Image-to-Image Steganography for Watermark Creation

Submission Deadline: Wednesday, 21 May 2025, at 3 PM.

1 Introduction

Steganography is the technique of hiding information inside another medium so that it goes unnoticed. Unlike cryptography, which scrambles a message to make it unreadable, steganography makes the message imperceptible. In **image-to-image steganography**, a secret image is embedded within a carrier image without noticeably altering its appearance. This allows information to be discreetly hidden while ensuring the carrier image looks unchanged. The hidden data can later be extracted using the appropriate method. One important application of steganography in the industry is watermarking, which helps ensure the authenticity and originality of images and videos.

In this assignment, your **objective** is to create your own watermark for digital images by exploiting keypoints detected with SIFT. The watermark should be inserted by altering the least significant bits (LSB) of the image pixels, which is a simple steganography strategy (See Appendix A).

Upon completion of this activity, you will be able to employ fundamental image processing techniques, feature extraction, and other computer vision methods learned in this module to address a real-world problem.

2 Descriptions and Requirements

In this assignment, you will implement image-to-image steganography to embed a watermark (image) within a cover image by leveraging key point detection using the SIFT algorithm (**Watermark Embedder**). Additionally, you will need to implement a tool to verify the authenticity of an image (**Watermark Recovery**) and, finally, a tool to check for any forgery based on the presence of watermarks (**Tampering Detector**).

Below are the standard requirements for a basic implementation, where the same watermark is applied to all key points detected using SIFT (see Fig. 1). However, you are encouraged to explore creative approaches for more compelling results. Innovation is a key marking criterion, so deviations from the basic approach are welcomed, as long as you justify your design choices in the report questionnaire.

You must follow these steps to create your **Watermark Embedder**:

1. **Input images**: Open an image to serve as the carrier for your watermark. This image should have enough complexity to hide the watermark without visual distortion. Choose a smaller image to be hidden in the cover image.

2. **Preprocessing**: Ensure both images are in the same format (preferably .png or .tif). Convert the carrier image to grayscale for easier SIFT point detection.

3. **Keypoint Detection with SIFT**: Apply the SIFT algorithm to detect keypoints in the cover image (Carrier Image). Extract the coordinates and feature descriptors of N keypoints to serve as the locations for embedding the watermark.

4. **Watermark Encoding and Embedding**: Prepare the watermark image for embedding. Depending on its size and the number of keypoints, adjust the watermark accordingly. Embed the watermark by subtly modifying the pixel values at the detected keypoints in

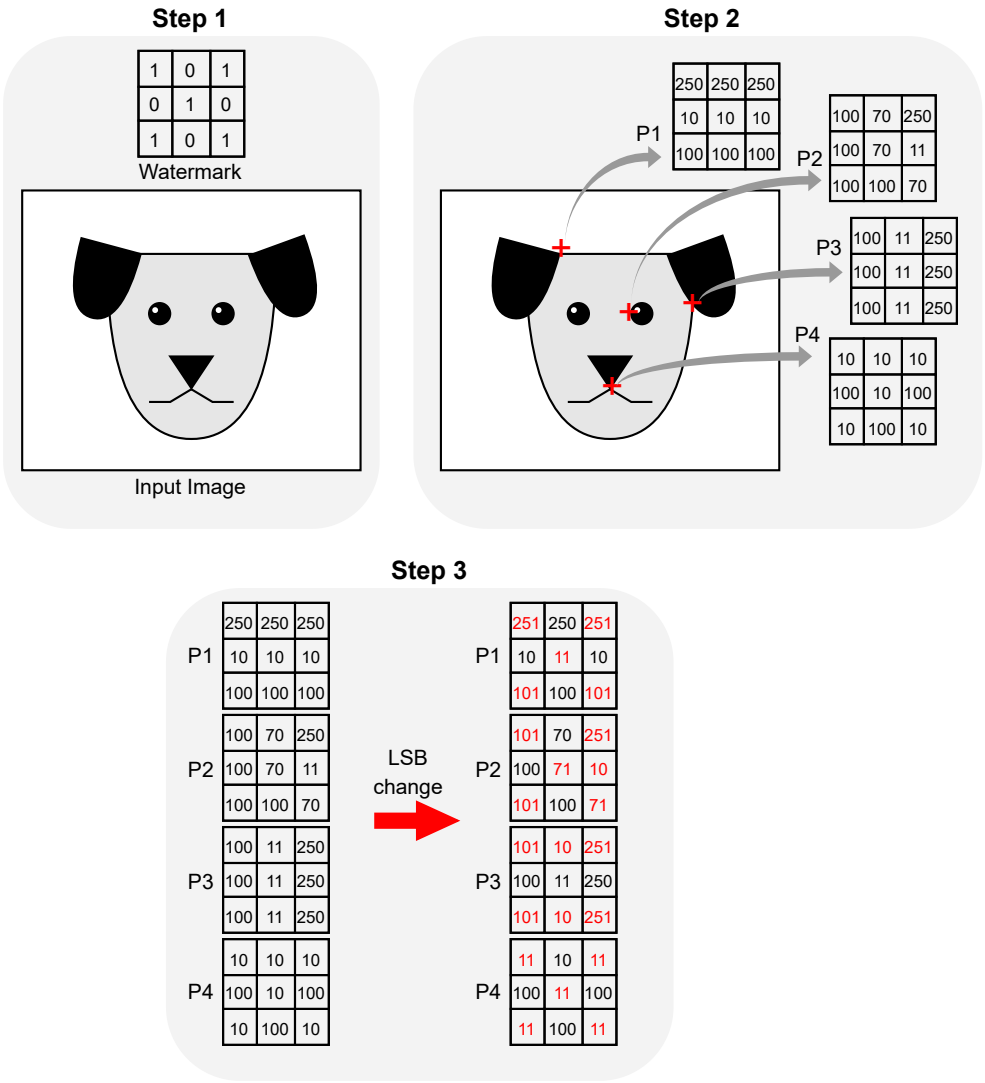

Figure 1: Watermark embedding process: Given an input image and a binary 3×3 watermark with an 'X' pattern (Step 1), four keypoints are detected using the SIFT algorithm (Step 2). The 3×3 pixel neighbourhoods around each detected point are then modified according to the watermark pattern (Step 3).

the cover image using LSB. Ensure the watermark remains imperceptible to the human eye but retrievable with the correct method.

You must satisfy the following steps to create your **Watermark Recovery**:

1. Reapply the SIFT algorithm to the modified image (with the watermark embedded).
2. Detect the same set of keypoints from the modified image and recover the watermark by retrieving the changes made to the pixel values at these points.
3. Keypoints that do not contain the watermark indicate that the image was not authenticated or has been modified.

You must satisfy the following steps to create your **Tampering Detector**:

1. Build a tool to upload an image and determine if it has been tampered with by comparing the watermark's consistency.
2. This tool should test for common tampering techniques (such as cropping, resizing, or rotating) and flag any discrepancies in the watermark extraction process.

You are also required to deliver a simple UI/Visualization tool as described in Section 3.

3 UI and Visualization

We expect a simple desktop app where the primary functions are accessible via clickable buttons, and the generated results are displayed. Your assignment will culminate in the development of a tool featuring three essential functionalities:

- **Watermark Embedder** – to embed a watermark into an image. Input and output are images.
- **Authenticity Verifier** – to verify if an image contains the watermark. Given an image, it should return "Yes" or "No".
- **Tampering Detector** – to detect whether an image has been altered based on watermark consistency. Given an input image, the tool should return "Yes" if tampering is detected and "No" otherwise. If tampering is detected, the output should include the image marked with keypoints that do not match the expected watermark. This is particularly useful for identifying specific manipulations or image composites.

4 Deliverables and Marking

You should submit a **ZIP file** via **Blackboard** containing your program code, image files, and a report detailing your implementation and results. The deliverables and marking criteria are outlined below.

Deliverables

1. **Report** (max. 2000 words, approximately 4 pages in 11pt font): A concise explanation of the entire process, including:

   - How the SIFT algorithm was used to detect keypoints.
   - How the watermark was encoded and embedded into the cover image.
   - How the watermark was recovered and evaluated.
   - A brief discussion on the invisibility and robustness of the watermark.
   - A summary of visibility, recovery, and resilience tests, including any challenges encountered and how they were addressed.

   You must also answer the following questions in your report:

   - How did you manage overlapping watermarks associated with nearby keypoints?
   - What modifications have you made compared to the standard specification? What are the advantages and disadvantages of your approach?
   - How effective was the implemented approach for image tampering detection? Mention limitations and possible improvements.
   - What do you consider the most innovative elements of your implemented approach, and why?

2. **Simple Desktop Application** with the following three tools:

   - Watermark Embedder – Embeds the watermark into the cover image.
   - Authenticity Verifier – Checks if an image contains the expected watermark.
   - Tampering Detector – Identifies potential modifications by analyzing watermark consistency.

   The tool should be simple and user-friendly, with minimal functionality—primarily consisting of buttons to trigger the tools and select the images to be processed.

3. **Images** – Example images for testing the tools (e.g., cover images and watermark).

4. **Source Code**:

   - Complete source code for keypoint detection (SIFT), watermark embedding, and watermark recovery.
   - Source code for the three tools developed in the application.
   - Include any necessary libraries or dependencies (e.g., OpenCV, Flask for a web app).
   - README file: provide clear instructions on how to set up and run the code.

5. **Demonstration Video**. Prepare a video (max. 3 min) demonstrating how to use the main functionalities of your tools. The video should be uploaded to YouTube as "Unlisted", ensuring that only those with the link can access it. If YouTube is not a viable option, alternative platforms such as Google Drive or OneDrive may be used, provided that the link is accessible without requiring special permissions. **The link to the video must be included in the report**.

Marking Scheme

The allocation of marks is 40% for the program code, 40% for the report, and 20% for innovation and creativity.

The program code

- Code structure, including comments and layout of the UI, use of data structures and use of functions (20%).
- How well the application works and the completeness of this (80%).

The report

- The descriptions of the following aspects will be evaluated: usage of SIFT algorithm (20%), watermark embedding (20%), watermark recovery (20%), tampering detection analysis (20%), robustness and resilience analysis/tests (20%).

Innovation and Creativity

Assessment will be based on the originality, complexity, depth, and practical application of the developed tools, as demonstrated in the report and code. The assessment will follow the rubric below:

- 0% = No evidence of innovation.
- 20% = Slight variation from standard implementation.
- 40% = Clear attempt at creativity but limited impact.
- 60% = Noticeable innovation with thoughtful execution.
- 80% = Highly innovative with strong implementation.
- 100% = Exceptionally creative and impactful approach.

In the report, you must justify whether your implementation is innovative. For instance, using the same watermark for each keypoint detected by SIFT does not qualify as innovative. A more innovative approach would involve varying the watermarks based on their position in the image, which would enhance the robustness of tampering detection.

5 Practical Considerations

- Unfair means

  The standard School rules for the use of unfair means will be applied, as described in the undergraduate student handbook: `https://sites.google.com/sheffield.ac.uk/comughandbook/your-study/assessment/unfair-means`

  We are aware that there are lots of tutorial sites on the Web. Do not copy them since that would be plagiarism. Instead, just learn from them.

  Do NOT use ChatGPT or any other AI tools for this assignment. This will be treated as plagiarism for the purposes of this assignment.

- Late hand-in

  Standard Department rules will be used for late hand-in – see the undergraduate student handbook: `https://sites.google.com/sheffield.ac.uk/comughandbook/your-study/assessment/late-submission`.

A Appendix: Image-to-image Steganography Example

In this example, we use a simple 5×5 grayscale matrix as the carrier image and embed a binary secret image by modifying the least significant bit (LSB) of the carrier.

$$C = \begin{bmatrix} 120 & 132 & 140 & 150 & 160 \\ 110 & 121 & 131 & 141 & 151 \\ 100 & 111 & 122 & 132 & 142 \\ 90 & 101 & 112 & 123 & 133 \\ 80 & 91 & 102 & 113 & 124 \end{bmatrix}$$

The secret image or "watermark" is a binary (black-and-white) image where each pixel is either 0 or 1:

$$S = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

To hide S in C, we replace the least significant bit (LSB) of each pixel in C with the corresponding value from S:

$$C' = \begin{bmatrix} 121 & 132 & 141 & 150 & 161 \\ 110 & 121 & 130 & 141 & 150 \\ 101 & 110 & 123 & 132 & 143 \\ 90 & 101 & 112 & 123 & 132 \\ 81 & 90 & 103 & 112 & 125 \end{bmatrix}$$

To recover S, we extract the least significant bit from each value in C':

$$S' = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

Since $S' = S$, we have successfully hidden and retrieved the secret image.

Note: In this assignment, the watermark will not be applied to the entire input image. Instead, it should be embedded only at a set of N keypoints detected using SIFT.