REPORT 02

Name: Pham Ngoc Tam
Teacher: Nguyen Thanh Hoa

I.Report content

1.Project purpose

- Use Pigo-Face Detection implement send image from Client to Server, then Server will response back one of two options:

Option 1: Response image that maked faces by square

Option 2: Reponse JSON data that is coordinates of faces in image

2.Ingredient of source code

- Server.go

```go
func main() {
    http.HandleFunc("/", Server)
    http.ListenAndServe(":8080", nil)
}
func Server(Rep http.ResponseWriter, Req *http.Request) {
    GetMultiFile(Rep, Req)
    Detection(Rep, Req)
}
```

Server will do two main step: Save image upload and Reponse result

+ Step 1: GetMultiFile()

```go
func GetMultiFile(Rep http.ResponseWriter, Req *http.Request) {
    if err := Req.ParseMultipartForm(1024 * 1024 * 10); err != nil {
        http.Error(Rep, err.Error(), http.StatusBadRequest)
        return
    }
    for key, value := range Req.MultipartForm.File {
        if key == "FileUpload" {
            for _, oneFileOfMultiFile := range value {
                saveFileintoServer(oneFileOfMultiFile)
            }
        }
    }
}
func saveFileintoServer(oneFileOfMultiFile *multipart.FileHeader) {
    File, _ := oneFileOfMultiFile.Open()
    FileByte, _ := ioutil.ReadAll(File)
    FiletoSave, _ := ioutil.TempFile("ImageIn", "image-*.jpg")
    FiletoSave.Write(FileByte)
    defer File.Close()
    defer FiletoSave.Close()
}
```

Function will save once image to ImageIn folder with random name

+Step 2: Detection()

```
func Detection(Rep http.ResponseWriter, Req *http.Request) {
    Listfiles, _ := ioutil.ReadDir("ImageIn")
    for _, oneFileOfListfiles := range Listfiles {
        if filepath.Ext(oneFileOfListfiles.Name()) != ".jpg" {
            continue
        }
        filename := oneFileOfListfiles.Name()
        cmd := exec.Command("pigo", "-in", "ImageIn/"+filename, "-
out", "ImageOut/"+filename, "-cf", "cascade/facefinder", "-json")
        err := cmd.Run()
        if err != nil {
            log.Fatal(err)
        }
        os.Remove("ImageIn/" + filename)
        if Req.FormValue("Option") == "1" {
            MakeResponseImage("ImageOut/"+filename, Rep)
        } else {
            MakeResponseJSONData("output.json", Rep)
        }
        os.Remove("ImageOut/" + filename)
    }
}
```

Function will get images sequentially and detect by run command line
"pigo -in <input.jpg> -out <output.jpg> -cf <Path/to/facefinder> -json"
Result will be saved in ImgaeOut folder. And then depend on option, function will choose MakeResponseImage() or MakeResponseJSONdata()

```
func MakeResponseImage(Path string, Rep http.ResponseWriter) {
    FileResponse, _ := os.Open(Path)
    defer FileResponse.Close()
    _, err := io.Copy(Rep, FileResponse)
    if err != nil {
        log.Fatal(err)
    }
}
func MakeResponseJSONData(Path string, Rep http.ResponseWriter) {
    var people []facePosition

    File, _ := os.Open("output.json")
    FileByte, _ := ioutil.ReadAll(File)

    _ = json.Unmarshal(FileByte, &people)

    var out bytes.Buffer
    json.Indent(&out, FileByte, "=", "\t")
    out.WriteTo(Rep)
}
```

- Client.go

+ main function

```go
func main() {
    Client()
}
func Client() {
    var Option string
    inputOption(&Option)
    PathIn := "Image/"
    uploadImage(PathIn, Option)
}
```

Client function get input of user and set up PathIn where image need detect

+ uploadImage function

```go
func uploadImage(Path string, Option string) {
    files, _ := ioutil.ReadDir(Path)
    for _, file := range files {
        if filepath.Ext(file.Name()) != ".jpg" {
            continue
        }
        var bodyRequest bytes.Buffer
        MultiWriter := multipart.NewWriter(&bodyRequest)
        FileImageUp, _ := os.Open(Path + "/" + file.Name())
        defer FileImageUp.Close()
        WriterFile, _ := MultiWriter.CreateFormFile("FileUpload", file.Name())
        _, err := io.Copy(WriterFile, FileImageUp)
        if err != nil {
            log.Fatal(err)
        }
        MultiWriter.WriteField("Option", Option)
        MultiWriter.Close()
        MakeRequest(MultiWriter, bodyRequest, Option)
    }
}
```

uploadImage() reads PathIn variable and make request to upload image to Server

```go
func MakeRequest(MultiWriter *multipart.Writer, bodyRequest bytes.Buffer, Option string) {
    Req, err := http.NewRequest("POST", "http://127.0.0.1:8080/", &bodyRequest)
    if err != nil {
        log.Fatal(err)
    }
    Req.Header.Set("Content-Type", MultiWriter.FormDataContentType())

    ClientObject := &http.Client{}
    Rep, err := ClientObject.Do(Req)
    if err != nil {
        log.Fatal(err)
    }
    defer Rep.Body.Close()
    if Option == "1" {
        Tempfile, _ := ioutil.TempFile("Result", "Image-*.jpg")
        defer Tempfile.Close()
        io.Copy(Tempfile, Rep.Body)
    } else {
        ByteFile, _ := ioutil.ReadAll(Rep.Body)
        fmt.Println(string(ByteFile))
    }
}
```

Depend on Option that function will get Image or JSON data.

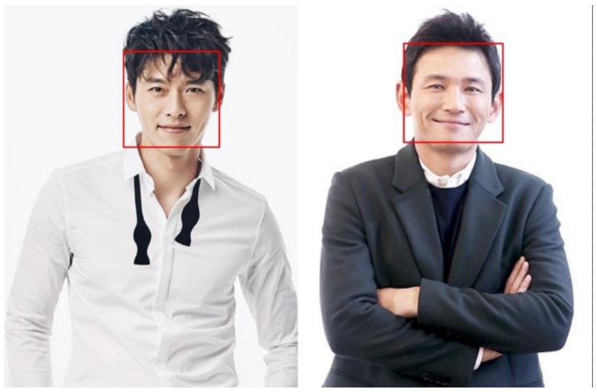
II.Test result

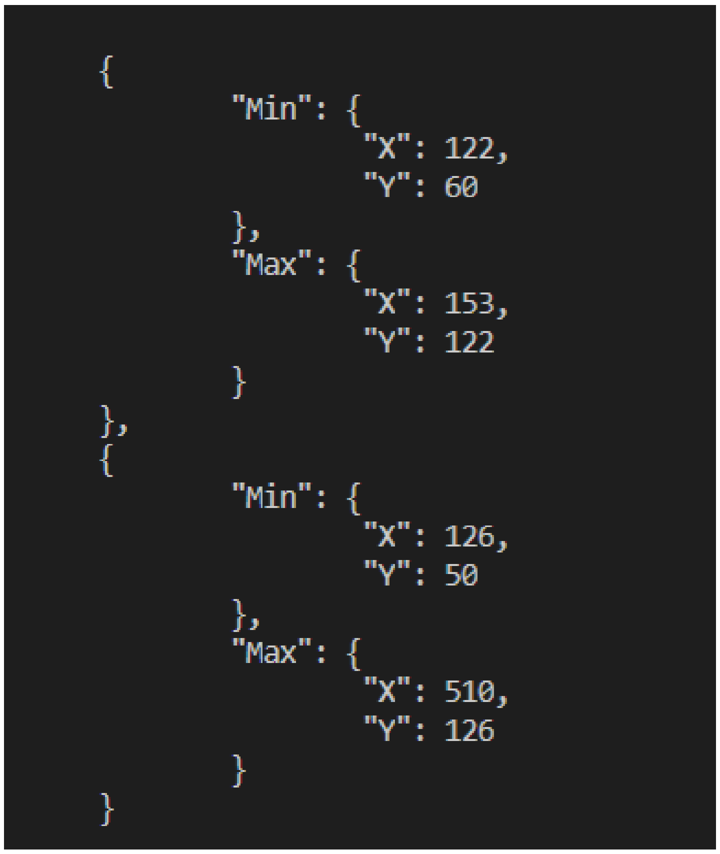
Option = "1"

Option = "2"

Before

After





```
{
        "Min": {
                "X": 122,
                "Y": 60
        },
        "Max": {
                "X": 153,
                "Y": 122
        }
},
{
        "Min": {
                "X": 126,
                "Y": 50
        },
        "Max": {
                "X": 510,
                "Y": 126
        }
}
```

------- END -------