Image Processing
lab 4

Klaas Kliffen Jan Kramer

January 8, 2016

Exercise 1 – Skeletons

The morphological skeleton of an image X with structuring element B is defined as:

$$SK(X) = \bigcup_{k=0}^{K} S_k(X)$$

where

$$S_k(X) = (X \ominus kB) - (X \ominus kB) \circ B$$

and $K \in \mathbb{N}$ such that

$$K = max\{k | (X \ominus kB) \neq \emptyset\}.$$

In addition $(X \ominus kB)$ is defined as follows:

$$(X \ominus kB) = \begin{cases} X & \text{if } k = 0 \\ ((\dots(X \ominus B) \ominus B) \ominus \dots) \ominus B) & \text{otherwise} \end{cases}$$

Note that in the second case $\ominus B$ is applied k times to X.

a. Let set $B = \hat{B} = \{0\}$ be given.

Claim. $SK(X) = \emptyset$

Proof. Note that for all $z \in \mathbb{Z}^2$ the following holds:

$$(B)_z = \{c | c = b + z, b \in B\} = \{c | c = z\} = \{z\}$$

Then we have by the definition of the erosion of X by B that

$$\begin{aligned} X \ominus B &= \{z | (B)_z \subseteq X\} \\ &= \{z | \{z\} \subseteq X\} \\ &= X. \end{aligned} \tag{1}$$

Similarly we have by the definition of the dilation of X by B that

$$\begin{aligned} X \oplus B &= \{z | (\hat{B})_z \cap X \neq \emptyset\} \\ &= \{z | \{z\} \cap X \neq \emptyset\} \\ &= X. \end{aligned} \tag{2}$$

The skeleton of image X with structuring element B is:

$$\begin{aligned} SK(X) &= \bigcup_{k=0}^{K} S_k(X) \\ &= \bigcup_{k=0}^{K} ((X \ominus kB) - (X \ominus kB) \circ B) \\ &= \bigcup_{k=0}^{K} ((X \ominus kB) - ((X \ominus kB) \ominus B) \oplus B) \end{aligned}$$

Now if we apply the definition of $(X \ominus kB)$ and the results of equation 1 and 2 we get:

$$\begin{aligned} SK(X) &= \bigcup_{k=0}^{K} ((X \ominus kB) - ((X \ominus kB) \ominus B) \oplus B) \\ &= \bigcup_{k=0}^{K} (X - X) \\ &= \bigcup_{k=0}^{K} \emptyset \\ &= \emptyset \end{aligned}$$

■

b. Let $X \ominus B = \emptyset$ be given.

Claim. *SK(X) = X*

Proof. Consider the definition of $(X \ominus kB)$:

$$(X \ominus kB) = \begin{cases} X & \text{if } k = 0 \\ ((\dots (X \ominus B) \ominus B) \ominus \dots) \ominus B) & \text{otherwise} \end{cases}$$

Note that $X \ominus B = \emptyset$. In addition for $X = \emptyset$ we have that every other erosion in the definition above is

$$X \ominus B = \emptyset \ominus B = \{z | (B)_z \subseteq \emptyset\} = \emptyset. \tag{3}$$

Hence the previous definition $X \ominus kB$ is equal to:

$$(X \ominus kB) = \begin{cases} X & \text{if } k = 0 \\ \emptyset & \text{otherwise} \end{cases} \tag{4}$$

Now the skeleton of image X with structuring element B is:

$$\begin{aligned}
SK(X) &= \bigcup_{k=0}^{K} S_k(X) \\
&= \bigcup_{k=0}^{K} ((X \ominus kB) - (X \ominus kB) \circ B) \\
&= \bigcup_{k=0}^{K} ((X \ominus kB) - ((X \ominus kB) \ominus B) \oplus B) \\
&= ((X \ominus 0B) - ((X \ominus 0B) \ominus B) \oplus B) \cup \bigcup_{k=1}^{K} ((X \ominus kB) - ((X \ominus kB) \ominus B) \oplus B)
\end{aligned}$$

If we use the equation 3, 4 and the definition of $X \oplus B$ we get:

$$\begin{aligned}
SK(X) &= (X - (X \ominus B) \oplus B) \cup \bigcup_{k=1}^{K} (\emptyset - (\emptyset \ominus B) \oplus B) \\
&= (X - (\emptyset \oplus B)) \cup \bigcup_{k=1}^{K} (\emptyset - (\emptyset \oplus B)) \\
&= \left(X - \{z|(\hat{B})_z \cap \emptyset \neq \emptyset\}\right) \cup \bigcup_{k=1}^{K} \left(\emptyset - \{z|(\hat{B})_z \cap \emptyset \neq \emptyset\}\right) \\
&= (X - \{z|\emptyset \neq \emptyset\}) \cup \bigcup_{k=1}^{K} (\emptyset - \{z|\emptyset \neq \emptyset\}) \\
&= (X - \emptyset) \cup \bigcup_{k=1}^{K} (\emptyset - \emptyset) \\
&= X \cup \bigcup_{k=1}^{K} \emptyset \\
&= X \cup \emptyset \\
&= X
\end{aligned}$$

■

c. The implementation of IPskeletondecomp is rather trivial once one has the basic morphologic functions implemented. The only non-trivial thing is the subtraction in $S_k(X)$. Since the second term is always smaller than the first one, because of the extra erosion, the xor function can be used to get the difference. Then it is just implementing the definition:

```
1  function skfr = IPskeletondecomp (x, b)
2
3    % create the output matrix
4    skfr = zeros(size(x), 'uint8');
5
6    k = 1;
```

```
  cur = x;
  while any(any(cur)) == 1
    next = IPerode(cur, b);
    % S_k(), since next dilate b is smaller than cur
    skel = xor(cur, IPdilate(next, b));
    skfr(skel) = k;
    cur = next;
    k++;
  end
end
```

The implementation of dilation is also rather simple, because the 2 dimensional convolution function can be used:

```
function di = IPdilate (x, b)
  bhat = flip(flip(b, 1), 2);
  di = conv2(x, bhat, "same") > 0;
end
```

The implementation of erosion can than be based on the one of dilation, because of their duality. Note however that the image has to be padded so that this works:

```
function er = IPerode (x, b)
  xd1 = size(x, 1);
  xd2 = size(x, 2);
  bd1 = size(b, 1);
  bd2 = size(b, 2);
  % pad the image
  a = zeros(size(x) + 2 * size(b));
  a(bd1+1:(bd1 + xd1), bd2+1:(bd2 + xd2)) = x;
  % based on (A erode B)^c = A^c dilate B^hat
  a = ~(IPdilate(~a, flip(flip(b, 1), 2)));
  % unpad
  er = a(bd1+1:(bd1 + xd1), bd2+1:(bd2 + xd2));
end
```

d. Basically the reconstruction works by taking the union of dilating the skeleton parts the correct amount. This can be done as follows:

```
function re = IPskeletonrecon (x, b)

  % create the output matrix
  re = zeros(size(x), 'logical');

  re(x == 1) = 1;
  % extract the skeleton subset counts
  maxk = max(x(x > 0));
  if maxk > 1
    for k = 2:maxk
      tmp = zeros(size(x), 'logical');
      tmp(x == k) = 1;
      for i = 1:(k - 1)
        tmp = IPdilate(tmp, b);
      end
      re = or(re, tmp);
```

```
    end
  end
end
```

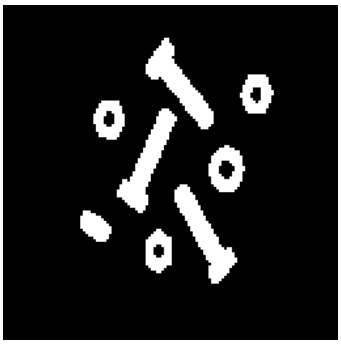
e. To demonstrate that the decomposition and the reconstruction works, we wrote the following script to compare the results:

```
x = imread('../images/nutsbolts.tif');
b = ~~[1 1 1; 1 1 1; 1 1 1];
skel = IPskeletondecomp(x, b);
recon = IPskeletonrecon(skel, b);
imwrite(recon, 'recon.png')
% check to test if any logical element differs, if equal the result
    should be 0
any(any(xor(recon, x)))
```

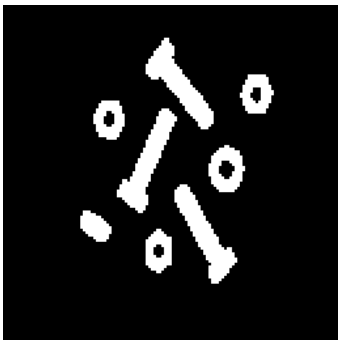
The result of running this script is as follows:

```
ans = 0.
```

This means that the original and the resulting image are identical. In addition these images can be seen in Figure 1.



Original image



The result of decomposition and reconstruction

Figure 1: The nutsbolts image and its skeleton decomposition reconstruction.

Exercise 2 – Grey-scale morphology

a. Based on the equation 9.2-1 in the book, the implementation of IPgerode is rather straightforward:

```
function y = IPgdilate (x, b)
#code based on IPfilter of lab 1 ex 3

# cast to double so nan exists
x = double(x);

#get image dimensions
nr = size(x, 1);
nc = size(x, 2);

```

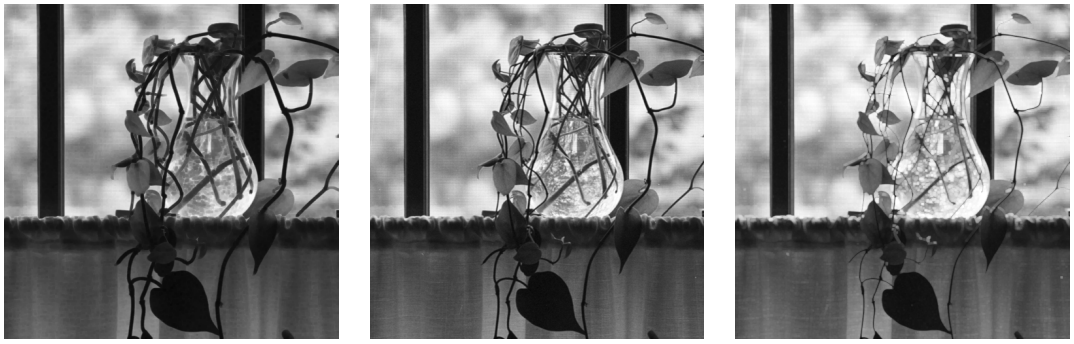
```
#now the shifted matrices can be computed:
x_u = [x(2:nr, :); NaN([1 nc])];
x_d = [NaN([1 nc]); x(1:(nr - 1), :)];
x_l = [x(:, 2:nc) NaN([nr 1])];
x_r = [NaN([nr 1]) x(:, 1:(nc - 1))];
x_ul = [x_u(:, 2:nc) NaN([nr 1])];
x_ur = [NaN([nr 1]) x_u(:, 1:(nc - 1))];
x_dl = [x_d(:, 2:nc) NaN([nr 1])];
x_dr = [NaN([nr 1]) x_d(:, 1:(nc - 1))];

#create a new structuring element with 0's for true and NaN for false
c = zeros(size(b));
c(~b) = nan;

#finally the value can be calculated (NaN's are ignored) by taking
    pairwise maxima of matrices
y = max(max(max(max(max(max(max(
          c(1, 1) + x_dr,  c(1, 2) + x_d), c(1, 3) + x_dl), ...
          c(2, 1) + x_r),   c(2, 2) + x),   c(2, 3) + x_l),  ...
          c(3, 1) + x_ur),  c(3, 2) + x_u), c(3, 3) + x_ul);

# cast the image back to uint8
y = uint8(y);
end
```

The only trick used here is to use the NaN value to ignore pixels that are outside the picture or not part of the structuring element. This works because $min(NaN, 3) = 3$ and $max(NaN, 3) = 3$. The only problem is that NaN does not exist for uint8's, therefore the image has to be converted to double before the calculations and back afterwards. Also note that maximum/minimum of the elements of the matrices is calculated by taking the the pairwise maxima/minima. The implementation of IPgdilate is based on equation 9.2-2 in the book and the implementation of IPgerode:

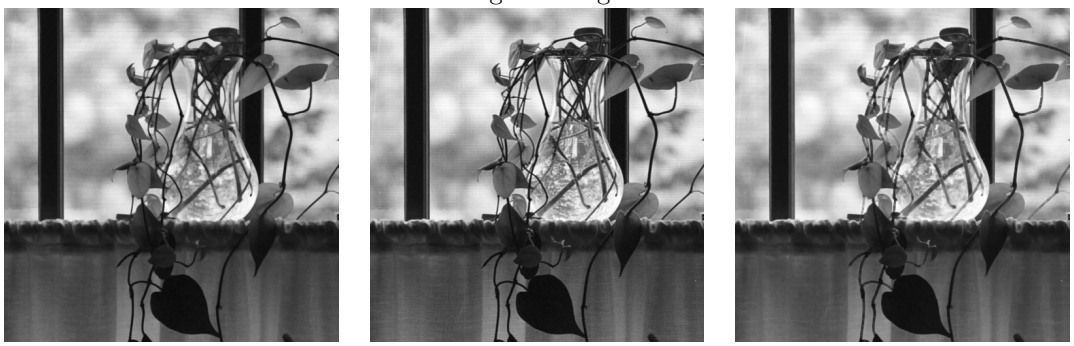
```
function y = IPgerode (x, b)
#code based on IPfilter of lab 1 ex 3

# cast to double so nan exists
x = double(x);

#get image dimensions
nr = size(x, 1);
nc = size(x, 2);

#now the shifted matrices can be computed:
x_u = [x(2:nr, :); NaN([1 nc])];
x_d = [NaN([1 nc]); x(1:(nr - 1), :)];
x_l = [x(:, 2:nc) NaN([nr 1])];
x_r = [NaN([nr 1]) x(:, 1:(nc - 1))];
x_ul = [x_u(:, 2:nc) NaN([nr 1])];
x_ur = [NaN([nr 1]) x_u(:, 1:(nc - 1))];
x_dl = [x_d(:, 2:nc) NaN([nr 1])];
x_dr = [NaN([nr 1]) x_d(:, 1:(nc - 1))];

#create a new structuring element with 0's for true and NaN for false
```

```
c = zeros(size(b));
c(~b) = nan;

#finally the value can be calculated (NaN's are ignored) by taking
    pairwise minima of matrices
y = min(min(min(min(min(min(min(min(
          c(1, 1) + x_dr,  c(1, 2) + x_d), c(1, 3) + x_dl), ...
          c(2, 1) + x_r),   c(2, 2) + x),   c(2, 3) + x_l),  ...
          c(3, 1) + x_ur),  c(3, 2) + x_u), c(3, 3) + x_ul);

# cast the image back to uint8
y = uint8(y);
end
```

b. The resulting images are shown in figure 2. Like mentioned in example 9.9 of the book the erosion increases the size and intensity of dark features, while decreasing the bright ones. For example the branches of the plant are darker and bigger, while the water in the vase is less bright. This is the result of the minimum operation in the definition of gray-scale erosion. Similarly that example in the dilation bright features are increased in size and intensity, while decreasing the size and intensity of dark features. In this case the branches are thinner and smaller than in the original and the water in the vase is brighter. Also note that smaller features may be more visible by erosion or dilation depending on if they are dark respectively bright. In addition example 9.10 in the book discusses gray-scale openings and closings. The opening decreases the intensity of bright features smaller than the structuring element, while keeping the dark ones relative the same. Note for example the decreased intensity of the spots in the vase. The closing decreased the intensity of dark features smaller than the structuring element, while keeping the bright ones. Again the dots in the vase show this.



The erosion | Original image | The dilation



The opening | Original image | The closing

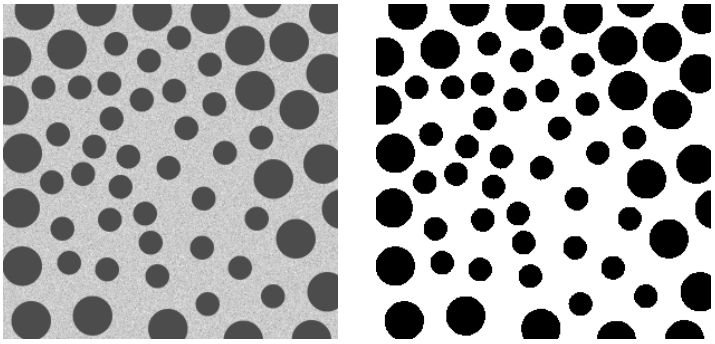
Figure 2: The grayscale dilation, erosion, opening and closing of vase.tif with a flat box structuring element.

Exercise 3 – Classification

To suppress the small disks in the image, they need to be eroded. By choosing a structuring element larger than the small disks, only the large disks will be left. However, the large disks are eroded and need to be reconstructed. This can be done by dilating the eroded image with a similar structuring element. To fully reconstruct the old disks, the original image and the dilated image can be considered a set. By taking the union of these two sets, the large disks will be restored, while the small disks will disappear. Since the possible overflow of the dilated image will be suppressed by the original image and the small disks are suppressed by the dilated image.

To be able to use set operations, the image is first converted to binary image with a foreground and background by a certain threshold. This will also suppress all noise in the image and only the disks will be shown as can be seen in figure 3.

```
% convert the image to a binary image using threshold
function retval = makebinary(img, threshold)
  schema = ones(size(img))*threshold;
  retval = img > threshold;
end
```



The original image The binary thresholded image

Figure 3: The original image is being converted to a binary image

To suppress the small disks an erosion is used. This is implemented by using a 2D convolution with the structuring element. This element is generated by creating a square matrix of ones and a circular element of zeros with a given radius. Zeros are being used, since the disks are also considered background. Since both images are binary, this will be the same as performing the opening operation. The result of the convolution can be seen in figure 4. The small disks have been removed by the convolution.

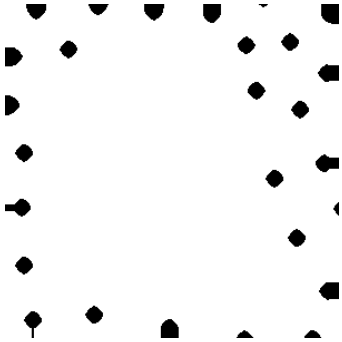


Figure 4: The erosion convolution of figure 3 with a circular structuring element of radius 8

```
% make a circular blob
function retval = makeblob(radius)
  out = ones(2*radius+1);
  for i=1 : (2*radius+1)
    for j=1 : (2*radius+1)
      if (sqrt(i*i+j*j) < 1.0)
        out(i,j) = 0;
      end
    end
  end
  retval = out;
end
```

Next the large disks have to be reconstructed. This is done by eroding the background instead of the foreground. Therefore the complement of figure 4 is taken for the next convolution step. A large structuring element is chosen, so to be sure that all pixels from the large disks in figure 3 are covered. Normally the and operation would be used to restore the final images, but since both images are actually the complement since the disks consist of 0 values, the or operation is used. This is derived by using De Morgan's laws for converting the and of two negated operand to the or operation.

Furthermore, using a too large structuring element will restore some parts of the small disks. Therefore the dilation is done in two steps: one with a circular structuring with radius 11 and one with radius 4. The resulting image can be seen in figure 5.

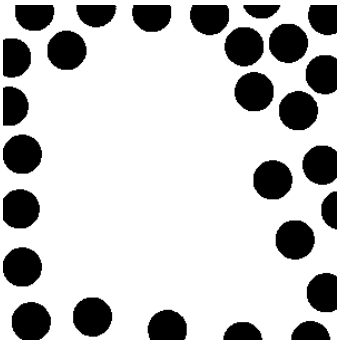
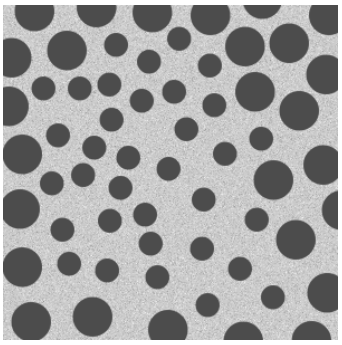
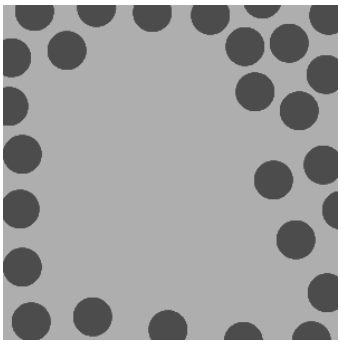


Figure 5: The dilation convolution of figure 4 with a circular structuring element of radius 11 and 4

To fully restore the original values of the large disks, the complement of figure 5 is element wise multiplied by the original image as seen on the left of figure 3. This will remove all the small disks, but also the noisy background. To somewhat represent the contrast of the original image, the mean value of the noise is used as the background which was deleted by thresholding in the first step. The final image can be seen in figure 6



The original image



The final image with the small disks and noise removed

Figure 6: A comparison of the original image with the final image

```
x = imread('../images/blobs.tif');
mask = makebinary(x,128);
imwrite(mask,'binarymask.png');

% Calculate the average background noise
background = x .* mask;
background2 = 256 - background;
background2 = background2 .* mask;
background2 = 256 - background2;
meannoise = mean([mean(mean(background)) mean(mean(background2)) ]);

% structuring element
blob = makeblob(8);
blob2 = makeblob(11);
blob3 = makeblob(4);

% erode the circles (all circles radius < 8 will be removed)
eroded = makebinary(conv2(mask,blob,"same"),1);
imwrite(eroded,'convolution.png');
% take the complement as eroding background = dilating foreground
erodedcomp = not(eroded);
% erode the background two times using different SE
firstdilate = not(makebinary(conv2(erodedcomp,blob2,"same"),1));
% or used, since the background uses 1 and foreground 0
firstdilate = or(firstdilate,  mask);
secondilate = not(makebinary(conv2(not(firstdilate),blob3,"same"),1));
secondilate = or(secondilate,  mask);
imwrite(secondilate,'restored.png');

% generate noise image component
noise = uint8(ones(size(x)) .* secondilate * meannoise);

restoredimage = not(secondilate) .* x + noise;
imwrite(restoredimage,'final.png');
```

The downside of using this method, is that the mask used is binary. Therefore, the edges of the large disks are quite sharp. This can be solved by applying a weak blurring filter. Also, the removed noise might be considered a downside of this method, since the visual contrast of the image is decreased. However, for computer vision application such as feature detection it might be a benefit.

Task distribution

ex1	design	implementation	answers questions	writing report
Klaas	20%	10%	20%	40%
Jan	80%	90%	80%	60%

ex2	design	implementation	answers questions	writing report
Klaas	50%	20%	25%	25%
Jan	50%	80%	75%	75%

ex3	design	implementation	answers questions	writing report
Klaas	80%	75%	75%	80%
Jan	20%	25%	25%	20%