Placement of polygonal shapes in a 2D image

Gilberto Galvis, MeridaTech

This document shows the operation of a MATLAB function called: polygon_shape_placement(). This function creates binary images with polynomial shapes placed on it. The interesting thing is that the center-to-center distance between the polygons and its sizes are variables and varies following a pseudonormal distribution. In the first section we will give a detailed review of the function where we will explain its input and output arguments. Then in the subsequent sections we will give examples and results making variation of parameters to verify the functioning of such function

Implementation Details

```
%--------------------------------------------------------------------------%
%---------------------- polygon shape placement function ------------------%
%                                                                          %
% im = polygon_shape_placement (imsize, polyshape, distparams, N, polyrotation)%
% creates the 2D binary image im of size imsize with the polinogonal shapes    %
% specified by polyshape placed along it. The size of the polygonal shapes va- %
% ries according to a normal distribution defined by the distribution parame-  %
% ters given by the polyshape input argument.The center-to-center distance     %
% along the x and y directions between the polygonal shapes follows a pseudo-  %
% normal distribution defined by distparams. The im image has the option to be %
% an image composed of N binary layers. Additionally, with the argument poly-  %
% rotation you can make random rotations on the polygonal shapes.              %
%                                                                              %
%  -imsize:       Vector of two elements specifying the dimensions of the 2D   %
%                 output image, im                                             %
%  >> imsize = [m, n]  % im of size m x n                                      %
%                                                                              %
%  -polyshape:    Array of two cells defining the pattern of the polygonal     %
%                 shape. The first element is a string specifying the type of  %
%                 shape desired (see the table below to know the types of sha- %
%                 pes available). The second element is a vector of two ele-   %
%                 ments specifying the mean & standar deviation of a pseudo-   %
%                 normal distribution used to define the size of each polygon. %
%  >> polyshape = {'square', [meansize, stdsize]}                              %
%                                    squares of a x a where 'a'                %
%                                    follows a ~ N(meansize, stdsize)          %
%                                                                              %
%  -distparams:   Vector of two elements specifying the mean & standar devia-  %
%                 tion error of a pseudo-normal distribution used to define    %
%                 the center-to-center distance                                %
%  >> distparams = [mean, std]                                                 %
%                                                                              %
%  -N:            Scalar defining the number of layers of the binary im image. %
%                 Each layer represents a subsample of the resulting image     %
%  >> N = 2  % Two layers                                                      %
%                                                                              %
%  -polyrotation  By default this variable is 'on'. You can set it to 'off' to %
```

```
%                deactivate the rotations.                                  %
%  >> polyrotation = 'off'  % deactivate the rotation                        %
%                                                                            %
%  Table: polyshape options                                                  %
%  +------------+-------------+--------------+---------+-----------------+  %
%  | shape      |  long form  |  short form  |  size   |  description    |  %
%  +------------+-------------+--------------+---------+-----------------+  %
%  | general    |             |  '4', '5'... | [d]     | d: diameter*    |  %
%  | polygon    |             |  '6', '7'... | [n]     | n: corners**    |  %
%  +------------+-------------+--------------+---------+-----------------+  %
%  | square     | 'square'    |  's'         | [a]     | a: side         |  %
%  +------------+-------------+--------------+---------+-----------------+  %
%  | rectangle  | 'rectangle' |  'r'         | [a, b]  | a: base         |  %
%  |            |             |              |         | b: lateral side |  %
%  +------------+-------------+--------------+---------+-----------------+  %
%  | circle     | 'circle'    |  'c'         | [d]     | r: diameter     |  %
%  +------------+-------------+--------------+---------+-----------------+  %
%  | triangle   | 'triangle'  |  '^'         | [b, h]  | b: base         |  %
%  |            |             |              |         | h: height       |  %
%  +------------+-------------+--------------+---------+-----------------+  %
%  | triangle   |             |  'v'         | [b, h]  | invertided      |  %
%  +------------+-------------+--------------+---------+-----------------+  %
%  | triangle   |             |  '<'         | [b, h]  | left orient..   |  %
%  +------------+-------------+--------------+---------+-----------------+  %
%  | triangle   |             |  '>'         | [b, h]  | right orient..  |  %
%  +------------+-------------+--------------+---------+-----------------+  %
%                                                                            %
%  *  The general polygons are created by means of a circumference, in which  %
%        the polygons are circumscribed. So, the diameter d is the diameter of %
%        the circumference                                                   %
%  ** n is the number of corners you want the polygon to have. This value is  %
%        taken directly from the numeric character that is placed as input to %
%        the general polygon option                                          %
%                                                                            %
%                                                                            %
%  NOTE: Now all the size parameters of the polygonal shapes (a, b, d and h)  %
%        will be defined by the distribution parameters specified in polyshape %
%                                                                            %
%----------------------------------------------------------------------------%
```

First example of usage

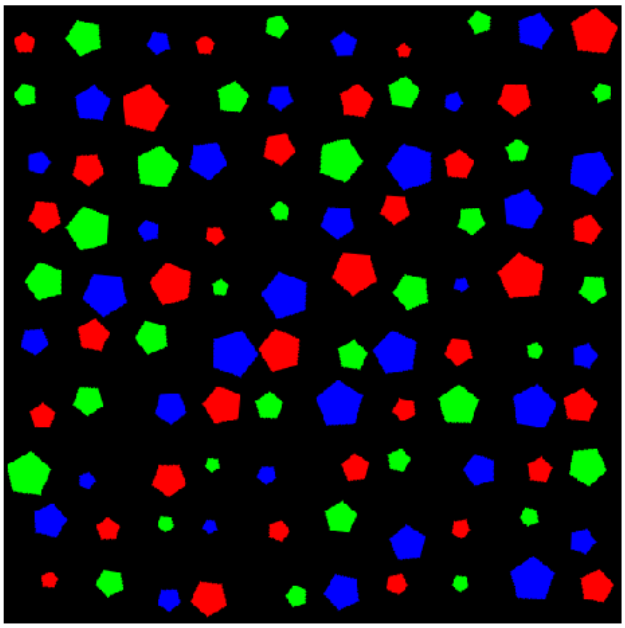
We will see how, in effect, the output image places the polygonal shapes according to a pseudo-normal distribution between the center-to-center distances. In addition, we will see how the size of the polygons also varies according to a pseudonormal distribution. For this purpose, we establish the following parameters:

- Size of 400×400 pixels for the desired output image.
- Pitch distribution of $mean = 40$ pixels with a standard deviation error of $std = 10$ pixels.
- Square as polynomial shape with size distribution defined by $mean = 20$ pixels and $std = 10$ pixels.

- N = 3. We purposely set this value to take advantage of the fact that the imshow operator sees the output image as an RGB image. This will show us the three superimposed layers, each with a different color: 1st layer (red), 2nd layer (green) and 3th layer (blue)

Then, we obtain

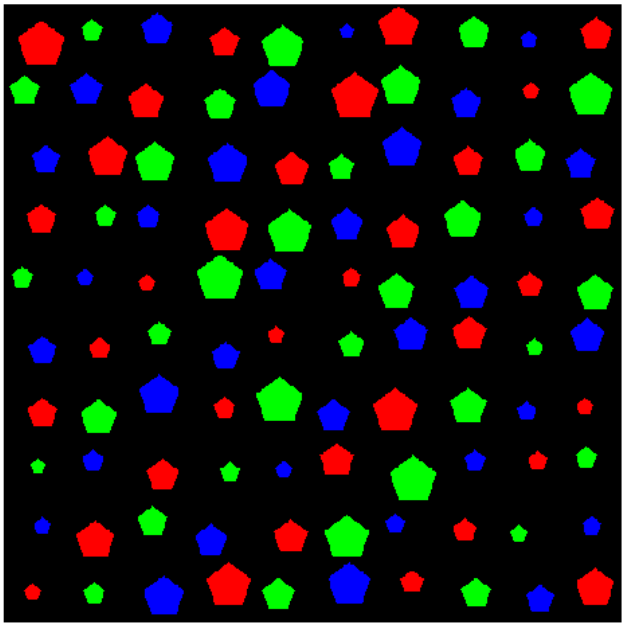
```
%--- Input parameter initializations
    imsize     = [550, 550];
    polyshape = {'5', [30,15]};
    distparams = [55, 25];
    N = 3;
    polyrotation = 'on';

%--- Run the function
    im = polygon_shape_placement(imsize, polyshape, distparams, N, polyrotation);

%--- Shown the output image
    figure;
    imshow(im)
    set(gcf, 'units','normalized', 'position', [0 0 .7 .7])
```



You can also disable the rotation option of the polygons. You can do this by setting the argument from entrana `polyrotation` to `'off'` (as a string of characters). In that case, the polygonal shapes do not rotate randomly as you can see below

```
    %--- Input parameter initializations
    imsize     = [550, 550];
    polyshape = {'5', [30,15]};
    distparams = [55, 20];
```

```
    N = 3;
    polyrotation = 'off';

%--- Run the function
    im = polygon_shape_placement(imsize, polyshape, distparams, N, polyrotation);

%--- Shown the output image
    figure;
    imshow(im)
    set(gcf, 'units','normalized', 'position', [0 0 .7 .7])
```



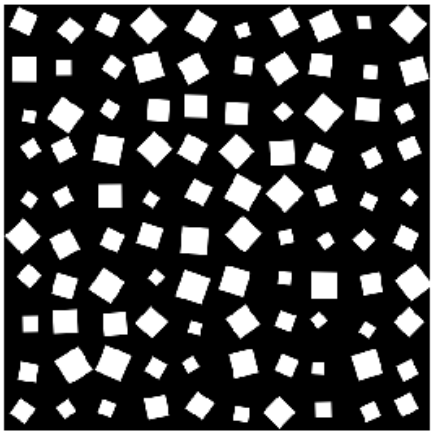
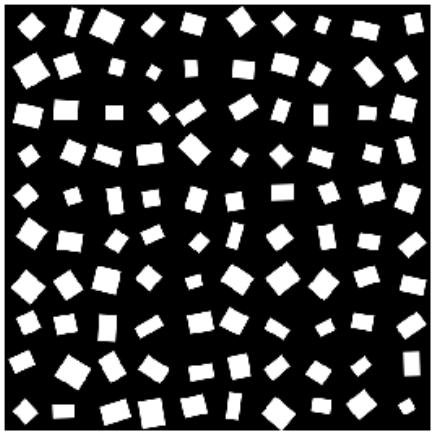
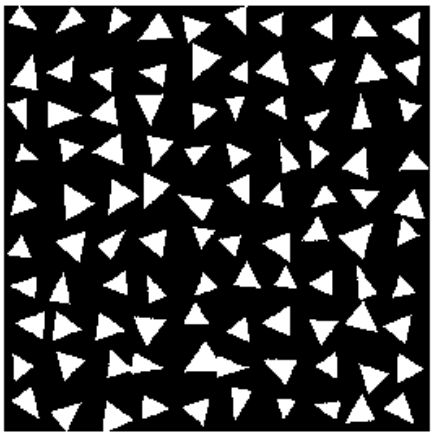
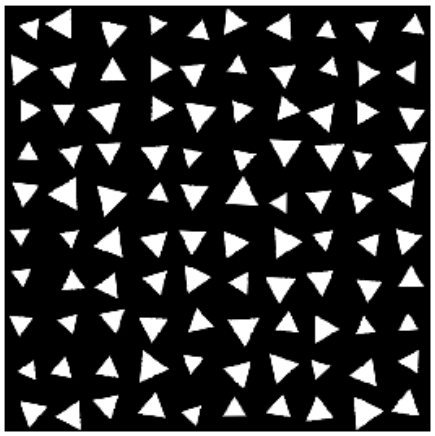
Now that we have seen that the output image is created for different layers and with pitch distribution, we will work with N = 1 for simplicity.

Varying the polygonal shapes

Here, keeping the same parameters above, we proceed to vary the polygonal shapes, that is, the first cell of the input argument `polyshape`

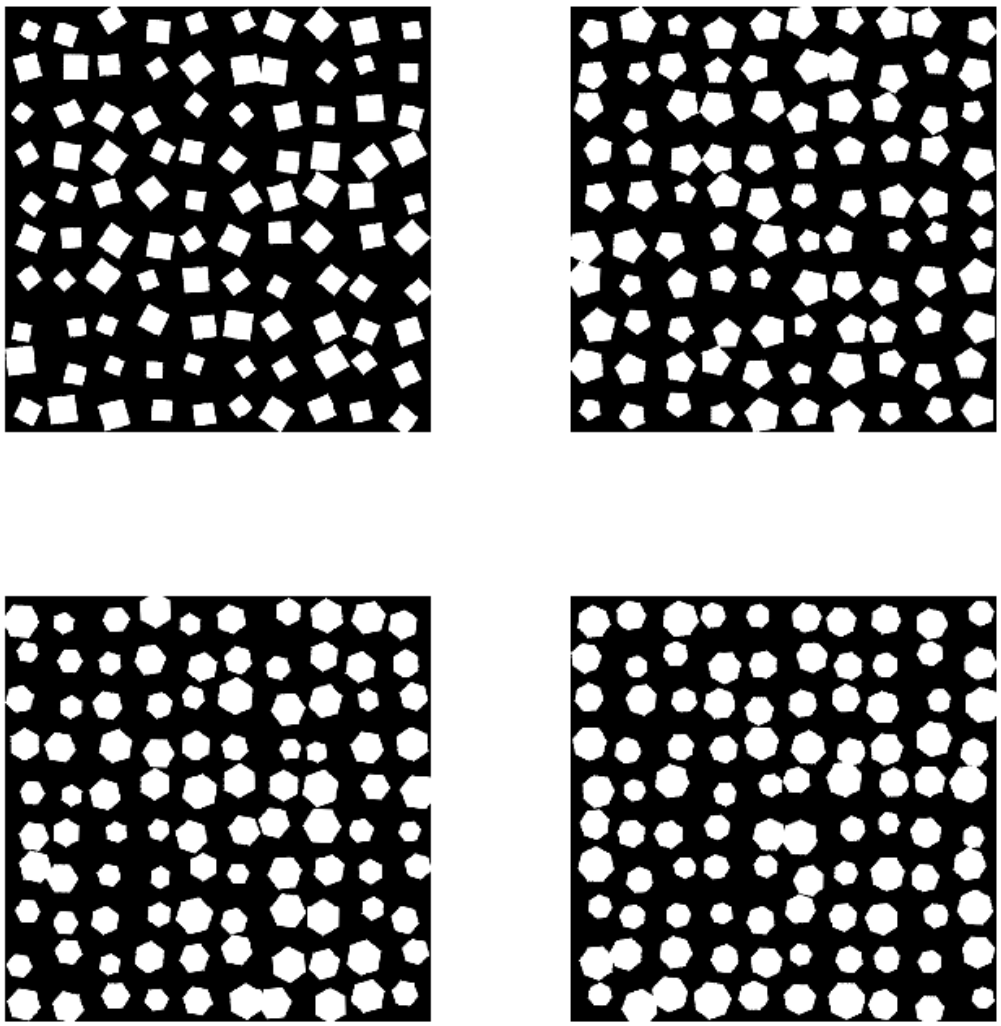
```
%--- Input parameter initializations
    imsize     = [550, 550];
    distparams = [55, 20];
    N = 1;
    polyrotation = 'on';

%--- Run the function for squares
    polyshape  = {'s', [25, 10]};
    im1 = polygon_shape_placement(imsize, polyshape, distparams, N, polyrotation);
```

```
%--- Run the function for rectangles
    polyshape  = {'r', [25, 10]};
    im2 = polygon_shape_placement(imsize, polyshape, distparams, N, polyrotation);

%--- Run the function for triangles with b non equal to h
    polyshape  = {'^', [35, 10]};
    im3 = polygon_shape_placement(imsize, polyshape, distparams, N, polyrotation);

%--- Run the function for triengles with b equal to h
    polyshape  = {'3', [40, 10]};
    im4 = polygon_shape_placement(imsize, polyshape, distparams, N, polyrotation);

%--- Show the resulting images
    figure;
    subplot(2,2,1); imshow(im1); subplot(2,2,2); imshow(im2);
    subplot(2,2,3); imshow(im3); subplot(2,2,4); imshow(im4);
    set(gcf, 'units','normalized', 'position', [0 0 .58 1])
```

```
%--- Input parameter initializations
    distparams = [55, 20];

%--- Run the function for polygonal shapes with 4 corners (diamond case)
    polyshape  = {'4', [40, 10]};
    im1 = polygon_shape_placement(imsize, polyshape, distparams, N, polyrotation);

%--- Run the function for polygonal shapes with 5 corners (pentagonal case)
    polyshape  = {'5', [40, 10]};
    im2 = polygon_shape_placement(imsize, polyshape, distparams, N, polyrotation);

%--- Run the function for polygonal shapes with 6 corners (hexagonal case)
    polyshape  = {'6', [40, 10]};
    im3 = polygon_shape_placement(imsize, polyshape, distparams, N, polyrotation);
```

```
%--- Run the function for polygonal shapes with 7 corners (heptagonal case)
    polyshape  = {'7', [40, 10]};
    im4 = polygon_shape_placement(imsize, polyshape, distparams, N, polyrotation);

%--- Show the resulting images
    figure;
    subplot(2,2,1); imshow(im1); subplot(2,2,2); imshow(im2);
    subplot(2,2,3); imshow(im3); subplot(2,2,4); imshow(im4);
    set(gcf, 'units','normalized', 'position', [0 0 .58 1])
```

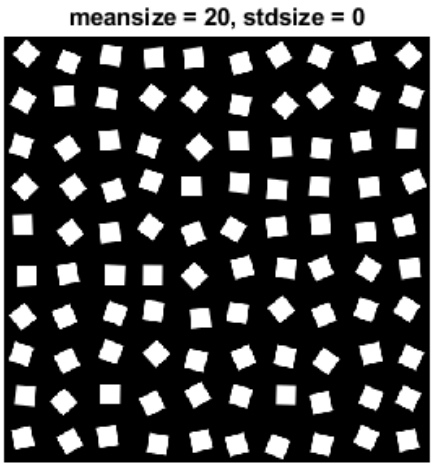


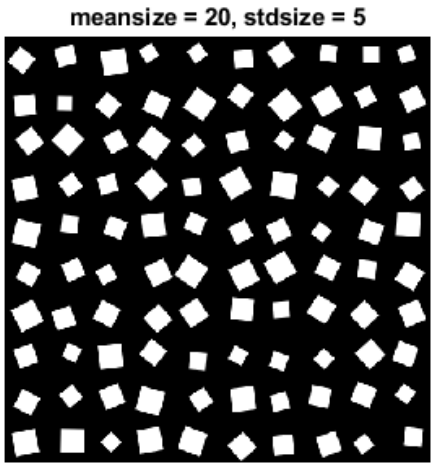
Varying polygonal size distribution for $0 \leq std \leq 15$ **with** $mean = 20$ **(units is a pixel)**

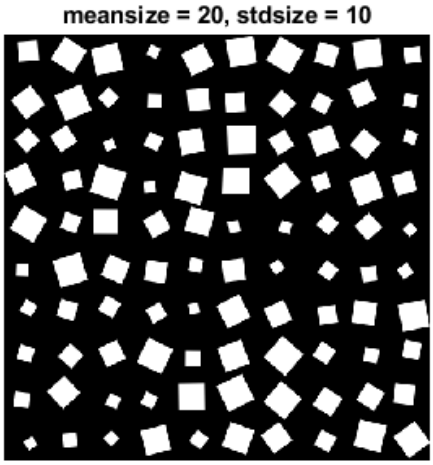
Here we have also defined:

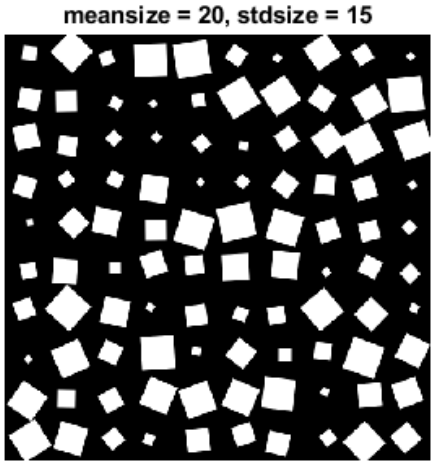
- Size of 450×450 pixels for the desired output image
- Pitch distribution of $mean = 45$ pixels with a standard deviation error of $std = 10$ pixels.
- Circle as polynomial shape

```
%--- Input parameter initializations
    imsize     = [450, 450];
    distparams = [45, 10];
    N = 1;
    polyrotation = 'on';

%--- Run the function for stdsize = 0
    polyshape  = {'s', [20, 0]};
    im0 = polygon_shape_placement(imsize, polyshape, distparams, N, polyrotation);

%--- Run the function for stdsize = 5
    polyshape  = {'s', [20, 5]};
    im5 = polygon_shape_placement(imsize, polyshape, distparams, N, polyrotation);

%--- Run the function for stdsize = 10
    polyshape  = {'s', [20, 10]};
    im10 = polygon_shape_placement(imsize, polyshape, distparams, N, polyrotation);

%--- Run the function for stdsize = 15
    polyshape  = {'s', [20, 15]};
    im15 = polygon_shape_placement(imsize, polyshape, distparams, N, polyrotation);

%--- Show the resulting images
    figure; subplot(2,2,1); imshow(im0);
    title('meansize = 20, stdsize = 0')
    subplot(2,2,2); imshow(im5);
    title('meansize = 20, stdsize = 5')
    subplot(2,2,3); imshow(im10);
    title('meansize = 20, stdsize = 10')
    subplot(2,2,4); imshow(im15);
    title('meansize = 20, stdsize = 15')
    set(gcf, 'units','normalized', 'position', [0 0 .58 1])
```









Varying polygonal pitch distribution for $0 \leq std \leq 15$ with $mean = 45$ (units is a pixel)

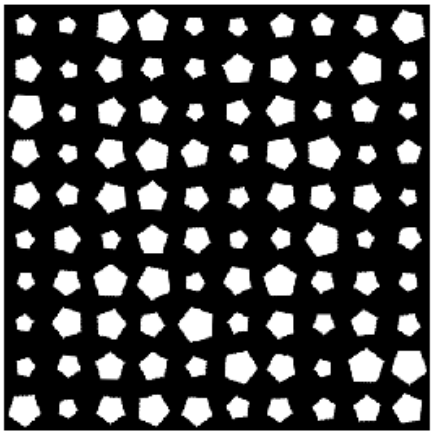
Here we have also defined:

- Size of 450×450 pixels for the desired output image
- Circle as polynomial shape with size distribution defined by $mean = 30$ pixels and $std = 10$ pixels.

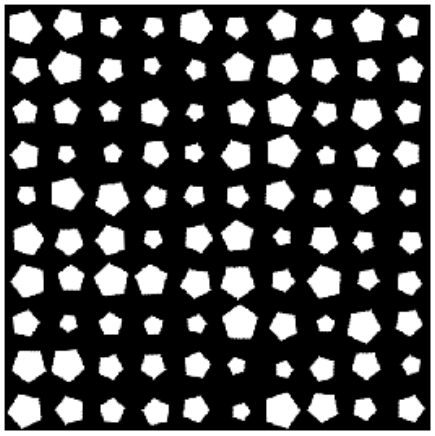
```
%--- Input parameter initializations
    imsize     = [450, 450];
    polyshape  = {'5', [30, 10]};
    N = 1;
```

```
%--- Run the function for std = 0
    distparams = [45, 0];
    im0 = polygon_shape_placement(imsize, polyshape, distparams, N);

%--- Run the function for std = 5
    distparams = [45, 5];
    im5 = polygon_shape_placement(imsize, polyshape, distparams, N);

%--- Run the function for std = 10
    distparams = [45, 10];
    im10 = polygon_shape_placement(imsize, polyshape, distparams, N);

%--- Run the function for std = 15
    distparams = [45, 15];
    im15 = polygon_shape_placement(imsize, polyshape, distparams, N);

%--- Show the resulting images
    figure; subplot(2,2,1); imshow(im0);
    title('mean = 45 and std = 0')
    subplot(2,2,2); imshow(im5);
    title('mean = 45 and std = 5')
    subplot(2,2,3); imshow(im10);
    title('mean = 45 and std = 10')
    subplot(2,2,4); imshow(im15);
    title('mean = 45 and std = 15')
    set(gcf, 'units','normalized', 'position', [0 0 .58 1])
```

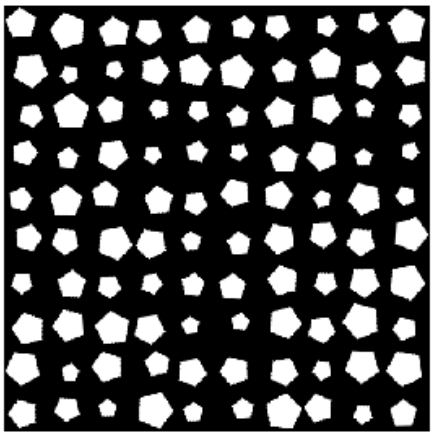
mean = 45 and std = 0



mean = 45 and std = 5



mean = 45 and std = 10



mean = 45 and std = 15