The following are the single functions which are stored in several scripts.

1) ErrorMeasure.m

```
function p = ErrorMeasure(u,f,method)

u = double(u);
f = double(f);
N = numel(u);

switch method
  case 'MAE'
    % TODO: Mean absolute Error
    p = 1/N * norm(u(:)-f(:),1);


  case 'MSE'
    % TODO: Mean square Error
    p = 1/N * norm(u(:)-f(:))^2;


  case 'PSNR'
    % TODO: PSNR
    p = 10 * log10( ((max(f(:)) - min(f(:)))^2) * N / norm(u-f,'fro')^2 );

end

end
```

2) mean_filter

```
function U = mean_filter(F,s)

% TODO: mean filter
mask = ones(2*s+1)/(2*s+1)^2;
U = conv2(F,mask,'same');

end
```

3) gaussian_filter

```
function U = gaussian_filter(F,s)

if s==0
  U = F;
```

```
  return
end

% Determine sigma from the mask size
sigma=s/3;

% TODO: Compute the mask
% Gaussian values
[X,Y] = meshgrid(-s:s,-s:s);
R = sqrt(X.^2 + Y.^2);      % for every pixel distance to central pixel
mask = 1/sqrt(2*pi*sigma^2) * exp(-(R).^2/(2*sigma^2));
% Ideas:
%   - Leave 1/sqrt(2*pi*sigma^2) since we normalize later on
%   - Alternatively use
% b = (2*pi*sigma^2)^(-1/4)*exp(-(-s:s).^2/(2*sigma^2));
% mask = b'*b;
%   - or compute U by conv2(conv2(A,b'),b) which is much faster for big
%     filter sizes. Do not forget normalization.
%     This works since mask = conv(b',b) and convolution is
%
% truncate
mask(R>3*sigma)=0;
% normalize
mask = mask ./ sum(mask(:));

% TODO: use the mask on F
% convolution (moving average)
U = conv2(F,mask,'same');

end
```

4) median_filter

```
function U = median_filter(F,s)

% median filter
sz = size(F);
U = F;
for k=s+1:sz(1)-s
  for l=s+1:sz(2)-s
    % TODO: compute median
    U(k,l) = median(F(k+(-s:s),l+(-s:s)),'all');
  end
  h = waitbar((k-s)/(sz(1)-2*s));
end
close(h);

end
```