

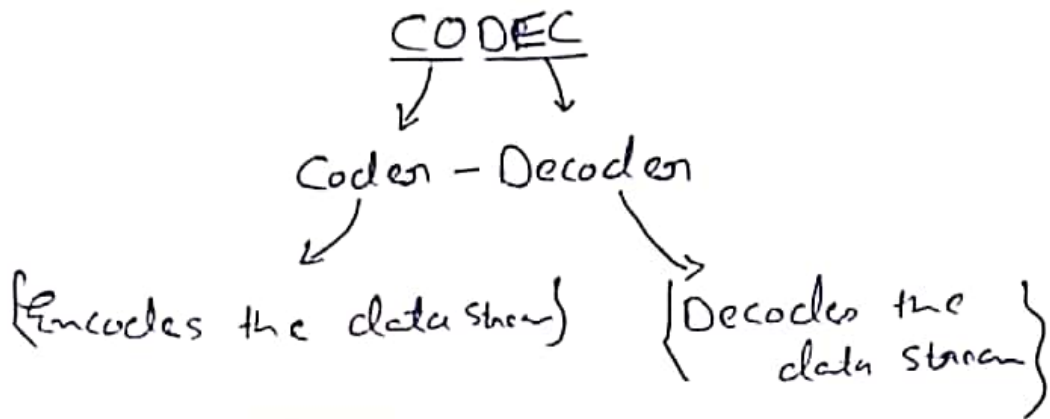
Highgui

①

⇒ Split into imgcodecs, videoio & highgui in OpenCV 3.0

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

CODEC



* imgcodecs

① Decoder

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

{ Path to the file }

{ Flags set how to interpret file }

Flags accepted by cv::imread()

① 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)

② cv::IMREAD_GRAYSCALE

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

③ `cv::IMREAD_ANYCOLOR`

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

④ `cv::IMREAD_ANYDEPTH`

⇒ Allow loading of more than 8-bit depth

⑤ `cv::IMREAD_UNCHANGED`

⇒ 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 returns an empty `cv::Mat` ”

② Encoder

```
bool cv::ImWrite (  
    const string & filename,  
    cv::InputArray image,  
    const vector<int> & params = vector<int>()   
    )  
;
```

{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 ”

- ①
- * The params argument expects a 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 \Rightarrow 8-bit 1 or 3 ch input.
 - * jp2 (JPEG2000) \Rightarrow 8-bit or 16-bit 1 or 3 ch input.
 - * tif or tiff (TIFF) \Rightarrow 8-bit or 16-bit 1, 3, 4 ch input.
 - * png (PNG) \Rightarrow 8-bit or 16-bit 1, 3, or 4 ch input.
 - * bmp (BMP) \Rightarrow 8-bit 1, 3, 4 ch input.
 - * ppm (PPM) \Rightarrow 8-bit 3 ch input
 - * pgm (PGM) \Rightarrow 8-bit 1 channel
- { Not 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

① `cv::WINDOW_AUTOSIZE`

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

② 0 (Default)

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

⇒ Destroying a window

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

Handle word to
identify window

③ `cv::imshow()`

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

Handle word to
identify window

Image to display
in the window

④ `cv::waitKey()`

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

Milliseconds until giving
up (0 = 'never')

⇒ It returns the key value when it is received. ③

↳ If no key press 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.

