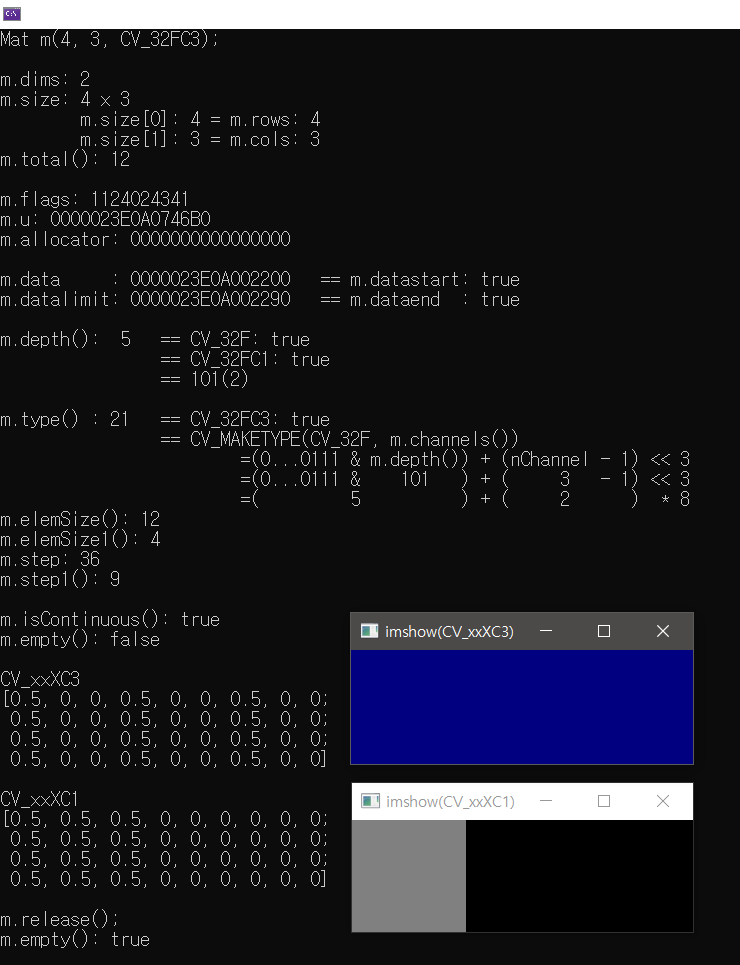
Computer Vision 과제

Mat m(4, 3, CV32FC3) 속성값 출력

2022-09-13

빅데이터/웹공학 트랙  
1891179 김현학

Result



Code

#include <cstdio>

#include <iostream>

#include <opencv2/opencv.hpp>

using namespace cv;

using namespace std;

int main()

{

// Mat (int rows, int cols, int type)

// type: CV\_<bit\_depth>{S|U|F}C (C는 채널 수)

printf("Mat m(4, 3, CV\_32FC3);\n\n");

Mat m(4, 3, CV\_32FC3); // 32비트 부동 소수점을 갖는 3개 채널 행렬: 화소 하나당 32\*3비트

// m = Mat::zeros/ones/eye(5, 4, CV\_32FC3); // Re-assign-able

printf("%s: %d\n", "m.dims", m.dims);

cout << "m.size: " << m.size << endl;

printf("\t%s: %d", "m.size[0]", m.size[0]); printf(" = %s: %d\n", "m.rows", m.rows);

printf("\t%s: %d", "m.size[1]", m.size[1]); printf(" = %s: %d\n", "m.cols", m.cols);

printf("%s: %zu\n", "m.total()", m.total());

puts("");

printf("%s: %d\n", "m.flags", m.flags);

cout << "m.u: " << m.u << endl;

cout << "m.allocator: " << m.allocator << endl;

puts("");

printf("%-11s: %p", "m.data", m.data);

cout << "\t== m.datastart: " << boolalpha << (m.data == m.datastart) << endl;

printf("%11s: %p", "m.datalimit", m.datalimit);

cout << "\t== m.dataend : " << boolalpha << (m.datalimit == m.dataend) << endl;

puts("");

printf("%s: %2d", "m.depth()", m.depth());

cout << "\t== CV\_32F: " << boolalpha << (m.depth() == CV\_32F) << endl;

cout << "\t\t== CV\_32FC1: " << boolalpha << (m.depth() == CV\_32FC1) << endl;

printf("\t\t== %s(2)\n", "101");

puts("");

printf("%s: %2d", "m.type() ", m.type());

cout << "\t== CV\_32FC3: " << boolalpha << (m.type() == CV\_32FC3) << endl;

printf("\t\t== CV\_MAKETYPE(%s, %s)\n", "CV\_32F", "m.channels()");

printf("\t\t\t=(0...0111 & m.depth()) + (nChannel - 1) << 3\n");

printf("\t\t\t=(0...0111 & 101 ) + ( %3u - 1) << 3\n", m.channels());

printf("\t\t\t=( 5 ) + ( 2 ) \* 8");

puts("");

cout << "m.elemSize(): " << m.elemSize() << endl; // 4Bytes(=32bits) \* m.channel(=3)

cout << "m.elemSize1(): " << m.elemSize1() << endl; // Mat::elemSize() / m.channel

cout << "m.step: " << m.step << endl; // Mat::cols \* Mat::elemSize()

cout << "m.step1(): " << m.step1() << endl; // Mat::step / Mat::elemSize1() == Mat::cols \* m.channel

puts("");

cout << "m.isContinuous(): " << m.isContinuous() << endl;

cout << "m.empty(): " << m.empty() << endl;

// image processing

Mat m2, m1 = ((m & 0) + 0.5); cout << "\nCV\_xxXC3\n" << (m2 = m1) << endl;

vector<Mat> a; split(m2, a); hconcat(a, m2); cout << "\nCV\_xxXC1\n" << m2 << endl;

cout << "\nm.release();\n"; m.release();

cout << "m.empty(): " << m.empty() << endl << endl;

namedWindow("imshow(CV\_xxXC3)", WindowFlags::WINDOW\_FREERATIO);

namedWindow("imshow(CV\_xxXC1)", WindowFlags::WINDOW\_FREERATIO);

imshow("imshow(CV\_xxXC3)", m1);

imshow("imshow(CV\_xxXC1)", m2);

waitKey(0);

return 0;

}