PCA Deconstruction of Images

George Daher

2025-11-30

Introduction

PCA Overview

Principal Component Analysis (PCA) is a tool that allows the dimensionality of the dataset to be reduced. This is done by decomposing the data matrix into a set of eigenvectors and eigenvalues. The sets of eigenvectors calculated in this way are orthogonal, meaning that the dot product of any 2 different eigenvectors is 0, and the dot product of any eigenvector with itself is 1. In statistics, this is important because it decomposes the data into set of uncorrelated vectors, meaning that all multicollinearity is eliminated in the Principal Component Decomposition. Then as a further result of the Linear Algebra behind this decomposition, each eigenvector explains some proportion of the overall variance in our data matrix. This proportion of variance explained is decreasing as we consider more eigenvectors of the dataset (considering all the eigenvectors will explain all the variance in the dataset). Principal Component Analysis is the idea that when a set of vectors has collinearity, by considering a smaller number of eigenvectors than the total dimension of the data, we can explain most of the variance in the data with Principal Components that are of a smaller number than the number of dimensions of the data itself.

As there is more and more collinearity in the data, then PCA will be more and more effective at decreasing the dimensionality of the data.

PCA of Images

Most images are encoded into computer memory by storing 3 color values for each pixel in the image. The actual meanings of these colors vary depending on the encoding of the image, but we can think of these generally as RGB values. We note that we are not considering opacity of the image in this case. The opacity of each pixel would add a 4th value to consider per pixel in the image. However, this is generally not an important aspect.

This means that any image can be represented using a dataset of $(width*height*3)$ values. The $width*height$ gives the number of pixels of the image, and then each of these pixels has 3 values to determine its color. There are a few ways this dataset could be organized, but importantly this dataset can be analyzed using PCA.

The main idea behind using PCA for images is that there will be a significant amount of collinearity in the dataset that can be used to represent an image. What this means is that we can generally expect different pixels in the image to be related to one another. This is easiest to think of if we have an image that is a solid color. The dimension of RGB values for all pixels in a solid color image will be exactly 0. This is because all the pixel values will be exactly the same so even though there are 3 vectors for the RGB values by pixel, because they are all constant, the dimension of the data is reduced to a point. This is more difficult to consider in a typical image, which will be much more complex, but the same idea applies.

Research Question

As explained above, an image can be encoded using $(width * height * 3)$ values. We must store this data in a matrix if we want to use PCA. There are a number of ways that this can be done. Depending on how we organize this into a matrix, the PCA will generally be different. More precisely, the PCA will almost always be different as long as the two matrices are not directly transposes or copies of each other, or transposes up to row or column permutation (swapping the order of the rows or columns, but not both).

My goal is to compare the performance of PCA when an image is transformed into a matrix in different ways. There are a very large number of possible different encodings of an image into a matrix that would differ by PCA (eigendecomposition). I've selected to focus on a few natural ways of encoding an image into a matrix. Then, I compare the performance of PCA across different types of images and different encodings, in order to determine if there is a best way to encode images into matrices for PCA, or if it depends on the type of image. In this context, best means that matrix-encoding of an images for which PCA reduces the dimension most effectively up to some proportion of variance explained by the decomposition.

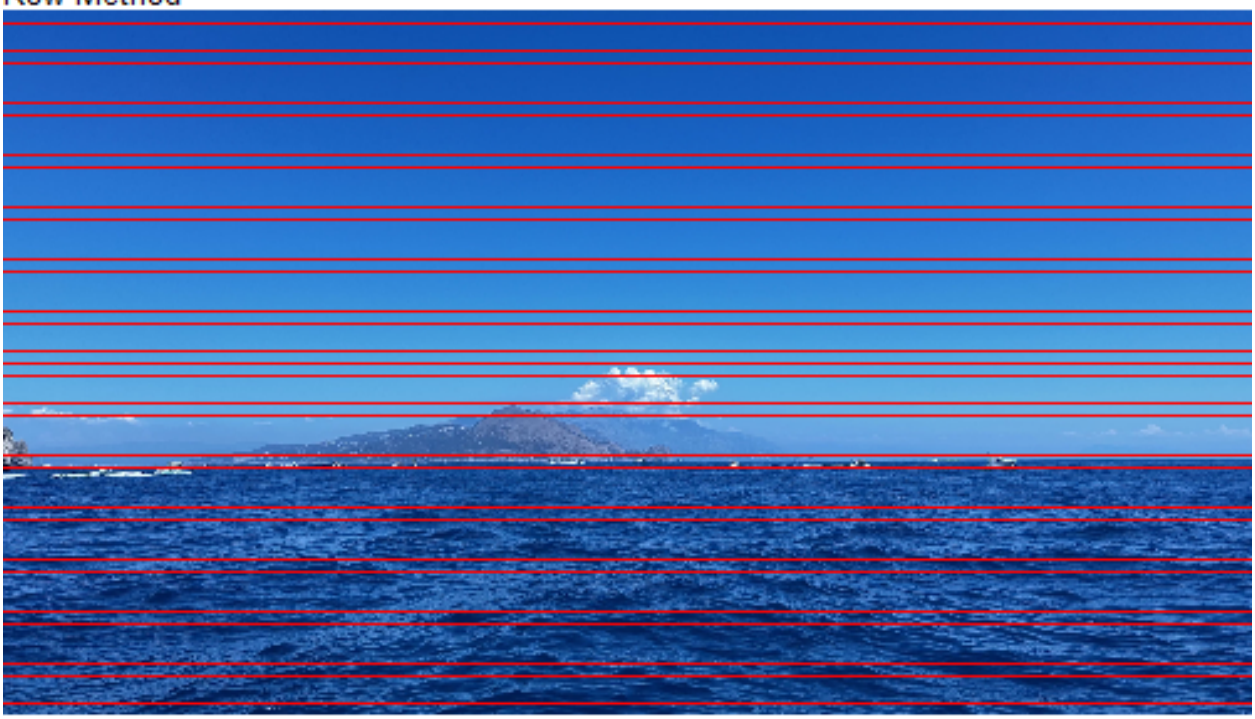
Matrix Encoding of an Image

There are 3 somewhat natural matrix encodings of an image that I considered, a decomposition by Row, a decomposition by column, and a decomposition following a grid. In a matrix decomposition according to each of these methods, a set of pixels in the same row, column, or grid square, would be grouped together in the final matrix transformation. It is important to note that the row and column decompositions are different (if they are encoded in the same way) because of the three color values for each pixel, so it is slightly more complicated than a matrix of dimensions $width * height$ of the image.

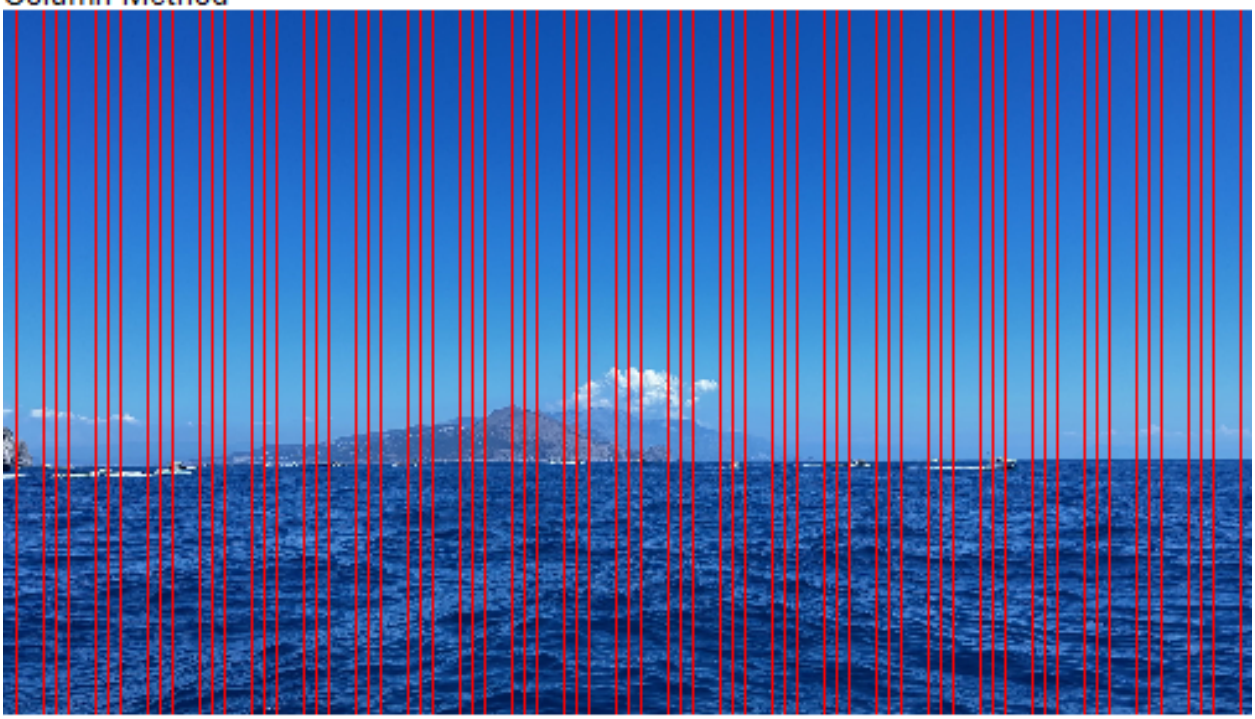
I considered two ways to put the pixel colors values into a 2D matrix for each of these transformations: in a wide format, and in a tall format. In the wide format, there would be one vector for a particular group of pixels in the image, including all three color values for each of the pixels. In the "tall" format, there would be three vectors, one for each of the three color values, for a particular group of pixels in the image. Prior to my analysis, I imagined that the relative performance of these methods would be somewhat similar, but could differ based on the colors in the image, and how they varied across row, column, or grid square.

We can visualize the decomposition in the following ways, where groups of pixels are separated by red:

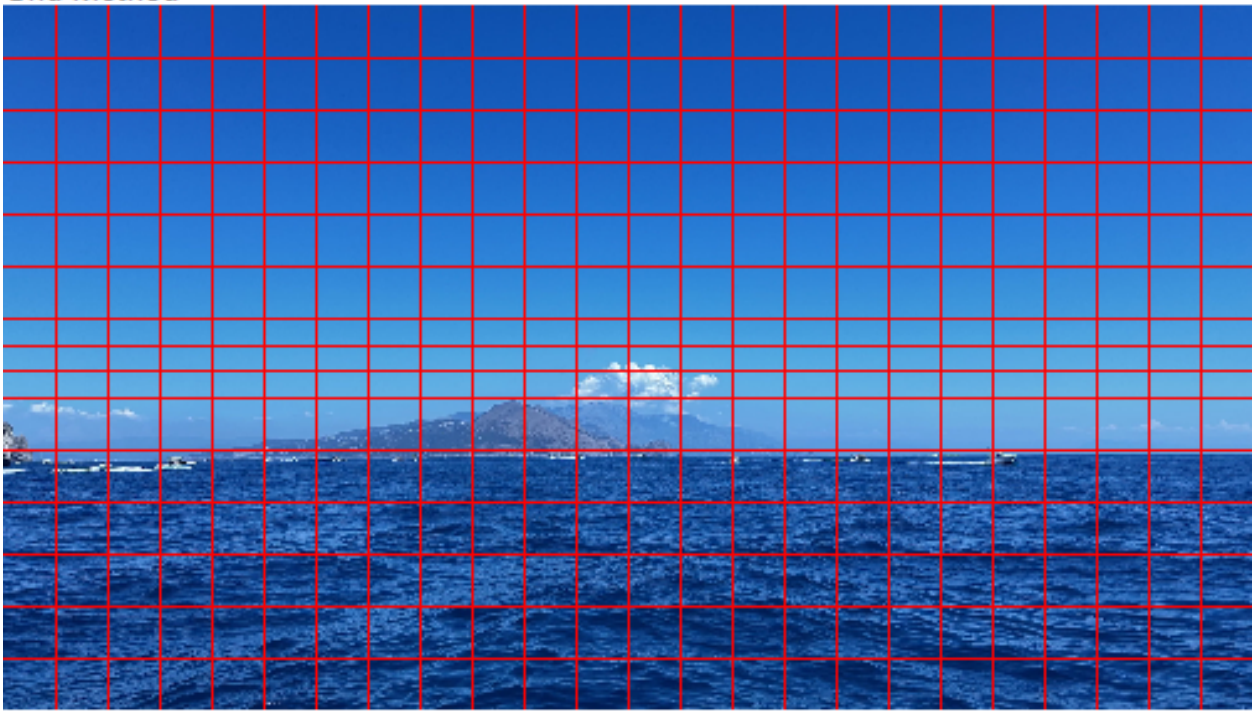
Row Method



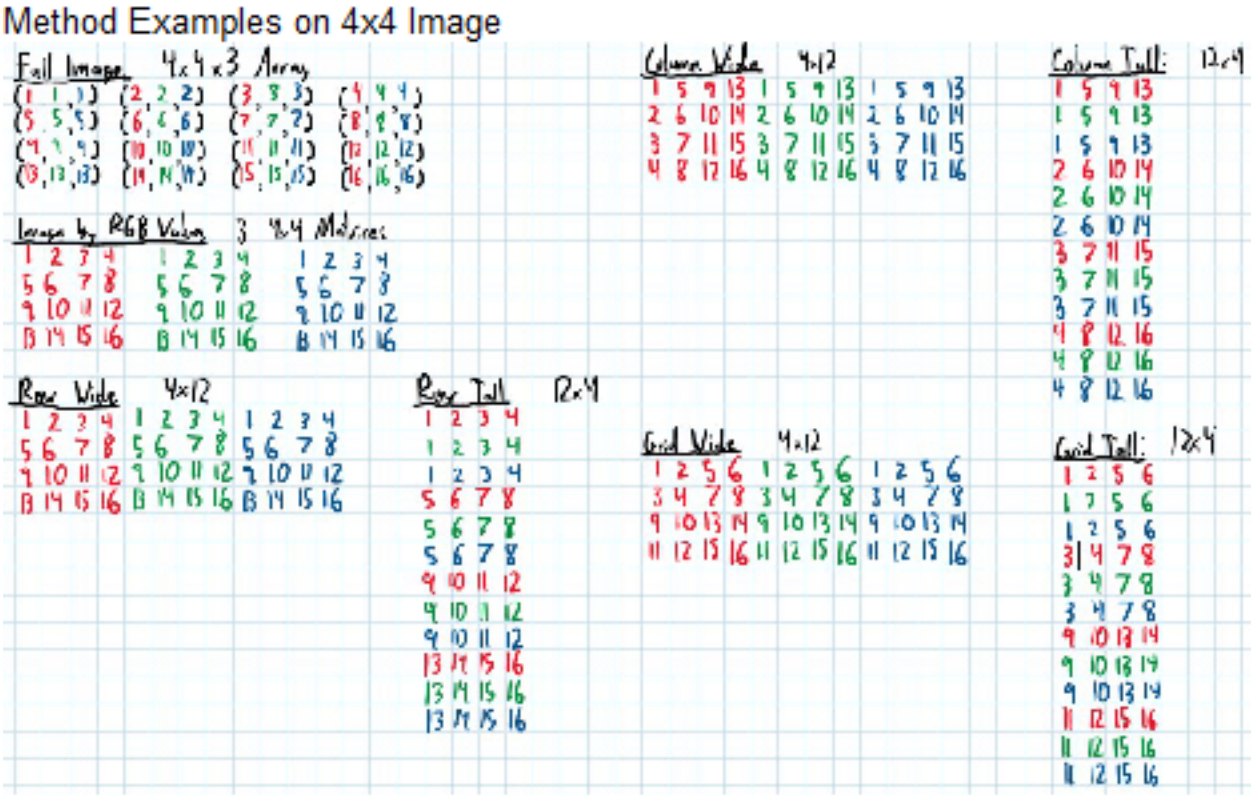
Column Method



Grid Method



Note: These column and row visualization are not completely accurate as they do not show the 1-pixel row/column slices as this is not visually clear. The images show 18-pixel row/column slices, with separating lines 2 pixels wide.



One thing to note that would be that the Wide Row and Tall Column, as well as Tall Row and Wide Column decompositions turn out to be transposes of each other. As explained before, by results in Linear Algebra, real matrices that are transposes of each other up to row permutation would have Eigendecomposition. Both of these methods were still done to compare the visual results in the image when either method was used, but in terms of variance explained per PC these results would yield the same thing.

Additionally, the size of the grid could be varied, but I fixed the grid to be made up of 40x40 squares. For a 1920x1080 image, this meant a decomposition into 1296 squares. This would lead to matrix decompositions of similar dimension for a 1920x1080 image as the row or column decompositions. I wanted to ensure this was the case because the number of Principal Components for a matrix is the number of its smaller dimension. With the row and column deconstructions, there would be 1920 and 1080 components in either case. With the grid deconstruction there would be 1296 or 1600 components, for the wide and tall versions respectively.

With these methods established, we can formally state our research question:

Is there an optimal way to decompose images into matrices for PCA?

Image Selection

The goal in selecting images was to select a variety of images that seemed to have different levels of detail, as well as different levels of horizontal and vertical variation. Based on the way the image was transformed into a matrix, certain types of variation could be accounted for in a better or worse way by PCA. For example, a image with a series of vertical stripes would require fewer principal components when transformed into a matrix by row horizontally as compared to when transformed into a matrix by column or by square on a grid. By using multiple images, selected to be different from each other (from a human perspective), a more robust comparison between methods can be made.

The images I ended up selecting were from my camera roll, including two drawings of mine and one drawing by a friend of mine. All drawings were in the same resolution, as specified before: 1920 by 1080, as I wanted this to be consistent so the number of principal components could be compared without scaling.

I also included one image that was just random noise, where all pixel RGB values were drawn from a uniform distribution. This would offer a comparison between how PCA explains the patterns in real images and drawings, as compared to how it explains the noise of a randomly generated image with no pattern.

Method

The steps needed to answer the Research Question are:

1. Extract RGB values from an image into an R dataset
2. Transform this Data into/from a matrix
3. Perform PCA on this matrix and Analyze Results
4. Reconstruct the Image using Principal Components

Step 4 is not integral to the analysis, but it offers a graphical understanding of what is happening and the fidelity of the reconstruction of the image.

Step 1 - Extracting RGB Values

In order to perform the image manipulations necessary, I used the `magick` and `imager` packages. The `imager` package allows an easy way an loading an image into R as an array with 3 dimensions: width, height, R/G/B. By default, each of the R/G/B values ranges from 0 to 1. RGB values are often considered on a scale from 0-255, but the difference in scales makes not difference, as scale is automatically accounted for in PCA.

Step 2 - Transforming the Data into a Matrix

I choose to look at 3 different main methods for converting the image to a matrix, and then 2 versions of each method. For each method I implemented a "wide" version and a "tall" version. The wide version puts all R/G/B values for some set of pixels in wide format as one row of the resulting matrix. The tall version puts R/G/B values for some set of pixels in tall format as three rows of the resulting matrix, one for R, one for G and one for B (this is same as transposing the row version). For each of these 6 methods, I created two functions, one to transform the image array to a matrix, and one to transform the matrix back into the image array. This way the image could be recreated from a matrix of the data reconstructed using some number of Principal Components from the PC deconstruction.

Step 3 - PCA

Principal Component Analysis is a built-in functionality for R with the function `prcomp()`, and this includes using a PCA to reconstruct the data matrix using some specified number of Principal Components.

One of the outputs of the `prcomp` is a matrix of importance that summarizes the proportion of variance explained by each particular Principal Component, and the cumulative proportion of variance explained by each number of Principal Components (as they are organized from most to least important). By analyzing the cumulative proportion of variance we are able to compare the performance of Principal Component Decomposition of an image using multiple different methods.

We can compare the performance between methods for a single image by comparing the lines of cumulative proportion of variance explained versus the number of principal components used. Whichever line shows the

highest cumulative proportion of variance explained for a certain number of PCs indicates the best method for a particular image. For this comparison, we can also consider scaling by the number of total principal components that the PCA extracts based on the image transformation technique used.

Determining overall best performance is slightly more difficult. We can determine the best performance based on our selected sample of images. However, this sample was not selected randomly, and was instead selected to try to account for images with different levels of detail, color profiles and horizontal/vertical variation.

If we want to determine overall best performance, we would need to establish the population of images we are considering. We could generally say all images, or possibly images people take on their phone, or something else. Then we would need to somehow select a representative sample of this population, which would be fairly difficult to do.

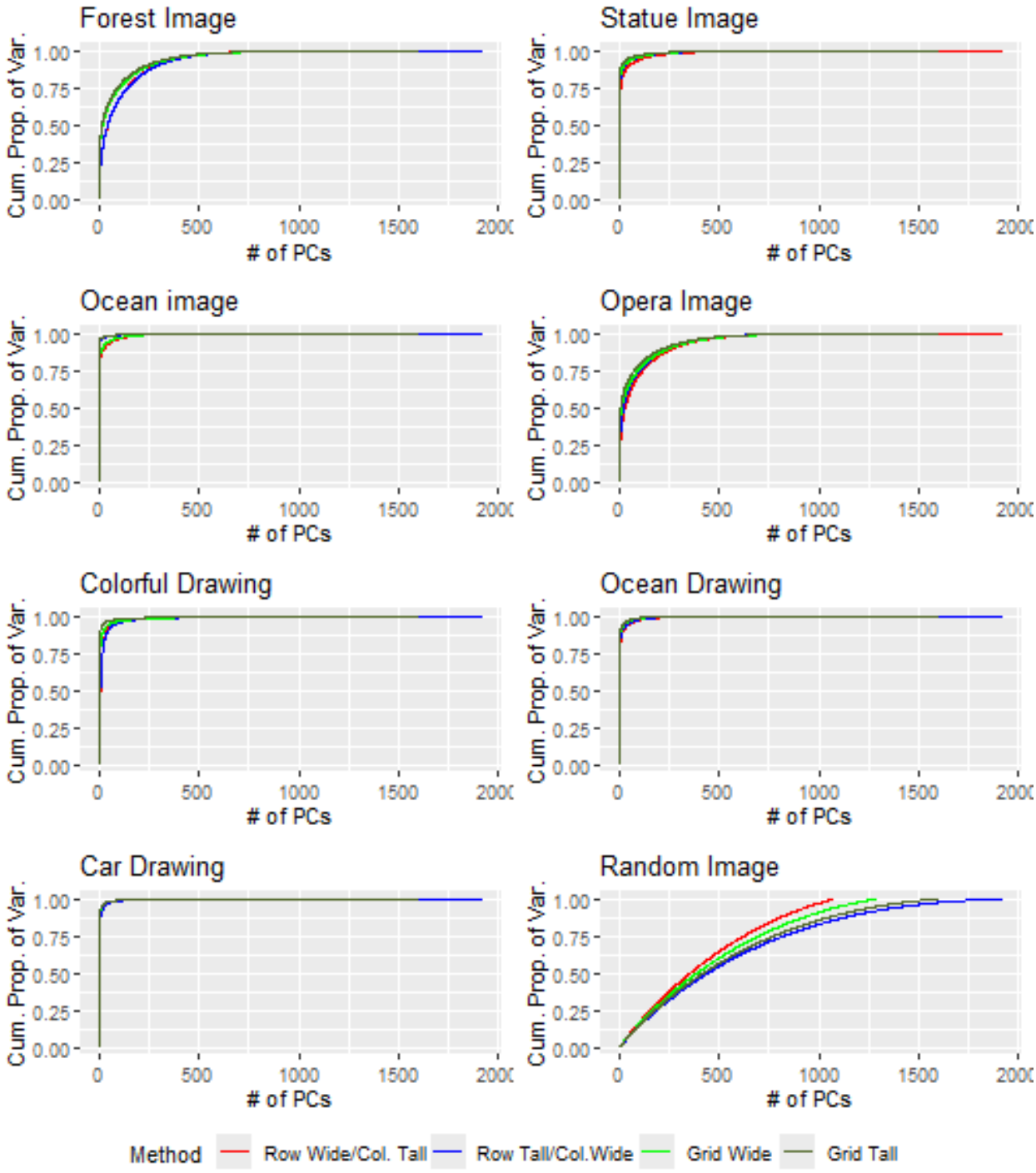
For the purposes of this study, we are considering the variety of images explained above.

Variance Analysis

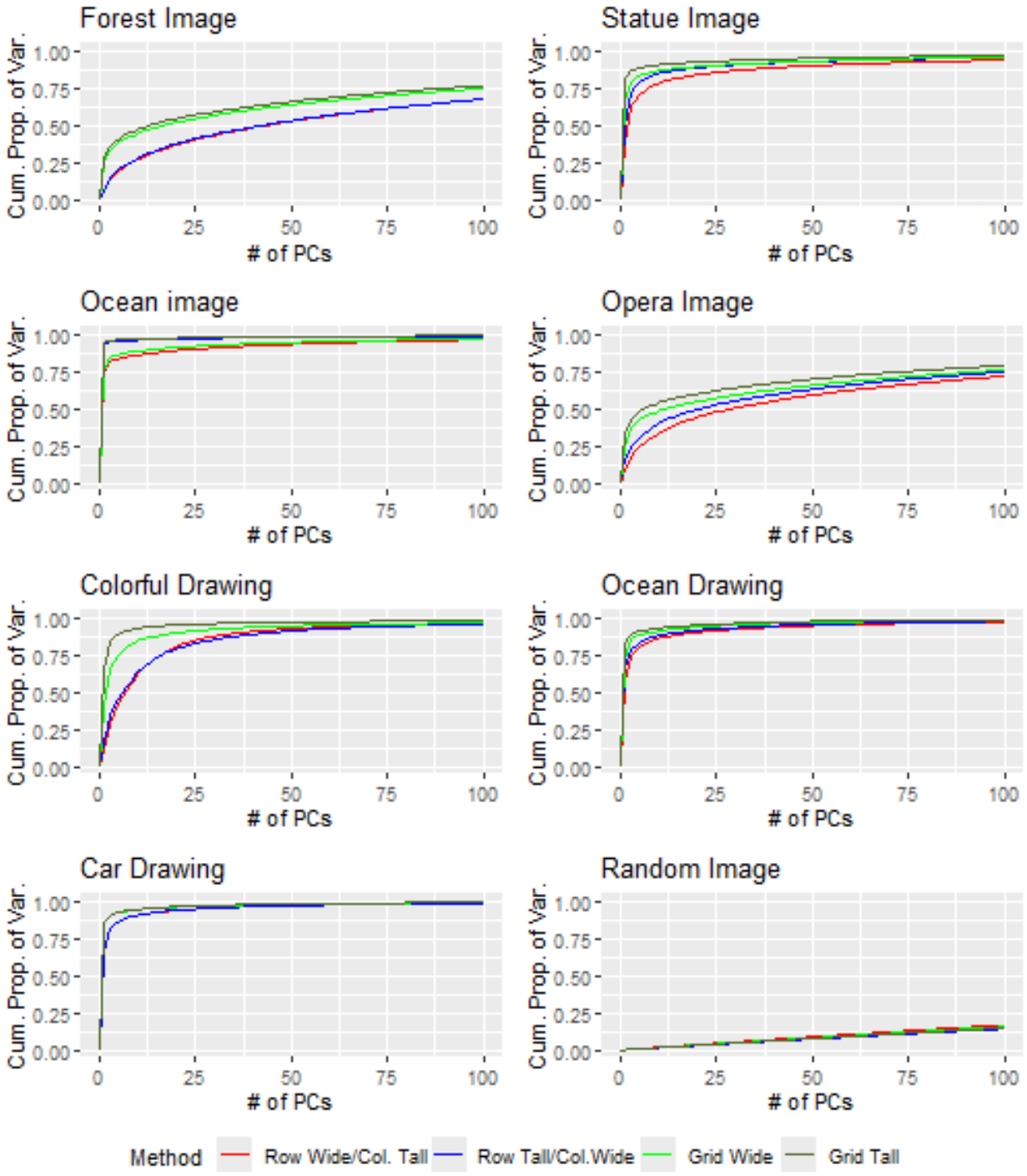
Before any analysis, we can note that the proportion of cumulative variance explained converges to 1 as we add principal components. We want to compare the performance of the methods with only a smaller number of PCs included.

By Image

We can first focus on comparing the performance between methods across all the images.

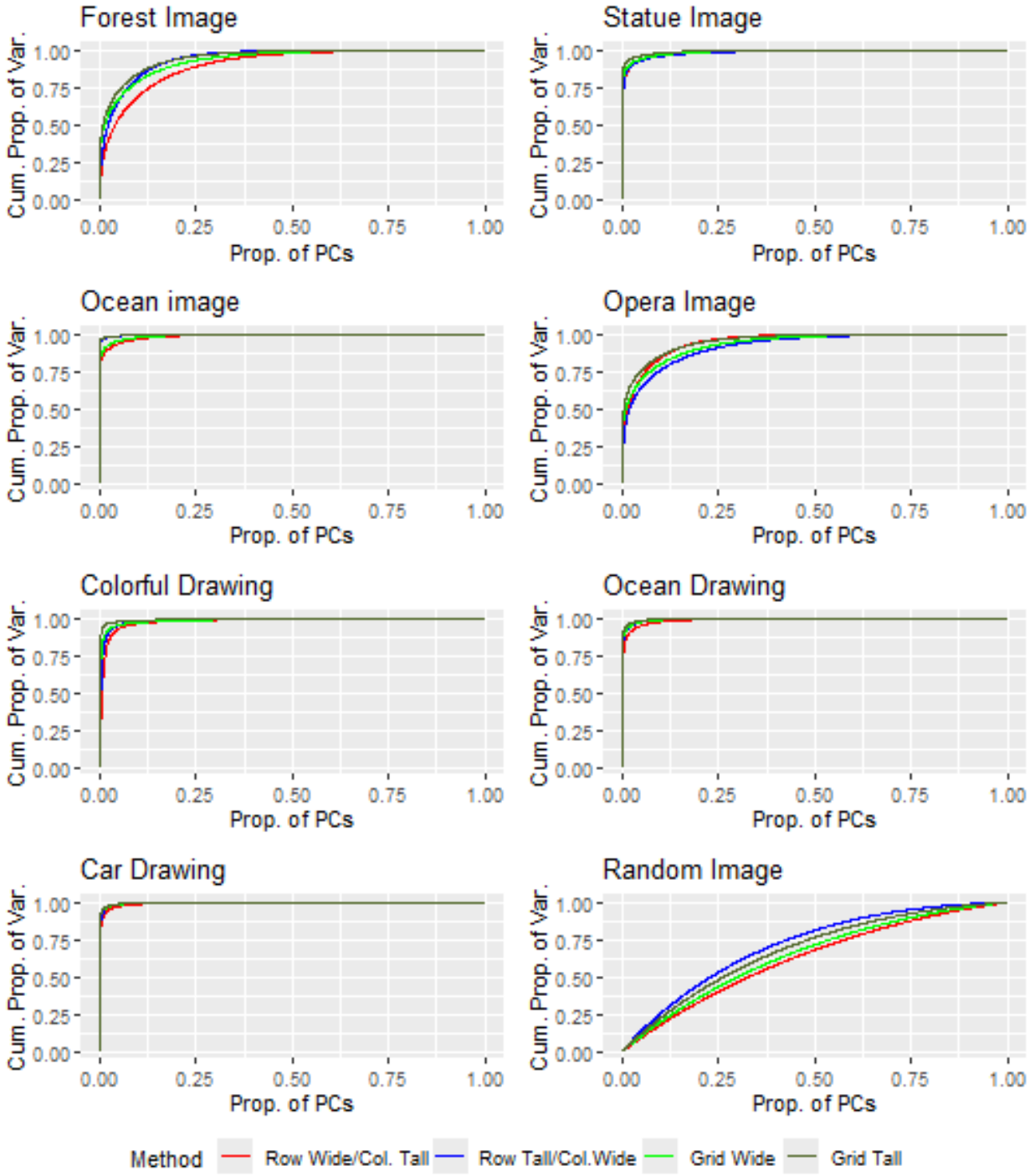


We can see that most of the variance is accounted for in the first few hundred PCs, except in the random image where there is no pattern. We can shrink the x-axis and focus on the first few hundred PCs, as the cumulative proportion of variance explained always converges to 1 and there is no clear visible separation in the Proportion of Variance explained between methods for any of the images besides the random image. The random image results may not actually indicate a separation between methods, but rather show the difference in the maximum number of principal components each method takes. In the random image, because of the complete randomness, we expect the proportion of variance explained by each Principal Component to be more constant than in the images where patterns are present, so the methods that admit more possible components in the PCA will explain less variance per principal component.

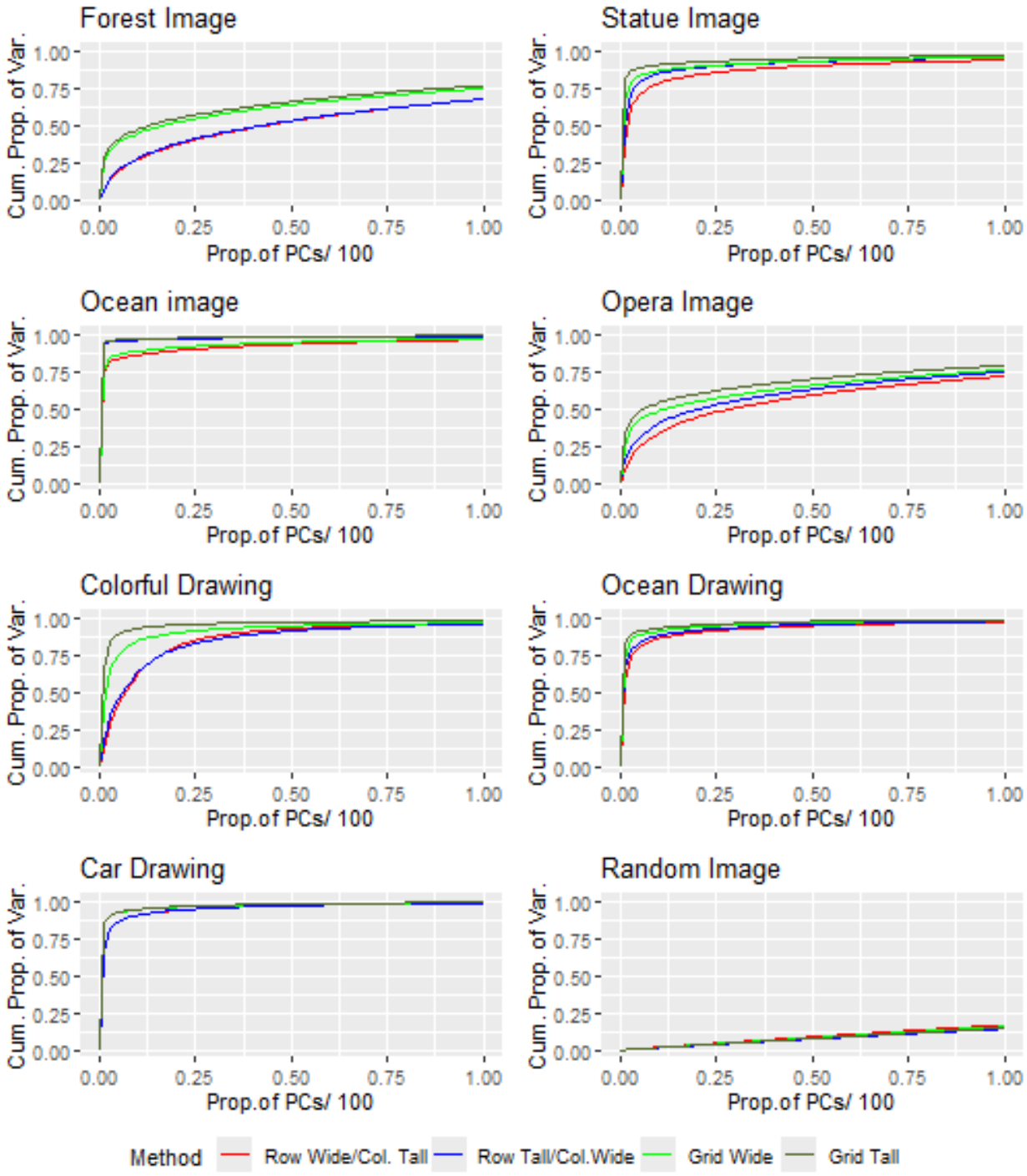


Even the first 100 PCs explain most of the variation in most of the images. There is not a large difference between the methods for proportion of variance explained as PCs are added. However, we can see that the Grid Tall method seems to yield the best results for all the images and drawings. Then the Grid Wide method is second best, and after that are the Column/Row methods, with Row Tall/Col. Wide being generally better than Row Wide/Col. Tall. There is no clear separation on the random image, so while still considering the prior analysis, there does not seem to be a difference between methods on the random image.

By Image - Scaled



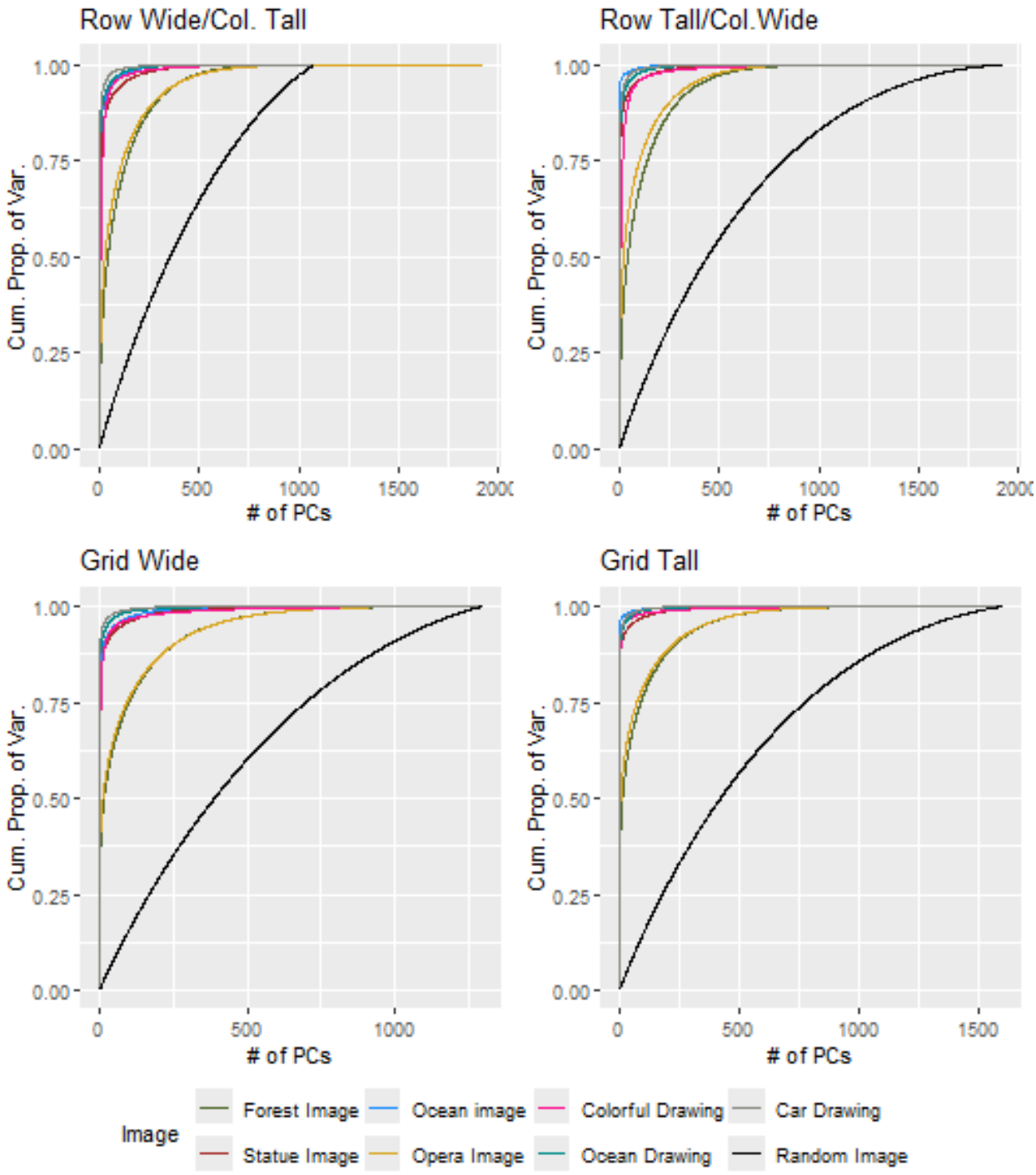
When we scale by the number of components in the PCA, the comparison between methods changes for the random image. However this is only present to establish a baseline.



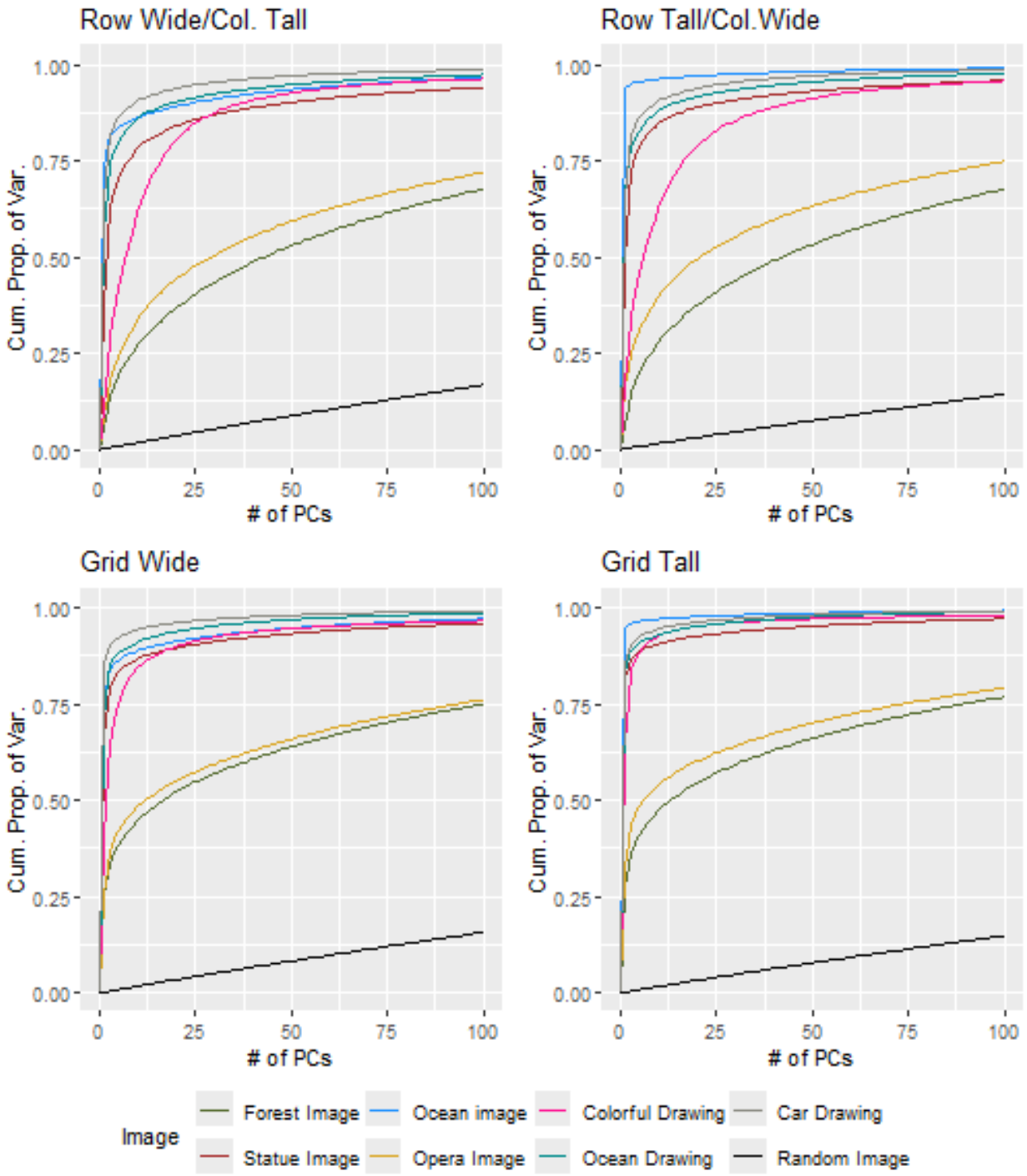
When plotting to the proportion of variance explained versus proportion of PCs used, it becomes clearer that methods in order from best to worse are: Grid Tall, Grid Wide, Row Tall/Col. Wide and then Row Wide/Col. Tall. Interestingly, the vertical images Statue and Opera, do not have different relative performance between Row Wide and Row Tall as the two horizontal images Forest and Ocean.

By Method

We can then focus on comparing the performance between images across the all the images.



We can see that by comparing the images by method, there is a clear pattern to the complexity of images according to the PCA deconstruction. All the images can have most of their variation explained by a few hundred PCs again. The more detailed images, Forest and Opera, require significantly more PCs to explain the same amount of variation. The random image requires a very large number of PCs to explain its variation, which comparatively makes sense as there are no overarching patterns that could reduce the dimensionality of the image matrix.



Even the first 100 PCs explain most of the variation in most of the images. There is not a large difference between the images other than Opera, Forest and Random for proportion of variance explained as PCs are added. The ordering of the images in terms of most variance explained by fewer Principal Components differs by method. However, there is some separation present. We can see that the car drawing, ocean image and ocean drawing are the simplest images in terms of most variance explained by fewer Principal Components. These are then followed by the statue image and colorful drawing, and then the opera image and forest image, followed finally by the random image.

By Method - Scaled

Scaling in this case is not important as the methods yield the same number of components per image when analyzed using PCA, so scaling is unnecessary. The result would be the exact same as the unscaled variance graphs by method, in the prior section.

Results

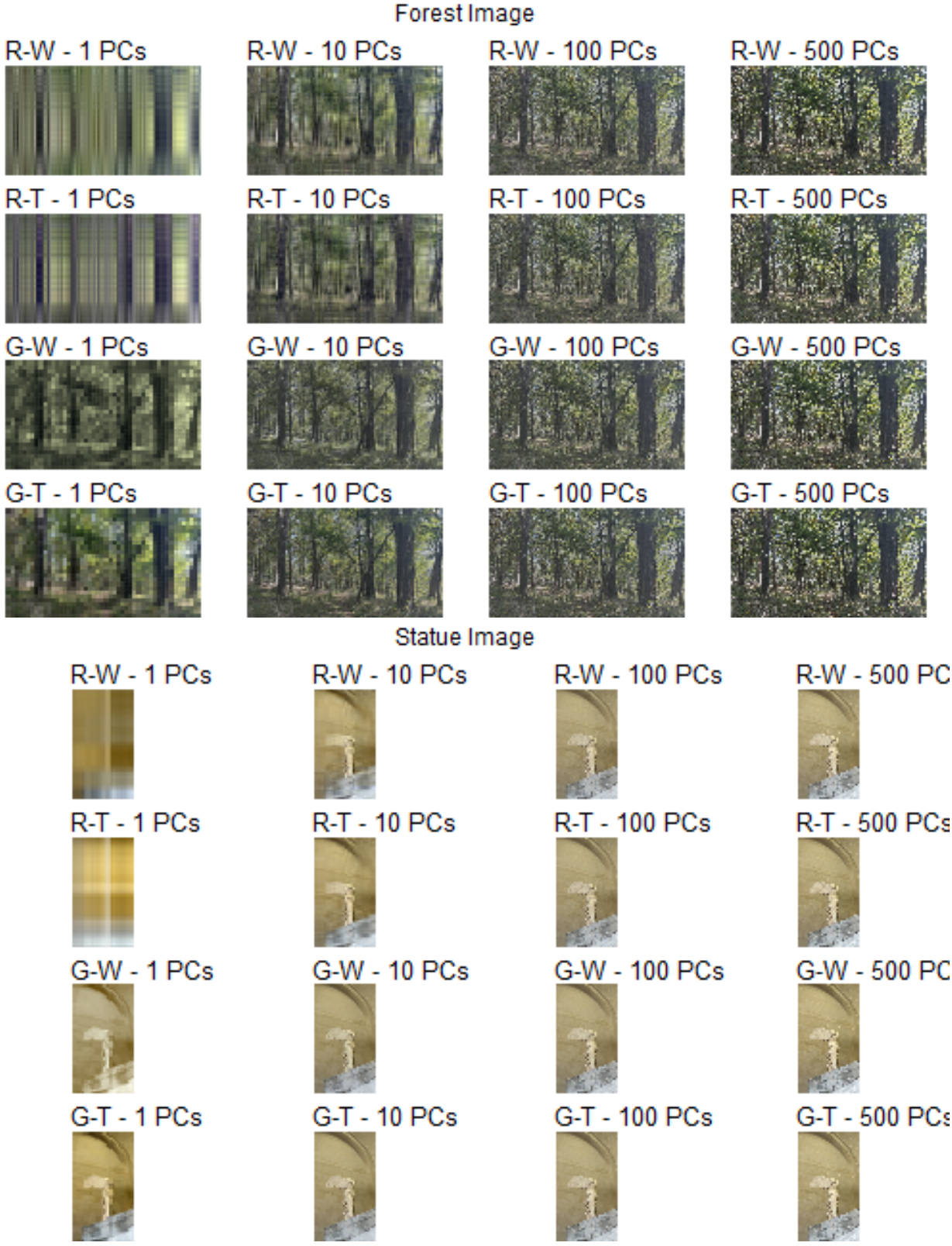
From analyzing the cumulative proportion of variance explained by the number of Principal Components considered, we are able to find some notable results:

First, it seems that the Grid tall transformation of an image into a matrix is the best method in order to decompose any type of image effectively into a small number of principal components. This is followed by the Grid Wide method, which often offers comparable performance. Then, noticeably worse, we have the row/column deconstructions that offer similar performance between the two versions. These results hold generally, to slightly different severity, when accounting for the relative difference in total number of Principal Components in the PCA per method (scaling), and not (not scaling).

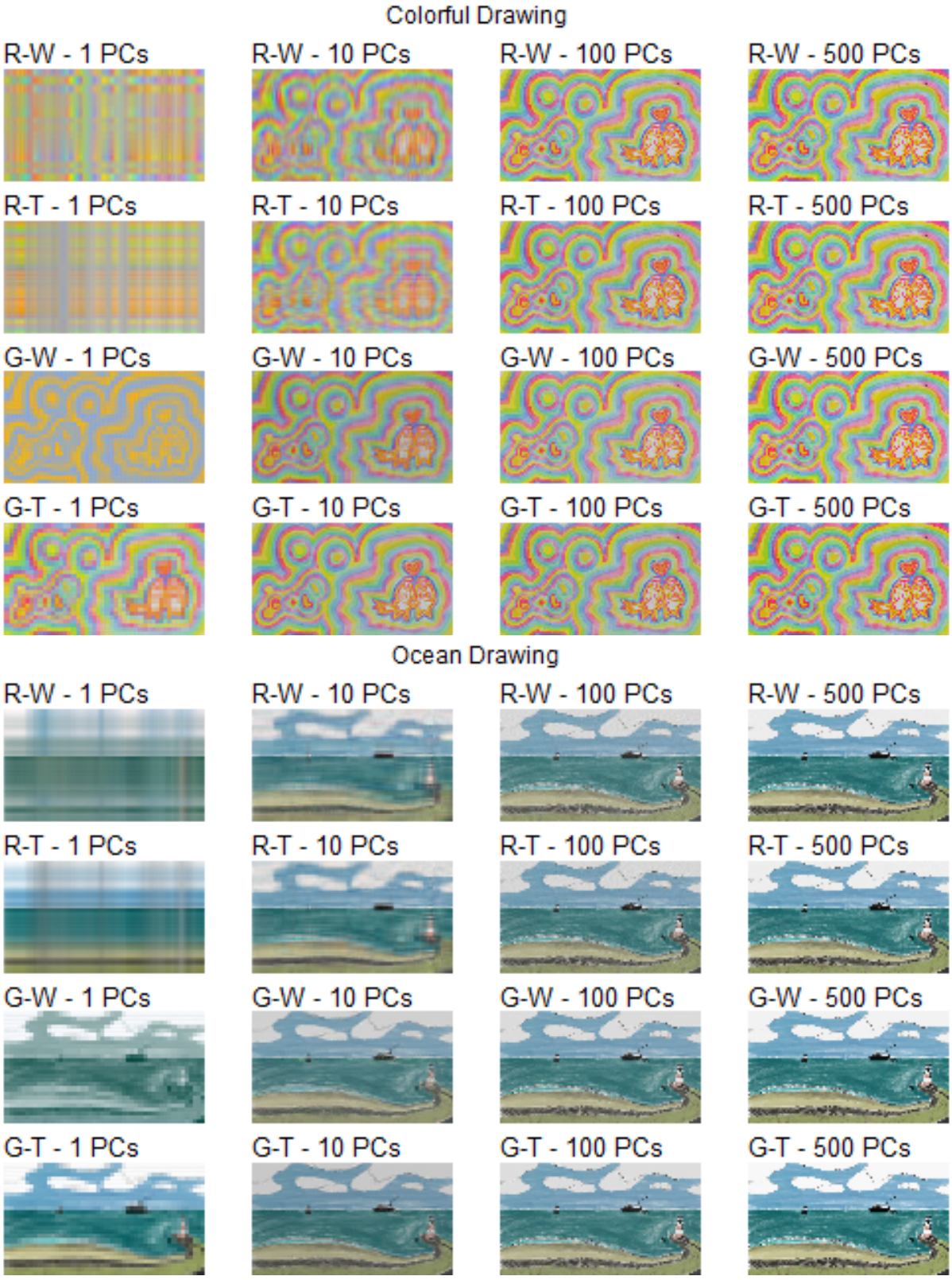
Second, the visual complexity of the images seems to be mostly explained by the number of principal components needed explain the variation in the image. The drawings generally required fewer principal components, as they were generally less detailed. Additionally, the two images with a simpler detail: the ocean image and statue image, both required a lower number of principal components to explain the variance than the rest of the images.This contrasted directly with the higher number of principal componets required to explain the variance in the more detailed images: the forest image and opera image. Finally, the image that was purely random pixels was not explained well at all by PCA, which makes sense as there were not patterns present in the image that could be extracted by their multicollinearity.

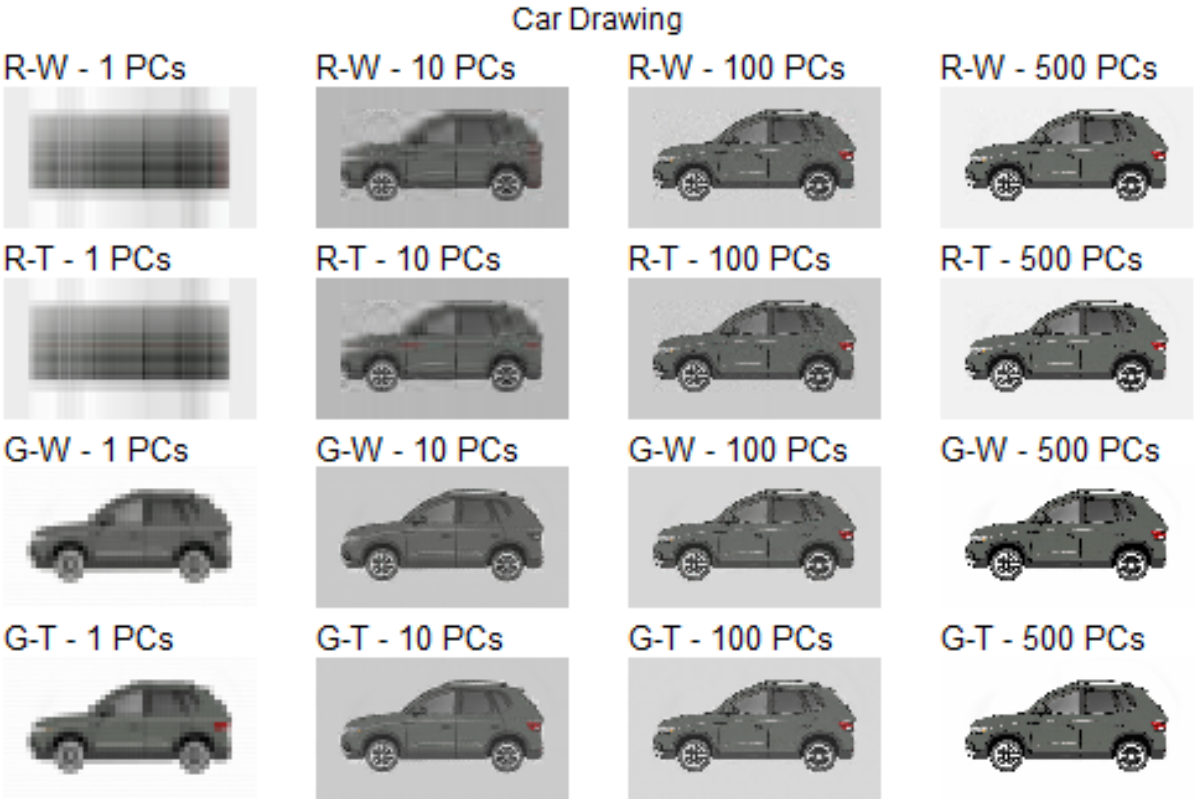
Visual Results

We can then compare the results found by analyzing the numerical variance found in the PCA with the visual results from reconstructing the images using various numbers of principal components. The visualization here is somewhat flawed as the images are shrunk in order to fit into the document for comparison, but the important trends can still be observed. ## By Image


Forest Image
R-W - 1 PCs
R-W - 10 PCs
R-W - 100 PCs
R-W - 500 PCs
R-T - 1 PCs
R-T - 10 PCs
R-T - 100 PCs
R-T - 500 PCs
G-W - 1 PCs
G-W - 10 PCs
G-W - 100 PCs
G-W - 500 PCs
G-T - 1 PCs
G-T - 10 PCs
G-T - 100 PCs
G-T - 500 PCs
Statue Image
R-W - 1 PCs
R-W - 10 PCs
R-W - 100 PCs
R-W - 500 PC
R-T - 1 PCs
R-T - 10 PCs
R-T - 100 PCs
R-T - 500 PCs
G-W - 1 PCs
G-W - 10 PCs
G-W - 100 PCs
G-W - 500 PC
G-T - 1 PCs
G-T - 10 PCs
G-T - 100 PCs
G-T - 500 PCs

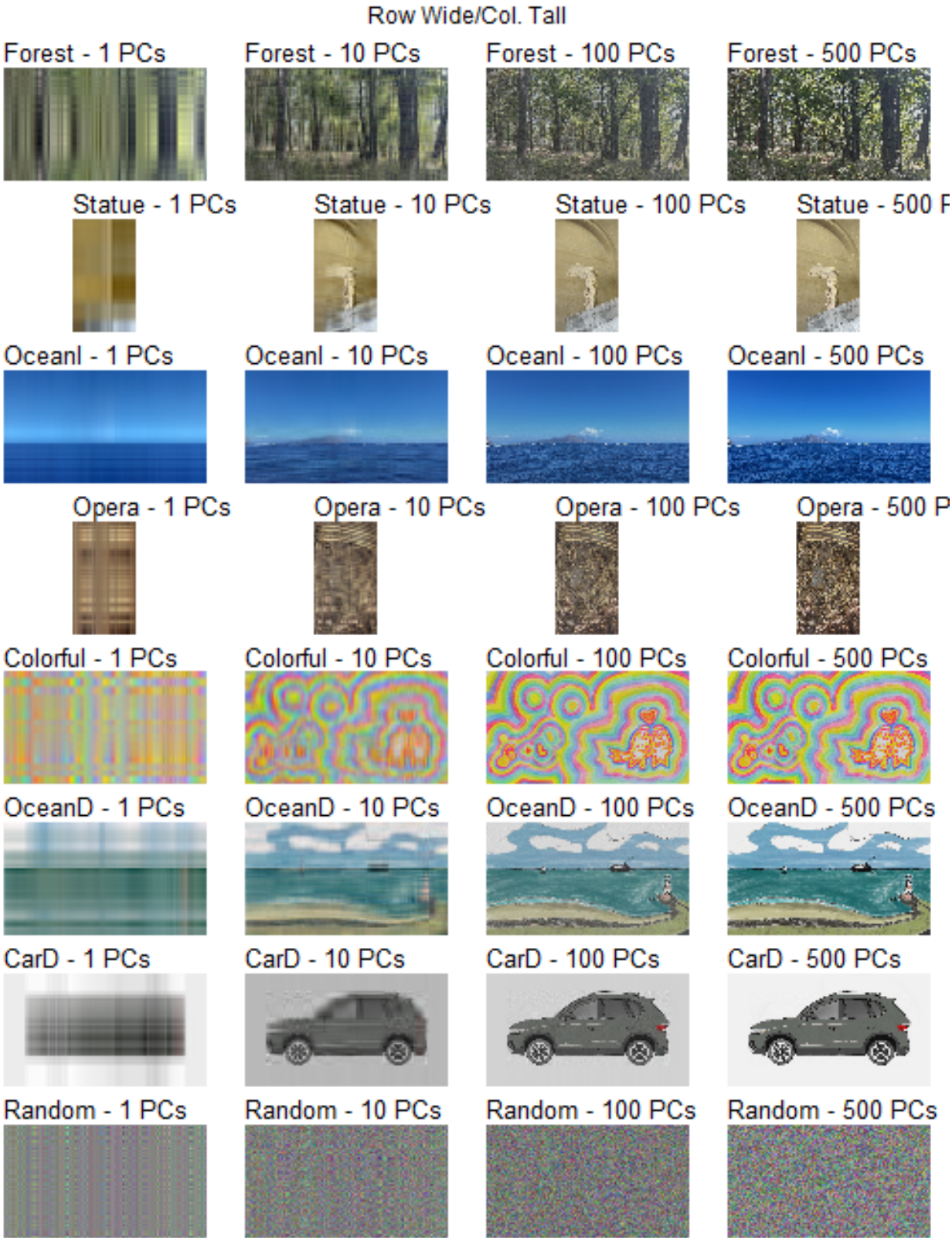

Ocean image
R-W - 1 PCs
R-W - 10 PCs
R-W - 100 PCs
R-W - 500 PCs
R-T - 1 PCs
R-T - 10 PCs
R-T - 100 PCs
R-T - 500 PCs
G-W - 1 PCs
G-W - 10 PCs
G-W - 100 PCs
G-W - 500 PCs
G-T - 1 PCs
G-T - 10 PCs
G-T - 100 PCs
G-T - 500 PCs
Opera Image
R-W - 1 PCs
R-W - 10 PCs
R-W - 100 PCs
R-W - 500 PC
R-T - 1 PCs
R-T - 10 PCs
R-T - 100 PCs
R-T - 500 PCs
G-W - 1 PCs
G-W - 10 PCs
G-W - 100 PCs
G-W - 500 PC
G-T - 1 PCs
G-T - 10 PCs
G-T - 100 PCs
G-T - 500 PCs

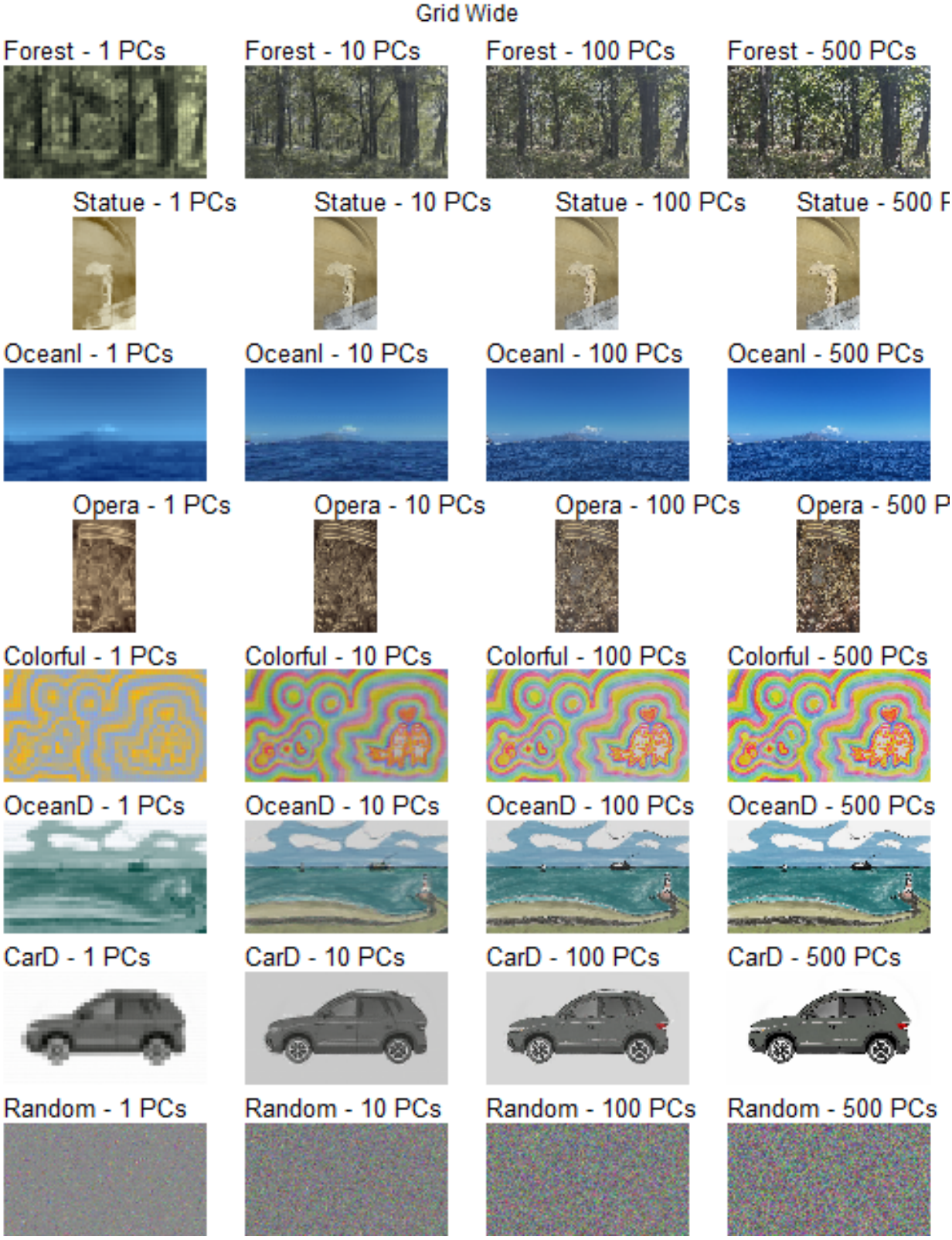

Colorful Drawing
R-W - 1 PCs
R-W - 10 PCs
R-W - 100 PCs
R-W - 500 PCs
R-T - 1 PCs
R-T - 10 PCs
R-T - 100 PCs
R-T - 500 PCs
G-W - 1 PCs
G-W - 10 PCs
G-W - 100 PCs
G-W - 500 PCs
G-T - 1 PCs
G-T - 10 PCs
G-T - 100 PCs
G-T - 500 PCs
Ocean Drawing
R-W - 1 PCs
R-W - 10 PCs
R-W - 100 PCs
R-W - 500 PCs
R-T - 1 PCs
R-T - 10 PCs
R-T - 100 PCs
R-T - 500 PCs
G-W - 1 PCs
G-W - 10 PCs
G-W - 100 PCs
G-W - 500 PCs
G-T - 1 PCs
G-T - 10 PCs
G-T - 100 PCs
G-T - 500 PCs

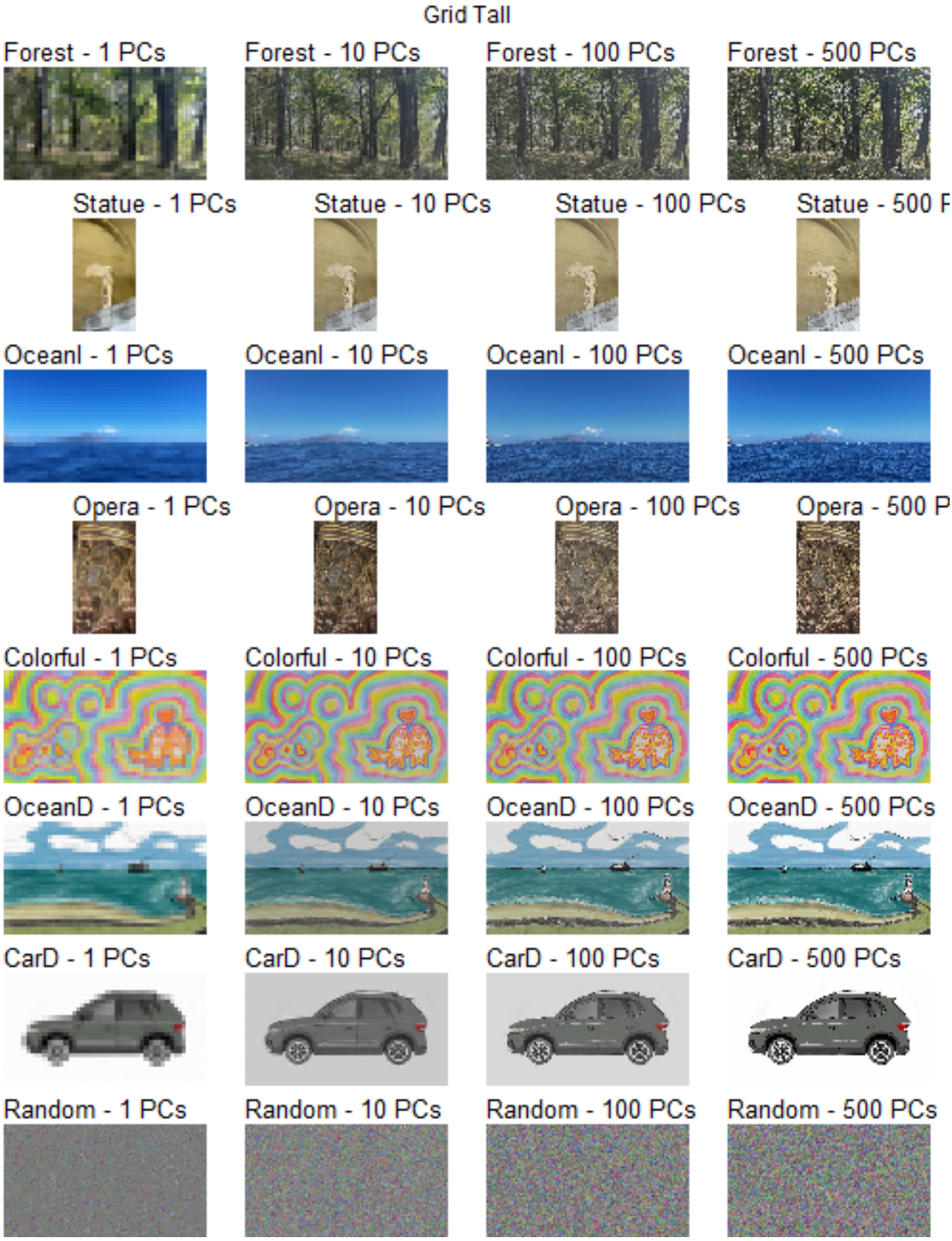


We can see that the grid methods offer the best reconstruction for the image at certain fixed numbers of PCs. At lower counts of PCs, it seems like a pixelation of the image. Comparatively, the row/column methods only offer blurry mess at low numbers of PCs. This lines up the results found by analyzing the variance explained by the Principal Components.

By Method


Row Wide/Col. Tall
Forest - 1 PCs
Forest - 10 PCs
Forest - 100 PCs
Forest - 500 PCs
Statue - 1 PCs
Statue - 10 PCs
Statue - 100 PCs
Statue - 500 F
Oceanl - 1 PCs
Oceanl - 10 PCs
Oceanl - 100 PCs
Oceanl - 500 PCs
Opera - 1 PCs
Opera - 10 PCs
Opera - 100 PCs
Opera - 500 P
Colorful - 1 PCs
Colorful - 10 PCs
Colorful - 100 PCs
Colorful - 500 PCs
OceanD - 1 PCs
OceanD - 10 PCs
OceanD - 100 PCs
OceanD - 500 PCs
CarD - 1 PCs
CarD - 10 PCs
CarD - 100 PCs
CarD - 500 PCs
Random - 1 PCs
Random - 10 PCs
Random - 100 PCs
Random - 500 PCs
Row Tall/Col.Wide
Forest - 1 PCs
Forest - 10 PCs
Forest - 100 PCs
Forest - 500 PCs


Grid Wide
Forest - 1 PCs
Forest - 10 PCs
Forest - 100 PCs
Forest - 500 PCs
Statue - 1 PCs
Statue - 10 PCs
Statue - 100 PCs
Statue - 500 P
Oceanl - 1 PCs
Oceanl - 10 PCs
Oceanl - 100 PCs
Oceanl - 500 PCs
Opera - 1 PCs
Opera - 10 PCs
Opera - 100 PCs
Opera - 500 P
Colorful - 1 PCs
Colorful - 10 PCs
Colorful - 100 PCs
Colorful - 500 PCs
OceanD - 1 PCs
OceanD - 10 PCs
OceanD - 100 PCs
OceanD - 500 PCs
CarD - 1 PCs
CarD - 10 PCs
CarD - 100 PCs
CarD - 500 PCs
Random - 1 PCs
Random - 10 PCs
Random - 100 PCs
Random - 500 PCs



We can see how the simpler images are better represented by smaller numbers of PCs than the more complicated ones. This also lines up with the results from the variance analysis.

In both cases, the image randomly generated by pixels does not behave like the other images, which we also

found in analyzing the variance in the previous section.

Conclusion

Through our analysis of the images reconstructed using various numbers of principal components, we further confirmed the results we found through the variance analysis in PCA.

First, we found that the Grid deconstructions were the best at explaining the highest proportion of variance in the image with the fewest number of PCs.

Next, we found that the visually simpler images can be explained in fewer PCs than more detailed images, as is predictable.

Further Work

There is a lot of opportunity for further work on this topic. The most obvious place for future work would be other ways of converting the image into a 2D matrix, In my analysis, the size of the squares in the grid decomposition was 40x40 pixels, but there may be an optimal square size in this method. This could be examined in depth by comparing different square sizes on the same images in a cross study.

This same type of analysis could be used on different types of media. Before honing on PCA on images, I experimented with PCA of sound files, .wav files in particular. I figured this would be much more difficult to communicate and show in a report so I focused on images.