ECE 510 EmbV I
Fall 2016
Lab02 notes

Morphological and filter operations

Task 01 – Gaussian filter complexity

In order to help visualize the problem I drew the following three pictures. Figure 1 shows a 10x10 image with a 1x3 filter, Figure 2 shows a 10x10 image with a 3x1 filter and Figure 3 shows a 10x10 image with a 3x3 filter.

	204	133	34	183	176	249	126	60	57	255	
	220	16	117	251	8	36	163	247	87	99	
	229	146	62	96	147	191	22	10	255	96	
	175	250	179	251	107	34	222	48	121	253	
	7	162	20	181	25	63	60	197	25	125	
	116	150	51	172	112	8	235	228	89	166	
	23	199	84	8	124	103	44	143	74	161	
	60	140	110	85	37	91	211	189	198	4	
	230	93	139	183	96	64	214	162	80	185	
	244	210	10	9	99	115	246	53	93	13	

Figure 1. NxN image with a 1xk kernel. Here N = 10 and k = 3.

For the convolution of a NxN image with a 1xk kernel with the output NxN (including the border pixels) we need k multiplications and k-1 additions per pixel. If we only take multiplications into consideration the complexity of the operation is $O(N^2*k)$.

	204	133	34	183	176	249	126	60	57	255	
	220	16	117	251	8	36	163	247	87	99	
	229	146	62	96	147	191	22	10	255	96	
	175	250	179	251	107	34	222	48	121	253	
	7	162	20	181	25	63	60	197	25	125	
	116	150	51	172	112	8	235	228	89	166	
	23	199	84	8	124	103	44	143	74	161	
	60	140	110	85	37	91	211	189	198	4	
	230	93	139	183	96	64	214	162	80	185	
	244	210	10	9	99	115	246	53	93	13	

Figure 2. NxN image with a kx1 kernel. Here N = 10 and k = 3.

For the convolution of a NxN image with a kx1 kernel with the output NxN we need k multiplications and k-1 additions per pixel. Based on the fact that most computing hardware supports multiply and add – MAD – instructions in one steps the complexity of the operation is $O(N^2*k)$.

	204	133	34	183	176	249	126	60	57	255	
	220	16	117	251	8	36	163	247	87	99	
	229	146	62	96	147	191	22	10	255	96	
	175	250	179	251	107	34	222	48	121	253	
	7	162	20	181	25	63	60	197	25	125	
	116	150	51	172	112	8	235	228	89	166	
	23	199	84	8	124	103	44	143	74	161	
	60	140	110	85	37	91	211	189	198	4	
	230	93	139	183	96	64	214	162	80	185	
	244	210	10	9	99	115	246	53	93	13	

Figure 3. NxN image with a kxk kernel. Here N = 10 and k = 3.

For the convolution of a NxN image with a kx1 kernel with the output NxN we need k^2 multiplications and k^2-1 additions per pixel. The complexity of the operation is $O(N^2*k^2)$.

For the convolution of a NxN image with a 1xk kernel followed by the convolution with a kx1 kernel we need 2*k multiplications and 2*(k-1) additions per pixel. The complexity of the operation is $O(N^2*2*k)$.

For the particular case where **N = 10 and k = 3**:

- separable convolution (first on rows then on columns) takes $10^2*2*3 = 600$ operations.
- 2D convolution takes $10^2*3^2 = 900$ operations.

This means that performing convolution with the 3x3 kernel requires **1.5x more operations** than performing the 1x3 and 3x1 convolutions separately.

Task 02 – Erode/Dilate/Open/Close

The code for this section can be found in lab02::Task02(…).
The output images produced by this task are saved to "C:\OpenCV\Solution\images\lab02\task02" and have suggestive names to help identify the morphological operations performed.

In order to test various combination of erode/dilate I implemented lab02::ErodeDilateTestAll(…) which generates all combinations of erode, dilate operations with rectangular, cross & ellipse structuring elements. In the following paragraphs 'bwImage' stands for 'black & white image'.

Q1: Erode(bwImage) == Negate(Dilate(Negate(bwImage)))?
A1: Yes. Figure 4 shows that the output is the same. Basically erode on the true image is performing the same operation as dilate on the negative image.

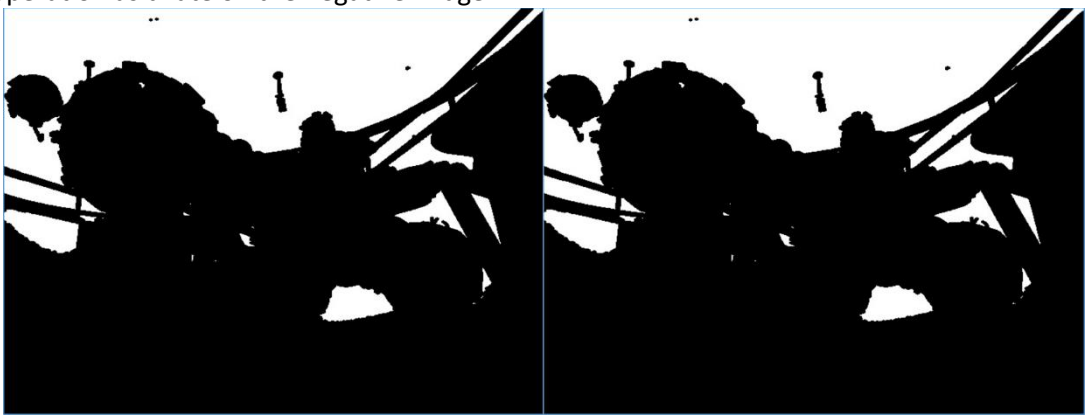

Figure 4. Erode true image (left) vs. Negate dilated negative image (right) with an ellipse kernel.

Q2: Dilate(bwImage) == Negate(Erode(Negate(bwImage)))?
A2: Yes. Figure 5 shows that the output is the same. Dilate on the true image is performing the same operation as erode on the negative image.

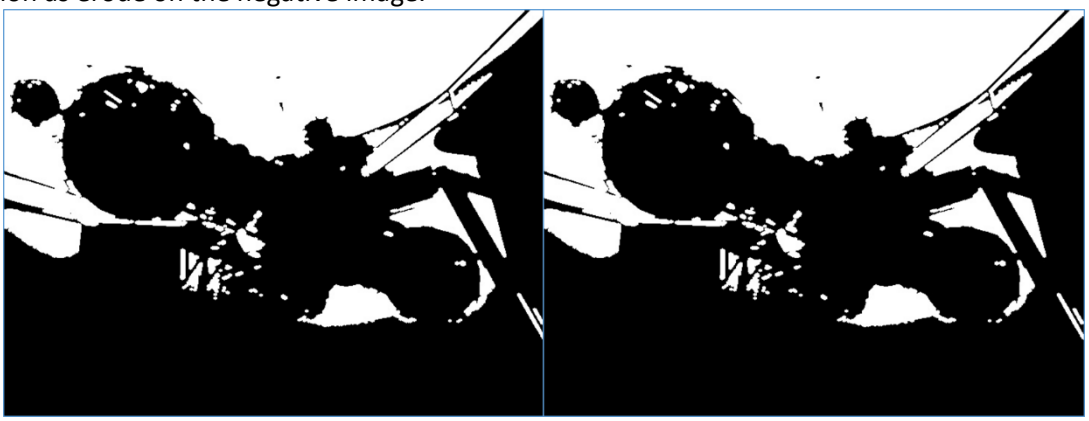

Figure 5. Dilate true image (left) vs. Negate eroded negative image (right) with an ellipse kernel.

Q3: Morphological gradient gives the same edges with either true or negative images?
A3: No. Figure 6 shows the gradient calculated as DILATED-ERODED. The eroded image is subtracted from the dilated image. We can clearly see differences between the left side (gradient based on the true image) and the right side (gradient based on the negative image).

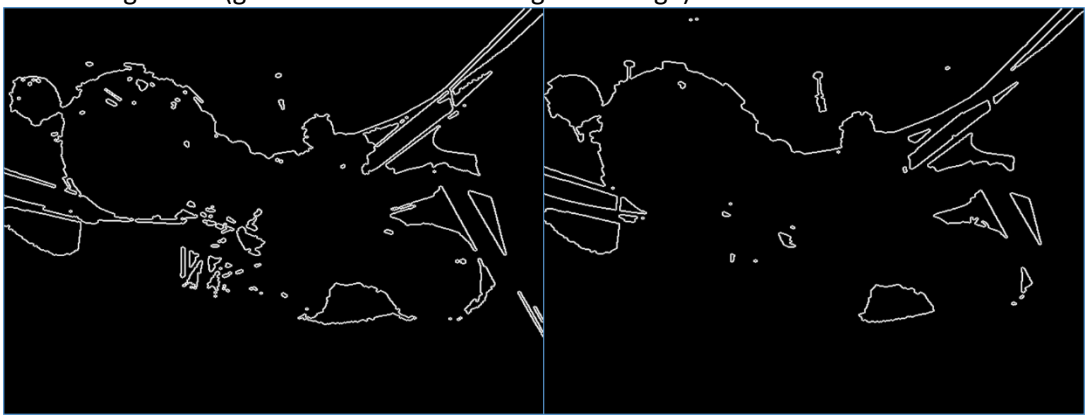

Figure 6. Morphological gradient DILATED-ERODEDs. We subtract from dilated image the eroded image. The left image is the output based on the true image. The right is the output based on the negative image.

Task 03 – Coffee cup

The code for this task can be found in lab02::Task03(…).
The output images produced by this task are saved to "C:\OpenCV\Solution\images\lab02\task03s" and have suggestive names to help identify the morphological operations performed.

Task03.a

Figure 7. The absolute difference.

Task03.b

I was unable to capture a clean image of the cup by applying just threshold. If the threshold is too high the resulting cup is incomplete. If the threshold is too small additional artifacts start to appear like in Figure 8.

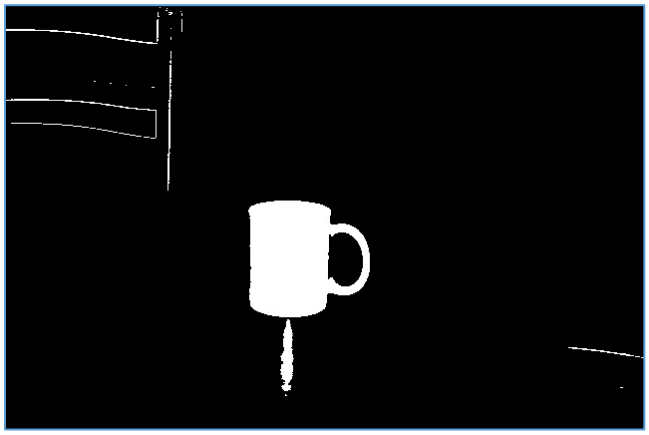

Figure 8. Binary image with a threshold of 40.
Please note the additional artifacts in the image. These are due to surface reflections.

In order to eliminate the reflections I used a median filter with a size of 7 on the absolute difference followed by thresholding with a larger value of 70 in order to preserve the shape of the cup. Figure 9. I have tried Gaussian blur and bilinear filtering as well but with less success.

Figure 9. The result of applying a median filter with a size 7 on the absolute difference image. This was followed by binary thresholding with 70.

I applied erode/dilate with an ellipse kernel size of 7 and obtained a good approximation of the cup as you can see in Figure 10.

Figure 10. Erode/Dilate with an ellipse kernel of size 7 removes the rest of the artifacts. This results in a good approximation of the cup.

Task03.c

Due to the additional filtering steps undertaken in section Task03.b I was able to successfully find the cup and flood fill its pixels with intensity value 100.

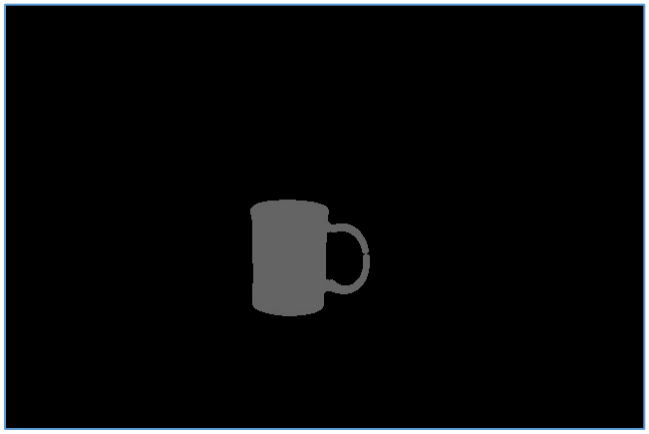

Figure 11. The cup flood filled with intensity value 100.

Task03.d

I addressed this question in section Task03.b. I was able to obtain a clean and reasonable approximation of the cup.

Task03.e

For this task I used the image from “c:\OpenCV\sources\samples\cpp\board.jpg” which comes with OpenCV. The image was slightly bigger than the mask so I cropped it.

Figure 12 shows the cup after the masking operation. Figure 13 shows the image of the board with a cutout in the shape of the mug. This image was obtained by masking the image of the board with the complement of the mask which was obtained by subtracting the mask from 255. Figure 14 shows the composite images which adds the image in Figure 12 to the image in Figure 13.

Figure 12. The cup after the masking operation.

Figure 13. The image after the masking operation with the cup complement.
Please note in the center there’s a region of all black pixels in the shape of a cup.

Figure 14. The composite image.

Additional notes

The images in this paper and some more can be generated by running the ‘lab02’ project. The images are saved to “C:\OpenCV\Solution\images\lab02\task02” and “C:\OpenCV\Solution\images\lab02\task03”.