Basics of openCV

Sung Soo Hwang

Basic data structures in openCV

- Mat
  - The basic data structure used in openCV
  - Declaration
    - Mat (int rows, int cols, int type)
    - Mat (Size size, int type)
    - Mat (const Mat & m)
    - Mat (Size size, int type, const Scalar& s)

Basic data structures in openCV

- Pixel type
  - CV_8U: 8-bit unsigned integer: uchar (0~255)
  - CV_8S: 8-bit signed integer: schar (-128~127)
  - CV_16U: 16-bit unsigned integer: ushort (0~65535)
  - CV_16S: 16-bit signed integer: short (-32768~32767)
  - CV_32S: 32-bit signed integer:
    int (-2147483648~2147483647)
  - CV_32F: 32-bit floating-point number:
    float (-FLT_MAX~FLT_MAX, INF, NAN)
  - CV_64F: 64-bit floating-point number:
    double (-DBL_MAX~ DBL_MAX, INF, NAN)
  - Multi-channel array:
    CV_8UC3, CV_8U(3), CV_64FC4, CV_64FC(4)

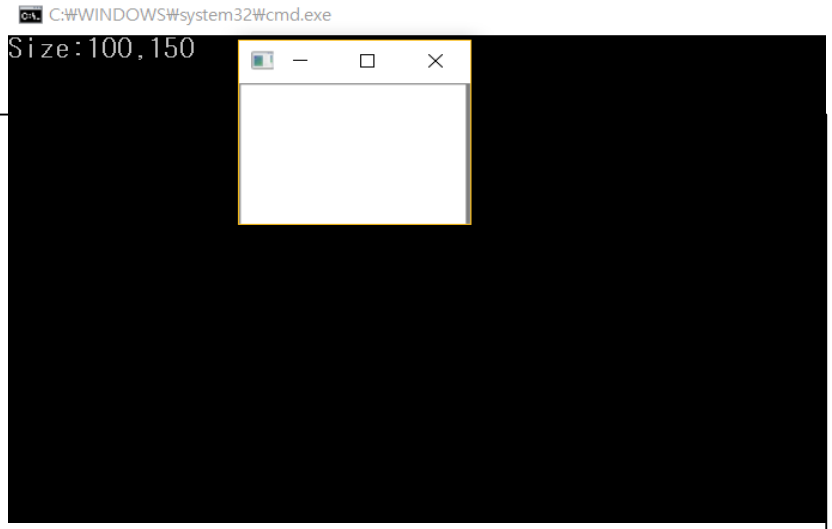
Basic data structures in openCV

- Example

```
Mat mtx(3, 3, CV_32F);
// make a 3x3 floating-point matrix

Mat cmtx(10, 1, CV_64FC2);
 // make a 10x1 2-channel floating-point matrix
(10-element complex vector)

Mat img(1080, 1920, CV_8UC3);
 // make a 3-channel (color) image of 1920 columns and 1080 rows.

Mat img(Size(1920, 1080), CV_8UC3);
 // make a 3-channel (color) image of 1920 columns and 1080 rows.
```

Basic data structures in openCV

- Example




```
#include "cv.hpp"
#include <iostream>

using namespace cv;
using namespace std;

int main()
{
        int w = 150, h = 100;
        Mat image(h, w, CV_8UC1, Scalar(255));
        cout << "Size: " << image.size().height << "," << image.size().width << endl;
        imshow("image", image);

        waitKey(0);
        return 0;
}
```

  - For a multi-channel image use Scalar function as
    - Scalar(255,0,0)

Read an image in openCV

- Read an image
  - Mat imread(const string& filename, int flags=1)
    - Flag value as 1:read image as color image
    - Flag value as 0:read image as gray scale image

```
int main() {
    Mat gray_image, color_image;

    // 0 on the 2nd parameter means read img in grayscale
    gray_image = imread("lena.png", 0);

    // blank 2nd parameter means 1, which means read img in colors
    color_image = imread("lena.png");

    imshow("gray image", gray_image);
    imshow("color image", color_image);

    waitKey(0);
    return 0;
}
```

Read a video in openCV

- Read a video from a file
  - Example 1

```
int main() {
    Mat frame;
    VideoCapture cap;

    // check if file exists. if none program ends
    if (cap.open("background.mp4") == 0) {
        cout << "no such file!" << endl;
        waitKey(0);
    }

    while (1) {
        cap >> frame;
        if (frame.empty()) {
            cout << "end of video" << endl;
            break;
        }
        imshow("video", frame);
        waitKey(33);
    }
}
```

Read a video in openCV

- Read a video from a webcam
  - Example

```
int main() {
    Mat frame;
    // capture from webcam
    // whose device number=0
    VideoCapture cap(0);

    while (1) {
        cap >> frame;

        imshow("web cam", frame);
        waitKey(16);
    }
}
```

Read a video in openCV

- Read a video
  - Use VideoCapture Class
  - Methods in VideoCapture Class

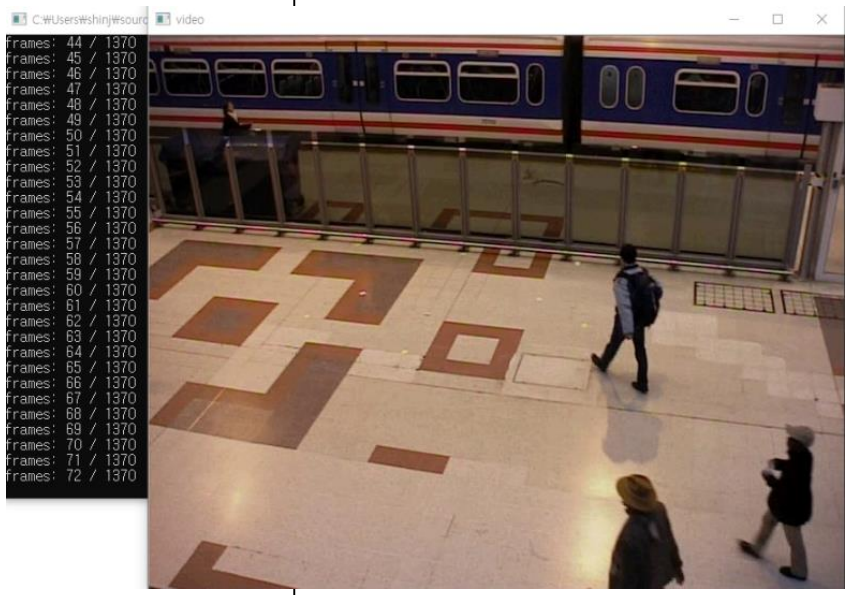
	VideoCapture ()
	VideoCapture (const **String** &filename)
	VideoCapture (const **String** &filename, int apiPreference)
	VideoCapture (int index)
virtual	**~VideoCapture** ()
virtual double	**get** (int propId) const Returns the specified **VideoCapture** property. More...
virtual bool	**grab** () Grabs the next frame from video file or capturing device. More...
virtual bool	**isOpened** () const Returns true if video capturing has been initialized already. More...
virtual bool	**open** (const **String** &filename) Open video file or a capturing device for video capturing. More...
virtual bool	**open** (int index)
virtual bool	**open** (const **String** &filename, int apiPreference)
virtual **VideoCapture** &	**operator>>** (**Mat** &image)
virtual **VideoCapture** &	**operator>>** (**UMat** &image)
virtual bool	**read** (**OutputArray** image) Grabs, decodes and returns the next video frame. More...
virtual void	**release** () Closes video file or capturing device. More...
virtual bool	**retrieve** (**OutputArray** image, int flag=0) Decodes and returns the grabbed video frame. More...
virtual bool	**set** (int propId, double value) Sets a property in the **VideoCapture**. More...

Read a video in openCV

- VideoCapture::get(int propId) – propId
  - **CAP_PROP_POS_MSEC** Current position of the video file in milliseconds or video capture timestamp.
  - **CAP_PROP_POS_FRAMES** 0-based index of the frame to be decoded/captured next.
  - **CAP_PROP_POS_AVI_RATIO** Relative position of the video file: 0 - start of the film, 1 - end of the film.
  - **CAP_PROP_FRAME_WIDTH** Width of the frames in the video stream.
  - **CAP_PROP_FRAME_HEIGHT** Height of the frames in the video stream.
  - **CAP_PROP_FPS** Frame rate.
  - **CAP_PROP_FOURCC** 4-character code of codec.
  - **CAP_PROP_FRAME_COUNT** Number of frames in the video file.
  - **CAP_PROP_FORMAT** Format of the Mat objects returned by retrieve() .
  - **CAP_PROP_MODE** Backend-specific value indicating the current capture mode.
  - **CAP_PROP_BRIGHTNESS** Brightness of the image (only for cameras).
  - **CAP_PROP_CONTRAST** Contrast of the image (only for cameras).
  - **CAP_PROP_SATURATION** Saturation of the image (only for cameras).
  - **CAP_PROP_HUE** Hue of the image (only for cameras).
  - **CAP_PROP_GAIN** Gain of the image (only for cameras).
  - **CAP_PROP_EXPOSURE** Exposure (only for cameras).
  - **CAP_PROP_CONVERT_RGB** Boolean flags indicating whether images should be converted to RGB.
  - **CAP_PROP_WHITE_BALANCE** Currently not supported
  - **CAP_PROP_RECTIFICATION** Rectification flag for stereo cameras (note: only supported by DC1394 v 2.x backend currently)

Video Capture example

```
int main() {
    Mat frame;
    VideoCapture cap;

    if (cap.open("background.mp4") == 0) return -1;
    double fps = cap.get(CAP_PROP_FPS);
    double time_in_msec = 0; int curr_frame = 0;
    int total_frames = cap.get(CAP_PROP_FRAME_COUNT);

    // video stops after 3 sec
    while (time_in_msec < 3000) {
        cap >> frame;
        if (frame.empty()) break;
        time_in_msec = cap.get(CAP_PROP_POS_MSEC);
        curr_frame = cap.get(CAP_PROP_POS_FRAMES);
        // printing current frames over total frames
        cout << "frames: " << curr_frame << " / " << total_frames << endl;
        imshow("video", frame);

        // calculating the right delay from given fps
        waitKey(1000 / fps);
    }
    waitKey(0);
    return 0;
}
```



Display an image/video in openCV

- Display an image
  - Example

```
int main() {
    Mat img;
    img = imread("lena.png", 1);
    imshow("Window", img);
    waitKey(0);
}
```

- Display a video
  - waitKey? int waitKey(int delay=0)
    - Delay in milliseconds.
    - 0 is the special value that means “forever”

```
int main() {
    Mat frame;
    VideoCapture cap("background.mp4");
    while (1) {
        cap >> frame;
        imshow("Window", frame);
        waitKey(30);
    }
}
```