Nota: Algunas de las imágenes que aparecen en esta presentación provienen del libro:
Visión por Computador: fundamentos y métodos.
Arturo de la Escalera Hueso. Prentice Hall.

Intelligent Systems Lab

Sistemas de Percepción

Visión por Computador

Arturo de la Escalera
José María Armingol
Fernando García
David Martín
Abdulla Al-Kaff

uc3m | Universidad Carlos III de Madrid

Intelligent Systems Lab

Etiquetado

Etiquetado

connectedComponents

computes the connected components labeled image of boolean image `image` with 4 or 8 way connectivity - returns N, the total number of labels [0, N-1] where 0 represents the background label. ltype specifies the output label image type, an important consideration based on the total number of labels or alternatively the total number of pixels in the source image.

C++: int `connectedComponents`(InputArray **image**, OutputArray **labels**, int **connectivity**=8, int **ltype**=CV_32S)

C++: int `connectedComponentsWithStats`(InputArray **image**, OutputArray **labels**, OutputArray **stats**, OutputArray **centroids**, int **connectivity**=8, int **ltype**=CV_32S)

Parameters:

- **image** – the image to be labeled
- **labels** – destination labeled image
- **connectivity** – 8 or 4 for 8-way or 4-way connectivity respectively
- **ltype** – output image label type. Currently CV_32S and CV_16U are supported.
- **statsv** –

  statistics output for each label, including the background label, see below for available statistics. Statistics are accessed via statsv(label, COLUMN) where available columns are defined below.

  - **CC_STAT_LEFT** The leftmost (x) coordinate which is the inclusive start of the bounding box in the horizontal direction.
  - **CC_STAT_TOP** The topmost (y) coordinate which is the inclusive start of the bounding box in the vertical direction.
  - **CC_STAT_WIDTH** The horizontal size of the bounding box
  - **CC_STAT_HEIGHT** The vertical size of the bounding box
  - **CC_STAT_AREA** The total area (in pixels) of the connected component
- **centroids** – floating point centroid (x,y) output for each label, including the background label

Etiquetado

Intelligent Systems Lab

RNG::uniform

Returns the next random number sampled from the uniform distribution.

C++: int `RNG::uniform`(int **a**, int **b**)

C++: float `RNG::uniform`(float **a**, float **b**)

C++: double `RNG::uniform`(double **a**, double **b**)

Parameters:
- **a** – lower inclusive boundary of the returned random numbers.
- **b** – upper non-inclusive boundary of the returned random numbers.

The methods transform the state using the MWC algorithm and return the next uniformly-distributed random number of the specified type, deduced from the input parameter type, from the range `[a, b)` . There is a nuance illustrated by the following sample:

```
RNG rng;

// always produces 0
double a = rng.uniform(0, 1);

// produces double from [0, 1)
double a1 = rng.uniform((double)0, (double)1);

// produces float from [0, 1)
double b = rng.uniform(0.f, 1.f);

// produces double from [0, 1)
double c = rng.uniform(0., 1.);
```

Etiquetado

- Ejemplo
  - Cargar una imagen
  - Comprobar que existe
  - Llamar a la función *connectedComponents*
  - Contar el número de etiquetas (objetos)
  - Dibujar las *bounding boxes* en colores distintos
  - Dibujar los centroides
  - Mostrar la imagen original y la dibujada
  - Esperar a pulsar una tecla
  - Liberar memoria

Etiquetado

- Etiquetado: encontrar los contornos en imagen binaria

C++: void **findContours**(InputOutputArray **image**, OutputArrayOfArrays **contours**, OutputArray **hierarchy**, int **mode**, int **method**, Point **offset**=Point())

C++: void **findContours**(InputOutputArray **image**, OutputArrayOfArrays **contours**, int **mode**, int **method**, Point **offset**=Point())

Parameters:

- **image** – Source, an 8-bit single-channel image. Non-zero pixels are treated as 1's. Zero pixels remain 0's, so the image is treated as `binary` . You can use **compare()** , **inRange()** , **threshold()** , **adaptiveThreshold()** , **Canny()** , and others to create a binary image out of a grayscale or color one. The function modifies the `image` while extracting the contours. If mode equals to `CV_RETR_CCOMP` or `CV_RETR_FLOODFILL`, the input can also be a 32-bit integer image of labels (`CV_32SC1`).
- **contours** – Detected contours. Each contour is stored as a vector of points.
- **hierarchy** – Optional output vector, containing information about the image topology. It has as many elements as the number of contours. For each i-th contour `contours[i]` , the elements `hierarchy[i][0]` , `hiearchy[i][1]` , `hiearchy[i][2]` , and `hiearchy[i][3]` are set to 0-based indices in `contours` of the next and previous contours at the same hierarchical level, the first child contour and the parent contour, respectively. If for the contour `i` there are no next, previous, parent, or nested contours, the corresponding elements of `hierarchy[i]` will be negative.

Etiquetado

- Etiquetado: encontrar los contornos en imagen binaria

C++: void **findContours**(InputOutputArray **image**, OutputArrayOfArrays **contours**, OutputArray **hierarchy**, int **mode**, int **method**, Point **offset**=Point())

- **mode** –

  Contour retrieval mode (if you use Python see also a note below).

  - **CV_RETR_EXTERNAL** retrieves only the extreme outer contours. It sets `hierarchy[i][2]=hierarchy[i][3]=-1` for all the contours.
  - **CV_RETR_LIST** retrieves all of the contours without establishing any hierarchical relationships.
  - **CV_RETR_CCOMP** retrieves all of the contours and organizes them into a two-level hierarchy. At the top level, there are external boundaries of the components. At the second level, there are boundaries of the holes. If there is another contour inside a hole of a connected component, it is still put at the top level.
  - **CV_RETR_TREE** retrieves all of the contours and reconstructs a full hierarchy of nested contours. This full hierarchy is built and shown in the OpenCV `contours.c` demo.
- **method** –

  Contour approximation method (if you use Python see also a note below).

  - **CV_CHAIN_APPROX_NONE** stores absolutely all the contour points. That is, any 2 subsequent points `(x1,y1)` and `(x2,y2)` of the contour will be either horizontal, vertical or diagonal neighbors, that is, `max(abs(x1-x2),abs(y2-y1))==1`.
  - **CV_CHAIN_APPROX_SIMPLE** compresses horizontal, vertical, and diagonal segments and leaves only their end points. For example, an up-right rectangular contour is encoded with 4 points.
  - **CV_CHAIN_APPROX_TC89_L1,CV_CHAIN_APPROX_TC89_KCOS** applies one of the flavors of the Teh-Chin chain approximation algorithm. See [TehChin89] for details.
- **offset** – Optional offset by which every contour point is shifted. This is useful if the contours are extracted from the image ROI and then they should be analyzed in the whole image context.

Etiquetado

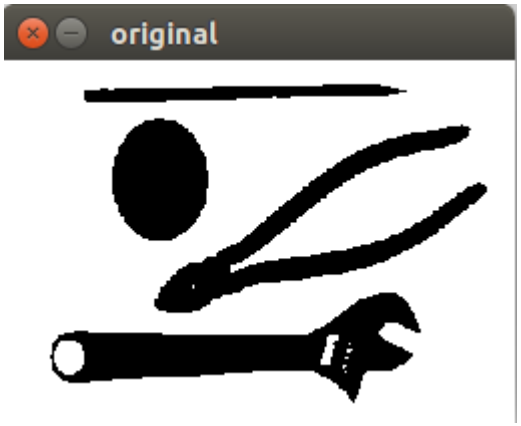
- Etiquetado : dibujar los contornos exteriores o interiores

C++: void **drawContours**(InputOutputArray **image**, InputArrayOfArrays **contours**, int **contourIdx**, const Scalar& **color**, int **thickness**=1, int **lineType**=8, InputArray **hierarchy**=noArray(), int **maxLevel**=INT_MAX, Point **offset**=Point())

Parameters:

- **image** – Destination image.
- **contours** – All the input contours. Each contour is stored as a point vector.
- **contourIdx** – Parameter indicating a contour to draw. If it is negative, all the contours are drawn.
- **color** – Color of the contours.
- **thickness** – Thickness of lines the contours are drawn with. If it is negative (for example, `thickness=CV_FILLED`), the contour interiors are drawn.
- **lineType** – Line connectivity. See **`line()`** for details.
- **hierarchy** – Optional information about hierarchy. It is only needed if you want to draw only some of the contours (see `maxLevel`).
- **maxLevel** – Maximal level for drawn contours. If it is 0, only the specified contour is drawn. If it is 1, the function draws the contour(s) and all the nested contours. If it is 2, the function draws the contours, all the nested contours, all the nested-to-nested contours, and so on. This parameter is only taken into account when there is `hierarchy` available.
- **offset** – Optional contour shift parameter. Shift all the drawn contours by the specified `offset` $= (dx, dy)$.
- **contour** – Pointer to the first contour.
- **externalColor** – Color of external contours.
- **holeColor** – Color of internal contours (holes).

Etiquetado




```
//thresholded image
Mat im_thresholded(img_src.size(), CV_8UC1);
threshold_value = MinMax(hist_vector);
threshold(img_src, im_thresholded, threshold_value, 255,THRESH_BINARY_INV);
 // Find the contours
 vector<vector<Point> > contours;

 findContours(im_thresholded, contours,RETR_EXTERNAL,CHAIN_APPROX_SIMPLE);

 cout << "Numero de contornos detectados es " << contours.size() << endl;
 Mat img_contornos(im_thresholded.size(), CV_8UC3, Scalar(0, 0, 0));
 for( size_t k = 0; k < contours.size(); k++ ){
         Scalar color( rand()&255, rand()&255, rand()&255 );
         drawContours(img_contornos,contours, k,color);
 }
```

Etiquetado

- Etiquetado: Ejercicio 1, contar los granos en “cafe.tif”
  - Calcular el histograma.
  - Umbralización de la imagen utilizando el histograma (función MinMax).
  - Erosionar la imagen binaria 5 veces.
  - Etiquetado.