Technische Universität München
Fakultät für Informatik
Prof. Gudrun Klinker, Ph.D.
Andreas Langbein, Adnane Jadid, David Plecher

Summer Term 2018
Exercise Sheet 3
May 2nd, 2018

Introduction to Augmented Reality

Exercise 7 (P,H) Rectangles with subpixel accuracy

In the last exercise, you have used `OpenCV` functions to locate rectangles. However, these results are not yet accurate enough to be used for pose estimation. In this exercise, the rectangles from `OpenCV` are used as guidelines to extract the edges with subpixel accuracy.

(a) At the subdivision points from the previous exercise, sample a strip of width 3 pixels and length proportional to the marker size, orthogonal to the preliminary edge. Take care that the center of the strip is on the center of the line found by `OpenCV`.
Note: it may be easier to use a simple array of `unsigned char` than a full-fledged `cv::Mat`.
Note: also see sketch solution in tutorial 3.
Note: starting from the subdivision point, you can define two vectors with a length of one pixel, one in the direction of the original edge and an orthogonal one. For each of the pixels in your subimage, use the following function to get the correct value from the original image:

```
int subpixSampleSafe ( const cv::Mat &pSrc, const cv::Point2f &p )
{
int x = int( floorf ( p.x ) );
int y = int( floorf ( p.y ) );

if ( x < 0 || x >= pSrc.cols  - 1 ||
 y < 0 || y >= pSrc.rows - 1 )
return 127;

int dx = int ( 256 * ( p.x - floorf ( p.x ) ) );
int dy = int ( 256 * ( p.y - floorf ( p.y ) ) );

unsigned char* i = ( unsigned char* ) ( ( pSrc.data + y * pSrc.step ) + x );
int a = i[ 0 ] + ( ( dx * ( i[ 1 ] - i[ 0 ] ) ) >> 8 );
i += pSrc.step;
int b = i[ 0 ] + ( ( dx * ( i[ 1 ] - i[ 0 ] ) ) >> 8 );
return a + ( ( dy * ( b - a) ) >> 8 );
}
```

(b) After the subimage has been created, apply the Sobel operator. It looks as follows:

$$\begin{pmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{pmatrix}$$

To get the new value for the center pixel, take the original value of each pixel, multiply with the respective coefficient and add the results. Store them in a second array of `int`.
Note: the above matrix only makes sense if your sub-image consists of three rows of pixels. If you have three columns instead, you have to transpose it.

(c) In order to check your results, create a new `cv::Mat` of appropriate height and width. Copy one of the stripes from exercise (b) into this image and display it in a new window.
Note: Take care not to overflow the `unsigned char` datatype of the image. You may have to normalize your image to a range of `[ 0;255 ]`.

(d) In the resulting array, first find the maximum (or minimum) pixel. Whether you need to look for a maximum or a minimum depends on your particular implementation.
Now, in order to find the edge location with subpixel accuracy, the quadratic equation $y = ax^2 + bx + c$ will be used. Use the maximum from the previous step and its two neighbors as positions 0, 1 and -1 on the x axis. The intensity values correspond to the y axis. Use the equation to determine the extremum and finally calculate the exact subpixel location of the edge.