Assignment #1

Prof. Hanbyul Joo
M1522.001000, Computer Vision

Assigned: April 1, 2025
Due: April 17, 2025, 11:59 PM

0 Instruction

In this assignment, you will implement SVD, Image Filters, HoG, Harris Corner, and SIFT, and solve related problems.

- **Submission Platform**: All homework must be submitted electronically on eTL.
- **Collaboration Policy**: Discussions with peers are encouraged, but you must solve the problems and write up the solutions independently.
- **Individual Assignment**: Each student is required to submit their own individual report and code.
- **Plagiarism Policy**: Do **not** copy code or reports from others. Any form of plagiarism may result in a score of zero.
- **Coding Requirements**:
  - Use **Python 3** for all programming tasks.
  - You are **not allowed** to use high-level image processing or computer vision functions such as `cv2.filter2D`.
  - If you're not sure whether a function or package is allowed, please post your question on the eTL Q&A board.
- **Reporting**:
  - Answers must be **clear, unambiguous**, and **supported by experimental results** (e.g., images, plots, brief quantitative analysis).
  - Only PDF submissions compiled using LaTeX (e.g., via Overleaf or other LaTeX tools) will be accepted. No specific template will be provided.
- **Notebook Submission**:
  - You must also submit the **Jupyter Notebook (.ipynb)** file.
  - Adding new cells is allowed.
  - Make sure that **all outputs are preserved**.
  - All images in the report should be **reproducible** from the notebook.
  - TAs will primarily check if your code is **fully reproducible**.
- **Questions**: We will only respond to questions posted on the eTL Q&A board.

- **Structure**: Submit the zip file as the following folder and file structure.

```
{YOUR_NAME}_{YOUR_STUDENT_ID_NUMBER}.zip
├─ {YOUR_NAME}_{YOUR_STUDENT_ID_NUMBER}.pdf (your report)
├─ problem1
│  └─ svd.ipynb
├─ problem2
│  ├─ hog.ipynb
│  ├─ my_template.png
│  └─ output.png
├─ problem3
│  └─ harris.ipynb
└─ problem4
   ├─ sift.ipynb
   ├─ 4-4c_img1.png
   └─ 4-4c_img2.png
```

1 Linear algebra (20 Points)

1-1. Geometric Interpretation of the Pseudo Inverse (7 Points).

Let $\mathbf{A} \in \mathbb{R}^{m \times n}$ with $m > n$, and assume that $\mathbf{A}$ has full column rank ($\text{rank}(\mathbf{A}) = n$). Let $\boldsymbol{b} \in \mathbb{R}^m$ be a vector that does *not* lie in the column space of $\mathbf{A}$, i.e., $\boldsymbol{b} \notin \text{Col}(\mathbf{A})$. Consider the system $\mathbf{A}\boldsymbol{x} = \boldsymbol{b}$, which has no exact solution.

(a) (3 Points) On the plane (paper or tablet pc), draw a simple sketch to represent the situation where:

- $\text{Col}(\mathbf{A}) \subset \mathbb{R}^m$ is a 2D plane through the origin,
- $\boldsymbol{b} \in \mathbb{R}^m$ is a vector not lying in $\text{Col}(\mathbf{A})$,
- $\hat{\boldsymbol{b}} \in \text{Col}(\mathbf{A})$ is the orthogonal projection of $\boldsymbol{b}$ onto $\text{Col}(\mathbf{A})$,
- $\hat{\boldsymbol{x}} \in \mathbb{R}^n$ is the solution to $\mathbf{A}\hat{\boldsymbol{x}} = \hat{\boldsymbol{b}}$.

Label all vectors and indicate the right angle between the residual vector $\boldsymbol{b} - \hat{\boldsymbol{b}}$ and the subspace $\text{Col}(\mathbf{A})$. Attach your sketch to the report.

(b) (3 Points) Based on your diagram, derive the expression for the least-squares solution $\hat{\boldsymbol{x}}$ that minimizes $\|\mathbf{A}\boldsymbol{x} - \boldsymbol{b}\|^2$. Begin by stating the orthogonality condition for the residual vector $\boldsymbol{r} = \boldsymbol{b} - \mathbf{A}\hat{\boldsymbol{x}}$, derive the corresponding matrix equation, and show that:

$$\hat{\boldsymbol{x}} = (\mathbf{A}^\top \mathbf{A})^{-1} \mathbf{A}^\top \boldsymbol{b}.$$

Remark: The matrix $(\mathbf{A}^\top \mathbf{A})^{-1} \mathbf{A}^\top$ is called the Moore–Penrose pseudo-inverse of $\mathbf{A}$, denoted $\mathbf{A}^\dagger$, so:

$$\hat{\boldsymbol{x}} = \mathbf{A}^\dagger \boldsymbol{b}.$$

(c) (1 Points) Using the pseudo-inverse $\mathbf{A}^\dagger$ defined above, describe the geometric meaning of the vector $\mathbf{A}\mathbf{A}^\dagger \boldsymbol{b} \in \mathbb{R}^m$.

1-2. Solving $\mathbf{A}\boldsymbol{x} = \mathbf{0}$ via SVD (7 Points).

Let $\mathbf{A} \in \mathbb{R}^{m \times n}$ be a matrix with $m > n$, and suppose that $\mathbf{A}$ is constructed from real-world observations that may contain small noise. In an ideal, noise-free setting, the underlying matrix $\mathbf{A}$ may be rank-deficient, meaning there exists a nonzero solution to the system

$$\mathbf{A}\boldsymbol{x} = \mathbf{0}.$$

However, in practice, small measurement noise can perturb $\mathbf{A}$ such that it becomes numerically full-rank, eliminating any exact nonzero solution. In this problem, we aim to find a **unit-norm vector $\boldsymbol{x}$** that **approximately satisfies $\mathbf{A}\boldsymbol{x} \approx \mathbf{0}$**, and to interpret this solution via Singular Value Decomposition (SVD).

(a) (2 Points) Let $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$ be the SVD of $\mathbf{A}$, with $\mathbf{V} = [\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n]$. Show that the vector $\boldsymbol{x} = \boldsymbol{v}_n$ minimizes $\|\mathbf{A}\boldsymbol{x}\|$ subject to $\|\boldsymbol{x}\| = 1$.

(b) (1 Points) As described above, nontrivial solutions to $\mathbf{A}\boldsymbol{x} = \mathbf{0}$ may not exist when $\mathbf{A}$ is perturbed by noise and appears numerically full-rank. Briefly explain why it is still useful to minimize $\|\mathbf{A}\boldsymbol{x}\|$ under $\|\boldsymbol{x}\| = 1$, and how this leads to a solution based on the singular values of $\mathbf{A}$.

(c) (3 Points) Consider the matrix

$$\mathbf{A} = \begin{bmatrix} 1 & 0 \\ 0.01 & 0.1 \end{bmatrix}.$$

Let $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$ be its singular value decomposition. On a piece of paper or tablet, draw the following:

- The unit circle in $\mathbb{R}^2$ centered at the origin,

- The right singular vectors $\boldsymbol{v}_1$, $\boldsymbol{v}_2$ as orthogonal input directions,
- The image of the unit circle under $\mathbf{A}$, which becomes an ellipse,
- The left singular vectors $\boldsymbol{u}_1$, $\boldsymbol{u}_2$ as the axes of the output ellipse,
- The lengths of the ellipse axes labeled as σ_1, σ_2.

Note that you should attach your drawing to the report. You may use NumPy or any other tool to compute the SVD numerically — no need to perform the decomposition by hand. Based on your drawing and SVD computation:

- Identify the input direction that is mapped to the shortest output vector by $\mathbf{A}$.

(d) (1 Points) In the context of solving $\mathbf{A}\boldsymbol{x} \approx \mathbf{0}$ with $\|\boldsymbol{x}\| = 1$, explain why the direction $\boldsymbol{x} = \boldsymbol{v}_2$ (the right singular vector corresponding to the smallest singular value) is a meaningful approximate solution. What does this direction represent in terms of how $\mathbf{A}$ transforms the input space?

1-3. Image Compression with SVD (6 Points).

Open the file `1_svd_compression.ipynb` to complete this problem. You may run the notebook either in Google Colab or in your local Python environment.

(a) (3 Points) Using the provided Steve Jobs image, first check its resolution (i.e., height and width in pixels). Suppose we perform rank-k SVD compression on the image. Derive the compression ratio, defined as:

$$\text{Compression Ratio} = \frac{\#\text{ of original pixel values}}{\#\text{ of parameters in the compressed representation}}.$$

Express your answer as a function of k, assuming that the compressed representation stores the top-k components of $\mathbf{U}$, $\mathbf{\Sigma}$, and $\mathbf{V}$.

Then, in the provided notebook, complete the implementation of the following two functions in Step 3:

- `svd_compress(u, s, vt, k)`: reconstructs the image using the top-k singular values and vectors,
- `compression_ratio(u, s, vt, k, original_shape)`: computes the compression ratio using your formula above.

(b) (1 Points) Compute the largest integer k^* such that the compressed representation is smaller than the original image, i.e., where compression is actually achieved.

(c) (2 Points) Perform SVD compression using the following values of k:

$$k \in \{5, 20, 50, 100, k^*\}.$$

Following Step 5 of the provided notebook, save the compressed images as JPG files, and attach them with the original image to your report (no need to upload them separately).

Compare the file sizes of the saved images, and briefly explain the results in terms of the compression ratio and visual quality.

2 HOG: Histogram of Oriented Gradients (30 Points)

Direction

You will use the provided skeleton Python code (`hog.ipynb`) that implements HOG (Histogram of Oriented Gradients) feature extraction, differential filtering, and sliding window-based face detection. You are not required to strictly follow the structure or framework of the provided code. You are free to upgrade or improve the code as you wish. Additionally, all parameters, including the HOG parameters, can be freely adjusted. Your tasks include analyzing the effect of various differential filters, applying the detection algorithm on provided images, and experimenting with your own template and target images.

Template-Based Face Detection and Analysis

The code utilizes sliding window matching with HOG features and NCC (Normalized Cross-Correlation) to detect faces.

(a) (10 Points) Detection Using Provided Images:
Use the provided `template.png` (containing a face region) and `target.png` images. Run the face detection algorithm and generate the output images (e.g., `detections.png` and `hog_visualization.png`). In your report:

- (3 Points) Attach the resulting images.
- (7 Points) Analyze the detection performance. Discuss any false positives, missed detections, or areas of improvement.

(b) (14 Points) Detection with Self-Selected Images:
Select your own template image (ensure it contains a clear face) and a target image that includes at least two people. Run the detection algorithm using these images. In your report, include:

- The target and template images you selected.
- (6 Points) The resulting detection output.
- (8 Points) A comparative analysis with the results from the provided `template.png` and `target.png`. Explain any differences in performance and possible reasons.

(c) (6 Points) Differential Filter Analysis:
Using the provided code, run experiments with each differential filter: Sobel, Prewitt, and Scharr. In your report, include:

- (2 Points) A brief description of each filter's kernel (using matrix notation) and how it computes image gradients.
- (4 Points) Provide a qualitative comparison of the face detection results and HOG visualizations obtained using each method. Briefly analyze which filter yielded the best results.

References

- `https://www.youtube.com/watch?v=0Zib1YEE4LU`
- `https://www.youtube.com/watch?v=92-NaIdIrEs`

3 Harris Corner Detection (20 Points)

3-1. Harris Detector: Gradient and Local Structure (10 Points).

(a) (3 Points) Visualize the horizontal gradient $|I_x|$, vertical gradient $|I_y|$, and gradient magnitude $\sqrt{I_x^2 + I_y^2}$ using the provided code. Describe where in the image these gradients are strong. What kind of image features do strong gradients correspond to?

(b) (5 Points) Use the interactive tool to explore the Harris matrix and eigenvalue ellipses at various pixel locations. Identify three representative regions in the image:

- A flat region (both eigenvalues small),
- An edge-like region (one eigenvalue large, one small),
- A corner (both eigenvalues large).

For each region, record:

- The pixel coordinates,
- The eigenvalues of the Harris matrix,
- The interpretation given by the tool.

(c) (2 Points) In your own words, explain how the shape and orientation of the eigenvalue ellipse reflect local image structure. What does a small round ellipse, an elongated ellipse, or a roughly circular but large ellipse imply about the region?

3-2. Harris Response and Parameters (10 Points).

(a) (2 Points) Run the complete Harris pipeline using the default parameters in the provided notebook. Specifically, use `sigma=1`, `k=0.05`, and `threshold_ratio=0.01`. Report the total number of detected corners. Visualize the result using `plot_corners`.

(b) (3 Points) Fix `sigma=1` and vary the Harris sensitivity constant k using values:

$$k \in \{0.01,\ 0.1,\ 0.2\}.$$

For each value of k:

- Count the number of detected corners.
- Display the corner overlay image.
- Briefly describe how the corner distribution and density change.

(c) (3 Points) Fix $k = 0.05$ and now vary the Gaussian smoothing parameter `sigma`:

$$\texttt{sigma} \in \{0.5,\ 1.0,\ 2.0\}.$$

Repeat the steps above for each value. What trade-offs do you observe in terms of noise, stability, and corner localization?

(d) (2 Points) **Effect of Non-Maximum Suppression.** Modify the corner detection function to *disable* the local maximum check in the `detect_corners` function, i.e., remove the condition:

```
R[y, x] == np.max(window)
```

and detect corners based on thresholding only.

- Visualize the result and compare it to the original version with NMS.
- How does the number of detected corners change?
- What visual differences do you observe in the corner distribution?
- What is the role of NMS in improving corner quality?

4 SIFT: Scale-Invariant Feature Transform (30 Points)

4-1. Gaussian and DoG Pyramid (7 Points)

(a) (3 Points) Run the function `build_gaussian_pyramid` by generating scale-space images per octave using Gaussian blur. Why do we generate `(num_intervals + 3)` images instead of just `num_intervals + 1`?

(b) (2 Points) Visualize the Difference-of-Gaussian (DoG) pyramid. Describe how the structures (edges, blobs, textures) change across octaves and scales.

(c) (2 Points) Suppose we double the initial `sigma` value to $\sigma = 3.2$. How would this affect the detection of keypoints at finer scales? What would happen if we set $\sigma = 0.5$ instead?

4-2. Scale-space Extrema Detection (6 Points)

(a) (2 Points) Describe how a local 3D patch in scale-space is used to detect extrema. Why do we check across three adjacent scales?

(b) (2 Points) How many candidate keypoints are detected for `contrast_threshold=0.03`? What happens if you increase or decrease this threshold?

(c) (2 Points) What types of regions (e.g., edges, flat areas, corners, blobs) tend to dominate the keypoint candidates? Use visual evidence from the DoG pyramid to support your claim.

4-3. Rotation and Scale Invariance of SIFT (10 Points)

SIFT achieves invariance to image rotation and scale by assigning dominant orientations and detecting extrema in scale-space. In this problem, you will verify and interpret these invariance properties both visually and quantitatively.

(a) (4 Points) **Rotation Invariance.** Rotate the input image by 90 degrees and rerun the keypoint detection and orientation assignment steps.

- Visualize the dominant orientations before and after rotation.
- Select a few matched keypoints from both images (e.g., top 5 by contrast or proximity).
- Compare their dominant orientation values numerically. What is the expected difference due to the 90-degree rotation?
- Does the assigned orientation change accordingly? What does this indicate about SIFT's rotation invariance?

(b) (4 Points) **Scale Invariance.** Resize the original image by a factor of 2 (either upsample or downsample), and rerun keypoint detection and orientation assignment.

- Visually compare the keypoints from the original and resized images.
- Are similar keypoints detected at corresponding positions? How do their orientations and scales compare?
- Summarize your findings. What aspects of the keypoint information are preserved under scaling, and what changes?

(c) (2 Points) **Why Orientation Histograms?** SIFT assigns orientations by constructing a weighted histogram of gradient directions within a patch. Explain why this method is more robust than simply selecting the single strongest gradient direction, especially in noisy or textured regions.

4-4. OpenCV Matching (7 Points)

(a) (2 Points) Match keypoints between two different viewpoints of the same scene using OpenCV's built-in `cv2.SIFT_create`. Report the number of keypoints in each image and the number of good matches retained after Lowe's ratio test.

(b) (3 Points) How does the value of the Lowe's ratio threshold (e.g., 0.4 vs 0.75) affect the number of good matches and the presence of false positives?

(c) (2 Points) Try switching the image pair to unrelated scenes (e.g., Taj Mahal vs Eiffel Tower). What happens to the number of matches? What does this indicate about SIFT's selectivity?

(*Hint:* SIFT is known for its selectivity, but not all descriptors behave this way. Consider what might happen if you used a very simple patch-based descriptor (e.g., raw pixel intensities from a small region). Would such a method still reject incorrect matches when comparing unrelated scenes?)