CAUSTA系统4.0开发计划

1. 前言

基本原则是数据模型、分析算法、展示子系统和计算引擎（与胡老师会商后确定相关方案，而且这块需要确定人选做对好接工作）相对独立。其中数据模型集成在API中，分析算法（包括数据IO、可视化[[1]](#footnote-1)及分析功能）以插件形式集成、展示子系统采用cesium架构，需要满足示范展示录屏和报告截图的需要。

1. 开发环境
   1. 数据模型和分析系统
      1. 开发环境：

VS2015(vc-14.0)、X64、Unicode字符集

* + 1. 库环境：

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 库 | 版本 | 备注 |
| 01 | SAGA | 7.8.0 |  |
| 02 | WxWidget | 3.1.3 |  |
| 03 | xerces | 3.1.1 | CityGML解析 |
| 04 | VTK | 9.0.1 |  |
| 05 | CLU | 2.0 |  |
| 06 | GASandBox | 1.0.7 | 基于Gaigen |
| 07 |  |  |  |

* 1. 展示系统

1. 模块功能说明
   1. 数据模型

集成至api，由于需要后期插件调用，这里采用源码级别的深度集成。目前已包含CaVector和STGrid两种数据模型，后期需进一步增加网络和张量数据模型。虽然是源码级的集成，由于代码量小，版本更新基本可以在1天内完成。

* + 1. CaVector

多维矢量数据模型，借鉴coverage数据模型，没有对拓扑进行显示表达，目前更侧重于不同格式2D/3D矢量数据的转换与统一存储。先直接集成进API,后期优化的方向是集成MVTree数据模型，并提供GA表达的输出，脚本化的GA算子运算，最终实现模板化运算。

目前该模型与dxf、citygml等数据读取绑定，导致api中会引入其它的第三方库，影响集成效率和稳定性。这次集成需要将相关的功能剥离，并以插件的方式松散集成。

* + 1. STGrid

时空场数据模型，基于原api中的grid修改，可以理解为基于时空立方体的时空场模型，没什么好优化的（后期可能需要考虑大数据量时空场数据读取问题），先直接集成到系统，后面主要跟张量模型配合作用。

* + 1. 网络模型

张季一老师开发中，已经有基本原型，具有矩阵和链接表两种存储方式，并可根据需要构建层次网络。模型上可供优化的或者展现亮点的不多，后期可能主要在生成式动态多约束网络分析上体现优势。

* + 1. 张量模型

开发中，有基本原型。提供从NC数据读取，张量分解，张量压缩，张量数据存取等功能。目前有模型插件框架，已基本实现NC数据管理。张量压缩有很琳耀和王健健两个不同的版本（但效率和功能的完整性都有待提升）。由于之前大量工作是在matlab上完成，迁移到C++上需要相关的数值分析库，这个王健健之前有对相关的库做过评估，由于是4年前的工作了，需要再去看看是否有新版本解决当时的问题。

* + 1. 时空框架
  1. 分析算法（功能）

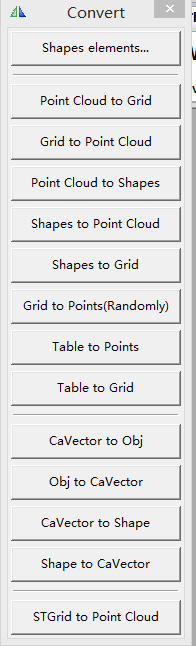
以插件的方式集成，基本上都是输入数据、设定参数然后输出。之前在老版本的saga上做过插件化，现在需要升级到新版本上来，难度应该不大。目前已有的插件有数据组织、几何代数空间转换、基本计算、网络分析、模板匹配等。后期可进一步集成的功能包括（以下功能可能不一定都能集成到系统，但需要做好代码、数据及说明文档的备份及梳理）：多约束网络/层次网络相关算法（朱帅、MCOP论文）、语义相关算法（石格格）、点云相关算法（袁帅）、张量相关算法（王健健、刘袁、李冬双）、台风场相关可视化和算法（邹宇、车骁宇）、PIR相关算法（冯琳耀、周春烨）、多粒度时空对象相关算法（傅蓉、张正方）、高速公路相关算法（项丽燕、高鸿）、数据溯源相关算法（胡靖瑶）、城市网络结构建模（杨力、高鸿、吴帆）

以下为目前已成熟可直接集成的功能插件。

* + 1. 数据转换

**Convert** --Convert

――Shapes element…

――――Points to Line[CLines\_From\_Points] CPoints\_to\_Lines

――――Line to points[CPoints\_From\_Lines] CLine\_to\_points

――――Line to Polygons[CPolygons\_From\_Lines] CLine\_to\_Polygons

――――Polygon to Lines[CLines\_From\_Polygons] CPolygon\_to\_Lines

――――Polygon to points[CPolygon\_To\_Points] CPolygon\_To\_Points

――――――――――――――

――Point Cloud to Grid[CPC\_To\_Grid] CPC\_To\_Grid

――Grid to Point Cloud[CPC\_From\_Grid] CGrid\_To\_PC

――Point Cloud to Shapes[CPC\_To\_Shapes] CPC\_To\_Shapes

――Shapes to Point Cloud[CPC\_From\_Shapes] CShapes\_To\_PC

――Shapes to Grid[CShapes2Grid] CShapes\_To\_Grid

――Grid to Points(Randomly)[CGrid\_To\_Points\_Random] CGrid\_To\_Points\_Random

――Table to Points[CPoints\_From\_Table] CTable\_To\_Points

――Table to Grid[CTable2Grid] CTable\_To\_Grid

――――――――――――――

--Citygml to CaVector

――CaVector to Obj[CCaVector2Obj]

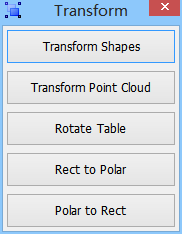
――Obj to CaVector[CObj2CaVector]

――CaVector to Shape[CCaVector2Shape]

――Shape to CaVector[CShapes2Cav]

――STGrid to Point Cloud[Cstgrid2ptc]

* + 1. 数据组织

**Transform** **--** Organize

――Transform Shapes[CTransformShapes] CTransform\_Shapes

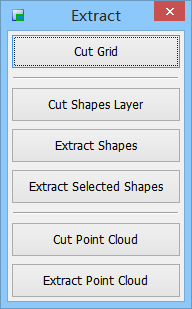
――Transform Point Cloud[CPC\_Transform] CTransform\_PC

――Rotate Table[CTable\_Rotate] CRotate\_Table

――――Rect to Polar[CCost\_RectToPolar] CCost\_RectToPolar

――――Polar to Rect[CCost\_PolarToRect] CCost\_PolarToRect

袁帅

**Extract** **--** Organize

――Cut Grid[CGrid\_Cut] CGrid\_Cut

――――――――――――――

――Cut Shapes Layer[CShapes\_Cut\_Interactive] CShapes\_Cut\_Interactive

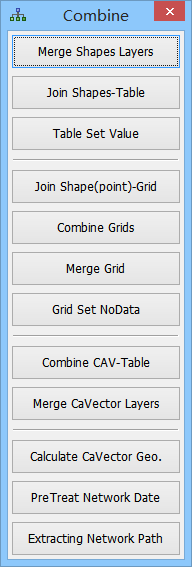
――Extract Shapes[CShapes\_Cut] CShapes\_Cut

――Extract Selected Shapes[CNewLayerFromSelectedShapes] CNewLayerFromSelectedShapes

――――――――――――――

――Cut Point Cloud[CPC\_Cut\_Interactive] CPC\_Cut\_Interactive

――Extract Point Cloud[CPC\_Cut] CPC\_Cut

**Combine** **--** Organize

――Merge Shapes Layers[CShapes\_Merge]

――Join Shapes-Table[CShapes\_Assign\_Table]

――Table Set Value[TableSetValue] --General**--** Organize

――――――――――――――

――Join Shape(point)-Grid[CGrid\_Values\_AddTo\_Points]

――Combine Grids[ CCombineGrids]

――Merge Grid[CGrid\_Merge]

――Grid Set NoData[CgridSetNoData] --General**--** Organize

――――――――――――――

――Combine CAV-Table [CCaVector\_Assign\_Table]

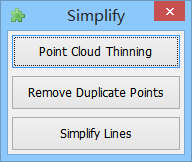
――Merge CaVector Layers [CCaVector\_Merge]

――――――――――――――

――Calculate CaVector Geo. [CCaVector\_Geometry\_Calc]

――PreTreat Network Date [CPreTreatNetworkDate] --General**--** Organize

――Extracting Network Path [Cbuildlinefromtxt] --General**--** Organize

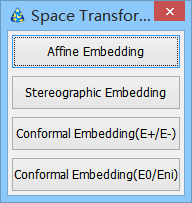
**simplify**  Organize

――Point Cloud Thinning (simple)[ CPC\_Thinning\_Simple]

――Remove Duplicate Points[CRemove\_Duplicates]

——Simplify Lines[CSimplifyShapes] //取消

* + 1. 几何代数空间转换



**Space Transform** -- GA Space

――Projective Embedding[Ctransform2affine]

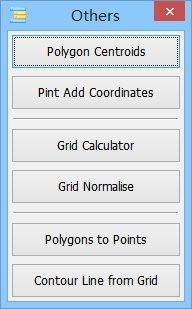
――Stereographic Embedding[transform2stereographic]

――Conformal Embedding[CONFORMAL1]

――Polynomial Embedding[CONFORMAL2]

* + 1. 基本计算

**Others** **--**Compute -- General

――Polygon Centroids[CPolygon\_Centroids]

――Pint Add Coordinates[CAddCoordinates]

――――――――――――――

――Grid Calculator[CGrid\_Calculator]

――Grid Normalise [CGrid\_Normalise]

――――――――――――――

――Contour Line from Grid[CGrid\_To\_Contour]

――Gradient from Grid[CGrid\_To\_Gradient]

* + 1. 网络分析

Network**--**Compute

――Data process 【CDataProcess】

――Max Flow 【Cmax\_flow】

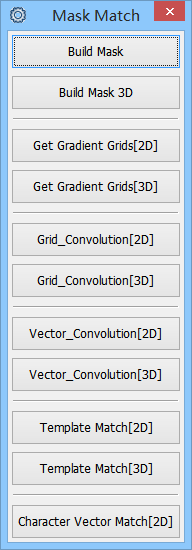
――Shortest Path[Module] 【CModelShortestPath】

――Node Constraint 【CNodeConstraint\_Path】

――Shortest Path 【CShortestPath】

网络分析/应急疏散等

* + 1. 模板匹配



**Mask Match** 

――Build Mask [Cbuild\_mask] 构建二维模板

――Build Mask 3D [Cbuild\_mask\_3D] 构建三维模板

――Gradient Grids[2D] [CgetGradientGrids] 二维梯度

――Get Gradient Grids[3D] [CgetGradientGrids3D] 三维梯度

――Convolution[2D] [CGridConvolution2D] 二维栅格卷积

――Grid\_Convolution[3D] [CGridConvolution3D] 三维栅格卷积

――Vector Convolution[2D] [CVectorConvolution2D] 二维矢量卷积

――Vector\_Convolution[3D] [CVectorConvolution3D] 三维矢量卷积

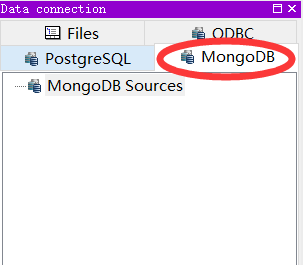
――Template Match[2D] [CTemplateMatch2D] 二维模板匹配

――Template Match[3D] [CTemplateMatch3D] 三维模板匹配

――Character Vector Match[2D] [CCharVectorConvolution2D] 特征模板匹配

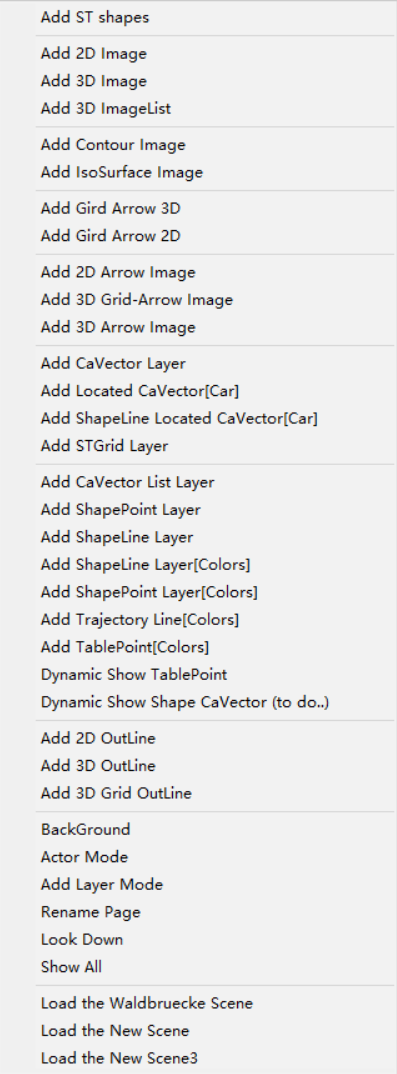
* 1. 数据管理与可视化功能

数据管理与可视化均是在GUI中以附加组件的方式实现。对于数据管理功能，之前已经在3.0版的系统中增加了mongoDB管理空间（如下图），利用回调函数可以实现基于插件的工作空间数据管理。数据项的右键菜单相关功能也可以绑定到插件上。这样只需要在GUI中增加一个用于GA数据管理的TreeCtrl，在API中添加一个全局变量，并提供相关的回调函数即可。与系统工作空间分离，可保证集成的便捷性和系统的稳定性。



可视化功能可基于GDI+VTK的方式实现（之前王健健做过原型实验），这样的好处是GUI跟VTK也是分离的，可极大提高稳定性。可视化方法可以借鉴之前的CCGA\_Visualization类，几乎实现了所有类型数据的可视化。但由于vtk版本升级了，代码需要微调，此外vtk可视化的效率还是不太高，特别是需要可视化的对象（Actor）较多的时候。可考虑OSG一类的替代方案，但由于会有专门的展示系统，这块也可以不做改动，只满足基本的可视化需求即可。

目前支持的可视化功能如下图，并可进行基本的图层管理，对象微调等。也支持批量场景文件导入。



* 1. 展示系统

主要用于前期成果的展示，需要设计相对大气的场景，体现几何代数的优势，并有一定的展示性。目前有以下场景可集成。

* + 1. 空间关系动态计算场景
    2. 海面时空场模板匹配场景
    3. 南极海地冰多层多时相建模场景
    4. 多数据融合的应急疏散场景
    5. PIR传感器模拟场景
    6. 中高层大气场景
    7. 六域模型语义综合场景（关系、图的表达）
    8. 物理和高维场景

1. 任务及进度安排
   1. 第一阶段：版本升级及迁移（2周）

主要完成API的升级、GUI基本框架的搭建、插件的升级。提供一个稳定的可供开发的环境。进度安排如下：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 任务名称 | 时间安排 | 责任人 | 备注 |
| 01 | API升级 | 20210323-25 | 王增杰/罗文 |  |
| 02 | GUI框架搭建 | 20210325-28 | 王增杰/罗文 |  |
| 03 | 插件升级 | 20210323-0405 | 潘炳煌/滕玉浩 | 28号之前完成基本插件升级，待框架搭建好后，完成GA插件升级 |

* 1. 第二阶段：数据模型及功能集成/展示系统（1个月）

主要完成网络模型和张量模型的开发与集成，并将部分前期成果集成到系统：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 任务名称 | 时间安排 | 责任人 | 备注 |
| 01 | 网络模型 | 20210323-31 | 张季一/王增杰 | 实现基本功能 |
| 02 | 张量模型 | 20210328-0405 | 王增杰/罗文 | 实现基本功能 |
| 03 | 多约束网络/层次网络相关算法（朱帅、MCOP论文）  语义相关算法（石格格）  点云相关算法（袁帅）  张量相关算法（王健健、刘袁、李冬双）  台风场相关可视化和算法（邹宇、车骁宇）  PIR相关算法（冯琳耀、周春烨）  多粒度时空对象相关算法（傅蓉、张正方）  高速公路相关算法（项丽燕、高鸿）  数据溯源相关算法（胡靖瑶）  城市网络结构建模（杨力、高鸿、吴帆） | 20210405-0505 | 王增杰/罗文/滕玉浩 | 待评估后再进行具体任务的划分 |
| 04 | 展示系统 | 20210405-0505 | 王增杰/罗文/潘炳煌 |  |

* 1. 第三阶段：数据模型及算法优化（2个月+）

主要是对前期矢量模型的优化，进一步完善张量模型、网络模型和时空框架。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 任务名称 | 时间安排 | 责任人 | 备注 |
| 01 | 网络模型 | 20210505-0705 | 张季一/王增杰 |  |
| 02 | 张量模型 | 20210505-0705 | 王增杰/罗文 |  |
| 03 | 矢量模型 | 20210505-0705 | 王增杰/罗文/潘炳煌/滕玉浩 |  |
| 04 | 时空框架 | 20210505-0705 | 王增杰/罗文/潘炳煌/滕玉浩 |  |

1. 项目管理

第一阶段由于涉及代码修改的工作量不大，且会有大量的源码提交任务，主要以文件交换的方式实现代码版本管理和共享。第二、三阶段拟采用私有git进行版本管理和进度控制。

1. 插件中的数据可视化满足基本过程测试并可部分满足发表论文的截图即可 [↑](#footnote-ref-1)