# **Open-source knižnica OpenCV**

**Knižnica pre manipuláciu s počítačovou grafikou**

OpenCV (Open Source Computer Vision, <https://opencv.org/>) je knižnica pre manipuláciu s obrázkami a real-time videom. Je napísaná v C++, ktorý je aj jej primárnym interface jazykom.

OpenCV transformuje obrázky do matice Mat, ktorá sa skladá z 2 častí:

* hlavičku (header) – obsahuje informácie o veľkosti (počet riadkov a stĺpcov), formát, v akom je obrázok uložený (napr. RGB alebo HSV), počet bitov pre každú hodnotu, či je signed, koľko hodnôt je na jeden pixel,…
* samotnú maticu (resp. pointer na ňu), ktorá predstavuje 2D pole s hodnotami, ktoré reprezentujú farbu každého bodu

Aby sa optimalizovala práca s veľkým objemom dát, Mat reálne obsahuje len header a pointer na samotné dáta, čím umožňuje zdieľanie rovnakých dát medzi metódami. Pointer môže ukazovať len na časť veľkého obrázka.

**cv::Mat::Mat ( int rows, int cols, int type )**

*Príklad inicializácie 2x2 matice s 8 unsigned bitmi na hodnotu a 3 hodnotami na pixel:*

Mat M(2,2, [CV\_8UC3](https://docs.opencv.org/master/d1/d1b/group__core__hal__interface.html#ga88c4cd9de76f678f33928ef1e3f96047), [Scalar](https://docs.opencv.org/master/dc/d84/group__core__basic.html#ga599fe92e910c027be274233eccad7beb)(0,0,255));

<https://docs.opencv.org/master/d3/d63/classcv_1_1Mat.html>

## **Manipulácia s videom**

#include "videoio.hpp"

OpenCV dokáže spracovávať nielen statické obrázky, ale aj real-time video (z webkamery) alebo video načítané z disku. Funkcionality na prácu s videom sú v triede: cv::VideoCapture, ktorá vychádza z FFmpeg open-source knižnice. Samotné video je spracovávané ako jednotlivé obrázky, frame-y, pomocou základných metód knižnice.

Načítanie videa sa spúšťa prostredníctvom cv::VideoCapture::VideoCapture alebo cv::VideoCapture::open.

// Opens a video file or a capturing device or an IP video stream for video capturing with API Preference

**cv::VideoCapture::VideoCapture (const String &filename, int apiPreference=CAP\_ANY)**

vstupné parametre metódy:

* filename – môže byť názov video súboru, sady obrázkov alebo URL video stream-u
* apiPreference – definuje preferované API pre načítavanie na backende

// Opens a camera for video capturing

**cv::VideoCapture:: VideoCapture (int index, int apiPreference=CAP\_ANY)**

vstupné parametre metódy:

* index – udáva ID zariadenia, ktoré kamere pridelil operačný systém; pre otvorenie defaultnej kamery je potrebné uviesť hodnotu 0
* apiPreference – definuje preferované API pre načítavanie na backende

// Opens a video file or a capturing device or an IP video stream for video capturing.

**virtual bool cv::VideoCapture::open (const String &filename, int apiPreference=CAP\_ANY)**

**virtual bool cv::VideoCapture::open (int index, int apiPreference=CAP\_ANY)**

Keďže aplikácia bude pracovať s video streamom z webkamery, bude použitý variant metódy s index. Minimálne rozdiely v metódach však umožnia testovanie aplikácie v priebehu vývoja na existujúcich súboroch.

Ukončenie spracovania videa môže byť uskutočnené automaticky, keď je zavolaný destructor objektu, alebo je možné ho ukončiť aj pomocou metódy

**virtual void cv::VideoCapture::release ( )**

Každý frame z videa je spracovávaný ako samostatný objekt, ktorý je vkladaný do jednej matice Mat. Jednotlivé obrázky sú z videa načítavané pomocou metódy **cv::VideoCapture::read** alebo pomocou overload >> operátora.

*Príklad použitia pomocou metódy read:*

for (;;)

{

// wait for a new frame from camera and store it into 'frame'

VideoCapture.read(frame);

// check if we succeeded

if (frame.empty()) {

cerr << "ERROR! blank frame grabbed\n";

break;

}

// show live and wait for a key with timeout long enough to show images

imshow("Live", frame);

if ([waitKey](https://docs.opencv.org/master/d7/dfc/group__highgui.html#ga5628525ad33f52eab17feebcfba38bd7)(5) >= 0)

break;

}

*Príklad použitia cez >> operator:*

Mat frameReference, frameUnderTest;

captRefrnc >> frameReference;

captUndTst >> frameUnderTest;

if( frameReference.empty() || frameUnderTest.empty())

{

// exit the program

}

V oboch príkladoch je zároveň ukázané, ako je možné testovať, či video už neskončilo (frame / frameReference / frameUnderTest je prázdny). V takom prípade je potrebné program ukončiť.

Metóda read vykonáva 2 kroky: prevezme frame a dekóduje ho. Tieto kroky je možné vykonať aj jednotlivo cez cv::VideoCapture::grab a cv::VideoCapture::retrieve.

Knižnica umožňuje načítavať aj ďalšie informácie obsiahnuté vo videu a to prostredníctvom metódy get().

**virtual double cv::VideoCapture::get ( int propId ) const**

vstupné parametre metódy:

* propid – určuje kód atribútu, ktorú metóda prečíta

výstupom metódy je hodnota vyžiadaného atribútu, alebo hodnota 0, ak atribút nie je podporovaný v backendovom API

*Príklad použitia metódy get() pre získanie veľkosti framov a počet framov vo videu:*

[Size](https://docs.opencv.org/master/dc/d84/group__core__basic.html#ga346f563897249351a34549137c8532a0) refS = [Size](https://docs.opencv.org/master/dc/d84/group__core__basic.html#ga346f563897249351a34549137c8532a0)((int) captRefrnc.get(CAP\_PROP\_FRAME\_WIDTH),

(int) captRefrnc.get(CAP\_PROP\_FRAME\_HEIGHT)),

cout << "Reference frame resolution: Width=" << refS.[width](https://docs.opencv.org/master/d6/d50/classcv_1_1Size__.html#abfe0367b32c407ddccf5ddf92667c73d) << " Height=" << refS.height

<< " of nr#: " << captRefrnc.get(CAP\_PROP\_FRAME\_COUNT) << endl;

Podobným spôsobom je možné kontrolovať tieto parametre prostredníctvom funkcie set().

**virtual bool cv::VideoCapture::set ( int propId, double value )**

vstupné parametre metódy:

* propid – určuje kód atribútu, ktorý metóda zmení
* value – hodnota, na ktorú bude atribút nastavený

výstupom metódy je true, ak je atribút podporovaný backend-ovým API; to ale nemusí znamenať, že hodnota bola zmenená.

*Príklad použitia metódy set() na nájdenie času alebo framu vo videu:*

captRefrnc.set(CAP\_PROP\_POS\_MSEC, 1.2); // go to the 1.2 second in the video

captRefrnc.set(CAP\_PROP\_POS\_FRAMES, 10); // go to the 10th frame of the video

// now a read operation would read the frame at the set position

Čítanie a nastavovanie atribútov prechádza cez viacero úrovní, na ktorých môže dôjsť k nesprávnej interpretácii parametra vzhľadom na rôzne interpretácie, najmä na úrovni API a driver zariadenia:

VideoCapture -> API Backend -> Operating System -> Device Driver -> Device Hardware`

viac informácií k VIdeoCapture: <https://docs.opencv.org/master/d8/dfe/classcv_1_1VideoCapture.html>

atribúty pre get() a set(): <https://docs.opencv.org/master/d4/d15/group__videoio__flags__base.html#gaeb8dd9c89c10a5c63c139bf7c4f5704d>

dodatočné atribúty pre get() a set(): <https://docs.opencv.org/master/dc/dfc/group__videoio__flags__others.html#ggaaaeecdd8a279c71a7ebf8836b6c52feaa0b6409999490a804d1a06df43105e5cb>

## **Komunikácia s GUI**

#include "highgui.hpp"

Modul highgui (High-level GUI and Media I/O) umožňuje základú komunikáciu s GUI. Umožňuje zobrazovanie obrázkov aj framov z videa buď v pôvodnej alebo upravenej verzii (napr. po zvýrazení určitého objektu alebo jeho stredu).

void cv::imshow (const String &winname, InputArray mat)

vstupné parametre metódy:

* winname – názov okna, v ktorom sa obrázok zobrazí
* mat – obrázok v Mat formáte, ktorý bude zobrazný

<https://docs.opencv.org/master/d7/dfc/group__highgui.html>

## **Spracovanie obrázkov**

#include "imgproc.hpp"

OpenCV obsahuje veľké množstvo metód na spracovanie obrázkov. Pre potreby aplikácie je možné využiť aj štrukturálnu analýzu a identifikáciu objektov.

V tomto module existuje viacero funkcií, ktoré dokážu identifikovať objekty rôznych tvarov (napr. funkcia moments vie identifikovať polygonálne objekty do 3. rádu). fitEllipse nájde elipsu obkolesujúcu 2D obejkt.

cv::moments (InputArray array, bool binaryImage=false)

cv::fitEllipse ( InputArray points )

Tieto metódy by mohli byť využité pre určenie kyvadla vo videu ako alternatíva k identifikácii cez farbu určenú užívateľom.

Pred takýmto spracovaním je potrebné obrázok modifikovať (zmena na grayscale a blur alebo grayscale a treshhold) na lepšiu identifikáciu objektov.

*Príklad identifikácie kruhu a jeho stredu na obrázku:*

Mat thr, gray, src;

// convert image to grayscale

cvtColor( src, gray, COLOR\_BGR2GRAY );

// convert grayscale to binary image

threshold( gray, thr, 100,255,THRESH\_BINARY );

// find moments of the image

Moments m = moments(thr,true);

Point p(m.m10/m.m00, m.m01/m.m00);

// coordinates of centroid

cout<< Mat(p)<< endl;

// show the image with a point mark at the centroid

circle(src, p, 5, Scalar(128,0,0), -1);

imshow("Image with center",src);

waitKey(0);



*Dalšie zdroje:*

<https://docs.opencv.org/2.4/doc/tutorials/imgproc/shapedescriptors/moments/moments.html>

<https://www.learnopencv.com/find-center-of-blob-centroid-using-opencv-cpp-python/>

<https://www.pyimagesearch.com/2016/02/01/opencv-center-of-contour/>

<https://docs.opencv.org/master/d3/dc0/group__imgproc__shape.html>

Alternatívne metódy na identifikáciu kyvadla – bude potrebné otestovať, či nejde o výkonovo náročné spracovania, ktoré nebudú použiteľné v real-time video streame:

**detekcia kruhov a ich stredov:** HoughCircles()

<https://docs.opencv.org/master/d4/d70/tutorial_hough_circle.html>

**background subtraction:** cv::BackgroundSubtractor

<https://docs.opencv.org/master/d1/dc5/tutorial_background_subtraction.html>

## **Použitie knižnice v aplikácii**

**Prípady použitia knižnice openCV v aplikácii:**

* identifikácia farby pre určenie kyvadla z obrázka a súradníc, kam užívateľ klikol myšou
* prístup ku kamere ako zariadeniu pripojenému k počítaču
* spracovanie streamu videa
* nastavenie atribútov videozáznamu
* identifikácia kyvadla na obrázku podľa nastavenej farby alebo podľa kontrastu
* určenie stredu kyvadla
* označenie kyvadla kontrastnou farbou