



Formation OpenCV

Philippe FOUBERT

OpenCV – Les fondamentaux



Les structures de données – Classe Mat

→ C'est **LA** classe à connaître !

NB: vous trouverez sur Internet de nombreux tutoriaux basés sur la classe « **IplImage** » issue de la librairie Intel « Image Processing Library ». Néanmoins, il s'agit d'une classe **obsolète** qui n'est plus maintenue.

→ La classe Mat représente une matrice ou une image

```
// Create 3 channels, 2 rows x 3 columns matrix and fill all values with 125  
cv::Mat M(2, 3, CV_8UC3, Scalar::all(125));
```

et permet un debug facile :

```
std::cout << M << std::endl;
```

3 /

Philippe FOUBERT – Septembre 2021



Les structures de données – Classe Mat

→ Comme nous l'avons vu, pour charger une image :

```
cv::Mat im = imread("path/to/image.jpg", IMREAD_COLOR);
```

→ Si « image.jpg » est une image couleur 48x24, cette ligne va créer une matrice de 3 canaux 24x48 de type CV_8U :



Remarque : par défaut, OpenCV stocke les pixels en **BGR**

4 /

Philippe FOUBERT – Septembre 2021



Pourquoi cet encodage en BGR ? (1/2)

- Pourquoi les chemins de fer des Etats-Unis ont-ils été construits avec un écartement de 4 pieds et 8 pouces et demi (1,435m) ?
 - Parce que les premières lignes de chemin de fer furent en fait construites par les mêmes ingénieurs qui construisirent les tramways, et que cet écartement était alors utilisé pour les tramways.
- Mais alors, pourquoi ont-ils utilisé cet écartement pour les tramways ?
 - Parce que les premières personnes qui construisaient les tramways étaient les mêmes qui construisaient les chariots et qu'ils ont utilisé les mêmes méthodes et les mêmes outils.
- OK, mais pourquoi les chariots avaient-ils un écartement de 4 pieds et 8 pouces et demi ?
 - Parce que partout en Europe et en Angleterre, les routes avaient déjà des ornières (traces bien définies dans le sol) espacées de cette façon et un espacement différent aurait causé la rupture de l'essieu du chariot en circulant sur ces routes.
- Et pourquoi ces routes présentaient-elles des ornières ainsi espacées ?
 - Parce que les premiers chariots étaient des chariots de guerre romains. Ils étaient tirés par deux chevaux. Ces chevaux galopèrent côte à côte et devaient être suffisamment espacés pour ne pas se gêner.

5 /

Philippe FOUBERT – Septembre 2021



Pourquoi cet encodage en BGR ? (2/2)

- La raison pour laquelle les premiers développeurs d'OpenCV ont choisi le format de couleur BGR, c'est parce qu'à l'époque, le format de couleur BGR était populaire parmi les fabricants et les fournisseurs de logiciels d'appareil-photo.
(sous Windows, quand vous spécifiez une couleur en utilisant COLORREF, le format utilisé est aussi le format BGR 0x00bbgrr)
- Ce choix permettait de stocker des palettes de couleurs dans la mémoire d'une manière facilement transmissible au composant VGA RAMDAC utilisé dans les cartes vidéo d'IBM (et compatible) VGA.
- BGR est un choix fait pour des raisons historiques (en d'autres termes, BGR est le cul de cheval dans OpenCV...).



Sculpture inox de François Levrat
« Poids du Passé »

6 /

Philippe FOUBERT – Septembre 2021



Les structures de données – Format de donnée

→ Pour spécifier le nombre de canaux et la nature des données :

CV_<bit_depth>(S|U|F)C<number_of_channels>, avec

S = signed integer

U = unsigned integer

F = float

→ Exemple :

CV_8UC3 = 8-bit unsigned matrix with three channels

CV_32FC2 = 32-bit float matrix with two channels

Les structures de données – Format de donnée

Type	Description	Nombre d'octets	Plage de valeurs
CV_8U	unsigned char	1	0...255
CV_8S	char	1	-128...128
CV_16U	unsigned short int	2	0...65535
CV_16S	short int	2	-32768...32767
CV_32U	-	-	N'existe pas
CV_32S	int	4	-2147483648..2147483647
CV_32F	float	4	-float_max...float_max
CV_64D	double	8	-double_max...double_max

Mat - Créer une matrice

```
// Create 2x3 matrix of uchar type, single channel
Mat a(row, col, CV_8U, Scalar(0));

// 2x3 matrix, float, 2 channels
// Set all elements of first channel to 1.5 and
// all elements of second channel to 2.5
Mat b(Size(col, row), CV_32FC2, Scalar(1.5, 2.5));

// Using Matlab-style initializers
Mat c = Mat::zeros(row, col, CV_64F);
Mat d = Mat::ones (row, col, CV_64F);
Mat e = Mat::eye  (row, col, CV_64F);

// Initialize with external data
float data[row][col] = { { 0.1f, 0.2f, 0.3f },
                          { 1.1f, 1.2f, 1.3f } };
Mat f(row, col, CV_32FC1, data);

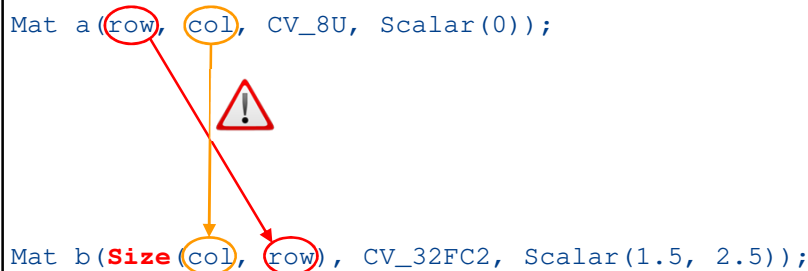
// Using comma-separated initializer
Mat g(Mat_<float>(row, col) << 0.1f, 0.2f, 0.3f,
                              1.1f, 1.2f, 1.3f);
```

With:
row=2
col=3

Mat - Créer une matrice (suite)

```
Mat a(row, col, CV_8U, Scalar(0));

Mat b(Size(col, row), CV_32FC2, Scalar(1.5, 2.5));
```



Les structures de données – Pixels

→ Pour spécifier un vecteur de 2, 3 ou 4 éléments :

Vec(2|3|4)(b|s|i|f|d), avec

b = uchar, s = short, i = int, f = float, d = double

→ **Scalar** : groupe de 4 nombres réels double précision

Comme Vec, ce type est généralement utilisé pour manipuler des valeurs de pixels :

Scalar a(1, 2, 3, 4);

Scalar b = Scalar::all(5);

Les structures de données – Coordonnées

→ Pour spécifier des coordonnées 2D :

Point_<T>

Point2(i|f|d), avec

i = int, f = float, d = double

→ Pour spécifier des coordonnées 3D :

Point3_<T>

Point3(i|f|d), avec

i = int, f = float, d = double