Use of Parallel Computing for Images Processing

Roksolana Kovtko
The Department of System Design
Faculty of Electronics and Computer Technologies
Ivan Franko National University of Lviv
Lviv, Ukraine
rsulymko@ukr.net

Serhij Grydzhan
The Department of System Design
Faculty of Electronics and Computer Technologies
Ivan Franko National University of Lviv
Lviv, Ukraine
serhiy.grydzhan@gmail.com

Roman Shuvar
The Department of System Design
Faculty of Electronics and Computer Technologies
Ivan Franko National University of Lviv
Lviv, Ukraine
shuwar@electronics.lnu.edu.ua

Andriy Prodyvus
High-performance Computing Systems Laboratory
Faculty of Electronics and Computer Technologies
Ivan Franko National University of Lviv
Lviv, Ukraine
andriy.produvys@lnu.edu.ua

Abstract—**The method of parallel program realisation for raster image processing in a mobile application is considered. The principle of developing an application in the iOS system is proposed. Developed iOS application for raster image processing. The influence of multithreading on the speed of data processing is investigated.**

Index Terms—**Image processing; parallel computing; Swift; queues; Dispatch Queues; multithreading; Xcode; raster image; iOS-system.**

I. INTRODUCTION

At the present stage of the development of information technologies, which are used to solve a wide range of problems, an important issue is the optimization of computations. To date, the theoretical limit of the processor speed has been practically achieved, and therefore the increase in power is achieved by increasing the number of cores. Therefore, the issue of parallelism becomes urgent, that is, the possibility of components of the program act independently, performed in parallel.

Parallel computing — is a way of organizing computer calculations, in which computer programs are developed as a set of computing processes that interact while working in parallel (simultaneously). The main difficulty in designing parallel programs is to ensure the correct sequence of interactions between different computing processes, as well as the coordination of resources that are split between processes.

Significant distribution of mobile electronic devices on different operating platforms necessitates the adaptation of existing and writing new applications to these platforms. The main part of all the content contained on mobile devices are photo and video files.

Digital filters are used for image to improve the quality and informativeness. The popularity of using photo content is causes to the development of new, modern mobile applications that will be able to process this content quickly and efficiently.

Most modern mobile devices provide the ability to use multithreading technology when creating mobile applications. Several computing elements (processes or cores) should be used to improve the speed of the mobile application. Multithreading is the reproduction in several copies of some hardware structure, which allows you to achieve increased productivity through the simultaneous operation of all elements of the structure that perform various parts of this task.

The paper proposes a methodology for developing a mobile application for the efficient processing of images using the principles of parallel computing.

II. REVIEW OF TECHNOLOGIES

The aim of the work is to study the principles of mobile application development on the iOS platform. The presence of several cores in today's mobile devices makes it possible to use parallel computing in mobile applications. The paper investigates the impact of the use of multiprocessing technology on the timing of the tasks of the mobile application.

To create an IOS application, the Xcode environment is selected. The project used the MVVM template (Model-View-ViewModel). This template is a fairly simple but effective suite for designing and implementing applications. It allows you to split all files, behavior patterns and data into three conditional groups (behavior patterns, visual part, and controllers).

The mobile application is written in the Swift programming language. This is a multi-paradigm compiled programming language developed by Apple to coexist with Objective C and be more stable to false code. Swift inherits the best elements of

C and Objective C, so its syntax is familiar to those for developer familiar with these languages. At the same time, Swift is characterized by the use of automatic allocation of memory and control overflow of variables and arrays, which greatly increases the reliability and security of the code. Swift programs are compiled into a machine code that provides high performance.

Also, the project uses the CocoaPods software interface using Parallel Computing technology. Cocoa is an object-oriented Application Programming Interface (API) for Apple's Mac OS X. Cocoa applications are applications that are written using the Cocoa software environment and usually have a distinctive look, because this environment greatly simplifies the support of the Apple Human Interface Guidelines (Apple Human Interface Guidelines). Cocoa consists of two libraries of objects Swift and Objective-C, called frameworks.

When developing a mobile application in the language Swift is best to use the technology of parallel computing GCD (Grand Central Dispatch).

Grand Central Dispatch (GCD) is a queuing application programming interface (API) that allows you to lock in work areas.

In other words, the locks containing the task to be executed can be added to the queue, which will execute them using a sequence of streams (sequentially), or in parallel, depending on the configuration of the queue. But regardless of the type of queue, the task will always be launched in the order "first came - the first went", that is, the task will always be run in accordance with the order of addition. The completion order will depend on the duration of each task.

This is a common template that can be found in almost all modern execution environments that handle parallel computing. The workflow space is easier to manage, check, and control than individual and unrelated threads.

The GCD allows you to create custom queues, as well as provide access to some queues defined by the system.

To create a basic serial line that will execute your loops consistently, you simply need to provide a line label that identifies it, it is usually advisable to use the reverse order domain prefix to facilitate tracking the queue owner in tracing the stack.

let serialQueue = DispatchQueue (label: "com.uraimo.Serial1")

let concurrentQueue = DispatchQueue(label: "com.uraimo.Concurrent1", attributes: .concurren)

The second queue we created is parallel, which means that the queue will use all available streams in its base workspase of threads when performing the tasks it contains.

From the DispatchQueue object the default queues can be obtained:

let mainQueue = DispatchQueue.main

let globalDefault = DispatchQueue.global()

The main queue - is the sequential main queue that handles the main event cycle for graphic applications on iOS or macOS, reacting to events and updating the user interface. As we know, each change to the user interface must be executed in this queue, and each long operation that is executed in this thread will make the user interface much slower.

The runtime environment also provides access to other global queues with different priorities that can be identified by the Quality of Service parameter (Qos). All priority levels are announced in the DispatchQoS class from top to bottom. It's important to note that on mobile devices that provide low-power mode, the background queues will be stopped when the battery is low.

To get a certain global queue by default, you must use the global (qos :) prefix that defines the priority you want:

let backgroundQueue = DispatchQueue.global (qos: .background)

III. DEVELOPMENT OF APPLICATION FOR IMAGE PROCESSING

In the Xcode environment, the appearance of the mobile application "ColorEditor" was developed. The user interface was created using the storyboard tool and the MVVM design model (Model-View-ViewModel). To ensure adaptive design, that is, proportional size and correct placement of interface elements at different screen resolutions, all elements have the "Constraints" setting. The app developed supports all iOS devices (iPad and iPhone) with different screen resolutions and with a version of the system no below iOS 11.3.4. The ability to select an image from the gallery or to get a photo from the camera of a mobile device is implemented.

The created mobile application allows the user to download an image, apply software digital filters to it, view changes and save the image.

It is suggested to download image pixels after download. In order to realize the multithreading technology, it is proposed to divide the image into several informative blocks in order to further process each block in a separate stream. The number of parts to which the image is divided must be multiple times to the number of processor cores on the device.

Example of reading code for each image pixel:

```
let queue = DispatchQueue.global(qos: .utility)
// initialize the parallel workspace
        queue.async        // run workspace asynchronously
        {
        image.image = UIImage(data: data)
        let pHeight = image.pixels.height
        let pWidth = image.pixels.width    // read the size of
the image
      if let data = try? Data(contentsOf: imageURL)
        {
                for int i=1..pHeight
                for int j=1..pWidth
        {
                        let pixArr[i,j] = pixelColor;
// write in the array data about each pixel
```

```
                                        }
                                        }
        DispatchQueue.main.async
        {
            print("Show image data")         }
                                         }
```

After uploading an image to it, a black-and-white filter is automatically applied, which can be removed completely or partially using the Splash feature.

You can apply features to the image: grayscale, sepia, brightness, contrast, blur (Gaussian filter), PencilSketch filter.

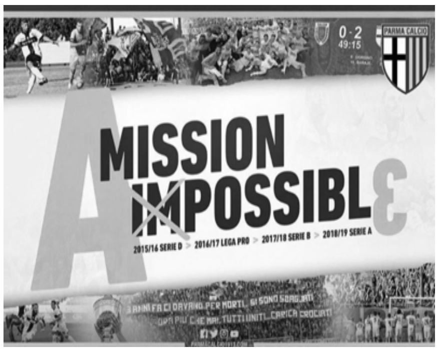
An example of applying a filter for change of brightness to an image. To do this, you need to change the values for each pixel:

```
let queue = DispatchQueue.global(qos: .utility)
  queue.async
        {
        image.image = UIImage(data: data)
        let pHeight = image.pixels.height
        let pWidth = image.pixels.width
    if let data = try? Data(contentsOf: imageURL)
        {
                for int i=1..pHeight
                for int j=1..pWidth
        {
                let pixArr[i,j] = pixelColor
        // read in the image the R,G,B-components of each
        pixel
                        }         }
       DispatchQueue.main.async
        {
        for int i=1..pHeight
        for int j=1..pWidth
        {
        pixArr[I,j]= pixelRGB(rElemet*0.3,
gElement*0.59+bElement*0.11)
        // filter application
print("Show image data")
       }
       print("Did download  image data")
        }
```

The images "b" and "c" illustrate examples of applying filters to the original image "a".

MISSION
IMPOSSIBLE
0-2
PARMA CALCIO

a) The initial image



b) Black and white filter image

MISSION
IMPOSSIBLE

c) Image with changed brightness

The example shows the use of multiprocessing technology (DispatchQueue object), where each part of the image is allocated a separate workspace, which is processed in parallel with others.

To investigate the effectiveness of using multithreading technology (reducing the execution time of certain operations), another iOS application was used, and a comparison was made between the time taken for the image to be blurred. The research was conducted on a mobile device iPhone 5 with two processor cores. The running time of the created mobile application is twice as low as running a regular application without the use of multithreading technology.

When using the created application, you should take into account features of the iOS system: the impossibility of processing images of a very large size. When using large sized images, the CPU is heavily loaded and the iOS operating system prevents the program from running, closing it.

CONCLUSIONS

Frequent use of photo content on mobile devices necessitates new applications. Modern mobile devices have multi-core processors. This feature allows developers to use parallel computing technology to create modern and efficient mobile applications. The basic idea behind this technology is to minimize task execution times by allocating loads between multiple computing devices.

With programming language Swift created a modern mobile application for devices on the iOS platform for image processing. The application uses the Grand Central Dispatch (GCD) application interface. His application made it possible to use the technology of parallel computing on the basis of queues.

The created mobile app is correctly displayed on devices with different screen resolutions. An application created for processing user images allows you to:

• upload images (choose from a gallery of a mobile device or receive photo from the camera);

• filter images (apply software digital filters and masks);

• save the image (overwrite existing or create a new file).

The created mobile application uses technologies of multithreading, which speeds up the time of tasks. The image is divided into parts that are processed in parallel on separate processor cores.

The use of parallel computing significantly accelerates the process of any computations. Multithreading technology has a significant effect in developing advanced applications for image processing on mobile devices.

REFERENCES

[1] Patterns for beginners: MVC vs MVP vs MVVM [Electronic resource in Ukrainian]. - Access mode: http://it-ua info/news/2014/03/17/patterni-dlya-novichkv-mvc-vs-mvp-vsmvvm.html.

[2] Model-View-Controller в .Net [Electronic resource]. – Access mode: http://rsdn.ru/article/patterns/ ModelViewPresenter.xml.

[3] Patterns: MVC, MVP and MVVM [Electronic resource in Russian]. – Access mode: http://outcoldman.com/ru/ archive/2010/02/22/паттерны-mvc- mvp-и-mvvm.

[4] Creating MVVM applications with Xamarin and MvvmCross [Electronic resource in Russian]. – Access mode : https://msdn.microsoft.com/ru- ru/magazine/dn 759 442.aspx.

[5] Using the Model-View-ViewModel (MVVM) Template [Electronic resource in Russian]. – Access mode : https://msdn.microsoft.com/ru-ru/library/windows/apps/jj88 3732.aspx.

[6] David Mark, Jack Natting, Kim Topley, Frederick Olsson Swift: Development of Applications in the Xcode Environment for iPhone and iPad using iOS BHV-Petersburg, St. Petersburg, 2016, 100-150 p. in Russian

[7] Khazaryan A.A. Swift language. Self-teacher BHV-Petersburg, Petersburg, 2016, 1-176 p. in Russian