In MICCAI 2014 Workshop on
Interactive Medical Image Computing

# An Interactive Visualization Tool for Nipype Medical Image Computing Pipelines

Ramesh Sridharan, Adrian V. Dalca, and Polina Golland

Computer Science and Artificial Intelligence Lab, MIT

**Abstract.** We present an interactive tool for visualization of medical imaging pipelines that are built with Nipype, a freely available tool for building pipelines programatically. Our tool enables researchers to interact with their pipelines, visualize the pipeline structure, and view their intermediate and final results. We also provide a video and live demonstration of our tool for a simple brain image registration pipeline.

## 1 Introduction

Medical image computing research frequently requires the development of elaborate analysis algorithms that are built as pipelines. Such pipelines are increasingly necessary as imaging datasets from population studies are growing larger, necessitating structured and consistent analysis across large patient cohorts. Building pipelines is an iterative process that involves substantial iterative refinement in order to attain meaningful results. Visualization of both the structure of a pipeline and of its intermediate results helps researchers and developers understand, debug, and improve the algorithm being developed. In this paper, we present an interactive tool for visualization of complex pipelines as well as their results. To facilitate use by computational researchers, we provide an extension to Nipype, a popularly used free and openly available tool for medical image computing workflows. We illustrate the proposed tool on a common medical image analysis task and provide a live demonstration and video detailing the features of our tool[1].

Nipype [4] provides a convenient way to build medical image computing pipelines. A pipeline in Nipype is specified programatically as a script; users can define arbitrary pipelines according to the needs of any particular task. Pipelines are represented internally as graphs whose nodes provide interfaces to various commonly used medical image analysis packages such as ANTS [1], FreeSurfer [3], and FSL [6]. A directed edge between two nodes indicates that an output of the parent is used as an input to the child. The result of each step is cached for ease of debugging and iteration. Additionally, Nipype tracks provenance information such as algorithm parameters and properties of data. In contrast to similar workflow tools that present a graphical interface, such as LONI Pipeline [10]

[1] The video and demonstration, along with code, are available at the following URL: `http://groups.csail.mit.edu/vision/medical-vision/pipelines.html`.

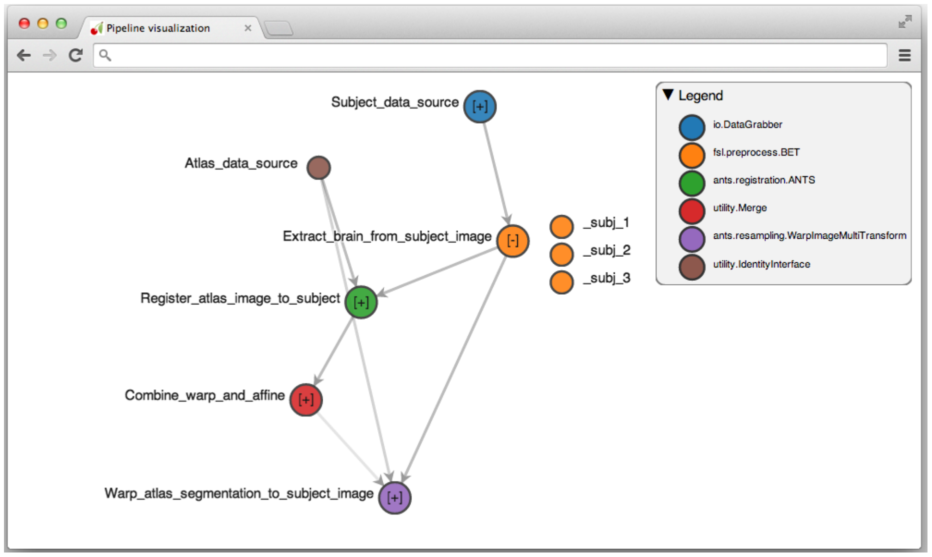

**Fig. 1.** A visualization of a simple registration pipeline. Nodes correspond to computational steps (the color represents the node type, as detailed in the legend) and edges correspond to data dependencies between processing steps. The smaller visual nodes beside the orange node correspond to iterations of the corresponding computational step across multiple subjects.

and SCIRun [9], Nipype pipelines are built exclusively through scripting. While this allows greater expressive power, it comes at the cost of limited visualization: Nipype pipelines can be visualized only statically using the `dot` library. This static visualization is not customizable, and more importantly, does not allow interaction with the results of the pipeline.

Our tool presents a graphical web-based interface to Nipype pipelines that enables interaction with and visualization of both the structure of a pipeline and its results. Users can quickly examine and interact with the dependencies of the pipeline graph. Furthermore, our tool harnesses Nipype's detailed record keeping to enable interaction with the results and status of every step. By enabling users to view a pipeline's resulting images as they are computed, our system facilitates examination and a better understanding of the pipeline structure, which can lead to improvement of the underlying algorithm.

## 2 Design

Visualization and interaction are crucial components of medical image computing research. Our lightweight browser-based tool provides the ability to visualize a graphical representation of a pipeline, which seamlessly integrates with visualization of image-valued results of Nipype nodes. The nodes in our graph-based pipeline visualization correspond directly to Nipype nodes.

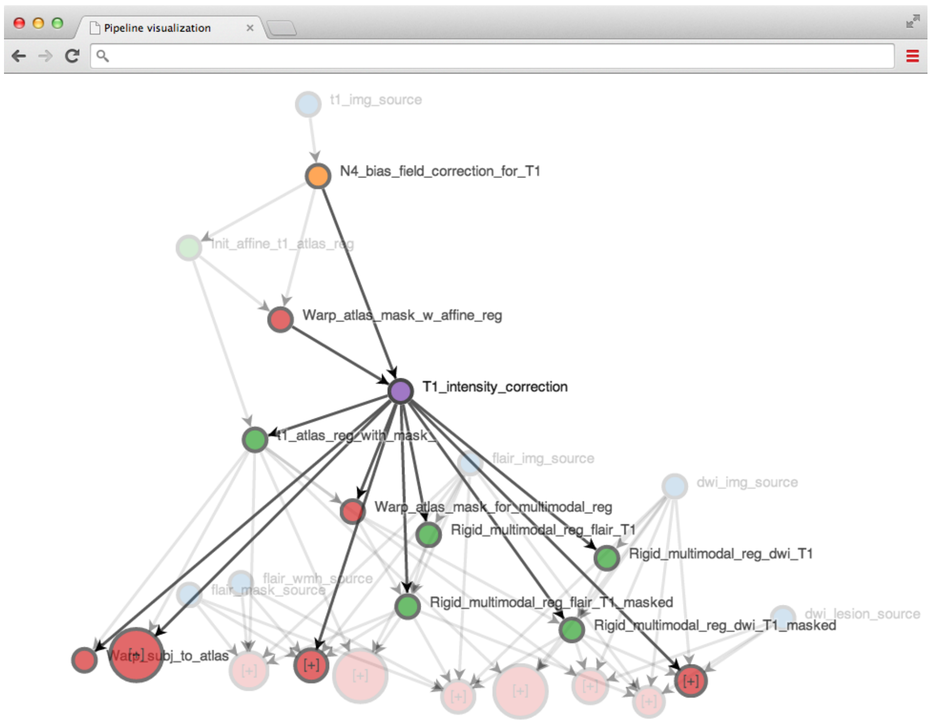
Pipeline visualization
t1_img_source
N4_bias_field_correction_for_T1
Init_affine_t1_atlas_reg
Warp_atlas_mask_w_affine_reg
T1_intensity_correction
t1_atlas_reg_with_mask
flair_img_source
dwi_img_source
Warp_atlas_mask_for_multimodal_reg
Rigid_multimodal_reg_flair_T1
Rigid_multimodal_reg_dwi_T1
flair_wmh_source
flair_mask_source
Rigid_multimodal_reg_flair_T1_masked
dwi_lesion_source
Rigid_multimodal_reg_dwi_T1_masked
Warp_subj_to_atlas
[+]

**Fig. 2.** A visualization of a complex pipeline. The size of a node is related to the number of replications of the processing step, e.g., for multiple parameter settings or subjects. In this case, the large red node corresponds to warping various features from an atlas image to a particular image modality. The legend is hidden to show the full pipeline.

Our tool's visualization is also compatible with various Nipype features. Nipype's *iterables* allow any Nipype node (and the subgraph of its children) to be executed multiple times for a range of parameter values. This enables a pipeline to easily scale to multiple subjects or parameter settings. Using our tool, any Nipype node that is replicated in this way is annotated with a plus sign, and can be clicked on to view the replications. When the number of replications is large, as is typically the case in pipelines that run over large image collections with many subjects, an abbreviated listing is presented instead.

Fig. 1 illustrates visualizing a registration-segmentation pipeline that, given a subject brain MRI, performs skull extraction, registers the result to an atlas image, and warps a label map from the atlas image to the subject image for use in segmentation. The pipeline is applied to three scans from the images used to construct the FreeSurfer atlas [2,7,8]. Fig. 2 illustrates a more complex pipeline that aligns multimodal images of the brain to an atlas, demonstrating the visualization of system with more replications and interactions.

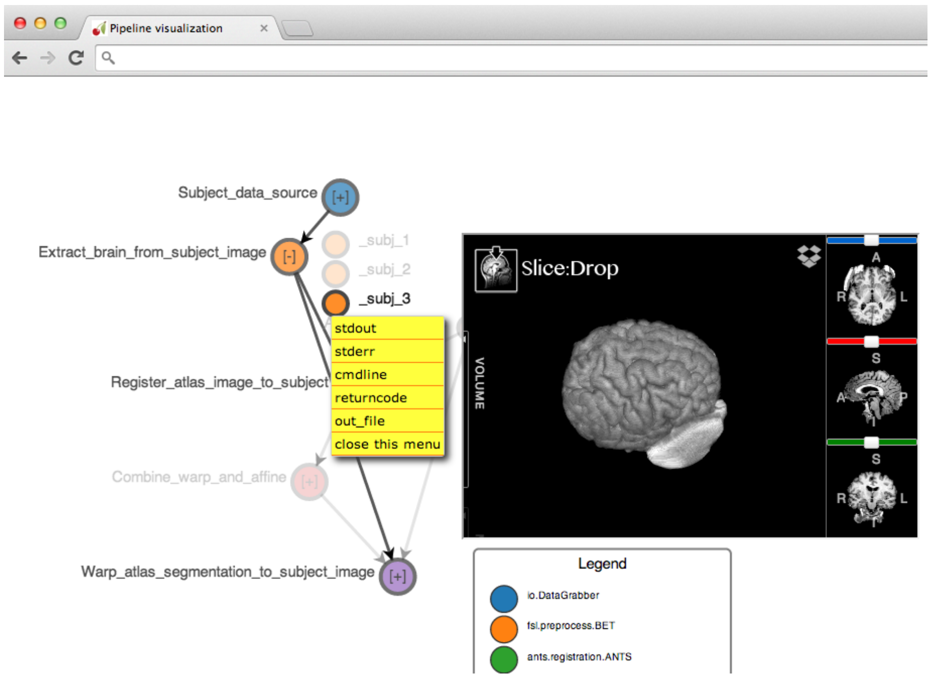

**Fig. 3.** Clicking on the orange visual node in Fig. 1 produces this visualization of the result of the brain extraction step using Slice:Drop. The visualization, which is overlaid on top of the pipeline visualization, makes it clear that the skull removal has mostly succeeded with a few remnants.

We also provide visualization of the output, status, and other metadata for any step within the pipeline. In particular, clicking on any node in the graph presents an interface that displays the result of the corresponding computational step, as well as the command used to produce it, any errors that resulted, and any command-line output from that step. While Nipype stores all this information by default, we make it easily accessible, enabling rapid debugging and pipeline iteration. Additionally, visualization of intermediate image-valued results facilitates understanding of the results of each node and speeds up the search for points of failure. Clicking on a node opens a menu that reports the exact command used to compute the corresponding Nipype node, as well as its command-line output or error status. Additionally, each output of the corresponding Nipype node can then be viewed interactively in the same window using Slice:Drop [5], a browser-based visualization tool that supports most common medical imaging formats. Fig. 3 illustrates such a visualization for a single Nipype node using Slice:Drop.

Our tool integrates directly with Nipype, requiring no additional input or annotation from the user. Because a paper is not an ideal medium for presenting an interactive visualization tool, we invite the readers to explore an interactive version of the demonstrations from Fig. 1 and Fig. 3, a short video, and the code for our system at the following site: `http://groups.csail.mit.edu/vision/medical-vision/pipelines.html`. In the near future, we plan to integrate our work into the main Nipype codebase.

### 2.1 Implementation details

Our tool consists of two components, a lightweight server (implemented in python using CherryPy[2]) and client (implemented in Javascript using d3.js[3]). The server interacts with the Nipype workflow and the filesystem to construct the underlying graph representation and data storage. Using the provided representation, the client renders the visualization. Actions such as clicking on a visual node to query the status of the corresponding Nipype node will result in the server retrieving the corresponding information using Nipype's internal representation and providing it to the browser. These actions all occur in real time, resulting in a quick, responsive interface.

## 3 Acknowledgments

We acknowledge the following funding sources: NSF GRFP, NSERC CGS-D, NIH NIBIB NAMIC U54-EB005149, NIH NIBIB NAC P41EB015902.

## References

1. B.B. Avants, N.J. Tustison, G. Song, P.A. Cook, A. Klein, and J.C. Gee. A reproducible evaluation of ants similarity metric performance in brain image registration. *Neuroimage*, 54(3):2033–2044, 2011.
2. E. Daly et al. Predicting conversion to Alzheimer disease using standardized clinical information. *Archives of Neurology*, 57(5):675, 2000.
3. B. Fischl. Freesurfer. *Neuroimage*, 62(2):774–781, 2012.
4. K. Gorgolewski, C.D. Burns, C. Madison, D. Clark, Y.O. Halchenko, M.L. Waskom, and S.S. Ghosh. Nipype: a flexible, lightweight and extensible neuroimaging data processing framework in python. *Front Neuroinform*, 5, 08 2011.
5. D. Haehn. Slice:Drop: Collaborative medical imaging in the browser. In *ACM SIGGRAPH 2013 Computer Animation Festival*, SIGGRAPH '13, New York, NY, USA, 2013. ACM.
6. M. Jenkinson, C.F. Beckmann, T.E.J. Behrens, M.W. Woolrich, and S.M. Smith. Fsl. *Neuroimage*, 62(2):782–790, 2012.
7. K.A. Johnson et al. Preclinical prediction of Alzheimer's disease using SPECT. *Neurology*, 50(6):1563–1571, 1998.
8. R.J. Killiany et al. Use of structural magnetic resonance imaging to predict who will get Alzheimer's disease. *Annals of neurology*, 47(4):430–439, 2000.
9. S.G. Parker and C.R. Johnson. SCIRun: a scientific programming environment for computational steering. In *Proceedings of the 1995 ACM/IEEE conference on Supercomputing (CDROM)*, page 52. ACM, 1995.
10. D.E. Rex, J.Q. Ma, and A.W. Toga. The LONI pipeline processing environment. *Neuroimage*, 19(3):1033–1048, 2003.

[2] `http://cherrypy.org/`
[3] `http://d3js.org/`