⑤ CvAux { Experimental algorithms }

(2)

Introduction to OpenCV

* Include Files

⇒ The main header files of interest in ../opencv2/opencv.hpp.

↳ This includs all the relevant header files.

⇒ You can use the include file opencv.hpp to include any or every possible OpenCV function.

↳ But it will slow down compile time.

↳ For fast compilation, use Specific header files.

⇒ The include files are located on disk under the /modules directory.

* First Program - Display a Picture

```
cv::Mat img = cv::imread("Path to file", -1);
```

OpenCV functions live within the namespace called cv

Data Structure to hold an image

↳ High-level routine that determines the file format to be loaded based on the filename; it automatically allocates the memory needed for the image data structure and returns cv::Mat.

img.empty()

↳ See if an image was in fact read.

cv::namedWindow("WindowName", cv::WINDOW_AUTOSIZE);

{Name of the window}

{Window property}

{It may be set either 0 (default) or cv::WINDOW_AUTOSIZE}

cv::imshow("WindowName", img);

cv::waitKey(0);

↳ Asks the program to stop and wait for a keystroke.

↳ If a positive argument is given, the program will wait for that number of milliseconds and then continue even if nothing is pressed.

↳ If the argument is set to 0 or to a negative number, the program will wait indefinitely for a key-press.

cv::destroyWindow("WindowName");

↳ Close the window and deallocate any associated memory usage.

* Second Program (Video)

`cv::VideoCapture cap;`

↳ This object can open and close video files of as many types as ffmpeg supports

`cap.open("filename");`

⇒ When created in this way, the object is initialized to the beginning of the video.

* Moving Around

`cap.set(cv::CAP_PROP_POS_FRAME, pos)`

- `cap` → a VideoCapture object
- `pos` → an integer
- `cv::CAP_PROP_POS_FRAME` → Set the frame of the video of pos frame

`frames = (int)cap.get(cv::CAP_PROP_FRAME_COUNT)`

↳ Returns number of frames in the video

`width = (int)cap.get(cv::CAP_PROP_FRAME_WIDTH)`

↳ Returns width of a frame in the video

`height = (int)cap.get(cv::CAP_PROP_FRAME_HEIGHT)`

↳ Returns height of a frame in the video

cap >> frame; → {cv::Mat}

↳ get the current frame of the video

current_pos = (int) cap.get(cv::CAP_PROP_POS_FRAMES);

↳ {Get current frame position}

cv::setTrackbarPos("label", "WindowName", &slider_position, frames, callback);

{number of frames} {Slider position address} {It creates trackbar in the Window name given}

Void Callback(int pos, void*);

{frame position}

★ A Simple transformation

cv::GaussianBlur(input_image, output_image, cv::Size(5,5), 3, 3);

{Standard deviation in x direction} σ_y Kernel size

* Input from a Camera

⇒ cv::VideoCapture object works the same for files on disk or from a camera.

⇒ For the former, you give it a path/filename, and for the latter you give it a Camera ID number.

- 0 if connected to system
- default is -1, which means 'just pick one'.

* Writing to an AVI File

```
cv::VideoWriter writer;
```

→ Video write object.

```
writer.open("filename", CV_FOURCC('M','J','P','G')
            , fps, size)
```

```
cv::Size size(width, height)
```

```
writer << frame
```

→ To write frame to the object

```
capture.release()
```

→ close the video file or capturing device.

★ Important Modules in Opencv

1. Core

The "core" section contains all of the basic object types and their basic operations.

2. imgproc

Contains basic transformations on images, including filters and similar convolutional operations.

3. highgui

This module contains user interface functions that can be used to display images or take simple user input.

4. Video

The video library contains the functions you need to read and write video streams.

5. Calib3d

This module contains implementations of algorithms you will need to calibrate single camera as well as stereo or multicamera arrays.

6. features 2D

This module contains algorithms for detecting, describing, and matching key-point features.

7. Objdetect

This module contains algorithms for detecting specific objects.

8. ML

Contains a wide array of ML algorithm implemented in such a way as to work with the natural data structure of OpenCV.

9. gpu

Contains most of the rest of the library functions optimized for operation on CUDA GPUs.

10. nonfree

You need to do some kind of special work in order to use them in a commercial product

g++ filename.cpp `pkg-config --libs --cflags opencv`

→ (Compiling an OpenCV program)