Máster de Visión Artificial

Asignatura: Aplicaciones Industriales

Práctica puntuable – Preparación de la imagen

Orión García Gallardo

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 ejemploextracción de ruido, eliminación de toda la información no relevante, corrección imágenes torcidas, etc.

Como resultado de esta práctica, partiendo de las imágenes de crotalescontenidas en una carpeta,se devolverá una serie de imágenesbitonales que sólo contendrán los dígitos presentes en lasimágenes.

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 eimprimirsobre 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 fasese 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 unumbralizado 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. Estecálculose 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 y_i).

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 x_i el punto inicial y w la anchura de la imagen, su "homólogo" x_h se calcularía como:

Si $x_i < (w/2)$:$x_h = x_i + (w/2)$;

Si $x_i > (w/2)$:$x_h = x_i - (w/2)$;

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

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 (x_i, y_i), (x_h, y_h). 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 se 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 representaningún número). Por otro lado en dicho algoritmo iterativo también se modificael 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. Dondese realizan unas operaciones básicas sobre la imagen:

1. Binarización de la imagen. Se convierte la imagen a bytes (8-bits).
2. Umbralización de la imagen.Estableciendo el valor de umbral al valor que lleve la variable de umbralizado en cada iteración.Se simplifica de esta manera la imagen porque no es relevante saber ni el color ni el valor de gris para esta práctica
3. 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 en un proceso iterativo. 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 máximo de valor del umbral (80) 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 pixeles por elemento). 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.

Intentando simular lo que un humano haría para detectar dígitos manualmente de un crotal se ha diseñado un algoritmo iterativo con dos bucles. En el bucle más interno se modificaría el valor de la densidad de pixel (bucle de densidad de pixel), disminuyendo dicho valor, y en bucle exterior el valor del umbral (bucle de umbralizado), aumentando dicho valor. 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 d_j el valor de la densidad de pixel de la iteración j en el *bucle de densidad de pixel*, n el número mínimo de partículas a buscar y p_{i-1} el número de partículas detectadas en la iteración anterior, el valor d_{j+1} se define como:

$$dj+ 1 = dj - ((200 * n)/(p_{i-1} + 1))$$

Sea t_i el valor del umbralizado de la iteración i en el *bucle de umbralizado*, m el máximo número de partículas obtenido de la última ejecución del *bucle de densidad de pixel*(inicialmente m = 0 para t_0) y n el número mínimo de partículas a buscar, el valor t_{i+1} de umbralizado de la iteración siguiente en el bucle de umbralizadose define como:

$$t_{i+1} = t_i + ((2 * n) + 1 – (2 * m))$$

De esta manera queda definido el algoritmo que detecta losdígitosde una imagen de un crotal. Dicho algoritmo finalizará siempre con los casos básicos descritos anteriormente en 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 superior del umbral que puede tener un dígito en un crotal, establecido por defecto en 80.

Una vez terminadas las iteraciones del algoritmo se procede, como comentamos anteriormente, a enderezar la imagen con el ángulo obtenido en la primera fase. Finalmente 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 encontrará en la misma ubicación que las imágenes de entrada.

PRUEBAS REALIZADAS SOBRE LA MUESTRA

Para mejorar la eficiencia del algoritmo hemos realizado pruebas modificando la actualización de los alores que se modifican en cada iteración, estos son: el valor de la densidad de pixel y el valor del umbral. Dicho valor es vienen dados por:

$dj+ 1 = dj - ((w * n)/(p_{i-1} + 1))$

$t_{i+1} = t_i + ((2 * n) + 1 – (2 * (m + v)))$

Siendo w y v los pesos que se han ido ajustando hasta obtener los resultados más satisfactorios.

A continuación se muestra unas tablas con los valores de estos pesos, el tiempo total del algoritmo y la eficiencia en porcentaje de imágenes cuyos dígitos del crotal han sido correctamente detectados, tras haber pasado el algoritmo al conjunto de 10 muestras pasado en la práctica.

W	V	Tiempo Total (mseg)	Eficiencia (%)
100	1	7249	20
200	1	4930	30
300	1	2916	20
200	0	3355	60
200	-1	4965	40
150	0	4289	50
250	0	2608	50

Antes de analizar los datos hay que tener en cuenta una serie de consideraciones. En la muestra procesada hay dos imágenes que nunca podrán devolver ningún dígito. Estas son la imagen crotal10 .tiff que no contiene ningún crotal y la imagen crotal9.tiff que no se puede abrir con la librería ImageJ. Por lo tanto la eficiencia máxima que se puede alcanzar es del 80%.

Como se observa en la tabla anterior, el algoritmo tarda menos cuando los valores de w y v son 250 y 0 respectivamente. Sin embargo, el nivel de eficiencia es un 20% menor que el obtenido cuando estos valores son 200 y 0 respectivamente. Por otro lado la diferencia en tiempo de procesado entre coger unos y otros valores es de unos 0.7 segundos, es decir, 0.07 segundos por imagen. Por lo tanto, como el tiempo no es significante,se puede determinar que los valores de w y v que optimizan bastante este algoritmo son 200 y 0 respectivamente.

CONCLUSIONES

- Resumen de resultados obtenidos.
  Según los resultados obtenidos, con este algoritmo se tarda como media una duración de 0.3355 segundos para procesar cada imagen. En el caso mejor, que se ha dado en una imagen con dígitos muy gordos, se ha tardado 124 milisegundos en procesar la imagen. Por otro lado, en el caso peor, que se ha dado en una imagen con los números muy finos, se ha tardado764milisegundos. Esto se debe a que el algoritmo planteado empieza con

una densidad de pixel por dígito muy grande y la va disminuyendo en cada iteración. Consecuencia directa de esto es que le cuesta más encontrar estos dígitos finos.

- Mejoras que podrían realizarse.
  El algoritmo presentado tiene algunas carencias que podrían solventarse en futuras implementaciones. Entre ellas cabe destacar:
  - Sólo es capaz de enderezar crotales en un ángulo menor a los 90º. Una solución a este problema podría ser intentar detectar números con el algoritmo descrito y en caso de no detectar ningún dígito girar la imagen 90º y probar el algoritmo de nuevo.
  - Hacer más eficiente el algoritmo. Aunque actualmente es bastante rápido podría buscarse alguna forma de mejorar el tiempo de detección de números por crotal.

- Problemas que no se sabe cómo solucionar.
  El punto más destacable a mejorar debería ser la eficiencia. Actualmente en el caso mejor un 60% de los casos se detectan correctamente y se debería buscar mejorar este porcentaje.
  Otro de los problemas que no se ha sabido cómosolucionar, y relacionado con lo anterior, es automatizar el número de dígitos que se van a detectar por crotal. Este problema produce que cuando se recibe un crotal que tienen muchos dígitos, muchas de las veces, sólo se detecten el número de dígitos pasado por parámetro.

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/