CS5390/7390 /Fall 2020 / Homework 5

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

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.)

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:

Provide a screenshot:

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

Write the values here:

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?

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?

4) (14 pnts) 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?

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

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

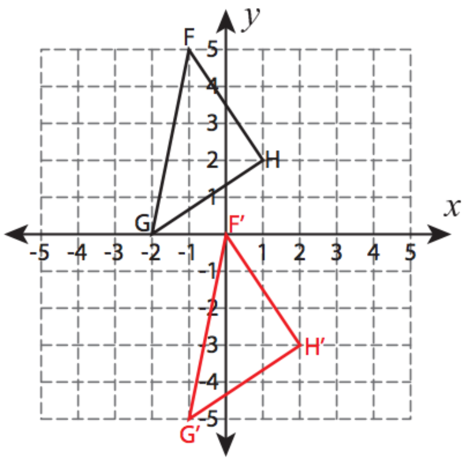
E) How sensitive is it to rotation (if applicable)?
(i.e. How does it perform when object rotates?)

(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)

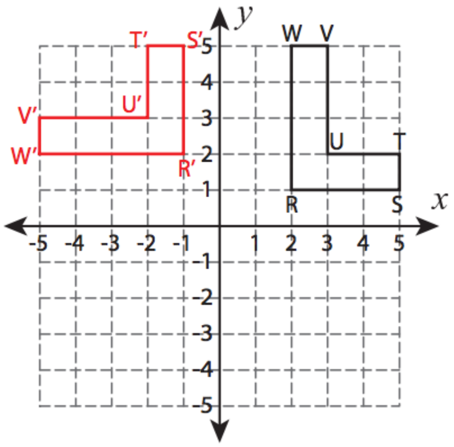
2D Geometric Affine Transformations (24 pnts)

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

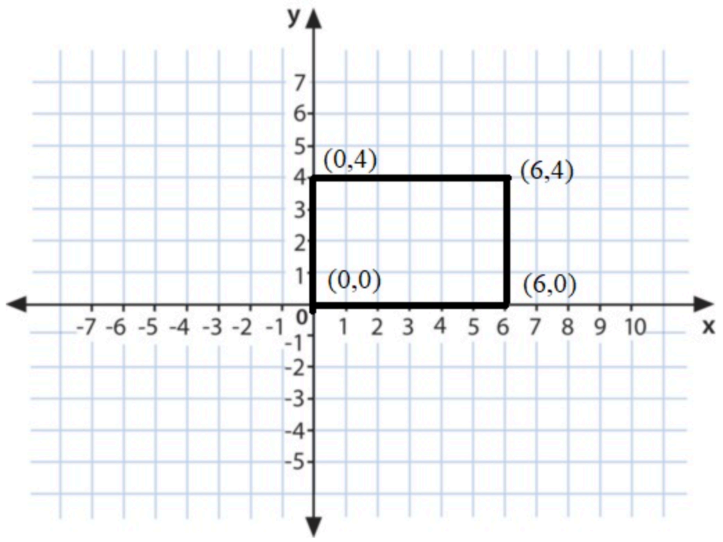


B) Verify your matrix by showing the computation of point R' =M R and U' = M U (HINT: It's a rotation!)

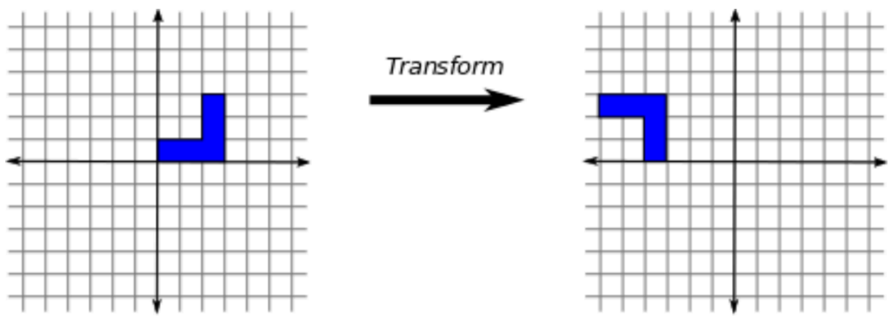


8) Given this figure, with coordinates (x,y) and the following transformation matrix, compute the transformed coordinates (x', y') and sketch its resulting position on the 2D plane. (You may show on the same coordinate grid for convenience.)

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



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').

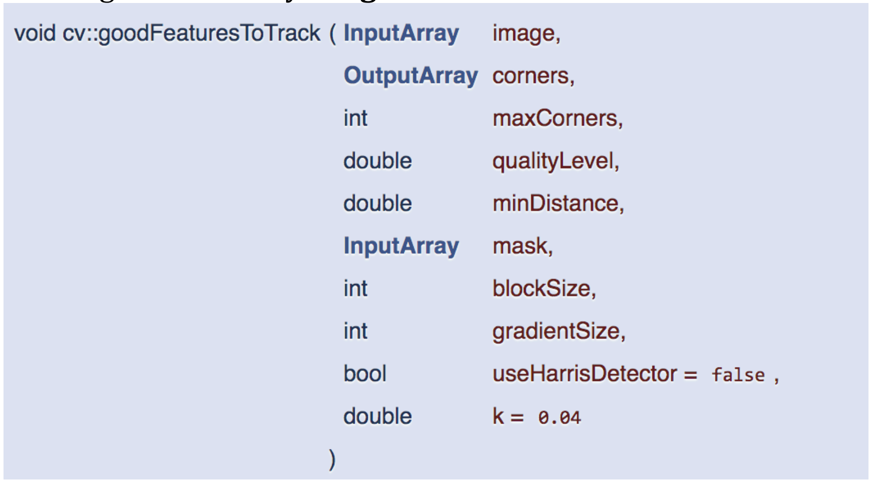


CS5390/7390 /Fall 2020 / Homework 5

Pre-Project: (20 points) Find or Create an Easily Detectable Object in your Own World, Suitable for Capturing with your Video Camera

Find or create a distinctive object that you can hold in view of your video camera. Create a single image with this object in it.

Apply a feature detecting algorithm in the OpenCV API to detect your object.
You might want to try the **goodFeaturesToTrack** API:



```
void cv::goodFeaturesToTrack ( InputArray   image,
                               OutputArray  corners,
                               int          maxCorners,
                               double       qualityLevel,
                               double       minDistance,
                               InputArray   mask,
                               int          blockSize,
                               int          gradientSize,
                               bool         useHarrisDetector = false ,
                               double       k = 0.04
                             )
```

Ideally, you will find an object, a background and a method so that your object is detected and nothing else is.

Use OpenCV drawing functins to mark objects returned with graphics on top of your image.

Upload screenshots of your object and a few screenshot samples of how effectively your object can be detected without false positivesBe prepared to share your objects and results.