Eric Miao

CS-7390 OpenCV

Homework-5: Feature Detection/AffineTransformations

Gaussian and Laplacian Pyramids (26 pnts)

1) (20 pnts) Write a program **LaplacianPyramid.cpp** to construct a Laplacian pyramid of 6 levels from an original 256x256 color image. (Consider the original the 1st level) Create a display that shows all images. (Customarily, the smaller images are displayed aligned along the horizontal right edge of the original image.) Upload your code and a screenshot of your pyramid.

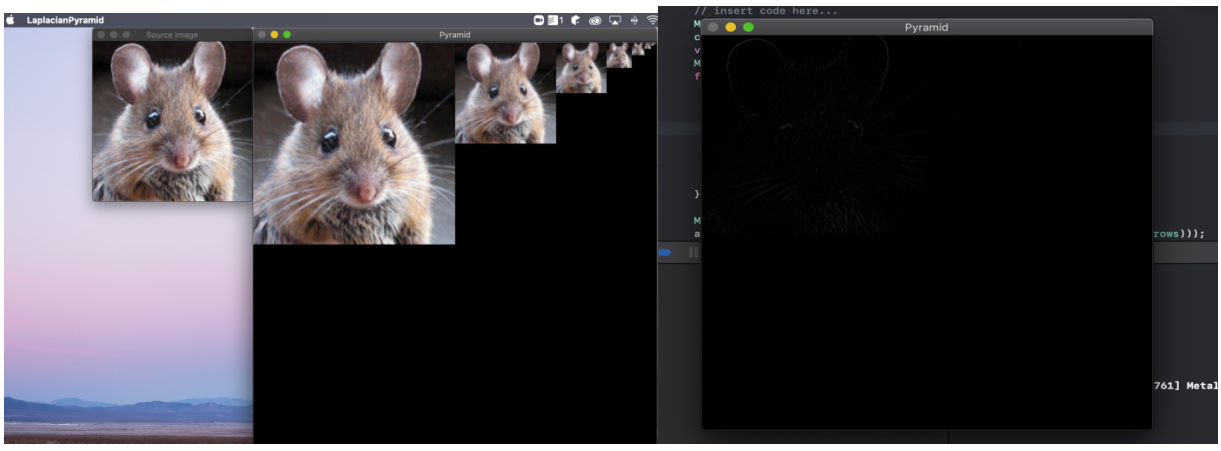
You may wish to use the cv:: pyrDown(..) in the imgproc module

```
void cv::pyrDown ( InputArray src, OutputArray dst,

const Size & dstsize = Size( ),

int borderType = BORDER_DEFAULT )
```

Here is the OpenCV tutorial to create a Gaussian Pyramid.
https://docs.opencv.org/3.4/d4/d1f/tutorial_pyramids.html

Before applying Gaussian Blur (Left)

After applying Gaussian Blur (Right) – It looks super dark I know…



I also attached the source code in the submission folder.

Eric Miao

2) (6 pnts) How many bytes does the pyramid require to store the image data itself? Can you come up with a general formula to determine how much storage is required for any general pyramid in terms of the storage of the original image? That is…if the original image requires NxN bytes, can you come up with an expression for how much storage a Gaussian pyramid will require? (HINT: Think in terms of a converging series.)

We assume that the original image is NxN

The number of images in the pyramid is log2(N)

$$\text{Storage} = \sum_{k=0}^{Log2(N)} \frac{N^2}{4^i}$$

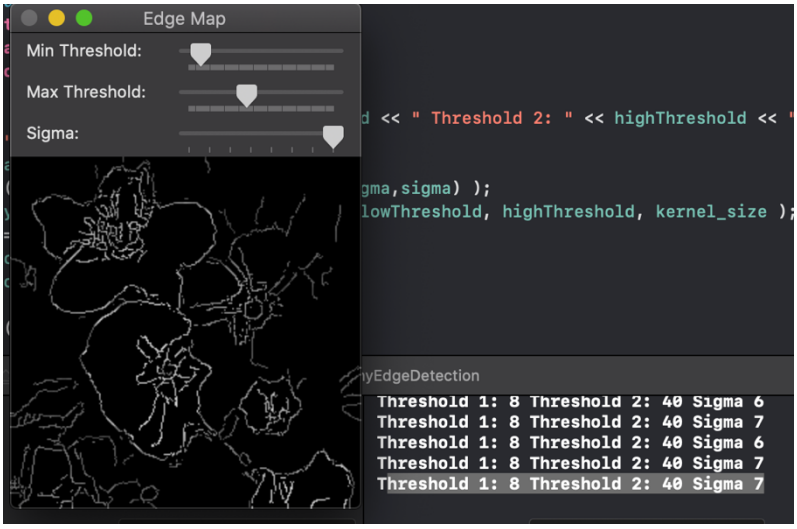
Feature Detectors (30 pnts)

3) (16 pnts) Run and experiment with the Canny Edge Detector using the supplemental code provided with this assignment. Explore settings for the two thresholds that provide a reasonable edge detection while minimizing detection of non-edges.

A) Find values for the two thresholds that keep a 1:5 ratio.

Write the values here: Threshold 1: 8 Threshold 2: 40 Sigma 7

Provide a screenshot:

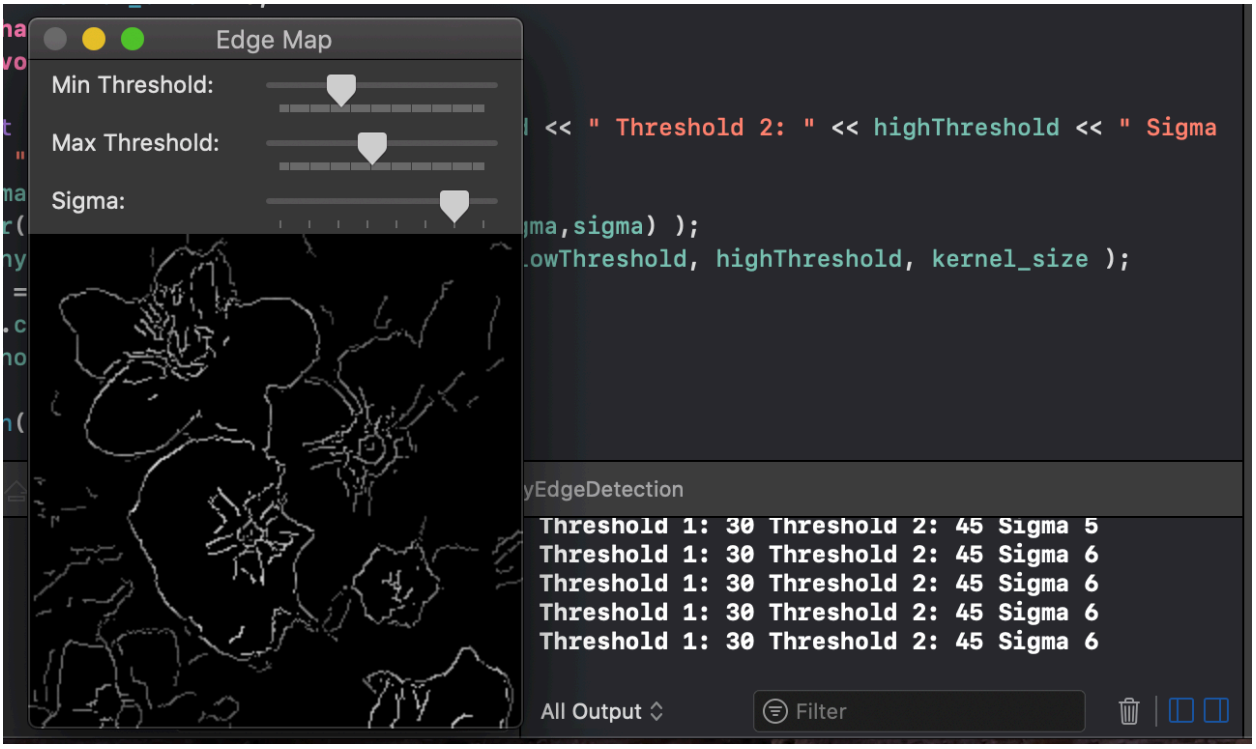


Eric Miao

B) Find settings that keep a ratio of 2:3 between the two thresholds.

Write the values here: Threshold 1: 30 Threshold 2: 45 Sigma 6

Provide a screenshot



C) What do you observe about the types of edges detected in parts A and B. Could you characterize the edges as fine or coarse?

Overall, I think it looks pretty fine to me. However, If I lower the min/max threshold values, it gets finer and if I higher the min/max threshold values, it gets a slightly coarser. Similarly, if I lower the sigma value, it gets finer. If I higher the sigma value, it gets coarser.

D) How does sigma affect this, visually, with respect to type of features detected? What is sigma related to in the Canny Edge detection algorithm?

I did some research and I found that the sigma plays the role of a scale parameter for the edges: large values of sigma produce coarser scale edges and small values of sigma produce finer scale edges. Larger values of sigma also result in greater noise suppression. Smaller filters(sigma) cause less blurring, and allow detection of small, sharp lines. A larger filter (sigma) causes more blurring, smearing out the value of a given pixel over a larger area of the image. Larger blurring radii are more useful for detecting larger, smoother edges

Eric Miao

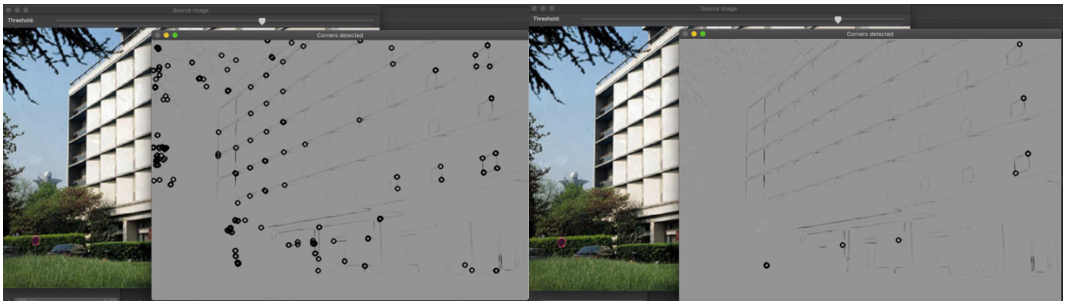
4) (14 point) Run and explore a few of the following demo programs that were installed with your OpenCV build. Choose one that applies to a static image and one that applies to a video stream and answer some questions about what you observe.

FEATURES IN STATIC IMAGES
<your OPENCV Install>/build/bin/example_tutorial_HoughCircle_Demo
<your OPENCV Install>/build/bin/example_tutorial_HoughLines_Demo
<your OPENCV Install>/build/bin/example_tutorial_cornerHarris_Demo
<your OPENCV Install>/build/bin/example_tutorial_findContours_demo

FACES AND EXPRESSIONS IN VIDEO
<your OPENCV Install>/build/bin/example_cpp_dbt_face_detection
<your OPENCV Install>/build/bin/example_cpp_facedetect
<your OPENCV Install>/build/bin/ example_cpp_smiledetect
A) Which one did you explore in detail? B) What types of features does it detect?
I explored example_tutorial_cornerHarris_Demo in detailed.
It detects all the corners on the image (interest points).

C) What types of parameters control its sensitivity to detection (if applicable)
blockSize, apertureSize

D) How good is it at avoiding falsely detecting features (if applicable)



I think it really depends on the threshold. As we can see if we higher the threshold, it detects most of the corners, but it also falsely found some points that are not what we want. On the other hand, if we lower the threshold, it correctly detects some points but we need more than that.

E) How sensitive is it to rotation (if applicable)? (i.e. How does it perform when object rotates?)
I think it should perform the same as the algorithm itself seems not be influenced by the direction of the image.

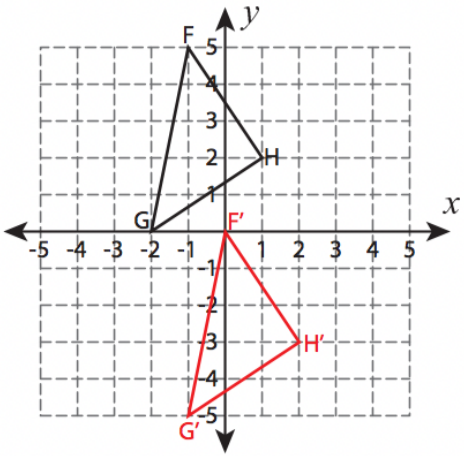
(F) What OpenCV API are specific to this type of detection?
(Look at the code! Find most relevant one or two APIs if used together)
cv::cornerHarris(InputArray src, OutputArray dst, int blockSize, int ksize, double k, int borderType = BORDER_DEFAULT)

Eric Miao

2D Geometric Affine Transformations (24 pnts)

7) For each of the following figures, you see a figure in black with vertex coordinates on xy-plane, each denoted with a letter here for easy reference, and a red version of the figure shown with a desired displacement in the plane. Determine the transformation matrix, M, need to achieve the desired transformation. Use homogeneous coordinates. Verify with actual values for particular.

A) Verify your matrix by showing the computation of point H'' =M H and G' = M G

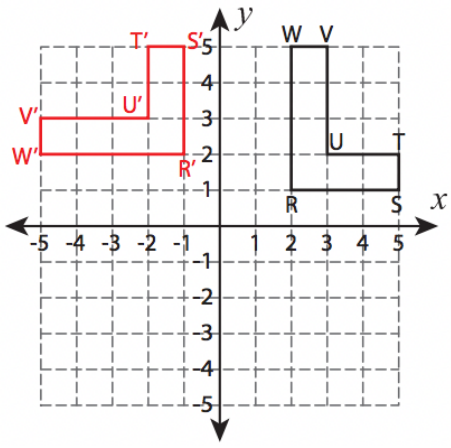


$$M = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & -5 \\ 0 & 0 & 1 \end{bmatrix}$$

$$H' = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & -5 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 2 \\ 3 \\ 1 \end{bmatrix}$$

$$G' = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & -5 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} -2 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} -1 \\ -5 \\ 1 \end{bmatrix}$$

B) Verify your matrix by showing the computation of point R' =M R and U' = M U (HINT: It's a rotation!)



$$M = \begin{bmatrix} \cos 90 & -\sin 90 & 0 \\ \sin 90 & \cos 90 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$R' = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} -1 \\ 2 \\ 1 \end{bmatrix}$$

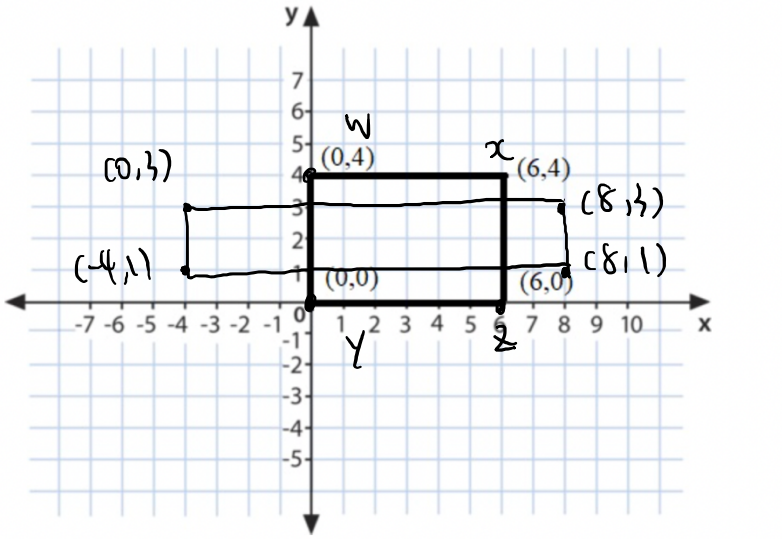
$$U' = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 3 \\ 2 \\ 1 \end{bmatrix} = \begin{bmatrix} -2 \\ 3 \\ 1 \end{bmatrix}$$

$R' = (-1, 2)$ $\quad$ $U' = (-2, 3)$

Eric Miao

8) Given this figure, with coordinates (x,y) and the following transformation matrix, compute the transformed coordinates (x', y') and sketch its resulting position on the 2D plane. (You may show on the same coordinate grid for convenience.)

$$M=\begin{bmatrix} 2 & 0 & -4 \\ 0 & 0.5 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$



$$W \cdot \begin{bmatrix} 2 & 0 & -4 \\ 0 & 1/2 & 1 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 3 \\ 1 \end{bmatrix}$$

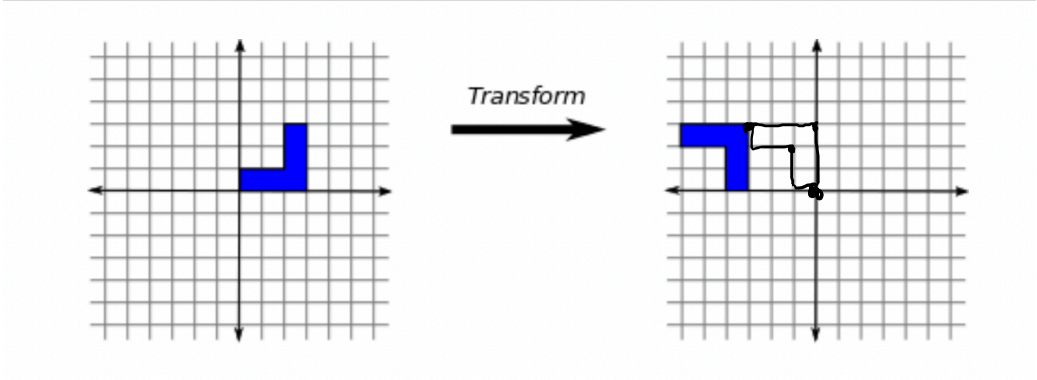
$$X \cdot \begin{bmatrix} 2 & 0 & -4 \\ 0 & 1/2 & 1 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 8 \\ 3 \\ 1 \end{bmatrix}$$

$$Y \cdot \begin{bmatrix} 2 & 0 & -4 \\ 0 & 1/2 & 1 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} -4 \\ 1 \\ 0 \end{bmatrix}$$

$$Z \cdot \begin{bmatrix} 2 & 0 & -4 \\ 0 & 1/2 & 1 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 8 \\ 1 \\ 0 \end{bmatrix}$$

Eric Miao

9) Consider the following transformation of a simple shape, shown in a traditional xy-coordinate system. Assume grid lines are 1 apart. Write the combination of basic transformations (scale, translate, rotate) that must be applied to achieve this in sequence. Use homogeneous coordinates, to illustrate clearly the order in which the transformations must be applied to each point (x,y) to get the transformed points (x'y').



$$M_1 = \begin{bmatrix} \cos 90 & -\sin 90 & 0 \\ \sin 90 & \cos 90 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$ (For Rotation)

$$M_2 = \begin{bmatrix} 1 & 0 & -3 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$ (For translate)

original cordinates:		tranformed cordinates:
$(0,0) \times M_1 = (0,0)$	$\times M_2$	$= (-3,0)$
$(0,1) \times M_1 = (-1,0)$	$\times M_2$	$= (-4,0)$
$(2,1) \times M_1 = (-1,2)$	$\times M_2$	$= (-4,2)$
$(2,3) \times M_1 = (-3,2)$	$\times M_2$	$= (-6,2)$
$(3,3) \times M_1 = (-3,3)$	$\times M_2$	$= (-6,3)$
$(3,0) \times M_1 = (0,3)$	$\times M_2$	$= (-3,3)$