# Project 2

## Due: 10/25/23

## Team Member Names:

_____

_____

_____

*Instructions:*

1. Fill in your name and attach this page as a cover sheet for your report.
2. This is a group project. Work with your group as assigned in the "ECE4510-F23 - Project Groups" document. List the contribution from each team member to this report, on the following page. Submit a single report for the group.
3. Attach all supporting material with the report.
4. Type / handwrite and sign the following honor pledge:
   "I pledge on my honor that I have not given or received any unauthorized assistance on this assignment."

*Honor Pledge:*

| |
|---|
| |

_____

*Signature*

## Grading Information:

| | |
|---|---|
| Task 1: MATLAB Image Analysis (40 Points)<br>a) Blob detection and boundary plotting (10 Points)<br>b) Find the different object types (20 Points)<br>c) Find the brightest washer in the image (10 Points) | |
| Task 2: Edge Detection in OpenCV (30 Points) | |
| Task 3: Keypoint Detection and Matching in OpenCV (30 Points)<br>Extra credit (25 Points) | |
| *Total* | **/ 100 Points** |

## Team member contributions to the report:

**Team member 1 name:**

*Contribution:*

**Team member 2 name:**

*Contribution:*

**Team member 3 name:**

*Contribution:*

**Team member 4 name:**

*Contribution:*

# Task 1: MATLAB Image Analysis

The purpose of this project is to get familiar with the image analysis and statistics functions in the image processing toolbox in MATLAB. Review the following functions before working on this project:

```
imfill          % Fill image regions and holes
bwlabel         % Label connected components in 2-D binary image
regionprops     % Measure properties of image regions
bwboundaries    % Trace region boundaries in binary image
```

This project will implement identification of different sized gears and washers in the “GearsAndWashers.tif” image using blob analysis in MATLAB. Print all intermediate images.

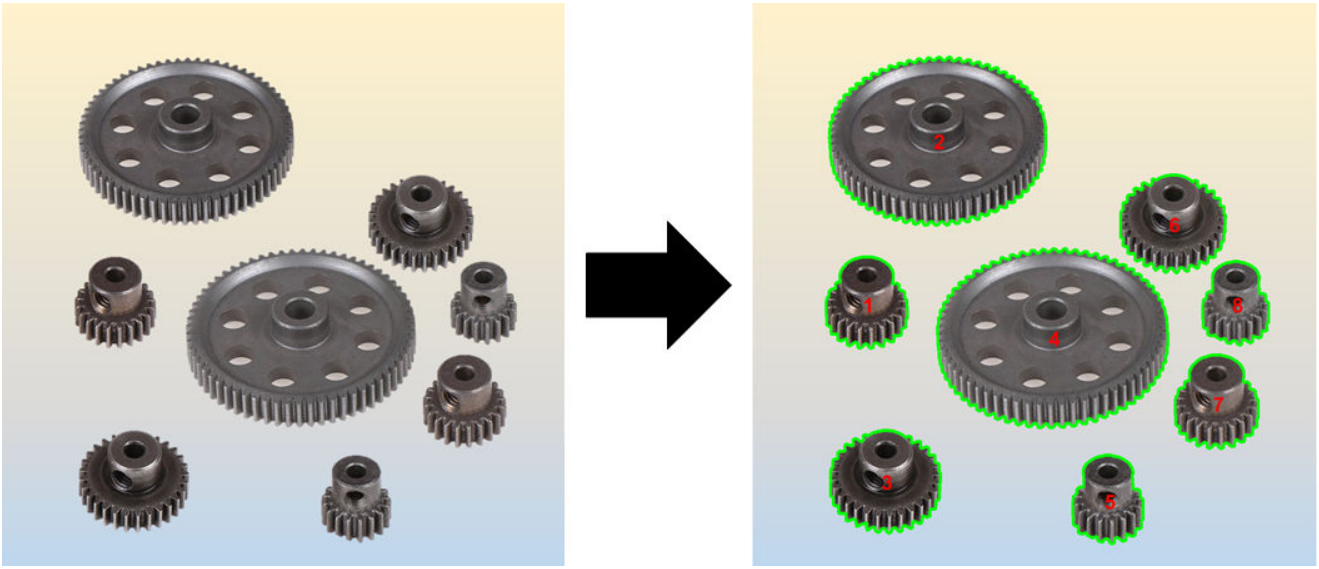
**Report:**

- Write your code in the MATLAB Live Scripts using the template provided.
- Comment your code sufficiently.
- Save the live scripts code with its output in a PDF file.

## a) Blob detection and boundary plotting

In the GearsAndWashers.tif image:

- Put a boundary on each of the object and a number label identifying them.
- Print the properties (mean intensity, area, perimeter, centroid, diameter) of each of the object.




i. Convert the image to a clean binary image with no holes by thresholding and **`imfill`** function.

ii. Use the **`bwlabel`** command to generate a labeled image showing each of the blobs in the image.% Label blobs in the image and show the resulting image.

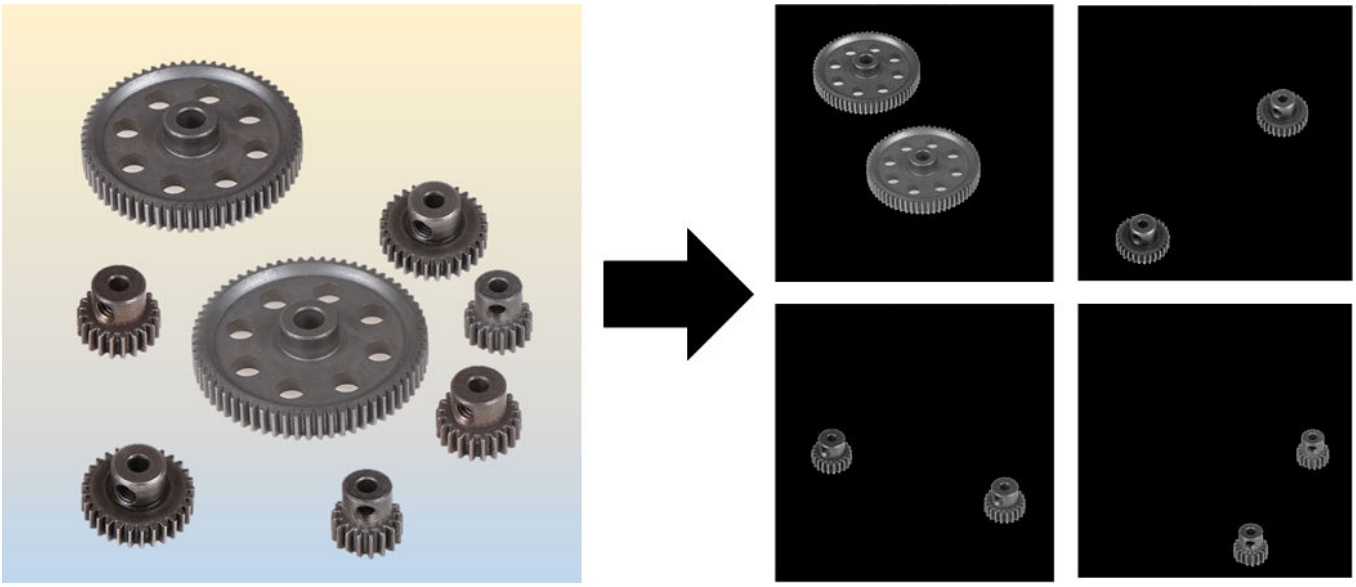
iii. Use the **`regionprops`** command to get the number of blobs found and the properties of each of the blobs.

iv. Use the **`bwboundaries`** command to plot the boundary of each of the blobs found on the original grayscale image to highlight the objects.

v. Print the blob properties (mean intensity, area, perimeter, centroid, diameter) of each blob found and plot blob number labels on the objects in the outlined grayscale image.

## b) Find the different object types in the image

Identify the different object types (large gears, medium-large gears, medium-small gears, and small gears) in the image using the blob properties obtained in part (a) and extract them in separate images.



i. Check blob properties from part (a) and identify which criteria to use to separate the four different types of the objects.

ii. Use the **`ismember`** function to extract the selected blobs in the labeled image that was generated in part (a) using the **`bwlabel`** command. Create a binary mask from the labeled image and filter each type of the object for display in the original grayscale image.

## c) Find the brightest medium-large gear in the image

Identify the brightest medium-large gear in the image using the blob intensity and other properties obtained in part (a) and extract it in a separate image.

i. Check blob properties from part (a) and identify which criteria to use to separate the brightest medium-large gear.

ii. Use the `ismember` function to extract the identified blobs in the labeled image that was generated in part (a) using the `bwlabel` command. Create a binary mask from the labeled image and filter the brightest medium-large gear for display in the original grayscale image.

# Project 2 - Task 2

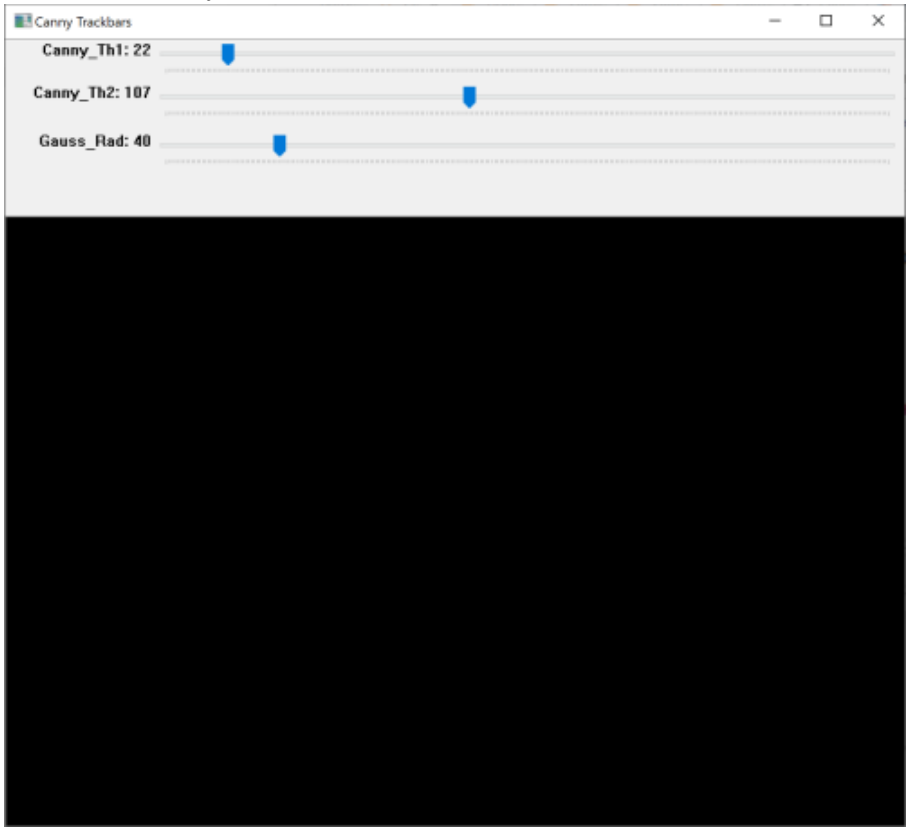
## Edge Detection in OpenCV

The purpose of this task is to get familiar with the Canny Edge detection function. Review the following basic functions in OpenCV before working on this task:

```
imread          Loads an image from a file
imshow          Displays an image in the specified window
imwrite         Saves an image to a specified file
createTrackbar  Creates a trackbar and attaches it to the specified window
getTrackbarPos  Returns the trackbar position
Canny           Finds edges in an image using the Canny algorithm
GaussianBlur    Blurs an image using a Gaussian filter
```

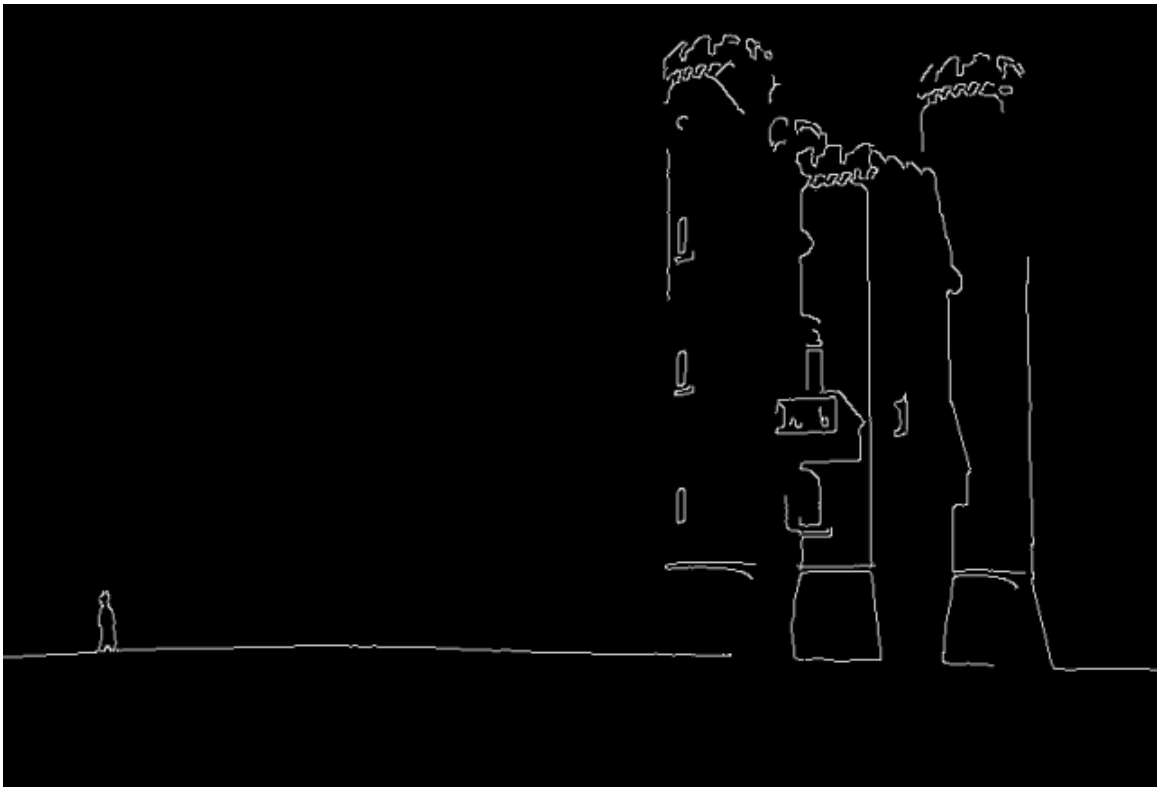
Write a program that does the following:

1. Creates 3 trackbars in a window to allow tuning of Canny function and Gaussian filter. Two of the trackbars adjust the first and the second threshold for the Canny function. The third trackbar sets the kernel radius of a Gaussian filter.

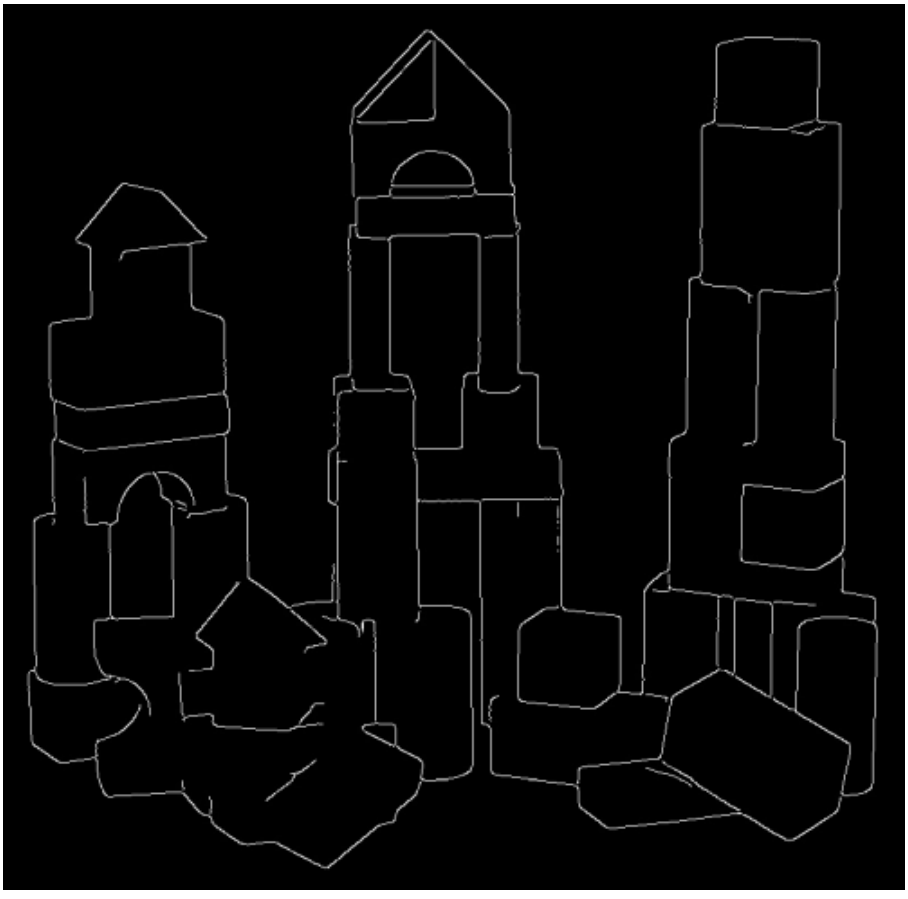



2. Open the given sample image file.
3. Apply Gaussian filter to the image with a kernel size computed using the radius selected by the trackbar position.
4. Obtain edges using the Canny function with the first and second thresholds selected by the two trackbar positions.
5. Show the edge image in the window and repeat last two step until desired edge image is obtained.
6. Save the final edge image in a file.

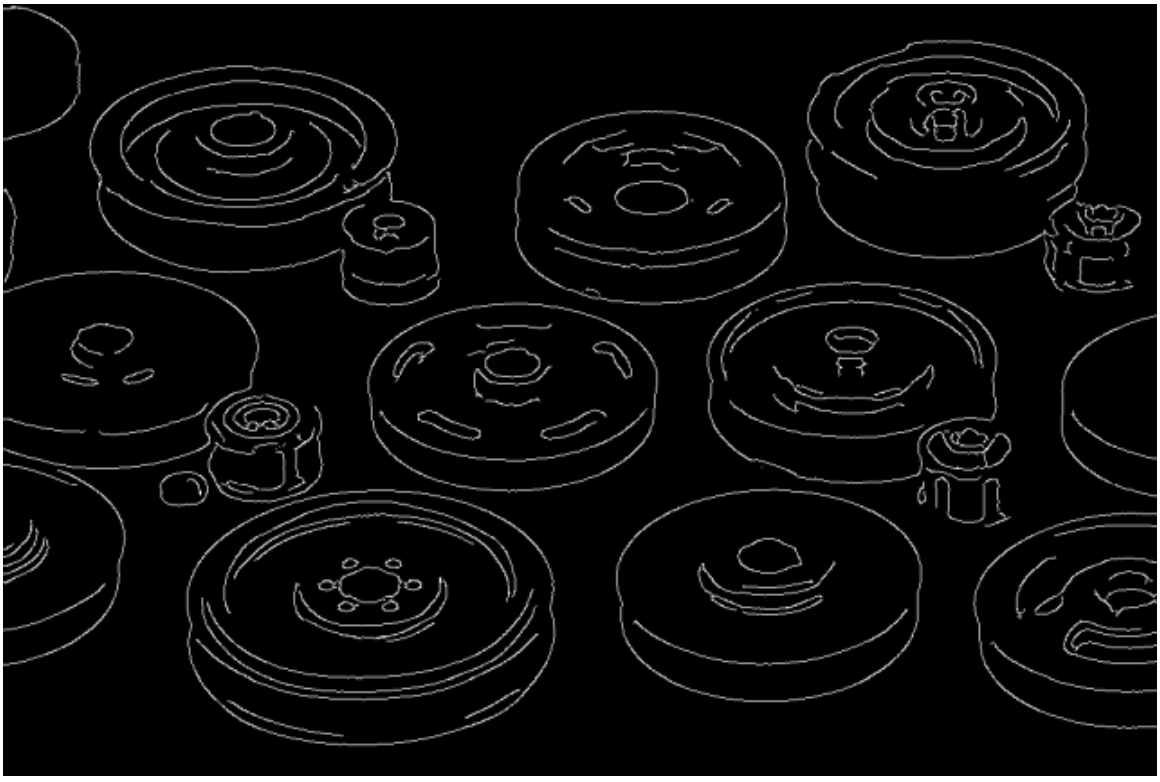
a) Process the “castle_small.tif” to obtain the edge image which looks as close as possible to that shown below. Note the Canny threshold and Gaussian filter size values.



b) Process the “Blocks.jpg” to obtain the edge image which looks as close as possible to that shown below. Note the Canny threshold and Gaussian filter size values.



c) Process the “Parts.jpg” to obtain the edge image which looks as close as possible to that shown below. Note the Canny threshold and Gaussian filter size values.



## Report:

- Write your code in the Jupyter Notebook.
- Comment your code sufficiently.
- Save the Jupyter Notebook code with its output in a PDF file.
- Submit the PDF file separately with the project report.
- Fill in the following table and attach it along with final edge images with your report:

| Image Name | Gaussian Radius | Canny Threshold 1 | Canny Threshold 2 |
|---|---|---|---|
| castle_small.tif | | | |
| Blocks.jpg | | | |
| Parts.jpg | | | |

# Project 2 - Task 3

## Keypoint Detection and Matching in OpenCV

The purpose of this task is to get familiar with the ORB (oriented BRIEF) keypoint detector and descriptor extractor and the Brute-force descriptor matcher. Review the following basic functions in OpenCV before working on this task:

| | |
|---|---|
| `ORB_create` | `The ORB constructor` |
| `BFMatcher_create` | `Brute-force matcher create method` |
| `detectAndCompute` | `Detects keypoints and computes the descriptors` |
| `match` | `Finds the best match for each descriptor from a query set` |
| `drawMatches` | `Draws the found matches of keypoints from two images` |

For extra credit:

| | |
|---|---|
| `findHomography` | `Finds a perspective transformation between two planes` |
| `perspectiveTransform` | `Performs the perspective matrix transformation of vectors` |
| `polylines` | `Draws several polygonal curves` |

Write a program that does the following:

1. Opens a source image.
2. Detects keypoints and descriptors on the source image (img1) using the ORB and BFMatcher.
3. Opens a video stream using the webcam.
4. Detects keypoints and descriptors on the destination image (img2) using the ORB and BFMatcher.
5. Finds the matches or correspondences between the two sets of descriptors.
6. Using the best 10 matches, draws lines between the matching points in img1 and img2.
7. Continue tracking the object in the video.
8. Save the last image in a file.

Example of a video frame tracking the object in the source image:

For extra credit:

1. Use the best 10 matches to form a transformation matrix.
2. Transform the rectangle around img1 based on the transformation matrix.
3. Add offset to put the bounding box at the correct position in img2.
4. Draw a bounded box around the matched object in the destination video stream to track it.


## Report:

- Write your code in the Jupyter Notebook.
- Comment your code sufficiently.
- Save the Jupyter Notebook code with its output in a PDF file.
- Attach 2-3 images of the detected object found in the video with the project report.
- For extra credit: Attach 2-3 images of the detected object found in the video with a box around it with the project report.