Review Computer Vision

1. This chapter have completed basic I/O programming and data structures in OpenCV. The following exercises build on this knowledge and create useful utilities for later use.

a. Create a program that (1) reads frames from a video, (2) turns the result to grayscale, and (3) performs Canny edge detection on the image. Display all three stages of processing in three different windows, with each window appropriately named for its function.

b. Display all three stages of processing in one image.

c. Write appropriate text labels describing the processing in each of the three slots.

2. Create a program that reads in and displays an image. When the user's mouse clicks on the image, read in the corresponding pixel (blue, green, red) values and write those values as text to the screen at the mouse location.

3. Create your own simple paint program.

a. Write a program that creates an image, sets it to 0, and then displays it. Allow the user to draw lines, circles, ellipses, and polygons on the image using the left mouse button. Create an eraser function when the right mouse button is held down.

b. Allow "logical drawing" by allowing the user to set a slider setting to AND, OR, and XOR. That is, if the setting is AND then the drawing will appear only when it crosses pixels greater than 0 (and so on for the other logical functions).

4. Load an image with interesting textures. Smooth t 1. he image in several ways using cvSmooth() with smoothtype=CV_GAUSSIAN.

a. Use a symmetric 3-by-3, 5-by-5, 9-by-9 and 11-by-11 smoothing window size and display the results.

b. Are the output results nearly the same by smoothing the image twice with a 5-by-5 Gaussian fi lter as when you smooth once with two 11-by-11 filters? Why or why not?

5. Take a picture of a scene. Then, without moving the camera, put a coffee cup in the scene and take a second picture. Load these images and convert both to 8-bit grayscale images.

a. Take the absolute value of their difference. Display the result, which should look like a noisy mask of a coffee mug.

b. Do a binary threshold of the resulting image using a level that preserves most of the coffee mug but removes some of the noise. Display the result. The "on" values should be set to 255.

c. Do a CV_MOP_OPEN on the image to further clean up noise.

6. Load an image of a scene and convert it to grayscale.

a. Run the morphological Top Hat operation on your image and display the results.

b. Convert the resulting image into an 8-bit mask.

c. Copy a grayscale value into the Top Hat pieces and display the results.

7. Load an image of a scene. Use cvPyrSegmentation() and display the results.

8 Load an image of an interesting or suffi ciently “rich” scene. Using cvThreshold(), set the threshold to 128. Use each setting type in Table 5.5 on the image and display the results. You should familiarize yourself with thresholding functions because they will prove quite useful.

a. Repeat the exercise but use cvAdaptiveThreshold() instead. Set param1=5.

b. Repeat part a using param1=0 and then param1 = -5.

Table 5-5. cvThreshold() threshold_type options

Threshold type	Operation
CV_THRESH_BINARY	$dst_I = (src_I > T)\ ?\ M : 0$
CV_THRESH_BINARY_INV	$dst_I = (src_I > T)\ ?\ 0 : M$
CV_THRESH_TRUNC	$dst_I = (src_I > T)\ ?\ M : src_I$
CV_THRESH_TOZERO_INV	$dst_I = (src_I > T)\ ?\ 0 : src_I$
CV_THRESH_TOZERO	$dst_I = (src_I > T)\ ?\ src_I : 0$