Highgui

⇒ Split into (imgcodecs), (videoio) & (highgui) in OpenCV 3.0

{ Codecs is a Computer program which encodes or decodes a digital data stream }

CODEC

Coder - Decoder

{ Encodes the data stream } { Decodes the data stream }

★ imgcodecs

① Decoder

{ Path to the file }

cv::Mat cv::imread (const string& filename,
int flags = cv::IMREAD_COLOR)

{ Flags set how to interpret file }

flags accepted by cv::imread()

ⓐ cv::IMREAD_COLOR { → Even if the image is actually grayscale in the file, resulting image will be 8 bit 3 channel with identical info }

⇒ Always load to three-channel array (with 8 bit per channel)

ⓑ cv::IMREAD_GRAYSCALE

⇒ Always load to single-channel array. (8 bit channel)

ⓒ cv::IMREAD_ANYCOLOR

⇒ Channels as indicated by the file (up to three)

ⓓ cv::IMREAD_ANYDEPTH

⇒ Allow loading of more the 8-bit depth

ⓔ cv::IMREAD_UNCHANCED

⇒ Equivalent to cv::IMREAD_ANYCOLOR | cv::IMREAD_ANYDEPTH

" cv::Imread() does not give a runtime error when it fails to load an image, it simply return an empty cv::Mat "

② Encoder

```
bool cv::Imwrite(
    Const String& filename,
    cv::InputArray image,
    Const vector<int>& params = vector<int>()
);
```

(optional)

Image to be stored

Third argument is used for Parameters that are accepted by the particular file type being used.*

filename, whose extension is used to determine the format in which the file will be stored

" Return value will be true if the save was successful and should be false if the save was not "

* The params argument expects an STL vector of integers, with those integers being a sequence of parameter IDs followed by the value to be assigned to that parameter.
  - → Alternating between the parameter ID and the parameter value.

* jpg or jpeg ⇒ 8-bit 1 or 3 ch input.
* jp2 (JPEG2000) ⇒ 8-bit or 16-bit 1 or 3 ch input.
* tif or tiff (TIFE) ⇒ 8-bit or 16-bit 1, 3, 4 ch input.
* png (PNG) ⇒ 8-bit or 16-bit 1, 3, or 4 ch input.
* bmp (BMP) ⇒ 8-bit 1, 3, 4 ch input.
* ppm (PPM) ⇒ 8-bit 3ch input
* pgm (PGM) ⇒ 8-bit 1 channel

{ Net PBM }

★ highgui

① cv::namedWindow()

```
int cv::namedWindow(
    const string& name,
    int flag = 0
);
```

{Name of the window}

{Appears on top of the window}

{Used as a handle for the window that can be passed to other HighGUI functions}

⇒ For now, the only valid options available for flags are to

(a) cv::WINDOW_AUTOSIZE

⇒ Resize the window to fit automatically whenever a new image is loaded, but users cannot resize the window.

(b) 0 (Default)

⇒ Indicates that the users are able to resize the window.

⇒ Destroying a window

```
int cv::destroyWindow(
    Const string& name
);
```

→ {Handle used to identify window}

(2) cv::imshow()

```
Void cv::imshow(
    const string& name,
    cv::InputArray image
);
```

→ {Handle used to identify window}

→ {Image to display in the window}

(3) cv::waitKey()

```
int cv::waitKey(
    int delay = 0
);
```

→ {Milliseconds until giving up (0 = 'never')}

⇒ It returns the key value when it is received.

↳ If no keypress comes before delay milliseconds has passed, cv::waitKey() will return -1.

⇒ There is a second, less obvious function of cv::waitKey(), which is to provide an opportunity for any open OpenCV window to be updated.

↳ This means that if you do not call cv::waitKey(), your image may never be drawn in your window.

↳ Or your window may behave strangely (and badly) when moved, resized, or uncovered.