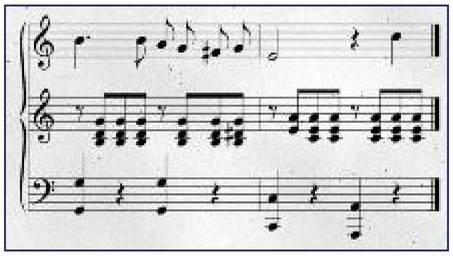
ELL715
Assignment 6 Report

Ankit Kumar (2017MT10727) , Naman Jhunjhunwala (2017MT10737)

April 20, 2020

1 Notes

1.1 Original Image



1.2 Dilation

There can be two approaches to this:

- Approach 1: Treating the music notes as 0, and the background as 1. This seems intuitive, because of course, value for black is taken to be 0 and that for white is taken to be 1 (or 255).
- Approach 2: Treating the music notes as 1 and the background as 0. This one seems correct, because dilation is supposed to expand objects, music notes in this case. Since the kernel is all 1 (in block kernel), dilation will expand the music notes only if the notes themselves are 1 too.

The problem statement doesn't explicitly mention which approach to use, and hence we have attached results corresponding to both of the approaches.

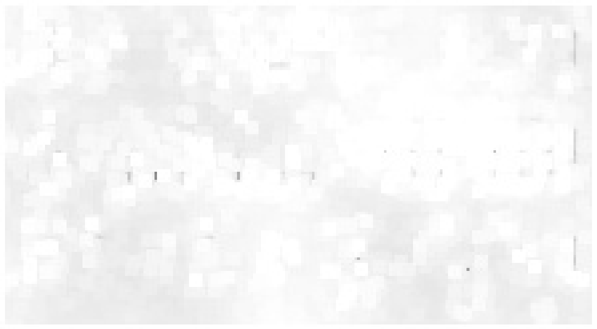
1.2.1 Approach 1

Figure 1.a corresponds to dilation of the original image using a $5 * 5$ block kernel:

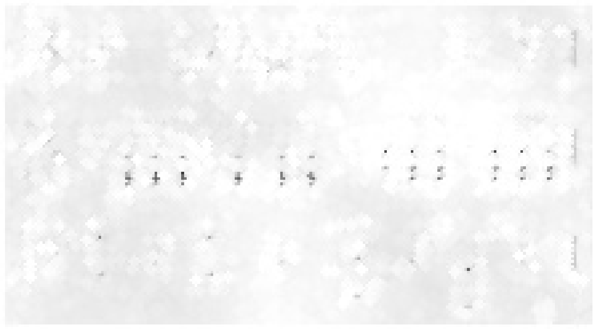
$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Similarly, figure 1.a corresponds to dilation of the original image using a $5 * 5$ diamond kernel:

$$\begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$



(a) 1X Dilate(Block, 5)



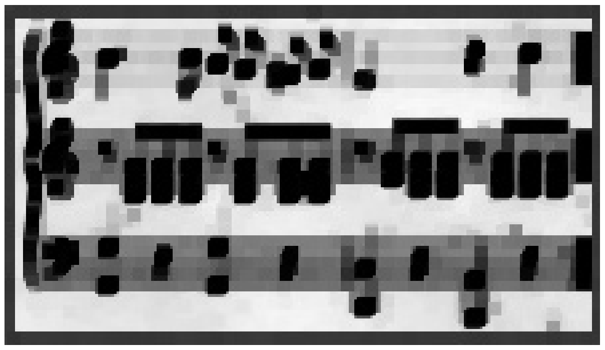
(b) 1X Dilate(Diamond, 5)

Figure 1: Images showing dilation output for notes.jpg for different kernels

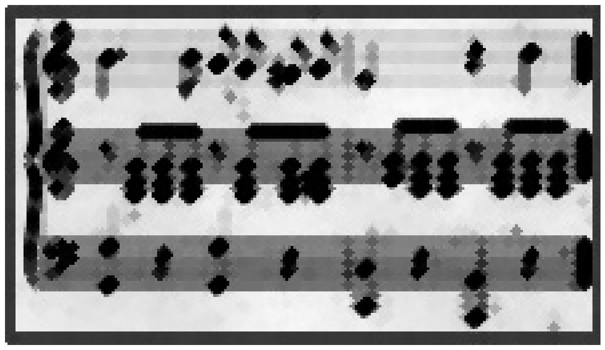
Outputs produced in this case are almost the same because most of the image is covered with white foreground (very less portion is black background which are basically spots having either length or width of single pixel). We know during process of dilation structuring element is aligned with each pixel and if it intersects with any part of foreground, then the output value in that pixel is set to 1. Since we have large structuring elements, even when it gets aligned with background pixel, it intersects with the foreground and hence most of the pixel are 1 in both images. But since the structuring elements are different, there are parts in the images (for instance, around the note heads) where there is a visible difference in both the images. Thus, although very similar, the outputs are **not exactly** the same.

1.2.2 Approach 2

For this, we take negative of the image, to make music notes have the value 255. Then we apply the appropriate morphological operation. Finally, we again take negative of the image to obtain image in the earlier format (with music notes having value 0 – or at least closer to zero than 255).



(a) 1X Dilate(Block, 5)



(b) 1X Dilate(Diamond, 5)

Figure 2: Images showing dilation output for notes.jpg for different kernels

Outputs are different because we used two different structuring elements with different properties. First one is block kernel giving rise to the square spots in the output image while other one is diamond kernel and hence diamond-like spots in the output image.

1.3 Closing

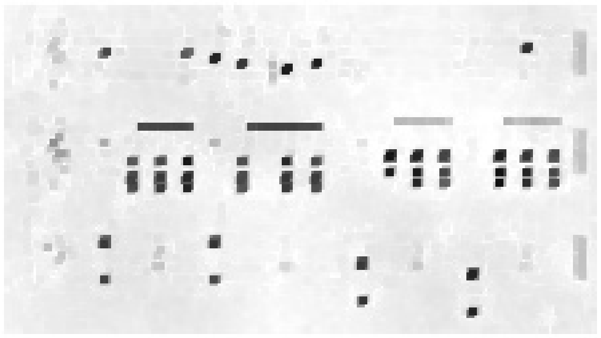
Similar to dilation, there can again be two approaches to this problem.

1.3.1 Approach 1

Both the outputs correspond to closing with a $3 * 3$ block kernel:

$$\begin{bmatrix}1 & 1 & 1\\1 & 1 & 1\\1 & 1 & 1\end{bmatrix}$$

Figure 3.a corresponds to a single closing iteration with the above kernel, while figure 3.b corresponds to 3 closing iterations with the same kernel.


(a) 1X Close(Block, 3)

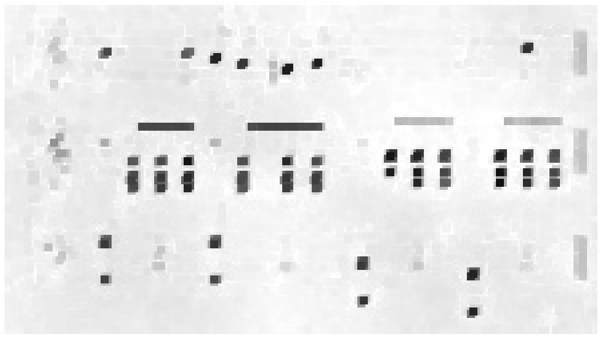
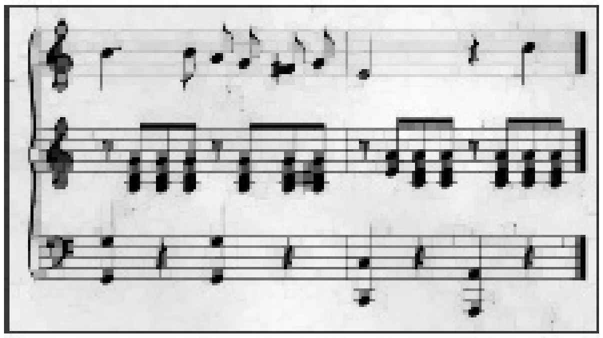

(b) 3X Close(Block, 3)

Figure 3: Images showing closing output for notes.jpg for different kernels

Both the outputs are the same. It is an expected result since opening and closing are idempotent operations. The multiple application of opening/closing doesn't have any effect after the first time you apply it.

1.3.2 Approach 2

We again take negative of image, work on it, and revert back to the original "format".


(a) 1X Close(Block, 3)

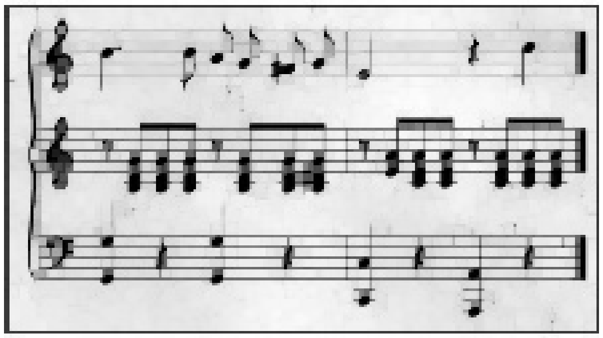

(b) 3X Close(Block, 3)

Figure 4: Images showing closing output for notes.jpg for different kernels

The outputs are again the same, as explained in the previous subsection (approach 1).

1.4 Detecting black note heads

1.4.1 Algorithm

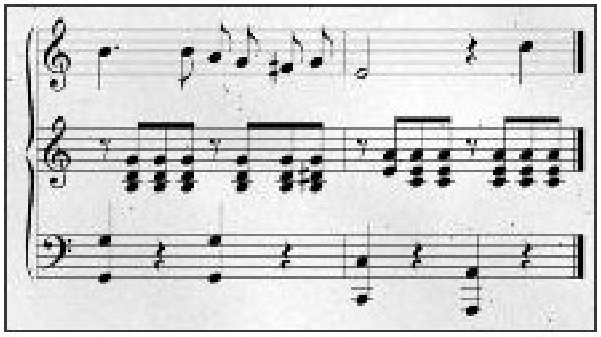
- Closing with a disk kernel with radius 2. Note that it will essentially be a matrix of size $(2r-1)*(2r-1)$ (or $2r+1$, depending upon how you define radius in your matrix – we have used the former one). The disk kernel is:

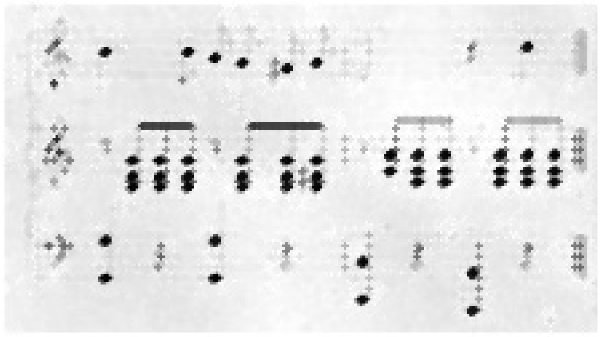
$$\begin{bmatrix}0 & 1 & 0\\1 & 1 & 1\\0 & 1 & 0\end{bmatrix}$$

- Thresholding the closing output by $(40, 255)$.

1.4.2 Explanation

Note that the shape of note heads is circular. But, the size of the image is too small, and hence so is the size of those "circles". If we increase the radius of the disk, the change is too drastic. A few of the note heads become blurred out too along with the background. On the other hand, if we decrease the radius, $r = 1$ is the only possibility, and for that, the image almost remains the same, since the kernel is just [1]. Thus, radius 2 seems to be the most appropriate.


(a) Closing with $r = 1$


(b) Closing with $r = 2$

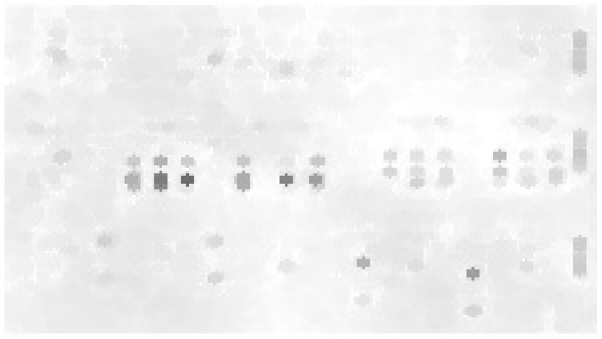

(c) Closing with $r = 3$

Figure 5: Images showing closing output for notes.jpg for different kernels

Note that the note heads are very dark. There are only a couple of lines with comparable intensity. Other than those, everything else in the image has a higher intensity (corresponding). Since the lines too have higher intensity then that heads, it is possible to threshold the image to a binary image so that only the note heads remain and everything else changes to white background. $(40, 255)$ achieves exactly that. The corresponding final output is:

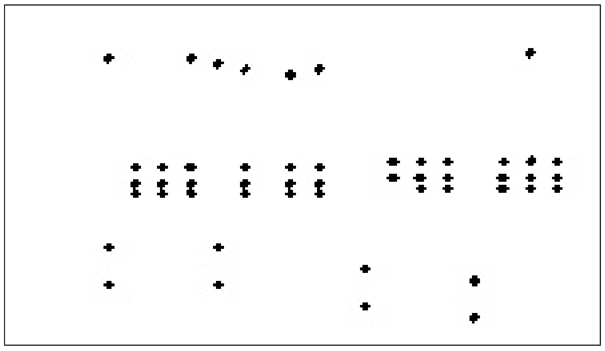
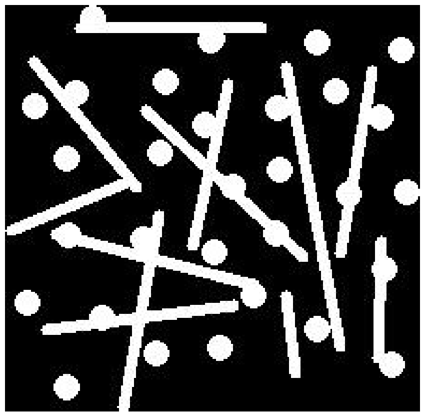


Figure 6: Final output of the algorithm (border added for clarity)

2 Counting Circles

2.1 Original Image



2.2 Algorithm

- Our algorithm is based on the simple fact that the size of circles is bigger than the size of the lines/rectangles present, i.e., if we take a circle with a sufficient enough radius, it will fit completely inside all of the circles, but it won't fit **completely** inside any of the lines/rectangles.
- Thus, keeping the above fact in mind, we perform opening using a disk kernel of radius 6. We perform opening since it is erosion followed by dilation. Although unlikely, but in some cases (images), it might happen that erosion leaves disconnected "circles" in the output. Opening prevents that from happening, since dilation is performed after erosion, filling any such "gaps" in the image.

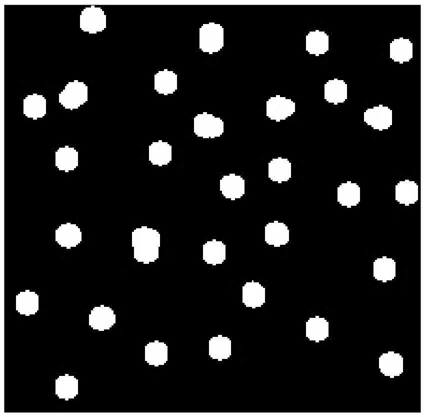


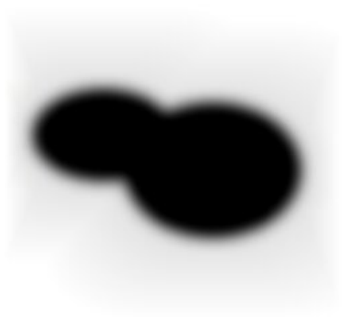
Figure 7: Output image after opening with disk kernel with radius 6

- After opening, we count the number of connected components in the output image. The answer to that comes out to be 31, which is # circles +1, since the background is a connected component too.

Thus, # circles = 30.

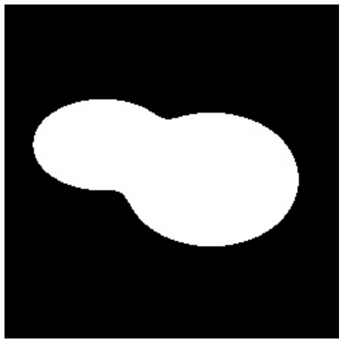
3 Segmenting Occluded Objects

3.1 Sample Image

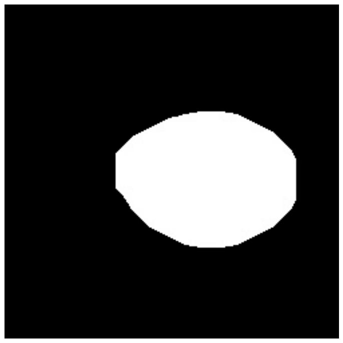


3.2 Algorithm

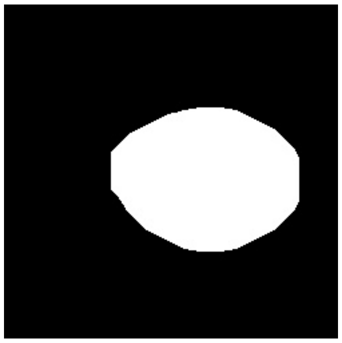
- The basic intuition of our algorithm stands on the observation that the two "disks" are of different sizes, hence we can separate them by using opening or erosion.
- First, we use thresholding operation with parameters (0, 255, Inverse Binary) to separate background and foreground



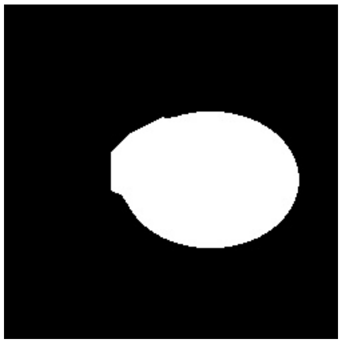
- Then, we perform opening with the disk kernel having radius 4 to remove the smaller circle.



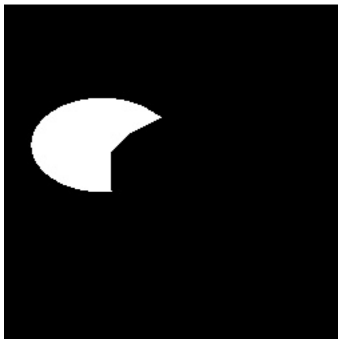
- Then, we increase the size of the obtained circle in order to "complete" it using dilation with a disk kernel of radius 4.



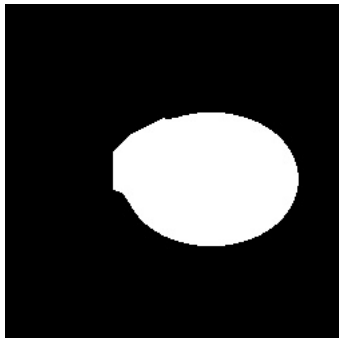
- But, note that this also increases the size of the circle in the portions that are not occluded. In order to limit this increase to only the occluded part, we take intersection of constructed image with the original image to get finer details of large circle in the parts that were not occluded.



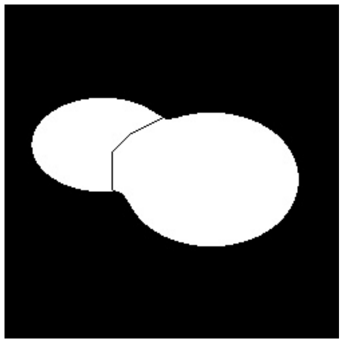
- Then we take image difference of the original image with the image constructed so far to segment out the smaller circle.



- Erosion of big circle with ellipse kernel having parameters (3 , 3) to decrease size of big circle so that the union of both the circles do not overlap



- Union of big circle with small circle to find the final image with separation line.

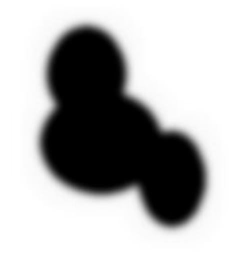


3.3 Failure

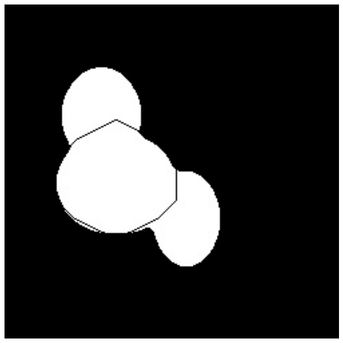
As mentioned above, our algorithm is based on the observation that both the objects(circles) are of different sizes. As a consequence, our algorithm might fail when the objects to be separated are of same size because in that case we would not be able to separate those objects by the opening operation.

3.4 Extra Credit: Additional Image

3.4.1 Sample Image



3.4.2 Output Image


(a) Output using our algorithm

3.4.3 Observation

We can see that our algorithm has performed satisfactorily to separate the three circles with the same parameters used with previous image. Since the size of the the middle circle and the 2 circles on the sides were different, our algorithm was successful in segmented the occluded objects.