GLOVE 기능 확장을 위한 가시화 파이프라인 기술

(Visualization pipeline for GLOVE)

KISTI Technical Report

2018. 07.

이중연, 김덕수

목차

1. 소개 (Introduction) ····· 3
2. 개요 (Overview) ····· 4
3. 파이프 (Pipe) ····· 5
  가. 구조 및 작업흐름 (workflow) ····· 5
  나. 입출력 데이터 (Input/Output data) ····· 6
  다. 파이프의 종류 (Types of pipes) ····· 6
    1) 가시화 파이프 (Visualization pipe) ····· 6
    2) 통신 파이프 (Interface pipe) ····· 7
  라. 주요 클래스 (Important classes) ····· 7
    1) gloreWorkerPipelineBase ····· 7
    2) gloreWorkerProcBase ····· 8
    3) gloreWorkerProcBarrierBase ····· 10
    4) gloreWorkerDataManager ····· 10
    5) gloreWorkerResultCollector ····· 11
  마. 파이프 클래스 관계도 ····· 11
4. 파이프라인 (Pipeline) ····· 13
  가. 파이프라인 생성 및 구조 ····· 13
  나. 작업흐름 (Workflow) ····· 13
  다. 주요 클래스 ····· 14
    1) gloreWorkerPipelineManager ····· 14
5. 결과 이미지 (Results) ····· 15
감사의 말 ····· 16

그림 목차

그림 1. 가시화 파이프라인을 이용한 가시화의 기본 흐름 ····· 4
그림 2. 파이프 구조 및 작업 흐름 ····· 5
그림 3. 파이프의 구분 ····· 6
그림 4. 주요 클래스와 가시화파이프 클래스 사이의 관계도 ····· 11
그림 5. 주요 클래스와 가시화파이프 클래스 사이의 관계도 (그림 4 에서에서 연결) ····· 12
그림 6. 파이프라인의 구조 (GLOVE에 구현한 예 기준) ····· 13
그림 7. 가시화 결과의 예 1 ····· 15
그림 8. 가시화 결과의 예 2 ····· 16

1. 소개 (Introduction)

과학적 현상 분석 또는 더 나은 제품의 개발을 위해 과학자과 공학자들은 방대한 양의 측정/실험 데이터를 수집한다. 과학적 가시화(scientific visualization)는 이와 같은 데이터들을 시각적 정보로 바꾸어 주는 방법들의 집합이다. 과학적 가시화를 통해 방대하고 복잡한 데이터로부터 유의미한 정보를 얻을 수 있으며, 의료, 기상, 항공/우주, 중공업 등의 분야에서 널리 사용되고 있다.

과학적 가시화 기법들은 각각 독립적으로 적용되기도 하지만, 특정 활용분야에서는 여러 기법들을 결합한 결과를 요구기도 한다. 예를 들어, 원자로 내부의 유동을 확인하기 위해서는 원본 데이터를 클리핑(clipping)한 후, 다른 가시화 기법을 적용해야 한다(그림 9).
여러 기법이 적용된 가시화 결과는 하나의 기법을 적용한 중간 데이터를 저장하고, 그 데이터에 다음 가시화 기법을 적용하는 방식으로 얻을 수도 있다. 하지만 현재 GLOVE에 이러한 stage-by-stage 방법을 적용하기 위해서는, 모든 연산 노드들로부터 중간 결과를 수집하고 통합한 중간 데이터를 생성하고, 새로운 가시화 연산을 시작해야 하는 큰 부담(overhead)가 따른다. 또 다른 접근법은, 응용이 요구하는 목표 가시화 기법의 조합을 새로운 가시화 질의(visualization query)를 정의 및 구현하는 방법이다. 하지만 응용분야 마다 요구하는 가시화 기법이 상이할 수 있으며, 이러한 모든 조합을 구현한다는 것은 비현실적이다.

이러한 한계점들을 해결하고 다양한 조합의 가시화 요청을 효율적으로 처리할 수 있도록 GLOVE를 위한 가시화 파이프라인(*G-Pipeline*) 프레임웍을 설계하였다. 제안된 가시화 파이프라인 기술은 다음과 같은 장점을 가진다.

- **유연함 (Flexible)** : 제안된 G-Pipeline에서 각각의 가시화 기법(query)은 독립된 가시화 파이프(visualization pipe)로 표현되며, 파이프들은 사용자의 요청에 따라 서로 동적으로 연결될 수 있다. 즉 G-Pipeline은 수동적인 조작 없이 주어진 가시화 기법의 조합을 표현하는 파이프라인을 구성하고, 요청된 결과를 생성한다.
- **효율적 (Efficient)** : G-Pipeline에서 각 노드는 주어진 블록(block)들 요청받은 일련의 가시화 기법을 적용하며, 그 결과는 마지막 파이프에 의해 취합된다. 즉 G-Pipeline는 꼭 필요한 경우를 제외하고는 중간 결과를 생성하지 않는다. 그 결과 불필요한 연산 및 메모리 소비를 줄일 수 있다. 또한 G-Pipeline는 병렬처리 알고리즘으로 쉽게 확장 가능한 구조를 가지고 있어, 병렬처리 자원을 효율적으로 활용할 수 있다.
- **이식성 (Portable)** : G-Pipeline는 가시화 SW의 다른 요소(component)와의 통신은 특별히 정의된 전담 파이프들(i.e. interface pipe)에 의해서만 이루어진다. 따라서 GLOVE가 아닌 다른 SW에 적용하기 위해서는 통신 전담 파이프만 구현하면 되며, 높은 이식성을 가진다.
- **확장성 (Extensible)** : 통신 전담 파이프를 제외한 내부 파이프들(i.e. visualization pipe)은 정형화된 파이프 형태(예, 입력/출력 형태)를 따르면서, 자신의 역할에 충실한 형태로 구현된다. 즉 새로운 가시화 기법의 추가는 해당 기법에만 충실하도록 독립적 구현이 가능하며, 기존 파이프들과도 유연하게 연결될 수 있다.

2. 개요 (Overview)

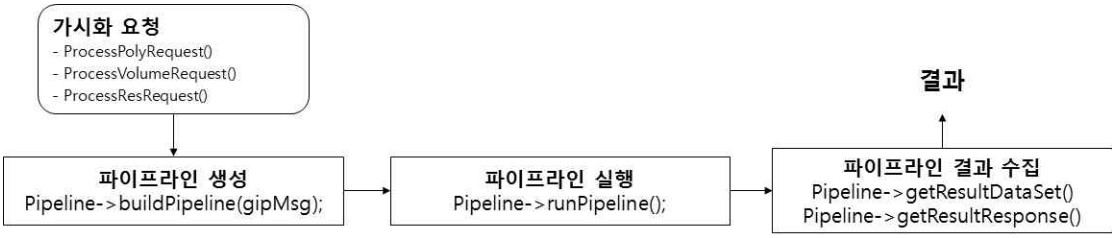


그림 2. 가시화 파이프라인을 이용한 가시화의 기본 흐름

제안하는 가시화 파이프라인(G-Pipeline)의 기본 일 단위(work unit)는 가시화 파이프(pipe)다. 각각의 파이프는 Iso-surface, glyph 등 하나의 가시화 기법을 표현한다. 파이프들은 사용자의 요청에 따라 서로 동적으로 연결되어 파이프라인을 생성하고, 최종 가시화 결과를 생성한다.

그림 2은 GLORE[1)]에서 G-Pipeline을 이용한 가시화의 기본 흐름을 보여준다. 사용자로부터 받은 요청(i.e. gipMsg)이 파이프라인 프레임웍에 전달되면, G-pipeline은 해당 요청을 해석(parsing)하고 그 순서에 맞추어 가시화 파이프라인을 자도응로 생성한다. 생성된 가시화 파이프는 통신 전담 파이프들과 연결되어, GLORE로부터 데이터를 입력 받고 가시화 결과를 되돌려 준다. 요청 해석 및 파이프라인 생성은 사용자의 수동적 작업 없이 자동으로 이루어지며, 다양한 사용자의 요청을 처리할 수 있다.
현재 G-Pipeline이 지원하는 가시화 기법 및 관련 파이프에 대한 정보는 표 1에서 확인할 수 있다. 각 파이프의 실제 구현한은 gloreWorkerProc{Pipe name} 클래스에서 확인할 수 있다.

본 기술문서의 3장에서는 파이프를, 4장은 파이프라인에 대해 상세히 기술하고, 4장에서는 G-Pipeline을 적용한 결과의 일부를 보여준다.

1) GLOVE에서 가시화 연산을 담당하는 가시화 서버 컴포넌트

표 1. 현재 G-Pipeline이 지원하는 가시화 파이프 정보

Pipe name	Viz. query	Input	Output	Connecting to next pipe
ClipData	Data clipping	volume	volume	O
CutPlane	Cutting plane	volume	surface	O
	Contour line	volume	line	Glyph only
ExtractSurface	Extract surface	volume	surface	O
IJKGraph	IJK Graph	volume	glvResponse	Terminal
IsoS	Iso-surface	volume	surface	O
		surface	line	Glyph only
ProbeByPoints	Point probing	volume	point	Glyph only
		surface	point	Glyph only
Glyph	Glyph	volume	point	Terminal
		surface	point	Terminal
GlyphOverPlane		line	point	Terminal
		point	point	Terminal
SurfaceDisplay	Scalar display	volume	surface	O
XYZGraph	XYZ Graph	volume	glvResponse	Terminal

3. 파이프 (Pipe)

본 장에서는 G-Pipelne의 기본 구성단위인, 파이프에 대해서 살펴본다.

가. 구조 및 작업흐름 (workflow)

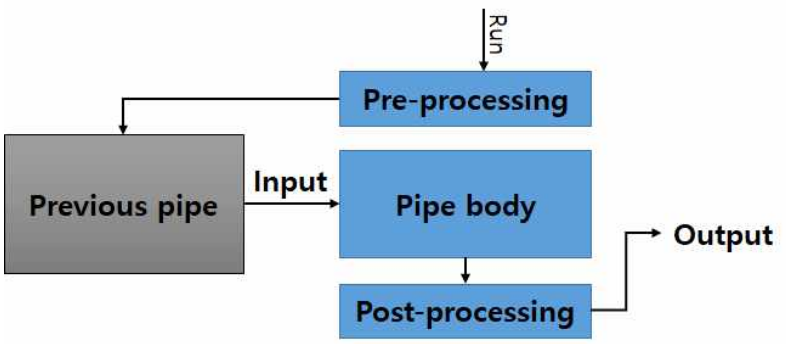


그림 3. 파이프 구조 및 작업 흐름

파이프의 역할은 입력 데이터(input data)를 받아 정의된 가시화 연산을 수행 한 후, 그 결과(output)를 다음 파이프로 전달하는 것이다. 파이프 사이의 유연한 연결을 위해 모든 파이프는 정형화된 구조(unified form)를 따른다(그림 3). 하나의 파이프는 다음 세 부분으로 구성된다.

- **Pre-processing** : 파이프에 진입 시 처음으로 호출되며, 파이프를 초기화 하고 가시화 연산에 필요한 사전 작업을 수행
- **Pipe body** : 선행 파이프로부터 입력 데이터를 받고, 가시화 연산을 수행 한 후, 그 결과를 Post-processing으로 전달
- **Post-processing** : 파이프가 모든 작업을 끝낸 후 수행되며, 파이프를 정리하고 최종 데이터의 가공 작업을 수행

파이프가 실행되면 1) 파이프를 초기화 한 후, pre-processing 단계 수행을 통해 가시화 연산을 준비한다. 다음으로 2) 선행 파이프에게 데이터(input data)를 요청하고, 3) 전달 받은 데이터에 대해 가시화 연산을 수행한다. Visualization-barrier pipe (3-다. 참조)의 경우는, 선행 파이프에서 전달 받을 데이터가 더 이상 없을 때 까지 2) 과정을 반복한다. 가시화 연산이 끝난 후에는 4) post-processing 단계에서 파이프의 작업을 마무리하고 결과 데이터(output data)를 생성한다. Pre-processing과 Post-processing 작업은 전달받을 데이터(input data) 없더라도 수행된다.

나. 입출력 데이터 (Input/Output data)

현재 G-Pipeline은 vtkDataset과 glvResponse 두 가지 데이터 형태(data type)를 지원한다. glvResponse는 vector 형태로 전달되며, 복수개의 glvRepose를 입/출력 데이터로 사용할 수 있다.

다. 파이프의 종류 (Types of pipes)

파이프는 크게 가시화 파이프와 통신 파이프로 구성된다(그림 4).

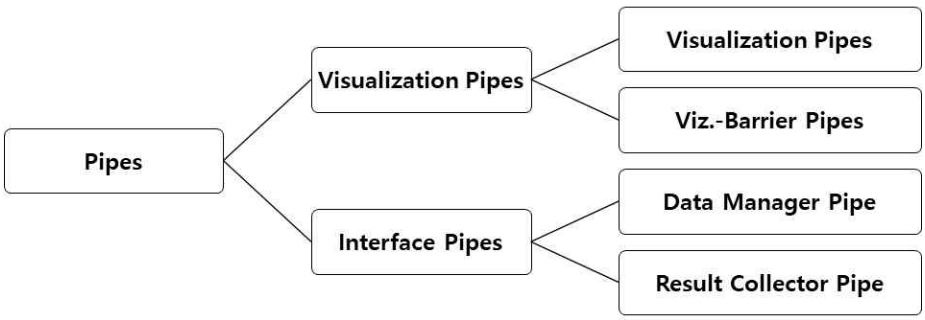


그림 4. 파이프의 구분

1) 가시화 파이프 (Visualization pipe)

가시화 연산을 수행하는 파이프로, 각각의 파이프는 하나의 가시화 연산을 표현한다. 가시화 파이프는 다시 기본 가시화 파이프(Visualization pipe)와 barrier 가시화 파이프(Viz.-barrier pipe)로 구분된다.

일부 가시화 연산은 여러 블록들 사이 또는 여러 time-step들 사이의 관계를 고려해야 한다(예, point probing). 이러한 경우 복수개의 데이터에 대한 가시화 연산 후, 최종 결과(output)를 만들 수 있다. Viz.-barrier pipe는 선행 파이프에 더 이상

출력 데이터가 없을 때까지 pipe body를 수행 한 후, post-processing 단계로 진입하여 최종 결과를 만들도록 설계 되어 있다.

Viz. pipe와 Viz.-barrier pipe는 각각 gloreWorkerProcBase class와 gloreWorkerProcBarrierBase class를 상속 받아 구현함으로써 정형화된 파이프의 형태를 갖춤을 보장 할 수 있다.

2) 통신 파이프 (Interface pipe)

G-Pipeline은 가시화 SW를 구성하는 하나의 컴포넌트(component)다. 통신 파이프는 G-Pipeline과 그것을 사용하는 SW 사이의 통신을 전담한다. 그 덕분에 가시화 파이프들은 가시화 SW와 독립적으로 구현 될 수 있으며, 자신의 가시화 연산에 집중할 수 있다. 또한 통신 파이프만 구현함으로써, 다른 가시화 SW에서 기존 구현된 가시화 파이프를 사용 할 수 있다.

가장 기본이 되는 통신 파이프는 데이터 관리 파이프(data manager pipe)와 결과 수집 파이프(result collector pipe)로, 이 목표 SW에 대응하는 이 두 파이프는 반드시 구현되어야 한다.

- **Data manager pipe** : 가시화 SW로부터 입력 데이터를 받아, G-Pipeline이 처리 할 수 있는 형태로 변환하여 전달하는 파이프
- **Result collector pipe** : G-Pipeline에서 생성된 결과들을 수집하고, 최종 결과를 가시화 SW가 원하는 형태로 변환 후 반환해주는 파이프

GLORE를 위한 통신 파이프는, gloreWorkerDataManager class와 gloreWorkerResultCollector class로 구현되어 있다.

라. 주요 클래스 (Important classes)

1) gloreWorkerPipelineBase

G-Pipeline의 가장 기본이 되는 최상위 클래스로, 모든 pipe가 상속받아야 한다. 파이프의 기본동작 및 파이프 간 데이터 흐름에 대한 정의를 담고 있다.

- 주요 멤버 변수 (member attributes)

Type	Name	Note
gloreWorkerPipelineBase*	prevPipe	이전 pipe 객체를 가리키는 포인터
int	dataTimeStep	현재 input data의 time step
vtkDataSet*	inputData	이전 파이프로부터 받은 input data
vector<glvResponse*>	inputResponses	
void*	inputExtraPtr	
vtkDataSet*	outputData	다음 파이프에 전달 할 output data
vector<glvResponse*>	outputResponses	
void*	outputExtraPtr	
vector<pair<int,int>>	varList	파이프가 유지하는 variable list

• 주요 멤버 메소드 (member methods)

이름	설명
runPipe()	pipe의 work flow 정의 (3-가. Work flow 설명 참조)
preProcessing()	Pipe의 동작 정의 (3-가. Work flow 설명 참조)
postProcessing()	
Process()	
setPrevPipe()	이전 파이프 연결
isMorePipelineInput()	남은 input data가 있는지 확인

2) gloreWorkerProcBase

가시화 파이프(Visualization pipe)의 가장 기본이 되는 클래스로, 모든 가시화 파이프가 상속받는다. 가시화 파이프들의 공통 변수 및 공통 루틴이 정의되어 있으며, 가시화 파이프 동작에 기본이 되는 메소드가 선언되어 있다. gloreWorkerProcBase를 상속받은 가시화 파이프들은, 이 메소드들을 정의함으로써 G-Pipeline의 가시화 파이프로 동작 가능하게 된다.

• 주요 멤버 변수 (member attributes)

Type	Name	Note
vector<int>	timeStepList	pipe가 처리할 time step 리스트
vector<int>	elementList	pipe가 다룰 element 리스트
int	fillerID scalarID	varible을 지정하기 위해 사용 struct의 union 형태로 정의 됨
int	fidx sidx	
int	geometryID gidx	
int	coutourID cidx	

• 주요 멤버 메소드 (member methods)

이름	설명
initProc()	주어진 gipMessage를 기반으로 파이프를 초기화 - variable, time step 및 파이프가 사용하는 인자 설정 (예, iso value 등) **<Development note>** 파이프를 구현할 때 반드시 정의해야함 - gloreWorkerProcIsoS.cpp, gloreWorkerProcGlyph.cpp 등 참조 - 관련매크로는 gloreWorkerProcBase.h 에 정의되어 있음 : _PROC_BASE_INPUT_DATA_CHECK : _PROC_BASE_SET_ELEMENT_LIST : _PROC_BASE_SET_TIMESTEP
setInitialDataManager()	해당 파이프가 첫 번째 파이프인 경우, data manager 파이프를 초기화하기 위해 호출 됨 (4-가. 파이프라인 생성 부분 참조) **<Development note>** 기본동작은 gloreWorkerProcBase 클래스에 정의되어 있으며, 특별한 경우가 아니라면 별도 정의가 필요 없음. Fetch 방법이 일반 파이프와 다른 경우, 해당 파이프에서 직접 구현 (예, ExtractSurface 파이프)
setResultCollector()	해당 파이프가 마지막 번째 파이프인 경우, result collector 파이프를 초기화하기 위해 호출 됨 (4-가. 파이프라인 생성 부분 참조) **<Development note>** 기본동작은 gloreWorkerProcBase 클래스에 정의되어 있으며, 특별한 경우가 아니라면 별도 정의가 필요 없음. 결과 수집 방법이 일반 파이프와 다른 경우, 해당 파이프에서 직접 구현 (아직 직접 구현한 파이프 없음)

getResultType()	파이프를 구현할 때 반드시 정의해야함 (구현된 소스 코드 참조)
	파이프의 결과 type을 정의. result collector 설정에 사용됨. gloreWorkerPipelineBase.h에 enum으로 정의되어 있음
	`namespace GLORE_PIPE_DATA_TYPE{`
	`enum Type {Unknown, vtkDataSet, glvResponse, Both, End};`
	`}`

getPipeName()	Pipe의 이름을 문자열로 돌려준다. 파이프 정보 출력 시 사용됨

3) gloreWorkerProcBarrierBase

Viz.-barrier pipe의 동작을 정의한 클래스로, 모든 Viz.-barrier 가시화 파이프가 상속 받아야 한다. Barrier로서 동작하게 하는 공통 루틴이 정의 되어 있다(예, Process_Barrier() 메소드). 일반적 가시화 파이프와 달리 subProcess() 메소드만 정의하여 사용할 수 있다. 일반적 Viz.-barrier pipe의 작업 흐름과 다른 barrier-style 파이프를 작성해야 하는 경우는 Process() 메소드를 정의하여 구현할 수 있다. 관련된 구현 사례는 CutPlane 파이프를 참조하면 된다.

4) gloreWorkerDataManager

GLORE를 위한 통신 파이프의 구현 중 하나로, GDM[2]과의 통신을 통해 입력 데이터를 가져와서 G-Pipeline으로 전달하는 역할을 한다. GDM에서 받은 데이터는 첫 번째 가시화 파이프가 요구하는 형태로 변환되어 전달된다.

- 기본동작
  - 첫 번째 파이프에의 setInitialDataManager()를 통해 초기화 됨.
  - 기본적으론, 파이프라인 관리자가 초기화 작업 담당 (gloreWorkerPipelineManager, 4장 참조)
  - 통신 파이프도 하나의 파이프로 runPipe() 메소드를 통해 동작이 실행 됨

- TargetElement 구조체
  - target element 정보 관리하는 구조체
  - gloreWorkerDataManager 내부적으로 TargetElement의 리스트를 관리하며, 요청에 따라 순차적으로 처리

```
typedef struct {
  int     eleID;            // element ID
  int     blockListStart;   // target block list
  int     blockListEnd;
  void*   extraPtr;         // pointer for extra information
} TargetElement;
```

- 주요 멤버 메소드 (member methods)

이름	설명
addTarget()	target data 들을 설정 함 - block list를 주지 않으면 모든 블록을 대상으로 설정 (FetchNext) - block list를 설정하면, 대상 block들을 가져옴 (FetchBlock)
addTargetTimeStep()	target time step을 설정

2) GLOVE의 데이터를 관리하는 컴포넌트

5) gloreWorkerResultCollector

GLORE를 위한 통신 파이프의 구현 중 하나로, G-Pipeline의 결과를 수집하고, 결과를 GLORE가 원하는 형태로 변환하여 돌려주는 역할을 한다. G-Pipeline의 가장 마지막 파이프로 동작하며, 기존 gloreWorker들의 동작과 같은 결과 형태로 돌려주도록 구현되어 있다.

- 주요 멤버 메소드 (member methods)

이름	설명
getResultType()	가지고 있는 output data의 type을 반환
getResultDataSet()	dataSet 형태 결과들을 반환
getResultResponse()	glvResonse 형태 결과들을 반환

마. 파이프 클래스 관계도

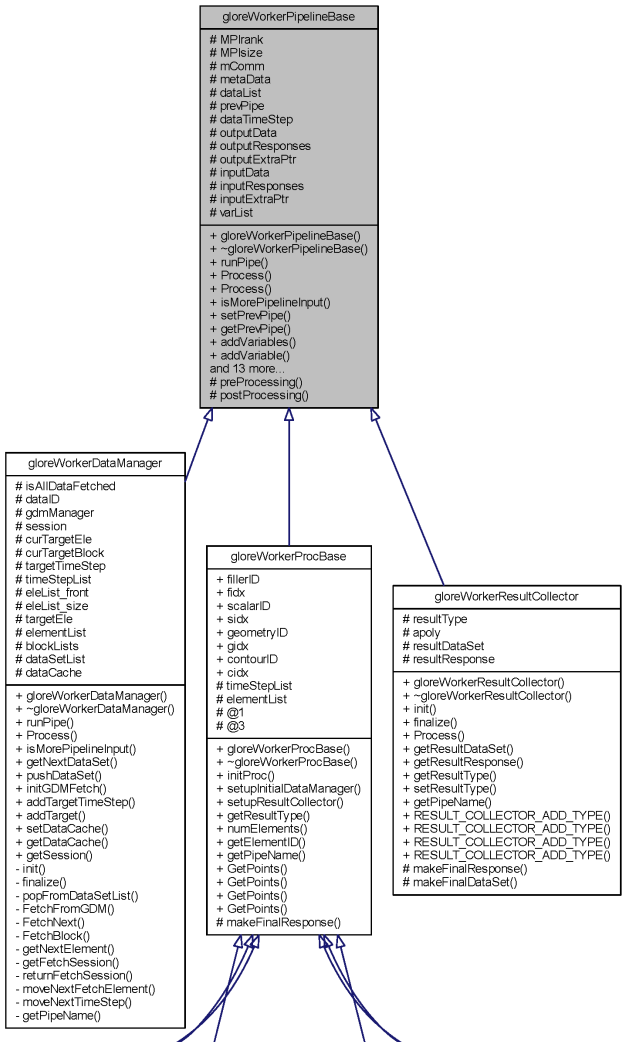


그림 5. 주요 클래스와 가시화파이프 클래스 사이의 관계도

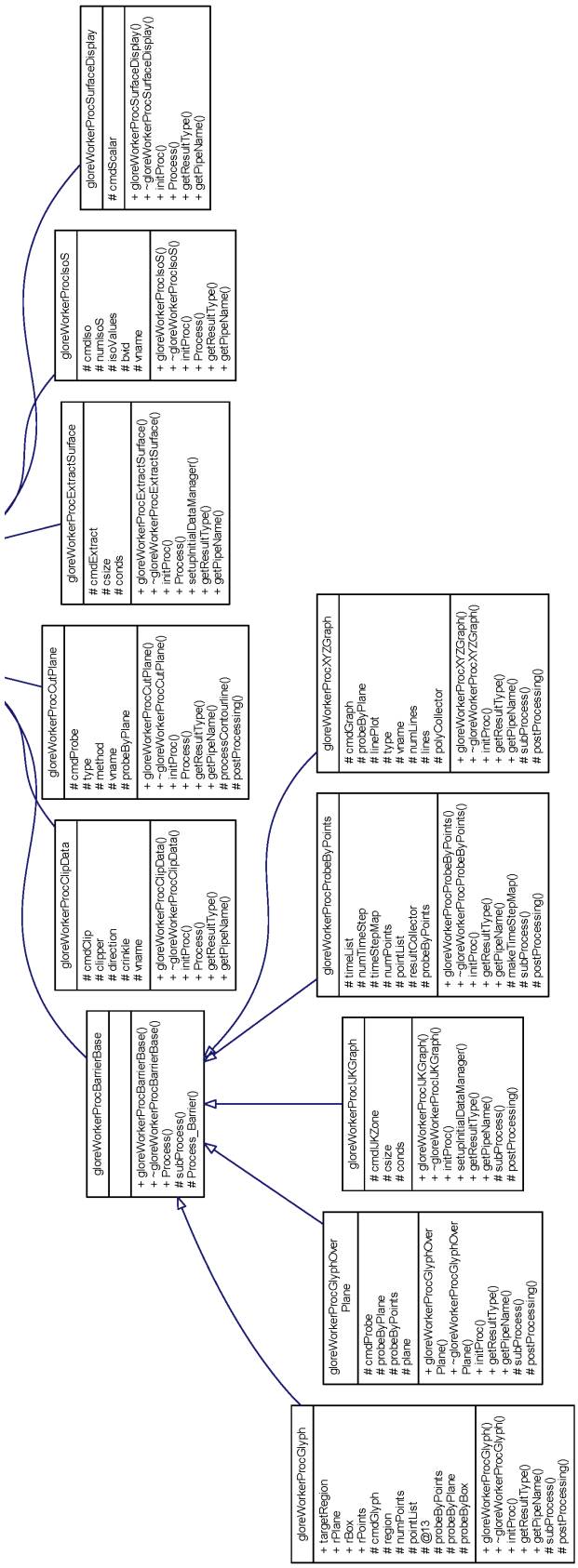


그림 6. 주요 클래스와 가시화파이프 클래스 사이의 관계도 (그림 4 에서에서 연결)

4. 파이프라인 (Pipeline)

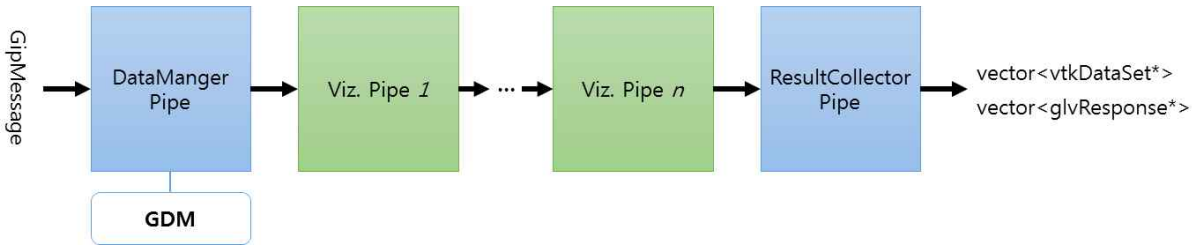


그림 7. 파이프라인의 구조 (GLOVE에 구현한 예 기준)

가. 파이프라인 생성 및 구조

G-Pipeline은 파이프라인 관리자(pipeline manager)에 의해서 생성되고 관리된다. 파이프라인 관리자는 사용자의 요청(예, GLOVE의 GipMessage)을 해석하고 최종 가시화 결과를 만들어 내기 위한 파이프라인을 실시간으로 생성한다.

가시화 파이프라인의 가장 앞쪽에는 data manager 파이프가 연결되며(예, gloreWorkerDataManager), 첫 번째 가시화 파이프(그림 7의 Viz. Pipe 1)에 종속적으로 초기화 된다. Data manager의 초기화시 고려되는 요소들은 필요한 time-step, target elements, target blocks 등이 있다. 가시화 파이프들의 연결 과정에서, 각 가시화 파이프가 요구하는 요소들을 누적해서 첫 번째 파이프로 전달한다. 그 결과 data manager는 파이프라인 실행에 필요한 모든 요소들(예, elements, blocks, variables, time-steps 등)을 알 수 있다. 가시화 파이프를 개발 할 때는, 해당 파이프가 가장 앞에 오는 경우에 수행해야할 data manager 초기화 방법을 정의해야하며, 그 정의는 setupInitialDataManager() 메소드 안에 하면된다.

가시화 파이프라인의 가장 마지막에는 result collector 파이프가 연결된다(예, gloreWorkerResultCollector). result collector의 초기화는 가시화 파이프라인의 마지막 가시화 파이프(예, 그림 7의 Viz. Pipe *n*)의 결과 형태에 종속적으로 이루어진다. 가시화 파이프를 개발 할 때는, 해당 파이프가 가장 마지막에 오는 경우에 수행해야할 result collector 초기화 방법을 정의해야하며, 그 정의는 setupResultCollector() 메소드 안에 하면된다.

나. 작업흐름 (Workflow)

G-Pipeline의 기본 처리단위는 block이다. 1) data manager는 가시화 SW로부터 하나의 block을 읽어서 파이프라인에 전달한다. 2) 가시화 파이프라인은 해당 block에 대한 일련의 가시화 연산을 수항한 후, 3) 결과를 result collector에게 전달하고 4) data manager에게 다음 block을 요청한다.

모든 block에 대한 처리가 끝나면, result collector 파이프는 최종 결과를 생성한다. 최종 결과는 getResultDataSets(), getResultResponses() 메소드를 통해 가시화 SW로 전달된다. 주어진 요청에 대한 모든 작업을 마친 후 G-Pipeline은 소멸된다.

다. 주요 클래스

1) gloreWorkerPipelineManager

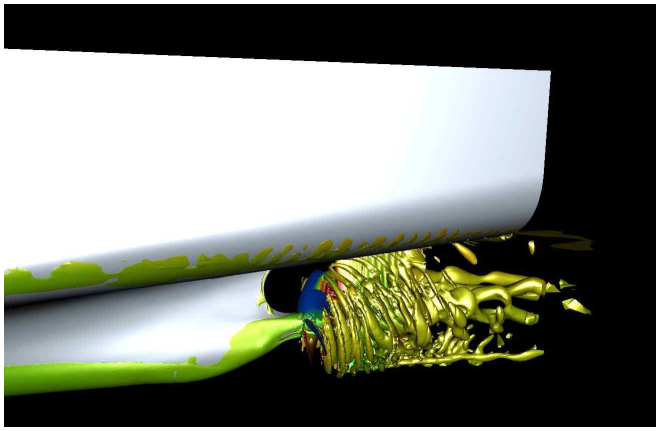
: 파이프라인을 생상하고 관리하는 클래스

- 주요 멤버 메소드 (member methods)

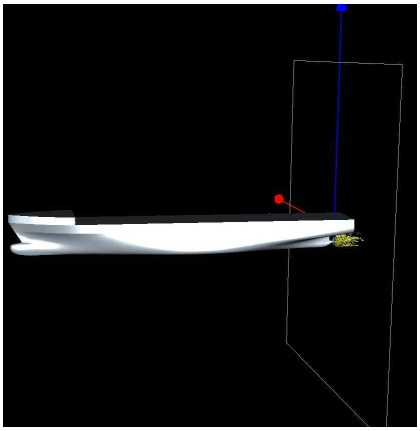
이름	설명
buildPipeline()	주어진 gipMessage를 처리하기 위한 파이프 라인 생성 : 내부적으로 data manager, result collector 파이프를 생성하고, 필요한 Viz. pipe를 생성하고 연결해서 파이프라인 생성
runPipeline()	파이프라인을 실행
getResultDataset()	파이프라인에서 생성된 vtkDataSet들을 반환 (vector)
getResultResponse()	파이프라인에서 생성된 glvResponse들을 반환 (vector)
Print()	구성된 파이프라인 정보 출력
getPipelineObject()	필요한 pipe 객체를 생성 (private method, 내부사용) <Development note> 새로운 Viz. pipe를 추가할 시, 해당 pipe 객체를 생성할 수 있도록 추가해야함. ADD_CASE_GET_PIPEOBJ 매크로 사용
getTargetMsg()	target msg를 확인 (private method, 내부사용) <Development note> 새로운 Viz. pipe를 추가할 시, 해당 pipe에 해당하는 msg를 반환하도록 추가해야 함. ADD_CASE_GET_TARGET_MSG 매크로 사용

5. 결과 이미지 (Results)

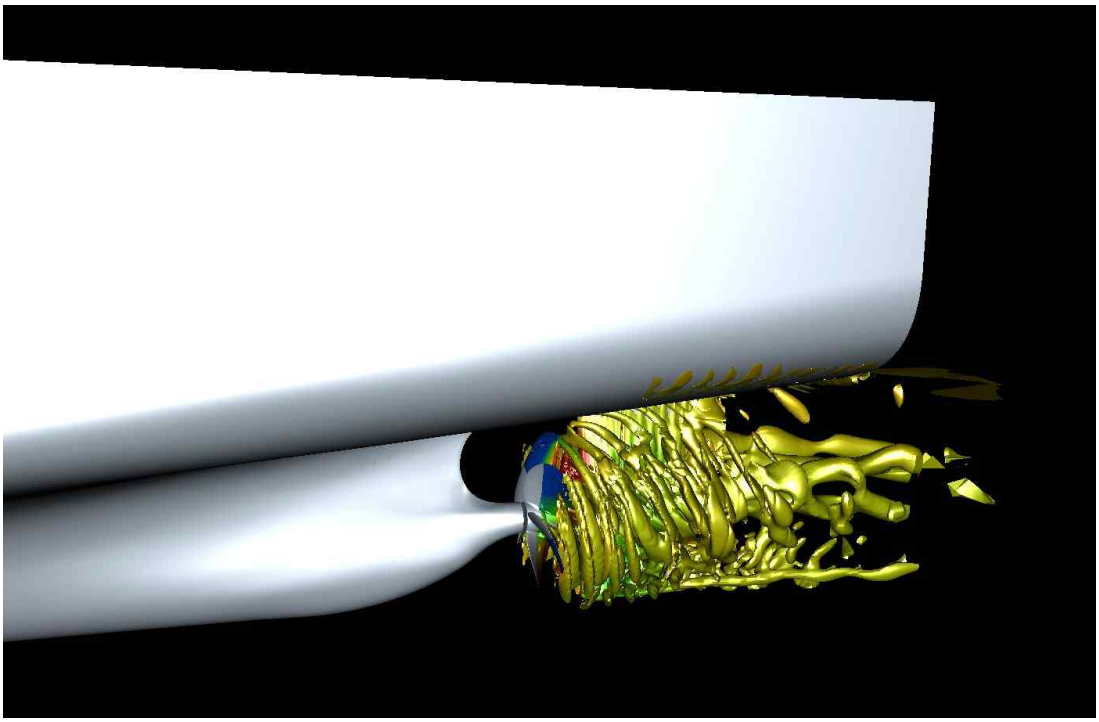
본 장은 GLOVE에 적용된 G-Pipeline을 이용한 가시화의 결과 이미지를 보여준다.



(a) Iso-surface

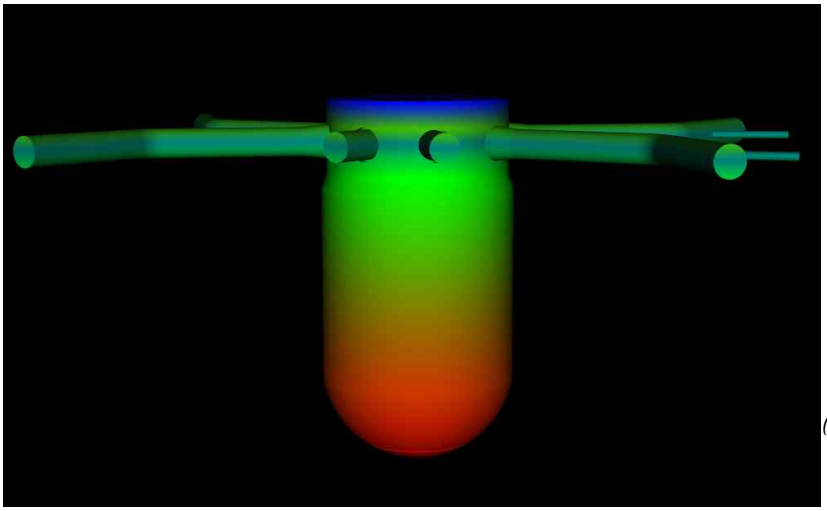


(b) Clipping surface

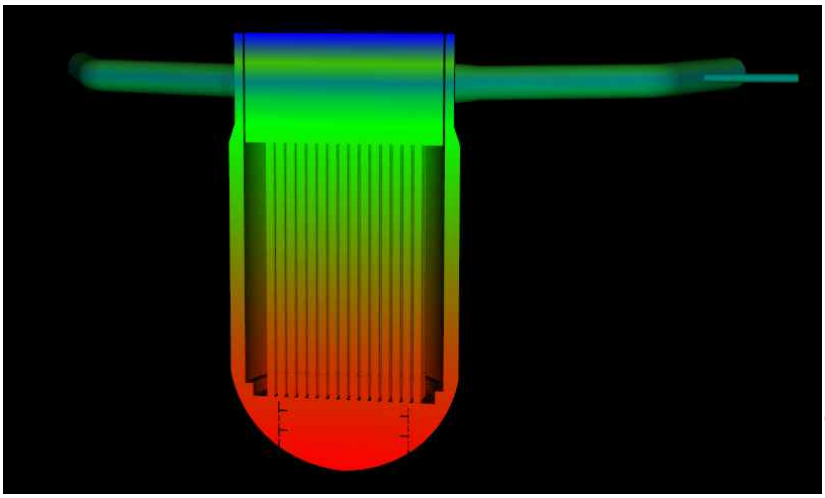


(c) Iso-surface + Clipping

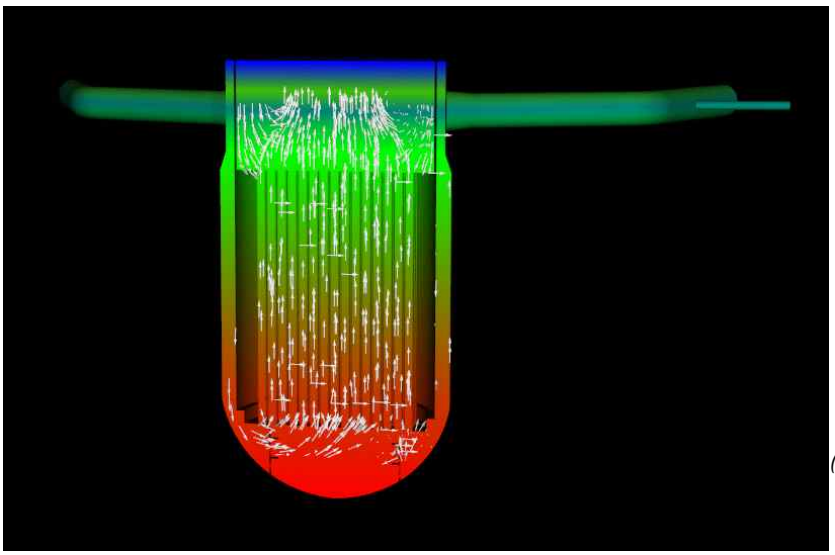
그림 8. 위 이미지들은 배의 프로펠러 실험 데이터에 대한, G-Pipeline의 가시화 결과를 보여준다. 이지지(c)를 생성하기 위해, Iso-surface와 Clipping 파이프가 연결되어 사용되었으며, 그 결과 선체 부분에 있던 노이즈가 제거된 것을 확인할 수 있다.



(a) Original data



(b) Clipping



(c) Clipping + Glyph

그림 9. 위 이미지들은 원자로 실험 데이터에 대한 가시화 결과다. 원자로 내부 유동을 볼 수 있도록, Clipping과 Glyph 파이프를 연결하여 가시화한 결과를 (c)에서 볼 수 있다.

감사의 말 (Acknowledgements)

본 기술 문서는 2017년 정부(미래창조과학부)의 재원으로 국가과학기술연구회 민군융합기술연구사업(No. CMP-17-03-KISTI) 과제의 지원을 받아 작성된 문서임.