Kỹ thuật cơ bản trong xử lý ảnh

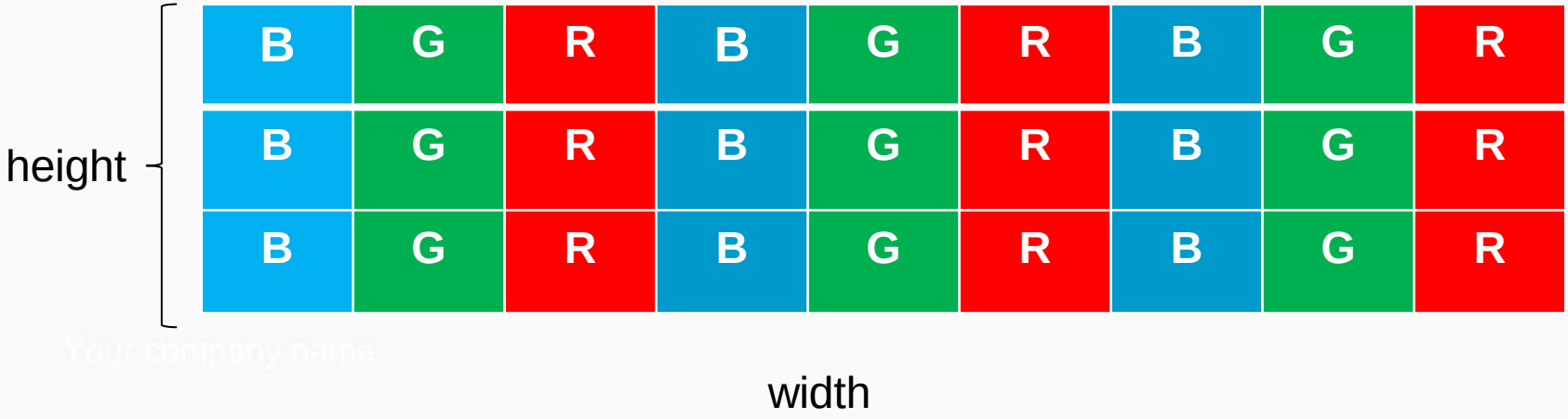
Phạm Minh Hoàng

Nội dung

- Kỹ thuật truy xuất điểm ảnh (pixel)
- Kỹ thuật dùng bảng tra (lookup table)
- Kỹ thuật truy xuất lân cận

Kỹ thuật truy xuất điểm ảnh

- Ảnh có cấu trúc như một mảng hai chiều nhưng thực tế được lưu bằng những vùng nhớ liên tiếp nhau



Kỹ thuật truy xuất điểm ảnh

- Cách 1: Để truy xuất vào từng pixel, sử dụng 2 vòng for lồng nhau và hàm at của OpenCV

```
for(int y = 0; y < height; y++)
{
    for(int x = 0; x < width; x++)
    {
        img.at<uchar>(x, y) = 255;//truy xuất pixel (x,y)
    }
}
```

Ưu điểm:

- Dễ hiểu
- Dễ viết

Nhược điểm:

- Chậm do truy xuất bằng hàm

Kỹ thuật truy xuất điểm ảnh

- Cách 2: Dùng con trỏ đầu mỗi dòng để truy xuất đến từng pixel trên dòng

```
int nCols = image.cols, nRows = image.rows;
//nChannels là số kênh màu
int nChannels = image.channels();

for(int y = 0; y < nRows; y++)
{
	//lấy con trỏ đầu mỗi dòng
	uchar* pRow = img.ptr<uchar>(y);
	for(int x = 0; x < nCols; x++, pRow += nChannels)
	{
		pRow[0]=...;//truy xuất pixel (x,y) channel thứ 0
		pRow[1]=...;//truy xuất pixel (x,y) channel thứ 1
		pRow[2]=...;//truy xuất pixel (x,y) channel thứ 2
	}
}
```

Kỹ thuật truy xuất điểm ảnh

- Cách 3: Dùng con trỏ data để quản lý vùng nhớ ảnh

```
int width = image.cols, height = image.rows;
//nChannels là số kênh màu
int nChannels = image.step[1];
//widthStep là khoảng cách tính theo byte giữa 2 pixel cùng cột trên 2 dòng kế tiếp
int widthStep = image.step[0];
//pData là con trỏ quản lý vùng nhớ ảnh
uchar* pData = (uchar*)image.data;
for(int y = 0; y < height; y++, pData += widthStep)
{
    //lấy con trỏ đầu mỗi dòng
    uchar* pRow = pData;
    for(int x = 0; x < width; x++, pRow += nChannels)
    {
        pRow[0]=...;//truy xuất pixel (x,y) channel thứ 0
        pRow[1]=...;//truy xuất pixel (x,y) channel thứ 1
        pRow[2]=...;//truy xuất pixel (x,y) channel thứ 2
    }
}
```

Kỹ thuật dùng bảng tra

- Bài toán: Tăng giảm độ sáng của ảnh theo công thức sau
- $I_{output}(x,y) = a*I_{input}(x,y) + b$
- Cài đặt:

```
for(int y = 0; y < height; y++)
    for(int x = 0; x < width; x++)
    {
        //tính giá trị độ xám val_src tại pixel (x,y) của ảnh input
        uchar val_src = …
        //tính giá trị độ xám val_dst của ảnh output theo công thức
        uchar val_dst = a*val_src + b;
        //gán val_dst vào vùng nhớ của pixel (x,y) của ảnh output
    }
```

Kỹ thuật dùng bảng tra

- Câu lệnh **uchar val_dst = a*val_src + b;** phải thực hiện đủ width*height lần => rất tốn kém trong khi miền giá trị của val_src và val_dst chỉ từ [0,255]
- Giải pháp: dùng mảng gồm 256 phần tử làm bảng tra

```
uchar lookup[256];
for(int i = 0; i < 256; i++)
    lookup[i] = a*i + b;
```

Kỹ thuật dùng bảng tra

- Cài đặt theo bảng tra:

```
for(int y = 0; y < height; y++)
    for(int x = 0; x < width; x++)
    {
        //tính giá trị độ xám val_src tại pixel (x,y) của ảnh input
        uchar val_src = …
        //dùng bảng tra để tính val_dst
        uchar val_dst = lookup[(int)val_src];
        //gán val_dst vào vùng nhớ của pixel (x,y) của ảnh output
    }
```

Ưu điểm:Thay vì phải thực hiện width*height phép tính chỉ cần làm 256 phép tính

Nhược điểm: chỉ áp dụng đối với các phép xử lý trên một điểm ảnh riêng rẽ

Kỹ thuật truy xuất lân cận

- Trong xử lý ảnh có nhu cầu từ 1 pixel truy xuất sang những pixel xung quanh
- Giả sử, p là con trỏ đến pixel (x,y), lân cận 8 của p được xác định như sau

p[-widthStep - 1]	**p[-widthStep]**	**p[-widthStep + 1]**
p[-1]	**p[0]**	**p[1]**
p[widthStep - 1]	**p[widthStep]**	**p[widthStep + 1]**

Kỹ thuật truy xuất lân cận

- Lưu các chỉ số lân cận vào mảng offset

```
//kWidth, kHeight là kích thước vùng lân cận
int kHalfWidth = kWidth >> 1;
int kHalfHeight = kHeight >> 1;
vector<int> offsets;
for(int y = -kHalfHeight; y <= kHalfHeight; y++)
      for(int x = -kHalfWidth; x <= kHalfWidth; x++)
            offsets.push_back(y*widthStep + x);

for(int y = 0; y < height; y++)
      for(int x = 0; x < width; x++)
      {
            //p là con trỏ đến pixel (x,y)
            for(int k = 0; k < offsets.size(); x++)
                  p[offsets[k]] =....//truy xuất lân cận của p
      }
```