Optimal Photo Mosaics: An Instance of the Assignment Problem

Tobias Weber
tm.weber@stud.unibas.ch

Fall 2023

Abstract

A photo mosaic is a picture that itself consists of many smaller pictures, called tiles. These tiles are carefully arranged in a rectangular grid such that a single large picture emerges. An optimal mosaic for a given set of tiles is a photo mosaic that resembles a target picture as closely as possible. This is determined by a distance function. Previous research has primarily focused on approximate solutions that scale well with large problem instances. In contrast, we present an algorithm that generates the optimal mosaic by solving the corresponding assignment problem. We additionally construct a family of distance functions based on downscaling tiles. A series of experiments were conducted to evaluate the algorithm and distance functions. The results suggest that realistic problem instances can be solved in a matter of seconds. We conclude that generating optimal photo mosaics should no longer be considered infeasible.

1 Introduction

The widespread adoption of smartphones with capable cameras has made digital photographs ubiquitous. Together with the availability of affordable storage, this results in an ever-expanding collection of photos. A popular technique that takes advantage of this abundance of photos are photo collages. A photo collage combines multiple, usually related photos in a single picture. This is achieved by reducing the size of each photo, possibly cropping it and arranging them in a visually appealing manner.

A special category of photo collages are *photo mosaics*. A traditional mosaic is made up of small colored tiles, which are arranged to appear as a single larger picture. Analogously, a photo mosaic is a picture which is itself made up of many small pictures, called *tiles*. Photo mosaics are not a recent invention and were originally created by hand. With the advent of computers, algorithms have been developed to produce photo mosaics for a given set of tiles and *target picture*. Today, photo mosaics are widespread. They can be found on magazine covers and posters and are popular birthday or wedding gifts.

In this paper, we discuss a simple algorithm for creating optimal photo mosaics. This is achieved by reducing the problem to the *linear assignment problem*, a fundamental combinatorial optimization problem. Creating optimal photo mosaics has mostly been ignored by research because of its high computational complexity. Instead, algorithms and heuristics were developed to approximate the optimal mosaic. While these have been very successful, they are often non-trivial to implement and use complex data structures. It would be ideal if we could opt for the simpler algorithm which also yields superior results. We pose the hypothesis that calculating the optimal solution has become feasible for typical problem instances.

A frequently used simplification of the problem first computes the average color of each tile and discards the color information of individual pixels. This is equivalent to *downscaling* the tile to just a single pixel. Using the presented algorithm, we conduct experiments to investigate the impact on mosaic quality when tiles are downscaled to different resolutions.

In the following section, we briefly review related work. Section 3 introduces our optimal algorithm for photo mosaics. The evaluation and its results are presented in Section 4. Finally, we discuss the results in Section 5 and potential future work is outlined in Section 6.

2 Related Work

One of the earliest scientific papers concerning photo mosaics was published by Finkelstein and Range [1] in 1998. It focuses on adjusting the colors of tiles after the mosaic has been created. The target picture is thus further emphasized. This is done by generalizing traditional halftoning techniques and applying a “shift-and-scale" rule.

Kim and Pellacini [2] describe the Jigsaw Image Mosaic algorithm for creating mosaics out of arbitrarily-shaped tiles. For this, the target picture is divided into single-colored containers. For pictures that are

made up of clearly discernible shapes it leads to great results, but it is not suited for arbitrary pictures.

A randomized iterative improvement algorithm for photo mosaics is proposed by Narasimhan and Satheesh [3]. It begins initially with an arbitrary feasible mosaic. Subsequent iterations swap random tiles or replace them with unused ones to find a better mosaic. This approach is also well suited if a tile may be repeated up to t times.

Battiato, Blasi, Farinella, *et al.* [4] give a comprehensive overview about existing techniques for digital mosaic creation. In particular, they compare the computational complexity of different photo mosaic algorithms. By using the *Antipole Tree Data Structure*, Blasi, Gallo, and Petralia [5] achieve the best computational complexity of just $O(n \log(m))$, where n is the number of target picture pixels and m the size of the tile set. A drawback of this approach is that it may result in the repetition of tiles.

3 The Optimal Mosaic Problem

We open this section by defining the problem of finding an optimal mosaic. The given input is a set of tiles $T = \{T_j\}$ for $j = 1, 2, ..., m$ and target picture P with desired number of tiles n (with $n \leq m$). The target picture P is subdivided into a regular grid of n square *blocks*[1] P_i, which are contained in a set we also call P for simplicity. Additionally, we define a *distance function*

$$d : P \times T \mapsto \mathbb{R}_{\geq 0}.$$

It measures the distance between each possible pair of block and tile. The selection of d has a substantial impact on the perceived quality of the optimal mosaic and will be discussed in Section 3.2.

A *mosaic* is a picture with the same dimensions as P. It is made up of n blocks, where each block corresponds to a unique tile from the set of tiles[2]. We thus define a mosaic to be an injective function that assigns each block of P to a tile of T. An *optimal mosaic* Q for a given distance function d is a mosaic which minimizes the total distance

$$D_Q = \sum_{p \in P} d(p, Q(p)).$$

3.1 Solving the Problem

The key insight for solving the optimal mosaic problem lies in recognizing that it is essentially an instance of the linear assignment problem, short LAP. It can be defined as finding the best assignment (lowest total cost) of elements of a set A to elements of a set B. The cost for a single assignment is stored in a $|A| \times |B|$ cost matrix C. We find that the set of blocks P corresponds to A, the set of tiles T corresponds to B and the entries of the cost matrix can be defined as $C_{i,j} = d(P_i, T_j)$. This leads us to the following algorithm for the optimal mosaic problem:

Algorithm 1 Solving the Optimal Mosaic Problem

Input: Target picture P_{target}, set of tiles T with size m, number of blocks n
Output: Optimal mosaic Q

Step 1: Subdivide P_{target} into n blocks and store them in set P.
Step 2: Calculate the cost matrix C with $C_{i,j} := d(P_i, T_j)$ using the distance function d.
Step 3: Solve the LAP for input C and store the optimal assignment in variable X.
Step 4: Replace the blocks of P_{target} with the tiles in T given by X to build Q.
Step 5: Return Q.

As the cost matrix C is constructed explicitly, Algorithm 1 has a storage complexity of $O(nm)$. Step 3 is the dominant part for the computational complexity. The LAP can be solved with the Hungarian method proposed by Kuhn [6]. An improved version is the Jonker-Volgenant algorithm [7]. It has a computational complexity of $O(n^2 m)$ for $n < m$.

3.2 Distance Functions

Up to this point, we have not yet stated an implementation for the distance function d. The choice of d can significantly alter the appearance of the resulting mosaic. We assume that blocks and tiles are represented as images in the RGB color space.

One way to define d is to first map each tile (and block) to a k-dimensional feature vector. Any distance measure for vectors can then be used, for instance the Euclidean distance or the cosine measure.

Ancient Roman mosaics used colored stones as tiles. We can mimic this simple approach by calculating the average color for tiles. This results in 3-dimensional feature vectors, one dimension per color channel[3]. While this is very cheap to compute, it also disregards the structure in tiles. The approach can be

[1] We assume that this subdivision is always possible. This might require cropping P and adjusting n to get blocks with integer dimensions.

[2] If the dimensions of a tile are not equal to the dimensions of a block, it can simply be rescaled. For simplicity, we assume that the dimensions are always equal.

[3] We operate under the assumption that the (Euclidean) distance in the RGB color space matches human perception of color distance ΔE. Many better measures based on uniform color spaces have been developed. A good compromise between complexity and accuracy is the *Lab color space*. The CIEDE2000 formula [8] is based on it and has become the standard measure for ΔE.

improved by dividing each tile into a grid of $r \times r$ subtiles and calculating the average color per subtile. The dimension of the resulting feature vector is $k = 3 \cdot r^2$. We can increase r up to the point where it is equal to the side length of a tile:

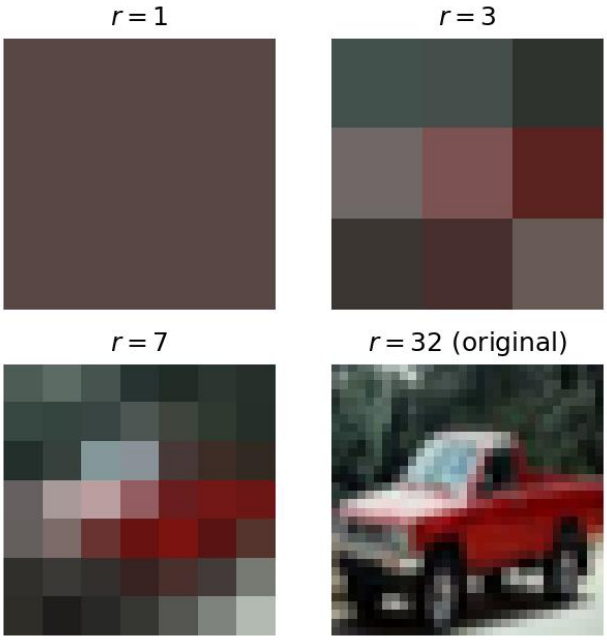

Figure 1: *Different values of r for a tile of the CIFAR dataset.*

This approach has been employed by Lee [9] as a cheap way to improve mosaic quality. Other features are also feasible. Finkelstein and Range [1] use wavelets-based features for tile matching, which were introduced by Jacobs, Finkelstein, and Salesin [10].

4 Evaluation

In this section, we compare the runtime and mosaic quality for different distance functions and problem instances.

Solving the optimal mosaic problem with Algorithm 1 is only feasible if its runtime for typical problem instances is acceptable. We focus on the use case where the goal is to create a single photo mosaic for a given target picture and set of tiles. A runtime of several seconds up to a few minutes is therefore considered acceptable. A visually pleasing mosaic consists of enough tiles for the target picture to emerge. If too many tiles are used, the individual tiles are no longer discernible. In practical scenarios, we thus anticipate values for n within the lower thousands. The size m of the set of available tiles is typically around a multiple of n in the single-digit range [4].

[4] Otherwise, m could first be reduced by removing tiles; either at random, or by taking the target picture into account.

4.1 Experiment Setup

While the generated mosaics are optimal, a measure that allows us to compare the effect of different distance functions is nonetheless valuable. We define the *error* of a mosaic Q for a target picture P as the mean of squared differences between corresponding pixel values of Q and P.

The algorithm is implemented in Python and uses the SciPy LAP solver[5]. For the experiments, we used the family of distance functions described in Section 3.2 with the Euclidean distance measure and $k = 3 \cdot r^2$ features. The target picture P was given by Figure 2. For the set of tiles T, a subset of the CIFAR-100 dataset [11] was used. The experiments were conducted on an Intel Core i7-13700k CPU.

Figure 2: *The target picture P for the experiments. Source: Andrew Pons on unsplash.com.*

Table 1: *Overview of the parameters used in the experiments.*

	Parameters		
Experiment	n	m	r
A	1 536	15 000	*variable*
B	1 536	*variable*	3
C	*variable*	15 000	3

Experiment A investigated the effect of r on the runtime and error. In experiment B, we varied the number of tiles m and the runtime was measured. Analogously, we varied the number of blocks n in

[5] scipy.optimize.linear_sum_assignment

experiment C and kept the other parameters constant. Together, the results of these experiments allow us to estimate for what problem size the algorithm becomes infeasible. The parameter values for each experiment can be found in Table 1.

4.2 Results

The outcomes of experiment A are shown in Figure 3 and Figure 4. We see that larger r result in a quadratic increase of the runtime. This corresponds to a runtime that is proportional to the number of features k. In practice, small values for r are sufficient. When looking at the error shown in Figure 4 we see that the error converges quickly. A value of $r = 3$ already yields very good results. This can also be observed visually in Figure 7.

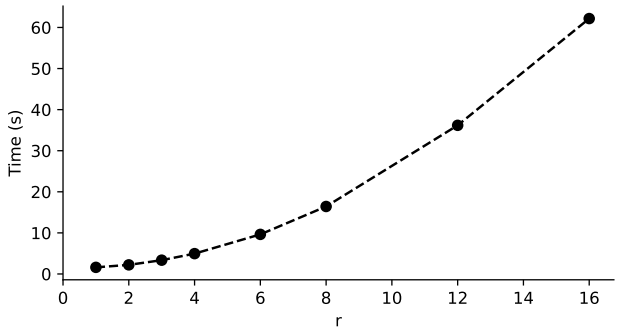

Figure 3: *The measured runtime in experiment A for varying r.*

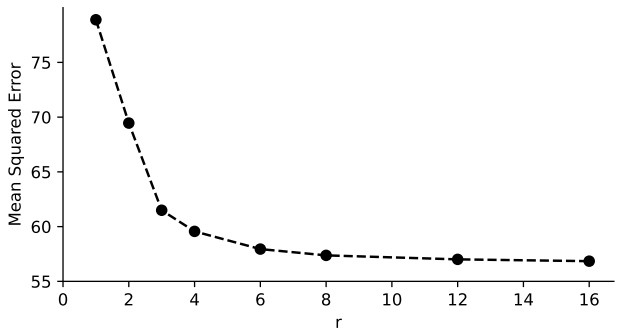

Figure 4: *The measured error in experiment A for varying r.*

Figure 5 depicts the results of experiment B. They suggest a linear relationship between m and the algorithm runtime. This is consistent with the theoretical computational complexity of $O(n^2m)$. Even for large sets of tiles, the algorithm finishes in a matter of seconds.

In Figure 6 the measurements of experiment C are shown. As expected, the runtime appears to increase with the square of n. For the measured problem instances, the algorithm still terminates in a reasonable amount of time. However, for even larger n ($> 10\,000$) the high runtime becomes an issue.

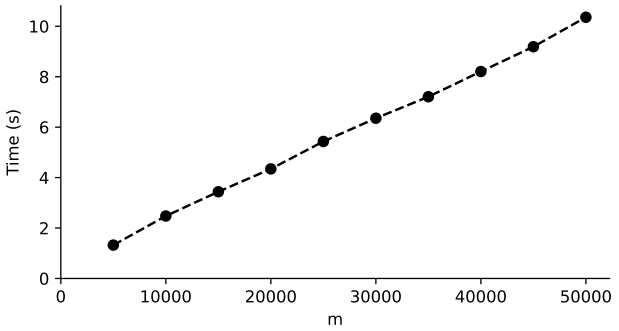

Figure 5: *The measured runtime in experiment B for varying m.*

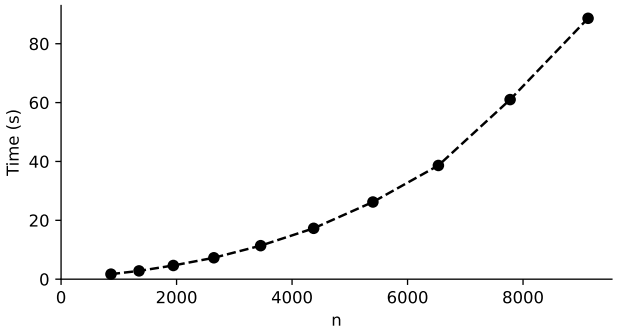

Figure 6: *The measured runtime in experiment C for varying n.*

5 Conclusion

Having presented the algorithm and performed the evaluation, we proceed to analyze the implications of the evaluation results and draw a conclusion.

From experiment A we learned what distance functions are suitable for constructing mosaics of high quality, while minimizing its impact on performance. A value of $r = 3$ can already notably reduce the error without resulting in a significantly higher runtime. A value of $r = 6$ (resulting in $k = 108$ features) is sufficient for all practical purposes, as any additional gain in quality becomes insignificant. This result should also hold for general distance functions based on feature vectors. Using vectors with hundreds of dimensions is unnecessary and leads to a considerable computational overhead.

In the past, the cubic computational complexity of Algorithm 1 has rendered this approach infeasible, and it was not further discussed in scientific literature. Contrary to that, the results of experiment B and C suggest that the presented algorithm is fast enough for most practical purposes, even on off-the-shelf consumer hardware. We assume that this is a consequence of the development of more efficient processors in the last years.

A variation of the optimal mosaic problem allows tiles to be repeated up to t times in the mosaic. For this problem, approximate solvers maintain an advan-

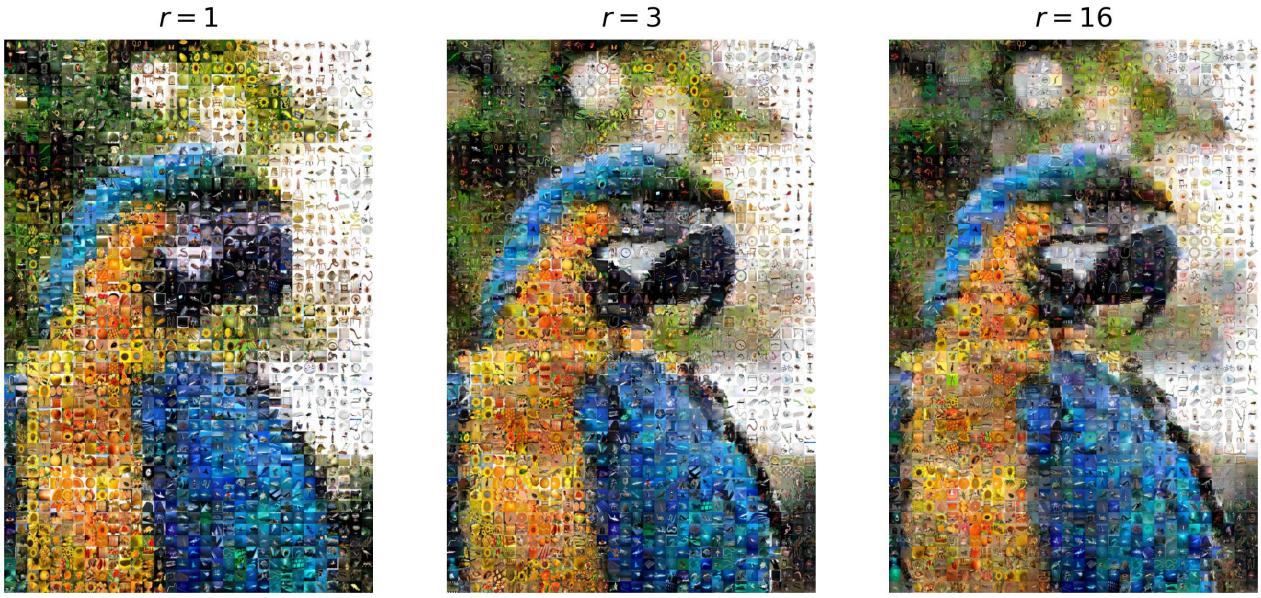

Figure 7: *Some of the resulting mosaics from experiment A ($n = 1\,536$). The perceived quality for $r = 3$ and $r = 16$ is very similar.*

tage over our approach. We can solve the problem by adding t copies of each tile to the set of tiles T. This increases m and therefore the runtime by a factor of t. Most approximate algorithms can leverage the repetition more efficiently.

We conclude that seeking an optimal solution for the mosaic problem is a reasonable approach. For typical problem sizes the presented algorithm terminates in a matter of seconds. Choosing a suitable distance function has a significant impact on performance and visual quality. The proposed distance function, which corresponds to downscaling each tile to a resolution of 3 by 3 pixels, resulted in mosaics with a high level of detail. This allows us to recognize features of the target picture in the mosaic that are smaller than individual tiles. With traditional monochrome tiles, this effect would not be visible. This adds a compelling element to the charm of photo mosaics.

6 Future Work

In the scope of this research project, we only evaluated the algorithm performance in isolation. A more thorough analysis could clarify whether the lower runtime of suboptimal algorithms is worth the decrease in quality.

Another promising topic for future work is to further explore various distance functions. It would also be interesting to see the effect of using different color spaces for measuring color difference.

Our proposed method can easily be generalized to tiles with a fixed non-square aspect ratio. A harder problem would be to allow tiles of different sizes based on the level of detail in the target picture. Blasi, Gallo, and Petralia [5] have proposed such mosaics under the name *QT-Photomosaics*.

References

[1] A. Finkelstein and M. Range, "Image mosaics," in *Electronic Publishing, Artistic Imaging, and Digital Typography*, Springer Berlin Heidelberg, 1998, pp. 11–22. DOI: 10.1007/bfb0053259.

[2] J. Kim and F. Pellacini, "Jigsaw image mosaics," in *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, ser. SIGGRAPH '02, San Antonio, Texas: Association for Computing Machinery, Jul. 2002, pp. 657–664, ISBN: 1581135211. DOI: 10.1145/566570.566633.

[3] H. Narasimhan and S. Satheesh, "A randomized iterative improvement algorithm for photomosaic generation," *2009 World Congress on Nature & Biologically Inspired Computing (NaBIC)*, pp. 777–781, Dec. 2009. DOI: 10.1109/NABIC.2009.5393882.

[4] S. Battiato, G. di Blasi, G. M. Farinella, and G. Gallo, "Digital mosaic frameworks - an overview," *Computer Graphics Forum*, vol. 26, no. 4, pp. 794–812, Jun. 2007. DOI: 10.1111/j.1467-8659.2007.01021.x.

[5] G. di Blasi, G. Gallo, and M. P. Petralia, "Smart ideas for photomosaic rendering," in *Fourth Eurographics Italian Chapter Conference 2006*, Eurographics, 2006, pp. 267–272. DOI: 10.2312/LocalChapterEvents/ItalChap/ItalianChapConf2006/267-271.

[6] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval Research Logistics Quarterly*, vol. 2, no. 1-2, pp. 83–97, Mar. 1955. DOI: 10.1002/nav.3800020109.

[7] R. Jonker and A. Volgenant, "A shortest augmenting path algorithm for dense and sparse linear assignment problems," *Computing*, vol. 38, no. 4, pp. 325–340, 1987. DOI: 10.1007/BF02278710.

[8] M. R. Luo, G. Cui, and B. Rigg, "The development of the CIE 2000 colour-difference formula: CIEDE2000," *Color Research & Application*, vol. 26, no. 5, pp. 340–350, Aug. 2001. DOI: 10.1002/col.1049.

[9] H.-Y. Lee, "Generation of photo-mosaic images through block matching and color adjustment," *International Journal of Computer and Information Engineering*, vol. 8, no. 3, pp. 457–460, 2014. DOI: 10.5281/zenodo.1337092.

[10] C. E. Jacobs, A. Finkelstein, and D. H. Salesin, "Fast multiresolution image querying," in *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, ser. SIGGRAPH '95, New York, NY, USA: Association for Computing Machinery, Sep. 1995, pp. 277–286, ISBN: 0897917014. DOI: 10.1145/218380.218454.

[11] A. Krizhevsky, G. Hinton, *et al.*, "Learning multiple layers of features from tiny images," 2009. [Online]. Available: https://api.semanticscholar.org/CorpusID:18268744.

Appendix

On the next pages, we present additional mosaics that were generated by the presented algorithm. The tiles are again sourced from the CIFAR-100 dataset.

Figure 8: *Mosaic of a horse ($n = 486$, $r = 5$).*
Source of target picture: Helena Lopes on https://unsplash.com.

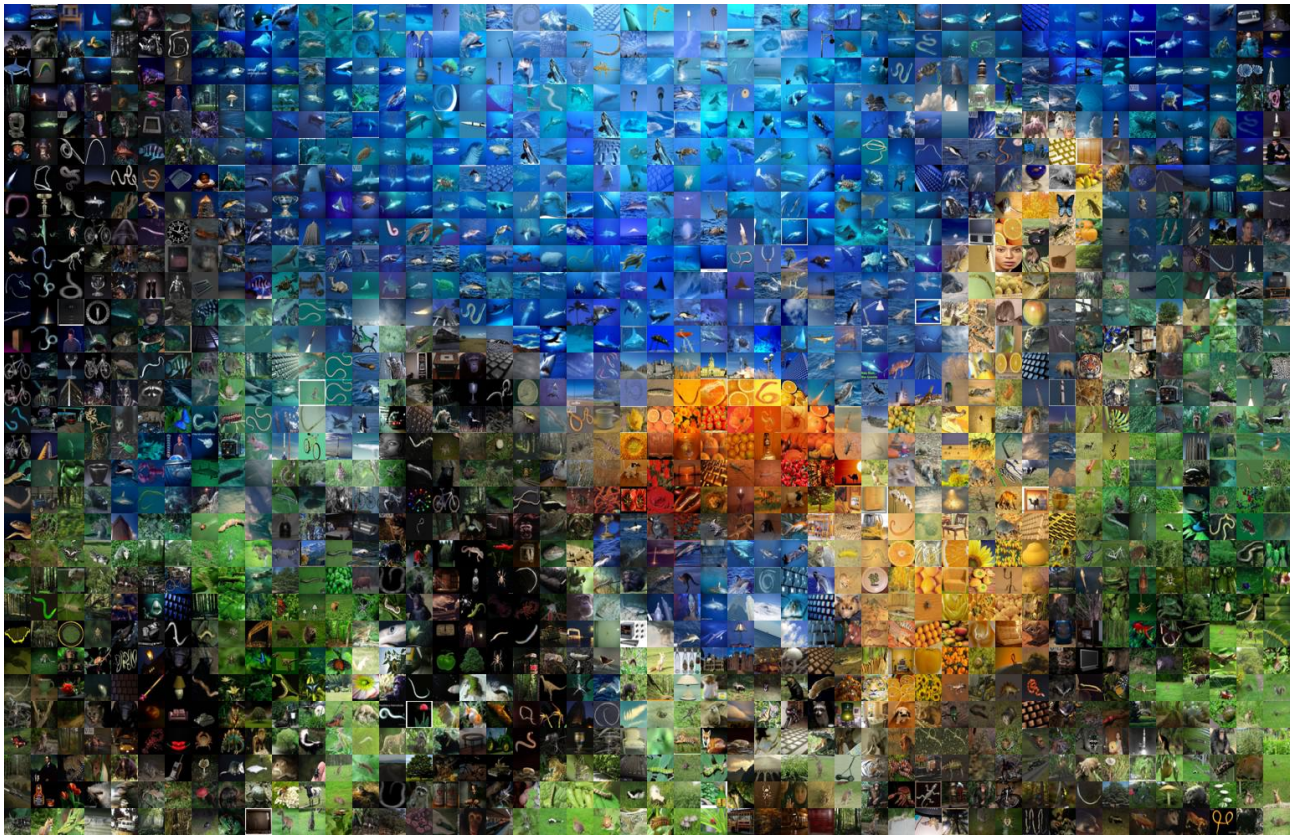

Figure 9: *Mosaic of a goldfish in an aquarium ($n = 1488$, $r = 5$).*
Source of target picture: redcharlie on https://unsplash.com.

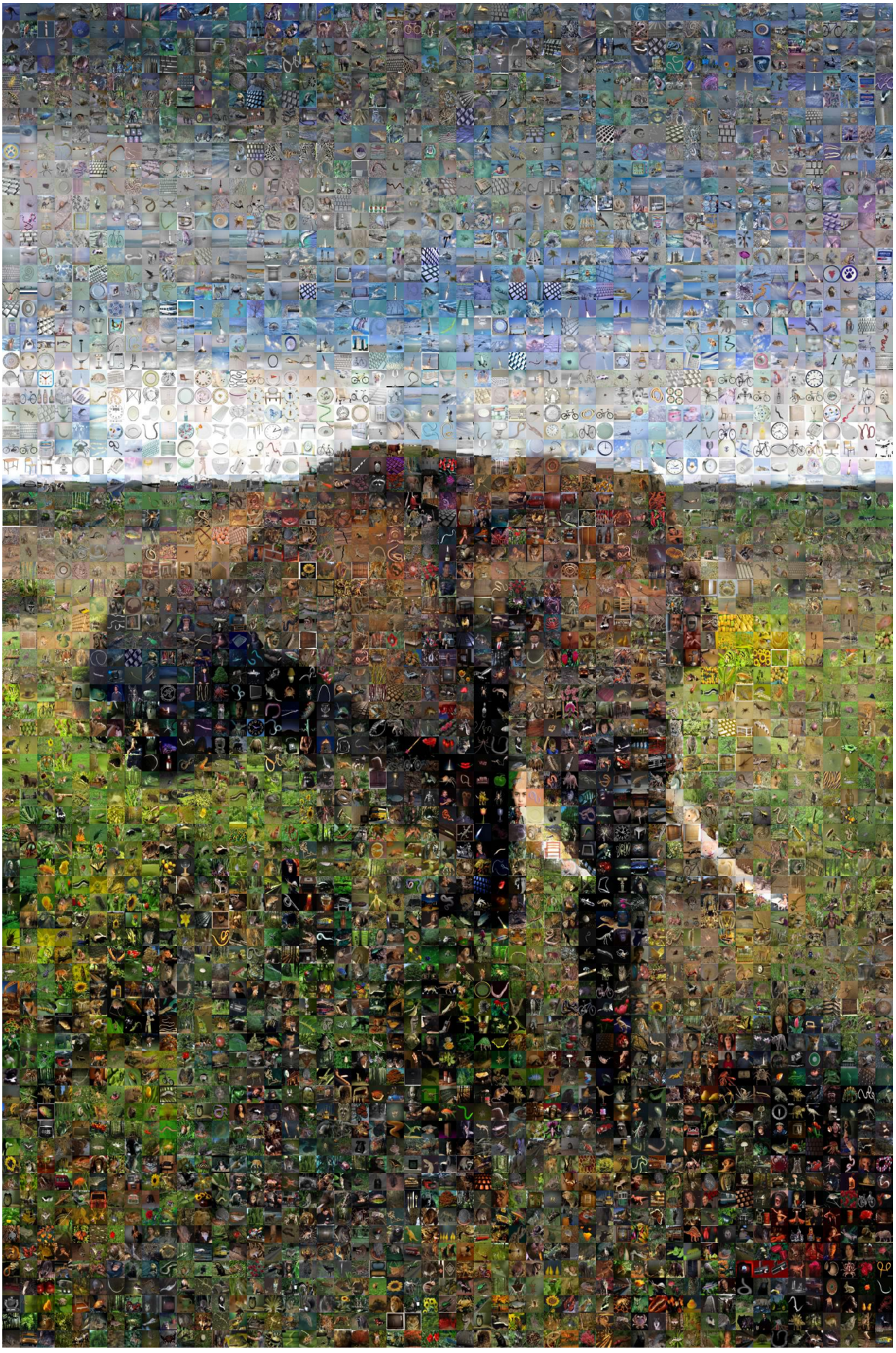

Figure 10: *Mosaic of an African elephant ($n = 3927$, $r = 5$).*
Source of target picture: Delbert Pagayona on https://unsplash.com.