CV TA - 2

Name : Ketan Pardhi ***Batch : B3*** ***Roll no. : 41***

1. SIFT

Original Images:

Image 1

Image 2

Detect KeyPoints

```
sift = cv2.SIFT_create()

# Detect keypoints and compute descriptors
keypoints1, descriptors1 = sift.detectAndCompute(img1_gray, None)
keypoints2, descriptors2 = sift.detectAndCompute(img2_gray, None)
```

```
# Draw keypoints on images
img1_kp = cv2.drawKeypoints(img1_gray, keypoints1, None, color=(0, 255, 0),
                            flags=cv2.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)
img2_kp = cv2.drawKeypoints(img2_gray, keypoints2, None, color=(0, 255, 0),
                            flags=cv2.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)

# Convert to RGB for matplotlib
img1_kp = cv2.cvtColor(img1_kp, cv2.COLOR_BGR2RGB)
img2_kp = cv2.cvtColor(img2_kp, cv2.COLOR_BGR2RGB)
```

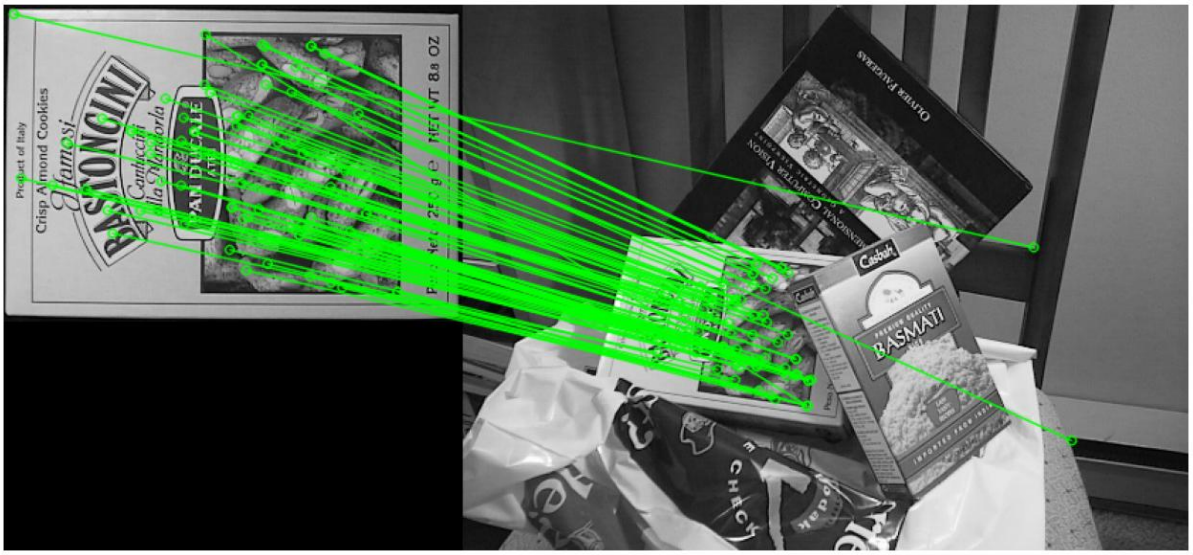
SIFT Keypoints - Image 1

SIFT Keypoints - Image 2

Match KeyPoints:

```python
# FLANN parameters
FLANN_INDEX_KDTREE = 1
index_params = dict(algorithm=FLANN_INDEX_KDTREE, trees=5)
search_params = dict(checks=50)

flann = cv2.FlannBasedMatcher(index_params, search_params)

# Convert descriptors to the appropriate format if necessary
if descriptors1 is not None and descriptors2 is not None and len(descriptors1) > 0 and len(descriptors2) > 0:
    descriptors1 = np.float32(descriptors1)
    descriptors2 = np.float32(descriptors2)

    # Find k best matches for each descriptor (k=2)
    matches = flann.knnMatch(descriptors1, descriptors2, k=2)

    # Apply Lowe's ratio test
    good_matches = []
    ratio_threshold = 0.75
    for m, n in matches:
        if m.distance < ratio_threshold * n.distance:
            good_matches.append(m)
```

SIFT Matches



2. *Grab Cut:*

```python
# Define the initial mask and rectangle for GrabCut
mask = np.zeros(image.shape[:2], np.uint8)
rect = (50, 50, image.shape[1] - 100, image.shape[0] - 100)

# Allocate memory for models
bgd_model = np.zeros((1, 65), np.float64)
fgd_model = np.zeros((1, 65), np.float64)

# Apply GrabCut
cv2.grabCut(image, mask, rect, bgd_model, fgd_model, 5, cv2.GC_INIT_WITH_RECT)

# Modify the mask to extract the foreground
mask2 = np.where((mask == 2) | (mask == 0), 0, 1).astype('uint8')
segmented_image = image * mask2[:, :, np.newaxis]
```

Original (img1)

Foreground (img_rgb)

3. *Shi-Tomasi*

```python
# Load the image
img = cv2.imread('image.png')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# Detect corners
corners = cv2.goodFeaturesToTrack(gray, maxCorners=100, qualityLevel=0.01, minDistance=10)
corners = corners.astype(np.int32)

# Draw the corners
for corner in corners:
    x, y = corner.ravel()
    cv2.circle(img, (x, y), 4, (0, 255, 0), -1)

# Convert BGR to RGB for matplotlib
img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
```

Shi-Tomasi Corner Detection