Enhancing the resolution of an image using Super Resolution techniques

By
ARINDAM PRASAD
ARNAB CHOUDHURY
SOHAN DUTTA
ARKOV PAUL

UNDER THE GUIDANCE OF
Prof. PRAMIT GHOSH

PROJECT REPORT SUBMITTED IN FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

BACHELOR OF TECHNOLOGY IN COMPUTERSCIENCE AND ENGINERING

RCC INSTITUTE OF INFORMATION TECHNOLOGY

Session 2014-2018

DEPARTMENT OFCOMPUTERSCIENCEANDENGINEERING

RCC INSTITUTE OF INFORMATION TECHNOLOGY
Affiliated to Maulana Abul Kalam Azad University of Technology
CANAL SOUTH ROAD, BELIAGHATA, KOLKATA-700015

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
RCC INSTITUTE OF INFORMATION TECHNOLOGY

श्रमम् बिना न किमपि साध्यम्

<u>TO WHOM IT MAY CONCERN</u>

I hereby recommend that the Project entitled 'Enhancing the resolution of an image using Super Resolution Techniques' **prepared under my supervision by** Arindam Prasad **(ClassRollNo.CSE2014/056),** Arnab Choudhury (roll: CSE2014/055), Sohan Dutta (roll: CSE2014/042), Arkov Paul (roll: CSE2014/046) **of B.Tech (8th Semester), may be accepted in fulfilment for the degree of** Bachelor of Technology in Computer Science & Engineering **under Maulana Abul Kalam Azad University of Technology.**

....
Pramit Ghosh, Associate Professor
Project Supervisor
Department of Computer Science and Engineering
RCC Institute of Information Technology

Countersigned:

....
Head
Department of Computer Sc.& Engg.
RCC Institute of information Technology
Kolkata-700015

DEPARTMEN TOF COMPUTER SCIENCE AND ENGINEERING
RCC INSTITUTE OF INFORMATION TECHNOLOGY

श्रमम् बिना न किमपि साध्यम्

CERTIFICATE OF APPROVAL

The foregoing Project is hereby accepted as a credible study of an engineering subject carried out and presented in a manner satisfactory to warrant its acceptance as a prerequisite to the degree for which it has been submitted. It is understood that by this approval the undersigned do not necessarily endorse or approve any statement made opinion expressed or conclusion drawn therein, but approve the project only for the purpose for which it is submitted.

FINAL EXAMINATION FOR EVALUATION OF PROJECT	**1.______________________________**
	2.______________________
	(Signature of Examiners)

DEPARTMENTOFCOMPUTER SCIENCEANDENGINEERING
RCCINSTITUTEOFINFORMATIONTECHNOLOGY

ACKNOWLEDGEMENT

We express our sincere gratitude to Mr.Pramit Ghosh, Associate Professor of Department of Computer Science & Engineering, RCCIIT for providing his valuable time for us to take up this topic as a Project.

ARINDAM PRASAD

ARNAB CHOUDHURY

SOHAN DUTTA

ARKOV PAUL

Table of Contents

1.Introduction:

For better pictorial view for human interpretation or for machine perception for making better decisions an image is required to be highly resolved. Resolution plays an important role for interpretation and analysis of an image. If the number of pixels is less than the image produces will be of low resolution (LR) and offers very less information. As the pixel density increases the image quality as well as the information offering by the image increases. Usually, the sensor limits the quality of an image due to its physical characteristics like size and density of detectors. The degradations in an image quality are caused at the recording process such as optical distortion, motion blur caused by limited shutter speed, noise and aliasing effects.
Optical Blur is a non-symmetric design of the lens and an aperture before or behind the optic center of the lens lead to image distortions. *Motions blur* results when the image being recorded changes during the recording of a single frame, either due to rapid movement or long exposure. *Noise* in an image is an undesirable by-product of image capture that adds spurious and extraneous information. *Aliasing effects* refer to an effect which can create confusion between different signals when sampled. Due to these, the final observed image is blurred and noisy. One way of producing a high resolution (HR) image, is by installing a high resolution sensor. But it is not very feasible to do so. It results in increase of a cost as well as increase in power consumption. A simple example of this is imaging system of satellite or a imaging system of medical, where it is infeasible to use a high resolution sensor. So, to come over these, post processing is required to develop a better resolved image that holds more information. One of the promising approaches for this is signal processing techniques to obtain HR image from multiple LR images. Nowadays such approach is more active in research area, and is called Super Resolution (SR) or Resolution Enhancement.

2.Review of Literature:

IMAGE SUPER RESOLUTION:

Super resolution is a process of achieving the best image quality through the single low-resolution image or multiple low-resolution (LR) images of the same scene. Super Resolution (SR) techniques combine the main feature of image
restoration and image interpolation. The dimension of the image can be changed via image interpolation whereas image restoration is used to recover a degraded image without changing its dimension. Thus image super resolution (SR) is a technique that restores the degraded image and also increases the size of the image. In SR from multiple LR images, it is a construction of HR image from several LR images, thereby increasing the high frequency components.
The basic idea behind this is to combine non-repetitive information contained by multiple LR images. While in SR from single LR image, resolution of the image can be increased either by enhancing the edges of the objects present in an image or by patch redundancy technique, where each LR patch is replaced by its corresponding HR patch. The main benefit of SR approach is that, a HR image can be obtained even with the existing LR imaging with lower cost and less power consumption.

In SR reconstruction from multiple LR images, the basic assumption is that the LR should have enough shifted in viewing the same scene. If LR has minor shift then the HR reconstructed image will not contain any new information. Suppose that four images are taken and one image out of four can be taken as reference and other be shifted horizontally, vertically or diagonally to a scale of half pixels. By taking that one image as reference, other three image pixels can be interleaved and a higher resolved image can be generated.

2	2	2
2	2	2
2	2	2

1	1	1
1	1	1

1	2	1	2	1	2
3	4	3	4	3	4
1	2	1	2	1	2
3	4	3	4	3	4
1	2	1	2	1	2
3	4	3	4	3	4

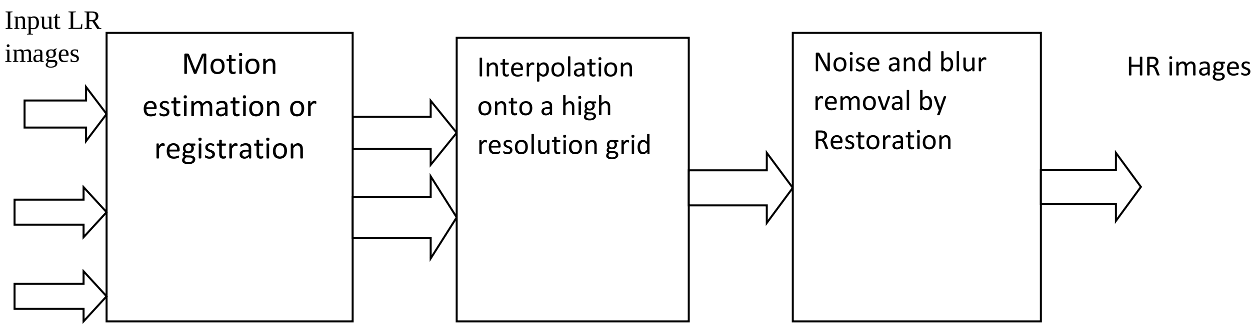
1	1	1

3	3	3
3	3	3
3	3	3

4	4	4
4	4	4
4	4	4

Ideal Super Resolution setup

Out of these left side four images first one can be taken as reference and other three images can be considered to be relative shifted to half a pixel in horizontal, vertical, and diagonal directions. These three image pixels can then be added to produce a high resolution image with increase in size of the right side-single image.
Usually, a super resolution method consists of the following basic processing steps: (1) Registration, (2) Interpolation and (3) DE blurring or noise removal.


Input LR images
Motion estimation or registration
Interpolation onto a high resolution grid
Noise and blur removal by Restoration
HR images

Basic SR reconstruction stages

Image registration is the process of overlapping more than one images of the same scene which has been taken from different angles by the sensors. In registration two or more images are align geometrically to obtain the information through image fusion or change detection.

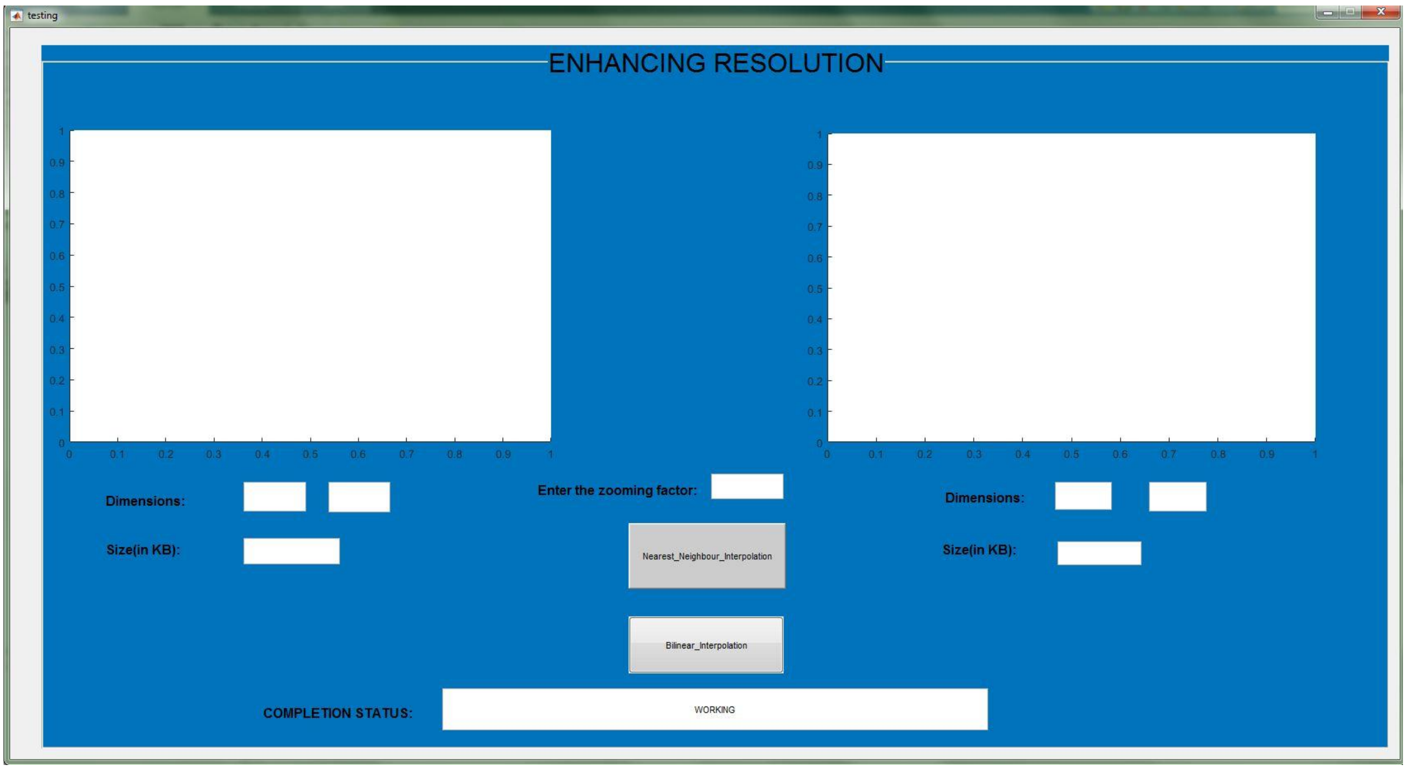
Interpolation is a process of estimating the intermediate pixels between the pixel values. When any image is converted from LR to HR, intermediate gaps are introduced and these values have to be estimated and filled with interpolation process.

As the process of interpolation introduces some artifacts the resultant image will be blurred or noisy. Through different filters and techniques noise will be removed and finally a super resolved image is generated.

3.Objective of the Project:

The objective of image super-resolution (SR) is to enhance the resolution of a given low-resolution (LR) image, which has always been a continuous ongoing process in image technology, through up-sampling, de-blurring, de-noising, etc. In order to restore an image into a high-resolution (HR) image correctly, it is necessary to infer high frequency components of a low-resolution image. In some applications like, video surveillance, forensic investigation, face recognition, medical diagnosis, satellite images and pattern recognition, it becomes essential to extract the useful information from the images. During this process, enlarging the image beyond a certain limit results in a blurred image with no peculiar information. Hardware limitations of sensors is one of the main cause behind this problem. Also, the main idea behind achieving the high-resolution images is not to tamper the observable quality of the image. Many super resolution techniques have been proposed to overcome the hardware limitations in order to achieve the best results. There are lots of techniques exist to increase the resolution of the input images. This paper provides comparative studies among them.

4.System Design:



The GUI was designed using Matlab R2017b. Here multiple low resolution image is first registered using Registration Estimator of matlab. Then the UI forms a high resolution image removing noise from that image, and displays the information of the image.

5.Methodology for implementation (Formulation/Algorithm)

Image Registration

Registration is the determination of a geometrical transformation that aligns points in one view of an object with corresponding points in another view of that object or another object. We use the term “view” generically to include a three dimensional image, a two-dimensional image, or the physical arrangement of an object in space. Three-dimensional images are acquired by tomographic modalities, such as computed tomography (CT), magnetic resonance (MR) imaging, single-photon emission computed tomography (SPECT), and positron emission tomography (PET). In each of these modalities, a contiguous set of two-dimensional slices provides a three-dimensional array of image intensity values. Typical two-dimensional images may be x-ray projections captured on film or as a digital radiograph or projections of visible light captured as a photograph or a video frame. In all cases, we are concerned primarily with digital images stored as discrete arrays of intensity values. In medical applications, which are our focus, the object in each view will be some anatomical region of the body. (See Volume I of this handbook for a discussion of medical imaging modalities.) The two views are typically acquired from the same patient, in which case the problem is that of intrapatient registration, but interpatient registration has application as well .

Overview:

Functions for aligning images by rotation and translation: image_registr_MI.m

MI2 - calculating Mutual information

joint_h - calculating Joint histogram

Mutual information is calculated using joint histogram calculation between two images.

For each angle of rotation all translation parameters are checked.

NOTE - the images must have correct relative sizes with respect to each other (no resizing is incorporated in this registration)

NOTE: image1 should be smaller than image2

Function allows to crop part of the image for registration to save computational time using IMCROP function.

For more help type:

help image_registr_MI.m

For users without IP toolbox download file im_reg_mi.zip without

% cropping option and with different rotation function

Nearest Neighbor Interpolation

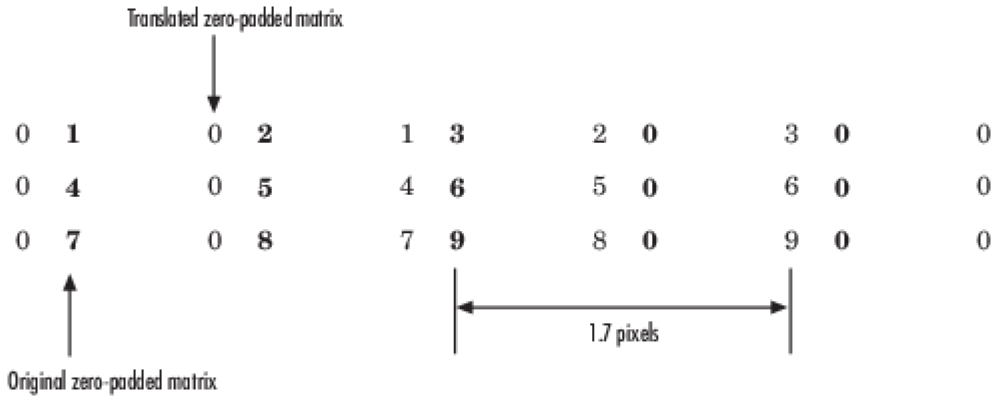
For nearest neighbor interpolation, the block uses the value of nearby translated pixel values for the output pixel values.

For example, suppose this matrix,

147258369

represents your input image. You want to translate this image 1.7 pixels in the positive horizontal direction using nearest neighbor interpolation. The Translate block's nearest neighbor interpolation algorithm is illustrated by the following steps:

1. Zero pad the input matrix and translate it by 1.7 pixels to the right.



2. Create the output matrix by replacing each input pixel value with the translated value nearest to it. The result is the following matrix:

000000147258369

Bilinear Interpolation

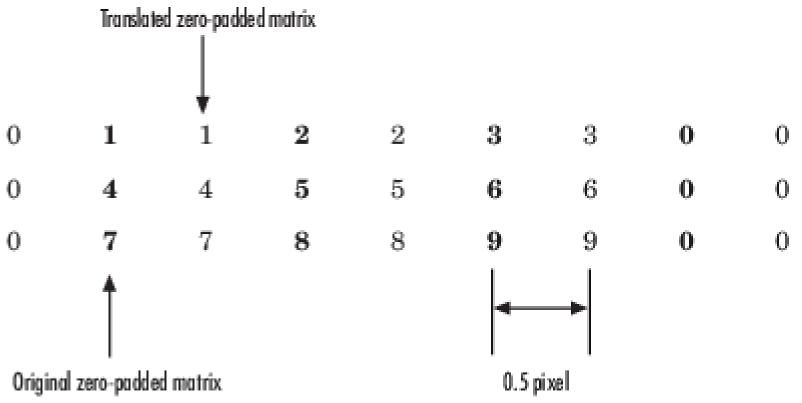
For bilinear interpolation, the block uses the weighted average of two translated pixel values for each output pixel value.

For example, suppose this matrix,

147258369

represents your input image. You want to translate this image 0.5 pixel in the positive horizontal direction using bilinear interpolation. The Translate block's bilinear interpolation algorithm is illustrated by the following steps:

1. Zero pad the input matrix and translate it by 0.5 pixel to the right.



2. Create the output matrix by replacing each input pixel value with the weighted average of the translated values on either side. The result is the following matrix where the output matrix has one more column than the input matrix:

0.523.51.54.57.52.55.58.51.534.5

Noise Removal

Digital images are prone to various types of noise. Noise is the result of errors in the image acquisition process that result in pixel values that do not reflect the true intensities of the real scene. There are several ways that noise can be introduced into an image, depending on how the image is created. For example:

- If the image is scanned from a photograph made on film, the film grain is a source of noise. Noise can also be the result of damage to the film, or be introduced by the scanner itself.
- If the image is acquired directly in a digital format, the mechanism for gathering the data (such as a CCD detector) can introduce noise.
- Electronic transmission of image data can introduce noise.

To simulate the effects of some of the problems listed above, the toolbox provides the `imnoise` function, which you can use to *add* various types of noise to an image. The examples in this section use this function.

Remove Noise by Linear Filtering

You can use linear filtering to remove certain types of noise. Certain filters, such as averaging or Gaussian filters, are appropriate for this purpose. For example, an averaging filter is useful for removing grain noise from a photograph. Because each pixel gets set to the average of the pixels in its neighborhood, local variations caused by grain are reduced.

Remove Noise Using an Averaging Filter and a Median Filter

We can remove salt and pepper noise from an image using an averaging filter and a median filter to allow comparison of the results. These two types of filtering both set the value of the output pixel to the average of the pixel values in the neighborhood around the corresponding input pixel. However, with median filtering, the value of an output pixel is determined by the median of the neighborhood pixels, rather than the mean. The median is much less sensitive than the mean to extreme values (called outliers). Median filtering is therefore better able to remove these outliers without reducing the sharpness of the image.

2-D median filtering:

Syntax:

```
B = medfilt2(A)
B = medfilt2(A,[m n])
B = medfilt2(___,padopt)
gpuarrayB = medfilt2(gpuarrayA)gpuarrayB = medfilt2(gpuarrayA,[m n])
```

6. Implementation Details:

The above project was implemented using Matlab.

Source Code:

GUI using Matlab:

```
function varargout = testing(varargin)
% TESTING MATLAB code for testing.fig
%      TESTING, by itself, creates a new TESTING or raises the existing
%      singleton*.
%
%      H = TESTING returns the handle to a new TESTING or the handle to
%      the existing singleton*.
%
%      TESTING('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in TESTING.M with the given input arguments.
%
%      TESTING('Property','Value',...) creates a new TESTING or raises the
%      existing singleton*.  Starting from the left, property value pairs
are
%      applied to the GUI before testing_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to testing_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help testing

% Last Modified by GUIDE v2.5 15-May-2018 18:33:00

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @testing_OpeningFcn, ...
                   'gui_OutputFcn',  @testing_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
```

```
% End initialization code - DO NOT EDIT


% --- Executes just before testing is made visible.
function testing_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to testing (see VARARGIN)

% Choose default command line output for testing
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes testing wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = testing_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA

% --- Executes on button press in near.
function near_Callback(hObject, eventdata, handles)
% hObject    handle to near (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
[filename pathname] = uigetfile({'*.bmp';'*.jpg'},'File Selector');
i=strcat(pathname,filename);
im=imread(i);
axes(handles.axes1);
imshow(im);
fac=str2num(get(handles.edit8,'string'));
out=nearest_neighbour_zoom(im,fac);
fileinfo=imfinfo(i);
sizew=fileinfo.Width(1,1);
sizeh=fileinfo.Height(1,1);
f_size=(fileinfo.FileSize(1,1))/1024;
axes(handles.axes2);
imshow(out);
set(handles.edit1,'string',sizew);
set(handles.edit2,'string',sizeh);
set(handles.edit3,'string',f_size);
imwrite(out,'near_zoom.jpg');
set(handles.edit4,'string','The zoomed file is saved as near_zoom ');
final=('D:\matlab projects\interpolation\near_zoom.jpg');
fileinfo=imfinfo(final);
sizew=fileinfo.Width(1,1);
sizeh=fileinfo.Height(1,1);
f_size=(fileinfo.FileSize(1,1))/1024;
set(handles.edit5,'string',sizew);
set(handles.edit6,'string',sizeh);
set(handles.edit7,'string',f_size);


% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
[filename pathname] = uigetfile({'*.bmp';'*.jpg'},'File Selector');
i=strcat(pathname,filename);
im=imread(i);
axes(handles.axes1);
imshow(im);
fac=str2num(get(handles.edit8,'string'));
out=bilinear_zoom(im,fac);
fileinfo=imfinfo(i);
sizew=fileinfo.Width(1,1);
sizeh=fileinfo.Height(1,1);
f_size=(fileinfo.FileSize(1,1))/1024;
axes(handles.axes2);
imshow(out);
set(handles.edit1,'string',sizew);
set(handles.edit2,'string',sizeh);
set(handles.edit3,'string',f_size);
imwrite(out,'bilinear_zoom.jpg');
set(handles.edit4,'string','The zoomed file is saved as bilinear_zoom ');
final=('D:\matlab projects\interpolation\bilinear_zoom.jpg');
fileinfo=imfinfo(final);
sizew=fileinfo.Width(1,1);
sizeh=fileinfo.Height(1,1);
f_size=(fileinfo.FileSize(1,1))/1024;
set(handles.edit5,'string',sizew);
set(handles.edit6,'string',sizeh);
set(handles.edit7,'string',f_size);


% --- Executes during object creation, after setting all properties.
function pushbutton1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called



function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1 as a
double


% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2 as a
double


% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit3_Callback(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit3 as text
%        str2double(get(hObject,'String')) returns contents of edit3 as a
double


% --- Executes during object creation, after setting all properties.
function edit3_CreateFcn(hObject, eventdata, handles)
```

```
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit4_Callback(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit4 as text
%        str2double(get(hObject,'String')) returns contents of edit4 as a
double


% --- Executes during object creation, after setting all properties.
function edit4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit5_Callback(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit5 as text
%        str2double(get(hObject,'String')) returns contents of edit5 as a
double


% --- Executes during object creation, after setting all properties.
function edit5_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
```

```
if ispc && isequal(get(hObject,'BackgroundColor'), 
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit6_Callback(hObject, eventdata, handles)
% hObject    handle to edit6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit6 as text
%        str2double(get(hObject,'String')) returns contents of edit6 as a 
double


% --- Executes during object creation, after setting all properties.
function edit6_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), 
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit7_Callback(hObject, eventdata, handles)
% hObject    handle to edit7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit7 as text
%        str2double(get(hObject,'String')) returns contents of edit7 as a 
double


% --- Executes during object creation, after setting all properties.
function edit7_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), 
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function edit8_Callback(hObject, eventdata, handles)
% hObject    handle to edit8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit8 as text
%        str2double(get(hObject,'String')) returns contents of edit8 as a
double


% --- Executes during object creation, after setting all properties.
function edit8_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

bilinear interpolation using matlab:

```
% bilenear_zoom.m
% Zooming function using bilenear method
function Img_zoomed = bilenear_zoom(Img, factor)

[h w c] = size(Img);
r = factor;
hn = r*(h-1)+1; wn = r*(w-1)+1;
Img_zoomed = zeros(hn, wn, c);

% Padding the scaled image with 0 pixel value
for i= 1:h
   for j= 1:w
      Img_zoomed(r*(i-1)+1, r*(j-1)+1, : ) = Img(i, j, : );
   end
end

% Interpolation
for k= 1:c
for i= 0:r:hn-r
for j= 0:r:wn-r
A = Img_zoomed(i+1, j+1, k);
B = Img_zoomed(i+1, j+r+1, k);
C = Img_zoomed(i+r+1, j+1, k);
D = Img_zoomed(i+r+1, j+r+1, k);

a0 = A;
         a1 = double((B-A)/r);
         a2 = double((C-A)/r);
         a3 = double((D-C-B+A)/(r*r));

         for l= 0:r
```

```
            for m= 0:r
               Img_zoomed(i+l+1, j+m+1, k) = a0 + a1*m + a2*l + a3*m*l;
            end
         end
      end
   end
end

Img_zoomed = uint8(Img_zoomed);

end
```

nearest neighbour interpolation using matlab:

```
% Zooming function using nearest neighbour method

function Img_zoomed = nearest_neighbour_zoom(Img, factor)

[h w c] = size(Img);
wn = w*factor;
hn = h*factor;
Img_zoomed = uint8(zeros(hn, wn));

for i= 0:hn-1
  for j= 0:wn-1
    x = floor(j/factor);
    y = floor(i/factor);
    for k= 1:c
      Img_zoomed(i+1, j+1, k) = Img(y+1, x+1, k);
    end
  end
end

end
```

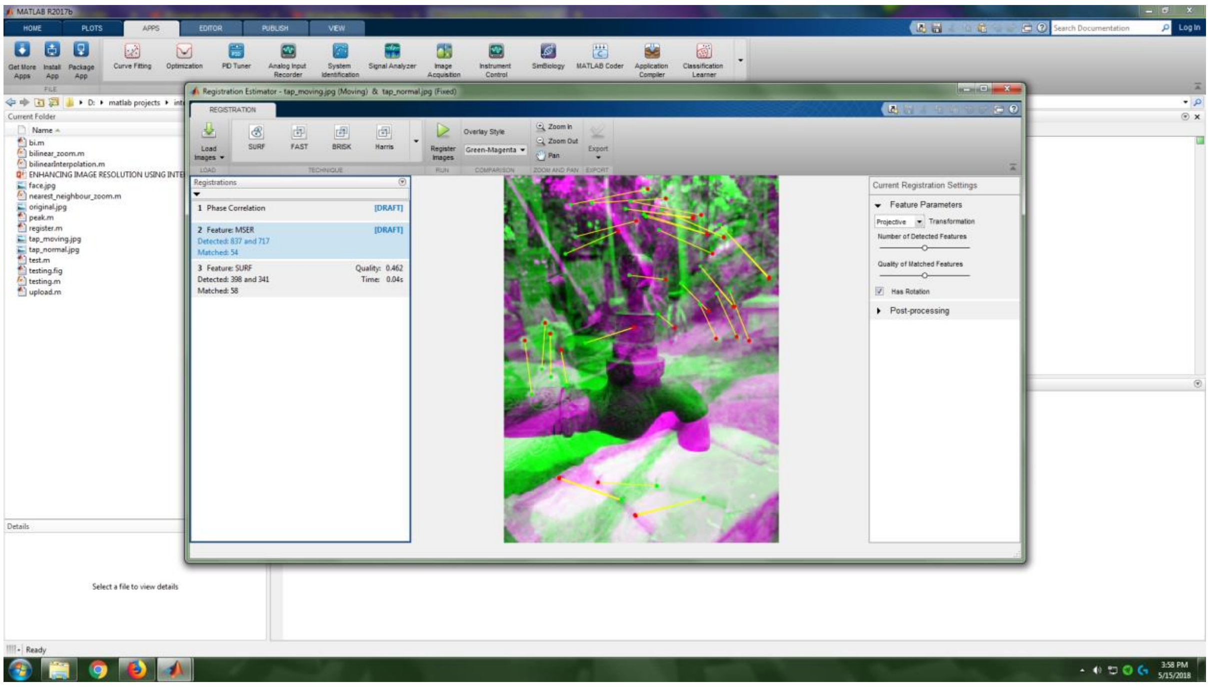
Noise Removal using Matlab:

```
i=imread('bi_zoomed.jpg');
s=imsharpen(i);
m=medfilt2(s);
imshow(m);
imwrite(m,'final.jpg');
```

7. Results/Sample output:

2 Low resolution images:

Image registration using matlab registration estimator:



Registration Estimator - tap_moving.jpg (Moving) & tap_normal.jpg (Fixed)

Registrations

1 Phase Correlation [DRAFT]

2 Feature: MSER [DRAFT]
Detected: 837 and 717
Matched: 54

3 Feature: SURF Quality: 0.462
Detected: 398 and 341 Time: 0.04s
Matched: 58

Current Registration Settings

Feature Parameters

Projective Transformation

Number of Detected Features

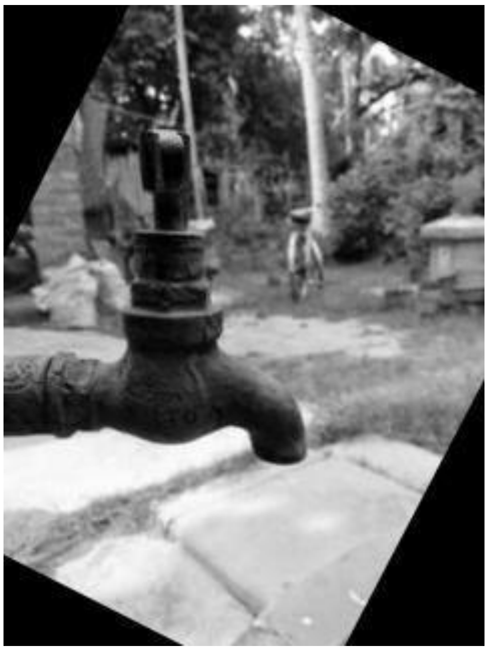
Quality of Matched Features

[✓] Has Rotation

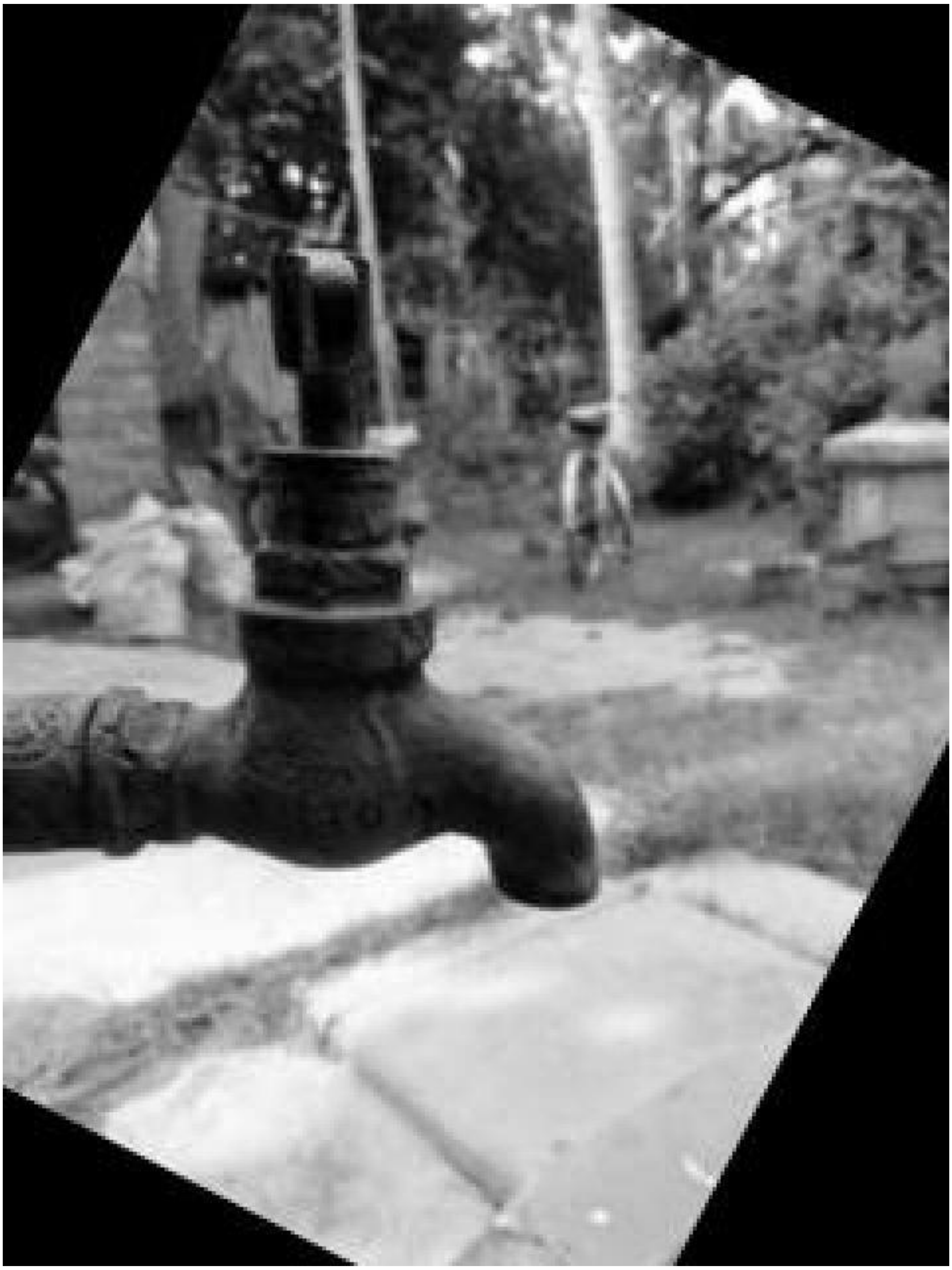
Post-processing

Overlay Style: Green-Magenta

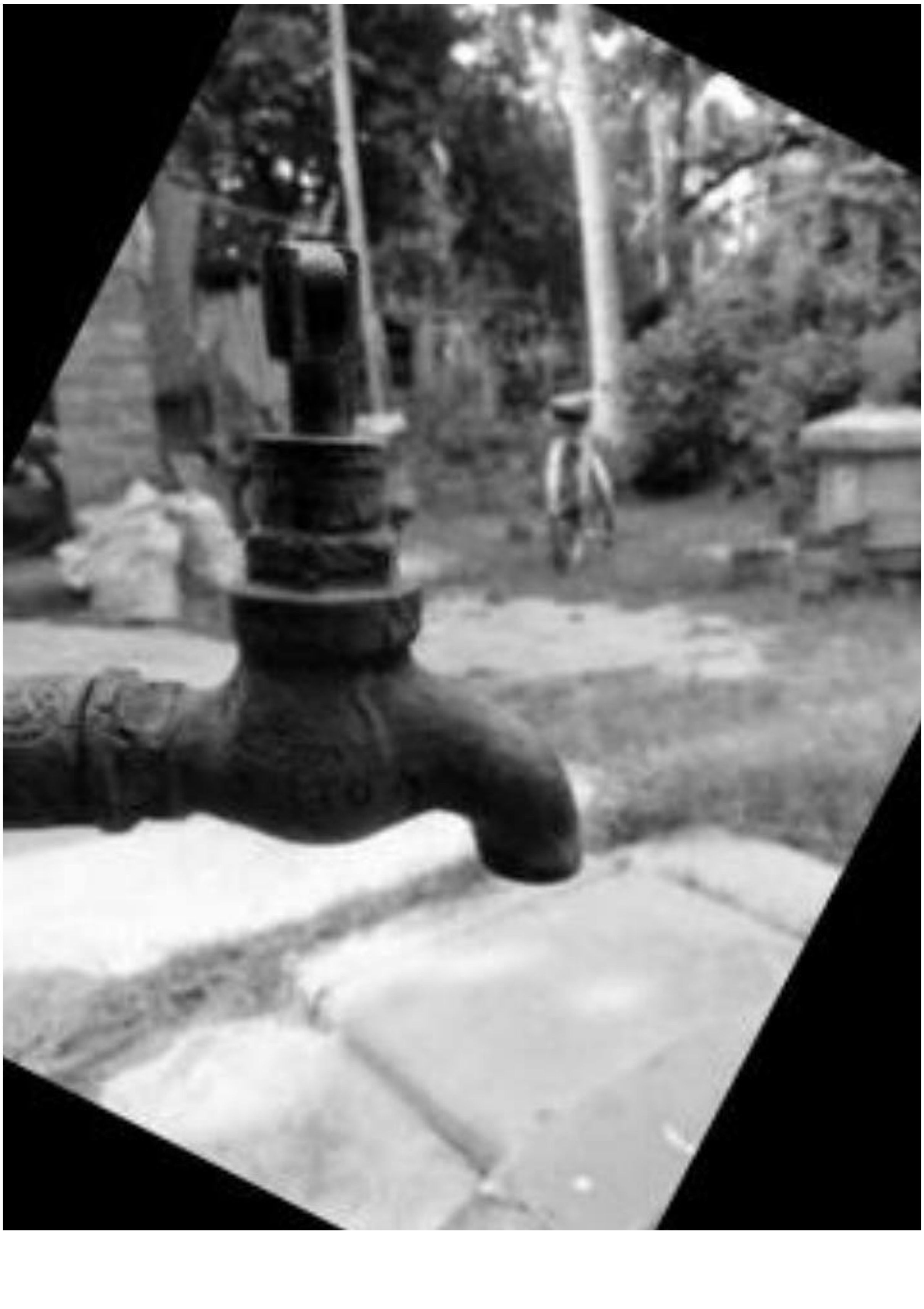
After registration:



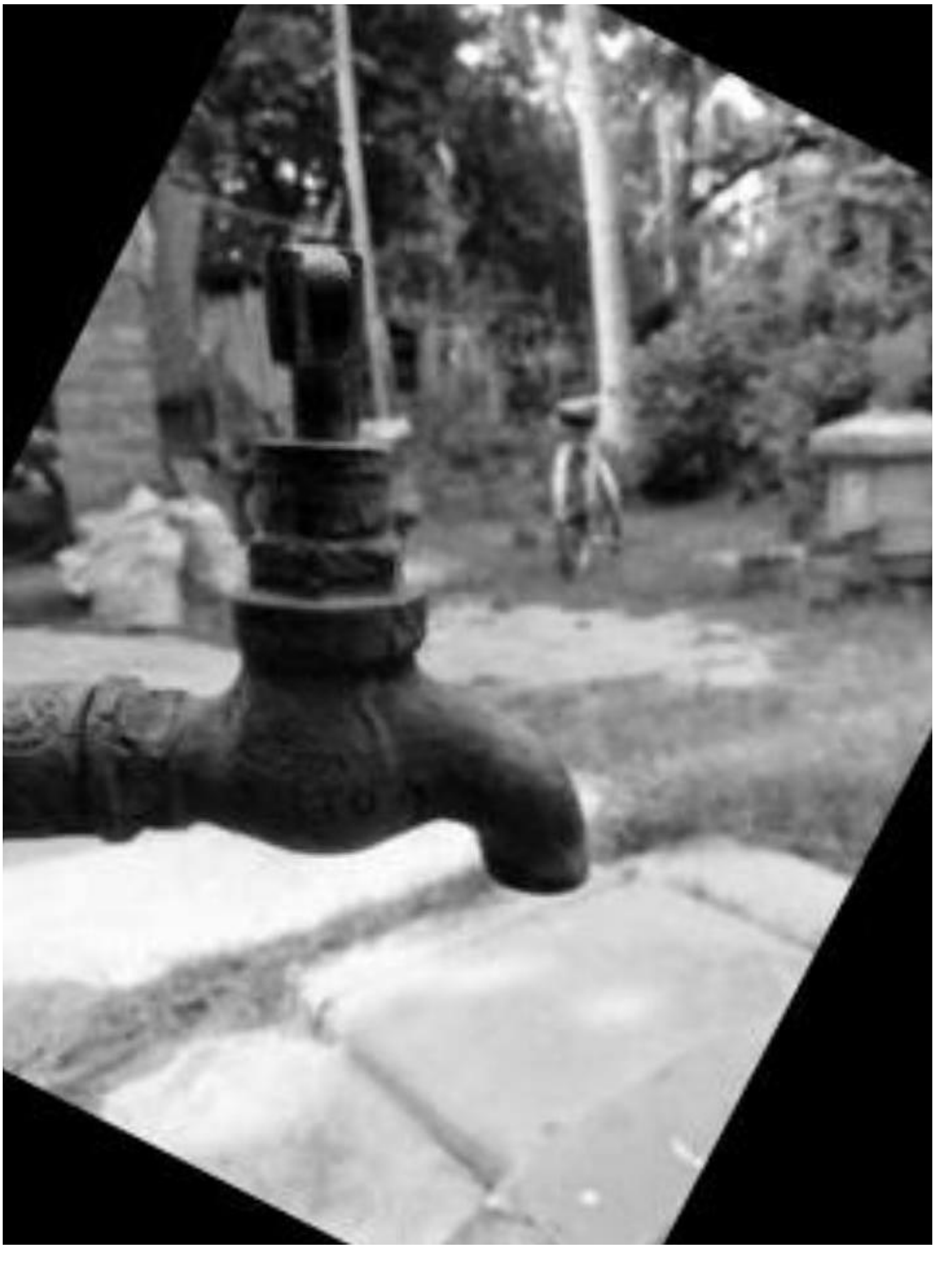
After nearest neighbour interpolation:



After bilinear interpolation:



After noise removal:



8.APPLICATIONS:

Several practical areas and applications of SR as follows:

1. **Biometrics** – Fingerprint recognition, Face recognition, Character recognition, DNA analysis.

2.**Medical Science** – MRI, CT, X-Ray, Ultrasound

3. **Satellite Imaging** – Planetary information, Weather forecasting, Target detection, Traffic detection.

4. **Surveillance Video** – Zooming region of interest (ROI). E.g. license plate recognition of vehicle, target recognition

5. **Entertainment** – HDTV, Photography.

6. **Commercial** – Barcode reading.

7. **Military** – Tracking and Detecting

9.Challenges for super resolution:

In practical building SR image, there are several challenges and issues regarding that. Some of them are as follows:

(i). Image Registration:

In an image, image registration is a well-known problem known by the name of ill-posed image. Image registration becomes more and more difficult when observed LR image is having very high aliasing effects. The registration error increases with decrease in the resolution of observed image. The degradation caused by these registration errors affects the quality of an image resolution more than that of interpolation

(ii). Computational Efficiency:

Real time application is always requiring good efficiency. As there are large numbers of unknowns in reconstructing SR images, matrix manipulation increases [10].

10.Conclusion:

We have explained the concept of SR technology in this article by providing an overview of existing SR algorithms and advanced issues currently under investigation. We specified interpolation based, reconstruction based and learning based approaches to achieve the goal. We also include applications and comparison of different SR approaches. SR image reconstruction is one of the most spotlighted research areas, because it can overcome the inherent resolution limitation of the imaging system and improve the performance of most digital image processing applications.