Assignment 2
SIFT

Parallel Programming
2025/10/03

Introduction

SIFT (Scale-Invariant Feature Transform)

- It is used to detect local features in an image by searching for extrema in the spatial scale and extracting their position, scale, and rotation invariance.

Steps

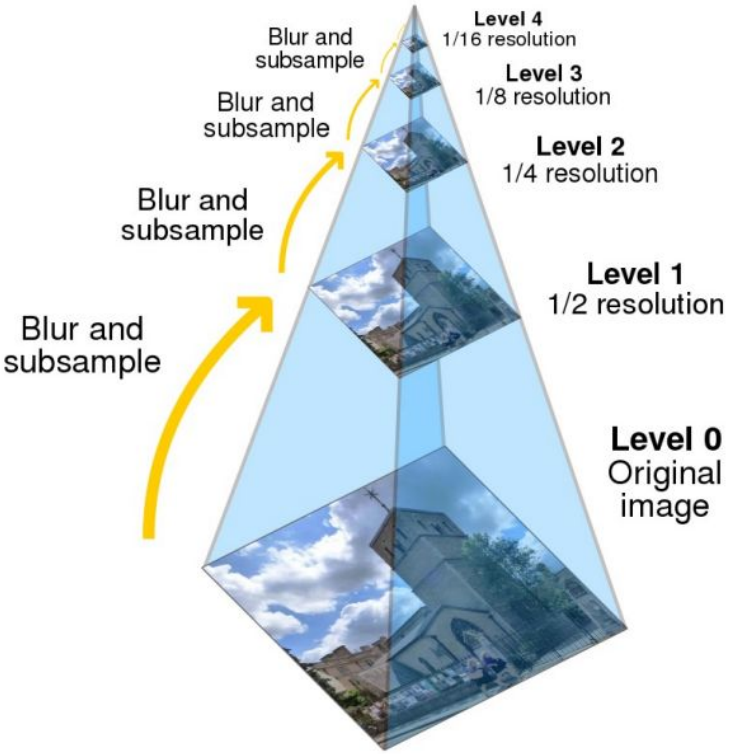
Only need to find keypoints & descriptors in an image.

1. Generate Gaussian Pyramid
2. Generate DoG Pyramid
3. Find Keypoints
4. Assign Orientation & Create Descriptor

Gaussian Pyramid

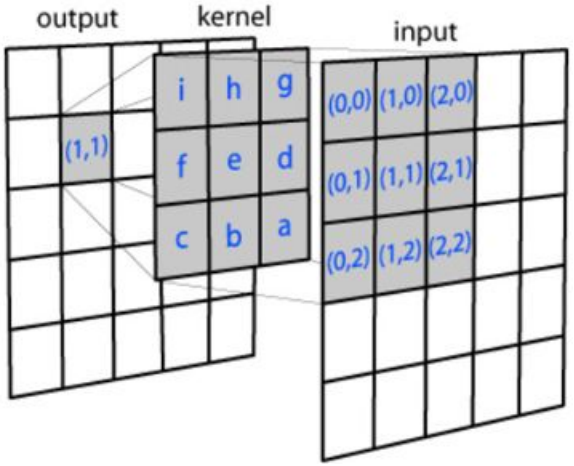
Construction process:

- Start with the original image, change the Gaussian blur sigma, and generate multiple images within the same octave.
- Downsample a selected image, and use it as the starting image of the next octave.



Gaussian Blur

- When downsampling an image, apply a low-pass filter to the image before resampling.
- Instead of performing a full 2D convolution, use two 1D convolutions: one horizontal and one vertical, in the sample code.

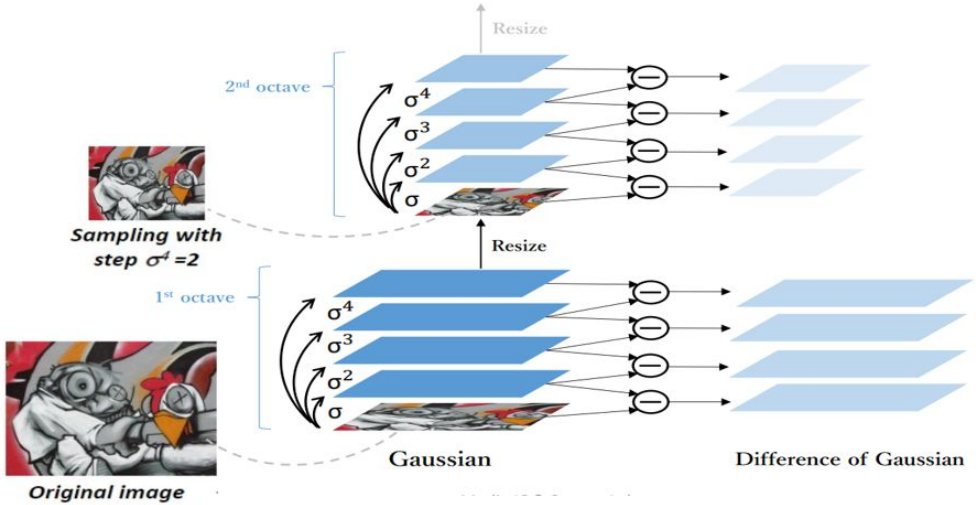


$$\frac{1}{256} \cdot \begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$$

$$G(x_i, y_i, \sigma) = \frac{1}{2\pi\sigma^2} exp(-\frac{(x - x_i)^2 + (y - y_i)^2}{2\sigma^2})$$

DoG (Difference of Gaussian) Pyramid

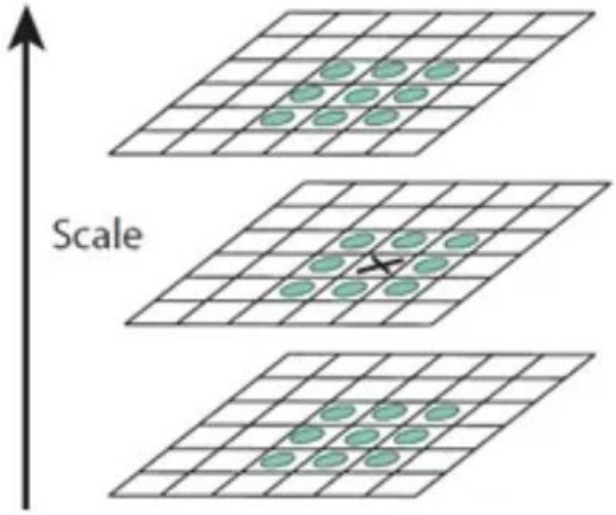
- Subtract adjacent Gaussian-blurred images within the same octave.



$$DoG = G(x, y, k\sigma) * I - G(x, y, \sigma) * I$$

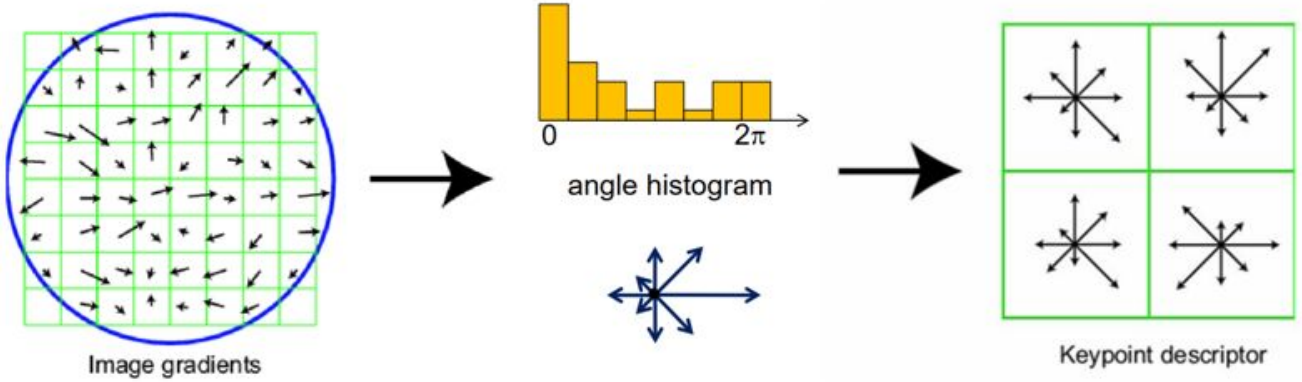
Finding keypoints (local extrema)

1. Detect Local Extrema
   - For each pixel in the DoG image, compare it with its 26 neighbors to find local maxima or minima. (Halo communication pattern)
2. Feature Point Filtering and Refinement
   - Discarding low-contrast points and points located along edges.



Assign Orientation & Create Descriptor

- An orientation is assigned to each keypoint based on the local gradient direction.
- For each keypoint, a descriptor is created by sampling the gradient magnitudes and orientations within a local region.



Goal

- You are asked to parallelize it using MPI and OpenMP (or pthread).
  - Understand the importance of Load Balancing.
  - Analyze the Arithmetic Intensity (communication vs computation) of different operations.
- C++ std::thread library is prohibited in this assignment !
- Only C++ is allowed!

Resources

- /work/b10502076/pp25/hw2
  - hw2.cpp
  - testcases
  - goldens
  - results
  - image.hpp, image.cpp
  - sift.hpp, sift.cpp
  - stb_image.h, stb_image_write.h
  - validate.py, validate
  - Makefile
  - scripts
    - env.sh, conda.sh

We provide a sequential version of the sample code

```
cp -r /work/b10502076/pp25/hw2 .
```

Your Implementation Files

- /work/b10502076/pp25/hw2
  - hw2.cpp The main function
  - testcases
  - goldens
  - results
  - image.hpp, image.cpp
  - sift.hpp, sift.cpp
  - stb_image.h, stb_image_write.h
  - validate.py, validate
  - Makefile
  - scripts
    - env.sh, conda.sh

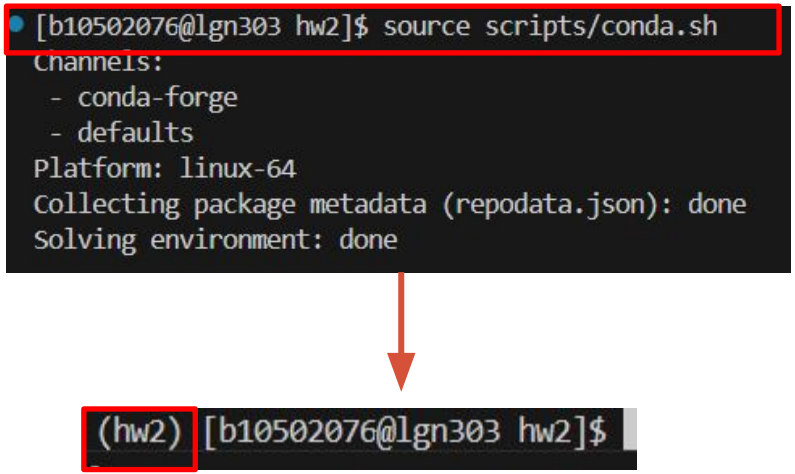
Implement some functions related to SIFT and Image processing

Create Miniconda Environment

- Create the environment for validation.
  - Under scripts/conda.sh

```
module load miniconda3
conda create -n hw2 python=3.12 -y
conda activate hw2

pip install --upgrade pip
pip install scikit-image
pip install opencv-python
```



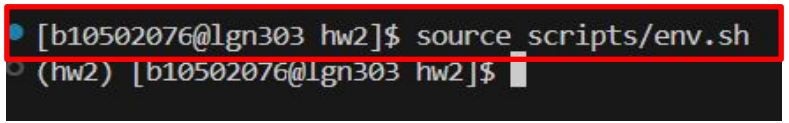
In this assignment, the environment needs to be created only once !!!

Environment Setting

- Setting the environment for MPI execution.
  - Under scripts/env.sh

```
module purge
module load miniconda3
conda activate hw2

module load gcc/13
module load openmpi
export UCX_NET_DEVICES=mlx5_0:1
```



After creating the conda environment,
you need to set the environment each time you open a new terminal !!!

Input

- Under testcases/xx.jpg
- Sample test cases
  - ❏ xx.jpg (xx : 01 ~ 08)

There are 8 public testcases, but we only test 4 testcases on the judge server to reduce the overhead.

However, TA will finally test all the 8 public testcases and the private testcases for score evalution.

Goldens

- Under goldens directory
  - xx.jpg (01~08)
  - xx.txt (01~08)

#keypoints (x, y, octave, scale) descriptor

```
45732
6108 333 0 1  4 0 0 0 0 6 127 38 18 0 0 0 0 0 80 24 19 4 2 1 1 0 3 1 0 5 52 2 0 0 0 0 32 12 8 0
1079 2621 0 2  0 18 33 41 53 0 0 0 3 25 6 12 60 6 0 0 6 18 12 28 17 2 1 4 1 2 3 4 6 2 12 12 148
```

Output

- Save the output (xx.jpg, xx.txt) under the results directory.

  ~/hw2/results

Makefile

- A simple way to compile your program.

`$ make`

to compile your hw2.cpp as hw2

`$ make clean`

to remove the executable file

```
(hw2) [b10502076@lgn303 hw2]$ make
mpicxx -std=c++17 -O3 -fopenmp -I. -c hw2.cpp
mpicxx -std=c++17 -O3 -fopenmp -I. -c sift.cpp
mpicxx -std=c++17 -O3 -fopenmp -I. -c image.cpp
mpicxx -std=c++17 -O3 -fopenmp -o hw2 hw2.o sift.o image.o
(hw2) [b10502076@lgn303 hw2]$
```

Execution

Remember to set the time limit when you submit your slurm job.

- `srun -A ACD114118 -N $node -n $proc -c $core --time=00:03:00 ./hw2 ./testcases/xx.jpg ./results/xx.jpg ./results/xx.txt`

```
$ srun -A ACD114118 -N 2 -n 4 -c 6 --time=00:03:00 ./hw2
./testcases/01.jpg ./results/01.jpg ./results/01.txt
```

Launch 4 processes on 2 nodes, Each process has 6 CPU cores

The resource allocation for each public testcase is range form -N : 1~3, -n : 1~36, -c: 6 on the judge system

Validation

- Use validate.py to check whether the output is correct.
  - Apply “Structural Similarity” to evaluate the position of keypoints
  - Apply “Euclidean Distance” to verify the direction of descriptors.
  - For a match to be valid, both indices must exceed 0.98.

```
$ python3 validate.py ./results/xx.txt ./goldens/xx.txt
./results/xx.jpg ./goldens/xx.jpg
$ Output: Pass / Wrong
```

This validation only takes few seconds, so you can run on the login node directly.

Judge

- Type hw2-judge under the directory of hw2.
- Scoreboard

```
(hw2) [b10502076@lgn303 hw2]$ hw2-judge
Looking for hw2.cpp: OK
Looking for image.cpp: OK
Looking for image.hpp: OK
Looking for sift.cpp: OK
Looking for sift.hpp: OK
Looking for stb_image.h: OK
Looking for stb_image_write.h: OK
Looking for Makefile: Not Found
Using fallback: /work/b11902043/PP25/.ta/hw2/Makefile: OK
Running: /usr/bin/make -C /home/b10502076/.judge.3441515764 hw2
make: Entering directory '/home/b10502076/.judge.3441515764'
mpicxx -std=c++17 -O3 -fopenmp -I. -c hw2.cpp
mpicxx -std=c++17 -O3 -fopenmp -I. -c sift.cpp
mpicxx -std=c++17 -O3 -fopenmp -I. -c image.cpp
mpicxx -std=c++17 -O3 -fopenmp -o hw2 hw2.o sift.o image.o
make: Leaving directory '/home/b10502076/.judge.3441515764'
```

Report

Answer the following questions in either English or Traditional Chinese. (you can add more as you like)

1. Implementation Detail
   - How do you partition the task?
   - What scheduling algorithm did you use: static, dynamic, guided, etc.?
   - What techniques do you use to reduce execution time?
   - Other efforts you make in your program.
   - What difficulty did you encounter in this assignment? How did you solve them?
2. Analysis:
   - Design your own plots to show the load balance of your algorithm between threads/processes.
   - Analyze your program's scalability in the following aspects:
     - number of nodes
     - number of processes per node
     - number of CPU cores per process
3. Conclusion:
   - What have you learned from this assignment?
   - (Optional) Any feedback or suggestions for this assignment or spec.

Submission

- Due: Fri, 2025/10/17 23:59.
- Zip the following files to a single archived file named <studentID>.tar and submit it to NTU Cool:
  - All .hpp / .h / .cpp files
  - report.pdf
  - Makefile (optional)
- Please note that the first character of your studentID must be lowercase.

Hints

- Profile the program first to identify the bottleneck that needs to be parallelized.
- Analyze the workload balancing and arithmetic intensity.
- Discussing with your friends.

Notes for Assignment !!!

- **Before emailing the TAs :**
  - **Read the specification carefully.**
  - **Review the discussion forum on NTU Cool to see if someone has already asked the same question. If the question is not answered, post it on the forum so all students can benefit.**
  - **For questions about grades or other private matters, email the TAs directly.**
- **Plagiarism is strictly prohibited.**
  - **You are not allowed to share code generated by AI tools with your classmates.**
  - **You may use AI tools for assistance, but you must do so independently.**

Notes for NCHC Server Usage !!!

- **Do not submit jobs aggressively. Each student is limited to 3 jobs (queuing and running).**
  - **Also don’t judge your program many times, because the slurm system can’t detect the reject message from judge server.**
- **Do NOT run or test any compute-intensive program on the login node (e.g., do not run hw2 directly).**
  - **The login node is used to Submit a Slurm job.**
    - **sbatch run.sh / srun …. (tutorial)**
  - **Only the hw2-judge (not compute node) and compilation can run on the login node.**

Notes for NCHC Server Usage !!!

- **Use the sacct command at any time to check queued jobs.**
  - **sacct -u <username> to check your job status.**

```
[b10502076@un-ln01 ~]$ squeue -u b10502076
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
            769633      gp2d      hw2 b1050207  R       0:38      1 gn1201
[b10502076@un-ln01 ~]$ sacct -u b10502076
       JobID    JobName  Partition    Account  AllocCPUS      State ExitCode
------------ ---------- ---------- ---------- ---------- ---------- --------
769632              hw2       gp2d  acd114010          1     FAILED      2:0
769632.exte+     extern             acd114010          1  COMPLETED      0:0
769632.0            hw2             acd114010          1     FAILED      2:0
769633              hw2       gp2d  acd114010          1  COMPLETED      0:0
769633.exte+     extern             acd114010          1  COMPLETED      0:0
769633.0            hw2             acd114010          1  COMPLETED      0:0
[b10502076@un-ln01 ~]$
```

Notes for NCHC Server Usage !!!

- **If you want to stop a running program, terminate it with Ctrl+C rather than suspending it with Ctrl+Z.**
- **Do not run long-duration programs.**
  - **Always set the time limit when you submit a Slurm job.**
- **Slurm job timeouts will not affect the final evaluation. We will detect such issues during testing and rerun the job if necessary.**
- **If you do not want to wait in the job queue, please run the program on your own personal computer.**

Reference for NCHC Server Usage

- If you are unsure about using the NCHC server, you can refer to the following resources or email the TAs.
  - **Service Overview:** https://man.twcc.ai/@TWCC-III-manual/H1bEXeGcu
  - **Module Tutorial:** https://man.twcc.ai/@TWCC-III-manual/ryDQk2yKO

Q & A

Feel free to ask if you have any questions.