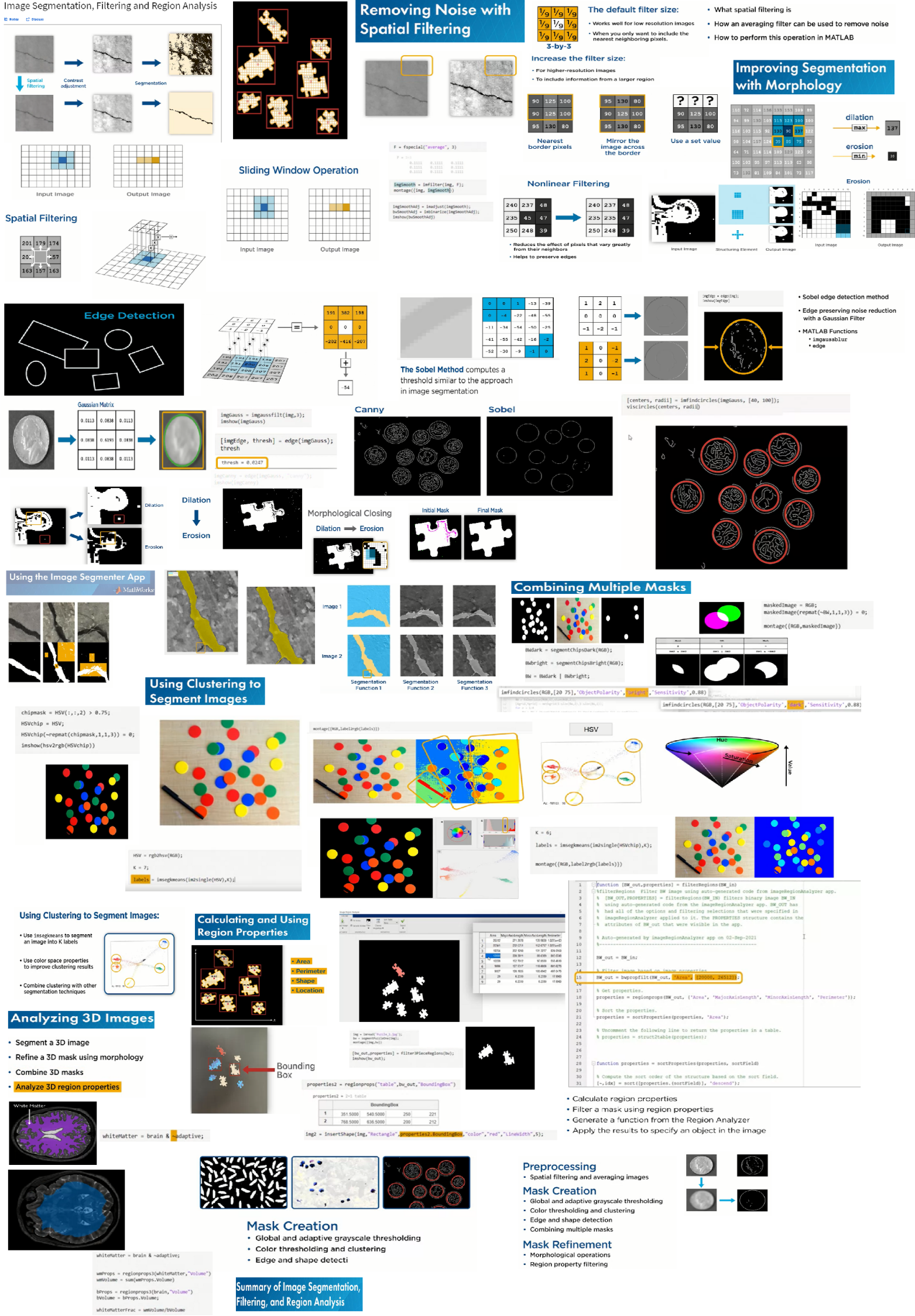


Image Segmentation, Filtering and Region Analysis
Spatial filtering
Contrast adjustment
Segmentation
Input Image
Output Image
Removing Noise with Spatial Filtering
The default filter size:
3-by-3
Works well for low resolution images
When you only want to include the nearest neighboring pixels.
What spatial filtering is
How an averaging filter can be used to remove noise
How to perform this operation in MATLAB
Increase the filter size:
For higher-resolution images
To include information from a larger region
Nearest border pixels
Mirror the image across the border
Use a set value
Improving Segmentation with Morphology
dilation
max
erosion
min
Erosion
Input Image
Structuring Element
Output Image
Sliding Window Operation
Input Image
Output Image
Nonlinear Filtering
Reduces the effect of pixels that vary greatly from their neighbors
Helps to preserve edges
Spatial Filtering
Edge Detection
The Sobel Method computes a threshold similar to the approach in image segmentation
Sobel edge detection method
Edge preserving noise reduction with a Gaussian Filter
MATLAB Functions
imgaussblur
edge
Gaussian Matrix
Canny
Sobel
Dilation
Erosion
Morphological Closing
Initial Mask
Final Mask
Using the Image Segmenter App
MathWorks
Image 1
Image 2
Segmentation Function 1
Segmentation Function 2
Segmentation Function 3
Combining Multiple Masks
Using Clustering to Segment Images
HSV
Hue
Saturation
Value
Using Clustering to Segment Images:
Use imsegkmeans to segment an image into K labels
Use color space properties to improve clustering results
Combine clustering with other segmentation techniques
Calculating and Using Region Properties
Area
Perimeter
Shape
Location
Analyzing 3D Images
Segment a 3D image
Refine a 3D mask using morphology
Combine 3D masks
Analyze 3D region properties
White Matter
Bounding Box
Calculate region properties
Filter a mask using region properties
Generate a function from the Region Analyzer
Apply the results to specify an object in the image
Mask Creation
Global and adaptive grayscale thresholding
Color thresholding and clustering
Edge and shape detecti
Preprocessing
Spatial filtering and averaging images
Mask Creation
Global and adaptive grayscale thresholding
Color thresholding and clustering
Edge and shape detection
Combining multiple masks
Mask Refinement
Morphological operations
Region property filtering
Summary of Image Segmentation, Filtering, and Region Analysis

Congratulations! You passed!

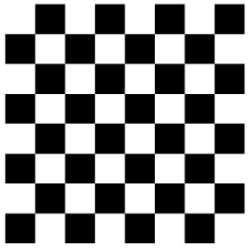
Grade received 100% | Latest Submission Grade 100% | To pass 80% or higher

Go to next item

1. Questions 1 and 2: Working with Test Patterns

1 / 1 point

One way to get a feel for how spatial filters affect an image is to apply them to very simple test patterns. The following checkerboard pattern is composed of alternating 16-by-16 pixel squares.



You can create this image in MATLAB with the following commands:

```
A = ones(16);
B = zeros(16);
C = [A, B; B, A];
C = [C, C; C, C];
img = [C, C; C, C];
```

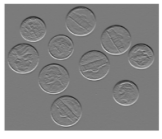
Note that the image produced is of type `double`

3. Visualizing Gradients

0 / 1 point

Load the '`coins.png`' image included with MATLAB. Detect edges using the `edge` function with the Sobel method, and return the vertical and horizontal gradients.

Which image best represents the gradient highlighting the vertical edges in this image?

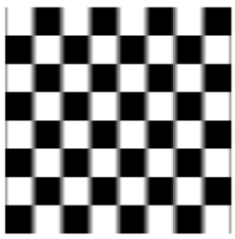


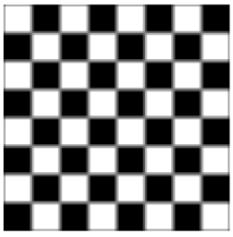
Incorrect
This contains the vertical gradient, but the negative values have all been set to zero. When passing an image of data type `single` to the `imshow` function, you also need to pass a set of empty brackets as the data range to correctly display all the values.

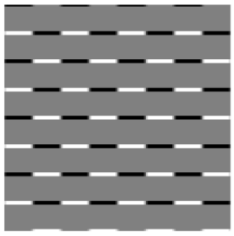
Correct
This image shows the vertical gradient. Note how the brightest regions are on the right edges of the coins and the darkest regions are on the left edges.

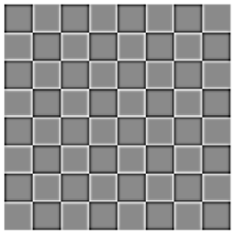
Spatial Filters

Which of the following images is the result of applying a 3x3 averaging filter to this image?







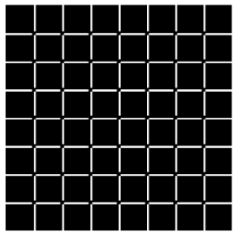


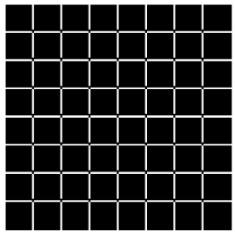
Correct
This image was produced with the averaging filter. Note how all of the boundaries between the white an black sections are equally blurred.

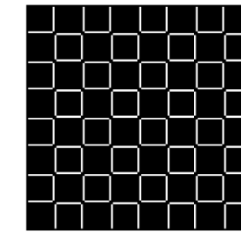
2. Edge Detection

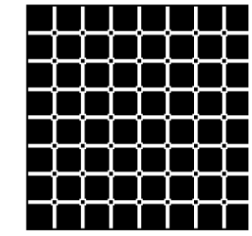
1 / 1 point

Which of the following images is the result of applying Canny edge detection to the checkerboard image?





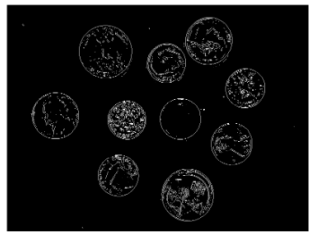


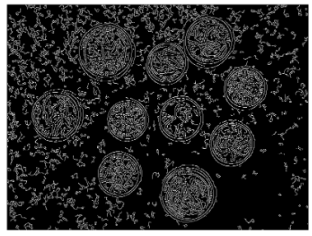


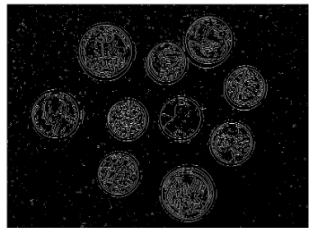
Correct
This is the result of Canny edge detection.

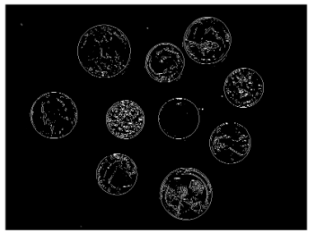
4. Detecting Edges in Coins

Load the `coins1.jpg` image and resize it to a resolution of 600 by 800 pixels. Apply a Gaussian filter with a standard deviation of 0.5. Which of these images shows the edges detected by the `edge` function with the Canny method?









Correct
This is the result of Canny edge detection.

5. Questions 5 and 6: Counting Coins

1 / 1 point

The next two questions involve using the `imfindcircles` and `viscircles` functions. For each question, you will need to load the image, resize it, and apply the specified Gaussian blur with the `imgaussfilt` function.

Counting Coins 1

Load the `coins3.jpg` image and rescale the image to 800 by 800 pixels. Blur the image using a Gaussian blur with a standard deviation (`sigma`) of 1. Use the `imfindcircles` function to find only the four largest coins. What is the lower value of the radius search to eliminate the smaller coins?

61

Correct

6. Counting Coins 2

1 / 1 point

Load the `coins2.jpg` image and rescale the image to 900 by 1200 pixels. Blur the image using a Gaussian blur with a standard deviation (`sigma`) of 1. Use the `imfindcircles` function to count the number of coins with radius less than 100.

8

Correct

Congratulations! You passed!

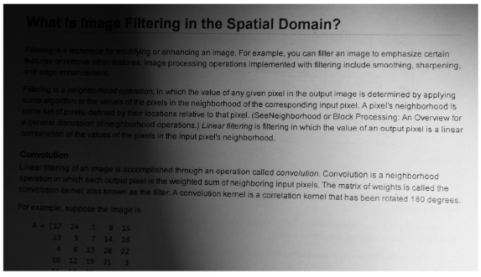
Grade received 100% | Latest Submission Grade 100% | To pass 80% or higher

Go to next item

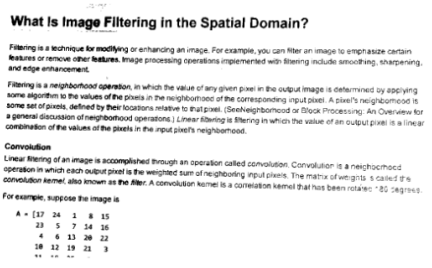
1. **For the questions in this quiz, you will be loading images into the Image Segmenter App. In each case, when prompted to "Adjust Image?", you need to click "Yes" to get the correct results.** 1 / 1 point

Use the following code to load and view an image of text data included with your version of MATLAB. Notice the illumination is very uneven.

```
textImage = imread("printedtext.png");
imshow(textImage)
```



Using the Image Segmenter App, which approach below gives you the following segmented image?



- ○ Manual threshold with a Threshold value of 60
- ◉ Adaptive Threshold with a bright Foreground Polarity and a Sensitivity value of 90
- ○ Adaptive threshold with a bright foreground polarity and a Sensitivity value of 50
- ○ Manual Threshold with a value of of 200
- ○ Global Threshold

Correct
An adaptive threshold is required due to the uneven illumination. A Sensitivity value of 90 works well to

4. Now assume you want to segment *only* the nickels (the larger of the two types of coin in this image). Can you find one or more ways that work?

- ☐ Auto Cluster, then Fill Holes
- ☐ Choose a manual threshold of 175 to differentiate between the two types of coins.
- ☑ Use the Find circles approach with a minimum diameter of 55 pixels.

Correct
Correct. All off the smaller coins are below this threshold and will not be included in the mask.

- ☑ Use Find Circles to find all the circles. Then use the "open" morphological operator with a disk shape and radius of 26.

Correct
Correct. Choosing a structuring element that is large enough to cover the small coins but not the large coins will remove the small coins from the mask.

5. To apply an approach developed in the Image Segmenter app on other images, the best practice is to:

- ○ The Image Segmenter App is meant for manual segmentation. You cannot repeat the process on other images.
- ○ Write down the steps you took. Load other images into the app and repeat the steps.
- ○ Load an entire folder of images into the Image Segmenter App to apply the steps to all images at once.
- ◉ Generate a function from the app and apply the function to your other images.

Correct

2. Use the following code to load, convert to grayscale, and view an image of a crack included with your course files. 1 / 1 point

```
imshow(crackImage)
```

Which approach or approaches below will produce the following mask (select all approaches that work)?

- ☑ 1. Global Threshold
  2. Invert mask
  3. Close mask with a disk of radius 3
  4. Open mask with a disk of radius 3

Correct
Correct!

This is one of the two correct sets of steps here.

- ☑ 1. Global Threshold
  2. Invert Mask
  3. Close mask with a disk of radius 3
  4. Erode mask with a disk of radius 3
  5. Dilate mask with a disk of radius 3

Correct
Correct!

Notice, erosion followed by dilation with the same structuring element is equivalent to opening.

This is one of the two correct sets of steps here.

- ☐ 1. Adaptive Threshold with bright Foreground Polarity and a sensitivity of 90
  2. Erode mask with a disk of radius 4
  3. Fill holes
  4. Invert mask

- ☐ 1. Manual Threshold with a value of 57
  2. Invert mask
  3. Close mask with a disk of radius 3

3. Use the following code to load and view an image of coins included with MATLAB. 1 / 1 point

```
coinImage = imread("coins.png");
imshow(coinImage)
```

Which approach or approaches below will segment the coins as foreground with no holes, negligible missing foreground, and no extra foreground artifacts:

- ☐ Manual Threshold with a Threshold value of 64.
- ☑ Auto Cluster, then Fill Holes

Correct
Correct!

This is one of the two correct approaches here.

- ☐ Find Circles with the following settings:
  - Min. Diameter: 50
  - Max. Diameter: 150
  - Number of Circles: Inf
  - Foreground Polarity: bright
  - Sensitivity: 0.85

- ☑ Find Circles with the following settings:
  - Min. Diameter: 30
  - Max. Diameter: 150
  - Number of Circles: Inf
  - Foreground Polarity: bright
  - Sensitivity: 0.90

Correct
Correct! All the coins in this image are well between 30 and 150 pixels in diameter.

This is one of the two correct approaches here.

Congratulations! You passed!

Grade received 100%

Latest Submission Grade 100%

To pass 75% or higher

Go to next item

1. Given two binary masks of puzzle pieces, one that identifies all puzzle pieces ("maskAll") and one that only identifies back-facing puzzle pieces ("maskBack"), how can you combine these masks to obtain only the front-facing puzzle pieces? **1 / 1 point**

- ○
```
maskAll & maskBack
```
- ○
```
maskAll | ~maskBack
```
- ◉
```
maskAll & ~maskBack
```
- ○
```
~maskAll | maskBack
```

Correct

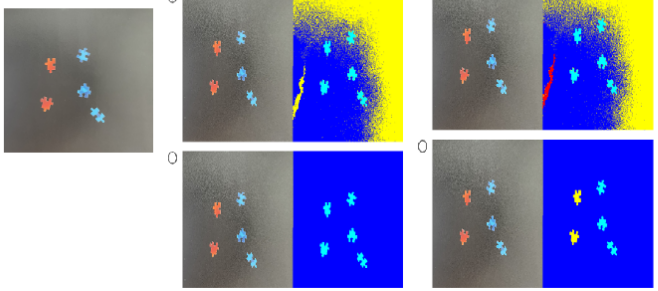
2. Which of the following pieces of code takes a color image, `img`, and uses a binary mask, `BW`, to create a masked image? **1 / 1 point**

- ○
```
maskedImage(repmat(~BW,3)) = 0;
```
- ○
```
maskedImage = img;
maskedImage(repmat(BW,1,1,3)) = 0;
```
- ○
```
maskedImage(~BW) = 0;
```
- ◉
```
maskedImage = img;
maskedImage(repmat(~BW,1,1,3)) = 0;
```

Correct

3. Import the image, "Puzzle_06.jpg", found in the course files and convert it to HSV. **1 / 1 point**

Assume you want to differentiate between the red and blue puzzle pieces. Perform K-means clustering to create a matrix with three labels, one for each color of puzzle piece and the background. Which image below most closely resembles your result?



Correct
The variation in the background causes all the puzzle pieces to be assigned to a single label while the background has multiple labels.

4. Which response below provides the best explanation for the result in the previous question?

- ○ Four clusters should be used for this image: two for the background variation and one for each color.
- ○ The `imsegkmeans` function returns a labeled matrix that accurately identifies the background and each color of puzzle piece.
- ○ Because the number of background pixels is so much larger than puzzle pieces, the `imsegkmeans` does not distinguish between the different colored puzzle pieces.
- ◉ The variation in the background pixels results in the background being divided into multiple labels rather than separating the puzzle pieces by color.

Correct
Yes - eliminating the background first would be a good first step before applying clustering.

5. Which of the following statement about morphology are true (select all that apply)?

- ☐ you can only use morphology when improving segmentation of grayscale images
- ☑ you must specify the size and shape of a structuring element

Correct

- ☑ you create a structuring element with the `strel` function

Correct

6. Assume you want to use a rectangular structuring element with size [3,6] to expand then shrink a foreground mask "BW". Which of the following code segments accomplishes this task? **1 / 1 point**

- ○
```
se = strel("rectangle",[3,6]);
BW = imopen(BW, se);
BW = imclose(BW, se);
```
- ◉
```
se = strel("rectangle",[3,6]);
BW = imdilate(BW, se);
BW = imerode(BW, se);
```
- ○
```
se = strel("rectangle")
BW = imdilate(BW, se);
BW = imerode(BW, se);
```
- ○
```
BW = imdilate(BW, "rectangle", [3,6]);
BW = imerode(BW, "rectangle", [3,6]);
```

Correct
You first need to set up the correct structuring element. Then, dilate and erode the mask.

Congratulations! You passed!

Grade received 100% **To pass** 80% or higher

Go to next item

1. To answer the next 4 questions, you must: **1 / 1 point**

- Create a 3-dimensional volume object, `vol`, from the `'T2 MRI Scan'` file folder provided in the course files
- Load `brainMask.mat` from the course files

What is the size of the variable `vol`?

- ○ `255x255`
- ○ `240x240x155`
- ○ `256x256x1x192`
- ◉ `256x256x192`

Correct

2. Each voxel (3-dimmensional pixel) in this image has a length, width, and height of 1mm. Calculate brain volume from `brainMask`. What is the volume of this brain in cm^3? **1 / 1 point**

1221.3

Correct

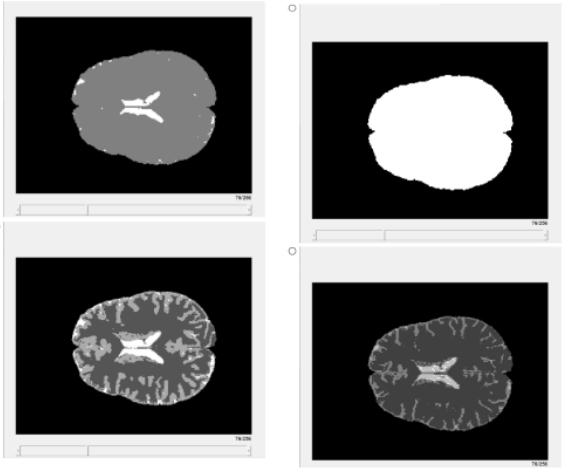
3. Which command will create a volume that is just brain (with the skull and scalp removed)? **1 / 1 point**

- ○ `vol(:).*brainMask(:);`
- ○ `vol(brainMask);`
- ○ `squeeze(brainMask);`
- ◉ `vol(~brainMask) = 0;`

Correct

4. In the video, we segmented the brain into white and gray matter using the Volume Segmenter app. Now you will segment the brain using the grayscale thresholding techniques we practiced with 2D images. **1 / 1 point**

Use thresholding to segment the brain volume into gray matter, white matter, CSF (the bright liquid in and around the brain), and background. Which of the following images shows the result?



Congratulations! You passed!

Grade received 100% **To pass** 66% or higher

Go to next item

1. Assume you are trying to combine two masks. Which of the following operators can you use when combining the masks? **1 / 1 point**

- [ ] All (:)
- [ ] At (@)
- [x] Or (|)

Correct
You can apply the "or" operator on two masks using "BW1 | BW2".

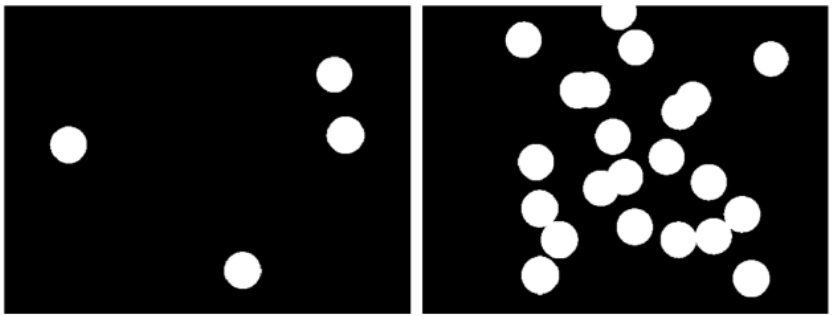
- [x] Not (~)

Correct
You can apply the "not" operation on a mask by running "~BW1". This will invert the mask.

- [x] And (&)

Correct
You can apply the "and" operation on two masks using "BW1 & BW2".

2. Below are two chip masks from the "Combining Multiple Masks" video, which we will refer to here as "MaskLeft" and "MaskRight." How might you combine the masks to get only the background pixels? Select all that apply. **1 / 1 point**



- [ ]
```
~MaskLeft | ~MaskRight
```
- [x]
```
~ (MaskLeft | MaskRight)
```

Correct

- [x]
```
~MaskLeft & ~MaskRight
```

Correct

3. Consider performing clustering on an image with blue and red objects on a black background. How many clusters should you specify when using the `imsegkmeans` function? **1 / 1 point**

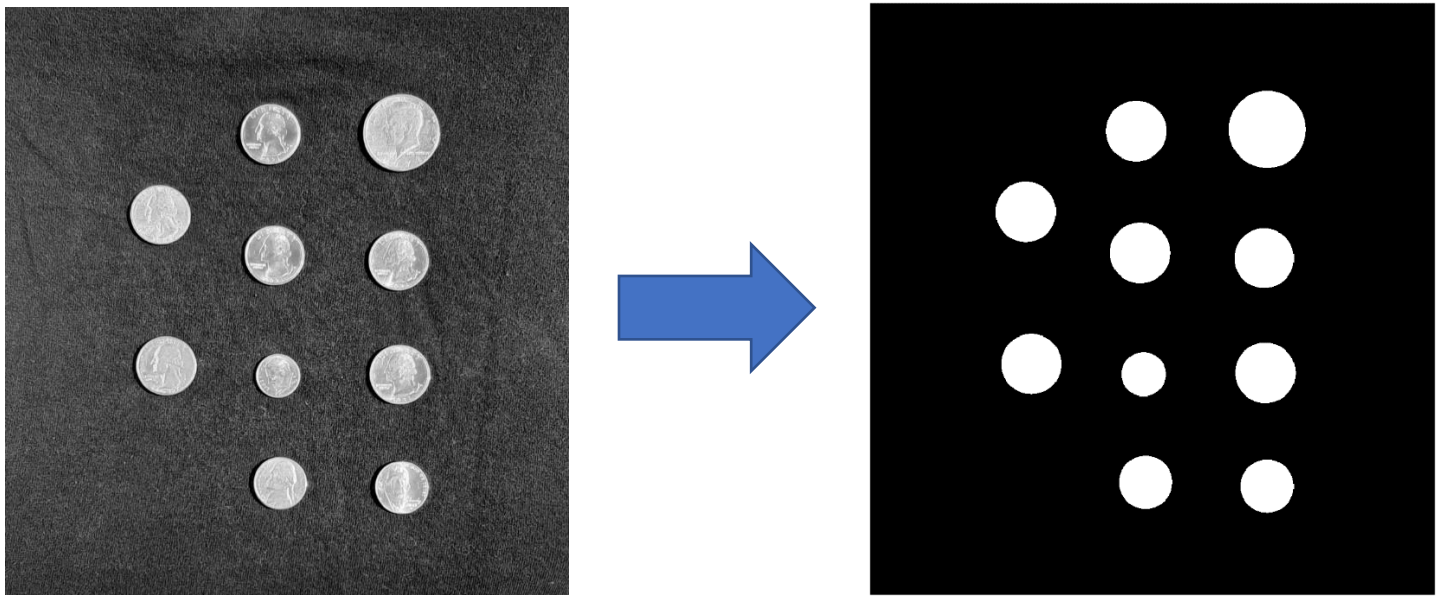
- ( ) 2
- (•) 3
- ( ) 4
- ( ) None of the above. The `imsegkmeans` function automatically chooses a value of K.

Correct
You need to differentiate the red and blue objects from each other and the background.

Counting US Coins by Size:

US coin type and value can be determined by their relative size in an image.

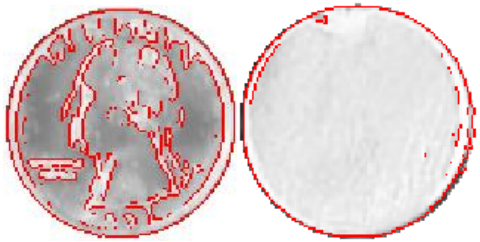
Using this fact, you previously developed code to segment US coins in an image, count each coin type, and determine the total value using the mask region properties.



Determining Valid vs. Blank Coins

Now, what if some of the coins are not valid currency, but rather blanks of the same size?

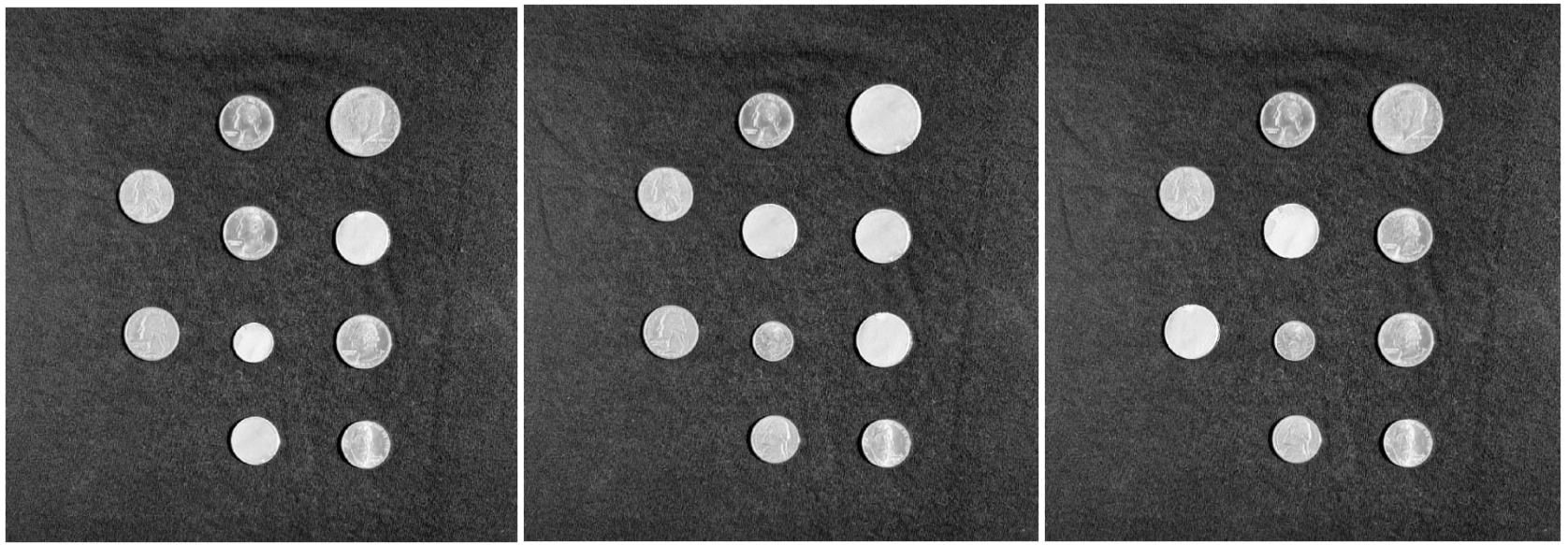
Using the mask region areas and perimeters as before will no longer be sufficient to count the number of each coin. You'll need to differentiate the blank coins from the valid coins. One feature that distinguishes the valid coins from the blanks is that valid coins have more pronounced edges near the center:



In this final project, you'll use the presence or absence of these edges to determine if a coin-sized region in an image is valid currency or a blank. Then you'll count the valid coin types and determine the total value.

Project Tasks

You'll work with three images, each with different coins replaced by blank versions:



There are four main stages in the workflow:

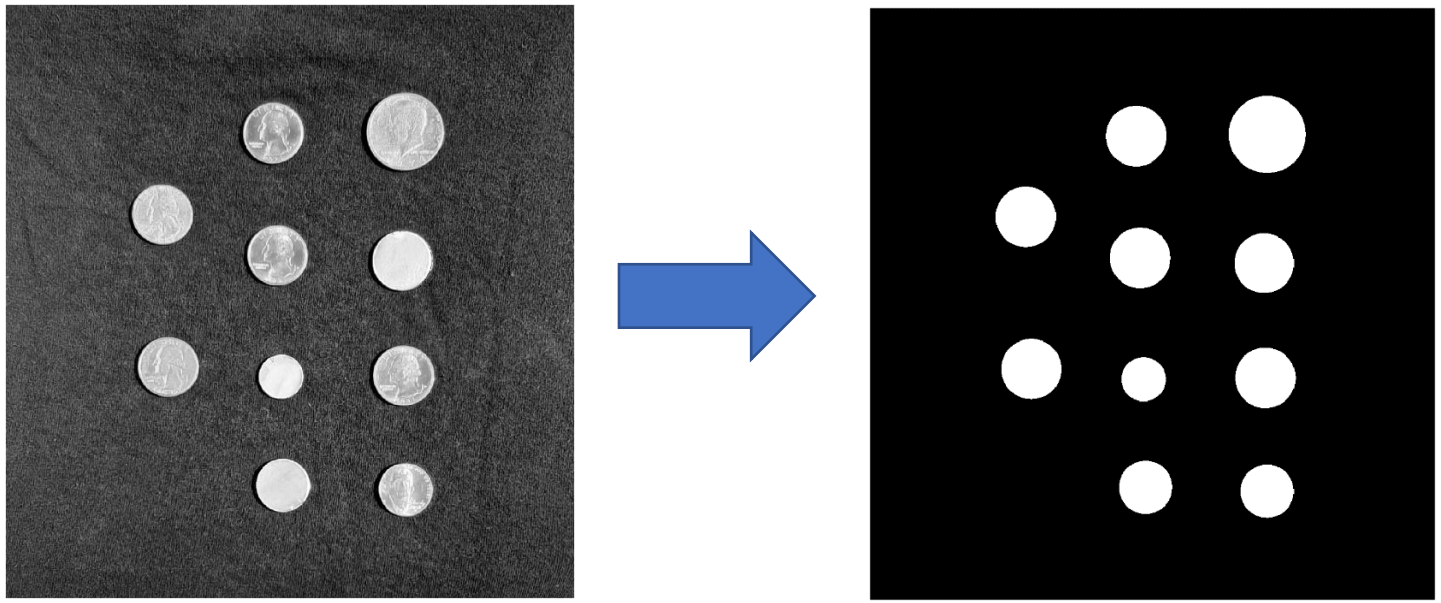
1. Segmenting the foreground (coins and blanks) from the background
2. Detecting and isolating valid coin faces
3. Segmenting only valid coins
4. Counting the number of each coin and calculating the total $ value

You will be assessed in MATLAB Grader at each stage using a random selection of the three test images. Each stage will most likely require you to copy the solution from the previous stage and add code for the new task

We encourage you to develop and test your code in MATLAB before submitting your solutions. The test images are provided with your course files: **testCoinImage1.png**, **testCoinImage2.png**, and **testCoinImage3.png**.

1. Segmenting Foreground

In step one, you need to create a mask that accurately segments the foreground, i.e., both the coins and blanks, from the background. Depending on how you previously segmented the image with all valid coins, the same code may work here, but you'll need to test it.



Using Code Generated from Apps

If you use an app to create the mask and masked image, you can bring your work into MATLAB Grader by generating a function and pasting the entire code at the bottom of the Script box. For example:

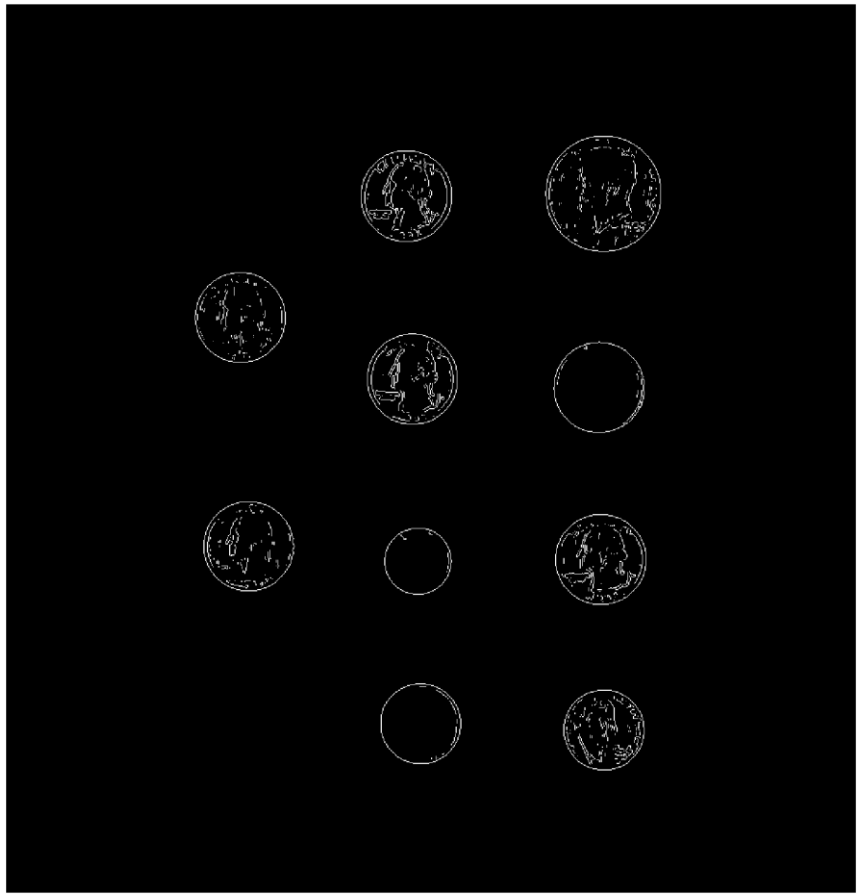
```
x = 1;
y = myFun(x);

function output = myFun(input)
% Auto-generated by an app on 21-Oct-2015
    output = 2*input;
end
```

2. Isolating Coin Face Edges

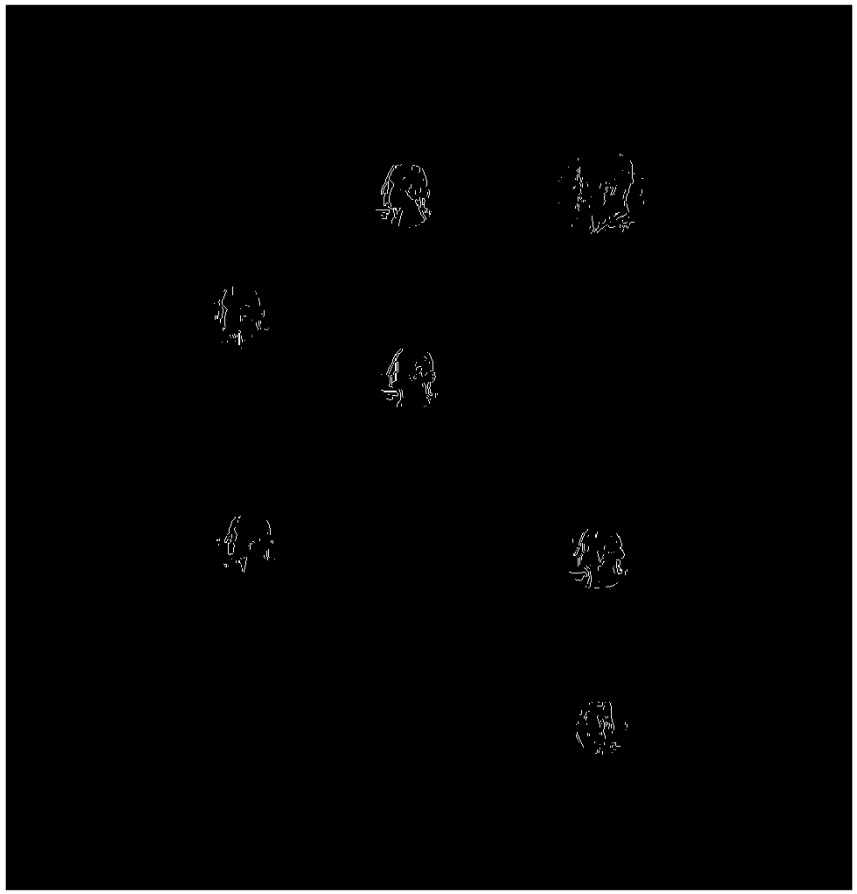
At this stage, you need to create mask with true pixels in the interior of each valid coin, but nowhere else. One workflow to accomplish this is given below:

- Create a binary image with strong edges on the surfaces of the valid coins and few to no edges on the surfaces of the blanks:



NOTE: The background of the original image is not smooth. Background edges could bias an automatically chosen edge threshold. Here, we used the masked foreground image from step one for edge calculations.

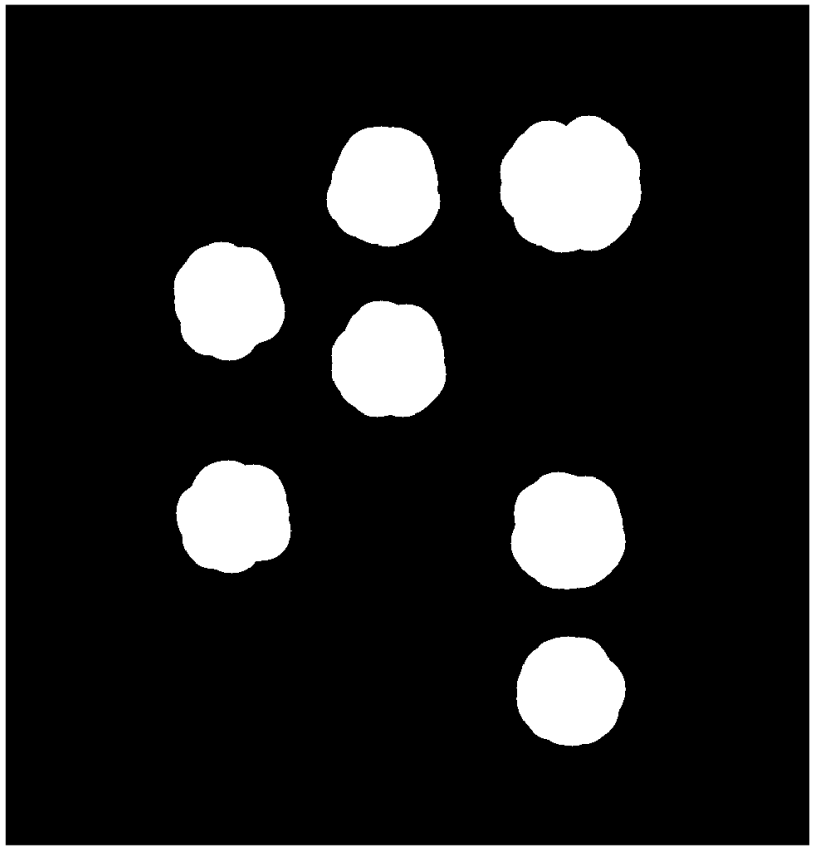
- Eliminate the pixels in the edge mask other than those in the valid coins. Logically combining your edge mask with an eroded version of your foreground segmentation mask should leave you with only the edges closer to the coin centers:



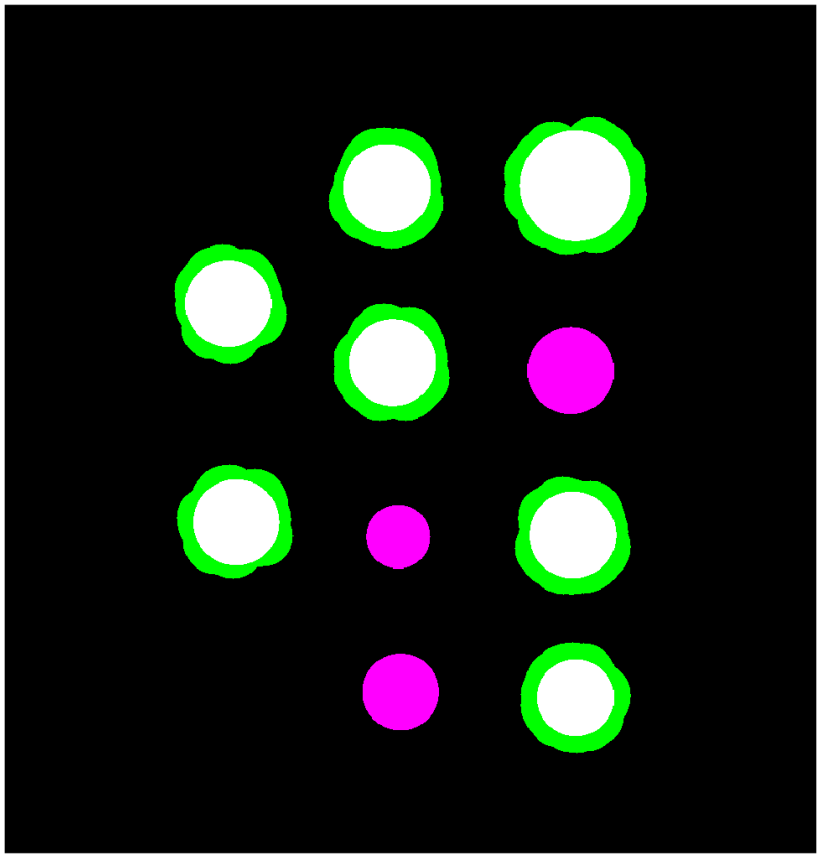
3. Creating a Valid Coin Mask

In this step, you need to create a mask that segments the valid coins. If you did step two correctly, then the valid coin regions are those that overlap the true pixels in your coin face edges mask. One approach to this task is given below:

- Expand the true pixel regions in the coin face edge mask to be greater than or equal to the size of the corresponding foreground regions:



- Combine the result with the foreground mask to extract the valid coin regions (white) and eliminate the blanks (magenta):



4. Counting Valid Coins

Finally, you need to:

- use the sizes of the valid coin regions to determine the type of coin each corresponds to
- count how many coins of each type are present in the image
- calculate the total value of coins in the image

You should be able to use your previous coin counting code for this final task.

Grades

You have completed all of the assignments that are currently due.

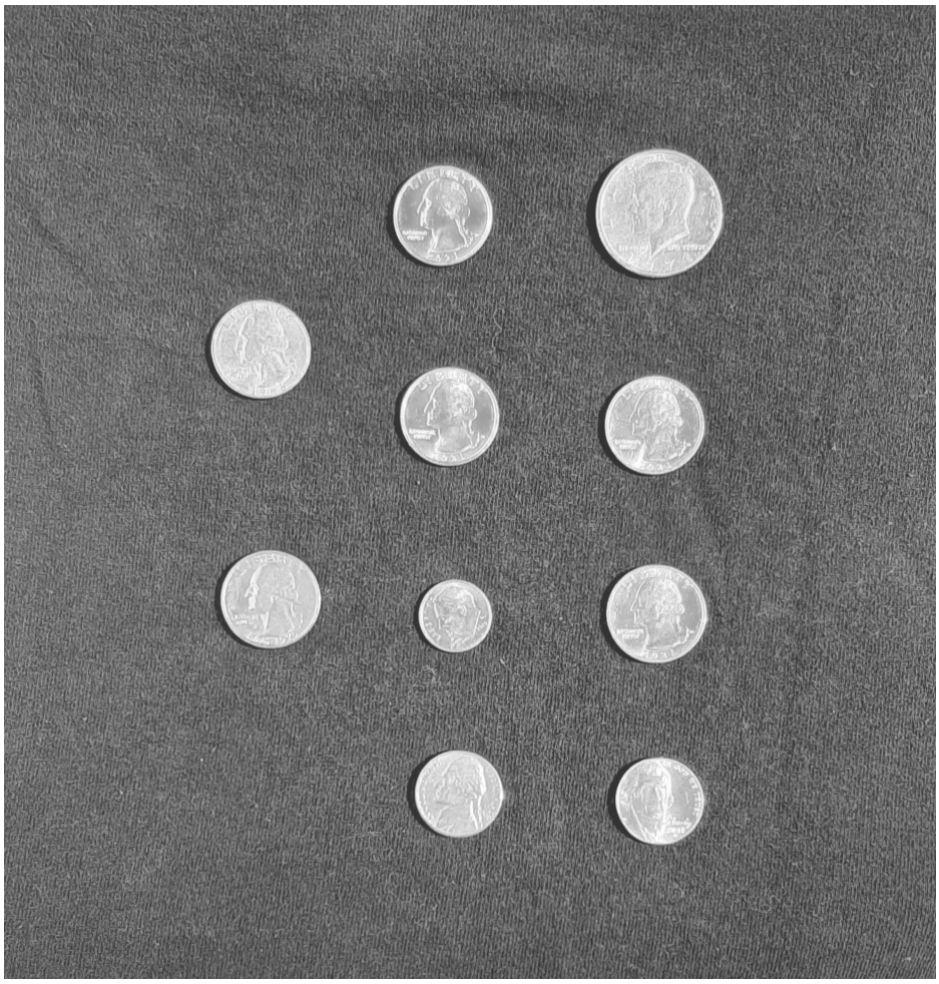
You passed this course! Your grade is 100%.

Item	Status	Due	Weight
Module 1 Quiz Quiz	Passed	Aug 14 11:59 PM IST	15%
Using the Image Segmenter App Quiz Quiz	Passed	Aug 21 11:59 PM IST	15%
Advanced Segmentation Approaches Quiz	Passed	Aug 28 11:59 PM IST	15%
Segment an Image with Code Graded External Tool	Passed	Aug 28 11:59 PM IST	15%
Segment the Foreground Graded External Tool	Passed	Sep 4 11:59 PM IST	10%
Valid Coin Face Edges Graded External Tool	Passed	Sep 4 11:59 PM IST	10%
Valid Coin Segmentation Graded External Tool	Passed	Sep 4 11:59 PM IST	10%
Final Analysis Graded External Tool	Passed	Sep 4 11:59 PM IST	10%

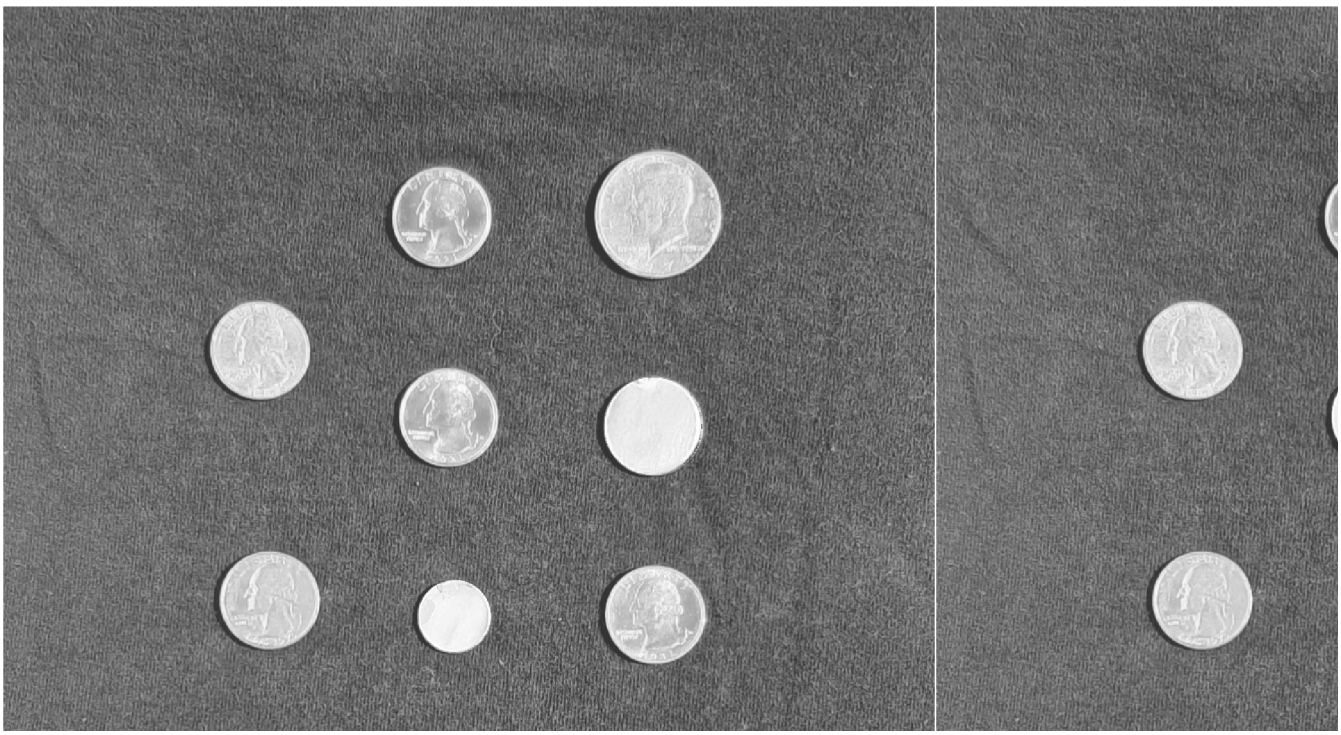
Segment the Foreground

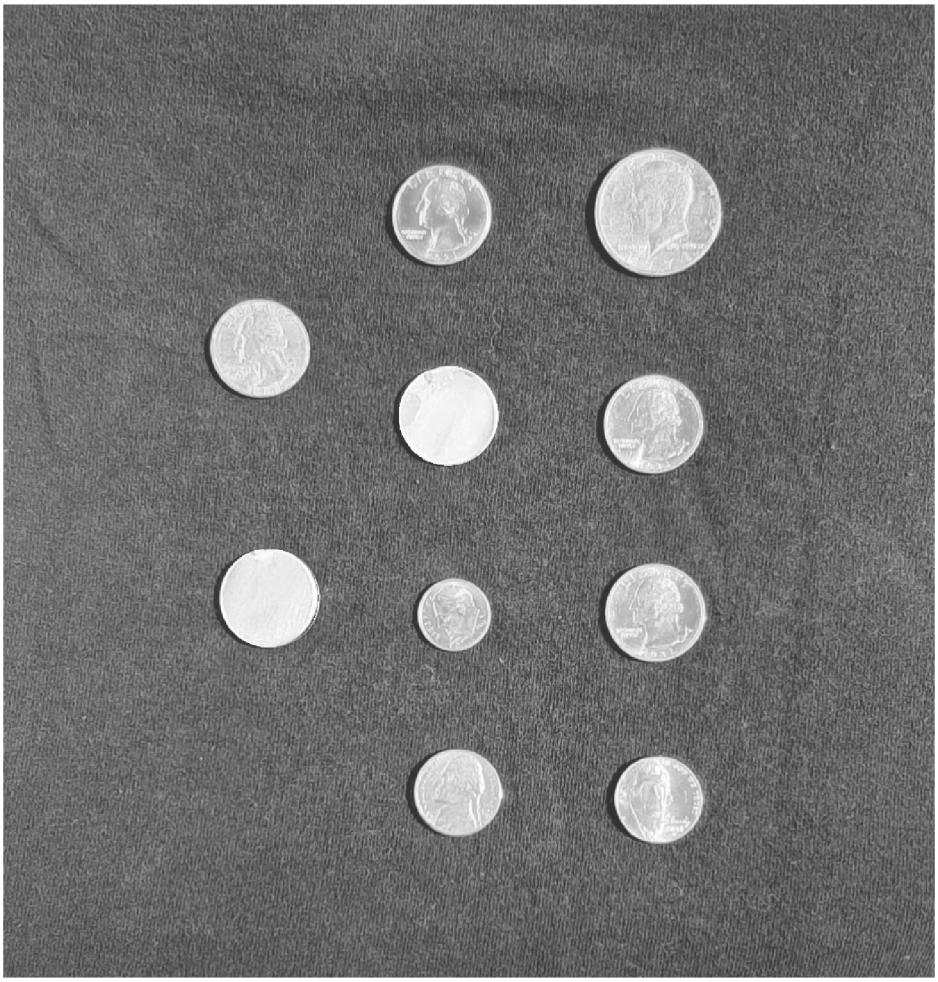
My Solutions ›

You've previously developed code to segment the US coins from this image and count the number of each ty



Now, what if some of the coins are not valid, but rather blanks of the same size? For example, the following t
testCoinImage1.png, **testCoinImage2.png**, and **testCoinImage3.png**. You'll need to differentiate the blank
segment both for further analysis.





For this problem you will be assessed using a randomly chosen selection from the three test images, so it is submitting here.

- Your code will need to create a mask to accurately segment the foreground (both the coins and blan
- Depending on how you segmented the image with all valid coins, the same code may work here, bu

NOTE: If you use an app to segment the images in MATLAB, generate your segmentation function and *copy* Grader to run it. For example:

```
testImageIdx = randi([1,3]);
testCoinImage = imread("testCoinImage"+testImageIdx+".png");
figure, imshow(testCoinImage)

% call segmentation function
[testCoinMask,maskedTestCoinImage] = yourSegmentationFunction(testCoinImageFiltere


% generated segmentation function
function [BW,maskedImage] = yourSegmentationFunction(X)
    % function code
```

```
12 end
13
```

Script

Save Reset MATLAB Documentation (https://www.mathworks.com/help/)

```
1  testImageIdx = randi([1,3]);
2  testCoinImage = imread("testCoinImage"+testImageIdx+".png");
3  figure, imshow(testCoinImage)
4  [testCoinMask, maskedTestCoinImage] = segmentCoins(testCoinImage);
5  figure, imshow(maskedTestCoinImage)
6
7  function [BW, maskedImage] = segmentCoins(X)
8      BW = imbinarize(X);
9      BW = imopen(BW, strel('disk', 3));
10     BW = imfill(BW, 'holes');
11     maskedImage = X;
12     maskedImage(~BW) = 0;
13 end
14
```

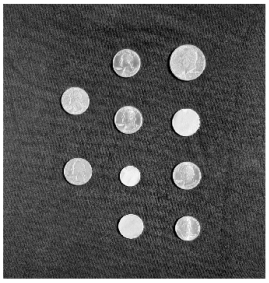
Run Script

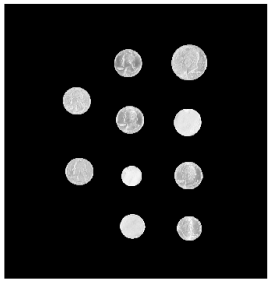
Assessment: All Tests Passed

Submit

✓ **Is the foreground mask accurate?**

Output

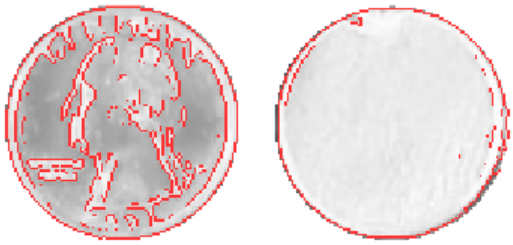




Valid Coin Face Edges

My Solutions ›

A major feature that distinguishes the valid coins from the blanks is that valid coins have more pronounced e
the **edge** function) overlayed on one of the quarters vs. one of the blanks.



For this problem your code will need to create a mask which only includes true pixels for all valid coin region
valid coin, but nowhere else. Use variable name **`faceEdgeMask`**.

One workflow to accomplish this is given below (see the project reading for more detials):

1. Use the **edge** function to create a binary image showing many edges on the coins, and few to no ed
   image is not smooth. Background edges could bias an automatically chosen edge threshold. You m
   edge calculations.
2. Eliminate the pixels in the edge mask other than those in the valid coins. Remember, you should ha
   while you have no true pixels other than near the outer rim of the blanks. Logically combining your e
   mask should leave you with only the edges closer to the coin centers.

As before, you will be assessed using a randomly chosen selection from the three test images, so it is a goo

Script

Save Reset MATLAB Documentation (https://www.mathworks.com/help/)

```
testImageIdx = randi([1,3])
testCoinImage = imread("testCoinImage"+testImageIdx+".png");
[mask, maskedImage] = segmentCoins(testCoinImage);
edges = edge(maskedImage);
erodedMask = imerode(mask, strel('disk', 17));
faceEdgeMask = edges & erodedMask;

function [BW, maskedImage] = segmentCoins(X)
    BW = imbinarize(X);
    BW = imopen(BW, strel('disk', 3));
    BW = imfill(BW, 'holes');
    maskedImage = X;
    maskedImage(~BW) = 0;
end
```

▶ Run Script

Assessment: All Tests Passed

Submit

Is the mask with edges only for valid coins correct?

Output

```
testImageIdx =

     2
```

Valid Coin Segmentation

My Solutions >

For this problem your code will need to create a mask which completely segments *only* the valid coins. Use t

The solutions to the previous assignments include a foreground mask with true pixel regions where any circu mask with true pixels only within the regions of valid coins (**`faceEdgeMask`**). Therefore, one approach to this than or equal to the size of the corresponding foreground regions in **`testCoinMask`**, and logically combine th Be careful not to dilate so far you overlap regions for blanks in the foreground mask. (See the project readin

As before, you will be assessed using a randomly chosen selection from the three test images, so it is a goo

Script

Save Reset MATLAB Documentation (https://www.mathworks.com/help/)

```
testImageIdx = randi([1,3])
testCoinImage = imread("testCoinImage"+testImageIdx+".png");
[testCoinMask, maskedImage] = segmentCoins(testCoinImage);
edges = edge(maskedImage);
erodedMask = imerode(testCoinMask, strel('disk', 17));
faceEdgeMask = edges & erodedMask;
faceEdgeMask = imdilate(faceEdgeMask, strel('disk', 85));
validCoinMask = testCoinMask & faceEdgeMask;

function [BW, maskedImage] = segmentCoins(X)
    BW = imbinarize(X);
    BW = imopen(BW, strel('disk', 3));
    BW = imfill(BW, 'holes');
    maskedImage = X;
    maskedImage(~BW) = 0;
end

```

Run Script

Assessment: All Tests Passed

Submit

Is the valid coin mask correct?

Output

```
testImageIdx =

     2
```

Final Analysis

My Solutions ›

For this problem your code will need to do the following:

- Accurately determine the number of each coin type present. Use variable names **nDimes**, **nNickels**
- Calculate the total $ value of coins present. Use variable name **USD**.

The coins can be differentiated and counted using your valid coin mask region properties. The USD values c

As before, you will be assessed using a randomly chosen selection from the three test images, so it is a goo

Script

Save Reset MATLAB Documentation (https://www.mathworks.com/help/)

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
```

```
testImageIdx = randi([1,3])
testCoinImage = imread("testCoinImage"+testImageIdx+".png");
[mask, maskedImage] = segmentCoins(testCoinImage);
edges = edge(maskedImage);
erodedMask = imerode(mask, strel('disk', 17));
faceEdgeMask = edges & erodedMask;
faceEdgeMask = imdilate(faceEdgeMask, strel('disk', 85));
validCoinMask = mask & faceEdgeMask;
maskedCoinImage = testCoinImage;
```

Run Script

Assessment: All Tests Passed

Submit

Is the number of each coin and the total value correct?

Output

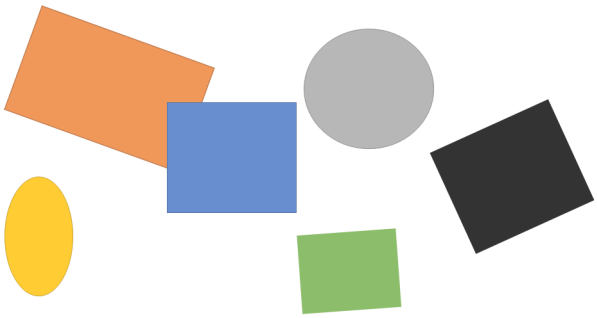
```
testImageIdx =

     2
```

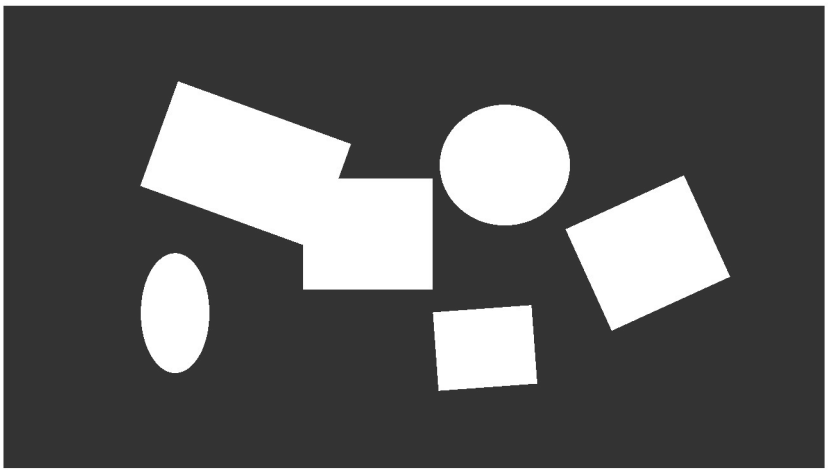
Edge Detection: Separating Overlapping Objects

My Solutions ›

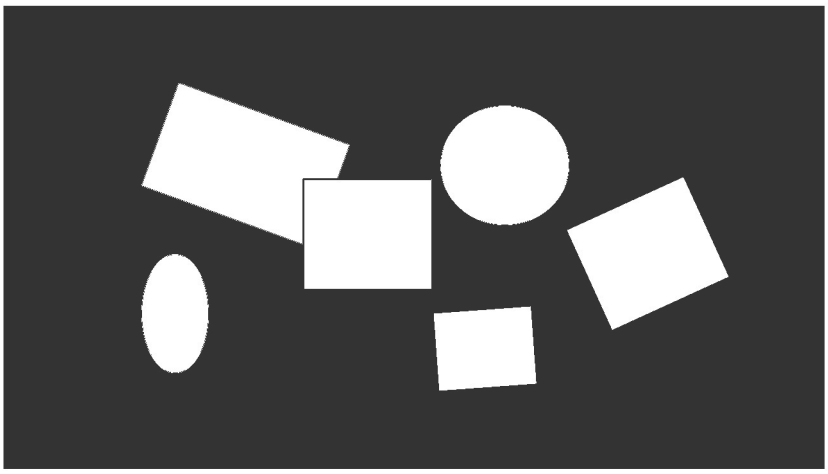
In the "Detecting Edges" video, you saw this example of overlapping shapes:



The orange and blue rectangles on the top left of the image are overlapping, leading initial segmentations to



Here, you will use edge detection to create a mask that separates the overlapping rectangles:



Create a script that:

- Converts the image to grayscale
- Segments the grayscale image and saves the result as **`BW`**. Your mask should have the shapes in th
- Detects the edges of the grayscale image using the sobel method and saves the result as **`imgEdges`** later segmentation more effective.
- Uses logical indexing to remove the edges from the original segmentation and saves the result as **`n`**

The original image, "DetectingEdgesSlide.png" is available in the course files. You are encouraged to work i appropriate for your segmentations. Your final segmentation should have 6 distinct regions instead of 5.

Uncomment the code available at the bottom of the script to display your results. Your edges will be enhance this later in this course.

Script

Save Reset MATLAB Documentation (https://www.mathworks.com/help/)

```
% Read in the image
img = imread("DetectingEdgesSlide.png");
img = im2gray(img);
% graythresh(img)
BW = ~imbinarize(img, 0.75);
%imshow(BW)
imgEdges = edge(img, "nothinning");
newBW = BW;
newBW(imgEdges) = 0;
% Uncomment below to Display
imshow(imerode(newBW,strel("disk",3)))
```

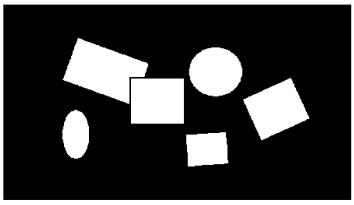
Run Script

Assessment: All Tests Passed

Submit

- **BW has the correct value**
- **imgEdges has the correct value**
- **newBW has the correct value**

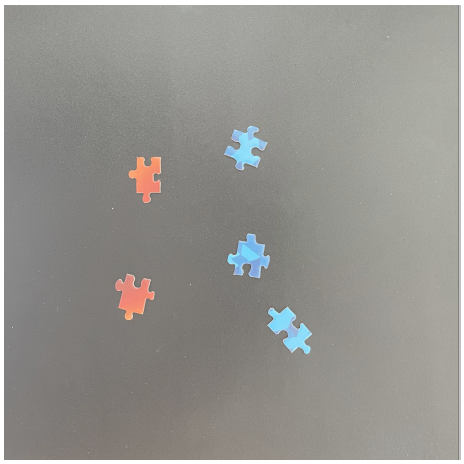
Output



Segment an Image into Multiple Labels

My Solutions >

In this problem, you will perform segmentation with code on the "Puzzle_06.jpg" image included in the cours differentiates the background, the red/orange puzzle pieces, and the blue puzzle pieces.



Your code should do the following:

1. Create a binary mask where the puzzle pieces are foreground and the rest of the image is backgrou
2. Segment the image by creating a labeled matrix with three groups, one for the background and one only contain the values 1, 2, and 3.

We recommend you work out a solution in MATLAB. If you get stuck, refer back to the "Segment Images witl

Script

Save Reset MATLAB Documentation (https://www.mathworks.com/help/)

```
img = imread('Puzzle_06.jpg');
img_gray = im2gray(img);
BW = imbinarize(img_gray, 'adaptive');
BW = imopen(BW, strel("disk", 3));
img(repmat(~BW,1,1,3)) = 0;
img_hsv = rgb2hsv(img);
labels = imsegkmeans(im2single(img_hsv), 3);
montage({img, label2rgb(labels)})
```

▶ Run Script

Assessment: All Tests Passed

Submit

✔ **Is the binary mask, BW, correct?**

Is the labeled image, labels, correct?

Output