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

Modificación del Contraste: Amplitud del Histograma, Ecualización y LUTs

Amplitud del histograma

- Funciones de interés:

minMaxLoc

Finds the global minimum and maximum in an array.

C++: void `minMaxLoc`(InputArray **src**, double* **minVal**, double* **maxVal**=0, Point* **minLoc**=0, Point* **maxLoc**=0, InputArray **mask**=noArray())

Parameters:
- **src** – input single-channel array.
- **minVal** – pointer to the returned minimum value; `NULL` is used if not required.
- **maxVal** – pointer to the returned maximum value; `NULL` is used if not required.
- **minLoc** – pointer to the returned minimum location (in 2D case); `NULL` is used if not required.
- **maxLoc** – pointer to the returned maximum location (in 2D case); `NULL` is used if not required.
- **mask** – optional mask used to select a sub-array.

The functions `minMaxLoc` find the minimum and maximum element values and their positions. The extremums are searched across the whole array or, if `mask` is not an empty array, in the specified array region.

Amplitud del histograma

- Funciones de interés:

normalize

Normalizes the norm or value range of an array.

C++: void **normalize**(InputArray **src**, OutputArray **dst**, double **alpha**=1, double **beta**=0, int **norm_type**=NORM_L2, int **dtype**=-1, InputArray **mask**=noArray())

C++: void **normalize**(const SparseMat& **src**, SparseMat& **dst**, double **alpha**, int **normType**)

Parameters:
- **src** – input array.
- **dst** – output array of the same size as `src` .
- **alpha** – norm value to normalize to or the lower range boundary in case of the range normalization.
- **beta** – upper range boundary in case of the range normalization; it is not used for the norm normalization.
- **normType** – normalization type (see the details below).
- **dtype** – when negative, the output array has the same type as `src`; otherwise, it has the same number of channels as `src` and the depth `=CV_MAT_DEPTH(dtype)`.
- **mask** – optional operation mask.

The functions `normalize` scale and shift the input array elements so that

$$\|\texttt{dst}\|_{L_p} = \texttt{alpha}$$

(where p=Inf, 1 or 2) when `normType=NORM_INF`, `NORM_L1`, or `NORM_L2`, respectively; or so that

$$\min_I \texttt{dst}(I) = \texttt{alpha}, \ \max_I \texttt{dst}(I) = \texttt{beta}$$

when `normType=NORM_MINMAX` (for dense arrays only). The optional mask specifies a sub-array to be normalized. This means that the norm or min-n-max are calculated over the sub-array, and then this sub-array is modified to be normalized. If you want to only use the mask to calculate the norm or min-max but modify the whole array, you can use **`norm()`** and **`Mat::convertTo()`**.

Amplitud del histograma

- Cargar la imagen
- Calcular el histograma
- Imprimir los valores del histograma (original) en el terminal
- Localizar el mínimo (Min) y el máximo (Max)
- Mostrar el histograma
- Mostrar las imágenes
- Esperar a que se pulse una tecla
- Liberar la memoria
- Finalizar el programa

Amplitud del histograma

- e01_minMaxLoc.cpp

```
14      original_img = imread("rayosx.tif", CV_LOAD_IMAGE_GRAYSCALE);
15      if (!original_img.data){
16          cout << "error loading image" << endl;
17          return 1;
18      }
19      // Initialize histogram parameters
20      /// Establish the number of bins
21      int histSize = 256;
22      /// Set the ranges GrayScale 0-255
23      float range[] = { 0, 256 };
24      const float* histRange = { range };
25      /// Set histogram image
26      int hist_w = 512; int hist_h = 400;
27      int bin_w = cvRound((double)hist_w / histSize);
28      Mat histImage(hist_h, hist_w, CV_8UC3, Scalar(0, 0, 0));
29
30      // Calculate histogram
31      Mat original_hist, normalized_hist;
32      calcHist(&original_img, 1, 0, Mat(), original_hist, 1, &histSize, &histRange, true, false);
```

Amplitud del histograma

- e01_minMaxLoc.cpp

```
34      // Print the values of the original histogram on console
35      cout << "Original histogram" << endl;
36      for (int h = 0; h < histSize; h++)
37      {
38          float binVal = original_hist.at<float>(h);
39          cout << " " << binVal;
40      }
41      cout << endl;
42
43      // Min max localization
44      Point min_point, max_point;
45      double min, max;
46      minMaxLoc(original_hist, &min, &max, &min_point, &max_point, Mat());
47
48      // Print min and max value and its localization
49      cout << "minimun value: " << min << " at index:" << min_point.y << endl;
50      cout << "Maximun value: " << max << " at index:" << max_point.y << endl << endl;
```

Amplitud del histograma

```
52      /// Normalize the result to [ 0, histImage.rows ]
53      normalize(original_hist, normalized_hist, 0, histImage.rows, NORM_MINMAX, -1, Mat());
54
55      // Print the values on console
56      cout << "Normalized histogram" << endl;
57      for (int h = 0; h < histSize; h++)
58      {
59          float binVal = normalized_hist.at<float>(h);
60          cout << " " << binVal;
61      }
62      cout << endl;
63
64      minMaxLoc(normalized_hist, &min, &max, &min_point, &max_point, Mat());
65
66      // Print min and max value and its localization
67      cout << "minimun value: " << min << " at index:" << min_point.y << endl;
68      cout << "Maximun value: " << max << " at index:" << max_point.y << endl;
69
70      /// Plot histogram
71      for (int i = 1; i < histSize; i++)
72      {
73          // Line from the previous vale to the current one. Unfilled. Width 2 (bin_w)
74          //line(histImage,
75          //  Point(bin_w*(i - 1), hist_h - cvRound(normalized_hist.at<float>(i - 1))),
76          //  Point(bin_w*(i), hist_h - cvRound(normalized_hist.at<float>(i))),
77          //  Scalar(0, 255, 0), 2, 8, 0);
78
79          //  Line of width 2 (bin_w = 512 width / 256 gray scale values) filled
80          line(histImage,
81              Point(bin_w*(i), hist_w),
82              Point(bin_w*(i), hist_h - cvRound(normalized_hist.at<float>(i))),
83              Scalar(255, 0, 0), bin_w, 8, 0);
84      }
```

Amplitud del histograma

- e01_minMaxLoc.cpp
  - ¡Importante! Liberar memoria de la imágenes (Mat)

```
87      namedWindow("Original picture", CV_WINDOW_AUTOSIZE);
88      namedWindow("histogram", CV_WINDOW_AUTOSIZE);
89
90      // Show image in the name of the window
91      imshow("Original picture", original_img);
92      //imshow("Normalized picture", normalized_img);
93      imshow("histogram", histImage);
94
95      // Function for show the image in ms.
96      waitKey(0);
97      // Free memory
98      histImage.release();
99      original_hist.release();
100     destroyAllWindows();
101     // End of the program
102     return 0;
103 }
```

Ecualización del histograma

Intelligent Systems Lab

equalizeHist

Equalizes the histogram of a grayscale image.

C++: void **equalizeHist**(InputArray **src**, OutputArray **dst**)

Parameters:
- **src** – Source 8-bit single channel image.
- **dst** – Destination image of the same size and type as `src` .

The function equalizes the histogram of the input image using the following algorithm:

1. Calculate the histogram H for `src` .
2. Normalize the histogram so that the sum of histogram bins is 255.
3. Compute the integral of the histogram:

$$H'_i = \sum_{0 \le j < i} H(j)$$

4. Transform the image using H' as a look-up table: $\mathtt{dst}(x,y) = H'(\mathtt{src}(x,y))$

The algorithm normalizes the brightness and increases the contrast of the image.

Ecualización del histograma

Intelligent Systems Lab

- e02_histogramEqualization:
  - Cargar una imagen
  - Calcular el histograma
  - Imprimir los valores del histograma de la imagen original en el terminal
  - Ecualizar el histograma
  - Imprimir los valores del histograma ecualizado en el terminal
  - Mostrar el histograma
  - Mostrar las imágenes
  - Esperar a que se pulse una tecla
  - Liberar la memoria
  - Finalizar el programa

Ecualización del histograma


```
8  int main(int argc, char* argv[])
9  {
10     // Object instantiation
11     Mat original_img, equalized_img;
12
13     // Load image from disk
14     original_img = imread("rayosx.tif", CV_LOAD_IMAGE_GRAYSCALE);
15     if (!original_img.data){
16         cout << "error loading image" << endl;
17         return 1;
18     }
19     // Initialize histogram parameters
20     /// Establish the number of bins
21     int histSize = 256;
22     /// Set the ranges GrayScale 0-255
23     float range[] = { 0, 256 };
24     const float* histRange = { range };
25     /// Set histogram image
26     int hist_w = 512; int hist_h = 400;
27     int bin_w = cvRound((double)hist_w / histSize);
28     Mat histImage(hist_h, hist_w, CV_8UC3, Scalar(0, 0, 0));
29     Mat equalizedHistImage(hist_h, hist_w, CV_8UC3, Scalar(0, 0, 0));
30
31     // Calculate histogram
32     Mat original_hist, normalized_hist, equalized_hist, equalized_normalized_hist;
33     calcHist(&original_img, 1, 0, Mat(), original_hist, 1, &histSize, &histRange, true, false);
```

Ecualización del histograma

Intelligent Systems Lab

```
35      // Print the values of the original histogram on console
36      cout << "Original histogram" << endl;
37      for (int h = 0; h < histSize; h++)
38      {
39          float binVal = original_hist.at<float>(h);
40          cout << " " << binVal;
41      }
42      cout << endl;
43
44      /// Normalize the result to [ 0, histImage.rows ]
45      normalize(original_hist, normalized_hist, 0, histImage.rows, NORM_MINMAX, -1, Mat());
46
47      // Print the values on console
48      cout << "Normalized histogram" << endl;
49      for (int h = 0; h < histSize; h++)
50      {
51          float binVal = normalized_hist.at<float>(h);
52          cout << " " << binVal;
53      }
54      cout << endl;
```

Ecualización del histograma

```
56      // Equalize histogram from a grayscale image
57      equalizeHist(original_img, equalized_img);
58      calcHist(&equalized_img, 1, 0, Mat(), equalized_hist, 1, &histSize, &histRange, true, false);
59
60      // Print the values on console
61      cout << "Equalized histogram" << endl;
62      for (int h = 0; h < histSize; h++)
63      {
64          float binVal = equalized_hist.at<float>(h);
65          cout << " " << binVal;
66      }
67      cout << endl;
68
69      // Normalize the equalized histogram to fit in the histogram image
70      normalize(equalized_hist, equalized_normalized_hist, 0, histImage.rows, NORM_MINMAX, -1, Mat());
71
72      // Print the values on console
73      cout << "Equalized normalized histogram" << endl;
74      for (int h = 0; h < histSize; h++)
75      {
76          float binVal = equalized_normalized_hist.at<float>(h);
77          cout << " " << binVal;
78      }
79      cout << endl;
80
```

Ecualización del histograma

```
81      /// Plot histograms
82      for (int i = 1; i < histSize; i++)
83      {
84          //  Line of width 2 (bin_w = 512 width / 256 gray scale values) filled
85          line(histImage,
86              Point(bin_w*(i), hist_w),
87              Point(bin_w*(i), hist_h - cvRound(normalized_hist.at<float>(i))),
88              Scalar(255, 0, 0), bin_w, 8, 0);
89          line(equalizedHistImage,
90              Point(bin_w*(i), hist_w),
91              Point(bin_w*(i), hist_h - cvRound(equalized_normalized_hist.at<float>(i))),
92              Scalar(0, 255, 0), bin_w, 8, 0);
93      }
94      // Windows for all the images
95      namedWindow("Original picture", CV_WINDOW_AUTOSIZE);
96      namedWindow("Equalized picture", CV_WINDOW_AUTOSIZE);
97      namedWindow("Original histogram", CV_WINDOW_AUTOSIZE);
98      namedWindow("Equalized histogram", CV_WINDOW_AUTOSIZE);
99      // Show image in the name of the window
100     imshow("Original picture", original_img);
101     imshow("Equalized picture", equalized_img);
102     imshow("Original histogram", histImage);
103     imshow("Equalized histogram", equalizedHistImage);
104     // Function for show the image in ms.
105     waitKey(0);
```

Ecualización del histograma

Intelligent Systems Lab

IMPORTANTE!! Liberar Memoria!

```
107     // Free memory
108     original_img.release();
109     equalized_img.release();
110     histImage.release();
111     equalized_img.release();
112     original_hist.release();
113     normalized_hist.release();
114     equalized_normalized_hist.release();
115     destroyAllWindows();
116     // End of the program
117     return 0;
118 }
```

Modificación del contraste y LUTs

- Cargar la imagen
- Calcular el histograma
- Mostrar los valores del histograma de la imagen original en el terminal
- Generar y aplicar la función de LUT
- Calcular el histograma con la LUT
- Mostrar los valores del histograma resultante de la LUT en el terminal
- Mostrar el histograma
- Mostrar las imágenes
- Esperar la pulsación de una tecla
- Liberar la memoria y finalizar el programa

Modificación del contraste y LUTs

Intelligent Systems Lab

LUT

Performs a look-up table transform of an array.

C++: void **LUT**(InputArray **src**, InputArray **lut**, OutputArray **dst**)

Parameters:
- **src** – input array of 8-bit elements.
- **lut** – look-up table of 256 elements; in case of multi-channel input array, the table should either have a single channel (in this case the same table is used for all channels) or the same number of channels as in the input array.
- **dst** – output array of the same size and number of channels as `src`, and the same depth as `lut`.

```
56      // Lut: functions from grayscale image
57      Mat lut(1, 256, CV_8U);
58      // Make the LUT function
59      for (int i = 0; i < 256; i++){
60          // Inverse function
61          lut.at<uchar>(i) = 255 - i;
62          // Square root function
63          //lut.at<uchar>(i) = pow((float)i * 255, (float)(1 / 2.0));
64          // Cubic function
65          //lut.at<uchar>(i) = pow((float)i, (float)3.0) / (255 * 255);
66      }
67      LUT(original_img, lut, lut_img);
```

Modificación del contraste y LUTs

```
8  int main(int argc, char* argv[])
9  {
10     // Object
11     Mat original_img, lut_img;
12
13     // Load image from disk
14     original_img = imread("backlighting2.jpg", CV_LOAD_IMAGE_GRAYSCALE);
15     if (!original_img.data){
16         cout << "error loading image" << endl;
17         return 1;
18     }
19     // Initialize histogram parameters
20     /// Establish the number of bins
21     int histSize = 256;
22     /// Set the ranges GrayScale 0-255
23     float range[] = { 0, 256 };
24     const float* histRange = { range };
25     /// Set histogram image
26     int hist_w = 512; int hist_h = 400;
27     int bin_w = cvRound((double)hist_w / histSize);
28     Mat histImage(hist_h, hist_w, CV_8UC3, Scalar(0, 0, 0));
29     Mat lutHistImage(hist_h, hist_w, CV_8UC3, Scalar(0, 0, 0));
30
31     // Calculate histogram
32     Mat original_hist, normalized_hist, lut_hist, lut_normalized_hist;
33     calcHist(&original_img, 1, 0, Mat(), original_hist, 1, &histSize, &histRange, true, false);
```

Modificación del contraste y LUTs

```
35      // Print the values of the original histogram on console
36      cout << "Original histogram" << endl;
37      for (int h = 0; h < histSize; h++)
38      {
39          float binVal = original_hist.at<float>(h);
40          cout << " " << binVal;
41      }
42      cout << endl << endl;
43
44      /// Normalize the result to [ 0, histImage.rows ]
45      normalize(original_hist, normalized_hist, 0, histImage.rows, NORM_MINMAX, -1, Mat());
46
47      // Print the values on console
48      cout << "Normalized histogram" << endl;
49      for (int h = 0; h < histSize; h++)
50      {
51          float binVal = normalized_hist.at<float>(h);
52          cout << " " << binVal;
53      }
54      cout << endl << endl;
55
56      // Lut: functions from grayscale image
57      Mat lut(1, 256, CV_8U);
58      // Make the LUT function
59      for (int i = 0; i < 256; i++){
60          // Inverse function
61          lut.at<uchar>(i) = 255 - i;
62          // Square root function
63          //lut.at<uchar>(i) = pow((float)i * 255, (float)(1 / 2.0));
64          // Cubic function
65          //lut.at<uchar>(i) = pow((float)i, (float)3.0) / (255 * 255);
66      }
67      LUT(original_img, lut, lut_img);
```

Modificación del contraste y LUTs

Intelligent Systems Lab

```
69      // Calculate lut histogram
70      calcHist(&lut_img, 1, 0, Mat(), lut_hist, 1, &histSize, &histRange, true, false);
71
72      // Print the values on console
73      cout << "Lut histogram" << endl;
74      for (int h = 0; h < histSize; h++)
75      {
76          float binVal = lut_hist.at<float>(h);
77          cout << " " << binVal;
78      }
79      cout << endl << endl;
80
81      // Normalize lut histogram to fit in the histogram image
82      normalize(lut_hist, lut_normalized_hist, 0, histImage.rows, NORM_MINMAX, -1, Mat());
83
84      // Print the values on console
85      cout << "Lut normalized histogram" << endl;
86      for (int h = 0; h < histSize; h++)
87      {
88          float binVal = lut_normalized_hist.at<float>(h);
89          cout << " " << binVal;
90      }
91      cout << endl << endl;
```

Modificación del contraste y LUTs

```
93      /// Plot histograms
94      for (int i = 1; i < histSize; i++)
95      {
96          //  Line of width 2 (bin_w = 512 width / 256 gray scale values) filled
97          line(histImage,
98              Point(bin_w*(i), hist_w),
99              Point(bin_w*(i), hist_h - cvRound(normalized_hist.at<float>(i))),
100             Scalar(255, 0, 0), bin_w, 8, 0);
101         line(lutHistImage,
102             Point(bin_w*(i), hist_w),
103             Point(bin_w*(i), hist_h - cvRound(lut_normalized_hist.at<float>(i))),
104             Scalar(0, 255, 0), bin_w, 8, 0);
105     }
106     // Windows for all the images
107     namedWindow("Original picture", CV_WINDOW_AUTOSIZE);
108     namedWindow("Lut picture", CV_WINDOW_AUTOSIZE);
109     namedWindow("Original histogram", CV_WINDOW_AUTOSIZE);
110     namedWindow("Lut histogram", CV_WINDOW_AUTOSIZE);
111     // Show image in the name of the window
112     imshow("Original picture", original_img);
113     imshow("Lut picture", lut_img);
114     imshow("Original histogram", histImage);
115     imshow("Lut histogram", lutHistImage);
116     // Function for show the image in ms.
117     waitKey(0);
```

Modificación del contraste y LUTs

```
119     // Free memory
120     original_img.release();
121     lut_img.release();
122     histImage.release();
123     lutHistImage.release();
124     original_hist.release();
125     normalized_hist.release();
126     lut_hist.release();
127     lut_normalized_hist.release();
128     destroyAllWindows();
129     // End of the program
130     return 0;
131 }
```