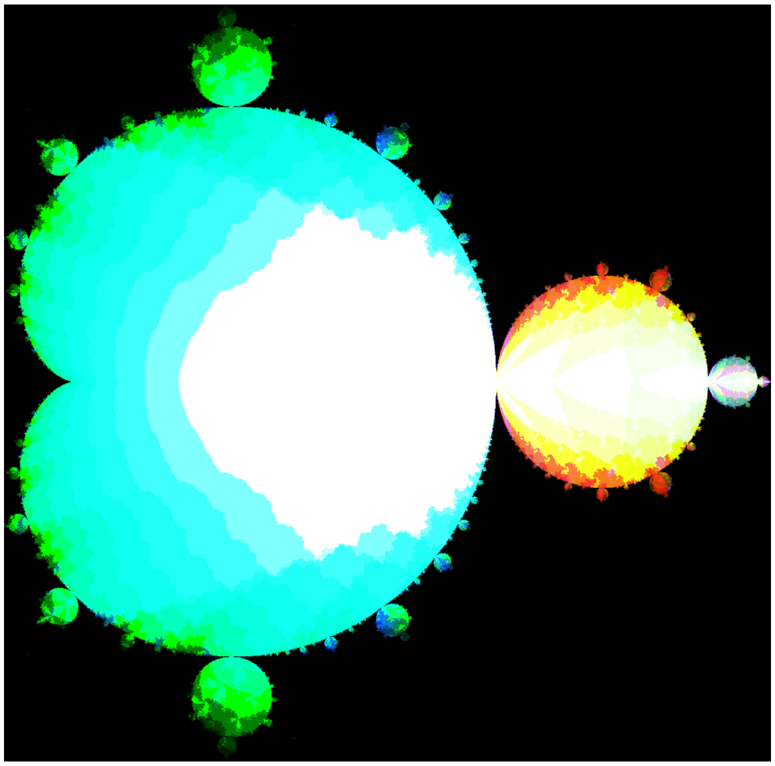
# XYTC: Lossless and Complete Representation of Four-dimensional Data Within Two-dimensional Displays

**Marc Dunand**
Harvard University
Cambridge, MA, USA



## ABSTRACT

Designing effective representations of higher dimensional structures (≥4 spatial dimensions) is an infamous challenge due to the human mind's innate inability to visualize a greater number of spatial dimensions than the three we perceive in the physical world. However, complex simulations and polychannel sensing devices, in conjunction with ever-increasing computing capacity have generated massive higher dimensional datasets [1]. In order to visualize such datasets, scientists must therefore represent additional dimensions in some other form. I present X-dimension, Y-dimension, Time-dimension, Color-dimension (XYTC), a four-dimensional (4D) data visualization which reparameterizes one dimension in terms of time and a second dimension in terms of color. XYTC is capable of representing the full complexity of any four dimensional data structure losslessly and without any manipulation or revectorization of the data. Through a quantitative and qualitative study, I evaluate the effectiveness of XYTC to visualize four-dimensional data as well as the relative success of two different subdesigns for XYTC.

## INTRODUCTION

When people need to visualize data, they may need ways to visualize four dimensions of data, however, existing methodologies for visualizing four-dimensional data often fall victim to a subset of a few obstacles:

**O1:** Existing visualizations lack the ability to represent multiple positions at once on the fourth spatial axis.
**O2:** Existing visualizations cannot represent the full resolution or accuracy of 4D space in a single figure.
**O3:** Existing visualizations can only accurately represent higher dimensional forms if they are regular or geometric.
I will begin by exploring each of these obstacles in turn and describing a characteristic of XYTC which helps to overcome the issue.

**O1: Existing visualizations lack the ability to represent multiple positions at once on the fourth spatial axis.**
Obstacle One refers to a class of visualizations which are capable of representing a single value in higher dimensional space but are unable to represent multiple values at once in that space. In essence, such visualizations are capable of representing a function of the standard three dimensions to some fourth dimension *w* like this: f(x, y, z) = w but they are unable to represent a four-dimensional structure where there is more than one value for w for any set of (x, y, z). A classic example of these limited higher dimensional visualizers are heatmaps. You can represent heat in a volume of space using the color at each voxel in the three-dimensional (3D) area. In this way, heat is almost treated as a fourth dimension. However, heat is not a true dimension. Only one value of heat can be represented at each (x, y, z), creating the limited function-based relationship between heat and the standard three spatial dimensions.

**O2: Existing visualizations cannot accurately represent 4D space in a single figure.**
Obstacle Two focuses on one of the core issues with rendering higher dimensional information in a meaningful way to a human viewer. When directly reducing to a lower number of dimensions, distances between data points cannot be maintained. As a clear example of this phenomenon, imagine how many points in 2-space can be a distance of 1 unit from some central point and a distance of at least 0.1 units from one another. These points form a dotted circle around the central point. Now imagine the same set of constraints for 3-space. This time, there are many more points, which form a sphere around the central point. If you would like to reduce the set of points in 3-space into 2-space, by the Pigeonhole Principle, one of our two invariants (distance to central point or distance between points) must be violated. When you try to reduce the number of dimensions, you literally run out of *space* to accurately represent the distances between all of your points, and you must sacrifice some of the accuracy of your visualization. Creating the least-inaccurate dimensional reduction is an area of active research [1].

**O3: Existing visualizations can only accurately represent higher dimensional forms if they are regular or geometric.**
Obstacle Three focuses on visualizations of four-dimensional structures that may not be entirely visual in nature but may also have a descriptive component. This description could be as simple as "The four-dimensional equivalent of a sphere with radius 1", or equivalently "$x^2+y^2+z^2+w^2 \leq 1$", or equivalently "A hypersphere of radius 1". A slightly more complex example would be "The four dimensional shape created by linearly interpolating between 3D shape A and 3D shape B through a distance of 1 in the fourth dimension", where A and B are any 3D shapes. Such descriptions produce well-defined four-dimensional structures, however, the set of 4D structures such mixed approaches can represent is a vanishingly small subset of all 4D structures.

XYTC was created with this set of obstacles in mind and was therefore designed with the following characteristics:

**C1:** XYTC can represent up to 24 values at once in the fourth dimension
**C2:** XYTC can represent four-dimensional structures accurately
**C3**: XYTC is capable of representing any 4D shape

**C1: XYTC can represent up to 24 positions at once in the fourth spatial dimension**
XYTC uses color to represent a dimension of information. As color is represented by a 24-bit value on a computer display, this allows me to represent the presence or absence of up to 24 points in the color dimension. As the three other dimensions of XYTC are the length, width, and time, these dimensions have no upper limit beyond file size and screen resolution. Reparameterization into color and time will be explored in much greater detail later in this paper.

**C2: XYTC can represent four-dimensional structures accurately**
As XYTC does not actually reduce the number of dimensions, but rather reparameterizes two of its four dimensions, the inaccuracies created by dimensional reduction do not apply. This means that XYTC dodges the issue of inaccurate representation of higher dimensional space altogether and can accurately represent four-dimensional structures.

**C3: XYTC is capable of representing any 4D shape**
By the same logic as C2 above, the fact that XYTC simply reparameterizes dimensions rather than compresses them guarantees that it can represent any 4D shape.

XYTC also has a number of limitations baked into it. Most importantly, the visualization is *only* capable of representing structures of four dimensions or less. The reparameterization approach is not generalizable, and it has no way of dealing with a 5th or n+1st dimension. Second, the resolution of whichever spatial dimension gets reparameterized in time is highly limited. Being able to represent 24 values at once is certainly much better than being able to represent only a single value (as discussed in Obstacle 1), but it is still a far cry from modern screen resolutions of 1080p or greater. The XYTC interface partially mitigates this limitation with two techniques. Firstly, it affords easy zooming and panning over 4D structures, allowing you to zoom in and inspect a section of the structure at higher fidelity when 24 pixels is insufficient. This works because the resolution of 24 applies to whatever is in frame. Zooming the frame in allows for the resolution of 24 to apply to a small slice of the structure, rather than the whole structure, effectively increasing the level of detail on that section inversely proportional to how small the section being zoomed into is. A second mitigation of the low resolution of the color dimension is that XYTC allows you to encode any dimension of your data into any dimension of its visualization. This means that users can strategically assign the color dimension to a dimension in their 4D data that is less important or contains less information.

## RELATED WORK

The research space of higher dimensional visualizations is quite large, with an excellent introduction being the survey piece *Visualizing High-Dimensional Data: Advances in the Past Decade*, Liu et al. [1]. There is additionally a wealth of data visualization research focused on the application of color or time to representing information. For example, *Polychromatic plots: Graphical display of multidimensional data*, Roesderer et al. [2] introduces PolyChromatic Plots which are data visualizations which use each color channel (red, green, and blue) to represent a distinct value. Another example of the well-established link between higher dimensional data representation and color is *U-CIE [/ju: 'si:/]: Color encoding of high-dimensional data*, Koutrouli et al. [3] which uses dimensional reduction techniques to encode arbitrarily higher-dimensional data into color.

There are also a number of papers which explore the applications of time to visualizing higher dimensions. Notably, *Time-varying volume visualization: a survey*, Bai et al. [4] provides a section discussing how time has been used to visualize four-dimensional information. This survey piece describes a set of four papers which treat time as another dimension identical to spatial dimensions (Bremer et al. [5]; Edelsbrunner et al. [6]; Lukasczyk et al. [7]; Weber et al. [8]) through the use of 4D Reeb graphs used to analyze a variety of datasets while treating the spatial and temporal dimensions identically. The technique of directly parameterizing a spatial dimension in time is also well established, with discussions of the technique present at least as far back as 2000 in *Isosurfacing In Higher Dimensions*, Bhaniramka et al. [10]. This is the very same technique which is used in XYTC to reparameterize a spatial dimension in terms of time.

All of this previous work naturally begs the question of what exactly does XYTC do that is novel. Expressions of higher dimensional information in terms of color and time have been well understood for decades. Indeed, my approach of reparameterizing one spatial dimension in terms of time through a sequence of 3D slices is not novel, though I will still discuss the approach in greater detail in the following section. It is instead my method for reparameterizing a spatial dimension in terms of *color* that is novel. Specifically, the fact that I encode dimensional data into color in such a way that makes the color dimension interchangeable with either of the spatial dimensions or the time dimension. This affords free rotation, scaling, and panning through the entire 4D space as mathematically, all four dimensions behave equivalently and interchangeably in the XYTC visualization.

## XYTC: THE SYSTEM

In this description of XYTC, I will begin by explaining the theoretical backing for my design approach, then proceed through descriptions of how the visualizer works at a low level of abstraction before closing with a list of features that XYTC provides to the user.

### Design Claim

The design of XYTC is supported by Structural Mapping Theory (SMT). SMT, first introduced primarily by Dedre Gentner in 1983, explains that when humans make analogies between things, we do so by matching their deeper relationships rather than their superficial traits [11]. Ultimately, XYTC is just that: a mapping. XYTC maps four spatial dimensions of data onto two-dimensional (2D) screens. This means that in order to make this mapping effective, it is more important to maintain the deeper relational structure of the four dimensional data rather than the superficial appearance of the visualizer. This theory informs the core design tenant of XYTC: **the visualization should always structurally behave exactly as the four dimensional structure does**. With this core tenant in mind, I have laid out my design claim in terms of *what* the design is, *when* it applies, and *how* it relates to SMT:

*A four-dimensional data visualization that maintains the structural relationships between dimensions helps users conceptualize the four dimensional data* ***(what)*** *when representing spatial dimensions in color and time is effective* ***(when)****. This is supported by Structural Mapping Theory (SMT) as the data visualization maintains the structural relationships between dimensions that are present in four-dimensional structures, making it easier for users to map the data visualization into its original four-dimensional form* ***(how)****.*

At the lowest level of abstraction, I will begin by explaining how the four-dimensional information is encoded into the visualizer. As a concrete example, we provide the Julia-Mandelbrot set as a demonstratively complex 4D structure for use in the XYTC visualization.

**What is the Julia-Mandelbrot set?**

The Julia-Mandelbrot set is defined in this paper as the four-dimensional hyperfractal generalization of both the 2D Julia set [16] and the 2D Mandelbrot set [17]. The Julia-Mandelbrot set is defined as all complex points ($Z_0$, C) such that the inductive sequence $z_{n+1}= (z_n)^2 + c$ converges as n approaches infinity [12]. This is the direct generalization of the Julia set and Mandelbrot set which are defined, respectively, as:

**Julia:** all complex values of $z_0$ for which $z_{n+1}= z^2_n+c$ converges for some constant c

**Mandelbrot:** all complex values of c for which $z_{n+1}= z^2_n+c$ converges for some constant $z_o$

As you can see, both sets hold one of the two variables constant, so by varying both at once, as you do in the defined Julia-Mandelbrot set, you are varying across two complex variables. If you parameterize one dimension as each of the real and imaginary components of both $z_0$ and c, you end up with our four dimensions. Then, each point in the four dimensional space is considered in the set if and only if its coordinates refer to values of $z_0$ and c that make the inductive sequence $z_{n+1}= z^2_n+c$ converge.

The Julia-Mandelbrot set is an ideal challenge for a four-dimensional visualization due to the infinite complexity and high irregularity of the fractal form.

**Reparameterization**

As previously described, XYTC functions as a 4D visualizer by reparameterizing two of its four dimensions from spatial to temporal and color and then mapping the remaining two spatial dimensions in terms of the x and y axis (width and height) of the computer display. The following sections will dive into detail on how reparameterizing a spatial dimension in terms of time and color can be achieved.

**Time**

The reparameterization of the spatial dimension in terms of time is by far the simpler of the two reparameterizations as time is already innately a dimension of our physical world and therefore already has the requisite characteristics. In fact, the methodology used to reparameterize a spatial dimension in terms of time is already well established [10]. For all descriptions of dimensional reparameterizations, I will describe the reparameterization in terms of a lower dimension count in order to make the transformation perceivable to the human mind. This is a reasonable abstraction as mathematical logic works independently of the dimensional count, meaning that these transformations are structurally identically at four dimensions.

A simple 3D -> 2D + time reparameterization which mirrors a 4D -> 3D + time reparameterization is slicing a sphere. Take a regular 3D sphere and then slice it along any plane into a series of thin pieces. Each of these slices now resembled a circle of some variable radius. Now, treat each of these circles as a frame of an animation and play these frames in order. The result will look like a circle that expands at first rapidly, before slowing and then reversing and shrinking at an equal accelerating rate. This animation exists in 2D but it holds all of the information of a 3D sphere (at some resolution). This means that we can reconstruct the 3D sphere by placing each 2D frame of our animation on top of the previous one.

Equivalently, you can 'slice' a 4-dimensional shape into 3D slices and then create an animation where each frame is a 3D slice, allowing you to represent a four dimensional shape in 3D space.

**Color**

Our reparameterization in terms of time almost seems like it should be sufficient on its own. Why do we need to reparameterize an additional dimension in terms of color when we already have been able to visualize our four dimensional structure in 3D space with our simple time reparameterization? With modern computers, it is a relatively simple thing to create a digital three-dimensional animation. You could zoom, pan, and rotate around this animation, allowing you to see it from all sides even if the screen itself is only two-dimensional. So given all of this, why even bother reparameterizing another dimension? The answer to this question is embedded in the very reasoning of the previous sentence. You can see this 3D animation from all *sides*. Even if you somehow create an object in our 3D world which behaves exactly like our 3D animation, all you can ever do is see its *sides*. Any complexity to the internal structure is invisible, occluded from all angles, and even if we make our 3D animation transparent, we simply cannot fully represent 3D spatial data to the human eye. This is due to a fundamental limitation of physics and biology. No matter how eloquently our brain interprets the light that strikes our retinae as 3D, the retina itself can only take in light across a two dimensional surface. In fact, the very concept of a three dimensional surface is meaningless in a three-dimensional world, as a surface *must* be of at least one dimension less than the space it exists within.

So we have run into just about as fundamental of a limitation as there is. If we want our eyes to take in truly three spatial dimensional information, the universe itself would have to contain four spatial dimensions that we can interact with. Note, however, the specification of *spatial* dimensions. There is one other dimension of information that our retina takes in about light other than the 2D position at which it strikes our retina. That is the light's energy, its *color*. This means that our eyes fundamentally take in three dimensions of information: x-position and y-position in the plane of our retina and color.

Technically, a fourth dimension perceptible to our eyes could be defined as the intensity of the light, how *much* of a certain color of light is hitting a specific point in our retina. Even more technically, this dimension could then also be subdivided into both saturation and value, giving you all three dimensions of the HSV color space (hue, saturation, value). However, this is not practical for a few reasons, most notably that human eyes are limited in their precision for detecting the difference between similar hues. This means that if you subdivide color into a couple of separate dimensions, each of these dimensions end up being so tiny as to be ineffective for encoding a dimension of information. Earlier in the paper, it was mentioned that the size of the color dimension ends up being only 24 units, which is already a serious limitation of the XYTC system.
The same issue appears if you approach the dimensionality of our eyes from the perspective of the color sensors in our retinae. The three cone types in our eyes; large, medium, and small, would appear to imply that color is in fact interpreted biologically as three dimensions, which is technically correct, but once again, each of these dimensions end up being so small as to be practically useless for encoding dimensional information. As a result of this, subdividing the color dimension as a form of visualization is not explored in this paper.

At this point, we have identified the three fundamental dimensions of information that our eyes take in: two spatial dimensions and one color dimension. But how exactly is color represented as a dimension? To understand this, I provide a visualization of how two spatial dimensions can be reparameterzied in terms of one spatial dimension and one color dimension in figure 1, and how three spatial dimensions can be reparameterized in terms of two spatial dimensions and one color dimension in figure 2.

**Figure 1**

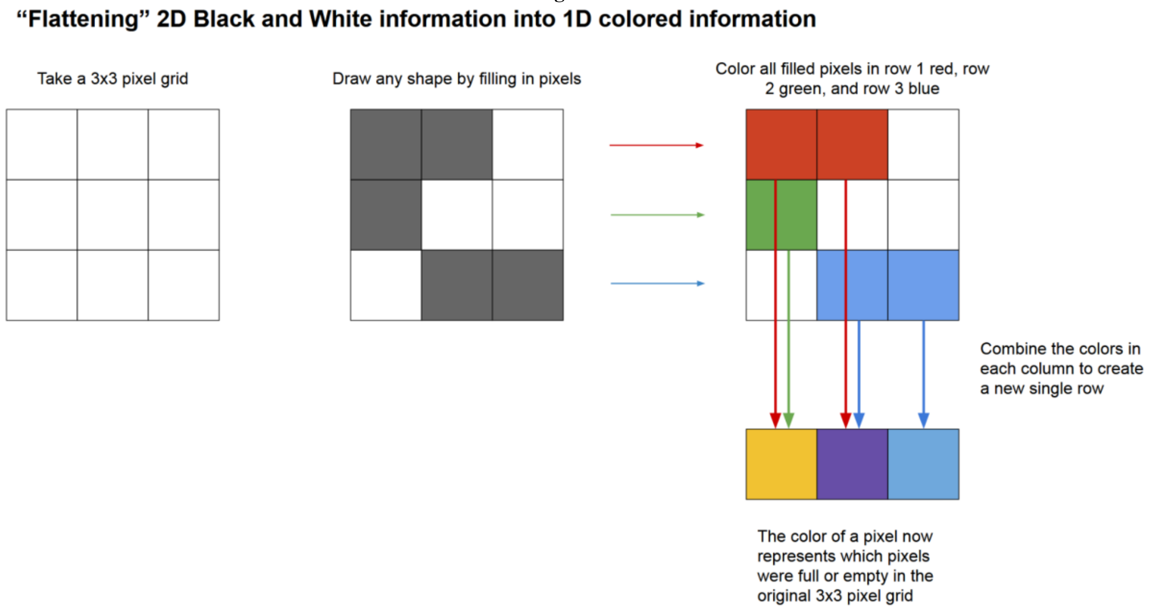

"Flattening" 2D Black and White information into 1D colored information
Take a 3x3 pixel grid
Draw any shape by filling in pixels
Color all filled pixels in row 1 red, row 2 green, and row 3 blue
Combine the colors in each column to create a new single row
The color of a pixel now represents which pixels were full or empty in the original 3x3 pixel grid

Figure 1 begins with a 2D grid which gets filled with some binary (black and white) information. Every black pixel is then assigned a color depending upon its row. Finally, the colors present in each column are additively combined to create a new colored pixel which represents the binary information of the entire column. A single row of these new colored pixels can therefore represent two dimensions of information. With careful choice to the color assignments of each row, you can guarantee that every unique column structure generates a unique color once flattened. This guarantees that if you know the color assignment, you can reconstruct the 2D information from the final 1D+color information.

**Figure 2**

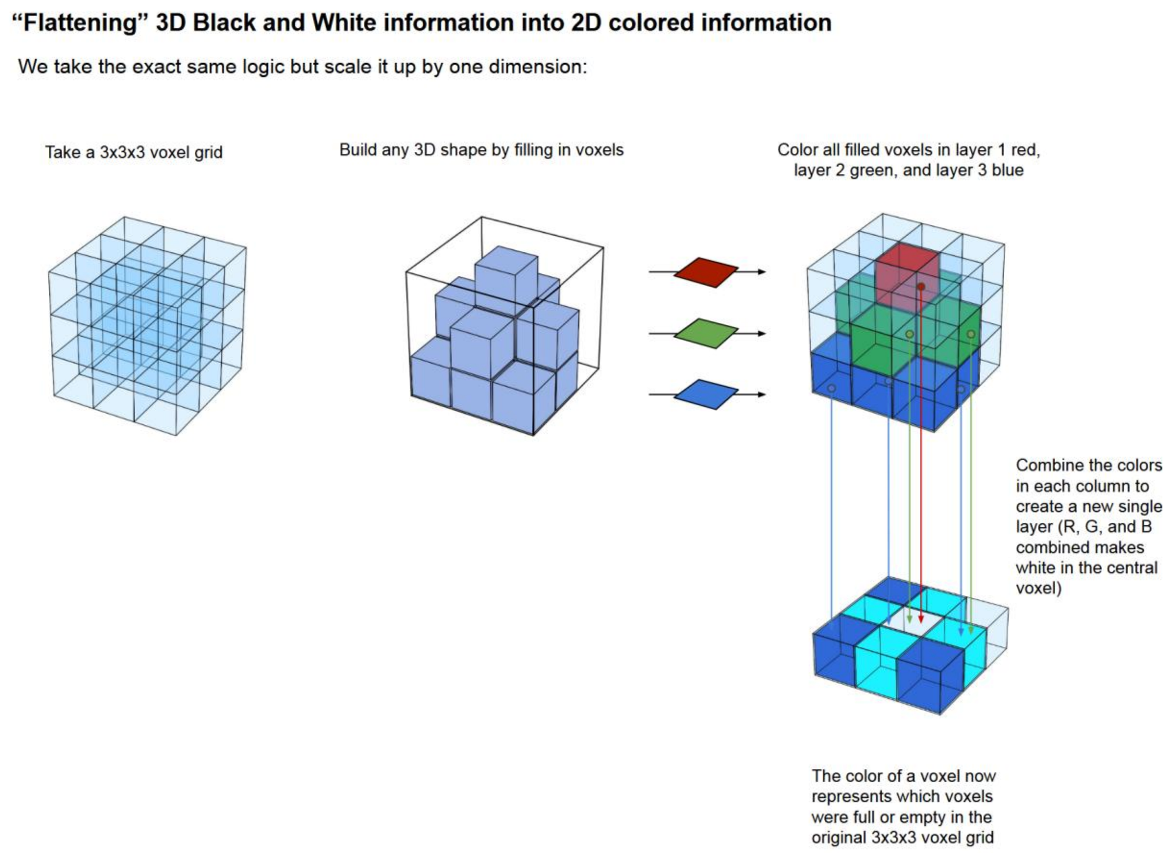

"Flattening" 3D Black and White information into 2D colored information
We take the exact same logic but scale it up by one dimension:
Take a 3x3x3 voxel grid
Build any 3D shape by filling in voxels
Color all filled voxels in layer 1 red, layer 2 green, and layer 3 blue
Combine the colors in each column to create a new single layer (R, G, and B combined makes white in the central voxel)
The color of a voxel now represents which voxels were full or empty in the original 3x3x3 voxel grid

Figure 2 begins with a 3D cube which gets filled with some binary (black and white) information. Every black pixel is then assigned a color depending upon its layer. The flattening process then proceeds exactly as it did in figure 2, resulting in one fewer spatial dimensions plus a color dimension, and maintaining the same guarantee of being able to reconstruct the three dimensional information given the final 2D + color information.

Figures 1 and 2 demonstrate a technique which I refer to as "flattening", effectively merging component colors (in this case red, green, and blue) across a spatial dimension to create a unique product color representative of that spatial dimension. As exhibited by the structural similarity of the approach in figure 1 and 2 despite their difference of a dimension, the flattening operation can be executed regardless of the number of dimensions at play. This means that 4D information can be reparameterized as 3D+color information (generating a volume of colored voxels) using this flattening operation.

There is still one critical difference between the flattening operation as it is described in figures 1 and 2 versus how it is applied in XYTC. That is a difference of size. The color dimension in XYTC is of size 24 units, however in the two figures, the color dimension is only of size 3 units. To get up to a size of 24 units we need to look at how computers construct the color of a pixel. Although certain specialty displays have different specifications, generally computers display a color as a 24-bit value, separated into 8 bits to control the intensity of the red channel, 8 bits for the green channel, and 8 bits for the blue channel. Every single one of the $2^{24} = 16{,}777{,}216$ values that a 24-bit binary number can represent corresponds to a unique color that a computer display can produce. This may look like a lot, but the number of values needed to encode a dimension scales exponentially relative to the size of that dimension. In figure 1, we have a color dimension size of 3. To represent whether each pixel contains or does not contain a point, we need one bit per pixel, so to represent a dimension of size 3, we need 3 bits, capable of representing $2^3 = 8$ color values, each one corresponding to one unique combination of empty and full pixels. This means that to represent a color dimension of size 24, we need 24 bits, or $2^{24} = 16{,}777{,}216$ values, which is the full size of the encoding of a color value for a standard display.

This is the methodology for reparameterizing a spatial dimension in terms of color at a resolution of 24 units. Continuing with our example of the Julia-Mandelbrot set, begin by taking each one-dimensional (1D) "column" along the dimension that you want to reparameterize[1]. For each position in that column, check if its four-dimensional coordinate is in the Julia-Mandelbrot set. If so, set the corresponding bit in a sequence of 24 bits to 1, if not, set that bit to 0. So, if the 1st, 3rd, 9th, and 24th positions in some 1D column are in the Julia-Mandelbrot set, that column would correspond to the binary value 10100000 10000000 00000001 which corresponds to the hexadecimal value 0xA08001, which corresponds to rich, dark yellow color. A pixel of that color then uniquely corresponds to the information that 1st, 3rd, 9th, and 24th positions in the color dimension are in the Julia-Mandelbrot set.

One can also note that by reparameterizing a third dimension in terms of color rather than space, you can see any internal details to the 3D structures. If you have a structure that is in some places hollow with thin walls of width two units, in the color dimension these places would be represented by the value 11000000 00000000 00000011, a bright red. However, when looking at this object in three spatial dimensions, you would be unable to tell that it is hollow as both ends of the structure are opaque. This could lead you to assuming that the structure is solid, or that it has some other internal structure. However, with that third dimension parameterized as color, if the structure were solid, its value would be 11111111 11111111 11111111, represented as pure white. If we see that the pixel is instead red, we know for certain that the internal structure is indeed mostly hollow.

By combining my two reparameterization techniques, slicing through time and encoding in color, I am able to decrease the number of dimensions visualized in space from four down to two. These two remaining dimensions are visualized directly on the x and y axes of the computer display, while the color of each pixel in the display parameterizes a third dimension and the animation of that 2D+color visual over time parameterizes a fourth dimension. This allows us not only to visualize one more dimension than what is directly possible, but it also gives us full information about the 4D structure, rather than simply an understanding of its surface.

## Interface

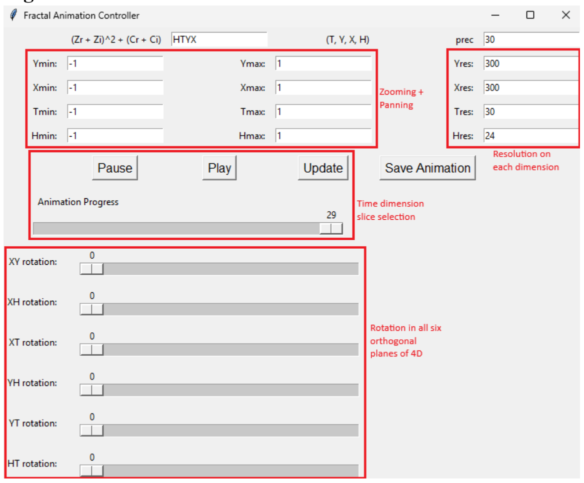
Where the previous section focused on explaining at a low level of abstraction how XYTC represents four dimensions, this section focuses on how users interact with the interface built atop the reparameterization.

Beginning again with the core tenant of XYTC, as informed by SMT, I focus on making sure that the relationship between the four dimensions remain that same for XYTC as they would in true four spatial dimensional space. By doing so, even if superficially XYTC does not appear like four dimensional space, the deeper relational connections between dimensions remain the same. SMT tells us that users will have a better chance of mapping XYTC to the four-dimensional data thanks to the underlying relational structure of both being the same. The previous section showed that rather than compressing and warping a higher number of dimensions into lower dimensionality, I still maintain all four dimensions, but simply show them in a different form. This means that all four spatial dimensions still behave mathematically like spatial dimensions in XYTC, they simply display differently. This allows us to treat XYTC like a four spatial dimension environment, because mathematically and relationally, it is.

## 4D Transformer

The core feature of the XYTC interface is the four-dimensional transformer. This interface (figure 3) allows the user to explore their four-dimensional data through freely panning and zooming each dimension independently, rotating the 4D object in any of the six orthogonal plans, and dynamically increasing or decreasing the resolution of any dimension.

**Figure 3**


Fractal Animation Controller
Zooming + Panning
Resolution on each dimension
Pause
Play
Update
Save Animation
Time dimension slice selection
Rotation in all six orthogonal planes of 4D

(Zr + Zi)^2 + (Cr + Ci) HTYX (T, Y, X, H) prec 30

Ymin: -1 Ymax: 1 Yres: 300
Xmin: -1 Xmax: 1 Xres: 300
Tmin: -1 Tmax: 1 Tres: 30
Hmin: -1 Hmax: 1 Hres: 24

Animation Progress 29

XY rotation: 0
XH rotation: 0
XT rotation: 0
YH rotation: 0
YT rotation: 0
HT rotation: 0

Figure 3 shows the XYTC 4D navigator which can be used to freely navigate in four dimensions to explore your data. The Y dimension is the height of your screen, the X dimension is the width of your screen, the T dimension is time, and the H dimension is color (H stands for hexadecimal value).

In figure 3, you may notice that each dimension has the exact same relationship with the navigator. Regardless of whether the dimension is represented directly, in color, or in time, you are able to rotate, scale, pan, and change the resolution of that dimension. This is possible because behind the scenes, the data is still four spatial dimensions, which simply gets reinterpreted visually in terms of color and time. This relationally equivalent treatment of all dimensions is

[1] Here, a 1D column is defined as a one-dimensional "slice" of a shape of two or more dimensions. In terms of a 3D object, a 1D column can be thought of as a "core sample", where you extract all z values at some set (x, y) position.

designed in order to make users' mapping of XYTC to 4D space easier, per SMT.

The 4D navigator does include one asymmetry between the controls for manipulating the dimensions. Time has one additional section, labeled in figure 3 as 'Time dimension slice selection'. This section is unfortunately necessary for selecting which 3D slice orthogonal to the time dimension to display. Buttons to play through the slices in the time dimension are also included in order for the user to see the entire 4D structure at once without having to focus on scrubbing across the time slices. Nonetheless, time can still be manipulated in all of the same ways as the other dimensions. Panning in the time dimension moves 3D time-orthogonal slices to the "left" or "right". Zooming in the time dimensions increases or decreases the distance between the 3D slice at t=0 and t=max. Rotating in a plane that includes the time dimension rotates the vector along which the 4D structure is orthogonally sliced. Increasing or decreasing the resolution in the time dimension increases or decreases the number of 3D slices in the total animation.

### Color Dimension Inspector

**Figure 4**

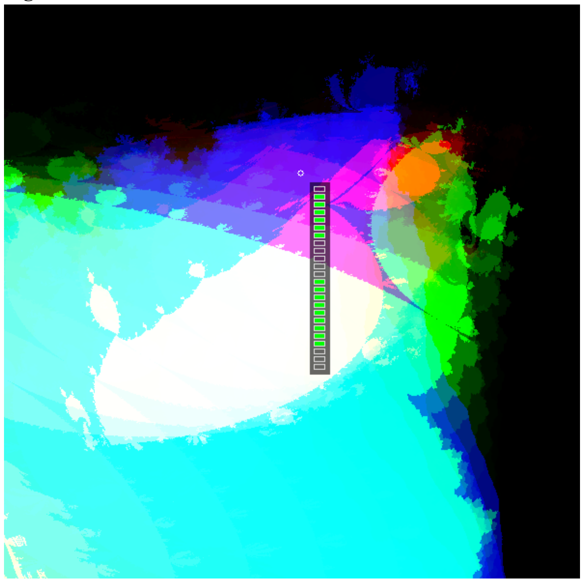


Figure 4 shows the XYTC visualizer with the color dimension inspector active. The small white circle represents the pixel that the mouse is located on and the green and empty bars represent the color dimension column of that pixel. In this case, we see that the color dimension is hollow in the middle at this mouse location, a feature that often corresponds to purple hues.

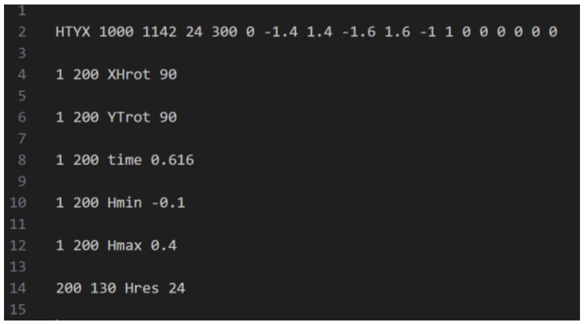
Much like the symmetry present in the dimensional transformations of the previous section, the color dimension inspector is also designed to reinforce the structural equivalence of XYTC to a four-dimensional space. Users can choose to toggle on the optional inspector with a keyboard button, at which point a spatial visualization of the color dimension follows the user's mouse around the screen. This visually reminds the user that the color dimension is still fundamentally a spatial dimension, simply communicated in a different manner, reinforcing the structural and relational mapping between XYTC and 4D space.

This visualization shows what the spatial equivalent of the color dimension is at the pixel the user is currently moused over. Green bars represent points in the Julia-Mandelbrot set (we are still using that example) while empty bars represent points outside of the set. Effectively, by mousing over a point with the inspector on, you are seeing a 1D column slice of our 4D object. Any still frame of XYTC is already a 3D slice of the object that is orthogonal to the time dimension, and then the user slices against a constant x and y coordinate by moving the mouse to a specific location on the screen. The inspector does not provide any information that is not already encoded into the color dimension of the visualization. It is instead designed to help users form an internal mapping between different colors and their spatial equivalents by allowing the user to see what a certain type of color maps to spatially.

### Transformational Language and Compiler

The Transformational Language and Compiler (TLC) is a simple instruction set built specifically for automated transformations in the full 4D space of XYTC. Due to the $O(n^4)$ size of 4D datasets on dimensions of size n, computers can often begin to seriously slow down when trying to render transformations from live input from the 4D Transformer on large datasets. CUDA GPU acceleration is used to provide what speedup it can, however, with high resolution datasets this is often not sufficient. The TLC can be used in such cases to predescribe the transformations you would like to do in 4-space around your dataset, at which point the computer will automatically compute these transformations in the background, allowing the user to do other things rather than waiting for frames to load. Once the transformations have been computed, a video representing the 4-space transformations is saved to the user's hard drive for viewing.

**Figure 5**



```
HTYX 1000 1142 24 300 0 -1.4 1.4 -1.6 1.6 -1 1 0 0 0 0 0 0

1 200 XHrot 90

1 200 YTrot 90

1 200 time 0.616

1 200 Hmin -0.1

1 200 Hmax 0.4

200 130 Hres 24
```

Figure 5 shows typical commands in the TLC which describe the initial settings of the four dimensions in XYTC and then a set of transformations to do to the object from there.

**Color Scheme**

Up until this point, the color scheme I have been using has been defined arbitrarily by the computer's hexadecimal representation of colors. This representation does make internal sense in the context of a color display. The first two hex values (eight binary values) represent how much red is in the color, the middle two hex values represent how much green, and the final two represent how much blue. This tracks neatly to the hardware of computer displays, where generally each pixel contains a red, green, and blue subpixel, each of which can be turned on to varying brightnesses. This color scheme, however, has nothing to do with dimensional representation. In some ways this RGB color scheme does hold up. Generally, colors that are similar represent spatial structures that are similar, and generally the brighter a pixel, the larger its corresponding structure.

The value of a bright color corresponding to large structure is supported by research on the intuitive human mapping between brightness and size, much of which is aggregated in the survey paper *The Psychological Meaning of Color in Design a Semantic Review* by Gor Meliksetyan [13]. This means that such a correlation being present in the RGB color scheme may improve users' intuitive mapping of the color dimension to a spatial dimension.

The RGB color scheme is, however, seriously limited. Most importantly, the hamming distance between the binary representations of two structures in the color dimension (the 24-bit values) does not correspond to the perceptual difference between their two corresponding structures. Hamming distance refers to the minimum number of bit flips needed to turn one binary value into another binary value. In the context of those binary values corresponding to 1D columns in the color dimension, the hamming distance between those values corresponds to how similar or different those two columns are. In all, this means that the degree of similarity or difference of two structures does not correspond to the degree of similarity or difference between the colors representing the two structures. This leads to serious issues with the RGB visualization. Two areas which are very similar in the color dimension may appear as highly distinct colors, giving the false impression that the colors actually correspond to very different structures. The inverse can also happen where highly distinct structures result in perceptually similar colors. Figures 6 and 7 show examples of this.

**Figure 6**

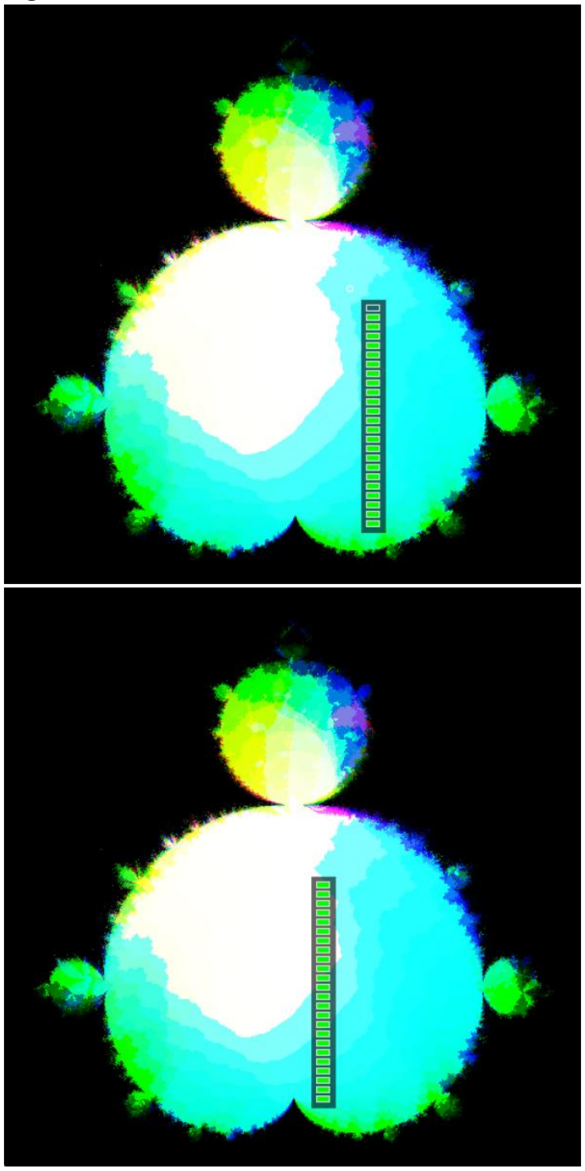


Figure 6 shows the color dimension inspector hovering over two different spots in the same frame of XYTC. In the top image, the mouse is hovering over a turquoise colored area while in the bottom image, the mouse is hovering over a pure white area. Despite the distinct difference of color, the color dimension inspector reveals that the color dimension structure at these two locations differ by only a single unit (have a hamming distance of 1).

**Figure 7**

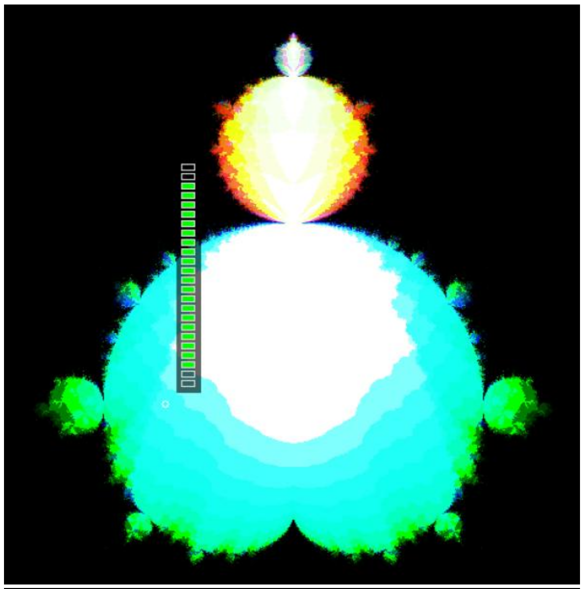
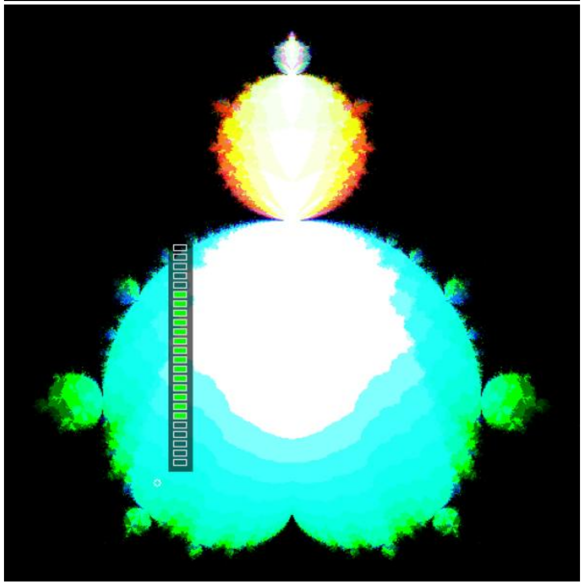



Figure 7 shows the color dimension inspector hovering over two different spots in the same frame of XYTC. In the top image, the mouse is hovering over a turquoise colored area while in the bottom image, the mouse is hovering over another area of almost exactly the same shade of turquoise. Despite the extreme similarity of the colors, the color dimension inspector reveals that the color dimension structure at these two locations differ by six units (have a hamming distance of 6).

In order to address the potentially unintuitive nature of this mismatch between perception and underlying structure, I provide an alternate bitcounting-based color scheme.

## The 25-Layer Color Scheme

Ultimately the RGB color scheme is simply a direct bijective mapping between 16,777,216 possible 1D structures and 16,777,216 possible colors. The RGB color scheme is naive in the sense that it maps each structure to a binary number corresponding to what points are present or absent in the structure and then interprets that number directly as a hexadecimal color code. However, just so long as we maintain a bijective relationship between the set of 1D structures and the set of colors, this bijection can be any relationship that we like. This gives us 16,777,216! (16,777,216 factorial) possible bijective relationships, which is quite a few.

Theoretically, the best color scheme would be one where hamming distance is proportional to difference in perceived color. This would mean that the visual difference between two colors corresponds precisely to the structural difference of their corresponding 1D columns. However, achieving this is not possible due to both the construction of the human eye and of standard computer displays.

We can use the Pigeonhole Principle to show that a perceptually uniform color scheme across hamming distance is impossible. Take any 24-digit binary value $b_0$. By definition, this value must correspond to some 1D structure. This structure must also have a set of exactly 24 other binary values $S_1 = [b_1, b_2, \dots b_{24}]$ that have a hamming distance of 1 from itself. $S_1$ can be generated by flipping bit i of $b_0$ for i in [1, 24] to generate $b_i$. In essence, flip exactly one bit of $b_0$, and since $b_0$ has 24 bits, this can be done in 24 unique ways. Now, each $b_i$ in $S_1$ has a hamming distance of exactly 2 from all other $b_i$ in $S_1$. This is the point at which we run into a problem. To maintain perceptual uniformity across hamming distance, each $b_i$ in $S_1$ needs to be equally far apart from one another in color space (since they all have a hamming distance of 2 from each other). However, since $S_1$ contains 24 different structures, this means that you would need 24 colors that are all equally different from each other. To understand why this is impossible, we need to look again at how color is constructed by standard computer displays.

Each colored pixel of a computer display has a red, green, and blue channel, the brightness of each defined by a two-digit hexadecimal value, allowing values of 0-255 on each channel. This means that the color space that displays can produce (as seen in figure 8) is a 3D cube of side lengths 256 with each of the three dimensions corresponding to how much red, green, and blue is present in the color. This cube, which has the expected volume of $256^3$ = 16,777,216 units, contains all possible colors a display can produce and organizes them spatially by how perceptually similar they are to one another[2]. This means that the color space of computer displays is three dimensional. However, in this 3D color space, we need to

[2] Note that this similarity is defined by the brightness of the red, green, and blue subpixels, not by the human eye's perception of the resulting color. The CEILAB color space and other color spaces designed for color perception uniformity to the human eye are better analogues in our case of creating a human-interpretable visualization, and this is an area that warrants more research. Nonetheless, the RGB color space is certainly a good enough representation of perceptual uniformity for our use here.

have 24 equidistant values. This is not possible. It is a standard geometrical result that for an n-dimensional space the maximum number of mutually equidistant points is n+1, so in 3-space, we can have a maximum of 4 equidistant values, and therefore 4 values of equal perceptual distance from one another. This whole issue boils down to the hamming distance of a 24-digit binary value existing in 24-space while color exists in 3-space, so it is impossible to create a mapping from 24-space to 3-space which does not warp the distances between points (and ultimately the difference between colors) to such an extreme degree that it renders the mapping mostly useless.

**Figure 8**

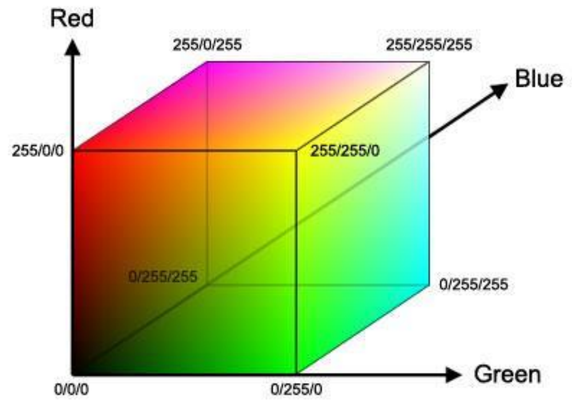


Figure 8 shows the RGB color space used by standard computer monitors. This figure is from *Face Detection and Recognition in Color Images under Matlab*, Maia et al. [14].

Instead, the 25-Layer color scheme (25L) is designed to be maximally perceptually uniform over bitcount. Bitcount refers to the sum of the individual bits in a binary number, so the bitcount of 0101 is 2, the bitcount of 0000 is 0, and the bitcount of 1111 is 4. If you group all possible 1D structures by their bitcounts, this gives us 25 groups, ranging from the group with a bitcount of 0 all the way up to the group with a bitcount of 24. Since each 1 in the binary value represents a point in the color dimension, and each 0 represents the absence of a point, a bitcount group n contains all 1D structures of a size n. As mentioned before, there is an intuitive human correlation between brightness and size [13], and the 25 groups are ordered by size. To then map these size groups to color brightness I order them in the following manner. First, take again the color cube in figure 7 and imagine it being placed on its tip so that the 0/0/0 point is at the bottom and the 255/255/255 point is at the top. This is done purely for ease of visualization and this step can be ignored if unhelpful. Now, take each successive bitcount group, starting with the group of bitcount 0 and going in order of increasing bitcount, and place all members of that group as close to 0/0/0 as has not already been filled by a previous group. This results in the 25 groups layering up the color cube from 0/0/0 to 255/255/255 with positions in the cube further from 0/0/0 being both brighter and corresponding to larger 1D structures in the color dimension. The specific ordering of 1D structures of the same bitcount within a layer is currently arbitrary and creating a meaningful ordering is an area for future research. What 25L achieves is creating a color mapping where the brightness of a color correlates to the size of the corresponding structure. Though this is not a perfect correspondence due once again to the issues of fitting 24-space data into 3-space color, the variation in this correspondence is far less than the variation in the RGB color scheme. The result of this new color scheme, as compared to the RGB color scheme is pictured below in figure 9.

**Figure 9**

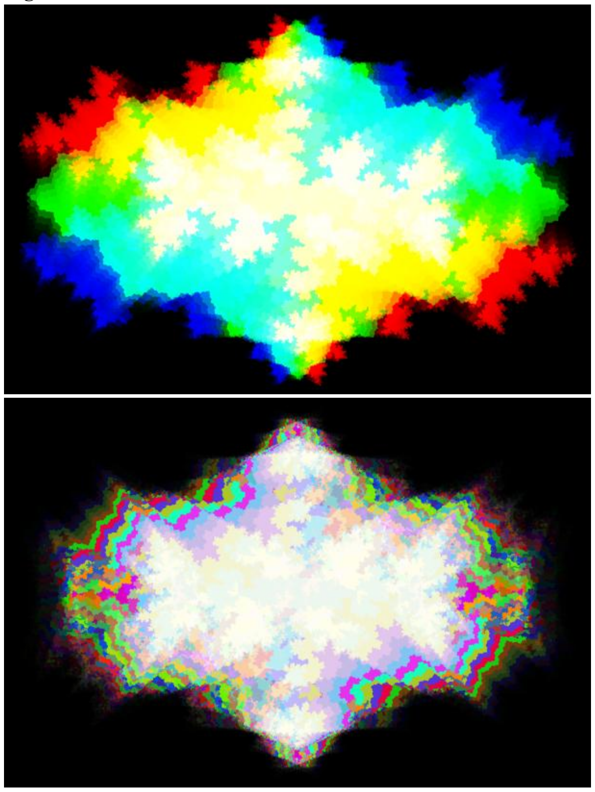


Figure 9 shows a direct comparison of the same frame of the XYTC visualizer in both the RGB color scheme (top) and the 25L color scheme (bottom).

Users can choose whether they would like the color dimension of their 4D data to be represented using the RGB color scheme, the 25L color scheme, or both. The option to show both at once is included as it provides the unique affordance of being able to compare the two color schemes at once. This allows users to learn to better interpret one color scheme from their existing understanding of the other color scheme.

Color schemes are an area of active ongoing research for XYTC and much of my ongoing and future research on this data visualization will work on creating more generalizable and more intuitive color schemes for mapping dimensional data onto color.

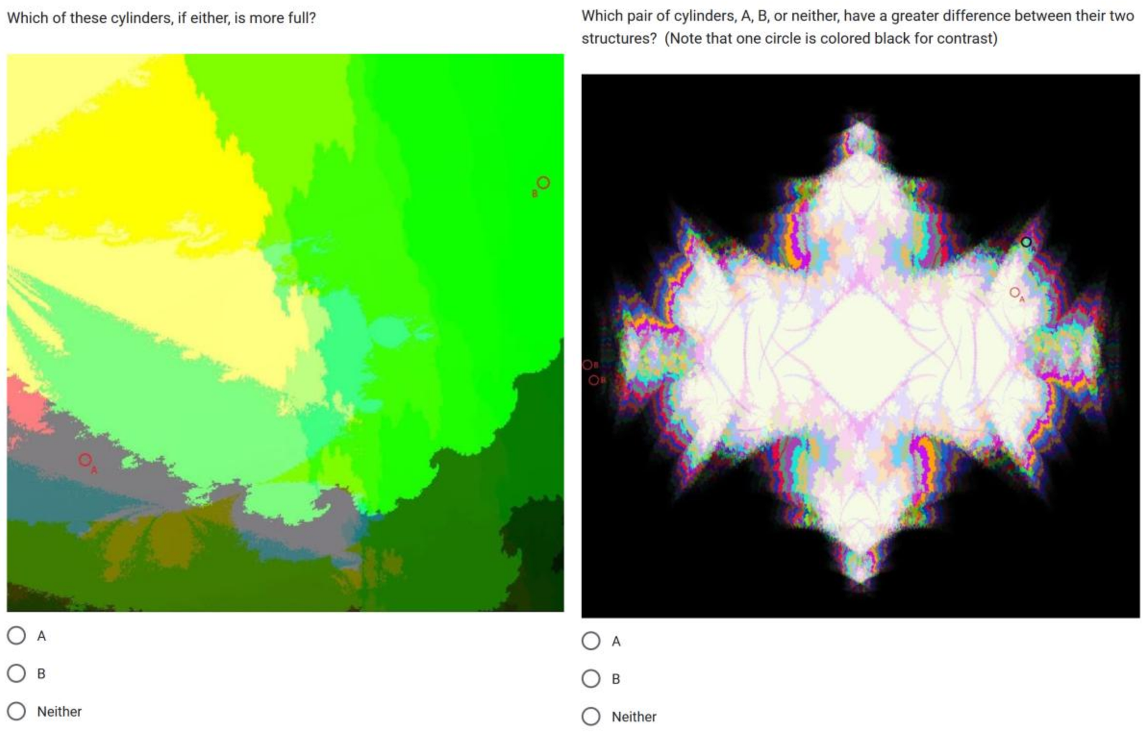
## USER STUDIES

My user studies are designed with two goals in mind. First, to evaluate the performance of XYTC as a visualization of 4D data, and second, to compare the effectiveness of the RGB and 25L color schemes.

### Quantitative Study

My first user study was a quantitative within-participants study where participants are asked to complete data analysis tasks on visualizations of 3D data reparameterized in terms of 2D+color. These visualizations are equivalent to still frames of XYTC (so 3D slices orthogonal to the time dimension). Some visualizations used the RGB color scheme while other questions used the 25L color scheme. Before beginning, participants were given brief descriptions of how both the RGB and 25L color schemes encoded 1D spatial data. Figure 10 shows typical questions from the quantitative study.

**Figure 10**



Which of these cylinders, if either, is more full?

A

B

Neither

Which pair of cylinders, A, B, or neither, have a greater difference between their two structures? (Note that one circle is colored black for contrast)

A

B

Neither

Figure 10 shows two questions from our quantitative study that participants were asked to answer. The left question uses the RGB color scheme while the right question uses the 25L color scheme.

### Participants

I recruited 5 participants for the quantitative study (3 female, 2 male; 3 in 20s, 1 in 30s, 1 in 60s). This same group of participants completed both the quantitative study and the qualitative study described in the next section.

### Results

All participants were able to successfully answer basic questions about which areas of the figure are completely full, partially full, and completely empty regardless of which color scheme was used. However, as seen in figure 11, when it came to participants being able to judge the magnitude of a color structure based upon its color (as well as the relative magnitudes of multiple color structures), there was significant variation within participants and across color schemes.

**Figure 11**

**RGB** **25L**

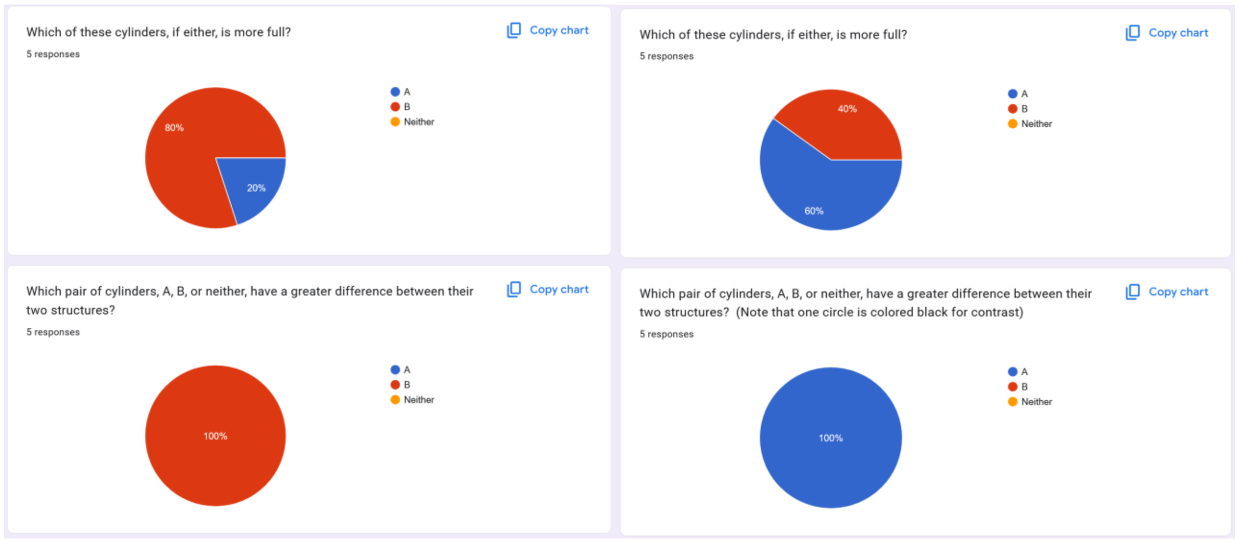


Figure 11 shows participant response to two questions across both color schemes. The two questions in the left column are in terms of the RGB color scheme and the two questions in the right column are in the 25L color scheme. In all four instances, the correct answer was 'A' and the same set of structures was used for both color schemes, it was only the representation of the color scheme that changed.

For each data analysis question asked, participants had to answer based on their interpretation of both the RGB and the 25L color schemes. In figure 11, the left instance of each question is for the RGB color scheme while the right instance is for the 25L color scheme, but, critically, the underlying data being visualized by both color schemes for a given question is the same. Yet, we see wildly different interpretations of this identical data across the color schemes.

For both questions, the correct answer was A, so we see that the 25L color scheme was much more effective at conveying accurate information. Since both of these questions are about magnitude or relative magnitude and this is what the 25L color scheme is designed to represent, these results are expected. In a post-survey interview, the two participants who answered incorrectly to which cylinder is more full in the 25L color scheme (that is, the upper-right subfigure of figure 11) explained their reasoning. They said that although they could tell that column A was brighter, it was also more washed out (closer to white), and the richer colors of column B felt more full to them.

**Qualitative Study**

Our second user study was a qualitative Comparative Structured Observation (CSO) where participants are asked to interact with the XYTC 4D Transformer. CSOs are a qualitative research method formalized in *Comparative Structured Observation*, Mackay et al. [15]. In our case, the comparison is the RGB color scheme against the 25L color scheme. Participants are given the general goal of trying to get a better understanding of the four-dimensional shape being visualized by XYTC. Participants are also given a brief explanation of how XYTC reparameterizes dimensions. For their first time exploring XYTC, participants use the RGB color scheme. Participants are encouraged to voice any thoughts that they are having while interacting with the system. Once the participant has indicated that they have finished exploring the system, I switch XYTC over to the 25L color scheme, at which point the process repeats. Once the participant has finished with the 25L color scheme, I first encourage them to share any thoughts they have about the differences between the color scheme and then I ask them a standard set of questions:

1. Which color scheme did you prefer?
2. Which color scheme helped you understand the 4D shape better?
3. Which color scheme made the shape more accessible, that is easier to naturally percieve?
4. In terms of the 4D geometry, what does it mean to play the animation?
5. In terms of the 4D geometry, what does it mean to scale in the color dimension?
6. In terms of the 4D geometry, what does it mean to scale in the time dimension?
7. In terms of the 4D geometry, what does it mean to rotate in terms of the color dimension?
8. In terms of the 4D geometry, what does it mean to rotate in terms of the time dimension?

**Results**

The CSO overall gave much more nuanced results to the question of which color scheme is best suited to representing the color dimension. Participants were regularly able to more rapidly interpret how rotation in the color dimension worked when using the RGB color scheme than when using the 25L color scheme. One participant explicitly noted that they could "see" how the object was rotating through the color dimension when using the RGB color scheme while they rotated in the x-color plane (labeled in the 4D Transformer as XH). As for the 25L color scheme, participants continued to get a better understanding of the magnitude of the color dimension with that color scheme. This result corroborates what we previously saw in the quantitative study.

Biases were also introduced by both color schemes. When using the RGB color scheme, participants focused on sharp color changes which are an artifact of how the RGB color scheme maps to color rather than any actual feature of the data itself. For the 25L color scheme, my explanation that the color scheme is designed to accurately represent the magnitude of the structure made participants focus on interpreting the magnitude of the structure to the exclusion of all other features.

Three participants additionally noted that the highly discrete color jumps present in the 25L color scheme were unpleasant with two of them adding that it actively took away from their ability to interpret the visualization when asked.

Beyond the comparative qualitative data that the CSO allowed us to collect, I was also able to collect some non-comparative data about the XYTC system as a whole, independent of color scheme. Most notably, all participants very rapidly developed an intuition about what to expect from various transformations of the visualizations. For example, the 4D Transformer currently has a glitch where sometimes the visualization will update incorrectly after a few rapid transformation inputs. Participants very often detected this glitch state and would use the 'update' button to fix the glitch. Although this feature of my CSO was completely accidental, I believe that it holds real information about how quickly participants learned what the visualization should look like upon a transformation. I hope to incorporate it into a future study.

Across all four dimensions, the x and y dimensions were absolutely the easiest to interpret. This is certainly expected as those two dimensions were not reparameterized in any way. The color dimension was also relatively easy for participants to understand. Though the understanding was not immediate, participants quickly built up an understanding of how the color dimension would behave and how to interpret it. Participants were also able to express relatively well what different transformations of the color dimension meant in terms of the 4D shape it helps to represent. The time dimension was absolutely the most difficult for participants to understand. The basic idea of the 4D shape being sliced orthogonal to the time dimension was clear and users were able to make sense of basic transformations in the time dimension, such as scrubbing through different slices, and scaling the time dimension. However all but one of the participants struggled to understand rotations in the time dimension or how the time dimension corresponded to a physical dimension. My current interpretation of this result is that while the x, y, and color dimensions are all visible in their entirety at any one time, by definition, you can only see one slice of the time dimension at once. As mentioned in my connection to SMT, this asymmetry between the time dimension and the other three dimensions makes my structural mapping between XYTC and a 4D shape less strong and potentially more confusing.

## CONCLUSION AND FUTURE WORK

XYTC is very much a work in progress. Much of both of my user studies were focused upon determining which of two color schemes is better suited to visualizing the color dimension of my reparameterization. Likely, XYTC will only end up using one of them, or perhaps even more likely it will use a different color scheme altogether that I have not yet discovered.

This preliminary paper outlines a set of obstacles which many existing 4D visualizers fall victim to and then explains how XYTC overcomes those issues. The paper then provides a brief overview of existing work in the field of higher dimensional visualization as well as past applications of both color and time for representing spatial information. Then a large section of the paper is dedicated to describe, in detail, how XYTC itself works. This begins with an explanation of how the design of XYTC is informed by Structural Mapping Theory. Then, low-level explanations of how a dimension of information can be reparameterized in terms of color and in terms of time. After this, the paper describes the higher-level interface with which users of XYTC interact, with a special focus on two different potential color schemes that can be used to describe the color dimension. Finally, both a quantitative and qualitative user study is described along with the results of the studies and analysis of those results.

XYTC represents a category of higher dimensional visualization with much potential: that of lossless higher dimensional visualization. By reparameterizing extra dimensions into other features of the data, XYTC describes a higher dimensional data both completely and losslessly.

There are a number of directions for future work. Within the bounds of XYTC, I intend to explore more color schemes beyond RGB and 25L as well as further explore the benefits and drawbacks of these two existing color schemes. Additionally, I intend to run more user studies which are focused on the efficacy of XYTC versus other existing four-dimensional visualization techniques in order to determine if XYTC is a viable alternative to these visualization approaches. Finally, I would like to explore the application of perceptually uniform color spaces like CEILAB to XYTC. Ultimately, the color dimension is only as useful as the accuracy of the perceptual data it conveys, so applications of such uniform color schemes would be appropriate to the goal of creating a maximally coherent visualizer.

Beyond the scope of XYTC specifically, I believe that future directions for research could focus on addressing some of the limitations of XYTC, such as the small size of its color dimension. Additionally, experimentations into parameterizing multiple dimensions into color (such as a red, green, and blue dimension or a hue, saturation, and value dimension) could generate interesting results.

Generative AI was not used in any way in the writing of this paper.

**References**

1. https://ieeexplore.ieee.org/document/7784854
2. https://onlinelibrary.wiley.com/doi/full/10.1002/cyto.a.20610
3. https://pubmed.ncbi.nlm.nih.gov/36040253/#:~:text=a%20visualization%20method%20that%20encodes,dimensional%20data.%20We%20illustrate%20its
4. https://link.springer.com/article/10.1007/s12650-020-00654-x
5. https://ieeexplore.ieee.org/abstract/document/5128904
6. https://www.sciencedirect.com/science/article/pii/S0925772107001071
7. https://www.researchgate.net/publication/319385751_Viscous_Fingering_A_Topological_Visual_Analytic_Approach
8. https://link.springer.com/chapter/10.1007/978-3-642-15014-2_20
9. https://dl.acm.org/doi/abs/10.1145/1476706.1476757
10. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=885704
11. https://onlinelibrary.wiley.com/doi/10.1207/s15516709cog0702_3
12. https://link.springer.com/chapter/10.1007/978-3-642-61717-1_13 pp 161–174
13. https://www.researchgate.net/publication/391197119_THE_PSYCHOLOGICAL_MEANING_OF_COLOR_IN_DESIGN_A_SEMANTIC_REVIEW
14. https://www.researchgate.net/publication/298734907_Face_Detection_and_Recognition_in_Color_Images_under_Matlab
15. https://ex-situ.lri.fr/content/7-workshops/1-comparative-structured-observation/1-course-handouts/comparative_structured_observation_tochi_2025.pdf
16. https://www.numdam.org/item/JMPA_1918_8_1__47_0.pdf
17. https://link.springer.com/chapter/10.1007/978-1-4757-4017-2_3

**Chi Visualization Papers used to model this paper:**

- https://dl.acm.org/doi/10.1145/2858036.2858307
- https://dl.acm.org/doi/pdf/10.1145/3290605.33+0272