Detecting and Counting Objects on a Known Background: Background Subtraction

In cases where the background image can be obtained beforehand, objects placed on that background can be easily detected through a process called background subtraction. Background subtraction is a commonly used method for generating a binary foreground mask corresponding to the pixels that are different from the background. For example, consider the following scenario: we place a static camera pointed at an unoccupied stage and take a snapshot, then at a later time, we take another snapshot while actors are performing on the same stage. We can detect the location and count the number of performers by simply looking at the difference between the two images. Needless to say, this method assumes that the lighting conditions are similar in both snapshots.

Preliminaries

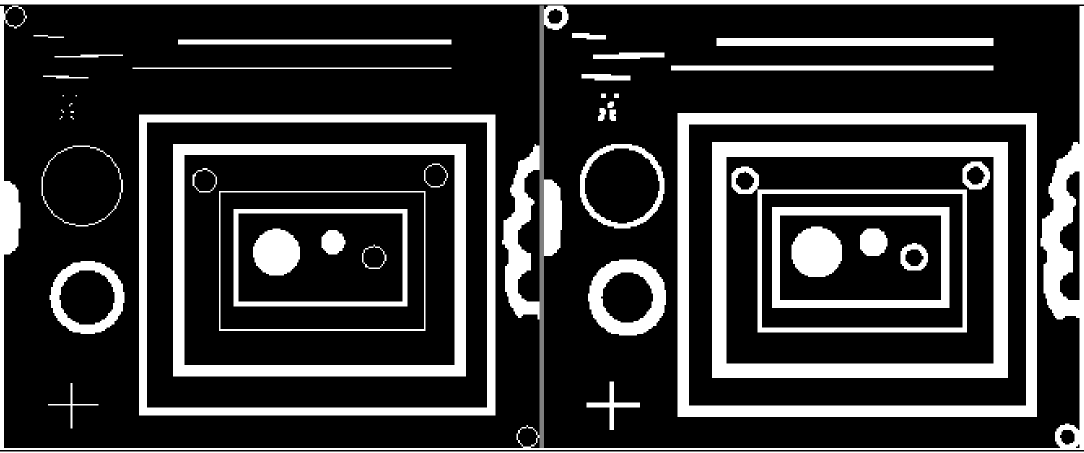
Morphological Dilation and Erosion (**MATLAB Documentation**)

The most basic morphological operations are dilation and erosion. Dilation adds pixels to the boundaries of objects in an image, while erosion removes pixels on object boundaries. The number of pixels added or removed from the objects in an image depends on the size and shape of the structuring element used to process the image. In the morphological dilation and erosion operations, the state of any given pixel in the output image is determined by applying a rule to the corresponding pixel and its neighbors in the input image. The rule used to process the pixels defines the operation as a dilation or an erosion.

Dilation
The value of the output pixel is the maximum value of all pixels in the neighborhood. In a binary image, a pixel is set to 1 if any of the neighboring pixels have the value 1.

Morphological dilation makes objects more visible and fills in small holes in objects. Lines appear thicker, and filled shapes appear larger.

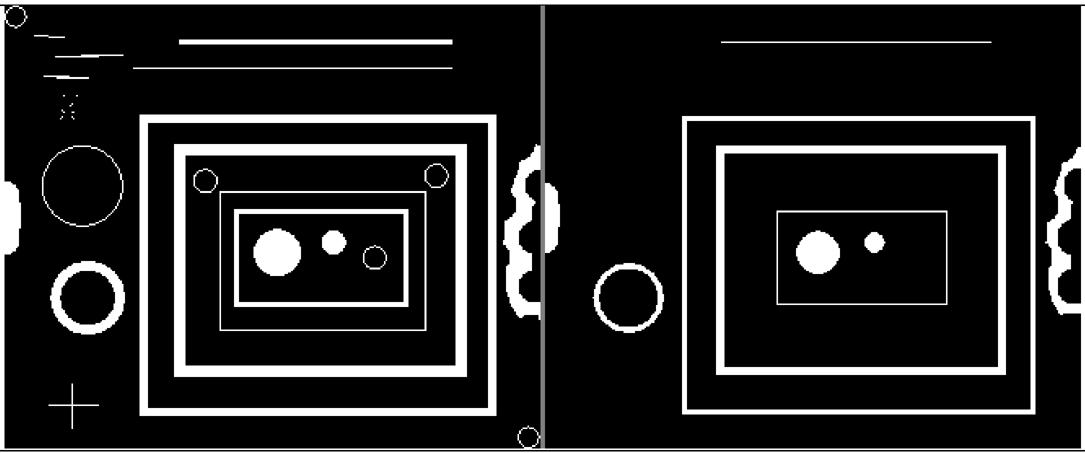
Function: imdilate



Erosion
The value of the output pixel is the minimum value of all pixels in the neighborhood. In a binary image, a pixel is set to 0 if any of the neighboring pixels have the value 0.

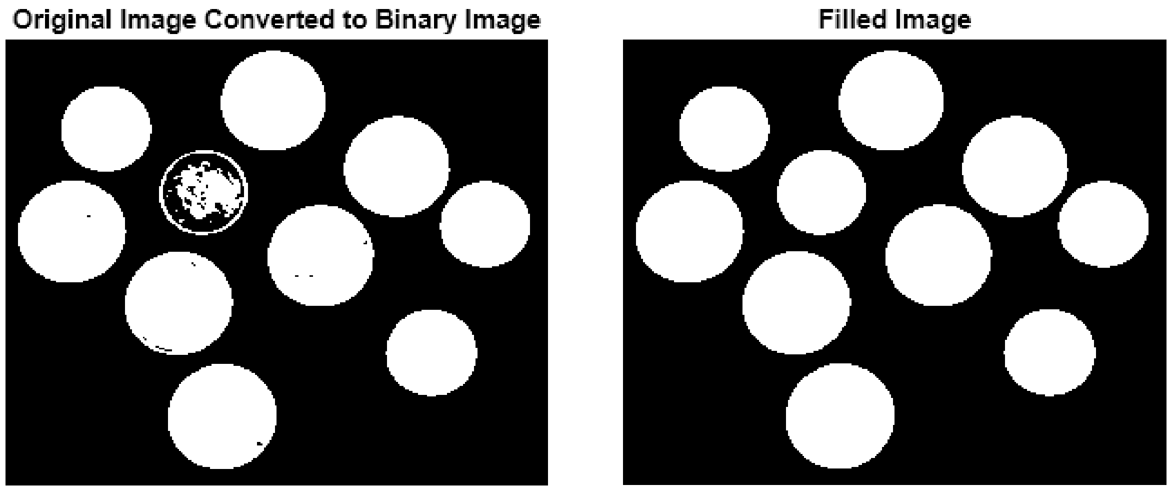
Morphological erosion removes floating pixels and thin lines so that only substantive objects remain. Remaining lines appear thinner and shapes appear smaller.

Function: imerode

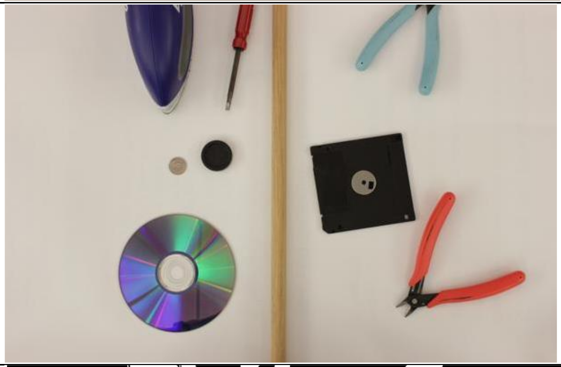
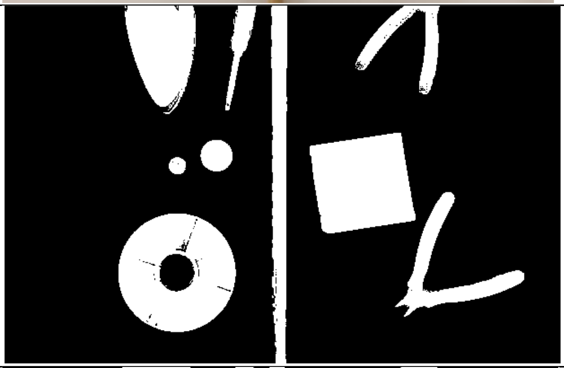
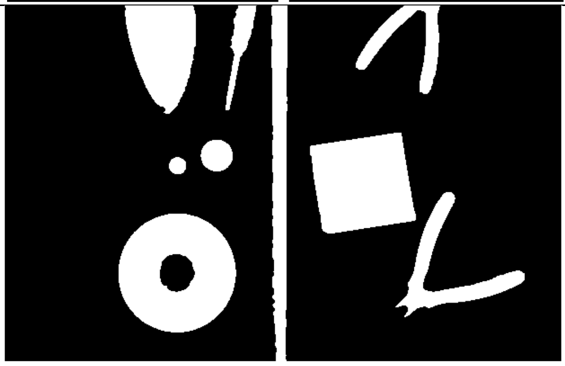


Flood fill
Performs a flood-fill operation on background pixels of the input binary image BW, starting from the points specified in locations.

Function: imfill



Background subtraction and finding the difference image in MATLAB

Read background image **BK**	
Read foreground image **A**	
Calculate the absolute sum of differences between RGB values of the images. **A_diff = sum(abs(double(A)-double(BK)),3);** Only the pixels that are different from the background will have large differences. Set a threshold (TH) to find a BW image where objects in the foreground are represented with white pixels. **A_diff = A_diff > TH;**	
Fill in the small holes using the imclose function **A_diff_closed = imclose(A_diff, strel('disk',4));** Imclose implements the morphological close operation: a dilation followed by an erosion, using the same structuring element for both operations. Strel creates a morphological object with the shape of a disk with radius 4. See this document for more info on dilation and erosion.	
Now we have the foreground objects marked by white pixels. E.g we can find the percentage of the total area of objects in the image. **sum(sum(A_diff_closed)) / (size(A_diff_closed,1)*size(A_diff_closed,2))**	0.1856

Counting objects in an image

For this problem, we will use the function called **imfill** which performs a flood fill starting from a given location (row, col). The function fills the connected black regions (0 pixels) with white (1 pixels).

Pseudocode for counting objects

```
Make a reverse image REV using A_diff_bw_close (see above) such that the objects are in black
and the remainder (background) is in white.

Set object_count to 0
Set min_area to a fixed number (# pixels area for an object to be counted)

for all pixels (r,c) in REV

    if   pixel at r, c is an object pixel (with value 0) in REV

        flood fill the area in REV containing location (r,c) with white. Use imfill

        if    area of object is > min_area

            increment object_count

        end

    end

end
```