

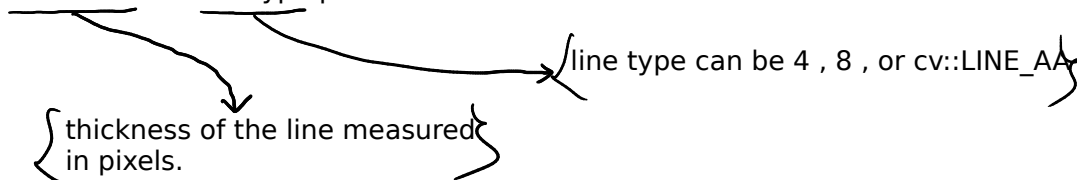
Drawing and Annotating

★ Drawing things

- ⇒ We often want to draw some kind of picture, or to draw something on top of an image obtained from somewhere else.
 - ↳ Functions that will allow us to make lines, squares, circles.
- ⇒ Most of the drawing functions support a color, a thickness, a line type, and subpixel alignment of objects.
- ⇒ When you specify colors, the convention is to use the `cv::Scalar` object, even though only the first three values are used most of the time.
- ⇒ It is sometimes convenient to be able to use the fourth value in a `cv::Scalar` to represent an alpha channel, but the drawing functions do not currently support alpha blending.

★ Line Art and Filled Polygons

- ⇒ Functions that draw lines of one kind or another will usually accept a thickness and `lineType` parameter.



- ⇒ Both thickness and `lineType` are integers.
- ⇒ For circles, rectangles, and all of the other closed shapes, the thickness argument can also be set to `cv::FILLED` (which is an alias for `-1`).
- ⇒ The `lineType` argument indicates whether the lines should be "4-connected," "8-connected," or anti-aliased.

Table 6-1. Drawing functions

Function	Description
<code>cv::circle()</code>	Draw a simple circle
<code>cv::clipLine()</code>	Determine if a line is inside a given box
<code>cv::ellipse()</code>	Draw an ellipse, which may be tilted or an elliptical arc
<code>cv::ellipse2Poly()</code>	Compute a polygon approximation to an elliptical arc
<code>cv::fillConvexPoly()</code>	Draw filled versions of simple polygons
<code>cv::fillPoly()</code>	Draw filled versions of arbitrary polygons
<code>cv::line()</code>	Draw a simple line
<code>cv::rectangle()</code>	Draw a simple rectangle
<code>cv::polyLines()</code>	Draw multiple polygonal curves

● `cv::circle()`

```
void circle(  
    cv::Mat&      img,           // Image to be drawn on  
    cv::Point     center,       // Location of circle center  
    int           radius,       // Radius of circle  
    const cv::Scalar& color,    // Color, RGB form  
    int           thickness = 1, // Thickness of line  
    int           lineType = 8, // Connectedness, 4 or 8  
    int           shift = 0,    // Bits of radius to treat as fraction  
);
```

● cv::line()

```
void line(
    cv::Mat&          img,           // Image to be drawn on
    cv::Point         pt1,          // First endpoint of line
    cv::Point         pt2,          // Second endpoint of line
    const cv::Scalar& color,        // Color, BGR form
    int               lineType = 8, // Connectedness, 4 or 8
    int               shift  = 0    // Bits of radius to treat as fraction
);
```

★ Fonts and Text

Table 6-2. Text drawing functions

Function	Description
cv::putText()	Draw the specified text in an image
cv::getTextSize()	Determine the width and height of a text string

cv::putText()

```
void cv::putText(
    cv::Mat&          img,           // Image to be drawn on
    const string& text,             // write this (often from cv::format)
    cv::Point         origin,       // Upper-left corner of text box
    int               fontFace,     // Font (e.g., cv::FONT_HERSHEY_PLAIN)
    double            fontScale,    // size (a multiplier, not "points")
    cv::Scalar        color,        // Color, RGB form
    int               thickness = 1, // Thickness of line
    int               lineType  = 8, // Connectedness, 4 or 8
    bool              bottomLeftOrigin = false // true='origin at lower left'
);
```