`lcreg` 0.1.2 — Documentation

Peter Rösch*

October 2019

Abstract

The software described in this document is concerned with the rigid and affine registration of large 3D images. The differentiating feature of `lcreg` is its capacity to efficiently register image data sets that do not fit into system memory.

*Peter.Roesch@hs-augsburg.de

Contents

1 3D Image Registration with lcreg

Registration of images allows for the combination of complementary information contained in different images and is frequently used as a preprocessing step in medical image processing applications [1]. The aim of this documentation is to enable the reader to efficiently apply the 3D image registration package lcreg [2]. Thus, this document focuses on practical aspects. The theoretical background of medical image registration in general and lcreg in particular can be found in the literature compiled in the references section.
After a brief overview of the lcreg approach, section 2 describes the installation of the software followed by a detailed discussion of parameter settings. Finally, a detailed hands-on tutorial guides the reader through the practical solution of typical registration tasks using a selection of example images and configuration files accompanying this document.

1.1 Distinguishing features of lcreg

Advances in imaging technology, in particular improved spatial resolution, lead to ever increasing image sizes. For example, high resolution nano CT data sets can reach sizes above 30 gigabytes. As most current image processing libraries require images to fit into random access memory (RAM), pairwise registration of theses images is not possible on standard desktop computers. With the design and implementation of lcreg, resource efficiency of 3D image registration could be significantly improved so that very large images can now be processed in tens of minutes even on notebook computers. This could be achieved by making use of recent advances in both hardware and software technology, namely the availability of high speed solid state disks (SSD) and fast on the fly compression software. These building blocks have been combined with image processing algorithms specifically tailored and optimised for the task at hand. More details concerning design and implementation are given in section 1.3.

1.2 Processing steps

Image registration with lcreg involves the following processing steps:

Image format conversion: Images need to be converted into the meta image (.mhd) format. This can be achieved using e.g. c3d [3].

Image import: Input images are converted to compressed data sets on disk based on bcolz arrays. Note that input images are processed slice by slice so that images not fitting into memory can be converted.

Creation of multi resolution representation: In order to improve the performance with respect to both speed and robustness, images and masks (see below) are resampled to lower resolutions where intensities of the low-resolution image voxels are calculated by averaging the grey values in the corresponding region of the high-resolution image.

Image masking: In order to exclude potentially disturbing structures like sample holders, the corresponding image area can be excluded from the registration process by providing binary masks.

Initialisation: Starting estimates for iterative optimisation can either be determined interactively using e.g. the manual registration functionality of ITK-SNAP or determined from the moments of inertia with respect to image edges. Furthermore, the identity transform can be used as a starting estimate.

Optimisation: Optimisation of the similarity measure with respect to the transformation is performed using a multi resolution approach [1] where registration results obtained for a particular resolution are used as starting estimate for the next higher resolution. Two different optimisation strategies are applied. For moderate initial displacements, the method of choice is Levenberg-Marquardt optimisation which has been applied previously for elastic registration [4] based on local correlation similarity measure [5] which is also used in `lcreg`. However, for larger initial displacements, a slow but robust gradient descent approach [6] can optionally be applied at the lowest resolution level.

Export of results: After registration, the 4×4 homogeneous transformation matrix representing the registration result is stored on disk. Optionally, the moving image can be resampled into the space of the fixed image thus allowing a voxel by voxel comparison. For validation purposes using the RIRE approach, output files according to the RIRE standard can be created.

Verification: Using the external tool ITK-SNAP, registration success can be assessed by overlaying the registered moving image to the fixed image or by investigating difference images. Furthermore, the image parts actually used for registration can be visualised in order to improve parameter settings in case of misregistration.

Cleanup: Finally, intermediate images data can be deleted

1.3 Building blocks and implementation

Three principles governed the implementation process of the `lcreg` application:

1. Performance-critical image processing operations are implemented from scratch using an optimising static compiler.
2. The core application `lcreg` does not require any input or intermediate image to fit into Random Access Memory (RAM) of the computer.
3. Rather than re-implementing common operations, existing libraries are used.
4. The number of dependencies on external libraries is kept to a minimum.

From these principles, the following decisions have been derived:

1. Low-level operations like interpolation, resampling, masking and similarity measure calculation have been implemented using cython [7]. Wherever reasonable, parallelisation via multiple threads has been applied. Optimised cython functions are aggregated into a single shared library.

2. The highly efficient bcolz [8] extension is used for on-the-fly compression and decompression of on-disk image data. Compressed on-disk arrays are accessed block by block analogous to `numpy` arrays.
3. Handling of transformations, in particular conversion between matrix and quaternion representation, is performed using the module `scipy.spatial.transform`.
4. For the sake of reproducibility, parameters are passed to `lcreg` by means of a configuration file which can be modified using a standard text editor or created automatically within a batch processing setup.
5. `scipy` [9] is used to reformat the moving image into the voxel space of the fixed image after registration. As `numpy` supports memory-mapped files via the `numpy.memmap` functionality, images need not to fit into physical RAM for this processing step.
6. In summary, `lcreg` depends on four external libraries, namely `numpy`, `scipy`, `bcolz` and `psutil`. Optionally, `py-cpuinfo` can be installed in order to incorporate CPU information into the log file.

1.4 Command line executables

The package contains the following executables:

`lcreg`: Start image registration using settings from the given parameter file.
`usage: lcreg parameter_file_name`

`lcreg_profile`: Start `lcreg` in profiling mode. Profiling resutls are printet to `stdout`.

`abs_difference_mhd`: Calculate the absolute grey value difference between two images in `mhd` formt specified by their names `im1_name` and `im2_name`. The result is stored in an image named `out_im_name`. Image dimensions need to agree.
`usage: abs_difference_mhd im1_name im2_name out_im_name`

`view_compressed_images`: Uncompress images stored in the internal format and use `ITK-SNAP` to visualise them.
The first parameter specifies a directory where temporary files, i.e. uncompressed images, are stored. Note that enough disk space to hold these uncompressed images needs to be available.
The parameters `path_name_1` ... `path_name_N` specify the paths of the `bcolz` directories containing compressed data.
`usage: view_compressed_images tmp_prefix path_name_1  path_name_2 ...`

`isq_to_mhd` Convert images from the ISQ format into the meta image format.
`usage: isq_to_mhd isq_image_name mhd_image_name`

2 Installation

2.1 Installing lcreg

Binary distributions of the software are available for the 64 bit versions of linux (64 bit), windows (64 bit) and Mac OS X. Using the source distribution, the software can be automatically installed on other operating systems where a C++ compiler is required to create the shared library containing optimised functions created from `cython` code.
`lcreg` depends on the following Python packages:

```
numpy  version >= 1.1
scipy  version >= 1.1
bcolz  version >= 1.2
psutil version >= 5.4
```

Optionally, the package `py-cpuinfo` (version ≥ 4) can be installed to allow for a more detailed description of system properties in the log file. Development and testing has been performed with the Anaconda distribution Using Anaconda, the dependencies can be installed with the following command:

```
conda install numpy scipy bcolz psutil py-cpuinfo
```

The `lcreg` package itself installed by typing

```
pip install lcreg
```

for Python (CPython) versions 3.6 and above.

2.2 Additional software: ITK-SNAP and c3d

ITK-SNAP allows for the interactive comparison of image pairs before and after registration. Furthermore, the manual registration mode can be applied to create text files containing homogeneous transformation matrices which can be imported from `lcreg` and used as starting estimates. The command line tool `c3d` which is contained in the `ITK-SNAP` distribution provides functionality for image format conversion and image reformatting. In particular, various image formats can be converted in `mhd` images required by `lcreg`. After registration, `c3d` can be applied to reformat the moving image using higher interpolation orders (e.g. cubic) rather than linear interpolation applied within `lcreg`.
Installers for `ITK-SNAP` (Version 3.8 and above) including the `c3d` command line tool can be found at the download page. The executables `c3d` and `itksnap` (Linux) or `ITK-SNAP` (Windows) should be part of the user or system search path (Linux: environment variable `$PATH`, Windows: Advanced system settings).

3 Configuration Files – Sections and Options

The command line tool takes exactly one command line argument specifying the name of configuration file in INI format, for example

```
lcreg post_to_pre.ini
```

3.1 DEFAULT

This section contains general registration settings.
Typical example taken from the tutorials:

```
[DEFAULT]
    ini_version = 1
    image_directory = ../mhd
    fixed_image_name = ${image_directory}/pre.mhd
    moving_image_name = ${image_directory}/post.mhd
    fixed_mask_image_name = None
    moving_mask_image_name = None
    working_directory_name = ../lcreg/endo
    output_directory_name = ${working_directory_name}/results
    reuse_existing_input_images = False
    log_level = DEBUG
    mem_limit_mb = -1
    nr_of_threads = -1
    remove_temporary_images = False
```

ini_version: `lcreg` ini file version, current setting: 1.

image_directory: Absolute or relative path to the directory containing image data. The value can be used as a file name prefix (see below).

fixed_image_name: Name of the fixed image file including suffix. Previously defined variables can be used applying the `${variable_name}` syntax.

moving_image_name: File name of the moving image which is transformed to match the fixed image.

fixed_mask_image_name: Name of binary mask image file for the fixed image or `None` for no masking.

moving_mask_image_name: Name of binary mask image file for the moving image or `None` for no masking.

working_directory_name: Relative or absolute path to store compressed image data. If the directory does not exist it will be created by `lcreg`.

output_directory_name: Directory where files representing registration results, i.e. transformation matrix, resampled moving image etc. should be stored.

reuse_existing_input_images: If set to `True`, existing compressed input files will be re-used rather than overwritten.

log_level: Level for the python logging system. Possible Values are `CRITICAL, ERROR, WARNING, INFO or DEBUG`. Logging output is appended to ${output_directory_name}/lcreg.log.
Recommendation: `INFO`
Setting for the tutorials: `DEBUG` to get more details for result assessment.

mem_limit_mb: Amount of RAM in MiB to be used for image registration. If the parameter is set to -1, all available RAM will be used.
Setting for the tutorials: -1 to use all memory available.

nr_of_threads: Number of threads to be launched for image processing. If the parameter is set to -1, all available CPU threads are used.
Setting for the tutorials: -1 to use all threads.

remove_temporary_images: If set to `True`, intermediate files, in particular resampled and masked image files at different resolutions, will be automatically removed after registration. Note that some registration assessment tools e.g. visualisation of masked images at different resolution levels can not be applied if this flag is set to `True`.
Recommendation: `False` (if sufficient disk space is available).
Setting for the tutorials: `False` to allow for result assessment.

3.2 transformation

This part defines the type of transformation to be optimised and the way the initial transformation is determined. If no initial transformation file is provided and initialisation from edge moments is disabled, the identity transform is used as starting estimate for optimisation.
Typical example taken from the tutorials:

```
[transformation]
    transformation_type = rigid
    initial_transformation_file_name = None
    initial_transform_is_RAS = True
    initial_transform_is_inverted = False
    store_as_RAS = True
    initialise_from_edge_moments = False
    initialise_shift_only = False
```

transformation_type: Type of transformation to be optimised. Possible Settings:
`rigid` (3D translation and rotation, 6 parameters) or
`affine` (3D translation, rotation, scale and shear, in total 12 parameters)
Setting for the tutorials: `rigid` except for the brain atlas example.

initial_transformation_file_name: Name of a text file containing a transformation in terms of a 4×4 transformation matrix in homogeneous coordinates or `None`.

initial_transform_is_RAS: The two most common anatomical coordinate systems are LPS and RAS (see e.g. 3D Slicer documentation). Internally, `lcreg` uses

LPS. However, according to the documentation, `c3d` uses the RAS frame. Thus, it is recommended to keep transformations stored on disk in the RAS system so that they can be read from `c3d`.
Recommended setting: `True`.
Setting for the tutorials: `True` so that the matrix can be used directly by `c3d`.

initial_transform_is_inverted: If set to `True`, the inverse of the matrix given in the text file is used as initial estimate.
Setting for the Tutorials: `False`.

store_as_RAS: See option `initial_transform_is_RAS`.
Recommended setting: `True`
Setting for the tutorials: `True`.

initialise_from_edge_moments: For large initial translations and rotations, the moments of inertia of the edge images can be used for initialisation. If not all anatomical structures contained in the fixed image are present in the moving image or vise versa, initialisation using edge moments can fail. In these cases, manual initialisation using ITK-SNAP as demonstrated in section 4.8 is preferable.
Setting for the tutorials: `False` except for the `caries` and `preparation_post_pre` examples.

initialise_shift_only: For small initial rotations it can be sufficient and mor stable (in particular for symmetrical structures) to use the centre of gravity with respect to image edges to initialise translation only.
Setting for the tutorials: `False` except for the preparation_post_pre example.

3.3 masking

In order to exclude irrelevant structures (e.g. sample holders) which might disturb registration, either a grey value range or a binary mask (see section 3.1) can be specified. As the local correlation similarity measure used in `lcrec` quantifies the alignment of corresponding edges, image voxels with relatively small values of the local absolute grey value gradient can be excluded from the registration process. This results in both an improved convergence behavior of similarity measure optimisation and a considerable speedup of similarity measure calculation as only a small fraction of voxels need to be taken into account.
Typical example:

```
[masking]
    fixed_grey_masking_range = 100 7500
    moving_grey_masking_range = None
    fixed_used_gradient_magnitude_fraction = 0.01
    moving_used_gradient_magnitude_fraction = 0.02
```

fixed_grey_masking_range: Lower and upper limit for fixed image thresholding prior to registration or `None`.
Setting for the tutorials: `None`.

moving_grey_masking_range: Lower and upper limit for moving image thresholding prior to registration or `None`.
Setting for the tutorials: `None`.

fixed_used_gradient_magnitude_fraction: Percentage of voxels of the fixed image to be used for registration. Voxels are sorted in decreasing order with respect to their local absolute gradient value. The given fraction corresponds to the percentile of voxels to be used for registration. Typical values range from 0.01 for high contrast μCT images to 0.2 for noisy MRT images.
Setting range in the tutorial: 0.01 – 0.05

moving_used_gradient_magnitude_fraction: Percentage of voxels of the moving image to be used for registration.
Setting range in the tutorial: 0.01 – 0.05

3.4 multiresolution

In order to increase the robustness of the optimisation process, registration starts with a low-resolution representation of fixed and moving image where the transformation is initialised by one of the methods described in section 3.2. For this purpose, a multi resolution pyramid of the images is created. Voxel size from level to level is either doubled or increased by a factor of $\sqrt{2}$. Note that the local correlation similarity measure requires fixed and moving image to have the same spatial resolution. Thus image resampling allowing for arbitrary scaling factors is required. Optionally, images can be resampled to isotropic resolution.
Typical example taken from the tutorials:

```
[multiresolution]
    pyramid_min_scale_factor = 1
    pyramid_sqrt2_scaling = True
    pyramid_cubic_voxels = False
    pyramid_min_voxel_nr = 64**3
```

pyramid_min_scale_factor: Scaling factor f_s to be applied for the highest resolution used during optimisation. Voxel spacing for the highest resolution level $\vec{s}_{\text{min}}$ is given by

$$\vec{s}_{\text{min}} = f_s \begin{pmatrix} \max(s_{\text{x,fixed}},\, s_{\text{x,moving}} \\ \max(s_{\text{y,fixed}},\, s_{\text{y,moving}} \\ \max(s_{\text{z,fixed}},\, s_{\text{z,moving}} \end{pmatrix}$$

For noisy images, this parameter can often be set to a value of 2 without significant loss of registration accuracy.
Setting for the tutorials: 1

pyramid_sqrt2_scaling: If set to `True`, the resolution change from level to level is $\sqrt{2}$, a setting of `False` results in a pyramid where voxel size is doubled from level to level. A setting of `False` is preferable unless images are noisy or the initial size of the images is small.
Settings for the tutorials: `True` or `False` depending on image properties.

pyramid_cubic_voxels: Resampling images to isotropic resolutions (setting: `True`) can speed up registration but tends to reduce accuracy as information in the directions with higher spatial resolution is lost.
Setting for the tutorials: `False`

pyramid_min_voxel_nr: Minimum number of image voxels in the multi resolution pyramid. Downsampling stops as soon as a resampled version of either fixed or moving images contains less voxels than the number specified here. The expression provided is evaluated and the result is taken as numerical parameter. Depending on original image sizes and on the percentage of the image volume relevant for registration, values between `32**3` and `128**3` have been successfully applied for a large set of test images.
Settings for the tutorials: `32**3, 64**3`

3.5 compression

Compression based on bcolz is one of the core components of `lcreg`. Image grey value data is converted from `mhd` raw files in an internal, compressed format controlled by parameters set in this section.
Typical example taken from the tutorials:

```
[compression]
    fixed_image_bits_per_pixel = 10
    moving_image_bits_per_pixel = 10
    cname = lz4
    clevel = 9
    shuffle = bcolz.BITSHUFFLE
```

fixed_image_bits_per_pixel: Number of bits to represent fixed image grey values. The lower the number specified here, the higher the compression factor. Note that reducing the number of bits per pixel increases quantisation error. Typical values are 10 for CT images and 8 for MRT images.

moving_image_bits_per_pixel: 10 Number of bits to represent fixed image grey values.
Setting for the tutorials: 10

cname: Name of the compressor to be used by bcolz. Possible options are described in the bcolz documentation and compared in the blosc tutorial.
Setting for the tutorials: `lz4`.

clevel: Compression level, possible values are in the range 0 . . . 9. Higher values result in smaller sizes of compressed image sizes on the cost of higher computation time for image conpression.
Setting for the tutorials: 9.

shuffle: Shuffling can improve the compression rate, see blosc tutorial.
Setting for the tutorials: `bcolz.BITSHUFFLE`.

3.6 levenberg_marquardt

The local correlation based similarity measure (1-LCC) used here allows for the analytical calculation of derivatives with respect to transformation parameters. Furthermore, 1-LCC is small in the vicinity of the optimum and exhibits a quadratic shape. Thus, the highly efficient Levenberg-Marquard optimisation scheme (LM) can be used.
Generally speaking, LM is the weighted combination of the two optimisation methods gradient descent and the Gauss-Newton algorithm where the relative weight of gradient descent is controlled by the parameter `lambda`. As gradient descent works well further away from the optimum but is inefficient close to the minimum of (1-LCC), `lambda` is decreased after each successful optimisation step thus increasing the relative weight of the Gauss-Newton algorithm which is very efficient as soon as the optimisation process approaches the minimum. If an optimisation step increases (1-LCC), `lambda` is increased and so is the influence of gradient descent. A detailed description of the procedure can be found in [11].

Typical example taken from the turorials:

```
[levenberg_marquardt]
    initial_lambda = 1
    lambda_upscale_factor = 10
    lambda_downscale_factor = 0.1
    max_nr_of_iterations = 50
    max_nr_of_upscale_steps = 4
    min_edge_voxel_displacement = 0.001
```

initial_lambda: Initial value for the parameter `lambda`.
Setting for the tutorials: 1.

lambda_upscale_factor: After an optimisation step which increased (1-LCC), the relative weight of gradient descent is increased by multiplying `lambda` with the factor defined here.
Settings for the tutorial: 10 for rigid, 1.1 for affine registration.

lambda_downscale_factor: 0.1 After a successful optimisation step which decreased (1-LCC), the relative weight of gradient descent is decreased by multiplying `lambda` with the factor defined here.
Settings for the tutorials: 0.1 for rigid, 0.9 for affine registration.

max_nr_of_iterations: Maximum number of iterations per resolution level. This termination condition stops the optimisation after a certain number of steps if none of the other two conditions described below is met.
Setting for the tutorial 50

max_nr_of_upscale_steps: Stop optimisation after a certain number of steps which increased (1-LCC) and thus `lambda`.
Setting for the tutorials: 4

min_edge_voxel_displacement: Parameter controlling the main termination criterion: Optimisation is stopped if the last update step which decreased (1-LCC) led to a negligible transformation update. A transformation update is considered negligible if the physical world position of image corners is modified by less than the specified fraction of a voxel diameter.
Setting for the tutorials: 0.01

3.7 simple_gradient

For large initial displacements LM optimisation might not converge. For these situations, a slow but robust simple gradient descent optimisation which modifies only one parameter per step [6] can be applied at the lowest resolution level prior to LM optimisation.
Typical example taken from the tutorials:

```
[simple_gradient]
    active = False
    delta_v_initial = 4
    delta_v_min = 0.5
```

active: Boolean variable switching simple gradient optimisation on or off.
Setting for the tutorials: `True` for examples `preparation, RIRE_training, veneer`.

delta_v_initial: Initial step size for simple gradient optimisation. If no further decrease of (1-LCC) can be achieved, simple gradient optimisation is continued with halved step size.
Setting for the tutorials: 4

delta_v_min: Minimum step size used in simple gradient optimisation in units of voxel sizes.
Setting for the tutorials: 0.25

3.8 output

This section specifies the output generated by `lcreg`.
Typical example taken from the tutorials:

```
[output]
    resample_moving_image = True
    store_RIRE_validation_file = False
    RIRE_header =
```

resample_moving_image: Boolean specifying whether the moving image should be resampled in the voxel space of the fixed image thus allowing for a comparison on voxel level or for calculation of difference images. If set to `True`, resampling

is performed using `scipy.ndimage`.
Setting for the tutorials: `True`.

store_RIRE_validation_file: If set to `True`, the transformation is written to a file that can be submitted to the Retrospective Image Registration Evaluation Project.
Setting for the tutorials: `True` only for the `RIRE_training` example.

RIRE_header: Header information written to the RIRE validation file. The header includes e.g. investigator, site and method information, see RIRE training data sets for templates.

4 Tutorial

In this section it is shown how typical registration tasks can be solved with `lcreg`. Specific parameter settings are derived from the properties of the actual registration problem.

4.1 How to benefit from this tutorial

This hands on tutorial aims at enabling the reader to practically solve registration tasks with `lcreg`. This requires the reader to actually apply the steps described in this documents using the provieded samples. Please note that just reading the tutorial is not effective as the interactive components (e.g. investigation of registration results) form an integral part of this approach.

4.2 Setup

Apart from the installation of `lcreg` (see section 2), the archive containing sample data sets and `lcreg` parameter files needs to be downloaded from `https://lcreg.de` and extracted. Free disk space of at least 1GB is required.

4.3 Conventions

Path and excecutable names are given in Linux style (e.g. `../endo/mhd`). For Windows, simply replace the path separator "`/`" by "`\`" and the command "`itksnap`" by "`ITK-SNAP`".
Furthermore, automatically generated scripts are named e.g. "`snap-reg.sh`" on operating systems and "`snap-reg.bat`" on Windows.

4.4 Compatibility testing

The software has been developed and tested using different versions of Python and the libraries mentioned in section 1.3. However, it can not be guaranteed that `lcreg` will work with future versions of these libraries. In order to test the compatibility of `lcreg` with new library versions, reference log files are provided for each example

discussed in the tutorial. In case of inaccuracies resulting from incompatibilities, the differences between log files created with new library versions and the reference log files can be useful to identify the origin of the problem.

4.5 Direct Levenberg-Marquard (LM) optimisation

If both the length of the translation vector and the rotation angles between fixed and moving image are relatively small i.e. within the capture range of LM optimisation, the identity transform can be used as starting estimate and simple gradient optimisation is not required. The configuration file used for the example described in this section is relatively simple. However, a profound understanding of these settings is important to master more complex registration tasks presented in the following sections.

4.5.1 Example `endo`

The image pair shows the root of a tooth before and after endodontic treatment. Isotropic resolution is 0.04 mm for both images. Image dimensions are $332 \times 283 \times 334$ voxels corresponding to 60 MiB per 16 bit integer image. As the sample had to be taken out of the μCT device to perform the procedure, the images are not aligned perfectly. `lcreg` is used to correct for the misalignment.

Visualising the images Prior to registration, the original images are overlaid using ITK-SNAP. Assuming that a command line interface (Linux-Terminal or Windows `cmd`) has been opened in the directory where the image data is stored (i.e. `lcreg_tutorial_data/endo/mhd`). On Linux systems, image visualisation is invoked by typing

```
itksnap -g pre.mhd -o post.mhd
```

As described in section 4.3, the corresponding Windows command is;

```
ITK-SNAP -g pre.mhd -o post.mhd
```

After loading the files, we automatically adjust contrast by typing `<ctrl> j` or via the menu `Tools` → `Image Contrast` → `Auto-Adjust Contrast`. Then, the display is switched to overlay mode by clicking on `post` in the Layer section with the right mouse button and choosing "Display as overlay". Finally, the relative weight of `pre` and `post` image can be changes with the `Opacity` slider (see figure 1). Varying the slider the following conclusions can be drawn:

- Tissue within the root canal has been removed in the course of treatment. This results in differences between the `pre` and `post` images. These differences are essential to assess the outcome of the procedure.
- Other differences in the vicinity of the outer root boundary result from repositioning inaccuracies. These differences should not be present any more after successful image registration.
- In this case, the images are shifted slightly with respect to each other. Apparently, no significant rotation between images is visible.

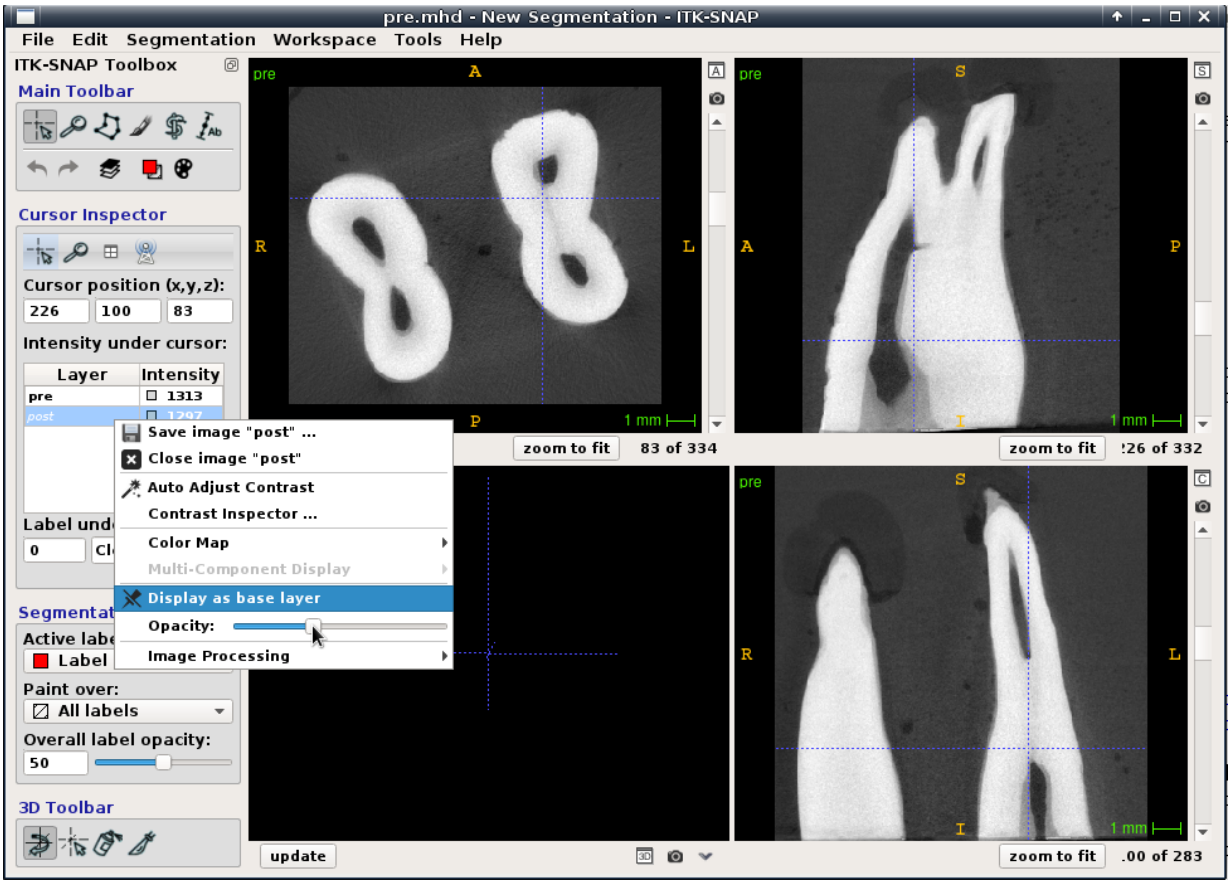

Figure 1: Overlay of endo pre and post images with ITK-SNAP

Setting up the parameter file In the parameter file `endo.ini` paths are specified relative to the `ini` directory so that `lcreg` has to be started from the `ini` directory. Note that this is not neccessary if absolute paths are given. The same holds for the working directory which is set to `../lcreg/endo` relative to the `ini` directory.

In the `[transformation]` section, rigid registration is selected. As both the options `initial_transformation_file_name` and `initialise_from_edge_moments` are set to `None`, the identity transform is used as initial estimate for iterative optimisation. In the `masking` section, no binary mask files are specified and the gradient magnitude fraction is set to 1 % (0.01) for both images. A value of 1 for the parameter `pyramid_min_scale_factor` in the `[multiresolution]` section make sure that the finest resolution used for registration corresponds to the original image resolution according to the description in section 3.4. The lowest resolution used for registration is selected such that the number of voxels in the corresponding image is above 64^3. Compression, optimisation and output settings correspond to the default values defined in section 3.

Starting lcreg The actual registration is started by typing

```
cd ../ini
lcreg endo.ini
```

Registration takes about 16 seconds on a notebook computer with a 2.6 GHz dual core Intel© Core© i5-3320M CPU.

Assessing registration results - Output files In order to examine registration results with command line tools, we change the working directory of our command line interpreter to the results directory
`lcreg_tutorial_data/endo/lcreg/endo/results`.
We notice that the following files have been created (note that Windows executable scripts have the suffix "`.bat`"):

lcreg_result.mat: Text file containing the transformation determined by `lcreg` in terms of a 4×4 matrix (homogeneous coordinates). Depending on the setting of the parameter `store_as_ras` in the `transformation` section, the matrix is stored either in the RAS or LPS coordinate frame.

moving_registered.mhd: This file which is only created for rigid registration is identical to the `.mhd` file of the moving image except for the `Offset` and `TransformMatrix` entries which have been subjected to the inverse of the transformation determined by the registration.
As `ITK-SNAP` takes into account image origin and orientation, an overlay of the original fixed image and the image represented by `moving_registered.mhd` should present correct overlays of corresponding anatomical structures.

lcreg.log: log file documenting individual steps of the registration process where individual entries contain e.g. a time stamp and the logging level.

snap_unreg.sh (snap_unreg.bat): Executing this script will load the unregistered image pair into ITK-SNAP and display them. For the `endo` example, the output corresponds to fig. 1 if images are displayed as overlay.

snap_reg.sh (snap_reg.bat): Executing this script will load the *registered* image pair into ITK-SNAP and display them.

c3d_call.sh (c3d_call.bat): Script to transform the moving image into the voxel space of the reference image using `c3d -reslice-matrix lcreg_result.mat` with cubic interpolation.
The name of the output image is `resampled_moving_image_c3d.mhd`.

view_masked_pyramid.sh (view_masked_pyramid.sh.bat): This script uncompresses the masked (i.e. edge) images of the multi resolution pyramid and displays all images with `ITK-SNAP`. Thus, the script allows the user to visualise the image parts used for registration and to correct the parameter `fixed_used_gradient_magnitude_fraction` if necessary.

lcreg_results.ini: Input file complemented by additional information e.g. the names of intermediate files, registration results for the individual resolution levels and the final value of the similarity measure.

resampled_moving_image_scipy.mhd (.raw): Moving image resampled in the voxel space of the fixed image using linear interpolation. This file is created only if the flag `resample_moving_image` is set to `True`.

Assessing registration results - concrete approach In order to visualise both intermediate images and results we change to the results directory

```
cd lcreg_tutorial_data/endo/lcreg/endo/results
```

In the first step, the situation prior to registration is visualised by typing

```
    # Linux
./snap_ungreg.sh
    REM Windows
snap_unreg.bat
```

To see which parts of the images have been used for registration, we type

```
    # Linux
./snap_ungreg.sh
    REM Windows
snap_unreg.bat
```

Changing the opacity slider, the relative weights of fixed and moving image can be modified. To overlay the registered moving image with the fixed image, the following commands are used:

```
    # Linux
./snap_reg.sh
    REM Windows
snap_reg.bat
```

In order to visualise the parts of the images that have been used for registration at the individual resolution levels, simply type

```
    # Linux
./view_masked_pyramid.sh
    REM Windows
view_masked_pyramid.bat
```

Note that voxels in the fixed images are used directly for similarity measure calculation whereas corresponding grey values in the moving images are determined by interpolation. Linear interpolation involves eight neighboring voxels. Thus more nonzero voxels are present in the masked moving images than in the masked fixed images. Visualisation of edge images allows to adjust the parameters `fixed_used_gradient_magnitude_fraction` and `moving_used_gradient_magnitude_fraction` in the input file. If relevant edges are not visible in the image, the value of this parameter should be increased. If background or noise-induced structures are present in the edge images, the value of the parameter should be reduced.
The absolute difference images between fixed and moving before and after registration are calculated by typing

```
abs_difference_mhd ../../../mhd/pre.mhd \
../../../mhd/post.mhd \
abs_difference_unreg.mhd

abs_difference_mhd ../../../mhd/pre.mhd \
resampled_moving_image_scipy.mhd \
abs_difference_reg.mhd
```

Note that in the Windows command line tool the character "^" rather than "\" is used for multi line commands. The resulting images can again be visualised using `ITK-SNAP`. In order to transform the moving image into the voxel space of the fixed image using `c3d` and cubic interpolation we type

```
    # Linux
./c3d_call.sh
    Rem Windows
c3d_call.bat
```

The transformation used by c3d is stored in the text file `lcreg_result.mat`. The output configuration file `lcreg_results.ini` containing intermediate information can be viewed using a text editor. Finally, the log file `lcreg.log` contains detailed information about the individual processing steps where the verbosity of logging is controlled by the parameter `log_level` in the `DEFAULT` section of the input file.

Conclusions In this case, registration using the identity transform as starting estimate for LM optimisation was succesful due to the small initial displacement of the images.

Please note In this section, both the Linux and Windows version of commands have been explained. For the sake of brevity, only the Linux version is given in the remainder of this document.

4.6 Combination with simple gradient (SG) optimisation

If the identity transform is used as starting estimate for iterative optimisation and the length of the initial displacement vector or rotation angles exceed a certain threshold, the Levenberg Marquardt algorithm may not converge to the correct solution. In these cases, an simple gradient optimisation method [6] with a larger capture range can be applied at the lowest resolution of the pyramid. The SG optimisation modifies only one parameter during each iteration where fixed step sizes for all parameters are used. Once the similarity measure can not be reduced further for the current step sizes, step sizes are halved and the procedure is repeated until the minimum step size has been reached. The results of SG optimisation are then used as starting estimates for the more efficient LM algorithm.

4.6.1 Example RIRE training with LM

The example used here is one of the training data set pairs (CT and rectified Proton Density weighted MRI images of the head) provided for the RIRE validation approach [12]. First of all, we change to the directory containing the images:

```
cd lcreg_tutorial_data/RIRE_training/mhd
```

Displaying an overlay of both images shows a relatively large initial displacement. Furthermore, the different contrast of CT and MR data set is clearly visible. The required command is[1]

```
tksnap -g training_001_ct.mhd -o training_001_mr_PD_rectified.mhd
```

Setting up the parameter file Right at the beginning of the parameter file `training_001_mr_PD_rectified_to_ct_LM_only.ini` the following new parameters are introduced:

```
patient_id = training_001
fixed_modality = ct
moving_modality = mr_PD_rectified
modality_combination = ${fixed_modality}_to_${moving_modality}
```

These settings are required to automatically create a file that can be submitted for evaluation according to the `RIRE` procedure.
In the `masking` section, fractions are set to 3 % and 5 % for fixed (CT) and moving (PD) image, respectively:

```
fixed_used_gradient_magnitude_fraction = 0.03
moving_used_gradient_magnitude_fraction = 0.05
```

These values are higher than for the `caries` example. The reason for this is that the strongest eges in the CT images correspond to transitions from bone to soft tissue whereas edges within the brain are more pronounced in the MR image. So in order to obtain *corresponding* edges contributing to the LCC measure, a larger fraction of edges in both images need to be preserved.

Starting `lcreg` The actual registration is started by typing

```
cd ../ini
lcreg training_001_mr_PD_rectified_to_ct_LM_only.ini
```

within the `ini` subdirectory. Registration takes about 7 seconds on a notebook computer with a 2.6 GHz dual core Intel© Core© i5-3320M CPU.

[1] Use the overlay feature of `itksnap` as described in the previous section

Assessing registration results After changing into the results directory, we first visualise the parts of the images used for registration:

```
cd ../lcreg/LM_only/results
./view_masked_pyramid.sh
```

Note that for lower resolutions, almost the complete anatomical content is used whereas for the highest resolution, LCC calculation is restricted to prominent edges. Finally, the overlay of the images after registration is visualised:

```
./snap_reg.sh
```

Obviously, the images are still misaligned so registration was not successful.

Conclusions In this case, registration was not succesful. One possible explanation is the significant initial displacement in combiniation with the identity transform that has been chosen as initial estimate. The limited capture range of LM optimisation prevented the optimisation process to converge to the correct transformation.

4.6.2 Example `RIRE training` with SG and LM

In order to successfully solve the regisration problem presented in the previous section, LM optimisation is preceded by a Simple Gradient (SG) descent starting from the identity transform at the lowest resolution. SG optimisation is less efficient but more robust than the LM algorithm. The result of the SG approach is used as initial estimate for LM optimistaion. At higher resolutions, only LM optimisation is applied as optimisation starts from the result of the next lower resolution step which should be close to the optimium already.

Setting up the parameter file The parameter file presented in the previous section is modified in the `simple_gradient` section:

```
[simple_gradient]
    active = True
    delta_v_initial = 4
    delta_v_min = 0.25
```

where the first line switches on SG optimisation and the second line sets the initial step width to a displacement of 4 voxels. Finally, the setting in the last line results in a minimum step width of SG descent corresponding to a displacement of 0.25 voxels. In many cases, the parameter `delta_v_min` can be set to 1.0. However, due to the relatively large slice thickness of 4 mm, a smaller value has been chosen in this case.

Starting `lcreg` The actual registration is started by typing

```
cd lcreg_tutorial_data/RIRE_training/ini
lcreg training_001_mr_PD_rectified_to_ct_SG_and_LM.ini
```

Registration takes about 8 seconds on a notebook computer with a 2.6 GHz dual core Intel© Core© i5-3320M CPU.

Assessing registration results Again, registered images are overlaid:

```
cd ../lcreg/SG_and_LM/results
./snap_reg.sh
```

This time, both position and orientation of the registered image are plausible indicating registration success.

Quantitative evaluation One big advantage of the RIRE data sets is the existance of ground truth transformations based on fiducial markers [12]. `lcreg` has been configured to store the registration result in the same format so that a direct comparison between ground truth and `lcreg` result is possible. In order to perform this comparison, the two text files can be opened in any text editor by typing e.g.

```
cd ../lcreg/SG_and_LM/results
vim -o RIRE_training_001_ct_to_mr_PD_rectified.transformation \
    ../../../standard/ct_PD_rectified.standard
```

Deviations in the colums `new_x, new_y, new_z` are well below the slice thickness of 5mm indicating successful registration. Details concerning the file format can be found at the RIRE home page.

Conclusions Adding SG optimisation at the lowest resolution level resulted in a successful automatic registration of a MR/CT image pair with significant initial displacement. Due to the larger capture range of SG descent, the identity transform could still be used as initial estimate.

4.7 Initialisation based on moments

If the initial translation vector and/or rotation angle exceeds a certain threshold, the identity transformation can no longer be used as as starting estimate for iterative optimisation due to limited capture range. However, initial parameters determined from edge moments can help to automatically register image pairs in many cases.

4.7.1 Example `caries`

Investigating the initial situation using

```
cd lcreg_tutorial_data/caries/mhd
itksnap -g pre.mhd -o post.mhd
```

we see that the initial translation is larger than in the examples discussed so far. Thus, optimisation starting with a zero translation vector will most likely not converge. However, the *orientation* of the images is almost identical so we can start with rotation angles of zero. Rather than finding the initial translation vector manually, we use grey value edge moments to automatically determine starting estimates for optimisation.

Setting up the parameter file The relevant settings for this example are

```
[transformation]
    ...
    initialise_from_edge_moments = True
    initialise_shift_only = True
    ...

[simple_gradient]
    active = False
    ...
```

where the initialisation from edge moments is enabled in the `transformation` section. Furthermore, only the translation (shift) is initialised automatically so that initial rotation angles are set to zero. Finally, SG optimisation is disabled.

Starting `lcreg` The actual registration is started by typing

```
cd ../ini
lcreg caries.ini
```

Registration takes about 19 seconds on a notebook computer with a 2.6 GHz dual core Intel© Core© i5-3320M CPU.

Assessing registration results Overlaying the images in itksnap with

```
cd ../lcreg/caries/results
./snap_reg.sh
```

indicates that registration was successful.

Exercise Modify the file `caries.ini` so that initialisation via edge moments is disabled. Verify that registration fails even if SG optimisation is switched on.

Conclusions In the presence of large initial translations, initialisation via edge moments allows for automatic registration in many cases. Furthermore, as only *corresponding* edges are used for registration, the method works even if anatomical structures are absent in one image due to preparation or resection.

4.7.2 Example `preparation_post_to_pre`

If initial translation and rotation are substantial, starting estimates for both can be determined automatically using edge moments. However, if image contents differ e.g. due to preparation or different image sections, SG optimisation needs to be applied at the lowest resolution level.

Image overlay prior to registration Comparing the original images

```
cd lcreg_tutorial_data/preparation/mhd
itksnap -g pre.mhd -o post.mhd
```

reveals that the images are not only shifted but also rotated. Furthermore cavities are present in the `post` image only.

Setting up the parameter file The relevant changes with respect to the previous example are

```
[transformation]
    ...
    initialise_from_edge_moments = True
    initialise_shift_only = False
    ...

[simple_gradient]
    active = True
    delta_v_initial = 4
    delta_v_min = 0.25
```

In this case, not only translation (shift) but also rotation is initialised from edge moments. SG descent is enabled for the lowest resolution level.

Starting `lcreg` The actual registration is started by typing

```
cd ../ini
lcreg post_to_pre.ini
```

Registration takes about 17 seconds on a notebook computer with a 2.6 GHz dual core Intel© Core© i5-3320M CPU.

Assessing registration results
Analysing the log file `lcreg/post_to_pre/results/lcreg.log` shows that the initial transform derived from edge moments (line 75)

```
2019-01-17 18:50:15,217 - DEBUG: initial euler angles (deg):
[18.16909784 12.52004392 67.12175554]
2019-01-17 18:50:15,217 - DEBUG: initial translation vector:
[14.05692041 -2.96317919 -0.71906692]
```

is pretty close to the final transform after optimisation (line 191)

```
2019-01-17 18:50:18,035 - DEBUG: final euler angles (deg):
[17.46343155 11.98068923 66.5885782 ]
2019-01-17 18:50:18,035 - DEBUG: final translation vector:
[14.08773353 -2.93067429 -0.74523916]
```

An overlay of transformed images

```
cd ../lcreg/post_to_pre/results
./snap_reg.sh
```

indicates successful 3D registration. Note that registration is impaired by imaging artifacts in the cusp area. These artifacts result from unwanted sample motion during image acquisition.

Conclusion Even in the presence of substantial initial mismatch and image differences due to preparation or resection, automatic registration is possible by combining initialisation via edge moments and SG descent optimisation at the lowest resolution level. However, imaging artifacts affected registration accuracy in this case.

4.8 Manual initialisation using ITK-SNAP

If the contents of the two images to be registered differ significantly due to resection or preparation, initial estimates based on edge moments may be located outside the capture range of the optimisation so that registration fails to converge. In these cases, the initial estimate can not be determined automatically. However, valid starting estimates can be derived from a set of corresponding anatomical landmarks [1]. Alternatively, `ITK-SNAP` provides a tool for 3D image registration featuring both an automatic and a manual mode [3]. Although `ITK-SNAP` requires the images to fit into memory, downsampled versions of large data sets can be used perform rough interactive alignment. Then, the transformation matrix can be stored to disk and used as initial estimate for `lcreg`.
Care must be taken to keep the geometry of the downsampled images, in particular the origins, consistent with modified voxel dimensions.

4.8.1 Example `veneer_post_to_pre`

Figure 2 shows overlaid μCT images of a tooth before and after preparation for a veneer. Note that preparation involved both shortening of the tooth and enamel removal. Furthermore, the two images show the tooth at different positions and orientations. In order to manually determine valid starting estimates, the images are loaded into `ITK-SNAP`

```
cd lcreg_tutorial_data/veneer/mhd
itksnap -g pre.mhd -o post.mhd
```

`Registration` is chosen from the `Tools` menu and the mode is set to `Manual` as shown in figure 3 on page 27. Clicking at the white circle in the top left image, the post image is rotated by about -25°. Afterwards, we click into the top left view at the center of the tooth to be moved, hold down the left mouse button and drag the tooth to the top by about 5 mm and to the right by about 0.6 mm. Then, we select the top right view and shift the tooth upwards by about 1.4 mm. The resulting view should be similar to figure 4 on page 28. Finally, the matrix representing our

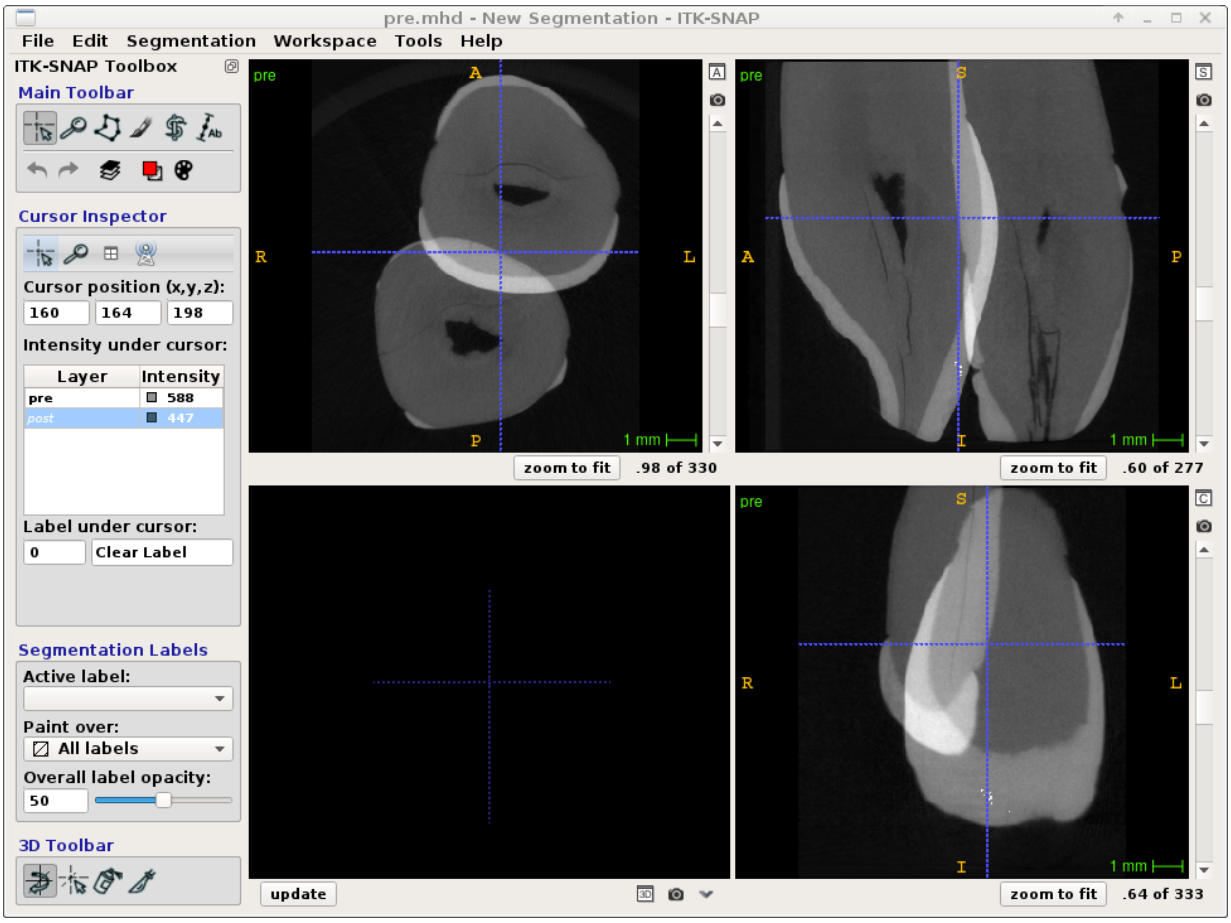

Figure 2: Overlay of veneer pre and post images with ITK-SNAP

transformation needs to be saved to disk. In order to achieve this, we click on the disk symbol at the bottom of the registration tab and select the file format `Convert 3D Transform Files`. Using the button `Browse` we can select the directory `../ini` and overwrite the existing file `veneer_post_pre.mat`. A copy of the initial file named `veneer_post_pre_backup.mat` is provided so that the original state can be restored. After the file has been saved, `ITK-SNAP` can be closed.

Setting up the parameter file In order to use the stored transformation as starting estimate, parameters in the `transform` section need to be set as follows:

```
[transformation]
    ...
    initial_transformation_file_name = veneer_post_pre.mat
    initial_transform_is_RAS = True
    initial_transform_is_inverted = False
    initialise_from_edge_moments = False
```

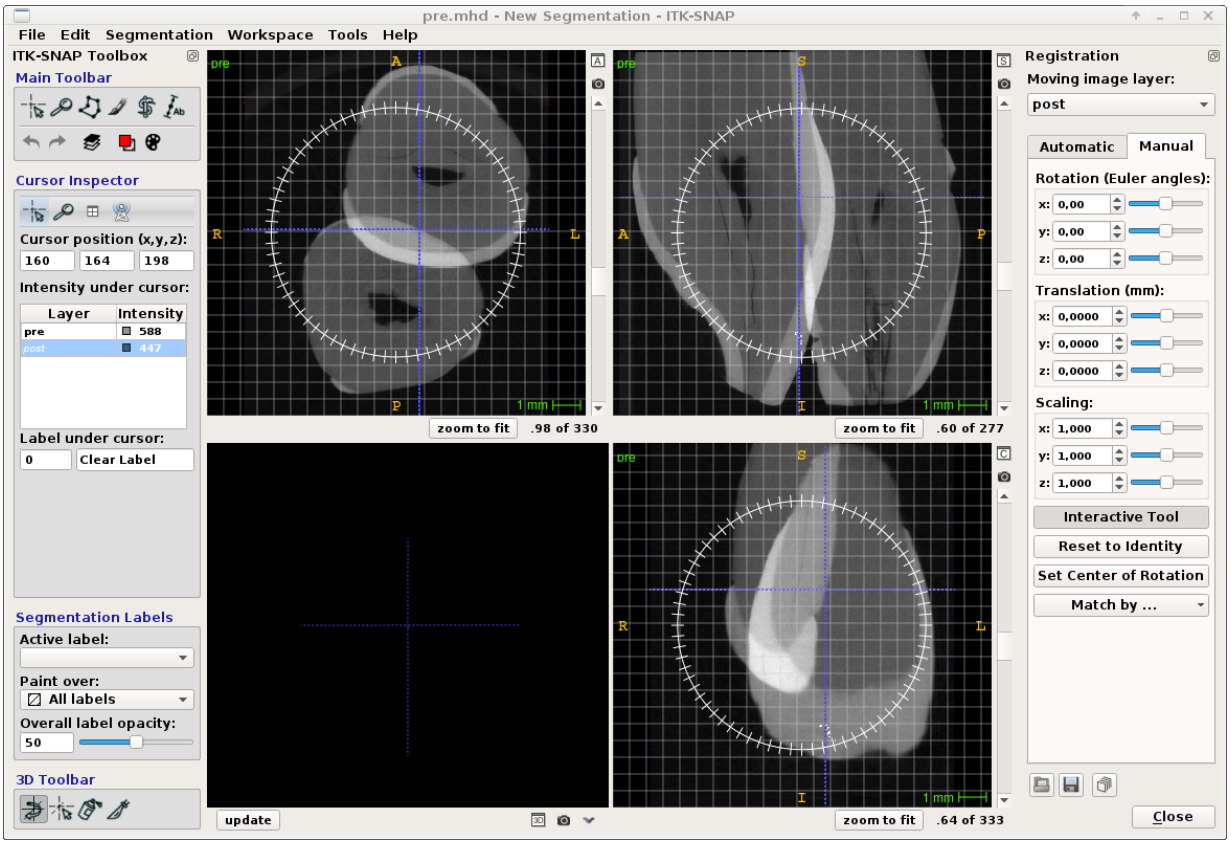

Figure 3: Veneer pre and post images prior to manual alignment.

...

Starting `lcreg` The actual registration is started by typing

```
cd ../ini
lcreg veneer_post_to_pre.ini
```

Registration takes about 17 seconds on a notebook computer with a 2.6 GHz dual core Intel© Core© i5-3320M CPU.

Assessing registration results An overlay of the images after registration can be created by typing

```
cd ../lcreg/post_to_pre/results
./snap_reg.sh
```

Results indicate succesful registration. Note that cracks within the dentine area are wider in the `post` image than in the `pre` image. A possible reason for this is sample dehydration.

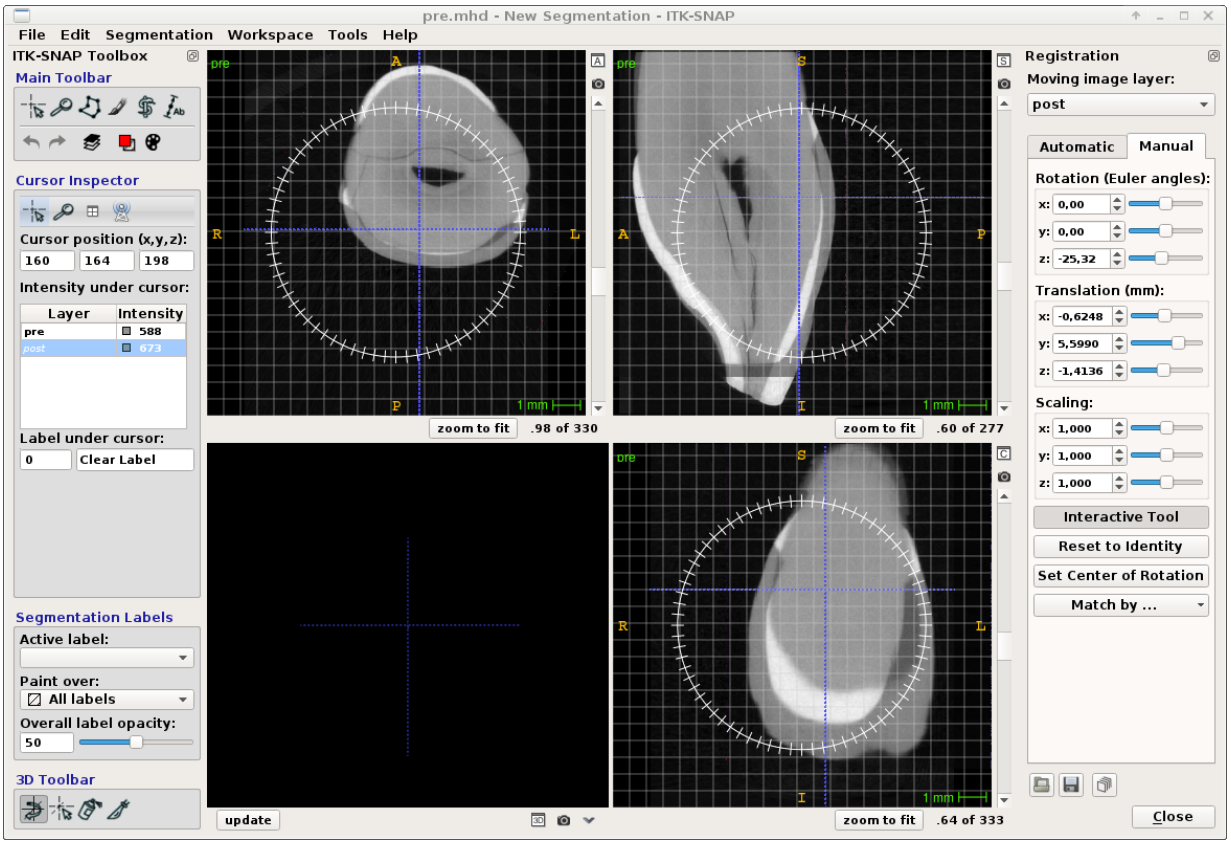

Figure 4: Veneer pre and post images after manual alignment.

Exercise Verify that an initialisation based on edge moments does not result in successful registration using the corresponding parameter file provieded in the `ini` directory.

Conclusions Using the manual registration feature of `ITK-SNAP`, to create an initial esitmate for the transformation matrix, two images of a tooth before and after preparation could be successfully registered dispite the significant large initial misalignment, the removal of enamel and the widening of cracks in the course of preparation.

4.9 Applying binary masks

Registration using `lcreg` is based on the alignment of edges, i.e. image regions with large local grey value gradients. The underlying assumption is that these edges correspond to borders between meaningful anatomical structures. It follows that edges originating from noise or non-anatomical structures like patient tables or sample holders can disturb the registration process. This unwanted interference can be avoided by excluding disturbing image parts from registration using binary masks. Note

that following the masking operation based on the binary image, the usual selection criteria controlled by the `gradient_magnitude_fraction` and `grey_masking_range` settings are still applied.

4.9.1 Example `root_canal_treatment`

The images to be registered show a tooth at different stages of root canal treatment. From an overlay of the images using

```
cd lcreg_tutorial_data/root_canal_treatment/mhd
itksnap -g pre.mhd -o post.mhd
```

three conclusions can be drawn:

1. The initial displacement is small.
2. The restoration takes up a small fraction of the `pre` and a significant fraction of the `post` image.
3. Due to inhomogeneities, spurious edges are visible in the restoration area.

Thus, it appears sensible to exclude the restoration from registration by masking. Two positive masks have been created by first segmenting the restorations using `ITK-SNAP` and then inverting the resulting binary image followed by a morphological erosion to exclude restoration borders. The resulting masks can be visualised as follows:

```
itksnap -g pre.mhd -s pre_mask.mhd
itksnap -g post.mhd -s post_mask.mhd
```

Setting up the parameter file First of all, the mask images are included:

```
[DEFAULT]
    ...
    fixed_mask_image_name = ${image_directory}/pre_mask.mhd
    moving_mask_image_name = ${image_directory}/post_mask.mhd
    ...
```

As the initial displacement is small, no SG optimisation step is required.

```
[simple_gradient]
    active = False
    ...
```

Starting `lcreg` The actual registration is started by typing

```
cd ../ini
lcreg root_canal_treatment.ini
```

Registration takes about 13 seconds on a notebook computer with a 2.6 GHz dual core Intel© Core© i5-3320M CPU.

Assessing registration results An overlay of the images after registration

```
cd ../lcreg/root_canal_treatment/results
./snap_reg.sh
```

shows that registration was successful. We can also investigate difference images before and after registration using the commands given below. Note that the sizes of the unregistered images do not agree so that a c3d -reslice-identity operation needs to be performed prior to voxel-by-voxel image subtraction.

```
c3d ../../../mhd/pre.mhd ../../../mhd/post.mhd \
    -reslice-identity -type ushort -o tmp.mhd
abs_difference_mhd ../../../mhd/pre.mhd \
    tmp.mhd abs_diff_unreg.mhd
rm tmp.mhd tmp.raw
abs_difference_mhd ../../../mhd/pre.mhd \
    resampled_moving_image_scipy.mhd abs_diff_reg.mhd
itksnap -g abs_diff_unreg.mhd -o abs_diff_reg.mhd
```

Exercise In order to compare results with and without binary mask, repeat the experiment with the configuration file `root_canal_treatment_no_mask.ini`. In this case, registration results are very similar as only `corresponding` edges contribute to the similarity measure so that the restoration for which no corresponding edges are present in the `pre` image does not contribute significantly.

Conclusions In cases where edges originating from non-anatomical structures are present, binary masks can be used to make sure that only relevant image parts influence the registration results.

4.10 Affine transformations

Although the main application area of `lcreg` is rigid registration with six parameters, the software also allows for affine (i.e. twelve-parameter) registration including scale and shear. This type of transform is frequently used as an intermediate step for elastic registration.

4.10.1 Example `atlas_affine`

In this example, an MR T_1 atlas image is registered to a simulated T_1 image of the brain. The affine registration which is modified to optimise the similarity measure is only an approximation of the "real" transform which is elastic. Thus, the convergence of LM optimisation scheme is expected to be slower than for rigid or "really" affine transformations. Due to the larger number of parameters, an initial SG descent is not feasible for affine registration. Thus, a valid initial estimate needs to be determined either from a rigid preregistration, a set of corresponding landmarks or by manually aligning the images using e.g. `ITK-SNAP`. Note that an initialisation based on edge moments is not supported for affine registration. Here, the atlas image has been shifted manually to the centre of the brain in the T_1 image as shown in figure 5.

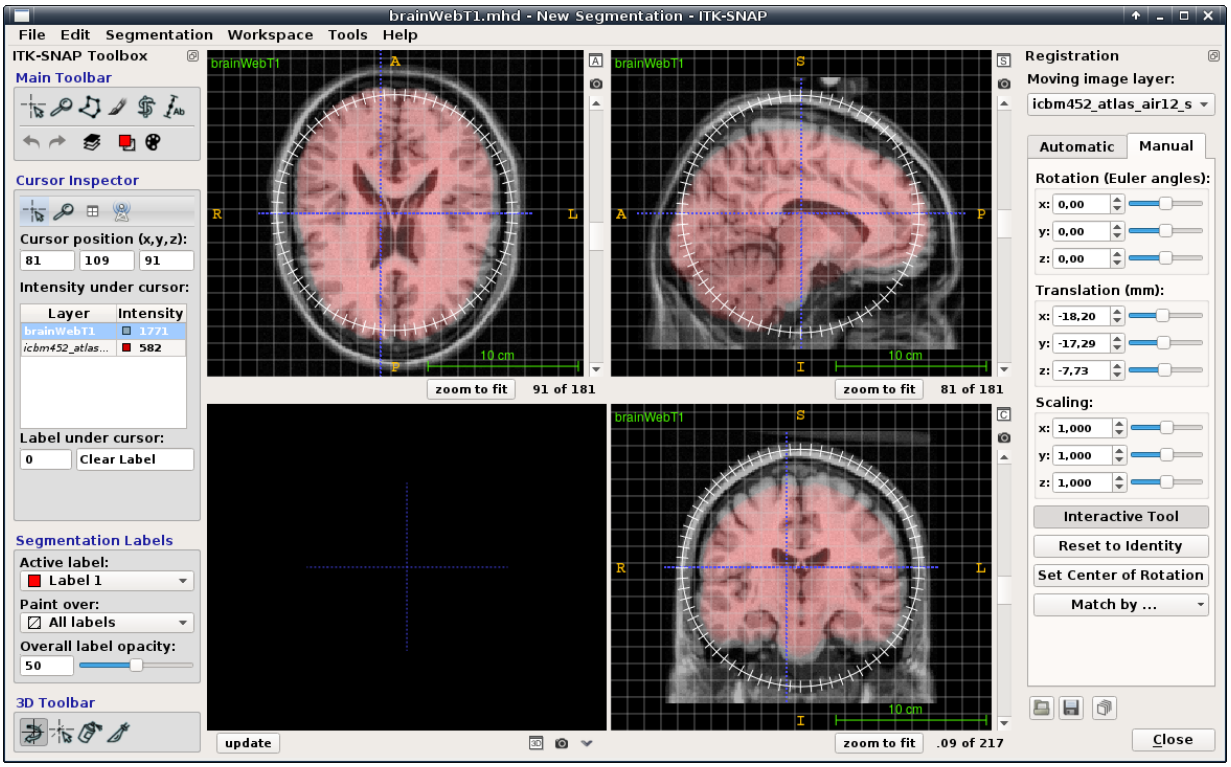

Figure 5: Atlas and T_1 image after manual alignment.

Setting up the parameter file The transformation type is set to affine and the initial transform is read from a file:

```
[transformation]
    ...
    transformation_type = affine
    initial_transformation_file_name = atlas_affine.mat
    ...

[levenberg_marquardt]
    ...
    initial_lambda = 1
    lambda_upscale_factor = 1.1
    lambda_downscale_factor = 0.9
    ...
```

In this case, elastic deformation is approximated by an affine transform, so strictly speaking, the preconditions for LM optimisation are not met. Starting with a `lambda` value of 1 and using upscale and downscale factors close to 1, the relative weight between gradient descent and Gauss-Newton optimisation does not change as rapidly as in the other examples where `lambda` changed by an order of magnitude with each

step.

Starting `lcreg` The actual registration is started by typing

```
cd ../ini
lcreg atlas_affine.ini
```

Registration takes about 71 seconds on a notebook computer with a 2.6 GHz dual core Intel© Core© i5-3320M CPU.

Assessing registration results Investigating an overlay after registration using

```
cd ../lcreg/atlas_affine/results
itksnap -g ../../../mhd/brainWebT1.mhd \
    -o resampled_moving_image_scipy.mhd
```

shows that although the differences are less prominent than in figure **??** on page **??**, the agreement between fixed and transformed moving image is worse than in the other examples. The reason for this is that the transformation applied for registration is only an *approximation* of the "real" elastic deformation as described above.

Conclusions Starting from manually pre-aligned images, optimisation of 1-LCC with respect to an affine transformation converged and the agreement between fixed and transformed moving image could be improved. However, as a elastic transform is required to accurately describe the relationship between images, only a part of the differences could be compensated for by affine registration.

5 How to cite `lcreg`

When referencing `lcreg`, please include a bibliographic reference to [2].

6 Acknowledgements

Many thanks to Karl-Heinz Kunzelmann for his support, many helpful discussions and for making dental test images available. This work benefited from the use of ITK-SNAP, an open-source software for image segmentation and visualisation. The University of Applied Sciences, Augsburg, in particular the Faculty of Computer Science supported this project by granting a sabbatical leave. Special thanks to Gisela Dachs, Andreas Gärtner, Evi Köbele, Stefan König, Dominik Lüder, Thomas Obermeier and Sigrid Podratzky for acquiring images and for keeping computers up and running.

References

[1] J. V. Hajnal, D. L. G. Hill, D. J. Hawkes (Eds.): *Medical Image Registration*, CRC Press (2001)

[2] P. Rösch, K.-H. Kunzelmann: *Efficient 3D rigid Registration of large micro CT Images.* International Journal of Computer assisted Radiology and Surgery **13 (Suppl. 1)** (2018) 118–119

[3] P. A. Yushkevich, J. Piven, H. C. Hazlett, R. G. Smith, S. Ho, J. C. Gee, G. Gerig: *User-guided 3D active contour segmentation of anatomical structures: Significantly improved efficiency and reliability.* Neuroimage **31** (2006) 1116–1128. http://www.itksnap.org

[4] S. Kabus, T. Netsch, B. Fischer, J. Modersitzki: *B-Spline Registration of 3D Images with Levenberg-Marquardt Optimization.* Proceedings of SPIE **5370** (2004), 304-313

[5] P. Rösch, T. Blaffert, J. Weese: *Multi-Modality Image Registration using Local Correlation.* In: H. U. Lemke, M. W. Vannier, K. Inamura, A. G. Farman (Eds.): Proceedings of CARS'99, Elsevier Science (1999) 228-232

[6] C. Studholme, D. L. G. Hill, D. J. Hawkes: *Automated 3-D registration of MR and CT images of the head.* Med. Image Anal. **1** (1996) 163–175

[7] https://www.cython.org.

[8] F. Alted et al.: *bcolz documentation.* http://bcolz.blosc.org (2019)

[9] https://www.scipy.org

[10] T. Netsch, P. Rösch, A. v. Muiswinkel, J. Weese: *Towards Real-Time Multi-Modality 3-D Medical Image Registration.* Eight IEEE International Conference on Computer Vision, ICCV 2001, 718-725

[11] A. Ranganathan: *The Levenberg-Marquardt Alogoritm.* http://www.ananth.in/docs/lmtut.pdf

[12] J. West, J. Fitzpatrick et al: *Comparison and Evaluation of Retrospective Intermodality Brain Image Registration Techniques.* Journal of Computer Assisted Tomography **21** (1997) 554–568.