

The MITK/ITK Pipeline



Concept, Usage, Pitfalls

Overview

- What it all is about
- How to use Pipeline Objects
- How to implement a pipeline object

Why Pipeline Execution?

- Intuitive Object representation of data flows
- Standardised Interface for "active" Objects
- Prevents Inconsistencies
- Easy handling by triggering long pipelines with one Function Call
- Easy ways to parallelise for Multicores

What does Update() do?

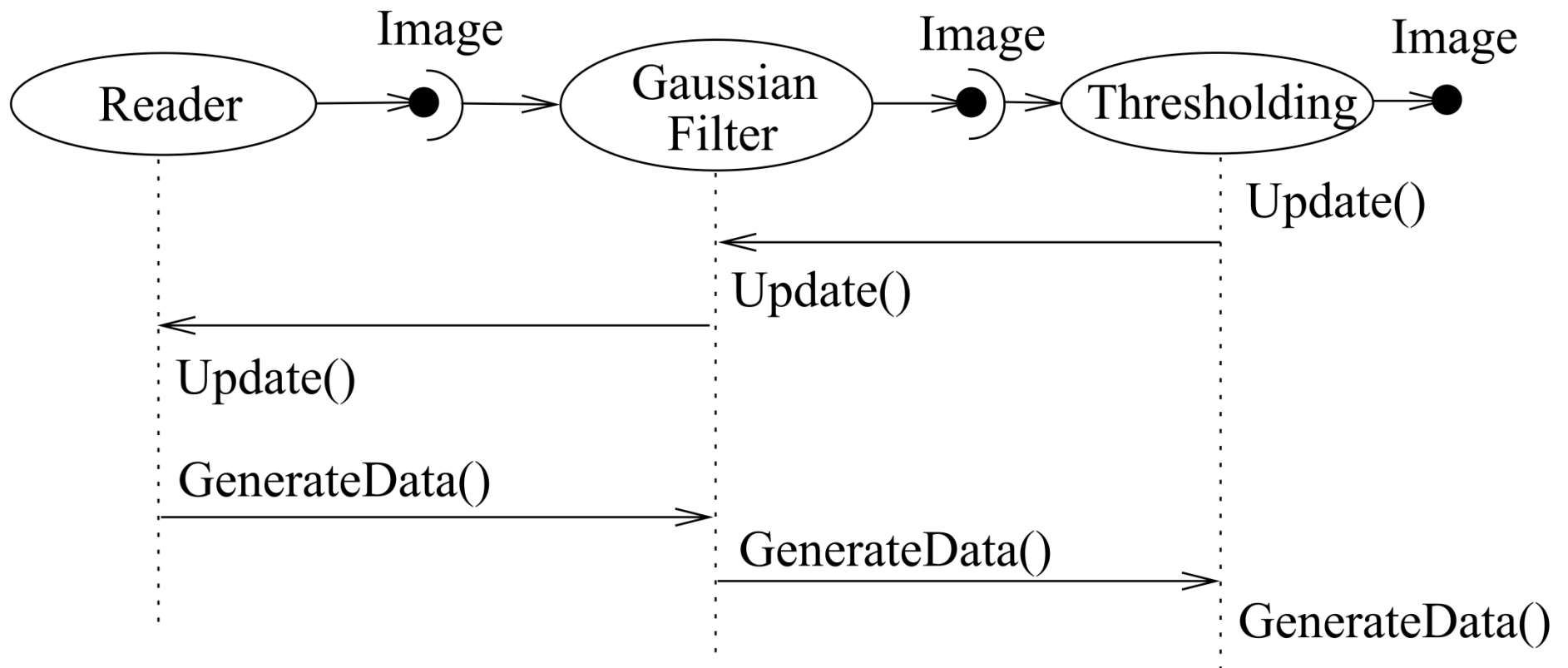


Figure 13.3: Sequence of the Data Pipeline updating mechanism

Two Kinds of Objects

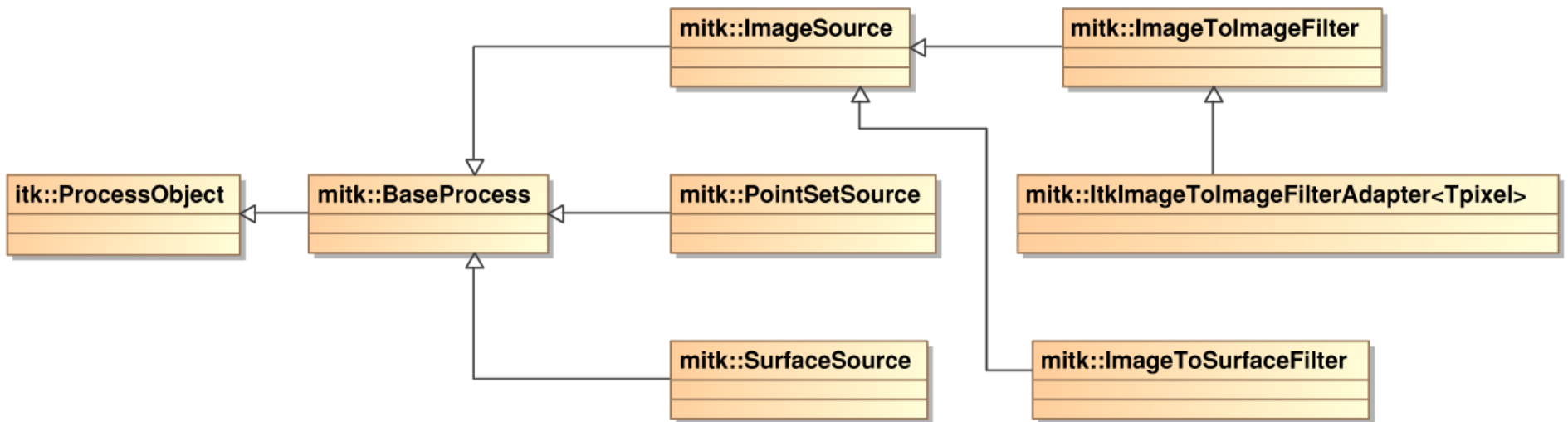
Mitk::BaseData

- Stores Data
- Knows its source
- Can be Input of a BaseProcess
- is child of itk::DataObject

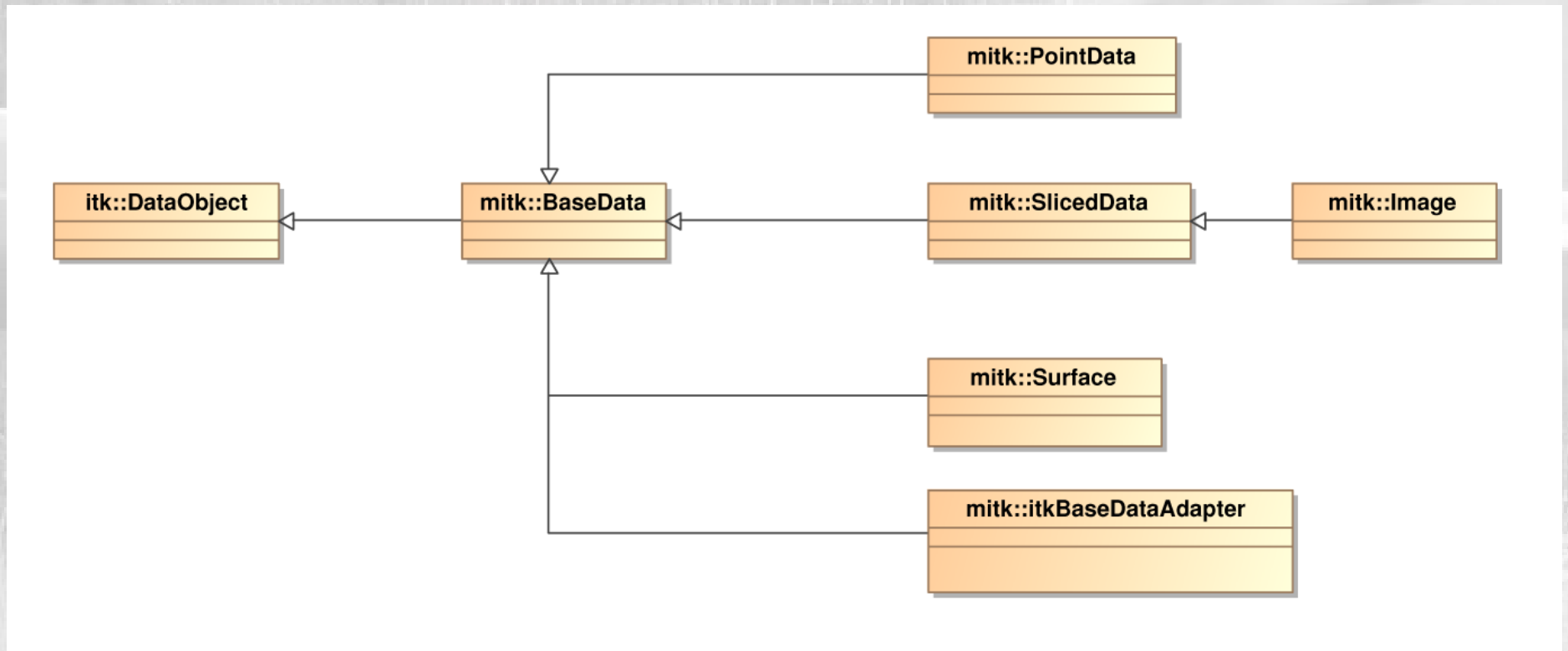
Mitk::BaseProcess

- Processes Data
- Has an Output
- Can have an Input
- Is child of itk::ProcessObject

mitk::BaseProcess



mitk::BaseData



Roadmap to a mitk::BaseProcess

- Find a base class (i.e. mitk::ImageSource)
- Implement the following methods:
 - GenerateData()
 - MakeOutput()
 - GetOutput()/SetInput()

Optional Methods to reimplement

- `GenerateInputRequestedRegion()`
- `GenerateOutputInformation()`
 - Vital if `InputType != OutputType`
- `AllocateOutputs()`

Handy Tools

- `itk::DataObject::DisconnectPipeline()`
- `itk::DataObject::Modified()`
- `itk::ProcessObject::GraftNthOutput(DObj)`

The background image is a faded, grayscale photograph of an industrial facility, likely a power plant, situated in a natural setting. A large, dark pipe runs horizontally across the middle of the frame. In the background, there are several tall, thin chimneys or smokestacks. The entire scene is set against a backdrop of a dense forest of tall, thin trees. The overall tone is muted and professional.

The End

Thank you for your attention.