https://github.com/Yujadeh

Cuestiones

¿Qué relación hay en OpenCV entre imágenes y matrices? Justificar la respuesta
Las imágenes podemos verlas como una sucesión de píxeles y almacenarlas así en forma de matriz para un mejor manejo de los datos o información de la que dispongamos. Cada casilla de nuestra matriz será un píxel de nuestra imagen.

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.

En OpenCV para declarar un objeto en el cuál guardar o crear alguna imagen tenemos el tipo Mat, basicamente una matriz en la que guardaremos los píxeles de las imágenes.

Los objetos de tipo Mat pueden tener varios aspectos atendiendo a la visualización que queramos dar a las images(escala de grises, a color, bordes, …) y así al tipo de objeto le daremos uno de los tipos del enunciado según el uso que le vayamos a dar la imagen. Así el tipo que le pasamos a un objeto Mat es:

CV_{nº bits profundidad}{U|S|F|D}C{nº channel}

Donde X es el número de bits que queramos dar al array, seguido del tipo de dato, U es Unsigned , S es Signed, F es Float y D Double. Por último indicamos el número de canales atendiendo al tipo de fotografía, esto es que a cada píxel de la imagen se le atribuye un conjunto de colores, así si indicamos que el número de canales es 1, estaremos ante una imagen en escalas de grises, si es 3 podemos visualizar una imagen a color, es decir, RGB o HSV, y si es 4 una imagen de tipo CMYK.

¿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 primer lugar usaremos el tipo de almacenamiento de un canal para visualizar imágenes en escala de grises o blanco y negro. El número de bits nos dirá la resolución de nuestra imagen. El tipo de dato a usar CV_X{U|S|F|D}C1.

Por otro lado para ver imágenes en color usaremos tipos de 3 canales. Tenemos dos tipos de imágenes de tres canales:

- RGB: Imagen que cuenta con los canales rojo, azul y verde. Según el número de bits que demos a cada canal cambiaremos la resolución de nuestra imagen. Así la imagen RGB estandar es de 24 bits contando con canales de 8 bits, usando así los tipos CV_8{U|S|F|D}C3 y la imagen en alta resolución es de 48 bits usando canales de 16 bits, es decir, el tipo usado es CV_16{U|S|F|D}C3
- HSV. A diferencia de el anterior tipo, solo usa dos canales con color y el restante lo usa para el brillo.

Por último tenemos las imágenes que usan 4 canales, para así disponer de una mayor gama de colores. Su principal uso es industrial. Una imagen de este tipo suele ser de 32 bits con canales de 8 bits cada uno.

¿Es posible realizar operaciones entre imágenes de distinto tipo visual? Justificar la respuesta
Debido a que las imágenes como hemos comentado las podemos codificar en matrices, estas matrices son distintas debido al distinto tipo visual al que nos enfrentamos en cada caso y por ello el tipo visual define la matriz a usar, dando así matrices con distintas dimensiones y haciendo imposible sus operaciones.

¿Cuál es la orden OpenCV que permite transformar el tipo de almacenamiento de una matriz en otro distinto?
Además del tipo Mat ya mencionado anteriormente podemos hacer uso de los tipos CvMat e IPLImage para poder guardar nuestras imágenes. Para ello si leemos una imagen y la guardamos en una estructura de tipo IPLImage podemos transformar la estructura a una de tipo Mat con la siguiente orden:

```
IplImage* img = cvLoadImage("lena.jpg", 1);
Mat mtx(img);
```

Por otro lado si deseamos guardar la información en una estructura de tipo IPLimage desde una estructura Mat simplemente hacemos esto:

```
IplImage dst_img = img;
```

Para guardar información en el tipo cvMat ejecutamos:

```
cvMat oldmat =img;
```

¿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? Solución

La orden para cambiar el tipo de visualización es:

```
cvtColor('imagen original', 'imagen nueva', 'flag del tipo');
```

Donde el parámetro “imagen original“ es la imagen en el tipo de visualización que sea, “imagen nueva“ será la estructura donde guardaremos esa misma imagen pero con distinto tipo de visualización y “flag del tipo“ es un parámetro en el cual indicaremos si queremos pasar de un tipo a otro. En este último parámetro haremos uso de las macros proporcionadas por openCV, siendo la macro CV_GRAY2RGB la macro para poder convertir una imagen en escala de grises a color y CV_RGB2GRAY la macro para convertir una imagen a color en escalas de grises.

Las operaciones de cambio de estructura y de visualización son tan distintas debido a que en la primera vamos a seguir trabajando con matrices, mientras que para una visualización distinta necesitaremos hacer uso de una reserva de más o menos espacio y una copia de lo que ya tengamos en nuestra imagen original.

Ejercicios

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

```
/**
* @brief Lectura de una imagen a color o escala de grises
* @param imagen, dirección en disco de la imagen a leer
* @param flagColor, numero que indica si la imagen pasada es a color o no
* @return Asigna la imagen leida a una variable
*/
Mat leeImagen(String imagen, int flagColor){
	Mat im;
	im = imread(imagen,1);

	if(flagColor == 1){
		cvtColor(im, im, CV_RGB2GRAY);
	}
	return im;
};
```

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

```
/**
* @brief Imprime una imagen por pantalla
* @param img, imagen que mostramos por pantalla
* @return Muestra por pontalla una ventana con la imagen y luego la borra
*/
void pintaImagen(Mat img){
	namedWindow("ventana",1);
	waitKey(1000);

	cout << "Ventana creada " << endl;
	imshow("ventana",img);
	waitKey(0);
	cout << "imagen mostrada " << endl;
	destroyWindow("ventana");

};
```

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)?

Si estamos ante imagenes de distinto tipo no podriamos mostrar todas las imágenes en una sola ventana, por lo que deberiamos o bien unificar los tipos o procurar pasas un conjunto de imagenes de mismo tipo.

```
/**
* @brief Muestra por pantalla varias imagenes
* @param imagenes, lista de imagenes que queremos mostrar
* @return Muestra varias imagenes en una ventana
*/
void pintaMI(vector <Mat> imagenes){
	/*
		Obtengo el tamaño de la ventana.
		- Ancho: suma de los anchos
		- Alto: máximo de las alturas
	*/

	int ancho = 0;
	int largo = 0;
	for(int i=0; i < imagenes.size();i++){
		ancho = ancho + imagenes[i].cols;
	};

	for(int i=0; i < imagenes.size();i++){
		int aux = imagenes[i].rows;
		if(aux > largo){largo = aux;};
	};

	//Creo un objeto Mat para incorporar todas las imagenes
	Mat ventana_mixta(largo, ancho, imagenes[0].type());

	int pivote = 0;

	for(int i=0; i < imagenes.size();i++){

	imagenes[i].copyTo(ventana_mixta(Rect(pivote,0,imagenes[i].cols,imagenes[i].rows)));
		pivote = pivote + imagenes[i].cols;
	};

	pintaImagen(ventana_mixta);
};
```

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

```
void modificaImagen(Mat imagen, vector <pixel> mascara, color valor){
	int coord_x, coord_y;
	for(int i=0;i < mascara.size(); i++){
		coord_x = mascara[i].x;
		coord_y = mascara[i].y;
		Point3_<uchar>* p = imagen.ptr<Point3_<uchar> >(coord_x,coord_y);
		p->x = valor.B; //B
		p->y = valor.G; //G
		p->z = valor.R; //R
	}

};
```