Algorithm for Implementing Lee and Gradient-Inverse filter using MATLAB

To load an image

```
>> begin
  >>browse for path and select an image input_image(i,j);
  #In MATLAB image is stored as 2-D matrix.
  >>compute image_size;
  #Required for padding image
  >>show it in user-interface;
>>end
```

To implement Lee filter

```
>>begin
  >>Select the kernel size;
  >>Obtain weight of center pixel weight_function;
   # for a 3*3 kernel, value of center pixel will be defined by 8 neighborhood pixels
  >>Repeat
   #Kernel will traverse through whole image
     >> Mean= Mean of all elements of kernel at a position;
     >>output_image= Mean+(weight_function*(input_image- Mean));
     >>Show output_image;
  >>end
>>end
```

To implement Gradient-Inverse filter

```
>>begin
  >>Select the kernel size;
  >>Repeat
    >>If
          input_image(i+k,j+l) ~= input_image(i,j)
          u( i,j,k,l) = 1/(| input_image(i+k,j+l) -input_image(i,j)|;
          #Traversing through each neighbor and through whole image.
    >>Else
          u( i,j,k,l) = 2.0
   >>End
 >>End
 >>Obtain weight of center pixel weight_center;
 #Weight can be any arbitrary value. Generally, it is takes as 0.5. Here, it is obtained as user input.
 >>Repeat
   >>h(I,j,k,l)= weight_function *  (u(I,j,k,l)/ sum of all elements in window);
   >>output_image = convolution of h(I,j,k,l) and f(i,j);
 >>end
 >>Show output_image;
>>end
```

To compare both outputs

```
>>begin
  >>compute histograms of both the output images;
  >>show images and their histograms in a single screen;
 #Histogram shows the number of pixels in a particular gray-level.
>>end
```