Lab: Part 2

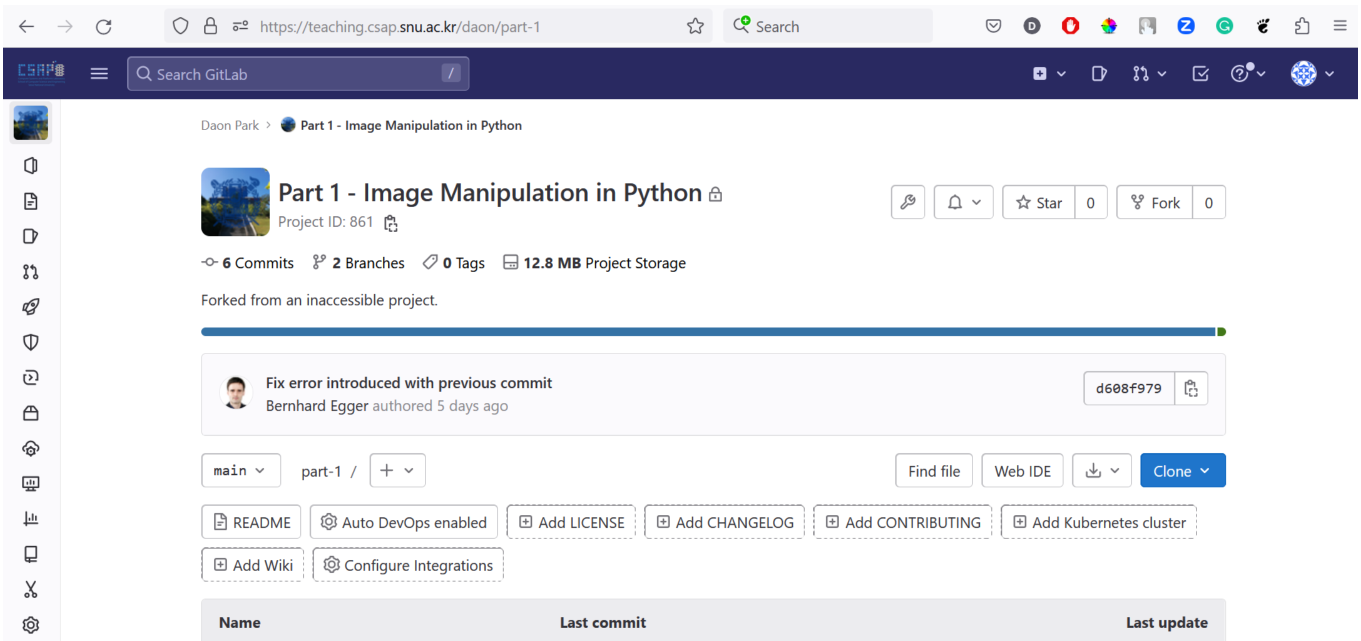
The Need for Speed: Image Manipulation in C

2023/04/10 ~ 2023/05/01

Computer Systems and Platforms Lab
Department of Computer Science and Engineering
Seoul National University

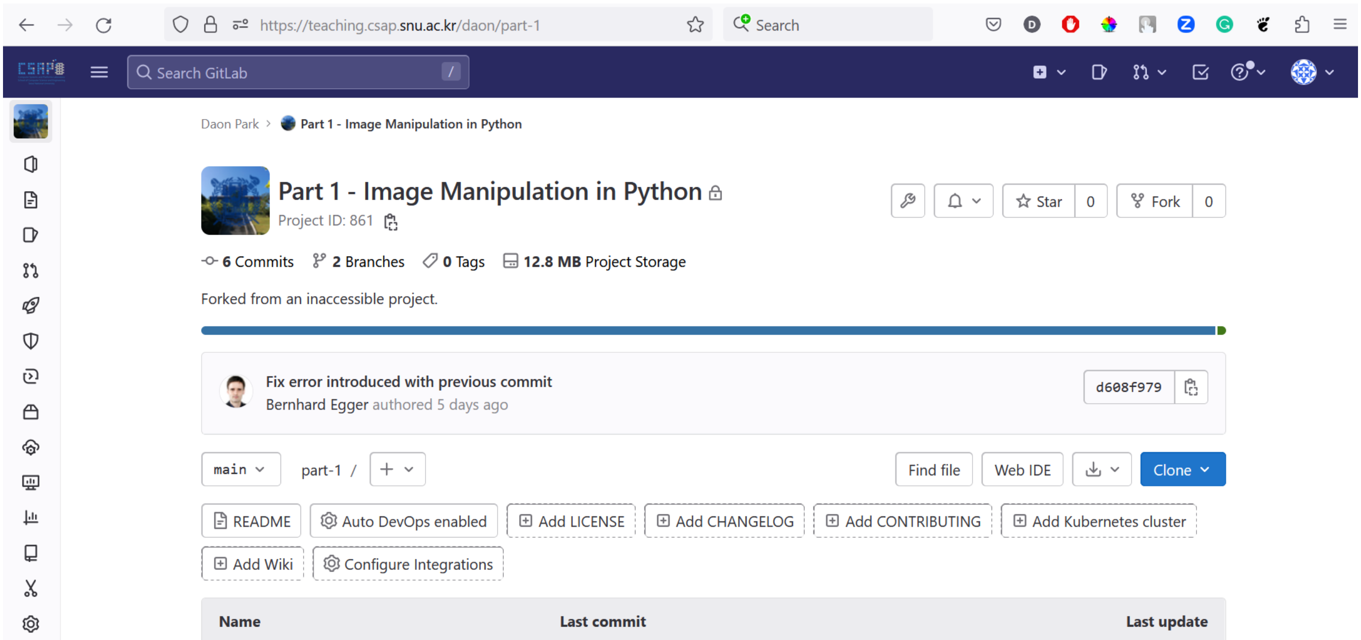
Username

- **Username not being the student ID will fail during grading**



Fork

- Set the visibility to private so that others cannot see your code
- Do not change your project name



Tag

- Create “Submission” tag
- Check for spelling error(s) in the tag

Part 1 - Image Manipulation...

Project information

Repository

Issues 0

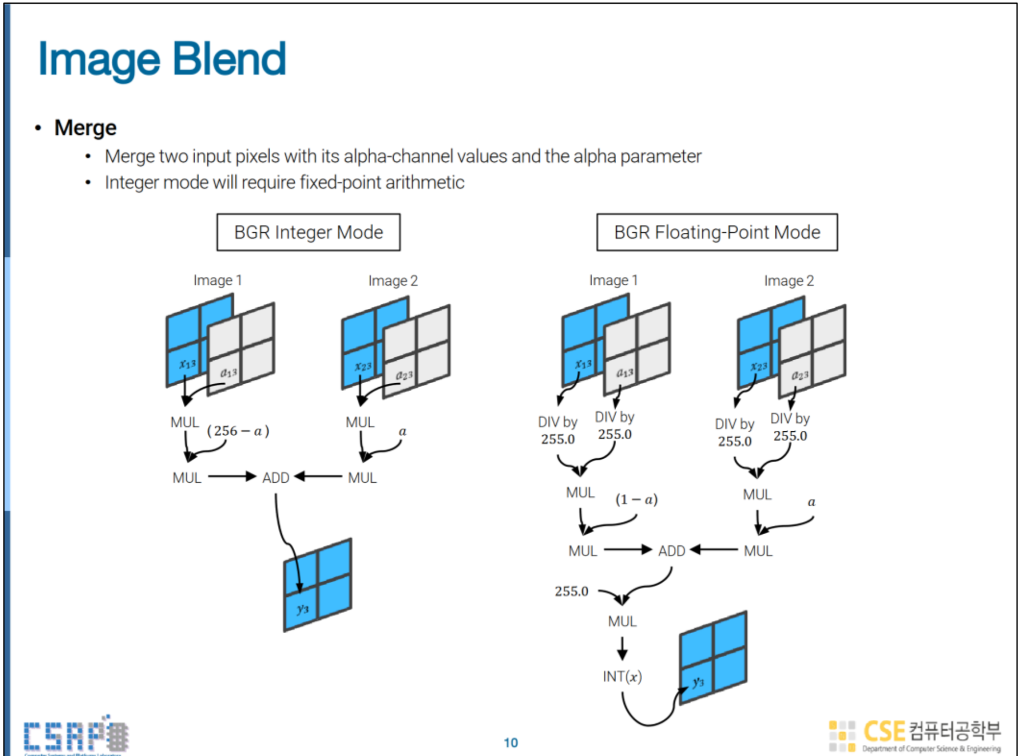
Submission

Commit and push your work frequently to avoid data loss. When you are ready to submit your code for grading, create a tag called "Submission". The timestamp of the "Submission" tag is considered your submission time.

To create a tag, visit the repository on GitLab and navigate to Repository -> Tags. Enter "Submission" as the Tag name then hit "Create tag". You can leave the other fields empty.

Fixed-Point Multiplication

- **The implementations are completely up to you**



Fixed-Point Multiplication

- The handout literally told you everything

- **Floating-point code**
  The the images store the intensities as 8-bit integers ranging from 0 to 255. You need to normalize all channels of a pixel by dividing the integer value by 255 before performing your calculations. The alpha parameter is provided as a float in the range 0.00 to 1.00, so you do not have to normalize it anymore.
- **Fixed-point code**
  To avoid the expensive floating point operations, many image manipulation libraries offer a faster but slightly inaccurate fixed-point implementation.
  In our fixed-point variant, we use 1.8 bit fixed-point values to represent a number range from 0.00000000_2 to 1.00000000_2. To avoid expensive scaling operations, we interpret the 8-bit pixel values directly as 1.8-bit fixed-point numbers, yielding a range from 0.00000000_2 to 0.11111111_2. Since the first bit is always zero, we can drop the leftmost bit and store the fixed-point values of a pixel in four 8-bit characters. Since we are unable to represent 1.0, we lose a little bit of intensity when calculating with fixed-point numbers. As a result, the result of the floating-point code and the integer fixed-point variant will not be bit-identical. When calculating with fixed-point numbers, remember what you have learned about fractional binary numbers, in particular, what happens to the radix point when multiplying numbers. Make sure to add/subtract only values that have the radix point in the same bit position and convert the number back to our .8-bit format before storing the values in the output pixel.

Lab 2

- **Project structure**

```
part-2
|-> blend_driver.c
|-> blend_float.c
|-> blend.h
|-> blend_int.c
|-> blur_driver.c
|-> blur_float.c
|-> blur.h
|-> blur_int.c
|-> imlib.c
 -> imlib.h
```

- **Commands**

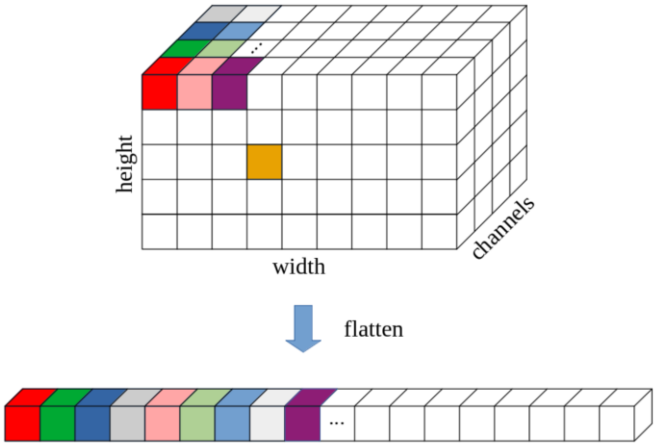
```
// compile
make all
// blur
./blend_driver [-h]
               [--type {int,float}]
               [--mode {overlay,merge}]
               [--alpha ALPHA]
               [--output OUTPUT]
               image1
               image2
// blend
./blur_driver [-h]
              [--type {int,float}]
              [--kernel {3x3,5x5,7x7}]
              [--output OUTPUT]
              image
```

Computer Systems and Platforms Laboratory

CSE 컴퓨터공학부 Department of Computer Science & Engineering

Raw Format

- **Raw image input(s) only**
  - Reads in raw image as the 1D format instead of 3D format



- **Macros**
  - `PIXEL()`
  - `INDEX()`

Dynamic Allocation

- **Dimension calculations**
  - First, need to know the dimensions to properly calculate how much to allocate
- **Memory allocation**
  - Allocate memory with `malloc()` or `calloc()`
- **Error checking**
  - Always check if there was an error after memory allocation
- **Memory freeing**
  - Once the data structure is not in use anymore, free the memory

Performance

- **Time measurements**
  - Warm-ups (usually for GPUs, but why not?)
  - Multiple iterations (at least 10 times)
  - Leave out the best and the worst performance