```
import cv2
import numpy as np
import matplotlib.pyplot as plt

def draw_keypoints(img):
    sift = cv2.SIFT_create()
    keypoints = sift.detect(img, None)
    img_with_keypoints = cv2.drawKeypoints(img, keypoints, None, color=(0, 255, 0), flags=0)
    return img_with_keypoints

def show_images(images):
    for i, img in enumerate(images):
        plt.subplot(1, len(images), i + 1)
        plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
        plt.axis('off')
    plt.show()

def compute_homography(img1, img2):
    sift = cv2.SIFT_create()
    kp1, des1 = sift.detectAndCompute(img1, None)
    kp2, des2 = sift.detectAndCompute(img2, None)

    # Match descriptors
    bf = cv2.BFMatcher(cv2.NORM_L2, crossCheck=True)
    matches = bf.match(des1, des2)
    matches = sorted(matches, key=lambda x: x.distance)

    # Draw matches
    img_matches = cv2.drawMatches(img1, kp1, img2, kp2, matches[:10], None,
flags=cv2.DrawMatchesFlags_NOT_DRAW_SINGLE_POINTS)

    # Extract location of good matches
    pts1 = np.float32([kp1[m.queryIdx].pt for m in matches]).reshape(-1, 1, 2)
    pts2 = np.float32([kp2[m.trainIdx].pt for m in matches]).reshape(-1, 1, 2)

    # Compute homography
    H, mask = cv2.findHomography(pts1, pts2, cv2.RANSAC, 5.0)
    matches_mask = mask.ravel().tolist()

    # Draw inliers
    draw_params = dict(matchColor=(0, 255, 0), singlePointColor=None, matchesMask=matches_mask, flags=2)
    img_inliers = cv2.drawMatches(img1, kp1, img2, kp2, matches[:10], None, **draw_params)

    return H, img_matches, img_inliers

# Load the images
img1 = cv2.imread('/root/turtlebot3_ws/src/turtlebot3_simulations/cam1.png')
img2 = cv2.imread('/root/turtlebot3_ws/src/turtlebot3_simulations/cam2.png')
img3 = cv2.imread('/root/turtlebot3_ws/src/turtlebot3_simulations/cam3.png')
img4 = cv2.imread('/root/turtlebot3_ws/src/turtlebot3_simulations/cam4.png')
```

```python
# Check if images are loaded correctly
if img1 is None or img2 is None or img3 is None or img4 is None:
    print("Error loading images")
    exit()

# Visualize keypoints
img1_kp = draw_keypoints(img1)
img2_kp = draw_keypoints(img2)
img3_kp = draw_keypoints(img3)
img4_kp = draw_keypoints(img4)

# Show keypoints
show_images([img1_kp, img2_kp, img3_kp, img4_kp])

# Compute and show homography between img1 and img2
H, img_matches, img_inliers = compute_homography(img1, img2)

# Show matches and inliers
plt.figure(figsize=(20, 10))
plt.subplot(1, 2, 1)
plt.title('Feature Matches')
plt.imshow(cv2.cvtColor(img_matches, cv2.COLOR_BGR2RGB))

plt.subplot(1, 2, 2)
plt.title('Inliers')
plt.imshow(cv2.cvtColor(img_inliers, cv2.COLOR_BGR2RGB))
plt.show()

print("Homography Matrix:\n", H)

# Stitch the images
stitched_image = stitch_images([img1, img2, img3, img4])
if stitched_image is not None:
    # Save the combined image
    cv2.imwrite('stitched_image.jpg', stitched_image)

    # Display the combined image
    cv2.imshow('Stitched Image', stitched_image)
    cv2.waitKey(0)
    cv2.destroyAllWindows()
else:
    print("Image stitching failed.")
```