Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

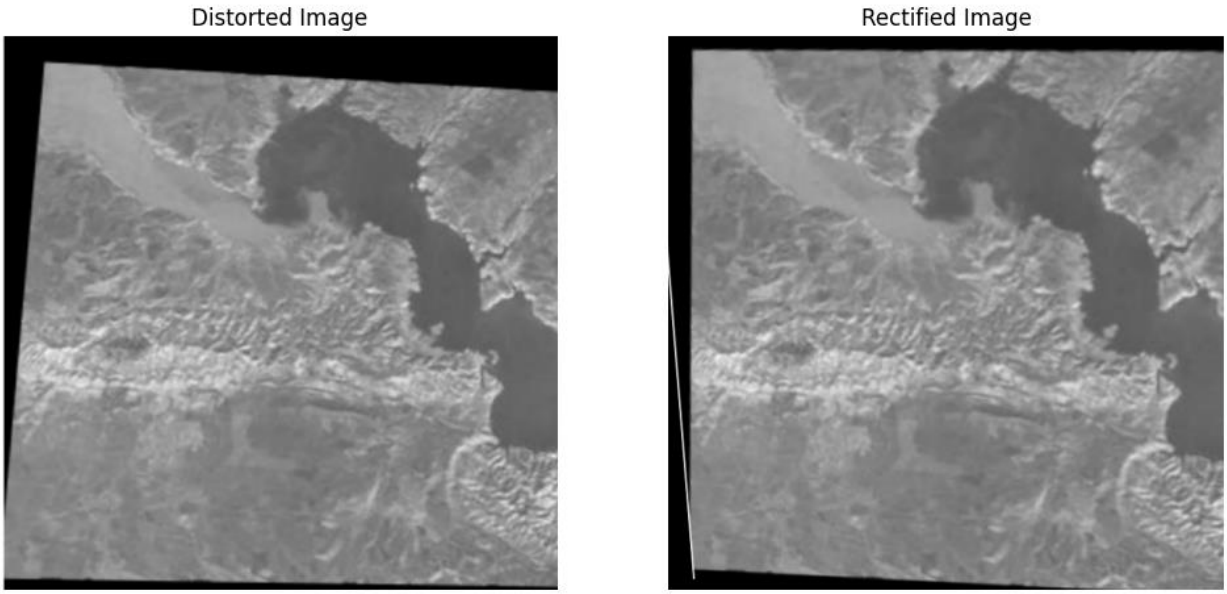
Lab Assignment 2 – Google Colab Link

Programme	:	B.Tech.	Semester	:	Fall 2024–25
Course Title	:	Machine Vision Lab	Code	:	BCSE417P
			Slot	:	L43+L44
Register Number	:	21BAI1106	Name	:	Ojas Patil
Faculty (s)	:	Dr. Saranyaraj D	Date	:	14/9/2024

Task 1 - Geometric Rectification of Satellite Imagery

```
# Define Ground Control Points (GCPs)

distorted_gcp = np.array([[110, 200], [400, 300], [600, 800], [900, 1000]], dtype=np.float32)

rectified_gcp = np.array([[100, 200], [400, 300], [600, 800], [900, 1000]], dtype=np.float32)

transformation_matrix = cv2.getPerspectiveTransform(distorted_gcp, rectified_gcp)


height, width = distorted_image.shape[:2]

rectified_image = cv2.warpPerspective(distorted_image, transformation_matrix, (width, height))

rectified_image_bilinear = cv2.warpPerspective(distorted_image, transformation_matrix, (width, height), flags=cv2.INTER_LINEAR)
```

Results



Learnings

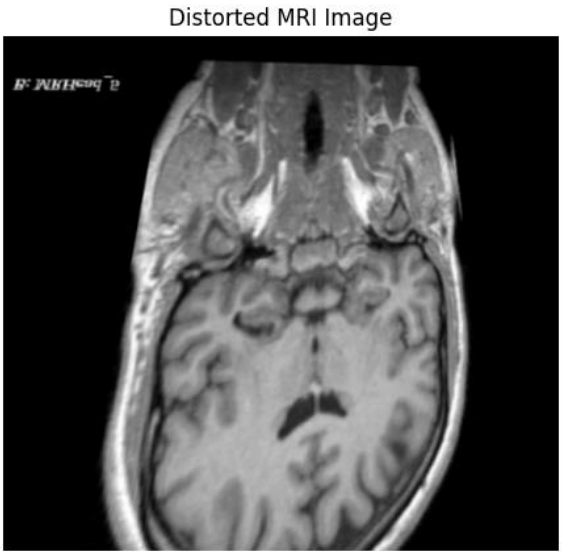
Using transformation matrices to map coordinates between images is a powerful method for geometric rectification. Bilinear interpolation helped preserve the visual quality of the rectified image, reducing pixelation and making transitions smoother.

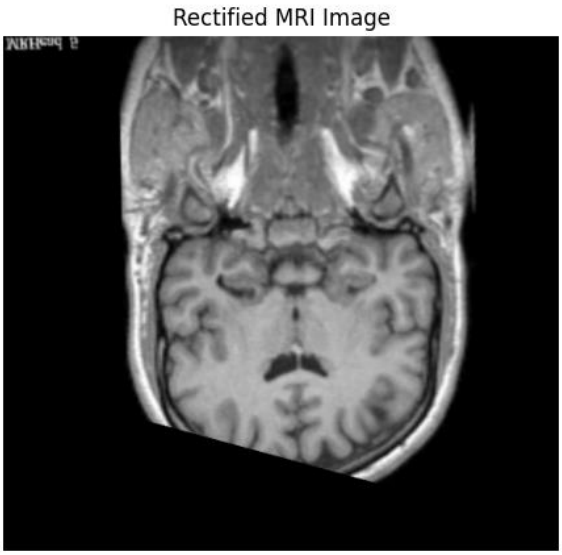
Task 2 - Medical Image Rectification

```
# distorted and reference points

distorted_key_points = np.array([[120, 270], [450, 300], [600, 500], [800, 700]], dtype=np.float32)

reference_key_points = np.array([[120, 220], [450, 300], [600, 500], [800, 700]], dtype=np.float32)



transformation_matrix = cv2.getPerspectiveTransform(distorted_key_points, reference_key_points)

height, width = distorted_mri.shape[:2]

rectified_mri = cv2.warpPerspective(distorted_mri, transformation_matrix, (width, height))
```

```
rectified_mri_bilinear = cv2.warpPerspective(distorted_mri,
transformation_matrix, (width, height), flags=cv2.INTER_LINEAR)
```

Results





Learnings

Bilinear interpolation produced smoother results, especially in regions with fine anatomical structures. The accuracy of the rectification heavily depends on the quality and precision of the chosen key points.

Task 3 – Drone Image Rectification

```
#computing basic statistics

distorted_control_points = np.array([[145, 350], [450, 450], [600, 750], [850,
950]], dtype=np.float32)

rectified_control_points = np.array([[150, 350], [450, 450], [600, 750], [850,
950]], dtype=np.float32)

transformation_matrix = cv2.getPerspectiveTransform(distorted_control_points,
rectified_control_points)

height, width = distorted_drone_image.shape[:2]
```

```
rectified_drone_image = cv2.warpPerspective(distorted_drone_image,
transformation_matrix, (width, height))


rectified_drone_bilinear = cv2.warpPerspective(distorted_drone_image,
transformation_matrix, (width, height), flags=cv2.INTER_LINEAR)
```

Results

Distorted Drone Image

Rectified Drone Image (Orthophoto)

Learnings

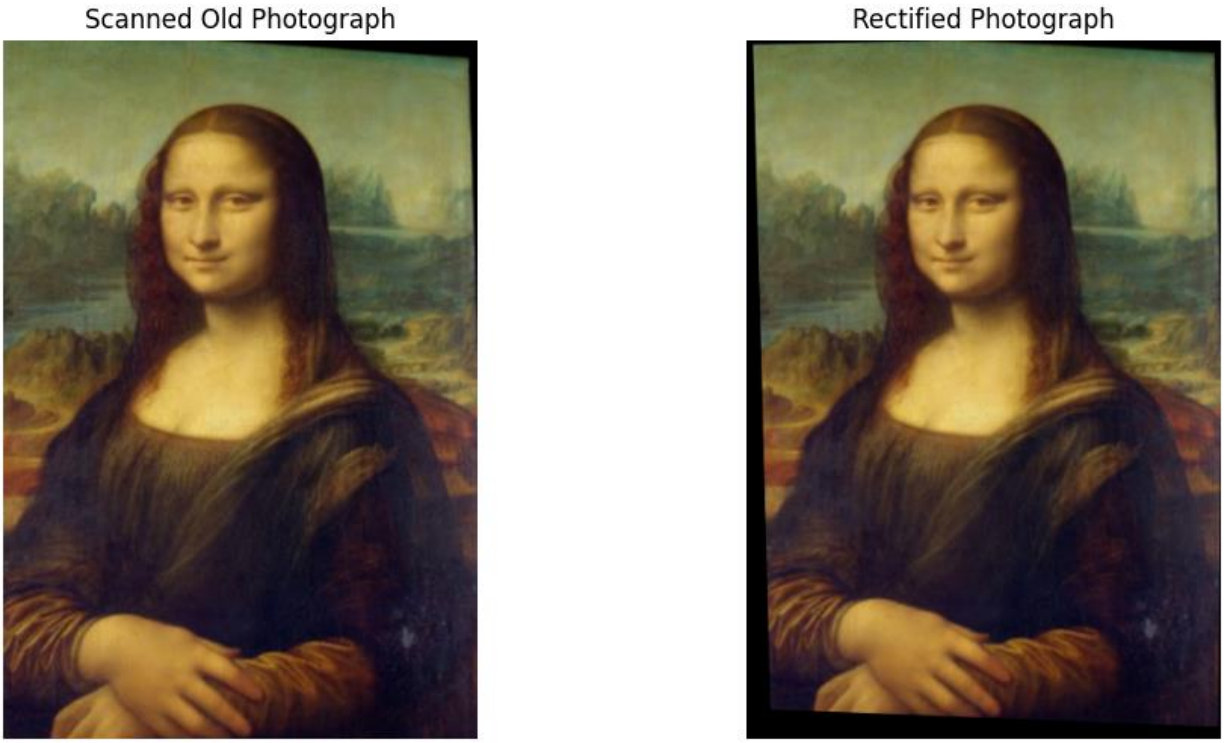
Bilinear interpolation is crucial in ensuring a smooth transition between pixels during rectification, especially in large drone images. The rectified orthophoto provides a more accurate representation of the agricultural field, aiding in better analysis.

Task 4 – Historical Photo Restoration

```
#computing basic statistics

distorted_photo_points = np.array([[100, 200], [380, 280], [560, 480], [730,
680]], dtype=np.float32)


reference_photo_points = np.array([[105, 200], [380, 280], [560, 480], [730,
680]], dtype=np.float32)

transformation_matrix = cv2.getPerspectiveTransform(distorted_photo_points,
reference_photo_points)

height, width = distorted_photo.shape[:2]
```

```
rectified_photo = cv2.warpPerspective(distorted_photo, transformation_matrix,
(width, height))


rectified_photo_bilinear = cv2.warpPerspective(distorted_photo,
transformation_matrix, (width, height), flags=cv2.INTER_LINEAR)
```

Results



Learnings

Bilinear interpolation helps in smoothly resampling pixel values, preserving the quality of the restored photograph. Historical photo restoration involves not just geometric correction but also a lot of careful observation to match the old image with the reference dimensions.

Task 5 – Architectural Image Rectification

```
distorted_key_points = np.array([[5000, 1010], [40, 20], [40, 60], [10, 60]],
dtype=np.float32)
```

```
rectified_key_points = np.array([[5000, 1005], [40, 20], [40, 60], [10, 60]], dtype=np.float32)


transformation_matrix = cv2.getPerspectiveTransform(distorted_key_points, rectified_key_points)

height, width = distorted_architecture.shape[:2]

rectified_architecture = cv2.warpPerspective(distorted_architecture, transformation_matrix, (width, height))


rectified_architecture_bilinear = cv2.warpPerspective(distorted_architecture, transformation_matrix, (width, height), flags=cv2.INTER_LINEAR)
```

Results

Learnings

Bilinear interpolation helps in preserving the architectural details during the rectification process, especially in straight edges. Perspective distortion correction is heavily reliant on the accuracy of the control points and known dimensions of the building.