SimpleElastix: A user-friendly, multi-lingual library for medical image registration

Kasper Marstal[1], Floris Berendsen[2], Marius Staring[2] and Stefan Klein[1]

[1]Biomedical Imaging Group Rotterdam (BIGR), Department of Radiology & Medical Informatics, Erasmus Medical Center, PO Box 2040, Rotterdam, 3000 CA, the Netherlands,
{kmarstal,sklein}@erasmusmc.nl

[2]Division of Image Processing (LKEB), Department of Radiology, Leiden University Medical Center, PO Box 9600, 2300 RC Leiden, the Netherlands, {f.berendsen,m.staring}@lumc.nl

Abstract

In this paper we present SimpleElastix, an extension of SimpleITK designed to bring the Elastix medical image registration library to a wider audience. Elastix is a modular collection of robust C++ image registration algorithms that is widely used in the literature. However, its command-line interface introduces overhead during prototyping, experimental setup, and tuning of registration algorithms. By integrating Elastix with SimpleITK, Elastix can be used as a native library in Python, Java, R, Octave, Ruby, Lua, Tcl and C# on Linux, Mac and Windows. This allows Elastix to intregrate naturally with many development environments so the user can focus more on the registration problem and less on the underlying C++ implementation. As means of demonstration, we show how to register MR images of brains and natural pictures of faces using minimal amount of code. SimpleElastix is open source, licensed under the permissive Apache License Version 2.0 and available at https://github.com/kaspermarstal/SimpleElastix.

1. Introduction

In the past decade there has been an increasing interest in relating information from different medical images spurred by a growing availability of scanners, modalities and computing power. This typically involves image registration for transforming images into a common coordinate system so corresponding pixels represent homologous biological points. Clinical applications include segmentation of anatomical structures, computer-aided diagnosis, monitoring of disease progression, surgical intervention and treatment planning.

A significant amount of research has focused on developing the registration algorithms themselves. However, less research has focused on accessibility, interoperability and extensibility of these algorithms. Scientific source code is typically not published [16], is lacking documentation or is difficult to use because it has not been written with other researchers in mind. This is a problem since image registration is a prerequisite for a wide range of medical image analysis tasks and a key algorithmic component for image-based studies. Open source, user-friendly implementations of scientific software make state-of-the-art methods accessible to a wider audience, promote opportunities for scientific advancement, and support the fundamental scientific principle of reproducibility. To this end, we have developed the SimpleElastix software package.

```
import SimpleITK as sitk
resultImage = sitk.Elastix(
  sitk.ReadImage("fixedImage.nii"),
  sitk.ReadImage("movingImage.nii")
)
```

Listing 1: Medical image registration in Python using SimpleElastix.

Elastix [19] is an open source, command-line program for intensity-based registration of medical images that allows the user to quickly configure, test, and compare different registration methods. SimpleElastix is an extension of SimpleITK [21] that allows the user to configure and run Elastix entirely in Python, Java, R, Octave, Ruby, Lua, Tcl and C# on Linux, Mac and Windows. In addition, we have developed an extensive test suite, online documentation, and a citable online software repository following recommended best practices for scientific software publication [24], all within an online environment that stimulates com-

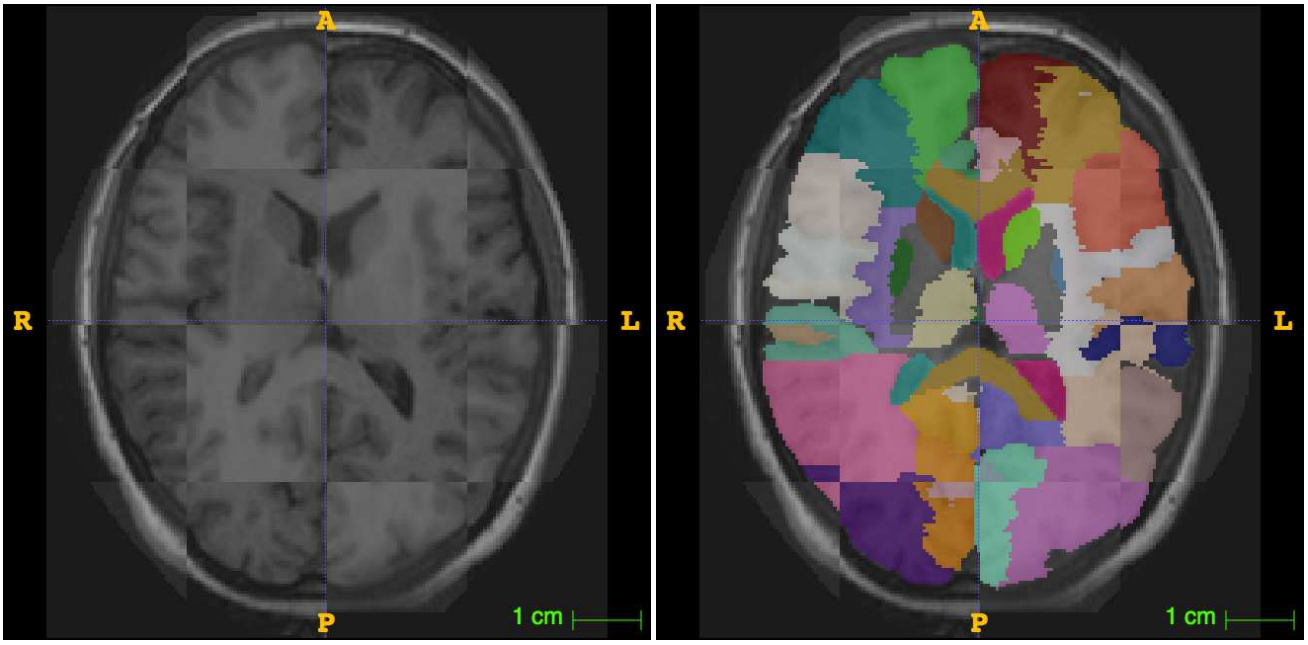

Figure 1: Example of two registered images from the Hammersmith83 dataset using default registration procedure as shown in Listing 1. Checkerboard pattern of fixed and result images (left) and overlay of associated segmentations (right).

munity involvement and contribution. SimpleElastix also includes SimpleTransformix, an accompanying program for subsequent warping of images, point sets and segmentations using transformation fields computed with SimpleElastix.

The software contributions are three-fold.

- A low-level ITK-filter-based C++ interface for Elastix and Transformix.
- A facade interface to these filters that enables SimpleITK to build Elastix and Transformix bindings for Python, Java, R, Ruby, Octave/Matlab, Lua, Tcl and C# on Linux, Mac and Windows.
- Preconfigured registration methods that serve as starting points for tuning registration algorithms to domain-specific applications.

Together, these contributions lower the entry barrier for performing image registration. For example, Figure 1 shows a registration of two brains from a publicly available data set consisting of 30 adult atlases each with 83 segmented regions [14] using the method in Listing 1. A Dice Similarity Coefficient (DSC) of 0.92 was obtained for the total brain volume and a mean DSC of 0.74 was obtained for the individually labeled regions for this particular example. In the experiments section we present the full setup used to obtain these results.

The remainder of the paper is laid out as follows. First, we introduce previous open source medical image registration libraries, discuss their influence on the design of SimpleElastix, and present the contributions that enable SimpleITK to link Elastix with other programming languages. We then outline the continuous integration (CI) testing framework, online documentation framework and online contribution process. Finally, we apply a built-in preconfigured registration method to MR images of brains and 2D natural pictures of faces. The focus of the paper is on reducing complexity of experimental setups for typical image registration problems. All experiments in this paper are available online.

1.1. Previous Work

A number of open source libraries are available online and the scientific community has been thriving around the use of these tools. The National Library of Medicine's Insight Segmentation and Registration Toolkit (ITK) is a collection of methods for image filtering, segmentation and registration. It is widely regarded as the standard reference for many medical image processing algorithms. Elastix is based on ITK's v3 registration framework [18] and makes extensive use of ITK's data structures for images, points, spatial objects, and several basic image processing components such as filtering, morphological operations, and interpolation schemes. Elastix adds an additional configuration layer that allows registration algorithms to be configured via text-based files. This allows users without expert programming knowledge to compare different registration

algorithms. However, command-line scripting and manual editing of text-files typically needed for using Elastix in larger studies makes experimentation time-consuming, difficult to reproduce and hard to maintain.

Recently, there has been considerable interest in developing bindings for other languages in order to streamline the use of ITK and bring its algorithms to a wider audience. The SimpleITK project [21] was developed specifically with usability in mind to overcome the steep learning curve of previous ITK wrapping efforts like WrapITK for Python, Java and Tcl [2] and ManagedITK for the .NET language family [23]. SimpleITK exposes simplified, native interfaces to ITK algorithms in a variety of scripting languages following the facade design pattern [9]. Each ITK algorithm is wrapped in facade bindings that hide implementation details such as the use of templates and the ITK pipeline architecture. This allows a user to setup an ITK pipeline in a programming environment of their choice and without knowing anything about the underlying C++ implementation. For example, a researcher may choose to prototype image processing pipelines in high-level scripting languages like Python, R or Octave and enterprise developers may choose to develop distributed computing software in languages like Java or C#. In this work we aim to bring the same level of versatility, convenience, and user-friendliness to the Elastix community.

Libraries such as Nipype [10] and ElastixFromMatlab [8] provide command-line wrappers for Elastix. Nipype is a Python library that facilitates comparison studies by providing uniform access to many popular image processing frameworks including ANTs, SPM, FSL, FreeSurfer, Camino, MRtrix, MNE, AFNI, Slicer and Elastix. ElastixFromMatlab makes it easier to use Elastix from Matlab environments. However, the command-line wrapping does not achieve the tight integration that SimpleElastix offers and consequently suffer from disk I/O before and after each registration and manual bookkeeping of intermediate files in the image processing pipeline. In this work we aim to remove the layer of complexity associated with command-line wrapping.

We opted to integrate Elastix with SimpleITK because of the support for multiple languages, interoperability with SimpleITK's image processing algorithms and native in-memory data structures for 2D, 3D and 4D images. We do not make any assumptions about workflow and file handling, as different approaches are preferred in different languages. Instead, we focus on providing a clean, flexible interface that supports both functional- and object-oriented programming paradigms so the user can setup experiments in whatever way she prefers.

In the following section we describe how Elastix was integrated with SimpleITK.

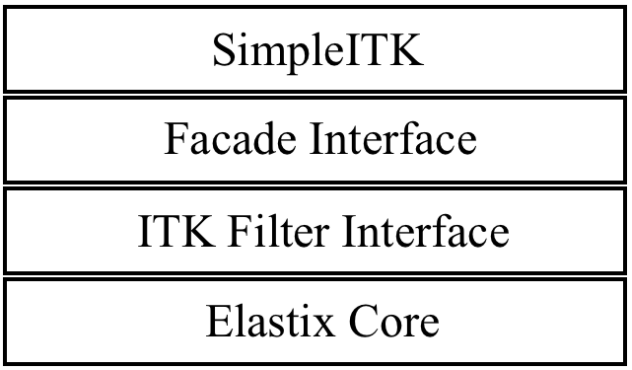

Figure 2: Software architecture of SimpleElastix. Three layers are built on top of Elastix, each wrapping the layer below. ITK filters control the elastix library, facade classes provide a simplified interface to ITK filters and SimpleITK uses the facade classes to generate and compile code for target languages.

2. Method

The proposed software package consists of three layers on top of the Elastix library as depicted in Figure 2. The first layer is an ITK filter-based C++ API that enables Elastix to be used in a regular ITK pipeline. The filter is templated over pixel types and dimensions of both the fixed and moving images. The second layer consist of two facade C++ classes, one for Elastix and one for Transformix, that define the external interface in target languages. This layer also hides the internal Elastix and Transformix filters and holds methods for passing data between SimpleITK and Elastix. The third layer is responsible for compiling the facade layer to target language libraries. It consists of an extended version of SimpleITK's build infrastructure that is able to pass C++ template information from SimpleITK's factory methods to Elastix' factory methods which are responsible for runtime instantiation of internal C++ registration algorithms.

In the following sections we briefly outline the registration formulation in Elastix and then describe the three API layers in detail.

2.1. Elastix

The Elastix library is a collection of medical image registration algorithms that facilitates research on medical image registration by allowing the researcher to quickly compare and tune different registration methods for domain-specific applications [19]. Elastix has been applied to many problems including multi-atlas segmentation of CT and MRI images [1], early diagnosis of dementia [6], estimation of pulmonary emphysema [25], groupwise registration [22], 2D-3D registration in x-ray guided interventions [26], ra-

```
#include "itkImageFileReader.h"
#include "itkImageFileWriter.h"
#include "itkElastixFilter.h"
#include "itkTransformixFilter.h"

elastixFilter->SetFixedImage(
  fixedImageReader->GetOutput()
);

elastixFilter->SetMovingImage(
  movingImageReader->GetOutput()
);

transformixFilter->SetInputImage(
  movingLabelReader->GetOutput()
);

transformixFilter->SetParameterObject(
  elastixFilter->
    GetTransformParameterObject()
);

resultImageWriter->SetInput(
  transformixFilter->GetOutput()
);

resultImageWriter->Update();
```

Listing 2: The C++ filter interfaces in a typical pipeline setup. Typedefs and input/output filenames omitted for brevity.

diotherapy treatment planning [20], liver segmentation [7], cervical segmentation with shape regularization [4], automatic localization of breast cancer in DCE-MRI [12] and brain segmentation [27].

Elastix treats the registration problem as the process of transforming a moving image $I_M(\mathbf{x})$ to a fixed image $I_F(\mathbf{x})$ by finding a coordinate transformation $T(\mathbf{x})$ that makes $I_M(T(\mathbf{x}))$ spatially aligned with $I_F(\mathbf{x})$. A transformation model $T_\mu(\mathbf{x})$ is defined, with parameters $\boldsymbol{\mu}$, and the registration problem is formulated as an optimization problem in which the cost function $\mathcal{C}$ is minimized with respect to $\boldsymbol{\mu}$:

$$\hat{\boldsymbol{\mu}} = \arg\min_{\boldsymbol{\mu}} \mathcal{C}\left(T_\mu; I_F; I_M\right) \tag{1}$$

The minization is solved by an iterative optimization method embedded in a multi-resolution setting.

Elastix supports many choices for cost functions, regularization terms, transformations, optimization methods, interpolators, samplers, image pyramids and multi-resolution strategies. The reader is referred to [19] for a complete overview.

2.2. Elastix And Transformix Filters

The first layer on top of Elastix and Transformix consists of a C++ API based on the ITK filter paradigm. The Elastix filter takes ITK images, masks and a parameter data objects as inputs, performs registration, and provides result images and transform parameter data objects as outputs. The Transformix filter can apply transform parameter data objects to

```
(NumberOfResolutions 4)
(Transform "BSplineTransform")
(GridSpacingSchedule 8.0 4.4 2.0 1.0)
(Interpolator "LinearInterpolator")
```

Listing 3: Text-based parameter file.

```
p = sitk.ParameterMap()
p["NumberOfResolutions"] = ["4"]
p["Transform"] = ["BSplineTransform"]
p["GridSpacingSchedule"] = ["8.0", "4.0", "2.0", "1.0"]
p["Interpolator"] = ["LinearInterpolator"]
```

Listing 4: Python parameters.

other images and point sets. It can also write the deformation field and transform Jacobian to disk. Advanced users can integrate this layer directly into ITK image processing pipelines and treat Elastix and Transformix as regular ITK filters as shown in Listing 2.

The previous Elastix and Transformix C++ API exposed only a subset of the functionality available via the command-line interface. These filters bring the C++ API on par with the command-line interface and allow SimpleElastix to support all components, methods and settings available in Elastix.

The user configures the registration procedure via the parameter data object. The parameter data object is an ITK data object that holds registration parameters in the form of key-value pairs that atomically define the registration method, its components, and any settings it might require. Previously, a user would manually enter parameters as text strings into a parameter file as shown in Listing 3, save the parameter file to disk and pass the path to Elastix via the command-line. This allows the user to configure Elastix without any C++ programming knowledge. However, the text processing adds an additional layer of complexity. Listing 4 shows the Python interface to the parameter data object. Using this approach, the parameter file is an in-memory data structure that can be programmatically manipulated as if it was a native data type in the target language. This eliminates the need for intermediate disk I/O and reduces bookkeeping code. It also allows multiple registrations to be run in parallel without having to take into account disk access.

2.3. Elastix And Transformix Facade Interface

SimpleITK uses a facade design pattern to simplify the interface to ITK algorithms. Each algorithm has a corresponding facade class that provides a minimum viable interface for configuring and running the algorithms, thus hiding the internal instantiations, templates and pipeline mechanisms. For SimpleElastix, facade classes were developed that hold methods for controlling input and output and dis-

```cpp
#include "SimpleITK.h"

namespace sitk = itk::simple;

// Functional-style interface
sitk::Image resultImage = sitk::Elastix(
  fixedImage,
  movingImage
)

// Object-oriented interface
sitk::SimpleElastix selx;
selx.SetFixedImage(fixedImage);
selx.SetMovingImage(movingImage);
sitk::Image resultImage = selx.Execute();
```

Listing 5: The C++ facade interfaces.

patching Elastix and Transformix filters when registration is executed. The facade layer is also responsible for converting SimpleITK data structures into native ITK objects, pass them to the filters and convert Elastix output to SimpleITK format.

Following the design conventions of SimpleITK, we developed a functional-style interface designed for rapid prototyping and an object-oriented facade interface suited for more advanced use cases and scripting purposes. The functional-style interface trades off flexibility for code simplicity and allows images to be registered with a single line of code as shown in Listing 1. The object-oriented interface is more flexible and allows for subsequent warping of points, meshes, segmentations, and automatic computation of inverse transforms as shown in Listing 7 and 8. Listing 5 shows the C++ facade interfaces.

In the experiments section we show how the functional-style interface, the object-oriented interface and SimpeITK's image processing algorithms can be used together in synergy.

2.4. SimpleITK Extensions

To compile Elastix and Transformix facade classes to target languages, SimpleITK was extended in a number of ways.

- The SimpleITK build system was enhanced to automatically download and compile Elastix.
- SimpleITK's type system was enhanced to be able to provide type information to Elastix' factory mechanism.
- The Elastix- and Transformix facade classes were developed and registered with SimpleITK's factory mechanism.
- Support for parameter maps was added to SimpleITK's code generation tools.

```
$ git clone https://github.com/kaspermarstal/
    SimpleElastix
$ mkdir build && cd build
$ cmake ../SimpleElastix/SuperBuild
$ make -j4
$ cd SimpleITK-build/Wrapping/PythonPackage
$ sudo python setup.py install
```

Listing 6: Installation procedure on Linux.

- Support for 4D images was added in order to support groupwise registration of 3D images with Elastix [22, 13].

With these extensions in place and the Elastix and Transformix filters fully encapsulated in facade classes, SimpleITK's code generation tools can compile Elastix and Transformix for the target languages. SimpleITK relies on SWIG [3] (Simplified Wrapper and Interface Generator) for generating target language binding code. SWIG inspects the C++ facade header files and automatically generates scripting language interfaces from these declarations.

Users can download and install SimpleElastix by following the commands in Listing 6. The build script will automatically detect available target languages on the host system and build packages for these languages.

2.5. Online Resources

SimpleElastix is developed as an open source project on Github. Here, community members can post issues and contribute code via pull requests (PRs). A PR is a unit of code that anyone can submit for merging into the main codebase. Before a PR is merged, it is reviewed by SimpleElastix developers and automatically tested against a comprehensive test suite with over two thousand tests. A PR can be reviewed, edited and merged via the GitHub web interface once everyone is satisfied with the changes.

The documentation includes installation instructions for Linux, Mac and Windows, a general introduction to image registration, how-to guides for using Elastix, and several examples for different registration methods.

The documentation is typeset using the Sphinx documentation system [5] and hosted online by Read the Docs [15]. The text files reside in the main repository so users can contribute to the documentation in the same way they contribute code. When a PR is merged into SimpleElastix, Read the Docs will download the repository, rebuild documentation and update the documentation website and the accompanying PDF file. The documentation is available at https://simpleelastix.readthedocs.io. In the next section, the proposed software package is tested on two different data sets.

```
import SimpleITK as sitk
import numpy as np
import os

# Load fixed image and associated segmentation
fixedImage = sitk.ReadImage("fi.nii.gz")
fixedLabel = sitk.ReadImage("fl.nii.gz")

# Turn the different brain regions into one
fixedBinaryLabel = sitk.Threshold(fixedLabel, 1.0)

selx = sitk.SimpleElastix()
selx.SetFixedImage(fixedImage)
overlapFilter = sitk.LabelOverlapMeasuresImageFilter()

images = os.listdir("images/")
labels = os.listdir("labels/")
totalDSC = []
meanRegionDSC.append = []
for image, label in zip(images, labels)
  movingImage = sitk.ReadImage(image)
  movingLabel = sitk.ReadImage(label)

  # Run registration
  selx.SetMovingImage(movingImage)
  selx.Execute()

  # Set interpolation scheme for labels
  tp = selx.GetTransformParameterMap()
  tp["ResampeInterpolator"] =
    ["FinalNearestNeighborInterpolator"]

  # Get mean DSC of brain regions
  resultLabel = sitk.Transformix(movingLabel, tp)
  overlapFilter.Execute(fixedLabel, resultLabel)
  totalDSC.append(overlapFilter.GetDiceCoefficient())

  # Get total brain volume DSC
  resultBinaryLabel = sitk.BinaryThreshold(
    resultLabel, 1.0)
  overlapFilter.Execute(
    fixedBinaryLabel, resultBinaryLabel)
  meanRegionDSC.append(
    overlapFilter.GetDiceCoefficient())

print np.mean(totalDSC), np.std(totalDSC)
print np.mean(meanRegionDSC), np.std(meanRegionDSC)
```

Listing 7: Experimental setup for registration of brains in Python.

```
import SimpleITK as sitk
import numpy as np
import os

fixedImage = sitk.ReadImage("face1.png")
fixedPoints = np.genfromtxt("face1.pts", skip_header=2)

selx = sitk.SimpleElastix()
stfx = sitk.SimpleTransformix()

selx.SetFixedImage(fixedImage)
selx.SetFixedPointSetFileName("face1.pts")

# Add the point metric
selx.AddParameter("Metric",
  "CorrespondingPointsEuclideanDistanceMetric")

images = os.listdir("images/")
points = os.listdir("pointsets/")
distances = []
for image, pointset in zip(images, points)
  movingImage = sitk.ReadImage(image)

  # Run registration
  selx.SetMovingImage(movingImage)
  selx.SetMovingPointSetFileName(pointset)
  selx.Execute()

  # Compute inverse transformation
  selx.ExecuteInverse()

  # Warp points
  stfx.SetPointSetFileName(points)
  stfx.SetTransformParameterMap(
    selx.GetInverseTransformParameterMap())
  stfx.Execute()

  # Get point distances
  outputPoints = np.genfromtxt("outputpoints.txt")
  resultPoints = outputPoints[:,(27,28)]
  distances.append(np.linalg.norm(fixedPoints-
    resultPoints))

print np.mean(distances), np.std(distances)
```

Listing 8: Experimental setup for registration of images of faces in Python.

3. Experiments

SimpleElastix can be used for many types of registration problems. Here, we present an intensity-based registration of 3D MR images and a point-based registration of 2D images faces. For both experiments, images are registered using a preconfigured registration method included in the package. The method and its parameters have been chosen conservatively and favor robustness over speed. The experiments are implemented in Python but any of the other supported languages could be used. Note that Listing 7 and 8 constitute the complete experimental setup used for this paper.

The preconfigured registration method consists of a Mutual Information (MI) similarity metric, a Transform Bending Energy Penalty metric, an Adaptive Stochastic Gradient Descent (ASGD) [19] optimizer and a sequence of transformation models of increasing complexity. First, a translation transform is applied followed by an euler- (translation, rotatation), affine- (translation, rotation, scaling, shearing) and B-spline transform. For each transform a multi-resolution strategy is used with four levels of image smoothing. The B-spline transform additionally employs four resolutions of B-spline knots. The grid spacing is halved in each subsequent resolution, resulting in a knot spacing of 10 mm in the final resolution. It takes around half a minute to register a typical 3D volume with this procedure.

3.1. Brain Registration

For this experiment we used 30 adult brain atlases available at brain-development.org [14, 11]. We loop over the population, register, warp associated segmentation and compute the overlap as shown in Listing 7. An example can be seen in Figure 1.

The experiment illustrates advantages of combining the

```
require 'SimpleITK'
fixed = SimpleITK.ReadImage('fixed.nii')
moving = SimpleITK.ReadImage('moving.nii')
result = SimpleITK.Elastix(fixed, moving)
```

Listing 9: Rapid prototyping in Lua.

```
library(SimpleITK)
fixed <- ReadImage('fixed.nii', 'sitkFloat32')
moving <- ReadImage('moving.nii', 'sitkFloat32')
result <- Elastix(fixed, moving)
```

Listing 10: Rapid prototyping in R.

```
import org.itk.simple.*
List<Image> movingImages
  = Arrays.asList(ReadImage("image1.nii"),
    ReadImage("image2.nii"));
movingImages
  .parallelStream()
    .forEach((moving) -> Elastix(fixed, moving));
```

Listing 11: Parallel processing in Java.

```
using itk.simple;
List<Image> movingImages
  = new List<Image> {ReadImage("image1.nii"),
    ReadImage("image2.nii")};
Parallel
  .ForEach(movingImages, moving =>
    Elastix(fixed, moving));
```

Listing 12: Parallel processing in C#.

native Python methods with the object-oriented interface, the functional-style interface and SimpleITK's image processing algorithms. The object-oriented interface facilitates re-use of the fixed image and registration parameters across the entire population. This reduces bookkeeping code and file handling. The transform parameters are edited with a Python-idiomatic interface so the user does not have to incorporate text processing or manually editing parameter text files, as is the case with Elastix. The procedural interface allows segmentations to be transformed with a single line of code. Lastly, the researcher can use the NumPy library to work with the result DSCs.

The total brain volume DSC in Table 1 shows that the preconfigured method can get new users up and running quickly or serve as a starting point for expert domain-specific tuning. For example, it is straightforward to add an additional loop over a list of different B-spline grid spacings or wrap the registration script in a function and use dedicated hyperparameter optimization packages.

3.2. Face Registration

For this experiment we used 1513 images from the BioID database [17] each annotated with 19 points as shown in Figure 3. The images are converted to PNG format and the point sets are converted to Elastix' PTS format. The population is then registered to a fixed reference image using the same approach as above with the addition of a point set similarity metric as shown in Listing 8. The point similarity metric acts as the main driving force in the registration while mutual information and a transform bending energy penalty regularize the registration between points. The metrics are weighted equally by default. The user could force the points to be fully aligned by adjusting the relative weights of the metrics.

Experiment	Result
Hammersmith83 (total)	0.92 ± 0.02 DSC
Hammersmith83 (regions)	0.72 ± 0.06 DSC
BioID	1.74 ± 2.25 pixels

Table 1: Results for registration of brains from the Hammersmith83 data set and registration of images of faces from the BioID data set.

To illustrate how the inverse transform can be obtained, we measure the registration error in the fixed reference frame. Elastix defines the transform from the fixed image to the moving image, so the inverse transform is needed to warp points from the moving image to the fixed image. This is done using the `ExecuteInverse` function. The result is calculated as the mean distance between corresponding points after registration and shown in Table 1. The misalignment is fairly small compared to the image size of size 384×286 pixels (width $\times$ height).

4. Discussion

Image registration is a prerequisite for a wide range of medical image analysis problems, but state-of-the-art methods are typically not made available or difficult to use. We have addressed this issue with the development of the SimpleElastix software package. The experiments demonstrates how to combine the object-oriented interface, the procedural interface and SimpleITK's existing image processing algorithms. Previously, a user would manually enter parameters as text strings in parameter files, save the files to disk and pass image and parameter file paths to Elastix and Transformix via the command-line. With the proposed software package, the command-line interface and text-based parameter files are superseded by native libraries and idiomatic data structures in target languages, requiring less boilerplate code and reduced disk I/O.

Bringing Elastix other programming languages allow users to call Elastix from environments suited for their particular use case. For example, the Python, Lua, and R interfaces facilitate rapid prototyping as shown in Listing 1, 9,

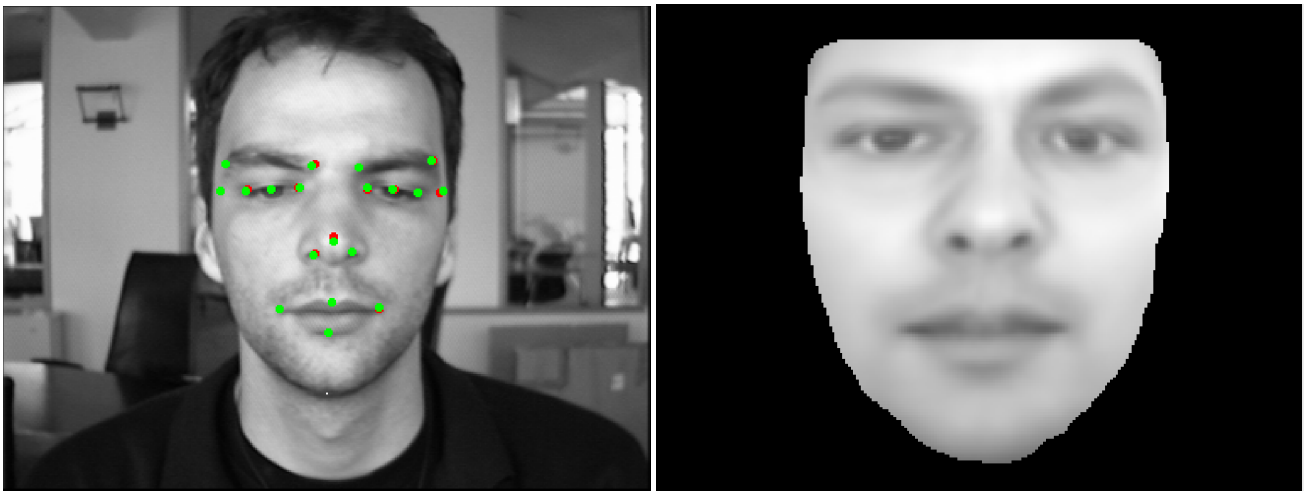

Figure 3: Example of two registered faces from the BioID dataset (left) and the mean of 1513 registered faces (right).

and 10, and the Java and C# interfaces provide langauge-specific constructs for easily processing many images in parallel as shown in Listing 11 and 12. It should be noted that it is still the responsibility of the user to select registration methods and appropriate parameters for a particular application.

SimpleElastix can also be used to automatically transfer additional or new registration functionality to supported languages. For example, a researcher developing a new registration metric can extend Elastix itself, after which SimpleElastix can be rebuilt with the extended version of Elastix. The new metric is then automatically available in all target languages.

Accessibility, extensibility and reproducibility has been the goal of SimpleElastix since development began. The development process is online, transparent and encourages contributions from users from all around the world. The source code is freely distributed under a license that allows anyone to use the code for any purpose.

Since SimpleElastix is based on SimpleITK and Elastix, both the underlying build infrastructure and registration algorithms are maintained upstream. Future work will therefore focus on ease of adoption by the user community. We will make binaries and target language packages available in order to remove the initial hurdle of compiling the software and provide updates as new versions of SimpleITK and Elastix are released.

5. Conclusion

We have developed a medical image registration software package that is easy to use in a wide variety of programming languages. We have applied the method to two registration problems and shown how to setup experiments using minimal amount of code. The software package, examples and documentation are available online.

6. Acknowledgement

The authors would like to acknowledge Bradley Lowekamp and fellow developers of SimpleITK without whom this work would not have been possible. The work was supported by The Dutch Technology Foundation (Stichting Technische Wetenschappen), grant number 13351, and the Netherlands Organisation for Scientific Research NWO, grant number 184033111.

References

[1] X. Artaechevarria, A. Munoz-Barrutia, and C. Ortiz-de Solorzano. Combination strategies in multi-atlas image segmentation: Application to brain MR data. *IEEE Transactions on Medical Imaging*, 28(8):1266–1277, 2009.

[2] R. Beare, R. Beare, G. Lehmann, and G. Lehmann. The watershed transform in ITK - discussion and new developments. *Insight journal*, pages 1–24, 2006.

[3] D. Beazley. SWIG. *http://www.swig.org/*.

[4] F. F. Berendsen, U. A. Van Der Heide, T. R. Langerak, A. N. T. J. Kotte, and J. P. W. Pluim. Free-form image registration regularized by a statistical shape model: Application to organ segmentation in cervical MR. *Computer Vision and Image Understanding*, 117(9):1119–1127, 2013.

[5] G. Brandl. Sphinx. *http://www.sphinx-doc.org/*.

[6] E. E. Bron, R. M. E. Steketee, G. C. Houston, R. A. Oliver, H. C. Achterberg, M. Loog, J. C. van Swieten, A. Hammers, W. J. Niessen, M. Smits, and S. Klein. Diagnostic classification of arterial spin labeling and structural MRI in presenile early stage dementia. *Human Brain Mapping*, 35(9):4916–4931, 2014.

[7] P. Campadelli, E. Casiraghi, and A. Esposito. Liver segmentation from computed tomography scans: A survey and a new algorithm. *Artificial Intelligence in Medicine*, 45(2-3):185–196, 2009.

[8] A. Coron, J. Mamou, E. Feleppa, and P. Laugier. ElastixFromMatlab: Register image with Elastix within Matlab. *https://sourcesup.renater.fr/elxfrommatlab/*, 2012.

[9] E. Freeman, E. Robson, B. Bates, and K. Sierra. *Head first design patterns*. " O'Reilly Media, Inc.", 2004.

[10] 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. *Frontiers in Neuroinformatics*, 5(13), 2011.

[11] I. S. Gousias, D. Rueckert, R. A. Heckemann, L. E. Dyet, J. P. Boardman, A. D. Edwards, and A. Hammers. Automatic segmentation of brain MRIs of 2-year-olds into 83 regions of interest. *Neuroimage*, 40(2):672–684, 2008.

[12] A. Gubern-Mérida, R. Martí, J. Melendez, J. L. Hauth, R. M. Mann, N. Karssemeijer, and B. Platel. Automated localization of breast cancer in DCE-MRI. *Medical Image Analysis*, 20(1):265–274, 2015.

[13] J.-M. Guyader, W. Huizinga, V. Fortunati, J. F. Veenland, M. M. Paulides, W. J. Niessen, and S. Klein. Groupwise image registration of multimodal head-and-neck images. In *Biomedical Imaging (ISBI), 2015 IEEE 12th International Symposium on*, pages 730–733. IEEE, 2015.

[14] A. Hammers, R. Allom, M. J. Koepp, S. L. Free, R. Myers, L. Lemieux, T. N. Mitchell, D. J. Brooks, and J. S. Duncan. Three-dimensional maximum probability atlas of the human brain, with particular reference to the temporal lobe. *Human Brain Mapping*, 19(4):224–247, 2003.

[15] E. Holscher, C. Leifer, and B. Grace. Read the Docs. *https://readthedocs.org/*.

[16] D. C. Ince, L. Hatton, and J. Graham-Cumming. The case for open computer programs. *Nature*, 482(7386):485–8, 2012.

[17] O. Jesorsky, K. J. Kirchberg, and R. W. Frischholz. Robust Face Detection Using the Hausdorff Distance. *In Proc. Third International Conference on Audio- and Video-based Biometric Person Authentication*, (June):90–95, 2001.

[18] H. J. Johnson, M. M. McCormick, and L. Ibanez. *The ITK Software Guide Book 1: Introduction and Development Guidelines-Volume 1*.

[19] S. Klein, M. Staring, K. Murphy, M. A. Viergever, and J. P. W. Pluim. Elastix: A toolbox for intensity-based medical image registration. *IEEE Transactions on Medical Imaging*, 29(1):196–205, 2010.

[20] S. Leibfarth, D. Mönnich, S. Welz, C. Seigel, N. Schwenzer, H. Schmidt, D. Zips, and D. Thorwarth. A strategy for multimodal deformable image registration to integrate PET/MR into radiotherapy treatment planning. *Acta Oncologica*, 52(October 2015):1353–1359, 2013.

[21] B. C. Lowekamp, D. T. Chen, L. Ibáñez, and D. Blezek. The Design of SimpleITK. *Frontiers in neuroinformatics*, 7(December):45, 2013.

[22] C. T. Metz, S. Klein, M. Schaap, T. van Walsum, and W. J. Niessen. Nonrigid registration of dynamic medical imaging data using nD + t B-splines and a groupwise optimization approach. *Med Image Anal*, 15(2):238–249, 2011.

[23] D. Mueller. ManagedITK: .NET Wrappers for ITK. 2007.

[24] T. Poisot. Best publishing practices to improve user confidence in scientific software. *Ideas in Ecology and Evolution*, 8(1), 2015.

[25] M. Staring, M. E. Bakker, J. Stolk, D. P. Shamonin, J. H. C. Reiber, and B. C. Stoel. Towards local progression estimation of pulmonary emphysema using CT. *Medical physics*, 41(February):021905, 2014.

[26] I. M. J. van der Bom, S. Klein, M. Staring, R. Homan, L. W. Bartels, and J. P. W. Pluim. Evaluation of optimization methods for intensity-based 2D-3D registration in x-ray guided interventions. *Proceedings of SPIE*, 7962(1):796223–796223–15, 2011.

[27] F. Van Der Lijn, M. De Bruijne, S. Klein, T. Den Heijer, Y. Y. Hoogendam, A. Van Der Lugt, M. M. B. Breteler, and W. J. Niessen. Automated brain structure segmentation based on atlas registration and appearance models. *IEEE Transactions on Medical Imaging*, 31(2):276–286, 2012.