In [ ]:
```
# using Pkg
# Pkg.add("Images")
# Pkg.add("ImageEdgeDetection")
# Pkg.add("ImageCore")
# Pkg.add("ImageFiltering")
# Pkg.add("HTTP")
# Pkg.add("ImageMagick")
# Pkg.add("ImageView")
# Pkg.add("FileIO")
```

In [ ]:
```
include("carving.jl")
```

Out[ ]:
```
carve_image (generic function with 1 method)
```

In [ ]:
```
using Images, ImageEdgeDetection, ImageCore, ImageFiltering
using HTTP, ImageMagick, ImageView
using FileIO
```

```
(julia.exe:3068): GLib-GIO-WARNING **: 05:15:52.122: Unexpectedly, UWP ap
p `51456GRIDGAME.APKForWin11_1.2022.2119.0_x64__g94vhgmp24j9g' (AUMId `51
456GRIDGAME.APKForWin11_g94vhgmp24j9g!App') supports 3 extensions but has
no verbs
```

In [ ]:
```
# load image
function loadImgLocal(img_path)
    img = load(img_path)
    img
end
```

Out[ ]:
```
loadImgLocal (generic function with 1 method)
```

In [ ]:
```
function loadImgOnline(url)
    r = HTTP.get(url)
    buffer = IOBuffer(r.body)
    img = ImageMagick.load(buffer)
    img
end
```

Out[ ]:
```
loadImgOnline (generic function with 1 method)
```

In [ ]:
```
function saveImg(type, path, img_source)
    if type == 1 #jpg
        # save file in JPG format
        save(string(path,"/saved_pic.jpg"), img_source)
    elseif type == 2 #png
        # save file in PNG format
        save(string(path,"/saved_pic.png"), img_source)
    else
        return "not supported file type"
    end
    return "file saved"
end
```

Out[ ]:
```
saveImg (generic function with 1 method)
```

In [ ]:
```
# edge dection function
function detEdge(img_source, scale)
    img_edge = detect_edges(img_source, Canny(spatial_scale=scale))
    img_edge
end
```

Out[ ]:
```
detEdge (generic function with 1 method)
```

In [ ]:
```
#laplacian edge detection
function lapEdge(img_source)
    img_edge = imfilter(img_source, Kernel.Laplacian())
    img_edge
end
```

Out[ ]:
```
lapEdge (generic function with 1 method)
```

In [ ]:
```
function sharpImg(img)
    gaussian_smoothing = 1
    intensity = 1
    # load an image and apply Gaussian smoothing filter
    imgb = imfilter(img, Kernel.gaussian(gaussian_smoothing))
    # convert images to Float to perform mathematical operations
    img_array = Float16.(channelview(img))
    imgb_array = Float16.(channelview(imgb))
    # create a sharpened version of our image and fix values from 0 to 1
    sharpened = img_array .* (1 + intensity) .+ imgb_array .* (-intensity)
    sharpened = max.(sharpened, 0)
    sharpened = min.(sharpened, 1)
    sharpened_image = colorview(RGB, sharpened)
    sharpened_image
end
```

Out[ ]:
```
sharpImg (generic function with 1 method)
```

In [ ]:
```
#controlled by Red, Green, Blue
function imgSaturate(img_source, r_par, g_par, b_par)
    img = copy(img_source)
    img_ch_view = channelview(img) # extract channels
    img_ch_view = permuteddimsview(img_ch_view, (2, 3, 1))
    x_coords = 1:size(img, 2)

    img_ch_view[:, x_coords, 1] = min.(img_ch_view[:, x_coords, 1] .* r_pa
    img_ch_view[:, x_coords, 2] = min.(img_ch_view[:, x_coords, 2] .* g_pa
    img_ch_view[:, x_coords, 3] = min.(img_ch_view[:, x_coords, 3] .* b_pa
    img
end
```

Out[ ]:
```
imgSaturate (generic function with 1 method)
```

In [ ]:
```
# grayscale image
function grayfilter(img_source)
    gray_img = Gray.(img_source)
    gray_img
end
```

Out[ ]:
```
grayfilter (generic function with 1 method)
```

In [ ]:
```
function resizeImg(img_source,perc_H::Float64, perc_W::Float64)
    H = trunc(Int,size(img_source)[1]*perc_H)
    W = trunc(Int,size(img_source)[2]*perc_W)
    resized_image = imresize(img_source, (H, W))
    resized_image
end
```

Out[ ]:
```
resizeImg (generic function with 1 method)
```

In [ ]:
```
# *smart horizontal stretch
function seamcarvImg(img_source, scale)
    sm_img = carve_image(img_source, scale)
    sm_img
end
```

Out[ ]:
```
seamcarvImg (generic function with 1 method)
```

In [ ]:
```
function blurImg(img_source, scale)
    blur_img = imfilter(img_source, Kernel.gaussian(scale))
    blur_img
end
```

Out[ ]:
```
blurImg (generic function with 1 method)
```

In [ ]:
```
#get img from disk
og_img = loadImgLocal("./2.png")

# get img from web
# url = "https://cdn.shopify.com/s/files/1/0337/7469/products/Tropical-Bea
# og_img = loadImgOnline(url)

##image processing functions##
# res = detEdge(og_img, 2)
# res = lapEdge(og_img)
# res = sharpImg(og_img)
res = imgSaturate(og_img, 0.8, 0.8, 0.7)
# res = grayfilter(og_img)
# res = resizeImg(og_img, 0.6,0.4)
# res = seamcarvImg(og_img, 200)
# res = blurImg(og_img, 4)

mosaicview(og_img, res; nrow=1) #comparesion

##image result save to disk##
#saveImg(type=, path=, img_source=) #type=1 is jpg, =2 is png
#saveImg(2,"E:/GWU/6221_Advanced Software Paradigms",res)
```

Out[ ]:

In [ ]:
```
#popout
imshow(res)
```

Out[ ]:
```
Dict{String, Any} with 4 entries:
  "gui"         ⇒ Dict{String, Any}("window"⇒GtkWindowLeaf(name="", par
ent, w…
  "roi"         ⇒ Dict{String, Any}("redraw"⇒50: "map(clim-mapped imag
e, inpu…
  "annotations" ⇒ 3: "input-2" = Dict{UInt64, Any}() Dict{UInt64, Any}
  "clim"        ⇒ 2: "CLim" = CLim{RGB{Float64}}(RGB{Float64}(0.0,0.0,0.
0), RG…
```