The World in Dalivision: Crowdsourced Photographic Mosaics

Brett Epps
University of Wisconsin
Madison, WI

http://eppsilon.com/projects/mosaics/

Abstract. "The World in Dalivision," or simply, "Dalivision," is an implementation of an "infinite" photographic mosaic inspired by a Salvador Dalí painting along with other pointillist work like that of Chuck Close. Using concepts derived from art such as luminance matching, Dalivision takes a source photo and converts it into a tiled mosaic, pulling pictures from an online photo-sharing website to produce a unique output image. This article describes a rudimentary mosaic-making method and then delves into the details of this specific implementation and the intriguing engineering problems that arise with this type of project. It concludes with a discussion of potential improvements to the program.

1 Introduction and Motivation

The mosaic is one of the earliest forms of artwork, with the earliest examples dating to the 4th century BCE. Since then, mosaics have become commonplace, decorating the ceilings and walls of buildings, the underwater floors of swimming pools, and even, in a sense, the pixilated screens of computers, televisions, and cell phones. More recently, in large part due to the work of Joseph Francis and Robert Silvers, mosaics comprised of tiny photographs have become popular.

A limitation of most current photographic mosaic software is the library of images available. The user is often prompted to select their own photos for inclusion in the mosaic, and sometimes a library of generic stock photography is also used. However, with the advent of online photo-sharing, many more images are available through the Internet than a user could possibly provide.

This project was conceived as a way to combine the traditional photographic mosaic-making approach with the variety of web photography. The work presented in this article achieves this basic goal, but the project has much potential for improvement and could eventually become a truly novel means for exploring large collections of photos.

2 Background

Most mosaics are created by taking a source image and partitioning it into tiny subdivisions, often called tiles or tesserae. The tiles are then colored in a way that approximates the color of the region of the source image they are replacing. When viewed from a distance, the human eye is unable to perceive the separations between the tiles, causing them to merge together into a representation of the source image. Close up, the tiles of the mosaic form patterns that are beautiful, though somewhat plain to the eye.

In recent decades, however, artists have challenged the idea that a mosaic must have just one layer of interpretation (the image you see from afar). In 1976, Salvador Dalí created a painting titled “Gala Contemplating the Mediterranean Sea” (also known as “Lincoln in Dalivision”), which is comprised mostly of large tile-like blocks [LSKL05]. When examined close-up, the viewer can see Dalí’s wife Gala framed by a blocky, Islamic-style window. But from a distance (or through a camera’s LCD), the blocks are compressed together and become much less defined, and a familiar image reveals itself.

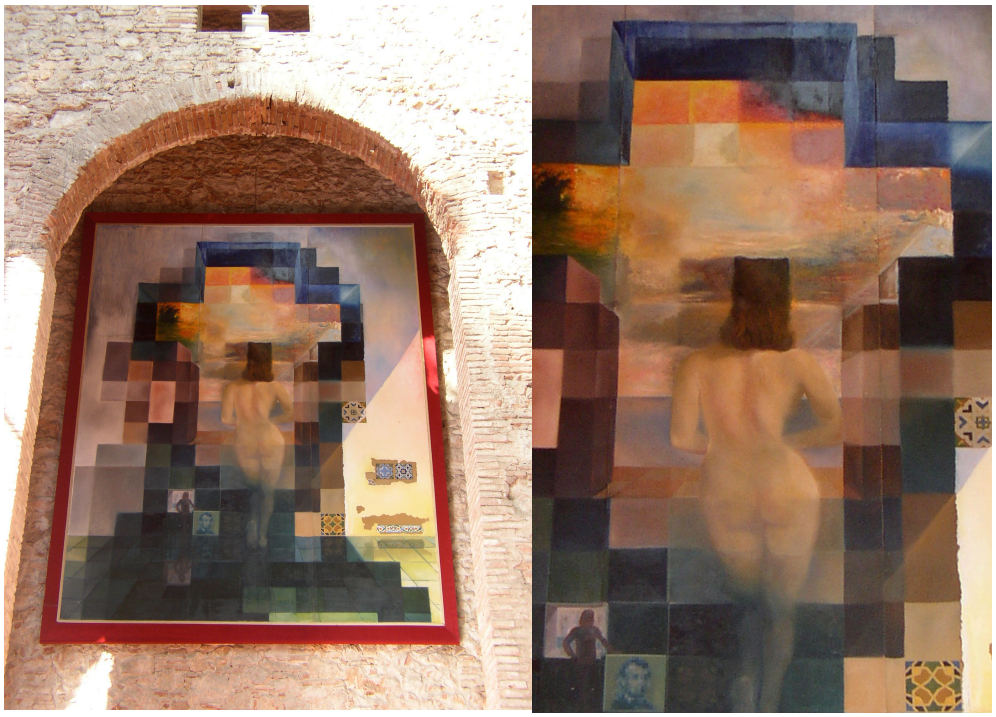

Figure 1: “Lincoln in Dalivision” photographed from afar compared to zoomed in.

Also in the 1970s, Chuck Close became known for his large portrait mosaics. Close would photograph a subject and then create a grid of diagonal tiles, filling each tile with layer upon layer of successively smaller blobs of color. The layers would sometimes alternate in color or luminance or both, but collectively they would build toward some target color. A 2005 paper by Luong, Seth, Klein, and Lawrence [LSKL05]

explores Close's pointillist method much further, developing an algorithm for simulating his technique and creating mosaics that mimic his unique style.

More applicable to this project is their paper's discussion of luminance and its effect on how artwork is perceived. One example they give is Monet's "Impression Sunrise," in which an orange sun rises on a teal background. In a comparison between the original and a version of the painting showing only luminance, they show that while the hues of the orange and teal colors differ significantly, their luminance levels are the same. This "produces perceptual tension between the luminance and color processing pathways of the human visual system," and makes a specific element, the rising sun, stand out. When Luong et al applied their Chuck Close filter to creating photographical mosaics, this concept of isoluminance enhanced their replacement tiles' ability to accurately replicate the color of the region being replaced. The program created for this project, Dalivision, will take advantage of this same property when adjusting web images and fitting them into tiles.

3 Problem Statement

Objectives

This project is comprised of several components and thus has a number of key objectives. The software (dubbed Dalivision) should:

- Take as input any user-specified RGB source image (of any size).
- Create a photographic mosaic – an output image comprised of tiles that contain small photographs.
- Allow the user to adjust the size of tiles used in the output image.
- Allow the user to adjust any post-mosaicking filters that correct the colors in the tiled photographs.
- Allow the user to easily choose a tiled photograph within the mosaic for which a new mosaic should be created, thus giving the impression of an "infinite mosaic."
- Complete all rendering and other user-facing actions in a reasonable amount of time (30 seconds to a minute, if possible).
- Attempt to store as little data as possible, downloading images on-the-fly instead of trying to maintain a gigantic offline pool of images.

Subproblems

These broad objectives break down naturally into several subproblems:

- Mosaic creation
  - What criteria should a photograph meet in order to be a suitable replacement for a region in the source image?

  - How should a photograph be adjusted after placement to better “fit” into its place in the source image?
- Tile photograph fetching
  - From which online collection should tile photographs be fetched?
  - How can tile photograph downloads be done as efficiently as possible?
  - What data about tile photographs should be stored to allow the application to choose a photo for a tile without having ready access to all available photos?

4 Method / Implementation

This section will describe the implementation details of Dalivision. The application has two main components: an indexer that crawls Flickr using its REST API and populates a database with information about available photos, and a mosaic creator that assembles tile photographs into an output image.

Tools, Libraries, and Other Resources

Dalivision is written in Python as an application designed to work with the popular web framework, Django. It uses a PostgreSQL database for storing data about available images and an nginx HTTP server for serving static content (output images). Dalivision uses many Python modules, but the only non-standard one is PIL, the Python Imaging Library. It should be noted that the author had to make a couple patches to PIL: one to get the library to build and another to get a function to work properly.

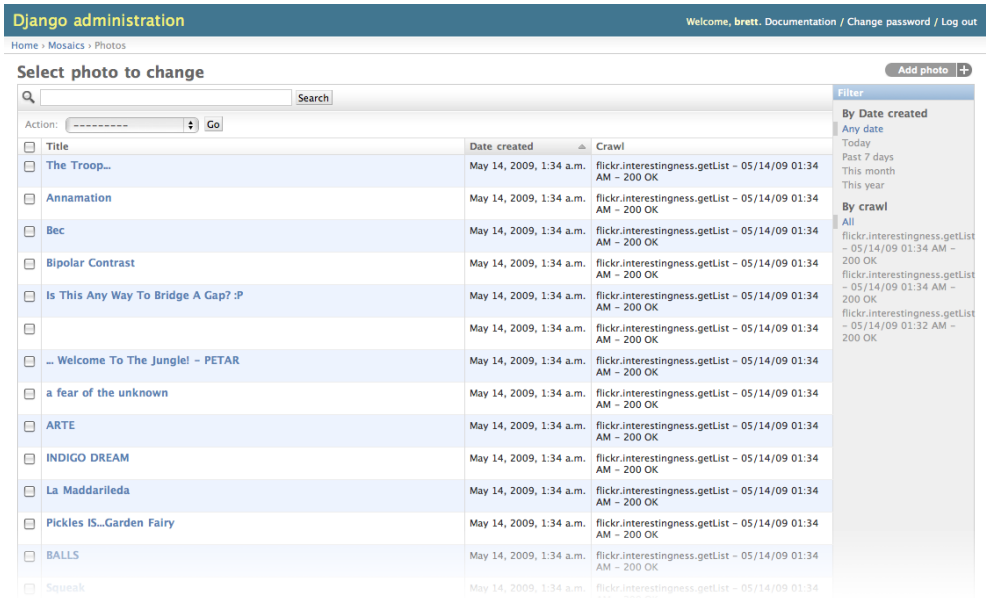

Figure 2: The Django admin area (much better than a regular database management tool).

Python was chosen for its combination of expressiveness and a large standard library, and Django was used because it provides all of the bits necessary to create a web application: HTTP server and view dispatcher, simple template language, standard database access API, and an added bonus: an automatically generated database administration area. PostgreSQL was utilized for its Django compatibility, but any SQL database should work.

Of course, none of these tools are absolutely essential to the project; a more standard web stack such as PHP, MySQL, and Apache would also suffice, but there are certain advantages to the Django/Python combination (the Django admin area and Python's imaging, threading, XML, and HTTP libraries, to name a few).

As mentioned in the introduction to this section, Dalivision uses Flickr's online image collection. Specifically, it uses photos classified by Flickr as "interesting". These images are usually high quality and highly varied – perfect for this use.

Lastly, all code was tested and executed using a custom-compiled build of Python 2.6 on Ubuntu Linux 8.10, running within a VMware virtual machine.

Implementing the Indexer

The indexer must perform several functions in order to complete a successful "crawl": get a list of photos from Flickr, add the photos to the database, and collect and save each photo's average color and luminance.

To get the photo list, the indexer connects to the Flickr API, asks for the 500 most recent interesting photos, and receives an XML response with information on how to look up each photo (certain data, such as farm, server, id, owner, and secret, is needed for each photo to be able to construct a URL from which to download it).

Each photo's XML information is then saved in the database. Also, a mapping is created between the photos' new database identifiers and their Flickr URLs. This is all very basic and not too difficult to get working. However, now we come upon a challenge – we need to download a copy of each photo and analyze its color information in a way that is efficient. A first attempt was to simply download each in order and process it, but this takes a significant amount of time, only some of which is actually spent doing anything CPU or I/O intensive. So for the second attempt, we parallelize the process, passing the dictionary of photo URLs to a Crawler object that will set up a pool of worker threads that can all download and analyze photos at the same time. The difference in speed between the two methods is large: two to five minutes and less than thirty seconds, respectively.

The Crawler object is based upon code from a snippet posted online that finds the fastest download mirrors for a popular burgundy-capped Linux distribution. It has

been adapted to be more general, and it can accept a callback function to run on each photo downloaded (in this case, an analysis function). This will prove useful later.

Implementing the Mosaic Creator

The mosaic creator requires a number of settings before it can begin its work: the name of a source photo, a tile size, ranges of acceptable average color and luminance values (for matching photographs with tiles), and alpha values for adjusting the color and luminance of the output image. After loading the source image and creating an empty image of the same size for output, the creator goes tile by tile in the source image, cropping out the tile region and finding its average color and luminance. The average color is used to create an image of average-colored tiles as the source image is read. (This will be blended with the output image later to make the colors more accurate.)

These values, along with the ranges specified earlier, are used to query the database for matching images. Of the matches, one is chosen at random and a Flickr URL is created for it and saved in a list. After the source image has been fully read, this URL list is handed to a Crawler object, which downloads the tile photographs in parallel and also crops and resizes each to the tile size (this is why the ability to specify a custom callback function was important). The Crawler object returns a dictionary of tile photographs loaded into memory and indexed by tile location. We can then run through all tiles in the image again, now placing the tile photographs in their correct location in the output image.

The final step is to do some color-correction. First, the output image is blended with the average-color version of the tiles (which, if viewed, would look like an ancient representation of the source image – just solid-colored tiles) using the alpha value specified earlier to control the amount of blending. Then the luminance is corrected, taking advantage of the isoluminance concept mentioned earlier: that keeping luminance the same while varying the hue and saturation will make two colors stand out more against each other. We traverse the output image pixel by pixel, convert the RGB color information for each to HLS, and push the luminance of the output pixel closer to that of the same pixel in the average-color image using alpha blending. The effect is the same as using the “Luminosity” blending mode in Photoshop.

Lastly, the output image is saved and displayed to the user.

5 Experimental Results

Starfish

The image below is a sample result from Dalivision. The source image was the ideal test candidate because it contains both detailed and smooth regions as well as a large, easily identifiable subject. Not including indexing time (about 30-60 seconds), this took about 45 seconds to render on a two-year-old dual-core laptop.

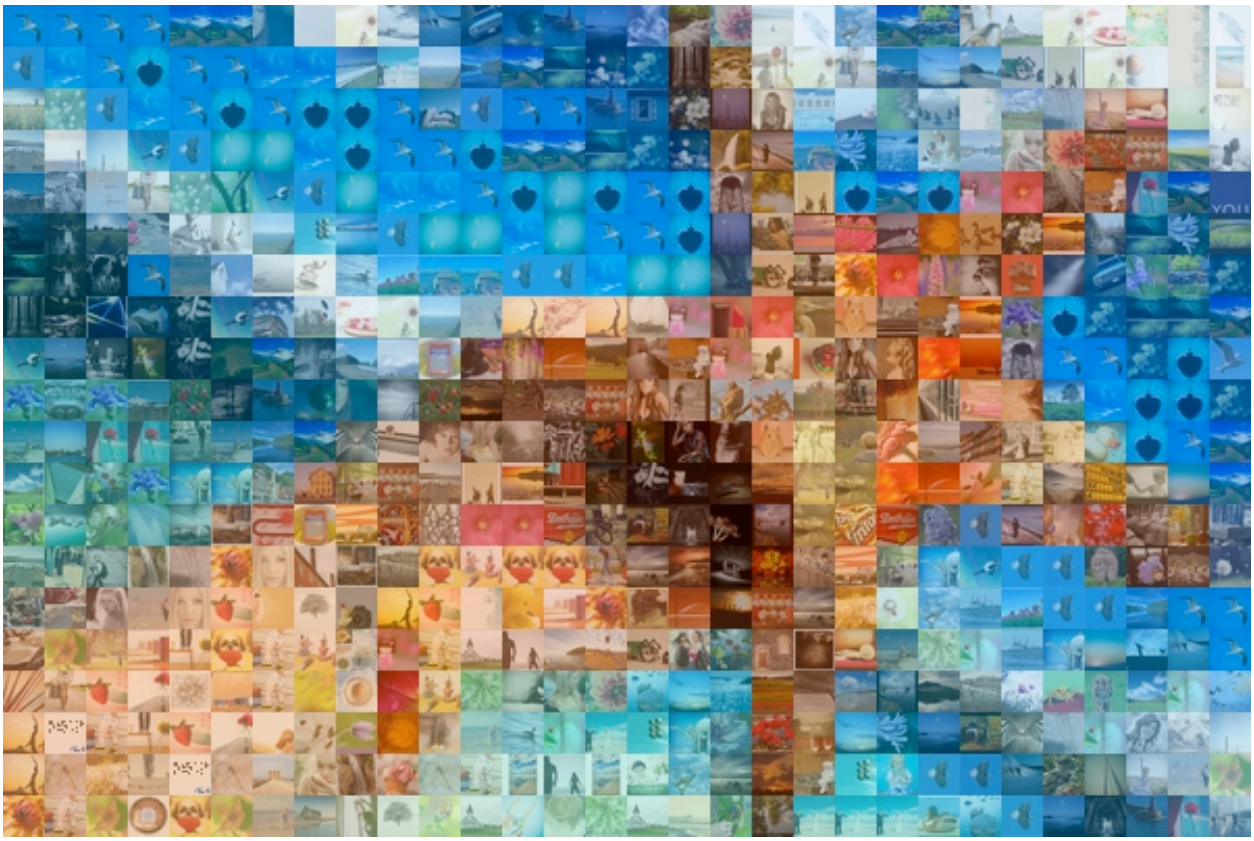

Figure 3: An average-color-blended photographic mosaic.

Figure 4.[1-2]: The mosaic from Figure 1 compared against its source photo.

Obama Presidential Portrait

Again, the source photo was chosen for its distinctive and recognizable subject. This series illustrates the effect of different settings available to the mosaic creator. As we can see, the addition of luminance blending does not impact the result significantly at low alpha values. The tiles have an increased smoothness and seem to have softer edges, especially in the face region. However, when increased near the maximum, the difference is quite significant: the content of the images stands out much more, but the subject remains visible and somewhat recognizable. Selecting an ideal representation is fairly subjective, but taking the average color alpha from the 3rd image and a somewhat lower luminance alpha from the 5th would yield good values of 45%/85%.

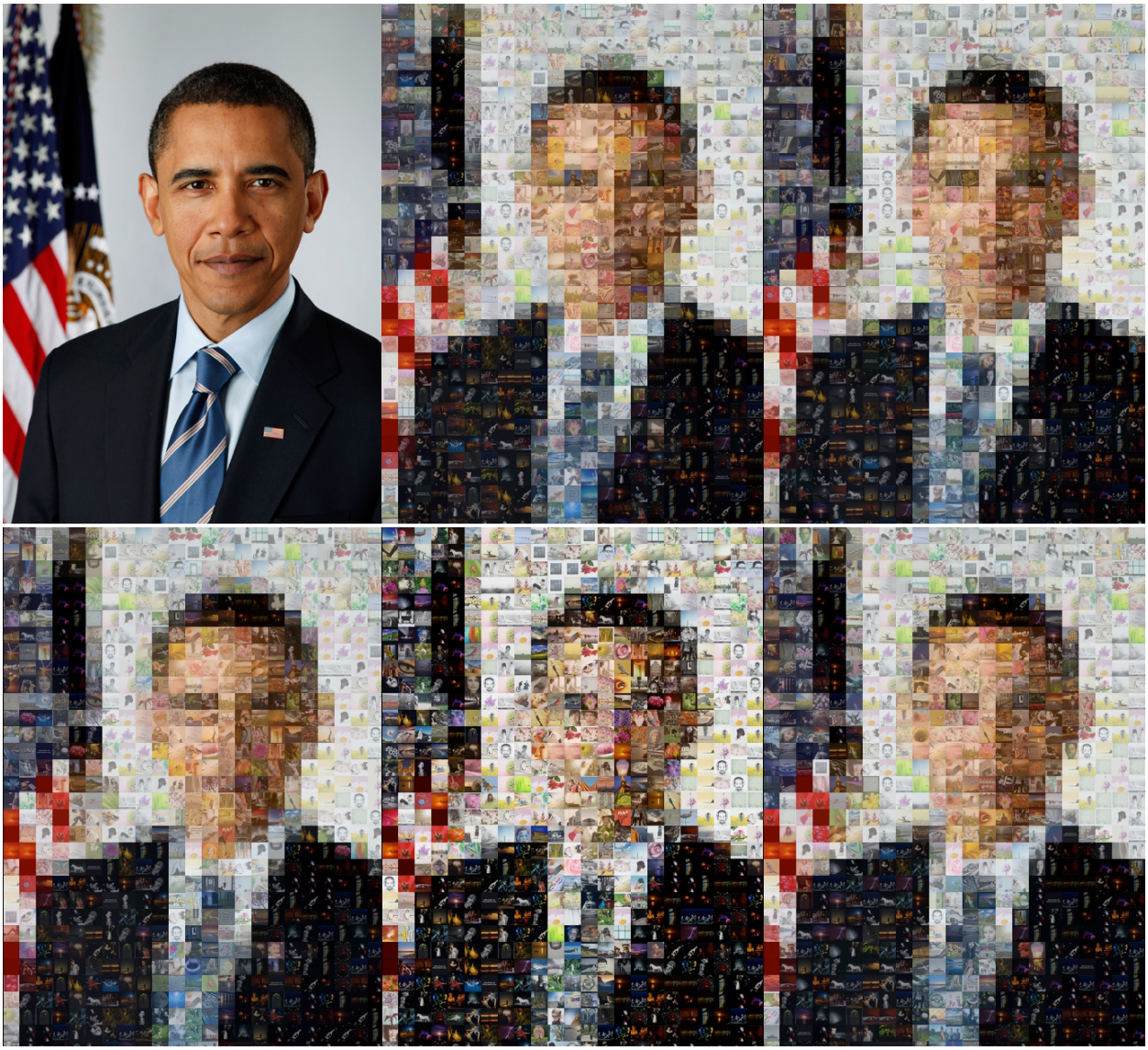

Figure 5.[1-6]: Left to right: the official portrait of President Barack Obama, the portrait with average color blending, with average color and luminance blending, with high average color and luminance blending, with average color and very high luminance blending, and with 45% average color and 85% luminance.

6 Related Work

The body of related work on this topic is surprisingly limited. Battiato, Di Blasi, Farinella, and Gallo recently published an overview of digital mosaic research which provides much insight into the state of the subject. Robert Silvers was one of the first to research the topic in 1997 and has created a company dedicated to producing photographic mosaics for companies and organizations. (He has also trademarked the term “Photomosaic,” hence the usage of “photographic mosaic” in this paper.) There are other companies and artists that create photographic mosaics as well. ArcSoft produces a commercial program (PhotoMontage) capable of creating mosaics, but it does not appear to be actively maintained. Other work has been done in video mosaicking and improving the color correction of the output image’s tiles.

However, none of these methods attempt to use a database of web images for generating the mosaic, and none tackle the problems that arise from the implementation of the mosaicking program, some of which are detailed in the next section. There are a few online tools for creating mosaics, but one does not lay out images in a representation of the source image and another (Mosaickr [1]) charges a fee for the creation (and the results are not too good). There is one more tool, written in Flash, which has a very smooth interface, but again the results are not too impressive, mainly because of the limited selection of images and the simple color-correction filters available.

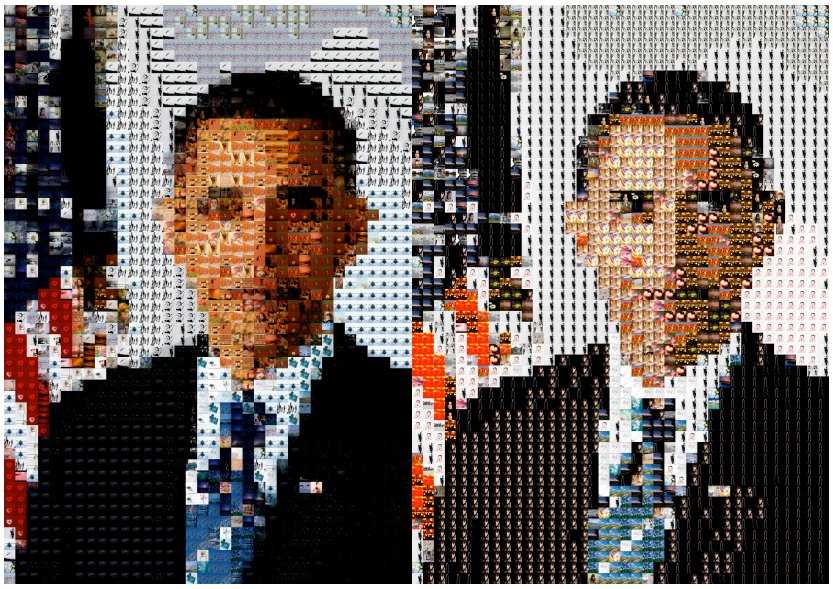

Figure 6.[1-2]: An uncorrected version of the Obama portrait compared to a colorized one. Note the heavy repetition of images in each.

7 Concluding Remarks / Ideas for Improvement

While this author is satisfied with the project (and he hopes his instructor is as well!) because it accomplishes the basic objectives mentioned in section 3, there are many areas in which it could be improved. Here is a far-from complete list:

- Performance issues: with a large database of images (more than 10,000), mosaic creation becomes very slow and memory intensive, even though the same number of database queries is run and the same number of tiles placed.
- Tile placement: a simple crop and resize is probably not the best method.
- Tile photograph choice: of the set of matching photos from the index, a random photo is chosen – using some heuristic to choose more wisely may help.
- Interface: the current code has a very rudimentary interface and requires the user to call views (via entering URLs) manually. This could be very easily improved using a HTML menu. Photos cannot currently be clicked to create a new mosaic for that photo. Preloading mosaics in the background might make the browsing experience more fluid. The source image should be displayed next to the output image for easier comparison. More information about the photo's owner should be provided to give them proper credit for their work. (The interface was not a huge priority for this first version of Dalivision, but it will be in the future. The Flash mosaic creator may provide helpful inspiration.)
- User inputs: currently changing alpha values or the source image means hand-editing the Python code. This could be easily done using an HTML form and Django's form handling and validation capabilities. A user could even upload or link to their own source photo instead of having to stick with a random or default one that the application specifies.
- Testing: there are many edge cases and off-by-one errors that are unaccounted for (viz. the black line on the left edge of some samples). Also, the threaded Crawler class needs some adjustment and further review to ensure deadlocks do not occur. It may be useful to have some sort of throttling built-in to keep from sending too many requests to Flickr at once. And, the Crawler depends on Python threading, which is limited to using one core due to an issue with the CPython interpreter. It could take advantage of multiple cores if the code were ported to use the Python multiprocessing module instead.

Certainly, there is much to be done before Dalivision could be considered production ready and deployable. But I plan to continue the development of the application in the hope that it can one day provide beautiful, freely available photographic mosaics for anyone with an Internet connection.

8 References

[BDFG07] BATTIATO, S., DI BLASI, G., FARINELLA, G., GALLO, G: Digital Mosaic Frameworks – An Overview. In Proc. *Computer Graphics Forum* (2007), 794-812.

[LSKL05] LUONG T.Q., SETH A., KLEIN A. W., LAWRENCE J.: Isoluminant Color Picking for Non-Photorealistic Rendering. In *Proc. of Graphics Interface 2005* (2005), 233-240.

[1] http://mosaickr.com

[2] http://www.inspirit.ru/exchange/mosaic/