SimpleITK Survey:

Preliminary Results

General Statistics

- 253* participants

- Non-C++ users:
  - 21.34% (54 participants) have never used or don't feel comfortable using C++
    - Specific distribution: 16 + 38
  - 51% (28) of non-C++ users have used ITK
  - 26 have never used ITK

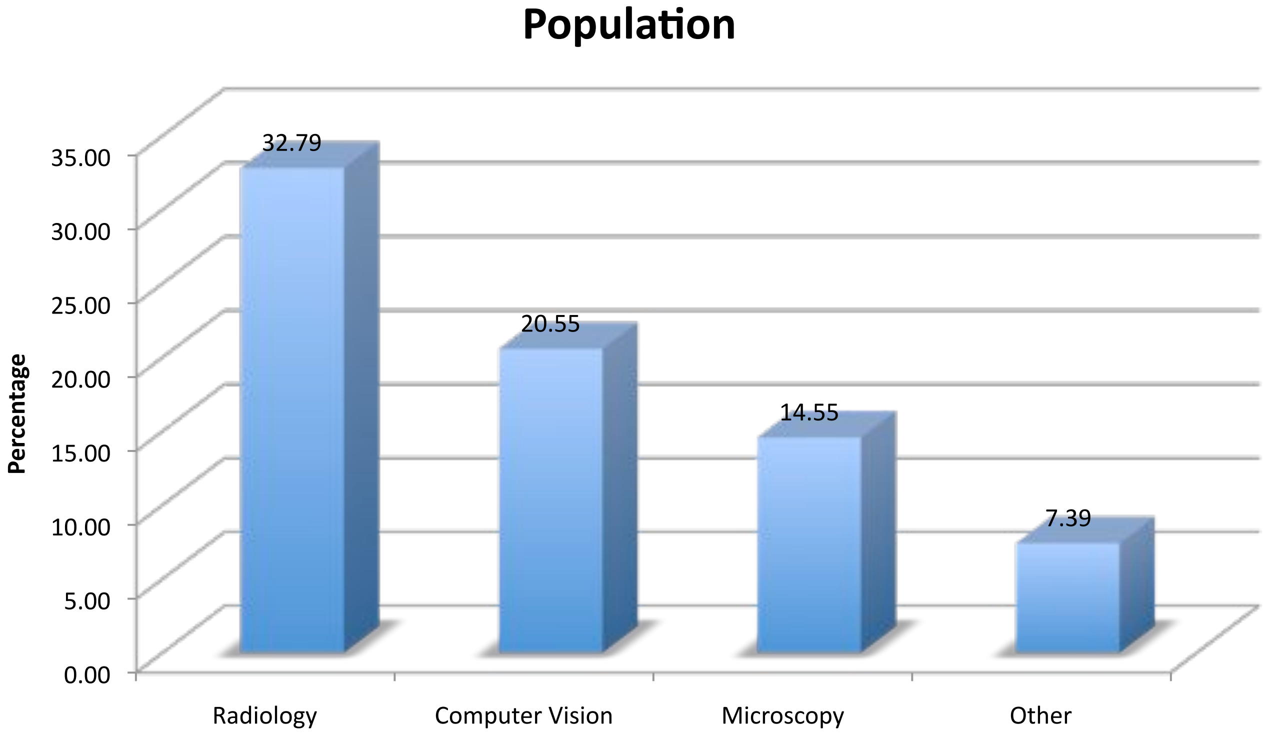
- Non-ITK users:
  - 17.78% (45 participants) have never used ITK
  - 207 have used ITK
  - 44.44% (20) of non-ITK users are C++ users

*as of 11/6/10

Survey: Target Audience

1. ITK user list
2. VTK dev/user list
3. Orfeo toolbox
4. Slicer
5. R imaging group
6. NAMIC
7. UNC Medical Imaging
8. UNC Computer Vision
9. SCI – U. Utah
10. Image World
11. V3d-workgroup
12. SIIM workshop
13. Yale (R. Papademetris)
14. JHU (R. Taylor et al.)
15. Image Processing (CC/NIH)
16. C++ users on LinkedIn
17. Medical Informatics on LinkedIn
18. etc….

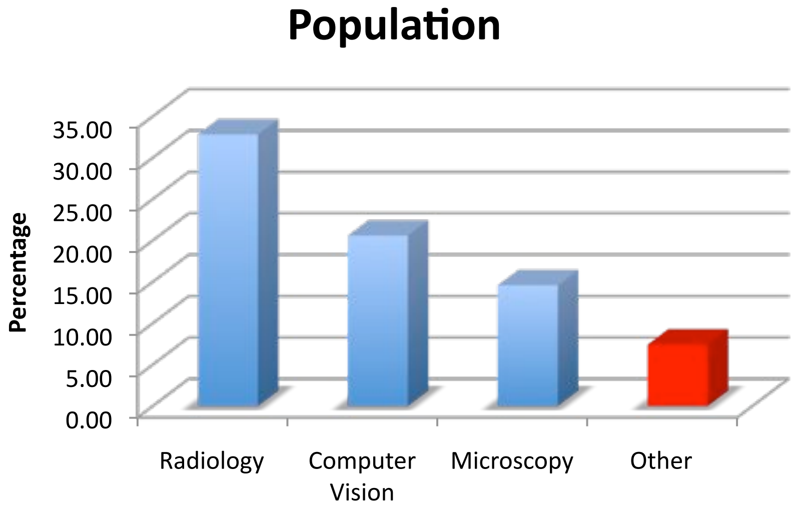
Population



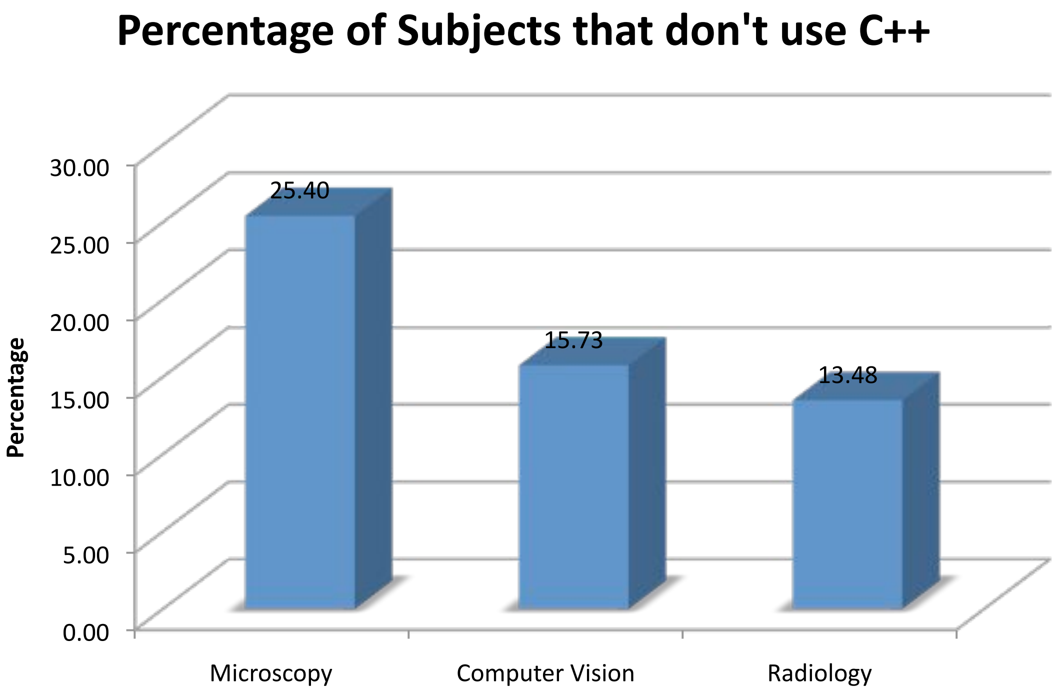
24.7% of the participants belong to academia

Population: Other

- Neuroimaging
- Ophthalmology
- Remote Sensing
- Statistics
- Microstructure modeling
- fMRI, DWI
- Historical paintings & preservation
- Radiotherapy
- Oceanography / Meteorology
- etc…



Population that don't use C++



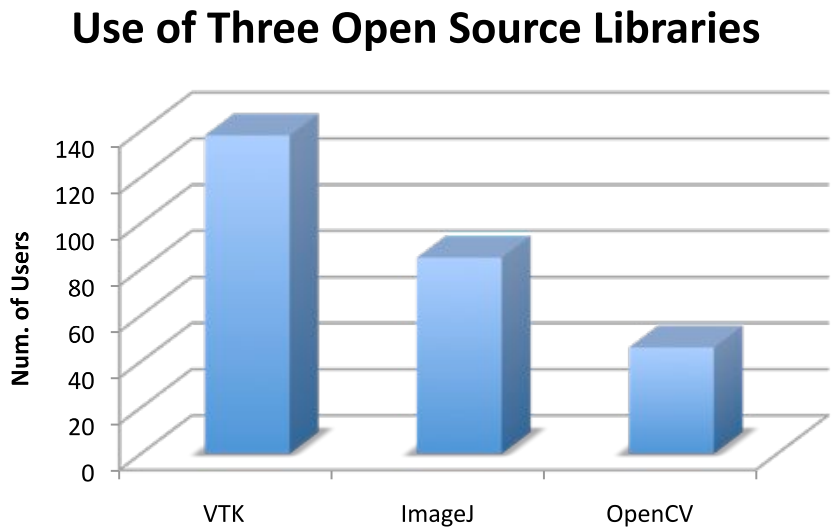
Note:

• In this survey, most of the people that don't use C++ are from microscopy

What other software / libraries?

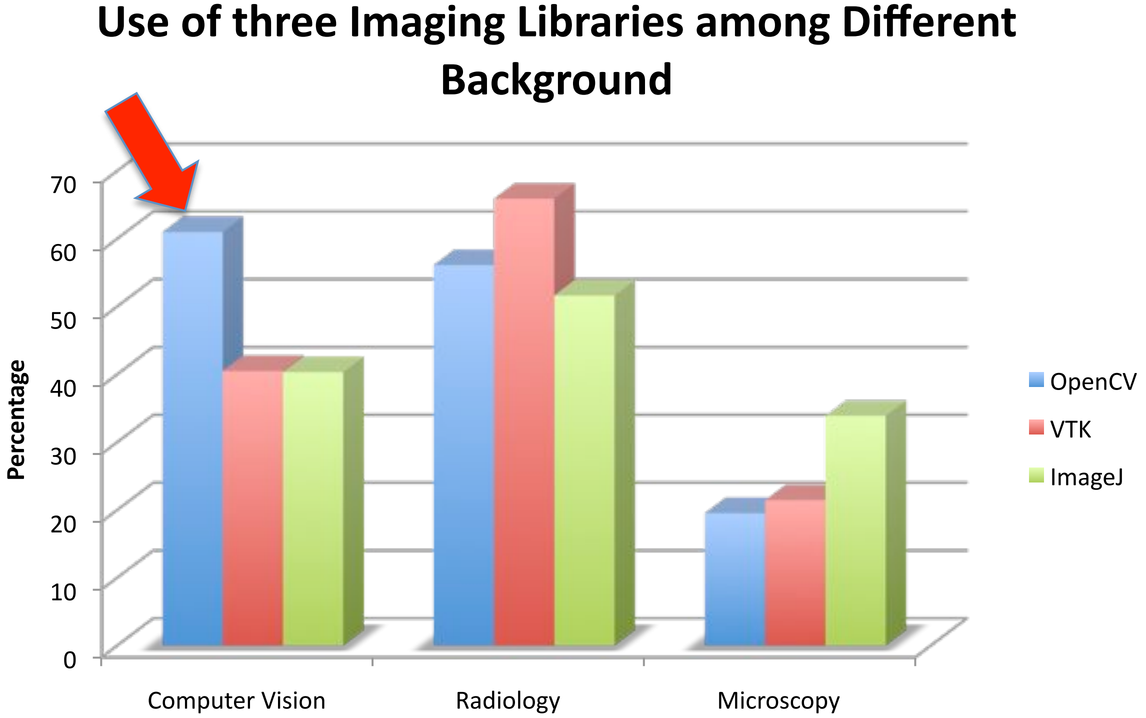
1. 3D Slicer
2. Analyze
3. Cimage
4. convert3D
5. MIPAV
6. DCMTK
7. ENVI
8. Orfeo Toolbox
9. FSL
10. Camino
11. ITK-snap
12. GDCM
13. Gimp
14. Fiji
15. Haskell
16. Irfanview
17. Vxl
18. Matlab
19. Octave
20. Qt
21. OsiriX
22. Simpleware ScanIP
23. Mathematica
24. GPU shaders
25. Mia.sf.net
26. Mipav
27. Etdips
28. Mitk
29. Vala
30. ~~My own code~~
31. Hugin
32. IDL
33. ~~In-house Tools~~
34. OpenSceneGraph
35. OTB
36. ParaView
37. PIL
38. Mayavi
39. R
40. Seg3D
41. V3D
42. VisualC++
43. ANTS
44. Numpy
45. Scipy
46. ImageMagick
47. Clojure
48. Javascript
49. VTK
50. ImageJ
51. OpenCV

What libraries are used to process images (besides ITK)?

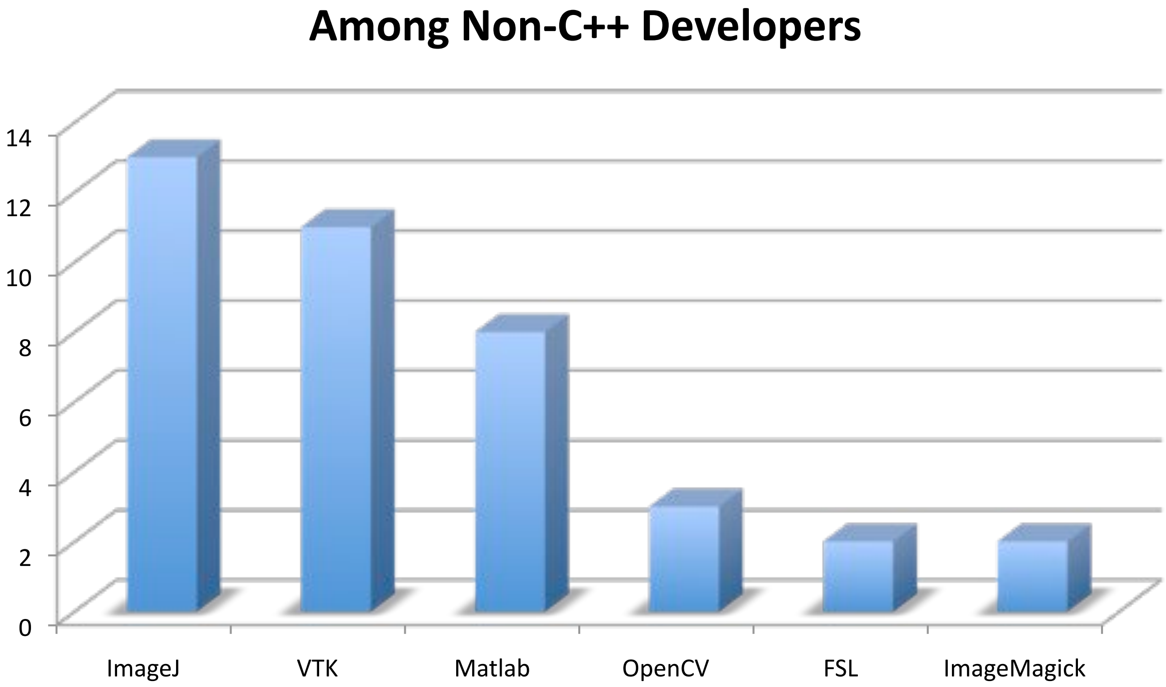

Use of Three Open Source Libraries

Q:

- Is ImageJ the primary software in microscopy?


Use of three Imaging Libraries among Different Background

Libraries used among non-C++ developers



Note that all of the top libraries / applications among non-C++ developers provide processing and visualization.

ITK

Functional vs. Object Oriented

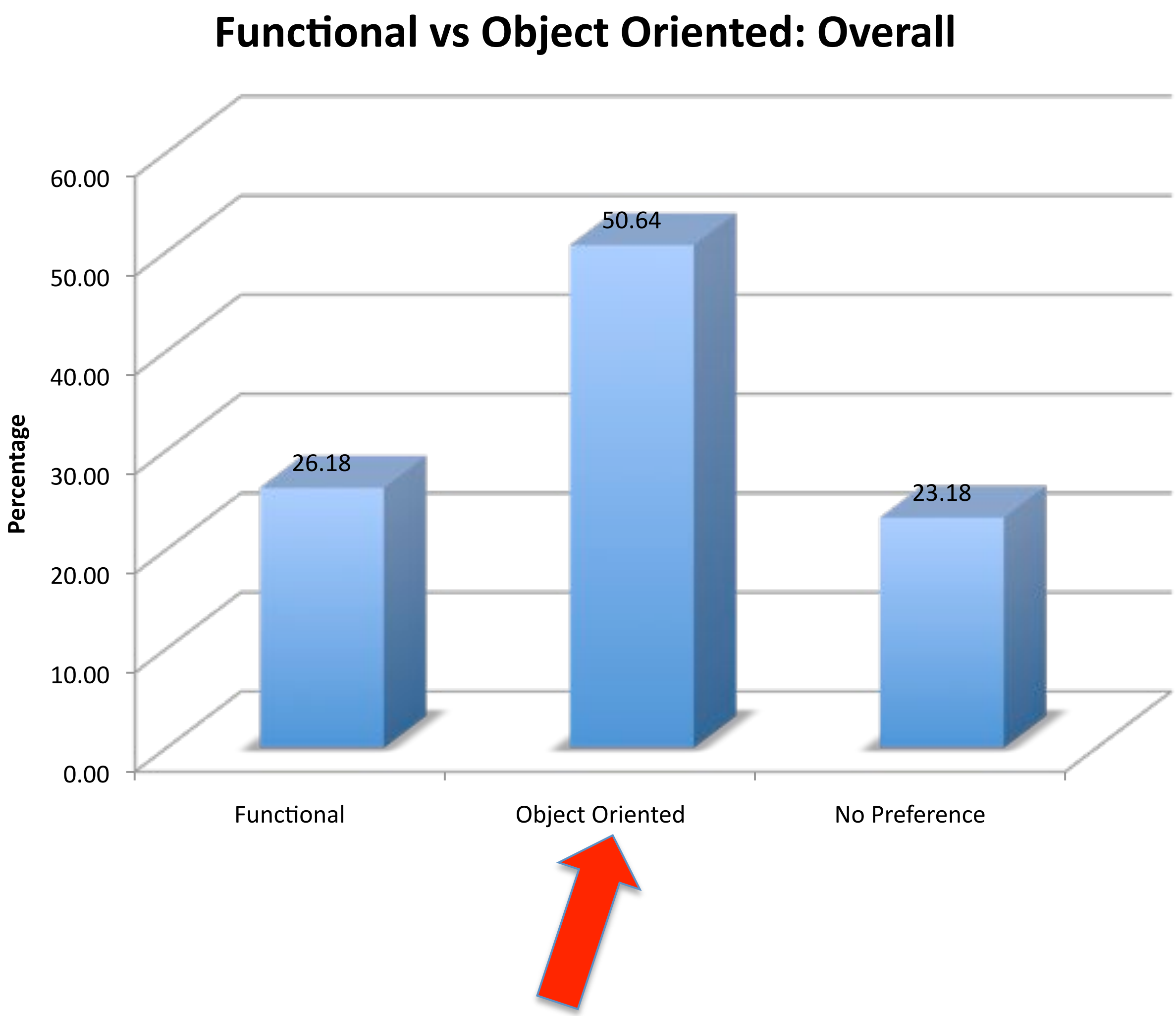
Example 1: Functional

```
Image out = GaussianFilter::Blur(inImage, 2);
```

Example 2: Object-oriented

```
GaussianFilter filter;
filter.SetSigma(2);
Image out = filter.Blur(inImage);
```

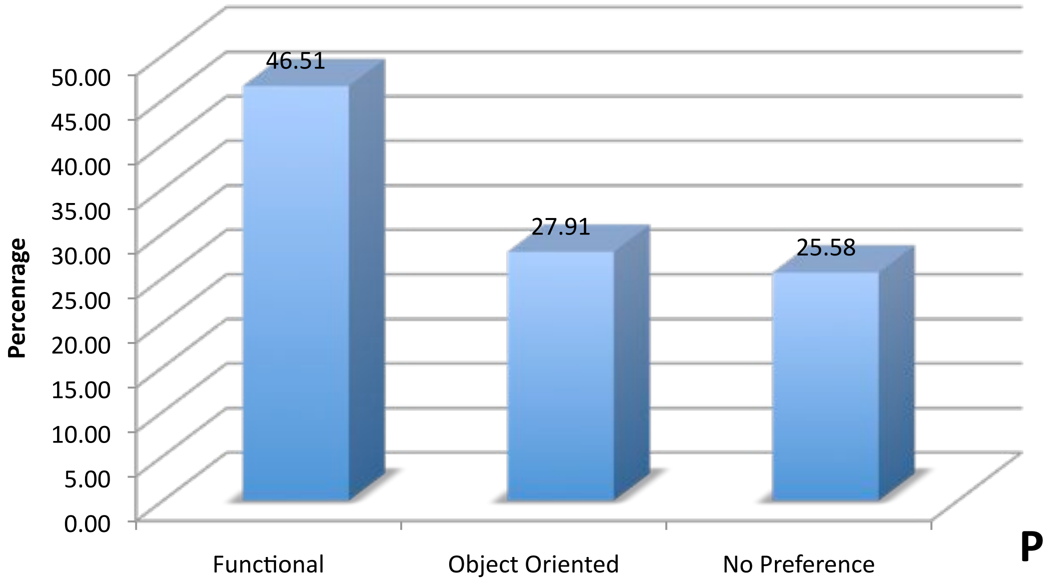
Functional vs. Object Oriented



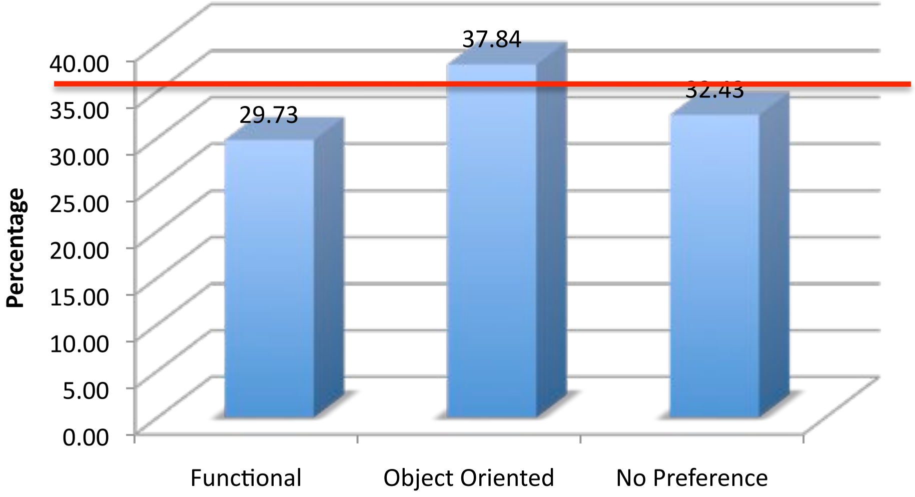
Overall, it seems that people prefer an *Object Oriented* programming style. However….

Functional vs. Object Oriented

Preference Among Non-C++ Programmers



Preference Among Non-ITK Users

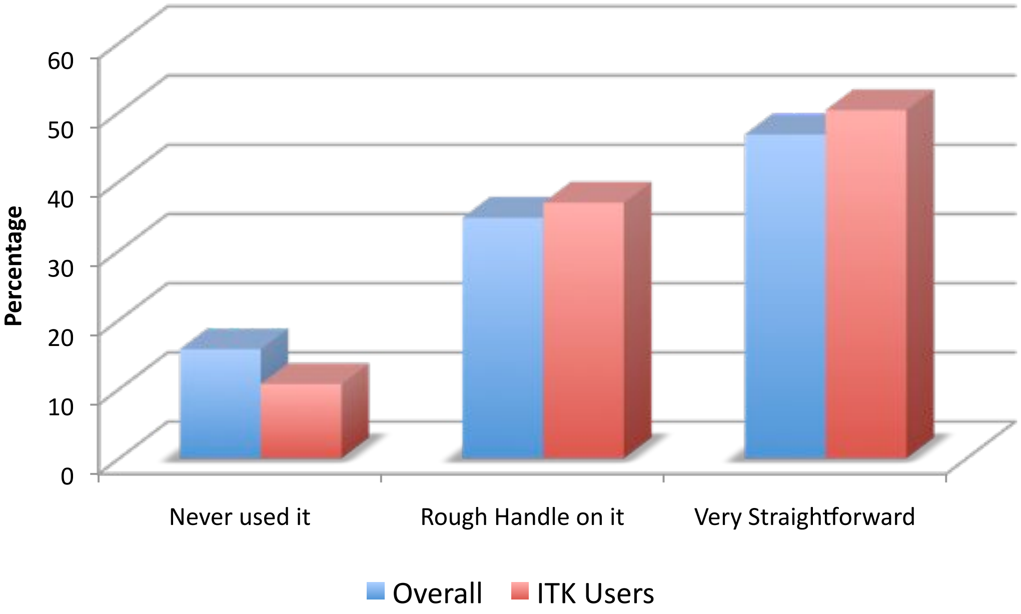


However:

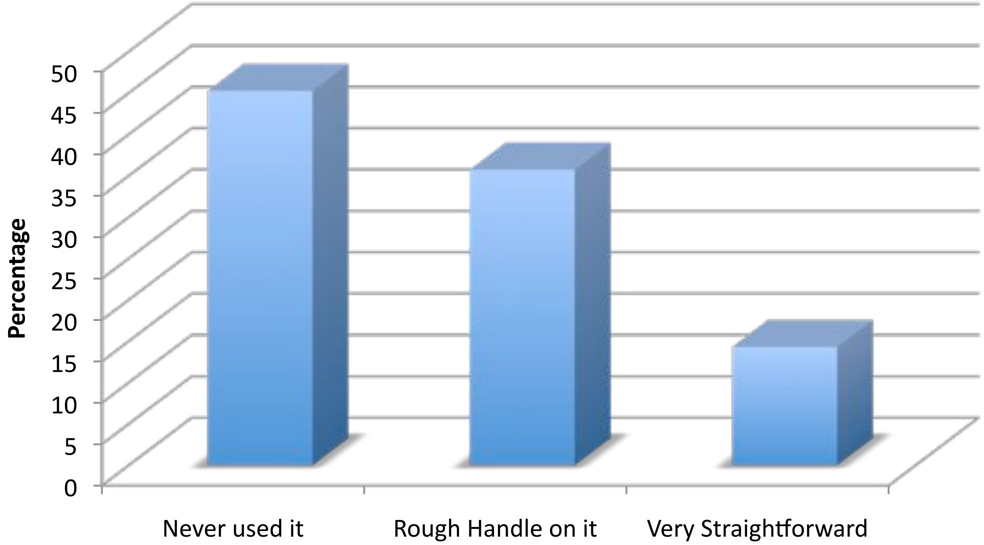
- •Non-C++ programmers prefer a Functional style
- •Non-ITK users don't have strong preferences

How well do you understand the pipeline?

Filter Pipeline



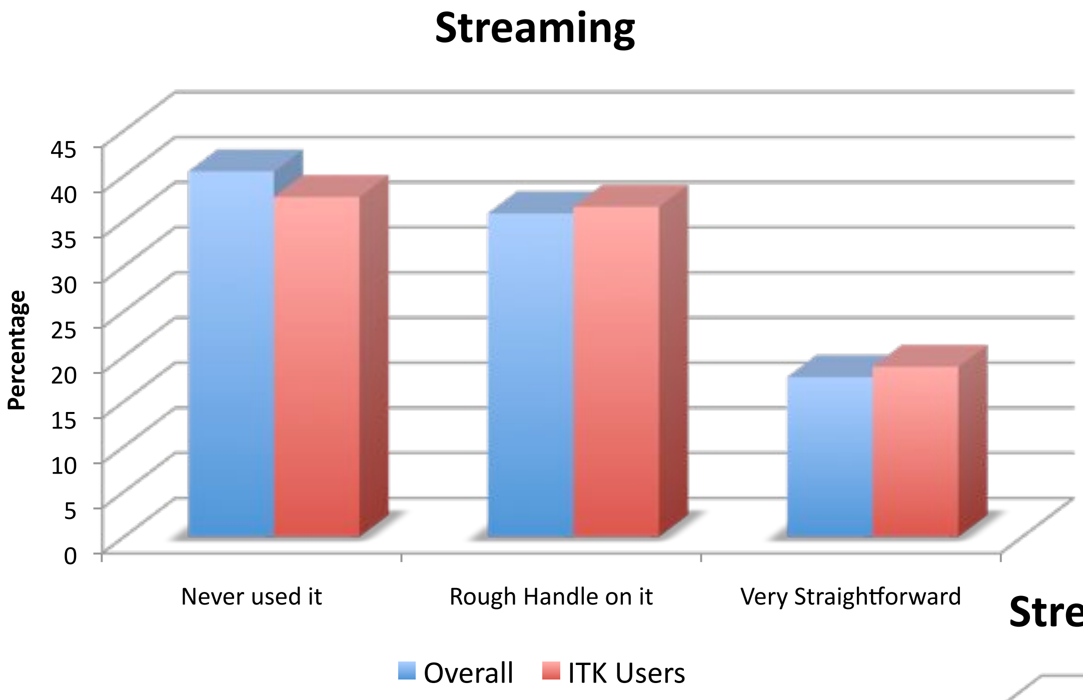
Filter Pipeline: Non-C++ Developers



Note:

Most non-C++ users (including the 51% that have used ITK), don't understand the pipeline.

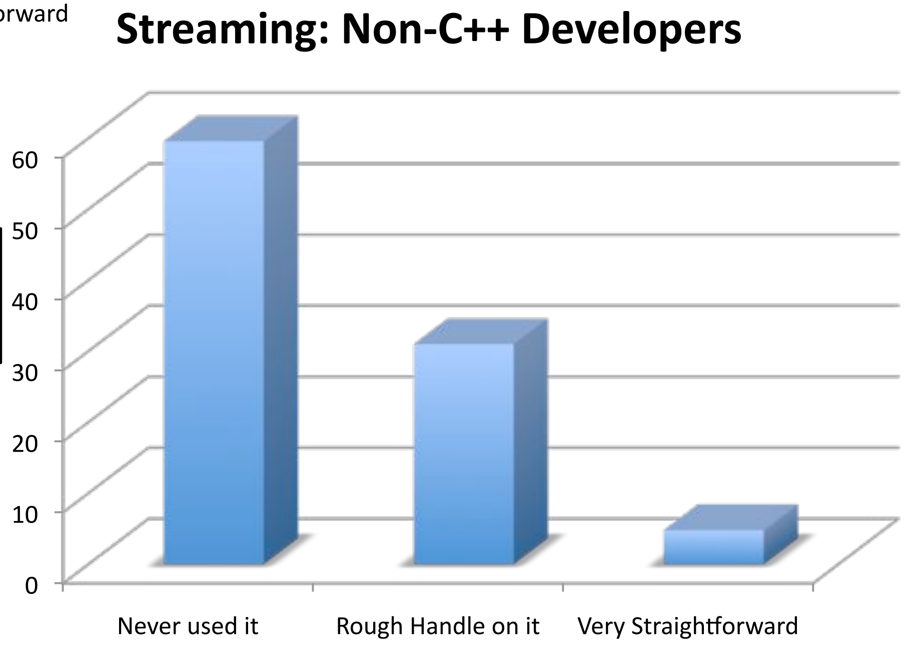
How well do you understand streaming?



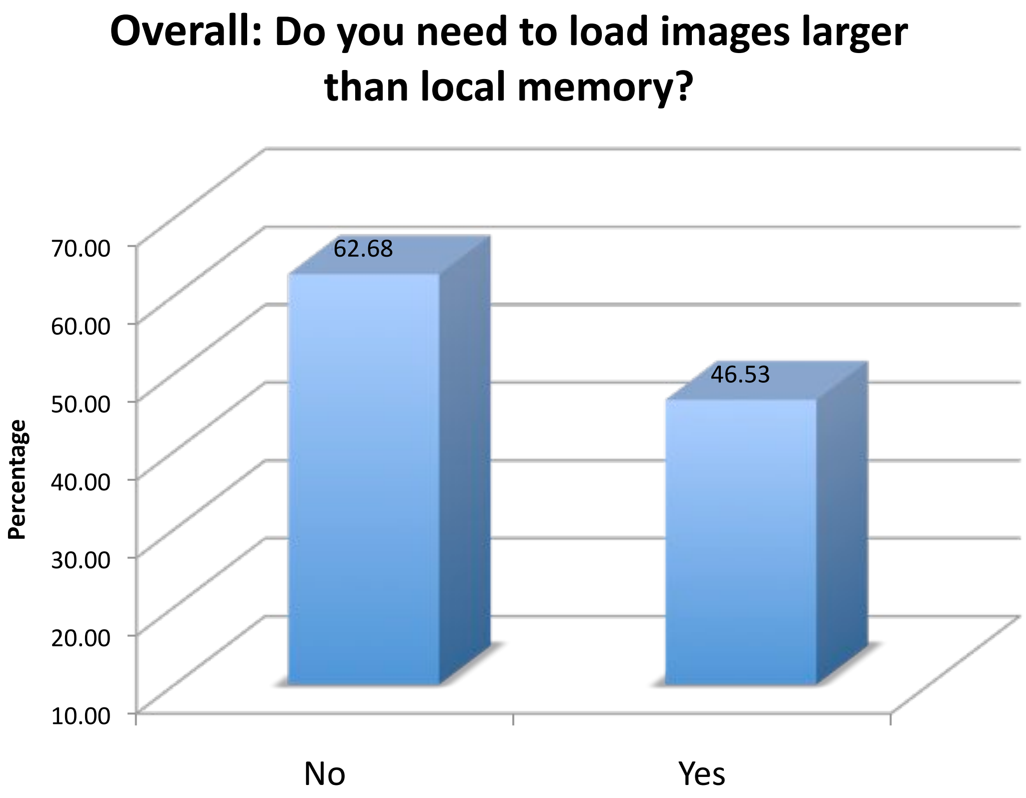
Note:

Most ITK, non-ITK, and C++ users don't have a clear understanding of streaming

However….

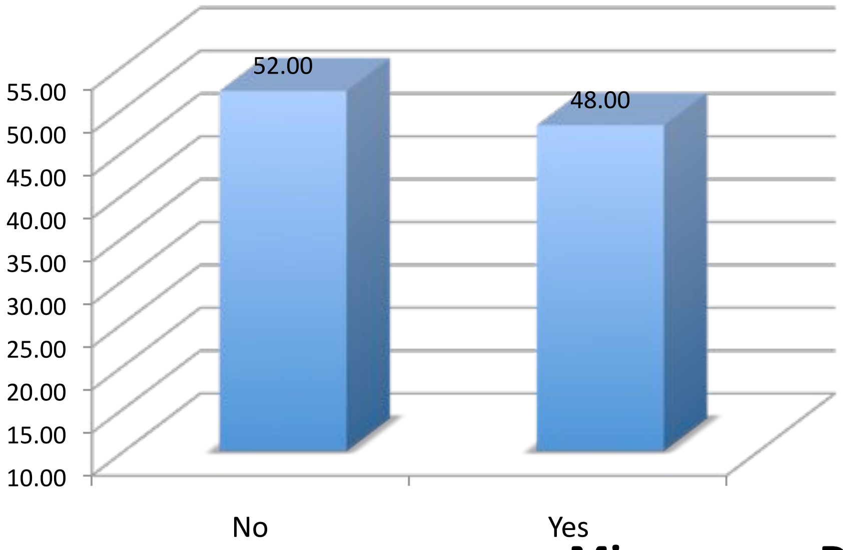


Loading images larger than RAM?

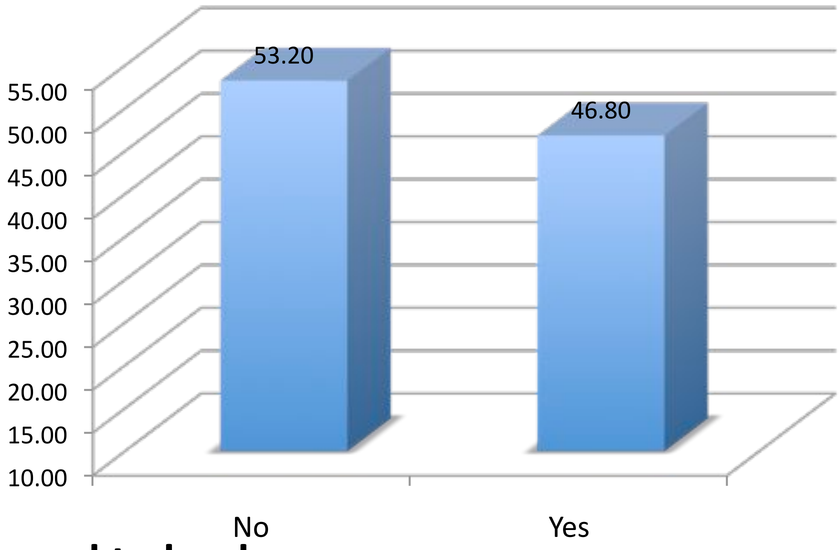


Loading images larger than RAM?

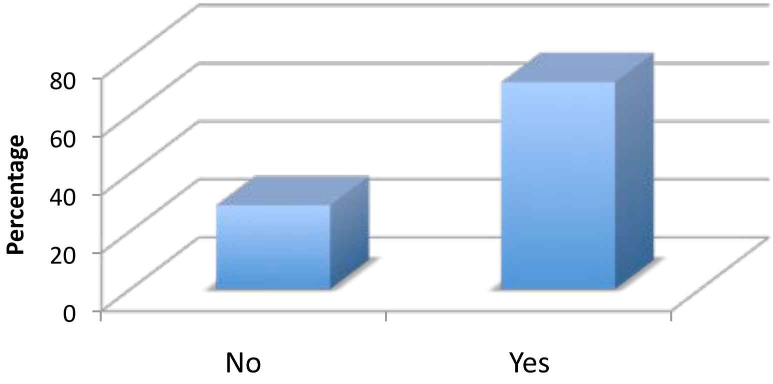
Non-C++ Users: Do you need to load images larger than local memory?



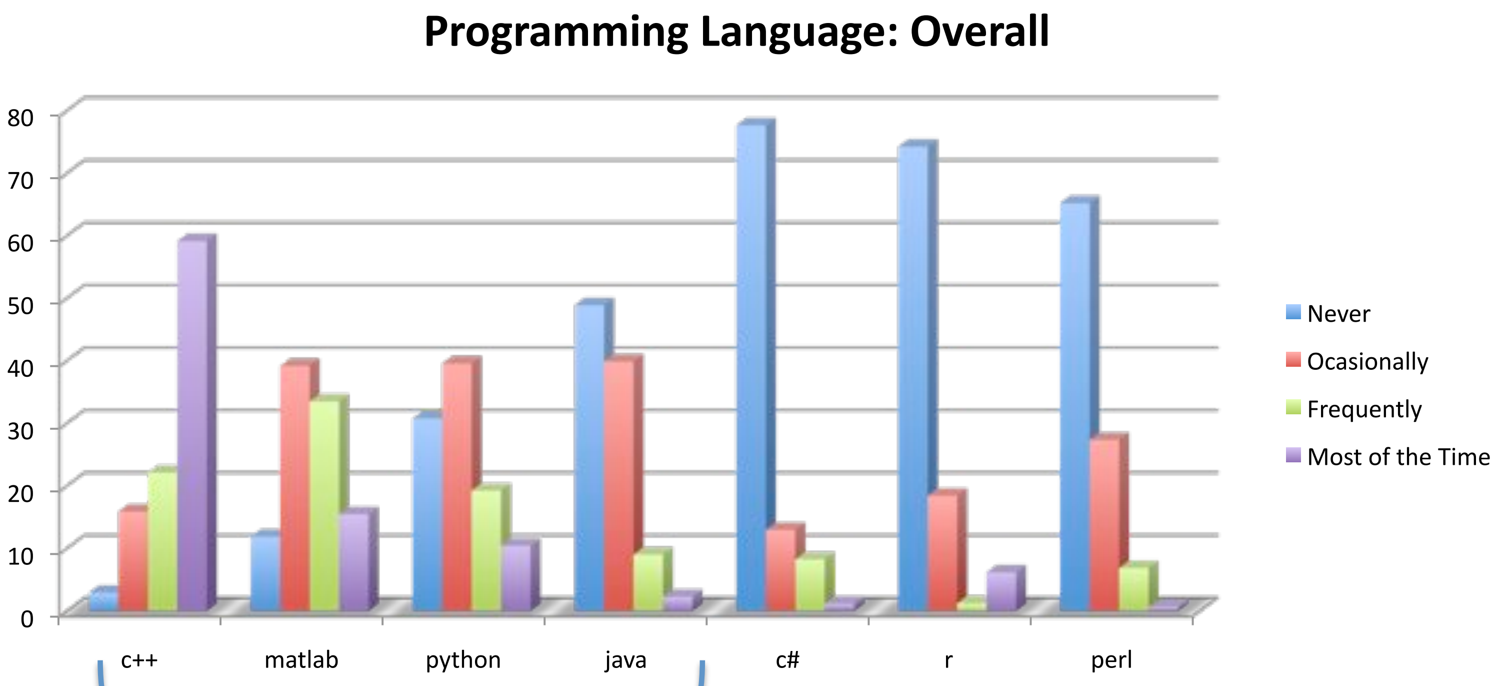
ITK Users: Do you need to load images larger than local memory?



Microscopy: Do you need to load images larger than local memory?



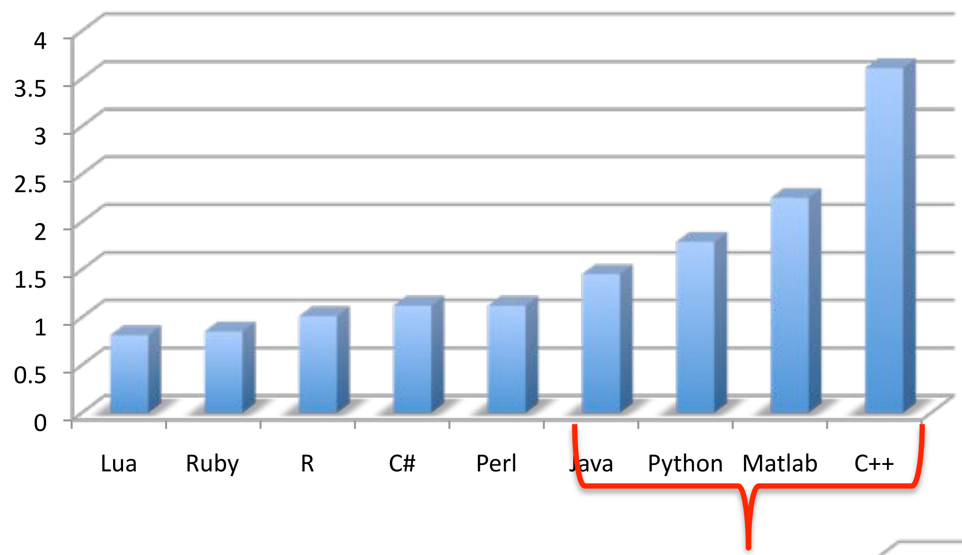
Programming Languages



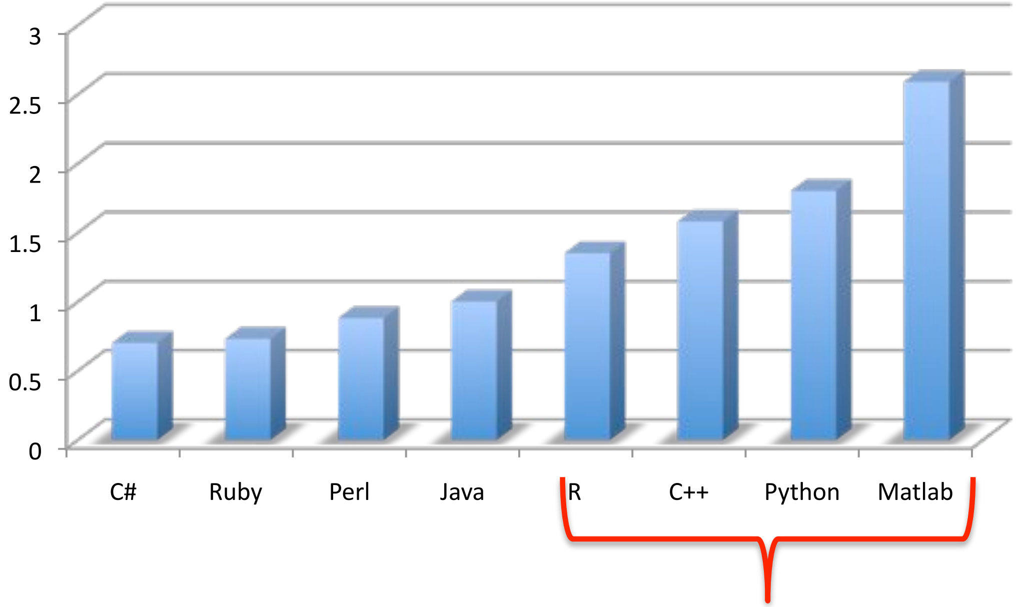
The most important programming languages?

Programming Languages

Preferences: C++ Experts



Preferences: Non C++ Developers



Q:

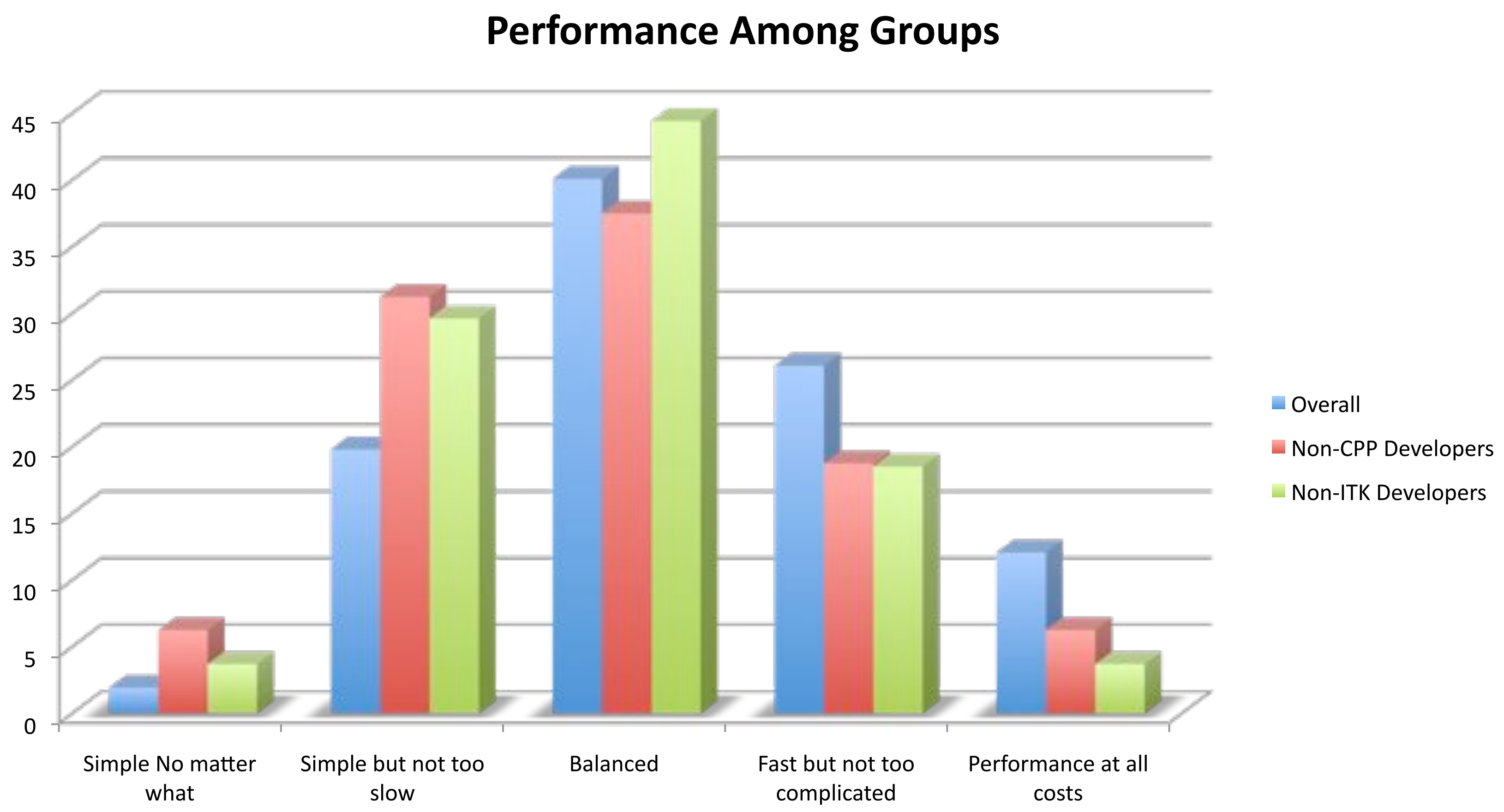
Can we say that Matlab, Python, C++, Java, and R are the most important programming languages?

Other programming languages

- Bash
- C
- Fortran
- JavaScript
- Mathematica
- Objective-C
- Octave
- PHP
- D
- Vala
- TCL
- Visual Basic

C/C++

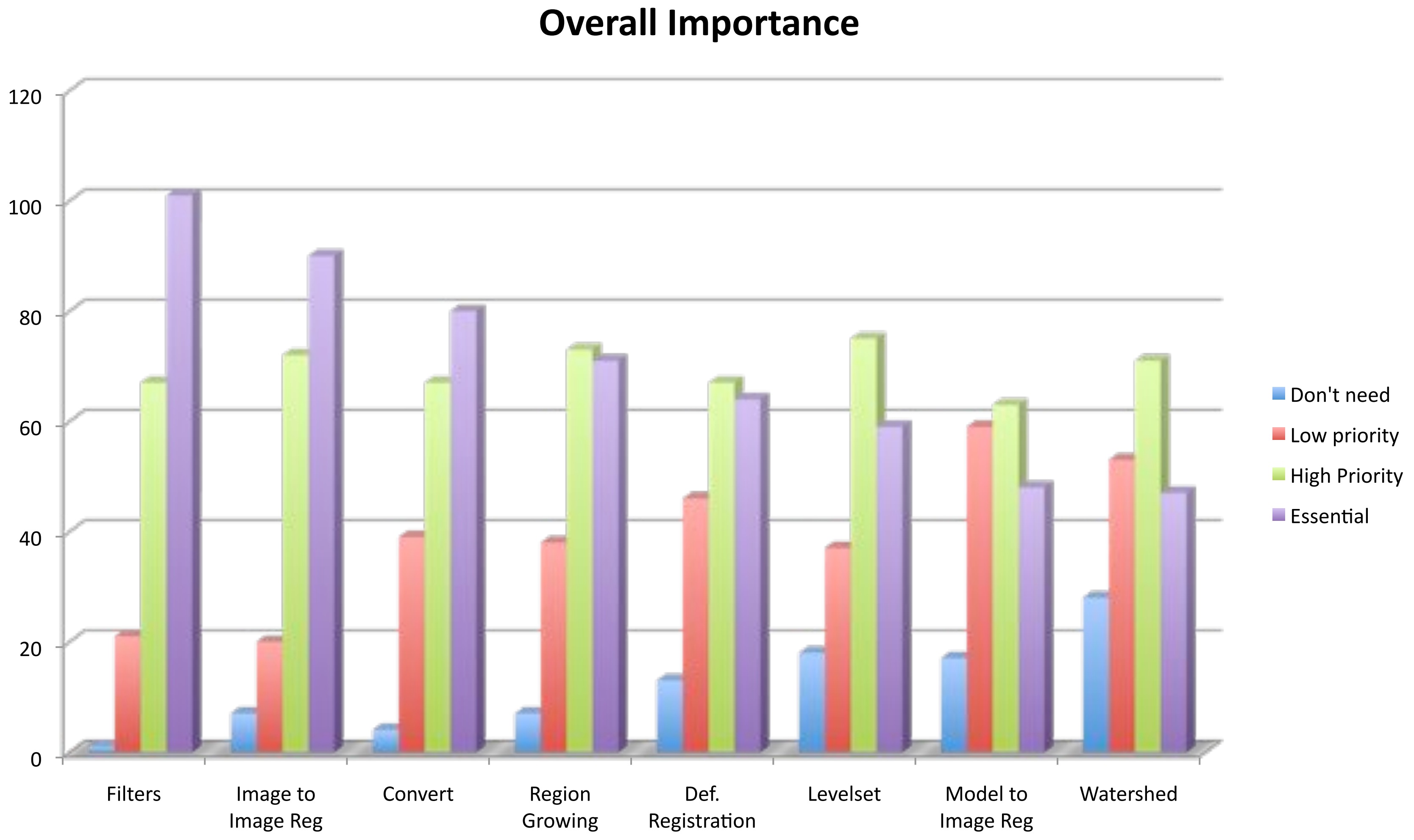
Simplicity vs. Performance



Order: 4 2 1 3 5

It seems that for non-C++ and non-ITK developers simplicity is more important than performance.

What ITK features are important?

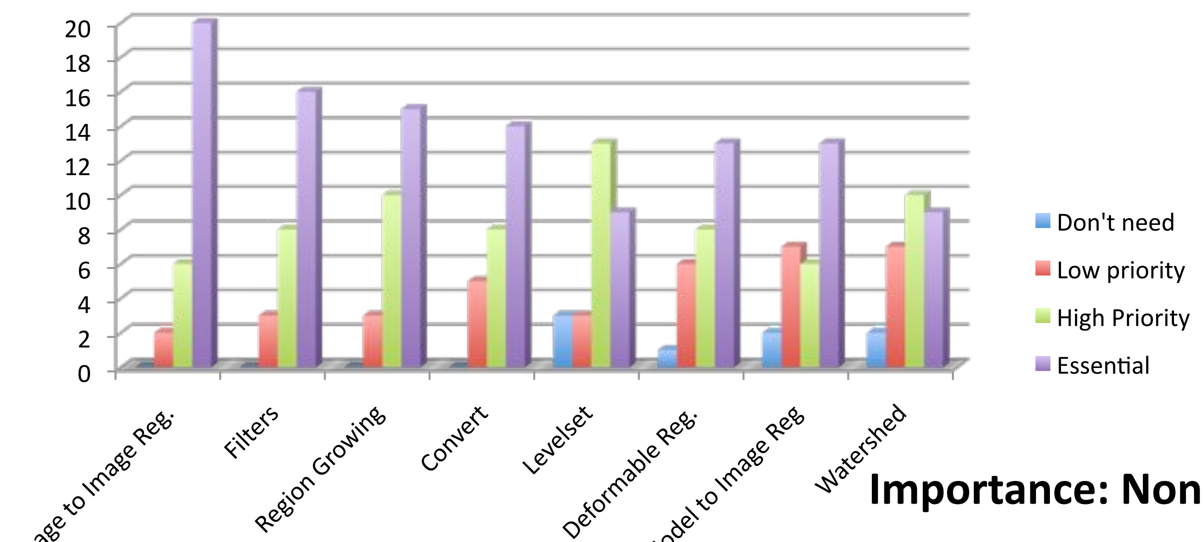


Note:

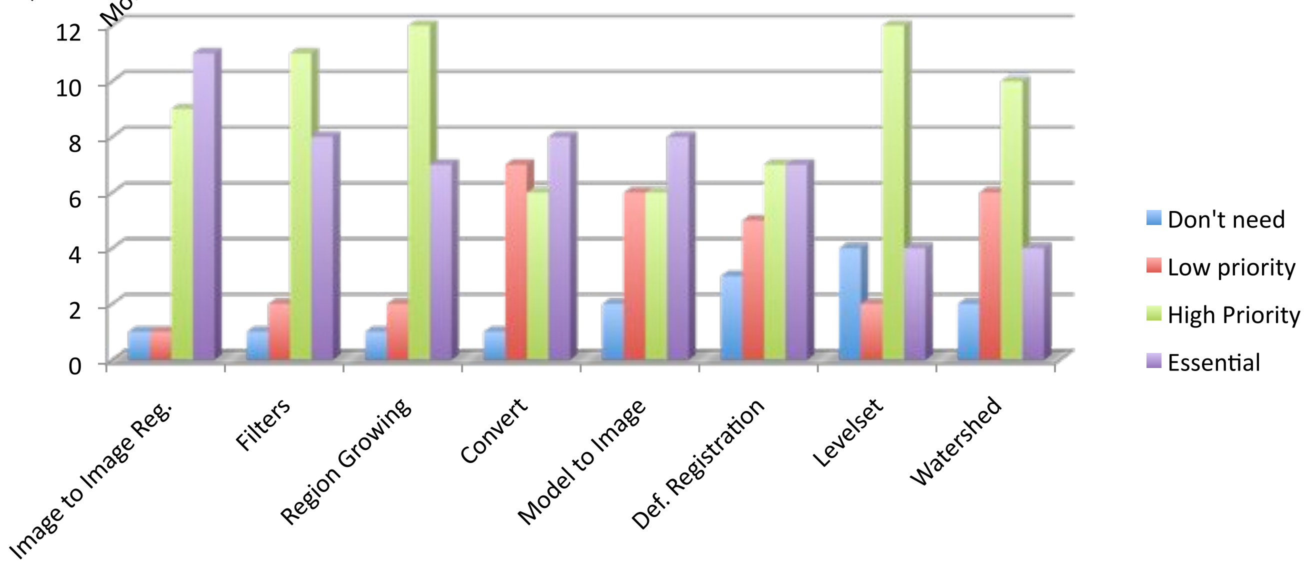
A good and simple segmentation technique is better than a complex and state-of-the-art approach?

What ITK features are important?

Importance: Non C++ Developer



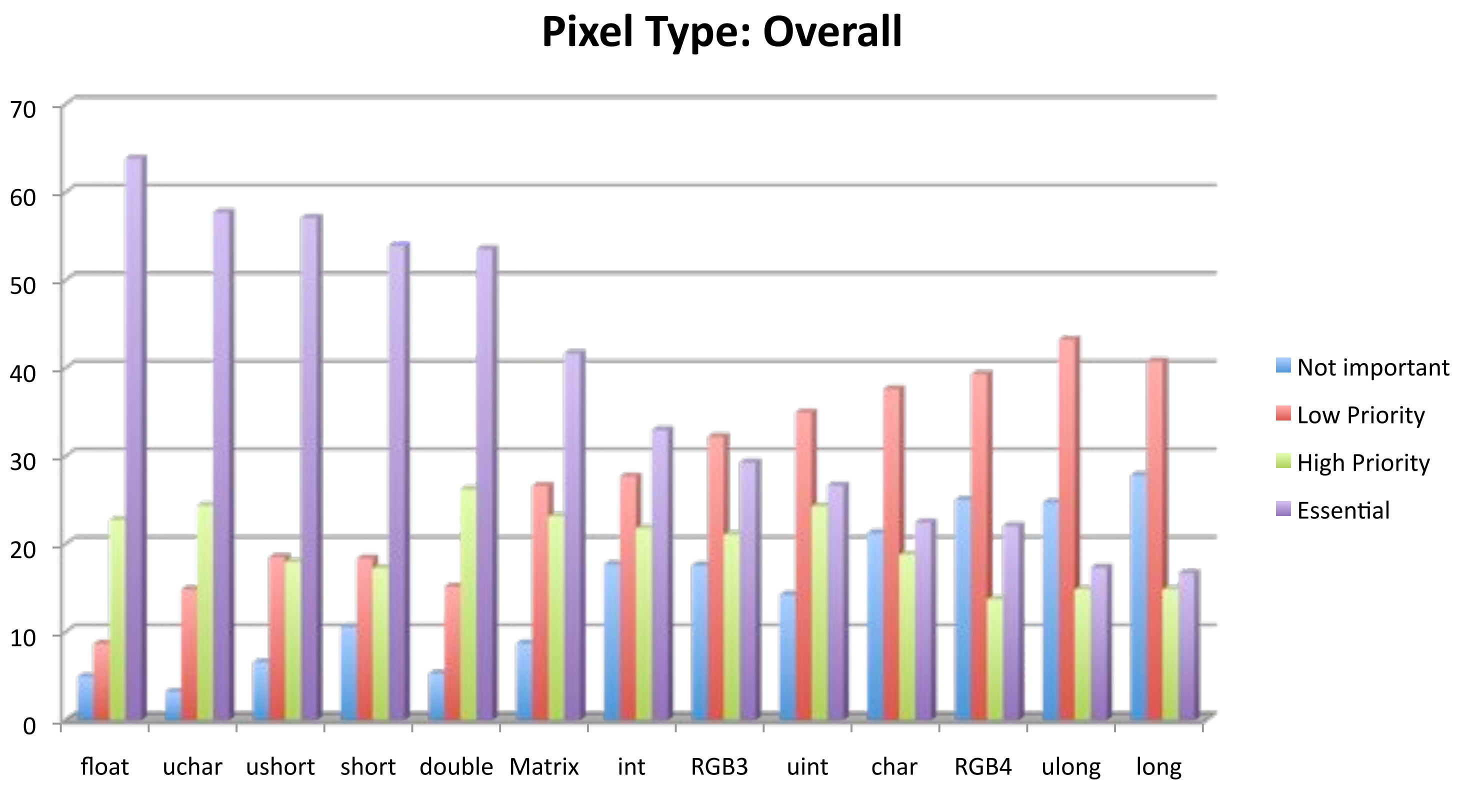
Importance: Non-ITK Developers



It seems that the most important features are:

1. Registration
2. Filters
3. Region Growing
4. Convert

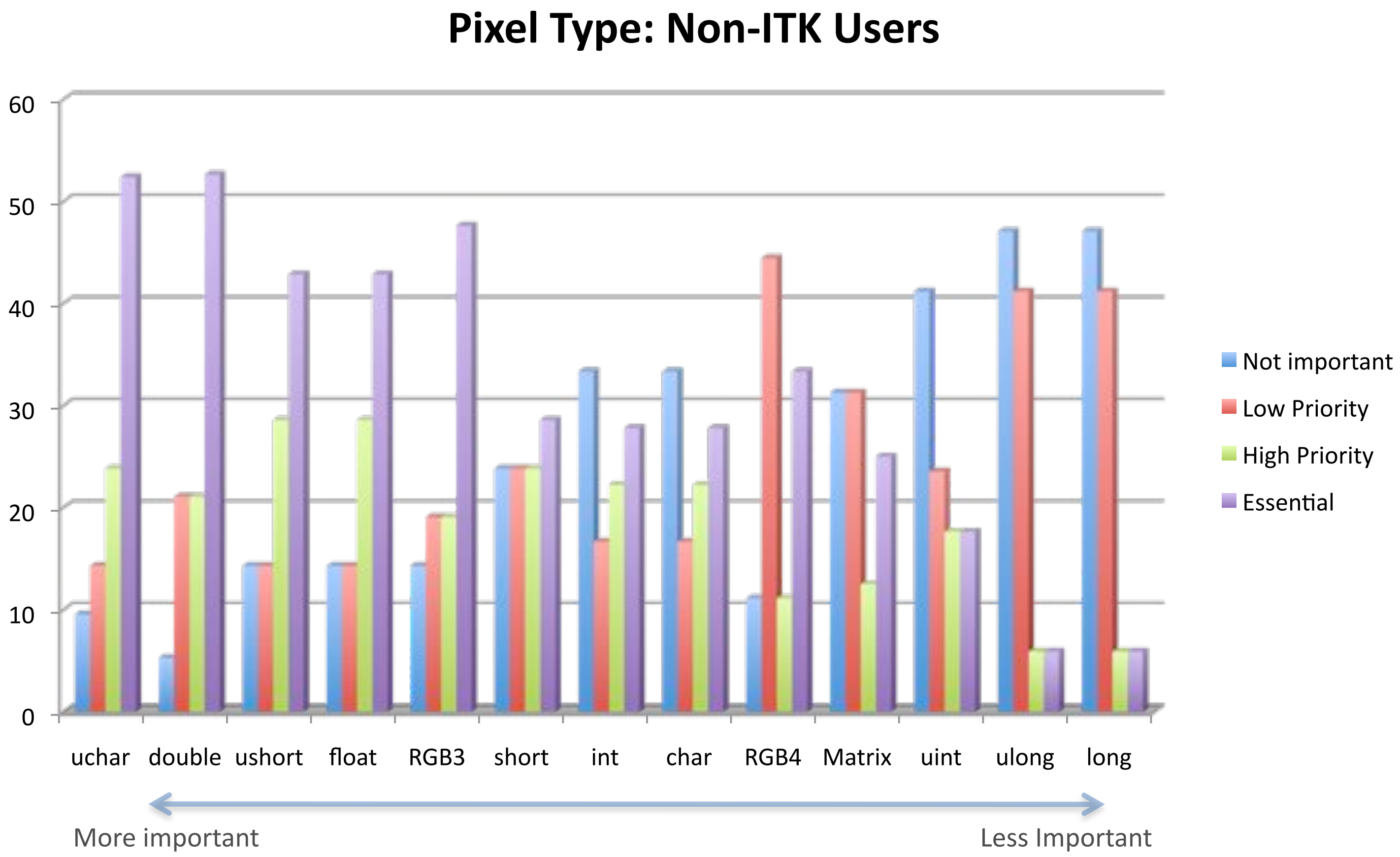
Pixel Type: Overall



More important

Less Important

Pixel Type: Non-ITK Users



Top 6 (Non-ITK users)

1. uchar
2. double
3. ushort
4. float
5. RGB3
6. short

1. float
2. uchar
3. ushort
4. short
5. double
6. matrix

Top 6 (Overall)

Pixel Type: Non-ITK Users

- Binary / bool images
- Imagej2
- Fiji
- Bool
- Tensor
- Multichannel
- NIFTI

Platform

- Other:
  - Cluster / grids
  - Phone / mobile apps
  - GPU-based cluster
  - iPad
  - Services on servers

Comments: Default Parameters

1. “Important to give users relatively straightforward advice on algorithm…”

2. “Simple defaults with optional alternatives”

3. “I generally want to be able to try something off the shelf with sensible defaults”

4. “Sensible defaults should be provided to avoid long code”

5. “Full flexibility is needed but a good selection of default parameters for all modules would be also important”

6. “initialized with reasonable default components”

Comments: Display Images

- “offer both...a GUI … and build scripts from there”
- “connect display to view intermediate images”
- “handling ability to visualize quickly (even at run time) images and histograms”
- “A GUI to build up the application pipeline by arranging icons of filters and I/O connectors”
- “visual tool for connecting filters to a pipeline

Comments: Documentation

- “writing ITK codes can be a complex task but with more examples and a thorough documentation these difficulties can be overcame”

- “One good way to compromise is more flexible but with lots of example code that can be cut-and-pasted for typical applications.”

- “Actually I think much of the motivation for SimpleITK could go away, if someone updated + improved the documentation”

- “some of the image filters modules are not sufficiently documented”

- “Lots of sample code that can be modified for specific purposes”

Comments: Other

- Features:
  - Graph-cut (e.g. boost graph-cut lib)
  - Tracking
  - Distributed I/O
  - Fast morphological operators

- General comments:
  - “It's better to provide fewer full-featured classes than to provide many partially-featured classes”
  - “I am not using ITK for registration purposes, and this is because of the complexity of it”
  - “Extensive use of templates makes debugging difficult in VisualC++”
  - “Most of filter called *VectorImageFilter are in fact not able to process itk::VectorImage ... This is confusing and sometimes frustrating”

- Frustrated folks:
  - Actually, ITK registration framework is not modular enough to be useful.
  - Ability to open differing file formats (dimensions, data types) without all the stupid c++ template nonsense.