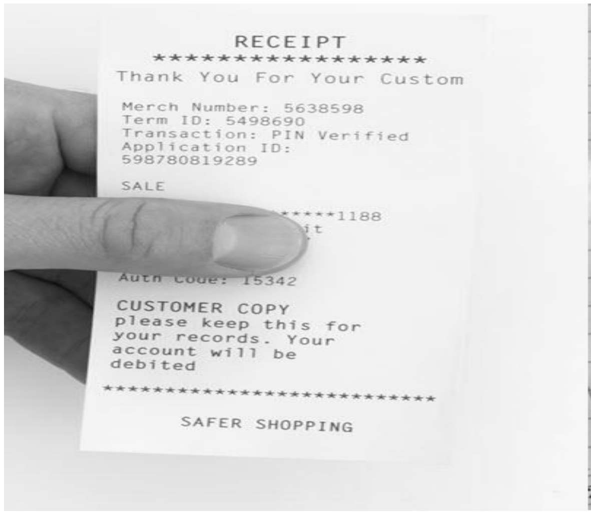
Contrast and Intensity Histograms

Even though these two receipt images are grayscale, the contrast is different.

If you are analyzing a set of images, normalizing the brightness can be an important preprocessing step, especially for identifying the black and white patterns of text in receipt images.



RECEIPT
****
Thank You For Your Custom

Merch Number: 5638598
Term ID: 5498690
Transaction: PIN Verified
Application ID:
598780819289

SALE

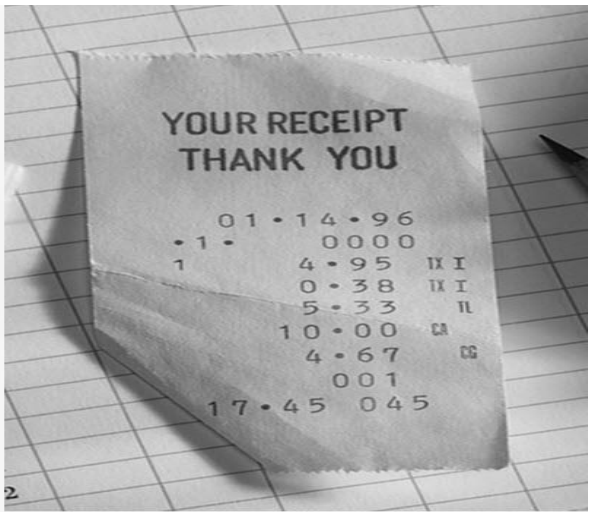
*****1188
it

Auth Code: 15342

CUSTOMER COPY
please keep this for
your records. Your
account will be
debited

****

SAFER SHOPPING

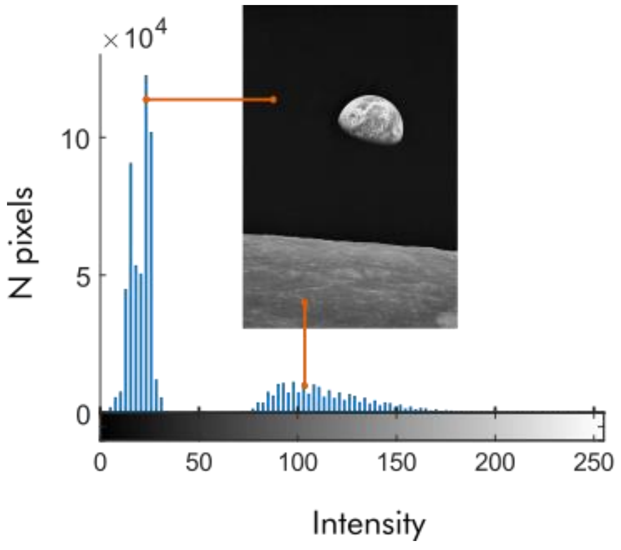


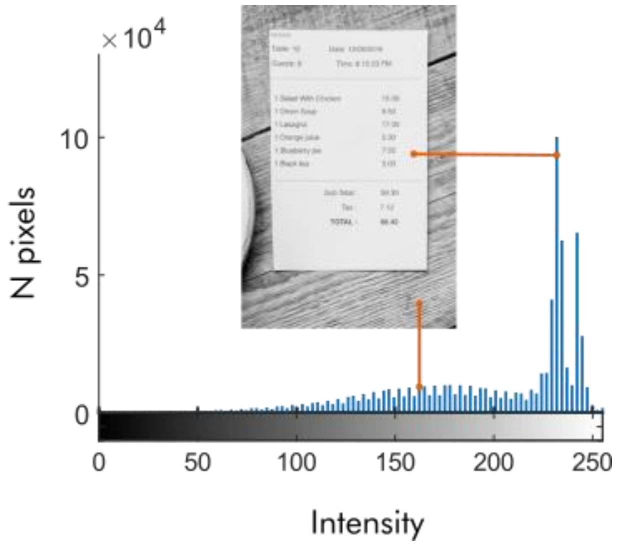
YOUR RECEIPT
THANK YOU

01•14•96
•1• 0000
1 4•95 TX I
0•38 TX I
5•33 TL
10•00 CA
4•67 CG
001
17•45 045

An intensity histogram separates pixels into bins based on their intensity values. Dark images, for example, have many pixels binned in the low end of the histogram. Bright regions have pixels binned at the high end of the histogram.

The histogram often suggests where simple adjustments can be made to improve the definition of image features. Do you notice any adjustments that could be made to the receipt image so that the text is easier to identify?

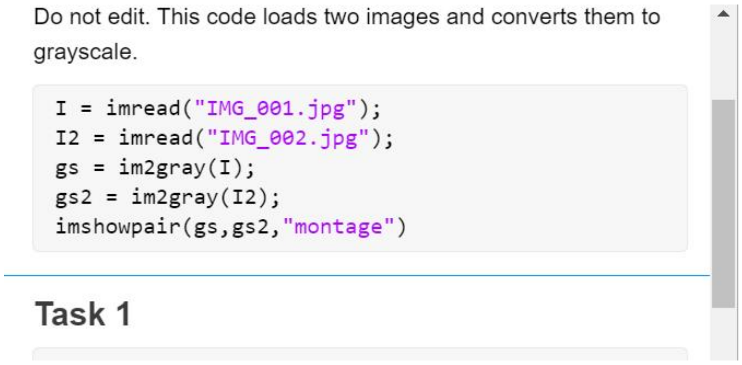




You can investigate the contrast in an image by viewing its intensity histogram using the `imhist` function.

```
imhist(I)
```

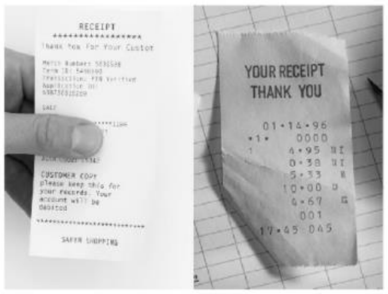
If the histogram shows pixels binned mainly at the high and low ends of the spectrum, the receipt has good contrast. If not, it could probably benefit from a contrast adjustment.

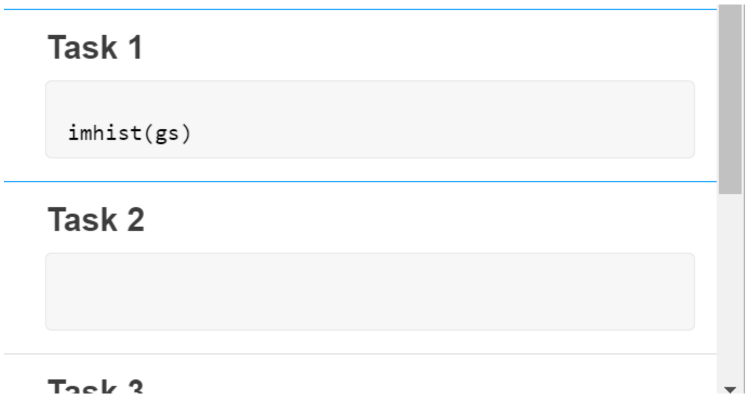


Do not edit. This code loads two images and converts them to grayscale.

```
I = imread("IMG_001.jpg");
I2 = imread("IMG_002.jpg");
gs = im2gray(I);
gs2 = im2gray(I2);
imshowpair(gs,gs2,"montage")
```

Task 1

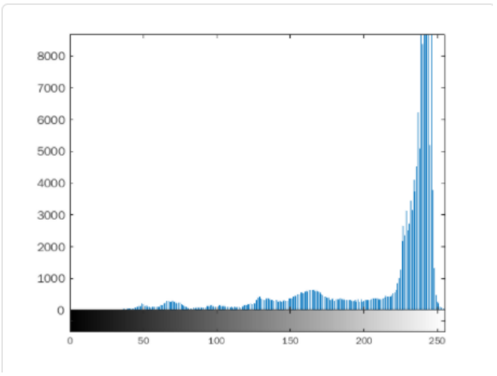


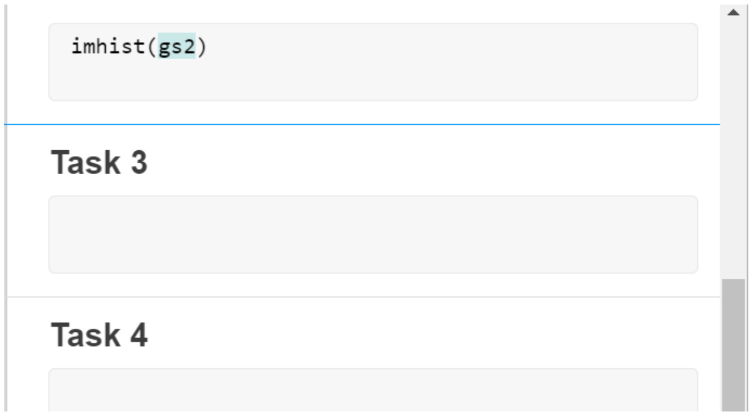


Task 1

```
imhist(gs)
```

Task 2

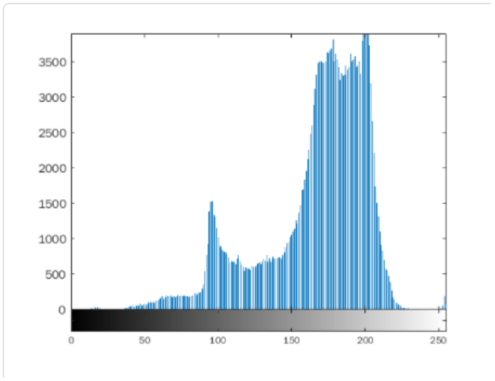




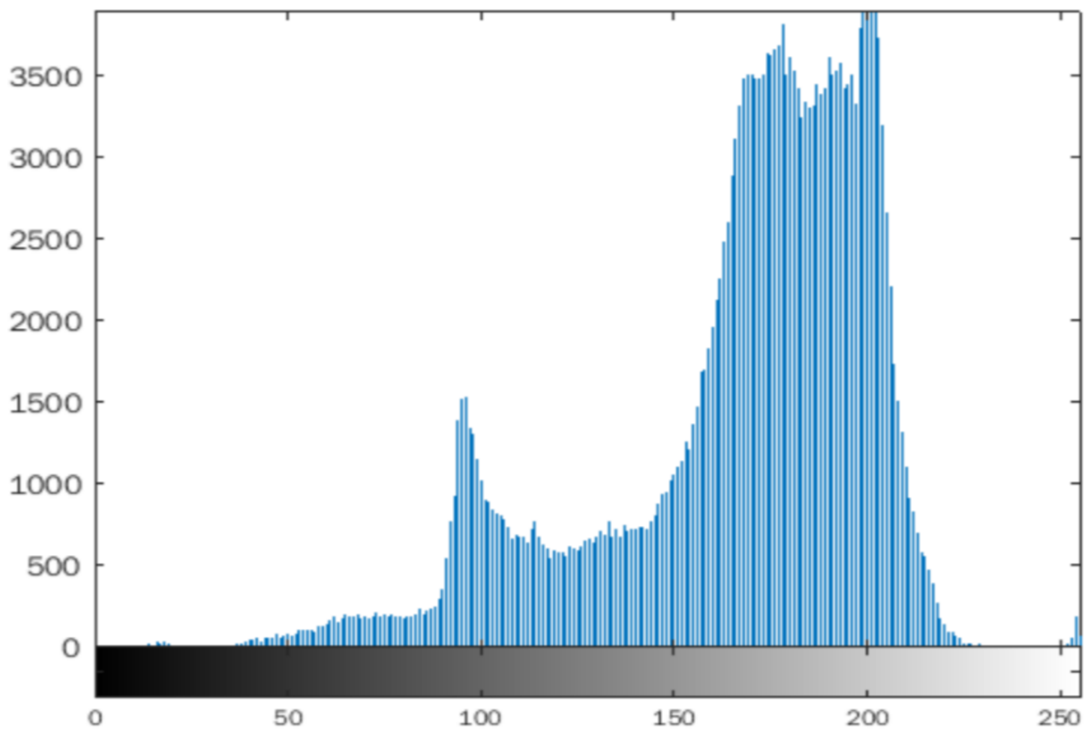
```
imhist(gs2)
```

Task 3

Task 4

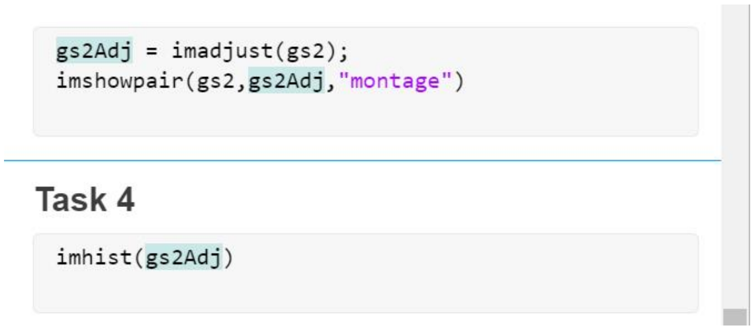


The intensity histogram of `gs2` shows lower contrast between the text and the background. Most of the dark pixels have intensity values around 100, and not many bright pixels have intensity values above 200. That means the contrast is about half of what it could be if the image used the full intensity range (0 to 255).



Increasing image contrast brightens brighter pixels and darkens darker pixels. You can use the `imadjust` function to adjust the contrast of a grayscale image automatically.

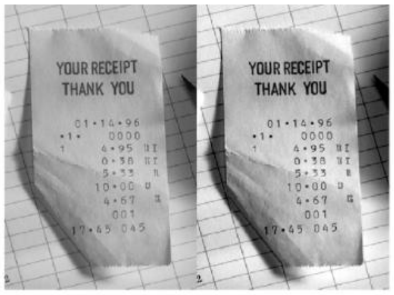
```
Aadj = imadjust(A);
```

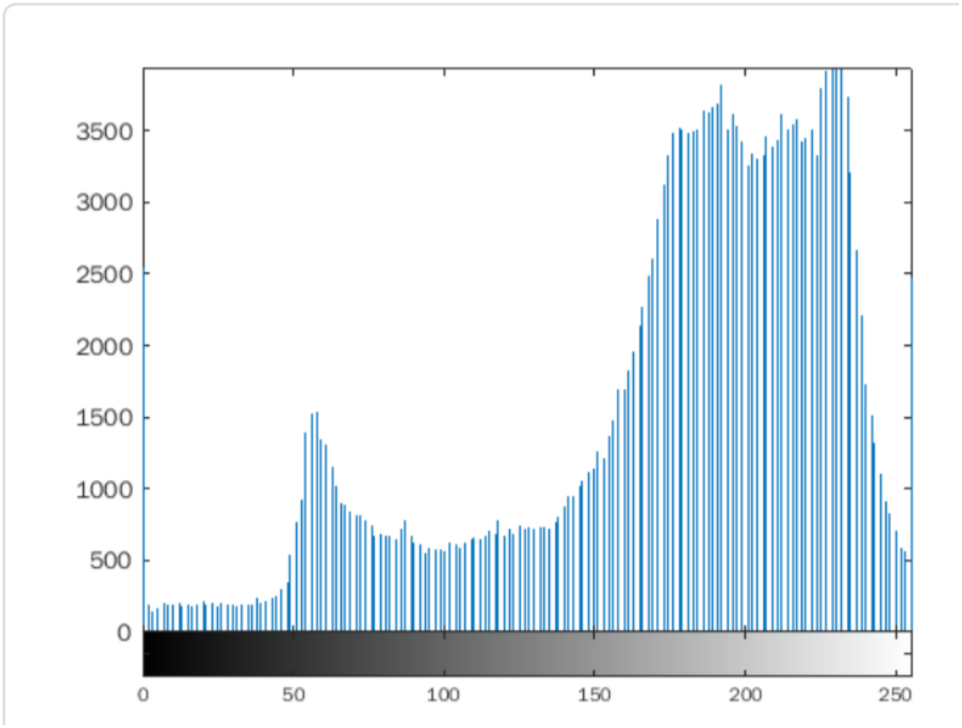


```
gs2Adj = imadjust(gs2);
imshowpair(gs2,gs2Adj,"montage")
```

Task 4

```
imhist(gs2Adj)
```



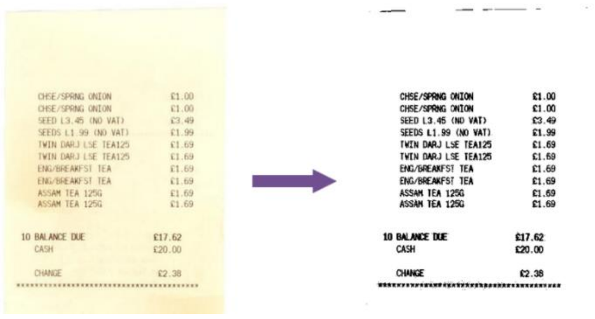


`imadjust` only works for grayscale images unless there are additional inputs to the function. You can, however, use the function `imlocalbrighten` to adjust the contrast of a color image. Try using `imlocalbrighten` on the color image `I2` and display the result.

```
I2adj = imlocalbrighten(I2);
```

Segmenting Text

The main feature of receipt images is dark text on a bright background. How can you leverage this? One method is to separate the text from the background. Separating an image into distinct parts is called *segmenting* an image.



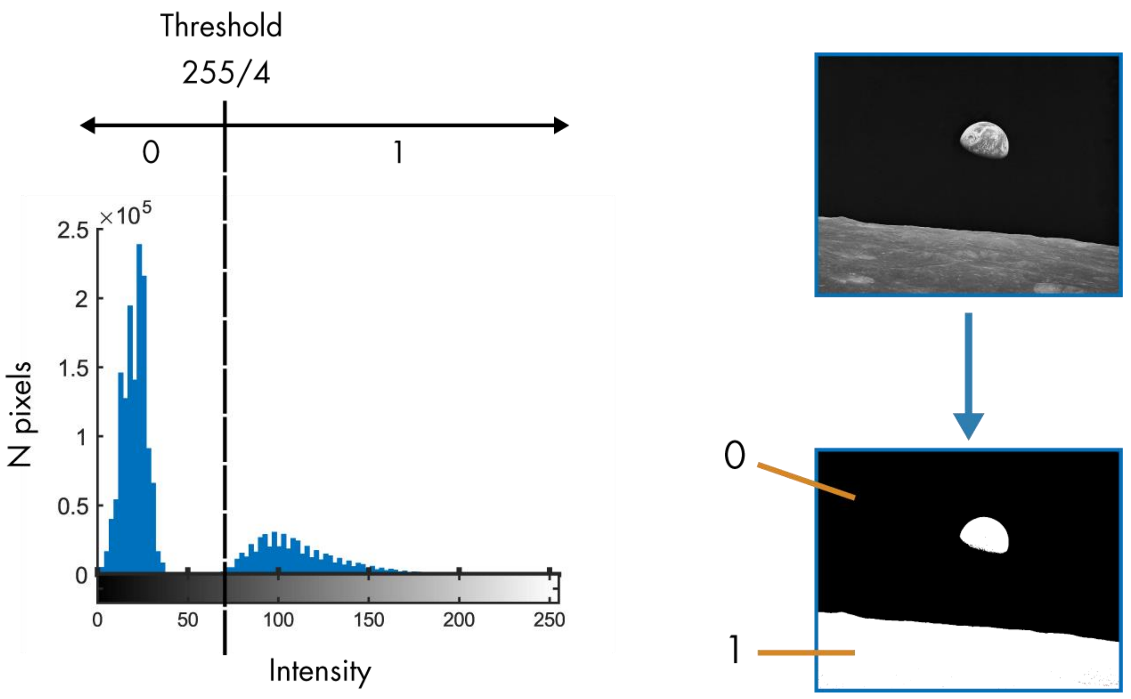
CHSE/SPRNG ONION	£1.00
CHSE/SPRNG ONION	£1.00
SEED L3.45 (NO VAT)	£3.49
SEEDS L1.99 (NO VAT)	£1.99
TWIN DARJ LSE TEA125	£1.69
TWIN DARJ LSE TEA125	£1.69
ENG/BREAKFST TEA	£1.69
ENG/BREAKFST TEA	£1.69
ASSAM TEA 125G	£1.69
ASSAM TEA 125G	£1.69
10 BALANCE DUE	£17.62
CASH	£20.00
CHANGE	£2.38

****

Intensity Thresholding

You can create a binary black and white image from a grayscale image by *thresholding* its intensity values. Values below the cutoff are assigned the value `0`, while those above are assigned the value `1`.

In the example below, a grayscale image was segmented using a threshold of 1/4 the maximum possible intensity of 255.



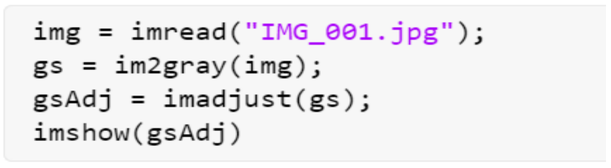
- When applied to arrays, logical operators like `<` and `>` generate arrays of the same size that contain logical values `1` (`true`) or `0` (`false`).

  You can use logical operators **to threshold the intensity values of a grayscale image**, creating a binary image.

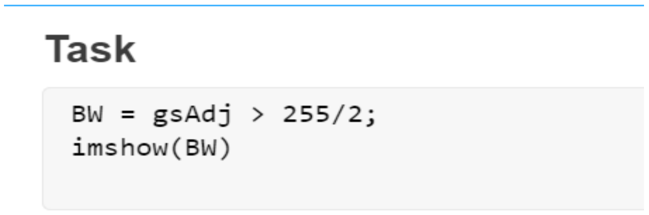
```
B = g > thresh;
```

TASK
Create a binary image `BW` by thresholding `gsAdj` at half the maximum possible intensity (the maximum intensity is 255).

Display `BW` using `imshow`.

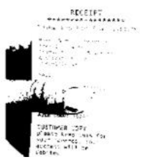


```
img = imread("IMG_001.jpg");
gs = im2gray(img);
gsAdj = imadjust(gs);
imshow(gsAdj)
```



Task

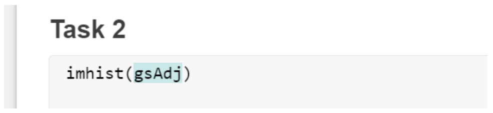
```
BW = gsAdj > 255/2;
imshow(BW)
```



That didn't quite work; some of the receipt text is light gray and gets washed out by the threshold. You can identify a better cutoff value by looking at the image's intensity histogram.

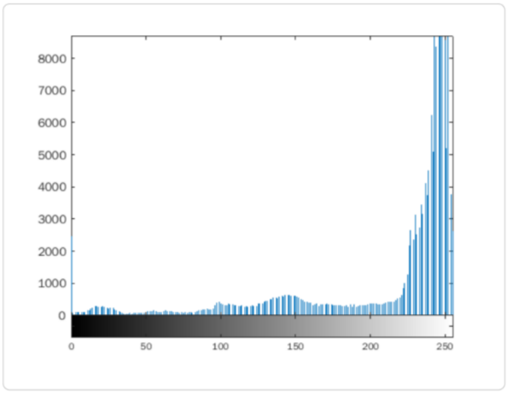
TASK

Plot the intensity histogram for `gsAdj`.



Task 2

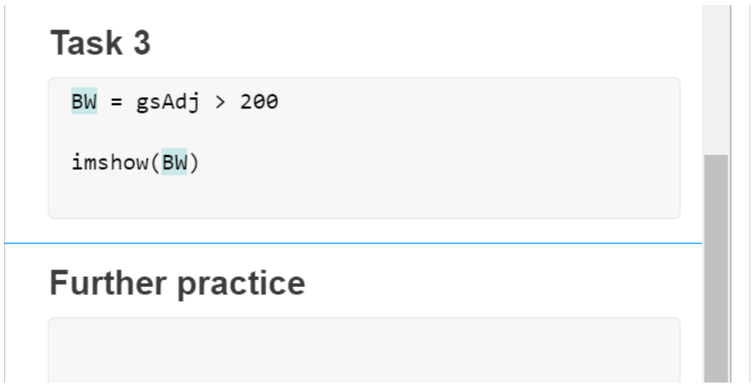
```
imhist(gsAdj)
```



The histogram shows a large peak between 220 and 250, which contains the bright pixels in the paper and background. To create a better segmentation, you can use a threshold that's just below this peak.

TASK

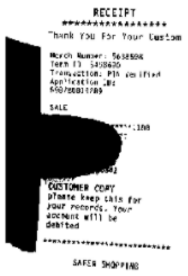
Threshold `gsAdj` at 200 and store the result in `BW`. Display `BW`.



Task 3

```
BW = gsAdj > 200

imshow(BW)
```

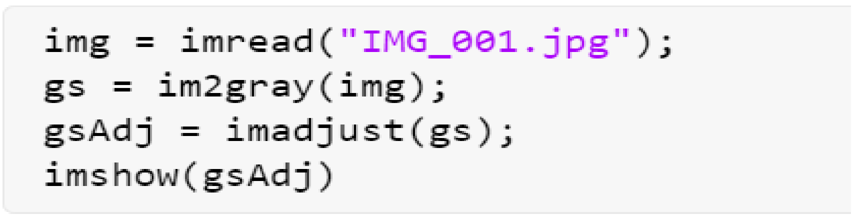
Further practice



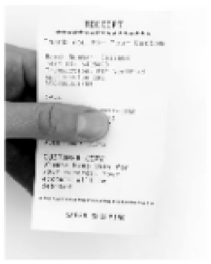
The ultimate goal is to create an algorithm that can process *all* of our images. Unfortunately, receipt images taken in different lighting conditions will have varied contrast and require different thresholds. Manually identifying a threshold using the intensity histogram works, but is impractical for thousands of images.

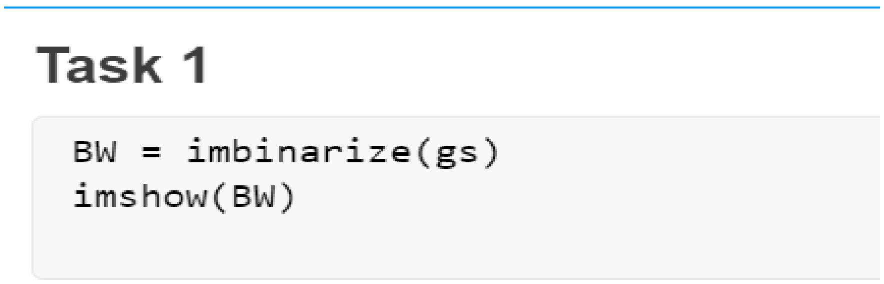
To automate the threshold selection process, you can use the `imbinarize` function, which calculates the "best" threshold for the image.

```
gBinary = imbinarize(g);
```



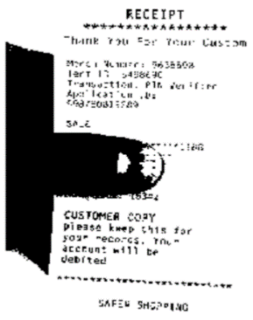
```
img = imread("IMG_001.jpg");
gs = im2gray(img);
gsAdj = imadjust(gs);
imshow(gsAdj)
```





Task 1

```
BW = imbinarize(gs)
imshow(BW)
```



That did ok, but manual threshold selection performed better. The text in the top half of the receipt isn't as clear as the text in the bottom half.

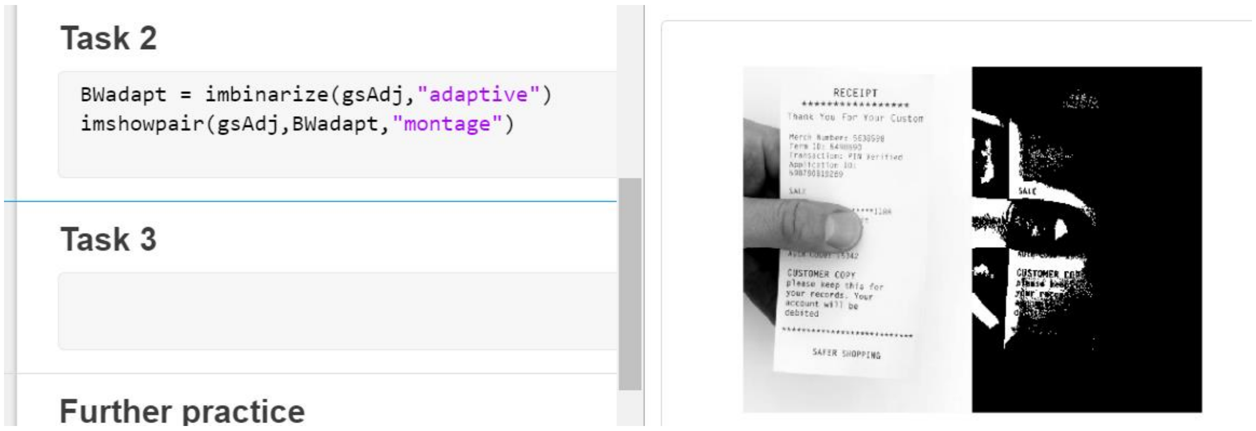
By default, `imbinarize` uses a global threshold value — the same threshold for every pixel in the image.

You can have `imbinarize` look at smaller portions of the image and pick the best threshold for that region by passing the `"adaptive"` option as the second argument to `imbinarize`.

```
gBinary = imbinarize(g,"adaptive");
```

TASK
Use `imbinarize` with the option `"adaptive"` to create a binary image from `gsAdj`. Store the result in `BWadapt`.

Display `gsAdj` and `BWadapt` together in a montage using `imshowpair`.



Task 2

```
BWadapt = imbinarize(gsAdj,"adaptive")
imshowpair(gsAdj,BWadapt,"montage")
```

Task 3

Further practice

That doesn't seem to have helped. The text now appears completely washed out in black. What happened?

By default, the foreground of an image is assumed to be bright and the background dark. But for receipts, the foreground is the dark text, and the background is bright.

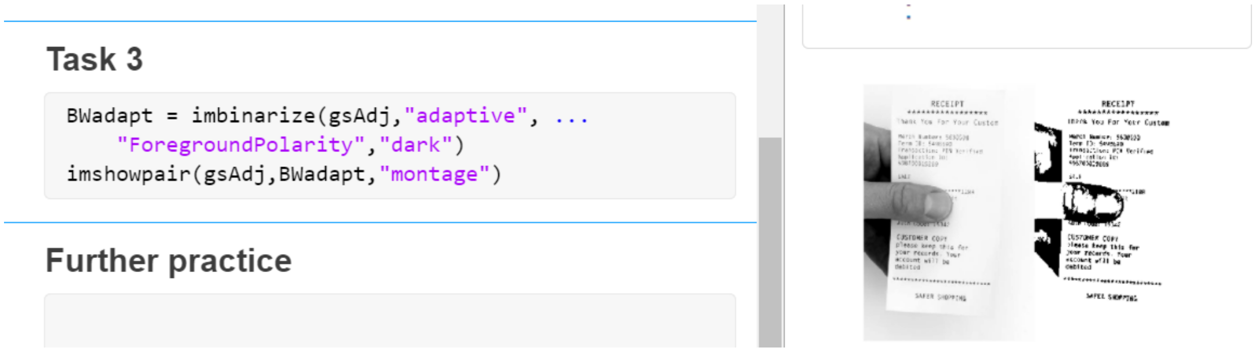
In `imbinarize`, you can designate whether the foreground is bright or dark by setting the `"ForegroundPolarity"` option.

```
gBinary = imbinarize(g,"adaptive",... "ForegroundPolarity","dark")
```

TASK

Use `imbinarize` with the `"adaptive"` option and `"ForegroundPolarity"` set to `"dark"` to create a binary image from `gsAdj`. Store the result in `BWadapt`.

Display `gsAdj` and `BWadapt` in a montage using `imshowpair`.



Task 3

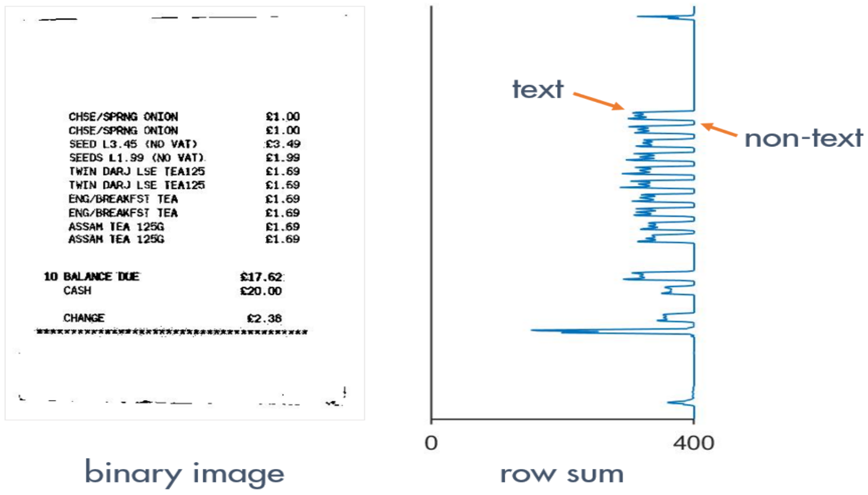
```
BWadapt = imbinarize(gsAdj,"adaptive", ...
    "ForegroundPolarity","dark")
imshowpair(gsAdj,BWadapt,"montage")
```

Further practice

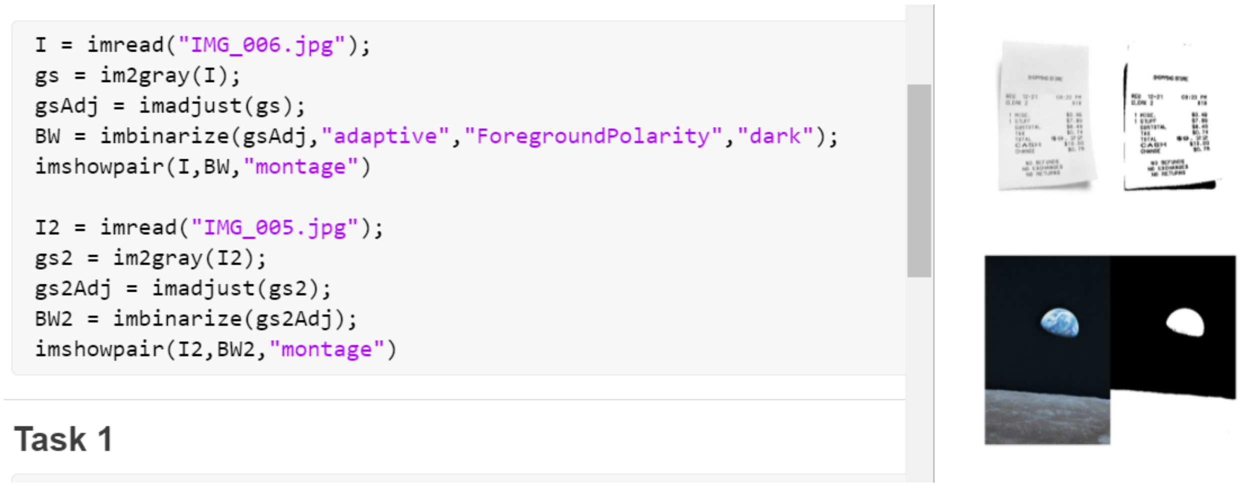
Identifying Text Patterns

You've improved the visibility of the text in a binary image. Now how can you determine if the image is a receipt? The answer lies in the pattern of text in the image.

Rows of pixels containing text have more `0` values, and the rows between lines of text have more `1`s. If you sum the values across each row, rows with text will have smaller sums than rows without text.



```
CHSE/SPRNG ONION          £1.00
CHSE/SPRNG ONION          £1.00
SEED L3.45 (NO VAT)       £3.49
SEEDS L1.99 (NO VAT)      £1.99
TWIN DARJ LSE TEA125      £1.69
TWIN DARJ LSE TEA125      £1.69
ENG/BREAKFST TEA          £1.69
ENG/BREAKFST TEA          £1.69
ASSAM TEA 125G            £1.69
ASSAM TEA 125G            £1.69

10 BALANCE DUE           £17.62
   CASH                  £20.00

   CHANGE                 £2.38
```

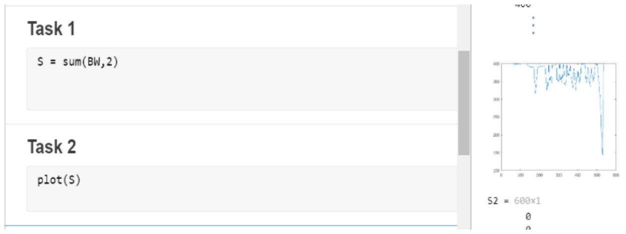


```
I = imread("IMG_006.jpg");
gs = im2gray(I);
gsAdj = imadjust(gs);
BW = imbinarize(gsAdj,"adaptive","ForegroundPolarity","dark");
imshowpair(I,BW,"montage")

I2 = imread("IMG_005.jpg");
gs2 = im2gray(I2);
gs2Adj = imadjust(gs2);
BW2 = imbinarize(gs2Adj);
imshowpair(I2,BW2,"montage")
```

Task 1

You can compute the sum across rows of an array using the `sum` function.

```
rSum = sum(BW,2)
```



Task 1

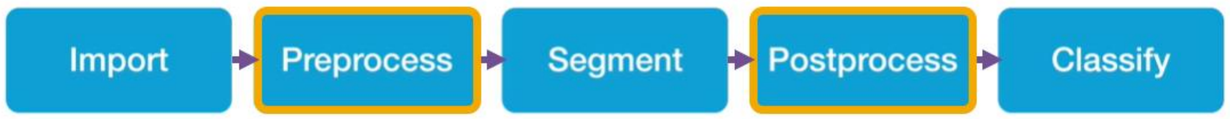
```
S = sum(BW,2)
```

Task 2

```
plot(S)
```

```
S2 = 600x1
        0
        0
```

Receipt Identification Workflow



Pre- and Postprocessing to Improve Segmentation

Segmentation can be improved in two ways: by preprocessing the image before binarizing and by postprocessing the binary image itself.

You've already done some preprocessing by converting each image to grayscale and adjusting the contrast. In the next few activities, you'll use three additional pre- and postprocessing techniques.

Segmentation can be improved in two ways: by *preprocessing the image before binarizing and by postprocessing the binary image itself.*

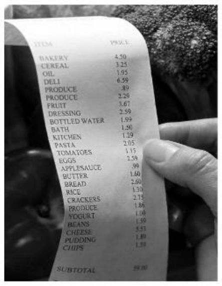
You've already done some preprocessing by converting each image to grayscale and adjusting the contrast. In the next few activities, you'll use three additional pre- and postprocessing techniques.

Noise Removal

Smooth pixel intensity values to reduce the impact of variation on binarization.

Background Isolation and Subtraction

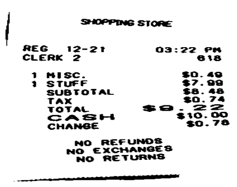
Isolate and remove the background of an image before binarizing.



```
ITEM              PRICE
BAKERY             4.50
CEREAL             3.25
OIL                1.95
DELI               6.59
PRODUCE             .89
PRODUCE            2.29
FRUIT              3.67
DRESSING           2.59
BOTTLED WATER      1.99
BATH               1.50
KITCHEN            1.29
PASTA              2.05
TOMATOES           1.15
EGGS               2.59
APPLESAUCE          .99
BUTTER             1.60
BREAD              2.60
RICE               [illegible]
CRACKERS           2.15
PRODUCE            1.86
YOGURT             [illegible]
BEANS              1.59
CHEESE             [illegible]
PUDDING            1.89
CHIPS              [illegible]

SUBTOTAL           [illegible]
```

Binary Morphology

Emphasize particular patterns or shapes in a binary image.



```
        SHOPPING STORE

REG   12-21          03:22 PM
CLERK  2                  618

 1 MISC.                $0.49
 1 STUFF                $7.99
   SUBTOTAL             $8.48
   TAX                  $0.74
   TOTAL             $9.22
   CASH                $10.00
   CHANGE               $0.78

       NO REFUNDS
      NO EXCHANGES
       NO RETURNS
```

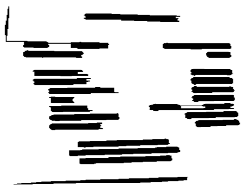
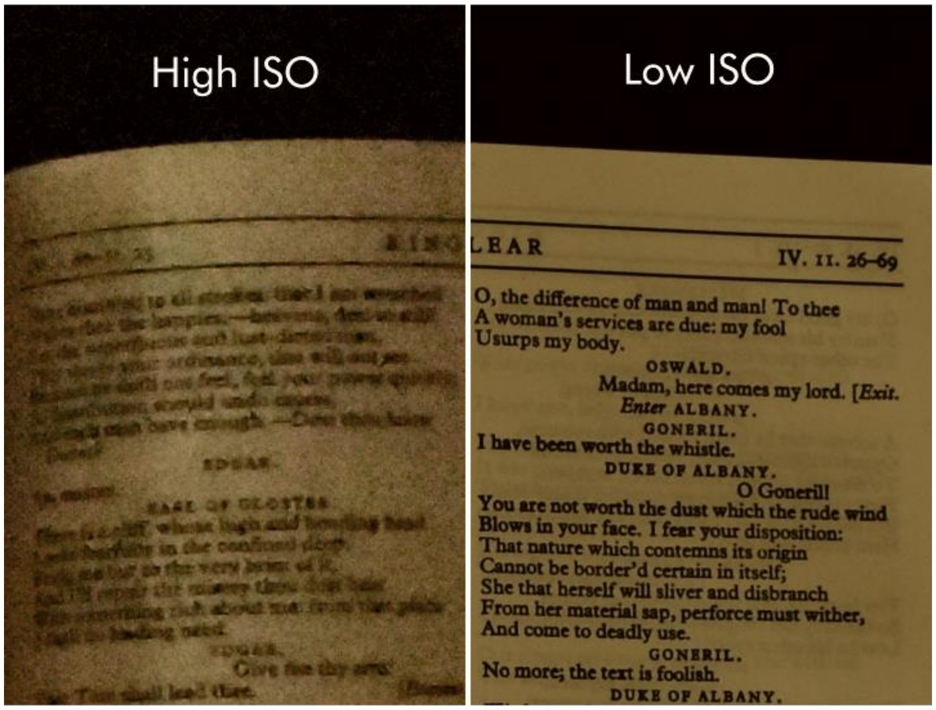


Image noise

You can increase the light sensitivity of a digital camera sensor to improve the brightness of a picture taken in low light. Many modern digital cameras (including mobile phone cameras) automatically increase the
ISO
in dim light. However, this increase in sensitivity amplifies noise picked up by the sensor, leaving the image grainy (shown on the left).



LEAR IV. II. 26–69

O, the difference of man and man! To thee
A woman's services are due: my fool
Usurps my body.

OSWALD.

Madam, here comes my lord. [*Exit.*

Enter ALBANY.

GONERIL.

I have been worth the whistle.

DUKE OF ALBANY.

O Goneril!
You are not worth the dust which the rude wind
Blows in your face. I fear your disposition:
That nature which contemns its origin
Cannot be border'd certain in itself;
She that herself will sliver and disbranch
From her material sap, perforce must wither,
And come to deadly use.

GONERIL.

No more; the text is foolish.

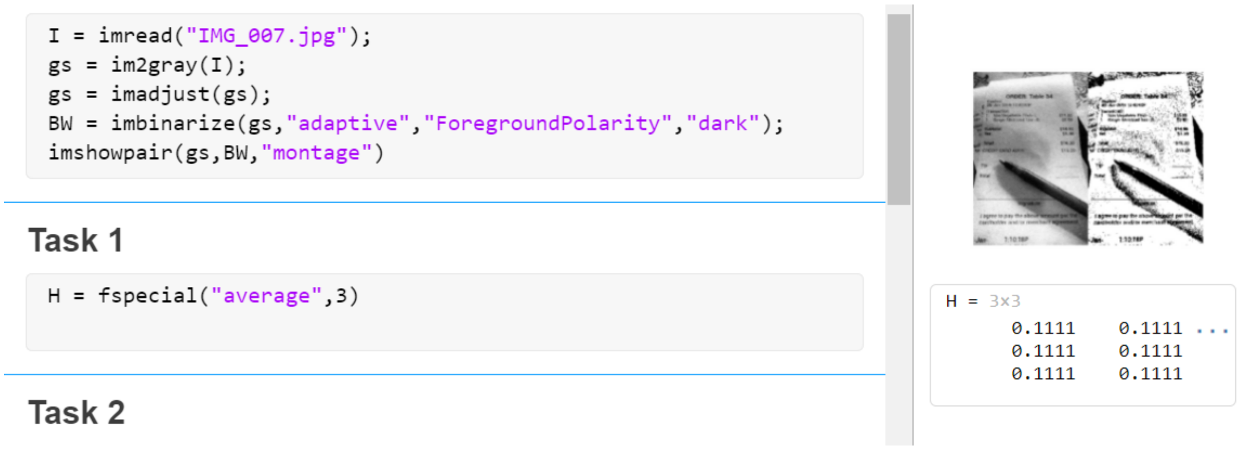
DUKE OF ALBANY.

Filtering noise

Images taken in low light often become noisy due to the increase in camera sensitivity required to capture the image. This noise can interfere with receipt identification by polluting regions in the binarized image.

To reduce the impact of this noise on the binary image, you can preprocess the image with an averaging filter.

Use the `fspecial` function to create an *n*-by-*n* averaging filter.



```
I = imread("IMG_007.jpg");
gs = im2gray(I);
gs = imadjust(gs);
BW = imbinarize(gs,"adaptive","ForegroundPolarity","dark");
imshowpair(gs,BW,"montage")
```

Task 1

```
H = fspecial("average",3)
```

```
H = 3x3
        0.1111     0.1111 ...
        0.1111     0.1111
        0.1111     0.1111
```

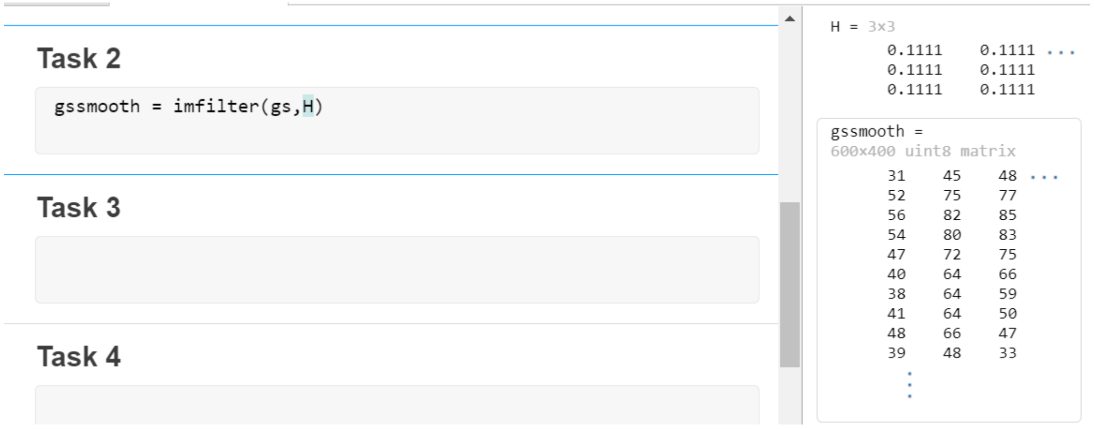
Task 2

```
F = fspecial("average",n)
```

You can apply a filter `F` to an image `I` by using the `imfilter` function.

```
Ifltr = imfilter(I,F);
```

TASK 2
Apply the filter `H` to the grayscale image `gs`. Store the result in `gssmooth`.



Task 2

```
gssmooth = imfilter(gs,H)
```

Task 3

Task 4

```
H = 3x3
        0.1111     0.1111 ...
        0.1111     0.1111
        0.1111     0.1111

gssmooth =
600x400 uint8 matrix
        31    45    48 ...
        52    75    77
        56    82    85
        54    80    83
        47    72    75
        40    64    66
        38    64    59
        41    64    50
        48    66    47
        39    48    33
         ⋮
```

Now that you have filtered the image, you can binarize it. Did the filter reduce the amount of noise in the binary image?

TASK 3
Create a binary image from `gssmooth` using `imbinarize` with the `"adaptive"` setting and the `"ForegroundPolarity"` set to `"dark"`. Store the result in `BWsmooth` and display it.

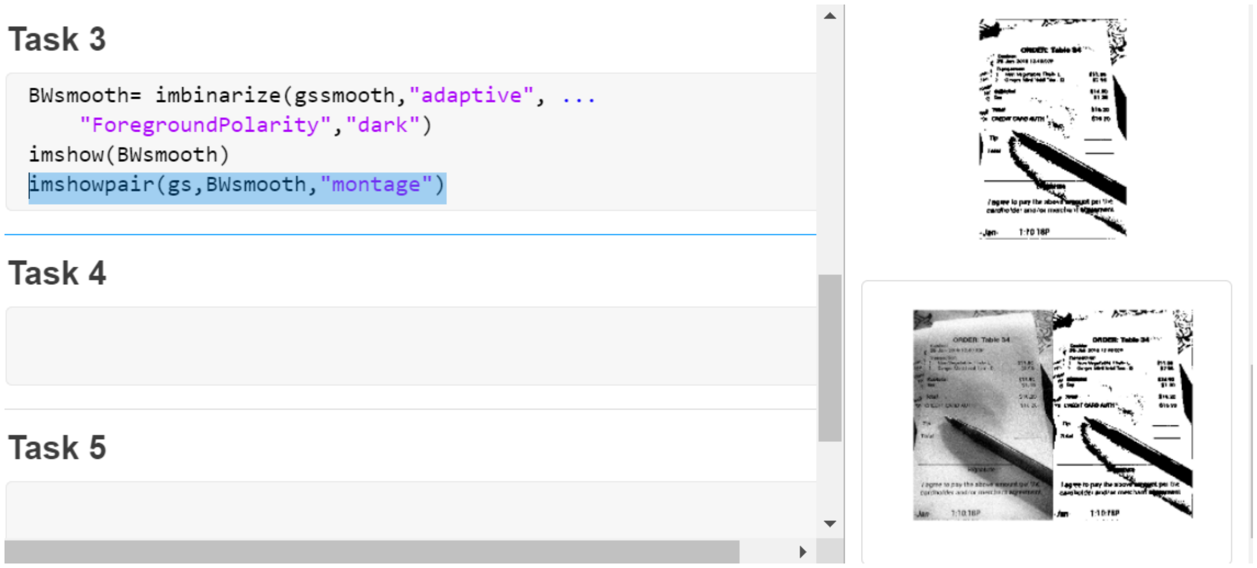
Task:4

If you look closely, you can see that filtering introduced a small, unwanted dark outline to parts of the binary image border. That's because the default setting of `imfilter` sets pixels outside the image to zero.

To adjust this behavior, use the `"replicate"` option.

```
  Ifltr = imfilter(I,F,"replicate");
```

Instead of zeros, the `"replicate"` option uses pixel intensity values on the image border for pixels outside the image.
TASK
Apply the filter `H` to the grayscale image `gs` as you did in Task 2, but use the `"replicate"` option. Store the filtered image in `gssmooth`.



Task 3

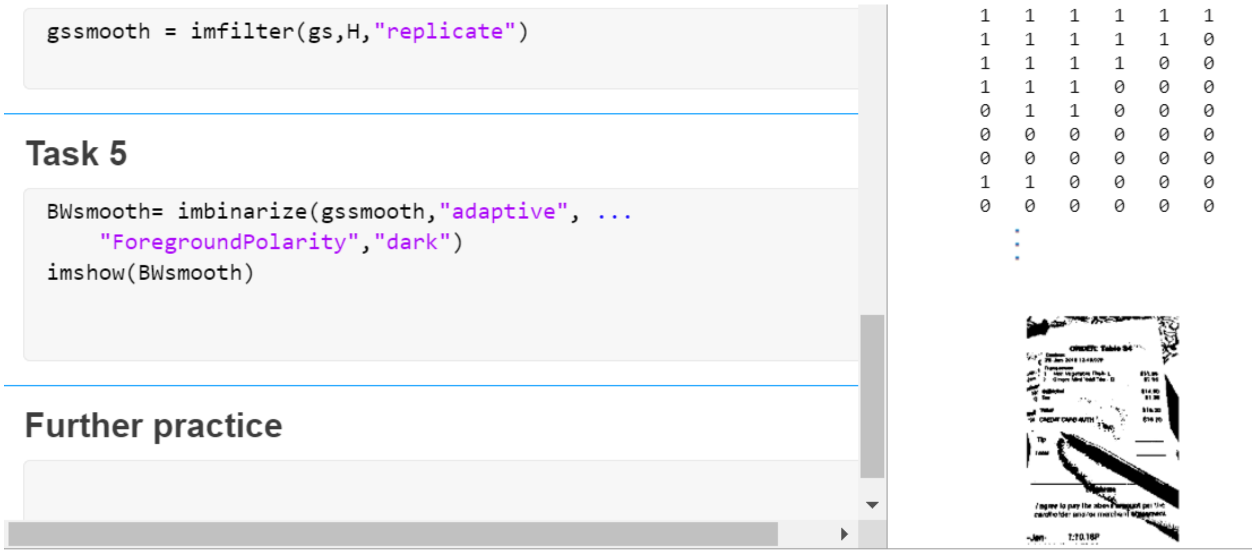
```
BWsmooth= imbinarize(gssmooth,"adaptive", ...
    "ForegroundPolarity","dark")
imshow(BWsmooth)
imshowpair(gs,BWsmooth,"montage")
```

Task 4

Task 5

Task:5

Because you used the "replicate" option, the filtered image should no longer contain an artificial border. Use the `imbinarize` function to create a new binary image. Has the border been removed?
TASK
Binarize `gssmooth` as you did in Task 3, and store the result in `BWsmooth`. Display `BWsmooth`.



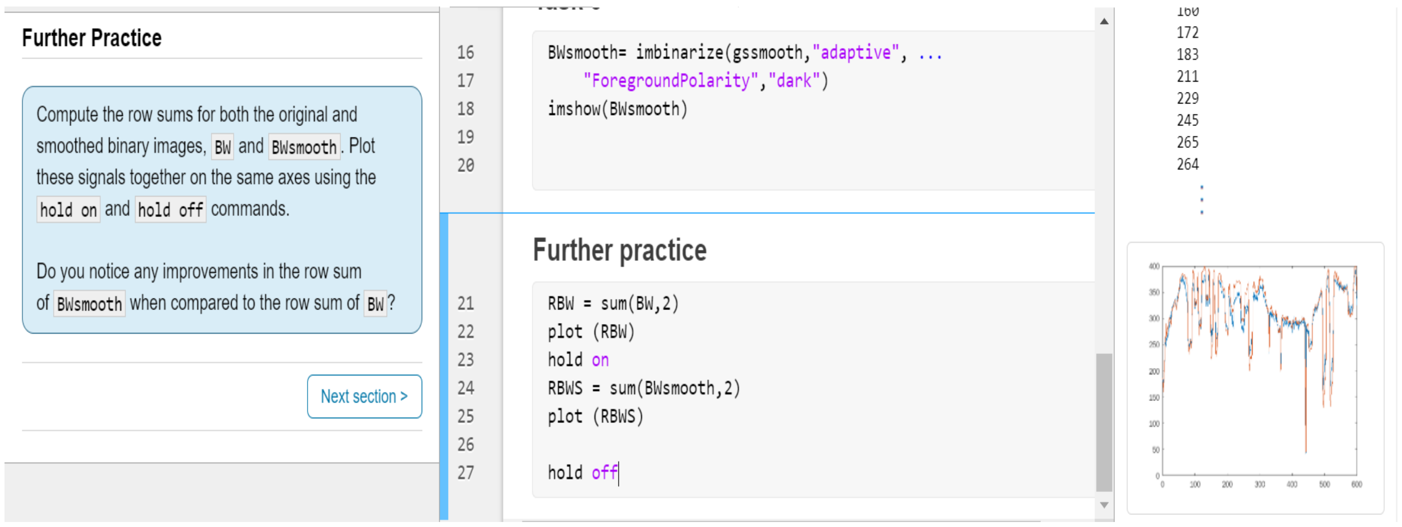
```
gssmooth = imfilter(gs,H,"replicate")
```

Task 5

```
BWsmooth= imbinarize(gssmooth,"adaptive", ...
    "ForegroundPolarity","dark")
imshow(BWsmooth)
```

Further practice

```
1    1    1    1    1    1
1    1    1    1    1    0
1    1    1    1    0    0
1    1    1    0    0    0
0    1    1    0    0    0
0    0    0    0    0    0
0    0    0    0    0    0
1    1    0    0    0    0
0    0    0    0    0    0
  :
```



Further Practice

Compute the row sums for both the original and smoothed binary images, `BW` and `BWsmooth`. Plot these signals together on the same axes using the `hold on` and `hold off` commands.

Do you notice any improvements in the row sum of `BWsmooth` when compared to the row sum of `BW`?

```
BWsmooth= imbinarize(gssmooth,"adaptive", ...
    "ForegroundPolarity","dark")
imshow(BWsmooth)
```

Further practice

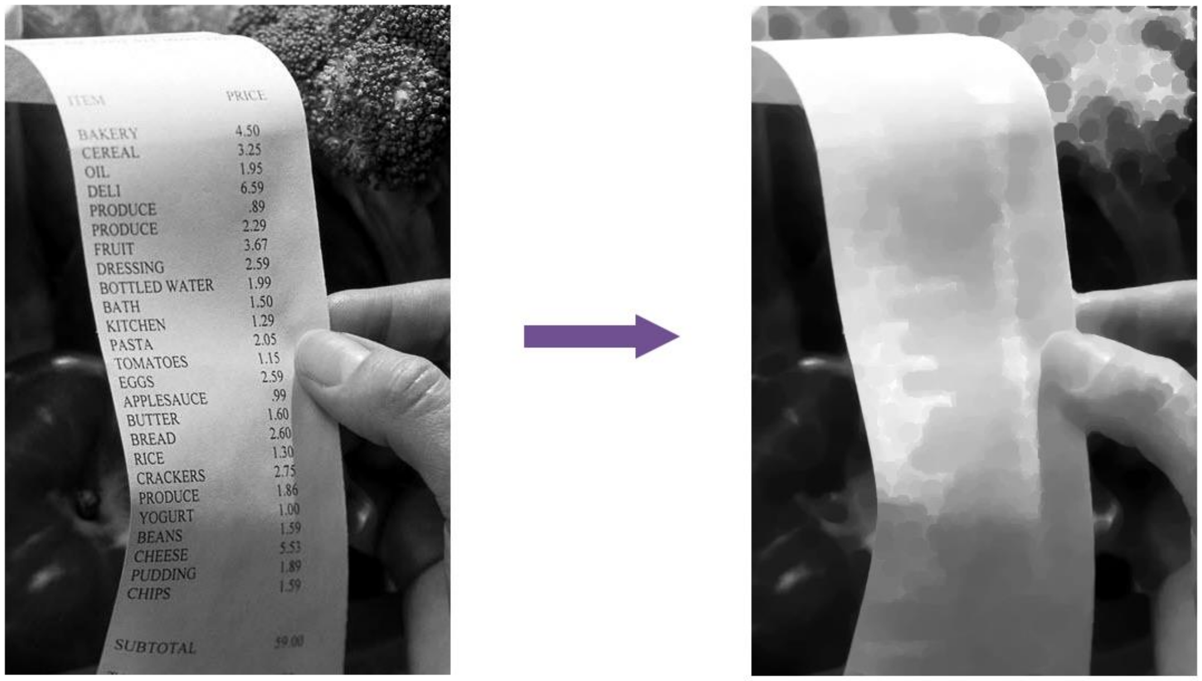
```
RBW = sum(BW,2)
plot (RBW)
hold on
RBWS = sum(BWsmooth,2)
plot (RBWS)

hold off
```

```
160
172
183
211
229
245
265
264
 :
```

Isolating the Image Background

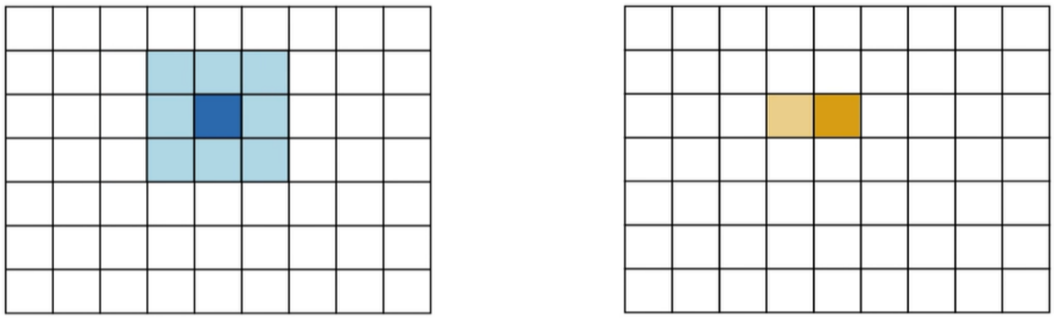
Receipt images that have a busy background are more difficult to classify because artifacts pollute the row sum. To mitigate this issue, you can isolate the background and then remove it by subtraction.

In a receipt image, the background is anything that is not text, so isolating the background can be interpreted as removing the text. One way to remove text from an image is to use *morphological operations*.

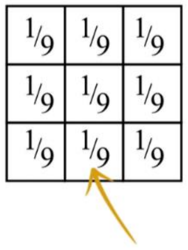


ITEM	PRICE
BAKERY	4.50
CEREAL	3.25
OIL	1.95
DELI	6.59
PRODUCE	.89
PRODUCE	2.29
FRUIT	3.67
DRESSING	2.59
BOTTLED WATER	1.99
BATH	1.50
KITCHEN	1.29
PASTA	2.05
TOMATOES	1.15
EGGS	2.59
APPLESAUCE	.99
BUTTER	1.60
BREAD	2.60
RICE	1.30
CRACKERS	2.75
PRODUCE	1.86
YOGURT	1.00
BEANS	1.59
CHEESE	5.53
PUDDING	1.89
CHIPS	1.59
SUBTOTAL	59.00

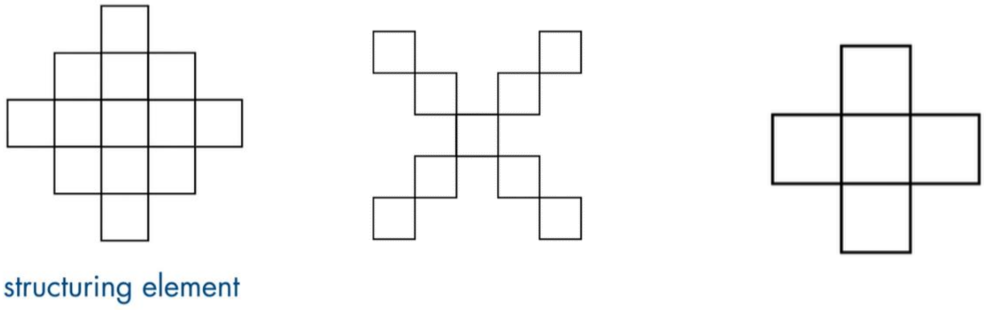
Morphological Operation is kind of sliding Window to look at the neighborhood of the pixel then perform some operation to compute the image.



In Linear output image filtering , we change the values of filter.



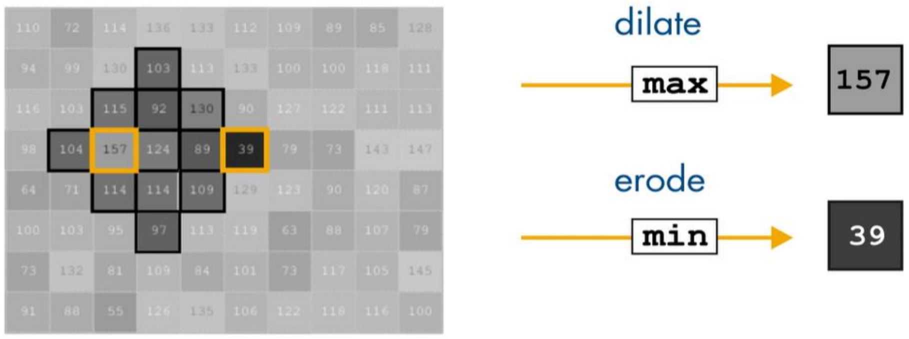
With morphological operation we change the output images by changing shape & size of window slide. This window is called Structuring element.



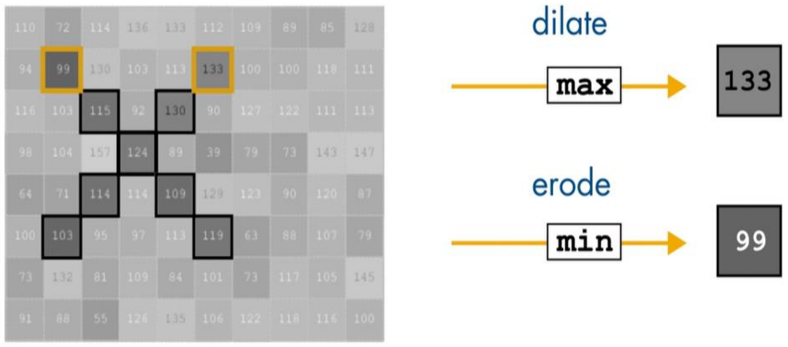
In Morphological operation , 2 operation is done:

Dilation -> Brightest pixel

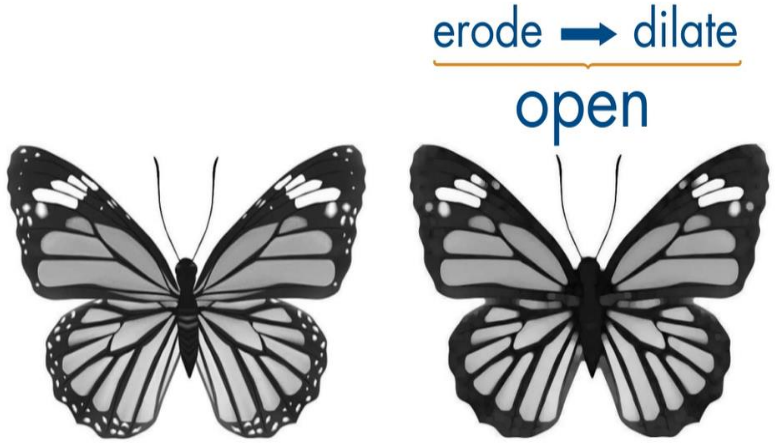
Erosion -> Darkest pixel



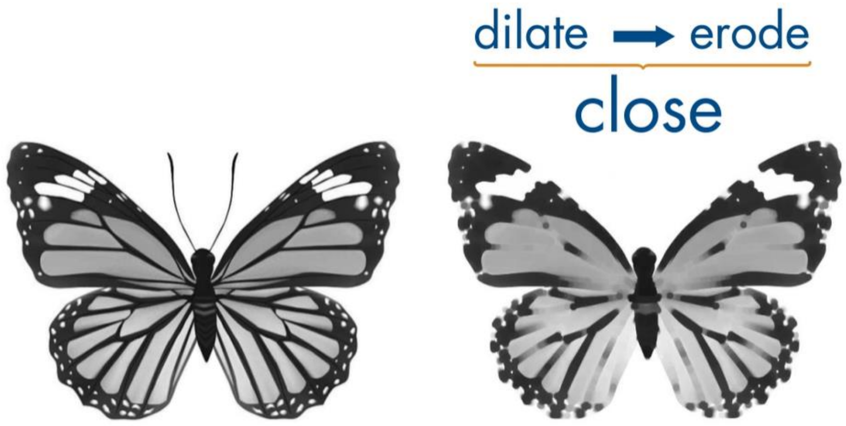
This doesn't result room for customization. To change the result we have to change the shape of window.



Connects darker part of the image and removes the brighter area which are narrow from structural elements and leaves the larger and wider.



Connects Brighter part of image.

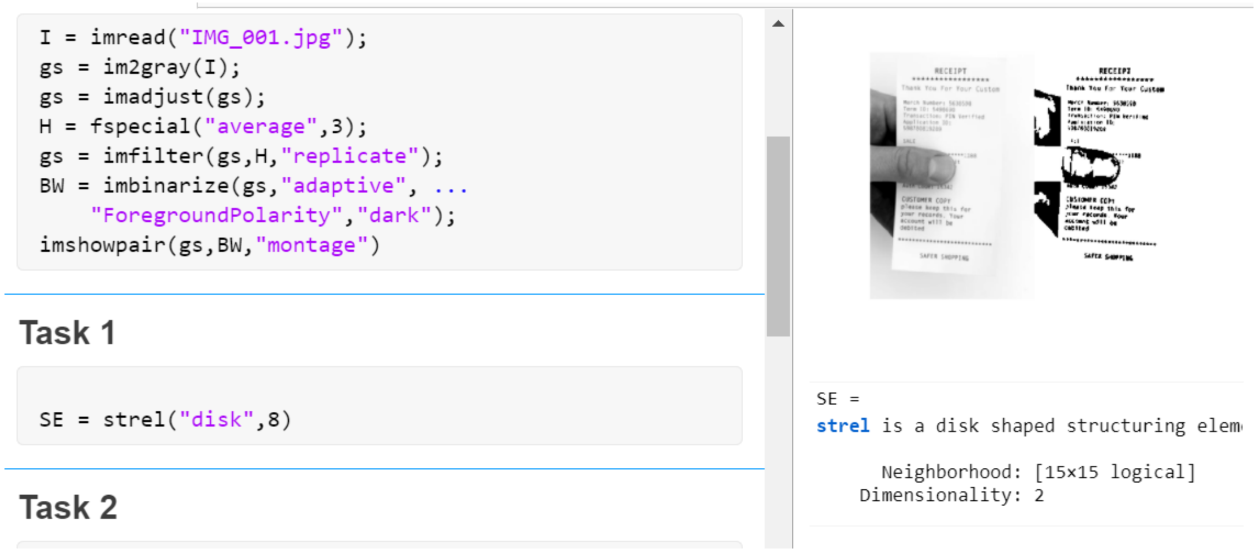


Remove smaller narrow darker arreas.

In a receipt image, the background is anything that is not text. By using a structuring element larger than a typical letter, each "window" captures what's behind the text and not the text itself.

Structuring elements are created using the `strel` function.

SE = strel("diamond",5)



```
I = imread("IMG_001.jpg");
gs = im2gray(I);
gs = imadjust(gs);
H = fspecial("average",3);
gs = imfilter(gs,H,"replicate");
BW = imbinarize(gs,"adaptive", ...
    "ForegroundPolarity","dark");
imshowpair(gs,BW,"montage")
```

Task 1

```
SE = strel("disk",8)
```

Task 2

```
SE =
strel is a disk shaped structuring elem

      Neighborhood: [15×15 logical]
    Dimensionality: 2
```

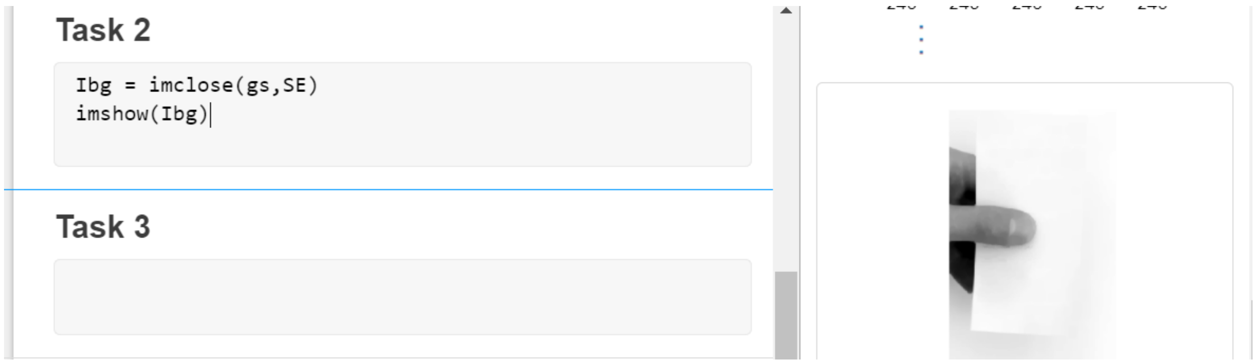
The foreground (the text) is dark, and the background is bright. *Closing* the image will emphasize and connect the brighter background, removing the thin dark lettering. Larger background objects, like the thumb, will remain. The final result: an image of the background without the text.

To perform a closing operation on an image `I` with structuring element `SE`, use the `imclose` function.

```
Iclosed = imclose(I,SE);
```

TASK

Apply the `imclose` function to `gs` using the structuring element `SE`. Store the result in `Ibg` and display it.



Task 2

```
Ibg = imclose(gs,SE)
imshow(Ibg)
```

Task 3

Now that you've isolated the background `Ibg`, you can remove it from `gs`. In general, you can remove one image from another by subtracting it.

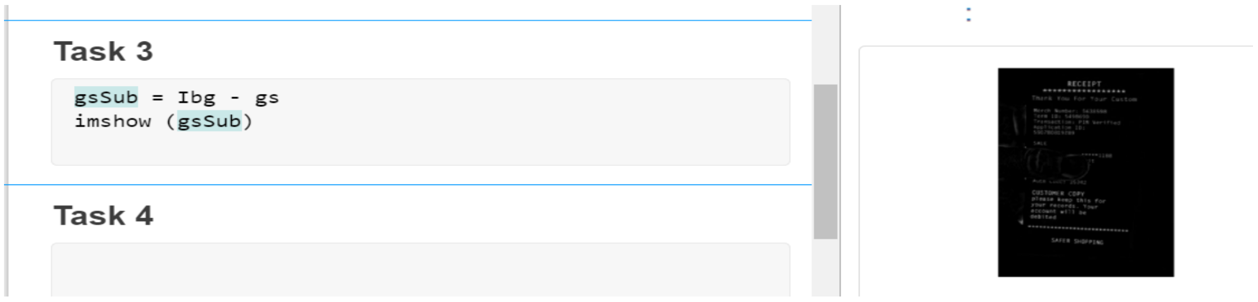
```
Isub = I2 - I1;
```

Because imclose emphasizes brightness in an image, the background image `Ibg` has higher intensities than `gs`. Subtracting `Ibg` from `gs` would result in negative intensities.

To avoid negative intensities, subtract `gs` from `Ibg`.

TASK

Subtract `gs` from `Ibg` and store the result in `gsSub`. Then display `gsSub`.



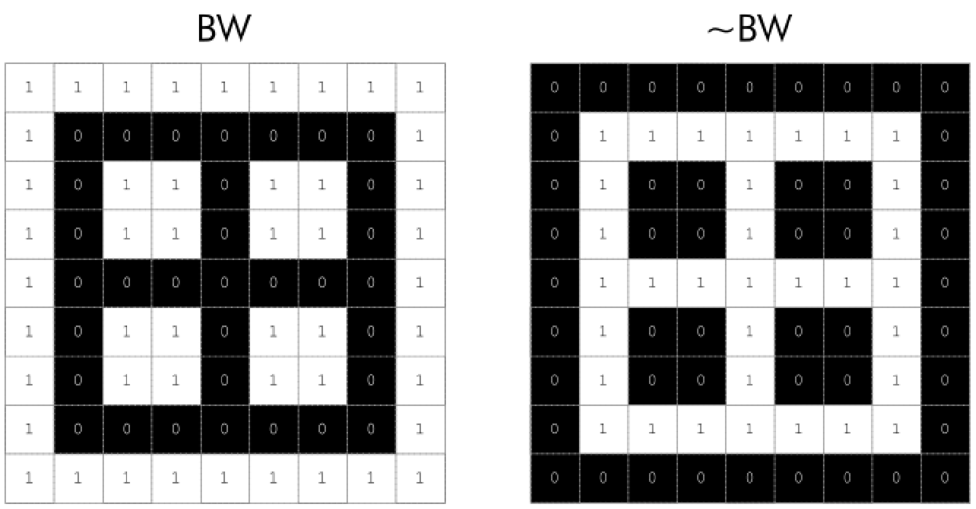
Task 3

```
gsSub = Ibg - gs
imshow (gsSub)
```

Task 4

Inverting a binary image

A binary image is composed of 0s and 1s. In logical terms, 0 = false and 1 = true. The logical NOT operator (**~**) reverses the logical state (true becomes false and vice versa). You can use the **~** operator to invert a binary image, which changes all 1s to 0s, and vice versa.



BW

1	1	1	1	1	1	1	1	1
1	0	0	0	0	0	0	0	1
1	0	1	1	0	1	1	0	1
1	0	1	1	0	1	1	0	1
1	0	0	0	0	0	0	0	1
1	0	1	1	0	1	1	0	1
1	0	1	1	0	1	1	0	1
1	0	0	0	0	0	0	0	1
1	1	1	1	1	1	1	1	1

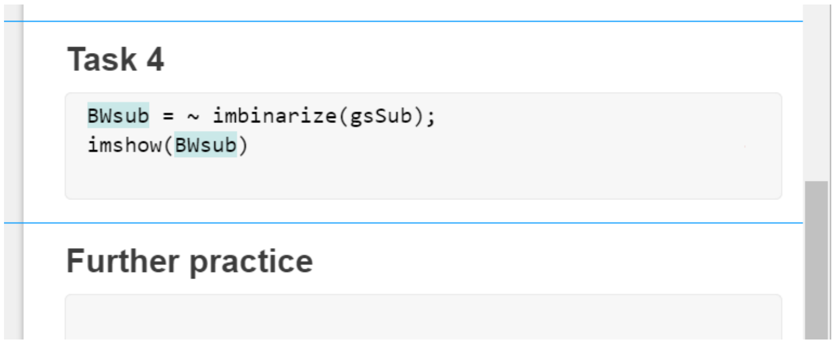
~BW

0	0	0	0	0	0	0	0	0
0	1	1	1	1	1	1	1	0
0	1	0	0	1	0	0	1	0
0	1	0	0	1	0	0	1	0
0	1	1	1	1	1	1	1	0
0	1	0	0	1	0	0	1	0
0	1	0	0	1	0	0	1	0
0	1	1	1	1	1	1	1	0
0	0	0	0	0	0	0	0	0

Did you notice anything strange about `gsSub`?

The subtraction in the previous step inverted the intensities, so the text appears white and the background black. To restore the original order, you need to invert the image again.

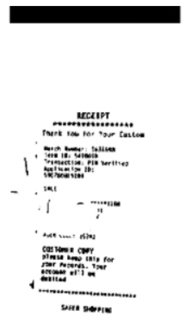
First, generate the binary image using `imbinarize`. Then invert the result using the NOT operator (~).
TASK
Use `imbinarize` on `gsSub` without any options, and use the NOT operator (~) to invert the binary image. Store the result in `BWsub` and display it.



Task 4

```
BWsub = ~ imbinarize(gsSub);
imshow(BWsub)
```

Further practice



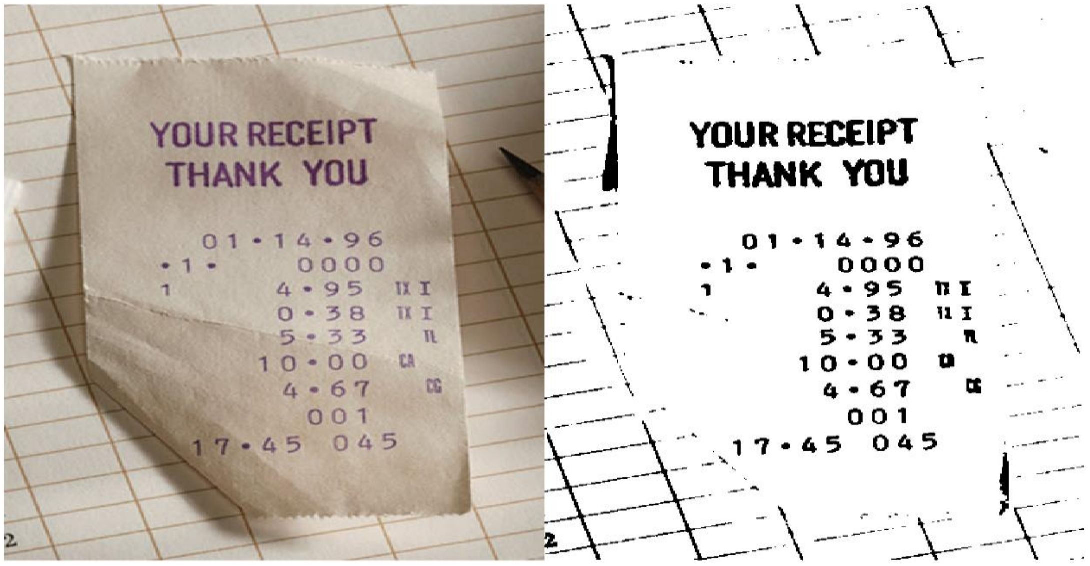
The sequence of operations used here is very common: closing an image, then subtracting the original image from the closed image. It's known as the "bottom hat transform," and you can use the `imbothat` function in MATLAB to perform it. Try applying `imbothat` to `gs` using the structuring element `SE`.

```
BW = imbothat(gs,SE);
```

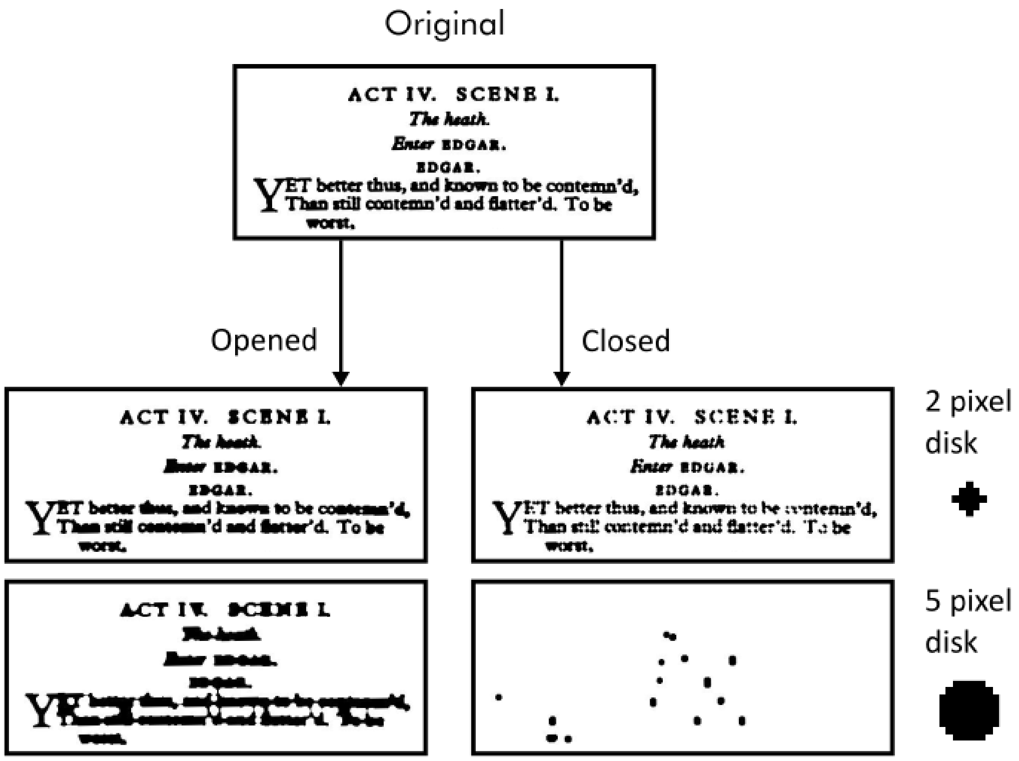
Enhancing text patterns

The process from the previous section doesn't remove the grid in the background of this receipt. The grid pattern is too thin to be included in the background generated by the closing operation.

Morphological operations are useful not only for removing features from an image but for augmenting features. You can use morphology to enhance the text in the binary image and improve the row sum signal.



Morphological opening expands the dark text regions, while closing diminishes them. Increasing the size of the structuring element increases these effects.

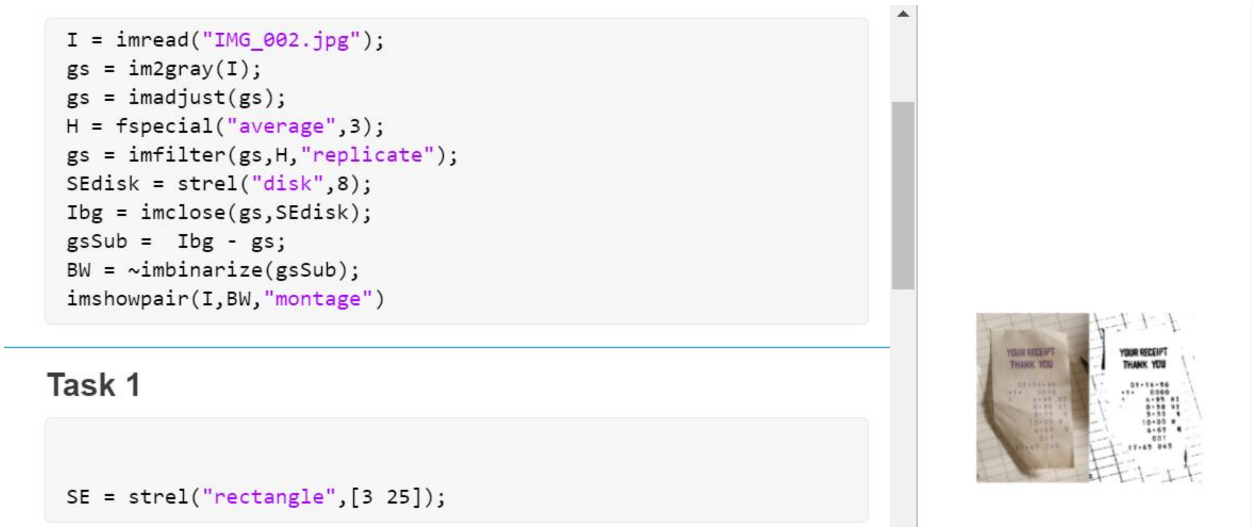


To enhance a particular shape in an image, create a structural element in the shape of what you want to enhance. Here, each row of text should become a horizontal stripe, which looks like a short, wide rectangle.

You can create a rectangular structuring element using `strel` with a size array in the format `[height width]`.

```
SE = strel("rectangle",[10 20]);
```

To determine the size of the rectangle, you can use features of a typical receipt. The rectangle should be short enough that the white space between rows of text is preserved, and it should be wide enough to connect adjacent letters.
TASK
Use `strel` to create a rectangular structuring element with height 3 pixels and width 25 pixels. Store the result in `SE`.



```
I = imread("IMG_002.jpg");
gs = im2gray(I);
gs = imadjust(gs);
H = fspecial("average",3);
gs = imfilter(gs,H,"replicate");
SEdisk = strel("disk",8);
Ibg = imclose(gs,SEdisk);
gsSub =  Ibg - gs;
BW = ~imbinarize(gsSub);
imshowpair(I,BW,"montage")
```

Task 1

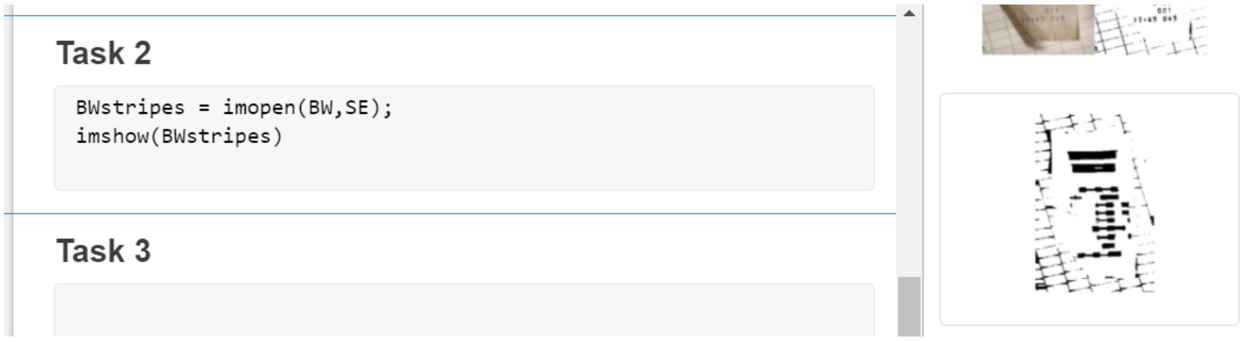
```
SE = strel("rectangle",[3 25]);
```

To isolate the background, you wanted to eliminate the dark text by emphasizing the bright regions. To do this, you used `imclose` to close the image.

Here**, you want to *emphasize* the dark text instead**, so you need to perform the opposite operation and open the image using `imopen`.

```
Iopened = imopen(I,SE);
```

Using `imopen` with a wide rectangular structuring element turns lines of text into black horizontal stripes, which augments the valleys in the row sum signal.
TASK
Apply a morphological opening operation to `BW` using the rectangular structuring element `SE`. Store the result in `BWstripes` and display it.



Task 2

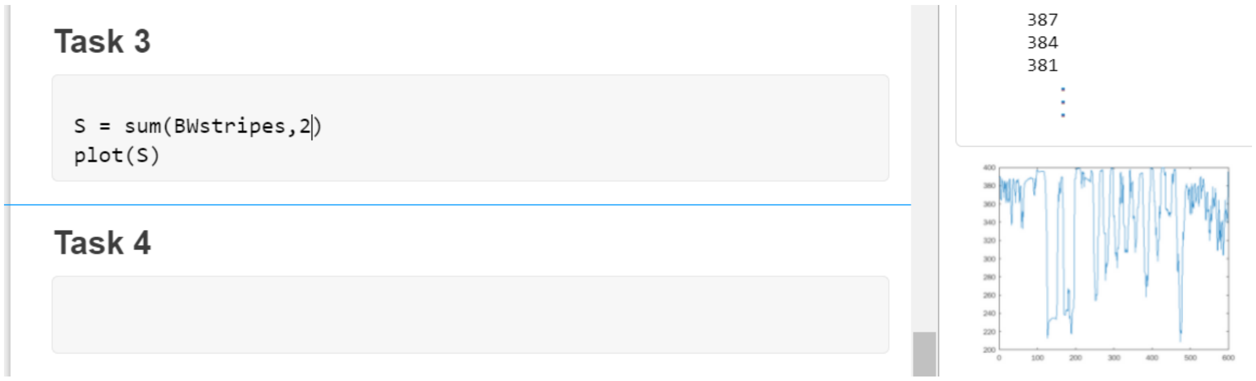
```
BWstripes = imopen(BW,SE);
imshow(BWstripes)
```

Task 3

You can analyze whether or not the opening operation has improved the image processing algorithm by plotting the row sum.

TASK

Sum the rows of `BWstripes` and store the result in `S`. Plot `S` to observe the pattern.



Task 3

```
S = sum(BWstripes,2)
plot(S)
```

```
387
384
381
 ⋮
```

Task 4

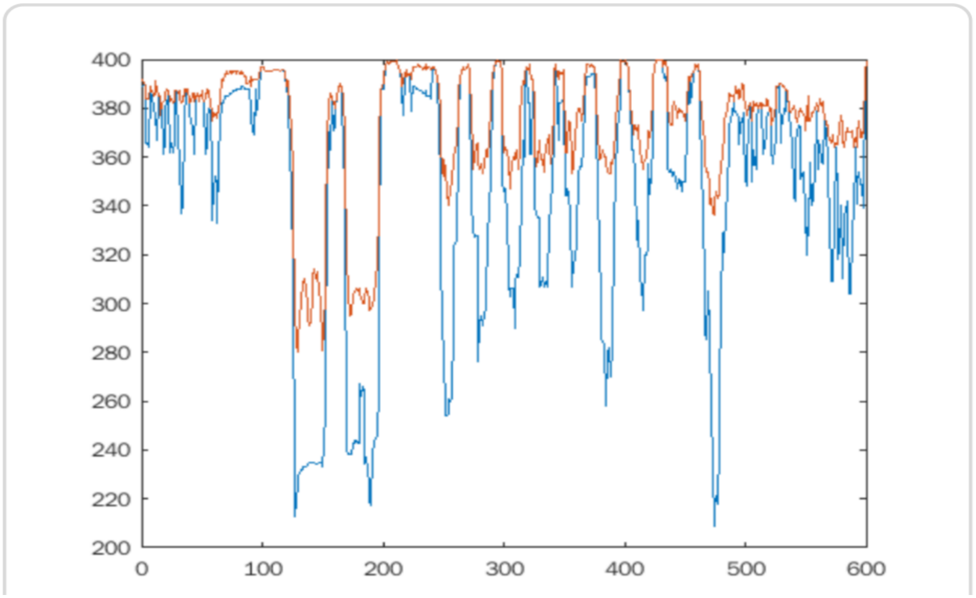
The improvement becomes more noticeable if you plot the original row sum signal alongside the opened image row sum.

TASK

Compute the row sum of the original signal `BW` and store the result in `Sbw`. Use `hold on` and `hold off` to add a plot of `Sbw` to your current plot.

Task 4

```
Sbw = sum(BW,2)
plot(S)
hold on
plot(Sbw)
hold off
```



The text rows generate more pronounced valleys after being enhanced with the opening operation. There is also little to no loss of row definition from the closing operation because the rectangular structuring element was only three pixels high.

By using a very wide rectangular structuring element, there is a risk of amplifying valleys and peaks in the row sums of non-receipt images, which could lead to misclassifications. This risk is acceptable as long as the amplified row sum pattern in receipt images is more pronounced than the unwanted amplification in non-receipt images.