Visión por computador, Trabajo 0

Cuestionario

1. ¿Qué relación hay en OpenCV entre imágenes y matrices? Justificar la respuesta

En OpenCV imagen y matriz son semejantes, ambos se tratan de igual forma y se representan utilizando objetos de la clase Mat la cual está incluida en OpenCV.

2. Diga el significado de los siguientes tipos OpenCV: 8UC1, 8UC2, 8UC3, 32SC1, 32SC2, 32SC3, 32FC1, 32FC2, 32FC3. ¿Cuáles de ellos están asociados a imágenes? Justificar la respuesta.

Matrices de celdas de 8-bit unsigned:
- 8UC1: de 1 canal por celda, correspondiente a imágenes.
- 8UC2: de 2 canales por celda.
- 8UC3: de 3 canales por celda, correspondiente a imágenes.

Matrices de celdas de 32-bit signed:
- 32SC1: de 1 canal por celda, correspondiente a imágenes.
- 32SC2: de 2 canales por celda.
- 32SC3: de 3 canales por celda, correspondiente a imágenes.

Matrices de celdas de 32-bit float:
- 32FC1: de 1 canal por celda, correspondiente a imágenes.
- 32FC2: de 2 canales por celda.
- 32FC3: de 3 canales por celda, correspondiente a imágenes.

3. ¿Qué relación existe entre cada tipo visual de una imagen: (color, grises, blanco y negro) y los tipos de almacenamiento de OpenCV ? Justificar la respuesta

En los tres tipos sería necesario 8 bits para almacenar la información de intensidad de color pudiendo representar hasta 256 valores por canal. Por lo tanto:
- Las imágenes en blanco y negro solo tienen dos niveles de intensidad: 0 a 255 con un solo canal.
- Las imágenes en escala de gris su intensidad varia entre 0 y 255 teniendo un solo canal.
- Las imágenes en color tienen que almacenar la información para la intensidad de los tres colores básicos (RGB) necesitando tres canales variando ellos entre 0 y 255, pudiendo tener un total de 255^3 valores de intensidad.

4. ¿Es posible realizar operaciones entre imágenes de distinto tipo visual? Justificar la respuesta

Si sería posible ya que en OpenCV usamos la clase Mat para representar imágenes. Podemos usar las operaciones para saber que tipo de imagen estamos tratando y así operar sobre ellas.

5. ¿Cuál es la orden OpenCV que permite transformar el tipo de almacenamiento de una matriz en otro distinto?

Tenemos el método convertTo de la clase Mat al cual se le pasa como parámetros el Mat de salida, y el tipo que se quiere obtener.

6. ¿Cuál es la orden OpenCV que permite transformar el tipo visual de una imagen en otro distinto? ¿Por qué es distinta de la que transforma un tipo de almacenamiento en otro?

Tenemos el método cvtColor que recibe la Mat origen, Mat destino y el int para el tipo de conversión de color que se quiere realizar.

Ejercicios

1. Escribir una función que lea una imagen en niveles de gris o en color (im=leeimagen(filename, flagColor))

Mi función leeimagen recibe el nombre del archivo y si será en tono de gris o en color para cargarla en un Mat que se devolverá.

```
1   Mat leeimagen(string filename, int flagColor) {
2       Mat res = imread(filename, flagColor);
3       return res;
4   }
5   void Ejercicio1Trabajo0() {
6       Mat img;
7       img = leeimagen("imagenes/lena.jpg", 1);
8   }
```

2. Escribir una función que visualice una imagen (pintal(im))

La función pintaI recibirá una Mat la cual visualizará en una ventana.

```
1   void pintaI(Mat im) {
2       namedWindow("Ventana", im.channels());
3       imshow("Ventana", im);
4       cvWaitKey();
5       destroyWindow("Ventana");
6   }
7   void Ejercicio2Trabajo0() {
8       Mat img;
9       img = leeimagen("imagenes/lena.jpg", 1);
10      pintaI(img);
11  }
```

3. Escribir una función que visualice varias imágenes a la vez: pintaMI(vim). (vim será una secuencia de imágenes) ¿Qué pasa si las imágenes no son todas del mismo tipo: (nivel de gris, color, blanco-negro)?

La función recibe un vector de Mat que irá mostrándolas una por una y finalmente cerrará todas las ventanas. Por lo tanto si las imágenes no son del mismo tipo, no pasará nada, ya que cada una se mostrará con su canal.

```
1   void pintaMI(vector<Mat> vim) {
2       cout << vim.size() << endl;
3       string windowname = "Ventana ";
4       string auxwindowname = windowname;
5       for (int i = 0; i < vim.size(); i++) {
6           auxwindowname.operator+=(to_string(i + 1));
7           namedWindow(auxwindowname, vim[i].channels());
8           imshow(auxwindowname, vim[i]);
9           auxwindowname = windowname;
10      }
11      cvWaitKey();
12      destroyAllWindows();
13  }
```

```
14  void Ejercicio3Trabajo0() {
15      Mat img1 = leeimagen("imagenes/lena.jpg", 1);
16      Mat img2 = leeimagen("imagenes/lena.jpg", 0);
17      vector<Mat> vim;
18      vim.push_back(img1);
19      vim.push_back(img2);
20      pintaMI(vim);
21  }
```

4. Escribir una función que modifique el valor en una imagen de una lista de coordenadas de píxeles.

En la función vamos a recibir un Mat, donde vamos a modificar los píxeles, y un vector de Point con las coordenadas del pixel a modificar.

```
1   void modificapixeles(Mat img, vector<Point> pixeles) {
2       Vec3b punto;
3       punto.val[0] = 255;
4       punto.val[1] = 0;
5       punto.val[2] = 0;
6       for (int i = 0; i < pixeles.size(); i++) {
7           img.at<Vec3b>(pixeles[i]) = punto;
8       }
9   }
10  void Ejercicio4Trabajo0() {
11      Mat img = leeimagen("imagenes/lena.jpg", 1);
12      vector<Point> pixeles;
13
14      Point pixel = (10, 10);
15      pixeles.push_back(pixel);
16      pixel.x = 20;
17      pixel.y = 20;
18      pixeles.push_back(pixel);
19      pixel.x = 50;
20      pixel.y = 50;
21      pixeles.push_back(pixel);
22
23      modificapixeles(img, pixeles);
24
25      pintaI(img);
26  }
```

NOTA: He adjuntado en el código los ejercicios de la guía de instalación de OpenCV.