Functional programming based image processing

Using functional paradigms and design patterns for easier and intuitive image processing

Archit Gupta
SCSE
Galgotias University
Greater Noida, India
guptaarchit137@gmail.com

Ayush Sharma
SCSE
Galgotias University
Greater Noida, India
kumarsharma908765@gmail.com

Mr. Arunendra Mani Tripathi
SCSE
Galgotias University
Greater Noida, India
arunendra.tripathi@galgotiasuniversity.edu.in

***Abstract*— In computer science, functional programming is a programming paradigm in which programs are built by integrating tasks. A declarative paradigm where functional descriptions are trees that map the values to other values, instead of a sequence of required statements that update the state of the system.**

The trend of Image processing has been shifting towards more mobile, fast and convenient methods for rapid processing.

The design and structure of functional programming can be applied to an image editing application with optimal structure integrity and with greater convenience and ease to the end user.

In this paper,we aim at deriving an understanding of the functional programming paradigm, particularly in the case of an image editing SaaS (Software as a Service) application.

Archit Gupta Ayush Sharma Mr. Arunendra Mani Tripathi

Keywords— functional programming, monoids, image processing, computer science, logic

I. Introduction

Abstract-Functional programming (FP) is a programming paradigm where the evaluation of mathematical functions is a major block of software creation.[1] The FP language is suitable for parallel processing and concurrent programming. Many different FP languages and programming styles have been proposed and investigated over the years.

In computer science, functional programming is a programming paradigm in which programs are built by integrating tasks. A declarative paradigm where functional descriptions are trees that map the values to other values, instead of a sequence of required statements that update the state of the system.

In the functional programming paradigm, functions are treated as first-class citizens[2], which means they can be tied to names (including local identifiers).

This allows programs to be written in an expressive and integrated style, where small tasks are grouped together in a modular way.

Recently, various programming languages and frameworks have started integrating declarative and functional programming patterns in their ecosystem for better integration in existing codebase, making it easy for software developers to get onboard with the functional programming paradigm.

The same patterns and approach to logic can also be applied to image processing, wherein an image can be treated as data and one single process or filter can be treated as a function.

II. Introduction to Programming Trends

There are many types of Programming Trends some of them are listed below –

A. *Functional programming paradigm*

Functional programming is a programming paradigm that seeks to bind everything in a purely mathematical functional style.

It is a declarative programming style. Its focus is on "what to solve" and in the imperative style it is on "how to solve".

It uses an expression instead of a statement. The expression is evaluated to generate a value, the statement is executed and the variable is assigned.

B. *Functional programming trends*

Functional programming has been gradually gaining traction in the software engineering and development community from as early as the late 1960s, and has been getting more and more prevalent in the engineering ecosystem.

Another reason for the growth is the rapid influx of various trends in the software ecosystem, which saw the creation, adoption and in many cases, discarding of many such trends.

Functional programming, however, has proved itself useful enough to be used in many production ready projects as well as hobby.

C. *Functional programming in imperative programming languages*

Many imperative programming have also added features for a more functional programming-eque style. Some of them are:

- **Python -** Lambda functions

```
triple = lambda x: x * 3
```

- **JavaScript(es6) -** Arrow functions

```
let triple = x => x * 3
```

III. Functional design patterns in image processing

Some of the functional design patterns in image processing are as follows –

A. *Functional Programming Patterns*

By design, functional programming, functional programming languages and their components follow a set of rules. Some of these roots are from group and category theory, and some from programming theory.

Some of the concepts relevant to the subject matter are:

- **Function composition -** combining smaller functions to create bigger and complex functions
- **No Side Effects -** functions do not manipulate existing state, instead create new state
- **Pure (idempotent) functions -** Functions do not alter state, and are predictable
- **Immutable Data -** Data is regarded as an immutable state
- **Function Currying -** Single argument functions, being called in series.

B. *Relevance to Image Processing*

Image processing is a way to perform some operations on an image to get a modified image or extract some useful information from the image.

This is a type of signal processing where the input may be an image and the output may be an image or a property / function associated with that image.

Image processing is one of the fastest growing technologies today. It is also the focus of research in the fields of engineering and computer science.

Image processing basically consists of the following three steps –

- The result will be a report based on the modified image or image analysis.
- Import the image via some image capture tool.
- Image analysis and manipulation.

The process of image processing can be easily mapped to a functional process, as both are observed to have overlapping concepts. Image processing can be broken down into smaller parts, wherein each smaller part can be mapped to a function.

Here, each function can be composed over one another to form the original process.

ex: If an image of a tourist attraction needs to be turned grayscale and tourists are to be removed from the image, then both of these components can be regarded as two different functions that can be applied to the image.

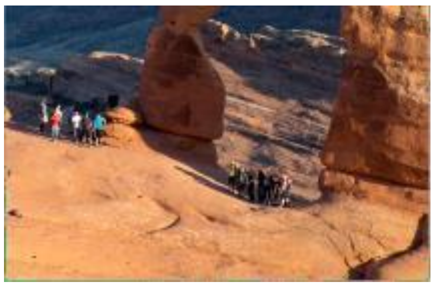

Fig 1. Initial Colored image with tourists

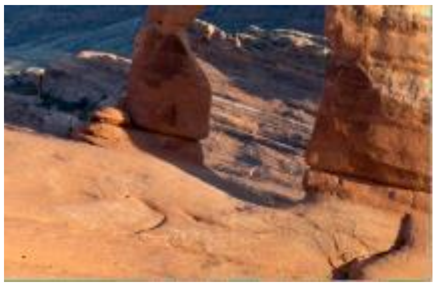

Fig 2. Intermediate Colored image with no tourists

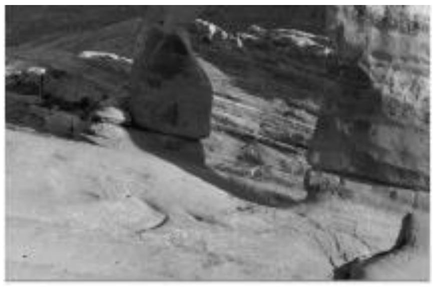

Fig 3. Final Grayscale Image with no tourists

Flow diagram of such a process is as follows:

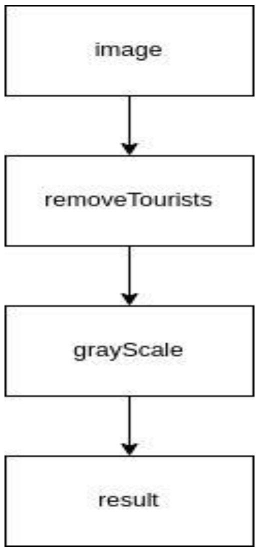

Fig4. flow diagram for tourist removal and grayscaling

Some of the processes / implementations that resemble functional programming patterns are:

- **Pure functions -** A lot of complex image processing pipelines can be broken down into smaller atomic functions / filters. Each of these filters can be regarded as a singular function being applied to an image without any intermediate state.
- **Shared Data -** The output of one filter / function can be used as input of another, thus creating complex processes through small building parts.
- **Function Composition -** Since these functions have a modular type signature, a number of functions can be composed to create complex filters and effects.
- **Modularity and ease of debugging -** Since these processes are just a series of smaller filters / functions, it is easier and more convenient to change, debug and reverse engineer these complex effects.

IV. IMPLEMENTATION OF FUNCTIONAL IMAGE PROCESSING

Some Functional image processing implementation examples –

A. *Cloud based image editing*

The Functional Approach to Image processing and editing can be implemented using a basic image editor SaaS (Software as a Service App), which can be hosted over a server.

A SasS (Software as a Service) is a better option for lightweight image editing as it can be accessed over the cloud easily, making it more accessible to new and experienced users alike

This can make such new and experimental features accessible and intuitive to newer users.

B. *Software based image editing*

Various Image Editing softwares that constitute large market shares are primarily desktop applications.

These applications are mainly used by professionals under heavy workloads. Here, functional programming can come into handy in two ways:

1. Many of the processes and flows that these professionals use can be simplified using the serialized and representative view of such processes.
2. The Advantages of functional programming patterns can greatly improve the quality of code, easing debugging, and potentially decreasing lines of code.

C. *Representational view of functional image processing pipelines*

An implementation of such an application can be made through a flow diagram, which can be easily implemented in a web app.

In such an implementation, the individual blocks of a flow diagram can represent each function / filter being applied to the image.

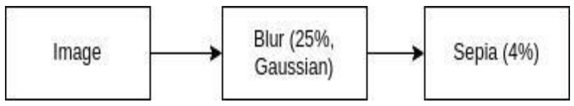

Fig 5. example of applying filters on images

D. *Function Currying for filter constraints / strength*

Since Each block is an atomic, single responsibility and single argument function, it is difficult to provide strengths of a particular filter (ex: blur intensity). [3]

Here, another component of functional programming - functional currying can be used.

A function takes the intensity of the filter and returns a function that applies the required amount of filter to an image.

These processes are just a series of smaller filters / functions, it is easier and more convenient to change, debug and reverse engineer these complex effects.

Below pseudocode shows an example –

```
image = blur(35)("gaussian")(image)

const blur = (amt) => {

    return (type) => {
      return (image) => {
        if(type == "gaussian") {
              // ...  code  with  gaussian  blur
  application returning new instance of image
        }
        else {
            // ... code with some other blur
  returning new instance of image
        }
      }
    }
  }
```

Instead of -

```
blur(image, "gaussian", 25)

Image blur(Image &image, string type, int
amount) {
        // ... applying blur on the reference
of the image object in the argument
}
```

This example shows the power of functional programming and how functional programming fits perfectly for scenarios like image processing.

V. Analysis of functional approach

Advantages and disadvantages of functional approach –

A. *Advantages of functional approach*

We've discussed the theory and implementation of this approach. However, the various advantages of such approaches are numerous.

Some of the most prominent ones are:

- **Predictable Behavior -** Such processes are predictable, i.e. we can be sure of the outcome. Such properties can be useful for users / designers with a specific editing signature.[2]

- **Easily changeable for the user -** such a modular approach is very convenient and easy for users to play with, ensuring larger pipelines can easily be understood and modified.

- **Easily debug -** Functional programming, by design, is easily debuggable.

  Hence, for such complex applications (image editing applications can grow to be

- **Data is immutable -** The functions in functional programming are unchangeable over time.

  We can add more functions without any hindrance. We can also create a new Data structure then can't change the already existing one in Functional programming.

- **Modularity -** This is one of the most important characteristics of functional programming which helps us to divide a large scale project into simpler modules.

  Every module thus created can be worked on independently and thus reduces the time that has to be spent on testing and debugging.

B. *Disadvantages of functional approach*

So far, the functional approach may seem ideal, but it may have some drawbacks depending upon the use case. It is important to know that even if the existence of such drawbacks are true, the advantages far outweigh them.

Regardless, it is critical to shed light on some of such disadvantages:

- **Immutability and Performance -** In Functional Programming, since the update of variables is not allowed and there isn't any state.

  The performance of the project is reduced and the process becomes memory intensive.

  The issue happens when we are dealing with a large size image and we need to perform a duplication operation even if it is only modifying a relatively smaller part of the code.

- **Inability to apply effects to some parts of image -** in some cases, one may need to apply an effect to one part on an image (ex: remove one particular person from image, crop out one animal from a wildlife image).

  Such cases are very difficult to implement with the functional approach, as it is made for applying effects to complete images.

- **Low prevalence of functional programming in industry -** Although functional programming is on the rise, it is not yet widely used in the industry, leading to less talent for functional programming in general.

VI. References

[1] Abdullah Khanfor, Ye Yang “An Overview of Practical Impacts of Functional Programming”, 2017 24th Asia-Pacific Software Engineering Conference Workshops (APSECW)

[2] Tomas Petricek with “Functional Programming for the real world” ©Manning Publications Co

[3] Robert C. Martin “Clean Code: A Handbook of Agile Software Craftsmanship” Stack Overflow 2021, Stack Overflow Annual Developer Survey 2021, accessed May 11, 2022.

[4] Nicole S. Young ,Remove Tourists from Images Using Photoshop, Photo Focus, Accessed May 11, 2022

[5] Hugles, J. (1989). Why functional Programming matters. The computer journal, 32(2), 98-107.

[6] Elliot, C. (2017). Graphics programming in haskell. Communications of the ACM, 60(1), 90-99.

[7] Meijer, E. (2004). Functional Programming for multimedia applications. Communications of the ACM, 47(11), 49-55.

[8] Bird, R. , & Wadler, P.(1998). Introduction to functional programming. Prentice Hall.

[9] Gallo, O, Grigoras, P & Ramponi, G. (2010). Functional image processing with scala and image Journal of real – time image processing, 5(4), 199-208.

[10] Hudak, P(1989). Conception, evolution, and application of functional programming languages. ACM Computing surveys (CSUR), 21(3).359-411.