**Máster de Visión Artificial**

**Asignatura: Aplicaciones Industriales**

**Práctica puntuable – Preparación de la imagen**

INTRODUCCIÓN

Desde 1998, en el sector vacuno se ha desarrollado un sistema de trazabilidad que permite un mayor control a lo largo de toda la cadena alimentaria en aras de una mayor transparencia, lo que ha permitido restablecer la confianza del consumidor tras la crisis de las encefalopatías espongiformes transmisibles y garantizar la seguridad alimentaria de los productos de carne de vacuno.

En la página web del Ministerios de Agricultura, Alimentación y Medio Ambiente [1] se describe el sistema de trazabilidad del ganado bovino, cómo debe actuar el ganadero en diferentes supuestos y se ofrece información sobre la normativa española y comunitaria sobre identificación y registro del ganado bovino.

Con el objetivo de facilitar la tarea de los ganaderos y trabajadores encargados de la manipulación del ganado está practica tiene como objetivo el de desarrollar un sistema en el cual el empleado se centraría principalmente en extraer los crotales de los animales y depositarlos en una cinta. A partir de este momento una cámara tomaría una instantánea de este crotal y mediante una aplicación se devolvería el número que contienen.

Dicho aplicación se va a dividir en dos prácticas. La primera, y que es objeto de este documento, se centraría en la preparación de la imagen, y la segunda en el reconocimiento del texto de la imagen resultante anterior.

La preparación de la imagen, como se explicará con más detalle a continuación, va a consistir en técnicas básicas como por ejemplo extracción de ruido, eliminación de toda la información no relevante, corrección imágenes torcidas, etc.

Como resultado de esta práctica, partiendo de las imágenes de crotales contenidas en una carpeta, se devolverá una serie de imágenes bitonales que sólo contendrán los dígitos presentes en las imágenes.

DESCRIPCIÓN DEL MÉTODO DESARROLLADO

Para la implementación de esta aplicación se hace uso de las funciones que aporta la librería ImageJ [2].

ImageJ es un programa de dominio público de procesamiento de imágenes inspirado en NIHImage [3] para los Macintosh. Se ejecuta en cualquier ordenador que tenga instalada máquina virtual de Java 1.4 o posterior y permite aplicar todo tipo de operaciones tales como mostrar, editar, analizar, procesar, guardar e imprimir sobre imágenes de 8-bits, 16-bits y 32-bits.

El diseño de esta práctica tiene en cuenta dos fases. En la primera se calcula el ángulo para enderezar la imagen y en segunda, una vez enderezada dicha imagen, se sacan los números que representan el crotal.

Para la realización de la primera fase se llevan a cabo una serie de pasos. En primer lugar se realiza un proceso de pre-preparación de la imagen. Esto consiste en realizar unas operaciones de la imagen para facilitar su manipulación.

Dichas operaciones se pueden resumir en:

1. Umbralización de la imagen. Como realizar un umbralizado no es vital en esta primera fase, se utiliza el auto-umbralizado que proporciona ImageJ. Dicho umbralizado es un procedimiento iterativo basado en el algoritmo de isodata [4] y está brevemente descrito en la documentación de ImageJ [5].
2. Eliminación de ruido. Para la eliminación del posible ruido se utiliza un proceso simple de erosión y dilatación.

Una vez realizadas estas operaciones se tiene la imagen preparada para poder calcular el ángulo para enderezar la imagen. Este cálculo se lleva a cabo detectando el margen inferior del crotal y midiendo su deviación con respecto al margen inferior de la imagen.

Para ello, en primer lugar, se detecta cual es el punto inferior del crotal de la imagen. Dicha detección consiste en un barrido horizontal de la imagen para obtener el punto en el cual el valor vertical del crotal es más próximo al margen inferior (dicho punto se denominará de aquí en adelante yi).

Una vez detectado este punto se calcula su “homólogo” en la otra mitad de la imagen. Hay diferentes maneras de interpretar cual es el “homologo” de un punto con respecto a otro. En está implementación para el cálculo de este punto se hace la suposición de que un crotal siempre será de largo horizontalmente mayor a la mitad de la anchura de la imagen. Con esta suposición se define el “homologo” de un punto como aquel punto que está a una distancia horizontal de la mitad de la anchura de la imagen. Siendo xi el punto inicial y w la anchura de la imagen, su “homólogo” xh se calcularía como:

Si xi < (w/2) : xh = xi + (w/2);

Si xi > (w/2) : xh = xi - (w/2);

Seguidamente partiendo del inferior de la imagen, con un proceso de rastreamiento vertical, se busca el punto vertical (llamese yh) donde se encuentra el margen inferior del crotal con respecto a este nuevo punto horizontal xh.

A continuación, gracias al uso de una librería proporcionada por apache [6] que permite el fácil uso de operaciones geométricas, se calcula la recta que pasa por los puntos (xi, yi), (xh, yh). A partir de esta recta se obtiene su pendiente con respecto a eje de las x gracias a la misma librería usada anteriormente. En este punto es importante tener en cuenta que en las imágenes el punto (0, 0) está en la parte superior izquierda, por lo que la recta calculada estará invertida y el ángulo obtenido nos permitiría enderezar la imagen en el sentido contrario a las agujas del reloj. Esto es importante porque la librería ImageJ tiene la funcionalidad de girar imágenes en el sentido de las agujas de reloj por lo que se tendría que multiplicar por -1 el ángulo obtenido para girar correctamente la imagen.

Conseguido este ángulo de giro se puede decir que la primera fase de la práctica estaría finalizada y entraría en juego la segunda fase. Esta fase se caracteriza por dos partes, una consiste en sacar los números asociados al crotal y la otra en aplicar el ángulo de giro para enderezar la imagen. La parte de sacar los dígitos del crotal se lleva a cabo a través de un proceso iterativo que busca un número de elementos determinado con una densidad de pixeles específica. Es por ello que en el algoritmo presentado es necesario definir el número mínimo de dígitos a buscar en la imagen. Dicho número es parametrizable y se puede pasar como parámetro a la hora de llamar al ejecutable (por defecto vale 4). En este algoritmo también hay otros valores que ser van modificando automáticamente hasta encontrar el número de dígitos que se anda buscando. Dichos valores son, por un lado, la densidad de los elementos que va desde los 2500 pixeles por elemento hasta los 500 pixeles por elemento (se considera que cualquier valor que no esté entre estos valores y el valor máximo, definido como 4000 pixeles por elemento, no representa ningún número). Por otro lado otro de los valores que se va modificando en dicho algoritmo iterativo es el valor del umbralizado de la imagen que va desde 40 a 80 (se considera que cualquier valor menor o mayor no permite distinguir dígitos correctamente).

Sabiendo esto se procede a explicar el funcionamiento básico del algoritmo representado en esta implementación de la práctica. En primer lugar y como en la fase anterior se realiza un pre-preparado de la imagen. Donde se realizan unas operaciones básicas sobre la imagen:

1. Umbralización de la imagen. Estableciendo el valor de umbral al valor que lleve la variable de umbralizado en cada iteración.
2. Binarización de la imagen. Se convierte la imagen a blanco y negro y se normaliza a valores de entre 0 y 1. Se simplifica de esta manera la imagen porque no es relevante saber ni el color ni el valor de gris para esta práctica.
3. Inversión de los colores de la imagen. Ya que el umbralizado junto con la binarización provocan que los dígitos salgan en blanco, hay que invertir los colores para que los dígitos salgan en negro y puedan en un futuro ser tratados por un reconocedor de texto OCR.
4. Eliminación de ruido. Para la eliminación del posible ruido se utiliza un proceso simple de erosión que se corregirá con una dilatación cuando hayamos conseguido los mejores valores de umbralizado y densidad de pixel de los dígitos.

A continuación y una vez preparada la imagen se realiza la detección de dígitos. Para ello se hace uso de la técnica de componentes conexas que aporta la funcionalidad de ImageJ. En esta librería está técnica se lleva a cabo con lo que denominan un analizador de partículas. Con dicho analizador se buscan partículas con densidades entre el valor actual de la densidad de pixel de la iteración en curso y el valor máximo de densidad (establecido en los 4000 pixeles por elemento).

De esta manera el algoritmo descrito anteriormente tendrá dos bucles iterativos y funcionará de la siguiente manera:

1. Si se detecta al menos el número de dígitos que andamos buscando (definido por parámetro de entrada) entonces el algoritmo para y devuelve la imagen sólo con los dígitos que ha encontrado.
2. En caso contrario (el número de dígitos encontrados es menor que el buscado por parámetro) se procede a la modificación de los valores de umbralizado y densidad de pixel para iterar de nuevo e intentar encontrar dígitos.
3. Si se llega al mínimo de densidad de pixel por dígito (500 pixeles por elemento) se devuelve una imagen con los dígitos detectados para el valor de umbralizado máximo (80) y el valor de densidad de pixel por elemento mínimo (500). Aunque no cumpliría las expectativas esperadas y la imagen no sería muy significativa. Este caso representaría el peor caso y el objetivo es no llegar a él.

Como se ha comentado el algoritmo es iterativo con un doble bucle porque, como puede interpretarse del caso 2, en el bucle más interno se modificaría el valor del umbral (bucle de umbralizado) y en bucle exterior el valor de la densidad de pixel (bucle de densidad de pixel). Con el objetivo de mejorar el algoritmo las modificaciones de estos valores se han hecho más eficientes comparado con la modificación lineal de ellos. Esto permite, si no mejorar el orden del algoritmo, disminuir el tiempo de ejecución. Para ello lo que se ha tenido en cuenta es el número de conexiones conexas reconocidas en cada iteración. De esta manera las variables de actualizarían en cada iteración como se define a continuación:

Sea ti el valor del umbralizado de la iteración i en el *bucle de umbralizado*, n el número mínimos de partículas a buscar y pi-1 el número de partículas detectadas en la iteración anterior, el valor ti+1 de umbralizado de la iteración siguiente en el bucle de umbralizado se define como:

ti+1 = ti + ((2 \* n ) + 1 – (2 \* (pi-1))

Sea dj el valor de la densidad de pixel de la iteración j en el *bucle de densidad de pixel*, m el máximo número de partículas obtenido de la última ejecución del *bucle de umbralizado* (inicialmente m = 0 para d0) y n el número mínimos de partículas a buscar, el valor dj+1 se define como:

dj+ 1 = dj - ((200 \* n)/(m + 1))

De esta manera queda definido el algoritmo que detecta los digitos de una imagen de un crotal. Dicho algoritmo finalizará como se ha dicho en los puntos 1 y 3. Es decir, cuando se detecten igual o más componentes conexas de las que se metió por parámetro que se andaban buscando. O cuando se llega al límite inferior de densidad de pixel que puede tener un dígito en un crotal, establecido por defecto a 500 pixeles.

Una vez terminadas las iteraciones del algoritmo se procede, como comentamos anteriormente, a enderezar la imagen con el ángulo obtenido en la primera fase. Seguidamente se guarda la imagen resultante en una carpeta, que si no se especifica lo contrario en los parámetros de entrada del ejecutable, se denominará *output* y se encontrara en la misma ubicación que las imágenes de entrada.

Como se ha mejorado el algoritmo

Mejoras:

Rapidez

Números detectados (automatico)

Girar imágenes torcidas más de 90 º

REFERENCIAS

[1] <http://www.magrama.gob.es/es/ganaderia/temas/trazabilidad-animal/identificacion-animal/bovino/>

[2] <http://rsb.info.nih.gov/ij/>

[3] http://rsb.info.nih.gov/nih-image/

[4] T.W. Ridler, S. Calvard, Picture thresholding using an iterative selection method, IEEE Trans. System, Man and Cybernetics, SMC-8 (1978) 630-632.

[5] <http://imagejdocu.tudor.lu/doku.php?id=faq:technical:what_is_the_algorithm_used_in_automatic_thresholding>

[6] http://commons.apache.org/math/