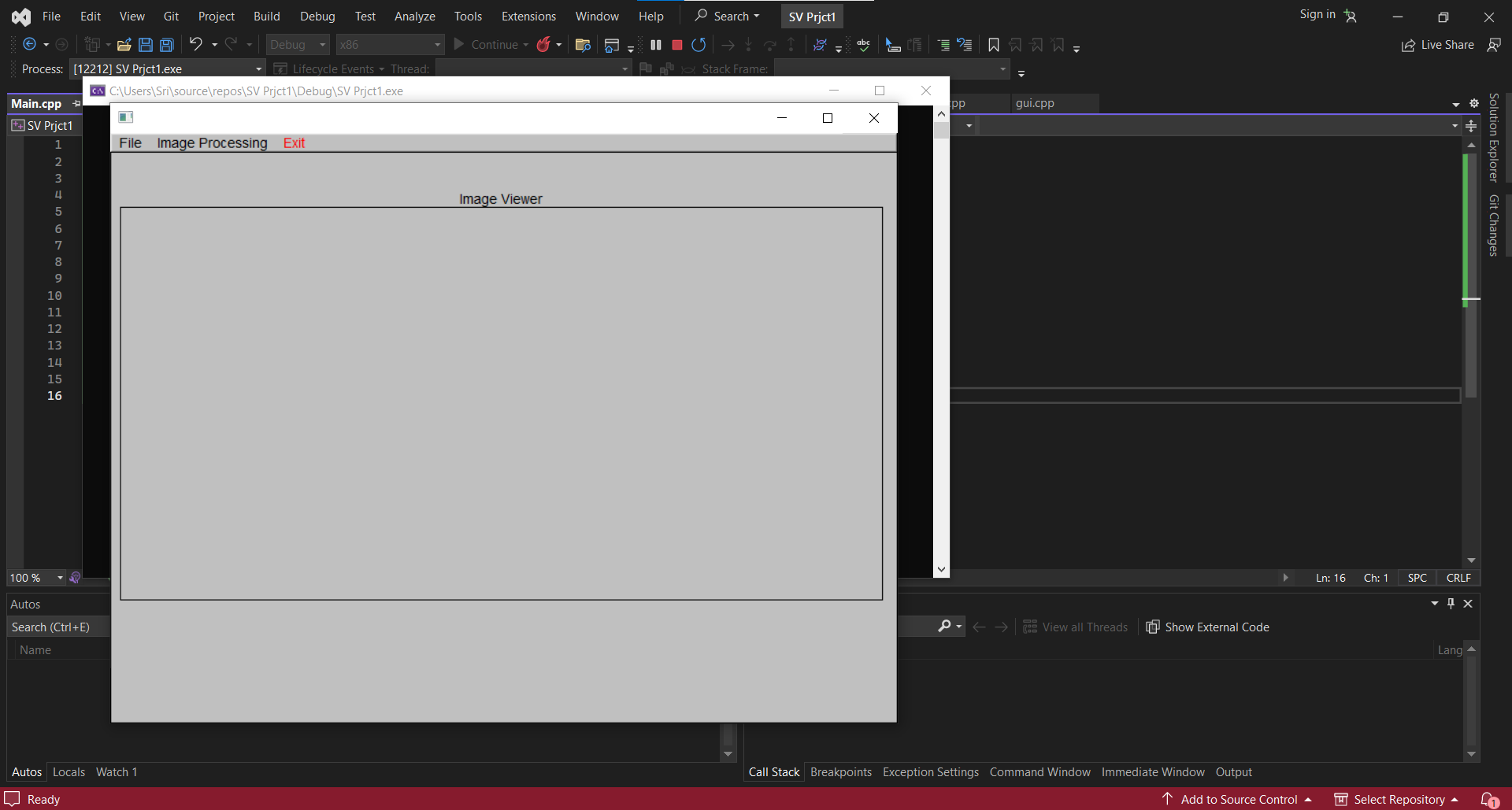
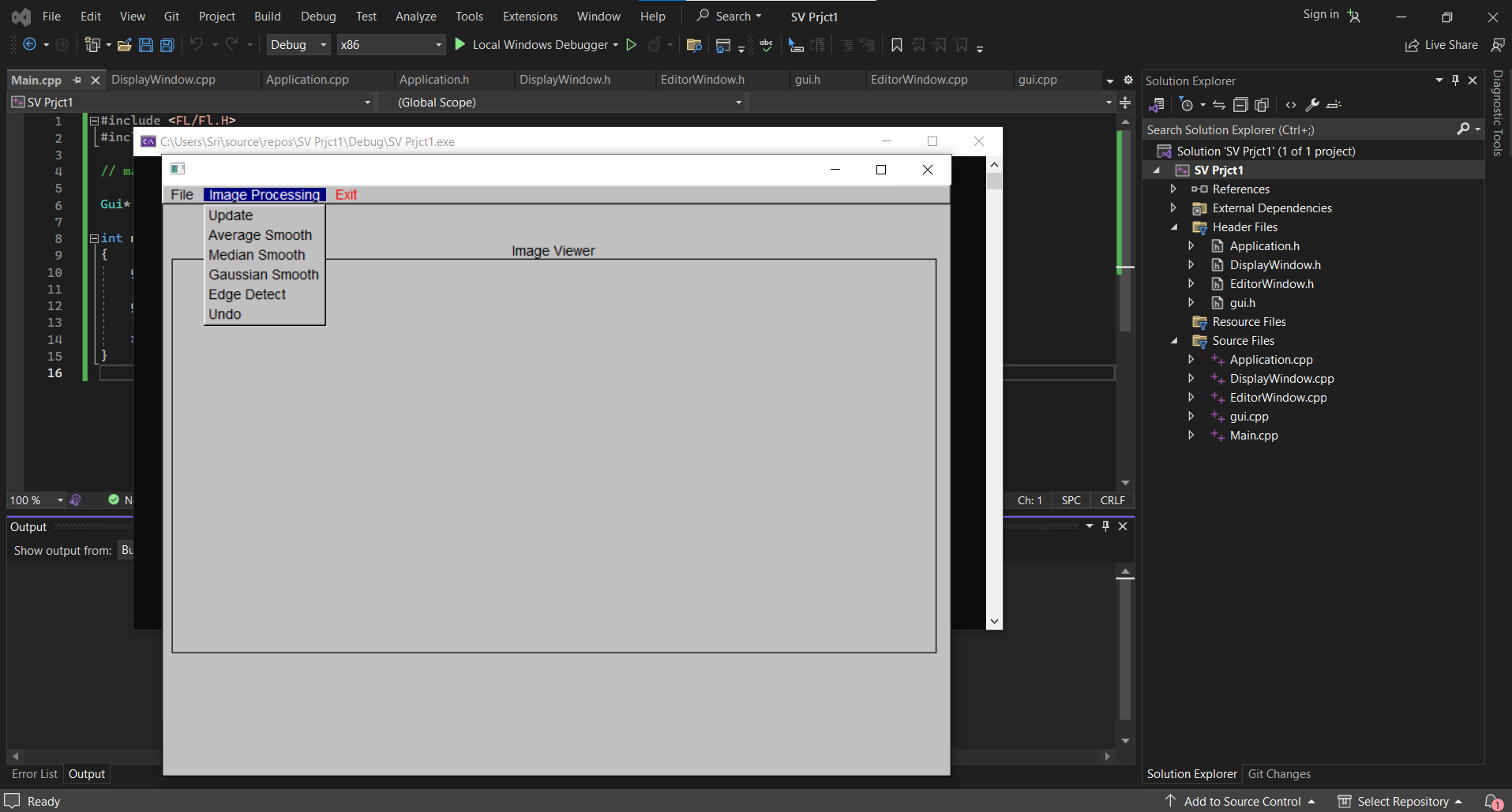
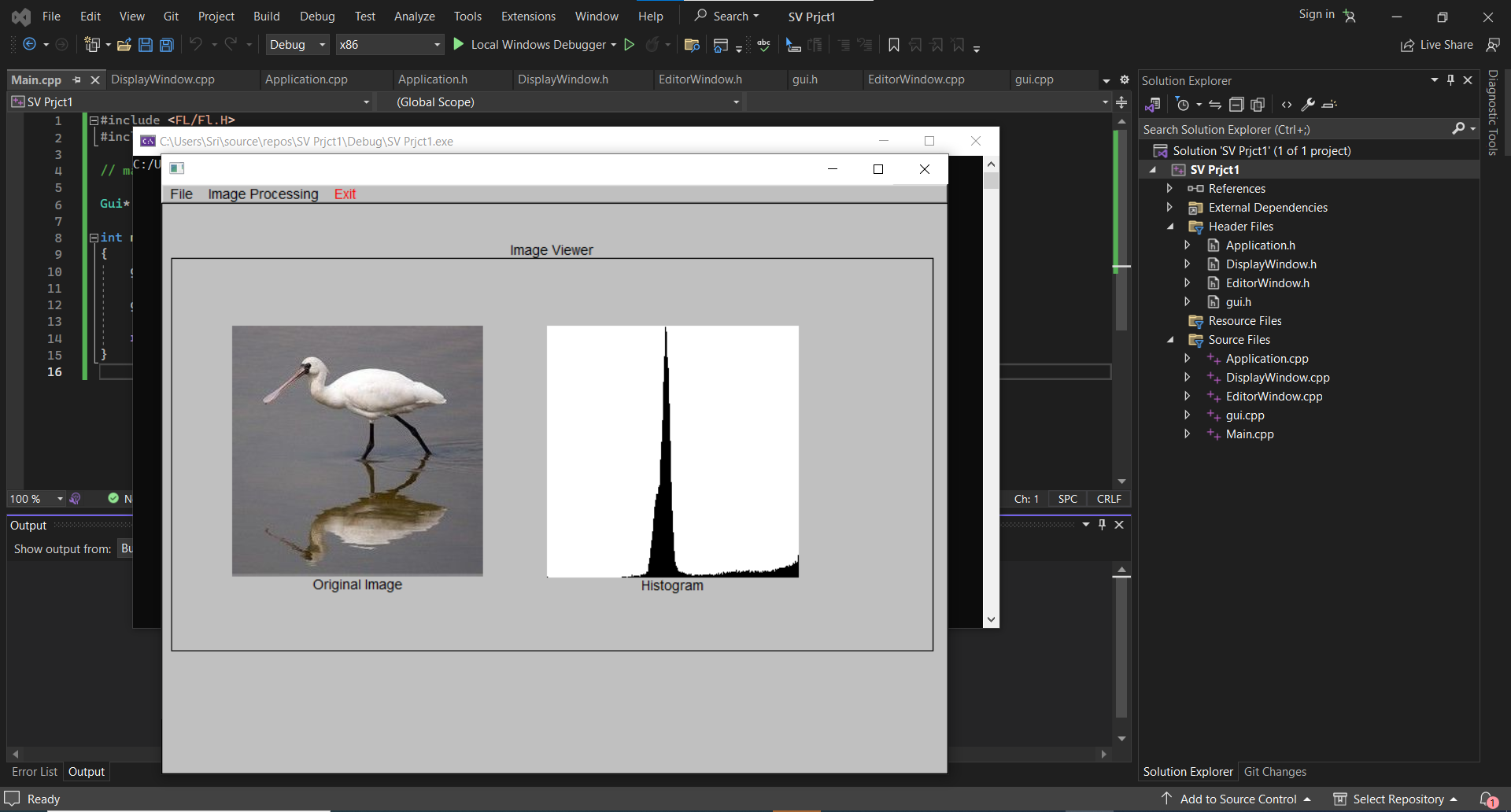
### Scientific Visualization Project 1



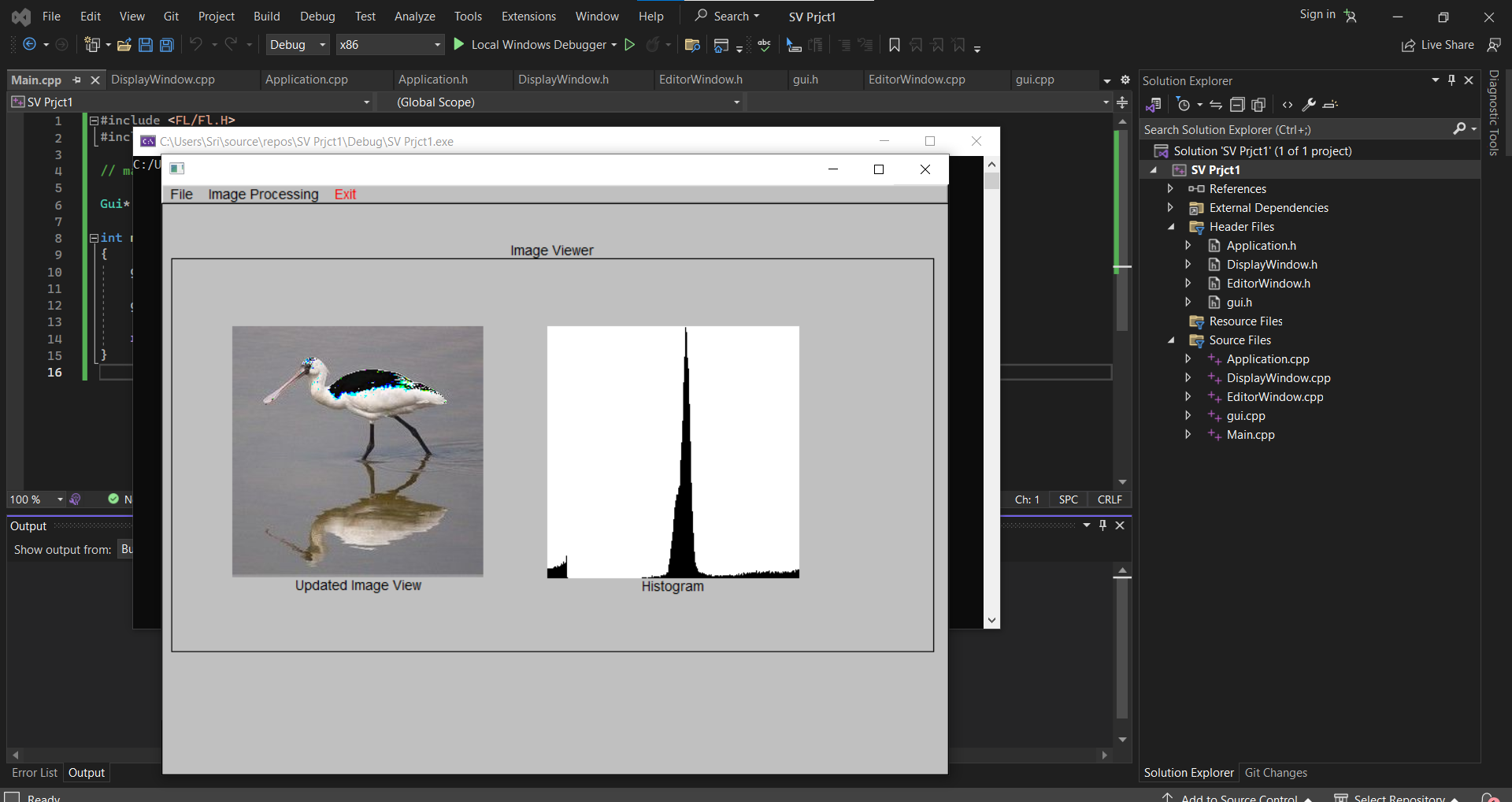
1. You are required to enhance the GUI of an application using the FLTK GUI editor 'fluid.' Specifically, you need to add a "Image Processing" menu on the main menu bar, featuring submenus like "Update," "Average Smooth," "Median Smooth," "Gaussian Smooth," "Edge Detect," and "Undo." These menu items should be added to the 'Gui.fl' file, and your changes will result in new versions of 'Gui.cpp' and 'Gui.h.' Additionally, you must create callback functions for these new menu items within the existing 'Application' class in 'Application.h' and 'Application.cpp.' The application is designed to work with .ppm or .pgm image formats, and it is suggested to start with gray scale images, although color images can be converted to gray scale by averaging color channel values.



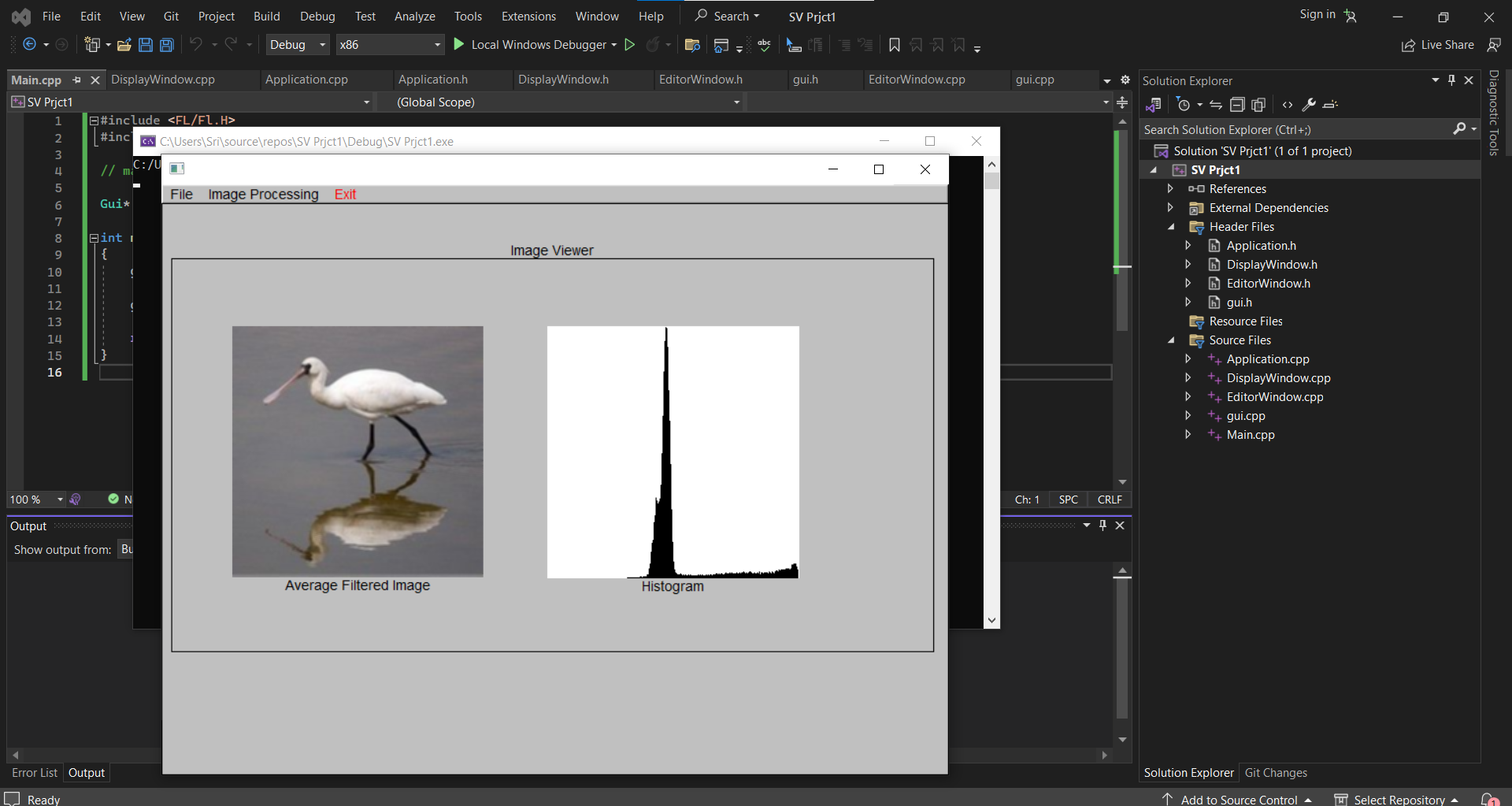
1. ***Histogram***: It displays the intensity of the current image as histogram. Different Image has different graphs. The pixel values of the image are converted to the histogram input.

******

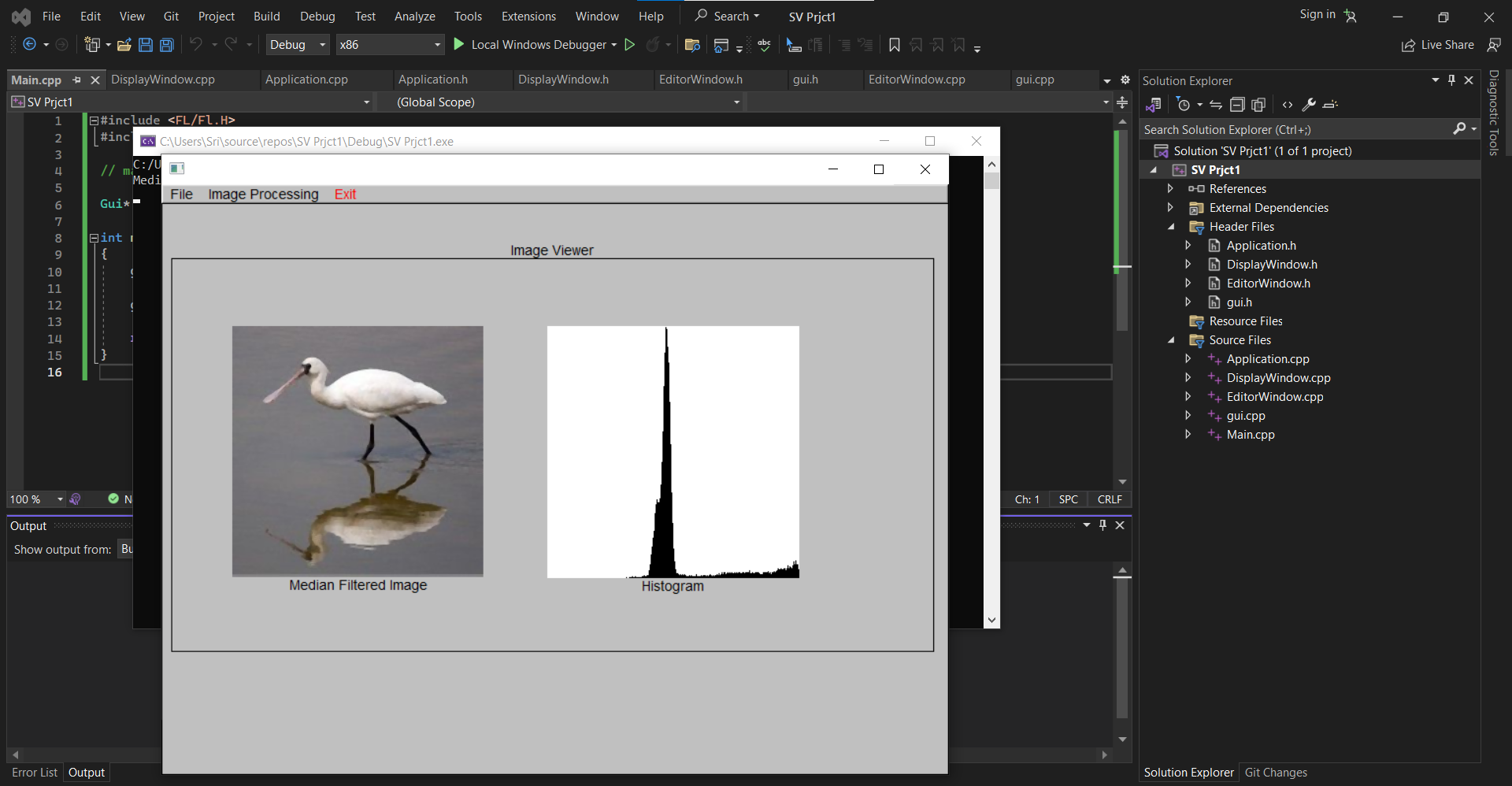
1. ***Update***: This update involves adding a feature to edit pixel intensities in the Editor window, using a transfer function defined in an array. When the "Update" menu is selected, the image is transformed.

******

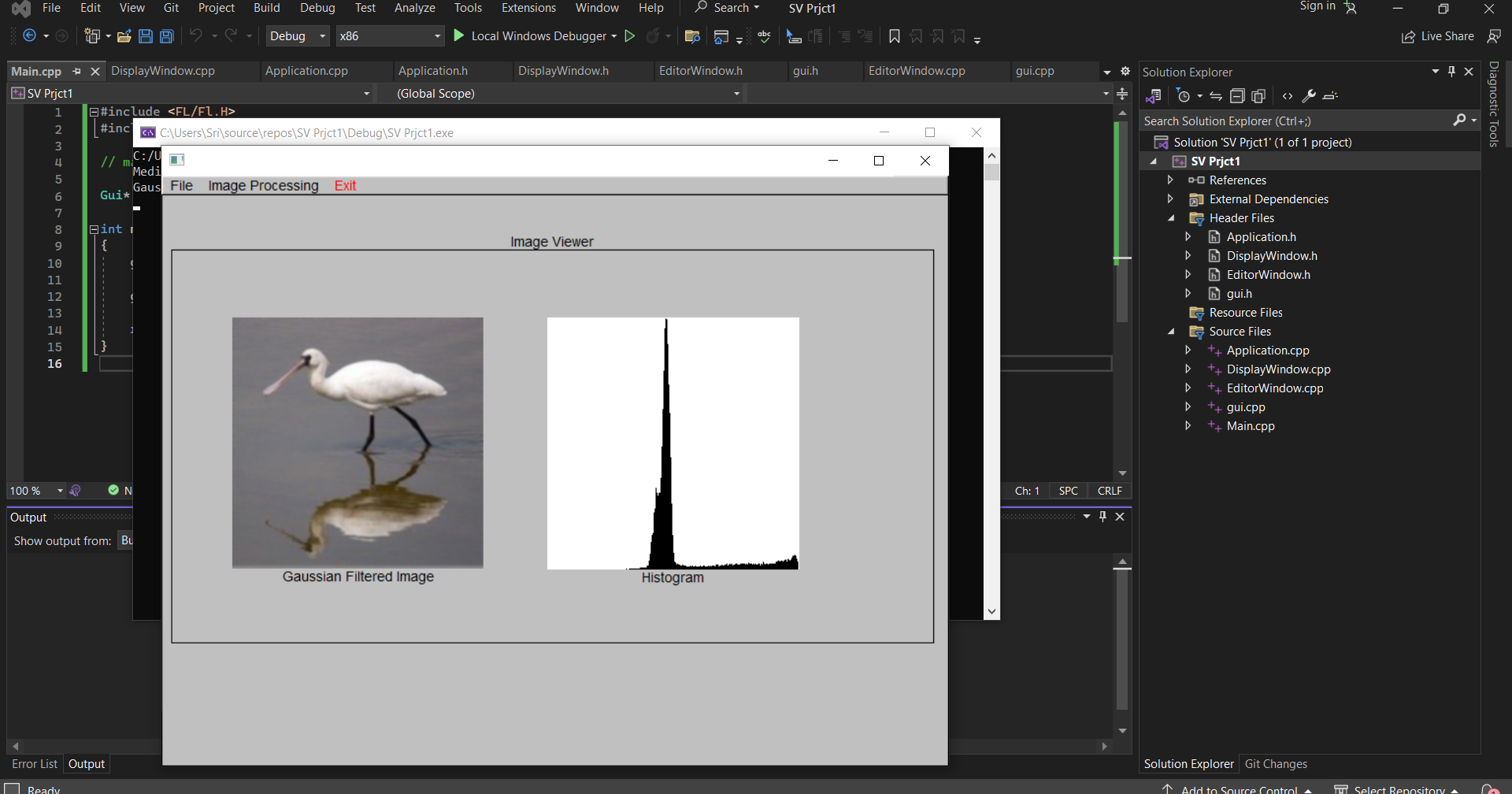
1. ***Average Smoothing***: using 3x3 averaging filter the image is transformed along with histogram, you will compute the average pixel value in a 3x3 neighborhood for each pixel in the original image and use this average value to create a new image. This process involves taking a weighted average of the pixel values in the neighborhood to reduce noise and create a smoother version of the image. The result is a new image with reduced high-frequency details, effectively applying a blur effect.



1. ***Median Smoothing***: using 3x3 median filter the image is transformed along with histogram.To apply 2D 3x3 median smoothing to an image, you will use a median filter to create a new image from the original image. This involves sliding a 3x3 window over the image and, for each pixel in the new image, replacing its value with the median value from the 3x3 neighborhood in the original image. This process effectively reduces noise and preserves edges and details in the image, making it a popular choice for image enhancement and noise reduction.

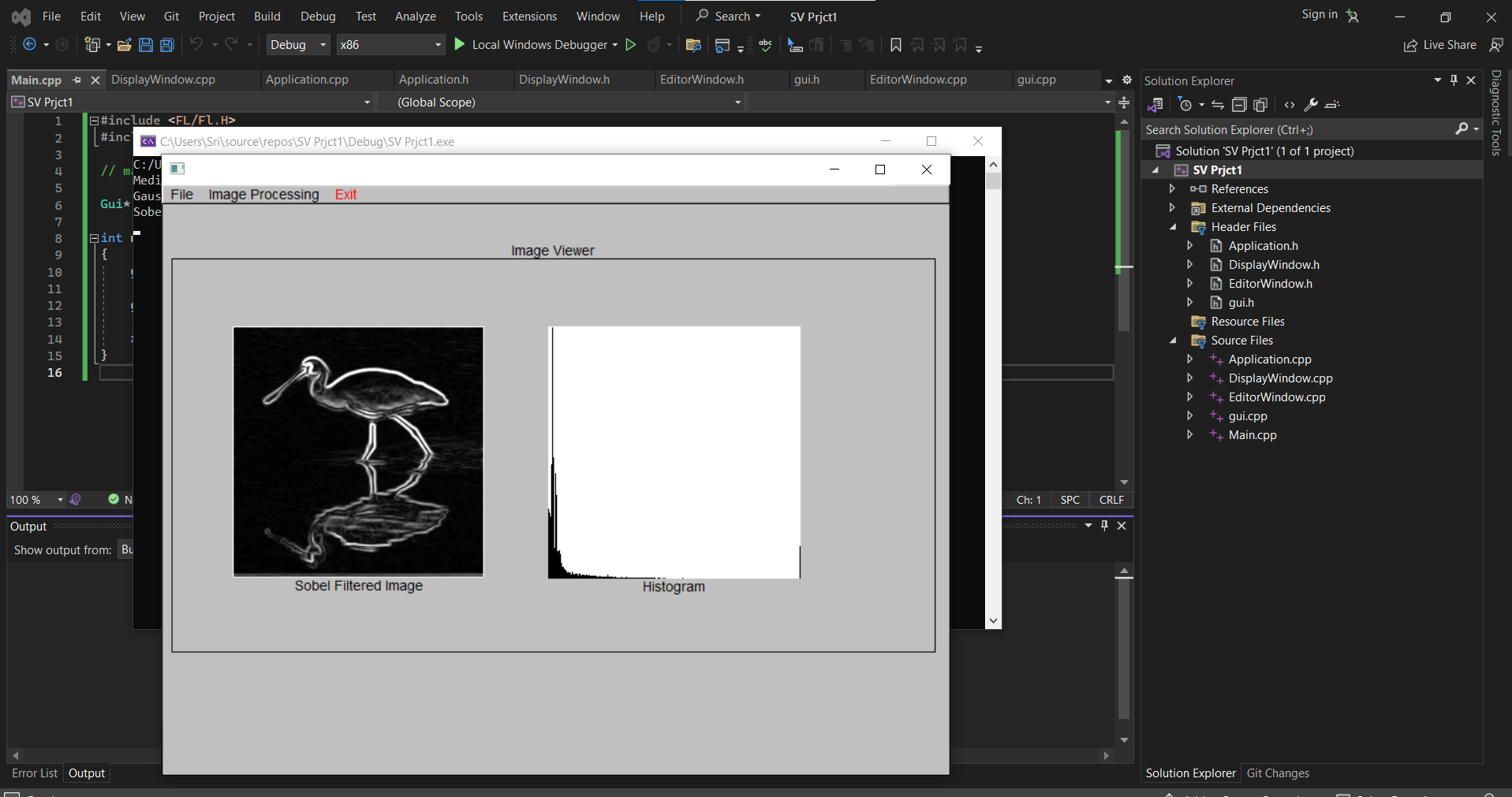


1. ***Gaussian Smoothing***: using 2D 5x5 Gaussian filter the image is transformed along with histogram. To apply 2D 5x5 Gaussian smoothing to an image, you will use a Gaussian filter to create a new image from the original image. This involves convolving the original image with a 5x5 Gaussian kernel, where each element of the kernel represents the weight assigned to a pixel's contribution in the neighborhood. This process results in a new image that is smoothed and reduced in high-frequency noise, with the degree of smoothing controlled by the parameters of the Gaussian kernel. Gaussian smoothing is commonly used for image blurring and noise reduction while preserving image details.



1. ***Edge Detection***: using 2D 3x3 Sobel kernel the image is transformed along with histogram.

To perform edge detection using a 2D 3x3 Sobel kernel, you will create a new image from the original image. This involves convolving the original image with two 3x3 Sobel kernels (one for detecting horizontal edges and the other for vertical edges). The resulting image, often called the gradient magnitude image, highlights edges in the original image by emphasizing changes in pixel intensities. This technique is used in computer vision and image processing to identify edges and contours within images.



1. ***Undo***: On clicking the undo submenu the changes made to the original image will be completely reflected back in both image and histogram outputs.

To implement the 'Undo' operation, your image processing application should maintain two entities of type Image, namely `curImage` and `oldImage`. Initially, both entities store the same image. After performing an image processing operation, the result is stored in `curImage`. When the user selects 'Undo,' the content of `curImage` and `oldImage` is swapped, effectively restoring the previous image. This enables users to revert to the prior image state if they are unsatisfied with the results of the latest operation, enhancing their ability to experiment with different image processing actions while maintaining a fallback option.

