Name: Drumil Kotecha

BTI Sem-XII C052

Aim: Use SIFT to determine interest points and descriptors

Task1 : Use SIFT to determine location and descriptors of the given image

Task2: : Modify the image

Task3: Match interest points original and modified images

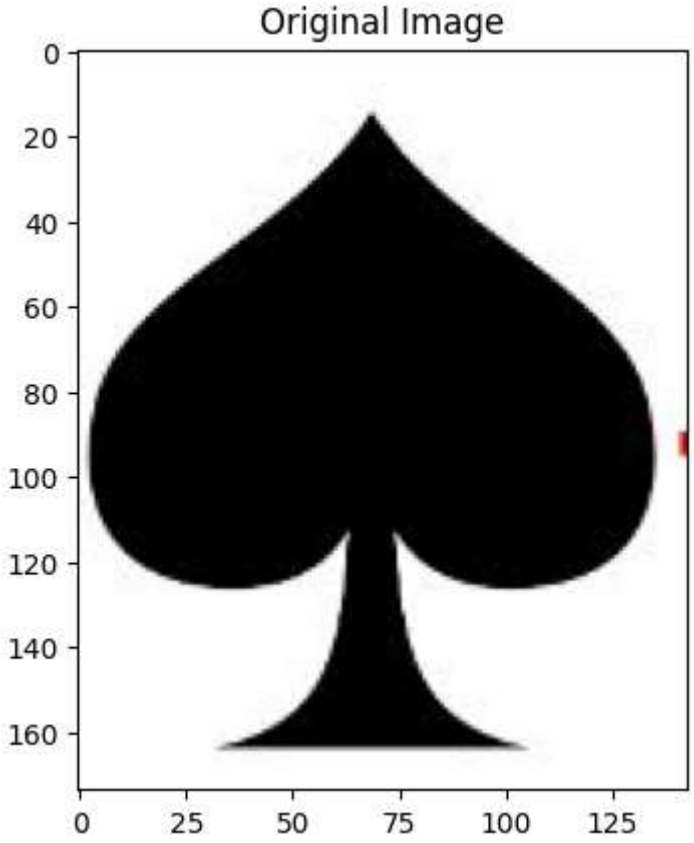
Task4: Observe the difference

In [1]:
```
import numpy as np
import cv2
import matplotlib.pyplot as plt
import imutils
```

In [4]:
```
img1 = cv2.imread('/content/card.JPG')
img1 = cv2.cvtColor(img1, cv2.COLOR_BGR2RGB)
```

In [5]:
```
plt.imshow(img1)
plt.title('Original Image')
```

Out[5]: Text(0.5, 1.0, 'Original Image')



In [6]:
```
img2 = img1.copy()
sift = cv2.SIFT_create(nfeatures=50)
k1, d1 = sift.detectAndCompute(img1, None)
# img2 = cv2.drawKeypoints(img2, k1, None, flags=cv2.DRAW_MATCHES_FLAGS_DRAW_R
ICH_KEYPOINTS)
```

In [7]:
```
len(k1), len(d1)
```

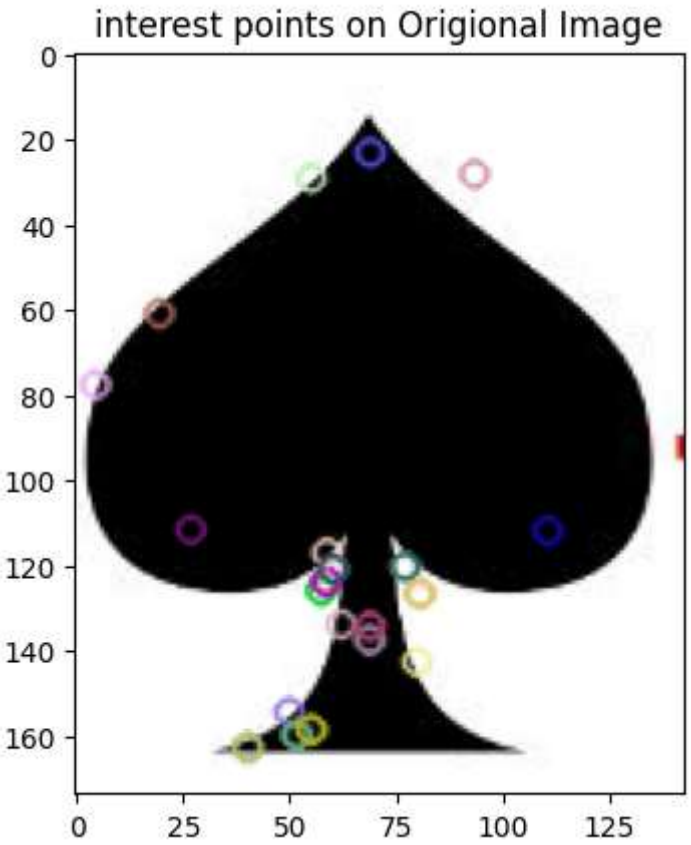
Out[7]: (31, 31)

In [8]:
```
len(d1[0])
```

Out[8]: 128

In [9]:
```
img1_k = cv2.drawKeypoints(img1, k1, None)
plt.imshow(img1_k)
plt.title('interest points on Origional Image')
```

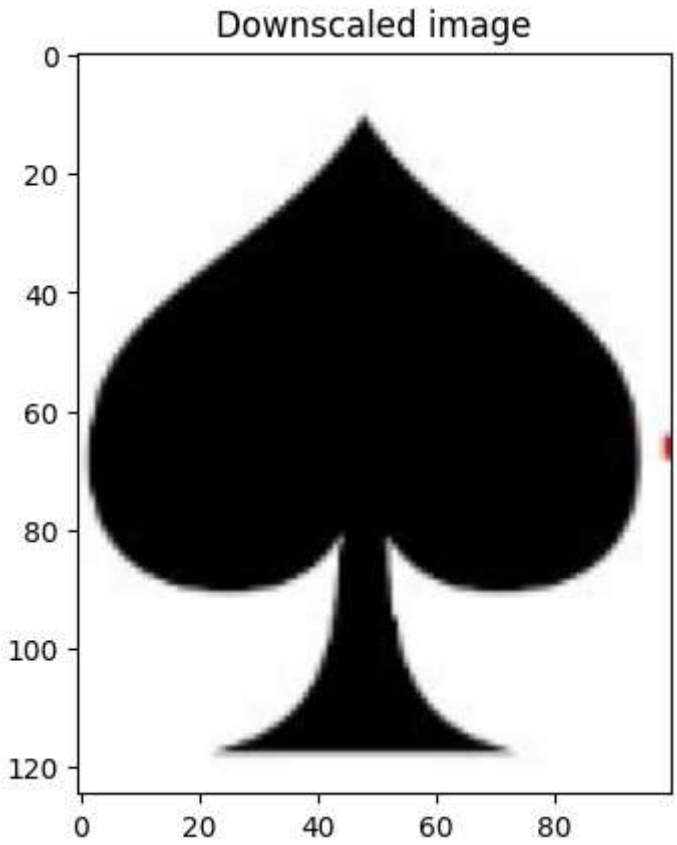
Out[9]: Text(0.5, 1.0, 'interest points on Origional Image')



In [10]:
```
img3 = img1.copy()
img4 = img1.copy()
img5 = img1.copy()
```

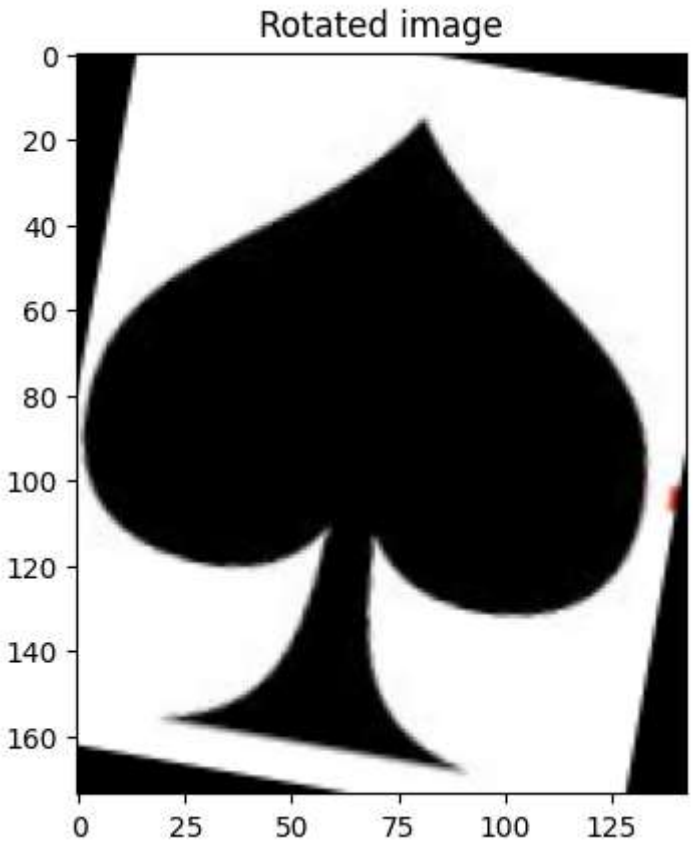
In [11]:
```
img3 = cv2.resize(img3, (100,125))
plt.imshow(img3)
plt.title('Downscaled image')
```

Out[11]: Text(0.5, 1.0, 'Downscaled image')



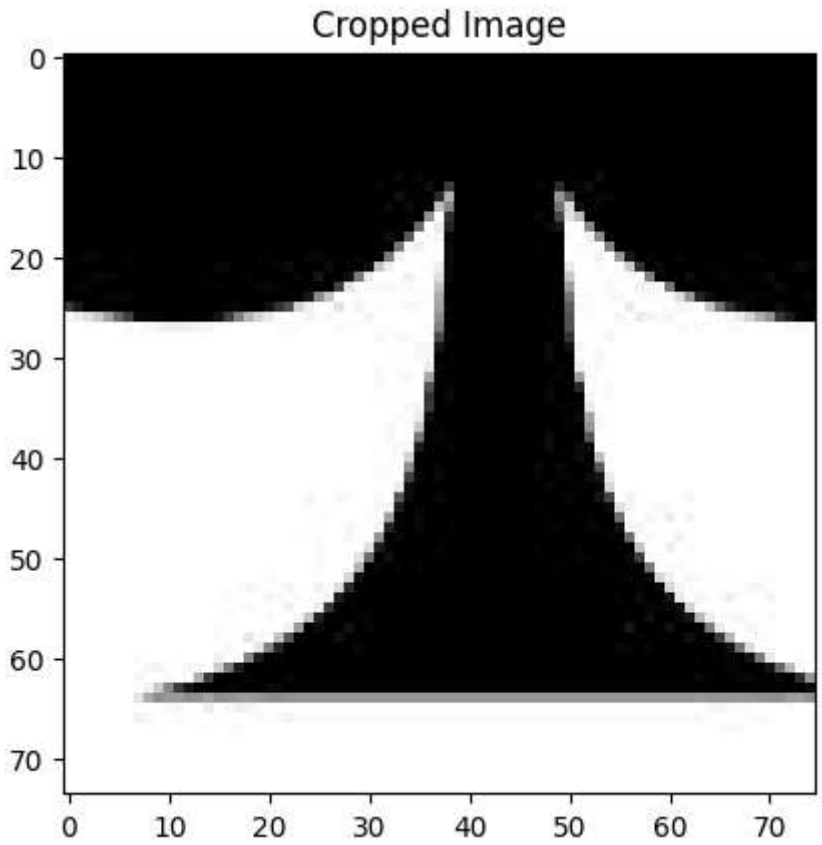
In [12]:
```
img4 = imutils.rotate(img4, -10)
plt.imshow(img4)
plt.title('Rotated image')
```

Out[12]: Text(0.5, 1.0, 'Rotated image')

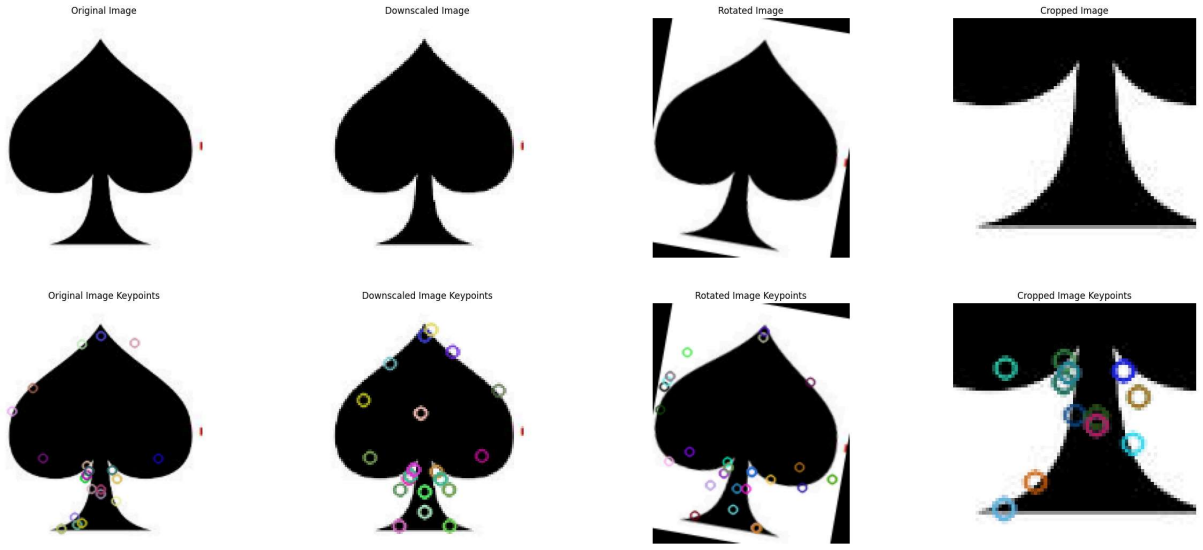


In [13]:
```
img5 = img5[100:, 25:100]
plt.imshow(img5)
plt.title("Cropped Image")
```

Out[13]: Text(0.5, 1.0, 'Cropped Image')



In [17]:
```
k3, d3 = sift.detectAndCompute(img3, None)
img3_k = cv2.drawKeypoints(img3, k3, None)

k4, d4 = sift.detectAndCompute(img4, None)
img4_k = cv2.drawKeypoints(img4, k4, None)

k5, d5 = sift.detectAndCompute(img5, None)
img5_k = cv2.drawKeypoints(img5, k5, None)

# Create a single figure with subplots in one row
fig, axes = plt.subplots(1, 4, figsize=(25, 5))  # 1 row, 6 columns

# Display the images in the subplots
axes[0].imshow(img1)
axes[0].set_title("Original Image")
axes[0].axis('off')

axes[1].imshow(img3)
axes[1].set_title("Downscaled Image")
axes[1].axis('off')

axes[2].imshow(img4)
axes[2].set_title("Rotated Image")
axes[2].axis('off')

axes[3].imshow(img5)
axes[3].set_title("Cropped Image")
axes[3].axis('off')

plt.tight_layout()
plt.show()

fig, axes1 = plt.subplots(1, 4, figsize=(25, 5))  # 1 row, 6 columns

axes1[0].imshow(img1_k)
axes1[0].set_title("Original Image Keypoints")
axes1[0].axis('off')

axes1[1].imshow(img3_k)
axes1[1].set_title("Downscaled Image Keypoints")
axes1[1].axis('off')

axes1[2].imshow(img4_k)
axes1[2].set_title("Rotated Image Keypoints")
axes1[2].axis('off')

axes1[3].imshow(img5_k)
axes1[3].set_title("Cropped Image Keypoints")
axes1[3].axis('off')

plt.tight_layout()
plt.show()
```



In [19]:
```
print(len(d1), len(d3), len(d4), len(d5))
```

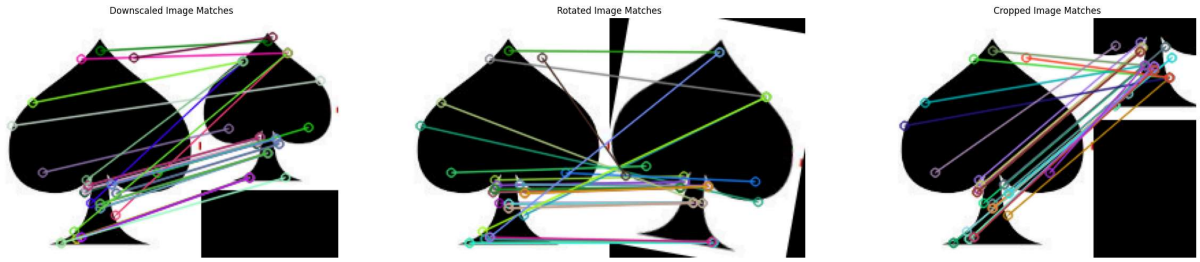
```
31 32 36 19
```

In [25]:
```
bf = cv2.BFMatcher()
matches2 = bf.match(d1, d3)
matches3 = bf.match(d1, d4)
matches4 = bf.match(d1, d5)

matches2 = sorted(matches2, key=lambda x: x.distance)
matches3 = sorted(matches3, key=lambda x: x.distance)
matches4 = sorted(matches4, key=lambda x: x.distance)

print(matches2[0].distance)
print(matches3[0].distance)
print(matches4[0].distance)
```

```
31.670175552368164
36.523963928222656
0.0
```

In [26]:
```
img1_3_matches = cv2.drawMatches(img1, k1, img3, k3, matches2[:50], None, flag
s=2)
img1_4_matches = cv2.drawMatches(img1, k1, img4, k4, matches3[:50], None, flag
s=2)
img1_5_matches = cv2.drawMatches(img1, k1, img5, k5, matches4[:50], None, flag
s=2)
```

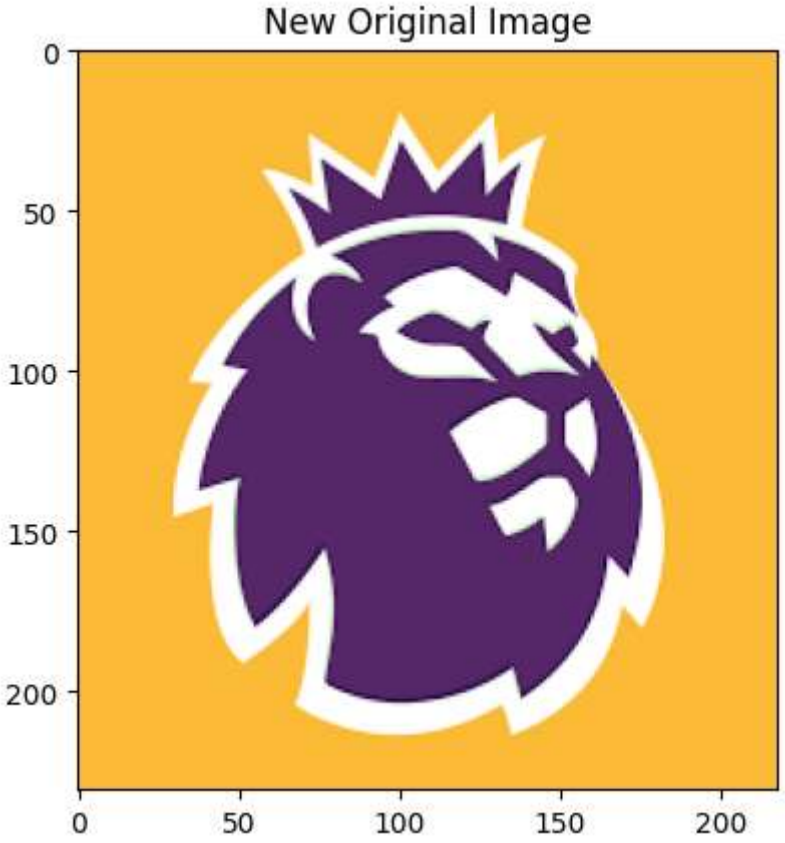
In [27]:
```
# Draw a single plot for all 3 matches
fig, axes = plt.subplots(1, 3, figsize=(25, 5))

axes[0].imshow(img1_3_matches)
axes[0].set_title("Downscaled Image Matches")
axes[0].axis('off')

axes[1].imshow(img1_4_matches)
axes[1].set_title("Rotated Image Matches")
axes[1].axis('off')

axes[2].imshow(img1_5_matches)
axes[2].set_title("Cropped Image Matches")
axes[2].axis('off')

plt.tight_layout()
plt.show()
```



In [36]:
```
new_img1 = cv2.imread('/content/pl1.png')
new_img1 = cv2.cvtColor(new_img1, cv2.COLOR_BGR2RGB)
```

In [37]:
```
plt.imshow(new_img1)
plt.title('New Original Image')
```

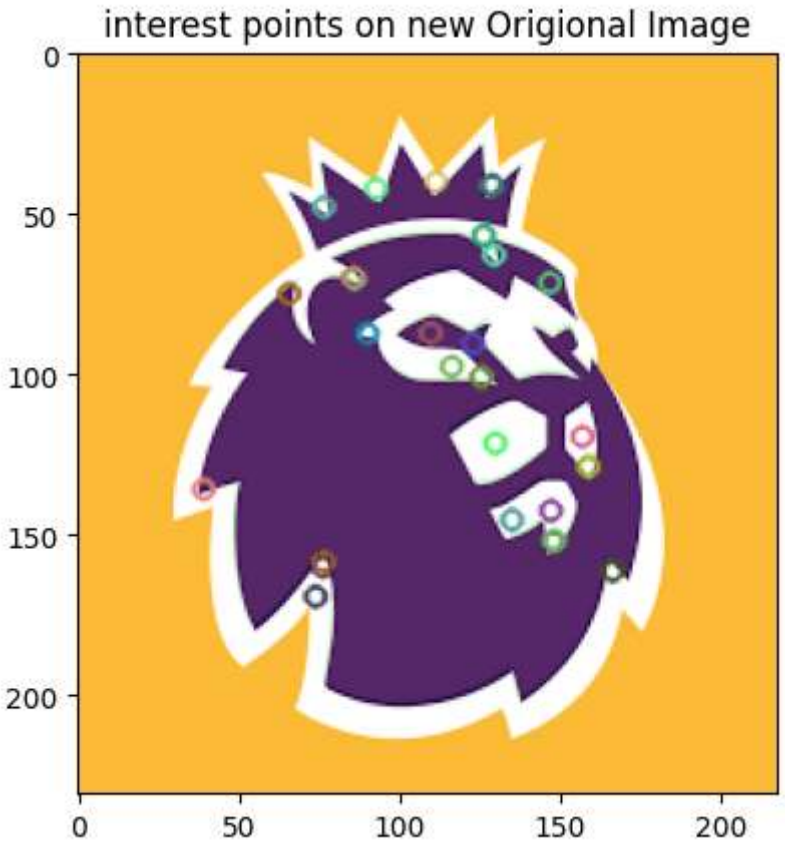
Out[37]: Text(0.5, 1.0, 'New Original Image')



In [39]:
```
new_img2 = new_img1.copy()
sift = cv2.SIFT_create(nfeatures=50)
k1, d1 = sift.detectAndCompute(new_img1, None)
# img2 = cv2.drawKeypoints(img2, k1, None, flags=cv2.DRAW_MATCHES_FLAGS_DRAW_R
ICH_KEYPOINTS)
```

In [40]:
```
new_img1_k = cv2.drawKeypoints(new_img1, k1, None)
plt.imshow(new_img1_k)
plt.title('interest points on new Origional Image')
```

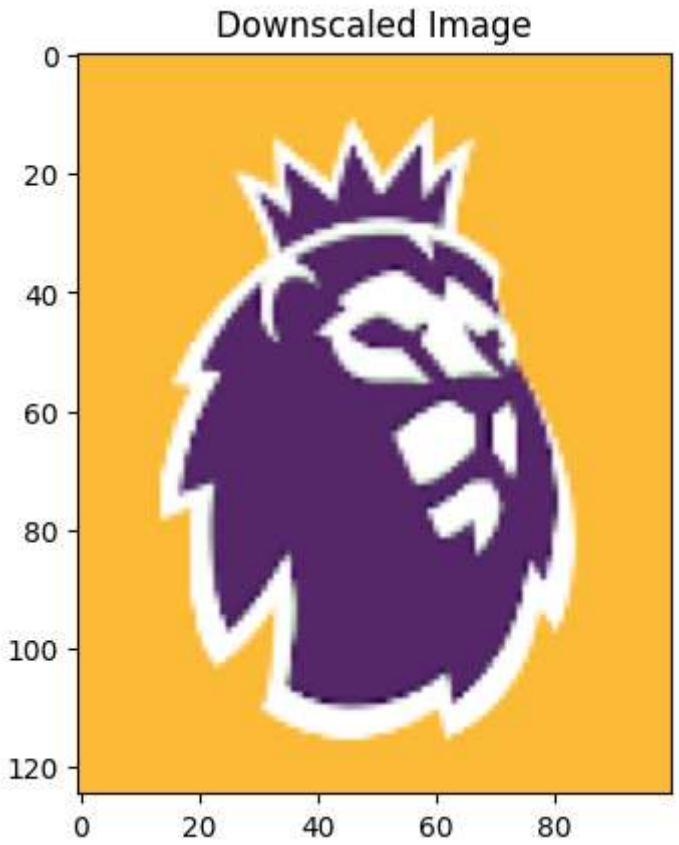
Out[40]: Text(0.5, 1.0, 'interest points on new Origional Image')



In [41]:
```
new_img3 = new_img1.copy()
new_img4 = new_img1.copy()
new_img5 = new_img1.copy()
```

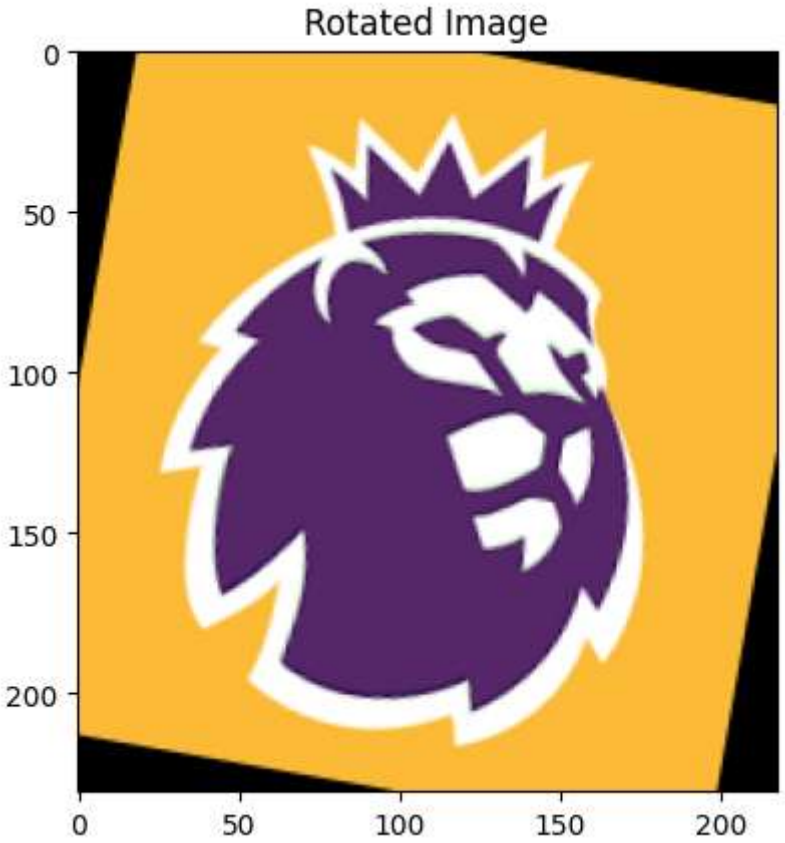
In [42]:
```
new_img3 = cv2.resize(new_img3, (100,125))
plt.imshow(new_img3)
plt.title("Downscaled Image")
```

Out[42]: Text(0.5, 1.0, 'Downscaled Image')



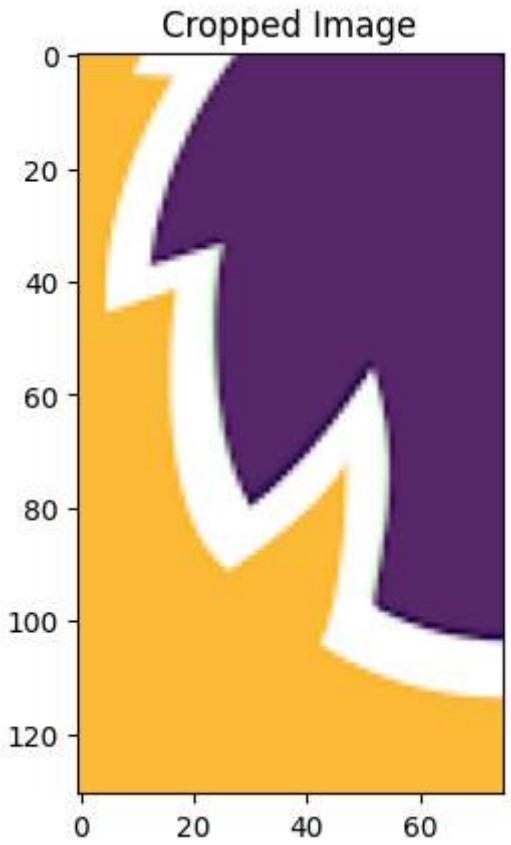
In [43]:
```
new_img4 = imutils.rotate(new_img4, -10)
plt.imshow(new_img4)
plt.title("Rotated Image")
```

Out[43]: Text(0.5, 1.0, 'Rotated Image')



In [44]:
```
new_img5 = new_img5[100:, 25:100]
plt.imshow(new_img5)
plt.title("Cropped Image")
```

Out[44]: Text(0.5, 1.0, 'Cropped Image')



In [46]:
```
k3, d3 = sift.detectAndCompute(new_img3, None)
img3_k = cv2.drawKeypoints(new_img3, k3, None)

k4, d4 = sift.detectAndCompute(new_img4, None)
img4_k = cv2.drawKeypoints(new_img4, k4, None)

k5, d5 = sift.detectAndCompute(new_img5, None)
img5_k = cv2.drawKeypoints(new_img5, k5, None)

# Create a single figure with subplots in one row
fig, axes = plt.subplots(1, 4, figsize=(25, 5))  # 1 row, 6 columns

# Display the images in the subplots
axes[0].imshow(new_img1)
axes[0].set_title("Original Image")
axes[0].axis('off')

axes[1].imshow(new_img3)
axes[1].set_title("Downscaled Image")
axes[1].axis('off')

axes[2].imshow(new_img4)
axes[2].set_title("Rotated Image")
axes[2].axis('off')

axes[3].imshow(new_img5)
axes[3].set_title("Cropped Image")
axes[3].axis('off')

plt.tight_layout()
plt.show()

fig, axes1 = plt.subplots(1, 4, figsize=(25, 5))  # 1 row, 6 columns

axes1[0].imshow(new_img1_k)
axes1[0].set_title("Original Image Keypoints")
axes1[0].axis('off')

axes1[1].imshow(img3_k)
axes1[1].set_title("Downscaled Image Keypoints")
axes1[1].axis('off')

axes1[2].imshow(img4_k)
axes1[2].set_title("Rotated Image Keypoints")
axes1[2].axis('off')

axes1[3].imshow(img5_k)
axes1[3].set_title("Cropped Image Keypoints")
axes1[3].axis('off')

plt.tight_layout()
plt.show()
```

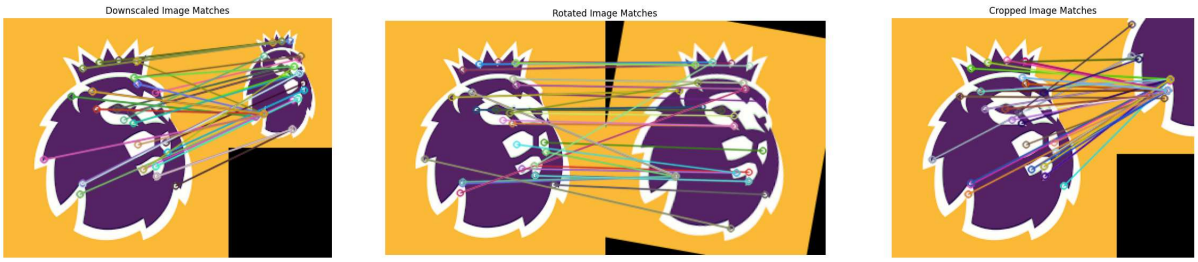
In [47]:
```
print(len(d1), len(d3), len(d4), len(d5))
```

```
50 50 51 20
```

In [48]:
```
bf = cv2.BFMatcher()
matches2 = bf.match(d1, d3)
matches3 = bf.match(d1, d4)
matches4 = bf.match(d1, d5)

matches2 = sorted(matches2, key=lambda x: x.distance)
matches3 = sorted(matches3, key=lambda x: x.distance)
matches4 = sorted(matches4, key=lambda x: x.distance)
```

In [49]:
```
img1_3_matches = cv2.drawMatches(new_img1, k1, new_img3, k3, matches2[:50], No
ne, flags=2)
img1_4_matches = cv2.drawMatches(new_img1, k1, new_img4, k4, matches3[:50], No
ne, flags=2)
img1_5_matches = cv2.drawMatches(new_img1, k1, new_img5, k5, matches4[:50], No
ne, flags=2)
```

In [50]:
```
# Draw a single plot for all 3 matches
fig, axes = plt.subplots(1, 3, figsize=(25, 5))

axes[0].imshow(img1_3_matches)
axes[0].set_title("Downscaled Image Matches")
axes[0].axis('off')

axes[1].imshow(img1_4_matches)
axes[1].set_title("Rotated Image Matches")
axes[1].axis('off')

axes[2].imshow(img1_5_matches)
axes[2].set_title("Cropped Image Matches")
axes[2].axis('off')

plt.tight_layout()
plt.show()
```



Conclusion

SIFT is used to detect the corners and determine their corresponding descriptors. For image 'Card1', the no. of key points is 31. Image is downscaled, rotated by 10 degree and also cropped. It shows that no. of keypoints for resized image = 31, 32, 36, 19. It is observed that some of the key points show mismatch, this is because descriptor these key points belong to similar patches of the image.

The same steps are repeated for the below image. It is found that for no. of features as 25, the key points are matched with the modified images. There are few mismatches as well.