Case Studies

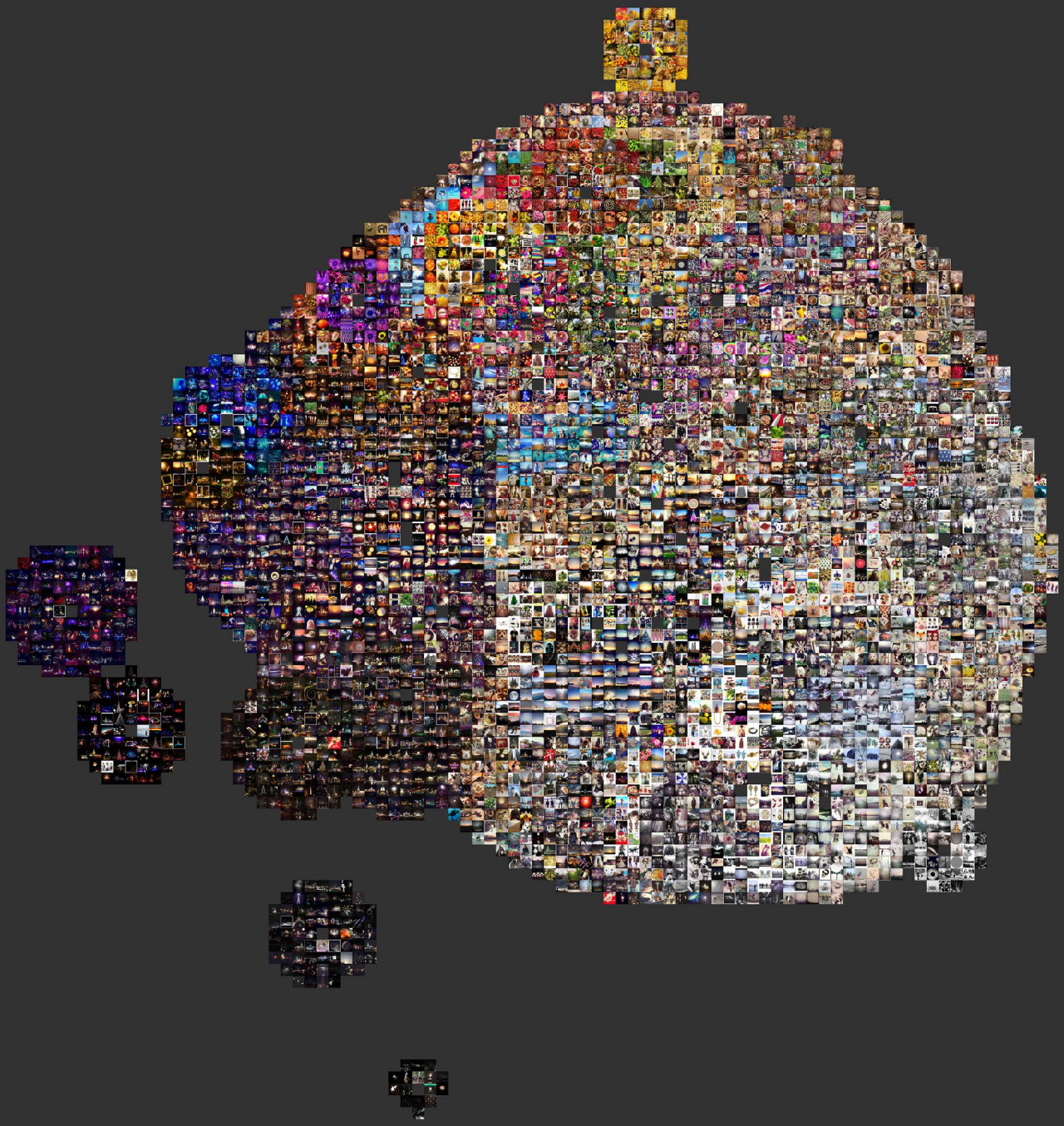


Fig. 1. Growing Entourage with 50 clusters of Instagram photos machine-tagged under the heading ‘nature’.

Invited Article

Direct visualization techniques for the analysis of image data: the slice histogram and the growing entourage plot

Damon Crockett

Abstract: The following is a description of two novel techniques for the direct visualization of image data. Direct visualizations of image data make use of the images in their original visible format. The first technique, the slice histogram, arranges slices of images as histograms, organized by both visual and non-visual variables. The second technique, the growing entourage plot, organizes high-dimensional clusters of images on a 2D canvas by projection. Both techniques are designed for exploratory analysis of image datasets.

Keywords: direct visualization, images, exploratory data analysis

Introduction[1]

In the following, I discuss two novel methods of direct visualization: (1) the slice histogram; and (2) the growing entourage plot. 'Direct visualization' is a term coined by Lev Manovich (2010) to describe visualizations of image or video data that make use of the data in its original visible format. The general form of direct visualization for static images is the 'image plot'—an arrangement of images on a single digital canvas.[2] The techniques described here are instances of this general type.

The active form of direct visualization for the digital humanities is media visualization (Manovich 2012)—direct visualization of cultural media collections—and the Software Studies Lab at UCSD has been largely responsible for its promotion and development. Several early forms of media visualization are cited in (Manovich 2010), and Manovich and collaborators have produced many examples of media visualizations, including Mapping Time, a chronologically sorted montage of every Time magazine cover from 1923 to 2009;[3] The Shape of Science, a visualization of over 10,000 pages each of the magazines Science and Popular Science;[4] Manga Style Space, a visualization containing over 1 million Manga pages sorted by their visual features, meaning simply their visual properties, basic things like average hue or brightness, or more sophisticated measurements like entropy;[5] and Phototrails, a series of image plots, each containing thousands of Instagram photos sorted by basic color features.[6]

In these and in many other projects, Manovich has demonstrated the power of this method for the digital humanities, and the tools I describe here are both extensions of this general method and, perhaps, attempts to show direct visualization relevant not only for cultural media but for all image data and to point in the direction of a new and purely perceptual paradigm for image data analysis.[7]

Direct Visualization

Direct visualizations are special cases of glyph visualizations, a class of statistical plots that present data points as glyphs—icons that carry information by way of their non-relational characteristics, things like size, shape, color, etc.

The dominant form of relational data visualization is the scatterplot—a collection of points plotted along (usually) two axes that reveals the relationship between the variables mapped to the axes. Because traditional scatter points have no non-relational characteristics—they are in fact point locations and not objects at all—they carry information only by their spatial positions. Traditional scatterplots, then, can present only as many informational dimensions as there are plotting axes. If, however, each scatter point is made a glyph with n non-relational characteristics, the dimensionality of the visualization is increased by n. Direct visualizations can therefore be understood as limit cases of glyph visualization, because they preserve, strictly speaking, all of the visual information in the dataset.

But the mere presence of the information in the visualization does not guarantee its being readable for the viewing subject. Important for direct visualizations are particular choices about sorting or otherwise organizing the images on the canvas in order to reveal patterns. Our lab has pioneered a suite of techniques for organizing images as plot elements on large digital canvases. We use sorted rectangular montages (Image Montage),[8] Cartesian scatterplots, and perhaps most identifiably, polar scatterplots (Image Plot).[9]

Image Histogram

Recently, we've pioneered a new technique, one that has roots in our Selfie City project:[10] the image histogram. The most basic form of the image histogram simply gives the distribution of a single variable—like average brightness or hue, time of day, geolocation, etc.—-and uses the images themselves as plot elements. Like the image montage, the image histogram gives every image its own place in the plot, and like the image plot, the image histogram uses an axis to organize data points. We have found this combination of characteristics to be very useful in presenting, e.g., temporal patterns in image data.

But because image histograms are direct visualizations, we needn't settle for the presentation of a single variable. The images themselves are right there on display, and although the particular choice of histogram (i.e., what gets assigned to the x-axis) dictates the horizontal sorting, the vertical sorting is still up for grabs and can reveal additional patterns in the data. We can therefore think of each histogram bin as a columnar montage that can be sorted as many times as we like. In Figure 2, we can see the results of multiple vertical sorts.

Slice Histogram

My own work with the lab began around the time we started making image histograms, and I confronted a problem: I wanted to make direct visualizations that reveal, with great clarity, the color properties of image datasets. Images wear their colors on their sleeves, of course, and so direct visualizations do carry color information—all of it, in fact—but again, particular choices about sorting can matter a lot.

Unfortunately, even our very best sorting choices for image histograms do not yield crystal clear presentations of color. And the problem is that images typically contain lots of different colors. When you sort them by color, how do you do it? Do you take the mean hue of the whole image? The mode? Do you look at all color dimensions, or just one? How you answer these questions will depend on your goals, of course, but the questions are less important if the images have very low standard deviation of their color properties. The more uniformly-colored the images, the easier it is to sort them by color. But we can't simply enforce uniformity in our data. What we can do is plot slices of images instead of whole images. This general idea is at work in various computer vision algorithms, and it makes good sense: images typically capture scenes, and scenes have parts, so we should at the very least be looking at those parts, whatever else we do. In computer vision (and human vision), the selection of parts can be very sophisticated (in human vision, this is roughly the function of visual attention), but it is computationally expensive. We need a way to get better color visibility without sacrificing computational speed.

I developed a technique that is fast and yields excellent color visibility (Figure 3; close-up in Figure 4). Each image is sliced into some number of equal-sized parts, color properties are extracted from the parts, and those parts are then plotted as an image histogram. The entire process, carried out with 1 million slices, can take less than an hour.

The number of parts will depend on the kind of images we use. In my own work, I set a criterion value for average standard deviation of hue (~0.1) and then find the minimum number of parts needed to meet the criterion. This approach ensures both

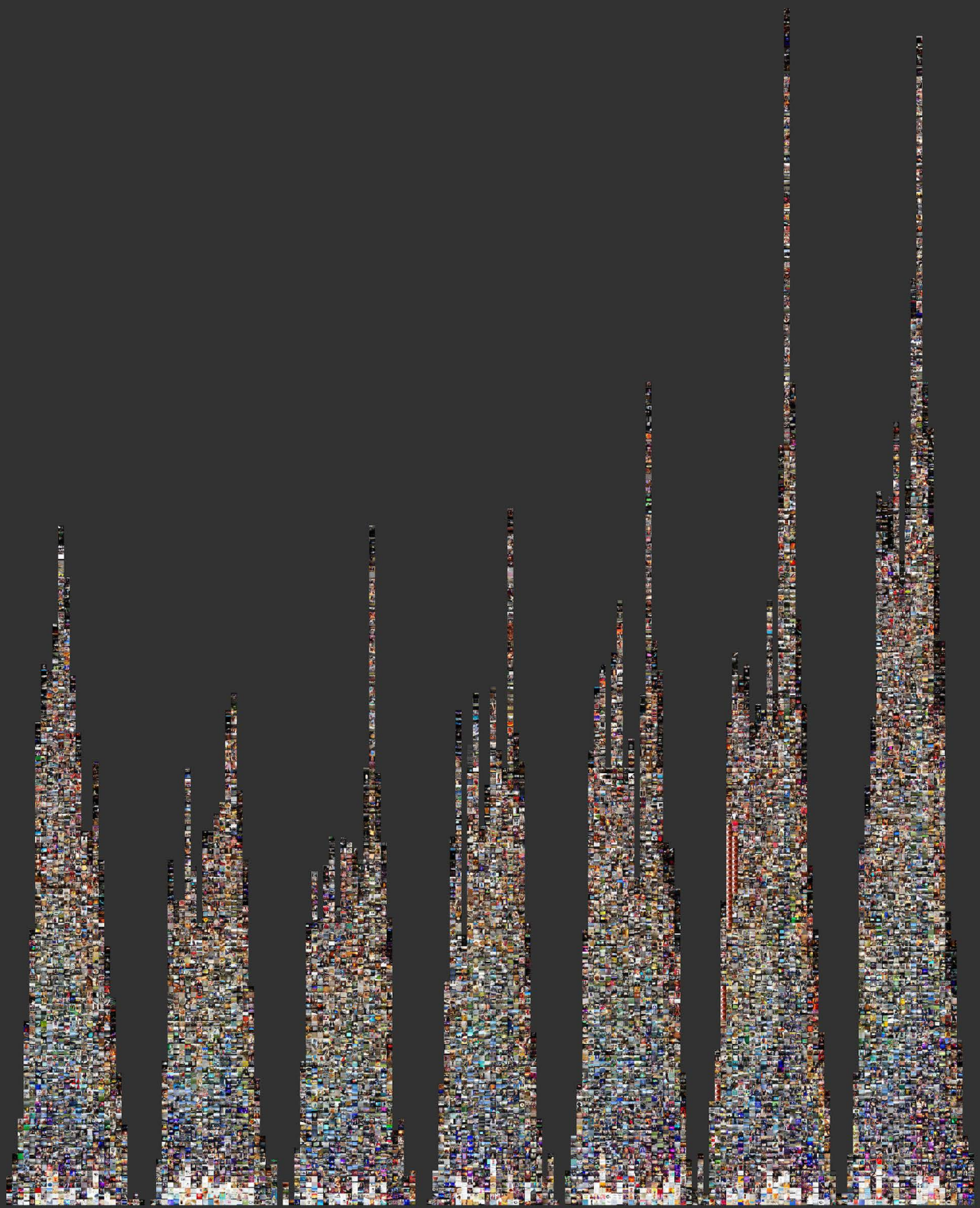
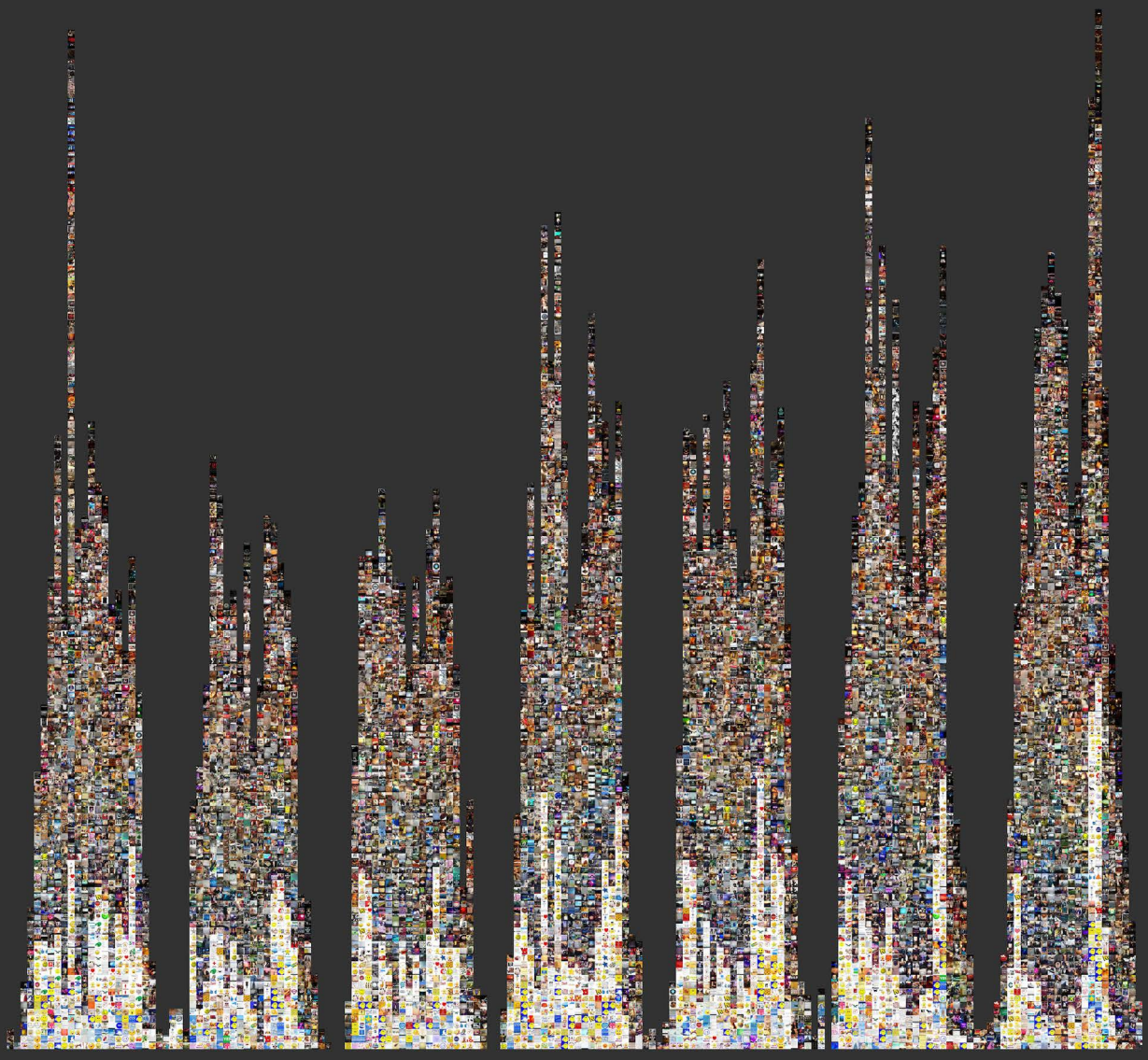


Fig 2. Image Histograms binned by hours over a week, sorted vertically by both brightness and hue.



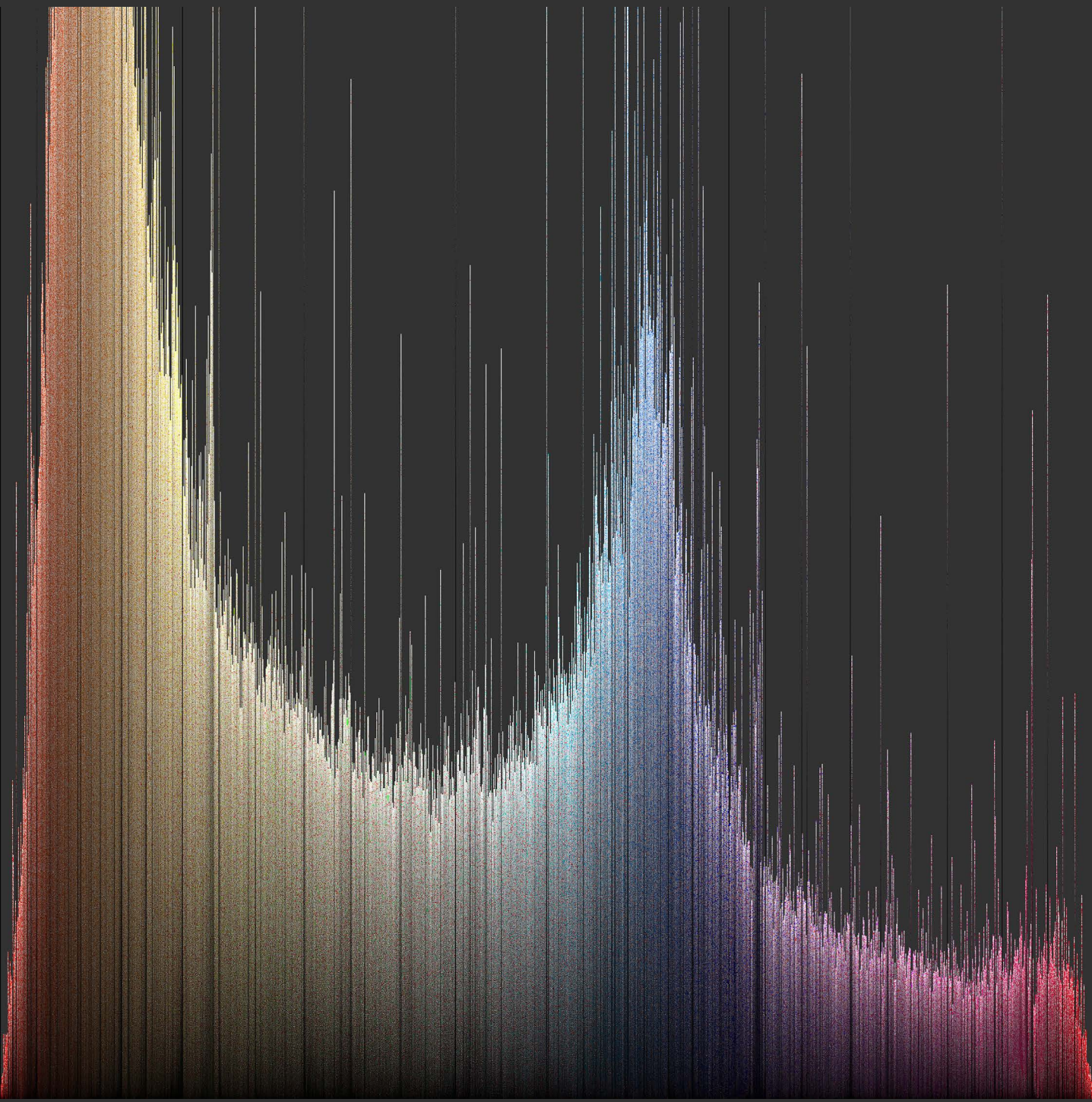


Fig 3. Hue histogram sorted vertically by brightness.

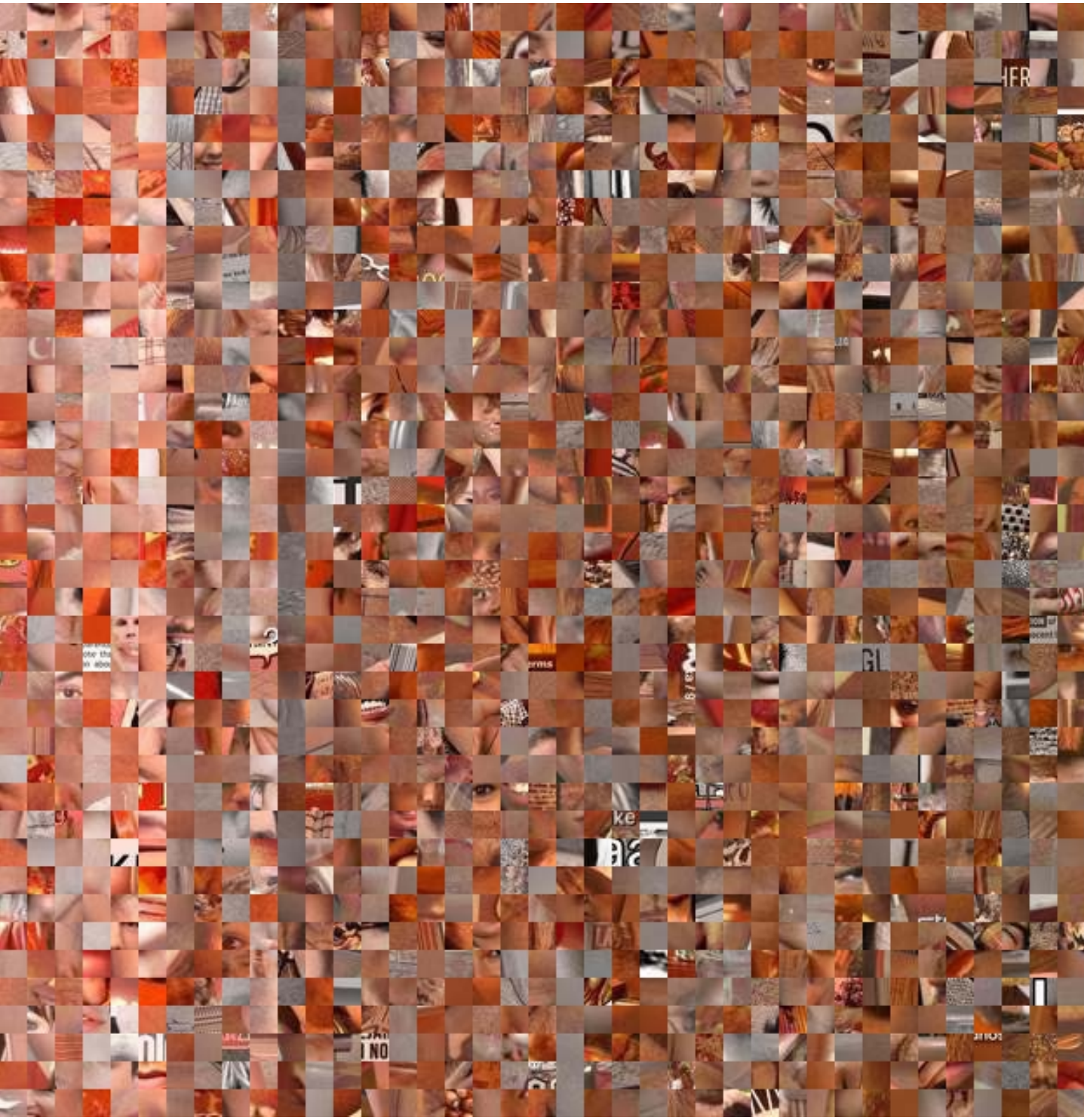


Fig 4. Close-up of slice histogram left.

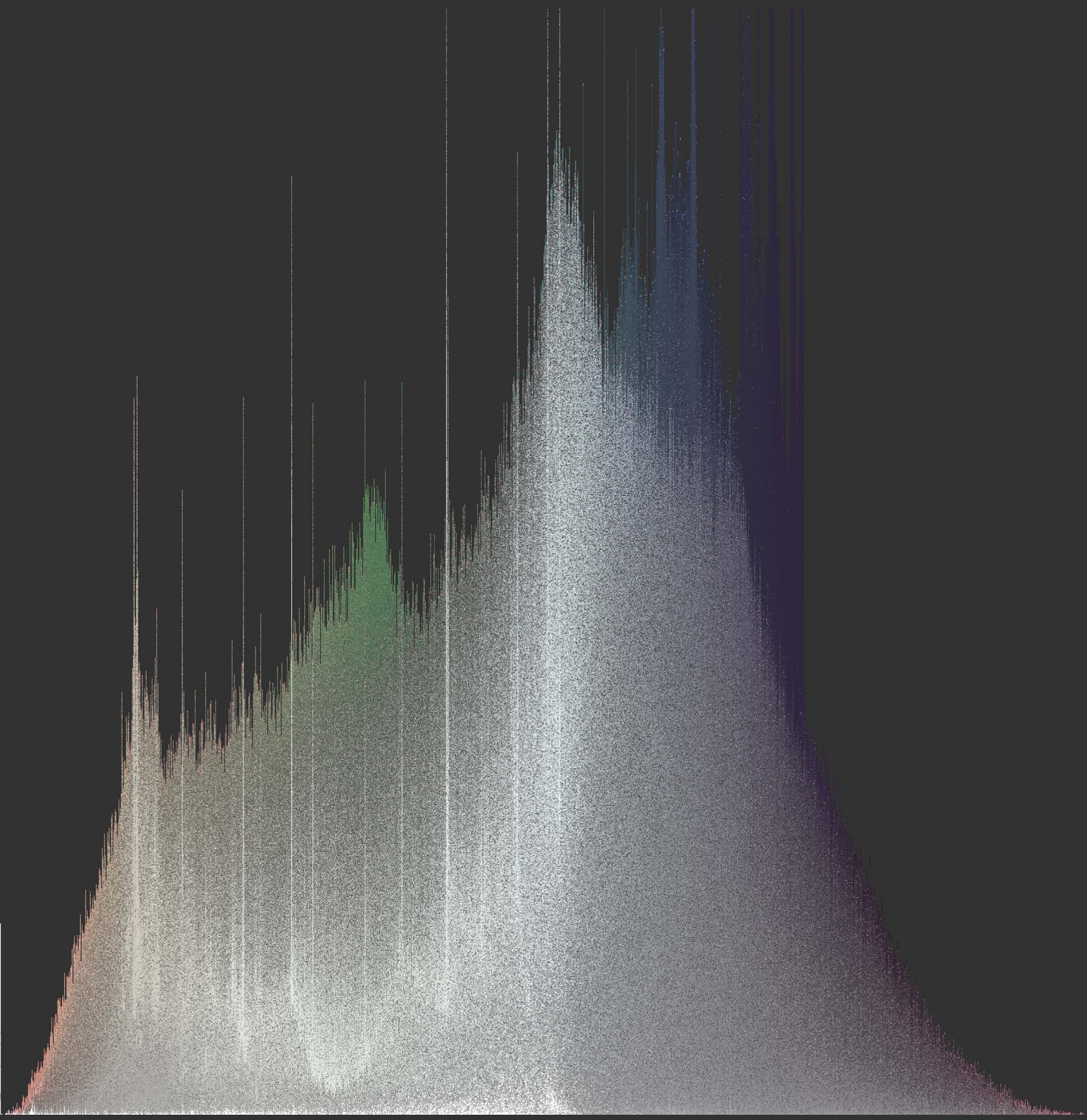


Fig 5. One million slices of a satellite image from downtown San Diego, arranged as a hue histogram sorted vertically by saturation.

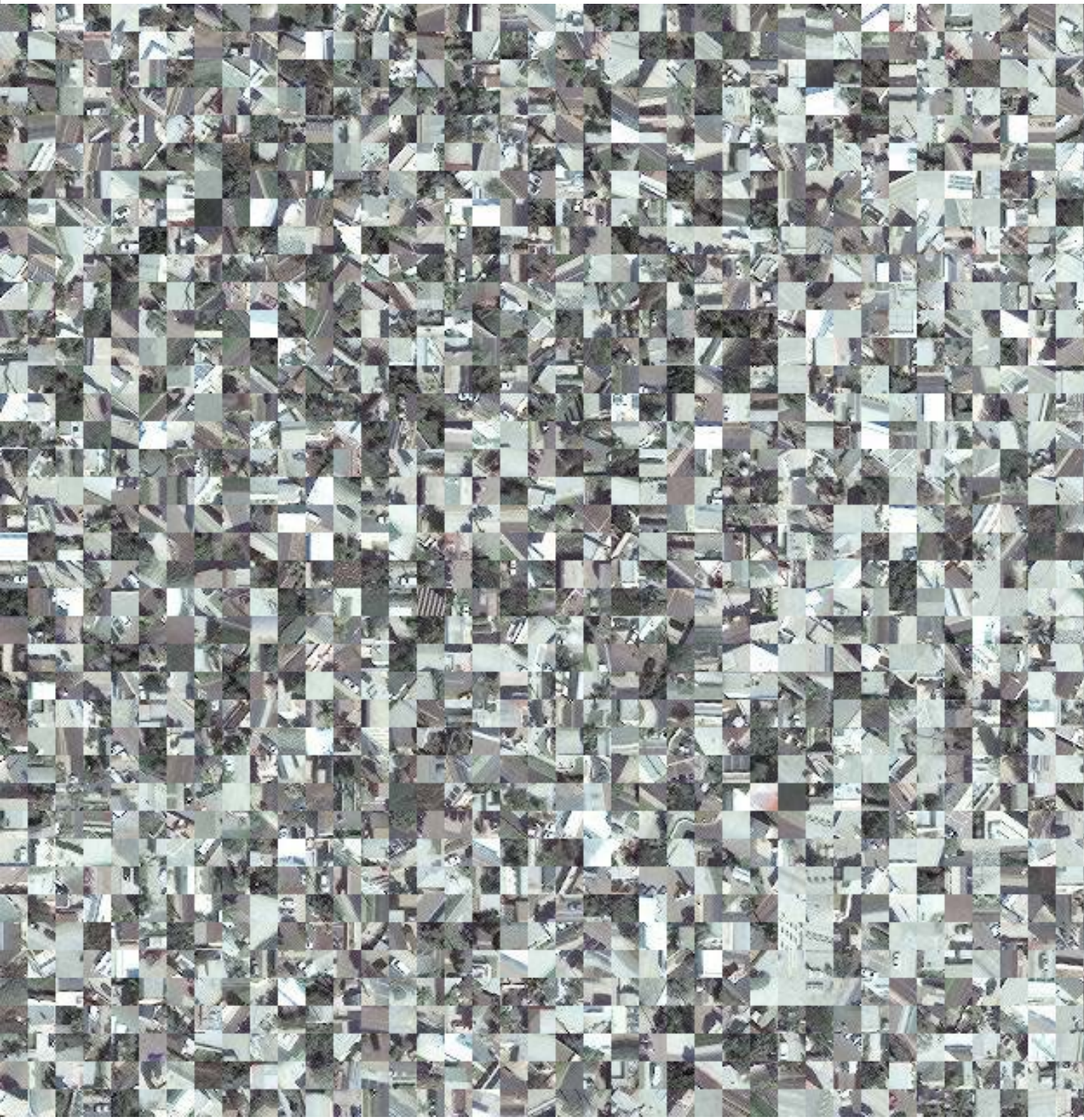
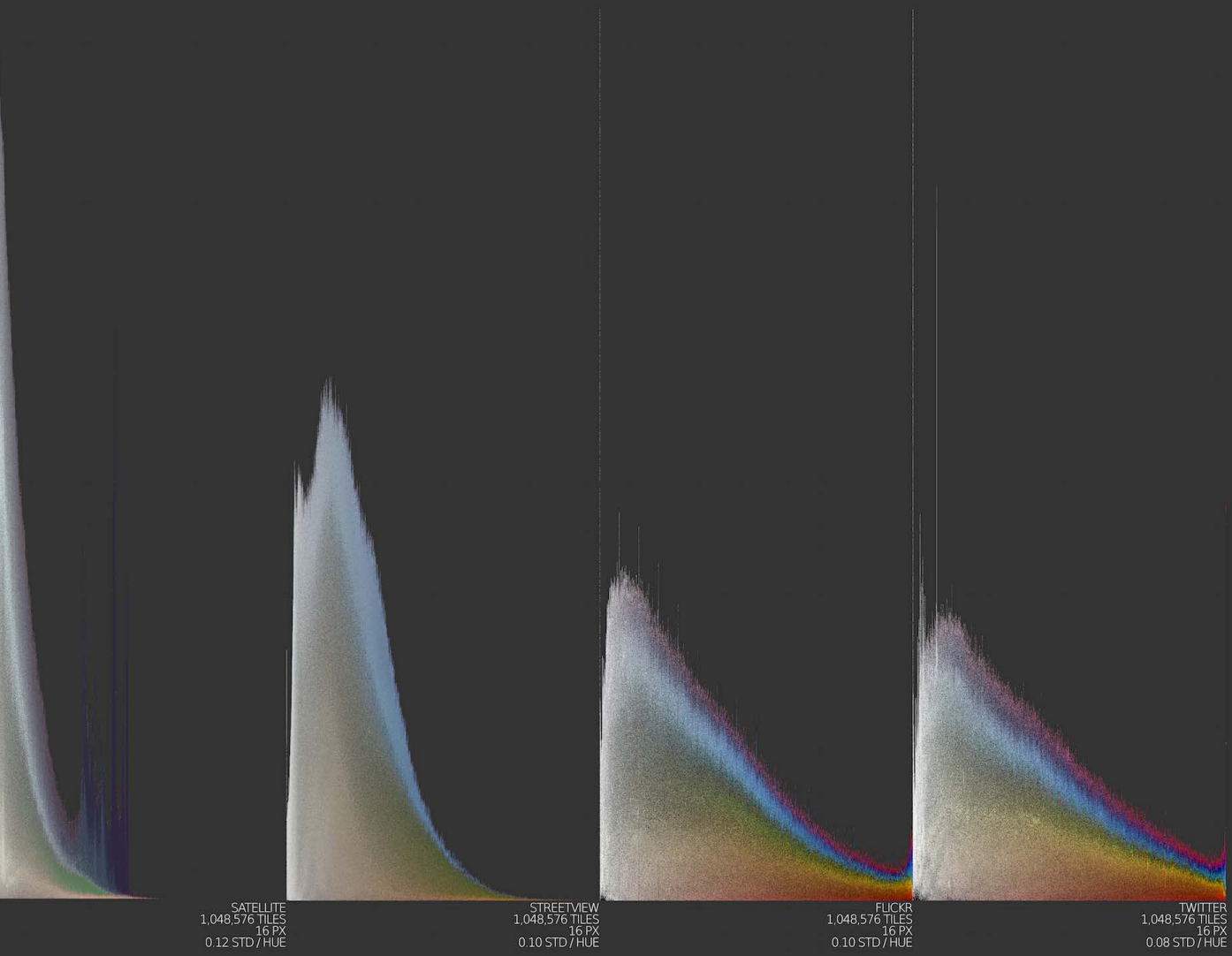


Fig 6. Close-up of slice histogram left.



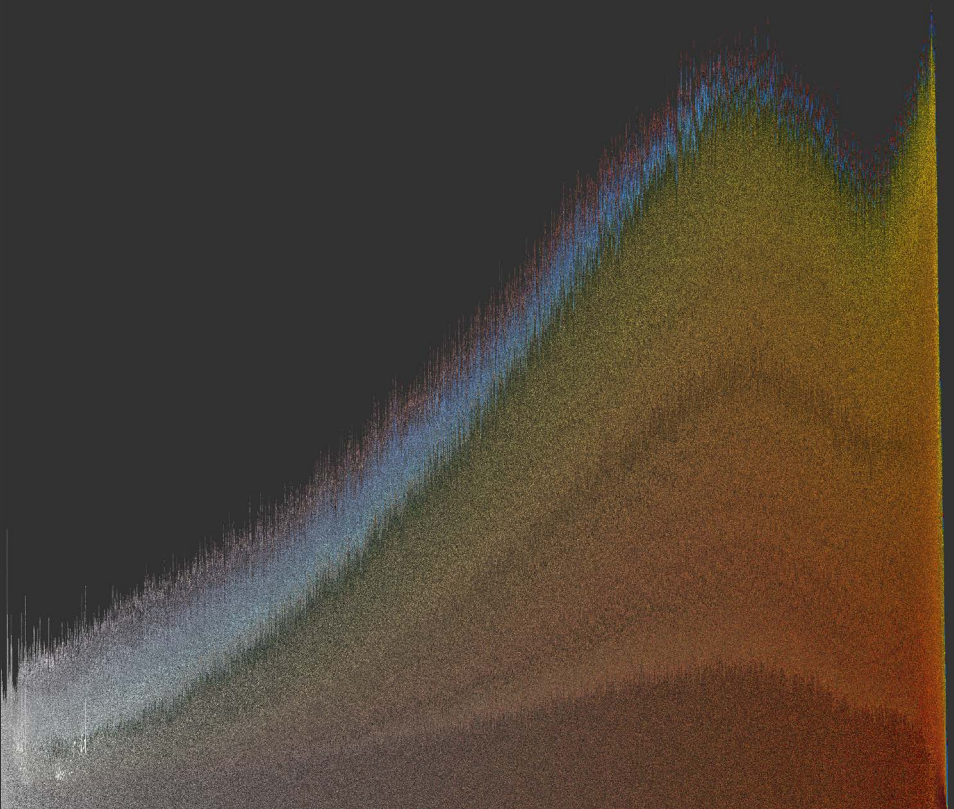
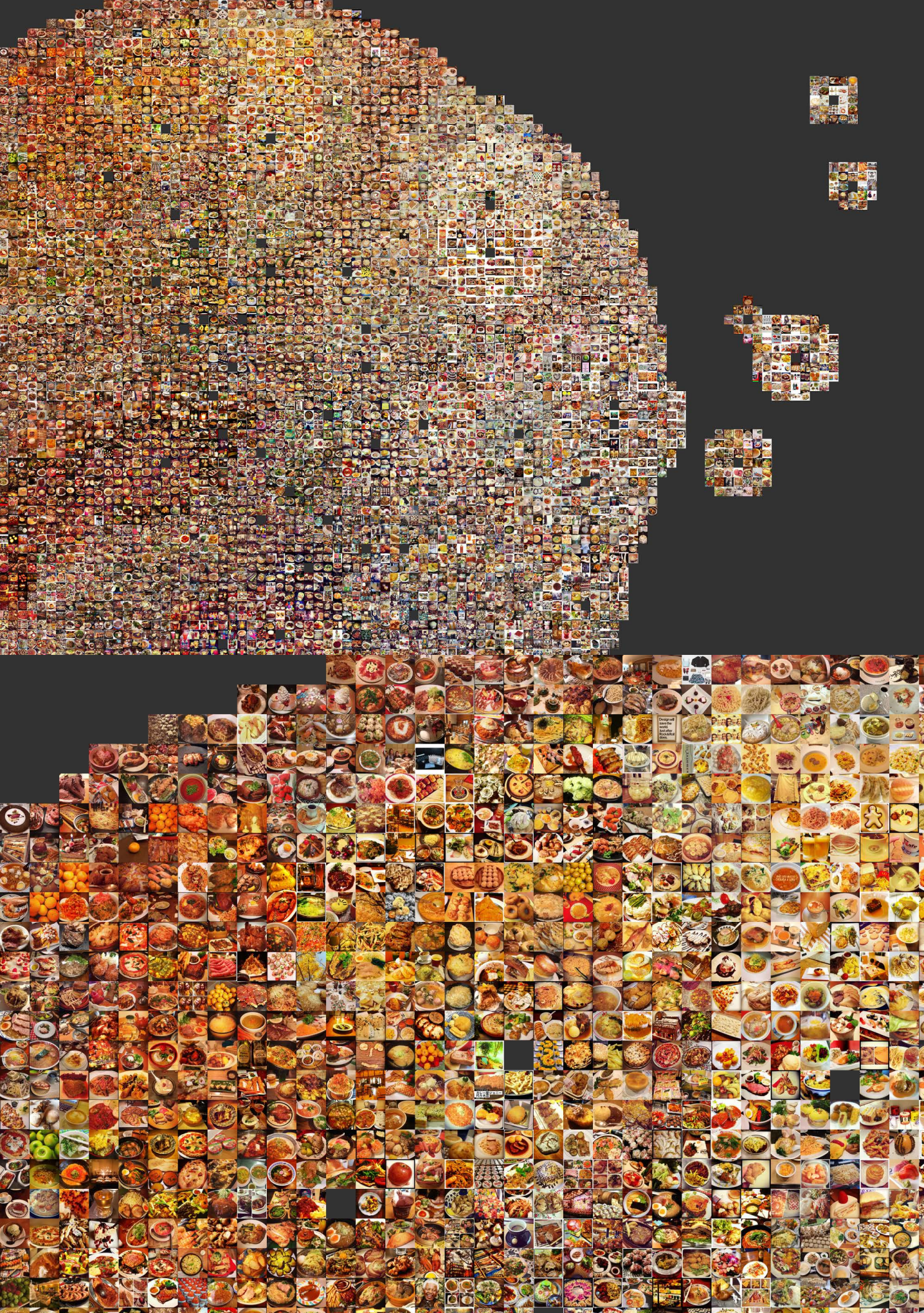


Fig 7. Slice histograms for four different sources of image data: satellite, Google Streetview, Flickr, and Twitter.

Fig 8. Slice histogram of Flickr images autotagged ‘autumn’ from New England.

Fig. 9. Growing entourage with 50 clusters of Instagram photos machine-tagged under the heading ‘food/drinks/meals’.

Fig. 10. Close-up of plot above. Empty grid squares are centroid locations.



that the plots will make a smooth (and consistent) presentation of color and that they will carry as much object content as is possible for that particular source of image data at that particular criterion value (this as opposed to slicing all types of image data into the same number of parts). As you might imagine, the more 'zoomed out' the original image data, the clearer the object content at any particular criterion value. Satellite photos give excellent results, for example (Figures 5 and 6).

The object content is an important product of this technique. Because the plot is, like all direct visualizations, composed of the images themselves, it still carries much of the information it would have carried had we used whole images—that is, unless we set the criterion value so low that our slices are uniform fields of color (even single pixels!). This would yield maximal color visibility, of course, but we'd lose all the other information. Choosing a criterion value, then, is choosing the balance between color visibility and object information. As we lower the criterion value, we look at progressively smaller parts of scenes, and gradually, they lose their structure. Just where we choose to stop this degradation will depend on our analytical goals.

It is important to note that both the image histogram and the slicing technique are very general use tools, and admit of great variation in their application. And of course, the tools have no particular analytical power without some intelligent selection of data. For example, we might want to compare the color signatures of different sources of image data for the same region (Figure 7).

Or, we might want to visualize the colors of a particular concept, like 'autumn' (Figure 8).

Additionally, in making slice visualizations, it's not essential that the visualization take the form of a histogram. What is essential is that the plot elements have low standard deviation of whichever visual properties we're interested in, and that there be some method of sorting that groups together similar elements. That's it. We can transform these histograms into any sort of plot, or montage, or map we like. In the following section, I'll discuss my efforts to expand the space of possible forms of direct visualizations can take.

Growing Entourage Plot

Since 2007, our lab has been visualizing large collections of cultural images. These visualizations have used either metadata variables such as date and location, or basic image features such as hue, saturation, brightness, number of lines, and texture. In particular, sorting by hue, saturation and brightness turned out to be very useful for quick exploration of large image collections. More re-

cently, however, we've expanded the scope of our analysis to include presence and characteristics of faces[11] and now a wide range of object and scene contents.[12]

Being able to use the latest computer vision techniques for the analysis of image content is very exciting, but it also brings new challenges. For example, how can we effectively visualize and explore the results of machine classification into many object categories? In this section, I'd like to discuss one particular method we've developed. This method visualizes high-dimensional image clusters using two dimensions. I call this method the 'growing entourage plot'.

The plots in this section are drawn from our current collaboration with Miriam Redi[13] on the clustering and visualization of large collections of Instagram images. Miriam extracted over 1000 image features that include image content (objects and scenes), photo style composition, style, texture, color and other characteristics. She then computed clusters of images using subsets of these features.[14]

The Challenge Of Visualizing High-Dimensional Data

In the field of information visualization, a great deal of energy is spent on the problem of how to present high-dimensional data on 2D canvases. There are at least three broad categories of solution: (1) preserve all features; (2) preserve some by selection; and (3) preserve some by redefinition. I'll discuss each of these in turn.

The first way of solution is simply to try visualizing everything. Such visualizations can be difficult both to design and to read. Direct visualizations are (or can be) officially of this type, but additional choices about sorting can make for big differences in readability. The effect of sorting on direct visualization is so important, in fact, that any feature not used for sorting is essentially invisible to the viewer.

The second way of solution is the one we've used most often: select some subset of features and use them for sorting. We might, for example, sort our high-dimensional image data by only brightness and hue. This is powerful and useful, but it does make invisible very complex sorts of similarity relations between images.

The third way of solution reduces the dimensionality of the data by defining a new, compact feature space. Principal Components Analysis (PCA) is the standard here, although there are others.[15] We can present images in, say, a 1000-feature hyperspace by projecting them to two dimensions. This approach has the advantage that similarity relations between images in 1000D feature space are preserved as best as possible during the projection

to 2D, meaning that our visualizations can reveal very complex sorts of similarity between images. This way of solution is quite popular and is used by our lab.[16]

Visualizing High-Dimensional Clusters

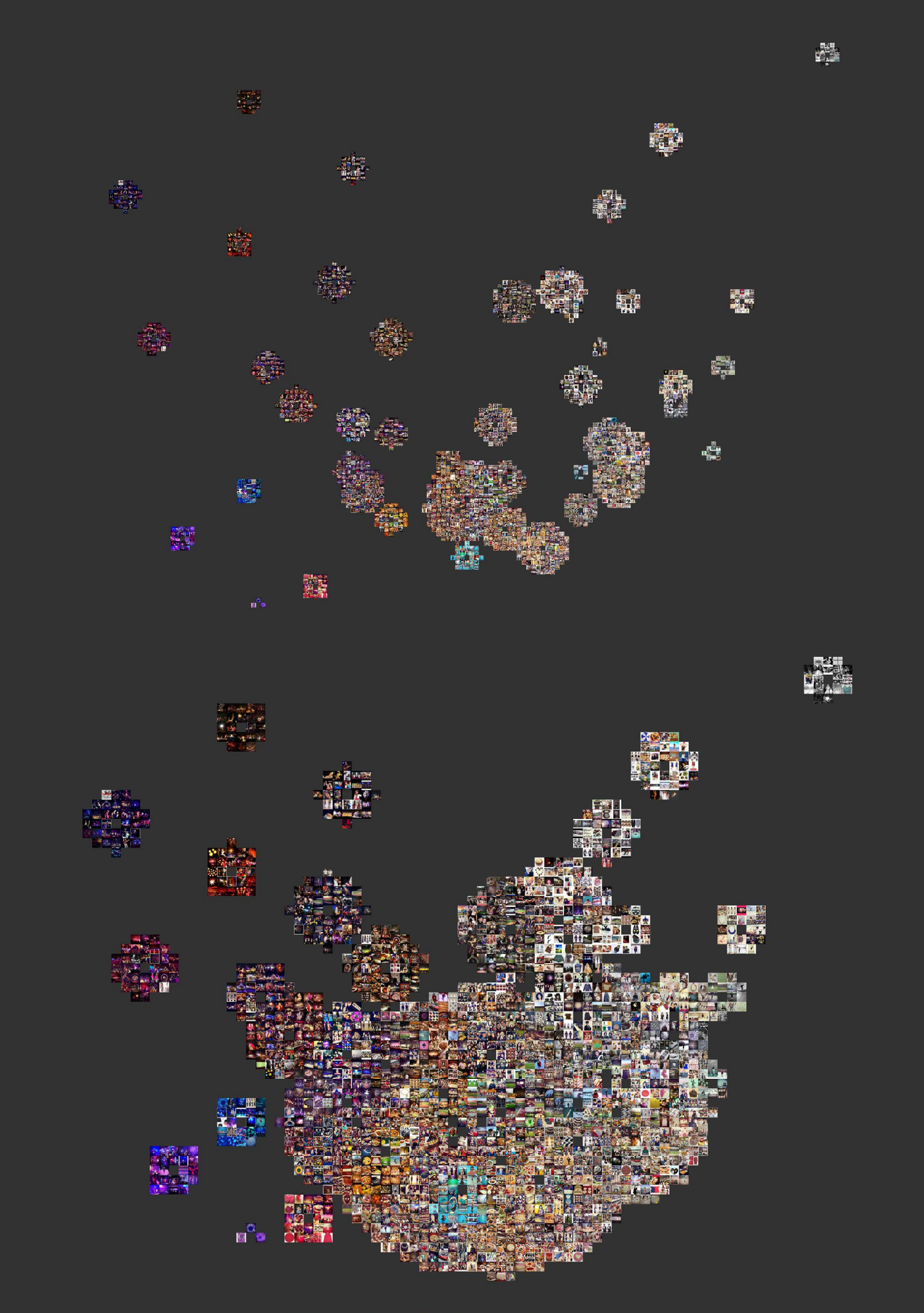
But I'd like here to talk about a different approach to dealing with dimensionality, one that is quite common in data analysis but whose use in information visualization is less common: clustering. Dimensionality reduction algorithms are powerful and useful but suffer major data loss in most cases. You simply can't preserve all the complexity of 1000D relations after projecting to 2D. Clustering algorithms, however (e.g., k-means)[17], preserve a greater share of the relational data, because they find groups of data points in your original feature space (or whichever subset you choose).

Now of course, there is still the problem of visualizing these clusters. This is particularly difficult for traditional sorts of statistical visualization, because their plot elements carry information only by their spatial positions, and the human visual system can parse a maximum of 3 spatial dimensions. Thus, if we want to see these clusters in their 'natural habitat', so to speak, we're probably out of luck. Additionally, seeing clusters of points, even in a 2D or 3D space, is not particularly illuminating, since clusters, unlike classifier outputs, are not 'classes' at all and have no conventional meaning or significance from the outset. We derive the meaning or significance of a cluster of points in a high-dimensional space from the feature values of those points, and in order to see those features in a plot, we'll have again to confront the problem of presenting high-dimensional data in 2D (or 3D). For these reasons, you don't see many cluster plots (what you'll see sometimes is PCA scatterplots with cluster memberships coded by color).

Direct visualizations are at an advantage here. Because direct visualizations use images as plot elements, a simple presentation of each cluster is actually quite illuminating, because the elements that make up the cluster are not points but images, and images, unlike the points in a scatterplot, carry information independently of their spatial positions. So even if our image data is very high-dimensional, we can present the image clusters in a simple way, ignoring the vexed matter of their spatial positions—perhaps by so many square montages—-and we nonetheless achieve some meaningful characterization of the clusters.

Fig. 11. Growing Entourage with wide grid, resulting in relatively isolated circular clusters.

Fig. 12. Growing Entourage using same data as above, but with a tighter grid. Some clusters are isolated, some have clumped with neighbors.



Growing Entourage Plot

The question now is how exactly to arrange these clusters on a 2D canvas. As I've said before, a simple presentation of each cluster is helpful. We could, for example, make square montages of each cluster and just leaf through them. But we might want more than this - we might want also to see the relations among the clusters. And now we confront a familiar problem: we have a set of data points in n-D, where n > 2, and we want to see them in 2D. The 'points' are now clusters, but the shape of the problem is exactly the same as before. The growing entourage plot is my solution to this problem. It projects cluster centroids—the middles of clusters—-to 2D and builds clusters around them by turn-taking and semantic priority.

We begin with high-dimensional image data and then use an algorithm like k-means to find k clusters in the original feature space. Each cluster has a centroid, given as a point location in the original feature space. We project the centroids to 2D using a dimensionality reduction algorithm, like PCA or t-SNE. We then bin these coordinates to a grid (making sure that no two centroids have the same grid location, which is typically not difficult). Now we have complex similarity relations among cluster centroids, and it remains only to build these clusters of images on the grid at the 2D centroid locations.

Every image in a given cluster is ranked according to its Euclidean distance[18] (in the original feature space) from the centroid. We can think of the centroid as the 'leader' of an 'entourage', and each image in the cluster is a member of the entourage. The closer they are to the centroid, by the aforementioned ranking, the closer they get to 'stand' near the centroid. Each cluster takes turns adding members of its 'entourage', starting with those closest to the leader. Each added member stands in the open grid space nearest its leader. Local conflicts between entourages are settled by this principle, since added members must occupy open grid squares.

This means that the look of the plot will depend on how we generate the original grid. We might end up with an array of circular clusters in 2D (Fig. 11), or we might end up with one large clump of images, with high-ranking members bunched up around their leaders and lower-ranking members scattered in nearby territories (Fig. 12).

This is not, of course, the only way to present clusters on a 2D canvas. It is, however, probably the best way to preserve as much of the complexity of intercluster relations as is possible in 2D. Additionally, it preserves similarity relations among images in the original feature space, something we lose by pure projection methods. Finally, it preserves intracluster relations by giving the semantically closest entourage members the privileged locations nearest their leaders.

Notes

1 This work is funded by a grant from the Frontiers of Innovation Scholars Program at UCSD. The code used to generate the visualizations described here can be found at https://github.com/damoncrockett/DataVisualizationTools/blob/master/learning.py (for the Image Histogram); and https://github.com/damoncrockett/SSI/blob/master/growing_entourage_plot/growing_entourage_general.ipynb (for the Growing Entourage Plot). These are not software packages but simple Python scripts. Please feel free to contact me at damoncrockett@gmail.com for implementation details.

2 I here use 'image plot' as a general term; it also refers to a software package developed by Lev Manovich and collaborators. See http://lab.softwarestudies.com/p/imageplot.html.

3 Jeremy Douglass and Lev Manovich, 2009. See http://www.flickr.com/photos/culturevis/4038907270/in/set-72157624959121129/.

4 William Huber, Lev Manovich, and Tara Zapel, 2010. See http://www.flickr.com/photos/culturevis/sets/72157623862293839/.

5 Lev Manovich and Jeremy Douglass, 2010. See http://www.flickr.com/photos/culturevis/4497385883/in/set-72157624959121129/.

6 Nadav Hochman, Lev Manovich, and Jay Chow, 2013. See http://phototrails.net/.

7 Content-based image retrieval systems in computer science often produce direct visualizations, typically involving small sets of images sorted by visual similarity. These systems are designed to support effective exploration and overview of image datasets (see Smeulders et al 2000 for review). The 'new paradigm' I point to here is meant to be considerably more extensive than this, although there is some clear overlap in our goals and methods.

8 See http://rsbweb.nih.gov/ij/plugins/image-montage/index.html.

9 See http://lab.softwarestudies.com/p/imageplot.html.

10 http://selfiecity.net.

11 For example, Selfie City: http://selfiecity.net.

12 For example, we've used deep learning image classification to analyze the contents of one million Twitter images (Yazdani and Manovich, 2015).

13 http://www.visionresearchwitch.com/.

14 Big thanks to the object detection team at Flickr for the object and scene tags. Their work is described here: http://code.flickr.net/2014/10/20/introducing-flickr-park-or-bird/.

15 For example, t-distributed stochastic neighbor embedding, or t-SNE (Van Der Maaten and Hinton 2008). Both PCA and t-SNE try to minimize the loss of relational information amongst points in high-dimensional space during projection to a lower-dimensional space. PCA does this by identifying axes along which the data points vary the most; t-SNE does this by privileging local relations between points over longer-range relationships.

16 See, e.g., https://www.flickr.com/photos/culturevis/albums/72157637904898314/with/10977282326/. The visualizations mentioned in note 7 are of this type.

17 K-means clustering partitions a set of data points into k clusters, for some choice of k. The algorithm attempts to minimize the intracluster distances between points.

18 Euclidean distance is the 'straight-line' distance between any two points in a space.

Bibliography

Manovich, L. (2011). What is visualization? Visual Studies, Vol. 26, no. 1, 36-49. Available at http://manovich.net/index.php/projects/what-is-visualization.

Manovich, L. (2011). Media visualization: Visual techniques for exploring large media collections. The International Encyclopedia of Media Studies Vol. VI: Media Studies Futures, edited by Kelly Gates (Chichester: Wiley Blackwell). Available at http://manovich.net/index.php/projects/media-visualization-visual-techniques-for-

exploring-large-media-collections.
Smeulders, A. W., Worring, M., Santini, S., Gupta, A., & Jain, R. (2000). Content-based image retrieval at the end of the early years. IEEE Transactions on Transactions on Pattern Analysis and Machine Intelligence, 22(12), 1349-1380.
Van der Maaten, L., & Hinton, G. (2008). Visualizing data using t-SNE. Journal of Machine Learning Research, 9 (2579-2605), 85.
Yazdani, M., & Manovich, L. (2015). Predicting social trends from non-photographic images on Twitter, 2015 IEEE International Conference on Big Data, 1653-1660. Available at http://manovich.net/index.php/projects/predicting-social-trends

Fig. 13. Close-up of Fig. 1. Empty grid squares are centroid locations.

Damon Crockett is an information visualization researcher for the Software Studies Lab at the Qualcomm Institute, UCSD. He is a recent Ph.D. in Philosophy and Cognitive Science from UCSD. His doctoral dissertation is an extended argument about the information-carrying function of human visual perception. His current research concerns the role of human perceptual judgment in information visualization.

Correspondence: damoncrockett@gmail.com