1 Discussion

The Topological Equivariant Artist Model (TEAM) is a functional model of the structure preserving maps from data to visual representation. TEAM expresses the specifications that graphic and data must have equivalent *continuity* equivalent to the data, and that the visual characteristics of the graphics are *equivariant* to their corresponding components. TEAM expresses these constraints in the encoding ν, assembly Q, and indexing ξ functions that make up the artist A. This decomposition of the artist into smaller components functional pieces allows TEAM to provide well specified guidance on implementing visualization components based on this architecture. The proof of concept prototype built using this model validates that it is usable for a general purpose visualization tool. Additionally, the decomposition facilitates iteratively integrating TEAM into existing architecture rather than starting from scratch, since ν, Q and ξ functions can be implemented independently. This prototype demonstrates that this framework can generate the fundamental point (scatter plot) and line (line chart) marks. Furthermore, combining Butler's proposal of a fiber bundle model of visualization data with Spivak's formalism of schema lets TEAM support a variety of data continuities, including discrete relational tables, multivariate high resolution spatio-temporal datasets, and complex networks. Although the prototype currently only implements 0D and 1D continuity, we expected it to generalize to the other continuities.

1.1 Limitations

The TEAM model is a specification visualization library developers can use to implement structure preserving library components. Implementing a TEAM based architecture involves developers explicitly describing the structure and continuity of the data and the structure and continuity the artist expect. TEAM does not provide a framework for recommending visualizations to the user, therefore effectiveness [1, 2] is out of scope. But, automatic recommendation tools could be built using TEAM components.

While TEAM specifies the components, the developers building libraries using TEAM components decide which compositions of components are semantically correct for the domain. For this reason, TEAM does not include data space transforms, as are incorporated into libraries like Tableau or ggplot, instead leaving choice of computations to implementors of the data object. TEAM's intentional ignorance of semantics also means it cannot evaluate whether a figurative glyph [3] is a semantically correct choice, but it can enforce equivariance constraints of glyphs generated from data components enforcing equivariance of figurative glyphs [3] generated from data components [4, 5]. TEAM also allows graphics to have a lower dimensional continuity than the source data when a retraction map from one continuity to the other exists. For example, TEAM components could transform 1D continuous segments into 0D discrete elements, e.g. bar charts or scatter plots. As with computations, it is the role of the domain specific library to determine which figurative glyphs and continuity downgrades are appropriate.

The prototype is deeply tied to Matplotlib's existing architecture, so it has not yet been worked through how the model generalizes to libraries such as R graphics [6], VTK, and D3. TEAM has only been tested using PNG files rendered with AGG [7], but is expected to work with all the file types Matplotlib currently supports, including svg, pdf, and eps. We have not yet addressed how this framework interfaces with high performance rendering libraries such as openGL [8] that implement different models of ρ.

1.2 Future Work

More work is needed to formalize the composition operators, equivalence class A', and the mathematical model of interactivity. We also need to implement artists that demonstrate that the model can underpin a minimally viable library, foremost an image [9, 10], a heat map [11, 12], and an inherently computational artist such as a boxplot [13]. In summary, the proposed scope of work is

work period	**milestones & tasks**
April - July 2021	prepare and submit **conference presentation** on new functionality enabled by model for *SciPy*: artists that do not inherit from existing Matplotlib artists, computational artists such as histograms, non tabular data, composite interactive artist
June - Sept 2021	prepare and submit **theory paper** on interactivity to *TCVG or Eurovis 2022*: fully work out and describe math of addition operators and lookups from graphic to data space, implement brush linked artist (shared base space) and artist that exploits sheafs
May - Nov 2021	prepare and submit **applications paper** on high dimensional to *TCVG*: math and implementation of computational artists, concurrent artists and data sources, non-trivial data bundles
Aug 2021 - Feb 2022	prepare and submit **systems paper** on building domain specific libraries based on this model to *Infoviz 2022*: domain specific structured data, composite artists, inference of meta data components, mathematical notion of a visualization (labeled, multiple artists, etc)
March 2022	**dissertation** writing: synthesize previous work on climate data, compile topological equivariant artist model work
April 2022	**defense**

Table 1

In acknowledgement that the schedule is optimistic, this work has various scales of data applications. We plan to apply this model to datasets with complex continuities, such as the trajectories of rats running around a maze and the positions of their limbs. We also potentially can look at large scale biology or climate datasets. The data applications could be further integrated with topological [14] and functional [15] data analysis methods. Since this model formalizes notions of structure preservation, it can serve as a good base for tools that assess quality metrics [16] or invariance [17] of visualizations with respect to graphical encoding choices. This specification of structure could also be used to develop a serialization structure that could then be use to allow Matplotlib to interface with other visualization libraries such as open GL via shared serialization protocol. While this paper formulates visualization in terms of monoidal action homomorphisms between fiber bundles, the model lends itself to a categorical formulation [18, 19] that could be further explored.

2 Conclusion

A TEAM driven refactor of visualizations libraries could produce more maintainable, reusable, and extensible code, leading to better building blocks for the ecosystem of tools built on top of libraries with a TEAM driven architecture. Building block libraries could better support downstream, including domain specific, libraries without having to explicitly incorporate the specific data structure and visualization needs of those domains back into the base library. Adopting this model would induce a separation of data representation and visual representation that, for example, in Matplotlib is so entangled that it has lead to a brittle and sometimes incoherent API and internal code base. A refactor that incorporated the generalized data model and functional transforms presented in TEAM would lead to building block libraries that provide a more consistent, reusable, flexible, collection of blocks.