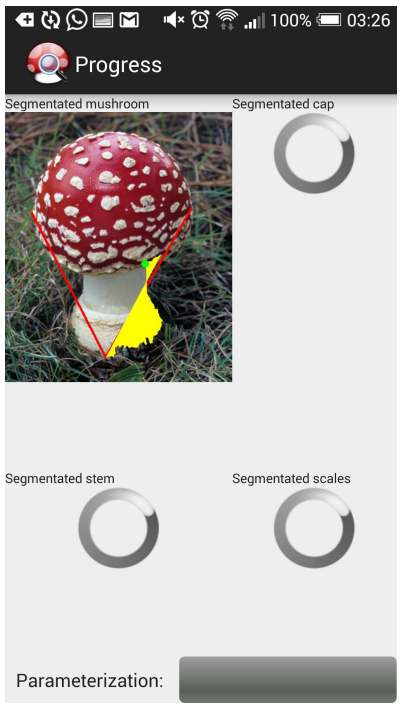


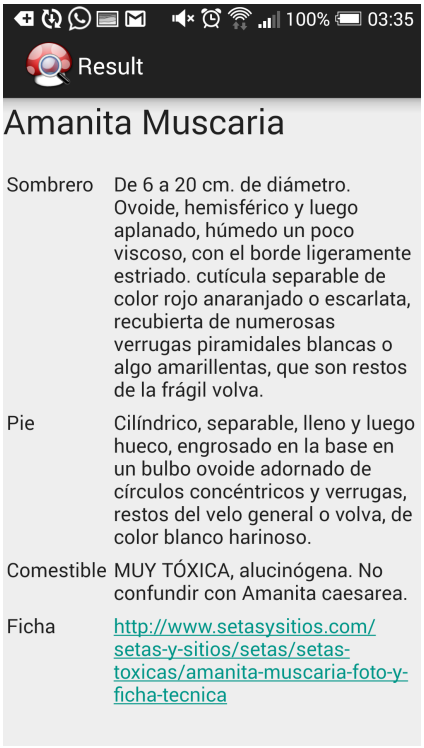
Ilustración 59: Android App - Límite sombrero derecho

En los siguientes pasos se irán mostrados las imagenes segmentadas descritas en el título de cada sección:

Ilustración 60: Android App - Parametrizando

Durante la parametrización la barra inferior irá mostrando el progreso.

Finalmente se envía la petición al servidor, con cuya respuesta se monta la página de Resultado:



En esta vemos una breve descripción de las características micológicas de la seta más un enlace a su ficha técnica en el site http://www.setasysitios.com