Abstracted Workflow Framework (Catena)

Adam J. Rossi

Chester F. Carlson Center for Imaging Science
Rochester Institute of Technology

November 9, 2012
Western New York Image Processing Workshop

Outline

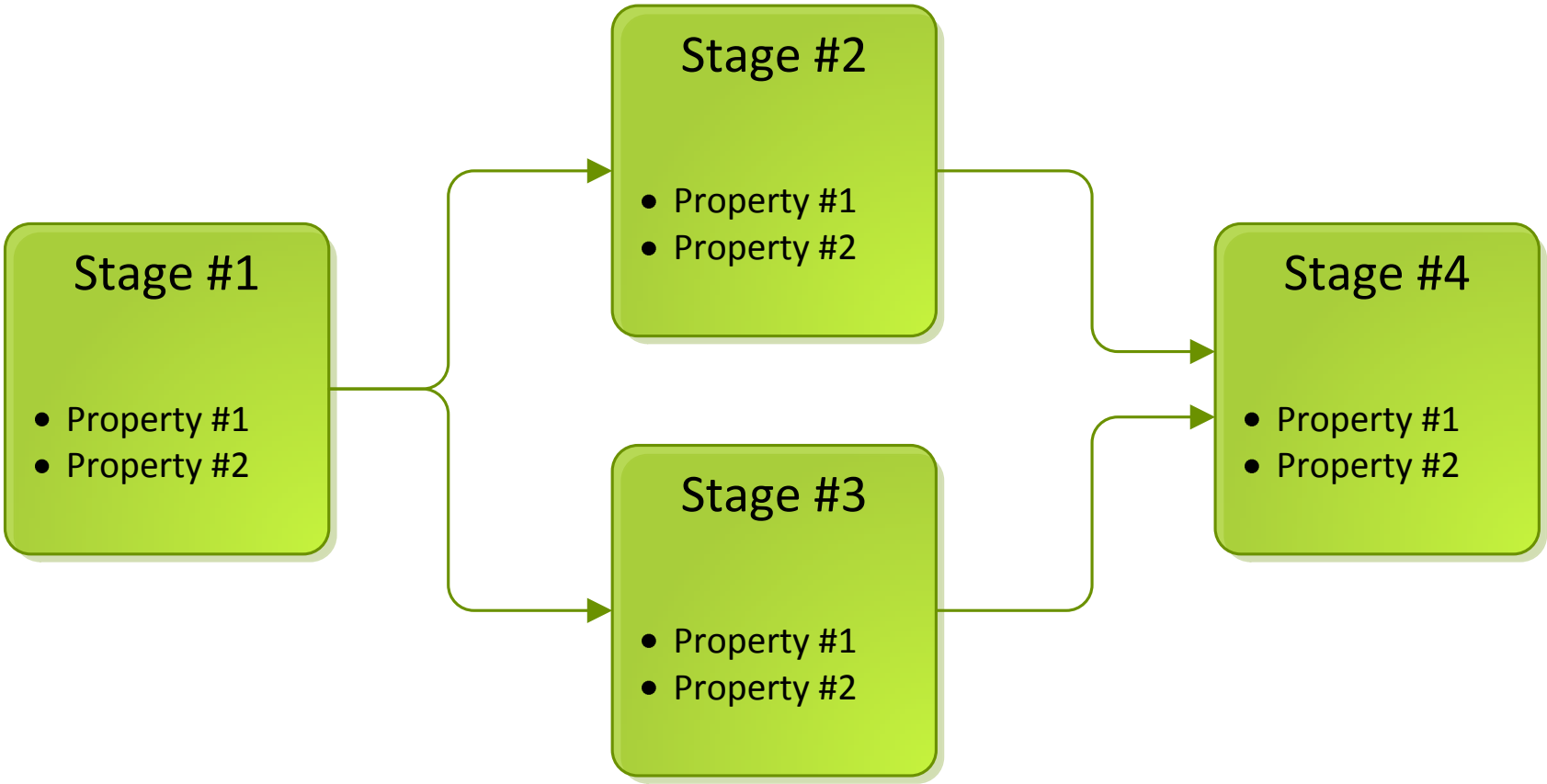
Problem Statement

- Developers write excellent software
  - Libraries, Applications, Scripts
- Work typically lost
  - Inaccessible
  - Documentation lacking
  - Application specific implementation
  - Easier for future users to start from scratch
- Multiple Implementations
  - Developers re-inventing each time
  - Incompatible tools
  - Unsupported software that eventually dies

Goals

- Eliminate re-implementation, maximize re-use
- Promote retention and ability to leverage software
- Componentize functionality into stages
- Quickly define new workflows / chains
- Ability to interchange stages with common interfaces
- Offer users of varying levels a stage library to build chains
  - Programmatically
  - Graphically

Chain Abstraction



Stage Developer Tasks

- Implement stage according to base class
- Default parameters to indicate types
- Documentation of stage, properties, and interfaces
- Interface description (Get Input/Output Interface)
- Implement execution method

Framework Benefits

- Stage / chain model, enforces
  - Self-documentation
  - Property and interface definition
  - Stage connection compatibility (based on interface)
  - Interface consistency and compatibility
  - Implementation correctness
- Caching
- Renders chains efficiently
- Poised for extensibility / scalability
- Build library of stages for re-use

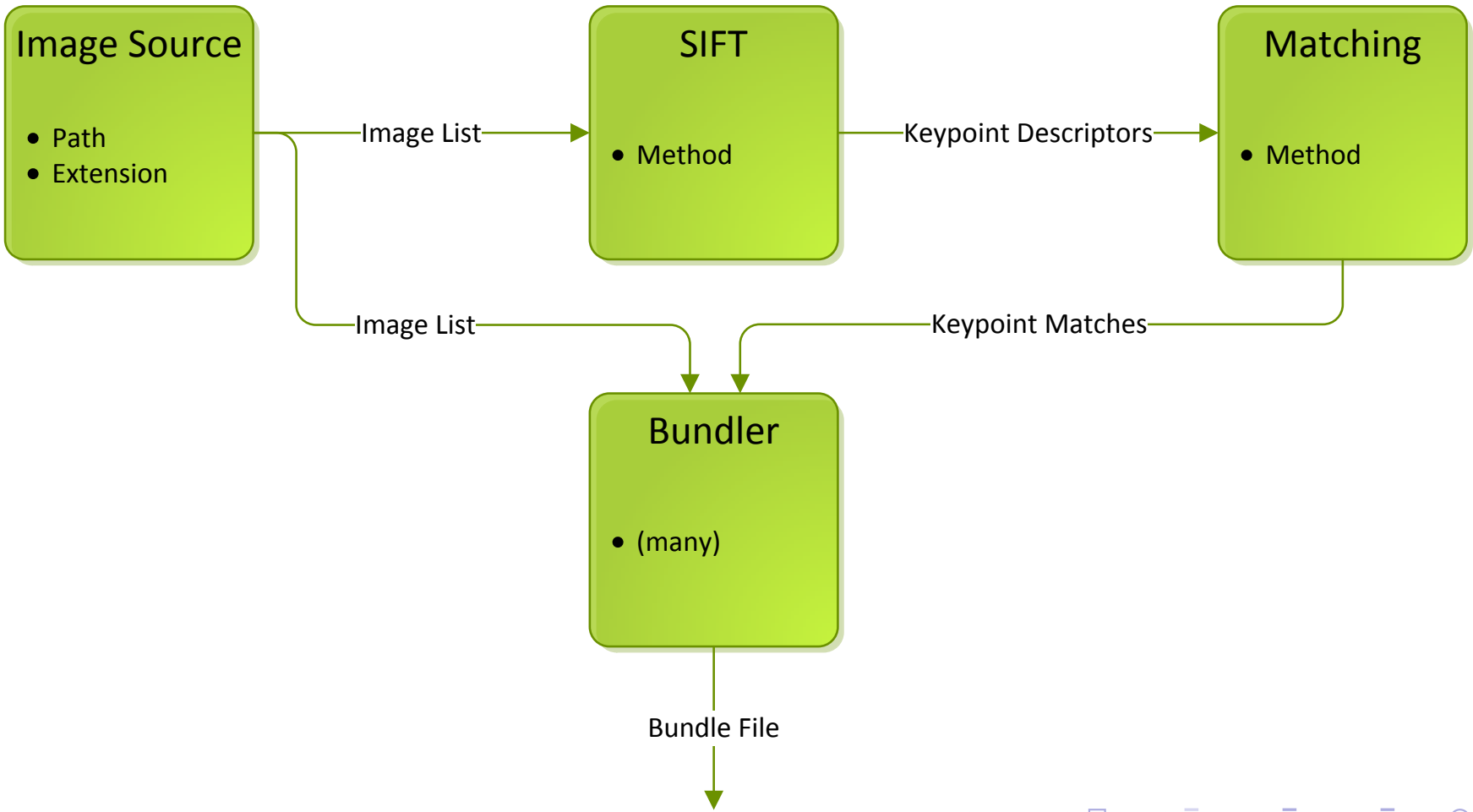
Chain Code

```
import Chain # Chain must be imported first, requirement of registry
import Sources, FeatureExtraction, FeatureMatch, BundleAdjustment

# build chain
imageSource = Sources.ImageSource("/data/images", "jpg")
sift = FeatureExtraction.Sift(imageSource, False, "SiftGPU")
keyMatch = FeatureMatch.KeyMatch(sift, False, "KeyMatchGPU")
bundler = BundleAdjustment.Bundler([keyMatch, imageSource])

# render chain
print Chain.Render(bundler)

# persist chain
Chain.StageRegistry.Save("chain.dat")
```

Restore Chain Code

```
# load chain
headStages, tailStages = \
    Chain.StageRegistry.Load("chain.dat")

# render tail stage (bundler)
print Chain.Render(tailStages[0])
```

Bundler Chain



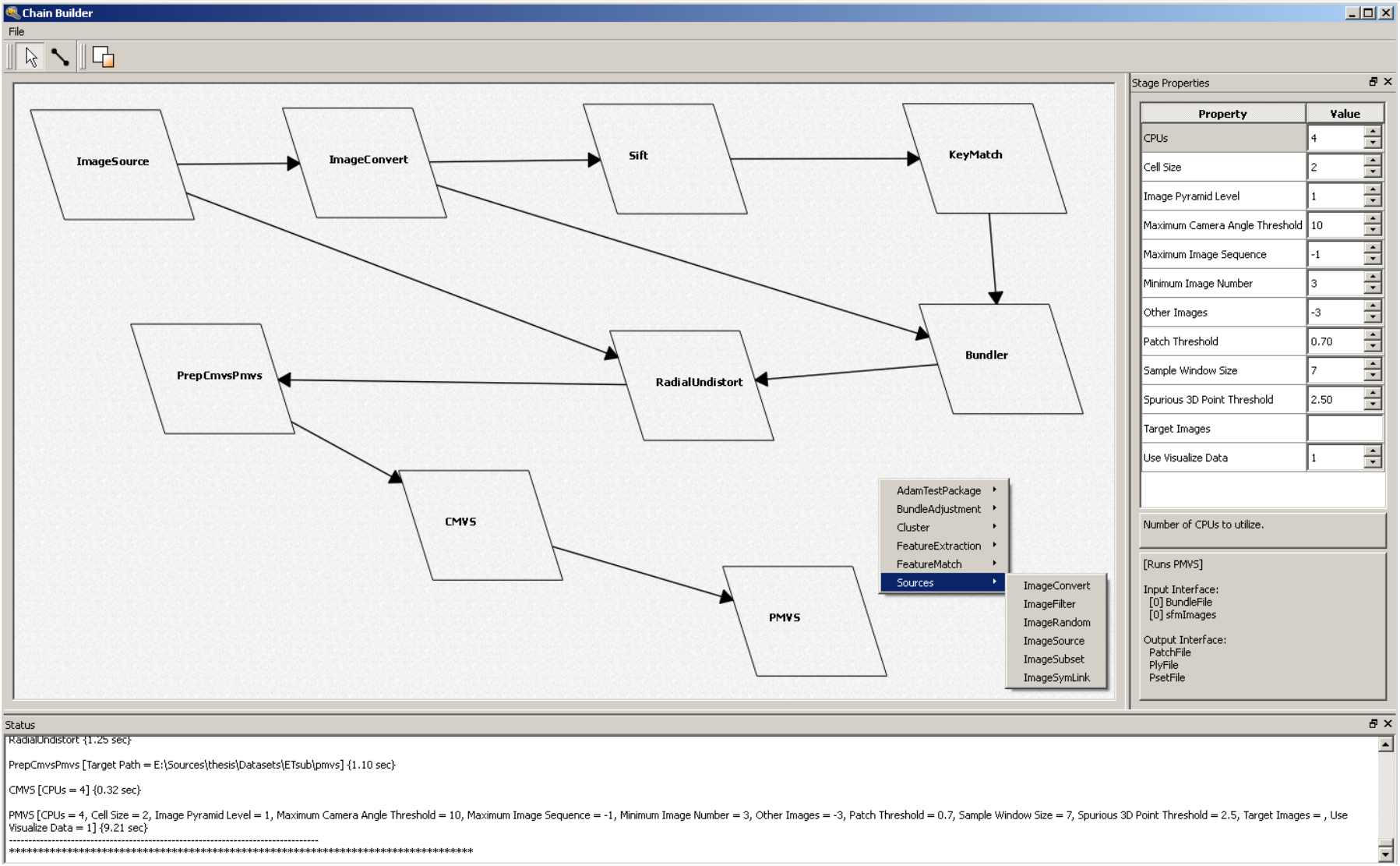
Stage Code (1/2)

```
import Chain

class StageExample(Chain.StageBase):

    def __init__(self, inputStages = None,
                 property1 = False, property2 = "",
                 property3 = 1,     property4 = 2.0):

        Chain.StageBase.__init__(self,
                                 inputStages,
                                 "Example Catena Stage",
                                 {"Property 1" : "Boolean property",
                                  "Property 2" : "String property",
                                  "Property 3" : "Integer property",
                                  "Property 4" : "Float property"})

        self._properties["Property 1"] = property1
        self._properties["Property 2"] = property2
        self._properties["Property 3"] = property3
        self._properties["Property 4"] = property4
```

Stage Code (2/2)

```
def GetInputInterface(self):
    return {"inputParameter" : (0, InputClass)}

def GetOutputInterface(self):
    return {"outputParameter" : OutputClass}

def Execute(self):

    inputParameter = self.GetInputStageValue(0, "inputParameter")

    self.StartProcess()

    # do work...

    outputParameter = OutputClass()

    self.SetOutputValue("outputParameter", outputParameter)
```

Chain Builder GUI



SfM Chain

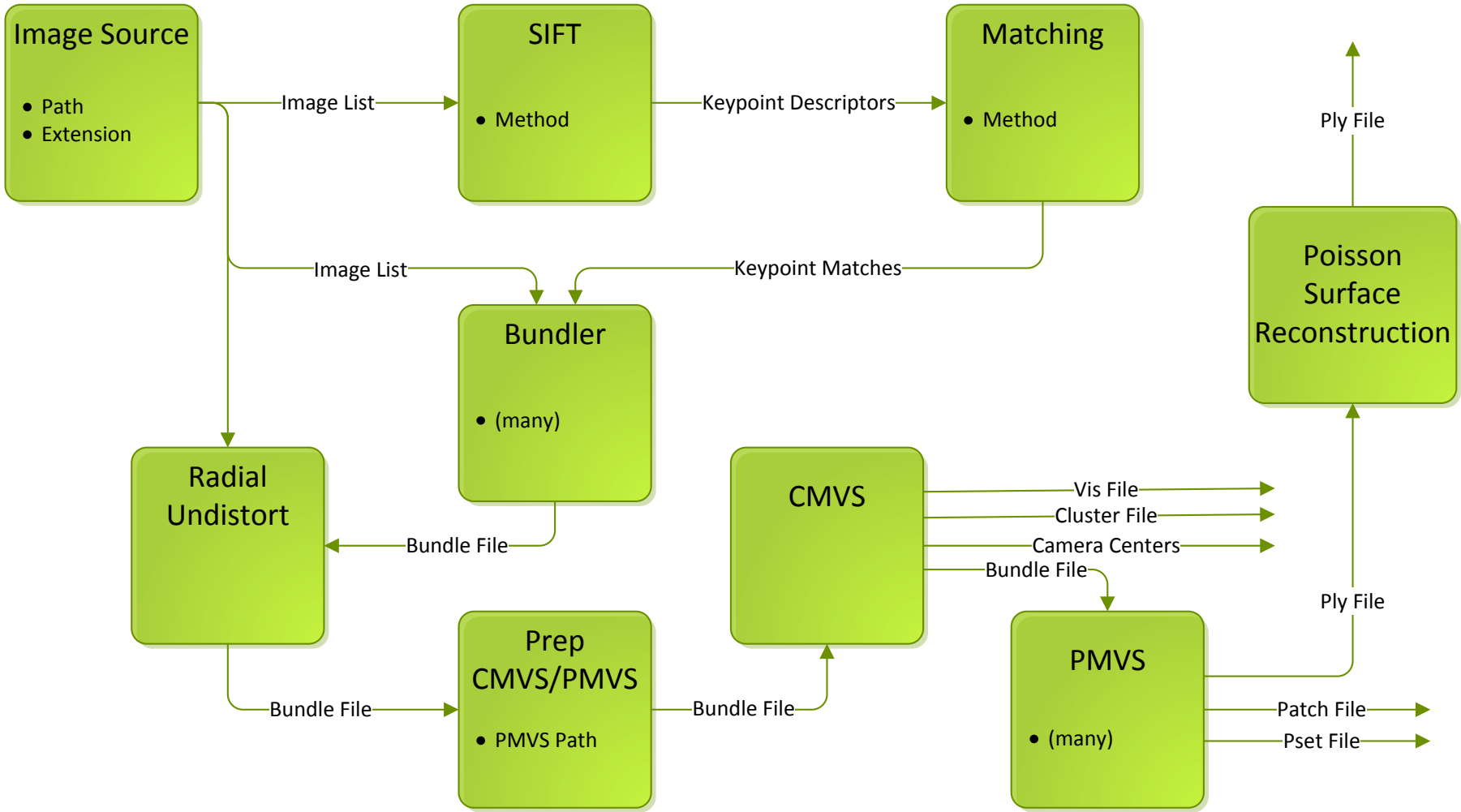
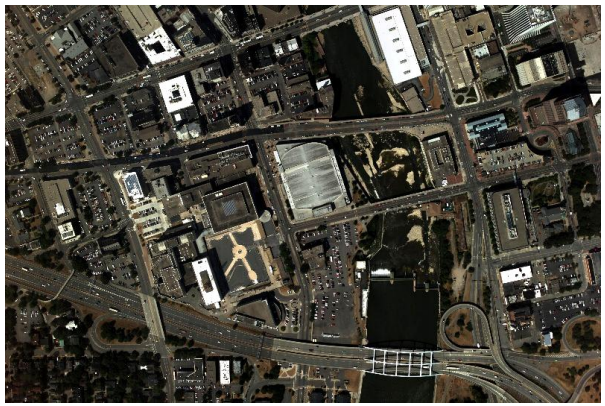
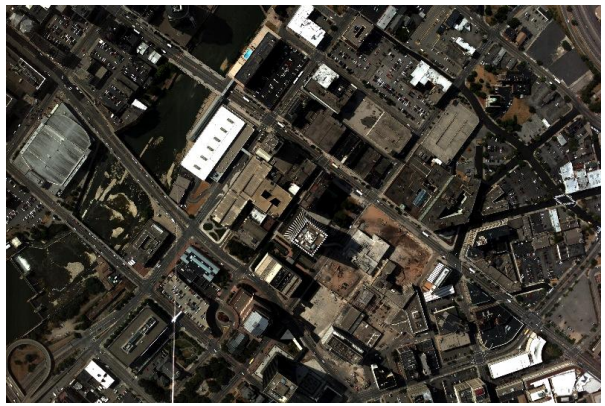


Image Source
Path
Extension
Image List
SIFT
Method
Keypoint Descriptors
Matching
Method
Ply File
Poisson Surface Reconstruction
Image List
Keypoint Matches
Bundler
(many)
Radial Undistort
Bundle File
CMVS
Vis File
Cluster File
Camera Centers
Bundle File
Ply File
Bundle File
Prep CMVS/PMVS
PMVS Path
Bundle File
PMVS
(many)
Patch File
Pset File

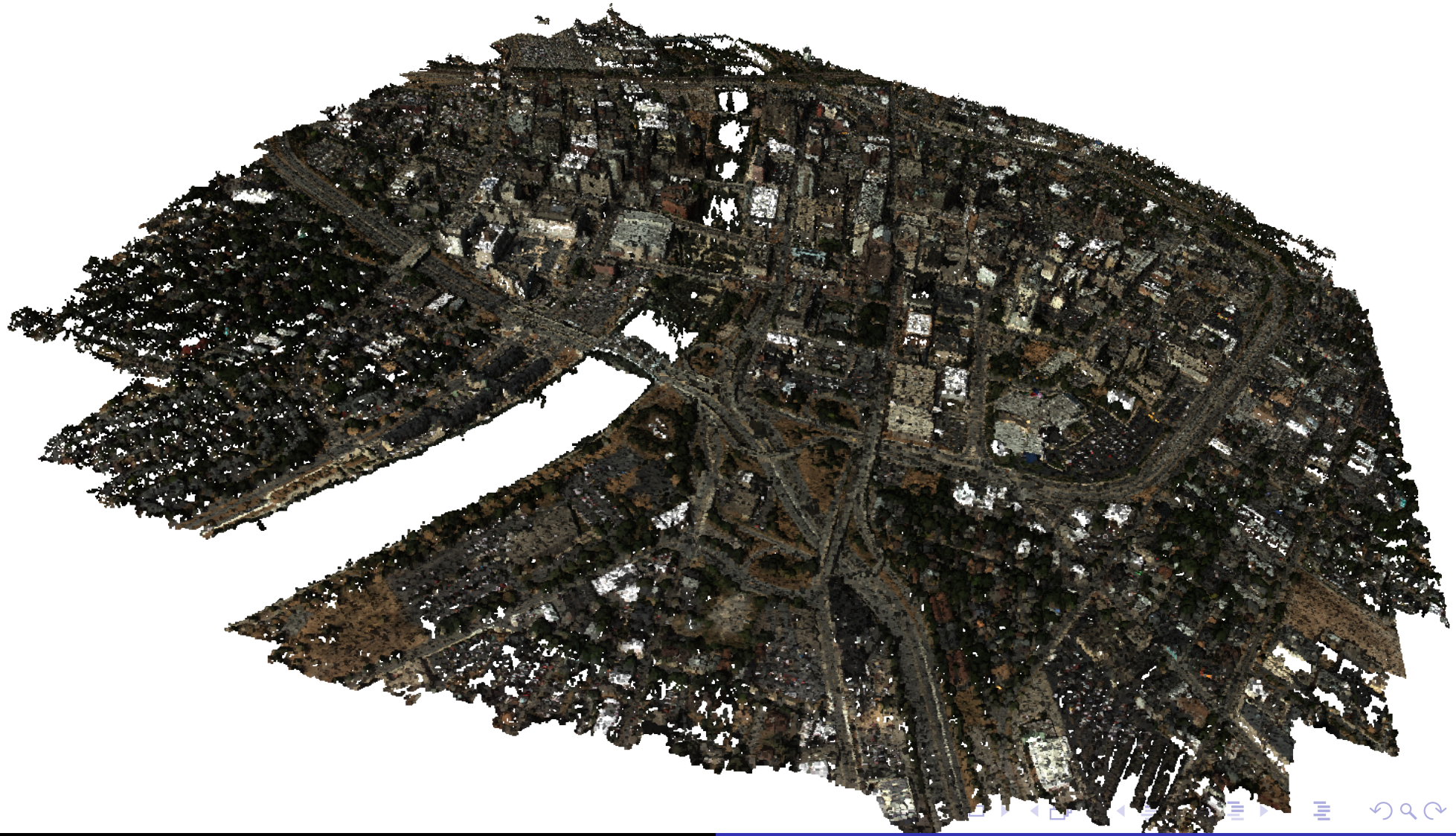
WASP Images



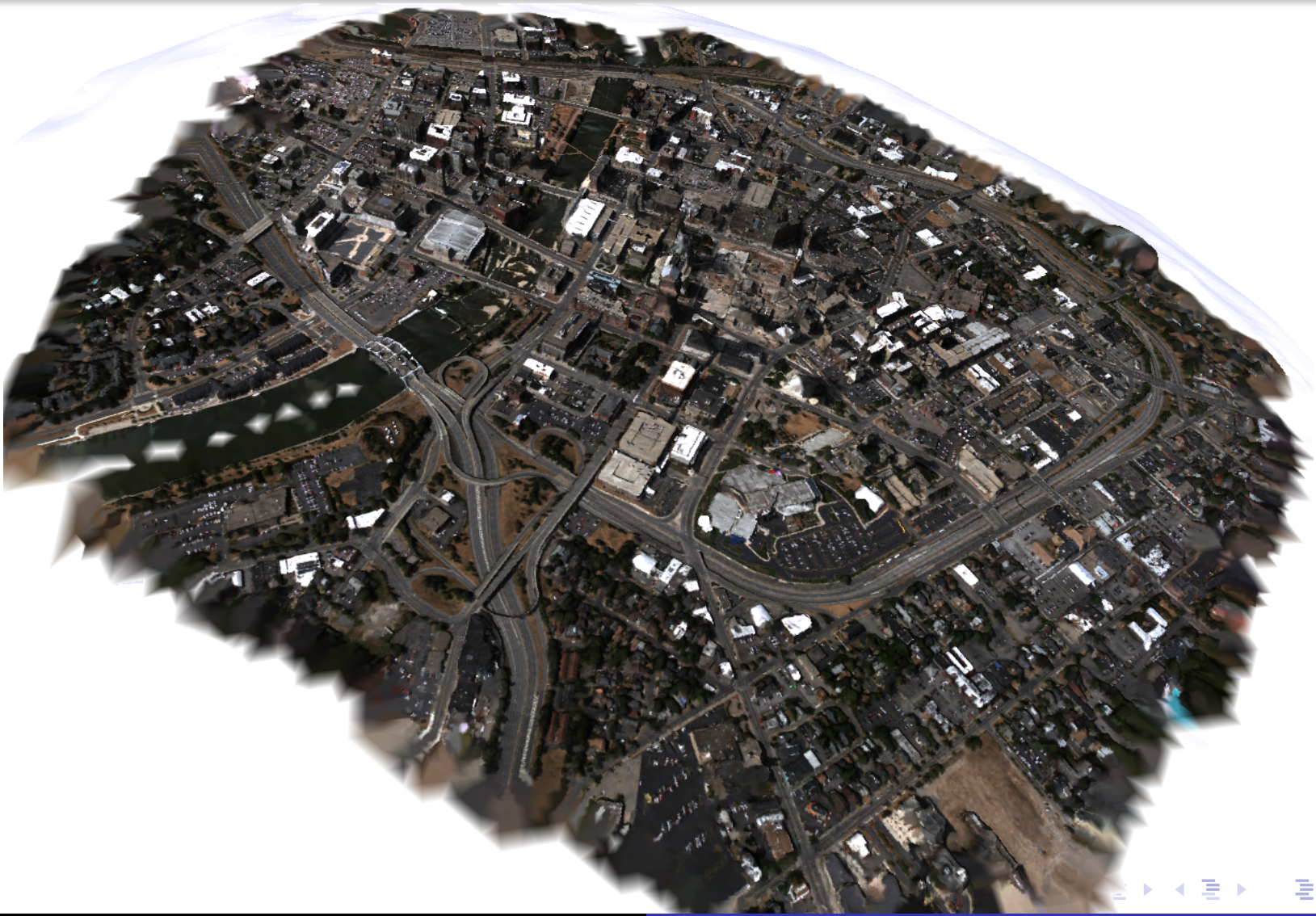





WASP Point Cloud

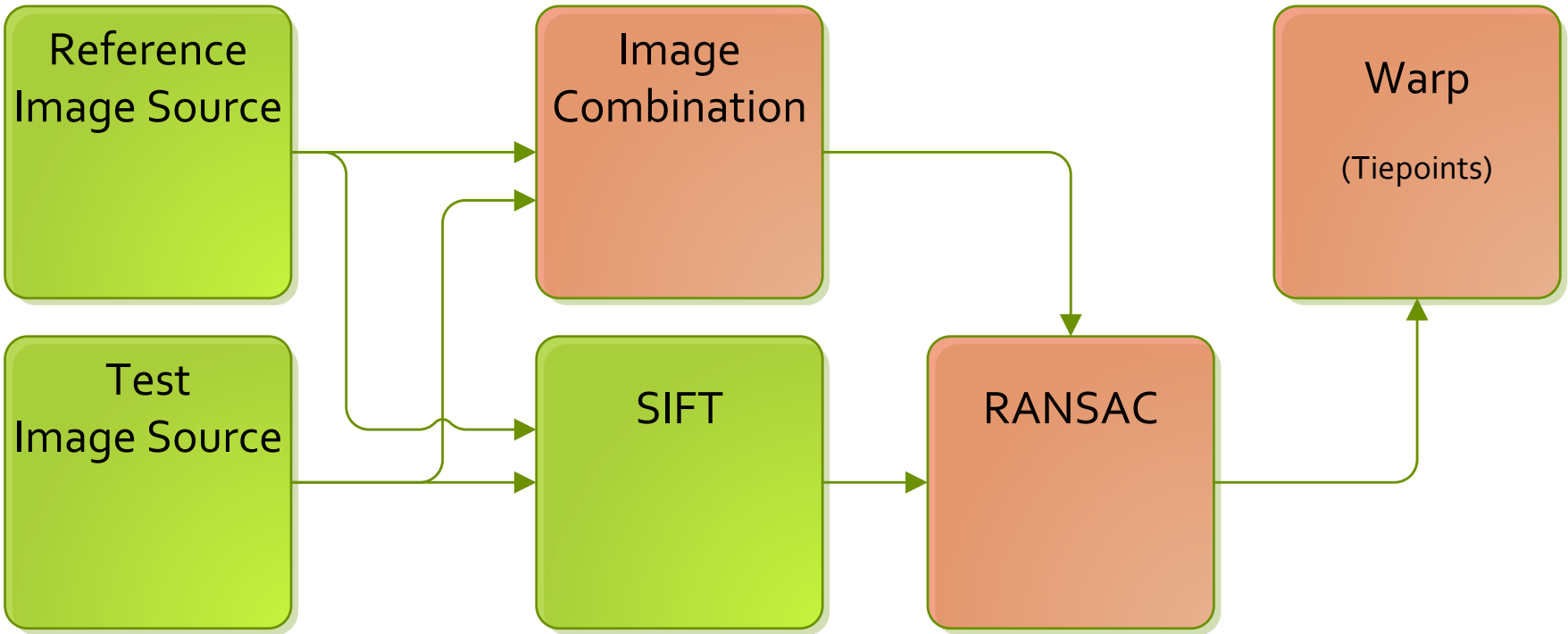


WASP Point Cloud (Poisson Surface Reconstruction)



WASP Fly-Through

Registration Chain (SIFT / RANSAC)



Summary

Generalized workflow framework implemented

- Automatic documentation
- Straight-forward implementation of stages
- Accessible to all user levels
  - Chain Builders
  - Algorithm Implementers
  - Integrators
  - Scientists
- Flexible and extensible (HPC, other domains)
- SfM and registration stages/chains

Acknowledgements

- Dr. Harvey Rhody (RIT CIS)
- Dr. Carl Salvaggio (RIT CIS)
- Dr. Derek Walvoord (ITT Exelis)
- Bernie Brower (ITT Exelis)

Thank You

Thank you for your attendance!