Report

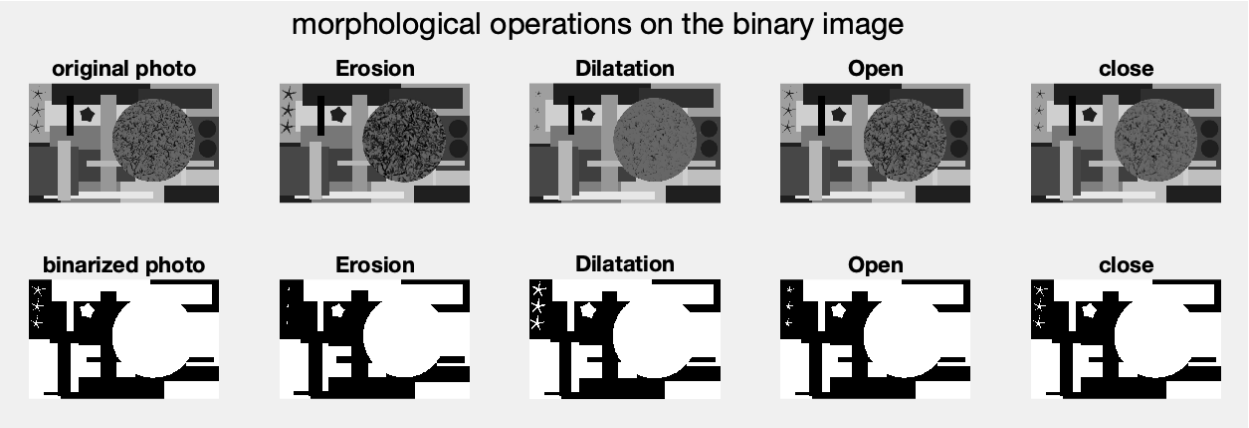
Isil Sonmez

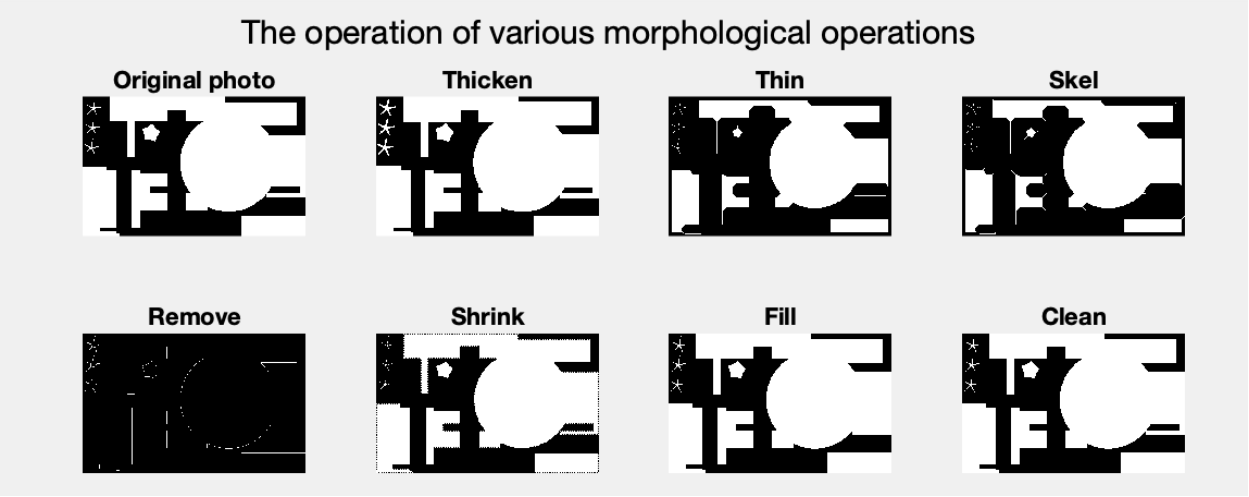
TASK 1

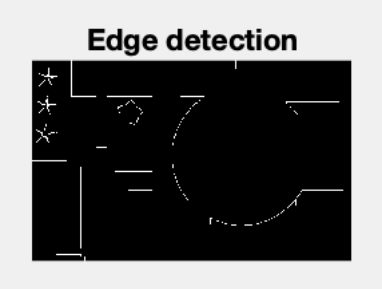
```
clear
close all
clc

%%
imRGB = imread('Thresholding1.jpg');

%%
imBW = rgb2gray(imRGB);

%% Binaration
thresh1 = graythresh(imBW);
imBW1 = im2bw(imBW,thresh1);
imBW1 = not(imBW1);

%% morphological
figure;
subplot(2, 5, 1);
imshow(imBW);
title('original photo');

subplot(2, 5, 6);
imshow(imBW1);
title('binarized photo');

%% Maska
se = strel('square', 3);

%% operation morphological

imErode = imerode(imBW, se);
subplot(2, 5, 2);
imshow(imErode);
title('Erosion');

imErode = imerode(imBW1, se);
subplot(2, 5, 7);
imshow(imErode);
title('Erosion');

%Dilalation
imDilate = imdilate(imBW, se);
subplot(2, 5, 3);
imshow(imDilate);
title('Dilatation');

imDilate = imdilate(imBW1, se);
subplot(2, 5, 8);
imshow(imDilate);
title('Dilatation');

% open
imOpen = imopen(imBW, se); % Open
subplot(2, 5, 4);
imshow(imOpen);
```

```
title('Open');

imOpen = imopen(imBW1, se); % Open
subplot(2, 5, 9);
imshow(imOpen);
title('Open');

% close
imClose = imclose(imBW, se); %close
subplot(2, 5, 5);
imshow(imClose);
title('close');

imClose = imclose(imBW1, se);
subplot(2, 5, 10);
imshow(imClose);
title('close');

%%
sgtitle('morphological operations on the binary image');

%%
imEdge = imBW1 - imErode
figure(2);
subplot(1, 3, 2);
imshow(imEdge);
title('Edge detection');

%%
%
figure;
subplot(2, 4, 1);
imshow(imBW1);
title('Original photo');

% Thicken
subplot(2, 4, 2);
imThicken = imdilate(imBW1, ones(3));
imshow(imThicken);
title('Thicken');

% Thin
subplot(2, 4, 3);
imThin = bwmorph(imBW1, 'thin', 10);
imshow(imThin);
title('Thin');

% Skel
subplot(2, 4, 4);
imSkel = bwmorph(imBW1, 'skel', 10);
imshow(imSkel);
title('Skel');

% Remove )
subplot(2, 4, 5);
imRemove = bwmorph(imBW1, 'remove');
imshow(imRemove);
title('Remove');

% Shrink
subplot(2, 4, 6);
```

```
imShrink = bwmorph(imBW1, 'shrink', 1);
imshow(imShrink);
title('Shrink');

% Fill
subplot(2, 4, 7);
imFill = imfill(imBW1, 'holes');
imshow(imFill);
title('Fill');

% Clean )
subplot(2, 4, 8);
imClean = bwareaopen(imBW1, 100);
imshow(imClean);
title('Clean');


sgtitle('The operation of various morphological operations');
```







Erosion: Shrinks the objects in the image.

Dilatation: Expands the objects in the image.

Opening: Removes small objects from the foreground (erosion followed by dilation).

Closing: Fills small holes in the objects (dilation followed by erosion).

Thicken: Expands the objects.

Thin: Thins the objects to single-pixel width.

Skeletonize: Reduces objects to their skeletons.

Remove: Removes isolated pixels.

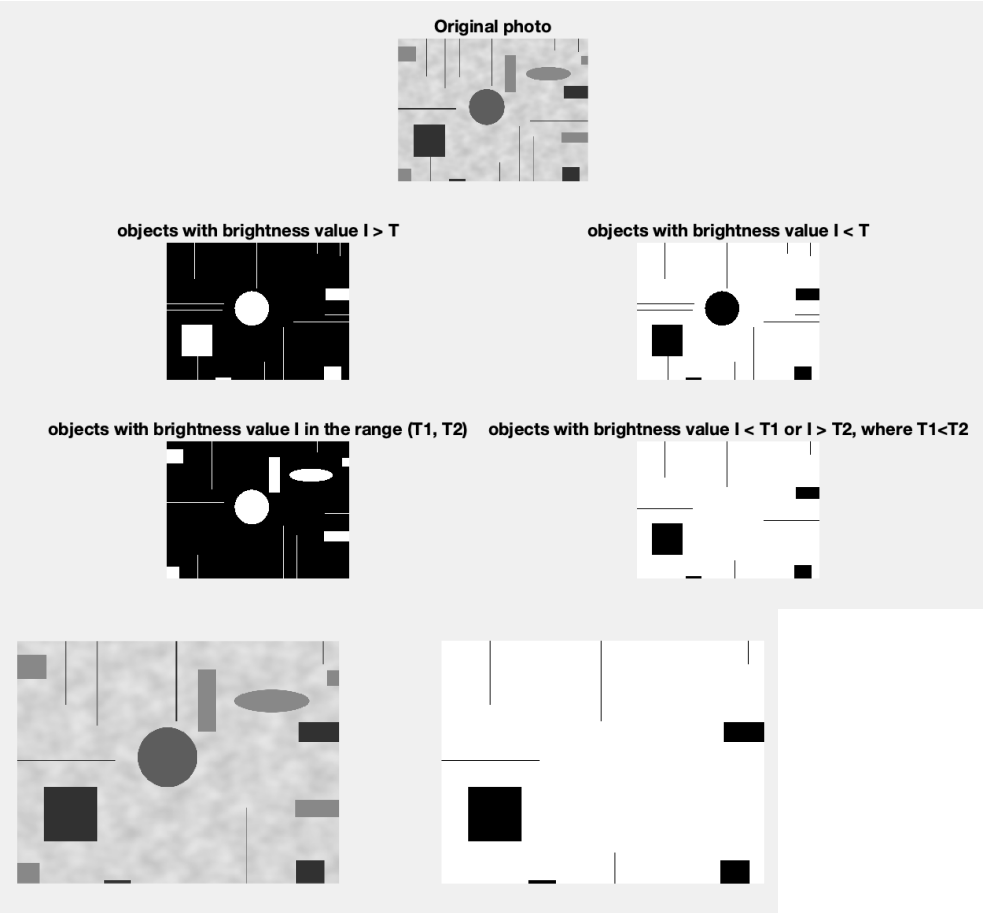
Shrink: Shrinks objects to points.

Fill: Fills holes in objects.

Clean: Removes small objects from the binary image.

Task 2

```
clear

clc

%%]
image = imread('Thresholding8.jpg');

%%
if size(image, 3) == 3
    gray_image = rgb2gray(image);
else
    gray_image = image;
end

%%
low_threshold = 50;
threshold = 115;
high_threshold = 150;
binary_image_T1 = gray_image < threshold;
binary_image_T2 = gray_image > threshold;
binary_image = gray_image > low_threshold | gray_image > high_threshold;
binary_image1 = gray_image > low_threshold & gray_image < high_threshold;


se = strel('square', 3);
%%
figure(1)

subplot(3, 2, [1 2]);
imshow(gray_image);
title('Original photo')

subplot(3, 2, 3)
imshow(binary_image_T1)
title('objects with brightness value I > T')

subplot(3, 2, 4)
imshow(binary_image_T2)
title('objects with brightness value I < T')

subplot(3, 2, 5);
imshow(binary_image1);
title('objects with brightness value I in the range (T1, T2)')

subplot(3, 2, 6);
imshow(binary_image)
title('objects with brightness value I < T1 or I > T2, where T1<T2');

figure(2)
subplot(1,2,1)
imErode = imdilate(gray_image, se);
imshow(imErode)

subplot(1,2,2)
binary_image = imErode > low_threshold | imErode > high_threshold;
imshow(binary_image)
```



Observation:
binary_image_T1: Pixels below the threshold (115) are set to 1 (white).
binary_image_T2: Pixels above the threshold are set to 1.
binary_image: Pixels either below the low_threshold (50) or above
the high_threshold (150) are set to 1.
binary_image1: Pixels within the range of low_threshold to high_threshold are set to 1.
Conclusion: Different binary images highlight pixels based on various intensity
thresholds, helping to identify different regions of interest in the image.

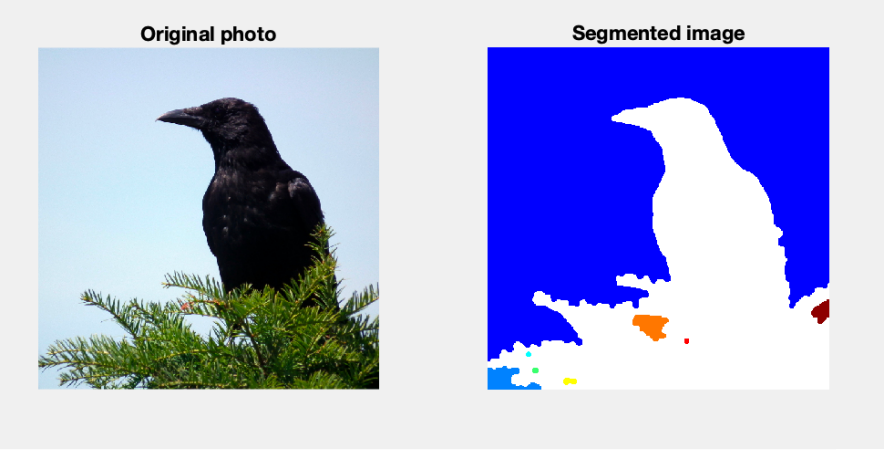
Observation: A square structuring element of size 3x3 is created for use in dilation.
Conclusion: This structuring element will be used to dilate the grayscale image,
affecting the binary image generation in the subsequent steps.

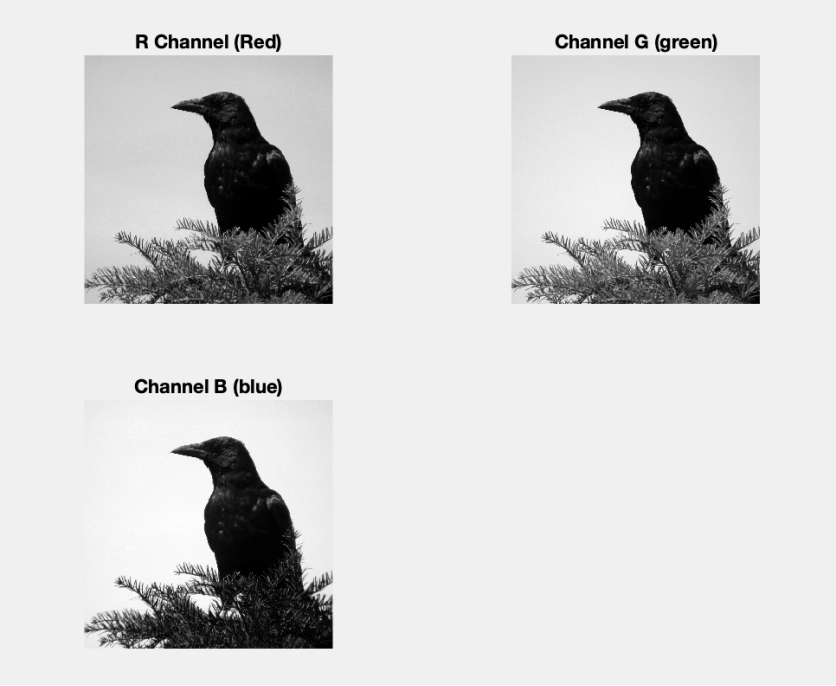
Observation: Different binary images are displayed alongside the original grayscale
image.
Conclusion: The various thresholding techniques allow for the examination of different
aspects of the image, highlighting regions based on intensity values.

Observation: The grayscale image is dilated using the square structuring element.A binary
image is created from the dilated image based on the low_threshold and high_threshold.
Conclusion: Dilation expands the bright regions in the image, which can be useful for
filling small holes and connecting disjoint objects in the image.

Task 3

```
clear
close all
clc


image = imread('Crow.JPG');


R = image(:,:,1);
G = image(:,:,2);
B = image(:,:,3);


figure;
subplot(2, 2, 1);
imshow(R);
title('R Channel (Red)');

subplot(2, 2, 2);
imshow(G);
title('Channel G (green)');

subplot(2, 2, 3);
imshow(B);
title('Channel B (blue)');


threshold = 120;
binary_image = R > threshold;

% Operation morphological
se = strel('disk', 5);
binary_image = imopen(binary_image, se);
binary_image = imclose(binary_image, se);
binary_image = imfill(binary_image, 'holes');


labeled_image = bwlabel(binary_image);
segment_props = regionprops(labeled_image, 'Area', 'BoundingBox');


figure;
subplot(1, 2, 1);
imshow(image);
title('Original photo');

subplot(1, 2, 2);
imshow(label2rgb(labeled_image));
title('Segmented image');
```



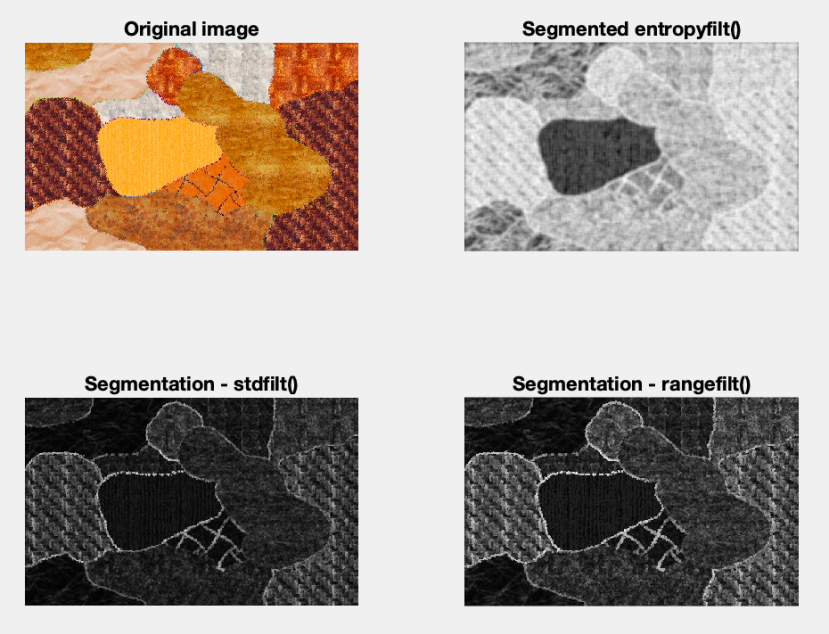


Observations: The thresholding on the red channel successfully isolates regions based on intensity. Morphological operations effectively remove noise and refine the segmented regions. The final segmented image highlights distinct objects in the image, which are color-coded for better visualization.

Conclusions: The script segments the image effectively using the red channel's intensity values and refines the segmentation with morphological operations. Comparing the original and segmented images visually confirms the segmentation's success, showing it can be useful for object detection and analysis tasks.

Task 4

```
clear
close all
clc

image = imread('texture3.jpg');
image_gray = rgb2gray(image);
entropy_filtered_image = entropyfilt(image_gray);
std_filtered_image = stdfilt(image_gray);
range_filtered_image = rangefilt(image_gray);
figure(1)
subplot(2, 2, 1), imshow(image), title('Original image');
subplot(2, 2, 2), imshow(entropy_filtered_image, []), title('Segmented entropyfilt()');
subplot(2, 2, 3), imshow(std_filtered_image, []), title('Segmentation - stdfilt()');
subplot(2, 2, 4), imshow(range_filtered_image, []), title('Segmentation - rangefilt()');
```



Conclusions
Entropy Filtering: The entropy-filtered image segments regions with high randomness or complexity. This is useful for identifying textured areas in the image.

Standard Deviation Filtering: The standard deviation filtered image highlights areas with high texture variation or contrast. This filter helps in detecting edges and textured patterns.

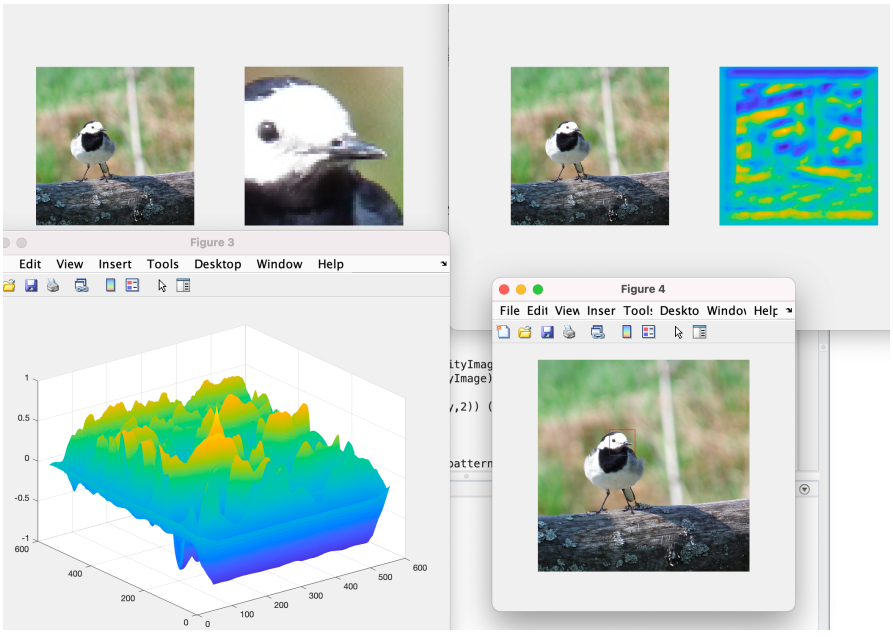
Range Filtering: The range-filtered image emphasizes changes in intensity, highlighting edges and texture boundaries. It's particularly good for edge detection and texture segmentation.

Overall Effectiveness: Each filtering method provides a unique view of the texture and variability in the image. By comparing these filtered images, we can identify and analyze different textural features and patterns.

Visual Comparison: Displaying the original and filtered images side-by-side allows for easy visual comparison, showing how each filter highlights different aspects of the image's texture.

Task 5

```
clc
clear all
close all
imRGB = imread('Wagtail.jpg');
patternRGB = imread('Wagtail_P2.jpg');
figure(1)
subplot(1,2,1)
imshow(imRGB)
subplot(1,2,2)
imshow(patternRGB)
imGray = rgb2gray(imRGB);
patternGray = rgb2gray(patternRGB);
similarityImage = normxcorr2(patternGray,imGray);
figure(2)
subplot(1,2,1)
imshow(imRGB)
subplot(1,2,2)
imagesc(similarityImage)
axis off
axis equal
figure(3)
surf(similarityImage)
shading flat
[maxSimilarity, imax] = max(abs(similarityImage(:)));
[ypeak, xpeak] = ind2sub(size(similarityImage),imax(1));

peak_offset = [(xpeak - size(patternGray,2)) (ypeak -size(patternGray,1))];
figure(4)
imshow(imRGB);
hold on
rectangle('Position',[peak_offset,size(patternGray)],'Edgecolor','red');
```



Conclusion: A 3D plot highlights high correlation areas.The best match location is marked with a rectangle on the original image. Normalized cross-correlation ensures robust and precise pattern matching, as shown by the accurate identification of the bird's head.