CHI 2021

IGScript: An Interaction Grammar for Scientific Data Presentation

Richen Liu [1], Min Gao [1], Shunlong Ye [1], and Jiang Zhang [2]

1) Nanjing Normal University
2) Peking University

Motivation

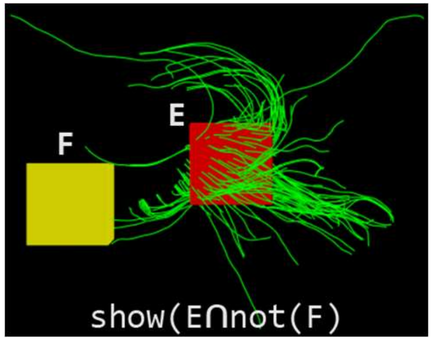
- Data story animation: reveal the dynamic changes and evolution processes

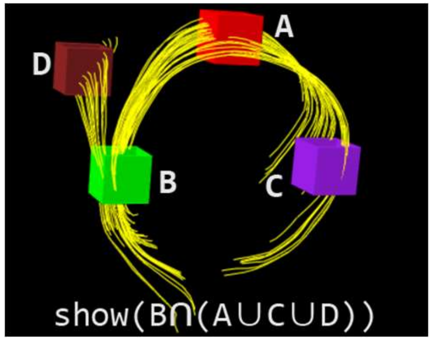
- Building data story animation is challenging: lack of tools on customizing animation contents, viewpoints, steps, styles of transitions, and shot changes

- Hard to customize set operator-based query expressions like "show (B∩(A∪C∪D))" to generate a dynamic animations by using GUI or traditional interactions

- Interpretive grammars: applied to customize static visualizations or even dynamic data-story animations flexibly

- We propose **IGScript**, a script-driven (interpretive grammars) and data-driven tool to help users build scientific data presentation animations





Background and Related Work

- Three classification criteria
  - External Grammars (Ext) v.s. Internal Grammars (Int)
  - Textual Grammars (Tex) v.s. Graphical Grammars (Gra)
  - DSVL, DSML, DSEL
- Most existing related wok focus on either the computation stage or the rendering stage. Few focus on the stages of interactive data presentations
- IGScript is more like a DSVL work while it focuses on **grammars for interactive data presentation animation** instead of **visualization**

	[6]Barringer et al.	[10]Barringer et al.	[11]Beyak et al.	[14]Brown et al.	[22]Cosentin et al.	[23]D'Amorim et al.	[36]Hong et al.	[40]Kindlmann et al.	[8]Barringer et al.	[28]Duke et al.	[34]Gunther et al.	[33]Gunther
Ext	■	■	■	■	■	■	■	■				
Int	■								■	■	■	■
Tex		■	■	■	■		■					■
Gra												
DSVL				■		■	■	■		■		
DSML												
DSEL									■		■	■

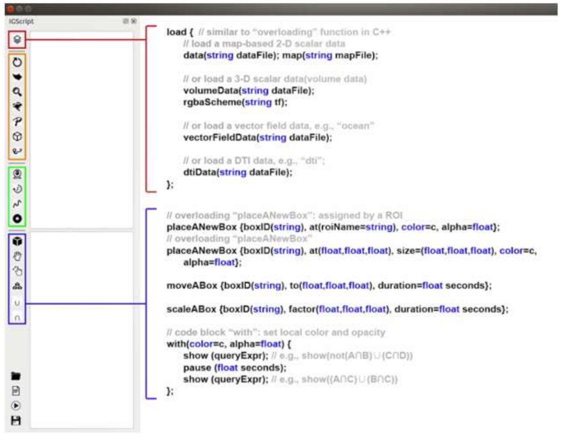
	[38]Karnick et al.	[2]Anderson et al.	[21]Choi et al.	[51]Ragan-K. et al.	[62]Silva et al.	[27]Diaz et al.	[43]Marques et al.	[48]Morgan et al.	[1]Almorsy et al.	[24]Dantra et al.	[32]Grundy et al.	[41]Li et al.
Ext												
Int	■	■	■	■								
Tex		■	■	■								
Gra					■	■	■	■				
DSVL	■	■	■					■	■	■	■	■
DSML	■		■	■								
DSEL		■	■	■								

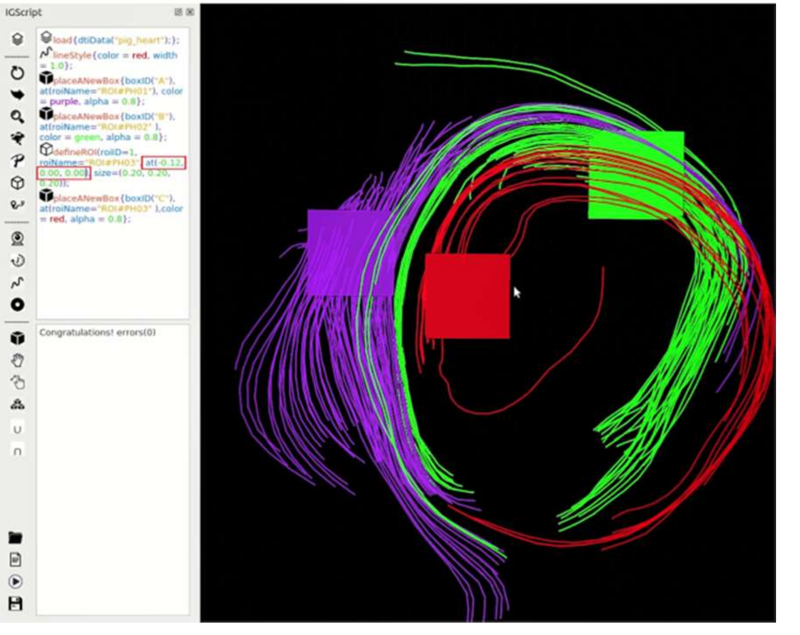
	[44]Marvie et al.	[52]Rath et al.	[56]Risoldi et al.	[59]Schulz et al.	[25]Deshayes et al.	Frank et al.	[16]Chafi et al.	[63]Sujeeth et al.	[4]Asenov et al.	[26]DeVito et al.	[5]Bachrach et al.	[12]Bostock et al.
Ext												
Int												
Tex							■					
Gra												
DSVL	■											
DSML		■	■	■	■	■						
DSEL							■	■	■	■	■	■

Table 1: Some of related work classified according to the three criteria [60]. They are external (Ext) or internal (Int), and the programming symbols are textual (Tex) or graphical (Gra), and DSVL is designed for visualization libraries, DSML is for modeling libraries, DSEL is designed for embedded libraries.

Design Rationale

- Design Goals
  - G1: help users define ROIs (region-of-interests) via coarse-grained and fine-grained interactions
  - G2: form a presentation animation by recording visual traversals or visual tracking across ROIs
  - G3: enable users to edit the animation clips of a data story in a semantic space



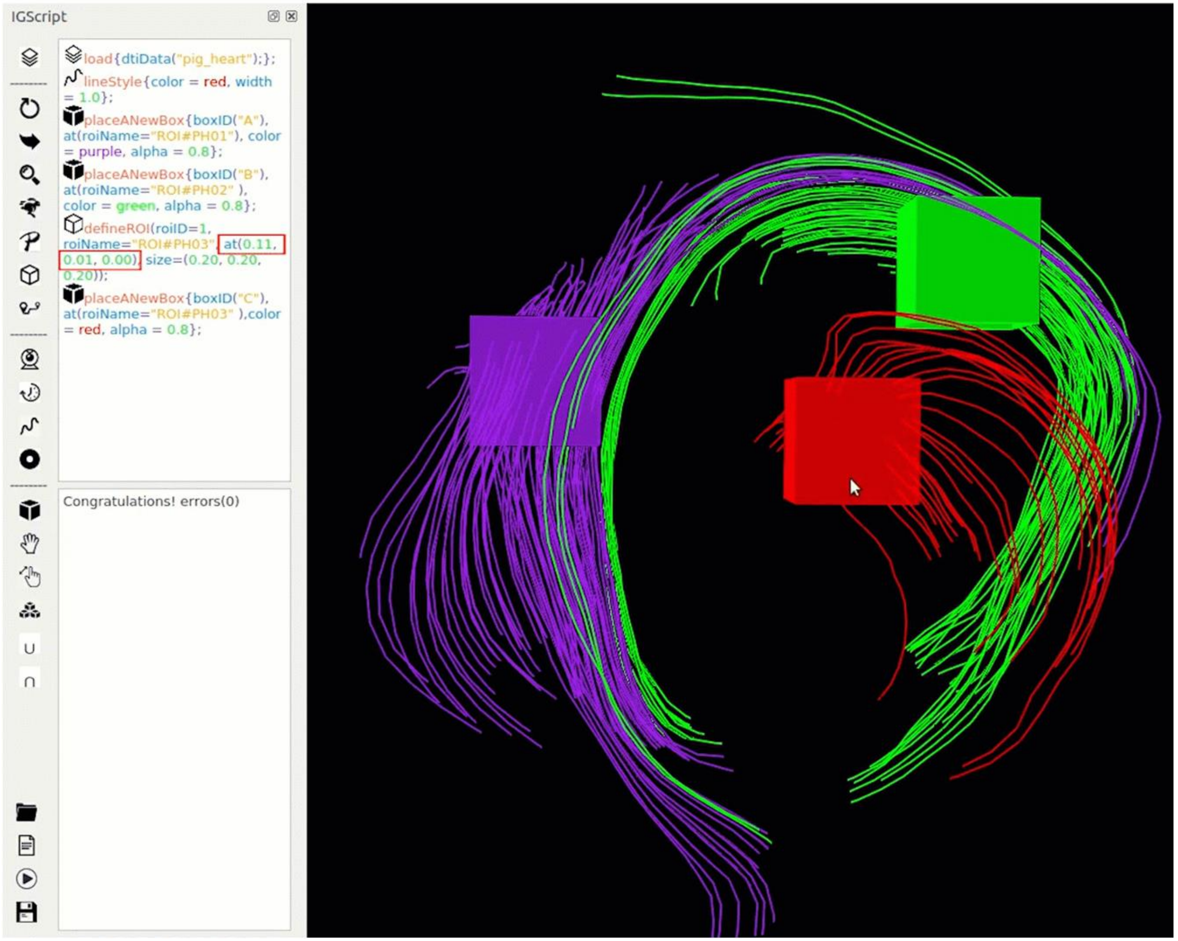


Overview & ROI Definitions

Video Figure

- Design Overview
  - A linked view: a visualization space view and a coding space view, providing **visual steering/visual feedback** for ROI definitions

- ROI Defs.
  - Place/move an ROI box in the visualization space by coarse-grained tuning (dragging)
  - Adjust the box slightly in the coding space by fine-grained tuning (text edit)



Grammar Design (1/2)

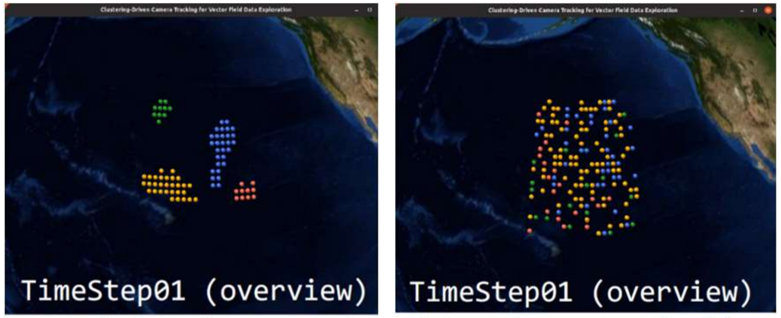
- General-purpose Grammars
  - Define ROIs: *defineROI*
  - Data loading*: load*, like a C++ overloading function to support different data types in run-time
  - Camera lens transformation around the ROIs: *rotate*, *translate*, *scale* (zoom), *parallel*
  - Data story animation: *animate*, *locate*

```
load { // similar to "overloading" function in C++
    // load a map-based 2-D scalar data
    data(string dataFile); map(string mapFile);

    // or load a 3-D scalar data (volume data)
    volumeData(string dataFile);
    rgbaScheme(string tf);// transfer function

    // or load a vector field data, e.g., "ocean";
    vectorFieldData(string dataFile);

    // or load a DTI data, e.g., "dti";
    dtiData(string dataFile);
};
```

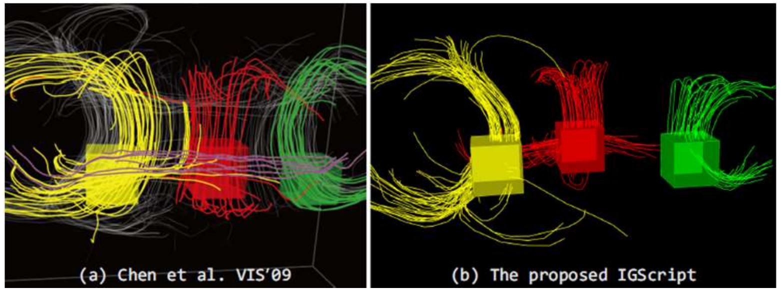
```
rotate {axis=string, angle=string, duration=float
    seconds};
rotate {axis=(float,float,float), angle=float
    degrees, duration=float seconds};
translate {to(float,float,float), duration=float
    seconds};
scale {factor=(float,float,float), duration=float
    seconds};
animate {speed=string}; // [low, moderate, high]
parallel { // executed concurrently
    rotate {axis=(float,float,float), angle=float
    degrees, duration=float seconds};
    scale {factor=(float,float,float), duration=
    float seconds};
    translate {to(float,float,float), duration=
    float seconds};
};

defineROI(roiID=int, roiName=string, at(float,
    float,float), size=(float,float,float));
locate {
    roiArray=string[roiName#1,roiName#2,roiName
    #3,...],
    foreach {
        duration=float seconds,
        interval=float seconds
```

Grammar Design (2/2)

- Application-specific Grammars
  - Generate data story animation for vector field data visualization and DTI data visualization
  - Vector field data visualization (OD query)
  - 3-D box queries and their arbitrary logic combinations for DTI fiber visualization



(a) mode = origin (b) mode = destination



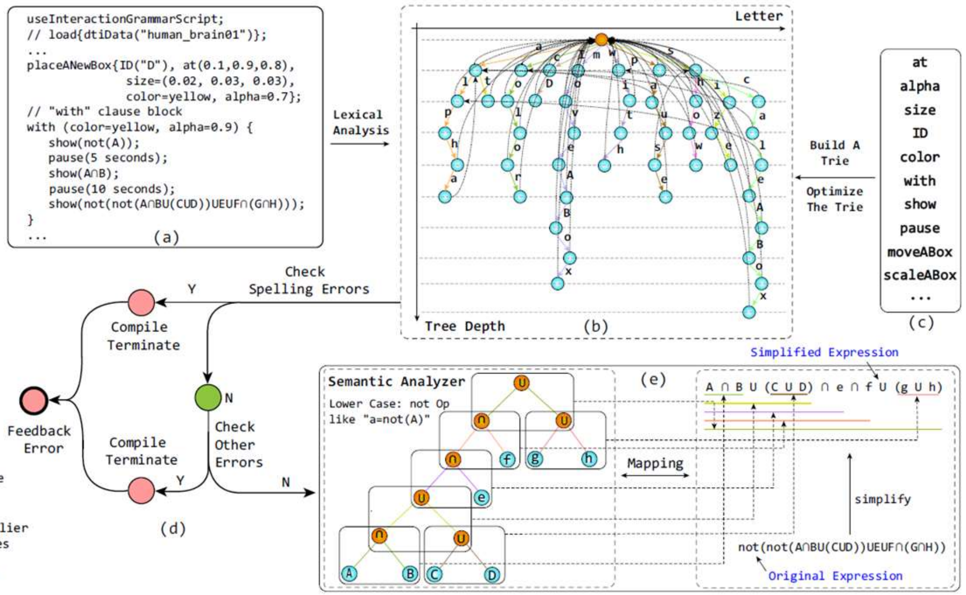

```
trace {
    mode=string, //[destination, origin, realtime]
    clusteringAlg=[dbscan, kmeans, pca, ...]
    lifeTime=float,//lifetime of traced fieldlines
    colorOfCenterPathline=[c1, c2, ...]
};
halfMergeSplit { // overview-to-details
    timeSlots=[overviewt1timeUnits, t2, t3, ...]
}
lineStyle {color=colorL, width=float};
tubeStyle {color=colorT, thickness=float};
```

```
// overloading "placeANewBox": assigned by an ROI
placeANewBox {boxID(string), at(roiName=string),
    color=c, alpha=float};
// overloading "placeANewBox"
placeANewBox {boxID(string), at(float,float,float)
    , size=(float,float,float), color=c, alpha=
    float};
moveABox {boxID(string), to(float,float,float),
    duration=float seconds};
scaleABox {boxID(string), factor(float,float,float
    ), duration=float seconds};

// code block "with": set local color and opacity
with(color=c, alpha=float) {
    show (queryExpr); //e.g., show(not(A∩B)∪(C∩D))
    pause (float seconds);
    show (queryExpr); //e.g., show((A∩C)∪(B∩C))
};
```

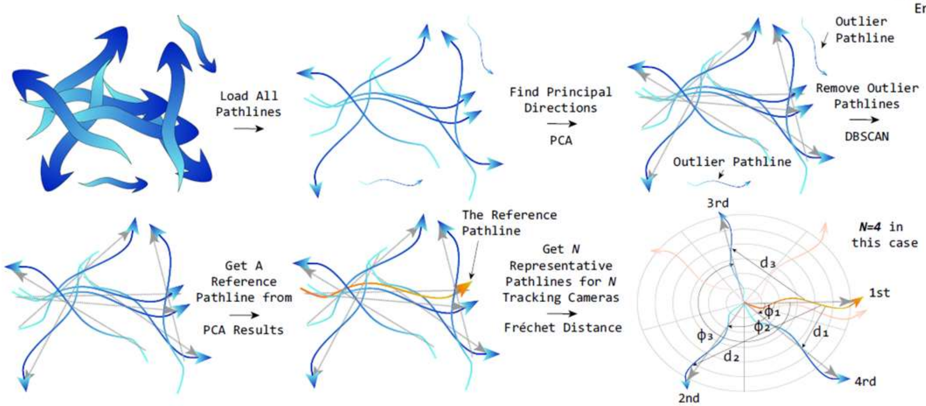
Design Details: Implementations

- A compiler converts textual grammar codes into a data story animation
- A code generator (decompiler) to translate the interactive data exploration animations back into the codes.

- IGScript makes the presentation animations **editable**, e.g., it allows to **cut**, **copy**, **paste**, **append**, or even **delete** some animation clips.



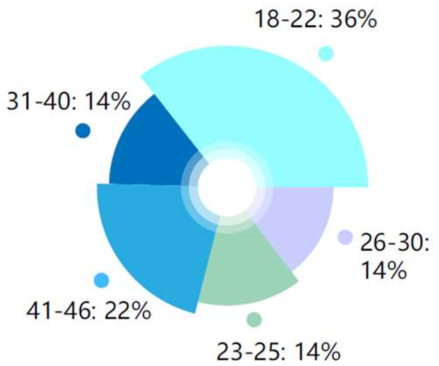
The compiler

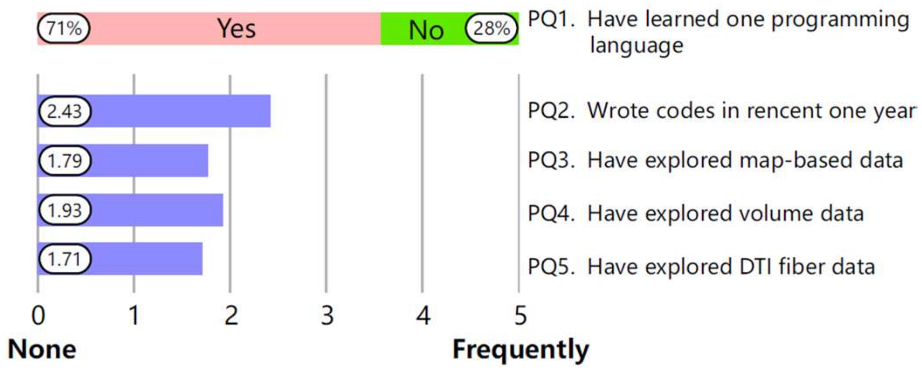


Camera tracking for pathline clusters (PCA components)

User Study

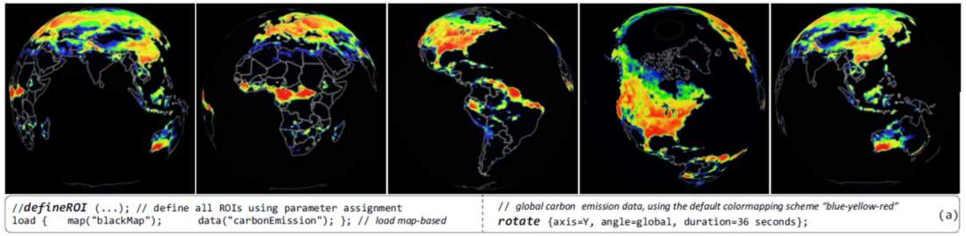
- Examine if users with limited programming skills could create their desired data presentation animations by IGScript.
- 14 participants: 8 doctors from different hospitals and 6 non-expert novice users with different majors
- Results
  - Users can define ROI easily (G1)
  - The generated animations are what users want (G2)
  - It is allowed to edit the small clips in an animation (**cut**, **copy**, **paste**, **append**, **delete**) (G3)



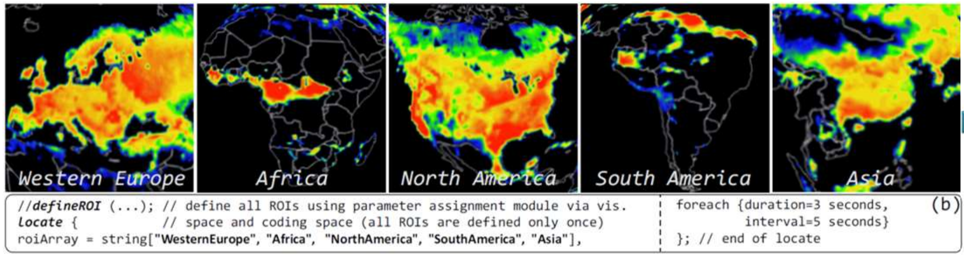


Case Study 01: 2-D Scalar Field Data

- 2-D scalar field: global carbon emission data

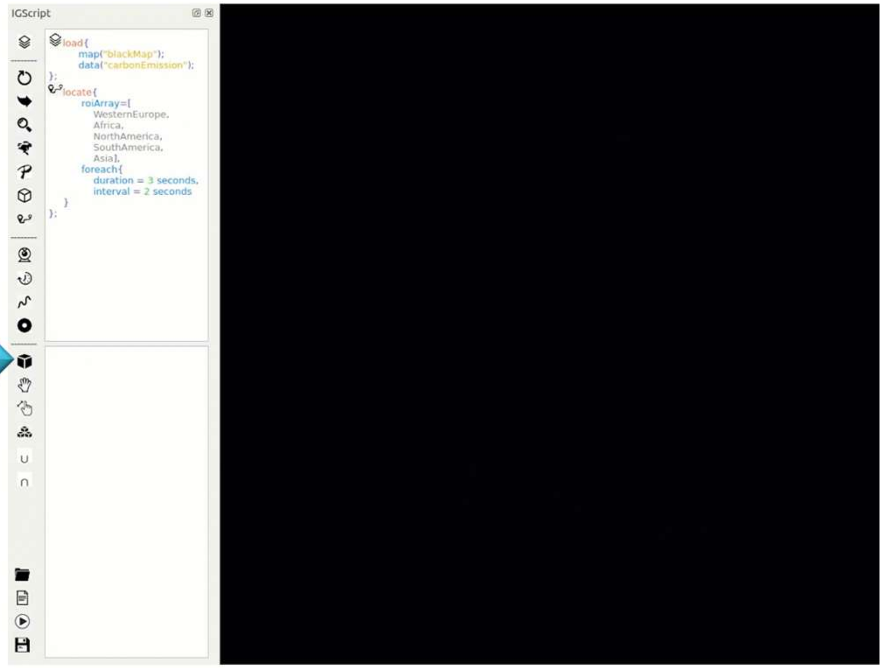


A 36-sec animation by the statement *rotate*



An animation with transitional scene switching across ROIs

Video Figure

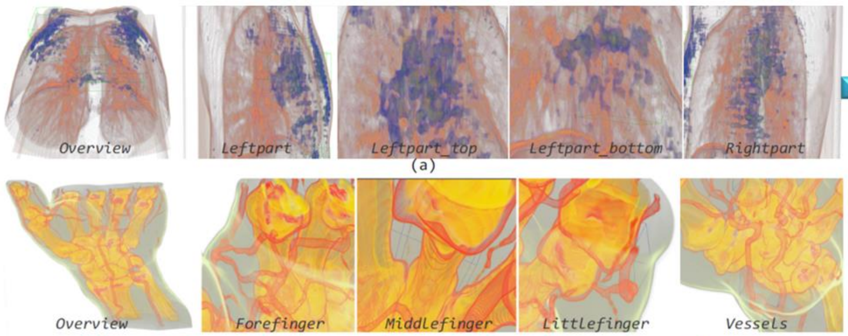


Case Study 02: 3-D Scalar Field Data

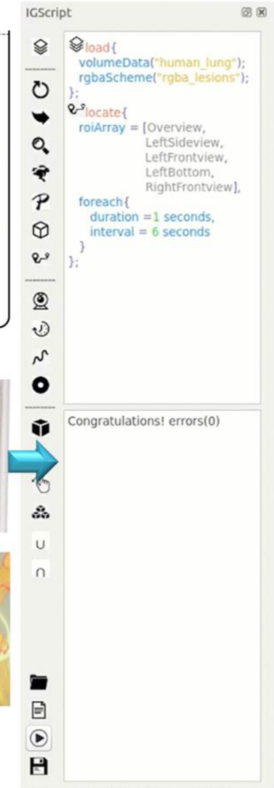
- 3-D scalar field data: CT scanning volume data LUNG (top) and HAND (bottom)
- The transitional scene switching follows the shot change style in film photography

```
//defineROI (...);
load { volumeData("human_lung");
rgbaScheme("rgba_lesions"); };
locate {
   roiArray=["Overview", "Leftpart",
       "Leftpart_top", "Leftpart_bottom",
       "Rightpart"],
   foreach {
       duration=25 seconds,
       interval=2 seconds }
}; // end of locate
```

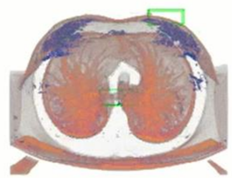
```
load { volumeData("hand");
rgbaScheme("boneSkinVessel");};
locate {
   roiArray=["Overview", "Forefinger",
        "Middlefinger", "Littlefinger",
        "Vessels"],
   foreach { duration=20 seconds,
        interval=2 seconds
   }
}; // end of locate
```



Snapshots of overview-to-details animations

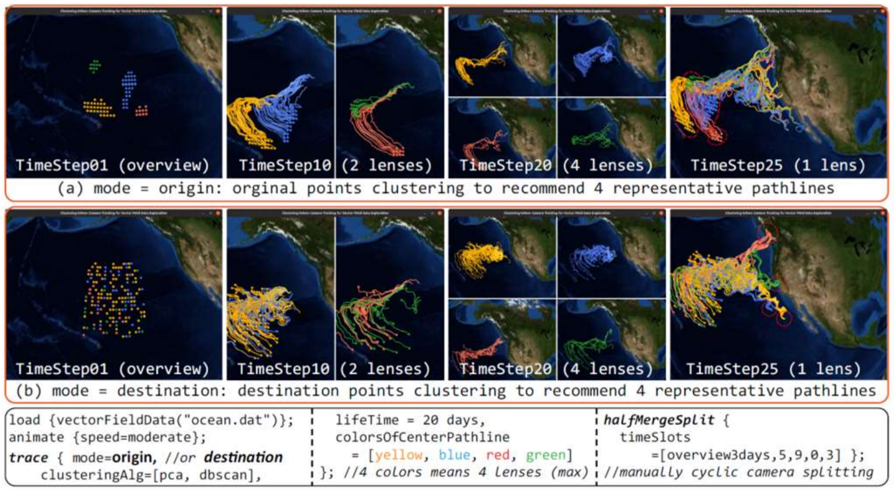


Video Figure



Case Study 03: Time-varying Vector Field Data

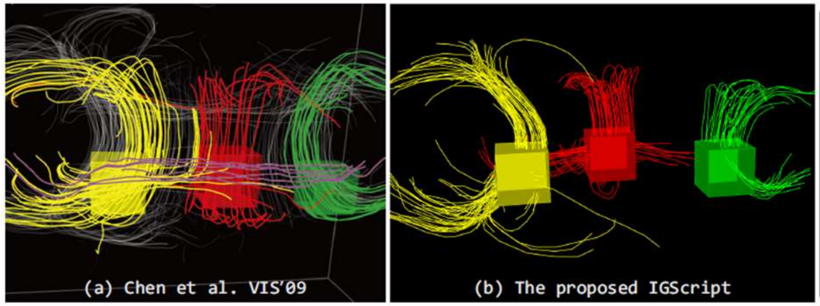
- Time-varying vector field data: global ocean pathline data
- The camera splitting/merging strategy is customized by ***halfMergeSplit***
- OD query animations: origin clustering (top) and the destination clustering (bottom)



Animation snapshots with four time-steps (#01, #10, #20, #25) for the ocean data

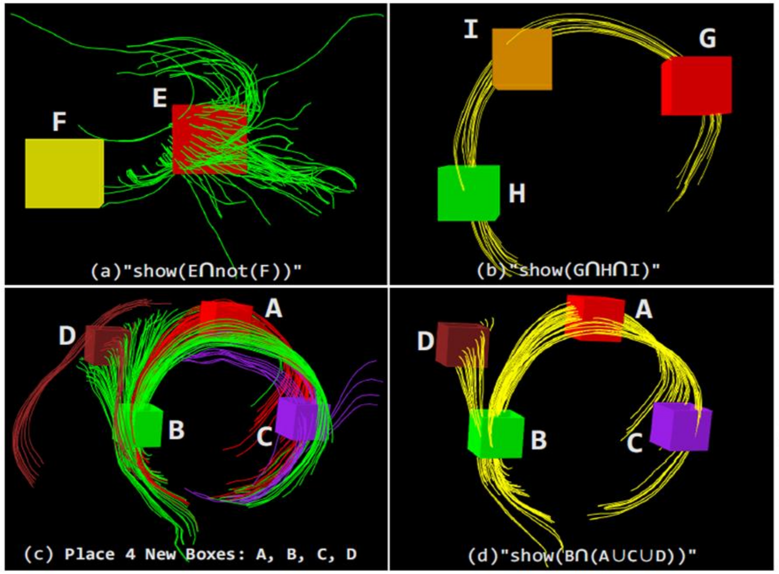
Video Figure

Case Study 04: Diffusion Tensor Imaging Data




```
// define all ROIs in parameter assignment module via visualization
// space and coding space, which will be invoked only once for each ROI
// defineROI (...); // defined ROIs are global variables

load {dtiData("human_brain01")};

lineStyle {color=green, width=1.0};

placeANewBox {boxID("A"), at("ROI#HB11"), color=green, alpha=0.5};
placeANewBox {boxID("B"), at("ROI#HB12"), color=red, alpha=0.6};
placeANewBox {boxID("C"), at("ROI#HB13"), color=yellow, alpha=0.6};
```

Easy to reproduce a similar result to the work [14] (left) without changing any rendering codes




```
// defineROI (...); // defined ROIs are global variables
load {dtiData("human_brain02")};
placeANewBox {boxID("E"), at("ROI#HB21"), color=red, alpha=0.8};
placeANewBox {boxID("F"), at("ROI#HB22"), color=yellow, alpha=0.8};   (a)
with (color=green,alpha=1.0) {
   show(E∩not(F));          }; // end of with
```

```
load {dtiData("pig_heart")};
placeANewBox {boxID("G"),at("ROI#PH11"), color=red,alpha=0.8};
placeANewBox {boxID("H"),at("ROI#PH12"), color=green,alpha=0.8};
placeANewBox {boxID("I"),at("ROI#PH13"), color=orange,alpha=0.8};    (b)
with(color=yellow,alpha=1.0) {
   show(G∩H∩I); }; // end of with
```

```
load {dtiData("pig_heart")};
lineStyle {color=red, width=1.0};
placeANewBox {boxID("A"), at("ROI#PH14"), color=red,alpha=0.8};      (c)
placeANewBox {boxID("B"), at("ROI#PH15"), color=green,alpha=0.8};
placeANewBox {boxID("C"), at("ROI#PH16"), color=purple,alpha=0.8};
placeANewBox {boxID("D"), at("ROI#PH17"), color=brown,alpha=0.6};
```

```
load {dtiData("pig_heart")};
with (color=yellow,alpha=1.0) {
   show(B∩(A∪C∪D));
   pause(5 seconds); // after 5-sec pause,
   // show(...);     // then show other queries
}; // end of with
```

$\textit{show}(B \cap (A \cup C \cup D))$

Complex expression-based queries for DTI fiber data

Discussion and Future Work

- Supported data currently: four general types of scientific data ("The Visualization Handbook" by Charles Hansen and Chris R. Johnson [28]):
  - 2-D/3D scalar field data (0th order tensors)
  - Vector field data (1st order tensors)
  - Tensor field data including DTI data (higher order tensors)
- Scalability on data types
  - It can be easily adapted to other use cases for these four types of data
  - Regarding a new data type, APIs on data loading and application-specific presentations (if any) should be added to the IGScripts library
- Suggestion by user study participants (future work)
  - Provide more sample codes
  - The identifiers and parameters could be redefined using their technical terms or closer to natural languages
  - Visualize the GUI flow charts for the textual codes

Conclusions

- We design an textual interaction grammar-based tool named *IGScript*
- A linked views, providing visual steering and visual feedback when customizing data presentation animations
- Script codes: look simple and follow the style of natural language-like grammars
  - A special-purpose compiler is designed to convert natural language-like grammar scripts into data presentation animations
  - A code generator (or a decompiler) is developed to translate the presentation steps into script codes.
- The animation clips are easy to be cut, copied, pasted, or even deleted
- Evaluations: a user study, four case studies and performance analysis demonstrate the usability and customizability of IGScript

Acknowledgement

- The reviewers for their valuable comments
- All the user study participants
- Funding
  - National Nature Science Foundation of China (61702271)
  - Postgraduate Research & Practice Innovation Program of Jiangsu Province (SJCX20_0445)

Thank You!