

ID2090 A6

Anton Beny M S, ME23B015

May 2024

Contents

1	Treasure Hunt II	2
1.1	Introduction	2
1.2	Hunt	2
1.2.1	Running the script	2
1.2.2	Searching for the pokemon	3
1.2.3	Merging with <code>master</code>	3
1.3	Observations	4
2	Image Processing	5
2.1	Introduction	5
2.2	Some Kernels	5
2.2.1	Brightness Change	5
2.2.2	Edge Detection	6
2.2.3	Blurring	6
2.2.4	Sharpening	8
2.3	Code	8
2.3.1	Reading the image	8
2.3.2	Defining the Kernels	8
2.3.3	Defining the Convolution Function	9
2.3.4	Generating the images	9
2.3.5	Denoising	10
2.4	Observations	12
2.5	Conclusion	13

Task 1

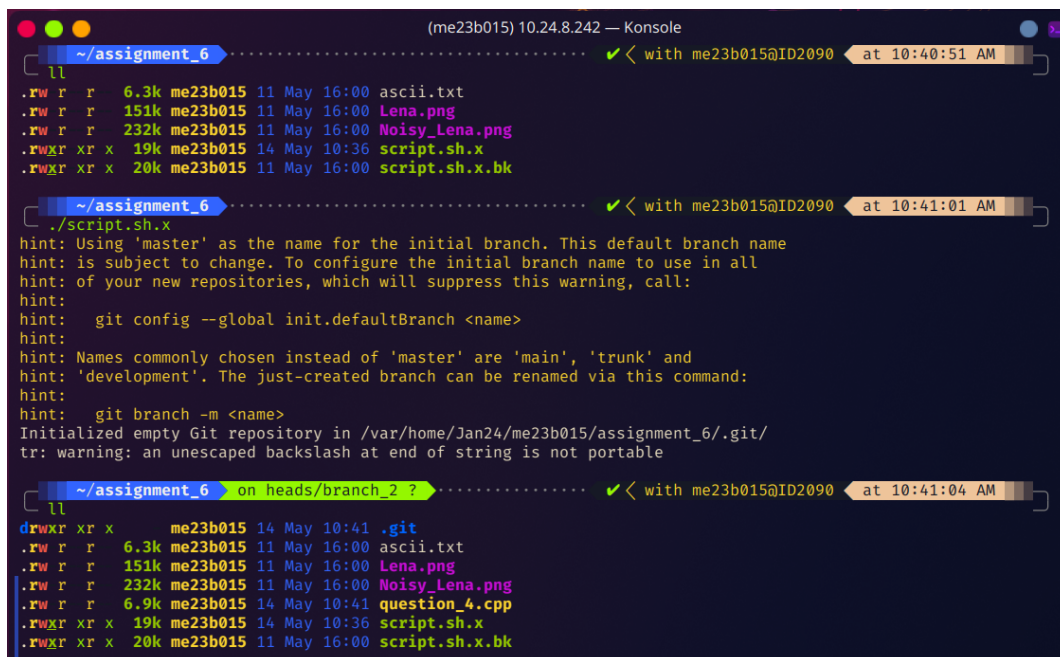
Treasure Hunt II

1.1 Introduction

In this task, we have to search for "legendary pokemon: Arceus". The pokemon is represented as ascii art, and we have to find it in the given repository, by going through all the commits.

1.2 Hunt

1.2.1 Running the script



```
(me23b015) 10.24.8.242 — Konsole
~/assignment_6 ..... ✓ < with me23b015@ID2090 at 10:40:51 AM
ll
.rw r r 6.3k me23b015 11 May 16:00 ascii.txt
.rw r r 151k me23b015 11 May 16:00 Lena.png
.rw r r 232k me23b015 11 May 16:00 Noisy_Lena.png
.rwxr x x 19k me23b015 14 May 10:36 script.sh.x
.rwxr x x 20k me23b015 11 May 16:00 script.sh.x.bk

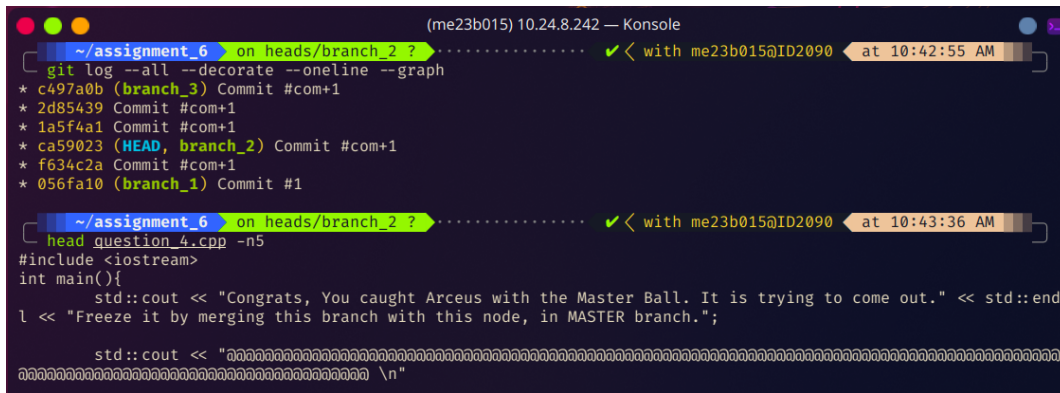
~/assignment_6 ..... ✓ < with me23b015@ID2090 at 10:41:01 AM
./script.sh.x
hint: Using 'master' as the name for the initial branch. This default branch name
hint: is subject to change. To configure the initial branch name to use in all
hint: of your new repositories, which will suppress this warning, call:
hint:
hint: git config --global init.defaultBranch <name>
hint:
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this command:
hint:
hint: git branch -m <name>
Initialized empty Git repository in /var/home/Jan24/me23b015/assignment_6/.git/
tr: warning: an unescaped backslash at end of string is not portable

~/assignment_6 on heads/branch_2 ? ..... ✓ < with me23b015@ID2090 at 10:41:04 AM
ll
drwxr x x me23b015 14 May 10:41 .git
.rw r r 6.3k me23b015 11 May 16:00 ascii.txt
.rw r r 151k me23b015 11 May 16:00 Lena.png
.rw r r 232k me23b015 11 May 16:00 Noisy_Lena.png
.rw r r 6.9k me23b015 14 May 10:41 question_4.cpp
.rwxr x x 19k me23b015 14 May 10:36 script.sh.x
.rwxr x x 20k me23b015 11 May 16:00 script.sh.x.bk
```

Figure 1.1: Running the script

Upon running the script, we get a git repository with one tracked file "question_4.cpp".

1.2.2 Searching for the pokemon



```
(me23b015) 10.24.8.242 — Konsole
~/assignment_6 on heads/branch_2 ? ..... ✓ < with me23b015@ID2090 at 10:42:55 AM
git log --all --decorate --oneline --graph
* c497a0b (branch_3) Commit #com+1
* 2d85439 Commit #com+1
* 1a5f4a1 Commit #com+1
* ca59023 (HEAD, branch_2) Commit #com+1
* f634c2a Commit #com+1
* 056fa10 (branch_1) Commit #1

~/assignment_6 on heads/branch_2 ? ..... ✓ < with me23b015@ID2090 at 10:43:36 AM
head question_4.cpp -n5
#include <iostream>
int main(){
    std::cout << "Congrats, You caught Arceus with the Master Ball. It is trying to come out." << std::end
l << "Freeze it by merging this branch with this node, in MASTER branch.";

    std::cout << "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa\n"
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa\n"
```

Figure 1.2: Searching for the pokemon

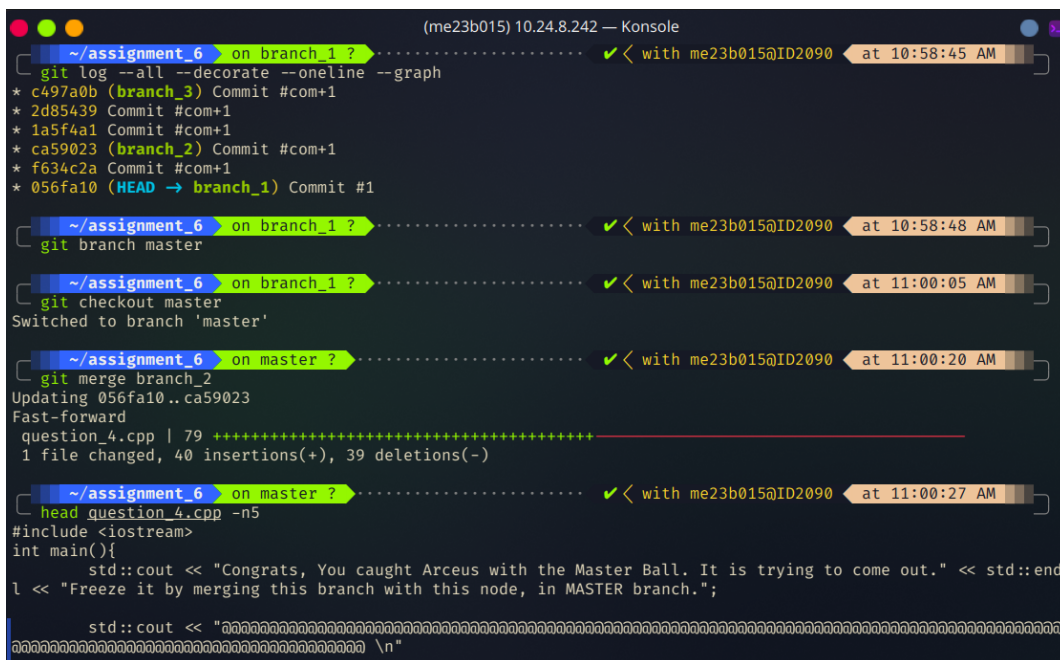
Using `git log --all --decorate --oneline --graph`, we can see that there are a total of 6 commits in the repository. We can use `git checkout <commit-hash>` in order to go through each commit and find the pokemon.

I found the pokemon in branch_2.

1.2.3 Merging with master

We need to merge branch_2 with the master branch. In order to do that, I will first create the master branch using the `git branch master`. Then, I will checkout to the master using `git checkout master`.

Finally, I will merge the branch_2 with the master using `git merge branch_2`.



```
(me23b015) 10.24.8.242 — Konsole
~/assignment_6 on branch_1 ? ..... ✓ < with me23b015@ID2090 at 10:58:45 AM
git log --all --decorate --oneline --graph
* c497a0b (branch_3) Commit #com+1
* 2d85439 Commit #com+1
* 1a5f4a1 Commit #com+1
* ca59023 (branch_2) Commit #com+1
* f634c2a Commit #com+1
* 056fa10 (HEAD -> branch_1) Commit #1

~/assignment_6 on branch_1 ? ..... ✓ < with me23b015@ID2090 at 10:58:48 AM
git branch master

~/assignment_6 on branch_1 ? ..... ✓ < with me23b015@ID2090 at 11:00:05 AM
git checkout master
Switched to branch 'master'

~/assignment_6 on master ? ..... ✓ < with me23b015@ID2090 at 11:00:20 AM
git merge branch_2
Updating 056fa10..ca59023
Fast-forward
 question_4.cpp | 79 ++++++
 1 file changed, 40 insertions(+), 39 deletions(-)

~/assignment_6 on master ? ..... ✓ < with me23b015@ID2090 at 11:00:27 AM
head question_4.cpp -n5
#include <iostream>
int main(){
    std::cout << "Congrats, You caught Arceus with the Master Ball. It is trying to come out." << std::end
l << "Freeze it by merging this branch with this node, in MASTER branch.";

    std::cout << "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa\n"
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa\n"
```

Figure 1.3: Merging with master

1.3 Observations

Here are a few things that I observed, while working on the Task

- In my case, after running the script, I was already in `branch_2`. Therefore, I didn't need to checkout to other branches.
- This was a pretty interesting task. I learnt the basics of git and how to go through the commits in a repository. I also learnt about how to merge branches in git.

Task 2

Image Processing

2.1 Introduction

In this task, we have to do some basic image processing, using kernel operations, or in better words, convolutions.

2.2 Some Kernels

2.2.1 Brightness Change

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

This kernel will increase the brightness of the image. It is a simple kernel, which will multiply the pixel value by 2, and is capped at 255.

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Similarly, we can create a kernel to decrease the brightness of the image.



(a) Brightened Image

(b) Original Image

(c) Darkened Image

Figure 2.1: Brightness Change

2.2.2 Edge Detection

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

This kernel works by emphasizing the current pixel value and subtracting the values of the surrounding pixels. This will help in detecting the edges in the image, as the value will be high where there is a sudden change in the pixel values.



(a) Original Image

(b) Edge Detected Image

2.2.3 Blurring

There are many types of blurring. One of them is box blurring, where the pixel value is the average of the surrounding pixels. This results in smoothing of the image. In this case, equal weights are given to all the surrounding pixels.

$$\frac{1}{9} \times \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$



(a) Original Image



(b) Box Blurred Image

Another method of blurring is the Gaussian Blur. In this case, the weights are given according to the Gaussian distribution, that is, a higher weight is given to the center pixel, and the weights decrease as we move away from the center. This results in a more natural blur.

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



(a) Original Image



(b) Gaussian Blurred Image

2.2.4 Sharpening

Another useful kernel, is the sharpening kernel. This kernel will emphasize the edges in the image, and make the image look sharper.

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$



(a) Original Image

(b) Sharpened Image

2.3 Code

The convolution was done using `octave`, in a Jupyter environment. The code is as follows:

2.3.1 Reading the image

Listing 2.1: Reading the image

```
image = imread('Lena.png');  
imshow(image);
```

2.3.2 Defining the Kernels

The following code defines the kernels [1] that were used in the task.

Listing 2.2: Defining the Kernels

```
brighten = [0, 0, 0; 0, 2, 0; 0, 0, 0];  
darken = [0, 0, 0; 0, 0.5, 0; 0, 0, 0];  
box_blur = [1, 1, 1; 1, 1, 1; 1, 1, 1] / 9;  
gaussian_blur_3x3 = [1, 2, 1; 2, 4, 2; 1, 2, 1] / 16;
```

```

gaussian_blur_5x5 = [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] / 256;
unsharp_masking_5x5 = [1, 4, 6, 4, 1; 4, 16, 24, 16, 4; 6, 24, -476,
    24, 6; 4, 16, 24, 16, 4; 1, 4, 6, 4, 1] / -256;
sharpen = [0, -1, 0; -1, 5, -1; 0, -1, 0];
outline = [-1, -1, -1; -1, 8, -1; -1, -1, -1];

```

2.3.3 Defining the Convolution Function

A function `convolution` is defined, whose parameters are the image and the kernel. The function returns the convoluted image.

Listing 2.3: The Convolution Function

```

function[output] = convolution(image, kernel)
    [row, col] = size(image);
    [k_row, k_col] = size(kernel);

    delta = (k_row - 1) / 2;

    padded_image = zeros(row + 2 * delta, col + 2 * delta);
    padded_image(delta + 1:row + delta, delta + 1:col + delta) =
        image;

    output = zeros(row, col);

    for i = 1+delta:col+delta
        for j = 1+delta:row+delta
            sub_matrix = padded_image(i-delta:i+delta, j-delta:j+
                delta);
            output(i-delta, j-delta) = sum(sum(sub_matrix .* kernel))
                ;
        end
    end
    output = uint8(output);
end

```

2.3.4 Generating the images

Finally, the convolved images are generated using the `convolution` function, and are saved as images.

```

brightened_image = convolution(image, brighten);
darkened_image = convolution(image, darken);
box_blurred_image = convolution(image, box_blur);
gaussian_blurred_3x3_image = convolution(image, gaussian_blur_3x3);
gaussian_blurred_5x5_image = convolution(image, gaussian_blur_5x5);

```

```
unsharp_masking_5x5_image = convolution(image, unsharp_masking_5x5);
sharpened_image = convolution(image, sharpen);
outlined_image = convolution(image, outline);

imwrite(brightened_image, 'brightened_image.png');
imwrite(darkened_image, 'darkened_image.png');
imwrite(box_blurred_image, 'box_blurred_image.png');
imwrite(gaussian_blurred_3x3_image, 'gaussian_blurred_3x3_image.png')
;
imwrite(gaussian_blurred_5x5_image, 'gaussian_blurred_5x5_image.png')
;
imwrite(unsharp_masking_5x5_image, 'unsharp_masking_5x5_image.png');
imwrite(sharpened_image, 'sharpened_image.png');
imwrite(outlined_image, 'outlined_image.png');
```

2.3.5 Denoising

To remove noise in the given image using kernel operations:

The easiest way to remove noise is to use a **box blur** kernel. This will average the pixel values and remove some of the noise. However, it also results in loss of detail in the actual image.



(a) Original Noisy Image



(b) Denoised Image (Gaussian Blur)



(a) Original Noisy Image

(b) Box Blur 1

(c) Box Blur 2



(d) Box Blur 3



(e) Box Blur 4

Figure 2.7: Denoising using multiple Box Blurs

This works because, blurring kernels tends to smoothen the image, and in this process, it ends up reducing the noise.

```
t0 = convolution(noised_image, gaussian_blur_5x5);
imwrite(t0, 'gaussian_blur_denoised_image.png');
```

```
t0=convolution(noised_image, box_blur);
imwrite(t0, 'box_blur1_denoised_image.png');
t0=convolution(t0, box_blur);
imwrite(t0, 'box_blur2_denoised_image.png');
t0=convolution(t0, box_blur);
imwrite(t0, 'box_blur3_denoised_image.png');
t0=convolution(t0, box_blur);
imwrite(t0, 'box_blur4_denoised_image.png');
```

In order to [denoise the image using Fourier Transform](#), we can use the following code:

```
[row, col] = size(noised_image);

F = fft2(double(noised_image));
F_shift = fftshift(F);
```

```

mask = zeros(row, col);
center = [row/2, col/2];
D0 = 30;
for i = 1:row/2
    for j = 1:col/2
        dist = sqrt((i-center(1))^2 + (j-center(2))^2);
        mask(i, j) = exp(-(dist^2) / (2*(D0^2)));
        mask(row-i+1, j) = mask(i, j);
        mask(i, col-j+1) = mask(i, j);
        mask(row-i+1, col-j+1) = mask(i, j);
    end
end

F_shift = F_shift .* mask;

F = ifftshift(F_shift);

noised_image_denoised = ifft2(F);
noised_image_denoised = real(noised_image_denoised);

imshow((noised_image_denoised));
imwrite(uint8(noised_image_denoised), 'fft_denoised_image.png');

```



(a) Original Noisy Image



(b) Denoised Image (Fourier Transform)

2.4 Observations

- Blurring is a simple way to denoise images, but it also results in loss of detail. It is suitable for images where the noise is random and not very large. Therefore, it is suitable for gaussian noise and salt and pepper noise.

- Denoising with Fourier Transform is much [more suitable for images with periodic noise](#), as it can be used to remove specific frequencies.
- The mask used in the Fourier Transform code is a Gaussian filter, which is used to remove the high frequency noise. For cutoff frequency $D_0 = 30$, the mask is shown here:

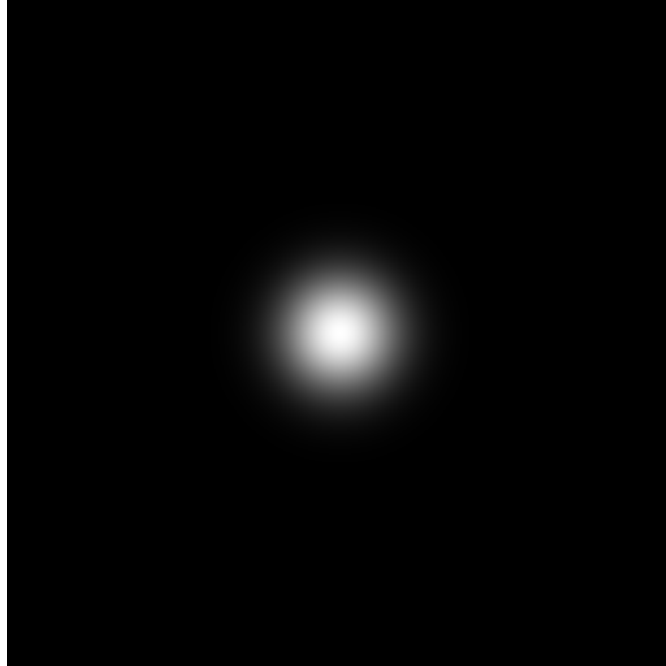


Figure 2.9: Mask used in Fourier Transform Denoising

- For the boundary pixels, I added the necessary padding, so that the convolved image has the same dimensions as the original image.

2.5 Conclusion

- I learnt a lot about image processing and how to use kernels to perform operations on images. I also learnt about the Fourier Transform and how it can be used to denoise images.

Bibliography

- [1] Wikipedia contributors. *Kernel (image processing)* — *Wikipedia, The Free Encyclopedia*. [Online; accessed 16-May-2024]. 2023. URL: [https://en.wikipedia.org/w/index.php?title=Kernel_\(image_processing\)&oldid=1180652863](https://en.wikipedia.org/w/index.php?title=Kernel_(image_processing)&oldid=1180652863).