BSCE417P-MACHINE VISION

LAB ASSIGNMENT 1

S ARAVIND KANDHASAAMMY - 21BAI1425

September 11,2024

By turning in this assignment, I declare that all of the following is of my own work.

Repository Link:

Re used Code:

import numpy as np

import matplotlib.pyplot as plt

import cv2

def grayscale(i): return np.dot(i[...,:3],[0.299,0.299,0.299])

def imshow(i): plt.imshow(i,cmap = 'gray')

def distort(img):

h, w = img.shape[:2]

pts\_original = np.float32([[0, 0], [w, 0], [w, h], [0, h]])

pts\_distorted = np.float32([[np.random.randint(0, 50), np.random.randint(0, 50)],

[w - np.random.randint(0, 50), np.random.randint(0, 50)],

[w - np.random.randint(0, 50), h - np.random.randint(0, 50)],

[np.random.randint(0, 50), h - np.random.randint(0, 50)]])

matrix = cv2.getPerspectiveTransform(pts\_original, pts\_distorted)

dst = cv2.warpPerspective(img, matrix, (w, h))

return dst

# Task1: Billinear Interpolation

img = grayscale(plt.imread("/Users/AravindKandhasaamy/Documents/Work/Sem 7/Machine Vision/sateliteimage.jpeg"))

dst = distort(img)

plt.figure(figsize = (10,5))

plt.subplot(1,2,1);plt.title("Original Image");imshow(img);plt.axis('off')

plt.subplot(1,2,2);plt.title("Distorted image");imshow(dst); plt.axis('off')

coordinates1 = []

def get\_GCPs1(event, x, y, flags, param,coordinates):

if event == cv2.EVENT\_LBUTTONDOWN:

coordinates1.append((x, y))

**# Mark the selected point on the image**

cv2.circle(dst, (x, y), 5, (0, 0, 255), -1)

cv2.imshow('Distorted Image', dst)

print(f'GCP selected: {x}, {y}')

cv2.imshow('Distorted Image', dst)

cv2.setMouseCallback('Distorted Image', get\_GCPs1)

while True:

if cv2.waitKey(1) & 0xFF == ord('q'):

break

cv2.destroyAllWindows()

print("Distorted GCPs: ", coordinates1)

**# Define method to get GCPs from the image**

coordinates2 = []

def get\_GCPs(event, x, y, flags, param):

if event == cv2.EVENT\_LBUTTONDOWN:

cv2.circle(dst, (x, y), 5, (0, 0, 255), -1)

cv2.imshow('Distorted Image', dst)

print(f'GCP selected: {x}, {y}')

cv2.imshow('Distorted Image', dst)

cv2.setMouseCallback('Distorted Image', get\_GCPs2)

while True:

if cv2.waitKey(1) & 0xFF == ord('q'):

break

cv2.destroyAllWindows()

print("Rectified GCPs: ", coordinates2)

GCP\_distorted = np.array(coordinates1, dtype=np.float32)

GCP\_rectified = np.array(coordinates2, dtype=np.float32)

transformation\_matrix = cv2.getPerspectiveTransform(GCP\_distorted, GCP\_rectified)

print("Transformation Matrix: ", transformation\_matrix)

r\_img = cv2.warpPerspective(dst, transformation\_matrix, (400, 400), flags=cv2.INTER\_LINEAR)

plt.figure(figsize=(10, 5))

plt.subplot(1, 2, 1)

plt.title('Distorted Image')

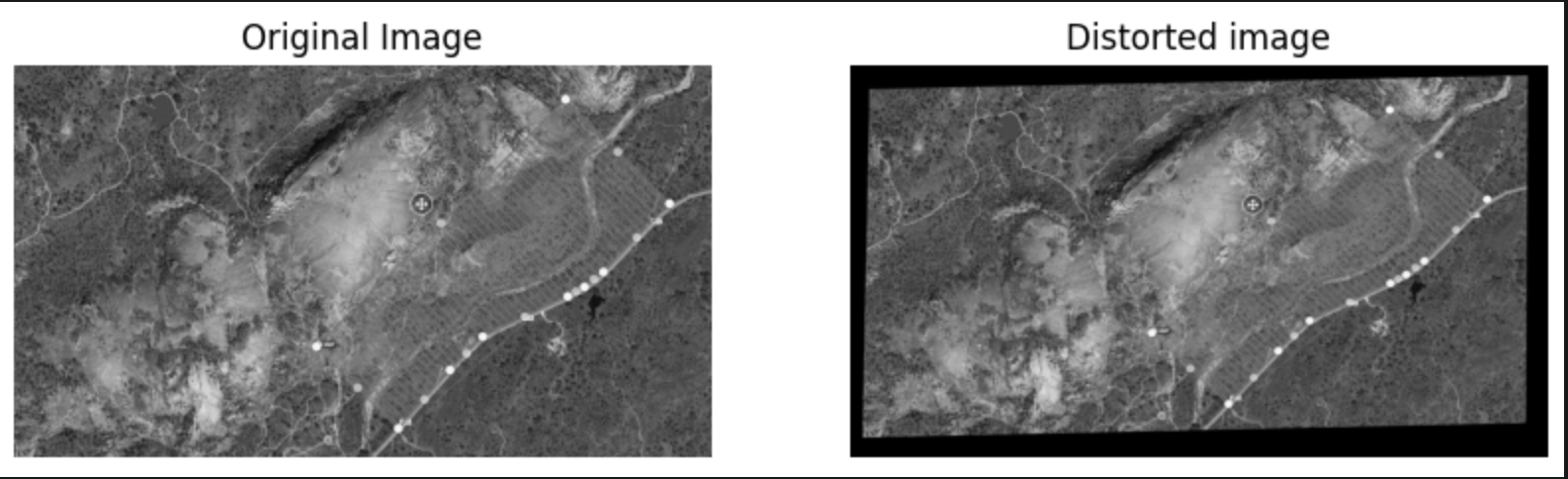
plt.imshow(cv2.cvtColor(dst, cv2.COLOR\_BGR2RGB))

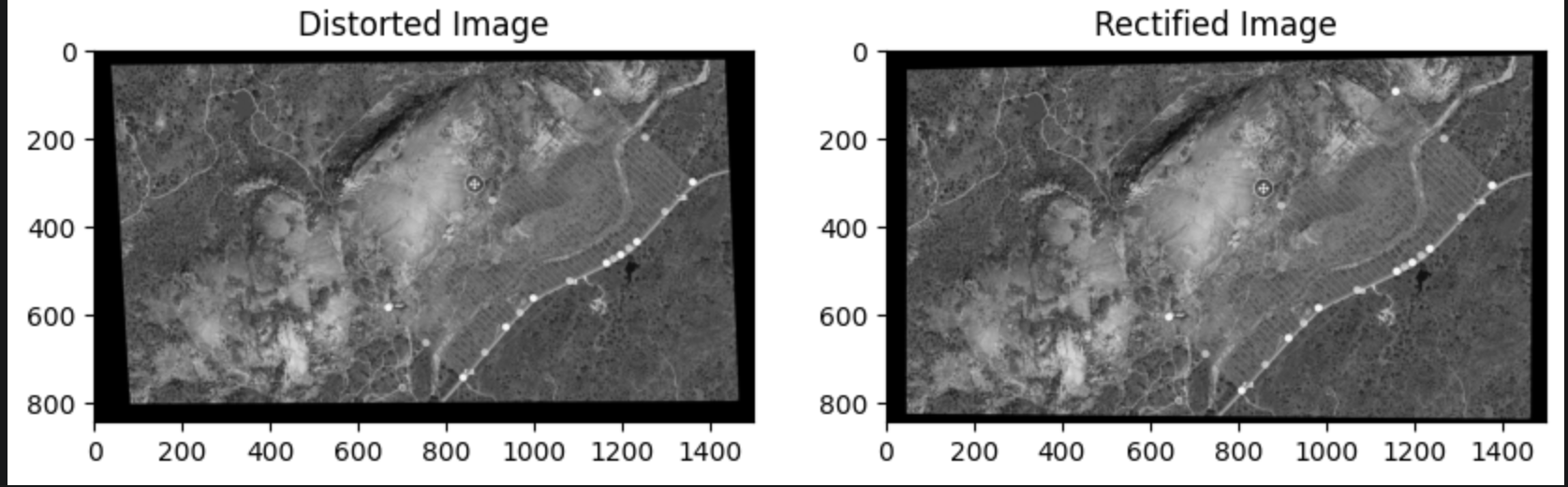
plt.subplot(1, 2, 2)

plt.title('Rectified Image')

plt.imshow(cv2.cvtColor(r\_img, cv2.COLOR\_BGR2RGB))

Output:

After Rectification:



## Observations

The procedure is as follows:

* Distorting an image
* Via a cv2 window, get 4 points (GCP) for the distorted image and where the rectified image shall be
* Get transform matrix and apply it

Challenges were faced in choosing the right GCP points, and the right choice was taken when choosing points in such a way that the dimensions of the image and its alignment were roughly preserved in the chosen points as well.

## TASK2: MRI IMAGE RECTIFICATION

Code:

img = grayscale(plt.imread("/Users/AravindKandhasaamy/Documents/Work/Sem 7/Machine Vision/mri.jpeg"))

dst = distort(img)

plt.figure(figsize = (10,5))

plt.subplot(1,2,1);plt.title("Original Image");imshow(img);plt.axis('off')

plt.subplot(1,2,2);plt.title("Distorted image");imshow(dst); plt.axis('off')

coordinates1 = []

def get\_GCPs1(event, x, y, flags, param,coordinates):

if event == cv2.EVENT\_LBUTTONDOWN:

**# On left mouse button click, append GCPs**

coordinates1.append((x, y))

**# Mark the selected point on the image**

cv2.circle(dst, (x, y), 5, (0, 0, 255), -1)

cv2.imshow('Distorted Image', dst)

print(f'GCP selected: {x}, {y}')

cv2.imshow('Distorted Image', dst)

cv2.setMouseCallback('Distorted Image', get\_GCPs1)

while True:

if cv2.waitKey(1) & 0xFF == ord('q'):

break

cv2.destroyAllWindows()

print("Distorted GCPs: ", coordinates1)

**# Define method to get GCPs from the image**

coordinates2 = []

def get\_GCPs(event, x, y, flags, param):

if event == cv2.EVENT\_LBUTTONDOWN:

cv2.circle(dst, (x, y), 5, (0, 0, 255), -1)

cv2.imshow('Distorted Image', dst)

print(f'GCP selected: {x}, {y}')

cv2.imshow('Distorted Image', dst)

cv2.setMouseCallback('Distorted Image', get\_GCPs2)

while True:

if cv2.waitKey(1) & 0xFF == ord('q'):

break

cv2.destroyAllWindows()

print("Rectified GCPs: ", coordinates2)

GCP\_distorted = np.array(coordinates1, dtype=np.float32)

GCP\_rectified = np.array(coordinates2, dtype=np.float32)

transformation\_matrix = cv2.getPerspectiveTransform(GCP\_distorted, GCP\_rectified)

print("Transformation Matrix: ", transformation\_matrix)

r\_img = cv2.warpPerspective(dst, transformation\_matrix, (400, 400), flags=cv2.INTER\_LINEAR)

plt.figure(figsize=(10, 5))

plt.subplot(1, 2, 1)

plt.title('Distorted Image')

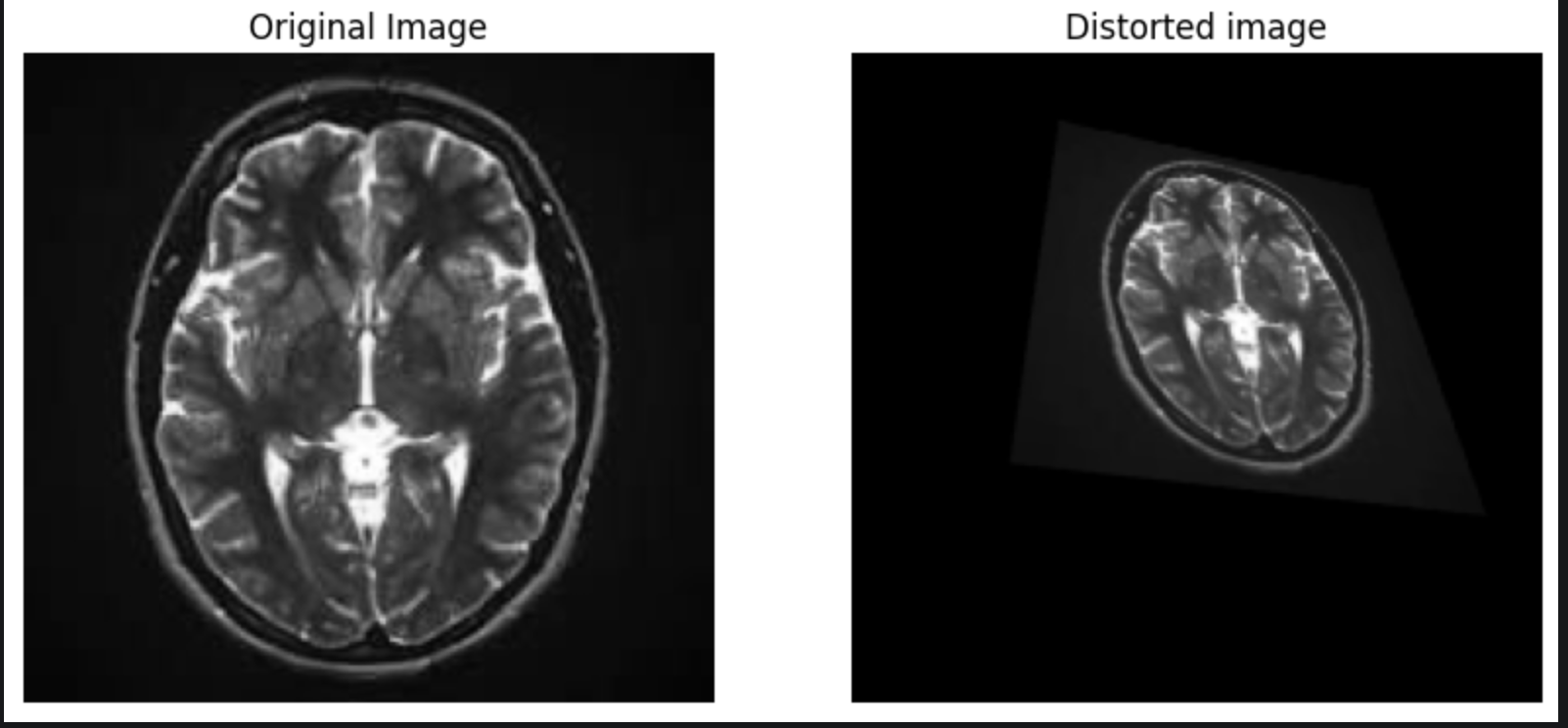
plt.imshow(cv2.cvtColor(dst, cv2.COLOR\_BGR2RGB))

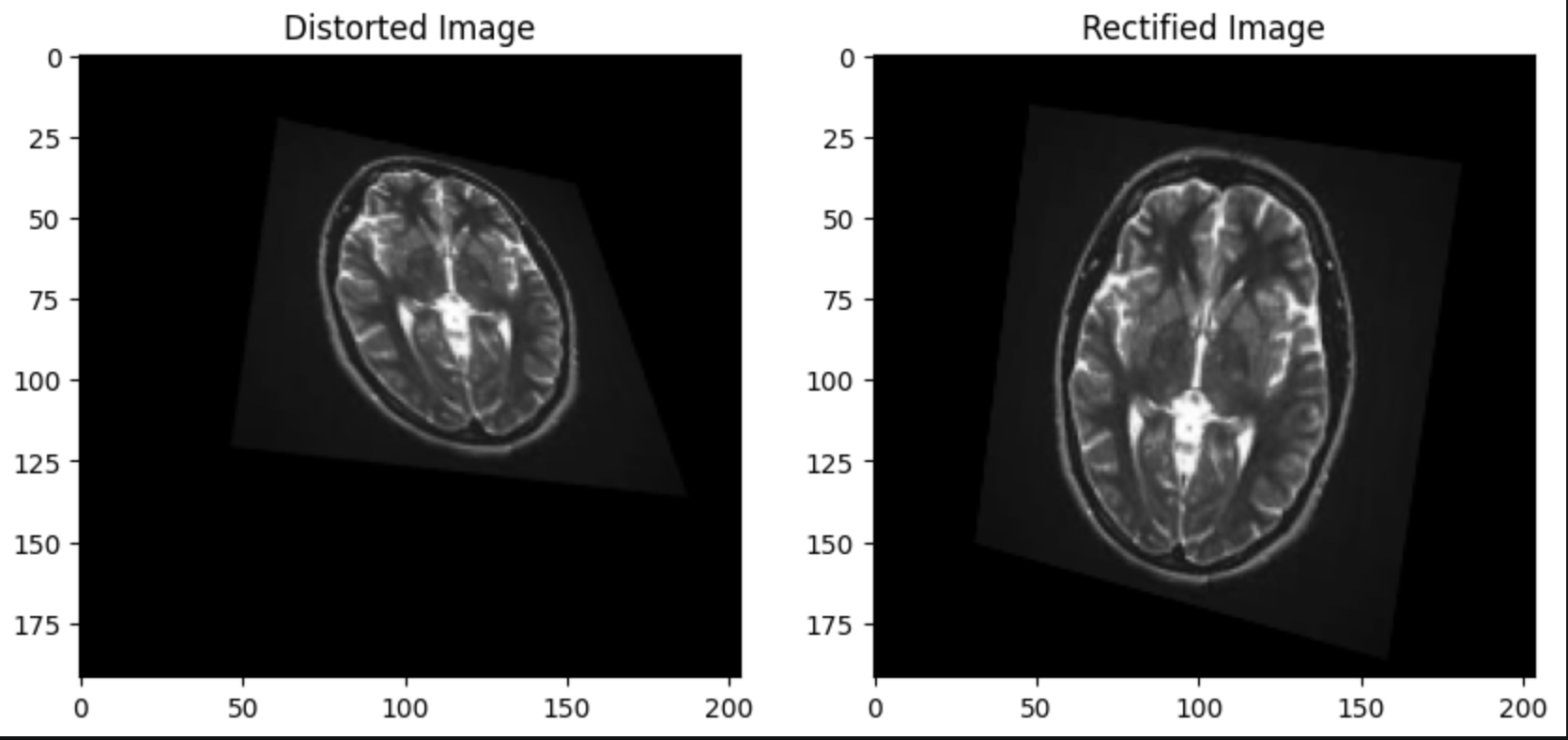
plt.subplot(1, 2, 2)

plt.title('Rectified Image')

plt.imshow(cv2.cvtColor(r\_img, cv2.COLOR\_BGR2RGB))

Output:





As noticeable above, even with simple bilinear interpolation the effects of the affine distortion, the change in alignment and focus are tough resolve and this was the best outcome possible with choosing random points for this.

## Task 3 – Drone Image

img = grayscale(plt.imread("/Users/AravindKandhasaamy/Documents/Work/Sem 7/Machine Vision/mri.jpeg"))

dst = distort(img)

plt.figure(figsize = (10,5))

plt.subplot(1,2,1);plt.title("Original Image");imshow(img);plt.axis('off')

plt.subplot(1,2,2);plt.title("Distorted image");imshow(dst); plt.axis('off')

coordinates1 = []

def get\_GCPs1(event, x, y, flags, param,coordinates):

if event == cv2.EVENT\_LBUTTONDOWN:

**# On left mouse button click, append GCPs**

coordinates1.append((x, y))

**# Mark the selected point on the image**

cv2.circle(dst, (x, y), 5, (0, 0, 255), -1)

cv2.imshow('Distorted Image', dst)

print(f'GCP selected: {x}, {y}')

cv2.imshow('Distorted Image', dst)

cv2.setMouseCallback('Distorted Image', get\_GCPs1)

while True:

if cv2.waitKey(1) & 0xFF == ord('q'):

break

cv2.destroyAllWindows()

print("Distorted GCPs: ", coordinates1)

**# Define method to get GCPs from the image**

coordinates2 = []

def get\_GCPs(event, x, y, flags, param):

if event == cv2.EVENT\_LBUTTONDOWN:

cv2.circle(dst, (x, y), 5, (0, 0, 255), -1)

cv2.imshow('Distorted Image', dst)

print(f'GCP selected: {x}, {y}')

cv2.imshow('Distorted Image', dst)

cv2.setMouseCallback('Distorted Image', get\_GCPs2)

while True:

if cv2.waitKey(1) & 0xFF == ord('q'):

break

cv2.destroyAllWindows()

print("Rectified GCPs: ", coordinates2)

GCP\_distorted = np.array(coordinates1, dtype=np.float32)

GCP\_rectified = np.array(coordinates2, dtype=np.float32)

transformation\_matrix = cv2.getPerspectiveTransform(GCP\_distorted, GCP\_rectified)

print("Transformation Matrix: ", transformation\_matrix)

r\_img = cv2.warpPerspective(dst, transformation\_matrix, (400, 400), flags=cv2.INTER\_LINEAR)

plt.figure(figsize=(10, 5))

plt.subplot(1, 2, 1)

plt.title('Distorted Image')

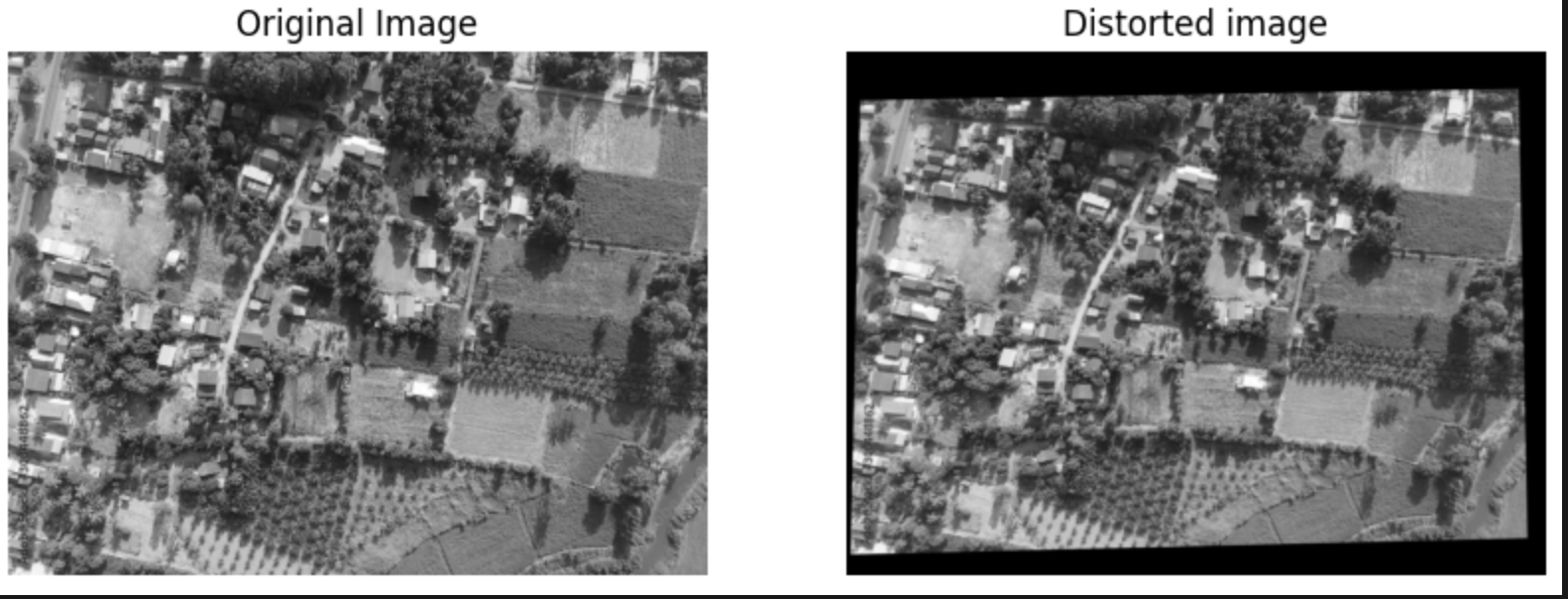
plt.imshow(cv2.cvtColor(dst, cv2.COLOR\_BGR2RGB))

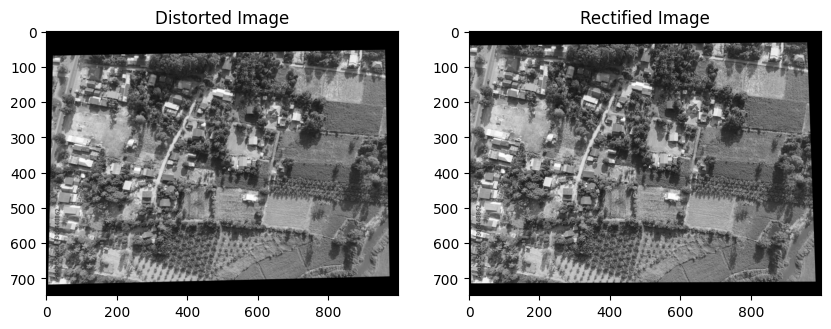
plt.subplot(1, 2, 2)

plt.title('Rectified Image')

plt.imshow(cv2.cvtColor(r\_img, cv2.COLOR\_BGR2RGB))

## Output





With user chosen points, this was the best possible result after an arduous phase of trial and error. Affine distortions are difficult to fix and the lopsided scales are one result of it.

## Task 4 – Historical image

## Code:

img = grayscale(plt.imread("/Users/AravindKandhasaamy/Documents/Work/Sem 7/Machine Vision/historical.jpeg"))

dst = distort(img)

plt.figure(figsize = (10,5))

plt.subplot(1,2,1);plt.title("Original Image");imshow(img);plt.axis('off')

plt.subplot(1,2,2);plt.title("Distorted image");imshow(dst); plt.axis('off')

coordinates1 = []

def get\_GCPs1(event, x, y, flags, param,coordinates):

if event == cv2.EVENT\_LBUTTONDOWN:

**# On left mouse button click, append GCPs**

coordinates1.append((x, y))

**# Mark the selected point on the image**

cv2.circle(dst, (x, y), 5, (0, 0, 255), -1)

cv2.imshow('Distorted Image', dst)

print(f'GCP selected: {x}, {y}')

cv2.imshow('Distorted Image', dst)

cv2.setMouseCallback('Distorted Image', get\_GCPs1)

while True:

if cv2.waitKey(1) & 0xFF == ord('q'):

break

cv2.destroyAllWindows()

print("Distorted GCPs: ", coordinates1)

**# Define method to get GCPs from the image**

coordinates2 = []

def get\_GCPs(event, x, y, flags, param):

if event == cv2.EVENT\_LBUTTONDOWN:

cv2.circle(dst, (x, y), 5, (0, 0, 255), -1)

cv2.imshow('Distorted Image', dst)

print(f'GCP selected: {x}, {y}')

cv2.imshow('Distorted Image', dst)

cv2.setMouseCallback('Distorted Image', get\_GCPs2)

while True:

if cv2.waitKey(1) & 0xFF == ord('q'):

break

cv2.destroyAllWindows()

print("Rectified GCPs: ", coordinates2)

GCP\_distorted = np.array(coordinates1, dtype=np.float32)

GCP\_rectified = np.array(coordinates2, dtype=np.float32)

transformation\_matrix = cv2.getPerspectiveTransform(GCP\_distorted, GCP\_rectified)

print("Transformation Matrix: ", transformation\_matrix)

r\_img = cv2.warpPerspective(dst, transformation\_matrix, (400, 400), flags=cv2.INTER\_LINEAR)

plt.figure(figsize=(10, 5))

plt.subplot(1, 2, 1)

plt.title('Distorted Image')

plt.imshow(cv2.cvtColor(dst, cv2.COLOR\_BGR2RGB))

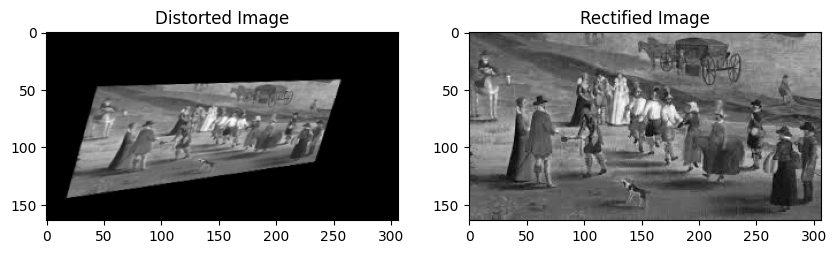
plt.subplot(1, 2, 2)

plt.title('Rectified Image')

plt.imshow(cv2.cvtColor(r\_img, cv2.COLOR\_BGR2RGB))

## Output:





With well-chosen and defined set of 4 points, it is possible to nearly fix all distortions albeit the rectified image loses a few pixels at its edges for a sharper re creation.

## Task 5- Architecture

## Code:

img = grayscale(plt.imread("/Users/AravindKandhasaamy/Documents/Work/Sem 7/Machine Vision/architecture.jpeg"))

dst = distort(img)

plt.figure(figsize = (10,5))

plt.subplot(1,2,1);plt.title("Original Image");imshow(img);plt.axis('off')

plt.subplot(1,2,2);plt.title("Distorted image");imshow(dst); plt.axis('off')

coordinates1 = []

def get\_GCPs1(event, x, y, flags, param,coordinates):

if event == cv2.EVENT\_LBUTTONDOWN:

**# On left mouse button click, append GCPs**

coordinates1.append((x, y))

**# Mark the selected point on the image**

cv2.circle(dst, (x, y), 5, (0, 0, 255), -1)

cv2.imshow('Distorted Image', dst)

print(f'GCP selected: {x}, {y}')

cv2.imshow('Distorted Image', dst)

cv2.setMouseCallback('Distorted Image', get\_GCPs1)

while True:

if cv2.waitKey(1) & 0xFF == ord('q'):

break

cv2.destroyAllWindows()

print("Distorted GCPs: ", coordinates1)

**# Define method to get GCPs from the image**

coordinates2 = []

def get\_GCPs(event, x, y, flags, param):

if event == cv2.EVENT\_LBUTTONDOWN:

cv2.circle(dst, (x, y), 5, (0, 0, 255), -1)

cv2.imshow('Distorted Image', dst)

print(f'GCP selected: {x}, {y}')

cv2.imshow('Distorted Image', dst)

cv2.setMouseCallback('Distorted Image', get\_GCPs2)

while True:

if cv2.waitKey(1) & 0xFF == ord('q'):

break

cv2.destroyAllWindows()

print("Rectified GCPs: ", coordinates2)

GCP\_distorted = np.array(coordinates1, dtype=np.float32)

GCP\_rectified = np.array(coordinates2, dtype=np.float32)

transformation\_matrix = cv2.getPerspectiveTransform(GCP\_distorted, GCP\_rectified)

print("Transformation Matrix: ", transformation\_matrix)

r\_img = cv2.warpPerspective(dst, transformation\_matrix, (400, 400), flags=cv2.INTER\_LINEAR)

plt.figure(figsize=(10, 5))

plt.subplot(1, 2, 1)

plt.title('Distorted Image')

plt.imshow(cv2.cvtColor(dst, cv2.COLOR\_BGR2RGB))

plt.subplot(1, 2, 2)

plt.title('Rectified Image')

plt.imshow(cv2.cvtColor(r\_img, cv2.COLOR\_BGR2RGB))

## Output:





Affine distortion was nearly resolved, albeit at the expense of pixels at the bottom of the image.

This was the best possible result after numerous trials.