ECE501_2025_12_Group_2

Project name: **Image Tampering Detection (Forgery Localization)**

Team Details:

Name	**AU ID**
Prasham Mehta	AU2340135
Dhairya Rupani	AU2340195
Vidhan Nahar	AU2340199
Drumil Bhati	AU2340211

Work done till week 4:

- We implemented a feature-matching algorithm in order to try and test how a prototype might look like.
- Our current model for the prototype includes a feature-matching algorithm that extracts features from smaller regions of the image in order to match them in the image.
- We used a brute-force matcher from the openCV library in the prototype, since efficiency is not our current requirement. We need to be able to highlight the correctness of our approach first and then improve our model further. The algorithm used by BFMatcher is k-nearest neighbour algorithm.

```python
import cv2

bf = cv2.BFMatcher()
matches = bf.knnMatch(descriptors, descriptors, 2)
```

- We then extract the good matches in a list.

Python

```
good_matches = []
for m, n in matches:
    if m.distance < 0.75 * n.distance:
        good_matches.append(m)

print(f"Found {len(good_matches)} good matches.")
```

- Then analyse the matched features to identify any potential forged areas.
- Extract keypoint coordinates for good matches, calculate the displacement vectors, group matched by similar displacement and filter groups with sufficient matches.

Python

```
import numpy as np
from sklearn.cluster import DBSCAN

# Extract coordinates of matched keypoints
src_pts = np.float32([keypoints[m.queryIdx].pt for m in
good_matches]).reshape(-1, 2)
dst_pts = np.float32([keypoints[m.trainIdx].pt for m in
good_matches]).reshape(-1, 2)

# Calculate displacement vectors
displacement_vectors = dst_pts - src_pts

# Group matches based on similar displacement vectors using DBSCAN
# Adjust eps and min_samples based on image resolution and expected forgery
characteristics
clustering = DBSCAN(eps=5, min_samples=5).fit(displacement_vectors)
labels = clustering.labels_

# Filter groups with significant number of matches
unique_labels, counts = np.unique(labels, return_counts=True)
potential_forged_regions = {}
for label, count in zip(unique_labels, counts):
    if label != -1 and count > 5:  # Exclude noise (-1 label) and small groups
        potential_forged_regions[label] = [good_matches[i] for i, l in
enumerate(labels) if l == label]

print(f"Found {len(potential_forged_regions)} potential forged regions.")
```

- Annotate the image with identified forged regions using matplotlib library.

```python
import numpy as np

# Create a copy of the original image to draw on
image_with_highlights = image.copy()

# Iterate through the identified potential forged regions
for region_label, matches_in_region in potential_forged_regions.items():
    # Draw circles around the keypoints of the matched pairs
    for match in matches_in_region:
        # Get the coordinates of the keypoints
        img1_idx = match.queryIdx
        img2_idx = match.trainIdx
        (x1, y1) = keypoints[img1_idx].pt
        (x2, y2) = keypoints[img2_idx].pt

        # Draw circles around the keypoints
        cv2.circle(image_with_highlights, (int(x1), int(y1)), 5, (0, 255, 0),
2) # Green circle for the first keypoint
        cv2.circle(image_with_highlights, (int(x2), int(y2)), 5, (0, 0, 255),
2) # Red circle for the second keypoint

# Display the resulting image with the highlighted regions
from matplotlib import pyplot as plt

plt.figure(figsize=(10, 10))
plt.imshow(cv2.cvtColor(image_with_highlights, cv2.COLOR_BGR2RGB))
plt.title('Image with Potential Forged Regions Highlighted')
plt.axis('off')
plt.show()
```

Original Image:

Forged Image:

Challenges faced:

- False Matches: The similarity in textures.
    - In high-texture images (e.g., grass, sky, water), incorrect key point matches are made, which lowers the accuracy of localising forgeries.

- Image Transformation Sensitivity.
    - The algorithm tends to miss matches when there are minor transformations in the copied area (e.g., rotation, scaling, or changes in illumination).

- Copy-move forgeries can include post-processing techniques such as blurring, compression, or noise injection. The existing algorithm does not perform well in such circumstances.

- Weakness in Ground Truth Localization.
    - Accurate forged region boundaries are not usually accessible, and thus, it is hard to measure performance quantitatively.

Plan for next phase:

- We aim to adopt a block approach in the future, where the image is cut into small overlapping blocks, and features are extracted from each block to identify similar areas. This will help in the detection of even the forgeries that do not possess good keypoints.

- To ensure that the detection is more stable and also less sensitive to changes in rotation or brightness.

- To enhance the accuracy of the algorithm, we propose using adaptive block size selection based on the image texture's resolution and complexity, as this allows for the correct identification of both large and small copied areas.

- We will also incorporate block-matching optimisation using efficient sorting or hashing methods, which will reduce the time and computational intensity required for block comparisons.

- We also plan to combine block-based and keypoint-based techniques in the future by leveraging the strengths of both, which include the accuracy of keypoints and the ability to cover large areas with blocks, thereby enhancing the detection of copy-move forgery.