Average Face

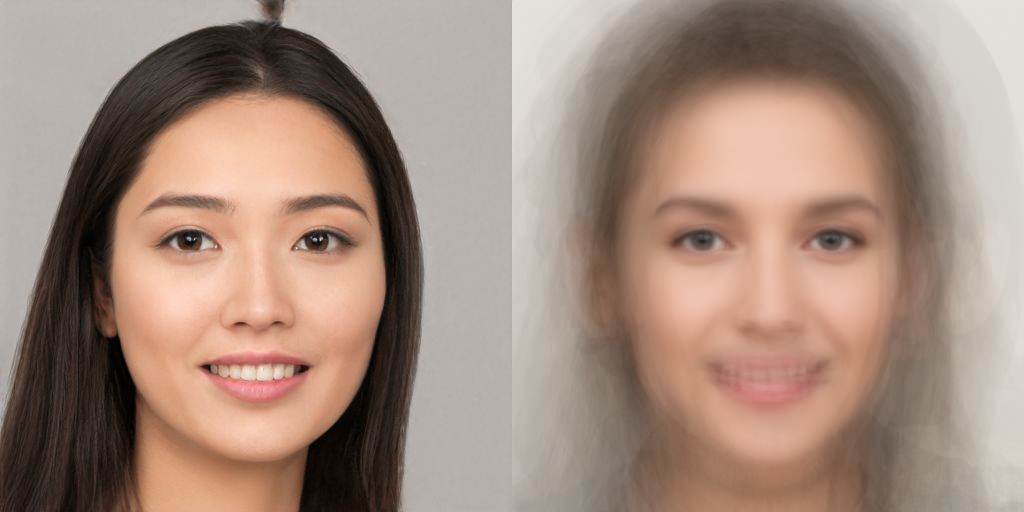
Some computational artists build whole careers based on averaging images. Jason Salavon created a whole series of special moments found [here](http://salavon.com/work/SpecialMoments/). Below are *Newlyweds* and *Kids with Santa,* two examples from the series, used with artist’s permission.



In this assignment we are going to use a data set of 30 faces. These faces are part of 100,000 faces belonging to people that have never existed. They have been generated using machine learning by a team of AI and photography professionals. More information on their work can be found [here](https://generated.photos/). The images used in this assignment have been used with permission by the authors.



Your goal in this assignment is to calculate the average face from these 30 images as seen in the image on the right below.



Start by downloading the averageFace-Template which contains the required data set.

## Steps to complete

Step 1: Let’s load the faces in memory. Inspect the assets folder. Notice there are 30 images with names starting from 0.jpg to 29.jpg. Use a for loop within the preload() function to load all 30 images into the *imgs* array. (Hint: Create a string called *filename* made up from the path to the images, the for-loop index and the file extension. You can use console.log to make sure you’ve built the right *filename* string)

Step 2: Update the createCanvas() line to create a canvas twice the width of the first image in the array, and equal to the first image’s height. Draw the first image on the left of the canvas. If you’ve done things right you should have one of the faces on the left and a grey area of equal size on the right.

Step 3: In the setup() function initialise the *avgImg* variable using the [createGraphics()](https://p5js.org/reference/#/p5/createGraphics) command. Set its size equal to the size of the first image in the array. This way, we have created an empty buffer to save the results of our calculations.

Step 4: We’ll need to access the pixel data of all images in the *imgs* array as well as the *avgImg*. Make sure you call the loadPixels() command on all image data inside the draw() loop.

Step 5: Create a nested for-loop looping over all pixels on the first image in the array. Convert the x and y coordinates from the for-loop to an pixel index value and use that value to set the corresponding pixel in the *avgImg* to red. After exiting the nested for loop, update the pixels of the *avgImg* to let p5js know that the image has had its data changed, and draw the *avgImg* to the right of the existing image. If you’ve done things right, the left side of the canvas should have the face of the first image in the array and the right side should be bright red. Add also a noLoop() at the end of the draw() function as the calculations we are about to do are intense and we only really need to do them once. No need for looping.

Step 6: Inside the nested for loop, create three variables *sumR, sumG, sumB* and initialise them to 0. This is where we are going to store the sum of each channel for that pixel. Create a for-loop just under these variables, looping through all the images in the *imgs* array and for each channel add its value to the corresponding sum variable. Just under this for-loop update each channel in the *avgImg*. (Hint: You’ll need to use the sum variables as well as the size of the *imgs* array.) If you’ve done things right you should be seeing the average image as displayed at the top of the exercise.

Step 7: Can you extend the sketch by implementing the ideas for futher development? HINT: Do not remove the noLoop() from the end of draw(), simply call loop() at the end of the user input functions mentioned below.

## Ideas for further development:

* How would you change the code so that the image drawn on the left is a random face from the array of faces rather than just the first one, with a new random face selected using the keyPressed() function?
* Could you have the pixel values of the second image transition between the randomly selected image and the average image based on the mouseX value? HINT: Use the p5 lerp() function, read the documentation to understand what you need to do.

You only need to submit the final version of your code after completing the steps. The step by step output in the rubric should only serve as a guide to understand what output should have been achieved at each step.

## Marking rubric

Step 1 - [2 point]: Images loaded successfully using a for-loop (check code).

Step 2 - [1 point]: Face appears on the left, grey canvas on the right.

Step 3 - [1 points]: Image initialised correctly within setup() function (check code).

Step 4 - [1 point]: Images are looped over and updatePixels() is called on them.

Step 5 - [2 points]: Face appears on the left, red canvas on the right. Conversion from 2D to 1D coordinates has taken place (check code).

Step 6 - [2 point]: Average image appears on right side of the canvas.

Step 7 - [3 point]: Points awarded based on whether the solutions to the ideas for further development where correctly implemented.

Full marks: Step has been achieved.

Half marks: Reasonable attempt but doesn't quite work.

Zero marks: Not attempted or little achievement.