Hallgatói segédlet

OpenCV Mat alapok

Létrehozás, inicializálás

Mat osztály inicializálásra mutat példákat a következő kódrészlet:

```
int cols = 10;
int rows = 10;
int type = CV_8UC1; // one channel, 8bit
int value = 5;

cv::Mat name; //empty instanced Mat object
cv::Mat name(rows,cols,type)
cv::Mat name(rows,cols,type,cv::Scalar(value))
```

A legelső esetben egy üres Mat osztályt hoztunk létre. A második esetben már megadtuk a méreteit és a típusát is, a harmadik esetében pedig konkrét értékekre be is állítjuk az összes elemet.

Lehetőség van arra is, hogy a példányosítást követően változtatjuk meg a Mat méretét:

```
int cols = 50;
int rows = 20;
int type = CV_8UC3; //three channel, 8bit

cv::Mat name; //empty instanced Mat object
name.create(rows,cols,type);
```

Műveletek

Az összes érték beállítása:

```
int value = 10;
cv::Mat name; //empty instanced Mat object
name.setTo(value);
```

Adott Mat osztály sor és oszlop elérése (sorrendben):

```
cv::Mat M(7, 7, CV_8UC1);
    //csak referenciakent tesszuk egyenlove a teljes matrixszal
    cv::Mat M2 = M;
    //adott sor es oszlop elerese ertekmasolas nelkul
    cv::Mat row = M.row(3);
    cv::Mat col = M.col(2);

    //adott matrix, sor, es oszlop ertekmasolassal
    cv::Mat M3 = M.clone();
    cv::Mat row2 = M.row(2).clone();
    cv::Mat col4 = M.row(4).clone();
```

A létrehozott Mat osztály paramétereinek elérése, kiíratása:

```
cv::Mat Image;

std::cout << "Image Properties: " << std::endl
        << "rows: " << Image.rows << std::endl
        << "cols: " << Image.cols << std::endl
        << "channels:" << Image.channels() << std::endl
        << "type: " << Image.type() << std::endl;

std::cout << Image;
```

A létrehozott Mat osztály elemeinek elérése:

```
cv::Mat M = cv::Mat::eye(10, 10, CV_8UC1);

    for (int i = 0; i < M.rows; ++i) {
        for (int j = 0; j < M.cols; ++j) {
            std::cout << M.at<uchar>(i, j) << " ";
        }
        std::cout << std::endl;
    }
```

Az at metódussal lehet elérni külön-külön a Mat osztály elemeit. Meg kell adni, hogy milyen típusú elemek találhatóak meg az osztályban (cast), és ezt követően pedig a sor és oszlop értékeket.

Mat objektum szátdarabolása és egyesítése:

```
cv::Mat input = cv::imread("image.jpg",1);
cv::Mat splitted[3], output;
cv::split(input, splitted);
cv::merge(splitted,3,output);
```

Képekkel kapcsolatos műveletek

Kép beolvasása:

```
int megnyitas modja = 1; // -1: modositatlan ertekek, 0: fekete-fehér mód, 1:színes
mód
cv::Mat Image = cv::imread("kep_eleresi_utvonala",megnyitas_modja);
```

Kép kiírása:

```
std::vector<int> formatParameters_png,formatParameters_jpg;
    formatParameters_png.push_back(cv::IMWRITE_PNG_COMPRESSION);
    formatParameters_png.push_back(5);

    formatParameters_jpg.push_back(cv::IMWRITE_JPEG_QUALITY);
    formatParameters_jpg.push_back(90);

    cv::imwrite("out.png", I, formatParameters_png);
    cv::imwrite("out.jpg", I2, formatParameters_jpg);
```

Kiíráskor szükséges egy vector osztályban felsorolni a kimeneti formátum jellemzőit.

Színterek

RGB

Három szín alapján keveri ki a végső színt. A három alapszín inexekkel együtt: Kék (0), Zöld(1), Piros(2)

HSV

Itt is három csatorna van, csak a jelentésük más:

- HUE: színérték
- SATURATION: mennyi a szürke aránya a színkomponensben
- VALUE: világosságérték

Konverziók

```
cv::cvtColor(I, gray, cv::COLOR_RGB2GRAY);
cv::cvtColor(I, hsv, cv::COLOR_RGB2HSV);
cv::cvtColor(I, hsv, cv::COLOR_HSV2RGB);
```

Képjavító módszerek

Szűrők,eljárások

Életlenítő szűrők (box,gauss,median):

```
cv::Mat input,output;
int value = 3; //always odd
int sigma_value = 2
cv::boxFilter(input, output, -1, cv::Size(value, value));
cv::GaussianBlur(input, output, cv::Size(value, value),sigma_value);
cv::medianBlur(input, output, value);
```

A box átlagértéket, a gauss a görbe által meghatározott értéket, a medián középértéket számol.

Hisztogram-kiegyenlítés:

```
cv::equalizeHist(input, output);
```

Gyenge kontrasztú képek esetében alkalmazzuk.

Élesítés:

```
cv::Mat image = cv::imread("kacsa.jpg", 0);
cv::Mat blurred, unsharped;

cv::GaussianBlur(image, blurred, cv::Size(sizeofKernel, sizeofKernel), 1);
cv::addWeighted(image, 1.5, blurred, -0.5, 0,unsharped);
```

Itt a lényeg, hogy először tetszőleges szűrővel kiszámoljuk az eredeti kép homályosabb verzióját, majd kivonjuk az eredeti képből, ezzel megkapjuk az élesebb változatot. Az

összegzés során a súlyok előjelei ellentétesek, és nagyon fontos, hogy az előjeles összegük egyenlő legyen **1-gyel**.

Küszöbölés

Sima globális küszöbölés:

```
cv::threshold(Img, thresh, thresh_value, 255, cv::THRESH_BINARY);
cv::threshold(Img, thresh, thresh_value, 255, cv::THRESH_BINARY_INV);
```

Globális küszöbölés algoritmussal (Otsu):

```
cv::threshold(Img, thresh, 0,255,cv::THRESH_BINARY | cv::THRESH_OTSU);
```

Adaptív küszöbölés:

```
int blockSize = 5;
cv::adaptiveThreshold(Img, thresh, 255, cv::ADAPTIVE_THRESH_MEAN_C,
cv::THRESH_BINARY,blockSize,2);
```

Morfológia

Alapműveletek

Dilatáció, erózió:

```
cv::Mat Img = cv::imread("morp_test.png", 0);
cv::Mat kernel = cv::Mat::ones(cv::Size(5, 5), CV_8UC1);
cv::Mat dilated,eroded;

cv::dilate(Img, dilated, kernel);
cv::erode(Img, eroded, kernel);
```

Nyitás:

```
cv::Mat Img = cv::imread("morp_test_open.png", 0);
cv::Mat kernel = cv::Mat::ones(cv::Size(3, 3), CV_8UC1);
cv::Mat eroded,opened;

cv::erode(Img, eroded, kernel);
cv::dilate(eroded, opened, kernel);
```

Zárás:

```
cv::Mat Img = cv::imread("morp_test_closed.png", 0);
cv::Mat kernel = cv::Mat::ones(cv::Size(7, 7), CV_8UC1);
cv::Mat dilated, closed;

cv::dilate(Img, dilated, kernel);
cv::erode(dilated, closed, kernel);
```

Struktúrális elem(kernel)

Deklarációk:

```
cv::Mat kernel = cv::Mat::ones(cv::Size(7, 7), CV_8UC1); // sima 7x7es négyzet alakú
cv::Mat kernel = cv::getStructuringElement(1, cv::Size(sizeofKernel,sizeofKernel));
```

Lehet használni a beépített metódust is, az első paraméter megadja a struktúrális elem típusát:

- 0: háromszög, cv::MORPH_RECT
- 1: kereszt, cv::MORPH_CROSS
- 2: ellipszis, cv::MORPH_ELLIPSE

Összetettebb műveletek

Kontúr:

```
cv::Mat image = cv::imread("kacsa.jpg", 0);
cv::Mat dilated, eroded, contures;
cv::Mat kernel = cv::getStructuringElement(1, cv::Size(sizeofKernel,sizeofKernel));

cv::dilate(image, dilated, kernel);
cv::erode(image, eroded, kernel);
contures = dilated - eroded;
```

Tophat (csúcsok detektálása):

```
sizeofKernel = 10;
cv::Mat image = cv::imread("kacsa.jpg", 0);
cv::Mat eroded, opened,tophat;
cv::Mat kernel = cv::getStructuringElement(1, cv::Size(sizeofKernel, sizeofKernel));

cv::erode(image, eroded, kernel);
cv::dilate(eroded, opened, kernel);
tophat = image - opened;
```

Blackhat (völgyrészek detektálása):

```
cv::Mat image = cv::imread("kacsa.jpg", 0);
cv::Mat dilated,closed,tophat, blackhat;
cv::Mat kernel = cv::getStructuringElement(1, cv::Size(sizeofKernel, sizeofKernel));

cv::dilate(image, dilated, kernel);
cv::erode(dilated, closed, kernel);
blackhat = closed - image;
```