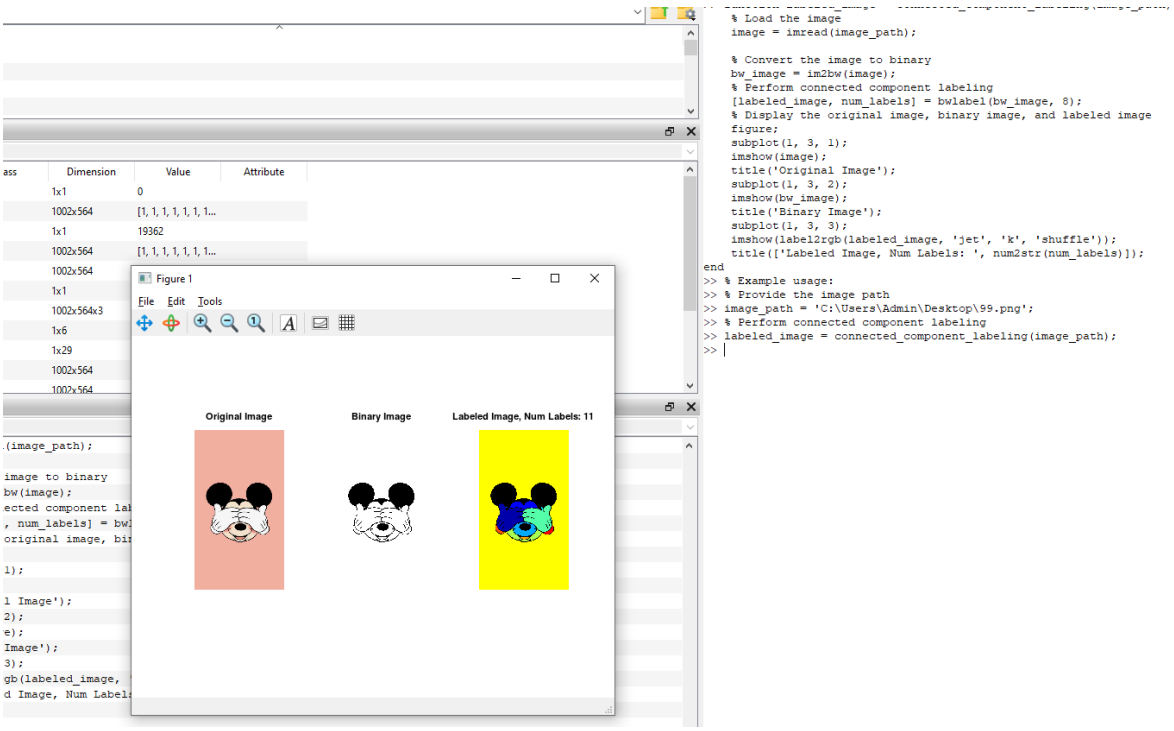
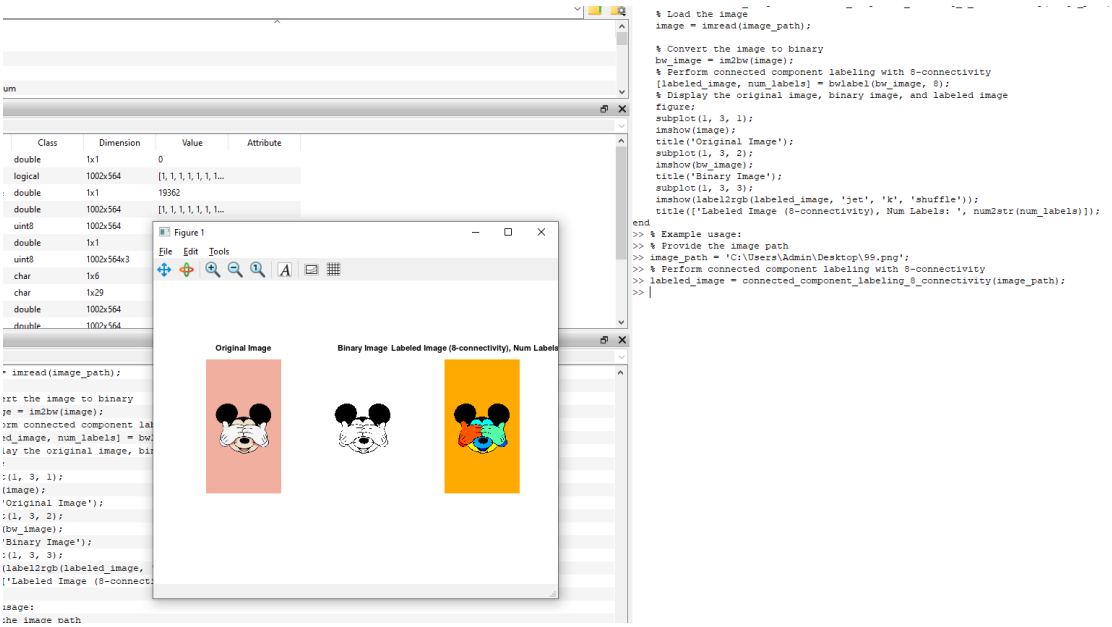
Output of Implement the component labeling algorithm



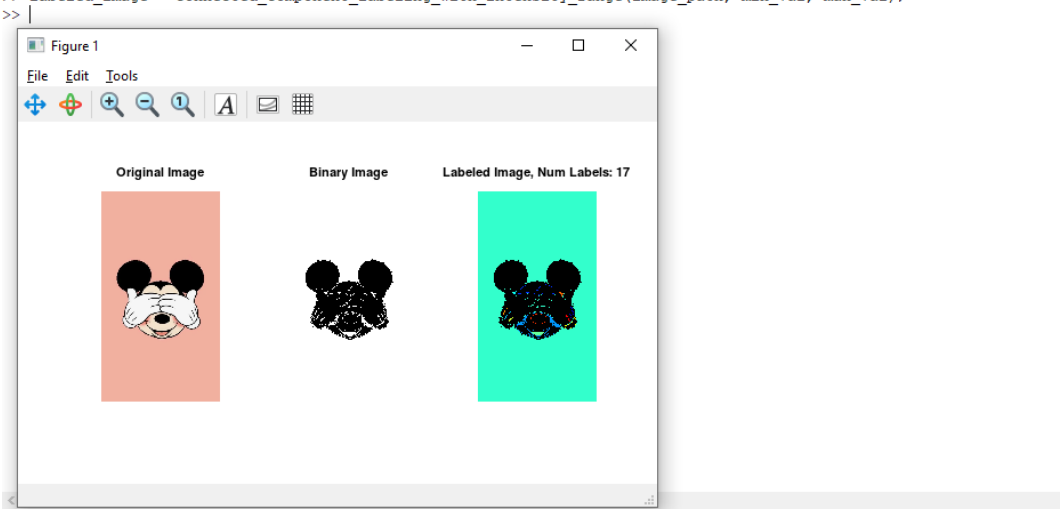
&Modify the component labeling algorithm from task 1 to consider the 8-connectivity.



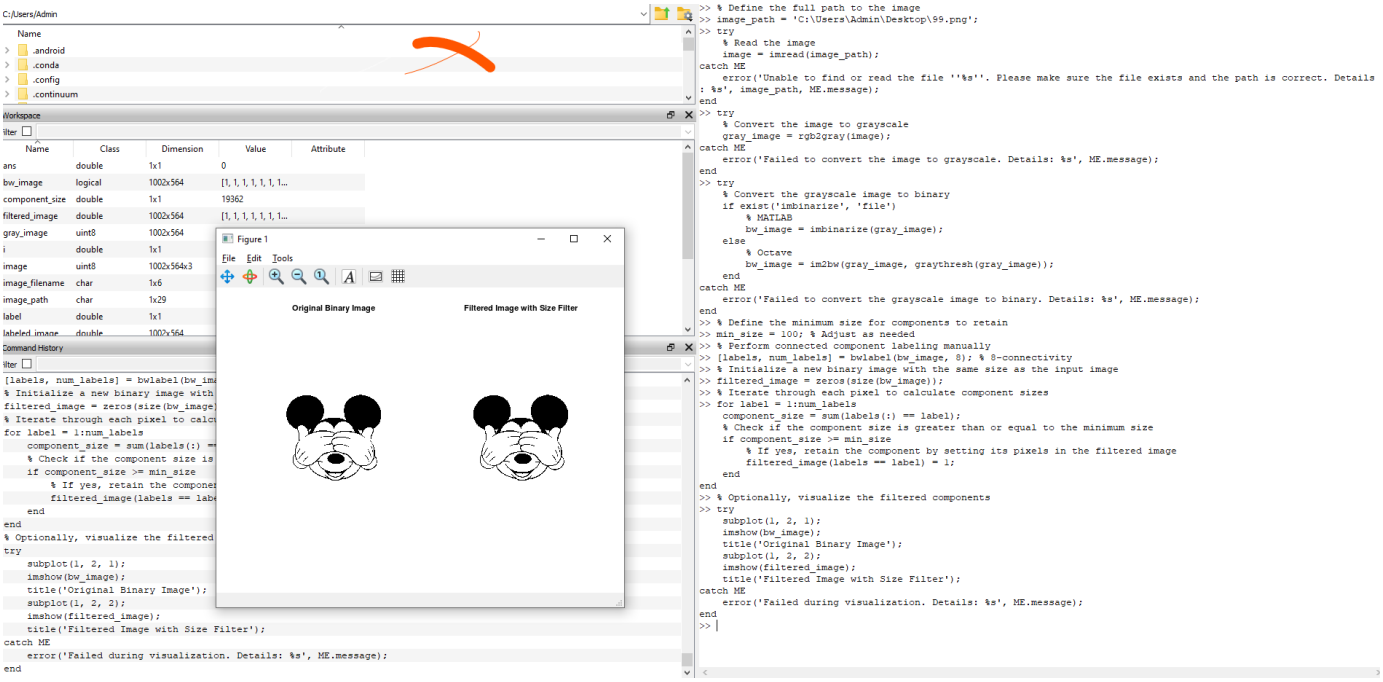
Output Modify the component labeling algorithm from task 1 to consider a range of intensity values within the set V, where V={min-max}, to determine pixel connectivity. The values of min and max are user-input parameters

```
>> function labeled_image = connected_component_labeling_with_intensity_range(image_path, min_val, max_val)
    % Load the image
    image = imread(image_path);

    % Convert the image to grayscale
    gray_image = rgb2gray(image);
    % Threshold the image based on intensity range
    bw_image = gray_image >= min_val & gray_image <= max_val;
    % Perform connected component labeling
    [labeled_image, num_labels] = bwlabel(bw_image, 8);
    % Display the original image, binary image, and labeled image
    figure;
    subplot(1, 3, 1);
    imshow(image);
    title('Original Image');
    subplot(1, 3, 2);
    imshow(bw_image);
    title('Binary Image');
    subplot(1, 3, 3);
    imshow(label2rgb(labeled_image, 'jet', 'k', 'shuffle'));
    title(['Labeled Image, Num Labels: ', num2str(num_labels)]);
end
>> % Example usage:
>> % Provide the image path and intensity range
>> image_path = 'C:\Users\Admin\Desktop\99.png';
>> min_val = 50;
>> max_val = 200;
>> % Perform connected component labeling with intensity range
>> labeled_image = connected_component_labeling_with_intensity_range(image_path, min_val, max_val);
>>
```



Implement the size filter algorithm




```
>> % Define the full path to the image
>> image_path = 'C:\Users\Admin\Desktop\99.png';
>> try
    % Read the image
    image = imread(image_path);
catch ME
    error('Unable to find or read the file ''%s''. Please make sure the file exists and the path is correct. Details
: %s', image_path, ME.message);
end
>> try
    % Convert the image to grayscale
    gray_image = rgb2gray(image);
catch ME
    error('Failed to convert the image to grayscale. Details: %s', ME.message);
end
>> try
    % Convert the grayscale image to binary
    if exist('imbinarize', 'file')
        % MATLAB
        bw_image = imbinarize(gray_image);
    else
        % Octave
        bw_image = im2bw(gray_image, graythresh(gray_image));
    end
catch ME
    error('Failed to convert the grayscale image to binary. Details: %s', ME.message);
end
>> % Define the minimum size for components to retain
>> min_size = 100; % Adjust as needed
>> % Perform connected component labeling manually
>> [labels, num_labels] = bwlabel(bw_image, 8); % 8-connectivity
>> % Initialize a new binary image with the same size as the input image
>> filtered_image = zeros(size(bw_image));
>> % Iterate through each pixel to calculate component sizes
>> for label = 1:num_labels
    component_size = sum(labels(:) == label);
    % Check if the component size is greater than or equal to the minimum size
    if component_size >= min_size
        % If yes, retain the component by setting its pixels in the filtered image
        filtered_image(labels == label) = 1;
    end
end
>> % Optionally, visualize the filtered components
>> try
    subplot(1, 2, 1);
    imshow(bw_image);
    title('Original Binary Image');
    subplot(1, 2, 2);
    imshow(filtered_image);
    title('Filtered Image with Size Filter');
catch ME
    error('Failed during visualization. Details: %s', ME.message);
end
>>
```