Boundary Detection and Morphological Operators

Alexis Baudron

abe3897

**Installing all dependencies**

Ubuntu 16.04 Linux machine with python3.5 or +

$ python3 -m venv venv

$ unset PYTHONPATH

$ source venv/bin/activate

$ pip install -r requirements.txt

**Run the code**

Activate virtual environment ($ source venv/bin/activate)

$ python morph.py

**Algorithm**

Python implementation of the basic morphological operators covered in class as well as Boundary detection. There are 5 functions covered in the morph.py file

1. Erosion

2. Dilation

3. Closing

4. Opening

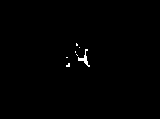
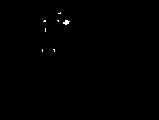
5. Boundary

1. Erosion

For the erosion operator we iterate through the rows and columns of the image. We define a sub-image that is the size of the kernel passed through to the function and a new array of zeros the size of the original image. If the kernel and the sub-image are the same, we set the corresponding center point of that array to 1 in the newly formed array of zeros mentioned above.

Once we are done iterating through the image we multiply our boolean array by 255 to obtained an image that we can visualize.

Here are the results with different kernel sizes:

Gun.bmp erosion with kernel size of ones of size 3 and 5

Palm.bmp erosion with kernel of ones of size 3 and 5

1. Dilation

For the dilation operator we iterate through the rows and columns of the image. We start by creating a copy of our original image. At each point we get the center of where the kernel is passing over on the image and compare that point that one. We then set the corresponding region size to ones that matches the kernel. Once we have iterated through the image we multiply by 255.



Gun.bmp dilation with kernel of ones of size 3, 5 and 7

Palm.bmp dilation with kernel of ones of size 3,5 and 7

1. Closing

Dilation followed by erosion with the same kernel

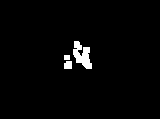
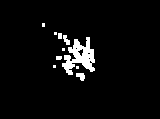
`

Gun.bmp with kernels of ones of size 3,5 and 7

Palm.bmp with kernels of ones of size 3,5 and 7

1. Opening

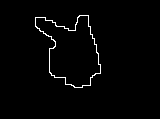
Erosion followed by dilation.

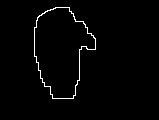


1. Boundary

Boundary detection algorithm, created a kernel of ones of size 3 that is used as the sliding window. A new image of zeros the size of the original image is also detected. Iterate through each pixel in the image and create a sub image of the same size as the kernel. If the center pixel of the sub image is equal to 255 and is not equal to the kernel, then we know it must be on the boundary of the of the image.

For the gun.bmp we started by dilating the image with a kernel of 9, eroding the image with a kernel of 11 and dilating again with a kernel of 9. We the processed the image through our boundary algorithm.



For the palm.bmp we did a closing operation with a kernel of 7.