Non-Overlapping LSB Embedding Using SIFT Keypoints and randomised Colour Channels

COM31006 Assignment Report

Aryan Golbaghi

Implementation code: `https://github.com/youngaryan/stif_steganography-`

Demonstration video: `https://youtu.be/EZfF1awYP58`

Word Count: 1670

Contents

1 Introduction **3**

2 SIFT Keypoint Detection **3**
2.1 Selecting the N Strongest Non-Overlapping Keypoints . . . 3

3 Watermark Embedding **4**
3.1 Pre–processing pipeline . . . 4
3.2 Keypoint-bit embedding . . . 4
3.3 Key-point correspondence during verification . . . 4

4 Watermark Recovery & Authenticity Verification **5**
4.1 Descriptor matching . . . 5
4.1.1 FLANN . . . 5
4.2 Geometric Consistency Verification . . . 5
4.3 Bit-pattern Consistency Verification . . . 5
4.4 Majority-vote watermark recovery . . . 5
4.5 Quantitative evaluation . . . 6

5 Tampering Detection Analysis **6**

6 Innovation **6**

7 Limitations, Challenges, and Future Work **7**
7.1 Current Limitations . . . 7
7.2 Explored but Abandoned Variant . . . 7
7.3 Future Work . . . 7

8 Conclusion **8**

1 Introduction

This project proposes a simple yet robust image steganography system that embeds a fixed binary watermark segment into the least significant bits of randomly chosen colour channels at non-overlapping SIFT keypoints. The coordinates, descriptors, and selected channels for each embedded patch are stored in a metadata file. During authentication, SIFT keypoints are re-detected, matched with stored descriptors using Fast Library for Approximate Nearest Neighbors (FLANN), and filtered via Lowe's ratio test. A watermark is considered authentic only if the total bit-error mismatches remain below 10% of the matched patches, and the inlier ratio of a Random Sample Consensus (RANSAC) estimated homography greater than 60%. A detection module highlights tampered regions by overlaying red circles on mismatches. A simple GUI allows users to embed, verify, detect, or recover watermarks in a few clicks. The approach is visually imperceptible, easy to implement and achieved 100% accuracy on the manipulated images.

2 SIFT Keypoint Detection

The embedding process starts by finding stable reference points in the image using the Scale-Invariant Feature Transform (SIFT). SIFT is used because it detects keypoints that remain consistent even when the image is rotated, resized, or slightly changed in lighting or viewpoint. This behavior ensures that the same keypoints can be found again later in a suspect image, which helps to recover the watermark and checking for tampering.

2.1 Selecting the N Strongest Non-Overlapping Keypoints

OpenCv's SIFT implementation usually detects thousands of keypoints, often grouped in very textured areas. The system therefore keeps only the top N response keypoints that do not overlap. Embedding a watermark all of these would waste space and could cause visible overlap between patches.

Algorithm 1 SelectNonOverlappingKPs(gray_carrier, N: desired number of points, half_seg = SEG_SIZE//2)

```
kps ← SIFT(gray_carrier)                                   ▷ All candidate keypoints
occupancy ← zeros(size(gray_carrier))                      ▷ Occupancy grid
sel ← empty list
for all kp in sort_descending(kps, kp.response) do         ▷ checks the stronger keypoints first
    (x, y) ← round(kp.pt)
    if the half_seg window around kp lies inside gray_carrier and occupancy is FALSE then
        mark occupancy[patch] ← True
        append kp to sel
        if sel.size == N then
            break
        end if
    end if
end for
return sel
```

How did you manage overlapping watermarks associated with nearby keypoints? Lines 4 to 13 ensure that, once a keypoint is accepted, an occupancy mask blocks out its surrounding watermark footprint; therefore, keypoints falling inside that region are ignored. By setting the mask radius to half the watermark size, the method ensures that no two patches touch. This eliminates the checkerboard patterns(local overwriting of LSBs) caused when adjacent patches share pixels. With $N = 200$ on $1080p$ images, which have $2,073,600$ pixels, A 9x9 patch is only 81 pixels, and placing 200 of them would use $200\text{x}81 = 16,200$ pixels. This is 0.78% of the entire image, this makes watermarks invisible while allowing robust authentication.

3 Watermark Embedding

3.1 Pre–processing pipeline

Given a carrier image I and a binary watermark W, the `Embedder` first *reads* the carrier in colour, then converts it to grayscale for keypoint detection. The watermark is loaded in grayscale, binarised (converts to black and white), and finally resized to a square `SEG_SIZE` $\times$ `SEG_SIZE` using nearest neighbour interpolation to preserve bit pattern integrity. This ensures that (i) every keypoint receives an identical $s \times s$ binary segment, and (ii) the watermark values already are either 0 or 1, simplifying the LSB overwrite step.

3.2 Keypoint-bit embedding

For each of the N strongest, non-overlapping SIFT anchors returned by Algorithm 1 (Section 2), the algorithm:

1. draws a *random* colour channel $c \sim \{B, G, R\}$ to spread the watermark across all channels and reduce vulnerability to single-channel filtering, this will further help to keeping the watermark invisible by eye.
2. overwrites the least significant bit of each pixel in the $s \times s$ window centered at the keypoint:

$$I'_{y+\Delta y,\, x+\Delta x,\, c} \;\leftarrow\; (I_{y+\Delta y,\, x+\Delta x,\, c}\,\&\,\sim 1) \;\mid\; W_{\Delta y+h,\, \Delta x+h},$$

where $-h \le \Delta x, \Delta y \le h$ and $h = \lfloor s/2 \rfloor$

Only one bit per pixel is modified. Since at most $\frac{N\,s^2}{|I|}$ pixels are affected (e.g. about 0.78% for $N = 200$, $s = 9$ on a 1080p image), the watermark remains imperceptible while introducing redundancy for robust recovery.

3.3 Key-point correspondence during verification

For each accepted SIFT keypoint, the metadata stores a tuple containing its (x, y) coordinates, the chosen colour channel (R, G, B), and its 128-dimensional SIFT descriptor. During verification, SIFT is rerun on the suspected image, and descriptors are compared with the stored ones using a FLANN-based matcher, followed by Lowe's ratio test to filter non-certain and ambiguous matches. A RANSAC homography is then estimated to reject outlier matches. Matching based on descriptor similarity rather than just spatial proximity recovers more matches, enabling reliable watermark reconstruction and robust tamper detection. (see Section 4). The metadata includes:

- keypoints assosited values (see Subsection:3.3);
- the (odd) segment size s;
- the binarised segment pattern.

This information is saved in `meta.json` so that the extraction can later locate and decode each bit segment.

4 Watermark Recovery & Authenticity Verification

4.1 Descriptor matching

The verification step starts by re-running SIFT on the *suspect* image and computing 128-D descriptors. The stored reference descriptors are then matched with a **FLANN** Kd-tree index; a pair is accepted if the best match distance is less than 0.7 times the second-best(Lowe's ratio test). This generates a one-to-one correspondence list M between reference and suspect keypoints.

4.1.1 FLANN

FLANN is used to efficiently match high-dimensional data such as SIFT descriptors. It is significantly faster than brute-force matching, especially when there are potentially thousands of keypoints.

During verification, for each reference descriptor, FLANN retrieves the two closest matches in the suspect set. Lowe's ratio test is then applied to reject ambiguous matches. This step is important to filter out false positives and ensure robust geometric consistency (see Subsection 4.2).

4.2 Geometric Consistency Verification

After matching descriptors (SIFT + FLANN + Lowe's ratio test), we need to verify that the image hasn't been warped, rotated, cropped, or tampered with geometrically.

We do that by checking if the matched keypoints align via a global geometric transformation (homography). If most matched keypoints can be transformed from the reference to the suspect image with one single homography, the image is considered geometrically consistent.

To do this the matched coordinates are fed into RANSAC homography estimation. A point is considered an *inlier* if it lies within the estimated plane; the image is accepted only when the inlier ratio $\gamma = \frac{|\text{inliers}|}{|M|}$ greater than 0.6. This protects against false matches and ensures that the global structure of the image is preserved (robust to temperaments).

4.3 Bit-pattern Consistency Verification

Around every retrieved match the verifier re-extracts the $s \times s$ patch in the *same colour channel* recorded during embedding and reads the least significant bit. The *bit-error rate* (BER) is calculated as $\mathrm{BER} = \frac{1}{s^2}|P\ XOR\ S|$, where S is the original segment and P is the recovered patch. A patch is marked as *mismatching* when $\mathrm{BER} > 0.1$; the image is considered *authentic* only if the total number of mismatches stratifies $(0.1 > \mid M|)$. Mismatching coordinates are returned and later will be used by the detector.

4.4 Majority-vote watermark recovery

When the authenticity test passes, we stack all the matched patches and perform a pixel-wise majority vote to reconstruct the original binary watermark. Small segments ($s < 20$) are upscaled ($\times 3$) with nearest neighbour interpolation for easier visual inspection.

4.5 Quantitative evaluation

Table 1 summerised the experiments on 6 carrier images and two different watermarks, each subjected to different manipulation attacks such as rotation, cropping, gaussian noise, gaussian blur, scaling, and filter colour channel.

Table 1: Verification accuracy under different attacks (N=200, s=9).

Attack	Matched Patches (%)	Inlier Ratio (%)	Authenticated	Detected Keypoints (out of 200)
None	100	100	Yes	196
Crop (remain 80%)	90.09	100	Yes	132
Crop (remain 15%)	92	97	Yes	92
Vertical Crop (remain 60%)	0	N/A	No	105
Horizontal Crop (remain 60%)	94	100	Yes	145
Rotate 1°	0	N/A	No	143
Rotate 15°	0	N/A	No	133
Rotate 90°	0	N/A	No	121
Scale 2×	0	N/A	No	146
Scale 1.2×	0	N/A	No	141
Scale 0.8×	0	N/A	No	70
Gaussian noise (mean=0, std=15)	0	N/A	No	140
Gaussian noise (mean=10, std=35)	0	N/A	No	77
Gaussian noise (mean=0, std=2)	0	N/A	No	189
Gaussian blur (ksize=(1,1), sigmax=0)	100	100	Yes	196
Gaussian blur (ksize=(5,5), sigmax=0)	1.53	96.9	No	65
Gaussian blur (ksize=(15,15), sigmax=0)	0	N/A	No	32
Filter out 1 colour channel (random)	0.36	N/A	No	169
Filter out 2 colour channels (random)	N/A	N/A	No	0
Filter out 1 different colour channel per pixel (random)	0	94.2	No	51

Summary of Verification Results The system performs well when no attack is applied, achieving 100% matched patches, a 100% inlier ratio, and successful authentication. Cropping preserves enough features to maintain high verification accuracy; however, vertical cropping breaks geometric consistency and fails for authenticity. Rotations and scaling transformations result in 0% matched patches and verification failure, showing the system is sensitive to global geometric alterations. The system is highly vulnerable to Gaussian noise and blur, even slight noise or moderate blur reduces matched patches, with most attacks leading to 0% successful matches. However, a minimal blur (kernel size 1) has a small impact. Most attacks that filter out colour channels cause the system to fail entirely, underscoring its reliance on intact multi-channel pixel values for recovery.

5 Tampering Detection Analysis

The detector first invokes the verifier; if verification failed, it draws red circles around the keypoints that expected to see watermakes, and saves the overlaid image.

6 Innovation

My system departs from the basic implementation in four ways that together form an innovative pipeline:

1. **Use the top N keypoint** To maximise recovery and improve efficiency and speed, we embed only the top N keypoints with the highest response score; these points are usually more stable and detectable than

the lower ranked points; this also helps the watermark stay less noticeable in the watermarked image, as we are using fewer keypoints.

2. **Non-overlapping keypoint selection** Instead of selecting the top-N SIFT responses, I apply an occupancy mask to greedily accept only those keypoints whose $segment\_size \times segment\_size$ neighborhood is unoccupied. This prevents spatial collisions between watermark patches, avoids mutual corruption during embedding, and results in cleaner, more imperceptible modifications compared to overlapping patches.

3. **Random-channel LSB embedding.** Each keypoint's (x, y) coordinate alternation is saved into a deterministic choice of R, G, or B channel. This ensures that only about third of the pixels in each channel are altered, significantly reducing the risk of detection. It also avoids patterns (e.g., chessboard artefacts) often caused by the same channel embedding.

4. **Homography-guided verification.** In addition to the bit-error threshold, I employ a geometric consistency check via RANSAC-based homography estimation. The image is only accepted as authentic if the inlier ratio $\gamma \geq 0.6$. This improves reliability under viewpoint changes.

7 Limitations, Challenges, and Future Work

7.1 Current Limitations

- **Lossy compression.** The prototype is reliable only on lossless carriers (PNG, TIFF). A single JPEG re-save destroys most of the of LSB bits, causing verification to fail.
- **Texture dependence.** Even though I have addressed this issue by limiting the number of keypoints to use; however, the system assumes a moderately textured carrier; strongly flat images produce fewer stable SIFT points, reducing capacity and robustness.
- **Basic GUI.** The front-end offers minimal options, limiting usability for users.

7.2 Explored but Abandoned Variant

I experimented with splitting the watermark into 5×5 sub blocks and embedding each at a different keypoint, then reconstructing the entire mark at verification. On highly textured images, this shows potential, but overall bit-error increased significantly and repeatability fails to verify correct images, especially for larger watermarks; this approach also required more stable keypoints, hence the approach was discarded for the final submission.

7.3 Future Work

1. **Learning-based optimisation** Train a lightweight steganographic auto-encoder to adaptively choose embedding locations, producing higher capacity and lower detectability.

2. **Adaptive keypoint choosing** Rank candidates by repeatability score and embed only in the top N_{stable} points, and choose a subset of keypoints from them that are more promising.

3. **Enhanced interface** Re-implement the GUI with drag-and-drop, and detailed tamper heatmap overlays for inspection.

8 Conclusion

By anchoring a binary watermark to top ranked non-overlapping SIFT keypoints, embedding its bits in randomly chosen LSB colour channels, and verifying authenticity with a homography-aware and bit-error threshold, our system is an inexpensive steganographic solution that correctly authenticates watermarked images.